Android programming

# Inleiding

Deze handleiding wordt gemaakt aan de hand van een programma, die je kan vinden op Github. <https://github.com/KennyDeblaere/Coaching>. Deze applicatie zal alle elementen bevatten en is gemaakt in de Android Studio omgeving.

# Opstarten project

We starten een project, waarbij we kiezen voor de API vanaf 4.4. De bedoeling van het project is er voor te zorgen dat de leerlingen tijdens een vak in eerste instantie kunnen meegeven als het vak te snel of te traag gegeven wordt. Later zou er dan een uitbreiding moeten voorzien worden dat een student vragen kan stellen.

# Icoon veranderen

Druk of File > New > Image Asset. Daarin kun je een gewenste icon toevoegen aan het project. We willen ook dat het programma Full-screen draait. Daarvoor voegen we in onze AndroidManifest.xml de volgende lijn toe in de activity. Denk er ook aan at je de MainActivity Activity laat extenden, niet een andere mogelijkheid.

<activity android:name=".MainActivity"

android:theme="@android:style/Theme.NoTitleBar.Fullscreen">

</activity>

# Het kiezen van een vak, Spinner

Het programma moet in eerste instantie de vakken weergeven. We kiezen hiervoor voor de vakken uit de opleiding Toegepaste Informatica uit semester 5. Deze worden weergegeven in een Spinner. We vullen deze waarden niet manueel op, dit doen we in de res > values folder. In de file strings.xml kunnen we volgende waardes toevoegen

<**string-array name="courses"**>  
 <**item**>Operationeel Onderzoek</**item**>  
 <**item**>Java EE</**item**>  
 <**item**>Linux Server</**item**>  
 <**item**>Node.js</**item**>  
 <**item**>VMWare</**item**>  
 <**item**>IT Governance</**item**>  
 <**item**>Internet of Things</**item**>  
 <**item**>Native Mobile Apps</**item**>  
 <**item**>Professionele Communicatie</**item**>  
 <**item**>Cloud Development</**item**>  
 <**item**>User Experience Design</**item**>  
 <**item**>Project</**item**>  
</**string-array**>

Nu dit toegevoegd is, zullen we deze toevoegen aan onze spinner.

Spinner courseSpinner = (Spinner) findViewById(R.id.course\_spinner);  
// Create an ArrayAdapter using the string array and a default spinner layout  
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,  
        R.array.planets\_array, android.R.layout.simple\_spinner\_item);  
// Specify the layout to use when the list of choices appears  
adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);  
// Apply the adapter to the spinner  
courseSpinner.setAdapter(adapter);

## Events

Events voor een Spinner worden zo meegegeven

**courseSpinner**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
  
 @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **courseTitle**.setText(**courseSpinner**.getSelectedItem().toString());  
 }  
  
 @Override  
 **public void** onNothingSelected(AdapterView<?> parent) {  
  
 }  
});

# Feedback op de snelheid van het vak

<**ProgressBar  
 style="?android:attr/progressBarStyleHorizontal"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/courseProgress"  
 android:layout\_below="@+id/feedback"  
 android:layout\_alignStart="@+id/titlebar"  
 android:max="100"  
 android:progress="50"  
 android:indeterminate="false"  
 android:layout\_margin="15dp"** />

We zullen met 2 ImageButtons aangeven als het gegeven onderwerp sneller of trager gegeven moet worden. Deze hebben een onClickListenener.

**slower**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **feedbackProgress**.setProgress(**feedbackProgress**.getProgress() + 1);  
 **int** input = **feedbackProgress**.getProgress() - 50;  
 **if**(input <= 50)  
 **feedbackNumber**.setText(**""** + input);  
 }  
});

Nu dat de feedback werkt, hebben we een volgend probleem. De studenten kunnen namelijk blijven drukken op de knop. Daarom gaan we dit nu verhinderen. We maken daarvoor eerst een globale variabele, waarbij we meegeven dat er geklikt kan worden. Zo hoeven we de UI niet up te daten en heeft de mainThread geen issues. Daarna maken we de volgende methode:

**private** Thread delayForButton(){  
 **return new** Thread(){  
 @Override  
 **public void** run() {  
 **super**.run();  
 **try** {  
 *sleep*(5000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **imageButtonClick** = **true**;  
 }  
 };  
}

Deze thread starten we dan op het einde van onze event.

# Met een button naar een andere activity

Omdat ons scherm te klein zou worden, gaan we de vragen op een andere pagina, activity zetten. Daarvoor maken we een button aan en als we er op klikken gaan we naar de vragen en antwoorden.

**final** Context context = **this**;  
**questionsButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(context, QuestionsActivity.**class**);  
 startActivity(intent);  
 }  
});

Als we het juiste vak willen meegeven, dan voegen we deze lijn toe in de methode.

intent.putExtra(**"course"**, **courseSpinner**.getSelectedItem().toString());

Die vragen we dan op in de onCreate van de nieuwe Activity:

intent.getStringExtra("course");

# Vragenlijst

Om vragen te kunnen beantwoorden, moeten we eerst vragen hebben. Deze stoppen we eerst hard-coded in een array. Deze zullen we dan inlezen in een ListView.

**private void** fillListView(){  
 ArrayAdapter<String> adapter =   
 **new** ArrayAdapter<String>(**this**,   
 android.R.layout.***simple\_list\_item\_1***,  
 android.R.id.***text1***,   
 getResources().getStringArray(R.array.***nativeQuestions***));  
 **questions**.setAdapter(adapter);  
}

Om dan een vraag toe te voegen aan de lijst, voeg je dit in

**private void** addListenerToAsk(){  
 **final** Context context = **this**;  
 **ask**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **questionList**.add((String) **question**.getText().toString());  
 ArrayAdapter<String> adapter =  
 **new** ArrayAdapter<String>(context,  
 android.R.layout.***simple\_list\_item\_1***,  
 android.R.id.***text1***,  
 **questionList**);  
 **questions**.setAdapter(adapter);  
 adapter.notifyDataSetChanged();  
 }  
 });  
}

Als we op een vraag klikken, dan willen we naar een andere activity, waar we oplossingen kunnen bekijken. We moeten natuurlijk deze vraag ook zien staan.

**private void** addListenerToListView(){  
 **final** Context context = **this**;  
 **questions**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 Intent intent2 = **new** Intent(context, ResponseActivity.**class**);  
 intent2.putExtra(**"question"**, parent.getItemAtPosition(position).toString());  
 startActivity(intent2);  
 }  
 });  
}

# Een antwoord geven op een vraag

Dit is hetzelfde principe als een vraag stellen. Ook de layout lijkt er goed op.

# Broadcasting

Als er een vraag gesteld wordt, zou iedere gebruiker deze moeten kunnen gebruiken. Dit gaan we zo doen.

# Marges in Android

Dit gebeurt op volgende wijze

<**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textAppearance="?android:attr/textAppearanceLarge"  
 android:id="@+id/titlebar"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginLeft="5dp"  
 android:layout\_marginRight="5dp"  
 android:layout\_marginBottom="10dp"  
 android:text="@string/app\_name"**/>

**android:gravity="center"**

Dit wordt gebruikt om de tekst te centreren.