

# Pokémon App Dokumentation

C:\Users\Isa\AndroidStudioProjects\MyApplication

Ersteller: Isabell Noack

Projektstart: 18.10.2023

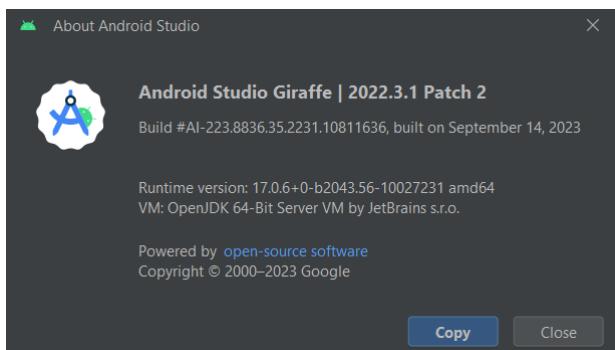
Modul: Mobile Apps Entwicklung

Prof. Dr.-Ing. Michael Stepping

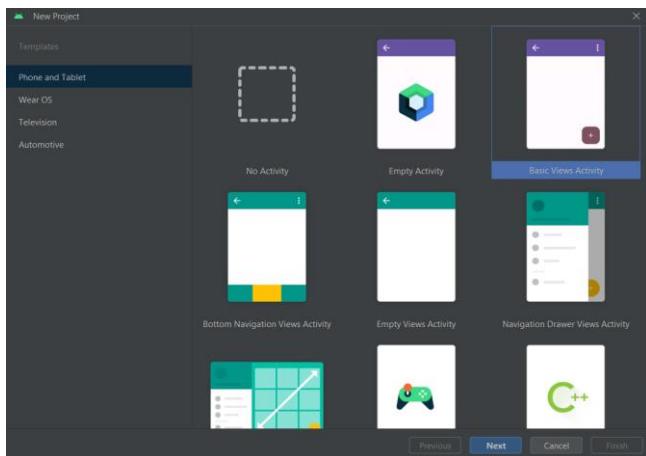
Ziel: Erstellung einer Mobile Android Applikation zum Aufrufen und detaillierten Einsehen von Pokémon

## Projekt erstellen

Version: Android Studio Giraffe | 2022.3.1 Patch 2



Template benutzt (Basic Views Activity)



Package: com.isabellnoack.myapp

## Github Repo erstellt

<https://github.com/IsaNck1/PokeApp>

## PokéAPI

<https://pokeapi.co/>

Um Daten zu Pokémon zu bekommen, wird PokéAPI benutzt. Es ist kostenlos und open-source.

PokéAPI stellt hierfür JSON Daten im Internet bereit. Beispiel: <https://pokeapi.co/api/v2/pokemon/>

## API JSON schön auslesen

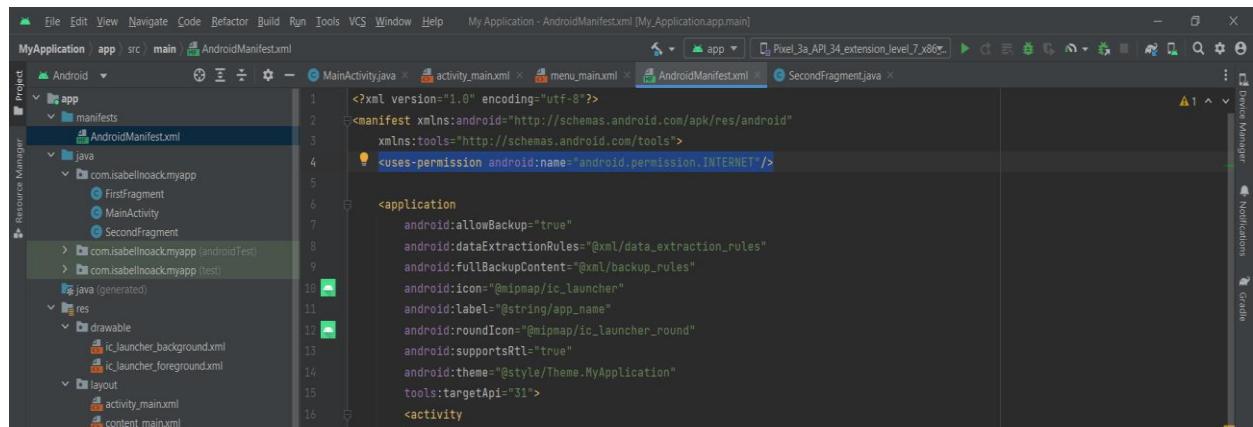
Hier reinkopieren, um es schön zu formatieren: <https://jsonformatter.curiousconcept.com/#>

## Verwendung des Internets

AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

Berechtigung das Internet zu benutzen



## Threads (Netzwerkkommunikation)

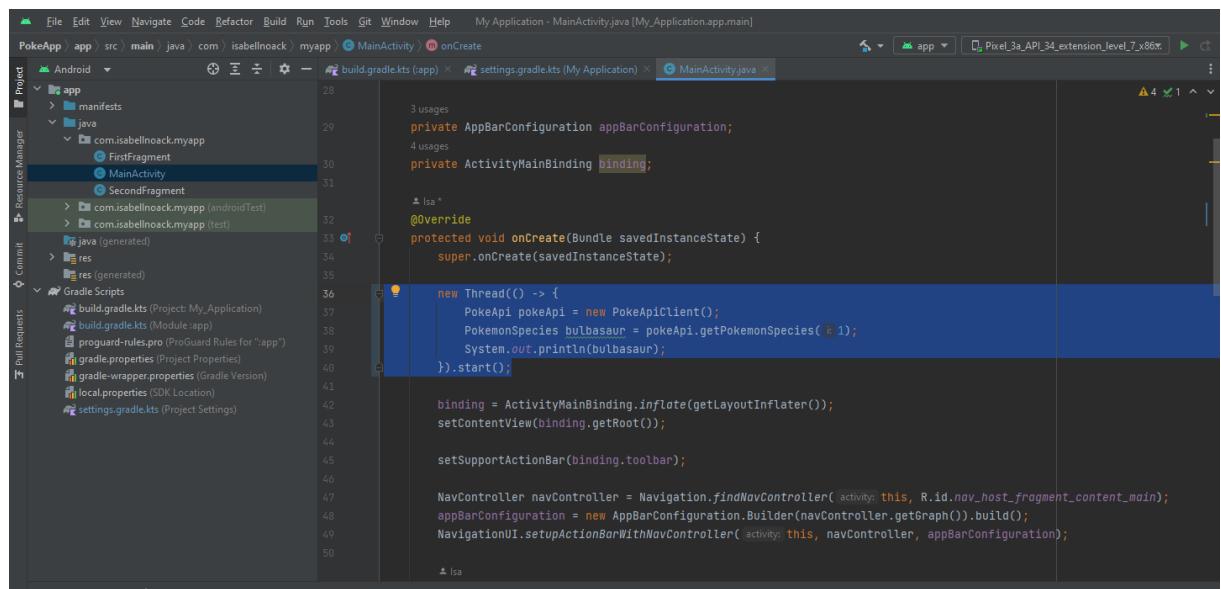
Allg. nützlich: Formatierung mit Strg+Alt+O / Strg+Alt+L

Fehlermeldung und App Crash:

FATAL EXCEPTION: main ... Caused by: android.os.NetworkOnMainThreadException

Grund: Internetkommunikation auf UI Thread ist schlecht.

Lösung: Abfrage auf neuem Download Thread

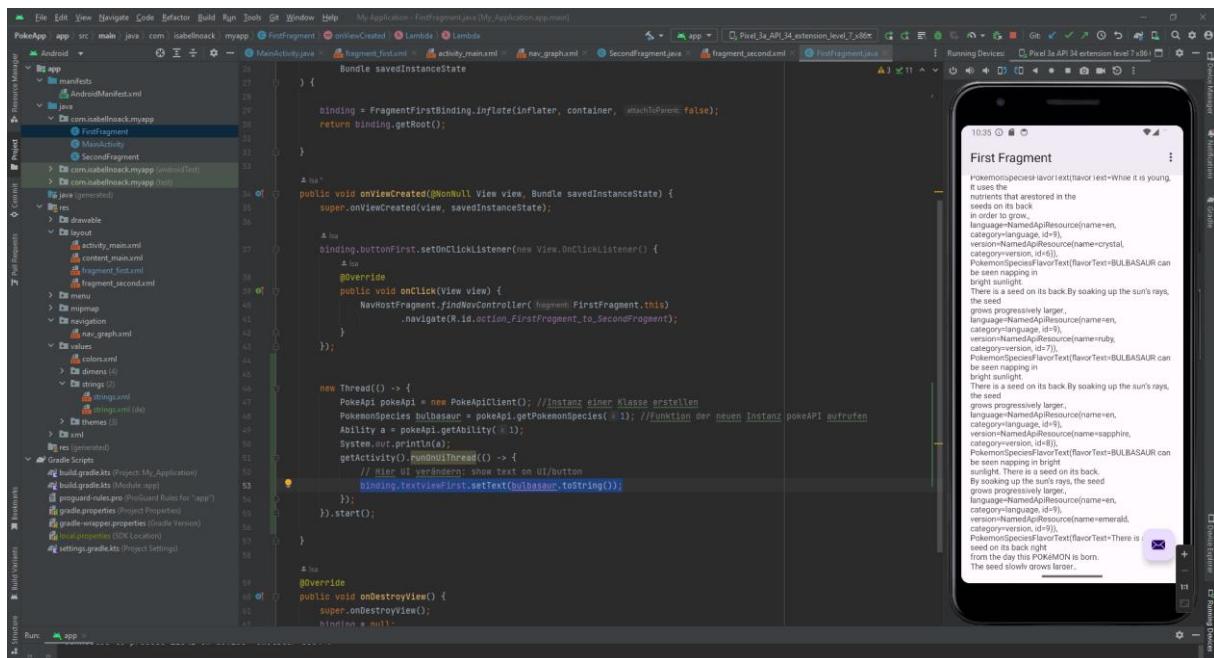


## Test Bulbasaur als Text erstellen und ersetzen

Dafür zuerst das Bulbasaur Test in FirstFragment gepackt, dort dann mit

```
binding.textViewFirst.setText(bulbasaur.toString());
```

ersetzt.



```
new Thread() -> {
    PokeApi pokeApi = new PokeApiClient(); //Instanz einer Klasse erstellen
    PokemonSpecies bulbasaur = pokeApi.getPokemonSpecies(1); //Funktion der
                                                               // neuen Instanz pokeAPI aufrufen
    Ability a = pokeApi.getAbility(1);
    System.out.println(a);
    getActivity().runOnUiThread(() -> {
        // Hier UI verändern: show text on UI/button
        binding.textViewFirst.setText(bulbasaur.toString());
    });
}.start();
```

## Sprach-Einstellungen

Sprach-Spezifisch unter res => values => strings.xml

In strings.xml(de) steht dann der Text für alles mit Deutschem System, ansonsten wird die Allgemeine strings.xml benutzt

```
<string name="hello_world">Hello World</string>      strings.xml
```

```
<string name="hello_world">Hallo Welt</string>           strings.xml(de)
```

## JSON auslesen (JSONReader)

Dokumentation zum JSON Reader <https://developer.android.com/reference/android/util/JsonReader>

Standard mäßig in Android drinnen. Klasse importieren da es in einem anderen Package ist.

```
import android.util.JsonReader;
```

JSON Reader in der Klasse PokeAPI (PokeAPI.java file)

### JSONObject vs. JSONReader

JSONObject liest JSON auf einmal aus

JSONReader geht Key für Key durch

=> Ich habe mit JSONReader angefangen, es geht aber beides

### Funktion requestPokemon (Ablauf JSONReader)

Funktion requestPokemon wird aufgerufen. (STRG + Links Klick zum Finden, wo es aufgerufen wird)

```
requestPokemon
```

Name der Funktion

```
(int id)
```

Parameter

```
Pokemon
```

Rückgabetyp (Eine Instanz der Klasse Pokemon)

```
throws IOException
```

Ausnahme (Verbindung bricht ab; übers Ende hinauslesen; Datei nicht gefunden)

```
//1
public Pokemon requestPokemon(int id) throws IOException {
    //Wo lesen wir unsere Daten her
    String url = "https://pokeapi.co/api/v2/pokemon/" + id; //Datenquelle
    JsonReader reader;
    reader = requestJsonReader(url);
    //Interpretiert Text aus Internet als JSON
    return readPokemon(reader); //Funktion für JSON in Pokemon Instanz
}
```

1. Festlegung, woher die Daten kommen: URL
2. JSON Reader nimmt die URL und liest von dort die Daten aus

3. Neue Instanz der Klasse JSON Reader „reader“ (Variable reader ist noch undefiniert)
4. Wir Speichern den Rückgabewert der Funktion requestJsonReader(url) in den reader
- 5. Funktion requestJsonReader(url) gibt uns den bufferedReader in der Variable reader zurück**
6. Wir rufen die Funktion readPokemon(JSONReader reader) mit dem Wert reader auf
- 7. Funktion Pokemon readPokemon(JsonReader reader)**
8. Der Rückgabewert wird zurückgegeben

## Funktion JsonReader requestJsonReader(String url)

```
//2
JsonReader requestJsonReader(String url) throws IOException { //JSON lesen
    HttpURLConnection connection = (HttpURLConnection) new
URL(url).openConnection();
    connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0
Safari/537.36");
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(
connection.getInputStream(), "UTF-8")); //Datenstrom als UTF-8 lesen
    return new JsonReader(bufferedReader); //Text/JSON als Antwort
}
```

JsonReader requestJsonReader(String url) throws IOException { //JSON lesen  
**Funktion mit Eingabewert String url**

```
HttpURLConnection connection = (HttpURLConnection) new
URL(url).openConnection();
```

Wir haben die Variable connection der Klasse HttpURLConnection.

Wir machen aus dem String url eine URL(url) damit wir .openConnection() aufrufen können.

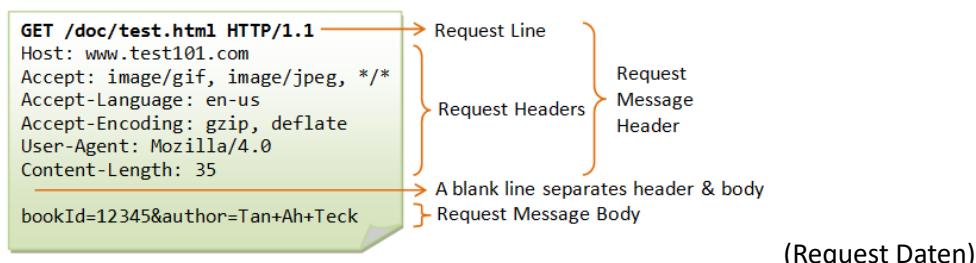
(.openConnection() ist in der Klasse URL definiert )

(Sie String url kommt von: String url = "https://pokeapi.co/api/v2/pokemon/" + id;)

```
(HttpURLConnection) new URL(url).openConnection()
```

Die open Connection kann viele verschiedene Typen von urls benutzen. Wir wissen, dass wir eine HttpURLConnection haben. Das nennt man casten (Datentyp einer Oberklasse zu spezieller Unterklasse, damit man das als Unterklasse benutzen kann muss man den Datentyp der variable ändern)

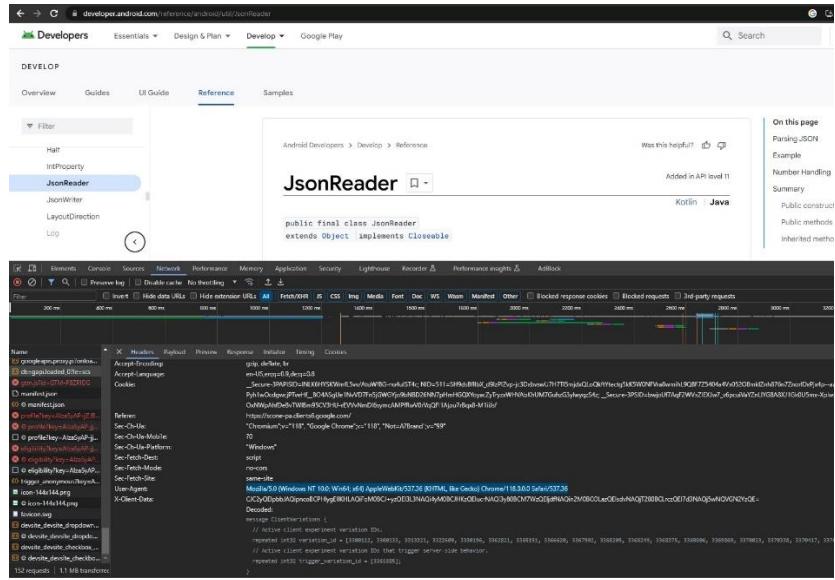
HttpURLConnection unterstützt Header (weil das http Protokoll Header unterstützt (Zusatzdaten))



```
connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36");
```

Funktion setRequestProperty der Klasse HttpURLConnection

Aussage: Ich unterstütze alle Features von Firefox der Version 5, Ich unterstütze alle Features die die Version „Chrome 118“, „Safari 537“ etc. hat.



Von hier genommen. Das machen alle Browser so.

Ansonsten antwortet die API nicht.

```
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(connection.getInputStream(), "UTF-8"));
```

Variable der Klasse BufferedReader „bufferedReader“ deklariert

```
new BufferedReader(new InputStreamReader(connection.getInputStream(), "UTF-8"));
```

Funktion connection.getInputStream() wird aufgerufen

Die Verbindung wird hergestellt und die Request Daten werden zum Server geschickt.

Wir bekommen einen InputStream zurück. (von der Funktion .getInputStream()) (Datenstrom als UTF-8 Format)

Der InputStream wird zu einem BufferedReader konvertiert. (und in unserem bufferedReader gespeichert)

```
return new JsonReader(bufferedReader);
```

Wir bekommen den bufferedReader als Rückgabewert.

## Funktion Pokemon readPokemon(JsonReader reader)

```
//3
Pokemon readPokemon(JsonReader reader) throws IOException {
    Pokemon pokemon = new Pokemon();
    reader.beginObject();
    while (reader.hasNext()) {
        switch (reader.nextName()) {
            case "name":
                pokemon.name = reader.nextString();
                break;
            case "base_experience":
                pokemon.baseExperience = reader.nextInt();
```

```

        break;
    default:
        reader.skipValue();
    }
}
reader.endObject();
return pokemon;
}

```

```
Pokemon readPokemon(JsonReader reader) throws IOException
```

Funktion readPokemon mit Input JsonReader reader

Rückgabetyp: Pokemon

```
Pokemon pokemon = new Pokemon();
```

Pokemon erstellen (Instanz der Klasse Pokemon mit der Bezeichnung „pokemon“)

Pokemon = Typ der Variable

pokemon = Name der Variable

new Pokemon() = Konstruktor (Besondere Funktion)

```
reader.beginObject();
```

Funktion beginObject() wird aufgerufen für den reader

Es liest die Daten vom Server bis er „{“ (Object) findet

```

while (reader.hasNext()) {
    switch (reader.nextName()) {
        case "name":
            pokemon.name = reader.nextString(); //Formatierung: Strg+Alt+O
Strg+Alt+L
            break;
        case "base_experience":
            pokemon.baseExperience = reader.nextInt();
            break;
        default:
            reader.skipValue();
    }
}

```

hasNext() schaut ob Klammer und Kommas (Symbole heißen Tokens) da sind um das Objekt/Eigenschaften zu lesen => sagt also ob es nächste Eigenschaft/Element gibt

reader.nextName() In dem Objekt sagt uns die Funktion wie die Eigenschaft heißt

Wenn er „name“ liest, wird in pokemon.name der darauffolgende String gespeichert

Wenn er „base\_experience“ liest, wird in pokemon.baseExperience der darauf folgende Integer gespeichert

Ansonsten werden die Werte übersprungen.

```
reader.endObject();
```

Objekt zuende gelesen

```
return pokemon;
```

Die Instanz der Klasse Pokemon mit den zugewiesenen Werten wird ausgegeben.

## Klassen

### Klasse Pokemon

```
public class Pokemon {  
    public String name;  
    public int baseExperience;  
  
    //magic mit: tostring, drei runter und enter (generate via wizard)  
    @Override  
    public String toString() {  
        return "Pokemon{" +  
            "name='" + name + '\'' +  
            ", baseExperience=" + baseExperience +  
            '}';  
    }  
}
```

Variable String name

Variable Integer baseExperience

```
public String toString()
```

Der Wizard erstellt uns die `toString()` Funktion. Die ist da um Dinge in der Konsole schön ausgeben zu können, für die App ist es irrelevant.

### Klasse Berry (Beispiel für Array)

```
public class Berry {  
    public String name = "Unknown";  
    public int size;  
    public int smoothness;  
    public ArrayList<BerryFlavor> flavors = new ArrayList<>();  
  
    @Override  
    public String toString() {  
        return "Berry{" +  
            "name='" + name + '\'' +  
            ", size=" + size +  
            ", smoothness=" + smoothness +  
            ", flavors=" + flavors +  
            '}';  
    }  
}
```

In der Klasse Berry ist eine Liste von Objekten der Klasse BerryFlavor

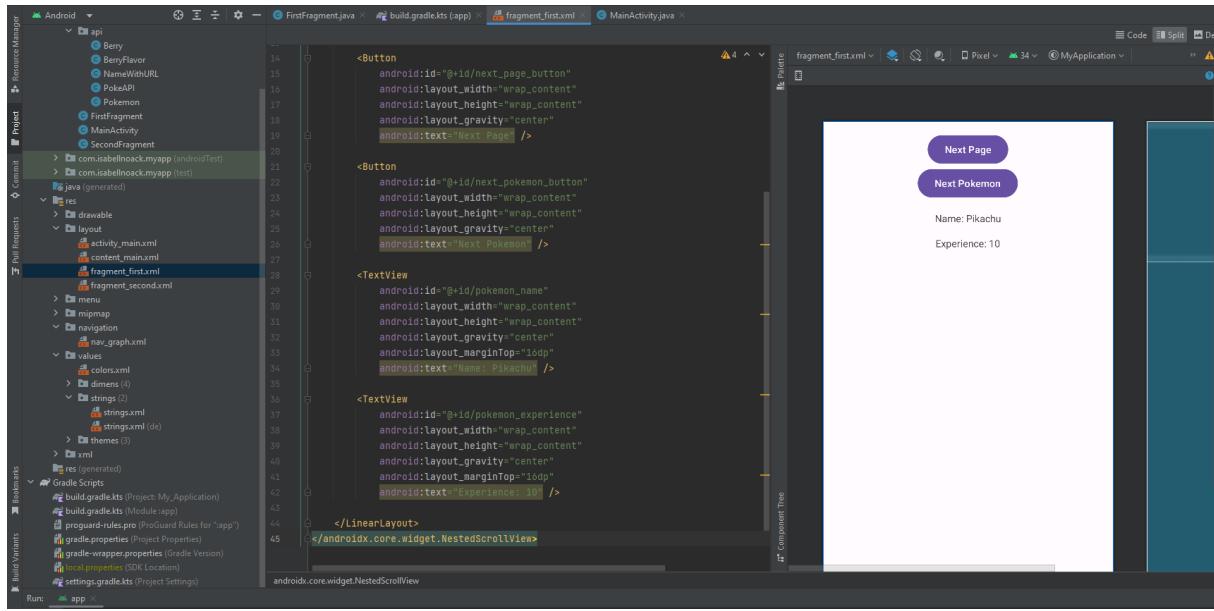
Jede Berry Instanz hat eine eigene Liste (array)

Notiz: Mit „static“ würde es für alle Berry eine gemeinsame Liste geben. Static heißt es braucht keine Instanz

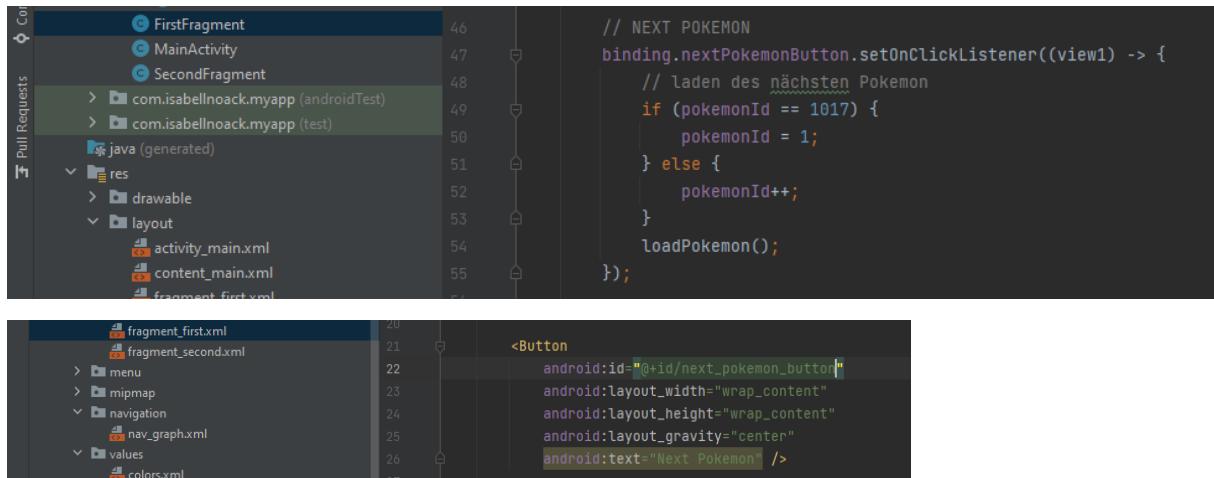
## Feature: Next Pokemon Button

Button angelegt für nächstes Pokemon

Button im Layout angelegt, ID zum Verknüpfen



Click Listener setzen (Funktion, die ausgeführt wird, wenn Button gedrückt wird)



Wir rufen Funktion loadPokemon() auf mit dem binding.nextPokemonButton

**Previous Button hinzugefügt auf die gleiche Art**

**Wrap Around (Bei last Pokemon (1017) -> next Pokemon ist 1)**

Problem: App beim Knopfdruck (Previous Pokemon) abgestürzt, wenn man beim ersten Pokemon war. Ansonsten hat alles geklappt.

Grund: Manche Pokemon haben bei der variablen Base\_Experience den Wert null

```

C:\Users\isabellinoack\AndroidStudioProjects\My_Application\app\src\main\java\com\isabellinoack\myapp\MainFragment.java
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

```

    // NEXT POKEMON
    binding.nextPokemonButton.setOnClickListener((view1) -> {
        // laden des nächsten Pokemon
        if (pokemonId == 1017) {
            pokemonId = 1;
        } else {
            pokemonId++;
        }
        loadPokemon();
    });

    //PREVIOUS POKEMON
    binding.previousPokemonButton.setOnClickListener((view1) -> {
        // laden des vorherigen Pokemon
        if (pokemonId == 1) {
            pokemonId = 1017;
        } else {
            pokemonId--;
        }
        loadPokemon();
    });
}

```

Wrap Around mit if und else:

```

// NEXT POKEMON
binding.nextPokemonButton.setOnClickListener((view1) -> {
    // laden des nächsten Pokemon
    if (pokemonId == 1017) {
        pokemonId = 1;
    } else {
        pokemonId++;
    }
    loadPokemon();
});

//PREVIOUS POKEMON
binding.previousPokemonButton.setOnClickListener((view1) -> {
    // laden des vorherigen Pokemon
    if (pokemonId == 1) {
        pokemonId = 1017;
    } else {
        pokemonId--;
    }

    loadPokemon();
});

```

Lösung:

Bei `readPokemon()` wenn `base_experience` ausgelesen wird, wird zuerst überprüft, ob der Token eine Zahl ist. Wenn nicht, wird der Wert übersprungen. (Hier wird dann der Standard Wert genommen, welcher bei der Deklaration der Variablen angegeben ist.)

```

PokeAPI
Pokemon
FirstFragment
MainActivity
SecondFragment
com.isabellnoack.myapp (androidTest)
com.isabellnoack.myapp (test)
java (generated)
res
drawable
layout
activity_main.xml
content_main.xml
fragment_first.xml
fragment_second.xml
menu
mipmap
navigation
values
xml
res (generated)
Gradle Scripts
build.gradle.kts (Project: My_Application)
build.gradle.kts (Module: app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)

```

```

37
38
39 @
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

//3
1 usage  ↳ ls
Pokemon readPokemon(JsonReader reader) throws IOException {
    Pokemon pokemon = new Pokemon(); //Pokemon erstellen (Instanz der Klasse Pokemon)
    reader.beginObject();
    while (reader.hasNext()) {
        switch (reader.nextName()) {
            case "name":
                pokemon.name = reader.nextString(); //Formatierung: Strg+Alt+0 Strg+Alt+L
                break;

            case "base_experience":
                if (reader.peek() == JsonToken.NUMBER) { //Überprüft ob Token einen Wert hat
                    pokemon.baseExperience = reader.nextInt();
                } else reader.skipValue();
                break;
            default:
                reader.skipValue();
        }
    }
    reader.endObject();
    return pokemon;
}

```

```

case "base_experience":
    if (reader.peek() == JsonToken.NUMBER) { //Überprüft ob Token einen Wert hat
        pokemon.baseExperience = reader.nextInt();
    } else reader.skipValue();
    break;

```

## Layout Type

```

PokeAPI
Pokemon
FirstFragment
MainActivity
SecondFragment
com.isabellnoack.myapp (androidTest)
com.isabellnoack.myapp (test)
java (generated)
res
drawable
layout
activity_main.xml
content_main.xml
fragment_first.xml
fragment_second.xml

```

```

8
9
10
11
12
13
14
15
16
17
18
19
20

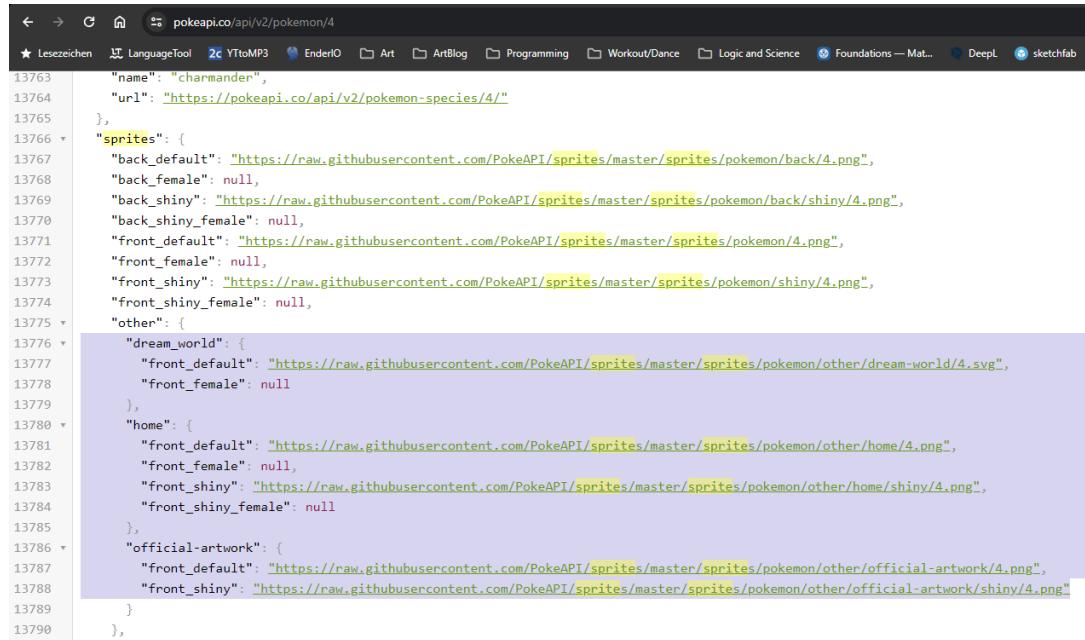
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <Button
        android:id="@+id/next_page_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Next Page" />

```

Constraint Layout Type zu Linear Layout Type geändert

## Sprites



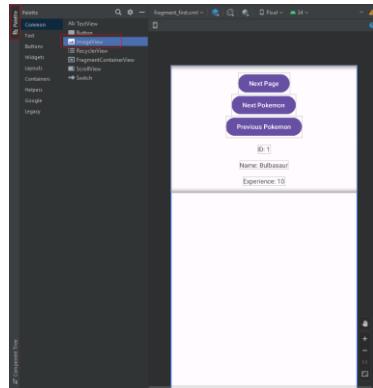
```
13763     "name": "charmander",
13764     "url": "https://pokeapi.co/api/v2/pokemon-species/4/"
13765   },
13766   "sprites": {
13767     "back_default": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/back/4.png",
13768     "back_female": null,
13769     "back_shiny": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/back/shiny/4.png",
13770     "back_shiny_female": null,
13771     "front_default": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/4.png",
13772     "front_female": null,
13773     "front_shiny": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/shiny/4.png",
13774     "front_shiny_female": null,
13775   },
13776   "other": {
13777     "dream_world": {
13778       "front_default": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/dream-world/4.svg",
13779       "front_female": null
13780     },
13781     "home": {
13782       "front_default": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/home/4.png",
13783       "front_female": null,
13784       "front_shiny": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/home/shiny/4.png",
13785       "front_shiny_female": null
13786     },
13787     "official-artwork": {
13788       "front_default": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/4.png",
13789       "front_shiny": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/shiny/4.png"
13790     }
13791   }
13792 }
```

Bei Pokemon Klasse String ergänzt

```
public String imageUrlDreamWorld = "";
```

Im Layout Image View rein (Über die Palette) => GEHT NICHT

⇒ Aber wir wissen, es muss „ImageView“ heißen



```
<ImageView
    android:id="@+id/pokemon_image"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_gravity="center"
    android:layout_marginTop="16dp"
    android:background="@color/black"/>
```

Im FirstFragment machen wir das Binding zum Layout:

```
binding.pokemonImage.setImageBitmap();
```

.setImageBitmap() setzt Bild ohne compression

Als nächstes in der PokeAPI / readPokemon() das ergänzen

```
case "sprites": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

    reader.beginObject(); //BeginObjekt hat immer Reader.hasNext()
    while (reader.hasNext()) {
        switch (reader.nextName()) {
            case "other": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

                reader.beginObject();
                while (reader.hasNext()) {
                    switch (reader.nextName()) {
                        case "dream_world": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

                            reader.beginObject();
                            while (reader.hasNext()) {
                                switch (reader.nextName()) {
                                    case "front_default":
                                        pokemon.imageUrlDreamWorld = reader.nextString();
                                        break; // Weil ganz unten, bitte verlasse das switch
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Dahinter muss jetzt überall noch nen break, Weil ganz unten, bitte verlasse das switch und dann der nächste drüber etc.

```
break;

case "sprites": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

    reader.beginObject(); //BeginObjekt hat immer Reader.hasNext()
    while (reader.hasNext()) {
        switch (reader.nextName()) {
            case "other": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

                reader.beginObject();
                while (reader.hasNext()) {
                    switch (reader.nextName()) {
                        case "dream_world": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

                            reader.beginObject();
                            while (reader.hasNext()) {
                                switch (reader.nextName()) {
                                    case "front_default":
                                        pokemon.imageUrlDreamWorld = reader.nextString();
                                        break; // Weil ganz unten, bitte verlasse das switch
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Als nächstes brauchen wir noch

```
default:
    reader.skipValue();

Überall – Werte die wir nicht kennen werden übersprungen
```

Als nächstes brauchen wir noch das Ende jeder while Schleife

```
reader.endObject();  
Objekt muss geschlossen werden
```

```
case "sprites": //Darunter weitere geschungene Klammer, daher neues BeginObjekt  
  
    reader.beginObject(); //BeginObjekt hat immer Reader.hasNext  
    while (reader.hasNext()) {  
        switch (reader.nextName()) {  
            case "other": //Darunter weitere geschungene Klammer, daher neues BeginObjekt  
  
                reader.beginObject();  
                while (reader.hasNext()) {  
                    switch (reader.nextName()) {  
                        case "dream_world": //Darunter weitere geschungene Klammer, daher neues BeginObjekt  
  
                            reader.beginObject();  
                            while (reader.hasNext()) {  
                                switch (reader.nextName()) {  
                                    case "front_default":  
                                        pokemon.imageUrlDreamWorld = reader.nextString();  
                                        break; // Weil ganz unten, bitte verlasse das switch  
                                    default:  
                                        reader.skipValue();  
                                }  
                            }  
                            reader.endObject();  
                            break;  
                        default:  
                            reader.skipValue();  
                    }  
                }  
                reader.endObject();  
                break;  
            default:  
                reader.skipValue();  
        }  
    }  
    reader.endObject();  
    break;  
default:
```

Ganz unten müssen wir jetzt noch schauen, ob ein Wert null ist

```
if (reader.peek() == JsonToken.STRING) {  
    pokemon.imageUrlDreamWorld = reader.nextString();  
} else reader.skipValue();
```

```

        case "sprites": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

            reader.beginObject(); //BeginObjekt hat immer Reader.hasNext
            while (reader.hasNext()) {
                switch (reader.nextName()) {
                    case "other": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

                        reader.beginObject();
                        while (reader.hasNext()) {
                            switch (reader.nextName()) {
                                case "dream_world": //Darunter weitere geschungene Klammer, daher neues BeginObjekt

                                    reader.beginObject();
                                    while (reader.hasNext()) {
                                        switch (reader.nextName()) {
                                            case "front_default":
                                                if (reader.peek() == JsonToken.STRING) {
                                                    pokemon.imageUrlDreamWorld = reader.nextString();
                                                } else reader.skipValue();
                                                break; // Weil ganz unten, bitte verlasse das switch
                                            default:
                                                reader.skipValue();
                                        }
                                    }
                                    reader.endObject();
                                    break;
                                default:
                                    reader.skipValue();
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Ergebnis:

Im FirstFragment fragen wir nach PokeAPI und fragen nach einem Pokemon. In dem Pokemon wurde die Variable „imageUrlDreamWorld“ ergänzt

Im First Fragment haben wir Binding zum Layout ergänzt mit Funktion setImageBitmap()

Wir haben jetzt nur die URL. Die Bild-Daten müssen noch aus dem Internet runtergeladen werden.

```

//2
2 usages ▲ Isa
JsonReader requestJsonReader(String url) throws IOException { //JSON lesen
    HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();
    connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36");
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(connection.getInputStream(), "charsetName: \"UTF-8\"")); //Datenstrom als UTF-8
    return new JsonReader(bufferedReader); //Text/JSON als Antwort
}

```

Hiermit lesen wir die Daten aus dem Internet. Jetzt lesen wir kein JSON sondern nach dem JSON zusätzlich noch ein Bild

**URL zu BILD**

## ImageLoader Klasse

In PokeAPI ImageLoader Klasse erstellt

```
public static class ImageLoader {
```

```
    public static Bitmap loadImageFromUrl(String imageUrl) throws IOException {
        HttpURLConnection connection = (HttpURLConnection) new URL(imageUrl).openConnection();
        connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36");
```

```

// Öffne einen InputStream zum Lesen der Bildressource
InputStream inputStream = connection.getInputStream();

// Dekodiere den InputStream in ein Bitmap
Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

// Schließe den InputStream
inputStream.close();

return bitmap;
}
}

```

```

My Application - 1 main

Android
app
manifests
java
com.isabelnoack.myapp
    api
        Berry
        BerryFlavor
        NameWithURL
        PokeAPI
        Pokemon
    Fragment
    MainActivity
    SecondFragment
    com.isabelnoack.myapp (androidTest)
    com.isabelnoack.myapp (test)
    java (generated)
    drawable
    layout
        activity_main.xml
        content_main.xml
        fragment_first.xml
        fragment_second.xml
    menu
    mipmap
    navigation
    values
    XML
    res (generated)
    Gradle Scripts

```

```

27
28
29
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
56

    Usage: Alsa
    Jseader requestJsonReader(String url) throws IOException { //JSON lesen
        HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();
        connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36");
        Bufferedreader bufferedreader = new Bufferedreader(new InputStreamReader(connection.getInputStream(), Charset.forName("UTF-8")));
        return new Jsonreader(bufferedReader); //Text JSON als Antwort
    }

    Usage: new
    public static class ImageLoader {
        Usage: new
        public static Bitmap loadImageFromUrl(String imageUrl) throws IOException {
            HttpURLConnection connection = (HttpURLConnection) new URL(imageUrl).openConnection();
            connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36");
            //öffne einen InputStream zum Lesen der Bildressource
            InputStream inputStream = connection.getInputStream();
            // Dekodiere den InputStream in ein Bitmap
            Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
            // Schließe den InputStream
            inputStream.close();
        }
        return bitmap;
    }
}

```

Dann noch im FirstFragment.java das binding ergänzt

```

// Bild laden und in ImageView setzen
try {
    Bitmap bitmap = PokeAPI.ImageLoader.loadImageFromUrl(pokemon.imageUrlDreamWorld); // Hier wird der entsprechenden Bild-URL-Pfad übergeben
    binding.pokemonImage.setImageBitmap(bitmap); //an Layout übergeben
} catch (IOException e) {
    e.printStackTrace(); //Fehlermeldung
}

```

```

My Application - 1 main

Android
app
manifests
java
com.isabelnoack.myapp
    api
        Berry
        BerryFlavor
        NameWithURL
        PokeAPI
        Pokemon
    Fragment
    MainActivity
    SecondFragment
    com.isabelnoack.myapp (androidTest)
    com.isabelnoack.myapp (test)
    java (generated)
    drawable
    layout
        activity_main.xml
        content_main.xml
        fragment_first.xml
        fragment_second.xml
    menu
    mipmap
    navigation
    values
    XML
    res (generated)
    Gradle Scripts

```

```

70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
109

    3 usages ▾ has *
    #SuppressLint("SetTextInBind") //Nicht hIGHLIGHTEN

    void loadPokemon() {
        //Neuer Thread da Hauptthread nicht blockiert werden darf
        new Thread() {
            try {
                Pokemon pokemon = new PokeAPI().requestPokemon(pokemonId); //Funktion der neuen Instanz pokeAPI aufrufen
                getActivity().runOnUiThread() -> {
                    //UI
                    String name = pokemon.name;
                    name = Character.toUpperCase(name.charAt(0)) + name.substring(1); //Erster Charakter groß geschrieben
                    binding.pokemonName.setText("Name: " + name);
                    binding.pokemonExperience.setText("Experience: " + pokemon.baseExperience);
                    binding.pokemonId.setText("ID: " + pokemonId);
                };
            } catch (Exception e) {
                e.printStackTrace(); //Fehlermeldung
            }
        }.start();
    }
}

```

Hier ist der Fehler

Android.os.NetworkOnMainThreadException

Das heisst, es muss wieder auf einem eigenen Thread laufen!!

Daher um das drumrum nochmal neuer Thread bauen:

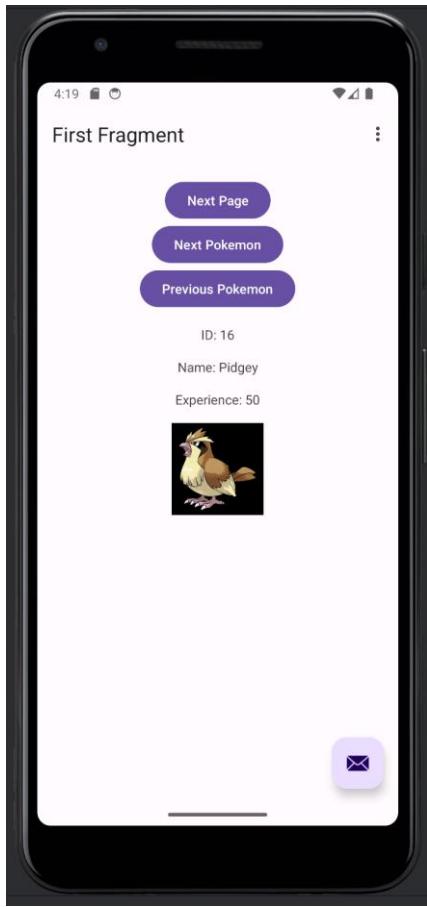
```
// Bild laden und in ImageView setzen
try {
    new *
    new Thread(new Runnable() {
        new *
        @Override
        public void run() {
            try {
                final Bitmap bitmap = PokeAPI.ImageLoader.loadImageFromUrl(pokemon.imageUrlDreamWorld);

                new *
                getActivity().runOnUiThread(new Runnable() {
                    new *
                    @Override
                    public void run() {
                        binding.pokemonImage.setImageBitmap(bitmap);
                    }
                });
            } catch (IOException e) {
                e.printStackTrace(); //Fehlermeldung
            }
        }
    }).start();
} catch (Exception e) {
    e.printStackTrace();
}
```

Bild wird nicht angezeigt weil ich die Sprites von Dream World nehmen wollte, die sind aber leider .svg was der ImageLoader nicht kann...

Hab stattdessen die “official-artwork” genommen, weil das png sind und funktionieren.

Zusätzlich sind die official artworks (FAST) vollständig, die dream world Bilder haben öfter null



## Pokemon Liste bauen

dafür Fragmente neu strukturieren

pokemon first fragment muss aus seite 2 kommen, auf seite first fragment dann die liste mit pokemon

**Umbenennungen:**

First Fragment zu PokemonFragment.java

SecondFragment zu ListFragment.java

fragment\_first.xml zu fragment\_pokemon.xml

fragment\_second.xml zu fragment\_list.xml

FragmentFirstBinding zu FragmentPokemonBinding

FragmentSecondBinding zu FragmentListBinding

ID Umbenennung im nav\_graph

Seiten-Bezeichnung Umbenennung im strings.xml

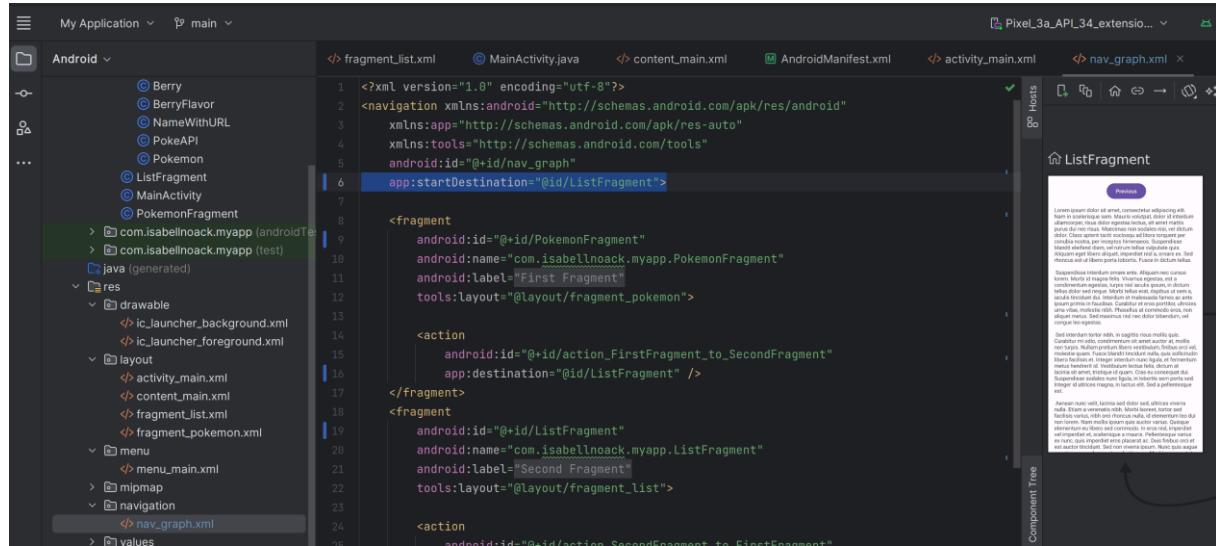
## Appstart mit ListFragment

=> in nav\_graph.xml

```
app:startDestination="@+id/PokemonFragment">
```

Umgeändert zu

```
app:startDestination="@+id/ListFragment">
```



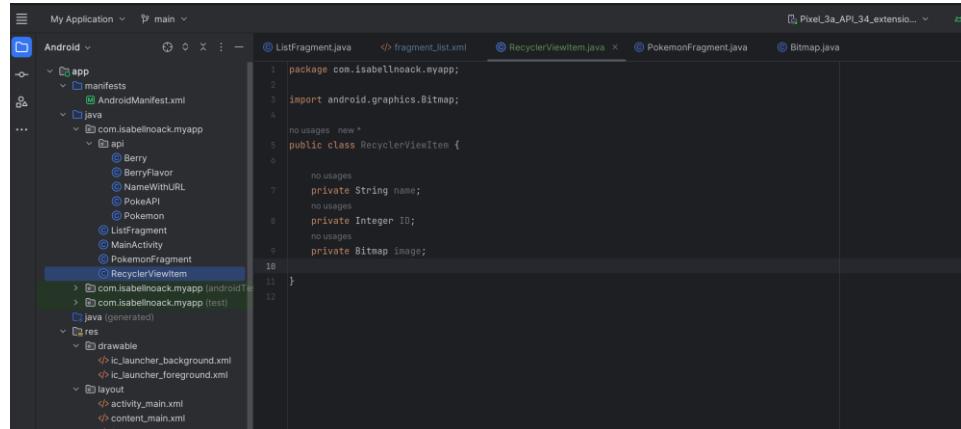
## Liste mit Pokemon Bildern

Anhand dem Tutorial <https://www.youtube.com/watch?v=jTo8aYcd-m8>



=> in Layout (fragment\_list.xml) RecyclerView erstellt

*Neue Klasse: RecyclerViewAdapter*

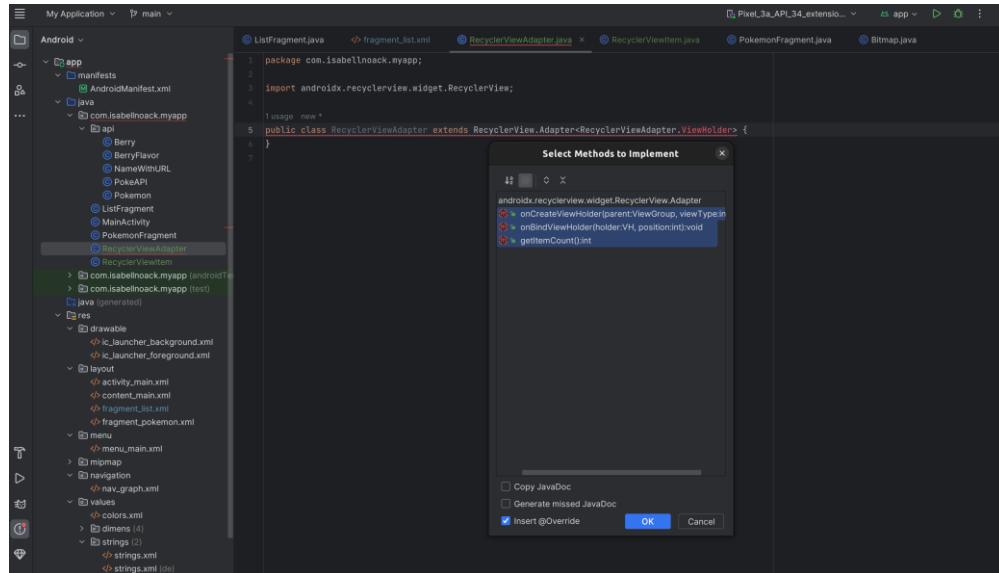


*Klasse angelegt: RecyclerViewAdapter*

Problem: Eine RecyclerList kann kein zeug einfach anzeigen, daher braucht man den Adapter...

Zusammengefasst: Der Adapter bindet Daten an eine RecyclerView, indem er die Daten in ViewHolders lädt, die dann in der RecyclerView angezeigt werden. Der ViewHolder enthält Ansichten für jedes Element, und der Adapter kümmert sich um das Handling der Daten und deren Darstellung.

Klasse RecyclerViewAdapter gebaut (extends RecyclerViewAdapter)



```
3 usages new *
public class ViewHolder extends RecyclerView.ViewHolder {
```

no usages new \*
public ViewHolder(@NotNull View itemView) {
 super(itemView);
}

}

}

```
3 usages new *
public class RecyclerViewAdapter extends RecyclerView.Adapter<RecyclerAdapter.ViewHolder> {
```

no usages
💡 private ArrayList<RecyclerItem> recyclerItems; //ArrayList<typ> variablen-name

Klasse ViewHolder gebaut (extends RecyclerView.ViewHolder)

Danach noch ArrayList aus den Items

Wie viele Items ich in meiner RecyclerView habe

```

new *

@Override
public int getItemCount() {
    return recyclerViewItems.size();
}

```

```

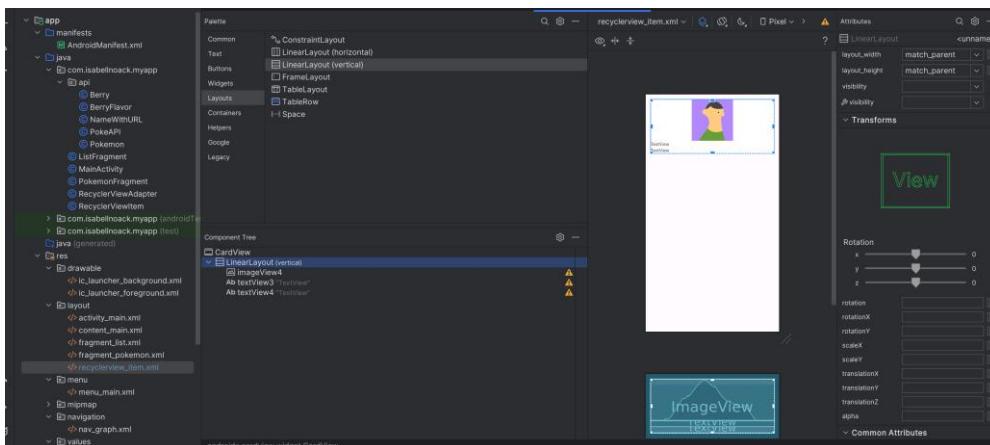
1 usage
private ArrayList<RecyclerViewItem> recyclerViewItems; //ArrayList<typ> variablen-name

//für was wird das verwendet?
no usages
private Context context;

new *
@NonNull
@Override

public RecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.recyclerview_item,parent, attachToRoot: false);
    return new ViewHolder(view);
}

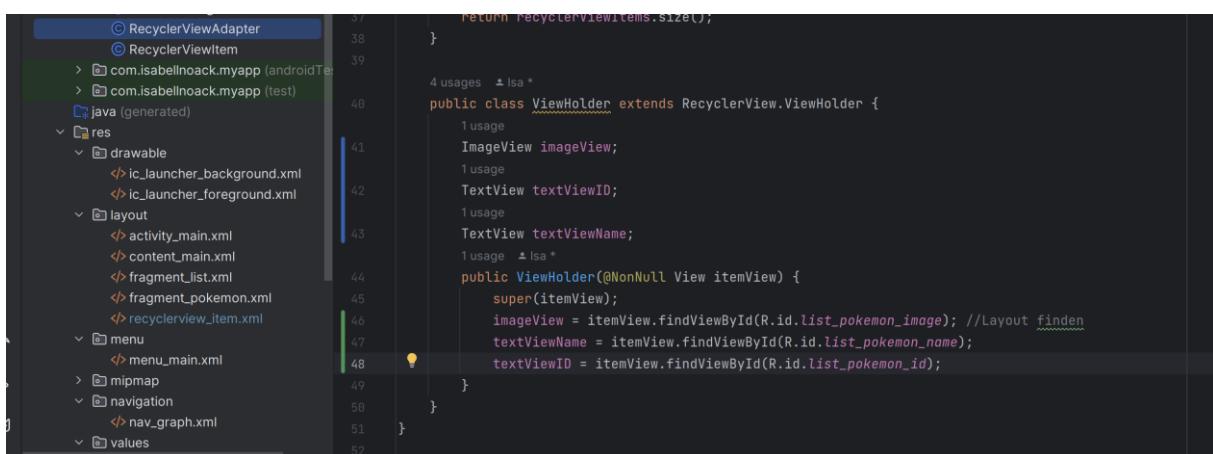
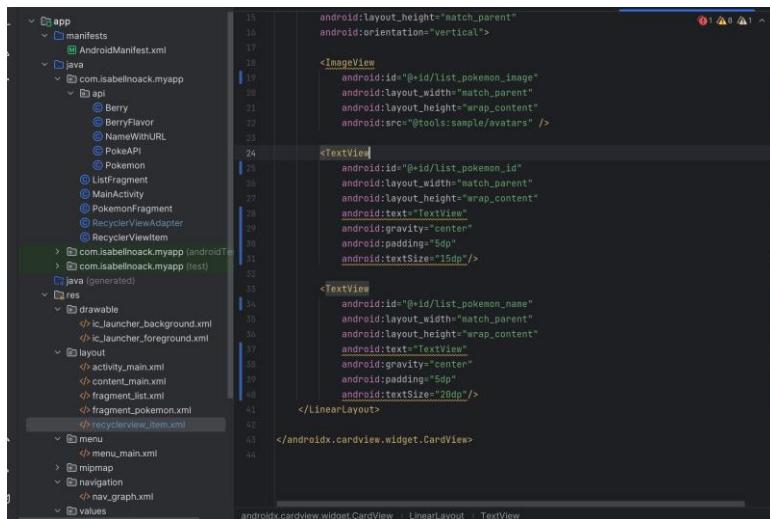
```



Neues Layout für die Items erstellt: recyclerview\_item

Das in der Klasse verknüpfen mit der jeweiligen ID (Funktion: findViewById)

für jedes der Elemente in meinem Layout



ausserdem den Holder erstellen für jedes der Elemente

und anderer kram den ich nicht verstehne

- ListFragment versteh ich nicht und das RecyclerViewItem und wie das alles zusammen kommt

## Liste mit Pokemon Bildern – 1,2, oder 3 Spalten nebeneinander

im ListFragment.java

```

//Hier wird der GridLayoutManager mit 3 Spalten erstellt
int numberOfColumns = 3;
GridLayoutManager layoutManager = new GridLayoutManager(context, numberOfColumns);
binding.recyclerView.setLayoutManager(layoutManager);

```

=> jetzt muss die grössze der Bilder und Texte angepasst werden

=> erstmal wenn 1 Column, dann soll die textView nebeneinander sein und ansonsten unter einander

Im bestehenden RecyclerViewAdapter-CodeAnpassungen vornehmen, um die Anordnung der TextViews basierend auf der Anzahl der Spalten zu ändern.

Alles im RecyclerViewAdapter!

The screenshot shows the Android Studio interface with the project tree on the left and the code editor on the right. The code editor displays the `RecyclerViewAdapter.java` file. The code defines a class `RecyclerViewAdapter` that extends `RecyclerView.Adapter<RecyclerView.ViewHolder>`. It has a private field `recyclerViewItems` of type `ArrayList<RecyclerViewItem>`, a private field `context` of type `Context`, and a private field `numberOfColumns` of type `int`. The constructor initializes these fields with their respective parameters. A note in the code indicates that `numberOfColumns` is a new variable for adding columns.

```

import ...
public class RecyclerViewAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    // Adapter als Übersetzer von der RecyclerView zu meinen Daten und zu Layout
    // Deklarierung von zwei private Variablen: recyclerViewItems ist eine Liste für die anzulegenden Daten,
    // und context wird für den Zugriff auf Android-Ressourcen verwendet
    private ArrayList<RecyclerViewItem> recyclerViewItems; //ArrayList<typ> variablen-name
    private Context context;
    private int numberOfColumns; // NEU: Variable für die Anzahl der Spalten hinzufügen
    //Konstruktor initialisiert die Klassenvariablen mit den übergebenen Werten
}

```

```

//Konstruktor initialisiert die Klassenvariablen mit den übergebenen Werten
1 usage  ↗ Isa *
public RecyclerViewAdapter(ArrayList<RecyclerViewItem> recyclerViewItems, Context context) {
    this.recyclerViewItems = recyclerViewItems;
    this.context = context;
    this.numberOfColumns = numberOfColumns; // NEU: Anzahl der Spalten setzen
}

```

Die TextViews finden wir durch unsere eigene Definition, wie die ViewHolder Klasse aussieht:  
textViewName und textViewID wird hier zu den Ressourcen (layout) vernetzt

Was wir alles ändern müssen um von 1 Spalte zu 2 oder mehr zu gehen:

- Layout orientation von horizontal zu vertical
- in den TextViews die android:layout\_width="0dp" zu ="match\_parent"

Für das Layout: Dafür auch im ViewHolder das Layout connected über die ID layoutchanged

The screenshot shows the code for the `ViewHolder` class, which extends `RecyclerView.ViewHolder`. The class contains fields for an `ImageView` (`imageView`), three `TextViews` (`textViewID`, `textViewName`, `textViewName`), and a `LinearLayout` (`layoutChanged`). The constructor takes a `View itemView` parameter and initializes the view fields. The `layoutChanged` field is specifically noted as being used for layout orientation changes.

```

// Definition ViewHolder Klasse
4 usages  ↗ Isa *
public class ViewHolder extends RecyclerView.ViewHolder {
    2 usages
    ImageView imageView;
    4 usages
    TextView textViewID;
    4 usages
    TextView textViewName;
    3 usages
    LinearLayout layoutChanged; // Layout Orientation
    1 usage  ↗ Isa *
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        imageView = itemView.findViewById(R.id.list_pokemon_image); //Layout finden
        textViewName = itemView.findViewById(R.id.list_pokemon_name);
        textViewID = itemView.findViewById(R.id.list_pokemon_id);
        layoutChanged = itemView.findViewById(R.id.layoutchanged); // Layout Orientation
    }
}

```

```

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

```

    // Anpassung der Anordnung der TextViews basierend auf der Anzahl der Spalten
    if (numberOfColumns == 1) {
        // Wenn es nur eine Spalte gibt, setze die TextViews nebeneinander und das Layout horizontal
        holder.textViewName.getLayoutParams().width = 0; // layout_width auf 0dp setzen
        holder.textviewID.getLayoutParams().width = 0; // layout_width auf 0dp setzen
        holder.layoutChanged.setOrientation(LinearLayout.HORIZONTAL);
    } else {
        // Wenn es mehr als eine Spalte gibt, setze die TextViews untereinander und das Layout vertikal
        holder.textViewName.getLayoutParams().width = ViewGroup.LayoutParams.MATCH_PARENT; // layout_width auf match_parent setzen
        holder.textviewID.getLayoutParams().width = ViewGroup.LayoutParams.MATCH_PARENT; // layout_width auf match_parent setzen
        holder.layoutChanged.setOrientation(LinearLayout.VERTICAL);
    }
}

```

## Liste mit allen Pokemon – Button erstellen für die Spalten-Änderung

Button erstellen, welches den parameter ändert. Danach muss die Liste geupdated werden.

```

<Button
    android:id="@+id/button_numberOfColumns"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Columns"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/button_search"
    app:layout_constraintTop_toTopOf="parent"/>

```

Wenn Button geklickt wird:

```

//Button Columns
new *
binding.buttonNumberOfColumns.setOnClickListener(new View.OnClickListener() {
    new *
    @Override
    public void onClick(View view) {
        if (numberOfColumns == 3) {
            numberOfColumns = 1;
        } else {
            numberOfColumns++;
        }
        updateRecyclerView();
    }
});

```

dafür die Funktion updateRecyclerView(); erstellt!

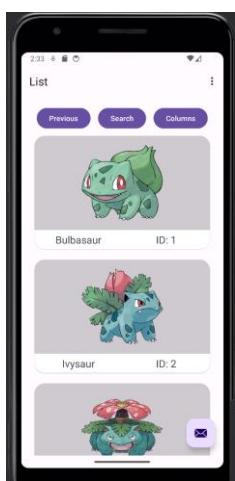
```

// Methode zur Aktualisierung der RecyclerView mit der neuen Anzahl von Spalten
1usage new *
private void updateRecyclerView() {
    GridLayoutManager layoutManager = new GridLayoutManager(getContext(), numberColumns);
    binding.recyclerView.setLayoutManager(layoutManager);

    // Adapter für die RecyclerView erneut setzen
    RecyclerViewAdapter recyclerViewAdapter = new RecyclerViewAdapter(recyclerViewItems, getContext(), numberColumns);
    binding.recyclerView.setAdapter(recyclerViewAdapter);
}

```

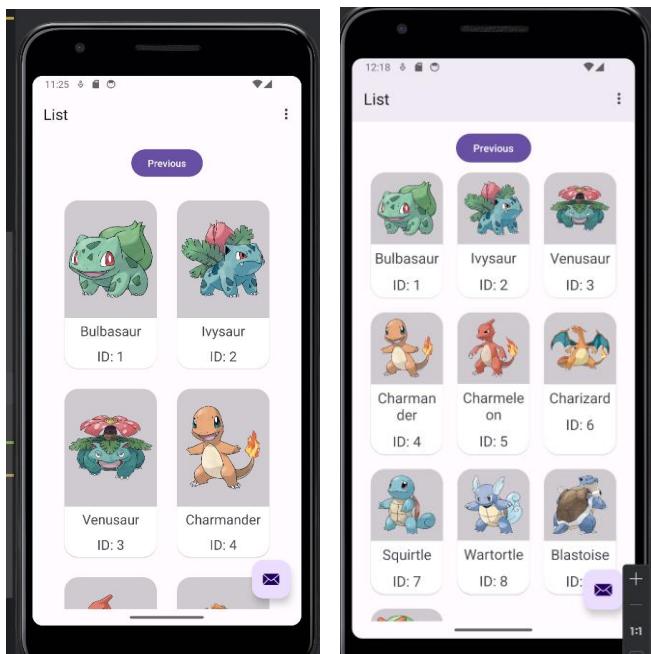
Funktioniert alles wie gewollt, durch drücken des Buttons „Columns“ wird zwischen 1,2 und 3 Spalten gewechselt.



### Liste mit allen Pokemon – mit Schleife

Höhe noch angepasst im XML

```
android:adjustViewBounds="true"
```



```

//Schleife, Listen Eintrag für alle Pokemon ID von 1 bis 1017
for (int id = 1; id <= 10; id++) {
    recyclerViewItems.add(new RecyclerViewItem(id)); //Listen Eintrag
}
//To-Do: performantes laden!

```

## Zu Große Pokemon-Abfrage auf einmal

Problem: wenn ich alle 1017 Pokemon vorlade – braucht die App mehrere Minuten bis etwas angezeigt wird!

Bei 100 Pokemon hat es bereits 40sek gedauert.

=> Cashing oder offset und limit stuff?

**Stattdessen Code umgebaut:**

Man hat keine ArrayListe „recyclerViewItems“ mehr, die alle Pokemon an RecyclerViewAdapter gibt.

Jetzt machen wir es so, dass der Adapter sich selber zieht was er braucht. Damit sollte er nur das anfragen, was man auf dem Handy sieht. Und nicht alles auf einmal.

Das Laden ist jetzt im Adapter (onBindViewHolder) statt ListFragment

=> ArrayList (aus dem listFragment) gelöscht und statt dass wir das bei erstellen von dem ViewHolder übergeben, ist das jetzt direkt im ViewHolder drinnen

```

75      //Array Liste neues Item hinzufügen
-      try {
-          //Schleife, Listen-Eintrag für alle Pokemon ID von 1 bis 1017
-          for (int id = 1; id <= 10; id++) {
-              recyclerViewItems.add(new RecyclerViewItem(id)); //Listen Eintrag
-          }
-          //To-Do: performantes laden!
-
-          } catch (IOException ignored) {
-              // todo send warning to user
-          }
-
```

das Array recyclerViewItems wird jetzt auch nicht mehr gebraucht

Jetzt neuer Code im RecyclerViewAdapter

```

56 +
57 +     int id = position + 1;
58 +
59 +     Thread t = new Thread(() -> {
60 +         try {
61 +             final RecyclerViewItem item = new RecyclerViewItem(id);
62 +             activity.runOnUiThread(new Runnable() {
63 +                 @Override
64 +                 public void run() {
65 +
66 +                     //holder.imageView.setImageURI(Uri.parse(item.pokemon.imageUrl));
67 +                     holder.imageView.setImageBitmap(item.image);
68 +                     holder.textViewID.setText("ID: " + item.id.toString()); //String mit "ID:" + ID
69 +                     holder.textViewName.setText(item.name);
70 +                 }
71 +             });
72 +
73 +         } catch (IOException e) {
74 +             throw new RuntimeException(e);
75 +         }
76 +     });
77 +
78 +     t.start();

```

Die Pokemon werden jetzt anhand der Position hier mit ID genommen

Code: final RecyclerViewItem item = new RecyclerViewItem(id);

## Pokemon IDs 1025 < 1302

Es gibt 1302 Pokemon

Pokemon mit ID 1013 hat kein Sprite!

Pokemon ab ID 1018 haben auch kein Sprite!

Ausserdem sind die IDs ab 1025 im 10.000-Bereich. Das sind anscheinend besondere Formen der Pokemon. Bestimmte entwicklungen, in verschiedenen Regionen etc.

### Keine Sprites - Error Behandlung

Pokemon mit ID 1013 hat kein Sprite!

Für das ListFragment:

Im RecyclerViewAdapter:

```

    //holder.imageView.setImageURI(Uri.parse(item.pokemon.imageUrl));
    if (item.image != null) {
        holder.imageView.setImageBitmap(item.image);
    } else {
        holder.imageView.setImageResource(R.drawable.empty);
    }
}

```

statt dem Bild wird das .xml „empty“ gezeigt

Für das PokemonFragment: (Im PokemonFragment)

Wenn das Bild gesetzt wird überprüfen ob es "" hat und wenn, dann empty.xml setzen

```

if (pokemon.imageUrl != "") { //wenn Sprite null > in RecyclerViewItem übersprungen > Standard Wert ""
    final Bitmap bitmap = PokeAPI.ImageLoader.loadImageFromUrl(pokemon.imageUrl);
    getActivity().runOnUiThread(new Runnable() {
        @Override
        public void run() {
            binding.pokemonImage.setImageBitmap(bitmap);
        }
    });
} else { //Hier statt "" dann empty.xml setzen
    binding.pokemonImage.setImageResource(R.drawable.empty);
}

```

empty.xml erstellen aus svg image:

Im ordner drawable rechtsklick => new => Vector Asset => Pfad auswählen

## App Icon

In der AndroidManifest.xml

```

android:icon="@mipmap/ic_pokeapp"
android:roundIcon="@mipmap/ic_pokeapp"

```

Bild benennung: ic\_pokeapp.png (lowercase)

## Pokemon Details aufrufen

**Von ListFragment zu Pokemon Fragment**

Pokemon-ID beim Klicken auf ein RecyclerView-Element an das PokemonFragment übergeben - über die Navigation mit dem NavController

### In RecyclerViewAdapter

In der ViewHolder-Klasse Methode hinzufügen (zum Öffnen des PokemonFragments)

```

147
148     //Methode openPokemonFragment für ListFragment zu PokemonFragment
149     usage new *
150     public void openPokemonFragment(int pokemonId) {
151         NavController navController = Navigation.findNavController(itemView);
152         Bundle bundle = new Bundle();
153         bundle.putInt("pokemonId", pokemonId);
154         navController.navigate(R.id.action_listFragment_to_pokemonFragment, bundle);
155     }

```

In **nav\_graph.xml** action benutzen/umbenannt in:  
android:id="@+id/action\_listFragment\_to\_pokemonFragment"

### OnClickListener für die CardView

```

106
107     // Von ListFragment zu PokemonFragment
108     holder.cardView.setOnClickListener(e -> {
109         // Bei einem Klick auf das CardView-Element in der RecyclerView wird dieser Listener aufgerufen.
110         // Hier wird die Position des geklickten Elements im RecyclerView-Adapter abgerufen und verwendet,
111         // um die entsprechende Pokemon-ID zu bestimmen.
112
113         int pokémonId = position + 1;
114         holder.openPokemonFragment(pokémonId); //Methode OpenPokemonFragment vom Holder wird aufgerufen mit der PokémonId
115     });
116 }

```

=> Mit Klick von ListFragment zu PokemonFragment funktioniert aber die ID wird nicht richtig übergeben

=> Die Methode openPokemonFragment muss angepasst werden, aber wie?

=> Man kann eine statische Variable für die ganze App festlegen.

*public static int pokémonIdToOpen;*

- public: von überall aus dem Programm zugänglich

- static: Statische Methoden können aufgerufen werden, ohne dass eine Instanz der Klasse erstellt werden muss;

- int: Datentyp

Variable definiert in der MainActivity.java

*public static int pokémonIdToOpen;*

RecyclerViewAdapter angepasst mit der neuen Variable

```

holder.cardView.setOnClickListener(e -> {
    pokémonIdToOpen = position + 1;
    holder.openPokemonFragment(pokémonIdToOpen);
});

```

Im PokemonFragment die pokémonId zu *pokémonIdToOpen* anstatt festem Startwert

*int pokémonId = pokémonIdToOpen;*

Aber Problem für später: Der Wert pokémonIdToOpen kommt aus der CardView Position. Die Position ist ja aber falsch wenn man die Liste nicht nach ID sortiert sondern nach Namen oder anderem.

## Handy Crash: null object reference

er wurde keine Activity gefunden.

Exception:

FATAL EXCEPTION: Thread-28  
com.isabellnoack.myapp, PID: 19749

Process: java.lang.NullPointerException: Attempt to

invoke virtual method 'void  
androidx.fragment.app.FragmentActivity.runOnUiThread(java.lang.Runnable)' on a null object  
reference

at

com.isabellnoack.myapp.PokemonFragment.lambda\$loadPokemon\$3\$com-isabellnoack-myapp-  
PokemonFragment(PokemonFragment.java:75)

at

com.isabellnoack.myapp.PokemonFragment\$\$ExternalSyntheticLambda3.run(Unknown Source:2)

at

java.lang.Thread.run(Thread.java:929)

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** My Application - PokemonFragment.java [My\_Application.app.main]
- Code Editor:** Shows the PokemonFragment.java file with the error line highlighted at line 75: `getActivity().runOnUiThread(() -> {`.
- Logcat:** Displays the error stack trace and log messages. The error message is:

```
E FATAL EXCEPTION: Thread-28
Process: com.isabellnoack.myapp, PID: 19749
java.lang.NullPointerException: Attempt to invoke virtual method 'void androidx.fragment.app.FragmentActivity.runOnUiThread(java.lang.Runnable)' on a null object reference
    at com.isabellnoack.myapp.PokemonFragment.lambda$loadPokemon$3$com-isabellnoack-myapp-PokemonFragment(PokemonFragment.java:75)
    at com.isabellnoack.myapp.PokemonFragment$$ExternalSyntheticLambda3.run(Unknown Source:2)
    at java.lang.Thread.run(Thread.java:929)
```
- Running Devices:** A sidebar showing device mirroring options.

Lösung anscheinend, statt getActivity() im Thread aufzurufen, zuvor definieren



This happened when you call `getActivity()` in another thread that finished after the fragment has been removed. The typical case is calling `getActivity()` (ex. for a `Toast`) when an HTTP request finished (in `onResponse` for example).



To avoid this, you can define a field name `mActivity` and use it instead of `getActivity()`. This field can be initialized in `onAttach()` method of Fragment as following:



```
@Override  
public void onAttach(Context context) {  
    super.onAttach(context);  
  
    if (context instanceof Activity){  
        mActivity =(Activity) context;  
    }  
}
```

In my projects, I usually define a base class for all of my Fragments with this feature:

```
public abstract class BaseFragment extends Fragment {  
  
    protected FragmentActivity mActivity;  
  
    @Override  
    public void onAttach(Context context) {  
        super.onAttach(context);
```

## Kleine Feinheiten von der UI/UX/Navigation:

1. Wenn ich im PokemonFragment die Pokemon durchklicke, soll beim nächsten Mal genau das Pokemon wieder sichtbar sein, wenn man im ListFragment auf den Button Pokemon drückt:

dafür beim Button Next und Previous die Variable „pokemonIdToOpen“ auch updaten.

The screenshot shows the Android Studio project structure on the left and the code for `PokemonFragment.java` on the right. The code handles button click listeners for navigating between pokemons.

```

43     //Button NEXT POKEMON
44     binding.nextPokemonButton.setOnClickListener((view1) -> {
45         // laden des nächsten Pokemon
46         if (pokemonId == 1025) { // 1017 ist das letzte Pokemon
47             pokemonId = 1;
48             pokemonIdToOpen = 1;
49         } else {
50             pokemonId++;
51             pokemonIdToOpen++;
52         }
53         loadPokemon();
54     });
55
56     //Button PREVIOUS POKEMON
57     binding.previousPokemonButton.setOnClickListener((view1) -> {
58         // laden des vorherigen Pokemon
59         if (pokemonId == 1) {
60             pokemonId = 1025;
61             pokemonIdToOpen = 1025;
62         } else {
63             pokemonId--;
64             pokemonIdToOpen--;
65         }
66         loadPokemon();
67     });
68
69     // Pokemon anhand ID wird geladen
70     loadPokemon();
71
72 }
73
74 }

```

2. Wenn man vom `PokemonFragment` zum `ListFragment` wechselt, soll die `RecyclerView` zum richtigen Item scrollen

Fehlermeldung:

Cannot scroll to position a LayoutManager set. Call `setLayoutManager` with a non-null argument.

=> Fehler konnte auf die schnelle nicht gelöst werden, daher erstmal weg gelassen

## PokemonFragment Design

- Aussehen (Farben)
- Inhalte
- Aufbau

In Figma Design erstellt

## Weitere Infos PokemonFragment – Weight, Height, Types

=> Ziel Weight, Height und die Types (Pokemon Art)

=> Pokemon Klasse ergänzt

```

public int height = 0;
public int weight = 0;
public ArrayList<NameWithURL> types = new ArrayList<>();

```

=> in der PokeAPI mit abgefragt

```

case "weight":
    if (reader.peek() == JsonToken.NUMBER) {
        pokemon.weight = reader.nextInt();
    } else reader.skipValue();
    break;

```

```

case "height":
    if (reader.peek() == JsonToken.NUMBER) {
        pokemon.height = reader.nextInt();
    } else reader.skipValue();
    break;

case "types":
    reader.beginArray();
    while (reader.hasNext()) {
        reader.beginObject();
        while (reader.hasNext()) {
            switch (reader.nextName()) {
                case "type":
                    pokemon.types.add(readNameWithURL(reader));
                    break;
                default:
                    reader.skipValue();
            }
        }
        reader.endObject();
    }
    reader.endArray();
    break;

```

=> in PokemonFragment Binding erstellt

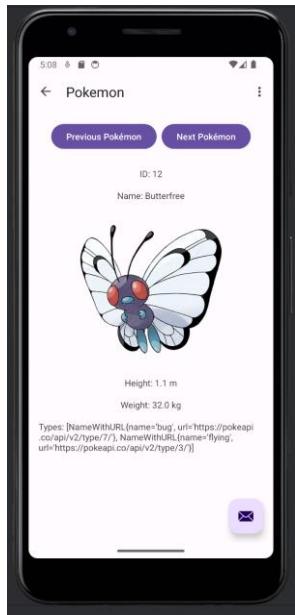
```

if (pokemon.weight != 0) {
    binding.pokemonWeight.setText("Weight: " + (pokemon.weight / 10.0f) + " kg"); //Angabe in
    Hectogramm // ausserdem mit float multiliziert, für Kommastellen
} else {
    binding.pokemonWeight.setText("Weight: undefined");
}

if (pokemon.height != 0) {
    binding.pokemonHeight.setText("Height: " + (pokemon.height / 10.0f) + " m"); //Angabe in Decimeter
} else {
    binding.pokemonHeight.setText("Height: undefined");
}

binding.pokemonTypes.setText("Types: " + pokemon.types);

```



Problem: Array wird noch nicht richtig angezeigt (soll nur die Namen zeigen, nicht alle Informationen aus dem Array)

=> for-each Schleife, um den Namen in einen anderen String zu speichern

```
//pokemon.types, nur name aus dem array ausgeben
String types = "none";
for (NameWithURL type : pokemon.types) {
    //for (Typ variablen-name : Datenquelle)
    if (types.equals("none")) {
        types = capitalize(type.name);                                //Wert überschreiben mit Pokemon Type
    } else {
        types = types + ", " + capitalize(type.name);      //Pokemon Type anhängen
    }
}
binding.pokemonTypes.setText("Types: " + types);
```

## IO Exception behandeln

IO Exception wenn man kein Internet hat oder dieses abbricht, oder Server ausfällt, File not found, etc.

```
} catch (IOException ignored) { //Hier im Download Thread, um etwas anzeigen zu können in den UI Thread wechseln
    // todo send warning to user
}
```

Hier im Download Thread, um etwas anzeigen zu können in den UI Thread wechseln

```
activity.runOnUiThread(() -> { //jetzt auf UI Thread
    // todo send warning to user
});
```

Jetzt Fehlermeldung anzeigen:

Klasse “Toast” benutzen – Fehlermeldung als kleiner Kasten der aufploppt

Das Toast zeigt die passende Exception für uns an!!

– Das macht uns aus dem Fehlertext lesbare UI

```
} catch (IOException exception) { //Hier im Download Thread, um etwas anzeigen zu können in den UI Thread wechseln
    activity.runOnUiThread(() -> { //jetzt auf UI Thread
        // send warning to user
        Toast.makeText(activity, exception.toString(),
        Toast.LENGTH_LONG).show(); //Toast Klasse mit: context(so anzeigen: activity ; Fehler als String anzeigen; Wie lange angezeigt)
    });
}
```

Der Fehlertext kommt aus exception.toString()

## Weitere Infos PokemonFragment - Ability

Neue Klasse Ability

```
public ArrayList<Ability> abilities = new ArrayList<>();
```

```
public class Ability {
    public NameWithURL nameWithURL;
    public boolean isHidden;

    @Override
    public String toString() {
        return "Ability{" +
            "nameWithURL=" + nameWithURL +
            ", isHidden=" + isHidden +
            '}';
    }
}
```

=> Datenstruktur fertig definiert

In PokeAPI, Werte in Datenstruktur füllen

```
case "abilities":
    reader.beginArray();
    while (reader.hasNext()) {
        pokemon.abilities.add(readAbility(reader));
    }
    reader.endArray();
    break;
```

neue Funktion damit der Text übersichtlicher ist

```
Ability readAbility(JsonReader reader) throws IOException { //Rückgabewert
Ability
    Ability ability = new Ability();
    reader.beginObject();
    while (reader.hasNext()) {
        switch (reader.nextName()) {
```

```

        case "ability":
            ability.nameWithURL = readNameWithURL(reader);
            break;
        case "is_hidden":
            ability.isHidden = reader.nextBoolean();
            break;
        default:
            reader.skipValue();
    }
}
reader.endObject();
return ability;
}

```

=> Parsing fertig

Jetzt Darstellung (UI)

Layout und PokemonFragment Binding

Binding:

```

//pokemon.abilities, nur name aus dem array ausgeben, und nach hidden filtern
String abilities = "none";
int numberofAbilities = 0;
for (Ability ability : pokemon.abilities) {
    NameWithURL nameWithURL = ability.nameWithURL;
    if (ability.isHidden) {
        continue;
    }
    //macht mit dem for weiter, und ignoriert damit die Ability wenn isHidden true
    numberofAbilities++;
    if (abilities.equals("none")) {
        abilities = capitalize(nameWithURL.name);
    } else {
        abilities = abilities + ", " + capitalize(nameWithURL.name);
    }
}
if (numberofAbilities == 1) {
    binding.pokemonAbilities.setText("Ability: " + abilities);
} else {
    binding.pokemonAbilities.setText("Abilities: " + abilities);
}

```

## Farbanpassungen Buttons

In Ordner values – colors definieren (colors.xml)

In Unterordner themes – colors zuweisen (themes.xml und themes.xml night version anpassen)

## Schütteln-Sensor

Um schütteln zu erkennen: Accelerometer

SensorManager integriert mithilfe <https://www.youtube.com/watch?v=gVsXHio7hU> und PDF

## Code – Button ausführen

```
•     public MainActivityFragment() {      }
•     @Override
•     public View onCreateView(LayoutInflater inflater, ViewGroup container,
•         Bundle savedInstanceState) {
•         mSensorManager = (SensorManager) getActivity().getSystemService(Context.SENSOR_SERVICE);
•         mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
•
•         View view = inflater.inflate(R.layout.fragment_main, container, false);
•         Button button = (Button) view.findViewById(R.id.btnGPSlocation);
•         button.setOnClickListener(new View.OnClickListener() {
•             @Override
•             public void onClick(View v) {
•                 // do something
•             }
•         });
•         return view;
•     }
```

Prof. Dr.-Ing. Michael Stepping, Cloud and Mobile Computing

157

[www.eah-jena.de](http://www.eah-jena.de)

## Sensor Manager

```
SensorManager sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL);
```

Um den Code ein try und catch um eine Fehlermeldung zu behandeln: falls das Handy keinen Sensor oder SensorManager hat (NullPointerException) dann soll dieses nicht abstürzen sondern einfach nichts machen (ignored)

Im Pokemon Fragment noch implements SensorEventListener ergänzt

```
public class PokemonFragment extends Fragment implements SensorEventListener {
```

## Schütteln erkennen

Erklärung in den Kommentaren

```
//Sensor
private static final float SHAKE_THRESHOLD = 30.0f; //g Beschleunigung
private long lastShakeTime;

@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        binding.sensor.setText("X: " + event.values[0] + ", Y: " +
event.values[1] + ", Z: " + event.values[2]);
    }

    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        // Beschleunigungssensors Werte
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        // Absolute Beschleunigung berechnen (Betrag der Differenz zur Erd-
```

```

beschleunigung)
    // .abs = absoluter Betrag: heißt es geht in + oder -
    float acceleration = Math.abs(x) + Math.abs(y) + Math.abs(z) -
(SensorManager.GRAVITY_EARTH); //Gravitation 9,81 m/s2, auf Z wenn es flach
auf Tisch liegt

    // Die aktuelle Zeit in Millisekunden
    long currentTime = System.currentTimeMillis();

    // Überprüfe, ob die absolute Beschleunigung den Schwellenwert für
ein Schütteln überschreitet
    if (acceleration > SHAKE_THRESHOLD) {

        // Cool-Down: Überprüfe, ob seit dem letzten Schütteln eine be-
stimmte Zeit vergangen ist
        if (currentTime > lastShakeTime + 1000) {           //Momentane
Zeit ist größer als letzteShake Zeit + 1sek
            onShakeDetected();                                //Wenn ja, wird
Methode aufgerufen
            lastShakeTime = currentTime;
        }
    }
}
}

```

Wenn schütteln erkannt wurde, dann: (Toast zum Testen)

```

//Schütteln erkannt
private void onShakeDetected() {
    Toast.makeText(getApplicationContext(), "Shake Detected!",
Toast.LENGTH_SHORT).show();
}

```

### Pokemon schütteln

Den Toast jetzt dazu ändern, dass sich stattdessen das Pokemon Bild bewegt 😊

=> Damit sich das Pokemon Bild schüttelt: ImageView soll animiert werden

360 Rotation:

```

ObjectAnimator rotationAnimator = ObjectAnimator-
ofFloat(binding.pokemonImage, "rotation", 0f, 720f);
rotationAnimator.setDuration(1000); // Dauer der Rotation in Millisekunden
(hier 500 Millisekunden)
rotationAnimator.start();

```

Stattdessen hin und her wackeln, ObjectAnimation Zeile austauschen:

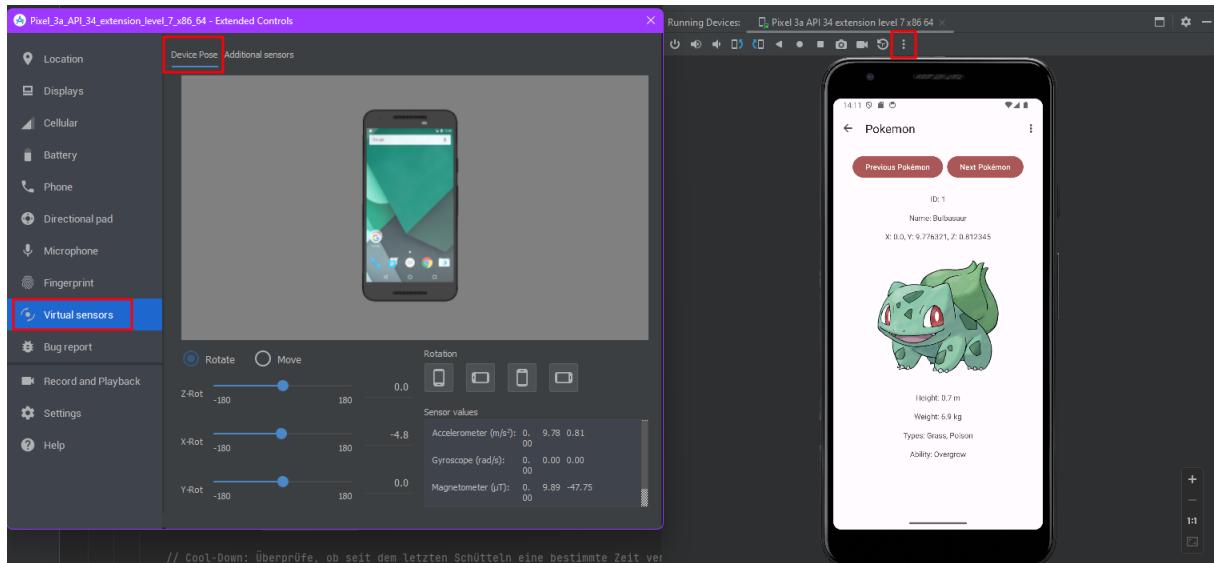
```

ObjectAnimator rotationAnimator = ObjectAnimator.ofFloat(
binding.pokemonImage, "rotation", 0f, -20f, 20f, -20f, 0f);

```

### Simulierte Handy testen

Schütteln des Handys testen über: (Eventuell SHAKE\_THRESHOLD anpassen)



## Pokemon Species: Neue URL einbauen

<https://pokeapi.co/api/v2/pokemon-species/1/>

Das will ich abfragen, dafür kann ich vieles von der Pokemon Abfrage für die PokemonSpecies abfrage benutzen! – d.h. alles kopiert und entleert

- In PokeAPI
  - o public PokemonSpecies requestPokemonSpecies(int id)
  - o PokemonSpecies readPokemonSpecies(JsonReader reader)
- Neue Klasse public class PokemonSpecies
- Im PokemonFragment neues void loadPokemonSpecies(); erstellt und aufgerufen in parallel laufendem Thread
  - o Passende Bindings

=> jetzt die passenden Werte abfragen

```
public class PokemonSpecies {
    public PokemonSpecies() {
    }

    public String name = "Unknown";

    public ArrayList<FlavorTextEntry> flavorTextEntries = new ArrayList<>();
}
```

```
public class FlavorTextEntry {
    public String flavorText;
    public NameWithURL language;
    public NameWithURL version;

    @Override
    public String toString() {
        return "FlavorTextEntry{" +
            "flavorText='" + flavorText + '\'' +
            ", language='" + language +
            ", version='" + version +'
```

```

        '}' ;
    }

PokemonSpecies readPokemonSpecies(JsonReader reader) throws IOException {
    PokemonSpecies pokemonSpecies = new PokemonSpecies();
    reader.beginObject();
    while (reader.hasNext()) {
        switch (reader.nextName()) {
            case "name":
                pokemonSpecies.name = reader.nextString(); //Formatierung:
Strg+Alt+O Strg+Alt+L
                break;

            case "flavor_text_entries":
                reader.beginArray();
                while (reader.hasNext()) {
                    pokemonSpe-
cies.flavorTextEntries.add(readFlavorTextEntry(reader));
                }
                reader.endArray();
                break;

            default:
                reader.skipValue();
        }
    }
    reader.endObject();
    return pokemonSpecies;
}

```

```

private FlavorTextEntry readFlavorTextEntry(JsonReader reader) throws IOException {
    FlavorTextEntry flavorTextEntry = new FlavorTextEntry();
    reader.beginObject();
    while (reader.hasNext()) {
        switch (reader.nextName()) {
            case "flavor_text":
                flavorTextEntry.flavorText = reader.nextString();
                break;
            case "language":
                flavorTextEntry.language = readNameWithURL(reader);
                break;
            case "version":
                flavorTextEntry.version = readNameWithURL(reader);
                break;
            default:
                reader.skipValue();
        }
    }
    reader.endObject();
    return flavorTextEntry;
}

```

=> Danach Binding, Text ausgeben anhand language und version

=> Per Klick auf den Text eine andere Textversion ausgeben, aber auch dazu schreiben welche Version das ist

## PokemonSpecies: FlavorTextEntry (Array List)

Array List da es im JSON eckige Klammern statt geschwungene hat: [ ... ]

In PokemonSpecies:

```
public ArrayList<FlavorTextEntry> flavorTextEntries = new ArrayList<>();
```

Neue Klasse

```
public class FlavorTextEntry {
    public String flavorText;
    public NameWithURL language;
    public NameWithURL version;

    @Override
    public String toString() {
        return "FlavorTextEntry{" +
            "flavorText='" + flavorText + '\'' +
            ", language=" + language +
            ", version=" + version +
            '}';
    }
}
```

In der API:

Wir lesen nur die englischen Texte und speichern sie in der Array List als Typ flavorTextEntry

```
case "flavor_text_entries":
    reader.beginArray();
    while (reader.hasNext()) {
        FlavorTextEntry flavorTextEntry = readFlavorTextEntry(reader);
        if (flavorTextEntry.language.name.equals("en")) { // nur
englische Texte speichern
            pokemonSpecies.flavorTextEntries.add(flavorTextEntry);
        }
    }
    reader.endArray();
    break;
```

Ausgabe im PokemonFragment

Hier gehen wir das Array durch

Mit dem OnClickListener (wenn jemand klickt) erhöht sich der Index und mit dem neuen Index wird der nächste Text in unseren String „flavorText“ geschrieben und ausgegeben

```
// PokemonSpecies.flavorTextEntries
versionIndex = 0;
String flavorText = "none";
String version = "";
if (!pokemonSpecies.flavorTextEntries.isEmpty()) { //Liste nicht empty,
dann:
    FlavorTextEntry flavorTextEntry = pokemonSpecies.flavorTextEntries.get(0); //Erster Eintrag
```

```

        flavorText = flavorTextEntry.flavorText;
        version = flavorTextEntry.version.name;
    }
binding.pokemonSpeciesFlavorText.setText("'" + flavorText.replace("\n", " "
) +
        "'" + "\n Version " + version);
binding.pokemonSpeciesFlavorText.setOnClickListener(v -> {
    if (++versionIndex == pokemonSpecies.flavorTextEntries.size()) {
        versionIndex = 0;
    }
    FlavorTextEntry flavorTextEntry = pokemonSpecies-
    .flavorTextEntries.get(versionIndex);
    String flavorText1 = flavorTextEntry.flavorText;
    binding.pokemonSpeciesFlavorText.setText("'" + flavor-
    Text1.replace("\n", " ") +
        "'" + "\n Version " + flavorTextEntry.version.name);
});

```

## Schriftart

Neuer Font Ordner

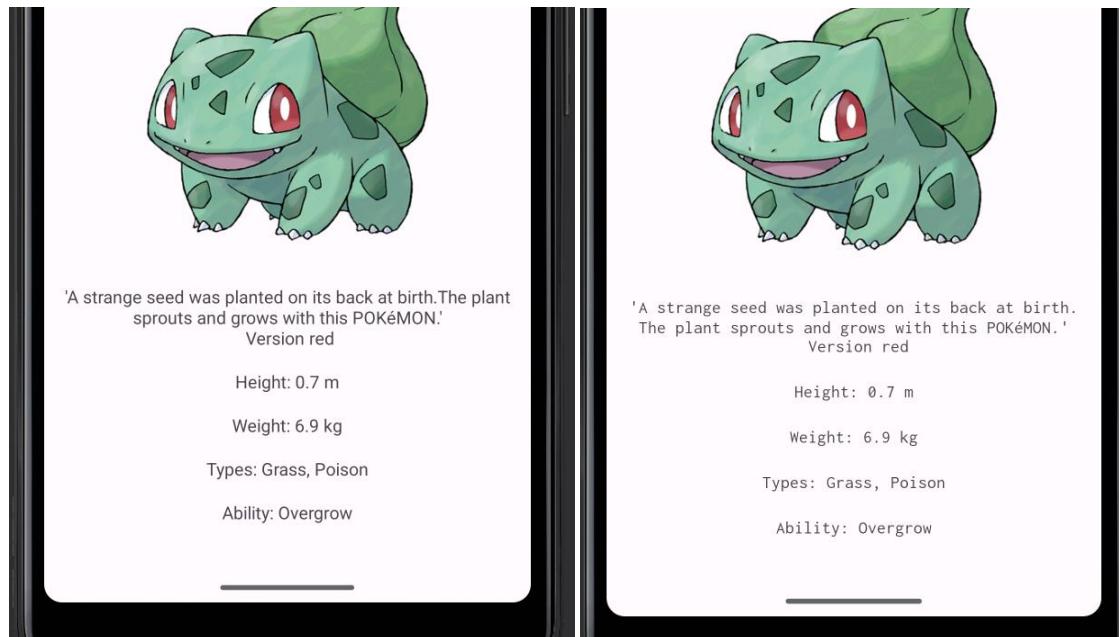
Darein die .ttf

!Achtung bei der Benennung, nur Kleingeschrieben und kein Minus

In Datei themes.xml

```
<item name="android:fontFamily">@font/inconsolataregular</item>
```

Font: **Inconsolata** (Google Font designed by Raph Levien)



## Größenänderung:

In den dimens.xml

```
<dimen name="button_size">18sp</dimen>
<dimen name="text_size_regular">12sp</dimen>
```

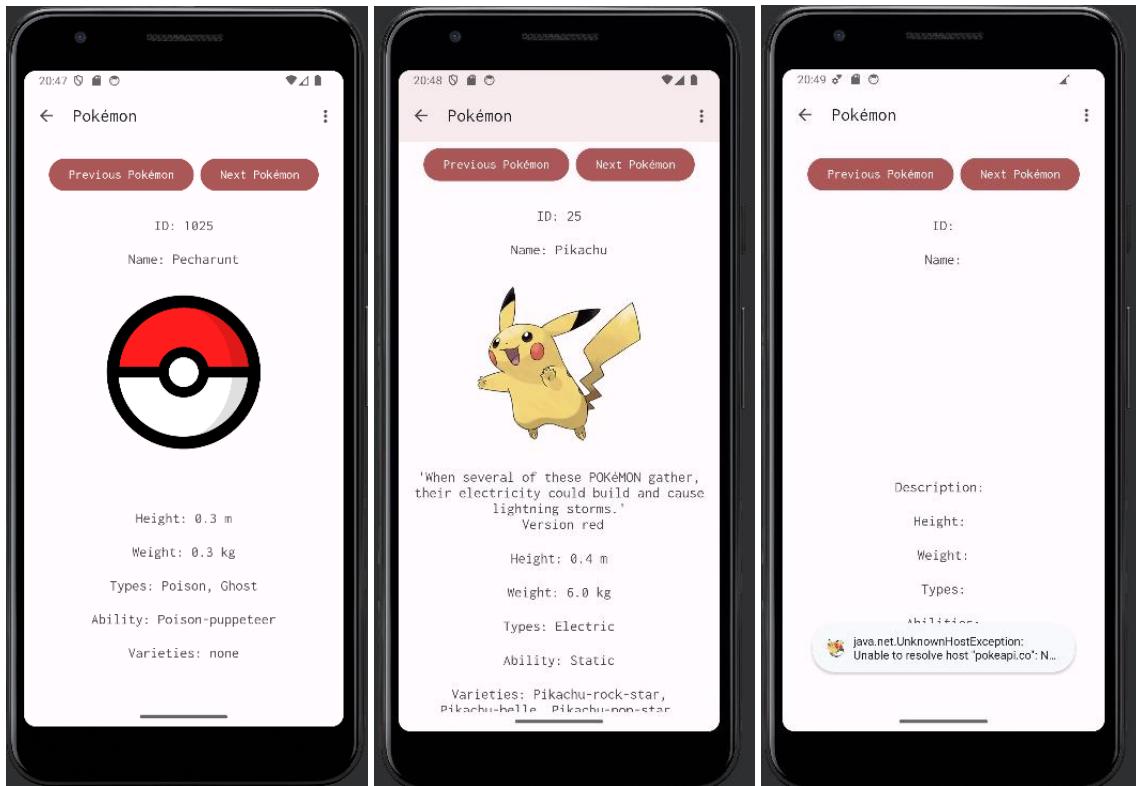
In der layout.xml dann diese Werte zuweisen – Beispiel Button:

```
android:textSize="@dimen/button_size"
```



## Stand am Ende

The image shows three screenshots of a mobile application. The first screenshot shows a list of the first six starters: Bulbasaur (ID: 1), Ivysaur (ID: 2), Venusaur (ID: 3), Charmander (ID: 4), Charmeleon (ID: 5), and Charizard (ID: 6). The second screenshot shows the detailed view for Ivysaur (ID: 2), which includes its image, a quote ("When the bulb on its back grows large, it appears to lose the ability to stand on its hind legs."), version information ("Version red"), height ("Height: 1.0 m"), weight ("Weight: 13.0 kg"), types ("Types: Grass, Poison"), ability ("Ability: Overgrow"), and varieties ("Varieties: none").



## Ausblick

Wenn man das Projekt fortsetzen würde, wären das Ideen zum fortsetzen:

- weitere Daten: wie Entwicklung der Pokemon
- Von den Variationen die vorhandenen Bilder in einer Liste
- Caching für besseres Laden
- Suche und Filtern nach Name, ID, Eigenschaften
- Pokemon Favoriten speichern
- verschiedene Sprachen unterstützen
- Design überarbeiten

## Weitere Datein

Vortrag Skript: <https://md.gafert.org/s/MUPoZI1ZK>

Übersicht des Codes (Modell) mit draw.io erstellt (Modell.drawio / .pdf)