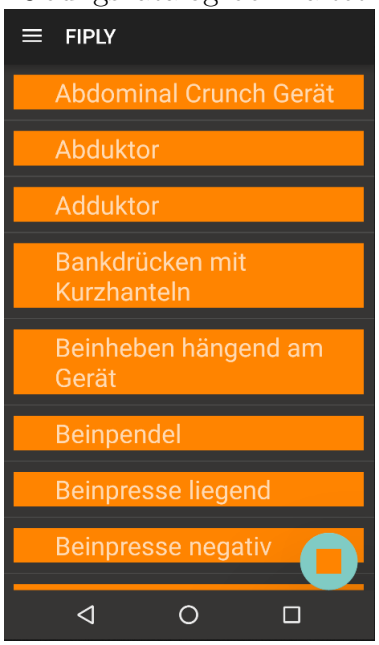
1 Übungskatalog

1.1 Beschreibung

Der Übungskatalog beinhaltet eine Liste aller verfügbaren Übungen.



1.2 Implementierung

1.2.1 Expandable List View

Die Expandable List View ist eine eigene Implementierung der standard List View. Sie erlaubt es bei tippen auf ein Element weitere Unterelemente darzustellen. Im Laufe der Entwicklung wurde festgestellt, dass eine standard List View, mit einem Verweis auf eine Detail-View vorteilhafter wäre.

1.2.2 List View

Die List View zeigt eine Liste aus Element an. Ein Klick auf ein bestimmtes Element löst eine Callback-Methode aus, von welcher die Detail View aufgerufen wird.

1.2.3 Einlesen der Übungen

Die Informationen über alle Übungen ist in der strings.xml im JSON-Format hinterlegt. Jede Übung ist somit ihr eigenes JSON-Objekt.

```
<string name="exercisecatalog">
[
...,
{
  \"Muskelgruppe\":\"Brust,Untere Brust\",
  \"Name\":\"Negativbankdruecken\",
  \"Beschreibung\":\"Mit geradem Ruecken auf Negativbank legen,
   Wenn ein Polster vorhanden ist - Beine einklemmen, Stange
   etwa schulterbreit greifen\",
  \"Durchfuehrung\":\"Mit fixierten Schultern die Stange
   kontrolliert in einer Linie zur Brust und wieder nach oben
   bewegen, Stange in einer Linie bewegen und Brust nicht
   beruehren\",
  \"Equipment\":\"Negativbank,Langhantel\",
  \"Schwierigkeit\":\"Mittel\"
}
 ,...
]
</string>
```

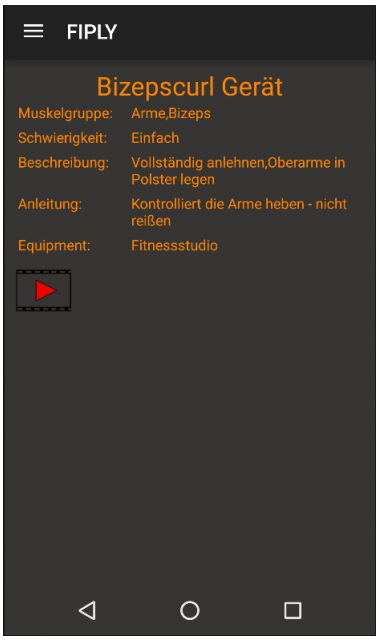
Wie in diesem Exampel erkennbar musste jedes Hochkomma mit einem Backslash escaped werden, da der JSON-String sonst nicht in der strings.xml abgelegt werden könnte.

Dieser JSON-String wird in einer Methode in der UebungenRepository eingelesen und die einzelnen Methoden werden in die Datenbank eingefügt.

```
1    public void insertAllExercises() throws JSONException {
2        reCreateUebungenTable();
3        String json = repoContext.getResources()
4                                 .getString(R.string.
   exercisecatalog);
5
6        JSONArray exercises = new JSONArray(json);
7        JSONObject temp;
8
9        for (int i = 0; i < exercises.length(); i++) {
10           temp = exercises.getJSONObject(i);
11           Log.wtf("Exercise: ", temp.getString("Name"));
12           insertUebung(temp.getString("Name"),
13                        temp.getString("Beschreibung"),
14                        temp.getString("Durchfuehrung"),
15                        temp.getString("Muskelgruppe"),
16                        temp.getString("Schwierigkeit"),
17                        "https://www.youtube.com/embed/..",
18                        temp.getString("Equipment"));
19       }
20
21   }
```

Diese Methode wird beim Startup der Applikation, während des SplashScreens, in einem Async-Task aufgerufen und ausgeführt.

1.3 DetailView



Für jede Übung gibt es eine Detailansicht, in welcher man genaures über jene Übung erfahren kann. Diese Detail-View wird aufgerufen indem man auf die korrespondierende Übung im Übungskatalog tippt.

Es werden folgende Details bereitgestellt:

- Name der Übung
- Die Muskelgruppe/n welche man mit dieser Übung trainiert.
- Schwierigkeit, wie anspruchsvoll ist diese Übung.
- Beschreibung, die Ausgangslage der Übung.
- Anleitung, wie wird die Übung, von der Ausgangsposition, richtig durchgeführt.
- Benötigtes Equipment, um die richtige Durchführung der Übung zu ermöglichen.
- Ein Video welches die Durchführung beschreibt. Dieses kann im mit Tipp auf das Videosymbol im linken mittleren Bereich der Detail View aufgerufen werden. Die App ändert dann automatisch in den Landschafts-Modus und stellt das Video im Vollbildmodus dar.

1.4 Filter

1.4.1 Beschreibung

Die Liste kann auch nach Name der Übung und Muskelgruppe gefiltert werden. Der Filter wird über den Floating Action Button aufgerufen, welcher sich in der rechten unteren Ecke des Übungskataloges befindet. Die Muskelgruppen-Auswahl erfolgt über das Tippen auf eine bestimmte Muskelgruppe, beim Namen wird rein nach Text filtriert

1.4.2 Implementierung

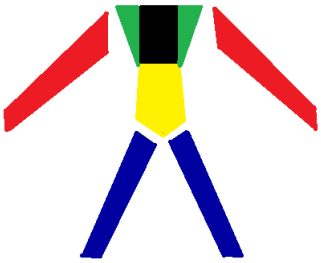
Für den Filter für Name und Muskelgruppe gibt es jeweils einen Eintrag in der Key-Value-Repository. Beim Einlesen der Uebungen werden die Filter automatisch angewandt.

Die folgende Lösung wurde diesem Tutorial entnommen [**ueClickableAreas** [**ueClickableAreas**]]

Um die geklickte Muskelgruppe zu ermitteln, wurden zwei Auswahlbereiche übereinander gelegt.

Jener Auswahlbereich, welchen der Benutzer zu sehen bekommt. Dieser Auswahlbereich ist nur die visuelle Stütze für den Benutzer, sie hat keinerlei funktionalen Nutzen.

Und eine farbkodierte Maske, welche der Benutzer nicht sieht, mit welcher er aber eigentlich interagiert. Bei einem Klick auf die Auswahlfäche kann nachvollzogen werden welche Farbe der angeklickte Bereich hat, somit wird festgestellt welche Muskelgruppe der Benutzer ausgewählt hat.



Bei einem Klick auf die Auswahlfläche wird folgende Callback-Methode aufgerufen.

```
public boolean onTouch(View v, MotionEvent event) {
    int tolerance = 25;
    int evX = (int) event.getX();
    int evY = (int) event.getY();
    int clickedColor = getHotspotColor(evX, evY);

    //RED area is arms
    if (closeMatch(getResources().getInteger(R.integer.redInt),
    clickedColor, tolerance)) {
        Log.wtf("Area clicked: ", "Arme");
        kvr.updateKeyValue("filterMuskelGruppe", "Arme");
    }
    //BLACK area is Breast
    else if (closeMatch(getResources().getInteger(R.integer.
    blackInt), clickedColor, tolerance)) {
        Log.wtf("Area clicked: ", "Brust");
        kvr.updateKeyValue("filterMuskelGruppe", "Brust");
    }
    //GREEN area is shoulders
    else if (closeMatch(getResources().getInteger(R.integer.
    greenInt), clickedColor, tolerance)) {
        Log.wtf("Area clicked: ", "Schultern");
        kvr.updateKeyValue("filterMuskelGruppe", "Schultern");
    }
    //BLUE are is legs
    else if (closeMatch(getResources().getInteger(R.integer.
    blueInt), clickedColor, tolerance)) {
        Log.wtf("Area clicked: ", "Beine");
        kvr.updateKeyValue("filterMuskelGruppe", "Beine");
    }
    //YELLOW area is core(stomach)
    else if (closeMatch(getResources().getInteger(R.integer.
    yellowInt), clickedColor, tolerance)) {
        Log.wtf("Area clicked: ", "Bauch");
        kvr.updateKeyValue("filterMuskelGruppe", "Bauch");
    }
    return true;
}
```

Diese Methode ermittelt die Farbe des gedrückten Bereiches und setzt den Key-Value Eintrag im Repository.

Die erste verwendete Hilfsmethode ermittelt ob zwei Farbwerte ähnlich oder gleich sind.

```
1     public boolean closeMatch(int color1, int color2, int
    tolerance) {
2         if (Math.abs(Color.red(color1) - Color.red(color2)) >
    tolerance)
3             return false;
4         if (Math.abs(Color.green(color1) - Color.green(color2)) >
     tolerance)
5             return false;
6         if (Math.abs(Color.blue(color1) - Color.blue(color2)) >
    tolerance)
7             return false;
8         return true;
9     }
```

Die zweite verwendete Methode ermittelt den Farbwert des gedrückten Bereichs.

```
1     public int getHotspotColor(int x, int y) {
2         bodyFilterMask.setDrawingCacheEnabled(true);
3         Bitmap hotspots = Bitmap.createBitmap(bodyFilterMask.
    getDrawingCache());
4         bodyFilterMask.setDrawingCacheEnabled(false);
5         return hotspots.getPixel(x, y);
6     }
```