THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Projektbericht

# Clothes Design App

## Modul „Android-Praktikum: Anwendungsentwicklung“ (CS2012) bei Steffen Vaupel – Sommersemester 2022


vorgelegt von

Omar, Hammad
Matrikelnummer 5311024

Technische Hochschule Mittelhessen (THM), Gießen
Fachbereich Mathematik, Naturwissenschaften
und Informatik (MNI)


10. Juli 2022

**Zusammenfassung und Hinweise**


Dieser Projektbericht wird Ihnen meine Reise zur Erstellung der Clothes Design App vorstellen. Warum ich diese Idee gewählt habe? wie ich das Projekt umgesetzt habe? welche Tests ich durchgeführt habe, um sicherzustellen, dass meine App gut funktioniert? sind alles Fragen, die in diesem Projektbericht beantwortet werden.

Kurz gesagt wird diese App eine Android App darstellen, die mit einem entfernten (Ktor) Server und einer lokalen Datenbank (Room) verbunden ist, die Kleidungsdesigns enthält, die von der Android App angezeigt, durchsucht, gefiltert und als Favoriten ausgewählt werden können.Die App wird auch in der Lage sein, neue Designs an den Server zu senden und für jeden anderen Benutzer sichtbar zu machen.Die App wird sowohl online als auch offline funktionieren.

Es war eine fruchtbare und freudige Reise, diese App zu entwickeln.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Fachlicher Hintergrund (Problem) und Zielsetzung

Das Problem war, dass die Bekleidungsdesign-Industrie immer mehr überflüssig wird. Designer brauchen mehr Inspiration und Ideen für ihre Designs.

Deshalb habe ich beschlossen, diese App zu entwickeln, damit die Leute sich von anderen Designs inspirieren lassen können.

## 1.2 Anforderung (Zusammenfassung)

Kleidungsdesign Wissen und die Designindustrie.

Ktor Server , Raum lokale Datenbank und allgemeine Kenntnisse der kotlin Sprache und Android

## 1.3 Ausgewählte Technologien/Architektur

Ich habe die MySQL-Workbench-Datenbank mit einem ktor-Server verbunden, um die Daten remote zu speichern, und eine Raumdatenbank, um die Daten lokal zu speichern, Retrofit, um die Daten vom Server abzurufen, Hilt für Dependency Injection.

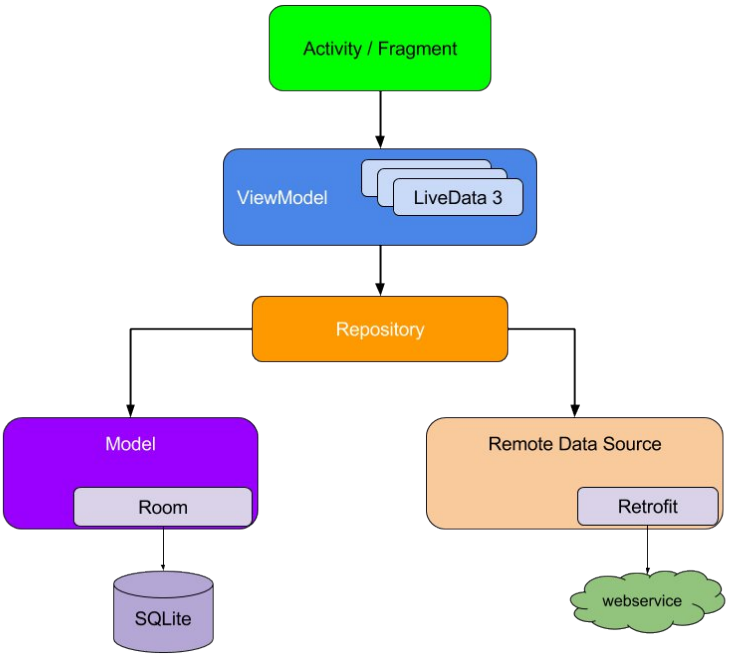
Die App sendet also Anfragen mit Retrofit an den Server, um die Daten zu erhalten und dann in der Room Datenbank zu speichern.

## 1.4 Organisation und Vorgehensmodell

Ich habe die empfohlene MVVM verwendet, bei der die GUI das ViewModel auf Änderungen hin überprüft. Das Viewmodel kommuniziert mit einem Repository, das die Daten aus der lokalen oder der entfernten Quelle bezieht.

Das Projekt soll zwischen 70 und 90 Stunden dauern.

20 für den Remote Server und den Rest für die Android App selbst



# 2 Anforderungen

* Einen Backend-Server, um Anfragen zu erhalten und Daten zu senden.

* Eine Datenbank zum Speichern der Elemente in.

* eine Android App zur Anzeige der Elemente .

## 2.1 Grundlegende Funktionsbeschreibung

Die App ermöglicht es Kleidungsdesignern :
- Neue Designs ansehen
-Ein Design auswählen und seine Details ansehen
- die Designs zu suchen
-die Kategorie der Designs zu filtern (Männer, Frauen, Kinder)
-Designs aus der Favoritenliste speichern oder löschen
- ein Design auf den Server hochladen

- Die App sollte sowohl online als auch offline funktionieren.

## 2.2 Funktionale Anforderungen

- Die App sollte in der Lage sein, Daten vom Backend-Server zu senden und zu empfangen.
- Die App sollte über eine interne Datenbank verfügen, um favorisierte Designs zu speichern.
- Die App sollte in der Lage sein, die Designs und deren Details anzuzeigen.

## 2.3 Nicht funktionale Anforderungen

- Die App sollte sowohl online als auch offline funktionieren.
- Die Daten des Favoritendesigns sollten nicht verloren gehen, auch wenn der Nutzer die App schließt.

## 2.4 (Benutzer) Schnittstellen / Ein- Ausgabeformate

Die Benutzeroberfläche sollte aus 3 Fragmenten bestehen :

- Home-Fragment -> zeigt alle Designs an
- Fav. Fragment -> zeigt favorisierte Designs an
- Add Fragment -> Hinzufügen eines Designs zu allen Designs

Eine Details-Aktivität zur Anzeige des Designs in Details

## 2.5 Fehlverhalten

## 2.6 Abnahmekriterien

Das Userinterface sollte dem Mock-up ähnlich sein.

Die App sollte auch Offline funktionieren.

Die Raumdatenbank sollte in der Lage sein, die Lieblingsdesigns zu speichern.

Der Backend-Server sollte fehlerfrei funktionieren.

# 3 Entwurf

Generell habe ich mich entschieden, XML für die GUI zu verwenden, mit einer unteren Navigationsleiste, die die Haupt-, Favoriten- und Zusatzfragmente enthält.

Für die Sprache werde ich kotlin verwenden

## 3.1 Technische Funktionen (und Funktionsabhängigkeiten)

Der Backend-Server empfängt die Daten des Benutzers und sollte mit 200 OK antworten.

Eine Singleton-Raumdatenbank sollte eine Tabelle für die Lieblingsdesigns haben, in der die Lieblingsdesigns gespeichert werden.

Wenn ein Benutzer auf ein Design geklickt hat, sollte er in einer Details-Aktivität landen, die dem Benutzer die Details eines Designs zeigt.

## 3.2 Datenmodell

Die Modellklasse des Designs sollte diese Werte für jedes Design enthalten:

**val about: String,**

**val category: String,**

**val color: String,**

**val image: String,**

**val likes : String,**

**val size: String,**

**val Titel: String**

```
import ...

@Parcelize
data class designsItem(
    @SerializedName("about")
    val about: String,
    @SerializedName("category")
    val category: String,
    @SerializedName("color")
    val color: String,
    @SerializedName("image")
    val image: String,
    @SerializedName("likes")
    val likes: Int,
    @SerializedName("size")
    val size: String,
    @SerializedName("title")
    val title: String
) : Parcelable
```

## 3.3 Strukturmodell (Aufbau)

Die Benutzeroberfläche wird drei Hauptfragmente enthalten:

Hauptentwurfsfragment, Lieblingsfragment und Hinzufügen eines neuen Entwurfsfragments.

Diese Fragmente werden das MainViewModel auf neue Daten hin beobachten.

Das MainViewModel sendet Daten von der lokalen Quelle, wenn die App im Offline-Modus ist, und holt die Daten von der Remote-Quelle, wenn die App online ist.

Die Remote-Quelle erhält die Daten, indem sie Anfragen an die Remote-API sendet.

Die lokale Quelle holt die Daten aus der Raumdatenbank

## 3.4 Funktionsmodell (Verhalten)

Wenn der Nutzer die Designs ansehen, suchen oder filtern möchte, sendet die App eine Anfrage an die API und zeigt dem Nutzer die beantworteten Designs an. Wenn dem Nutzer ein Design gefällt, führt die App eine Insert-Abfrage an die Raumdatenbank durch, um das Design in der Tabelle der Lieblingsdesigns zu speichern.

Wenn der Benutzer sein Design hochladen möchte, sendet er eine Post-Anfrage an den Server, um es zu speichern.

## 3.5 Testfälle

Ich habe das Backend mit Postman getestet und viele POST- und GET-Anfragen ausprobiert, um die App zu testen.

Für die App selbst habe ich die GUI getestet und versucht, jeden Button anzuklicken, um zu sehen, ob alles gut funktioniert.

Wegen der kurzen Zeit, die für das Projekt zur Verfügung stand, konnte ich kein Unittesting für die App durchführen.

# 4 Umsetzung

**GUI:**

Um das Senden von Daten und die Durchführung von Aktionen zwischen den Aktivitäten und dem Fragment zu erleichtern, habe ich eine Navigationsansicht verwendet.

Die App hat 2 Aktivitäten:

1. die Hauptaktivität, die die drei Hauptfragmente und die untere Navigationsleiste enthält.

2.Details Aktivität, die erscheint, wenn ein Benutzer auf ein Element klickt, das seine Details zeigt.

**lokale Datenbank:**

Die lokale Datenbank wird 2 Tabellen haben, eine für alle Designs (damit sie verwendet wird, wenn das Gerät offline ist) und eine für die Lieblingsdesigns, so dass jeder Benutzer sein eigenes Lieblingsdesign lokal speichern kann.

**entfernte API :**

Retrofit sendet GET-Anfragen, um Daten für alle Designs oder ein bestimmtes Design (durch die Suchfunktion) zu erhalten, oder POST-Anfragen, um neue Designs an den Benutzer zu senden.

**Injektion von Abhängigkeiten:**

Hilt stellt sicher, dass wir von einer Singleton-Datenbank und einer Singleton-Retrofit-Instanz versorgt werden

Ein Haupt-Ansichtsmodell, das die GUI auf Änderungen von der API oder der Raumdatenbank beobachten kann. und eine Funktion zum Senden von Daten an beide

**Offline vs Online :**

diese Funktion löst aus, ob eine Internetverbindung besteht oder nicht

sie gibt true für Internet und false für kein Internet zurück.

```
private fun hasInternetConnection(): Boolean {
    val connectivityManager = getApplication<Application>().getSystemService(
        Context.CONNECTIVITY_SERVICE
    ) as ConnectivityManager
    val activeNetwork = connectivityManager.activeNetwork ?: return false
    val capabilities = connectivityManager.getNetworkCapabilities(activeNetwork) ?: return false
    return when {
        capabilities.hasTransport(NetworkCapabilities.TRANSPORT_WIFI) -> true
        capabilities.hasTransport(NetworkCapabilities.TRANSPORT_CELLULAR) -> true
        capabilities.hasTransport(NetworkCapabilities.TRANSPORT_ETHERNET) -> true
        else -> false
    }
}
```

# 5 Qualitätssicherung (Testprotokoll)

Um die App zu testen, habe ich beschlossen, einige Leute die App ausprobieren zu lassen und mir ihr Feedback zu geben.

Im Allgemeinen war das Feedback so gut, außer für einige Bugs, die ich später behoben:

z.B. die Farbe des Favoriten-Buttons und das Löschen aller Elemente aus dem Favoriten-Fragment hatte einen Fehler, den ich aber später behoben habe.

Ein Bug ist nicht behoben , die ist die Like Button Color in die Details-Activity .

ich habe eine 4.5/5 bewertung bekommen

# 6 Evaluation

- Die GUI ist so nah am Mock-up wie möglich.
- So kann der Benutzer Designs ansehen, suchen, filtern, als Favorit speichern und hinzufügen.
- Die App funktioniert sowohl online als auch offline
- Das Backend empfängt Anfragen und beantwortet sie ohne Fehler.

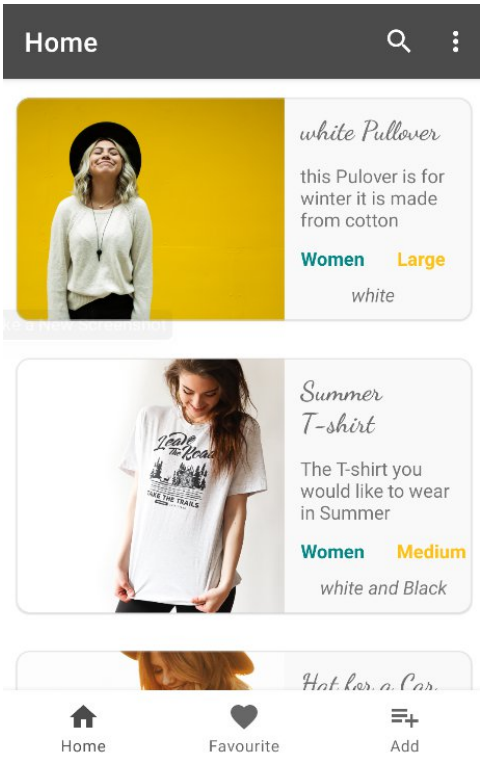
Im Großen und Ganzen würde ich sagen, dass ich mit dem Ergebnis, das ich erhalten habe, zufrieden bin.

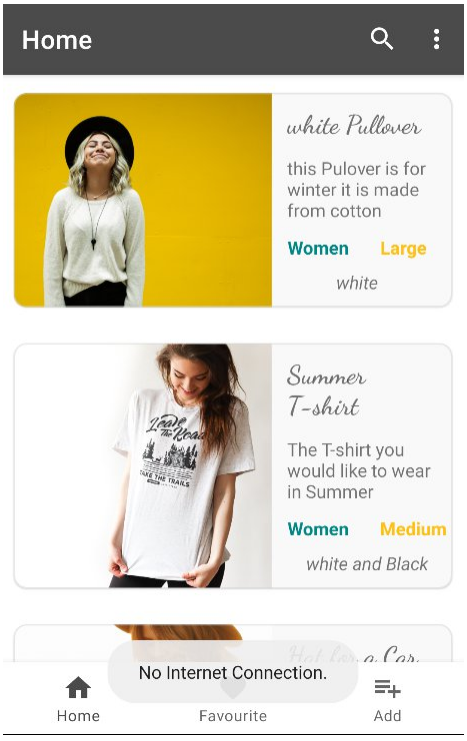
# 7 Anwenderdokumentation

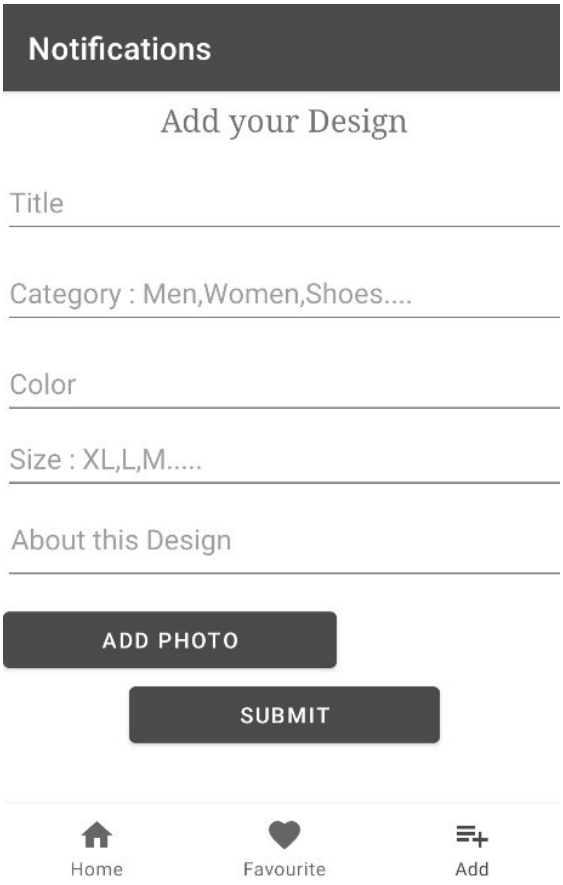
Wenn Sie die App zum ersten Mal öffnen, landen Sie in der Hauptansicht. Hier können Sie durch die verschiedenen Designs scrollen oder das Design im Menü im oberen Bereich suchen oder filtern.

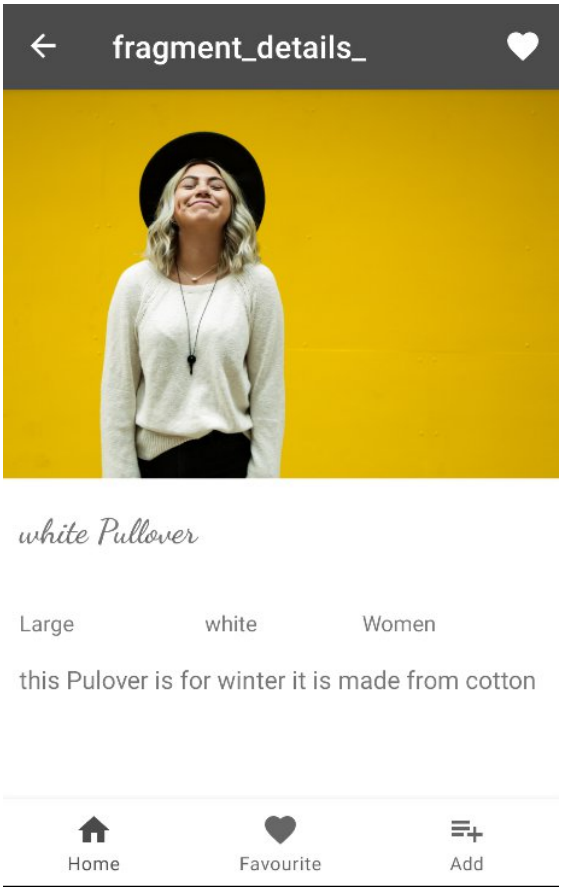
Wenn Sie mehr über ein Design wissen möchten, klicken Sie darauf und Sie erhalten eine neue Ansicht mit den Details des Designs. Wenn Ihnen das Design gefällt, finden Sie ein Herzbild im oberen Menü, klicken Sie darauf und es wird in Ihren Favoriten gespeichert. Klicken Sie nun auf den Zurück-Button und Sie gelangen wieder zur Hauptansicht. Sie finden unten eine Navigationsleiste, mit der Sie zum zweiten Abschnitt gehen können. Dort finden Sie alle Designs, die Ihnen gefallen haben, wenn Ihnen eines nicht mehr gefällt, klicken Sie es einfach lange an und es wird gelöscht oder klicken Sie es einfach an, wenn Sie wieder zu den Details gehen wollen.

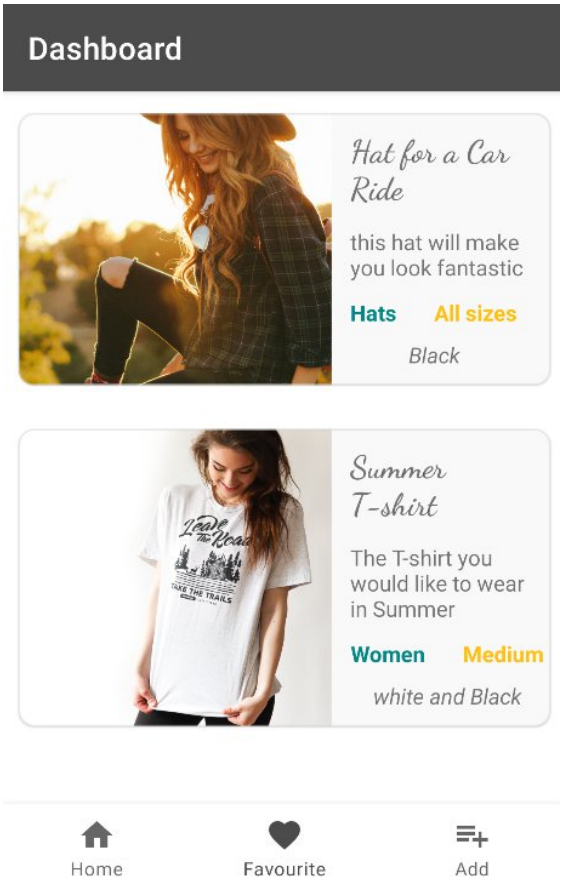
Gehen Sie nun zum letzten Abschnitt in der unteren Navigationsleiste. Hier können Sie Ihr eigenes Design hinzufügen, geben Sie einfach die Daten ein, laden Sie das Foto hoch und klicken Sie auf "Add", dann wird es dem Server hinzugefügt und andere können Ihr Design sehen und mögen.

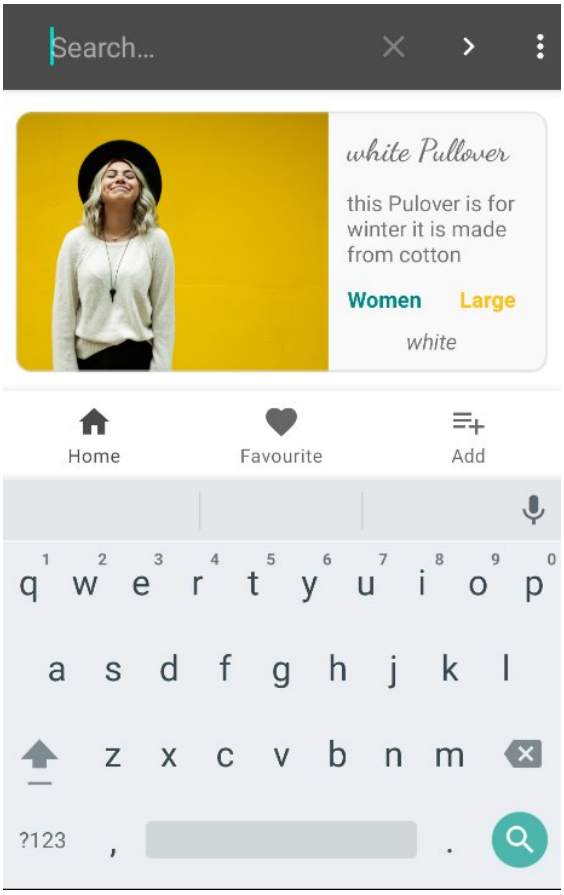



Notifications
Add your Design
Title
Category : Men,Women,Shoes....
Color
Size : XL,L,M.....
About this Design
ADD PHOTO
SUBMIT
Home
Favourite
Add


fragment_details_
white Pullover
Large
white
Women
this Pulover is for winter it is made from cotton
Home
Favourite
Add


Dashboard
Hat for a Car Ride
this hat will make you look fantastic
Hats
All sizes
Black
Summer T-shirt
The T-shirt you would like to wear in Summer
Women
Medium
white and Black
Home
Favourite
Add


Search...
white Pullover
this Pulover is for winter it is made from cotton
Women
Large
white
Home
Favourite
Add

## Resourcen :

https://developer.android.com/

https://ktor.io/

https://stackoverflow.com

**Backend Gitlab :**

https://git.thm.de/ohmm61/backend

**MYSQL Database Schema :**

```
create database designme ;

use designme;

create table designs(
id int not null auto_increment,
title varchar(1500) ,
category varchar(1500),
image varchar(1500),
about varchar(5999),
likes int,
size varchar(1500),
color varchar(1500),
primary key(id)
);
```