Dokumentation ProductAR

Maximilian Rehberger

July 27, 2019

1 Inhaltsverzeichnis

2 Einleitung

2.1 Zweck

Produkte können zum Beispiel beim Einkaufen mit dem Smartphone gescannt werden und erkannt werden. Informationen werden angezeigt wie zum Beispiel Bilder oder ein Preisvergleich. Mithilfe der App soll man einen Barcode einscannen können und Informationen zu den Produkten erhalten. Weiterhin kann der Nutzer ein Produkt in Augmented Reality (AR) testen und sieht somit wie es in Wirklichkeit aussehen wird, wenn er es kaufen würden.

3 Allgemeine Übersicht

3.1 Beschreibung Ausgangssituation

Es gibt bereits viele Shopping-Apps wie zum Beispiel Ikea, H&M oder S'Oliver. Das Problem ist, dass jeder am Ende für jedes Geschäft eine eigene App auf dem Smartphone hat. Diese App soll die Möglichkeiten geben mehrere unterschiedliche Produkte in einer App zu speichern und zu verwalten. Also eine App für alle Produkte.

3.2 Produkteinsatz

Die App kann zum Beispiel als Einkaufsliste oder Wunschliste für Produkte eingesetzt werden. Darüber hinaus bieten sich noch viele weitere Möglichkeiten.

3.3 Produktumfeld

Die App wird hauptsächlich im privaten Umfeld umgesetzt, beim Einkaufen in Geschäften oder Online-Einkauf.

3.4 Produktfunktionalität

Scannen von Produkten, Informationen zu Produkten, Preisvergleich, Bilder hochladen für Produkte, Produkte in AR testen.

3.5 Personas

3.5.1 Nutzer

3.5.2 Verkäufer

3.5.3 Admin

4 Architekturkonzept und Entwurf

4.1 Ursprüngliches Architekturkonzept

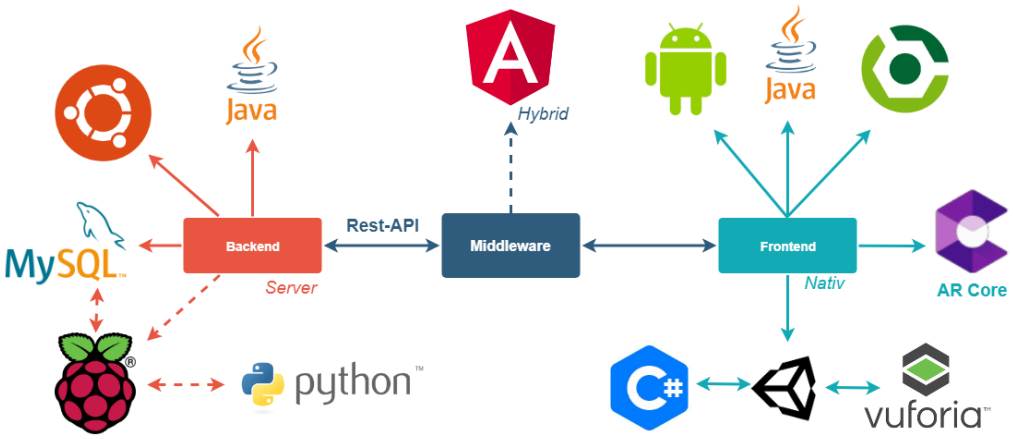

Figure 1: Ursprüngliches Architekturkonzept

4.2 Aktualisiertes Architekturkonzept

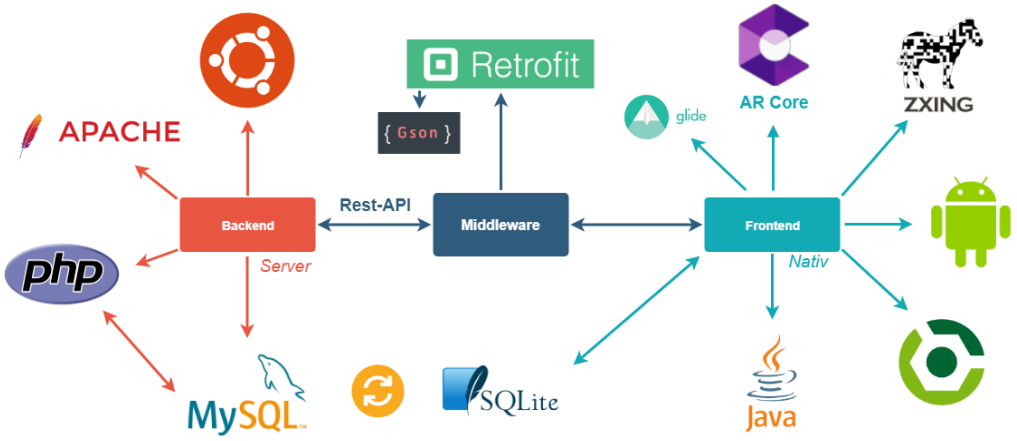

Figure 2: Aktualisiertes Architekturkonzept

4.3 Anfängliche Skizze Datenbankentwurf

4.3.1 MySQL Datenkbank (Remote)

Ursprünglich war geplant, dass die Daten ausschließlich auf dem Server in einer MySQL Datenbank gespeichert werden.

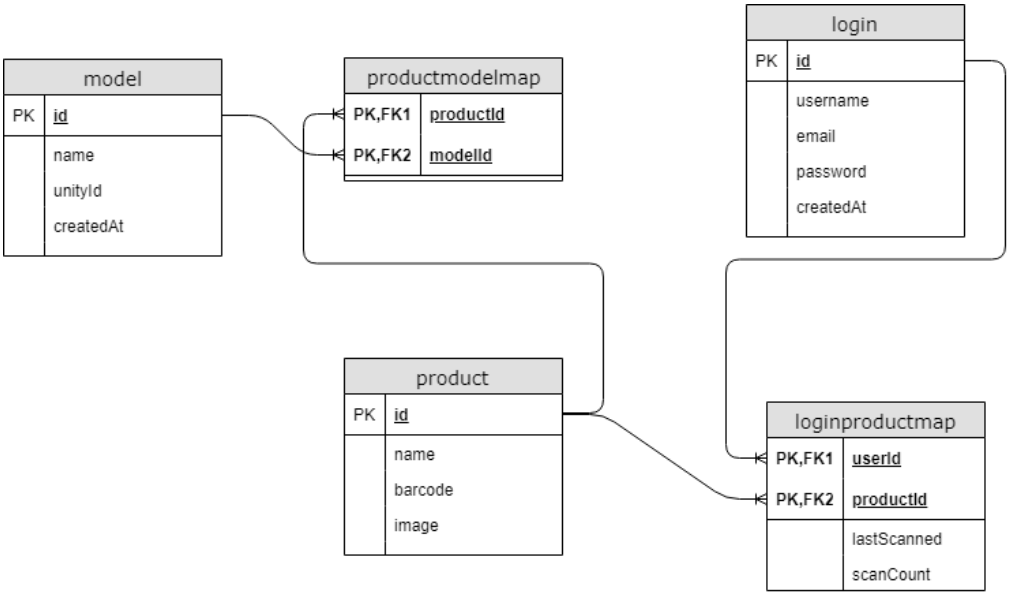

Figure 3: Anfängliche Skizze Datenbankentwurf

4.4 Anfängliche Skizze Java Klassen

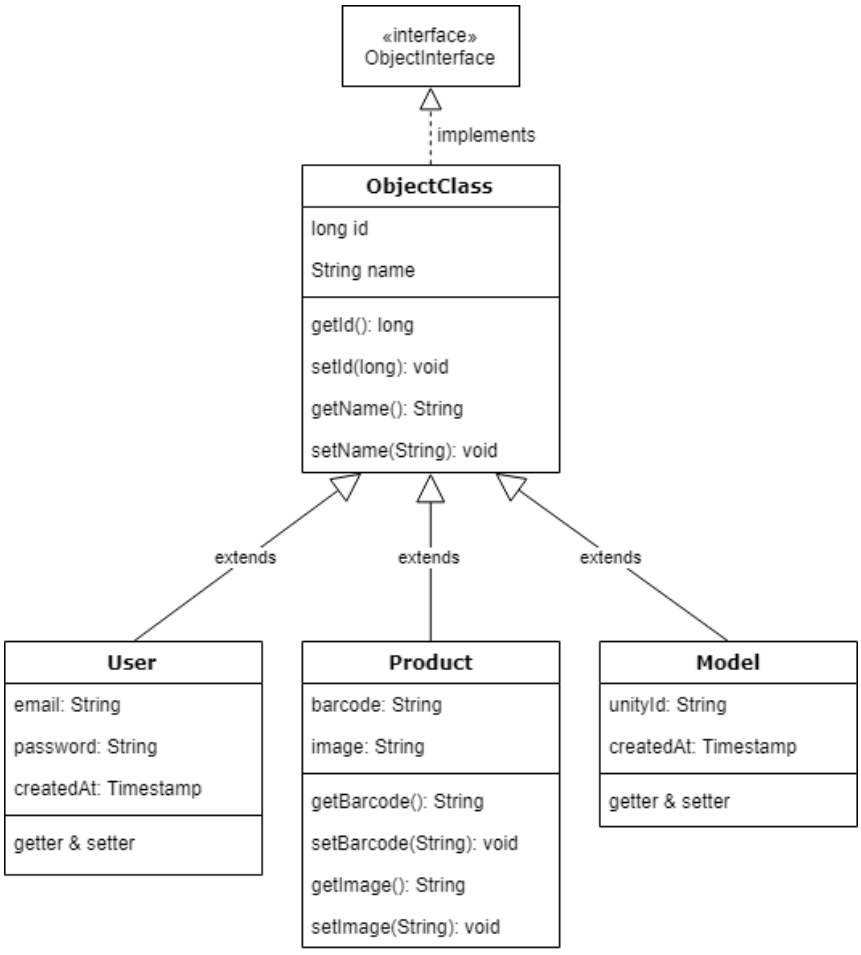

Figure 4: Anfängliche Skizze Datenbankentwurf

4.5 Endgültige Skizze Datenbankentwurf

4.5.1 SQLite Datenbank (Lokal)

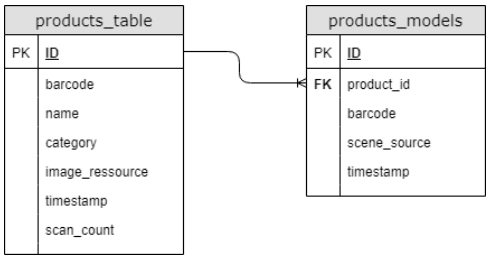

Figure 5: Skizze Datenbankentwurf: SQLite

4.5.2 MySQL Datenbank (Remote)

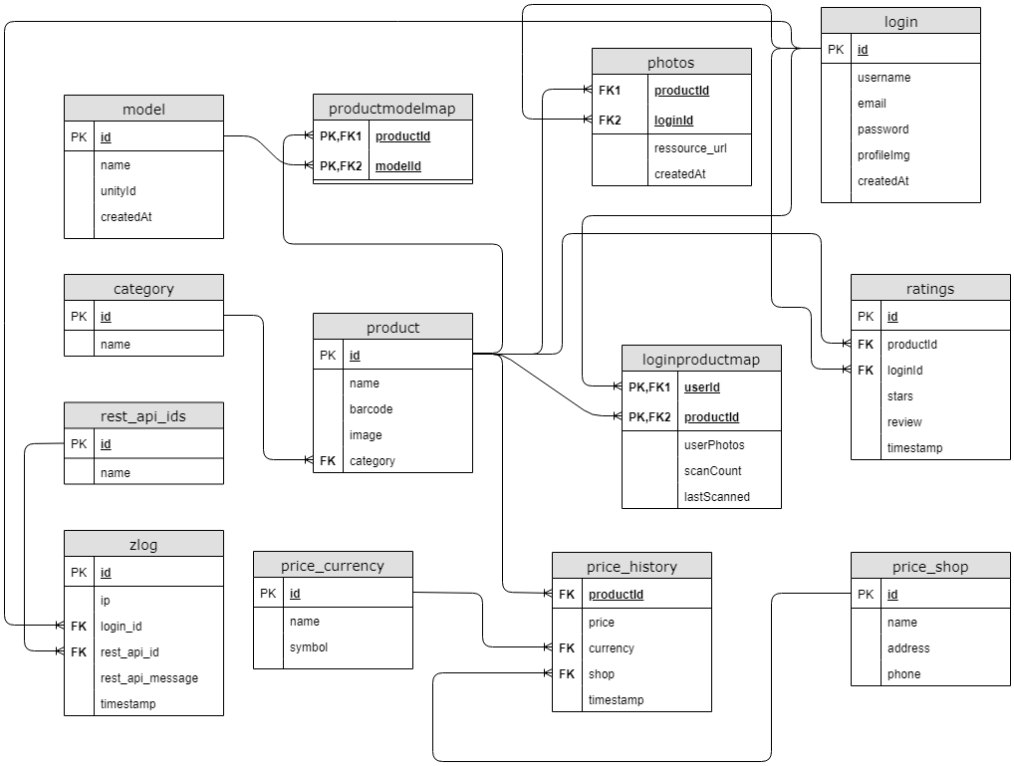

Figure 6: Aktualisierte Skizze Datenbankentwurf: MySQL

4.6 Endgültige Skizze Java Klassen

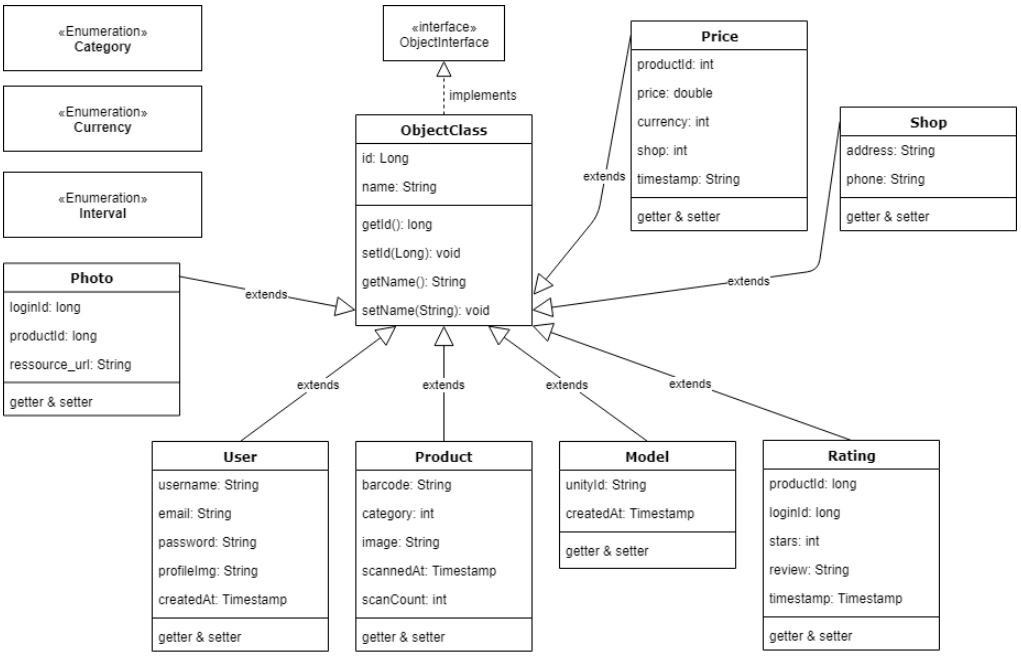

Figure 7: Aktualisierte Skizze: Java Klassen

4.7 Übersicht Backend Server

Der Backend Server ist ein gemieteter Server von Hosteurope.
Produktbezeichnung: "Virtual Server Linux Advanced 8.2".
Dieser hat folgende Linux Version installiert: Ubuntu 16.04.6 LTS.

Die Technischen Spezifikationen lauten wie folgt:

4 virtuelle Kerne
6 GB RAM
200GB SSD

Es handelt sich hierbei um einen virtuellen Server, das bedeutet, dass sich der Server mit anderen "Containern" die Hardware eines Servers teilen.

Der Server hat eine eigene Domain: www.nimoo.de.

4.8 Übersicht REST API

Die Rest Schnittstelle wurde mit PHP auf dem Webserver umgesetzt welcher vom Backend Server bereits zur Verfügung gestellt wurde. Für jede Ressource existiert ein Pfad, mit entsprechender PHP Datei.

Der Hauptpfad für die App auf dem Webserver: "https://www.nimoo.de/apps/productar"

Folgende Pfade existieren auf dem Webserver:

../products/
../products/images/
../products/photos/
../products/prices/
../products/ratings/
../users/
../users/images
../models/

4.9 Technische Entscheidungen

4.9.1 Warum Android?

Die Entscheidung, die App für Android zu entwickeln wurde getroffen, da Android zumindest in Deutschland einen höheren Marktanteil besitzt als iOS. Vor allem die Studenten der Fakultät Informatik und Wirtschaftsinformatik (FIW) und in der Vertiefung Mobile Solutions benutzen mehrheitlich Android Smartphones. Ein weiterer Grund ist, dass Android Java basiert ist und dafür sehr gut geeignet ist, wenn bereits fortgeschrittene Erfahrungen mit der Programmiersprache Java gegeben sind. Weiterhin gibt es beim Entwickeln keine Mehrkosten, da es bereits viele Open-Source Erweiterungen (Bibliotheken) gibt und Anleitungen, die das Entwickeln weiter vereinfachen.

4.9.2 Welche Androidversion?

Als minimal unterstützte Android Version (minSdkVersion) für die App musste die Api 24 (Android 7) verwendet werden. Dies liegt daran, dass die AR Funktionialität mit der Google AR Core Erweiterung erst ab Android Version 7 (Api 24) unterstützt wurde und alle vorherigen Versionen keine Unterstützung haben. Dies hat den Nachteil, dass nur ca. 37,1 % aller Android Geräte unterstützt werden im Vergleich zu den 95,3 % die mit Android 4.4 (Api 19) unterstützt würden.

4.9.3 Welche Entwicklungsumgebung?

Zum Entwickeln der App wurde hauptsächlich die Entwicklungsumgebung von Android Studio und IntelliJ genutzt.

4.9.4 Wieso Google AR Core?

Googles neuestes Framework für Augmented Reality Anwendungen heist "AR Core". Im Vergleich zu einer AR Anwendung mit Unity lässt es es sich sehr einfach in die App integrieren (Als Fragment oder View in der Aktivity). Weiterhin lassen sich Modelle (.OBJ) sehr einfach mit dem Sceneform Plugin einbinden und bearbeiten.

4.9.5 Wieso eine MySQL Datenbank?

Zum einen war die MySQL Datenbank ebenfalls schon auf dem Backend Server aufgesetzt, somit war keine weitere Konfiguration notwendig. Weiterhin ist es sehr einfach eine Datenbank mit SQL zu erstellen und Abfragen durchzuführen.

4.9.6 Wieso eine REST API?

Die Rest API ist die Schnittstelle zwischen der App und der Datenbank auf dem Server. Diese wird benötigt, da man aus Sicherheitsgründen keine direkte Verbindung zwischen App und Datenbank zulassen darf.

4.9.7 Vergleich mit Alternativlösungen

4.9.7.1 Firebase von Google .

Die Backendlösung von Google ist "FireBase" und wäre erheblich einfacher umzusetzten und hätte ebenfalls den Vorteil, dass kein externer Server benötigt wird. Warum wurde diese Lösung in diesem Projekt jedoch nicht verwendet? Die Datenbank enthält sensible Daten, wie zum Beispiel Nutzerdaten. Diese sollen nicht an Google gesendet werden.

4.9.7.2 Alternative Datenbankmodelle .

PostgreSQL und MongoDB.

5 Technische Dokumentation

5.1 Android Manifest

5.2 Java Interfaces

5.2.1 ObjectInterface

5.2.2 ScanResultReceiver

5.2.3 IRetrofitCRUD

5.2.4 JsonPlaceHolderApi

5.3 Java Klassen

5.3.1 Objekt Klassen

5.3.1.1 Object Class (Abstract)

5.3.1.2 Product

5.3.1.3 User

5.3.1.4 Model

5.3.1.5 Photo

5.3.1.6 Price

5.3.1.7 Shop

5.3.1.8 Category (Enum)

5.3.1.9 Currency (Enum)

5.3.1.10 Interval (Enum)

5.3.2 Aktivity Klassen

5.3.2.1 MainActivity

5.3.2.2 SplashScreen

5.3.2.3 ProductArActivity

5.3.2.4 ProductScanActivity

5.3.2.5 CaptureActivityPortrait

5.3.2.6 LastScannedProductsActivity

5.3.2.7 CreateProductActivity

5.3.2.8 ProductDetailActivity

5.3.2.9 ProductPhotoGalleryActivity

5.3.2.10 ProductPhotoDetailActivity

5.3.2.11 CreatePriceActivity

5.3.2.12 PriceHistoryActivity

5.3.2.13 RegisterActivity

5.3.2.14 LoginActivity

5.3.2.15 ProfileActivity

5.3.2.16 SettingsActivity

5.3.2.17 InfoActivity

5.3.3 Adapter Klassen

5.3.3.1 ProductListAdapter

5.3.3.2 PhotoAdapter

5.3.4 Hilfs Klassen

5.3.4.1 GeneralHelper

5.3.4.2 BarcodeHelper

5.3.4.3 QRCodeHelper

5.3.4.4 LoginHelper

5.3.4.5 SettingsHelper

5.3.4.6 ImageHelper

5.3.4.7 PhotoHelper

5.3.4.8 UploadHelper

5.3.4.9 PriceHelper

5.3.5 Fragment Klassen

5.3.5.1 ScanFragment

5.3.5.2 CustomArFragment

5.3.6 Retrofit Schnittstelle

5.3.7 Network Monitor

5.3.8 Background Service

5.3.9 Notifications

5.4 Ressourcen

5.4.1 Layout

5.4.2 Drawable Icons

5.4.3 App Icon

5.4.4 Animation

5.4.5 Menu

5.4.6 Assets

5.4.7 Values

5.5 Rest Api

6 Veröffentlichung im Google Play Store

6.1 Store Eintrag

6.2 Alpha Test

6.3 Beta Test

7 Zukünftige Entwicklungen

8 Fazit

9 Verwendete Technologie, Frameworks und Software

10 Verlinkung Repositories

11 Verlinkung Tutorials

12 Quellenangabe