Hochschule Luzern
Departement für Informatik

Projekt FoodPrint

Programmieren fürs IOS

Studierende: Frederico Fischer, Nico Iseli
Dozenten: Prof. Dr. Ruedi Arnold, Nicolas Märki
Abgabedatum: 13. Dezember 2020

1 Einleitung

Das vorliegende Projekt hat zum Ziel, eine App zu entwickeln, die Leute mehr zu umweltfreundlichen Einkäufen ermutigt. Die App soll dabei bedürfnisbasiert regionale und saisonale Produkte vorschlagen. Die Implementierung ist vorerst für User aus der Schweiz vorgesehen. Das Datenbankschema im Backend ist jedoch so aufgebaut, dass sich die App auch auf weitere Länder skalieren lässt.

2 Architektur

Die Architektur ist so aufgebaut, dass Produktdaten im JSON-Format über einen Webserver, der durch das EnterpriseLab gehostet wird, zur Verfügung gestellt werden. Die Produktdaten werden über HTTP-Requests geladen und mittels Core-Data persistiert. Beim Laden wird jeweils geprüft, ob es auf dem Server Änderung der Produktdaten gab. Falls nein, werden die Produktdaten im App auch nicht angepasst. Neben den Produktdaten werden auch die User-Einstellungen mittels Core-Data festgehalten. Die User-Daten befinden sich jedoch nur App-Intern. Die User-Daten werden genutzt, um die Produkte user-spezifisch zu filtern. Für das User-Interface setzt das Projektteam auf UI-Kit.Die Architektur ist als Abbildung (vgl. Abbildung 1 auf Seite 2) festgehalten.

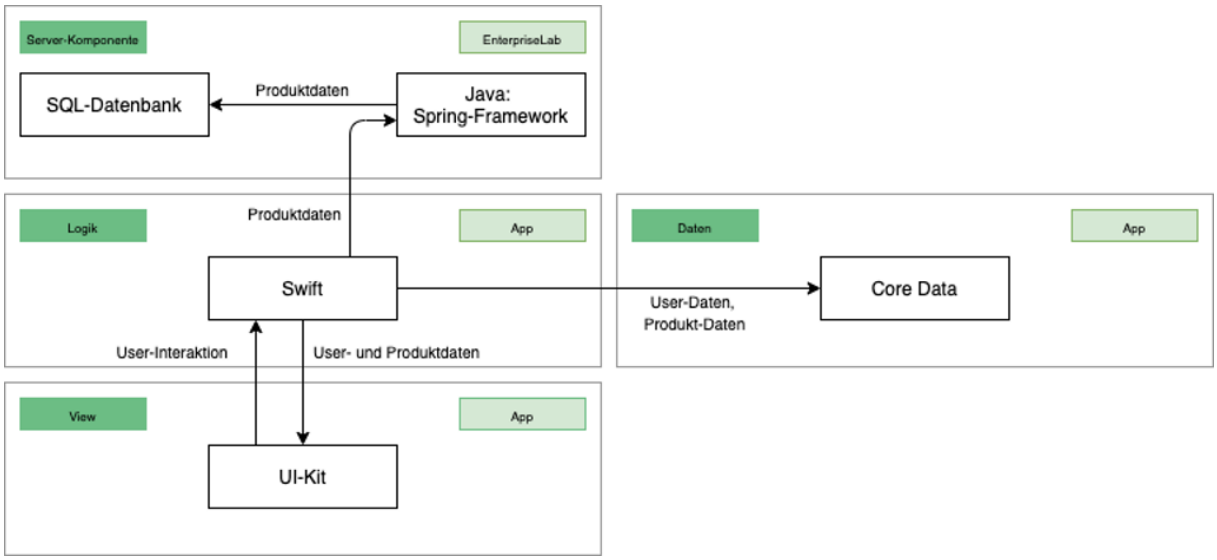

Abbildung 1: Architektur,
Quelle: Projektteam

3 Feature-List

- **User-Settings:** Bedürfnisorientierte Einstellungen, wie die Produkte gefiltert werden sollen
- **Userspezifische Produktauflistung:** Auflistung der gefilterten Produktdaten gemäss den User-Einstellungen

- **Produkt-Details:** Detail-View für jedes Produkt, in der ein vertiefterer Einblick möglich ist
- **About-View:** View mit Informationen über die Intention und Funktionsweise der App

4 Umsetzung technischer Anforderungen

4.1 Server-Komponente

Für die Server-Komponente wurde eine Ressource im EnterpriseLab reserviert. Dort werden die Produktdaten in einer SQL-Datenbank festgehalten und mit Java (beziehungsweise dessen Framework Spring) verarbeitet. Die Produktdaten werden auf dem Server (also in der Java-Applikation) in ein JSON-Format konvertiert und über den Webserver zur Verfügung gestellt (das JSON ist ersichtlich unter: `http://ffischer-ios-h20.el.eee.intern:8080/api/v1/products/Schweiz`). Aktuell läuft der Server nicht in der DMZ. Daher ist ein Aufruf nur mit bestehender VPN-Verbindung möglich.

Code-Referenz: `foodprint-backend-distribution`

4.2 HTTP-Kommunikation

Die JSON-Daten des Webservers werden über den Aufruf der URL (`http://ffischer-ios-h20.el.eee.intern:8080/api/v1/products/Schweiz`) geladen und mittels des JSON-Decoders von Swift in ein Array von Swift-Objekten konvertiert.

Code-Referenz:
`FoodPrint/FoodPrint/ViewControllers/ProductListViewController.swift`
In diesem File ist der entsprechende Code unter dem Abschnitt **„MARK: - API“** ersichtlich.

4.3 Persistierung mit Core-Data

Mittels Core-Data werden die User-Daten sowie die Produktdaten lokal persistiert. Die App verwaltet dabei ingesamt einen User und mehrere Produktdaten. Diese können gelesen (über die Fetch-Methode im NSManagedContext) und persistiert (über die Save-Methode im NSManagedContext) werden. Der persistierte User dient dazu, die Bedürfnisse zu ändern und demzufolge die entsprechenden Produktdaten zu filtern.

Code-Referenz:
`FoodPrint/FoodPrint/Utils/CoreData.swift`

4.4 Unit-Tests

Unit-Tests kamen besonders bei der Logik zur Anwendung. Damit die Daten richtig gefiltert werden, war es wichtig, verschiedene Fälle zu berücksichtigen. Daher wurde versucht, möglichst alle Ausnahmefälle zu evaluieren und somit verschiedene Tests für die unterschiedlichen Funktionen durchzuführen.

Code-Referenz:
`FoodPrint/FoodPrintTests/FoodPrintTests.swift`

4.5 Auto-Layout

Für das Projekt wurden die verschiedenen Inhalte der einzelnen View-Controller mittels Auto-Layout formatiert. Dafür wurden entsprechende Constraints gesetzt. Die Nutzung dieses Features wurde jedoch zu wenig gut geplant. Dieses wurde ohne grosses Vorwissen genutzt, was dazu führte, dass die Constraints mehrmals gelöscht und wieder neu erstellt werden mussten. Zudem wurden auch zu wenig Geräte getestet. Aktuell sieht die Formatierung auf neueren beziehungsweise grösseren Geräten gut aus. Hingegen auf kleineren Geräten (beispielsweise auf dem I Phone SE) überlappen sich einige Inhalte. Um dies zu beheben, wurde versucht, im Nachhinein noch eine Scroll-View zu integrieren. Diese kam dann jedoch mit den bestehenden Constraints in Konflikt. Daher ist die App aktuell nicht auf alle Geräte ausgelegt.

Code-Referenz:
`FoodPrint/FoodPrint/Utils/Main.storyboard`

5 Erkenntnisse

Das Erstellen einer eigenen App war eine sehr interessante Erfahrung. Das Wissen über das IOS sowie Swift hat sich im Team durch das Projekt stark verbessert. Insgesamt wurden einige Fehler gemacht, ohne die man im Nachhinein sicherlich effizienter gewesen wäre. Ein Beispiel dafür ist die Nutzung von Auto-Layout. Ein Problem dabei war es, dass zu schnell damit begonnen wurde. Es mussten immer wieder Constraints gelöscht und neu erstellt werden. Ein anderes Beispiel ist das Datenmanagement. Hier wurden die bisherigen Produktdaten von Hand abgetragen, weil keine passende Referenz gefunden wurde. Kämen neue Produkte dazu, müsste die Datenerhebung überdacht werden. Eine weitere Hürde war das Aufsetzen der Logik. Diese wurde unterschätzt. Beim Filtern der Daten gab es diverse Ausnahmefälle, die betrachtet werden mussten. Abgesehen davon ist man aber mit dem Resultat sehr zufrieden und motviert, die App auch im Anschluss an die Projektabgabe weiterzuführen.