Fuel Radar - ein Xamarin Forms Projekt

Lukas Körfer

Aachen, am 30. Juni 2016

Unter dem Namen *Code Competition* ruft die Plattform IT-Talents monatlich zu einem Programmierwettbewerb auf. Die Aufgabe von Juni 2016 sieht die Implementierung eines Programmes oder einer App zur Ermittlung von Tankstellen und Kraftstoffpreisen vor. Die entsprechenden Daten sollen von der Tankerkönig API abgerufen und dann präsentiert werden. Neben der Bearbeitung dieser Aufgabe von IT-Talents soll Fuel Radar für mich auch als erster Kontakt zu Xamarin dienen. Xamarin vereint zum einen die Möglichkeit der Cross-Plattform-Entwicklung mit den Vorteilen einer nativen App auf den Plattformen Android, iOS und Windows Phone, zum anderen baut es auf der freien Mono-Implementierung des mächtigen .NET-Frameworks sowie der weit verbreiteten Programmiersprache C# auf. Im Rahmen dieses Projekte wurde Xamarin sowie das Cross-Plattform-UI-Kit Xamarin Forms verwendet, um eine entsprechende App zu entwerfen. Zwar ist das erstellte Projekt durch den Cross-Plattform-Ansatz theoretisch auch für iOS und Windows Phone konzipiert, jedoch führten fehlende Testmöglichkeiten und notwendige plattformspezifische Eingriffe zu einer Fokussierung auf Android.

Inhaltsverzeichnis

1 Programmaufbau

Die Verwendung von Xamarin mit portablen Klassenbibliotheken (engl. portable class library, PCL) erzeugt für jede Anwendung mehrere Projekte, zum einen die erwähnte Klassenbibliothek um für alle Plattformen gemeinsamen Code auszulagern und zum anderen für jede gewünschte Plattform ein spezielles Projekt, die den jeweiligen Code zum Starten der App auf dieser Plattform beinhaltet (Android, iOS, Windows, Windows Phone). Zudem können in Fällen, in denen der Cross-Plattform-Ansatz nicht funktioniert, plattformspezifische Implementierungen einer gemeinsamen Schnittstelle in den jeweiligen Projekten abgelegt und dann im geteilten Programm über den *DependencyService* aufgerufen werden.

Die geteilte Klassenbibliothek ist zusätzlich unterteilt in folgende Kategorien:

- Model (das zugrundeliegende Datenmodell)
- UI (sämtliche Elemente der Oberfläche)
- ViewModel (die Schnittstelle zur Oberfläche für MVVM)
- Data (die Schnittstellen zur internen Datenbank)
- Requests (für den Zugriff auf die Tankerkönig API)
- Settings (für die Verwaltung der Einstellungen)
- Geo (Zugriff auf APIs für die Bestimmung von Koordinaten)

Fuel Radar verwendet intern zum Teil das MVVM-Schema, um zwischen den Daten und der Oberfläche zu trennen. Durch MVVM können die Daten eines Datenmodells im sogenannten *ViewModel* verwaltet und dann an Elemente der Ansicht gebunden werden. Sämtliche Änderungen im *ViewModel* werden in der Oberfläche automatisch aktualisiert und vice versa. Leider befindet sich Xamarin Forms derzeit noch in der Entwicklung, wodurch manche Bedienelemente dieses Binden von Eigenschaften (noch) nicht unterstützen, so beispielsweise das Auswahl-Bedienelement *Picker* oder die Cross-Plattform-Karte aus *Xamarin.Forms.Maps*. Bei der Verwendung dieser Elemente wurde MVVM somit verletzt und direkt auf die Elemente zugegriffen.

2 Benutzeroberfläche

Xamarin Forms bietet verschiedene *Pages*, um die Navigation innerhalb einer App zu gestalten. All diese *Pages* ermöglichen es, eine graphische Benutzeroberfläche einmalig zu gestalten und dennoch die Besonderheiten der einzelnen Plattformen zu respektieren, da bei der Umsetzung auf die jeweiligen nativen Bedienelemente zugegriffen wird. So erscheint bei der Verwendung einer *TabbedPage* die entsprechende Leiste mit Tabs mal am unteren (iOS) und mal am oberen Bildschirmrand (Android).

Um mehrere Ansichten möglichst schnell zugänglich zu machen, wird eine *Master-Detail-Page* verwendet, diese erzeugt eine ausziehbare Navigationsleiste (engl. navigation drawer). In dieser Navigationsleiste befinden sich neben den weit verbreiteten Elementen wie *Start*, der Einstiegsansicht der App, *Einstellungen* und *Info* zudem die Elemente *Suche*, *Favoriten* und *Statistiken*, welche die Kernfunktionalität abdecken. Der Nutzer kann hier nach Tankstellen suchen, Informationen zu zuvor favorisierten Tankstellen ermitteln und Graphen zu bisherigen Preisanfragen betrachten.

Die Suche nach Tankstellen kann der Nutzer in einer ersten Ansicht starten und wird bei Erfolg auf eine zweite Seite weitergeleitet, welche die Ergebnisse präsentiert. Diese Ergebnisansicht bietet drei Tabs, um die gefundenen Tankstellen zu vergleichen. Der Tab *Liste* listet alle gefundenen Tankstellen auf, im Tab *Detail* werden die Informationen zu der in der Liste ausgewählten Tankstelle präsentiert und der Tab *Karte* zeigt alle gefundenen Tankstellen auf einer Plattform-spezifischen Karte an (Google Maps auf Android, Apple Maps auf iOS, Bing Maps auf Windows Phone).

In den Ansichten für die Favoriten und die Statistiken werden *CarouselPages* verwendet, diese erlauben das Durchblättern mehrerer Elemente durch horizontale Wischgesten über den Bildschirm. In den Ansichten *Einstellungen* und *About* kommen *ScrollViews* zum Einsatz, um durch eine vertikale Wischgeste mehrere Bedienelemente zugänglich zu machen.

Aufgrund der geringen Zeit für die Implementierung musste der designspezifische Aspekt der Oberflächengestaltung auf die bloße Auswahl eines Farbschemas und die Auswahl von passenden Icons reduziert werden, wodurch das UI an vielen Stellen noch Verbesserungspotenzial hat. Dies hat jedoch auch den Vorteil, dass die App sehr einfach zu bedienen ist, da für fast alle Anwendungsfälle nur wenige Nutzeraktionen notwendig sind, meist reicht die Verwendung einer Schaltfläche aus. Die Ausarbeitung der App in nur einem Monat sowie fehlende Testmöglichkeiten führten zudem zu der fehlenden Unterstützung von unterschiedlichen Bildschirmgrößen, wodurch die Oberfläche auf abweichenden Bildseitenverhältnissen oder Bildschirmauflösungen möglicherweise eine suboptimale Nutzererfahrung anbietet.

Zudem benötigt die Verwendung von Xamarin Forms bei jedem Start der App eine ausgesprochen lang andauernde Initialisierung, die standardmäßig unter Android leider

nicht durch einen Begrüßungsbildschirm (engl. splash screen) überbrückt werden kann, wodurch ebenfalls die Nutzererfahrung leidet.

3 Ausblick

Aufgrund des Zeitmangels und der geringen Erfahrung mit Xamarin (Forms) blieben leider sowohl Grundfunktionalität und zusätzliche erdachte Features auf der Strecke. Nach dem Einreichen des Projektes bei der *Code Competition* können diese nun noch implementiert werden.

Zum einen klappte die Ermittlung der Daten für die Präsentation der Statistiken leider nicht, wodurch an dieser Stelle bisher Beispieldaten angezeigt werden, obwohl die Preise durchaus bereits in einer Datenbank gesammelt werden. Auch weitere Aufbereitungen für zusätzliche Statistiken sind für die Zukunft denkbar.

Eine wichtige Funktionalität, die auch von dem Betreiber der Tankerkönig API gewünscht wird, ist das Speichern von Abfragen, um wiederholende Zugriffe auf die gleichen Daten zu verhindern. Obwohl die dazu benötigten Daten bereits in der Datenbank hinterlegt sind, fehlt das Durchsuchen dieser um den Webrequest möglicherweise zu ersetzen.

Weitere Änderungen würden dann eine Anpassung und Verschönerung der grafischen Benutzeroberfläche bedeuten, um zum einen eine bessere Nutzererfahrung zu ermöglichen und zum anderen mehr Funktionalität (beispielsweise durch zusätzliche Suchparameter) zu ermöglichen.

Sollten sich zudem Testmöglichkeiten für die anderen Plattformen (iOS und Windows Phone) ergeben, wäre natürlich auch eine schnelle Adaption auf diese Plattformen durch Xamarin kein großes Problem.