Richtlinie zum „Software-Sprint“

StreetComplete für iOS

Schlussbericht

Zuwendungsempfänger:

Tobias Zwick

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen **01IS24S02** gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Ziel

Ziel war die Entwicklung eines Minimum Viable Products (MVP) von StreetComplete für iOS.

StreetComplete[1] ist eine Android-App, mit der jeder auch ohne technische Vorkenntnisse gamifiziert Beiträge zur OpenStreetMap, der freien Wiki-Weltkarte, leisten kann.

OpenStreetMap selbst ist eine datenschutzfreundliche Alternative zu Online-Kartendiensten wie etwa dem Quasi-Monopol Google Maps. Man darf nicht vergessen, womit diese Unternehmen ihr Geld verdienen: Nutzerdaten. Dieser Datensammlung zu entkommen, ist schwierig. Insbesondere die Geo-Position und ihr Verlauf ist ein wertvolles und schützenswertes Datum.

Nicht nur für Endnutzer-Karten und Applikationen sind OpenStreetMap-Daten (die u.a. durch StreetComplete-Nutzer beigetragen werden) interessant, sie werden zum Beispiel auch in der Forschung[2] und (in Produkten) für Behörden[3] genutzt.

Damit OpenStreetMap aber eine echte Alternative ist und bleibt, reicht keine freie Lizenz, die Daten müssen auch flächendeckend erfasst worden, korrekt und aktuell sein. Kurz, OpenStreetMap wird

[1] https://streetcomplete.app/?lang=de

[2] Zum Beispiel https://heigit.org/ , https://media.ccc.de/v/sotm2020-4344-earthquakes-and-openstreetmap

[3] Zum Beispiel https://www.fixmycity.de/

von seinen Beitragenden getragen. Je mehr Menschen beitragen, desto besser und nützlicher wird die Karte für alle.

Wie die Wikipedia ist OpenStreetMap ein Gemeingut, doch als Geodatenbank ist es wesentlich komplizierter und technischer, daran mitzuwirken. Darum ist es umso wichtiger, Möglichkeiten zu schaffen, dies zu vereinfachen.

iOS wird von etwa einem Drittel der Deutschen genutzt. Mehr Menschen zu ermöglichen, einfach beizutragen, führt letztlich zu einer vollständigeren und aktuelleren Karte, welche wiederum den Menschen, die OSM-basierte Karten nutzen, zugutekommt.

Umsetzung

Auf technischer Ebene sollte die iOS-Version der App nicht parallel zur bestehenden Android App neu entwickelt werden, sondern so umgeschrieben werden, dass sie letztlich plattformunabhängig funktioniert. Der Aufwand für eine komplette Neuentwicklung und Wartung der App mit ihren rund 100.000 Zeilen Quelltext wäre untragbar.

Für die iOS-Version sollte also so wenig zusätzlicher Code zu StreetComplete hinzugefügt werden wie möglich, sowie so viel Code wie möglich gemeinsam genutzt werden, um die anschließende Pflege des Projektes nicht zu erschweren.

Die App ist in der Programmiersprache Kotlin geschrieben, welche grundsätzlich plattformunabhängig genutzt werden kann. Um also eine iOS-Version auf Basis derselben Codebasis zu ermöglichen, sind folgende zusätzliche Schritte notwendig:

Erstens, Abhängigkeiten der App zu Java, Java-Bibliotheken und Android müssen schrittweise durch *Kotlin Multiplatform*[4] (KMP) Bibliotheken ersetzt werden, um von den plattformabhängigen Implementationen unterschiedlicher Funktionalitäten zu abstrahieren. Dies umfasst vor allem den Austausch oder im Einzelfall die Neuimplementierung von Programmbibliotheken wie etwa die Kommunikation per HTTP über Internet, Parsen von verschiedenen Datenformaten, Zugriff auf eine Datenbank, Logging, Unit-Tests und so weiter.

Zweitens, die Nutzeroberfläche muss schrittweise vom Android-View-System zu dem UI-Framework *Compose Multiplatform*[5] (zusammen mit *ViewModels*[6] und *Navigation*[7]) migriert werden. Dies verringert zwar nicht den initialen Aufwand gegenüber zum Beispiel der Alternative, die UI für iOS in SwiftUI nativ neu zu entwickeln, bietet aber den klaren Vorteil, dass der Code für die Nutzeroberfläche hinterher je Plattform nicht getrennt, also doppelt, gepflegt und erweitert werden muss. Zudem kann die Migration schrittweise

[4] https://kotlinlang.org/docs/multiplatform.html

[5] https://www.jetbrains.com/lp/compose-multiplatform/

[6] https://developer.android.com/topic/libraries/architecture/viewmodel

[7] https://developer.android.com/guide/navigation

erfolgen[8]: der Wartungsaufwand wird so während der Entwicklungsphase nicht erhöht, da die Migration Stück für Stück integriert, statt parallel entwickelt wird.

Insgesamt handelt es sich hier um ein inkrementelles Vorgehen: Es ist geplant, die kleinen Schritte, die letztendlich zu einer iOS-Version führen, kontinuierlich zu integrieren und zu veröffentlichen, um das Risiko zu minimieren und früh mögliche Probleme zu erkennen.

Der Gesamtaufwand für dieses Vorhaben wurde auf etwa ein Jahr geschätzt. Für das Ende der sechsmonatigen Förderung war geplant, einen Prototypen der App auf iOS mit sehr eingeschränktem Funktionsumfang zu veröffentlichen.

Dadurch, oder schon durch die Ankündigung und Entwicklung dieses Vorhabens, sollte die Community aktiviert werden, zur schrittweisen Vervollständigung der iOS-App beizutragen - insbesondere diejenigen, die bereits Beiträge in diese Richtung geleistet hatten. Die Aktivität im Vorlauf der Förderung ließ darauf hoffen, dass mit wesentlichen Beiträgen innerhalb des Förderungszeitraums zu rechnen sei.

Ein wichtiger Meilenstein auf dem Weg zu diesem Ziel war der Austausch der Komponente, die für die Anzeige der Karte und allem, was darauf zu sehen ist, zuständig ist. Diese zentrale Aufgabe wurde bisher von der Bibliothek *tangram-es* wahrgenommen, welche jedoch seit Jahren nicht mehr gewartet oder weiterentwickelt wurde. Das war ein Problem, denn Fehler wurden nicht mehr behoben und zukünftig würde diese Bibliothek nicht mehr auf modernen iOS-Geräten eingesetzt werden, weil sie Apples neue Grafikschnittstelle *Metal* nicht unterstützt. *tangram-es* sollte durch *MapLibre-native* ausgetauscht werden. *MapLibre* wird aktiv weiterentwickelt und kann auf modernen iOS-Geräten eingesetzt werden, da es *Metal* unterstützt.

Weitere Details zum Vorgehen wurden im Issue Tracker detaillierter umrissen:

https://github.com/streetcomplete/StreetComplete/issues/5421

Beitrag des Projektes zu den Zielen der Förderinitiative „Software-Sprint“

Die Zielgruppe der App sind Menschen, die OpenStreetMap-Karten nutzen oder schon einmal etwas von OpenStreetMap gehört haben und gerne unterwegs gelegentlich einen Beitrag dazu leisten würden. Dies schließt OpenStreetMap-Enthusiasten mit ein, doch richtet sich vor allem an Menschen, denen es bisher zu zeitaufwändig oder technisch erschien, an der OpenStreetMap mitzuwirken.

Die iOS-App soll im App-Store verfügbar und auf https://streetcomplete.app sowie im OpenStreetMap-Wiki verlinkt sein. Die Android-Version ist in der OpenStreetMap Community bereits sehr bekannt und tief darin integriert.

[8] https://developer.android.com/jetpack/compose/migrate/strategy

Ausführliche Darstellung der Ergebnisse

Die für eine iOS-Version notwendigen Änderungen wurden kontinuierlich integriert, so wurden im Zeitraum der Förderung die Versionen **v57.0** bis **v59.0-alpha3** der App veröffentlicht, welche bereits die meisten der unten gelisteten Änderungen enthalten.

Der folgende Text ist in zwei Sektionen gegliedert: Erstens, das Auflösen von Abhängigkeiten zu Java etc. und zweitens die Neu-Implementierung der Nutzeroberfläche.

Eine ausführliche Darstellung findet sich ebenfalls im Kanban-Board der Projektseite: https://github.com/orgs/streetcomplete/projects/1

A. Auflösen von Abhängigkeiten

Zum Auflösen der Abhängigkeiten zu Java, Java-Bibliotheken und Android:

1. Sämtliche Kommunikation der App mit Services im Internet, allen voran der OpenStreetMap API, wurde neu implementiert unter Verwendung der Bibliothek **Ktor-Client**[9]. Vorher wurden dafür Klassen aus der Java Standardbibliothek verwendet. Dies beinhaltete ebenfalls die Neuimplementierung des Parsers, der Antworten der OpenStreetMap API unter Verwendung der KMP Bibliothek **XmlUtil** verarbeitet.

2. Analog zum letzten Punkt musste der Lese- und Schreibzugriff auf Dateien mit Hilfe von **kotlinx-io** neu implementiert werden, da vorher dafür Klassen aus der Java-Standardbibliothek verwendet wurden.[10]

3. Die Java Bibliothek **osmfeatures**[11] wurde zu KMP portiert. Diese Bibliothek wird verwendet, um Kartenelemente (*Points of Interest*) in der App korrekt lokalisiert zu benennen, sowie ist unabdingbar, um neue Elemente zur Karte hinzuzufügen.

4. Die Android-spezifische Implementierung persistent Einstellungen zu speichern wurde durch die KMP Bibliothek **Multiplatform Settings**, die dem selben Zweck dient, ersetzt.[12]

5. Nutzer können ihre eigene Einstellung, welche Aufgaben in der App angezeigt werden, mit anderen per QR Code teilen. Bisher wurde eine Java-Bibliothek zur Erkennung von QR-Codes genutzt, diese wurde durch die KMP Bibliothek **QRose** ersetzt.

6. Es wurde ein simpler HTML Parser geschrieben, da es für KMP (noch) keine geeigneten Bibliotheken gibt, die dieses ermöglichen würden. Dieser wird benötigt, da einige lokalisierte Texte in der App mit simplen HTML-tags (in fett, kursiv, als Spiegelstriche, und so weiter) stilisiert sind und entsprechend gerendert werden müssen.

---

[9] https://github.com/streetcomplete/StreetComplete/pull/5686 und https://github.com/streetcomplete/StreetComplete/pull/5483

[10] https://github.com/streetcomplete/StreetComplete/pull/5656

[11] https://github.com/westnordost/osmfeatures/

[12] https://github.com/streetcomplete/StreetComplete/issues/5419

Die aktuelle Schätzung ist, dass in diesem Themabereich noch ein Drittel des Weges für eine komplette iOS-Version zu gehen ist.

38 %	31 %	31 %
Bereits vor Beginn der Förderung fertiggestellt	**Fortschritt innerhalb der Förderung**	**Noch offen**

B. Neu-Implementierung der Nutzeroberfläche

Absolut gesehen ist der Aufwand für eine Neu-Implementierung der Nutzeroberfläche in Compose Multiplatform etwa doppelt so hoch wie für die Auflösung von Abhängigkeiten zu Java etc. und macht damit den Großteil des Gesamtaufwandes aus.

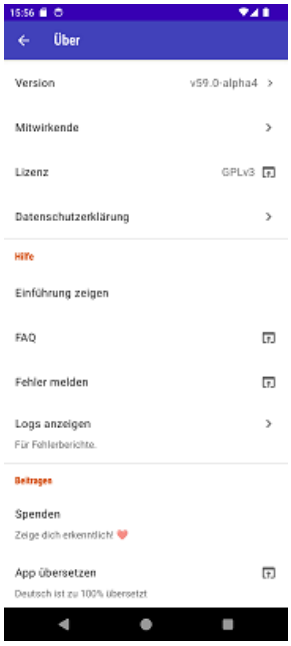
Schon vor Beginn der Förderung wurde zusammen mit einem Entwickler aus der Community als Vorbedingung für die Migration zu Compose damit begonnen, zu **MapLibre-native** zu migrieren. [13]

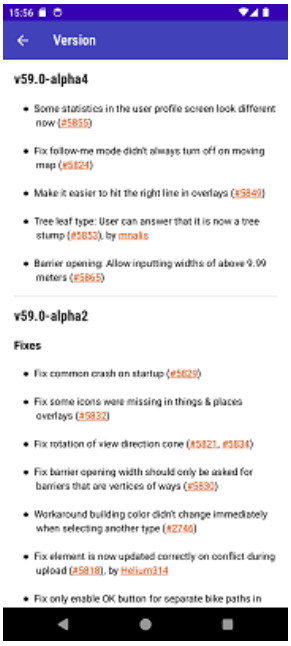
Weiterhin wurden alle Bildschirme und Unterbildschirme bis auf den Hauptbildschirm komplett zu Compose migriert, sowie alle Elemente, Dialoge, Seitenleisten usw., die auf dem Haupt-Bildschirm zu sehen sind. In einigen Fällen wurde die Neu-Implementierung dafür genutzt, die UI zu verbessern. Im Einzelnen:

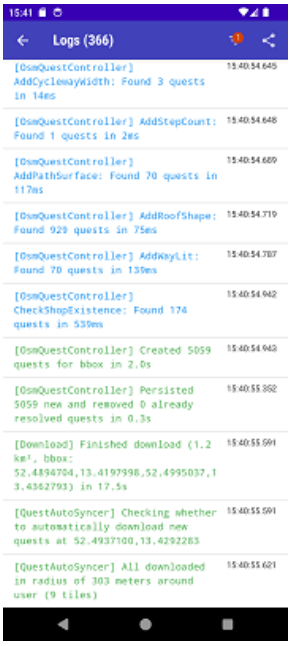
1. **Über-StreetComplete-Bildschirm**, Versionshistorie-Bildschirm, Mitwirkende-Bildschirm, Datenschutzerklärung-Bildschirm und Logs-Bildschirm[14]

[13] https://github.com/streetcomplete/StreetComplete/pull/5693

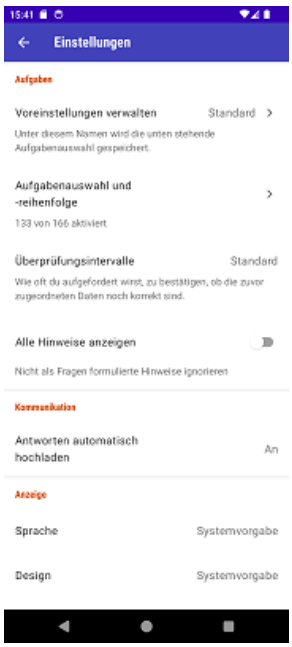
[14] https://github.com/streetcomplete/StreetComplete/pull/5719

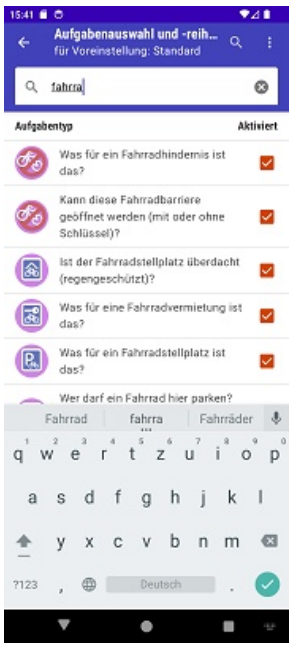






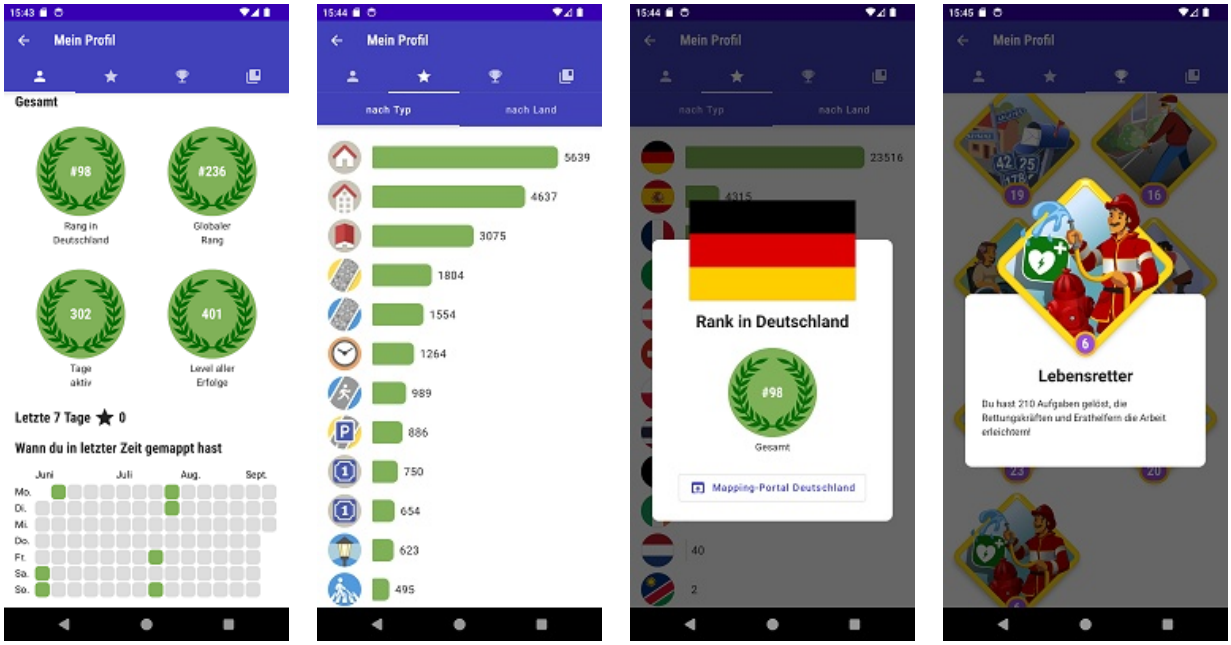
2. **Einstellungen-Bildschirm**, Aufgaben-Auswahl-Bildschirm, Voreinstellungen-Verwaltungs-Bildschirm inklusive Teilen von Voreinstellungen per QR Code[15]



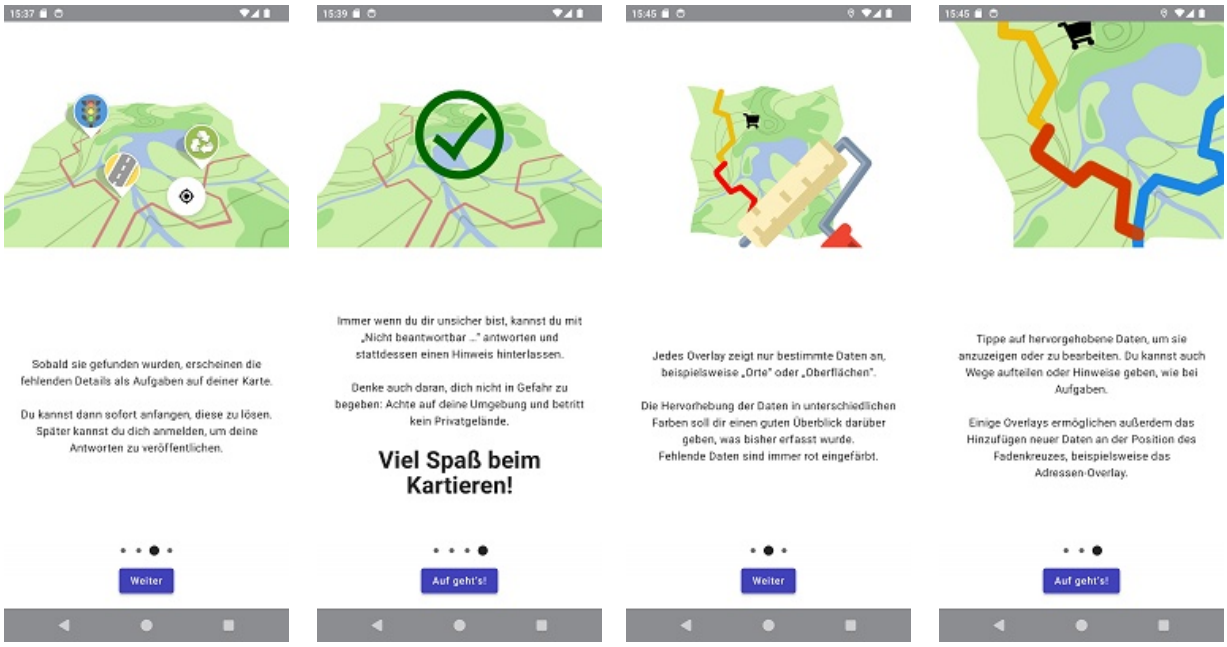


[15] Siehe 13

3. **Nutzer-Bildschirm**, Mein-Profil-Bildschirm, Login-Bildschirm, Aufgaben-Statistiken-Bildschirm (jeweils nach Aufgabentyp und nach Land), Errungenschaften-Bildschirm, Link-Sammlungs-Bildschirm[16]



4. **Einführungs-Bildschirm**, Einführung-in-Overlays-Bildschirm[17]



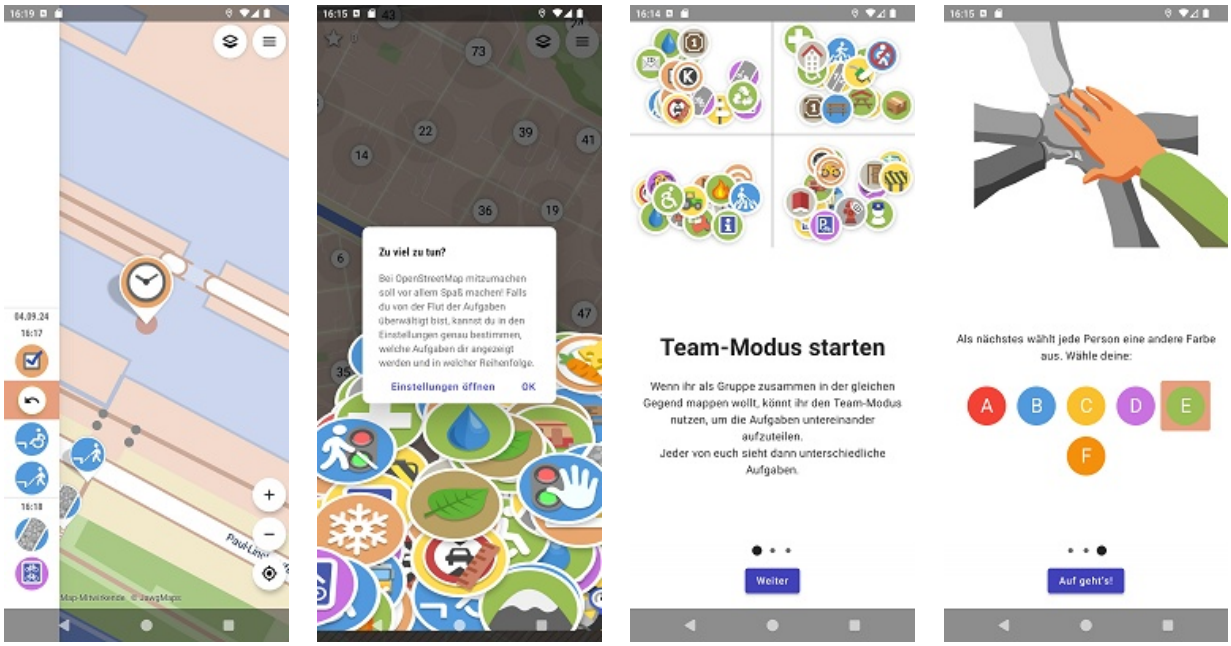
5. im **Haupt-Bildschirm** wurden folgende Komponenten neu implementiert: alle Nachrichten-Dialoge, Hauptmenü, Team-Modus, alle Bedienelemente, Änderungshistorie-Seitenleiste mit

[16] https://github.com/streetcomplete/StreetComplete/pull/5607

[17] https://github.com/streetcomplete/StreetComplete/pull/5746

Rückgängig-Dialog, Versionshistorie-Dialog, Voreinstellungs-Importierungs-Dialog und Fehlerdialoge[18].

Der Dialog für den Team-Modus wurde zudem bei dieser Gelegenheit grundlegend neu designt.



Bei der Implementation der neuen Nutzeroberfläche wurde besonders darauf geachtet, dass diese auch gut funktionieren

- im Dark-Theme,
- im Querformat (Landschaftsmodus),
- auf unterschiedlichen Bildschirmgrößen (von iPhone SE bis hin zu großen Tablets) und
- in Sprachen, in denen von rechts nach links geschrieben wird, d.h. alle Bedienelemente ebenfalls von rechts nach links angeordnet sein sollen. Die App ist in knapp fünfzig Sprachen verfügbar.

Noch migriert werden müssen alle Formulare für Aufgaben und die Kartenansicht (MapLibre) muss für Compose implementiert werden. Schätzungsweise sind für eine komplette iOS-Version noch 46% des Weges zu gehen, denn die Formulare sind für viele Aufgabentypen maßgeschneidert.
Jedoch müssen für eine Version mit ausdrücklich begrenztem Funktionsaufwand nicht unbedingt 100% des UI-Codes migriert werden.

54 %	46 %
Fortschritt innerhalb der Förderung	**Noch offen**

[18] https://github.com/streetcomplete/StreetComplete/pull/5799

MVP

Das Ziel, am Ende der Förderung einen Prototyp für iOS veröffentlichen, konnte leider nicht erreicht werden.

Der Grund dafür waren in erster Linie Probleme bei der Integration der Komponente, die das Rendern der Karte übernehmen sollte (*MapLibre-native*). Da diese Komponente zentral für die Funktionalität der App ist, war es nicht sinnvoll, bestimmte Teile der Migration zu Compose anzugehen, bevor die Integration von MapLibre abgeschlossen war.

Einerseits hat die Integration selbst nicht nur etwa anderthalb Monate länger gedauert als erwartet, es stellte sich auch heraus, dass die Komponente noch viele Fehler beinhaltet, von denen die kritischen erst behoben werden müssen, bevor MapLibre produktiv eingesetzt werden kann.[19]

Dieser Umstand hat den Plan blockiert, zuerst nur die für ein MVP wichtigen Teile der App zu migrieren. Erst zwei Wochen vor Ende der Förderung wurde ein Workaround für den kritischsten Fehler gefunden, so dass MapLibre nun doch produktiv ab dem nächsten Release der App eingesetzt werden kann.

Statt also zuerst nur selektiv die Teile zu migrieren, die für einen MVP unabdingbar sind, wurden während der Förderung zuerst die Teile der App migriert, die nichts mit der Kartenansicht zu tun haben.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Da das Ziel innerhalb des Förderungszeitraums, zumindest einen Prototypen der App für iOS zu veröffentlichen, nicht erreicht wurde, können iOS-Nutzer diese App leider noch nicht benutzen.

Die bisher in das Projekt geflossene Arbeit ist jedoch nicht verloren, denn diese wurde kontinuierlich in die bereits im Einsatz befindliche App integriert und ist nun Teil der veröffentlichten App, wird also weiterhin auf Stand gehalten. Das bedeutet, dass dem Endziel, nicht nur einen Prototypen, sondern eine Komplettversion der App für iOS auf Basis einer einzigen Codebasis für alle Plattformen zu schaffen, wesentlich näher gekommen wurde. Und zwar wesentlich näher als es mit einem versuchten Durchstich zu einem Prototypen möglich gewesen wäre, da dafür kurzfristig viel Code hätte entstehen müssen, der hinterher nicht weiterverwendet werden hätte können.

Sehr viele Personen haben ein starkes Interesse an einer iOS-Version von StreetComplete geäußert, darunter auch Entwickler. Da es generell eine aktive Entwickler-Community rund um StreetComplete gibt[20], wäre eine mögliche Strategie, hier auf die Community zu setzen, um dieses Projekt zu Ende zu führen. Extrapolierend von Frequenz und Umfang bisheriger Beiträge würde dies bis zu einer Komplettversion für iOS jedoch (wenn überhaupt) mehrere Jahre dauern, daher ist es ratsam, auch in Zukunft sofern keine Anschlussförderung gefunden wird zuerst einen MVP einer iOS-Version der App

[19] Siehe die von uns erstellten Fehlerberichte im Bugtracker von MapLibre – insgesamt 26: https://github.com/maplibre/maplibre-native/issues?q=+is%3Aissue+author%3Awestnordost und https://github.com/maplibre/maplibre-native/issues?q=+is%3Aissue+author%3AHelium314

[20] Über 100 Mitwirkende: https://github.com/streetcomplete/StreetComplete/graphs/contributors

zu forcieren. Eine schon bestehende und funktionierende aber abgespeckte Version der App erhöht das Potenzial, weitere Entwickler anzuziehen.

Über eine mögliche iOS-Version hinaus wäre eine Weiterentwicklung für weitere Plattformen wie beispielsweise einem Desktop-Client wesentlich vereinfacht. Es ist allerdings zu früh, um über solche Weiterentwicklungen nachzudenken, solange eine Komplettversion für iOS noch nicht besteht.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Kotlin Multiplatform ist eine junges Ökosystem. Im Gegensatz zu Java-Ökosystem kann man hier nicht davon ausgehen, dass es selbst für übliche Anwendungen bereits sicherlich *multiplatform-fähige* Kotlin Programmbibliotheken mit kompatibler Lizenz gibt.

So blieb die Suche nach Programmbibliotheken mit folgenden Zuständigkeiten bisher erfolglos:

- **Gebietsschemata (Locale).** Das heißt, die Benennung unterschiedlicher Länder und Sprachen, sowie regionaler Formatierung und Benennung von Datum, Monaten, Wochen, Wochentagen usw.
- **MapLibre für Compose.** Rendering der Karte im Kontext des UI-Frameworks Compose Multiplatform.

Diese Situation kann sich jedoch in einem jungen Ökosystem wie diesem schnell ändern, so wurde die Suche danach bzw. die Arbeit daran in folgenden Fällen abgebrochen, da sich die Voraussetzungen während der Förderung geändert hatten:

- **YAML-Parser.** Der aktuell schon genutzte Parser soll demnächst komplett multiplatform-fähig werden.
- **Jetpack ViewModels.** Ein Ersatz für Android Jetpack ViewModels ist nicht mehr nötig, da Google selbst angekündigt hat, diese multiplatform-fähig machen zu wollen.
- **Jetpack Navigation.** Ein Ersatz für Android Jetpack Navigation ist ebenfalls aus den gleichen Gründen nicht mehr notwendig.
- **Mock-Bibliothek für Unit-Tests.** Ein Entwickler aus der Community hatte damit begonnen, zu einer multiplatform-fähigen Mock-Bibliothek zu migrieren. Die Entwicklung an dieser wurde jedoch eingestellt, da sie nicht mehr mit dem aktuellen Kotlin-Compiler kompatibel ist.
- Die Arbeit an mehreren Aufgaben, die Entwickler aus der Community übernehmen wollten, sind nach einigem hin- und her leider im Sand verlaufen, so dass ich diese Aufgaben letztlich selbst implementiert habe. (Dies betraf den überwiegenden Teil der Aufgaben die im Abschnitt ***A. Auflösen von Abhängigkeiten*** gelistet sind)

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Zum Projekt „StreetComplete für iOS“

Master-Ticket: https://github.com/streetcomplete/StreetComplete/issues/5421

Projekt-Management Board: https://github.com/orgs/streetcomplete/projects/1

Folien des Vortrages bei der State of the Map Europe (jährliche OpenStreetMap Konferenz) im Juli 2024, mit Notizen. Leider wurden keine Aufnahmen gemacht:

https://github.com/streetcomplete/StreetComplete/raw/master/res/documentation/StreetComplete%20for%20iOS.pptx

Allgemein zur App

Website – https://www.streetcomplete.app

GitHub - https://github.com/streetcomplete/StreetComplete

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Es wurde anfangs geschätzt, dass die Portierung der App zu iOS etwa ein Jahr Aufwand für eine Person in Vollzeit bedeuten würde. Diese Schätzung erscheint mir weiterhin zutreffend.

Das ursprüngliche Ziel war also, in dem Zeitraum der Förderung, selektiv die Teile davon zu implementieren, die zu einen Prototypen mit einem minimal praktikablen Funktionsumfang (MVP) führen.

Dass dies vermutlich nicht möglich sein würde, stellte sich schon früh im Projektverlaufs heraus, da sich der Einsatz einer zentralen Komponente als wesentlich zeitaufwändiger und problematischer herausstellte, als angenommen: Durch *MapLibre-native* war bis kurz vor Ende des Förderungszeitraumes die Arbeit an allem, was in irgendeiner Form mit der Karte interagierte, geblockt. Zudem blieb die erhoffte Mitwirkung der StreetComplete-Community an diesem Projekt innerhalb des Förderungszeitraumes im erhofften Umfang weitgehend aus.

Aufgrund dessen wurden die Prioritäten geändert:

1. Die Schnittstellen zwischen neuem UI-Code und altem UI-Code sollten möglichst reduziert werden, um nach Projektende keine Baustellen oder Inkonsistenzen im Code zu hinterlassen und damit die zukünftige Arbeit am Projekt nicht unattraktiver zu machen.
   Dies ist ein Grund, warum zuerst alle Bildschirme außer dem Hauptbildschirm mit der Karte neu implementiert wurden.

2. Wichtige Komponenten wurden selbst implementiert statt auf Mitwirkung der Community zu setzen und damit zu riskieren, dass andere Aufgaben oder Folgeaufgaben über eine unbestimmte Zeit blockiert werden.
   Mitwirkung der Community ist immer eine schöne Sache und Überraschung, aber es ist keine gute Idee, darauf zu zählen.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Keine.