Master-Projekt

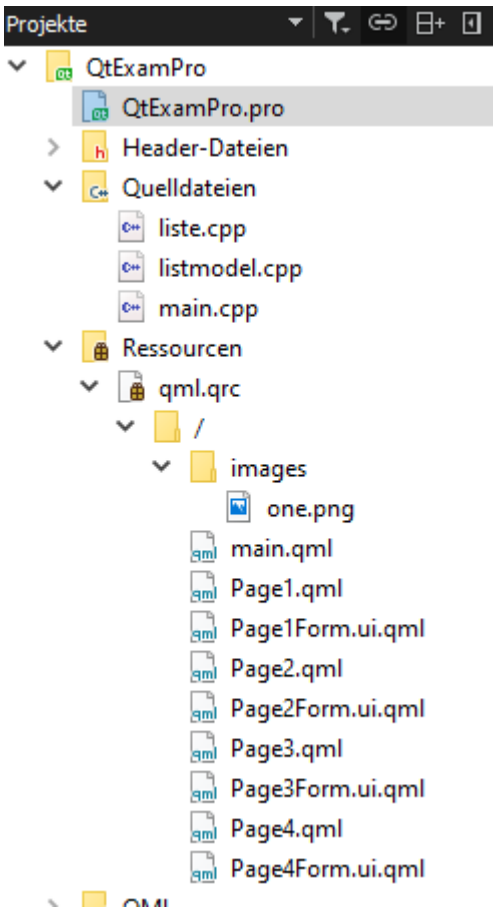
Vergleich von zwei erarbeiteten mobilen Android-Applikationen in unterschiedlichen Sprachen (JAVA und C++/QML) und Umgebungen (Android Studio und Qt Creator)

Aufgabe

Die Aufgabe dieses Projektes ist die Erarbeitung einer Beispiel-Applikation für das Betriebssystem Android. Diese dient als Vorlage für die Erstellung einer ähnlichen App in der Cross-Plattform Qt.
Die Beispiel-App wird im *Android Studio* in der Programmiersprache Java erstellt, wohingegen die zweite Mobilanwendung in der Qt-Umgebung *Qt Creator* in C++ und QML modelliert wird. Es soll gezeigt werden, in welchen Punkten sich die Apps unterscheiden und übereinstimmen.

Installationen

Für die Anwendung des Android Applikation ist es empfehlenswert Android Studio zu installieren und die App darüber zu kompilieren.



Für die Qt-App benötigt man den Qt Creator. Dort kann man ein exisitierendes Projekt über „Datei“ → „Datei oder Projekt öffnen“ die *.pro*-Datei auswählen und das Projekt mit seinen Dateien sollte in der Ansicht „Projekte“ angezeigt werden. Dort wird in der obersten Hierarchiestufe der Projektname angezeigt. Klappt man diesen Ordner auf sollten weitere Unterordner wie Header-Dateien, Quelldateien usw. erscheinen. Im Ordner „Ressourcen“ befindet sich die Datei „qml.qrc“. In dieser müssen alle QML-Dateien aufgelistet sein und das one.png, damit sie in dem Projekt eingegliedert sind. Es gibt auf der 2.Hierarchieebene den Ordner QML, aber es ist wichtiger, dass die Dateien im Ordner Ressourcen → qml.qrc → / enthalten sind.

Anforderungen an die Apps

Beide Applikationen sollen möglichst darstellungsähnlich und gleichwertig aufgebaut sein. Dafür bekommen beide eine Start-, Listen-, Karten- und Formularansicht.

Darstellung / Umsetzung

Menüführung: In Qt gibt es für Android nicht die typischen Android-Designs und Hilfsmittel. Das Menü ist in der Qt-App mit einer Tab-Bar umgesetzt, die am unteren Rand zu sehen ist. Dadurch kann man die einzelnen Tabs anklicken, um zu den verschiedenen Fenstern zu gelangen oder durch Wischen kann der Nutzer zwischen den Fenstern wählen. Angefangen beim Home-Bildschirm, hin zum Listenbereich und über die Kartenansicht zum Formular.
In der Java-App gibt es die ActionBar am oberen Rand der App. Links steht der Name der Applikation und rechts ist das Overflow-Menü, wo man zu den drei Fenstern *Liste*, *Karte* und *Formular* gelingt beim Anklicken. Über den „Zurück-Button“, den jedes Android-Smartphone besitzt, kommt der Anwender von jedem Fenster zurück zum Home-Bildschirm.

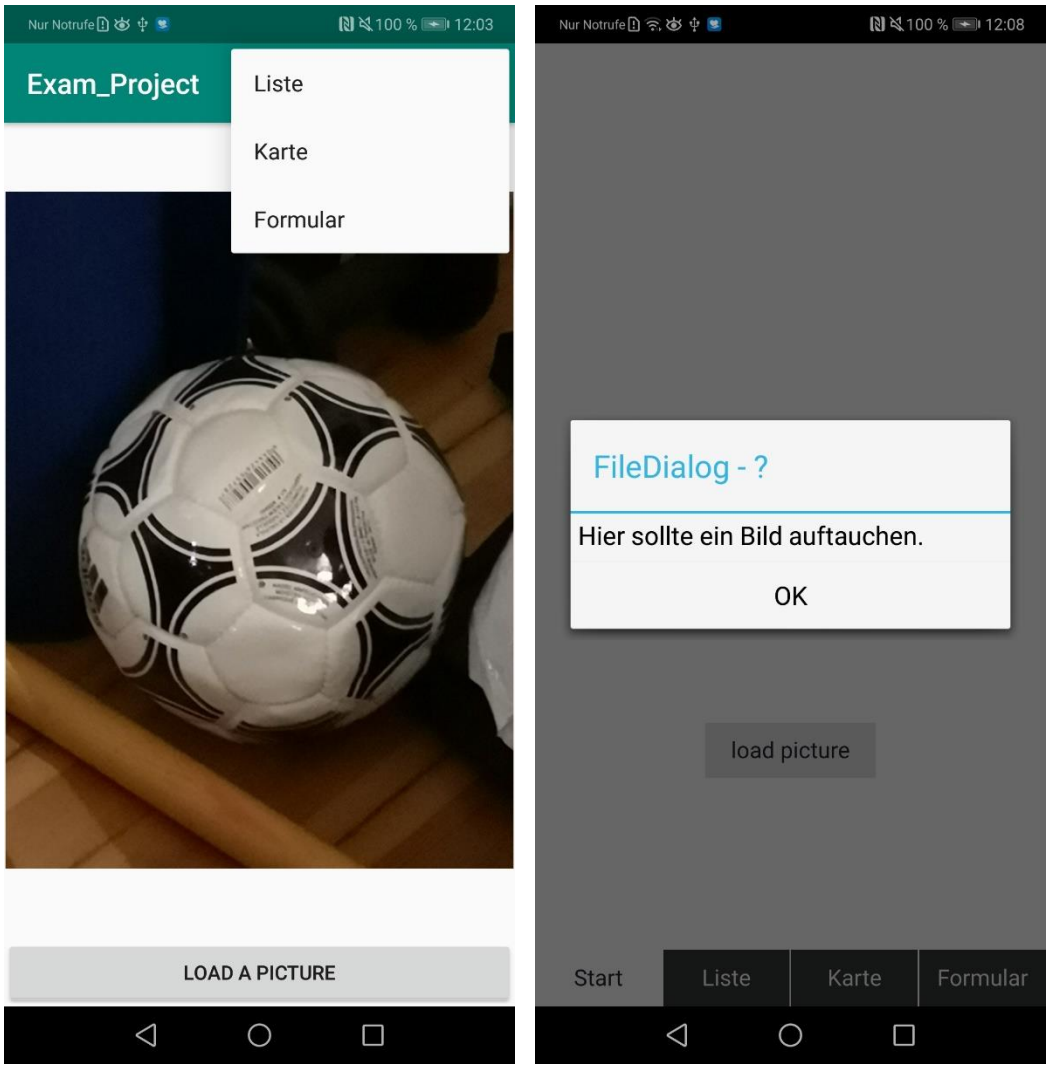


Abbildung 1:Startbildschirm Android Studio und Qt

Start-Bildschirm: Auf dem Startbildschirm der Apps (Abbildung 1) soll eine ImageView und ein Button sein. Der Button in der Java-App führt in die Fotogalerie des Smartphones, wenn man die Berechtigungen freigeschaltet und Zugriff auf diese hat. In der Qt-App kann man leider kein Bild öffnen, da es nicht die Möglichkeit gibt, über QML auf den internen Speicher zu zugreifen. Deswegen wird in dieser Version nur ein Dialogfenster nach Betätigung des Buttons geöffnet, der auf das Fehlen hinweist.

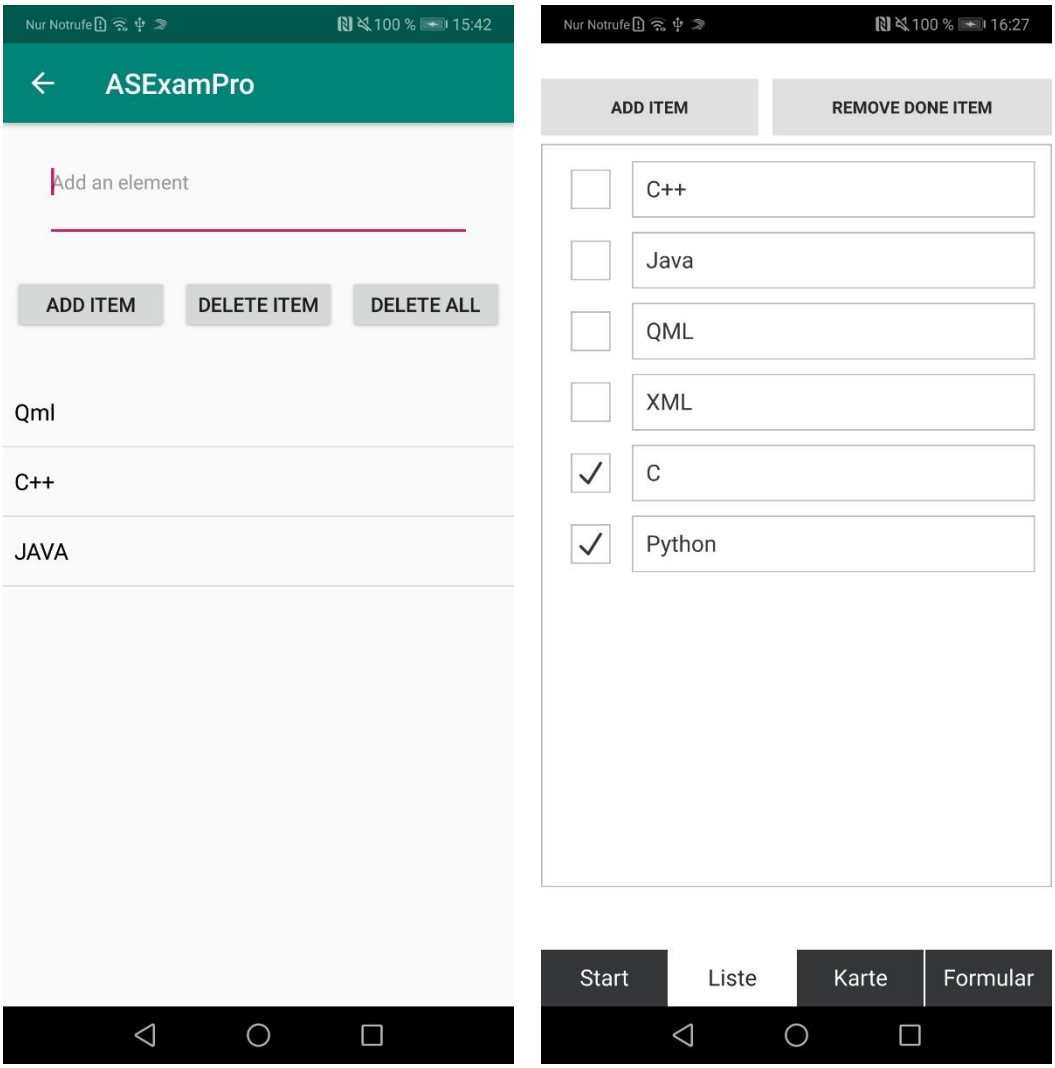


Abbildung 2: Listenansicht Android Studio und Qt

List-Bildschirm: Die zweite Ansicht, ist die der Listview (siehe Abbildung 2). In der Java-App gibt es in dieser Activity ein beschreibbares Textfeld, drei Button und die Listview. Mit dem *Add Item*-Button wird der im Textfeld eingetragene Inputtext der Liste hinzugefügt. Der mittlere Button löscht das oberste Element der Liste. Ein Element kann ebenfalls durch ein Drücken auf seine Position gelöscht werden. Der *Delete All*-Button löscht alle Elemente der Liste.

In der Qt-App gibt es zwei Button und ein Listenareal. Der linke Button fügt der Liste ein leeres Element (*ADD ITEM*) hinzu. Dieses kann mit einem Text gefüllt werden. Neben dem Textfeld ist das Check-Feld, welches mit dem Text automatisch generiert wird. Dadurch kann Zeilenelement aktiviert und deaktiviert werden. Als letztes gibt es noch den *Remove Done Item*-Button. Mit diesem können die aktivierten bzw. mit einem Haken versehenden Listenelemente gelöscht werden. Einzeln oder mehrere zugleich.

Maps-Bildschirm: Dieser Bereich der App zeigt dem User die Kartenansicht (siehe Abbildung 3). In Qt kann man über die QML direkt verschiedene offene Kartenmodule einbinden. Zu wählen ist bspw. zwischen HERE Maps und OpenStreetMap. Ich habe mich für die Anwendung der OSM entschieden. Im Android Studio wurde die Kartenansicht mit Google Maps eingearbeitet. Dafür benötigt man einen Google API-Schlüssel, den man als Entwickler durch ein Google-Konto erhält.

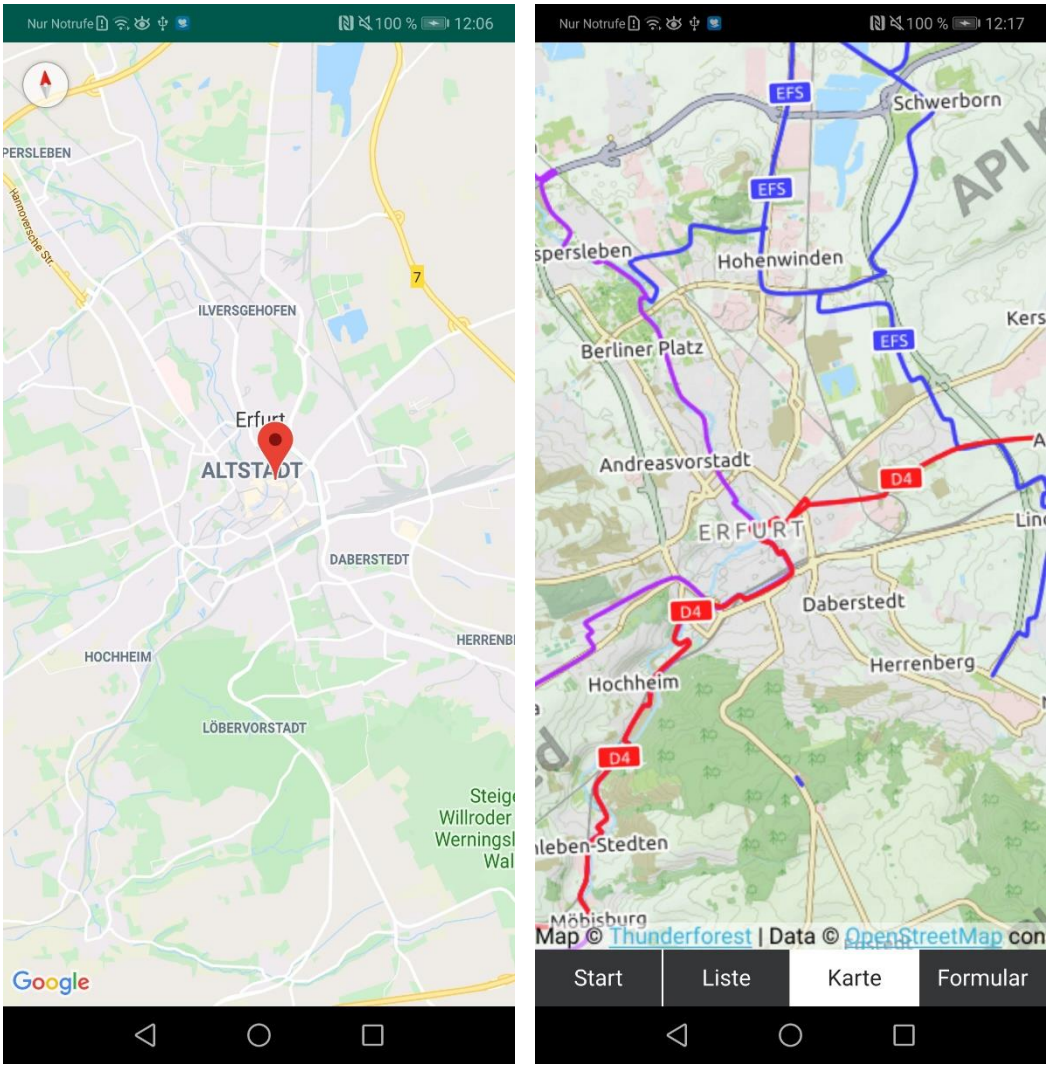


Abbildung 3:Android Studio (Google Maps) und Qt (OpenStreetMaps)

Formular-Bildschirm: Das Formular (siehe Abbildung 4) besteht aus dem Textfeld „Eingabeformular“ und Eingabefeldern für die Angaben Nickname, Email-Adresse, Telefonnummer und Geburtstag sowie einen Button.
Im Android-Studio kann direkt bestimmt werden, was für ein Typ das jeweilige Item ist, sodass keine Fehleinträge gemacht werden können. Diese sind in der Qt-Anwendung ebenfalls gegeben, aber die Felder sind trotzdem nicht auf diese beschränkt. Der Button soll die Angaben speichern, es gibt aber kein angelegtes Personenprofil, weshalb sich nur eine Dialogfenster mit einem Hinweis öffnet.
Während im Android-Studio die Ansicht des Formulars sehr benutzerfreundlich ist sowie die Eingabe der Daten, muss der Benutzer bei der Eingabe der Daten in der Qt-Anwendung etwas feinhändig sein.

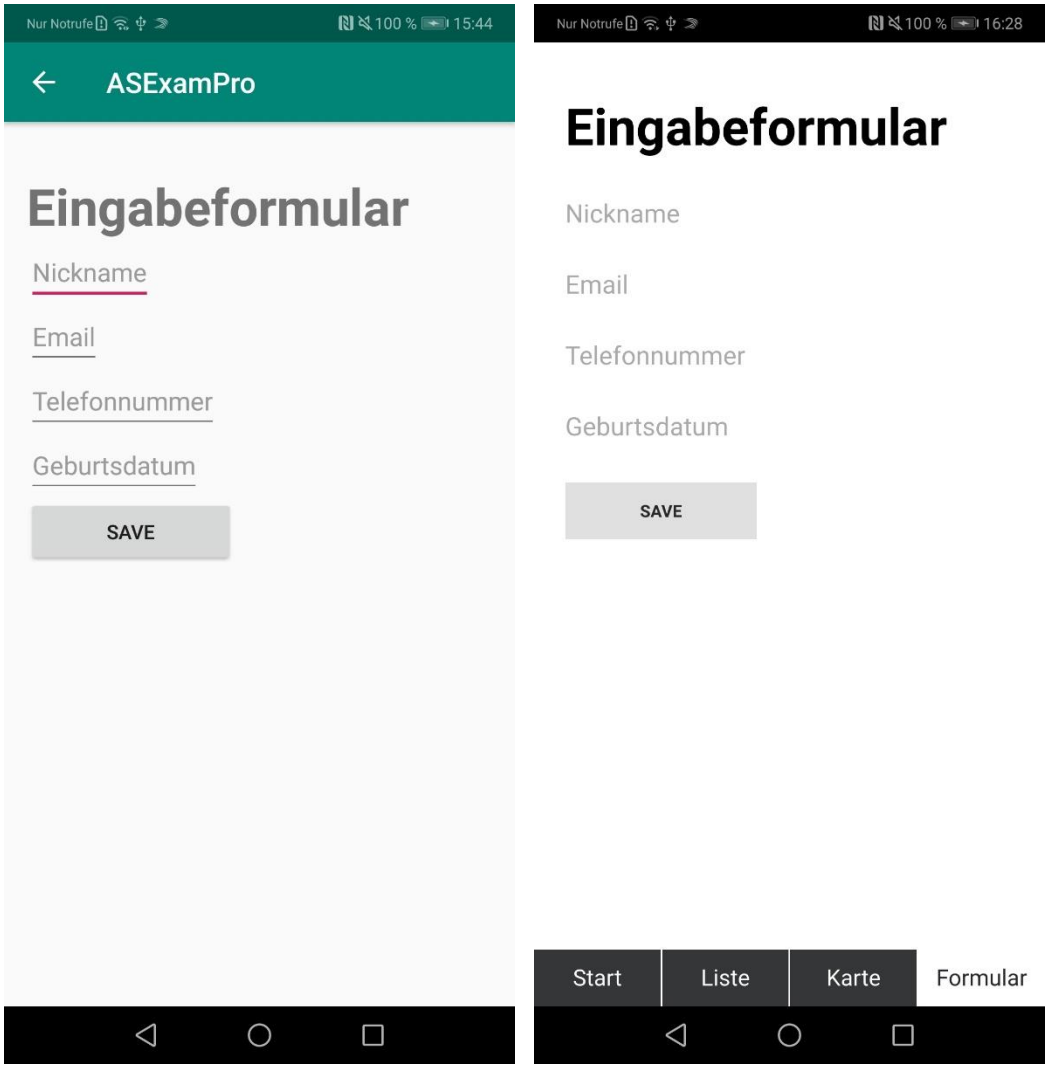


Abbildung 4:Formular in Android Studio und Qt

speziell zur Umsetzung: Im Android Studio werden die Activities durch Klassen definiert und das Layout mithilfe von XML-Strukturen erstellt. In Qt gibt es ui.qml-Dateien, mit denen man das Design der Activities bearbeiten kann und die wie in Android Studio eine graphische Vorschau haben. Hinzu kommen die QML-Dateien, die ebenfalls graphische Elemente enthalten können, diese aber nicht in einer Vorschau zwingend angezeigt werden können. Deshalb muss auch nicht jede qml-Datei mit einer ui.qml-Datei verbunden sein. Die QML-Dateien sind vermehrt für die Logik verantwortlich und in ihnen werden zumeist die Aufgaben der graphischen Elemente definiert. In der Qt-App gibt es für jede Ansicht eine ui.qml-Datei, denn nur so kann die SwipeView funktionieren.
Mit Hilfe von C++-Klassen können die Umsetzungen teilweise genauer erarbeitet werden, wie es hier in dieser Anwendung bei der Liste der Fall ist.

Bewertung

Die Darstellung und die klaren Strukturen im Android-Studio geben dem Nutzer mehr Freiheiten. Gleichzeitig sind die Anpassungen und Eingaben sehr übersichtlich für den Entwickler und den Anwender. In Qt hat der Entwickler ebenfalls eine Reihe von grafischen Hilfsmitteln, aber nicht in der Qualität für eine Android-Anwendung, wie im Android Studio. Für Desktopanwendungen sieht dies wiederum anders aus.

Nach dem ich verschiedene Erfahrungen mit Qt sammeln durfte, habe ich festgestellt, dass diese Umgebung für die Entwicklung von Desktop-Anwendungen am besten geeignet ist. Sie bietet für die C++-Entwicklung mit dem *Qt Creator* bzw. *Qt Designer* oder bspw. durch die Einbindung in Visual Studio eine unterstützende Arbeit in diesem Bereich an.
Weniger positiv sieht für die Entwicklung von Android-Anwendungen. Es war früher angedacht, dass die Zukunft der Entwicklung von mobilen Apps in dieser Umgebung stattfinden könnte. Aber es wurden nur die Grundsteine für Symbian - Nokia sowie iOS gelegt. Für die Android-Entwicklung gibt es nicht die umfänglichen Bibliotheken und Hilfsmittel, wie im Android Studio.