HOCHSCHULE DER MEDIEN

Herausgeber Prof. Dr. Barbara Dörsam und

Schriftreihe Bachelor-Resümee

Forschungsbereich: **Native iOS-App-Entwicklung**

Evaluation eines Konzeptes zur nativen iOS-App-Entwicklung mit webbasiertem Frontend

Welche Potenziale und Einschränkungen bietet ein Ansatz zur nativen iOS-App-Entwicklung mit Swift-basiertem Backend und webbasiertem Frontend im Vergleich zu einer vollständig mit Swift entwickelten iOS-App?

Niklas Burger

Impressum

Hochschule der Medien

Nobelstrasse 10

70569 Stuttgart

www.hdm-stuttgart.de

0711 8923-0

Autor

Niklas Burger

Betreuer

Prof. Dr. Barbara Dörsam und Mario Hamann

Datum

Juli 2023

Wirtschaftsingenieurwesen Medien

www.hdm-stuttgart.de/wing

hitzges@hdm-stuttgart.de

0711/8923-2634

Layout

Jochen Riegg

Fotos und Illustrationen

Innenteil: Niklas Burger

Bachelor-Resümee

Evaluation eines Konzeptes zur nativen iOS-App-Entwicklung mit webbasiertem Frontend

Welche Potenziale und Einschränkungen bietet ein Ansatz zur nativen iOS-App-Entwicklung mit Swift-basiertem Backend und webbasiertem Frontend im Vergleich zu einer vollständig mit Swift entwickelten iOS-App?

Niklas Burger

Juli 2023

Über den Autor

Niklas Burger studierte Wirtschaftsingenieurwesen Medien mit dem Schwerpunkt Digital Publishing Technologies an der Hochschule der Medien in Stuttgart. Die Bachelorarbeit wurde in Kooperation mit der Digital-Agentur Virtual Identity AG geschrieben, die ihren Fokus auf die Entwicklung moderner Web-Anwendungen legt.

Inhaltsverzeichnis

1. Einleitung

Das "iPhone ist ein wegweisendes und magisches Produkt, das jedem anderen Mobiltelefon um buchstäblich fünf Jahre voraus ist", sagte der mittlerweile verstorbene ehemalige CEO von Apple, Steve Jobs, als Apple mit der Einführung des ersten iPhones das Mobiltelefon neu erfand (Apple Inc., 2007). Heute ist das US-amerikanische Unternehmen drei Billionen US-Dollar wert und damit 1,6-mal so viel, wie die 40 umsatzstärksten Unternehmen Deutschlands (Mohr, 2023). Vergangenes Jahr erwirtschaftete Apple mit seinen mobilen Endgeräten, dem iPhone und dem iPad, einen Umsatz von 234,8 Milliarden US-Dollar. Dies entspricht rund 60 % des Gesamtumsatzes des Unternehmens (Apple Inc., 2022).

Unter Bezugnahme der Umsatz-Kennzahl, lässt sich festhalten, dass der Fokus des Milliardenkonzerns auf dem Hardware-Markt liegt. Nichtsdestoweniger spielt der Softwaremarkt eine ebenso bedeutsame Rolle. Insbesondere der Markt für mobile Anwendungen, die über den App Store auf den mobilen Endgeräten heruntergeladen werden können, stellt ein großes Potenzial dar. So verzeichneten der App Store von Apple und der Google Play Store 2021 einen Bruttoumsatz von beinahe 130 Milliarden US-Dollar (Sensor Tower Inc., 2022). Mobile Applikationen sind nutzungsfreundliche und unkomplizierte Anwendungen, die auf den mobilen Endgeräten ausgeführt werden.

Laut Apple stellt die Programmiersprache Swift in Kombination mit dem UI-Framework SwiftUI die beste Methode zur App-Entwicklung für iOS-betriebene Endgeräte dar (Apple Inc., 2019). Gleichwohl kamen in den vergangenen Jahren immer mehr Ansätze auf, die es ermöglichen, unter dem Einsatz unterschiedlicher Technologien native Anwendungen für iOS-Endgeräte zu entwickeln. Die gemeinsame Zielsetzung all dieser Ansätze besteht in der Entwicklung von Anwendungen für mobile Endgeräte, um das Potenzial des mobilen Applikationsmarkts auszuschöpfen. Die Bachelorarbeit knüpft an diese Zielsetzung an, indem sie eine Evaluation eines Konzepts zur Entwicklung nativer iOS-Apps mit einem webbasierten Frontend vorsieht.

2. Theorie

Gegenstand: Native iOS-App-Entwicklung

Die Bachelorarbeit beschäftigte sich mit der Evaluation eines neuartigen Konzepts zur Entwicklung nativer iOS-Apps mit einem webbasierten Frontend. Hierbei wurde folgende Fragestellung untersucht:

„Welche Potenziale und Einschränkungen bietet ein Ansatz zur nativen iOS-App-Entwicklung mit Swift-basiertem Backend und webbasiertem Frontend im Vergleich zu einer vollständig mit Swift entwickelten iOS-App?“

Diese Potenziale und Einschränkungen wurden anhand konkreter Anforderungen und Funktionalitäten evaluiert. Der geplante Ansatz soll einerseits maximale Funktionalität in Bezug auf die Integration mit dem mobilen Endgerät, den Zugriff auf Gerätefunktionen und die Performance liefern. Andererseits soll das Konzept Softwareentwickler:innen aus dem Bereich der Web-Entwicklung die Möglichkeit bieten, bekannte und etablierte webbasierte Konzepte nutzen zu können.

Eingrenzung

- **Betriebssystem**

  Gegenstand der Arbeit ist die Entwicklung nativer iOS-Anwendungen. Dementsprechend grenzt sich das Vorhaben von anderen Betriebssystemen ab. Des Weiteren grenzt sich das Vorhaben von Technologien ab, die auf die Entwicklung von Desktop- oder Web-Anwendungen abzielen.

- **Zielgruppe**

  Das geplante Konzept soll eine Möglichkeit für Softwareentwickler:innen im Bereich Web-Entwicklung darstellen, native iOS-Apps zu entwickeln und dabei auf webbasierte Konzepte zurückgreifen zu können.

- **Zu vergleichende Konzepte**

  Die in der Bachelorarbeit vorgenommene Marktanalyse dient dem inhaltlichen Vergleich des anvisierten Konzeptes mit etablierten Konzepten, zur Entwicklung mobiler Anwendungen. Dennoch ist an dieser Stelle hervorzuheben, dass das Ziel des geplanten Ansatzes die Darstellung einer adäquaten Alternative zu dem Standard der mobilen iOS-App-Entwicklung ist. Angesichts dieser Tatsache wurde das vorgesehene Konzept technisch nur mit dem „Swift-Konzept“ verglichen.

Native-Inertia-Ansatz

Der Native-Inertia-Ansatz basiert auf Inertia.js, einer opensource-Bibliothek zur Entwicklung vollständig clientseitig-gerenderten SPAs (Single Page Applications). Inertia.js funktioniert wie eine serverseitig-gerenderte Applikation, mit der Besonderheit, dass es sich bei den gerenderten Ansichten um Komponenten von einem JavaScript-Framework handelt. Inertia.js ist kein Framework, sondern wird als Verknüpfung zwischen serverseitigen- und clientseitigen Frameworks genutzt. Infolgedessen kann Inertia.js als Adapter zwischen Entwicklungsplattformen betrachtet werden.

Native-Inertia entstand als experimenteller Ansatz aus Inertia.js, der von der Virtual Identity AG, in Person von Mario Hamann, entwickelt wurde. Es handelt sich bei Native-Inertia um eine neuartige Herangehensweise zur Entwicklung nativer iOS-Applikationen, für die noch keine Erkenntnisse vorliegen. Der Ansatz wurde in der Bachelorarbeit evaluiert.

Das Native-Inertia-Konzept sieht eine Kommunikation zwischen nativen, mit Swift programmierten, Technologien im Backend und webbasierten Technologien im Frontend vor. Diese Kommunikation wird nicht standardmäßig von Inertia.js unterstützt, da Inertia.js nicht auf die Entwicklung nativer Applikationen abzielt. Der Grund für die Inkompatibilität ist, dass es sich bei XMLHttpRequests und XMLHttpResponses um Web-Technologien handelt, die im Swift-Kontext standardmäßig nicht unterstützt werden. Um diese Kommunikation zu ermöglichen, ist eine Erweiterung des bestehenden Intertia.js-Konzepts erforderlich. Es wird ein zusätzlicher „Backend-Adapter“ für Swift benötigt, sowie eine Zwischenschicht, die zwischen Web-Technologien und Swift-Technologien vermittelt. Die Zwischenschicht wird im Folgenden als „Native-Interceptor“ bezeichnet. Diese Erweiterungen wurden im Kontext des Native-Inertia-Konzeptes von der Virtual Identity AG umgesetzt und sind in der folgenden Abbildung blau markiert:

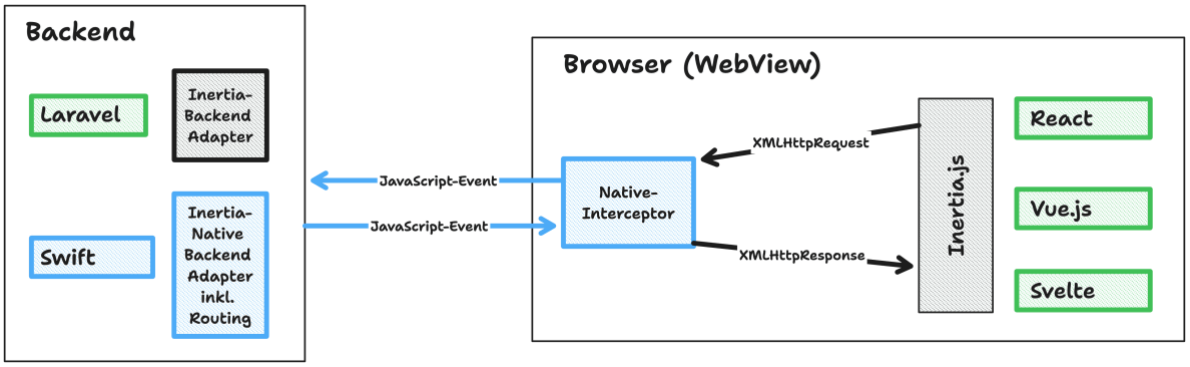

Abbildung 1 Native-Inertia. Kommunikation zwischen Backend und Frontend.

- Eine Native-Inertia-Applikation wird in einer WebView dargestellt. Hierbei handelt es sich um einen in die native App eingebetteten Browser. Diese WebView reagiert auf JavaScript-Events.
- Der „Native-Interceptor“ vermittelt als Bestandteil des Frontends zwischen den Web-Technologien im Frontend und den Swift-Technologien im Backend.
- Die Frontend-Backend-Kommunikation innerhalb von Native-Inertia findet eventbasiert statt. Wenn Events im Frontend ausgelöst werden, werden XMLHTTPRequests an das Backend gesendet. Der „Native-Interceptor“ fängt diese Requests ab und wandelt sie in JavaScript-Events um, damit diese verarbeitet werden können.
- Für das Backend wurde ein weiterer Adapter speziell für Swift entwickelt, der unter anderem die Routing-Funktion ermöglicht. Der „Backend-Adapter“ sorgt dafür, dass sich das Swift-Backend wie ein Web-Server verhält.
- Das Backend empfängt die JavaScript-Events und sendet JavaScript-Events als Response zurück an das Frontend. Der „Native-Interceptor“ fängt diese Responses ab und wandelt diese in XMLHTTPResponses um, die von Intertia.js verarbeitet werden können.
- Für den Inertia.js-Anteil verändert sich in dem Native-Inertia-Konzept nichts: Es werden XMLHTTPRequests gesendet und XMLHTTPResponses empfangen.

Die Erweiterungen des Inertia.js-Konzepts in Native-Inertia ermöglicht die Kommunikation zwischen nativen, mit Swift programmierten, Technologien im Backend und webbasierten Technologien im Frontend.

Marktanalyse

Im Rahmen einer Marktanalyse wurden bestehende Konzepte zur Entwicklung mobiler Anwendungen unter Verwendung von Web-Technologien vorgestellt. Die Charakteristiken und sich daraus ergebende Vorteile und Einschränkungen der jeweiligen Konzepte wurden dabei untersucht.

Nachdem im Zuge der Marktanalyse verwandte Konzepte zu Native-Inertia analysiert wurden, wurden die Erkenntnisse genutzt, um diese Ansätze von dem Native-Inertia-Ansatz abzugrenzen:

- **Native Inertia - Native iOS-App**

  Eine native iOS-App sieht eine vollständig mit Swift entwickelte Applikation vor. Im Gegensatz zu Native-Inertia sieht dieser Ansatz keine Aufteilung in Frontend und Backend vor, die die Präsentationsschicht von der Anwendungs- und Datenschicht trennt. Das Konzept der nativen App-Entwicklung schließt standardmäßig die Nutzung von Web-Technologien aus. Es ist möglich, innerhalb einer nativen iOS-App eine WebView einzubetten. Das ist eine vom nativen Code abgekapselte Komponente, die keine Auswirkungen auf den nativ programmierten Programmteil hat und nur zu Darstellungszwecken dient.

- **Native Inertia - Nativ-kompilierte App**

  Bei einer nativ-kompilierten App handelt es sich um eine auf Web-Technologien beruhende Codebasis, die mithilfe dafür vorgesehener Frameworks in nativen Code kompiliert wird. Durch die Komplexität des nativ kompilierten Codes sollten darin keine Änderungen vorgenommen werden, was mit einer eingeschränkten Kontrolle aus Entwicklersicht einhergeht. Native-Inertia sieht keine Kompilierung des Frontend-Codes vor.

- **Native Inertia - Full-Stack JavaScript App**

  Der wesentliche Unterschied zwischen den hier betrachteten Entwicklungssystemen ergibt sich aus der jeweils unterschiedlichen Codebasis der Konzepte.

  Die Basis einer Full-Stack JavaScript App ist eine Web-Anwendung, die durch die Verwendung von Frameworks in einem Browser dargestellt wird, der in eine native Umgebung integriert ist. Somit handelt es sich bei einer Full-Stack JavaScript App um eine JavaScript-Anwendung, die es ermöglicht, Swift zu nutzen.

  Die Basis einer Native-Inertia-Anwendung ist eine Swift-Applikation. Native-Inertia sieht eine klare Trennung des mit Swift programmierten Backends und des mit Web-Technologien programmierten Frontends vor. Somit verfolgt das Native-Inertia-Vorhaben den umgekehrten Ansatz einer Full-Stack JavaScript App. Es handelt sich um eine Swift-App, die die Möglichkeit bietet, Web-Technologien im Frontend zu verwenden.

- **Native Inertia - Progressive Web App**

  Die Abgrenzung dieser beiden Konzepte ist offensichtlich, da es sich bei einer PWA um eine Web-Anwendung handelt, die für die Verwendung auf mobilen Endgeräten optimiert ist. Eine Native-Inertia-App ist hingegen eine Swift-App, die die Möglichkeit bietet, Web-Technologien im Frontend zu verwenden.

3. Anforderungsanalyse

„Welche Anforderungen muss eine Native-Inertia-App erfüllen, um eine adäquate Alternative zu einer vollständig mit Swift entwickelten App darzustellen?“

Im Rahmen einer Literaturrecherche wurden folgende Anforderungskategorien an Konzepte zur nativen Anwendungsentwicklung ermittelt und im Rahmen der Arbeit detailliert beschrieben:

- Anforderungskategorie 1: Zugriff auf Gerätehardware
- Anforderungskategorie 2: Zugriff auf plattformspezifische Funktionalitäten
- Anforderungskategorie 3: Gute Performance
- Anforderungskategorie 4: Natives Look-and-Feel
- Anforderungskategorie 5: Entwicklerunterstützung
- Anforderungskategorie 6: Wirtschaftlichkeit

Die Anforderungskategorien eins bis vier wurden unter dem Begriff „anwendungsbezogene Anforderungskategorien“ zusammengefasst, da sich diese auf verschiedene funktionelle oder qualitative Aspekte einer Anwendung beziehen. Demgegenüber beziehen sich die Anforderungskategorien fünf und sechs auf übergeordnete Aspekte, wie die wirtschaftliche Effizienz und die Unterstützung für Entwickler:innen. Diese Faktoren stehen nicht unmittelbar mit der zu entwickelnden Anwendung in Zusammenhang. Da der Fokus der Arbeit auf Funktionalität und der Zielgruppe Softwareentwickler:innen aus dem Bereich der Web-Entwicklung lag, waren die Anforderungskategorien fünf und sechs kein Gegenstand der Bachelorarbeit.

Aus den priorisierten anwendungsbezogenen Anforderungskategorien wurden konkrete Anforderungen abgeleitet, die mit dem Native-Inertia- und dem „Swift-Ansatz“ implementiert und gegenübergestellt wurden. Die Vorgehensweise sah die Wahl von funktionellen Anforderungen vor, die den Zugriff auf die Gerätehardware oder plattformspezifische Funktionalitäten bieten. Diese wurden nach der Implementierung auf Basis der qualitativen Anforderungskategorien Performance und Look-and-Feel verglichen und evaluiert. Daraus ergab sich folgende Anforderungsliste:

- **Zugriff auf plattformspezifische Funktionalitäten**

  Anforderung 1: Routing

  Anforderung 2: Lokale Datenverwaltung mit Core Data

- **Zugriff auf Gerätehardware**

  Anforderung 3: GPS-Tracker

  Anforderung 4: Kompass

  Anforderung 5: Kamera und Bilder

Es wurde die Hypothese aufgestellt, dass sofern es mit dem Native-Inertia-Ansatz möglich ist, die ausgewählten funktionellen Anforderungen umzusetzen, ohne im Bereich Performance und Look-and-Feel dem „Swift-Ansatz“ nachzustehen, dass Native-Inertia unter Berücksichtigung des Untersuchungsbereiches der Arbeit eine adäquate Alternative zu einer vollständig mit Swift entwickelten App darstellt.

4. Evaluation

Die Zielsetzung dieses Kapitels war die Evaluation des Native-Inertia-Ansatzes in Bezug auf die ermittelten Anforderungen. So konnte überprüft werden, welche der in der Anforderungsanalyse ermittelten Anforderungen umgesetzt werden können. Wenn das Vorhaben umgesetzt werden konnte, wurde dieses auf verschiedene Aspekte überprüft, um auftretende Einschränkungen zu evaluieren und Potenziale zu bewerten. Für jede dieser Anforderungen wurden folgende Schritte durchgeführt:

- **Konzeption**

  Es wurde eine Anforderungsspezifikation vorgenommen und ein Konzept angefertigt.

- **Methode**

  Der Methoden-Abschnitt umfasste die Implementierungsdetails der jeweiligen Anforderung. Jede Anforderung wurde durch ein Minimal Viable Product (MVP), einer möglichst minimalistischen Implementierung, überprüft.

- **Ergebnis**

  Im Ergebnisteil wurden die implementierten Leistungsmerkmale auf verschiedene qualitative Aspekte, wie Performance und Look-and-Feel geprüft und ein Vergleich zwischen der Swift-Implementierung und der Native-Inertia-Implementierung vorgenommen.

  In Bezug auf das Look-and-Feel soll erwähnt sein, dass die in den MVPs entwickelten Anwendungen in erster Linie der Evaluation dienen, ob der Native-Inertia-Ansatz die jeweilige Anforderung umsetzen kann. Somit liegt der Fokus auf der Funktionalität. Des Weiteren ist an dieser Stelle hervorzuheben, dass die von Apple verwendeten Technologien den Referenzwert für natives Look-and-Feel im iOS-Kontext stellen. Daher war das Ziel nicht die Replikation des mit den Apple-Technologien erzielten Look-and-Feels, sondern vielmehr im Bereich Funktionalität gleichwertige Lösungen zu bieten. So soll durch die Schnittstellenoffenheit die Möglichkeit zur individualisierten Gestaltung vorhanden sein, die es unter entsprechendem Aufwand zulässt, das von Apple gewohnte Look-and-Feel nachzustellen.

- **Diskussion**

  In dem Diskussionsabschnitt wurden die Erkenntnisse aus dem Methoden- und dem Ergebnis-Abschnitt zusammenfassend evaluiert und Bezug genommen auf Limitationen und Potenziale.

Diese Schritte wurden für alle der fünf ermittelten Anforderungen umgesetzt. Des Weiteren wurde ein Belastungstest durchgeführt, der der Ermittlung diente, wie viele Informationen das Frontend innerhalb des Native-Inertia-Ansatzes in einem vorgegebenen Zeitintervall von dem Backend entgegennehmen kann. Die Intention dieses Tests war der Beweis, dass der Native-Inertia-Ansatz eine Möglichkeit bietet, auf Echtzeitdaten basierte Anwendungen zu entwickeln.

5. Zusammenfassende Evaluierung

Alle ermittelten Anforderungen konnten mit dem Native-Inertia-Ansatz erfolgreich und effizient umgesetzt werden. In der Gegenüberstellung mit dem „Swift-Ansatz“ sind unter dem Aspekt Funktionalität keine Limitierungen des Native-Inertia-Konzeptes erkennbar. Somit kann an dieser Stelle festgehalten werden, dass unter Berücksichtigung des in der vorliegenden Arbeit festgelegten Untersuchungsgebietes, der Native-Inertia-Ansatz im Bereich Funktionalität und Leistungsumfang eine adäquate Alternative zu dem „Swift-Ansatz“ darstellt.

Aufgrund der implementierten Anforderungen konnte verifiziert werden, dass sich der Native-Inertia-Ansatz für die Umsetzung verschiedener Anwendungen mit unterschiedlichen Schwerpunkten eignet:

- **Navigations-App**

  Native-Inertia ermöglicht die Entwicklung einer Navigations-App zur Routenplanung. Eine Native-Inertia-App kann zudem zur Orientierung einen Kompass und eine Angabe zur aktuellen Geschwindigkeit anbieten.

- **Produktivitäts-App**

  Native-Inertia ermöglicht die Entwicklung von Produktivitäts-Apps zur Verwaltung von Notizen, Aufgaben oder Finanzen. Durch die Möglichkeit zur lokalen, appspezifischen Datenverwaltung bietet der Ansatz zusätzliche Sicherheit der eigenen Daten.

- **Fitness-App**

  Native-Inertia ermöglicht die Entwicklung einer Fitness-App zum Aufzeichnen von Strecken. Außerdem kann eine mit Native-Inertia entwickelte App die aktuelle Geschwindigkeit angeben, Trainingsleistungen lokal speichern und daraus Statistiken generieren.

- **Lifestyle-App**

  Der Native-Inertia-Ansatz ermöglicht die Entwicklung einer Lifestyle-App zur Erstellung von Rezepten in einer Kochbuch-Anwendung oder von Tagebucheinträgen in einer Reisetagebuch-Anwendung.

Des Weiteren lieferten die implementierten Anforderungen folgende Potenziale und Einnschränkungen:

Einschränkungen:

- Native UI-Komponenten sind nur durch zusätzlichen Implementierungsaufwand umsetzbar.
- Keine bidirektionale Kommunikation zwischen dem Web-Frontend und dem Swift-Backend möglich.
- Beschränkte Kapazität für eine zeitgleiche Datenübertragung zwischen Frontend und Backend.

Potenziale:

- Unbeschränkter Zugriff auf plattformspezifische Funktionalitäten und die Gerätehardware.
- Zahlreiche Möglichkeiten zur nativen App-Entwicklung durch die Abdeckung einer Vielzahl von Anwendungsgebieten.
- Flexible Integration webbasierter Technologien im Frontend.
- Umsetzung komplexer nativer iOS-Anwendungen ohne fortgeschrittene Swift-Kenntnisse.
- Flexible Integration nativer Swift-Komponenten im Backend einer Web-Anwendung.
- Möglichkeit für individuelles Design durch den Verzicht auf standardisierte UI-Komponenten aus der Apple-Komponentenbibliothek.
- Performante Kommunikation zwischen Web-Frontend und Swift-Backend.
- Entwicklung einer auf Echtzeitdaten basierenden Anwendung.

Anforderungen

Die Auswahl der Anforderungen ergab sich primär aus der Eingrenzung der Arbeit auf den Schwerpunkt Funktionalität. Durch die Auswahl der Anforderungen wurden Applikationen mit unterschiedlichen plattformspezifischen- und Hardware-Funktionalitäten entwickelt. Dadurch konnten vielfältige Einsatzmöglichkeiten des Native-Inertia-Ansatzes ermittelt werden. Außerdem konnten durch die Anforderungswahl folgende Aspekte bestätigt werden:

- Unkomplizierte Integration von Drittanbieterwerkzeugen
- Unbegrenzter Zugriff auf native APIs
- Unabhängigkeit von Frameworks
- Keine Limitierung durch Browser-Funktionalitäten
- Maximale Kontrolle im Swift-Backend
- Native-Inertia weist keine wahrnehmbare Performanceminderung auf Benutzerseite im Vergleich zu einer vollständig in Swift entwickelten Anwendung auf.

Aus diesem Blickwinkel lieferten die Anforderungen eine Vielzahl an Erkenntnissen, wodurch eine erfolgreiche Anforderungsanalyse resümiert werden kann. Durch die Implementierungsergebnisse der Anforderungen konnten Potenziale sowie Einschränkungen des Native-Inertia-Ansatzes ermittelt werden und somit zur Beantwortung der eingangs aufgestellten Fragestellung beitragen.

6. Ausblick

Funktionelle Erweiterungspotenziale

In Bezug auf die funktionellen Erweiterungsmöglichkeiten bietet sich ein weites Spektrum an potenziellen Funktionalitäten und Aspekten, die innerhalb der vorliegenden Arbeit aus Zeitgründen nicht weiter berücksichtigt werden konnten:

- **Face ID**

  Die Authentifizierung durch Face ID würde das Anwendungsfeld des Native-Inertia-Ansatzes um sicherheitsrelevante Aspekte erweitern.

- **Plattformspezifische Informationen**

  Eine weitere relevante Anforderung ist das Auslesen von plattformspezifischen Informationen, wie beispielsweise dem Akkustand, um darauf basierend das Styling oder Leistung einer Native-Inertia-App anzupassen.

- **Beschleunigungssensoren**

  Durch die Einbindung der Beschleunigungssensoren wäre es möglich, die Ausrichtung des Bildschirms zu ermitteln und somit das Erscheinungsbild der Applikation an die Ausrichtung beziehungsweise Orientierung des mobilen Gerätes anzupassen.

- **Gesten**

  Ein weiterer Aspekt, der in die Arbeit hätte aufgenommen werden können, ist die Gesten-Funktionalität. Gesten werden als wichtig angesehen, da diese den Nutzer:innen ermöglichen, spezielle Aktionen auf vertraute Weise auszuführen.

- **Nicht-funktionale Aspekte**

  Nicht-funktionale Aspekte, wie die Wirtschaftlichkeit oder die Entwicklerunterstützung, waren kein Bestandteil der Arbeit, sind jedoch ebenfalls relevante Indikatoren, um einen Ansatz zur nativen Applikationsentwicklung umfassend beurteilen zu können.

Dokumentation

Um den Native-Inertia-Ansatz für andere Entwickler:innen zugänglich zu machen, könnte ein GitHub-Repository erstellt werden, dass eine detaillierte Dokumentation, eine Schritt-für-Schritt-Anleitung zur Projekteinrichtung und ein Boilerplate mit der Grundstruktur des Ansatzes und beispielhaften Umsetzungen der analysierten Anforderungen enthält.

Weitere Vergleichsmöglichkeiten

Der Native-Inertia-Ansatz wurde mit dem Entwicklungssystem von Apple unter Verwendung von Swift als Programmiersprache evaluiert. Eine weitere Betrachtungsweise wäre ein Vergleich zwischen Native-Inertia und den anderen Ansätzen, die in der Marktanalyse betrachtet wurden, um einen umfassenden Marktvergleich und eine differenzierte Markteinordnung des Ansatzes zu ermöglichen.

Plattformübergreifende Entwicklungspotenziale

Eine mögliche Weiterentwicklung des Native-Inertia-Ansatzes wäre die Erweiterung um plattformübergreifende nativ-kompilierte App-Entwicklung unter Verwendung von Flutter als Ersatz für Swift im Backend.

Durch die Umsetzung der in diesem Abschnitt betrachteten Potenziale könnte der Native-Inertia-Ansatz weiterentwickelt und ausgereift werden. Dadurch kann der Native-Inertia-Ansatz eine umfangreich analysierte und adäquate Alternative zu dem „Swift-Ansatz“ darstellen, der von Apple selbst als die beste Methode für die iOS-App-Entwicklung beworben wird.

7. Literaturverzeichnis

[1] Apple Inc. (9.. Januar 2007). *Apple Newsroom.* Abgerufen am Juli 2023 von Apple erfindet mit dem iPhone das Mobiltelefon neu: https://www.apple.com/de/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/#:~:text=Das%20iPhone%20wird%20in%20den,mit%20PC%20oder%20Mac%20zusammenarbeiten.

[2] Apple Inc. (03. 06 2019). *Apple Newsroom.* Abgerufen am Juli 2023 von https://www.apple.com/de/newsroom/2019/06/apple-unveils-groundbreaking-new-technologies-for-app-development/

[3] Apple Inc. (2022). *Annual Report 2022.* Abgerufen am Juli 2023 von https://s2.q4cdn.com/470004039/files/doc_financials/2022/q4/_10-K-2022-(As-Filed).pdf

[4] Huy, N. P., & Thanh, D. v. (2012). Selecting the right mobile app paradigms. *2012 Fifth IEEE International Conference on Service-Oriented Computing and Applications (SOCA).* Taiwan: IEEE.

[5] Krusche , S., & Sommer, A. (2013). Evaluation of cross-platform frameworks for mobile applications. *Software Engineering 2013 - Workshopband.* Bonn: Gesellschaft für Informatik e.V.

[6] Lange, S., & Bender, R. (2007). Median oder Mittelwert? *Deutsche Medizinische Wochenschrift*, S. 132.

[7] Mohr, D. (29. 06 2023). *FAZ.* Abgerufen am Juli 2023 von Der Kurswahnsinn von Apple geht weiter: https://www.faz.net/aktuell/finanzen/apple-ist-3-billionen-dollar-wert-1-6-mal-mehr-wert-als-der-dax-18999146.html

[8] Rieger, C., & Majchrzak, T. (2016). Weighted Evaluation Framework for Cross-Platform App Development Approaches. *EuroSymposium on Systems Analysis and Design.* Springer.

[9] Sensor Tower Inc. (2022). *Sensor Tower.* Abgerufen am Juli 2023 von Mobile App Market Outlook 2022 — A Forecast of the Mobile App Market in 2022: https://go.sensortower.com/rs/351-RWH-315/images/mobile-app-market-outlook-2022.pdf