Wie wird Android in Zukunft entwickelt und für welche Technologien wird es verwendet werden?

Niclas Schneider
INF-B
6. Semester
Matrikelnr: 98378

***Zusammenfassung*—In der heutigen Hi-Tech Welt, in der man alle wichtigen Erledigungen unterwegs mit dem Smartphone erledigen kann, sind diese kaum mehr wegzudenken. Das Betriebssystem Android ist das weltweit meist Verbreitete und bietet Vorteile, wie die Open-Source Entwicklung und Flexibilität der Geräte, die es installieren können. Dadurch, dass Smartphones in immer mehr Bereichen PCs oder anderen Geräten bevorzugt werden, spielt die Entwicklung von Android-Applikationen eine immer größere Rolle. In dieser Arbeit werden die native Entwicklung und die dazugehörigen Programmiersprachen sowie Cross-Platform Ansätze und ihre Frameworks analysiert. Außerdem werden Technologien vorgestellt, die für die Zukunft von Android maßgeblich sind.**

***Index Terms*—Android, Entwicklung, nativ, Cross-Platform, Augmented Reality, Internet of Things, künstliche Intelligenz, Machine Learning**

I. Einleitung

Seit dem Release von Android im Oktober 2008 hat sich viel bei der Entwicklung des Google-Betriebssystems getan. Damals noch musste jede Anwendung in Java geschrieben und manuell mit der Android SDK zum laufen gebracht werden, heute gibt es dafür weitaus angenehmere Möglichkeiten. Android-Geräte weisen doppelt so hohe Verkaufszahlen wie die von seinem größten Konkurrent Apple auf. Außerdem ist Android ein Open-Source Betriebssystem und kann auf vielen verschiedenen Geräten installiert werden. Hierzu gehören Auto-Navis, Kameras oder sogar Kühlschränke. [1] [2]

Da Android für den Betrieb vieler digitaler Geräte sehr wichtig ist, sollen einige der beliebtesten Ansätze für dessen Entwicklung vorgestellt werden. Anhand verschiedener Kriterien soll analysiert werden, ob diese Entwicklungsansätze in der Zukunft relevant sind. Um einen Ausblick auf die Zukunft der Android-App Entwicklung zu geben, sollen im Anschluss Technologien vorgestellt werden, die dafür maßgeblich sind.

II. Native Entwicklung

Der klassische Ansatz, Anwendungen zu entwickeln, ist diese nativ für das Betriebssystem zu programmieren. Dafür wird im Falle von Android eine SDK (Software Development Kit) zur Verfügung gestellt, welche alle Pakete, Applikations-Frameworks und Klassenbibliotheken für die Entwicklung enthält. [3]

Integrierte Entwicklungsumgebungen, wie Eclipse oder das von Google bereitgestellte Android Studio, helfen bei der Erstellung eines neuen Projekts. Diese bieten eine bereits eingerichtete Basis-Applikation und helfen dem Entwickler bei der Strukturierung der Arbeit. Sobald ein Projekt erstellt ist, kann dieses mithilfe der Android SDK und der Entwicklungsumgebung kompiliert und gestartet werden. [4]

A. Programmiersprachen

Zu Release mussten Android-Anwendungen mit der Java-Programmiersprache ”Java Android Library” entwickelt werden. Sie hat die exakt gleiche Syntax wie Java. Das betrifft Operanden, Iterationen, die Zuweisung von Werten, den Umgang mit Werten, usw. Den Unterschied machen Android-spezifische Klassen und Pakete wie die Activity- und View-Klasse. [3]

Die Benutzeroberfläche in Android benutzt ähnliche UI-Komponenten und Konzepte wie Java-Oberflächen. Alle Komponenten der Benutzeroberfläche werden durch Views und ViewGroups gebaut, welche in eine hierarchische Struktur eingeordnet sind. Diese können entweder im nativen Java-Code oder in XML-Dateien definiert werden. [5]

Eine Programmiersprache, die immer mehr an Beliebtheit gewinnt, ist Kotlin. Die Entwicklung begann 2010 durch das Software-Unternehmen JetBrains, die unter anderem auch die Entwicklungsumgebung IntelliJ entwickelt haben. Auf dieser IDE basiert auch Android Studio. Im Februar 2016 wurde die erste stabile Version veröffentlicht und im Mai 2017 wurde Kotlin von Google in Android Studio eingeführt. [6] [7]

Kotlin läuft, wie mehrere hundert andere Programmiersprachen, auf der Java Virtual Machine (JVM), was die darin geschriebenen Anwendungen leicht portabel für andere Software und Hardware Plattformen macht. [6]

Laut Google ist Kotlin die meist genutzte Programmiersprache zur professionellen Android-Entwicklung [8]. 80% der 1000 meistgenutzten Apps wurden mit Kotlin programmiert, ebenso über 70 Google Anwendungen wie

Drive, Home und Maps. Ein Grund dafür ist, dass Kotlin wegen der Verwendung der JVM die Vorteile einer modernen Programmiersprache mitbringt, ohne neue Restriktionen wie beispielsweise Kompatibilität, Leistung oder Interoperabilität einzuführen. [9]

B. Performance

Im Rahmen einer Studie des "2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)" wurde eine Analyse, welche die beiden Programmiersprachen Java und Kotlin in Bezug auf die Anzahl der Zeilen Code, Interoperabilität, Zeit zum Kompilieren und APK Größe vergleicht, durchgeführt. [9]

Aus dieser Studie sind folgende Ergebnisse hervorgegangen [9]:

- Das Ökosystem für beide Programmiersprachen ist vollständig und die Interoperabilität zwischen Kotlin und Java ist einfach und nahtlos. Daher können beide verfügbare Entwicklungstools, Bibliotheken und Frameworks verwenden.
- Beim Testen eines Applikationsprojekts ist aufgefallen, dass Kotlin ca. 15% weniger Zeilen Code braucht. Allerdings besitzt die Java APK nur 77% der Größe von der Kotlin APK.
- Wenn man den Aufbau der Programmiersprachen an sich betrachtet, bringt die Nutzung von Kotlin mehr Vorteile als Nachteile mit. Außerdem ist Java-Code um einiges schwieriger zu lesen als der von Kotlin, was zu mehr Bugs oder Code-Smell führen kann.
- Bezieht man alle Ergebnisse mit ein, kommt man zu der Schlussfolgerung, dass Java benutzt werden sollte, wenn in der Entwicklung die Priorität auf der APK-Größe und Build-/Kompilationszeit liegt. Legt man allerdings mehr Wert auf einen prägnanten Code, weniger Bugs und schnellerer Entwicklungszeit, sollte Kotlin für die Entwicklung genutzt werden.

C. Zukunftssicherheit

Trotz des Aufstiegs der Cross-Platform-Ansätze wie Hybrid- und Web-Apps bleibt der native Ansatz für Android der mit der besten Performance. Ein 2016 aufgestelltes Experiment zum Vergleich von hybriden und nativen App-Paradigmen hat dies bestätigt. Wie in Tabelle I zu sehen ist, liegt die native App in allen getesteten Bereichen vorne. [10]

Tabelle I
PERFORMANCE HYBRID UND NATIV

Performance Parameter	***Hybrid***	***Nativ***
Installationszeit (Sekunden)	9,37	7,64
Startzeit (Sekunden)	1,58	1,11
CPU Auslastung (%)	11,76	5,54
Speicher Auslastung (MB)	101,66	58,77
Batterie Temperatur (°C)	45	37
Netzwerkfluss (KB)	330,56	323,89

Eine im "IEEE Transactions on Mobile Computing" vom Mai 2018 veröffentlichte Studie vergleicht die nativen und Web Apps von großen Serviceanbietern wie Google, Amazon oder Facebook in Bezug auf Performance. Auch aus dieser Studie geht hervor, dass die native App in den meisten getesteten Bereichen (ca. 69%) vorne liegt. [11]

Die aufgezeigten Studien und Experimente sollen nicht zeigen, dass die native Entwicklung schlichtweg besser ist. Sie sollen lediglich darauf aufmerksam machen, dass native Apps eine bessere Performance aufweisen als andere Ansätze. Diese besitzen jedoch viele andere Vorteile gegenüber dem nativen Ansatz, was folgend noch behandelt wird. Trotzdem ist anzunehmen, dass die native Entwicklung bei einer Priorität auf Performance immer relevant bleiben wird. [10]

III. CROSS-PLATFORM ENTWICKLUNG

In der heutigen Zeit sollen die meisten Programme und Applikationen nicht mehr exklusiv für eine Plattform, sondern auf so vielen Geräten wie möglich erhältlich sein. Will man diese Herausforderung mit dem nativen Ansatz bewältigen, investiert man viel Zeit darin, die Software für jedes Betriebssystem einzeln zu implementieren. Hinzu kommt noch, dass das Entwicklerteam gute Kenntnisse in jeder für die Betriebssysteme benötigten Programmiersprache haben muss. Das erhöht nicht nur die Entwicklungsdauer und -kosten, sondern auch den Anspruch an die Fähigkeiten des Entwicklerteams. [12]

Im Folgenden sollen einige Ansätze zur Cross-Platform-Entwicklung vorgestellt und auf die Kriterien Performance und Sicherheit untersucht werden.

A. Hybride Entwicklung

Mittlerweile gibt es für die Cross-Platform Problematik im Mobilapp-Bereich einige gute Lösungen. Die weit verbreiteten Hybrid-Apps setzen dabei auf die Verwendung von HTML5 zur Entwicklung der Software. Die Hybrid-Frameworks stellen eine Schnittstelle zwischen der HTML5-Anwendung und dem Betriebssystem dar. Damit wird sichergestellt, dass diese von möglichst vielen Entwicklern angewendet und die Applikation auf den gängigen Betriebssystemen verwendet werden kann. In Abbildung 1 sieht man den Aufbau einer solchen Architektur am Beispiel Cordova. Die Cordova Middleware ermöglicht es der Anwendung, auf HTML5 und mit dem darin implementierten UI und logischem Code, auf den verschiedenen Betriebssystemen zu laufen. [12] [10] [13]

In Abbildung 2 ist der unterschiedliche Distributions- und Updateprozess von nativen und hybriden Anwendungen im Vergleich dargestellt. In der Abbildung ist zu erkennen, dass man sich durch das Entwickeln mit dem hybriden Ansatz nicht nur die Entwicklungszeit selbst spart, sondern auch die Zeit für das Testen und Bereitstellen der Software. Dies trifft nicht

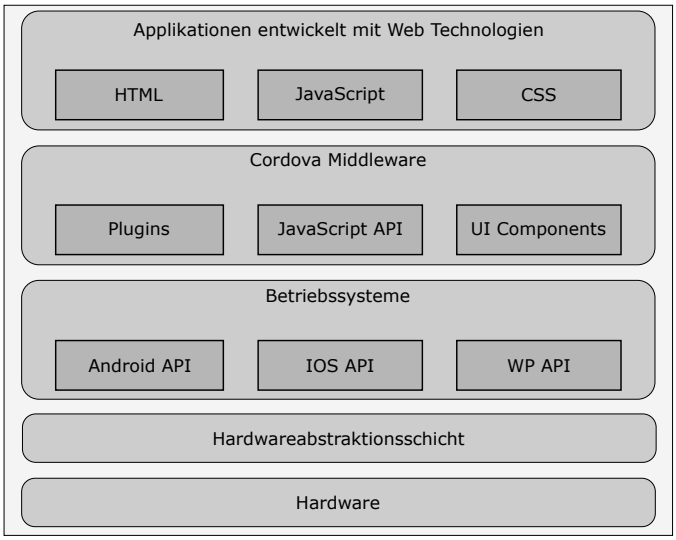

Abbildung 1. Architektur von Cordova

nur bis zum ersten Release, sondern auch für jedes Update zu.

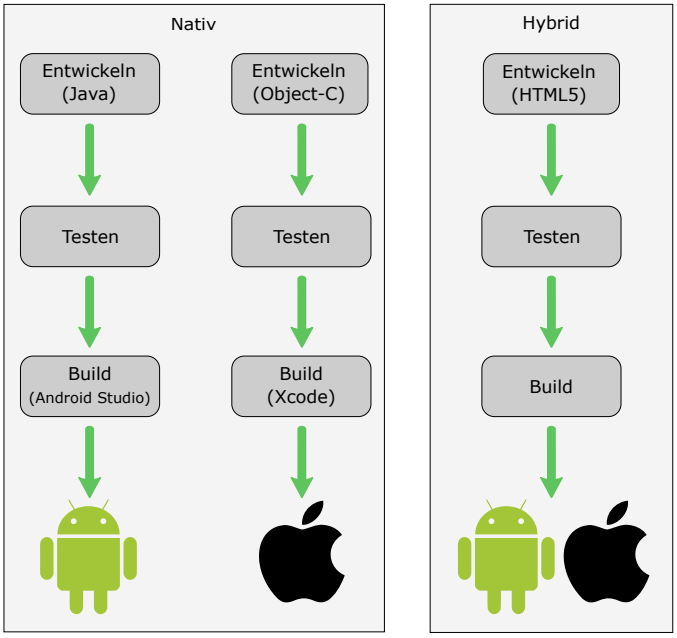

Abbildung 2. Vergleich Native vs Hybrid-Entwicklung

B. Cross-Compilation mit nativer API Einbindung

Ein weiterer Ansatz ist es den Code in einer Programmiersprache zu schreiben, welcher aber in Binärdateien für andere Plattformen zu kompiliert wird. Dafür werden native APIs eingebunden, damit diese auch in einer anderen Programmiersprache verwendet werden können. Ein Beispiel hierfür ist Xamarin, welches Entwicklern ermöglicht mobile Anwendungen in C# zu schreiben und diese dann in Binärdateien für andere Plattformen zu kompilieren. Durch die Einbindung nativer APIs weist diese Applikation dann nahezu identische Ergebnisse in Bezug auf Verhalten und Performance auf. [12] [14]

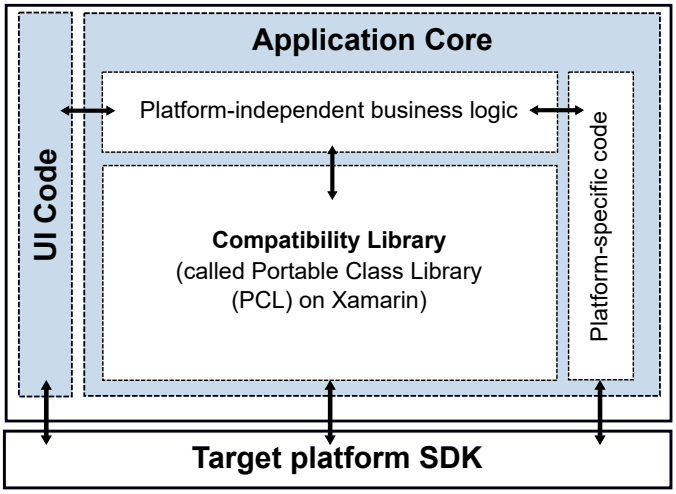

Abbildung 3. Struktur einer Xamarin Cross-Platform App

In Abbildung 3 ist die Struktur einer solchen Xamarin App abgebildet. Aus der Abbildung geht hervor, dass die Anwendung aus zwei Teilen besteht, dem Application Core und dem UI Code. Der UI Code beinhaltet native GUI Elemente für jede Plattform, wie beispielsweise Buttons und Widgets, die aber im plattform-unabhängigen Code alle auf einmal angesprochen werden können. Es gibt auch die Möglichkeit Code, der für die einzelnen Plattformen zu spezifisch ist, als das man ihn für diese unabhängig in C# schreiben könnte, nativ zu implementieren. Dieser muss dann gesondert in der Sprache und mit der Syntax für die jeweiligen Plattform programmiert werden. Das Ziel von Cross-Compilation, wie sie auch in Xamarin verwendet wird, ist es so wenig nativen Code wie möglich schreiben zu müssen. Um dieses Ziel zu erreichen werden sogenannte PCLs (Portable Class Libraries) verwendet. Diese können von Entwicklern in Visual Studio erstellt werden und bieten die Möglichkeit, Methoden und Eigenschaften im eigenen Code zu implementieren. Diese sind über mehrere Plattformen hinweg portabel sind. Dies wird bewerkstelligt, indem die Methoden und Eigenschaften zur Laufzeit den richtigen Plattform-Bibliotheken zugewiesen werden. [15] [14]

C. Interpretierte JavaScript Anwendungen

Die Mitte zwischen den beiden angehenden Ansätzen scheint das Interpretieren von JavaScript Code auf verschiedenen Plattformen zu sein. Hierbei benutzt man die JavaScript Engine der jeweiligen Plattform, um eine Bridge zu den nativen APIs zu erstellen. In Abbildung 4 sieht man am Beispiel React Native eine solche Architektur. Hieran ist zu erkennen, dass man mit React Code über JavaScript eine Bridge zu den verschiedenen Features der nativen Architektur aufbauen kann. Dabei kann, wie auch

in der nativen Entwicklung, auf alle Services sowie OEM Widgets zugegriffen werden. [4] [16] [17]

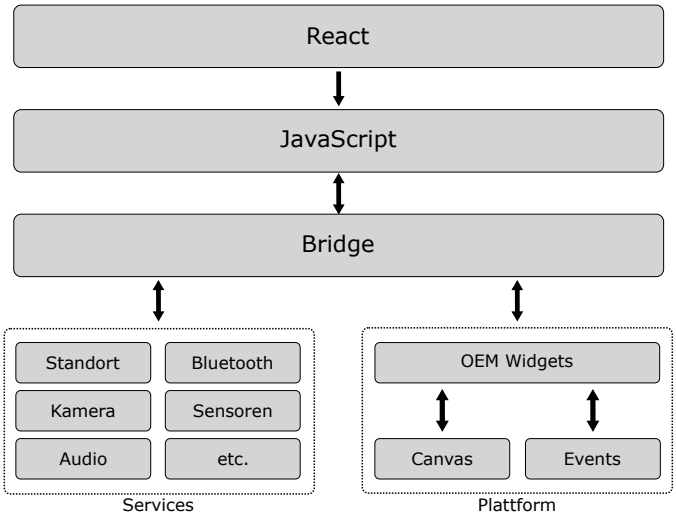

Abbildung 4. Architektur React Native

Der große Vorteil dieser Anwendungen ist, dass nur JavaScript-Kenntnisse benötigt werden, um sie entwickeln zu können. Im Vergleich zu anderen Cross-Plattform Ansätzen weisen JavaScript-Applikationen bessere Ergebnisse in Bezug auf Performance und Verhalten auf. Dies ist auf die nativen APIs, welche von der JavaScript Engine benutzt werden, zurückzuführen. [16]

D. Performance und Verhalten

In Tabelle II sind die vorgestellten Entwicklungsansätze aufgelistet und anhand von verschiedenen Kriterien bewertet. Aus dieser Tabelle geht hervor, dass die Cross-Platform Ansätze von der Performance und dem Verhalten der UI nicht besser sind als die native Entwicklung. Einige Ansätze kommen aber nahe ran. Der native Ansatz bringt allerdings höhere Entwicklungskosten und einen erhöhten Anspruch an die Entwickler mit. Die interpretierten und cross-compiled Apps sind beides gute Alternativen für die native Entwicklung und der Entwicklungsaufwand ist geringer. Hybrid Apps weisen eine niedrigere Qualität wie die vorangehenden Programmierparadigmen auf, sind allerdings in der Entwicklung billiger. Web Apps wurden in dieser Arbeit nicht behandelt, da sie nicht im App Store veröffentlicht werden können und auch nicht speziell für mobile Geräte entwickelt werden. Sie sind für die Vollständigkeit allerdings trotzdem in der Tabelle aufgelistet. [16] [10] [12]

E. Sicherheit

Smartphones sind mittlerweile nicht mehr nur ein Telefon oder ein Mittel zum Zeitvertreib, mit ihnen wird auch gearbeitet oder es werden private Gelegenheiten erledigt. Viele nutzen ihr mobiles Endgerät um Mails zu überprüfen, Einsicht in ihre Konten zu bekommen oder online einzukaufen. Da hier viele persönliche Daten eingegeben und verschickt werden, ist es wichtig, dass diese Apps sicher sind und keine Daten in falsche Hände geraten. [18]

Eine Studie, die in der "2019 International Conference on Cyber Security and Internet of Things (ICSIoT)" veröffentlicht wurde, vergleicht native und Cross-Platform Banking Apps auf ihre Datensicherheit. Die Studie ergibt, dass native Applikationen insgesamt weniger Zugriffsrechte anfordern, mit denen auf kritische Daten zugegriffen werden kann. Allerdings kann auch mit Cross-Platform Ansätzen eine gute Sicherheit gewährleistet werden. Hierbei muss auf die richtige Nutzung von APIs, sowie auf die korrekte Anfrage nach Zugriffsrechten, geachtet werden. Außerdem muss man stetig auf eine aktuelle targetSdk achten, um das Datenschutzniveau des Nutzers möglichst hoch zu halten. [19]

Wenn es um das Thema für die Entwicklung von JavaScript Apps geht müssen noch wichtige Anmerkungen beachtet werden. Angreifer haben zwei einfache Möglichkeiten Schwachstellen in JavaScript-Patterns auszunutzen [20]:

- *Schädlicher third-party Inhalt:* Schädlicher Inhalt kann in lokale HTML Dateien von third-parties, lokaler Malware, third-party Bibliotheken und Remote Webseiten eingebettet werden. Von dort können App Interfaces in einer Weise aufgerufen werden, in der es vom Entwickler nicht vorhergesehen war
- *Netzwerkattacke:* Befindet man sich in einem unsicheren Netzwerk (z.B. unsicherer Hotspot) und die Applikation fordert mittels HTTP eine WebView an, kann ein man-in-the-middle-Angreifer eine Seite zurückschicken, die schädlichen JavaScript-Code enthält.

IV. Schlussfolgerung zu Entwicklungansätzen

Insgesamt kann man sagen, dass es für jedes Entwicklungsbudget und für alle Anforderungen an die Qualität einer mobilen App einen zugehörigen Entwicklungsansatz gibt. In Abbildung 5 sollen diese Ansätze mit zugehörigem Entwicklungsaufwand und Softwarequalität graphisch eingeordnet werden. Dabei steht der Entwicklungsaufwand für die Kosten und Anforderungen an die Entwickler des Projekts, während die Softwarequalität

Tabelle II
Vergleich von Cross-Platform Entwicklungsansätzen

Entwicklungsansätze	***Nativ***	***Web Apps***	***Hybrid***	***Interpretiert***	***Cross-compiled***
UI/UX	Exzellent	Moderat	Moderat	Gut	Sehr Gut
Entwicklungskosten	Hoch	Niedrig	Niedrig	Moderat	Moderat
Performance	Exzellent	Schlecht	Moderat	Gut	Sehr Gut

für die Performance, UI/UX-Qualität und die vorraussichtliche Sicherheit der App steht.

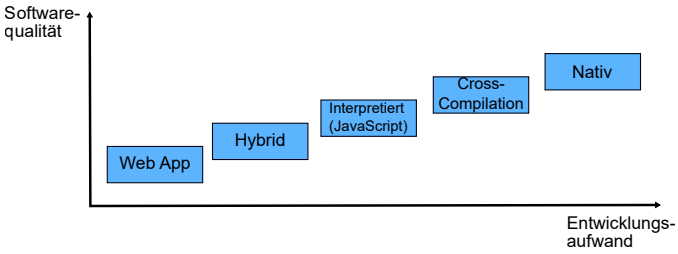

Abbildung 5. Aufwand und Softwarequalität von Entwicklungsansätzen

V. Technologien

Smartphones und deren Anwendungen wie M-Commerce, Social Media etc. werden immer mehr ein Teil des modernen Lebens. Wenn es um den Medienkonsum geht, ist das Smartphone in Deutschland mittlerweile gleichauf mit dem TV-Gerät. Beide wurden 2020 von 90% der Deutschen zum Konsum von Medien genutzt, wobei über die letzten sechs Jahre die Nutzung von TV-Geräten eher abgenommen hat während das Smartphone stark zugenommen hat. Fast 70% der Nutzer verwenden ihr Smartphone für Soziale Netzwerke oder Navigation und nahezu 50% für Banking oder Shopping. [21] [22]

Des weiteren gewinnen mobile Geräte für den kommerziellen Zweck immer mehr an Bedeutung. Über sie können z.B. Geschäftsprozesse gesteuert oder Termine verwaltet werden. In vielen Bereichen stellen Smartphones PCs in den Schatten, da sie von überall genutzt werden können und handlich zu bedienen sind. Android-Geräte werden auch für die Arbeit in bestimmten Branchen benutzt, beispielsweise als Scanner bei der Post. Aus dieser hohen Relevanz resultiert, dass Android auch für die Bedienung von Technologien der Zukunft sehr relevant sein wird. [18]

Im folgenden soll Androids Relevanz anhand von einigen Beispielen solcher Technologien erläutert werden.

A. Augmented Reality

In den letzten Jahrzehnten hat die Augmented-Reality-Technologie einen großen Fortschritt gemacht und ist jetzt bereit von PCs, Smartphones und weiteren Geräten benutzt zu werden. AR bringt virtuelle Dinge in die reale Welt. Im Bereich Bildung kann es beispielsweise dafür genutzt werden, den Schülern Inhalte auf ihren Geräten zu visualisieren. Dadurch können Lehrer ihren Unterricht lehrreicher und auch attraktiver gestalten. Dies kann als Schlüsselkomponente bezeichnet werden, welche eine von Soft- und Hardware unterstützte Lernatmosphäre ermöglicht. Hierfür könnte eine Anwendung das Scannen eines 2D Objekts, welches anschließend als 3D Objekt dargestellt wird, sein. Eine solcher Ablauf wird durch 6 visualisiert. [23]

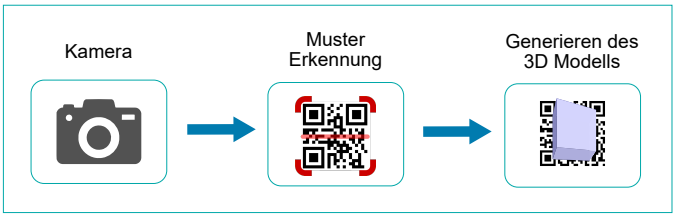

Abbildung 6. Diagramm zur Anzeige von 3D Modellen

B. Internet of Things

Das Konzept der Smart Homes wird für viele Menschen immer interessanter, da viele Geräte im Haus per Remote oder über Sensoren gesteuert werden können. Smartphones können den Vorteil bringen, dass sie diesen Prozess einfacher und günstiger gestalten. Es muss nicht mit einem speziellen Interface oder Gerät gearbeitet werden, stattdessen kann alles vom Smartphone aus erledigt werden. Ein Beispiel hierfür ist ein Sicherheitssystem mit Kamera am Haus. Klingelt jemand an der Tür und möchte eintreten, muss man nicht einen stationären PC benutzen, sondern kann das Kamerabild über eine Pop-Up Benachrichtigung direkt am Smartphone überprüfen und über die Anwendung weitere Schritte steuern. Ein weiterer Vorteil ist, dass man nicht für jedes Gerät eine eigene Fernbedienung braucht, sondern den Smartcontroller, mit dem diese verbunden sind, über das Handy steuern kann. [24] [25] [26]

C. Künstliche Intelligenz und Machine Learning

Bereits jetzt gibt es künstliche Intelligenzen auf dem Smartphone: Google Smart Assistant, Amazon Alexa, Apple Siri. Diese Assistenzen können helfen einem im Alltag aus, allerdings ist ihr Potential noch lange nicht ausgeschöpft. Features wie Chatbots, fließende Übersetzer oder eine Verbesserung des prädiktiven Texts werden in Zukunft wahrscheinlich ein fester Bestandteil sein. Außerdem kann eine Kombination aus KI und Machine Learning Entwicklern helfen, die Mensch-Maschine Kommunikation zu analysieren. Anhand dieser Daten können die Entwicklungsprozesse weiter verbessert werden. [27]

Eine Anwendung hierfür wäre ein persönlicher KI Assistent für sehbehinderte Menschen. Ein solcher Prototyp wurde 2018 entwickelt. Das Ziel der Entwickler war, eine KI zu entwickeln, welche über eine Bild- und Texterkennung sehbehinderte Menschen unterstützt. Der Assistent soll Objekte in der Umgebung erkennen und den Nutzer über diese informieren, sowie ihm Text auf Schildern oder ähnlichem vorlesen. Solch eine Software würde den Alltag von sehbehinderten Menschen komplett verändern und hilft ihnen diesen besser zu bewältigen. [28]

VI. Ausblick

Anhand des Überblicks, den diese Arbeit auf Methoden und Technologien für Android, gegeben hat lässt sich eine gute Konfiguration für die Entwicklung von Android-Apps

finden. Es kann allerdings nicht genau bestimmt werden, wo sich diese noch hinbewegt. In der Verbesserung für die verschiedenen Cross-Platform Ansätze kann jederzeit ein Durchbruch gelingen, der die anderen überbietet.

Die maßgeblichen Technologien für Android scheinen zwar schon in den Startlöchern zu stehen, scheinen aber nur träge Anwendung im Alltag zu finden. Zwar werden sie von immer mehr Menschen, Haushalten und Unternehmen benutzt, aber sind noch nicht zum Mainstream geworden und komplett ausgereift. Deswegen ist die Motivation alte Systeme abzulösen, neue Technologien einzusetzen und weiter auszuarbeiten noch gering. In Zukunft könnten diese allerdings schnell die Oberhand gewinnen.

VII. Conclusion

Abschließend lässt sich zusammenfassen, dass es für die Entwicklung von Android Apps viele gute Ansätze gibt. Die native Entwicklung wird alleine aus Performance- und Sicherheitsgründen in naher Zukunft sehr wahrscheinlich relevant bleiben. Dabei kann man sich, je nach Priorität, entweder für Java oder Kotlin entschieden. Bei den Cross-Platform Ansätzen hat man eine breite Palette an Auswahlmöglichkeiten. Generell kann man hier gut eine Lösung für die jeweiligen Anforderungen an die Software und das Entwicklungsbudget finden. Die Sicherheit gewinnt bei dieser Auswahl immer mehr an Priorität, da Smartphones mittlerweile oft bevorzugt werden, wenn es um Bank- oder Mailanwendungen geht. Diese enthalten oft private Daten, weswegen hier besonders auf die Sicherheit geachtet werden sollte. Android wird eine Plattform für viele maßgebliche Technologien der Zukunft, wie die in dieser Arbeit behandelten Themen AR, IoT oder KI, bieten und für deren Verwendung wichtig sein.

References

[1] Kantar. *Vergleich der Marktanteile von Android und iOS am Absatz von Smartphones in Deutschland von Januar 2012 bis März 2021*. 2021. URL: https://de.statista.com/statistik/daten/studie/256790/umfrage/marktanteile-von-android-und-ios-am-smartphone-absatz-in-deutschland/ (visited on 05/29/2021).

[2] Skye Hudson. *Not Just For Phones And Tablets: What Other Devices Run Android?* 2014. URL: https://www.makeuseof.com/tag/phones-tablets-devices-run-android/ (visited on 05/29/2021).

[3] Benny Skogberg. "Android Application Development". In: Malmö högskola/Centrum för teknikstudier, 2010, p. 11.

[4] William Danielsson. "React Native application development : A comparison between native Android and React Native". In: 2016, pp. 5–14.

[5] Seung-Ho Lim. "Experimental comparison of hybrid and native applications for mobile systems". In: *International Journal of Multimedia and Ubiquitous Engineering* 10.3 (2015), pp. 1–12.

[6] Daniela Gotseva, Yavor Tomov, and Petko Danov. "Comparative study Java vs Kotlin". In: *2019 27th National Conference with International Participation (TELECOM)*. 2019, pp. 86–89.

[7] Neha Verma, Sarita Kansal, and Huned Malvi. "Development of Native Mobile Application Using Android Studio for Cabs and Some Glimpse of Cross Platform Apps". In: *International Journal of Applied Engineering Research* 13.16 (2018), pp. 12527–12530.

[8] C. Haase. *"Google I/O 2019: Empowering developers to build the best experiences on Android + Play"*. 2019. URL: https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html (visited on 05/22/2021).

[9] Bambang Purnomosidi Dwi Putranto et al. "A Comparative Study of Java and Kotlin for Android Mobile Application Development". In: *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. 2020, pp. 383–388.

[10] Peixin Que, Xiao Guo, and Maokun Zhu. "A Comprehensive Comparison between Hybrid and Native App Paradigms". In: *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*. 2016, pp. 611–614.

[11] Yun Ma et al. "A Tale of Two Fashions: An Empirical Study on the Performance of Native Apps and Web Apps on Android". In: *IEEE Transactions on Mobile Computing* 17.5 (2018), pp. 990–1003.

[12] Aline Ebone, Yongshan Tan, and Xiaoping Jia. "A Performance Evaluation of Cross-Platform Mobile Application Development Approaches". In: *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. 2018, pp. 92–93.

[13] Shahrooz Pouryousef, Mariam Rezaiee, and Ata Chizari. "Let me Join Two Worlds! Analyzing the Integration of Web and Native Technologies in Hybrid Mobile Apps". In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2018, pp. 1814–1819.

[14] Dan Hermes. *Xamarin mobile application development: Cross-platform c# and xamarin. forms fundamentals*. Apress, 2015.

[15] Nader Boushehrinejadmoradi et al. "Testing Cross-Platform Mobile App Development Frameworks (T)".

In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2015, pp. 441–451.

[16] Kewal Shah, Harsh Sinha, and Payal Mishra. "Analysis of Cross-Platform Mobile App Development Tools". In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. 2019, pp. 1–7.

[17] Jun Kaneko. *"Understanding React Native Architecture"*. 2020. URL: https://dev.to/goodpic/understanding-react-native-architecture-22hh (visited on 05/28/2021).

[18] Aneet Shilpa George and Tanya Singh. "M-commerce: Evaluation of the technology drawbacks". In: *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*. 2015, pp. 1–6.

[19] Edward Danso Ansong and Thomas Quansah Synaepa-Addision. "A Comparative Study of User Data Security and Privacy in Native and Cross Platform Android Mobile Banking Applications". In: *2019 International Conference on Cyber Security and Internet of Things (ICSIoT)*. 2019, pp. 5–10.

[20] Wei Song, Qingqing Huang, and Jeff Huang. "Understanding JavaScript Vulnerabilities in Large Real-World Android Applications". In: *IEEE Transactions on Dependable and Secure Computing* 17.5 (2020), pp. 1063–1078.

[21] Bitkom. *Anteil der befragten Smartphone-Nutzer, die die folgenden Funktionen mit ihrem Smartphone nutzen. Statista.* 2017. URL: https://de.statista.com/statistik/daten/studie/166150/umfrage/nutzung-von-smartphone-funktionen-in-deutschland/ (visited on 05/29/2021).

[22] Seven.One Media GmbH (Ein Unternehmen der ProSiebenSat.1 Media SE). (2020). *Persönliche Gerätenutzung für den Medienkonsum in Deutschland in den Jahren 2014 bis 2020. Statista*. 2020. URL: https://de.statista.com/statistik/daten/studie/476467/umfrage/persoenliche-geraetenutzung-fuer-den-medienkonsum-in-deutschland/ (visited on 05/29/2021).

[23] Yogita Bahuguna, Aashish Verma, and Kunal Raj. "Smart learning based on augmented reality with android platform and its applicability". In: *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*. 2018, pp. 1–5.

[24] G. M. Madhu and C. Vyjayanthi. "Implementation of Cost Effective Smart Home Controller with Android Application Using Node MCU and Internet of Things (IOT)". In: *2018 2nd International Conference on Power, Energy and Environment: Towards Smart Technology (ICEPE)*. 2018, pp. 1–5.

[25] Gauri S. Girme and Sandip. R. Patil. "Internet of Things Based Intelligent Security using Android Application". In: *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2019, pp. 1101–1105.

[26] Kshirod Kumar Rout, Samuchita Mallick, and Sivkuinar Mishra. "Design and Implementation of an Internet of Things based Prototype for Smart Home Automation System". In: *2018 International Conference on Recent Innovations in Electrical, Electronics Communication Engineering (ICRIEECE)*. 2018, pp. 67–72.

[27] Audrey Throne. *5 Trends That Define Future of Android App Development*. 2020. URL: https://www.newgenapps.com/blog/5-trends-that-define-future-of-android-app-development/ (visited on 05/29/2021).

[28] Shubham Melvin Felix, Sumer Kumar, and A. Veeramuthu. "A Smart Personal AI Assistant for Visually Impaired People". In: *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*. 2018, pp. 1245–1250.