SPENGERGASSE

Diploma Thesis

title

Joachim GRÜNEIS, Klaus UNGER

Version 1.0 - 2018-06-18

Table of content

Colophon

Spengergasse Press, Vienna

Schuljahr 2018/19

Datum:	**übernommen von:**

Table 1. Abgabevermerk

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Wien, am 08.04.2019	**VerfasserInnen**
	Joachim GRÜNEIS
	Klaus UNGER

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Arbeit Weine

Bedürfnisse

In diesem Kapitel wird erarbeitet, dass es wichtig ist seine Anforderungen an die App zu kennen, bevor man sich für eine Sprache und ein Framework entscheidet. Wie in den folgenden Kapiteln nämlich herausgerarbeitet wird hat alles seine Vor- und Nachteile, daher ist es umso wichtiger zu wissen, welche Sachen man benötigt und welche Funktionen nicht gebraucht werden. Vor der Programmierung sollte daher eine Liste erstellt werden, die beinhaltet welche Funktionen/Möglichkeiten die Sprache und das Framework haben sollten. Ebenso ist es wichtig zu wissen, ob man Dinge wie Biometrische Daten (FaceID, FingerPrint, etc...) benötigt. Auch hier gibt es unterschiede beim Programmieren.

Die Arten der Apps

Native Apps

Unter nativen Apps versteht man, Mobile-Anwendungen, die speziell für ein bestimmtes Betriebssystem entwickelt werden Jede Plattform (iOS, Android) hat ihre eigenen Programmiersprachen. Während für den Apple-Stack Swift/Objective-C verwendet wird , so sieht es bei Android mit Java wieder ganz anders aus. Beim Windows Phone wiederum wird vor allem mit C++ und JavaScript gearbeitet. Bei Nativen Apps muss somit für jedes Betriebssystem die App in der jeweiligen Programmiersprache erarbeitet werden. Native Apps zeichnen sich auch dadurch aus, dass diese keine Internetverbindung brauchen, um funktioinsfähig zu sein.

Vorteile: Die Native Programmierung hat den Vorteil, dass diese zu 100% den Funktionalitäten der Geräte angepasst sind, daher können alle Features der Smartphones genutzt werden. Der Zugang zu den Hardware Eigenschaften (Kamera, GPS, etc...) ist bei der Native App ebenfalls sehr einfach.

Nachteile: Eine App pro Betriebssystem, das bedeutet möchte man iOS und Android abdecken, so müssen 2 Apps programmiert werden. Das kostet Zeit bei der Entwicklung, Zeit bei der Wartung und Schlussendlich jede Menge Geld.

Web Apps

Die Art der Web App Entwicklung ist darauf ausgerichtet, die App von jedem Gerät und Browser zugänglich zu machen. Die App wird hier im Gegensatz zur nativen Programmierung wird hier eine App für alle Betriebsysteme entwickelt. Die Web App macht sich HTML und CSS zu nutze und wird in einem Webbrowser durch eine URL ausgeführt. Nach dem Öffnen der App passt sich diese dann an das jeweilige Gerät an. Durch diese Umsetzung ist es ebenfalls nicht erforderlich die App auf dem Smartphone zu installieren. Deshalb sind diese meist auch nicht im App Store

vertreten. Auf Apple wäre dies via Safari oder anderen heruntergeladenen Browsern möglich. Auf Android wäre es zum Beispiel mit Google Chrome verwendbar.

Vorteile: Eine App für alle Geräte, das spart Zeit und Wartungsaufwand, sowie viel gespartes Geld am Ende. Es wird kein App Store für den Betrieb benötigt, man ist also nicht auf die Nutzungsbedingungen der App Stores beschränkt

Nachteile: Der größte Nachteil ist der schlechte bzw. fehlende Hardware Zugriff auf den Geräten. Ein ebenso nicht unwesentliches Kontra, ist dass die App nur mittels Internet funktioniert.

Fazit: Daher sollte vorab bedacht werden, wo und wie die App in der Verwendung sein wird.

Cross-Plattform (Hybride) Apps

Die Hybride Programmierung kombiniert die native und die Web App miteinander. Es wird hauptsächlich mit Web-Sprachen programmiert, dennoch wird der Hardware Zugriff nicht eingeschränkt. Cross-Platform Apps werden via App Store installiert.

Vorteile: Durch die Kombination hat man bei der Hybriden App ein besseres Erlebnis als bei den Web Apps

Nachteile: Die Unterstützung ist nicht zu 100% die Selbe, wie bei nativer Programmierung, da neue Funktionen des öftern etwas länger auf sich warten lassen. Einige Elmente müssen Seperat programmiert werden, da sowohl iOS als auch Android bestimme Vorgaben bei einigen Elementen haben, die mittels Cross-Platform bisher noch nicht gelöst werden konnten. Daher muss an manchen Stellen ebenso wie bei Nativen Apps doppelt programmiert werden. Dennoch hält sich der Aufwand meist in Grenzen.

Fazit: Aus der technischen Sicht, würde derzeit Cross-Platform am meisten Sinn machen, da man die vollen Funktionen der Geräte nutzen kann, ohne dass man eine App für jedes mobile Betriebsystem machen muss. Wenn man finanzielle Aspekte hinzuzieht, würden Web-Apps durchaus auch Sinn machen, da man somit die 30% Abgaben auf In-Game-Köufe nicht bezahlen muss, da die App nicht über den App Store vertrieben werden muss. Allerdings, kann man bei den WebApps oft nicht so gut oder teilweise erst später den vollen Funktionsumfang der Hardware Features der Smartphones nutzen.

Welche Sprachen verwendet werden

Swift

Swift ist eine neuere Alternative zu Objective-C von Apple. Zwar basiert Swift auf Objective-C, dennoch soll Swift um einiges unkomplizierter und komfortabler sein. Diese Programmiersprache ist vor allem für iOS, macOS, watchOS und tvOS Apps zu benutzen, da diese logischerweise darauf ausgelegt ist. Swift ist eine intuitive und gut lesbare Sprache im Bereich der Mobile-Applications

Java

Java ist eine Objektorientiere Programmiersprache. Java ist eine durchdachte und eher leichtgewichtige Sprache, die bewusst auf einige Funktionien, die C++ bietet, verzichtet. Die Syntax von Java ist an C und C++ angelehnt. Durch die Ähnlichkeit zu C und C++ fällt ein Umstieg auf Java leicht.

C++

C++ ist eine von der ISO genormte Erweiterung der Programmiersprache C

HTML 5

TypeScript

Frontend Kompaktlösungen

Appcelerator

Erklärung: Appcelerator ist ein Komplettpaket im Bereich der Mobilen Programmierung, denn es bietet die Möglichkeit eine App mit dem App-Designer zu bauen, sowie man ebenfalls ein Dashboard inkludiert hat, das mit einigen Statistiken zur App glänzen kann. Es wird außerdem die Möglichkeit geboten die App im Cross-Platform stil zu programmieren. Dies wird mithilfe von JavaScript umgesetzt. Für viele ebenso relevant ist der inkludierte API Builder, der sicherlich einiges an Zeit sparen kann. Einstellungen zu Push-Benachrichtigungen sind auch ein angepriesenes Feature.

Preis: Die Studio IDE und der API Builder sind gratis,. Für den App Designer und die API Calls sowie die App Preview muss man 99$ pro Monat bezahlen. Ebenso besteht die möglichkeit noch mehr zu kaufen, dies muss man sich allerdings selbst zusammenbauen und dementsprechend variiert der Preis. Hier besteht zum Beispiel die Möglichkeit noch eine Crash detection und Performance Analysen zu bekommen, sowie auch automatiertes Testen zu benutzen. Als Extra werden noch Cloud Kapazitäten geboten, die mit 15$ / Monat anfangen.

RhoMobile

Erklärung: Rhomobile Suite ist ein Software Stack für App-Entwickler, der unter anderem die Möglichkeit bietet mit Ruby zu programmieren, was den Focus auf die Einfachheit und Produktivität lenkt. Es wir auf Cross-Platform Entwicklung gesetzt und zusätzlich ist es auch möglich HTML/CSS/JS zu verwenden. Programmiert wird mittels RhoStudio Extension in Eclipse. Der Sinn von RhoMobile besteht laut Hersteller darin, dass Firmen sichere, aber dennoch dem Customer-Standart entsprechende Apps progammieren können.

Preis: Das Basis App Framework (Rhodes, RhoStudio, RhoElements) ist gratis. Gegen Bezahlung erhält man besseren Support sowie einige extra Features wie das Lesen von Barcodes oder automatische Daten verschlüsselung. Die Preise sind auf Anfrage.

MoSync

Erklärung: Ist ein gratis open-source Software Development Kit. Mit MoSync greift man ebenfalls auf C++, HTML5 und JavaScript zurück. MoSync ist ebenfalls mittels Eclipse verwendbar. Einer der Vorteile von MoSync ist, dass man sicher und schnell Files in der Cloud mit anderen Usern (sogar Personen die keinen MoSync-Account besitzen) teilen kann. Mittels der Platform ist es möglich, dass man überall und jederzeit daran Arbeiten kann. Ebenso soll die Datensicherung und Wiederherstellung sehr gut funktionieren.

Preis: Open Source SDK. Daher keine Kosten.

Sencha Ext JS

Erklärung: Ist eine Komplettlösung mit App-Baukasten der durch Drag and Drop einiges an Zeit beim Programmieren spart Ebenfalls ist es möglich mit Sencha Test zusätzlich zu Testen, hierbei geht es um Unit und End-To-End Tests. Es besteht die Option Statistiken und Heatmaps zu verwenden um Monitoring und Datenauswertung zu machen.

Preis: Ab 1800€ / Jahr für je einen Entwickler

Fazit: Die meisten oben genannten Lösungen sind kostenpflichtig, dafür bekommt man wirklich etwas geboten, das durchaus sehr viel Zeit und Ressourcen spart. Wenn man eine App schnell auf den Markt bringen will, so sind diese Lösungen sicherlich von Vorteil, da sie Arbeit abnehmen. Ebenso ist vermutlich auf lange Sicht auch eine Kostenreduktion bei den Mitarbeitern ein positiver wirtschaftlicher

Aspekt. Von der technischen Sicht, kriegt man einige Hilfestellungen, die vor allem den Erstellungs Prozess der App verkürzen, aber auch das Überwachen und Testen, sowie einige Analysen anbieten, was für kommerzielle Programmierung sicherlich einen starken Vorteil bringt.

Im Diplomprojekt wurde von so einer Lösung abgesehen, da es für zu teuer gewesen wäre und die Features bis auf die App-Baukasten und das automatisierte Testen, für das Projekt im aktuellen Stadium nicht relevant gewesen wären. Ebenso hätten es vermutlich zu viel Arbeitszeit gespart, da die App zu schnell Fertig geworden wäre.

Frontend Frameworks

Bei der Auswahl bei den Frameworks gibt es entscheidende Kritierien, die natürlich bei jeder App unterschiedlich sind. Daher ist eine allgemeine Aussage schwer zu treffen. Für diese Untersuchung gibt es folgende wichtige Kriterien. Alle Frontend Frameworks müssen schon etwas länger existieren und sollten auch in naher Zukunft nicht ohne Weiterführung und Support auskommen müssen. Aufgrund dieser zwei Punkte ist Ionic mit Vue aus der möglichen Auswahl rausgefallen, da sich dieses derzeit noch in einer Betaphase befindet.

Weitere Kriterien:

Übersetzung

Kann man in dem Framework eine Internationalization umsetzen? Im Jahr 2021 sollten Apps in mehreren Sprachen verfügbar sein. Auch hier wird unterschieden, wie einfach sich eine Internationalisierung umsetzen lässt.

Anpassbares Design (während der Runtime)

Wie leicht ist es Designs umzusetzen und vor allem lässt sich das Design während der Nutzung ändern.

Hardwarezugriff

Viele Apps benötigen Zugriff auf die Kamera, auf Biometrische Sensoren und auch auf andere mögliche Funktionen der Smartphones. Hier wird unterschieden, wie einfach das Framework solche Schnittstellen zulässt.

Support

Es ist wichtig, dass regelmäßige Updates erfolgen, um die App auch zukunftssicher

machen zu können. Regelmäßige Updates sind hierfür wichtig, allerdings ist der Abstand der Updates ebenfalls subjektiv zu werten, da für viele Entwickler zu häufige updates für mehraufwand sorgen können für andere dennoch kein Problem darstellen.

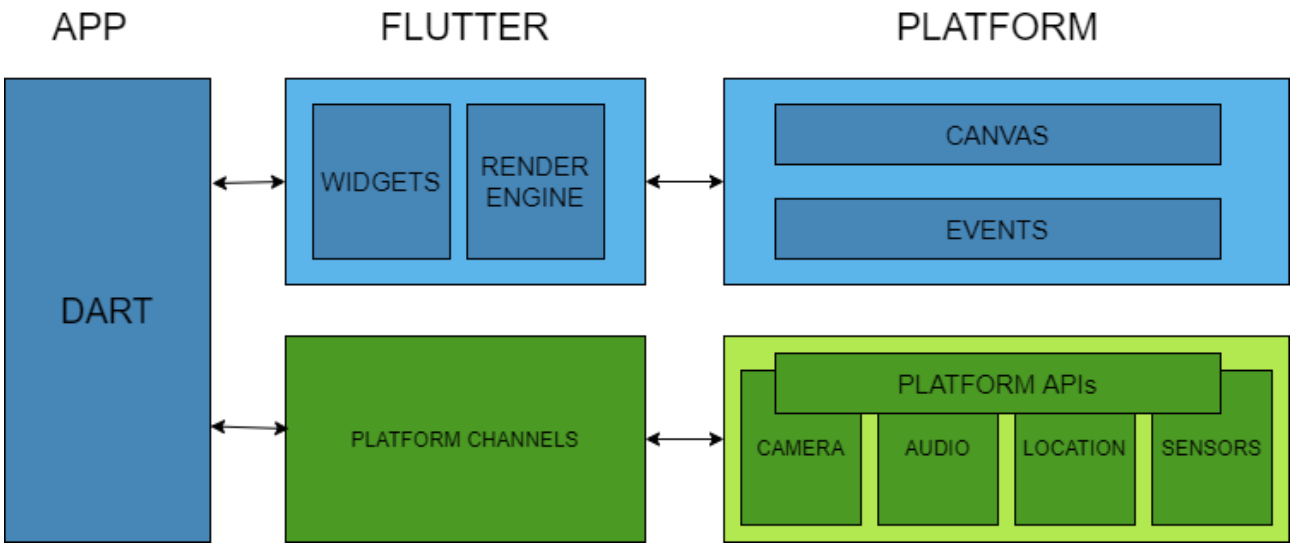
Dokumentation

Gibt es eine gute Dokumentation? Wie ausgereift und verständlich sind die Dokumentationen?

Flutter

Disclaimer: Da im Diplomprojekt mit Flutter gearbeitet wurde, ist in diesem Teil auch erworbenes Wissen eingeflossen, deshalb ist die Erklärung / Analyse genauer und auch teils detailierter.

Flutter ist ein Open-Source UI Entwicklungs-Kit. Die zugrundeliegende Programmiersprache ist Dart. Das Framework wird für die Programmierung von Apps verwendet. Das Framework selbst ist mittels C++ geschrieben worden.



Funktionsweise von Flutter [1]

Übersetzung

Die Übersetzung in Flutter ist relativ einfach vor allem sobald man diese aufgesetzt hat. In der laufenden Entwicklung hat man dann für jede Sprache, die man unterstützen will, ein JSON file in dem man die verschiedenen Elemente dann übersetzt. Im Code selbst, werden dann statt Strings einfach die Feld-Namen verwendet, die als Key für die Übersetzung fungieren.

Anpassbares Design

Flutter ermöglicht es während der Runtime die Designs zu verändern. Hier geht es vor allem um das Ändern der Farben während dem Benutzen der App. Ebenso können natürlich alle Widgets während der Runtime geändert werden, dazu muss man nicht viel machen, da dies mittels Navigator funktioniert.

Hardwarezugriff

Da Flutter sehr eng mit der Hardware kommuniziert, ist der Hardwarezugriff einfach. Für diese Use Cases gibt es bereits fertige Packages, die eingebaut werden können.

Support

Flutter versucht ungefährt jedes Quartal ein stable Update zu releasen. Erst im März 2021 kam Flutter 2.0 auf den Markt. Updates sind einfach mit dem Befehl "flutter upgrade" durchzuführen.

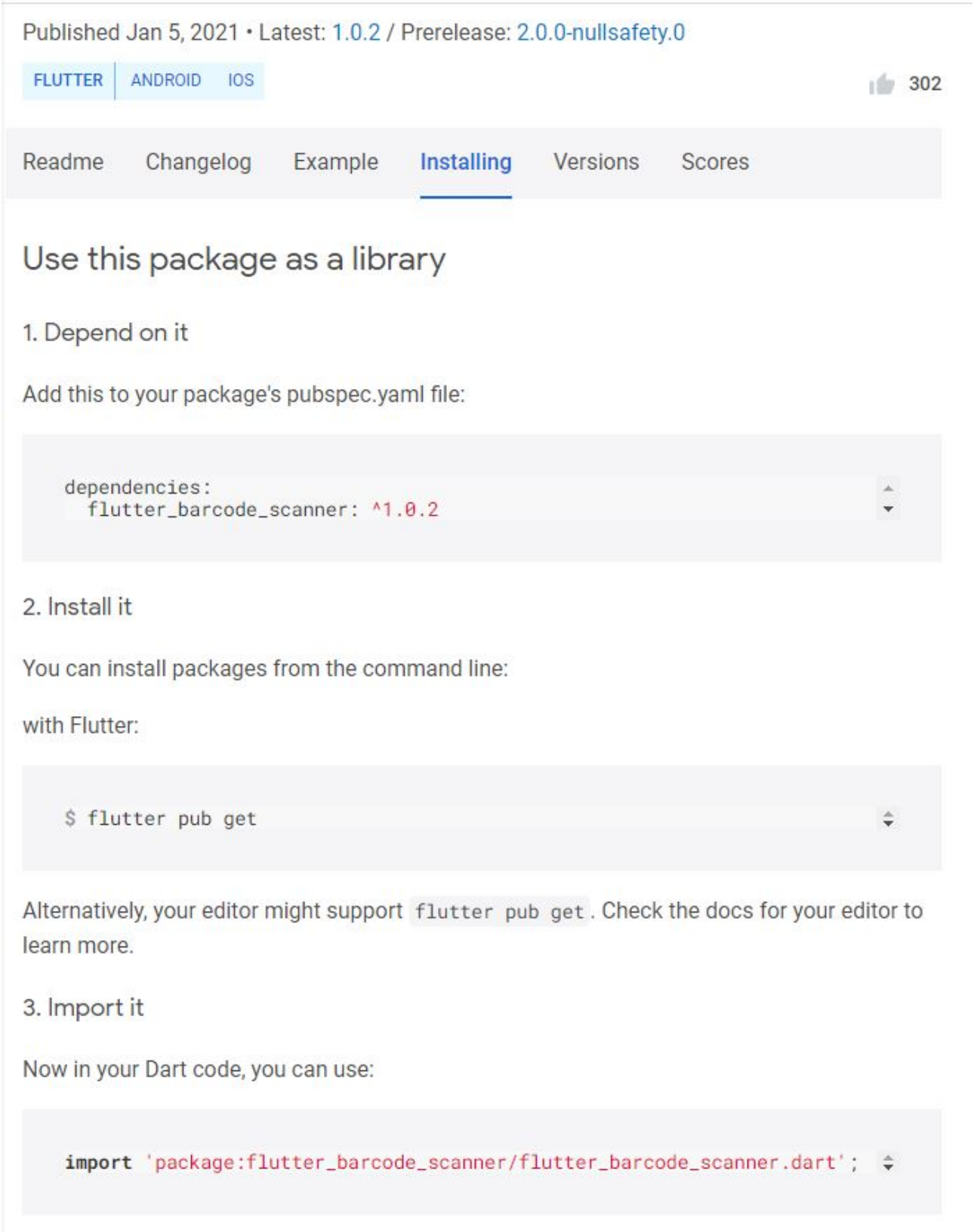
Dokumentation

Obwohl Flutter noch (im Vergleich zu Anderen) relativ "neu" ist, wird es sehr stark von Google unterstützt und es gibt eine durchaus beachtliche Dokumentation. Ebenfalls gibt es viele Kurzvideos zu bestimmten Widgets oder Funktionen, die

einem die Arbeit beim einlesen/einarbeiten erleichtern. Die Flutter Dokumentation ist vor allem sehr organisiert und einfach zu lesen.

Bonus

Für Flutter gibt es unzähloge fertige Packages, die einem das Leben als Entwickler erleichter, da man nicht alles von Grund auf neu machen muss. Für viele Use Cases gibt es bereits fertige Umsetzungen, die in die App eingebaut werden können. Ein Beispiel dafür wären Barcode Scanner. Hierfür ist es lediglich notwendig auf pub.dev danach zu suchen und eine dependency zu setzen. Dies ist, wie im unten stehenden Bild ersichtlich, alles detailiert unter dem Reiter "Installing" nachzulesen. Das verwenden von Packages ist simpel, die einzige Hürde ist es packages zu finden, die auch noch supported werden und laufend auf updates auch reagieren.



Gut zu wissen

In Flutter dreht sich alles um Widgets. Alles was in der App dann sichtbar ist, ist ein Widget. Ein Widget kann wiederum in ein anderen Widget gepackt werden. Was ist

nun also ein Widget? Es ist die Komponente, die Logik, Interaktion und Darstellung bündelt.

Zusammengefasst

Flutter kann Apps erstellen, die sehr gut aussehen und sehr starke Performancevorteile bieten. Im Gegensatz zu React Native, Ionic und anderen Webtechnologien ist es näher am Betriebssystem, was auch den Zugriff auf Gerätefunktionen wie die Kamera erleichtert. Die Community wächst und somit auch der Support unter den Entwicklern.

Code Snippets

Beispiel eines Home-Screens in Flutter

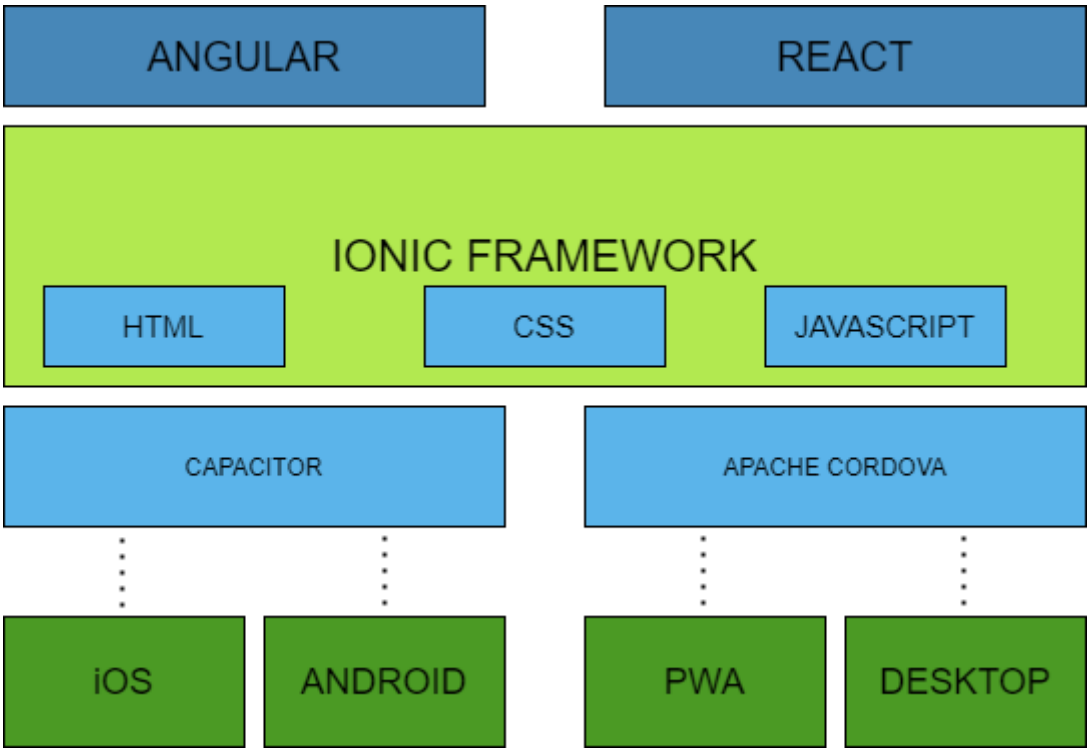
```
class _StartPageState extends State<StartPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('NoQuePOS'),
        centerTitle: true,
        actions: [
          IconButton(
            icon: Icon(
              Icons.shopping_cart,
              color: Colors.white,
            ),
            onPressed: () {
              _navigateToCartPage(context);
            },
          )
        ],
      ),
      bottomNavigationBar: NavigationBar(),
      drawer: DrawerMenu(),
      body: Container(
        decoration: BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/images/start.png'),
            fit: BoxFit.cover,
          ),
        ),
        child: Align(
          alignment: Alignment.topCenter,
          child: RaisedButton(
            onPressed: () {
              _navigateToScanPage(context);
            },
            color: Colors.blue,
            child: Text(
              translate('start_page.enter_button'),
              style: TextStyle(
                fontSize: 20,
                color: Colors.white,
              ),
            ),
          ),
        ),
      ),
    );
  }
```

Beispiel einer Appbar in Flutter

```
Scaffold(
  appBar: AppBar(
    iconTheme: IconThemeData(
      color: Color(0xffFF0000), //OR Colors.red or whatever you want
    ),
    title: Text("Title"),
    backgroundColor: Color(0xffFAFAFA),
  ),
)
```

Ionic/Angular

Ist ein Open-Source Webframework, dass vor allem für Cross-Platform und Progressive Webs Apps geeignet ist. Ionic mit Angular basiert, wie Angular auf TypeScript.



Architektur Ionic [2]

Übersetzung

Die Übersetzung ist mittels rxweb Package möglich, allerdings ein wenig umständlicher in der Handhabung, als andere Frameworks. Dennoch gibt es für die

Internatinalisierung bei Angular eine gute Dokumentation, die eine Step-by-Step Anleitung bereitstellt.

Anpassbares Design

Das Anpassen von Designs während der Runtime ist prinzipiell möglich, aber im Vergleich zu anderen Frameowrks eher unhandlich.

Hardwarezugriff

Der Hardwarezugriff bei Angular ist sehr gut und auch schon ausgereift. Im Ionic Framework gibt es das cordova-plugin-camera Plugin, welches die Schnittstelle zur Kamera bereitstellt.

Support

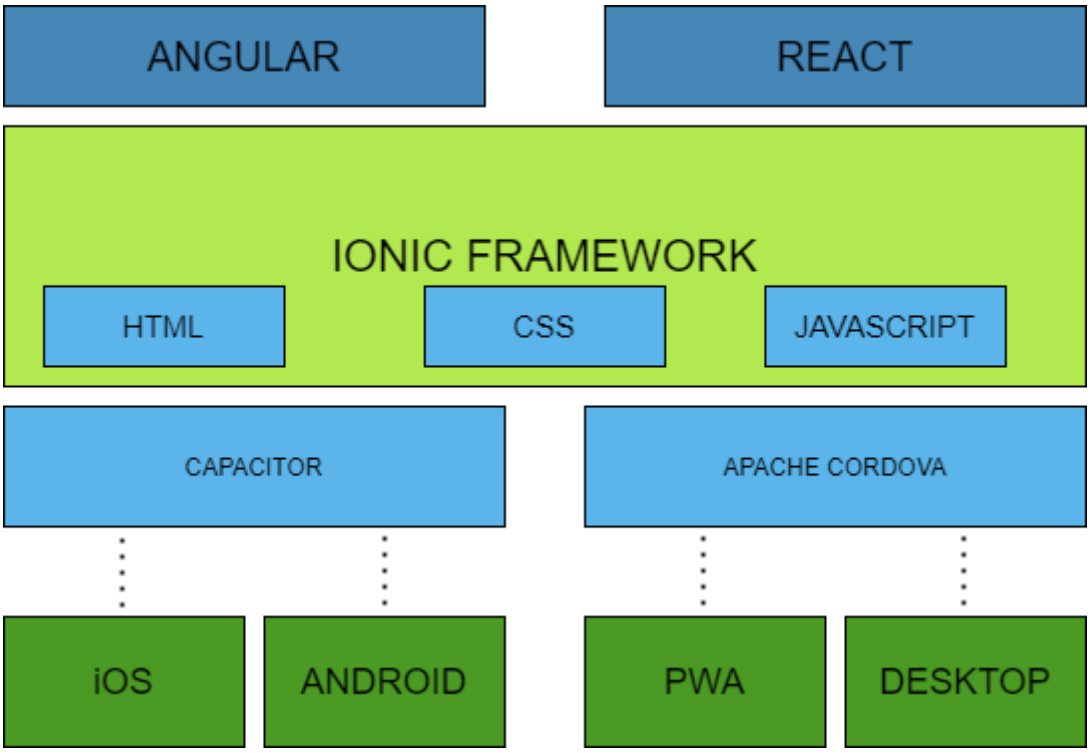
Major Releases werden alle sechs Monate veröffentlicht. Daneben gibt es noch Minor Releases, die sich mit API changes befassen, die keinen großen Eingriff vornehmen. Diese werden ungefähr ein Mal pro Monat released.

Dokumentation

Ionic hat eine übersichtliche und auch weitreichende Dokumentation, die ebenfalle jedes mal nach Major updates auch angepasst wird und somit auch die User Experience weiter verbessert wird.

Ionic/React

Ist ein Open-Source UI und Native API Projekt, dass vor allem für Cross-Platform und Progressive Webs Apps geeignet ist. Ionic mit React basiert, wie React auf JavaScript.



Architektur Ionic [3]

Übersetzung

Die Übersetzung in Ionic/React ist relativ einfach vor allem sobald man diese aufgesetzt hat. In der laufenden Entwicklung hat man dann für jede Sprache, die man unterstützen will, ein JSON file in dem man die verschiedenen Elemente dann übersetzt. Im Code selbst, werden dann statt Strings einfach die Feld-Namen verwendet, die als Key für die Übersetzung fungieren. Als zusatz ist es ebenso möglich, direkt mit dem String zu Arbeiten. Im JSON File wird also kein Key verwendet, sonder direkt der Text und mittels ":" dann die Übersetzung dahinter gemacht. Ermöglicht wird dies durch i18next.

Anpassbares Design

Ionic/React ermöglicht es während der Runtime die Designs zu verändern. Hierfür kann CSS oder ein bereits vorhandenes Theme Switcher Package verwendet werden. Durch die gute Dokumentation, stellt auch das Ändern des Designs während dem Benutzer der App kein Problem dar.

Hardwarezugriff

Im vergleich zu Flutter, fällt hier der Kamera Zugriff etwas schwerer aus, dennoch ist mittels Ionic/React der Hardwarezugriff generell auch relativ einfach möglich.

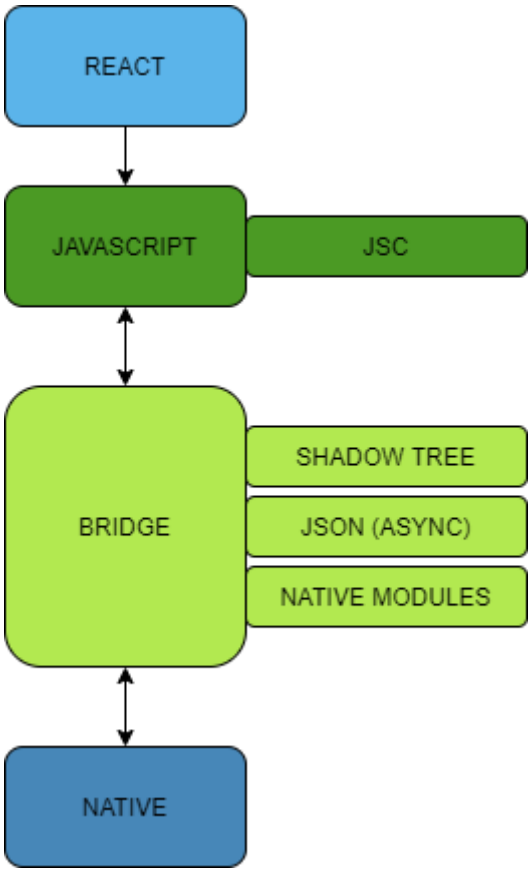
Support

Major Releases werden alle sechs Monate veröffentlicht. Daneben gibt es noch Minor Releases, die sich mit API changes befassen, die keinen großen Eingriff vornehmen. Diese werden ungefähr ein Mal pro Monat released.

Dokumentation

Ionic hat eine übersichtliche und auch weitreichende Dokumentation, die ebenfalle jedes mal nach Major updates auch angepasst wird und somit auch die User Experience weiter verbessert wird.

React / Native



Architektur React Native [4]

Übersetzung

Die Übersetzung in React Native ist relativ einfach vor allem sobald man diese aufgesetzt hat. In der laufenden Entwicklung hat man dann für jede Sprache, die man unterstützen will, ein JSON file in dem man die verschiedenen Elemente dann übersetzt. Im Code selbst, werden dann statt Strings einfach die Feld-Namen verwendet, die als Key für die Übersetzung fungieren. Als zusatz ist es ebenso möglich, direkt mit dem String zu Arbeiten. Im JSON File wird also kein Key verwendet, sonder direkt der Text und mittels ":" dann die Übersetzung dahinter gemacht. Ermöglicht wird dies durch i18next.

Anpassbares Design

React Native ermöglicht es während der Runtime die Designs zu verändern. Hierfür kann CSS oder ein bereits vorhandenes Theme Switcher Package verwendet

werden. Durch die gute Dokumentation, stellt auch das Ändern des Designs während dem Benutzer der App kein Problem dar.

Hardwarezugriff

Dadurch, dass es Native Framework ist, fällt die Einbindung der Hardware relativ einfach und ist a

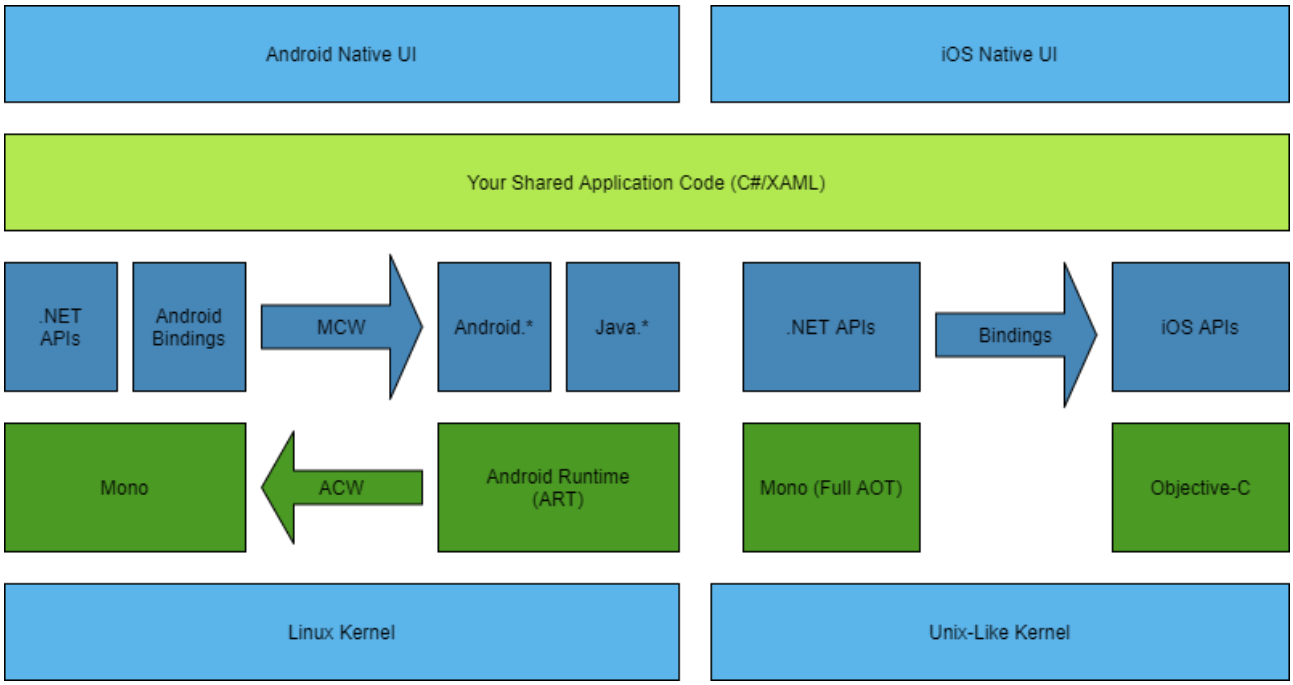
Support

Es werden regelmäßig jedes Monat neue Updates released. Die Updates werden über das GitHub Repository ausgerollt. Bevor das update eingebaut wird, gibt es eine Testphase für 1 Monat, wo reviewt werden kann. Ebenso können sich die Entwickler in der Phase mit den Änderungen vertraut machen.

Dokumentation

Es gibt eine generelle Dokumentation, dennoch ist diese bei weitem nicht so ausgereift, wie bei anderen Frameworks

Xamarin



Architektur Xamarin [5] https://docs.microsoft.com/en-us/xamarin/xamarin-forms/release-notes/

Übersetzung

Die Übersetzung in Xamarin erfolgft mittels RESX Files. Im Vergleich zu Flutter und React ist, hier allerdings etwas mehr Aufwand zu betreiben, dennoch gibt es dazu eine detailierte Dokumentation auf docs.microsoft.com. Ebenfalls muss man iOS und Android etwas seperat behandeln, da es nicht einheitlich ist und man somit mehr Aufwand erfordert.

Anpassbares Design

Es ist möglich die Design während der Runtime zu ändern

Hardwarezugriff

Der Hardwarezugriff ist etwas schwieriger, da man für iOS und Android jeweils 2 unterschiedliche Codebases benötigt. Es ist grundseätztlich möglich sowohl auf Apple Geräten, als auch bei Android auf die Hardware zuzugreifen, dennoch ist es mit mehr Aufwand verbunden, als bei anderen Frameworks. Hierfür gibt es

Libraries aus Xamarin.Essentials.

Support

Xamarin hat regelmäßige Updates, die ebenfalls auch immer in der Roadmap angepriesen werden, man kann sich also schon vorab darauf einstellen, was in der Zukunft auf einen zukommt. Ebenfalls steht auch dabei, wie lange diese Version supported wird.

Dokumentation

Xamarin hat eine vollständige Dokumentation samt intuitiver Navigation, die sehr weitreichend ist. Vom Anfänger bis zum Profi ist alles dabei.

Code Snippets:

```
private FahrzeugContext GetContext()
        {
            var db = new FahrzeugContext();
            db.Database.EnsureDeleted();
            db.Database.EnsureCreated();
            db.Import("data.sql");
            return db;
        }
```

===

[1] medium.com:Cross-platform mobile apps development in 2021: Xamarin vs React Native vs Flutter vs Kotlin Multiplatform, https://medium.com/xorum-io/cross-platform-mobile-apps-development-in-2021-xamarin-vs-react-native-vs-flutter-vs-kotlin-ca8ea1f5a3e0 abgerufen am 06.04.2021

[2] ICT-BZ.ch:Ionic Architektur, https://m335.ict-bz.ch/tag-1/ionic-architektur abgerufen am 06.04.2021

[3] ICT-BZ.ch:Ionic Architektur, https://m335.ict-bz.ch/tag-1/ionic-architektur abgerufen am 06.04.2021

[4] formidable.com : The New React Native Architecture Explained: Part Four, https://formidable.com/blog/2019/lean-core-part-4/ abgerufen am 06.04.2021

[5] docs.microsoft.com : What is Xamarin? , https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin#how-xamarin-works abgerufen am 06.04.2021

Auswertung / Vergleich

Für jedes Kriterium, das für die Bewertung der Frameworks heranzgeogen wurde, können maximal 10 Punkte erreicht werden.

	Übersetzung	Anpassbares Design	Hardwarezugriff	Support	Dokumentation
Flutter	8	8	8	8	8
Ionic Angular	9	5	8	8	8
Ionic React	9	8	7	8	8
React Native	9	8	8	8	6
Xamarin	6	7	5	9	9

Table 2. Auswertungs Tabelle

Wie man in der Tabelle oben sehen kann, liegen die Frameworks alle sehr nah bei einander. Durch diese Tabelle wird die Annahme am Beginn der Arbeit nochmals deutlich. Das perfekte Framework ist immer abhängig von den Anforderungen. Allgemein kann man sagen, dass fast alle sehr gut Dokumentiert sind und ebenfalls laufen Updates bekommen. Für das Diplomprojekt wurde Flutter verwendet, da das Team etwas komplett neues Lernen wollte und Flutter auch sehr ansprechend ist.

Schlussfolgerung der Arbeit

Es gibt zahlreiche Möglichkeiten im Frontend Bereich eine App zu entwickeln. Von Kompaktlösungen, die Geld kosten, bis zu Open Source Frameworks ist alles enthalten. Wichtig ist, dass vorab Kritierien festgelegt werden, die das jeweilige Framework erfüllen muss, um die Applikation umzusetzen. "Das Framework" gibt es hierbei nicht, denn jeder hat andere Anforderungen und Angewohnheiten, die eine Auswahl am Ende dann festlegen, denn die Frameworks sind im Großen und Ganzen alle sehr gut. Flutter hebt sich mit seiner besonderen Art dennoch ein wenig hervor, da es mit den Widgets eine doch sehr Bildhafte und einfache Programmierung ist und dadurch auch relativ schnell zu lernen ist. Die Syntax Highlighter vereinfachen die Lesbarkeit und auch die Fehlerbehebung sehr. Ebenfalls scheint die Zukunft von Flutter als relativ sicher.

Glossary

software

: invisible

hardware

: accessible

References

- [pp] Andy Hunt & Dave Thomas. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley. 1999.

Index

Appendix A: Appendix

A.1. The schema behind the data

```
{
  "bla": "bla"
}
```

```
public class HelloWorld {

  public static void main(String[] args) {
  }
}
```

```
// A function
void printInteger(int aNumber) {
  print('The number is $aNumber.'); // Print to console.
}

// The point where the application starts executing
void main() {
  var number = 10; // Declaration and initialization of  a variable.
  printInteger(number); // Call a function.
}
```