Current Progress in Cross-Platform Application Development

Evaluation of Frameworks for Mobile Application Development

Jan Christoph, Daniel Rösch, Thomas Schuster, Lukas Waidelich
Pforzheim University
Tiefenbronner Straße 65, Germany
{jan.christoph | daniel.roesch | thomas.schuster | lukas.waidelich}@hs-pforzheim.de

Abstract — **Cross-platform development is increasingly driven by web frameworks. Modern frameworks typically support application deployment for different platforms as well as the creation of progressive web apps. This approach is also driven by the increasing number of different device types and platforms. Development efforts can be significantly reduced by utilization of modern frameworks. Hence, several modern frameworks that have proven to be suitable for cross-platform development will be compared in this article. This article will extend our previous research on cross-platform development by several dimensions: at first, research on literature and technology developments regarding cross-platform development is extended. Secondly, we added further frameworks into our analysis. Thirdly, the evaluation approach is systematically extended to discuss each framework on an individual basis. This is driven by a reference architecture and implementation. To create a sound and objective framework comparison, the reference architecture is utilized to implement applications by means of each framework. Subsequently tests for different mobile devices and platforms are defined. All frameworks are compared according several key metrics. Finally, we describe current strengths and weaknesses of all approaches before giving an outlook on future steps of research.**

Keywords — cross-platform development; web component; web application framework; progressive web app.

I. INTRODUCTION

This article is an extended version of a former conference publication, see [1] for further details. Mobile devices have become an important platform for today's software applications. Especially, the utilization of smartphones increased rapidly within the last couple of years [2][3]. Since smartphones are often utilized to consume or orchestrate services, this process includes a vast range of applications. Smartphones also connect to other domains such as the Internet of Things (IoT) and often utilize smart cloud-based services [4].

The introduction of smartphones rapidly increased the need and development of mobile software. The development of mobile software applications is a special case of software engineering. Mobile applications are often also referred to as apps, which implies that the application is intended to be used on a smartphone or wearable device [5]. Thus, development must cope with specific aspects such as: short application lifecycles, limited device capabilities, mobility of users and devices, availability of network infrastructure as well as security and privacy issues [6]. The difference in devices also generates a variety of different resolutions and display sizes.

While developers are enacted to create and distribute applications in a large scale, they also have to deal with these inherent differences and limitations of mobile devices (i.e., battery life or small displays). Furthermore, it is necessary to address different operating systems (especially for smartphones, and, to a limited extent, for feature phones as well). Since the market for smartphones has consolidated recently, some operating systems (i.e., Windows Phone, BlackberryOS and other OS hold a market share of 3.2%) vanished again. Still, to address the smartphone market, applications for both, Android (market share: 72.4%) and iOS (market share: 24.4%) need to be provided [7]. In addition, Android is split into different versions, manufacturers and various system customizations. Despite vendor customization and just considering the Android version, current most widely used is Oreo (8.0 and 8.1 with 28.3%), followed by Nougat (7.0 and 7.1 with 19.2%) and finally, the latest version Pie (9; with 10.4%) [8].

In order to reach as many users as possible, applications need to support all major device platforms and versions of operating systems [6][9]. This introduces the need to either develop platform specific or platform agnostic applications. Platform specific implementations (native apps) literally require almost as many application implementations as platforms that are intended to be addressed. Therefore, this approach generates correspondingly high development expenditures. On the other hand, with a more generic approach, a single application or some core components could serve as the basis for multiple platforms. Besides reduced developments efforts, a generic approach also strengthens reuse of code and components.

Currently, generic approaches can be further subdivided into Web and hybrid applications (see Figure 1). Web applications can be used virtually under any platform, as a Web browser is preinstalled on almost all devices. The most salient advantage is application portability, which basically comes at no cost. Web apps are typically optimized by means of Hyper Text Markup Language (HTML5), Cascading Style Sheet (CSS) and JavaScript [10]. Numerous frameworks (such as Angular, Bootstrap, React or Vue) provide additional functionality on top of Web standard technologies and help to speed up development of Web apps.

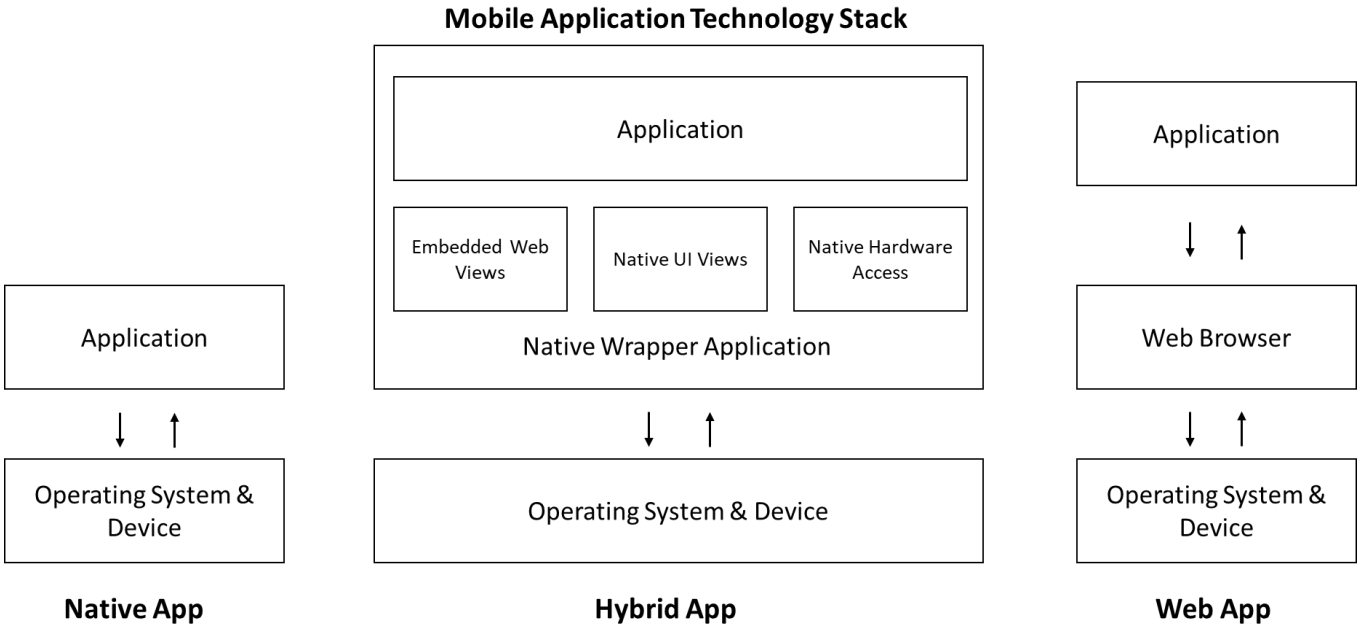

Figure 1: Mobile App Technology Stack.

Major disadvantages of Web applications are that they do not possess platform specific look and feel and often are restricted in functionality – especially access to system functions and device sensors. Furthermore, they must be interpreted and suffer performance losses compared to native applications [11].

Hybrid applications are built on frameworks such as Apache Cordova or Adobe PhoneGap. Often they rely on Web technologies also, and enact access to native device functions and sensors [6]. Hybrid apps utilize a specialized browser to present the user interface (UI). This results in a presentation layer, which is identical or very near to widgets used in native apps. Today's hybrid framework technologies are mainly extensions of Cordova and PhoneGap, as they extend and simplify the development of cross-platform applications. Therefore, the frameworks Cordova and PhoneGap are not included in the evaluation. While hybrid apps overcome some issues of Web apps (such as access to system functions and sensors), they still experience a loss of performance compared to native applications. However, it is notable that performance of hybrid apps has improved a lot with latest developments [6][10]. Comparing the short development lifecycles of devices and operating systems on the one hand to that of hybrid app frameworks on the other, it is noticeable that the latest developments are implemented with delays by the frameworks. As a result, access to new functionalities can be gained earlier when development is based on native apps.

Issues of supported functionality, performance and the generic question of maintenance of cross-platform applications lead us to the evaluation of multiple cross-platform frameworks. With this paper we want to record the current state of the art in the development of cross-platform apps. In addition, we want to uncover innovations and differences that arise with the deployment of new frameworks and versions. We achieve this with our reference app. The remaining parts of this article are structured as follows: First, Section II introduces the literature research method. This work can be seen as a starting point for further work on the article. Section III provides an overview of current mobile app development. In Section IV, a reference architecture is presented and five framework-based implementations of this architecture are discussed. The reference implementations are being evaluated in Section V. Finally, Section VI outlines the conclusion and outlook for further research.

II. LITERATURE RESEARCH

A comprehensive literature research was initiated to identify significant literature on the topic of cross-platform development and cross-platform frameworks. In addition, important scientific articles from the same field of research should be identified, which will form the basis for the later main part of the article. For this purpose, the proven literature research of Brocke et al. [12] was used. A five-phase model is defined by Brocke et al., which enables a systematic literature search. In the process, they combine approaches from Cooper [13] and Webster & Watson [14] . The method, the five-phase process is shown in Figure 2.

A. Definition of review scope

Phase I defines the different characteristic values of literature research. We apply the taxonomy of Cooper [13] in this phase as a basis. The applied taxonomy according to Cooper is explained below and also shown in TABLE I. The focus of

this article is on research outcomes in the area of cross platform technologies. In addition, the practical application of these technologies plays an important part in our research. If the technologies in focus are not applied in a broad common sense, they might lack in one or more aspects (e.g., they are just not well-known, they are error prone, they do not offer simplified multi-platform deployment, they do not support well supported programming languages or constructs or they are simply not applicable to specific project requirements).

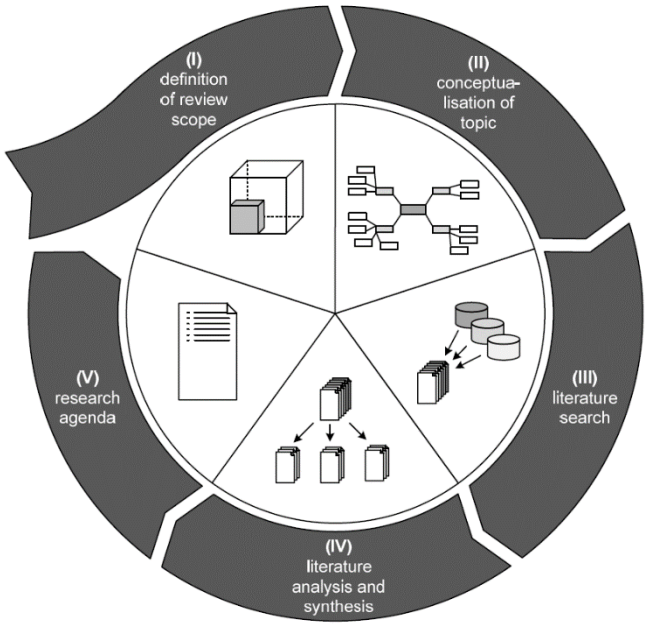

Figure 2: Literature review framework used for literature research according to Brocke et al. [12].

The goal of our work is to answer central questions on cross-platform development. We intend to do so by analysing different cross-platform frameworks and comparing them with each other. In a broader sense, the organisation of the work is to be characterised as conceptual. The different frameworks are described individually and compared step by step. The perspective of the authors is classified as neutral. The authors intend to provide the reader with a no-obligation perspective and to perform the subsequent tests from a neutral perspective. The article is aimed at general scholars who are interested in the cross-platform concept and also want to acquire knowledge about the existing frameworks and their distinctions. In addition, practitioners dealing with cross-platform development will be addressed. The underlying literature research does not claim to be exhaustive, because of its topicality in this thematic area. Therefore, a short overview of the current state of research is given, with the aim of a central literature search, which focuses on the above defined topics.

TABLE I: APPLIED TAXONOMY OF COOPER [13].

Characteristic	**Categories**			
(1) Focus	Research outcomes	Research methods	Theories	Applications
(2) Goal	Integration	Criticism		Central issues
(3) Organisation	Historical	Conceptual		Methodological
(4) Perspective	Neutral representation		Espousal of position	
(5) Audience	Specialised scholars	General scholars	Practitioners / politicians	General public
(6) Coverage	Exhaustive	Exhaustive and selective	Representative	Central/ pivotal

B. Conceptualisation of the topic

In Phase II, the conceptualization of the cross-platform development topic field is carried out. In the current work, this is done on the one hand by brainstorming and on the other hand by systematic mapping of topics on a canvas, which supported the research team in developing a common standpoint on the topic analysed. In addition, initial concepts were identified, which will then be searched for within the scope of the third phase.

C. Literature search

Phase III describes the procedure for literature search. In this section, the reader has the opportunity to gain insights into the authors' systematic approach.

The framework conditions are set at the beginning. Therefore, the key terms are defined as follows. The literary search uses two keyword strings. The first keyword string is "cross-platform development", the second search string equals "cross-platform frameworks". The string was searched in English language.

Furthermore, the media types are defined on the basis of Brocke et al. [12]. The sources of the current research consist of the latest journal and conference articles as well as web documents. The aim of this selection is to identify the main findings on the state of scientific research. Different databases are selected for the two source types. Peer-review articles in journals or conferences are found in current databases such as ScienceDirect, Emerald Insight and especially through the Google Scholar search engine. Web documents have been investigated by the search engine Google.

The next step is to perform the electrical search. In addition to the keyword-string entered, the search settings in the database for articles are set to journals and conferences and the web search is focused on documents. In addition, the last three years are taken into account. The first 20 hits of all types of researched publications are considered, which are sorted in advance according to relevance.

According to later findings, the search was extended to the five specific frameworks Ionic, Xamarin, React Native, Oracle Jet and Flutter. The same search process was used for these five concepts.

D. Literature analysis and synthesis

Phase IV comprises the literature analysis and synthesis of the collected works. This is done through a multi-level filtering approach to obtain the final list of documents relevant to the following activities. These steps apply to the two media types from the previous section.

The first type of filter refers to the title that has been published. The presence document must refer to the keyword string. If this requirement is fulfilled, the literature will continue to be considered. Secondly, abstraction is examined. There, the existing literature must contain elements of the keyword string. If this requirement is also met, the literature will be considered further. The full text is then filtered. If the document contains or refers to the contents of the keyword string, it is included in the final selection. The final step is the redundancy analysis, which eliminates redundant entries from journals and Internet documents.

Afterwards, the relevant literature is then transferred to a spreadsheet in, which the results are first classified according to the following criteria: author, title, publication type, year of publication, language, publisher/journal and found database/search engine. In the next step, the works were examined for their relevance in terms of content. The spreadsheet is converted into a so-called concept matrix according to Webster & Watson [14]. The identified focal points and concepts for cross-platform development and cross-platform frameworks are added to the spreadsheet. The concepts are assigned to the individual documents in the form of crosses. As a result of this step, the concept matrix can be used.

E. Research Agenda

Finally, Phase V aims to identify topics within the so-called Research Agenda. These topics are under-represented in the identified literature. This allows a recommendation; which concepts should be part of future research. Further fields of research are addressed in section VI Conclusion and Outlook. Based on the identified documents, the main focus will mainly be on the cross-platform frameworks. In other words, which of the identified frameworks are particularly useful for the suitability of mobile applications? As a result of Section II, the identified works will be presented in an additional step. This will be addressed as status quo of cross-platform development hereafter in Section III Related Work.

III. Related Work

This section covers different solutions and technologies which enable fast and efficient cross-platform-development of applications.

A. Cross-Platform Development

As stated above, there are several approaches for cross-platform development. This type of development is subject to typical challenges of ubiquitous computing. In addition, further challenges are typical to cross-platform development [6], [15], the most important issue being associated with:

1. User Interface (UI)
2. Limited Resources
3. Device Management
4. Application Maintenance

The design of UI is associated with questions of simplicity and intuitiveness. For mobile cross-platform development, this is extended by design guidelines defined by the different operating systems. It is further restricted because of different device capabilities (e.g., screen sizes and resolution) [16]. Limited resources is a typical issue in mobile software engineering; for cross-platform development the application size and resource consumption (especially power and memory management) is a typical issue [6][17]. Since cross-platform development addresses a vast variety of devices, their management in terms of appropriate usage of hardware and sensors (i.e., CPU, memory, Bluetooth, or camera) becomes another typical challenge. Furthermore, different operating systems must be handled as well. Finally the application has to be maintained by following short lifecycles of devices, operating systems and frameworks [6][16].

A lot of different methods that address cross-platform development can be observed in science and industry. Some are based on model-driven software engineering [18]. The advantage of model-driven methods is that developers and users, which are less familiar with specific programming paradigms are enabled to efficiently implement applications. As Object Management Group (OMG) standard, the Interaction Flow Modeling Language (IFML) offers model-based and platform-independent development of applications for different types of devices. Following the Model-Driven Architecture (MDA) it is based on a meta-model and it is built upon Web Modeling Language (WebML). A Web-based and an eclipse-based modelling environment is provided for IFML. Furthermore, extensions for Apache Cordova and PhoneGap are provided [18][19]. An open challenge is to keep the extensions up-to-date. Other solutions, such as WebView, utilize native code and combine it with Web technologies. Native components are used as containers to render Web pages that contain application logic and presentation layer definitions. Native components serve to access device-specific functions (i.e., push notifications or sensor data). Although WebView is a native application, it can internally use Web technologies without switching to a standard browser. WebView also supports CSS and JavaScript for custom interface development [11]. However, WebView does have two main drawbacks: 1) custom styling is necessary to gain a native look and 2) its performance is below average [20]. In summary, we observe three general approaches to cross-platform development:

1. Native application
2. Transformation- or generator-based application
3. Interpreted application (parser-based)

With native development, an application is developed for each specific device (and operating system). Benefits include the native look and feel, the ability to use all platform-specific features and a comparatively high performance of the app. The most prevalent disadvantage is high efforts for development and maintenance.

TABLE II: COMPARISON OF FRAMEWORK POPULARITY [21].

Source	Unit	Ionic	Xamarin	React Native	Oracle Jet	Flutter
StackShare	Stacks	2.360	461	3.880	N/A	105
	Fans	2.310	549	3.740	N/A	156
	Jobs	98	47	726	N/A	5
	Votes	1.660	646	821	N/A	49
Hacker News	Posts	669	995	917	8	N/A
Reddit	Posts	989	1.080	N/A	55	1.080
Stack Overflow Stats	Posts	4.430	35.000	45.100	280	9.890
GitHub Stats	Stars	36.700	N/A	73.200	295	51.400
	Forks	12.800	N/A	16.200	81	5.360

The latter is a result of redundancy in code and support because each platform has to be served by a separate application [11][15].

The use of generators employs a meta-implementation, which is then transformed to specific platforms (e.g., as used in Cordova or Ionic). Similarly, model-driven development approaches (such as IFML) may use transformations to produce platform specific code. An advantage is that the application logic is platform agnostic [18]. Applications, which are interpreted rely on some kind of parser. The parser interprets application code during runtime in order to create platform specific instructions. Fabrik19 utilizes an interpreted approach in its Mobility Suite (MOS) framework.

B. Cross-Platform Frameworks

As discussed above, there are a lot of cross-platform frameworks like IFML, Cordova, Corona Software Development Kit (SDK), Appcelerator Titanium, TheAppBuilder, PhoneGap, Native Script, SenchaTouch, Framework7, Apache Weex, Flutter, Oracle Jet, Jasonette or Manifold – also see [9]. All of them utilize one or a combination of the three methods to create platform specific applications. In our evaluation we also regard the popularity of the different frameworks. We consider communities like StackShare, Hacker News, Reddit, Stack Overflow Stats as well as GitHub Stats. In our comparison, we strive to evaluate the most frequently used and most progressively developed frameworks as illustrated in TABLE II.

Ionic offers a generator-based approach [22]. The framework is free to use and available as open source. Additionally, several services are available via pay on demand. The generator utilizes a Web application as input. Thus, development of cross-platform applications is based on Web technologies (JavaScript/TypeScript, HTML5 and CSS; see Figure 3). Ionic also relies on Angular [22] in order to foster component based development and reuse of templates. Ionic officially supports Android, iOS and Universal Windows Platform (UWP) [23]. Since Ionic is based on Web applications that are generated into platform specific applications through Apache Cordova, these source applications may also be executed in any Web browser. Native operating system functions and access to sensors is only available after generation of platform specific code. The utilization device specific functionalities often also rely on plugins that have to be declared as dependency [19].

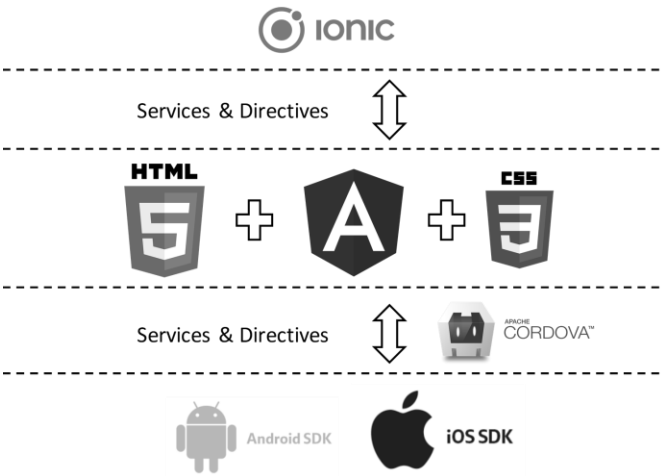

Figure 3: Ionic Architecture [23].

Xamarin is another framework to develop cross-platform apps for Android, iOS and UWB [24]. Other platforms such as Linux are not supported and macOS support was recently added with the launch of Xamarin.Mac.

Xamarin is based on .Net and utilizes C# as programming language. Xamarin is divided into two major parts: 1) Xamarin platform and 2) Xamarin.Forms. The Xamarin platform (Xamarin.Android, Xamarin.iOS) provides APIs to share code for application logic between all platforms. The UI is written individually for each platform. Xamarin.Forms allows to create additional platform-independent UI, which are mapped into native UI in a second step (see Figure. 4). The

development environment is based on Visual Studio (or Xamarin Studio for macOS) [24].

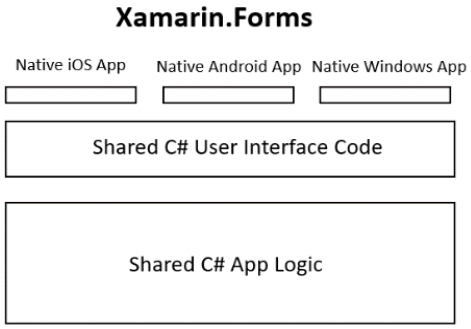

Figure 4: Xamarin Architecture [24], [25].

React Native is a parser based open-source framework for building cross-platform applications [26]. It is based on React. Both frameworks are being developed by Facebook. React Native currently supports Android and iOS and uses a NativeScript Runtime environment to execute the application code (see Figure 5). However, with a little more effort, it is also possible to deploy to UWP. Since React is built on JavaScript, this holds true for React Native as well. React Native invokes Objective-C APIs to render to iOS components and Java Application Programming Interface (APIs) to render to Android components. This means that no code generation is utilized in React Native. Facebook promises that the performance of apps would be almost as good as that of native applications. Components for React Native may either be built as functional components or class components [26].

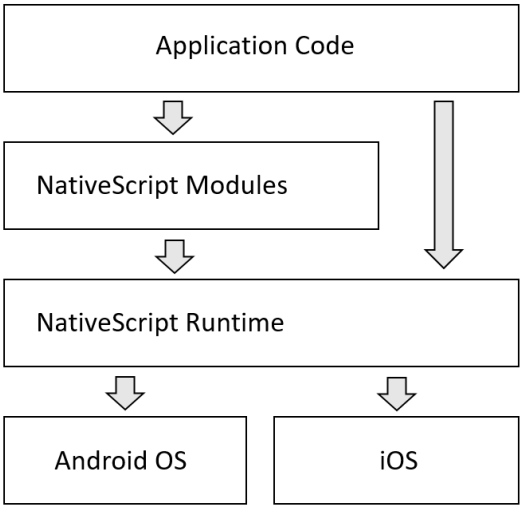

Figure 5: React Native Architecture.

Oracle Jet follows a similar philosophy as Ionic and was developed in the software house Oracle. The framework and its tools are freely available as open source. Oracle Jet also uses the generator-based approach. Web technologies (JavaScript, Knockout, jQuery, HTML5 and CSS, etc. see Figure 6) are used for the development. Platform-specific versions can be derived from the web application. Apps for Android, iOS and UWP can be officially released. Also, an execution in the browser is possible without further ado, particularly since the application is converted by means of Web technologies. For the use of platform-specific sensors, Cordova plug-ins must be used [27][28].

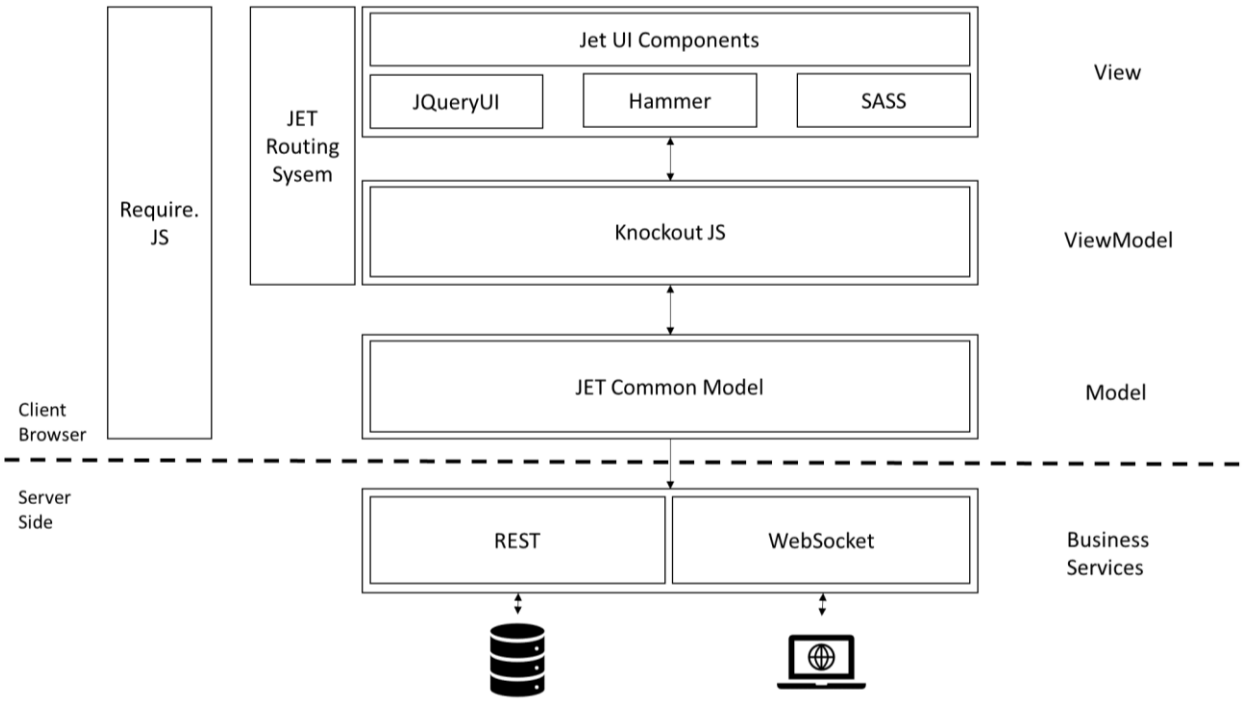

Figure 6: Oracle Jet Component Architecture [29].

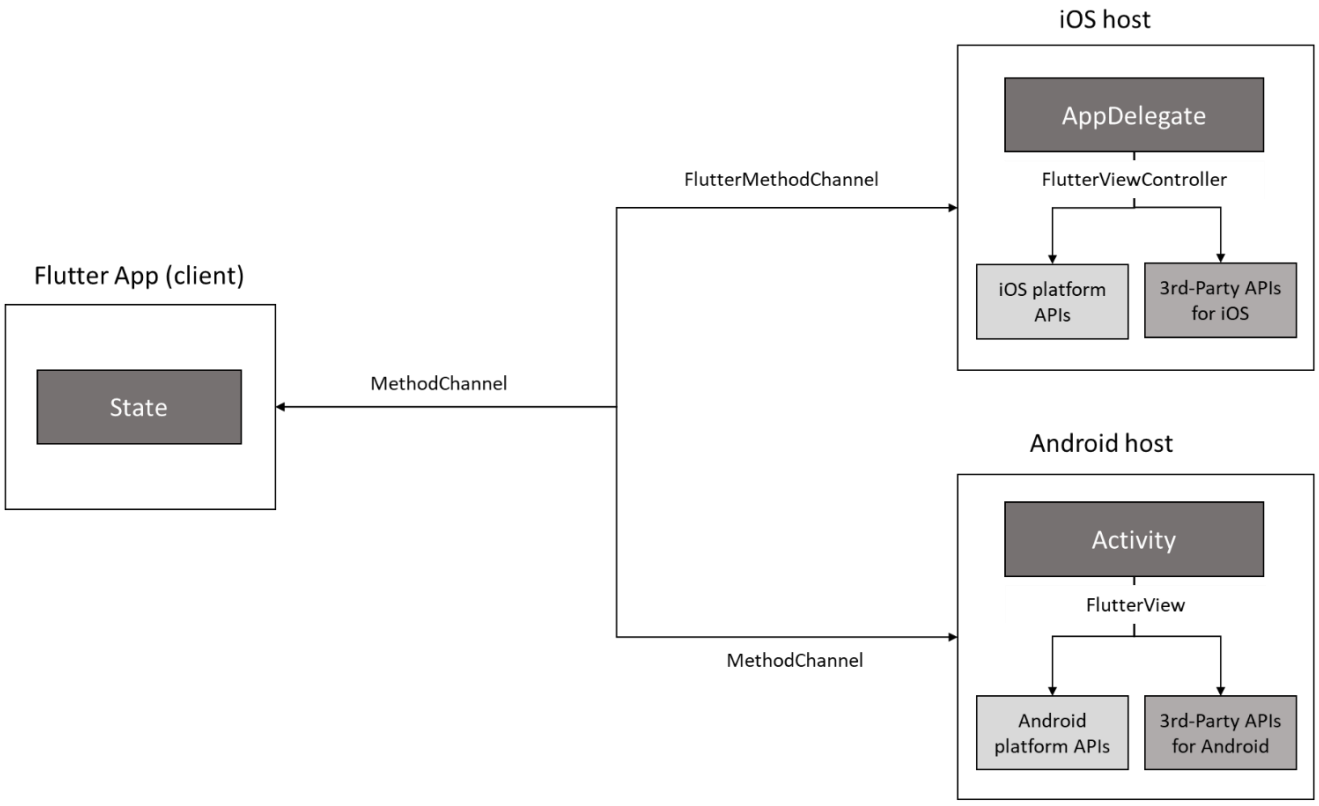

Figure 7: Flutter Architecture.

Flutter is a parser based, open source SDK, which offers an easy way to develop high-fidelity and high-performance mobile apps for android and iOS. Flutter uses a rich set of Cupertino (iOS) and Material Design behaviours and widgets. Furthermore, Flutter implements platform-specific code like navigational patterns, fonts and more. Flutter apps are written in Dart. Its syntax looks a lot like Java, JavaScript, C# or swift. Dart uses the standard Android and iOS toolchains to compile your code [30]. Flutter does not separate views, controllers, layouts and other properties like other frameworks. It uses one consistent, unified object model, so called widgets. Widgets can define structural elements (like buttons or menus), stylistic elements (like fonts or colour schemes, aspects of layouts (like margins) and so on [31]. Messages between the client (UI) and the host (platform) are passed using platform channels as illustrated in Figure 7. These messages and their responses are passed asynchronously. This way the user interface will remain responsive.

IV. Reference Architecture and Implementation

This article follows the constructivist paradigm of design science [32]. Thus, insights will be retrieved by creating and evaluating artefacts in the form of models, reference architectures and, in our case, specific implementation variants and efforts spent on their creation. Contrary to empirical research, the goal is not necessarily to evaluate the validity of research results with respect to their truth, but to the usefulness and feasibility of the different approaches in order to solve a common problem – here, to deploy with ease to different mobile platforms. Following this line of thought, requirements will be imposed by the definition of a reference application architecture. The reference architecture is derived using common hypotheses, practitioner interviews and literature review. The reference architecture serves as requirements model for the implementation of different alternatives and tests in a real environment.

Thus, the reference application architecture is defined to compare most utilized frameworks against each other and to identify strengths and weaknesses. To enact a comprehensive comparison [9][33], the application should access native system functionalities and provide a platform specific UI. In short, the frameworks should generate applications, which are close to native applications. Thus, we also evaluated against platform specific UI guidelines for Android and iOS [34]. We defined the following functional reference criteria:

1. Layout: Grid
2. Layout: Tab
3. Operating System Function: Access current time
4. Sensor Function: Access current position (GPS)
5. Sensor Function: Access the phone camera

In addition to functional criteria, it is also important to measure quality aspects, such as development efforts and application performance. Therefore, we analysed two different types of layouts mentioned in the list above, which are often used in today's apps – Mock-ups are depicted in Figure 8,

which serve as system templates for the reference app. Three tabs can be used to test the system time, GPS and camera functions.

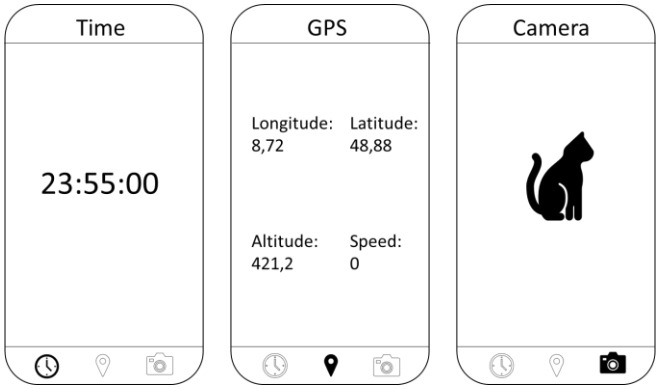

Figure 8: Wireframes.

A. Ionic

Layout – Grid: Ionic provides a typical Grid-View with the `<ion-grid>` component [23]. Furthermore, styling of the GridView can be set individually. **Layout – Tab**: Using Tabs in Ionic is easy as wall, it may be just derived by use of the starter template (which provides this from scratch). Precise instructions may also be found in the documentation [23].

Access system time: This is derived by simple and built-in JavaScript function calls (e.g., date().getHours() is used to get the current hour). **Access current position (GPS)**: To determine the position, the Cordova plug-in Geolocation has to be installed via npm. Then, it can be integrated in the project [23]. As shown in Listing 1 the position can be retrieved, if the necessary sensors are available and permissions are given. **Access to the camera**: To use the camera, the Cordova plugin Camera is required and has to be integrated into the project [23].

Listing 1

```
getThePosition(){
   this.geolocation.getCurrentPosition().
      then((resp) =>{
         this.longitude = resp.coords.longitude;
         this.latitude = resp.coords. latitude;
         this.altitude = resp.coords. altitude;
         this.speed = resp.coords. speed;
   }).catch((error) => {
      console.log("Error getting location",
error);
   });
}
```

Debugging & testing: Ionic offers several methods to debug and test apps. If the application is not utilizing sensor information, a clean Web test can be driven (by `ionic serve`). Web tests may be carried out as known for Web applications in general – such as debugging by means of the browser's developer console (`F12` shortcut) or employing Web driver test scripts. If sensor information is utilized the application has to be deployed to a platform specific device or an emulator. With Ionic this can be done by calling `ionic cordova build android|ios` to build the app and `ionic cordova emulate android|ios` to execute the app on an emulator. If a test device is being utilized instead of emulation (by calling `ionic cordova run android|ios`) the application may again be tested in a browser, e.g., using Google Chrome (`chrome://inspect/#devices` has to be called and the specific device has to be selected). In order to automate unit testing typical tooling as known for other JavaScript-based frameworks can be used. To test the reference implementation, we could simply employ the well-known frameworks Karma and Jasmin. The application source code is publically available via GitHub, the repository URL is:

https://github.com/futureLABHsPforzheim/ionic_blog_prototype.git

B. Xamarin

Layout – Grid: In Xamarin the layout differs, depending on the chosen platform. For Android `GridView` and for iOS `uicollectionview` has to be used [24]. **Layout – Tab**: In Xamarin tabs have to be set up manually. There is no standard template available to support this layout. Typically, a tabbed page will be used to reference other content integrated as tabs.

Access system time: To retrieve the system time, a `ViewModel` is created, and a `DateTime` attribute tracks the current time. For updates a `PropertyChanged` event is fired. The reference is made possible by the data binding. **Access current position (GPS)**: The current position is determined by the plugin `Xam.Plugin.Geolocator` [24] (installed via NuGet). Adjustments are needed to support Android. In addition, necessary privileges for querying the position must be granted. After configuration, the logic can be implemented. Attributes for longitude and latitude have to be mapped to determine the location (see Listing 2). **Access to the camera**: Camera access is realized with the plugin Xam.Plugin.Media [24]. It has to be configured by means of xml. In important step is the definition of a resources folder to determine where to store captured pictures and videos. The camera itself can be called asynchronously (`getTakePhotoAsyncCommand`).

Listing 2

```
public async System.Threading.Tasks.Task
   getLocationAsync()
   {
     var locator = CrossGeolocator.Current;
     locator.DesiredAccuracy = 50;
      if (locator.IsGeolocationAvailable &&
             locator.IsGeolocationEnabled) {
               var position = await
                 locator.GetPositionAsync();
               this.Longitude="Longitude" +
                  position.Longitude.ToString();
               this.Latitude="Latitude" +
                 position.Latitude.ToString();
      }
   }
```

Debugging & testing: Xamarin enables unit testing and debugging with Visual Studio. For Xamarin, Visual Studio basically offers the same mechanisms as known for any other component, which is developed within Visual Studio (such as break points and live debugging). Visual also offers support for asynchronous testing and mock object creation, e.g., if the Model View Viewmodel (MVVM) pattern is applied and view models invoke service operations asynchronously.
Visual Studio also provides a well sophisticated profiler, which provides monitoring of memory utilization and object allocation. Finally, Xamarin also offers also a test cloud for UI-Tests – where automated testing for native and hybrid applications is done by employing the App-Center. The source code of the Xamarin-Reference-App is publically available via GitHub, the repository URL is:

https://github.com/futureLABHsPforzheim/xamarin_blog_prototype.git

C. React Native

Layout – Grid: React Native does not provide a grid layout immediately. To resemble a grid-layout within the reference implementation, a ScrollView component was used and individual views had been adapted by means of CSS. Alternatively, third-party grid components could be utilized as well to resemble a grid layout. React Native Easy Grid and React Native Layout Grid are just two examples of these components, which may be installed via npm. **Layout – Tab**: React-Native Expo IDE can create a starter app, which directly operates with tabs. Manual creation is not as easy as in Ionic but efforts are still considerably low.

Access system time: Is achieved by simple JavaScript calls. `this.state` [26] is needed for the databinding and `new Date().getHours()` retrieves the current hour. **Access current position (GPS)**: The determination of the current position is already integrated in the React Native API [26]. The position is retrieved by calling `navigator.geolocation.getCurrentPosition`, further details can be seen in Listing 3. **Access to the camera**: Camera and access rights have to be configured and `hasCameraPermission` has to be set to zero. The `componentWillMount` method the permissions are checked and we the status is updated. The asynchronous method `takePicture` is utilized tJano check if the camera is available and if it was possible to take a picture.

Listing 3

```
Navigator.geolocation.getCurrentPosition(
   (position) => {
      this.setState({
         latitude: position.coords.latitude,
         longitude: position.coords.longitude,
         error: null,
      });
},
(error) => this.setState({ error: error.message }),
{ enableHighAccuracy: true, timeout: 20000,
maximumAge: 1000},
);
```

Debugging & testing: React Native similarly offers multiple ways to debug and test apps. Debugging mode can be activated from a developer menu. This can be called by keyboard shortcuts or, if running on a test device, by shaking the smartphone. To debug the JavaScript code in Chrome, a remote debugging session can by created when select `Debug JS Remotely` is selected from the developer menu. This will open `http://localhost:8081/debugger-ui` in a new browser tab. Other debugger implementations may be used as well and a recommendation then would be to use the standalone version of React developer tools. These can be installed via `npm install -g react-devtools` and may be called via `react-devtools`. To set up unit testing for React Native it is recommended to utilize Jest and execute tests via node. For integration testing, several different options exist. Integration testing always relies on platform specific environments; thus, those have to be set up first. The application source code is publicly available, the repository URL is:

https://github.com/futureLABHsPforzheim/react_blog_prototype

D. Oracle Jet

Layout – Grid: Oracle Jet offers several components for displaying data. With the tag `<oj-data-grid>` data can also be displayed in a GridView. **Layout – Tab**: Oracle Jet offers ready-made templates for tabs that can be reused. An independent implementation can be done by using the tag `<oj-tab-bar>` in combination with a `<ul>` [27].

Access system time: Oracle Jet also uses JavaScript technologies, so the time query is similar to Ionic (`date().getHours()`. **Access current position (GPS):** Oracle supports the use of Cordova plug-ins. After installing the Cordova plugin Geolocation, it can be integrated into the project. As can be seen in Listing 4, the values are stored in variables latitude and longitude after a successful determination of the location. **Access to the camera:** In Oracle Jet, the Cordova Camera Plugin can be used to provide camera functionality. In order to use the camera in Oracle Jet, an appropriate authorization must be granted.

Listing 4

```
self.getMapLocation = function (info) {

    navigator.geolocation.getCurrentPosition
        (onMapSuccess, onMapError, {
            enableHighAccuracy: true });
        };
        var onMapSuccess = function (position) {

        self.latitude(position.coords.latitude);
        self.longitude(position.coords.longitude);
    };
```

Debugging & testing: Oracle Jet applications can be examined with any browser. In our development environment we used Chrome for debugging. As soon as the application was started with `ojet serve ios|android|windows -browser`, debugging in the local browser is possible. Debugging on an emulator or a physical device is also possible. The commands `ojet serve ios|android|windows -emulator` and `ojet serve ios|android|windows -device` are used

for this purpose. As soon as the app is started, it can be examined in the browser. Chrome offers a selection of available devices via `chrome://inspect/#devices`. Thus, console outputs, network communication, memory utilization etc. can be examined. To test the application and its functionality, different frameworks can be used that are JavaScript compatible. Oracle itself uses QUnit [27]. The source code is publically available via GitHub, the repository URL is:

https://github.com/K1ngdan1/JetRefApp.git

E. Flutter

Layout – Grid: To use the GridView in Flutter with Dart you define a new GridView.count(). CrossAxisCount defines the amount of children in a Row and childApsectRatio their size, both spacing options define how much space is between the grid items, which are defined by children. Here children attribute is filled with a list of widgets (see Listing 5).

Listing 5

```
return new Container(
      margin: const EdgeInsets.only(top: 200.0),
      color: Colors.white30,
      child: new GridView.count(
          crossAxisCount: 2,
          childAspectRatio: 1.0,
          padding: const EdgeInsets.all(4.0),
          mainAxisSpacing: 4.0,
          crossAxisSpacing: 4.0,
          children: widgets),
    );
```

Listing 6

```
return Scaffold(
      appBar: AppBar(
        title: Text('Colla Test App'),
        elevation: 0.7,
        bottom: new TabBar(
          controller: _tabController,
          indicatorColor: Colors.white,
          tabs: <Widget>[
            new Tab(icon: new Icon(Icons.timer)),
            new Tab(icon: new
                    Icon(Icons.location_searching)),
            new Tab(icon: new
                    Icon(Icons.camera_alt)),
          ],
        ),
      ),
      body: new TabBarView(
        controller: _tabController,
        children: <Widget>[
          new Time(),
          new Gps(),
          new Camera(widget.cameras),
        ],
      )
    );
```

Layout – Tab: Dart defines Tabs with a widget named TabBar, which creates the look of the Tab Bar. To use the TabBar a TabController is needed to pass different options e.g., the amount of tabs to the TabBar. The TabBarView creates the functionality for the GridView (see Listing 6) [35]. **Access system time:** To access the system time new DateTime.now() is called. **Access current position (GPS):** To access your location you first need to import the package location. Then define a new variable and initiate it (see Listing 7).

Listing 7

```
Location _location = new Location();
StreamSubscription<Map<String,double>>
    _locationSubscription;
```

As illustrated in Listing 8 it can then be listened to the OnLocationChanged method, which returns the current location [36].

Listing 8

```
_locationSubscription=
_location.onLocationChanged().listen((Map<String,
double> result) {
      setState(() {
        lng = result["longitude"];
        lat = result['latitude'];
        alt = result['altitude'];
        speed = result['speed'];
      });
    });
```

Access the camera: First the camera package needs to be imported and I must be verified if a camera is available (see Listing 9).

Listing 9

```
List<CameraDescription> cameras;
cameras = await availableCameras();
```

Afterwards it is mandatory to pass the found camera and the desired resolution to the controller (see Listing 10).

Listing 10

```
controller = new CameraController(widget.cameras[0],
ResolutionPreset.medium);
```

The asynchronous method takePicture then lets the user take a picture and return its file path (see Listing 11) [36].

Listing 11

```
await controller.takePicture(filePath);
return filePath;
```

The source code of the Flutter-Reference-App is publicly available via GitHub, the repository URL is:

https://github.com/futureLABHsPforzheim/flutter_blog_prototype.git

V. FRAMEWORK EVALUTION

The evaluation and comparison of all frameworks is based on the reference architecture and the implementation of the corresponding test app. We selected several evaluation criteria based on the evaluation framework developed by Heitkötter et al. [37]. The aforementioned covers different evaluation criteria, especially for infrastructure (including the lifecycle as well as the functionality and usability of the app) and app development (including testing, debugging and developing the app). We also extended and removed some criteria (e.g., scalability). Hence, we base the evaluation on the following application properties:

1. Supported platforms
2. Supported development environment
3. Access to platform-specific functions
4. Application look and feel
5. Application portability
6. Simplicity of development
7. Application performance

At first, it is important, which platforms (Android, iOS, UWP, etc.) and to which extent these are supported by each framework. The next criterion discusses all possible development platforms and environments (Windows, MacOS and Linux). With the help of our test app we intend to analyze if platform-specific functions are available. Also, an evaluation of the UI is conducted to measure platform specific look and feel. Moreover, we want to unveil if the source code is reusable and if it can be integrated into other frameworks (portability). Also, the development efforts play a major role and will be evaluated within criterion 6. In order to assess and evaluate efforts and feasibility of the frameworks, we asked five experienced developers to implement our test application according the reference architecture. The following evaluation is also based on their feedback. Finally, we conducted an assessment of the application's performance. Therefore, the test app is used to measure: start time, used memory and execution speed of internal functionalities such as GPS polling. For this purpose, three test devices (Honor 9, Sony XZ1, Samsung Galaxy S7, IPhone 8) were used. The specifications of the test devices are listed in TABLE III. In order to stabilize test results, 100 test runs were conducted for each device. The following subsections will outline our observations for each framework individually. Finally, this section is concluded by a comparison of all frameworks.

A. Ionic

Configuring a system for Ionic and creating a first app only takes a few minutes. Regarding ramp up, the majority of our developers found that Ionic is the easiest framework to start with. It has to be mentioned that it is necessary to ensure that all dependencies (to plug-ins) are installed according their declared version. This can be error prone, especially when Ionic is updated. In case of multiple app development projects, conflicts may also arise between dependencies of different projects. Hence, previously deployed Ionic projects should be removed from the test device to prevent side effects during testing. As a prerequisite to start and develop Ionic applications only knowledge in the typical Web development stack (HTML, JavaScript and CSS) is required. TypeScript as an extension of JavaScript and thus is easy to learn if JavaScript is already known. TypeScript provides additional benefits compared to JavaScript (especially type safety) – some extension have been adopted into ECMAScript-6 (such as classes, inheritance or generics) [38].

In addition, Ionic reuses Angular, which makes it easier to keep the code clean, separate concerns and speed up development of the application itself. The project structure in Ionic is logically well structured according to Web component architecture. Since Ionic relies on Web technologies, the user is free to choose the development environment [23]. The use of Cordova is another advantage, especially because it enables access to system specific functionality and device sensors. Furthermore, Cordova improves re-use of application components, since a single code base can be utilized for all platforms. However, since Ionic is based on Web-technologies and packaged into native wrapper applications, the performance is behind native applications. Former evaluations also indicated that the performance is behind Xamarin and React Native, especially for larger applications [20].

B. Xamarin

Xamarin projects can be set up in Visual Studio. With the use of C#, Xamarin is the best choice for developers, who also work conventionally with C#. Another advantage is the native UI [24]. Users will not recognize any difference to native applications. Xamarin offers to share a single code base between platforms, to develop application logic. Platform specific extensions may be integrated with a subproject feature of Xamarin. As for all cross-platform frameworks, problems may arise with third-party plugins (installed via NuGet). We recognized several issues with outdated plug-ins. In general, our experience has shown that new device and operating features of mobile devices had been adopted very fast by Xamarin. Hence, in most cases its framework-based services can be used instead of third-party plugins. Regarding testing and debugging applications, the developers stated that Xamarin would be the most convenient framework to use. This may be the case because of extended possibilities instantly provided by Visual Studio.

C. React Native

React Native is easy to set up as well. React Native is built upon React and is also based on JavaScript. Applications developed in React Native interpreted directly and the design appears near to native. Interesting features include a well-designed live debugging. With Expo, React Native offers an open source toolchain to simplify deployment on test devices.

TABLE III: TEST DEVICES

	Honor 9	Sony XZ1	Samsung Galaxy S7	IPhone 8
Processor type	HiSilicon Kirin 960 octa-core processor (four 2.4 GHz cores and four 1.8 GHz cores)	Qualcomm Snapdragon 835 MSM8998 Octa-core (quad 2.35 GHz + quad 1.9 GHz) 64-bit Kryo processor	Exynos: Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)	Apple, Hexa-Core (64 Bit) 1. CPU: A11 Bionic, 4 x 2,24 GHz 2. CPU: A11 Bionic
Memory	4 GB RAM	4 GB	4 GB	2 GB
Graphic chip	Mali-G71 MP8 GPU	Adreno 540	Adreno 530	A11 Bionic
Display size	5,15 inches (1080x1920)	5,2 inches (1080x1920)	5,5 inches (1440x2560)	4,7 inches (750x1334)
OS Version	Android 7.0 Nougat	8.0 Oreo	Android 6.0.1 Marshmallow	iOS 12.1.2

Although this may result in some benefits, we observed that the apps that are generated by the Expo are structured differently than those set up by the console. Additionally, these apps have different access to native functions. Another disadvantage compared to the other frameworks is interface development. React utilizes a lot of specific HTML-Tags, which we recognized as somewhat difficult to use and configure. This makes it more difficult to get started than with other frameworks, even if experience in Web technologies is pre-existent.

Oracle Jet

Oracle Jet also offers its own CLI commands for the creation of cross-platform projects. Web technologies such as JavaScript, HTML and CSS are used to build individual pages of the app. To simplify the retrieval and processing of information within the application, the jQuery library is used. In addition, Oracle Jet offers a variety of UI elements to provide arbitrary display options. Oracle indicates, which plugins have already been tested and verified. Other plugins can also be used in Oracle Jet projects, but incompatibilities may occur, and their functionality is not guaranteed. Oracle Jet does not offer in-house debugging features, but existing test and debug tools can be integrated into the system. In addition, Oracle Jet offers its own packaging functions for Android, iOS and Web.

D. Flutter

Flutter is easy to setup. You literally need to download a zip folder, unzip it in a directory with no security restriction (e.g., on Windows C:\). Flutter comes with its own command line tool, but you can integrate it into your system by adding it to your environmental variables. Flutter also offers hot reload and fast native like performance, which enhances the speed of development and the feeling of a native app. Flutter uses its own programming language called Dart. It is oriented on concepts known by Java and JavaScript but offers a lot of specific extensions like the handling of asynchronous calls with their new Future class. It also has its own virtual machine and comes with 2D-Rendering. In the beginning you need to get used to Dart because there are some differences in creating the UI but since their documentation is really good you will be able to adjust yourself pretty quick.

E. Comparative Evaluation

To evaluate all frameworks comparatively and in an objective manner, we implemented a test application according the reference architecture (as defined in Section IV). In a second step, we measured the criteria defined at the beginning of Section V, to reason about benefits and limitations of all frameworks. Based on upon the evaluation criteria presented and measured below, the overall results are summarized in TABLE IV. For each evaluation criterion, we applied a nominal scale that rates the observed framework behaviour in comparison to our general expectations and in relation to the other frameworks. The nominal scale contains the following ratings: “++”=very good, “+”=good, “0”=neutral, “-“=poor, and “--"=very poor.

Supported platforms: Ionic officially supports Android, iOS and since 2016 also UWP development, although the documentation is still very limited here. Xamarin offers full support for Android, iOS and UWP. Limited support is provided for MacOS. React Native supports iOS, Android and with a little extra effort also UWP applications. Oracle Jet also supports iOS UWP and Android platforms. Flutter supports Android and iOS and with the help of Hummingbird it is possible to deploy your app in a specialized browser, which means you can use it on your desktop computer as well. The UWP is not yet supported and probably never will be because the market share is just too small to benefit from it. In a developers perspective Xamarin is rated best, because its platform support is broader than all other frameworks.

Supported development environment: Ionic applications can be developed on Windows, macOS and Linux. The development platform for Xamarin is Visual Studio for Windows and Xamarin Studio for macOS. React Native supports Windows, macOS and Linux. Oracle Jet projects, on the other hand, can be developed on Windows, macOS and Linux. Flutter can be used on any kind of operating system. Literally, since the web-based frameworks depend on technologies available on all common development environments, they all share a best in class rating. The latter does not hold true for Xamarin, since it is based on .net technologies and hence restricts use of development platforms. Thus, for this criterion Xamarin is rated less than all other frameworks.

Access to platform-specific functions: Ionic provides access to iOS, Android, Microsoft and browser-based features. Platform-specific functions can be used via various Cordova

plugins. With Xamarin, all platform-specific functions can be used in a similar fashion. However, Xamarin offers different possibilities to access platform specific functions. The fastest possibility is to install corresponding NuGet packages. A second option would be the definition of interfaces with platform or device specific implementations and expose this shared code via the dependency service. Then there is also the possibility to use native libraries, for example written in pure Java for Android, via binding. While React-Native is JavaScript-based, and many native functions are not supported, it is possible to include native SDKs and libraries. However, this requires specific code for Android (in Java) and for iOS (in Swift) which results in higher development efforts. In addition, these features are currently often not mature enough. Oracle Jet projects use Cordova like Ionic to address platform-specific functions. A common code base is therefore sufficient to address all different platforms and use their device-specific functions. Flutter uses Dart to directly interact with the native API of the client as illustrated already in Figure 8. Overall, the most comprehensive support for use of platform specific functions is given with Xamarin which it is rated best in this category.

Application look and feel: Ionic offers its own widgets for the UI. Navigation elements (e.g., back button) are provided in platform-specific style, so the differences to native apps are small. As already described in Section V, the use of a GridView in Ionic is very simple. Xamarin creates completely native UI, thus the interface is familiar to the user. Xamarin also supports styling with themes and the interface is not different to native apps. Xamarin Android also supports material design. React Native uses specialized widgets. Setting up a GridView it is not as easy as in Ionic or Xamarin, CSS has to be used to achieve this layout. In general, Ionic and React Native ignore style guidelines of platforms partially and some widgets break them explicitly. For example, tabs in Android are at the top of the screen in native apps, while this is not the case in apps developed with Ionic or React Native. Xamarin, in contrast, uses tabs as expected. Oracle Jet offers native themes to display the applications adapted to the platform used. In addition, different templates can be used to adapt the styling to your own needs. Since Flutter uses its own widgets, which are accessing the native API and Widgets via Dart. That brings the look and feel of a native written app. The most native appearance of apps is given when Flutter and Xamarin are utilized. While the other frameworks also get close to a native user experience, they still fall behind.

Application portability: Since Ionic represents a hybrid approach, portability of the source code is given and further supported through Cordova. Since Ionic modules are well-structured and based on Web technologies, they can be transferred to other Web frameworks. However, as many other frameworks, Ionic uses specific HTML tags that may not be supported in other frameworks, thus there is limited transferability of this module part. Since Xamarin separates application logic and UI related code, it offers the best portability and reuse of the logic. Furthermore, Visual Studio offers tolling (portability analyser) to transfer the UI related parts as well. Of course, it has to be said that this is restricted to .Net and mono frameworks. The UI (defined by eXtended Application Markup Language, XAML), could in principle be transformed into HTML or similar languages, which, however, requires further manual efforts in a second step. Similarly, React Native offers portability to different platforms. React-Native code is relatively easy to transfer to other frameworks that use JavaScript, HTML, and CSS. Comparable to Ionic, specialized HTML tags have to be XAML handled manually. However, since as React Native Logic, UI and CSS are typically implemented in a single file, this tends to be tedious. Like Ionic, Oracle Jet uses a hybrid approach. JavaScript is used for the application logic, HTML for the page structure and CSS for the styling. This allows this code to be transferred to other hybrid approaches that use web technologies. However, the Oracle Jet specific HTML tags represent limitations. These cannot be transferred to other frameworks and must be replaced. Even though Flutter uses Dart is possible to use the code elsewhere, since there are possibilities to compile or reuse the code in web apps like Hummingbird, DartPad and dart2js [36][39][40]. Overall, we expected frameworks to be much more advanced already, nevertheless neither of them could really fulfil our expectations.

Simplicity of development: Through a lot of documentation (tutorials, community discussion, API documentation, quick start and programming templates) a quick and efficient start in development of Ionic Apps is possible. Because of the short development lifecycle, confusion may occur through different version documents and some outdated plugins. Occasionally, the framework reveals unexpected behavior (some builds end up with broken apps, while a rebuild without code change is successful). We intend to examine this further. Currently we believe that this is related to generator issues. In principle, the development with Xamarin is fast as well, since the framework also possesses a very good documentation (tutorials, sample projects and a very precise API documentation). The programming language underneath (C#) also is very sophisticated and in our opinion much better then JavaScript. In terms of simplicity, Visual Studio or NuGet may pose a certain barrier for developers not used to it in the beginning. The entry into the development with React Native is comparable to Ionic. The use of the framework-specific UI elements is different from the other frameworks but does not impose an obstacle. The ability to see and debug all changes in real-time eases troubleshooting. A larger issue is related to external libraries and modules. Since many of these modules and libraries are not officially supported, regular maintenance and support is not guaranteed. In addition, we observed that the installation of node modules consumes much more time compared to Ionic. For Oracle Jet, the vendor provides extensive documentation from application setup to testing and debugging Oracle Jet applications. A cookbook is available for existing elements (collections, controls, forms, framework, layout and patterns). Additionally, sample projects and starter templates are offered to speed up the start of development. Thus, detailed documentations are available, which simplify the development with Oracle Jet. A detailed debugging using the browser tools promotes the development and simplifies the finding and elimination of errors.

TABLE IV: EVALUATION OF CROSS-PLATFORM FRAMEWORKS

Evaluation Criteria	Ionic	Xamarin	React Native	Oracle Jet	Flutter
Supported platforms	+	++	+	+	+
Supported development platforms	++	0	++	++	++
Access to platform-specific functions	+	++	0	+	+
Application Look & Feel	+	++	+	+	++
Application Portability	0	0	0	0	0
Simplicity of development	++	+	0	+	+
Application performance	+	+	0	+	++

Flutter comes with a very good documentation that offers a cookbook for different widgets and use cases. It also has a large number of code labs/tutorials to help you getting into Flutter and Dart. In terms of debugging and testing Dart has some tools like the Dart Analyzer to check our code for errors or the Dart Observatory, which lets you define your own breakpoints in the code without using an IDE to do so. Like in every mentioned framework you can print out logs as well. Furthermore, Flutter has integrated the possibility of writing unit tests, widget tests and integration tests, which is well documented on their official page. We faced very little errors and bug while developing though the framework is still young. In this category we rated how quick and flawless a developer can initiate and develop an application. Overall, all frameworks provide developer support but Ionic currently provides the most comprehensive developer guidelines.

Application performance: In order to stabilize test results, we deployed all test applications and measured results of 100 test runs with four test devices. Hence, we could observe the following numbers: The required start time of the Ionic test app is between 2s and 2.44s, while Xamarin requires 3 to 3.3s, React Native 4 to 4.5s and Oracle Jet 1,2s. Flutter just needed 0.5s to start the app The size of the Ionic app is 10 MB, while Xamarin requires 24 MB, React Native requires 11 MB, Oracle Jet requires 19.13 MB and the Flutter app requires 34.12 MB. The time the Ionic app takes to retrieve the current location (with high signal strength of GPS) is approximately 0.1s, while Xamarin needs 3.0s, React Native 0.4s, Oracle Jet 1.2s and Flutter 0,3s.

VI. Conclusion and Outlook

In this article, we reviewed a couple of cross-platform frameworks and discovered that they can significantly reduce development efforts, especially if code shall be shared between different platforms. This of course, is not the case with native app development. Hence, in our point of view cross-platform development is superior to native app development. This especially holds true, if a cost-benefit relation is applied. In addition, there are almost no limitations in the use of native functionalities when cross platform libraries are employed.

In this article, we focused the evaluation on five different frameworks and evaluated, which of them provides the best development support for multiplatform deployment. In order to decide, which frameworks and approaches to choose for evaluation, we conducted a literature review and did some research about popularity of frameworks. Hence, we evaluated developer-oriented web portals and usage statistics of cross-platform development frameworks. While we had been able to discover several scientific approaches to multiplatform development, we recognized that current development is mainly driven by standardized web technologies. These are typically extended by generator-based technologies, which are utilized in order to transform and package applications for platform specific rollout (deployment). Further approaches where defined in literature, such as transformation-based or interpreted application (parser-based) development.

We analysed and evaluated all frameworks on the basis of the following categories: a) supported platforms, b) supported development environment, c) access to platform-specific functions, d) application look and feel, e) application portability, f) simplicity of development, and g) application performance. The results of our evaluation conclude that Flutter is the best suited framework for cross-platform development. In our case study, we examined Ionic, Xamarin, React-Native, Oracle Jet and Flutter in detail. Our results revealed several differences to other comparable articles. First, numerous articles have reported that Ionics's performance is rather poor compared to the other frameworks. However, Ionic scored surprisingly well in our tests. Together with the Flutter framework, Ionic had the best performance score. Compared to other studies [6], performance enhancements of all evaluated cross-platform frameworks could be observed in general. This applies to response and processing times as well as sensor access. The latter can be a result of framework improvements within the latest versions as well as mobile device platform improvements. From a user's point of view, there are almost no observable performance issues when compared to native applications. All evaluated frameworks offer full access to system functionalities and sensors. Although new versions of operating systems can lead to different functionalities and sometimes completely different APIs, the rate and speed of their adoption in cross-platform frameworks is quite high [41]. All cross-platform frameworks allow generic development for different operating systems, although there are still limitations. As we mentioned before, apps are sometimes not fully portable and may require platform-specific customizations. This holds true for all approaches. Furthermore, the statement that the reuse of components between

different mobile applications is not yet fully supported is not valid. Ionic has released betas, which support the use of other components written in React, Vue and Angular [23]. Nevertheless, we could not observe that components developed in one framework could be integrated in other frameworks with ease. Since the web-component standard is promising in this context, we plan to do research of cross-component integration between different frameworks in a next step.

As another important feature of a framework, long-term support is essential to reach a wide range of users and ensure support for current applications. In newer versions of the frameworks examined, it was observed that downward compatibility with APIs of earlier releases was not given. This is common in web development and is known at least since Angular 2 completely broke with the API of its predecessor AngularJS. This can be observed repeatedly for new releases, and its likelihood that this is repeated for future framework releases is high. Thus, it is important for mobile software engineering, whether standardized component development (e.g., the web component architecture) will be adopted in all frameworks. Supplementary important aspects are, if framework consolidation is promoted or whether the spread of new programming languages and techniques will further divide the market. Similar arguments apply to the downward compatibility of the API. This will give rise to further research questions on API issues.

As mentioned above, an important question will be whether it is possible to transfer code from current framework applications into new releases and preserve their functionality (even if the API changes). Therefore, we intend to define a model-driven approach to address this problem in a next step. In this context, we plan to compare parser-based methods with transformation-based cross-platform approaches and to derive API mappings. As a broader outlook, a more detailed study in the area of web components will be performed. In this context, we are planning to assess reusability of web components. Our goal is to evaluate web component programmed utilization across multiple different frameworks.

ACKNOWLEDGEMENT

This article is result of our work conducted in the project "EDV – Einfaches Digitales Vergessen". The research project was supported by the German Federal Ministry of Economics and Energy (BMWi) as part of the smart data funding line.

REFERENCES

[1] J. Christoph, D. Rösch, and T. Schuster, "Cross-Platform Development. Suitability of Current Mobile Application Frameworks," in *The Eighth International Conference on Advanced Collaborative Networks, Systems and Applications*, Venice, Italy, 2018, pp. 13–20.

[2] Statista, "Internet of Things, Forecast, Number of networked devices worldwide by 2020," *Statista*. [Online]. Available: https://de.statista.com/statistik/daten/studie/537093/umfrage/anzahl-der-vernetzten-geraete-im-internet-der-dinge-iot-weltweit/. [Accessed: 07-Mar-2018].

[3] Statista, "Number of smartphone users worldwide 2014-2020." [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed: 15-May-2018].

[4] L. Knoll, "Developing The Connected World Of 2018 And Beyond," *Forbes*. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2018/03/16/developing-the-connected-world-of-2018-and-beyond/. [Accessed: 30-Jan-2019].

[5] L. Corral, A. Janes, and T. Remencius, "Potential advantages and disadvantages of multiplatform development frameworks–a vision on mobile environments," *Procedia Computer Science*, vol. 10, pp. 1202–1207, 2012.

[6] W. S. El-Kassas, B. A. Abdullah, A. H. Yousef, and A. M. Wahba, "Taxonomy of Cross-Platform Mobile Applications Development Approaches," *Ain Shams Engineering Journal*, vol. 8, no. 2, pp. 163–190, Jun. 2017.

[7] Statista 2019, "Marktanteile der führenden mobilen Betriebssysteme an der Internetnutzung mit Mobiltelefonen weltweit von September 2009 bis November 2018," 01-Nov-2018. [Online]. Available: https://de.statista.com/statistik/daten/studie/184335/umfrage/marktanteil-der-mobilen-betriebssysteme-weltweit-seit-2009/. [Accessed: 31-Jan-2019].

[8] Android Developers, "Android distribution dashboard." [Online]. Available: https://developer.android.com/about/dashboards. [Accessed: 01-Jun-2019].

[9] I. Dalmasso, S. K. Datta, C. Bonnet, and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, 2013, pp. 323–328.

[10] M. Lachgar and A. Abdali, "Decision Framework for Mobile Development Methods," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 2, 2017.

[11] V. Ahti, S. Hyrynsalmi, and O. Nevalainen, "An Evaluation Framework for Cross-Platform Mobile App Development Tools: A case analysis of Adobe PhoneGap framework," 2016.

[12] J. vom Brocke, A. Simons, B. Niehaves, and K. Reimer, "Reconstructing the Giant on the importance of Rigour in documenting the Literature search Process," *ECIS 2009 Proceedings*, vol. Paper 161, pp. 1–14, 2009.

[13] H. M. Cooper, "Organizing knowledge syntheses: A taxonomy of literature reviews," *Knowledge in Society*, vol. 1, no. 1, pp. 104–126, Mar. 1988.

[14] J. Webster and R. T. Watson, "Analyzing the Past to Prepare for the Future: Writing a literature Review," *MIS Quarterly*, vol. 2, no. 26, pp. xiii–xxiii, 2002.

[15] P. Smutný, "Mobile development tools and cross-platform solutions," in *Proceedings of the 2012 13th International Carpathian Control Conference*, 2012.

[16] J. Perchat, M. Desertot, and S. Lecomte, "Component based framework to create mobile cross-platform applications," *Procedia Computer Science*, vol. 19, pp. 1004–1011, 2013.

[17] M. E. Joorabchi, M. Ali, and A. Mesbah, "Detecting inconsistencies in multi-platform mobile apps," in *Software Reliability Engineering (ISSRE), 2015 IEEE 26th International Symposium on*, 2015, pp. 450–460.

[18] R. Acerbis, A. Bongio, M. Brambilla, and S. Butti, "Model-Driven Development Based on OMG's IFML with WebRatio Web and Mobile Platform," in *Engineering the Web in the Big Data Era*, 2015, pp. 605–608.

[19] Apache Cordova, "Architectural overview of Cordova platform." [Online]. Available: https://cordova.apache.org/docs/en/7.x/guide/overview/index.html. [Accessed: 09-Mar-2018].

[20] A. Holzinger, P. Treitler, and W. Slany, "Making Apps Useable on Multiple Different Mobile Platforms: On Interoperability for Business Application Development on Smartphones," in *Multidisciplinary Research and Practice for Information Systems*, 2012, pp. 176–189.

[21] "Ionic vs React Native vs Xamarin 2018 Comparison | StackShare." [Online]. Available: https://stackshare.io/stackups/ionic-vs-react-native-vs-xamarin. [Accessed: 07-Mar-2018].

[22] M. Ramos, M. T. Valente, R. Terra, and G. Santos, "AngularJS in the wild: A survey with 460 developers," in *Proceedings of the 7th International Workshop on Evaluation and Usability of Programming Languages and Tools*, 2016, pp. 9–16.

[23] Ionic, "Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular," *Ionic Framework*. [Online]. Available: https://ionicframework.com/. [Accessed: 20-May-2019].

[24] "Developer Center - Xamarin." [Online]. Available: https://developer.xamarin.com/. [Accessed: 07-Mar-2018].

[25] A. Sandu, "Build Cross-Platform Android and iOS UIs with Xamarin Forms," 26-Feb-2016. [Online]. Available: https://www.sitepoint.com/build-cross-platform-android-ios-uis-xamarin-forms/. [Accessed: 31-Jan-2019].

[26] Facebook, "React Native · A framework for building native apps using React." [Online]. Available: https://facebook.github.io/react-native/index.html. [Accessed: 09-Mar-2018].

[27] Oracle, "Oracle Jet Developer's Guide." [Online]. Available: https://docs.oracle.com/en/middleware/jet/6.1/develop/index.html. [Accessed: 18-Jan-2019].

[28] Oracle, "Welcome to Oracle JET." [Online]. Available: https://www.oracle.com/webfolder/technetwork/jet/index.html. [Accessed: 18-Jan-2019].

[29] Oracle, "The Oracle JET Architecture," *JavaScript Extension Toolkit (JET) Developing Applications with Oracle JET*. [Online]. Available: https://docs.oracle.com/middleware/jet410/jet/developer/GUID-293CB342-196F-4FC3-AE69-D1226A025FBB.htm#JETDG113. [Accessed: 31-Jan-2019].

[30] "Building Beautiful UIs with Flutter." [Online]. Available: https://codelabs.developers.google.com/codelabs/flutter/#0. [Accessed: 25-Jan-2019].

[31] "Flutter - Beautiful native apps in record time." [Online]. Available: https://flutter.io/. [Accessed: 25-Jan-2019].

[32] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, Mar. 2004.

[33] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, 2013, pp. 15–24.

[34] M. Willocx, J. Vossaert, and V. Naessens, "A Quantitative Assessment of Performance in Mobile App Development Tools," in *2015 IEEE International Conference on Mobile Services*, 2015, pp. 454–461.

[35] Flutter, "Dart API docs," 07-May-2019. [Online]. Available: https://docs.flutter.io/. [Accessed: 25-Jan-2019].

[36] "Dart Packages," *Dart Packages*. [Online]. Available: https://pub.dartlang.org/. [Accessed: 25-Jan-2019].

[37] C. Rieger and T. A. Majchrzak, "Weighted Evaluation Framework for Cross-Platform App Development Approaches," in *Information Systems: Development, Research, Applications, Education*, 2016, pp. 18–39.

[38] Microsoft, "TypeScript is a superset of JavaScript that compiles to clean JavaScript output," 07-Jun-2018. [Online]. Available: https://github.com/Microsoft/TypeScript. [Accessed: 07-Mar-2018].

[39] Dart, "dart2js: The Dart-to-JavaScript Compiler," 17-Apr-2019. [Online]. Available: https://webdev.dartlang.org/tools/dart2js. [Accessed: 28-Jan-2019].

[40] Y. Jbanov, "Hummingbird: Building Flutter for the Web," *Flutter*, 04-Dec-2018. [Online]. Available: https://medium.com/flutter-io/hummingbird-building-flutter-for-the-web-e687c2a023a8. [Accessed: 28-Jan-2019].

[41] M. Martinez and S. Lecomte, "Towards the Quality Improvement of Cross-Platform Mobile Applications," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, Buenos Aires, Argentina, 2017, pp. 184–188.