Master Thesis

Choosing the right framework for Android development: which mobile development frameworks are chosen and why?

Author: Ashwin Banwarie
Student number: 2509037
Email: a.banwarie@student.vu.nl
Study: Information Sciences
Supervisor: Dr. Sieuwert van Otterloo
Second reader: Dr. Ivano Malavolta
Date of submission: July 15, 2022

Abstract. In recent years, the use of apps has grown significantly. This demands a fast time to market for mobile apps. Mobile developers can choose different strategies to develop mobile apps. For instance, developers for the Android platform can develop apps in Android native, React Native, Flutter, Xamarin, etcetera. Since every framework has advantages and disadvantages, it is challenging for developers to select the most suitable framework. Research has been conducted to gain insight into the current ratio of the frameworks used in the Google Play Store. In addition, a survey and interviews were conducted with various developers. This research indicated that 74.4% of the apps are developed in Android native, and 25.6% are developed in cross-platform frameworks. The results also showed that Android native and Flutter are the most popular frameworks. Considering the strengths and weaknesses, it can be concluded that if performance is the most crucial aspect for the user, then apps should be developed in Android native. The user receives the best UX/UI, and the most complex app features can also be used. If the user experience is not the most crucial factor, Flutter is the best choice for developing a mobile app. In terms of performance and UX/UI, Flutter is close to native. In addition, there is a shorter development time, which means that the development costs are much lower than Android native and the other frameworks. There is also the possibility to develop apps for multiple platforms such as Android, iOS, Web, Desktop, etcetera. It is expected that Android native will remain the largest in the coming years. However, the differences are becoming smaller as the cross-platform frameworks are continuously improving, and the popularity of cross-platform frameworks, such as Flutter and React Native, are increasing.

Keywords: Android development, Google Play Store, Flutter, React Native, Xamarin.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Mobile apps are being developed at an increasing rate to fulfill human needs. According to statistical data, mobile app downloads worldwide have increased dramatically in recent years. While in 2020, 218 billion apps were downloaded by users, that number increased in 2021 to 230 billion app downloads, an increase of 5.5% [1]. In addition, the number of mobile apps available in the leading App Stores, Google Play, and Apple App Store has also increased in recent years. While in 2015, the number of available apps in the Google Play Store was 1.6 million, this number increased in 2021 to 3.5 million [2,3]. Similarly, the number of available apps in the Apple App Store increased from 1.4 million in 2015 to 2.2 million in 2021[1] [3,4]. From the latter, it can be concluded that the demand for mobile apps among various users has been growing continuously over the years.

iOS and Android are the two large competing operating systems for mobile apps [5]. Organizations must develop a mobile app for each platform to service all customers. This is challenging since each platform's design, and development requirements are different [6]. To solve this problem, various cross-platform mobile development frameworks have been developed by tech companies. Developing with cross-platform frameworks makes it possible to develop two apps in one code-based for both the iOS and Android platforms [7]. This research aims to provide more insights into the various mobile development frameworks used within Google Play Store and whether developers indeed prefer cross-platform frameworks.

1.2 Problem definition

Mobile development frameworks are powerful toolkits for building robust mobile apps [8]. Developers can choose different strategies to build mobile apps. For instance, developers can build mobile apps in iOS native, Android native, React Native, Flutter, Xamarin, Ionic, Cordova, Unity, NativeScript, Kotlin Multiplatform, etcetera.

iOS native can only be used to create iOS apps. Similarly, Android native can only be used to create Android apps. To develop a native app for the iOS platform, Objective-C or Swift can be used as a programming language, whereas for the Android native app, Java or Kotlin can be used as a programming language. React Native, Flutter, Xamarin, etcetera are cross-platform mobile development frameworks. Cross-platform mobile development frameworks are used to create apps that run on both iOS and Android platforms [8].

According to Lachgar & Abdali [9], native apps are more improved in performance than apps developed in other frameworks. Shevtsiv & Striuk [7] also stated that the app development cost of native apps is more expensive than cross-platform apps when targeting multi-platforms. This also means that two development teams are working on

[1] Apple and Google constantly remove low-quality content from their app stores. Therefore, the precise quantity of applications may vary.

two different apps for iOS and Android. Shevtsiv & Striuk [7] conjecture that the costs for native mobile development are much higher than when the apps are developed in a cross-platform mobile development framework.

It appears that there are many different cross-platform development frameworks with their advantages and disadvantages. Since developers struggle to choose a particular framework that meets their needs, it is essential to clarify the deciding factors to make it easier for developers to choose a particular type of framework.

1.3 Research question

This research will answer the following research questions:

- **RQ1: What are the most used mobile development frameworks for developing Android mobile apps?**

The following sub-questions have been formulated and will help to answer the first main question:

- SQ1.1: What type of frameworks are used in the Google Play Store?
- SQ1.2: How does the type of frameworks relate to the different categories?
- SQ1.3: What is the relationship between the app downloads and the type of frameworks?
- SQ1.4: What is the relationship between the app ratings and the type of frameworks?

The goal of RQ1 is to identify whether a mobile app is developed with Android native or a cross-platform mobile development framework. Since there are various types of cross-platform mobile development frameworks, it is also essential to check which frameworks have been used to develop the mobile apps. This will be conducted by analyzing the app data from the Google Play Store.

- **RQ2: What are the deciding factors for developers to choose a mobile development framework for developing Android mobile apps?**

The following sub-questions have been formulated and will help to answer the second main question:

- SQ2.1: How do developers rate the various frameworks?
- SQ2.2: How positive or negative are developers about the various frameworks?
- SQ2.3: Which processes do developers go through before choosing a framework?
- SQ2.4: Which deciding factors do developers consider the most when choosing a framework?
- SQ2.5: What impact does the framework have on the product and the development process?
- SQ2.6: For what purposes are the developers using the various frameworks?

The goal of RQ2 is to investigate the most important deciding factors for developers when considering a mobile development framework for Android mobile apps. This will be achieved by conducting a survey and interviews.

The scope of the research will focus on examining free Android apps in the Google Play Store. This is because the APK (Android Package) files of the various apps from the Google Play Store can be mined via AndroZoo[2] (a dataset of Android apps collected from the Google Play Store by the University of Luxembourg). During the literature research, no dataset of IPA (iOS Package) files or tool was found to mine iOS apps.

1.4 Scientific and practical contribution

This research will provide insight into which development frameworks are used the most in the Google Play Store. Furthermore, the focus will be on the deciding factors that should be considered while choosing a particular framework. Besides the thesis that will be written, the research results will also be posted by writing a blog[3] on the Medium platform. This allows developers or experts to gain insight into the research results while using the acquired knowledge in their further work experience.

2 Related literature

Several studies related to this topic have already been conducted in this domain. Malavolta et al. [10,56] investigated the different hybrid mobile apps in the Google Play Store. Using the APK Category Checker tool, data of 11.917 free apps was extracted from the Google Play Store, and the apps were analyzed on various mobile development frameworks. The study's data is outdated since the most popular frameworks currently on the market are not included in this research. For instance, popular frameworks such as React Native was launched in May 2015, and Flutter was launched in December 2018 [11,12]. The focus was on exploring the apps (number of hybrid apps in Google Play Store, most used frameworks, etcetera). Unfortunately, this study has not investigated the deciding factors for choosing a particular framework [10].

Allix et al. [13] described how AndroZoo works and what can be done with it. AndroZoo retrieves metadata from millions of Android apps from the Google Play Store. It contains metadata such as APK data, manifest, releasing, etcetera. AndroZoo has developed specialized crawlers that automatically retrieve the metadata from various apps. Researchers who need the dataset in their research can request access to a selected dataset. The dataset may only be used for research purposes. The goal of AndroZoo is to contribute to ongoing research and enable new potential research topics in Android apps.

A study by Lachgar & Abdali [9] presented a framework to select the best technology to develop a specified mobile application in a given context. The framework

[2] https://androzoo.uni.lu/

[3] https://medium.com/@a.banwarie/choosing-the-right-framework-for-android-development-which-mobile-development-frameworks-are-c813339149a9

determined a mobile development method (native, hybrid, or web) and was based on a set of relevant questions to make a decision. They created a decision tree to adopt the correct development method. After this, the appropriate tool for the implementation based on a set of relevant criteria could be determined [9].

Another research by Nygård [15] conducted a literature review comparing the different platforms and approaches in mobile app development. Nygård [15] mainly looked at aspects such as development costs, supported platforms, performance, quality of UX, sensor and device access, monetization, and app maintenance. His conclusion emerged that it is essential to do further research on modern cross-platform frameworks in the future as they are constantly innovating.

3 Research strategies and research methods

The type of scientific research is quantitative research using data collection and software analysis tools, combined with a survey and interviews to gain qualitative insights. Based on an inductive research approach, data from the Google Play Store was extracted. After this, the extracted data was analyzed, and a theory was formulated for the used mobile development frameworks in the Google Play Store. Also, a theory was formulated for the deciding factors for choosing a framework.

3.1 Quantitative descriptive research

To answer RQ1: "What are the most used mobile development frameworks for developing Android mobile apps?" quantitative descriptive research was performed. By performing quantitative descriptive research, insight was gained into the number of apps (native apps and cross-platform apps). Also, the mobile development frameworks in which the mobile apps are built can be determined.

Data collection. The top 50 free apps from the Google Play Store were selected from 11 categories, resulting in a dataset of 550 apps. The categories such as *Finance, Lifestyle, Shopping,* etcetera were randomly selected. The APK data were collected from February 25, 2022, to April 8, 2022. A data set of at least 550 apps had been chosen to keep the data set not too limited. As a result, a broader scope was examined during the analysis, and frameworks were not excluded.

Data collection was performed using the dataset available via AndroZoo [13]. The tool is actively maintained, and the dataset was accessed by sending a request to AndroZoo. Via AndroZoo, an API key was received. After this, a CSV[4] file was downloaded via the AndroZoo website. The next step was to search the package name of the apps. This was possible by searching the app in the Google Play Store, and via the browser, the package name could be copied from the URL (Appendix 1). Subsequently, the CSV file could be opened, and the SHA256[5] number of the app could be found with

[4] https://androzoo.uni.lu/lists

[5] The APKs are made unique with SHA256 hashes in AndroZoo [13].

the package name (Appendix 1). By entering the API key and the SHA256 number in the link, which is available via AndroZoo[6], the APK file could be downloaded as a .apk file (Appendix 1). The last step was to extract the .apk file so that the APK data of the app could be analyzed.

App downloads and ratings were collected for each app from the Google Play Store. The data of the downloads in the Google Play Store are rounded to whole numbers. For the app ratings, the number of stars per app was collected.

Measurement & data analysis. Measurement and data analysis were performed by looking at the frameworks used to develop the apps in each category. To analyze the various apps, reverse engineering was applied by looking at the source files in the APK data. To open the APK files and analyze the used framework, Android Studio was used. A few examples are provided as a recognition method for the various frameworks. For example, if the *flutter_assets* folder in the project AND the file *"libflutter.so"* in *"/lib/x86_64"* is present, the app is developed using the Flutter framework (Appendix 2). If the file *"libreact_nativemodule_core.so"* AND *"libreactnativejni.so"* in *"/lib/x86_64"* is present, then the app is developed using the React Native framework. Another example is if the file *"libxamarin-app.so"* AND *"Xamarin.AndroidX.Core.dll"* is present, then the app is developed using the Xamarin framework. The other frameworks have been analyzed similarly. The data was processed in a table to visualize the results. The frameworks were indicated by category in a table and were visualized in pie charts and bar charts. Subsequently, the relations between the various categories were analyzed. Based on the data from all the selected apps of all the categories, the used frameworks within the Google Play Store were indicated.

The median downloads of the apps developed per framework were examined to provide insight into the downloads. Due to this, the results are not influenced by outliers. Three categories have been created to determine how well the apps built in a particular framework are rated. The categories are good ratings (3.6-5.0 stars), fair ratings (2.1-3.5 stars), and poor ratings (0-2.0 stars). The number of apps was sorted per framework in the three categories, and based on this, a percentage was calculated.

3.2 Quantitative survey and qualitative interviews

To answer RQ2: "What are the deciding factors for developers to choose a mobile development framework for developing Android mobile apps?" a quantitative survey and qualitative interviews were conducted. The research was conducted to gain insight into the specific sub-questions of RQ2. The interviews were obtained to make the deciding factors measurable for the various frameworks.

Data collection. A survey and interviews were conducted as the main data collection technique for answering RQ2. An online survey was conducted to collect data (Appendix 8). In order to collect data for SQ2.1 and SQ2.2, inspiration has been gained for the survey [15]. In a survey conducted by the State of JS [15] in the figure below, the

[6] https://androzoo.uni.lu/api_doc

satisfaction, interest, usage, and awareness rankings were determined for the JavaScript frameworks from 2016 to 2021.

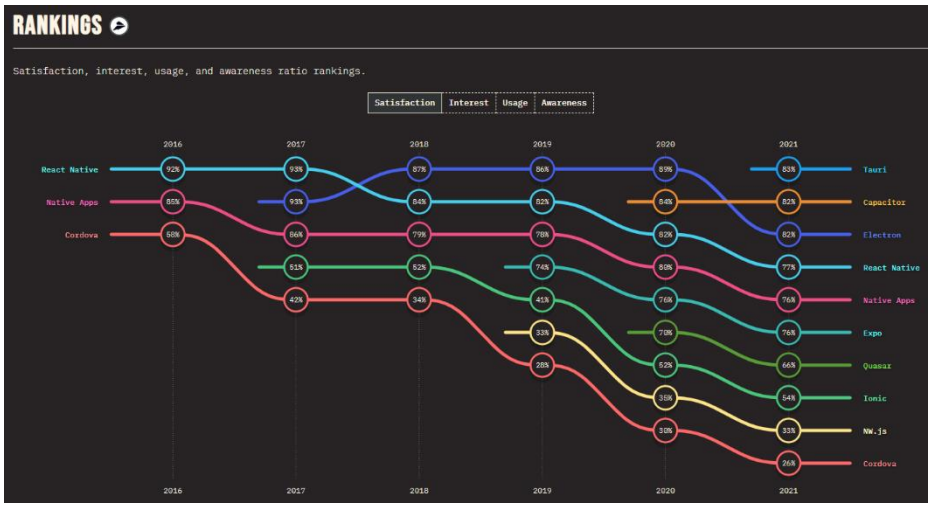


Fig. 1. Satisfaction, interest, usage, and awareness ratio rankings

The interviews were personal online conversations, and the interview questions were prepared based on the survey and literature research. The type of open-ended interview used was semi-structured [16]. Probing questions were asked based on the information provided during the interview. As a result, the same structure was not always followed for all interviewees. In general, a structure was followed, and several questions were prepared (Appendix 7). To collect the data, 10 interviews were conducted with lead developers. The background of the lead developers was investigated via LinkedIn, and an invitation for an interview was sent. For instance, Android native developers, and cross-platform developers in Flutter, React Native, Xamarin, etcetera were interviewed.

Measurement & data analysis. The survey answers were analyzed by using Excel. The responses of SQ2.1 were sorted and calculated based on the formulas presented in table 1 [15]. For instance, for Android native, 26 developers indicated that they would use it again, two developers indicated that they would not use it again, 12 indicated that they were interested, and none of the developers indicated that they had never heard of it. Based on these numbers the scores were calculated. The answers to the survey questions SQ2.2-SQ2.6 were sorted and calculated in percentages. By dividing the number of answers by the total number of respondents, the percentage could be calculated, e.g., to calculate the score of the deciding factor performance, 38 of the 44 developers indicated that they considered performance as an essential factor when choosing a framework. Hence, a percentage of 86.4% is calculated.

Table 1. Formulas for ranking the various frameworks

Ranking of frameworks	Formulas
Satisfaction	would use again / (would use again + would not use again)
Interest	want to learn / (want to learn + not interested)
Usage	(would use again + would not use again) / total
Awareness	(total – never heard) / total

The interviews were analyzed based on grounded theory [17]. After the data was collected, the data was prepared. For instance, pseudonyms were assigned to the developers' names, and the interviews were organized by source. Subsequently, open coding was used to break the data down analytically. The data collected from the interviews was coded per type of developer. For instance, if three interviews with three Android developers were taken, the codes (labels) were compared for similarities and differences. Following that, a codebook was created to help navigate through the data. In addition, axial coding was applied. With axial coding, the codes formed in open coding were made into categories. Open codes were grouped based on similarities, and as a result, broader patterns could be noticed in the data. For instance, an Android developer explains that an app developed in native Android always performs better than one developed in Flutter. Similarly, another Android developer claims that an app developed in Android native performs better than one developed in Flutter. Based on similarities, we can conclude that performance is a category. Also, the literature was revisited to see if the categories connect or differ from the literature. Furthermore, links and relationships between the various categories were explored. Finally, selective coding was applied to identify central categories representing the research's central phenomenon.

4 Results of RQ1

This chapter describes the results of RQ1: "What are the most used mobile development frameworks for developing Android mobile apps?". Subsection 4.1 shows the results of the used frameworks in the Google Play Store. In addition, the relationship between the frameworks and the app downloads and app ratings is shown in subsections 4.3 and 4.4.

4.1 Type of frameworks used in Google Play Store

Table 2 shows an overview of the various mobile development frameworks per category in numbers and percentages. The analysis shows that 74.4% of the apps are developed in Android native. The results indicate that most Android apps are still developed in a native framework, and 25.6% are developed in a cross-platform mobile development framework.

Table 2. Distribution of frameworks in Google Play Store

Category	Android native	React Native	Flutter	Xamarin	Ionic	Cordova	Unity	Other
Communication	43 (86%)	4 (8%)	1 (2%)		1 (2%)	1 (2%)		
Finance	34 (68%)	10 (20%)	1 (2%)	5 (10%)				
Food & Drink	32 (64%)	8 (16%)	2 (4%)	3 (6%)	4 (8%)	1 (2%)		
Health & Fitness	36 (72%)	5 (10%)	5 (10%)	2 (4%)	1 (2%)		1 (2%)	
Lifestyle	28 (56%)	9 (18%)	8 (16%)		2 (4%)	1 (2%)	2 (4%)	
Medical	30 (60%)	5 (10%)	4 (8%)	2 (4%)	2 (4%)	7 (14%)		
Music & Audio	41 (82%)	6 (12%)	1 (2%)				2 (4%)	
News & Magazines	45 (90%)	2 (4%)	2 (4%)		1 (2%)			
Shopping	40 (80%)	5 (10%)	3 (6%)		2 (4%)			
Sports	39 (78%)	3 (6%)	4 (8%)	2 (4%)		2 (4%)		
Travel & Local	41 (82%)	7 (14%)		2 (4%)				
Total apps	409 (74.4%)	64 (11.6%)	31 (5.6%)	16 (2.9%)	13 (2.4%)	12 (2.2%)	5 (0.9%)	0 (0.0%)

Figure 2 illustrates that after Android native (74.4%), the following cross-platform frameworks were used as mobile development frameworks: React Native (11.6%), Flutter (5.6%), Xamarin (2.9%), Ionic (2.4%), Cordova (2.2%) and Unity (0.9%). No other frameworks were found in the dataset of 550 apps.

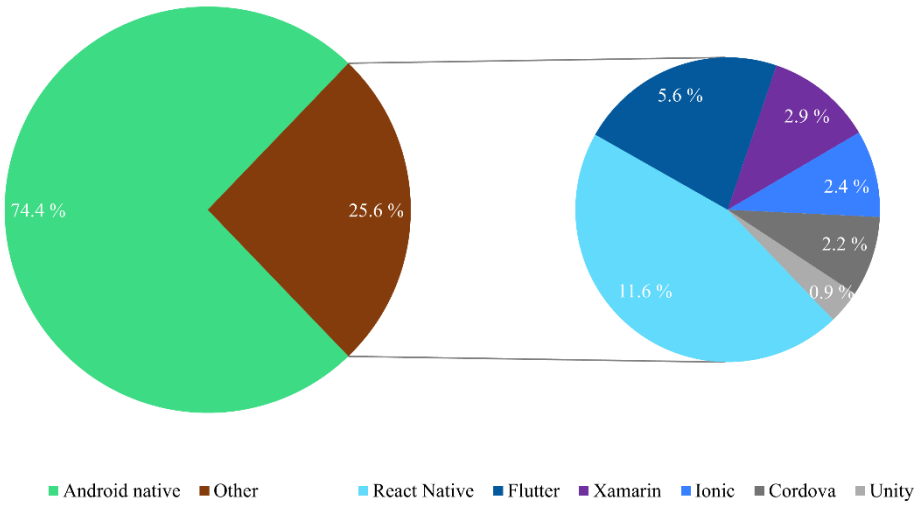


Fig. 2. Android native vs. other frameworks in the Google Play Store

4.2 Type of frameworks in relation to the different categories

Figure 3 illustrates that Android native is used the most in all categories to develop an app. In *News and Magazines* category, 90% of the apps are developed in Android native. It is also noticeable that React Native is used in all categories. Flutter is also used within most categories except in *Travel & Local*.

When analyzing the different categories, React Native is used the most (20%) in *Finance* and the least in *News & Magazines* (4%). In addition, it appears that Flutter is used the most in the category *Lifestyle* (16%) and the least in *Music & Audio*, *Finance*, and *Communication* (2%). Cordova is also used the most in the category *Medical* as a cross-platform mobile development framework (14%). Xamarin, Cordova, and Ionic do not appear in every category and are used relatively few in various categories (between 2-14%).

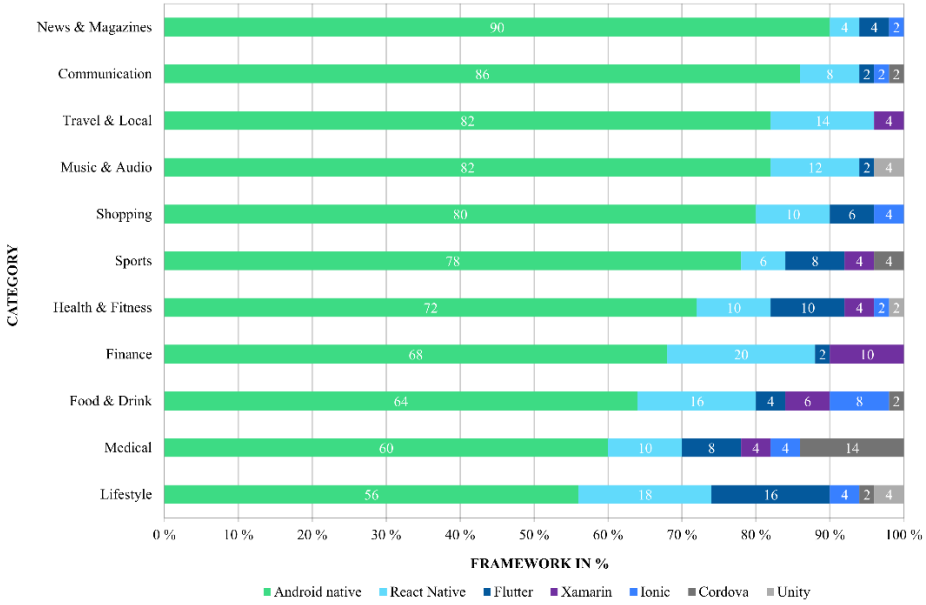


Fig. 3. Type of frameworks in relation to the various categories

4.3 Relationship between app downloads and type of frameworks

Figure 4 shows the relationship between the median of app downloads developed in various frameworks. The results indicate that apps developed in Android native, and Flutter are the most downloaded (1.000.000). The results also show that apps developed in React Native are often downloaded (500.000). Apps developed in Xamarin, Ionic, and Cordova are downloaded the least (between 100.000 and 10.000).

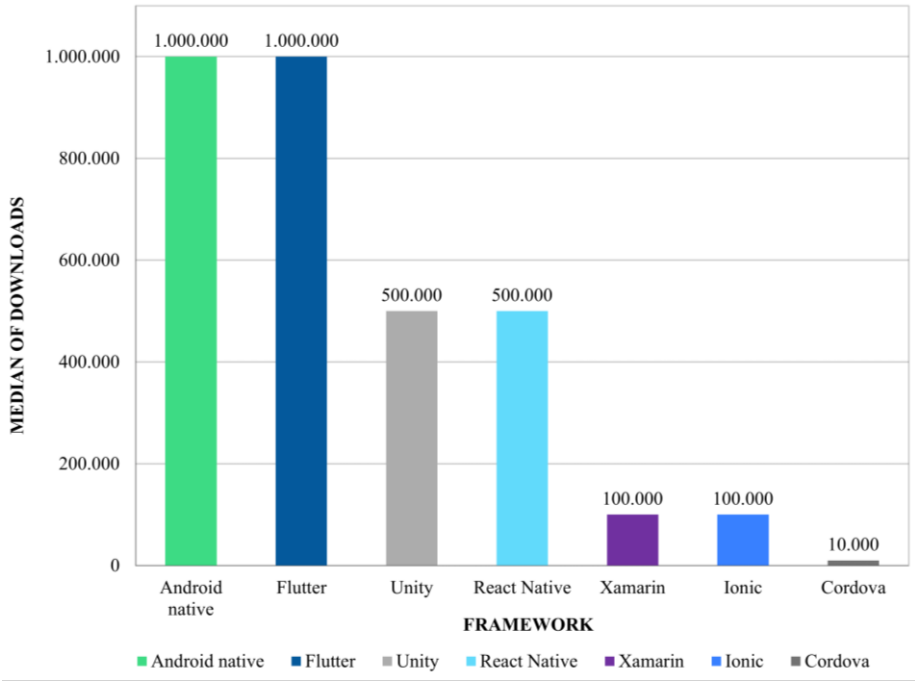


Fig. 4. Median of downloads in relation to frameworks

4.4 Relationship between app ratings and type of frameworks

Figure 5 shows the ratings per framework in three different categories, namely *good ratings (3.6 – 5.0 stars)*, *fair ratings (2.1 – 3.5 stars)*, and *poor ratings (0 – 2.0 stars)*. The results show that apps developed in Android native are the best rated. Of the 409 apps, 325 apps (79.5%) have a good rating, of which 80 apps (19.6%) receive a fair rating, and only four apps (1.0%) have a poor rating. Since the apps developed in Android native are much more common than those developed in the other frameworks, the results indicate that Android native apps are the best rated by the users.

It also appears that apps developed in Flutter receive a relatively large number of good reviews (77.4%). Also, it is noticeable that the highest percentage of poorly rated apps occur in Flutter (9.7%) and React Native (4.7%).

Most apps developed in React Native and Xamarin receive above 60% good ratings. Apps developed in Ionic, and Cordova generally receive fewer good ratings (equal to or less than 50%).

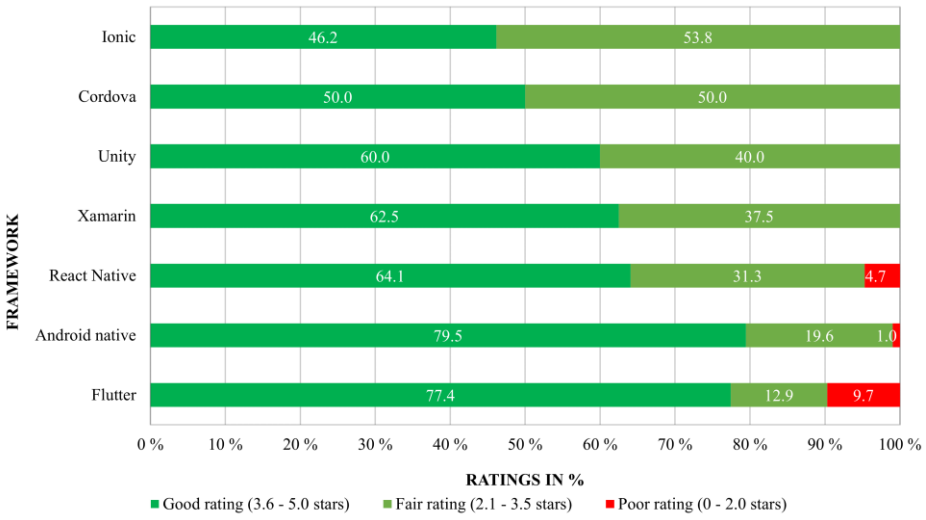


Fig. 5. Ratings in relation to frameworks

5 Results of RQ2

This chapter describes the results of RQ2: "What are the deciding factors for developers to choose a mobile development framework for developing Android mobile apps?". Each subsection describes a sub-question of RQ2.

RQ1 indicated that Android native, React Native, Flutter, Xamarin, and Unity are used as a mobile development framework in the Google Play Store. Based on these frameworks, a survey was created in which 44 mobile developers participated. The results show that the majority (61.4%) of developers have more than five years of mobile development experience, 20.5% have three to five years of experience, 15.9% have one to three years of experience, and 2.3% have less than one year experience in mobile development.

5.1 Rating of various frameworks

All respondents were asked the survey question: "Suppose you would have to start a new mobile project next week. Would you consider using the following frameworks?". Each framework was assessed with the following options: *would use again, would not use again, interested, not interested,* and *never heard*. As indicated in section 3, the *satisfaction, interest, usage,* and *awareness* score were calculated based on the provided options [15].

From figure 6, the results show that 95.2% are satisfied with the mobile development framework Flutter. 92.9% are satisfied with Android native, 82.4% with React Native, and 40% of the developers were satisfied with the mobile development framework

Xamarin. Only 10% to 12.5% of the developers were satisfied with Ionic and Cordova as a mobile development framework.

Figure 6 also shows that most (86.4%) of the developers are interested in Flutter. 75% of the developers indicated being interested in Android native, 57.7% in React Native, 22% in Xamarin, and only 12.9% of the developers showed an interest in Ionic and Cordova.

In addition, most developers (63.6%) said they use Android native as a mobile development framework. Flutter is used by 47.7% of developers as a cross-platform framework, whereas React Native is used by 38.6%, Xamarin by 34.1%, Cordova by 22.7%, and Ionic by 18.2%.

Figure 6 shows that all developers are aware of Android native as a mobile development framework, whereby 97.7% are aware of React Native and Flutter as a cross-platform framework. 95.5% were aware of Xamarin, 93.2% of Cordova, and 88.6% of Ionic. Between 2.3% and 11.4% of the developers were unaware of the frameworks mentioned. The results show that most developers were familiar with the frameworks indicated in the survey.

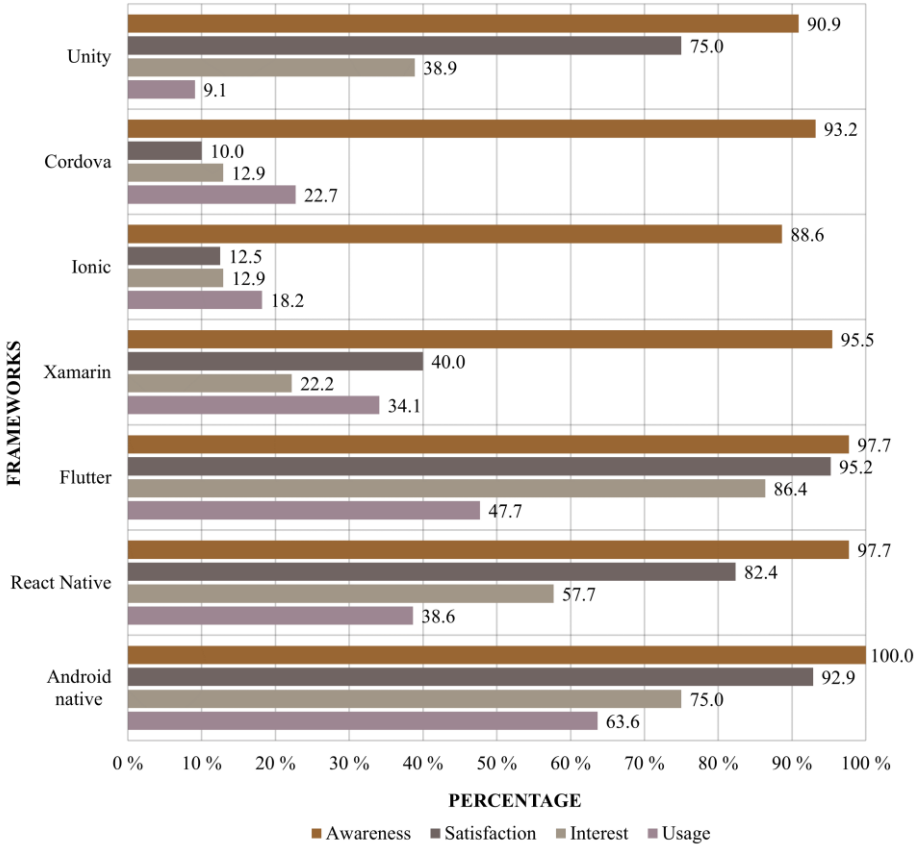


Fig. 6. Rating of various frameworks

5.2 Positive and negative experience with various frameworks

Based on the survey question mentioned in the previous paragraph, the positive and negative experiences of the developers with the various frameworks are visualized in figure 7. The options *would not use again* and *not interested* indicate that developers have a negative experience with the framework (this is visualized in red in figure 7), whereas *would use again* and *interested* indicate that developers have a positive experience with the framework (this is visualized in blue in figure 7). *Never heard* was also one of the options in the survey, but this is not included in the figure because the developers could not indicate whether they are positive or negative about the framework.

Figure 7 shows that 88.7% of developers had a positive experience with Flutter as a mobile development framework, whereas 86.4% and 65.9% of the developers had a positive experience with Android native and React Native. Only 31.8% of the developers had a negative experience with React Native, 13.6% with Android native, and 9.1% with Flutter. The results indicate that Flutter was experienced as the positivist mobile development framework with the least negative score compared to the other frameworks. Less than 30% of the developers appeared to have a positive experience with Xamarin (27.2%), Ionic, and Cordova (11.4%). 81.5% of developers had a negative experience with Cordova, 77.3% with Ionic, and 68.2% with Xamarin. The results show that the developers experienced Cordova as the negativist mobile development framework, with the least positivist score compared to the other frameworks.

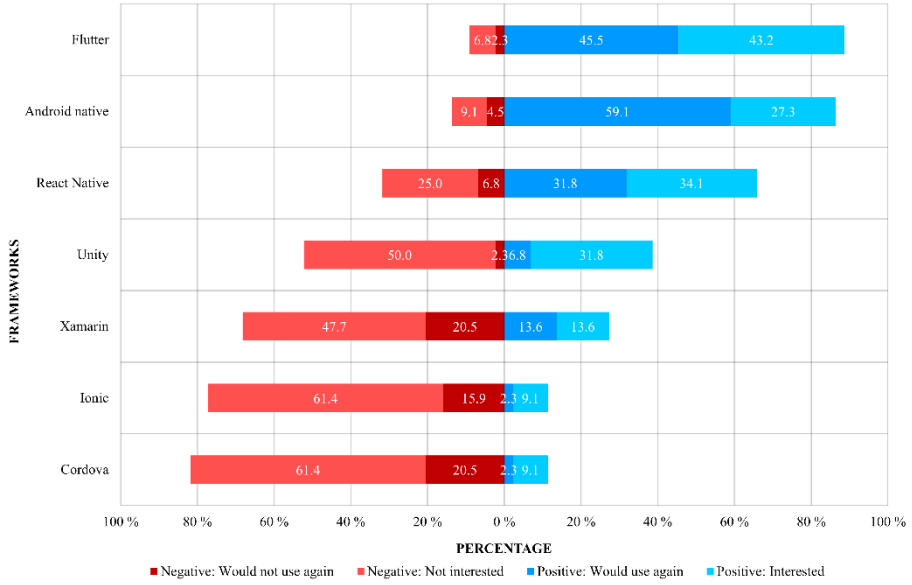


Fig. 7. Positive and negative experience

5.3 Processes before choosing a framework

The respondents were also asked to answer the question: "Suppose you have to choose the most suitable framework. Which processes do you consider when choosing a mobile development framework?". The developers could select multiple options. Figure 8 shows that 81.8% of developers use their expertise before choosing a framework. In other words, a developer with experience with JavaScript and who has previously developed apps in React Native will prefer React Native as a framework. In addition, developers also find it essential to research other frameworks (65.9%) and perform research on similar projects (52.3%). Less than 40% opted for processes such as comparison analysis with previous projects (36.4%), use of in-house guidelines (20.5%), use of external guidelines (18.2%), and feasibility studies (13.6%). In addition, 4.5% (mentioned as other in figure 8) indicated that they also find it essential to ask around within the communities about mobile developers' experiences with frameworks.

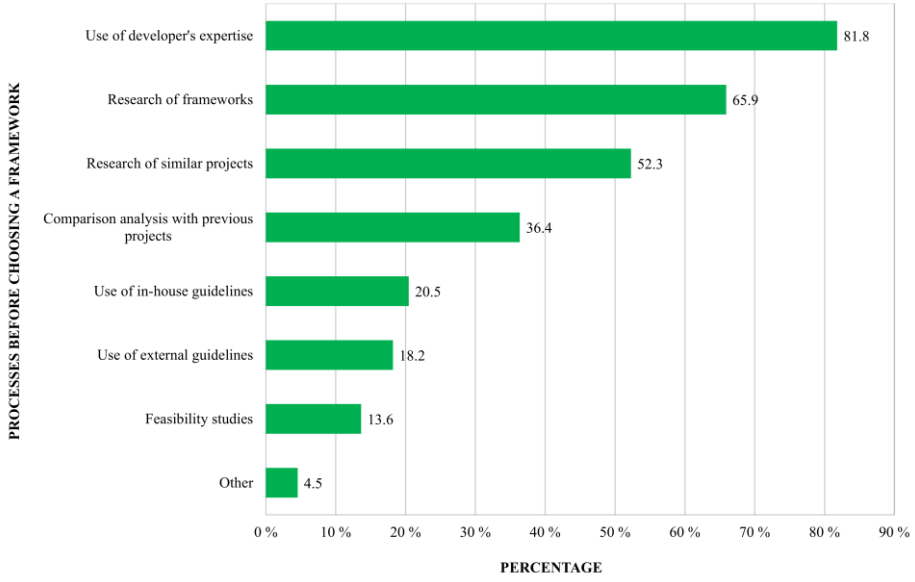


Fig. 8. Processes before choosing a framework

5.4 Most important deciding factors when choosing a framework

The respondents were also asked to indicate the most important deciding factors when choosing a framework. Therefore, the survey asked the following question: "Suppose you have to choose the most suitable framework. Which deciding factors do developers consider the most when choosing a framework?" The developers could select multiple options. Figure 9 indicates that performance (86.4%), development skills (79.5%), development time (75.0%), target platforms (70.5%), app functionalities (features) (68.2%), and UX/UI (63.6%) are in the top six when it comes to the most important deciding factors that developers consider when choosing a framework. License cost

(27.3%) and development cost (38.6%) were chosen by the developers as the least essential deciding factors when choosing a framework. 4.5% of developers (mentioned as other in figure 9) also indicated that they would consider support of IDE when choosing a framework.

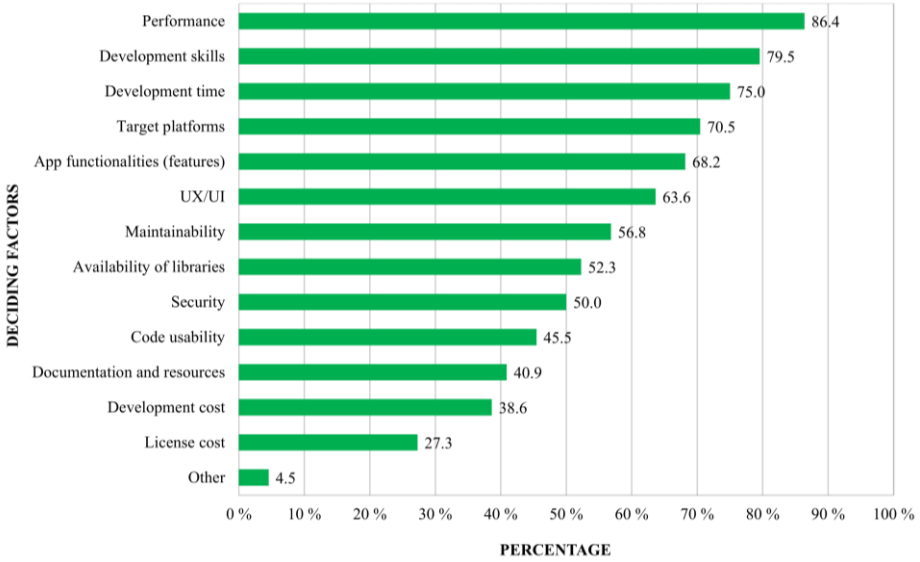


Fig. 9. Deciding factors when choosing a framework

5.5 Impact of frameworks on the product and development process

To show the impact of the different factors, the deciding factors from figure 9 are described. These factors are subdivided into product and development process. With the product, the developed product is meant, in this case, an app developed for the user. The development process refers to the various processes required to develop an app. The impact of the various frameworks on the development process is also described.

For the product and the development process, relative ranks were assigned to each factor per framework so that a comparison can be made on how well particular factor ranks per framework.

For SQ2.5 and SQ2.6 of RQ2, limited or no data could be collected during interviews and survey for the frameworks Unity, Cordova, and Ionic. As a result, no statement could be made for the above frameworks. Also, security was not made measurable during the research because of a lack of data. Hence, no statement could be made.

Impact of various frameworks on the product.

Performance

- **Android native:** *Excellent performance* as Android native apps are compiled using the platform's core programming language and API. They are built for the Android

platform without using layers. Android native apps use direct access to the hardware of the devices (GPS, camera, microphone, sensors, bluetooth, etcetera). In general, the best performance can be achieved with native apps [18]. **Rank: 5.0**

- **React Native:** *Good performance* as React Native communicates through a JavaScript bridge. The JavaScript bridge is between the React Native application layer and the hardware components, and each interaction with the device has to pass through that bridge, which affects performance [18]. In terms of performance React Native is near-native. **Rank: 4.0**
- **Flutter:** *Very good performance* as Flutter is slightly more efficient than React Native and Xamarin. Flutter renders the UI directly. It does not require JavaScript bridges. This allows developers to build complex apps without affecting performance and startup times [18]. In terms of performance Flutter is close to native. **Rank: 4.5**
- **Xamarin:** *Good performance* as Xamarin uses platform-centric hardware stimulation for apps. In terms of performance, Xamarin is near-native apps as the cross-platform capabilities are mainly about sharing the business logic and not the codebase [18]. **Rank: 4.0**

App functionalities (features)

- **Android native:** *Very mature* to use features such as sensors (NFC), camera, GPS, microphone, and bluetooth [19]. **Rank: 5.0**
- **React Native:** *Relatively mature* for apps that use sensors (NFC), camera, GPS, microphone, and bluetooth [20]. **Rank: 4.0**
- **Flutter:** *Relatively mature* for apps that use sensors (NFC), camera, GPS, microphone, and bluetooth [21]. **Rank: 4.0**
- **Xamarin:** *Relatively mature* for apps that use sensors (NFC), camera, GPS, microphone, and bluetooth [22]. **Rank: 4.0**

UX/UI

- **Android native:** *Excellent UX/UI* as Android provides a variety of pre-built UI components such as structured layout objects and UI controls that allow developers to build the graphical user interface for apps. Android also provides other UI modules for special interfaces such as dialogs, notifications, and menus [23]. **Rank: 5.0**
- **React Native:** *Good UX/UI* as React Native implements native UI components, allowing apps to look like native apps and providing a high-quality user interface [24]. **Rank: 4.0**
- **Flutter:** *Very Good UX/UI* as Flutter offers an extensive library of pre-built widgets. Developers can also create their own widgets or customize pre-existing widgets [24]. **Rank: 4.5**
- **Xamarin:** *Good UX/UI* as Xamarin.Forms use standard interface elements and provide a library of templates that can be reused. Xamarin.iOS and Xamarin.Android can be used for manual customization if needed [24]. **Rank: 4.0**

Development cost

- **Android native:** *Expensive* because the apps are built for an individual platform, and code reusability is not possible [18]. **Rank: 3.0**
- **React Native:** *Cost-saving* because the apps are built for multiple platforms, and code reusability is possible [18]. **Rank: 4.5**
- **Flutter:** *Cost-saving* because the apps are built for multiple platforms, and code reusability is possible [18]. **Rank: 5.0**
- **Xamarin:** *Cost-saving* because the apps are built for multiple platforms, and code reusability is possible [18]. **Rank: 4.0**

The relative ranks of development cost are influenced by license cost, development time, target platform, and code usability.

License cost

- **Android native:** *Open-source* [25]. **Rank: 5.0**
- **React Native:** *Open-source* [26]. **Rank: 5.0**
- **Flutter:** *Open-source* [27]. **Rank: 5.0**
- **Xamarin:** *Open-source* [28]. However, developers and enterprises still need to pay between $540 to $3000 per year for Visual Studio Professional/ Enterprise, depending on the license used [29]. **Rank: 4.0**

In figure 10, a relative rank of the impact of the various frameworks on the product is visualized.

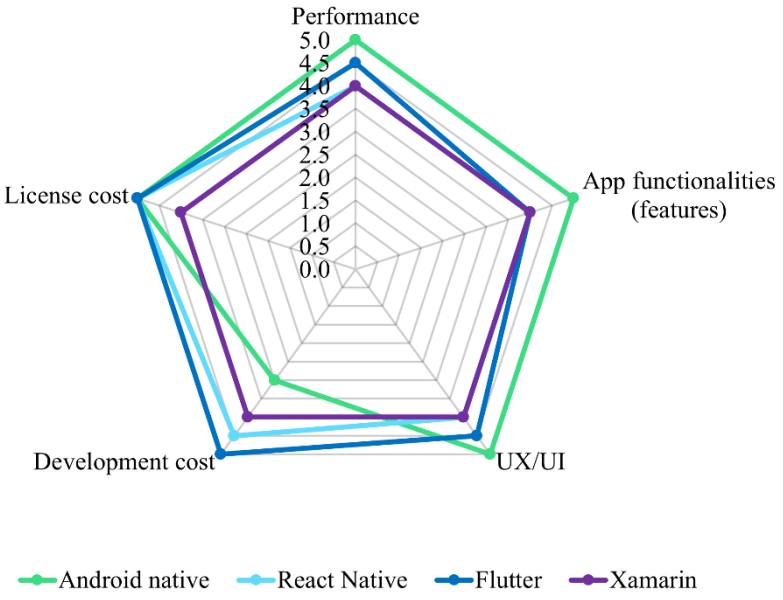


Fig. 10. Relative ranks of deciding factors on the product

Impact of various frameworks on the development process.

Development skills

- **Android native:** Uses *Kotlin or Java (typed)* as a programming language. The availability of developers is *high*, and the learning curve is *easy to learn* [30]. **Rank: 4.5**
- **React Native:** Uses *JavaScript (dynamic)* as a programming language. The availability of developers is *high*, and the learning curve is very *easy to learn* [30]. **Rank: 5.0**
- **Flutter:** Uses *Dart (typed)* as a programming language. The availability of developers is *limited*, and it requires *more time to learn the framework* because it uses the new Dart programming language [30]. **Rank: 3.5**
- **Xamarin:** Uses *C# (typed)* as a programming language. The availability of developers is *limited*, and the learning curve is *easy to learn* [30]. **Rank: 4.0**

A dynamic language like JavaScript has more issues during development because type bugs cannot be checked at compile time but only occur at runtime. However, typed languages like Kotlin or Dart do type checking at compile time [32].

Development time

- **Android native:** *Time-consuming* because the app codes have to be written from scratch for individual platforms. Android native apps can only be used for the Android platform [24]. **Rank: 3.0**
- **React Native:** *Time-saving* because with hot and live reload feature, the development time can be further reduced. React Native offers a vast library of UI components, allowing for faster development time [24]. **Rank: 4.5**
- **Flutter:** *Time-saving* as it uses a single tech stack and shareable codebase that reduces the development time. Developers need to make only minor changes to release apps across various platforms because of a robust set of fully customizable widgets to develop native-like interfaces in a few moments. With the hot-reload feature, the development time is further reduced [24]. **Rank: 5.0**
- **Xamarin:** *Time-saving* as it uses a single tech stack and shareable codebase that reduces the development time. Developers need to make only minor changes to release apps across various platforms. With the hot-reload feature, the development time can be further reduced [24]. **Rank: 4.0**

The relative ranks of development time are also influenced by the target platform and code usability.

Target platforms

- **Android native:** *Mobile (Android)* [19]. **Rank: 1.0**
- **React Native:** *Mobile (Android, iOS)* [33]. **Rank: 4.0**
- **Flutter:** *Mobile (Android, iOS), Web, Desktop (Windows, Linux, macOS), Embedded* [35]. **Rank: 5.0**
- **Xamarin:** *Mobile (Android, iOS)* [35]. **Rank: 4.0**

Maintainability (Updates of operating systems)

- **Android native:** *Always* up to date with the latest version of Android [19]. **Rank: 5.0**
- **React Native:** *Slightly* delayed support for the latest platform updates [36]. **Rank: 4.0**
- **Flutter:** *Slightly* delayed support for the latest platform updates [36]. **Rank: 4.0**
- **Xamarin:** *Slightly* delayed support for the latest platform updates [37]. **Rank: 4.0**

Availability of libraries

- **Android native:** Above *58.9K* Android specific packages [38]. **Rank: 5.0**
- **React Native:** Above *1.0K* React Native specific packages [39]. **Rank: 3.5**
- **Flutter:** Above *23.4K* Flutter specific packages [40]. **Rank: 4.0**
- **Xamarin:** Above *1.3K* Xamarin specific packages [41]. **Rank: 3.5**

Code usability

- **Android native:** Code reuse is not possible [18]. **Rank: 1.0**
- **React Native:** Code reuse is possible up to 90% [18]. **Rank: 5.0**
- **Flutter:** Code reuse is possible up to 85% [34]. **Rank: 4.0**
- **Xamarin:** Code reuse is possible between 80-90% [28]**. Rank: 4.0**

Documentation and resources

- **Android native:** *Very clear and accessible* as Android provides very detailed and easy-to-apply documentation. Developers can read standard documents, watch video training, or even complete lab exercises to master their skills [19]. **Rank: 5.0**
- **React Native:** *Clear and accessible* as there are sufficient documentation and additional resources (user-friendly documentation, guides, tutorials, and Q&A sites) [20]. **Rank: 4.0**
- **Flutter:** Very c*lear and accessible* as it provides detailed and easy-to-apply documentation. Developers can read standard documents, watch video training, or even complete lab exercises to master their skills [21]. **Rank: 5.0**
- **Xamarin:** *Clear and accessible* as it has been on the market for a while and, therefore, provides quality documentation. Developers can dive into use cases, step-by-step tutorials, Q&As, snippets, videos, overviews, and other materials [28]. **Rank: 4.0**

In figure 11, the relative ranks of the impact of the various frameworks on the development process is visualized.

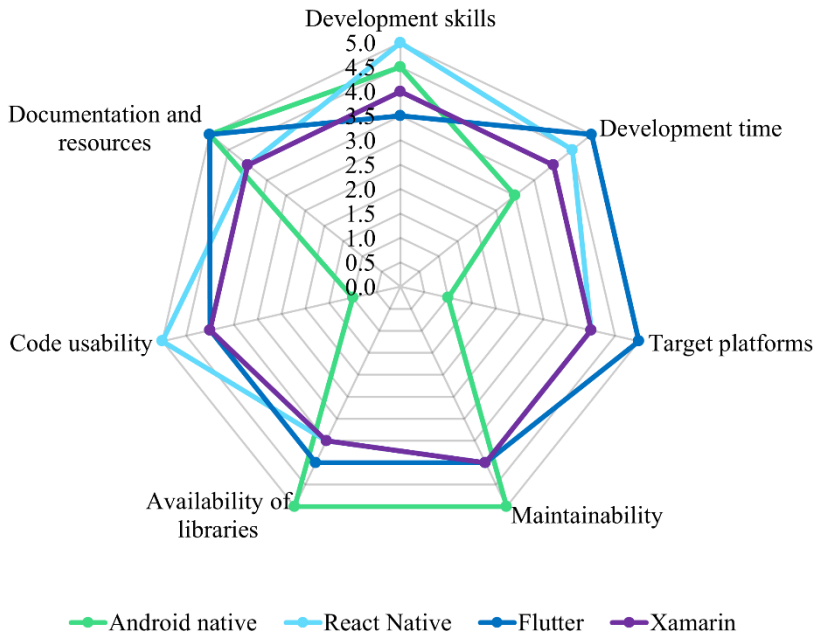


Fig. 11. Relative ranks of deciding factors on the development process

The survey and interviews have indicated that other deciding factors are also important when choosing a framework. In the description below, the other deciding factors are described. However, no rank has been assigned as these deciding factors are not part of figure 9.

Developer community

- **Android native:** Stars are not available on GitHub [42]. There are *103K articles* on Medium [43], while the framework activity is also *descending* on Stackoverflow [31]. *Kotlin is loved by 61.55%* and *Java by 47.15%* of the developers [30].
- **React Native:** There are *104K stars* on GitHub [44] and *17.8K articles* on Medium [45], while the framework activity is *moderate* on Stackoverflow [31]. *JavaScript is loved by 61.55%* of the developers [30].
- **Flutter:** There are *143K stars* on GitHub [46] and *22K articles* on Medium [47], while the framework activity is also rising on Stackoverflow [31]. *Dart is loved by 63.77%* of the developers [30].
- **Xamarin:** There are *5.6K stars* on GitHub [48] and *2.3K articles* on Medium [49], while the framework activity is also descending on Stackoverflow [31]. *C# is loved by 61.96%* of the developers [30].

Developer experience

- **Android native:** Hot and live reloading is not possible. The code is easy to debug, and testing is supported in the framework (Unit, UI, screenshot tests, and performance testing) [59]. Official supported IDEs are Android Studio and IntelliJ IDEA [50].
- **React Native:** Hot and live reloading is possible. The code is difficult to debug, and there is no official support in the framework. Testing is done by third-party tools and frameworks [58]. Official supported IDEs are Visual Studio Code, Visual Studio, Atom, and IntelliJ IDEA [51].
- **Flutter:** Hot reloading is possible. The code is easy to debug, and testing is supported in the framework (Unit, widget & integration testing) [60]. Official supported IDEs are Android Studio, IntelliJ IDEA, and Visual Studio Code [52].
- **Xamarin:** Hot reloading is possible. The code is easy to debug, and testing is supported in the framework (Unit and UI testing) [61]. The official supported IDE is Visual Studio [53].

5.6 Purposes of using the various frameworks

Finally, the respondents were also asked for what purposes they are using the frameworks, whereby multiple options could be selected. In figure 12, the results indicate that Flutter scores the highest for all the purposes compared to the other frameworks. 82.6% of the developers use Flutter for developing proof of concepts, 78.3% for developing new apps from scratch, 52.2% for rebuilding apps from other frameworks, and 47.8% for personal projects.

In addition, React Native also scores high in developing new apps from scratch (72.7%) and developing proof of concepts (63.6%). Moreover, developers use React Native the least for personal projects. Xamarin scores the lowest in developing new apps from scratch (43.8%). Android native is used the least for rebuilding apps from other frameworks (14.7%) and for developing proof of concepts (23.5%).

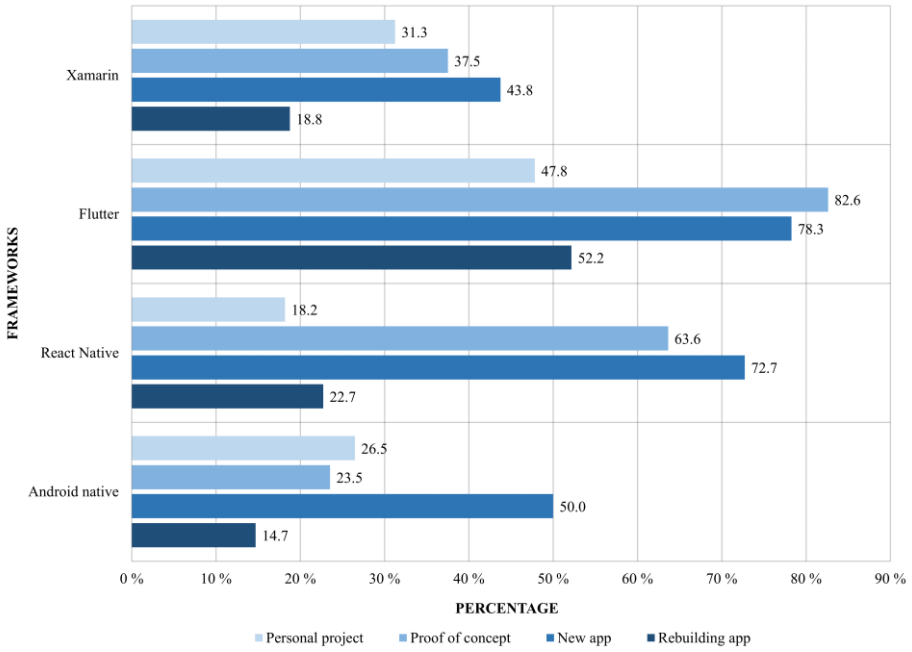


Fig. 12. Purpose of using frameworks

6 Strengths and weaknesses of the various frameworks

This chapter describes the strengths and weaknesses of the various frameworks. Based on these, a framework can be chosen.

Android native	
Strengths	**Weaknesses**
• Excellent performance • Very mature to use features such as sensors (NFC), camera, GPS, microphone, and bluetooth • Excellent UX/UI • Open-source • Uses Kotlin or Java (typed) as a programming language • The availability of developers is high • The learning curve is easy to learn • Always up to date with the latest version of Android	• High development costs when developing apps for multiple platforms • Code reuse is not possible • Development time is high when developing apps for multiple platforms • Apps can be developed only for the Android platform • Hot and live reloading is not possible

• Documentation and resources are very clear and accessible • Very large developer community • The code is easy to debug • Testing is supported in the framework (Unit, UI, screenshot tests, and performance testing) • Jetpack compose makes it much faster and easier to build android native UI	

React Native	
Strengths	**Weaknesses**
• Good performance (near-native) • Relatively mature for apps that use sensors (NFC), camera, GPS, microphone, and bluetooth • Good UX/UI • Low development cost when developing apps for multiple platforms • Code reuse is possible up to 90% • React Native uses pre-developed components • Open-source • The availability of developers is high • The learning curve is very easy to learn • Development time is low when developing apps for multiple platforms • Apps can be developed for the Android, and iOS platforms • Documentation and resources are clear and accessible • Large developer community • Hot and live reloading is possible • The possibility to build platform-specific apps with a native look and feel	• Uses JavaScript (dynamic) as a programming language • Slightly delayed support for the latest platform updates • The code is difficult to debug • There is no official support in the framework. Testing is done by third-party tools and frameworks • Some companies are reluctant to use React Native as it is supported by Meta (Facebook)

Flutter	
Strengths	**Weaknesses**
• Very good performance (close to native) • Relatively mature for apps that use sensors (NFC), camera, GPS, microphone, and bluetooth • Very good UX/UI • Low development cost when developing apps for multiple platforms • Code reuse is possible up to 85% • Flutter is a widget-based framework • Open-source • Uses Dart (typed) as a programming language • Development time is low when developing apps for multiple platforms • Apps can be developed for multiple platforms Mobile (Android, iOS), Web, Desktop (Windows, Linux, macOS), Embedded • Documentation and resources are very clear and accessible • Large developer community • Hot reloading is possible • The code is easy to debug • Testing is supported in the framework (Unit, widget & integration testing) • The possibility to build platform-specific apps with a native look and feel	• The availability of developers is limited • Developers require more time to learn the framework because it uses the new Dart programming language • Slightly delayed support for the latest platform updates • Some companies are reluctant to use Flutter because it is still new and not so mature yet

Xamarin	
Strengths	**Weaknesses**
• Good performance (near-native) • Relatively mature for apps that use sensors (NFC), camera, GPS, microphone, and bluetooth • Good UX/UI • Low development cost when developing apps for multiple platforms • Code reusability is possible • A shared .NET standard library and individual platform projects	• For the supporting IDE (Visual Studio), a license fee has to be paid when using for enterprise and commercial purposes • The availability of developers is limited • Slightly delayed support for the latest platform updates • Small developer community

• Open-source • Uses C# (typed) as a programming language • The learning curve is easy to learn • Development time is low when developing apps for multiple platforms • Apps can be developed for the Android, and iOS platforms • Code reuse is possible between 80-90% • Documentation and resources are clear and accessible • Hot reloading is possible • The code is easy to debug • Testing is supported in the framework (Unit and UI testing) • The possibility to build platform-specific apps with a native look and feel	

7 Discussion

7.1 Implications

When the results are compared with the research conducted by Malavolta et al. [10] in 2015, it appears that the use of cross-platform frameworks has increased in recent years. The research by Malavolta et al. [10] showed that about 96% of the apps are developed in Android native. About 4% of the apps were developed in cross-platform frameworks such as Cordova, Titanium, etcetera. The results of section 4.1 show that approximately 75% of the apps are developed in Android native, and 25% are developed in a cross-platform framework. This indicates that developers have created more apps using cross-platform frameworks over time while the popularity is also growing. It is expected that Android native will remain the largest in the coming years. However, the differences are becoming smaller because the cross-platform frameworks are continuously improving, and the popularity of some frameworks, such as Flutter and React Native, are increasing.

When the results of the first research question are compared with the research conducted by Malavolta et al. [10], it is also noticeable that there are differences in the types of frameworks. The popular cross-platform frameworks, such as React Native, and Flutter, were introduced to the market after 2015 [11,12]. Also, the demand for apps has increased significantly in recent years, leading many companies to consider cross-platform frameworks so that they can quickly bring their app to the market [2,3].

This research has indicated that Android native is the best solution for developing high-volume apps. This is because users can get the best performance and UX/UI in Android native. It is also possible to build the most complex app features in Android

native when using sensors, microphone, camera, etcetera. The cross-platform frameworks can be used when users are satisfied with close to native performance and UX/UI. Also, the development cost is low when developing apps for multiple platforms with cross-platform frameworks. For companies with a small budget that need an app, the best solution would be to develop an app in a cross-platform framework. It is also interesting for start-ups to build MVP (Minimum Viable Product) apps in a cross-platform framework and quickly bring it on the market. If the app needs more complex features after a while, then the app could be rebuilt in Android native.

When looking at the cross-platform frameworks, many apps have been developed in React Native. This is probably because React Native was launched in 2015 [11], and at that time, it was the best cross-platform framework based on the strengths described in section 6.

React Native also originates from the React.js framework, which is famous for developing web applications. As a result, they have excellent integration with each other [57]. Many web developers with JavaScript skills are also available, making React Native famous among them. Flutter was launched in 2018 [12], and given its strengths described in section 6, this is a better framework than React Native and thus a significant competitor for React Native. Also, it is currently the most popular framework among developers [55].

Cordova and Ionic score the poorest as a framework on the various sub-questions of the first research question. These are mainly web-based apps that render in the form of an app on a device. As a result, apps developed in Cordova and Ionic have poor performance, UX/UI, and app features compared to apps developed in other frameworks [54].

7.2 Research limitations

The research contained some limitations. The top 50 apps from 11 categories were analyzed for the first research question. In total, a dataset of 550 apps. From the analysis of these apps, seven types of frameworks were observed. Due to the time available for this research and the time-consuming process of mining and analyzing the apps, it was not possible to investigate a much larger dataset. Perhaps if a much larger dataset of, for example, the top 500 apps of multiple categories in the Google Play Store were examined, other frameworks could be observed than those found in this research.

A second limitation is that for SQ2.5 and SQ2.6 of RQ2, limited or no data could be collected during the survey and interviews for the frameworks Unity, Cordova, and Ionic. As a result, no statement could be made for these frameworks.

Another limitation is that the security of the various frameworks was not measured during the research. Some developers argued that security depends on the developer and how accurately the apps are developed. In other words, a developer determines how secure an app is, for instance, by applying encryption. Another argument was that if there are existing bugs in the frameworks, they are continuously solved by the developing organization of the framework. For instance, Google continuously solves bugs in the Flutter framework. Some developers could not specifically indicate how secure a particular framework was. Their general reasoning was that all the popular frameworks

developed by big tech companies are pretty secure. Based on the arguments, no statements could be made about the security of the various frameworks.

7.3 Recommendations

Similar research could be conducted for the iOS platform to get insight in the various mobile development frameworks and why developers are choosing for a particular framework. A test app could also be developed in multiple frameworks. As a result, the factors such as performance, UX/UI, development time, etcetera can be explicitly measured per platform so that the differences between the various frameworks can be observed on both Android and iOS platforms. Conducting a good experiment can be time-consuming, which could be a research by itself.

7.4 Experiences with AndroZoo

The use of AndroZoo for this study was generally a positive experience. Mining APK files from AndroZoo was quite simple and fast. However, it was not possible to find data such as ratings of apps from the Google Play Store in AndroZoo. In addition, the apps can only be recognized in AndroZoo based on the Android package names. This means that for each app, the package name must be searched via the Google Play Store. After this, the app can be searched in AndroZoo using the package name and downloaded based on the SHA256 number. This is time-consuming as the CSV file contains millions of lines of data, and the search result takes a long time to load. The process of mining the apps took about two weeks for the used data set, and about four weeks were needed to analyze the APK files.

8 Conclusion

This research aimed to gain insight into the various mobile development frameworks chosen for Android development. The conclusions are described for each research question.

8.1 Mobile development frameworks used in the Google Play Store

It can be concluded that about three-quarters (74.4%) of the developed apps are built in Android native. In addition, it was noticed that most apps are built in Android native (>55%) in almost all categories. About a quarter of the apps is developed with cross-platform frameworks (25.6%), such as React Native (11.6%), Flutter (5.6%), Xamarin (2.9%), Ionic (2.4%), Cordova (2.2%) and Unity (0.9%). From the latter, it can be concluded that React Native is the most used cross-platform. This is probably due to the strengths described in section 6 and because React Native has been on the market since 2015 [11]. React Native is also a mature framework with a large developer community. When analyzing the categories, it is noticeable that React Native and Flutter are also used in almost all categories.

When analyzing the relationship between app downloads and the type of framework, it can be concluded that apps developed in Android native, and Flutter are downloaded the most (median 1.000.000). In addition, it can be concluded that apps developed in Cordova are downloaded the least (median 10.000).

Concerning the app ratings and type of frameworks, it can be concluded that apps developed in Android native generally receive the best ratings from the users. It can also be concluded that Flutter and React Native apps receive a lot of good ratings, but most of the poor ratings also occur in these frameworks (between 4.7% and 9.7%).

8.2 Deciding factors when considering a mobile development framework

Considering the strengths and weaknesses in section 6, it can be concluded that if performance is the most crucial aspect for the user, then apps should be developed in Android native. The user also receives the best UX/UI, and the most complex app features can also be used, such as sensors, microphone, camera, etcetera, without affecting the user experience. In situations where apps are developed for consumers for multiple platforms (iOS, Android, etcetera), the development cost and development time are higher during the development process in comparison to Flutter, React Native, and Xamarin.

If performance is not the most crucial aspect for the user, then apps can be developed in cross-platform frameworks such as Flutter, React Native, and Xamarin. Each framework has its advantages and disadvantages, but in general, it can be concluded that Flutter is the best cross-platform framework. Flutter scores the best on most of the deciding factors based on product and development process in comparison to Xamarin and React Native. In terms of performance and UX/UI, Flutter is close to native. In addition, there is a shorter development time, which means that the development costs are much lower than Android native and the other frameworks when developing apps for multiple platforms. There is also the possibility to develop apps for multiple platforms such as Android, iOS, Web, Desktop, etcetera.

When comparing React Native and Xamarin, React Native scores better than Xamarin on most deciding factors based on product and development process. The development time is also lower for React Native than Xamarin, which means the development cost is also lower. Although React Native and Xamarin are both open-source, Xamarin uses Visual Studio as an IDE for which license costs must be paid for commercial purposes.

React Native is recommended for organizations that already have web developers and are looking for a cross-platform framework for mobile development. The framework is very easy to learn for web developers because it is JavaScript-based and very popular among them. Hence, web developers can develop mobile apps easily and fast with React Native. Thus, the development cost is also low when developing apps for multiple platforms. In terms of performance and UX/UI, React Native is near-native.

References

1. Statista. 2022. Annual number of mobile app downloads worldwide 2021 | Statista. [online]. Available at: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-download. [Accessed 25 January 2022].
2. Statista. 2022. *Number of apps available in leading app stores as of 2022 | Statista*. [online]. Available at: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Accessed 25 January 2022].
3. Statista. 2022. *Number of available applications in the Google Play Store from December 2009 to March 2022*. [online]. Available at: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-download. [Accessed 25 January 2022].
4. Statista. 2022. *Number of available apps in the Apple App Store from 2008 to 2022*. [online]. Available at: https://www.statista.com/statistics/268251/number-of-apps-in-the-itunes-app-store-since-2008/. [Accessed 25 January 2022].
5. Statista. 2022. *Mobile operating systems' market share worldwide from January 2012 to January 2022*. [online]. Available at: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/. [Accessed 20 January 2022].
6. Biørn-Hansen, A., Grønli, T. and Ghinea, G., 2019. A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development. *ACM Computing Surveys*, [online], 51(5), pp.1-34. Available at: https://dl.acm.org/doi/pdf/10.1145/3241739?casa_token=xHmNK4vGgycAAAAA:-U0oiSTHo1Jtj3Q9dyJfEEx05Oa7nL1VfyohtY8XHHEb-drGv08aD5NevctInCHcbqbM_Wmn1WtE.
7. Shevtsiv, A.N., & Striuk, M.A. (2020). Cross platform development vs native development. *CEUR Workshop Proceedings*, [online], 2832, pp. 75-83. Available at: http://ceur-ws.org/Vol-2832/paper09.pdf.
8. Admin, 2022. *How to Choose the Right Mobile App Development Framework? - 6 Tips Inside*. [online]. Explorate Global. Available at: https://www.explorate-global.com/blog/choose-mobile-app-development-framework/ [Accessed 19 January 2022].
9. Lachgar, M. and Abdali, A., 2017. Decision Framework for Mobile Development Methods. *International Journal of Advanced Computer Science and Applications*, 8(2), pp.110-118. https://doi: 10.14569/IJACSA.2017.080215.
10. Malavolta, I., Ruberto, S., Soru, T. and Terragni, V. (2015). End Users' Perception of Hybrid Mobile Apps in the Google Play Store. *2015 IEEE International Conference on Mobile Services*, 2015, pp. 25-32, https://ieeexplore.ieee.org/document/7226668.
11. Reactnative.dev. 2022. *React Native · Learn once, write anywhere*. [online]. Available at: https://reactnative.dev/. [Accessed 18 June 2022].
12. Docs.flutter.dev. n.d. *FAQ*. [online]. Available at: https://docs.flutter.dev/resources/faq#:~:text=Flutter%201.0%20was%20launched%20on,hundreds%20of%20millions%20of%20devices. [Accessed 11 June 2022].
13. Allix, K.T. F., Bissyandé, J. Klein & Traon, Y.L. (2016). AndroZoo: Collecting Millions of Android Apps for the Research Community, *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 2016, 468-471. Available at: https://androzoo.uni.lu/static/papers/androzoo-msr.pdf.
14. Nygård, J. (2019). Mobile application platform selection. Available at: http://jultika.oulu.fi/files/nbnfioulu-201905242064.pdf.

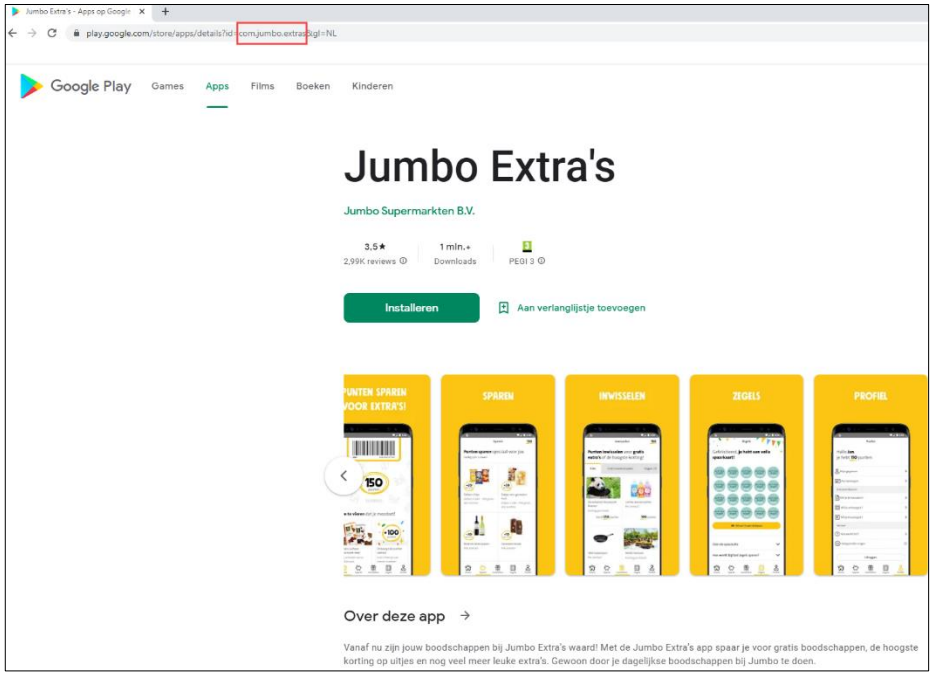
15. State of JS. (2021). *Mobile & Desktop*. Available at: https://2021.stateofjs.com/en-US/libraries/mobile-desktop.
16. Patton, M. Q. (2002). *Qualitative interviewing*. Qualitative research and evaluation methods, 3, 344-347.
17. Corbin, J. & Strauss, A. (1990). *Grounded Theory Research: Procedures, Canons, and Evaluative Criteria*. Qual Sociol 13, 3–21 (1990). Available at: https://doi.org/10.1007/BF00988593.
18. Kurale, R., & Bala, K. (2021). A Comparative Study of Flutter with other Cross-Platform Mobile Application Development. *INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS – IJCRT*, 9(12), pp. 368-372. Available at: https://ijcrt.org/papers/IJCRT2112036.pdf.
19. Android Developers. 2021. *Developer Guides.* [online]. Available at: https://developer.android.com/guide. [Accessed 20 June 2022].
20. Reactnative.dev. 2022. *Introduction*. [online]. Available at: https://reactnative.dev/docs/getting-started. [Accessed 18 June 2022].
21. Docs.flutter.dev. n.d. *Cookbook*. [online]. Available at: https://docs.flutter.dev/cookbook. [Accessed 17 June 2022].
22. Docs.microsoft.com. 2021. *Get started with Xamarin*. [online]. Available at: https://docs.microsoft.com/en-us/xamarin/get-started/. [Accessed 12 July 2022].
23. Android Developers. 2021. *User Interface & Navigation*. [online]. Available at: https://developer.android.com/guide/topics/ui. [Accessed 20 June 2022].
24. Rees, J. (2020, June 3). *Flutter Vs React-Native Vs Xamarin.* DZone. Available at: https://dzone.com/articles/flutter-vs-react-native-vs-xamarin
25. Android Open Source Project. n.d. *Android Open Source Project*. [online]. Available at: https://source.android.com/#:~:text=Android%20is%20an%20open%20source,source%20project%20led%20by%20Google. [Accessed 12 June 2022].
26. Ramos, H., 2018. *Open Source Roadmap*. [online]. Reactnative.dev. Available at: https://reactnative.dev/blog/2018/11/01/oss-roadmap. [Accessed 9 June 2022].
27. Flutter.dev. n.d. *Flutter - Build apps for any screen*. [online]. Available at: https://flutter.dev/#:~:text=Flutter%20is%20an%20open%20source,applications%20from%20a%20single%20codebase. [Accessed 9 June 2022].
28. Microsoft.com. 2021. *What is Xamarin?* [online]. Available at: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin. [Accessed 9 June 2022].
29. Microsoft. n.d. *Pricing and Purchasing Options | Visual Studio*. [online]. Available at: https://visualstudio.microsoft.com/vs/pricing/?tab=enterprise. [Accessed 9 June 2022].
30. Stack Overflow. 2022. *Stack Overflow Developer Survey 2022*. [online]. Available at: https://survey.stackoverflow.co/2022/. [Accessed 13 June 2022].
31. Insights.stackoverflow.com. 2022. *Stack Overflow*. [online]. Available at: https://insights.stackoverflow.com/trends?tags=android%2Cflutter%2Creact-native%2Cxamarin. [Accessed 13 June 2022].
32. Ortin, F., Zapico, D., Perez-Schofield, J. and Garcia, M., 2010. Including both static and dynamic typing in the same programming language. *IET Software*, 4(4), pp.268-282. Available at: https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2009.0070.
33. Reactnative.dev. 2022. *Cross Platform Implementation · React Native*. [online]. Available at: https://reactnative.dev/architecture/xplat-implementation. [Accessed 14 June 2022].
34. Flutter.dev. n.d. *Multi-Platform*. [online]. Available at: https://flutter.dev/multi-platform. [Accessed 11 June 2022].

35. Microsoft.com. 2021. *Xamarin.Forms supported platforms*. [online]. Available at: https://docs.microsoft.com/nl-nl/xamarin/get-started/supported-platforms?tabs=windows. [Accessed 9 June 2022].
36. Anwar, S. (2021). Comparison and evaluation of cross-platform framework and development of a digital health platform using selected framework. Available at: http://uu.diva-portal.org/smash/get/diva2:1626535/FULLTEXT01.pdf.
37. Beladiya, K. (2021). *Flutter Vs React Native Vs Xamarin – Top Cross Platform Mobile App Development Framework*. Available at: https://theonetechnologies.com/blog/post/flutter-vs-react-native-vs-xamarin-top-cross-platform-mobile-app-development-framework.
38. mvnrepository.com. n.d. *MVN Repository*. [online]. Available at: https://mvnrepository.com/search?q=android&c=android. [Accessed 16 June 2022].
39. Reactnative.directory. n.d. *React Native Directory*. [online]. Available at: https://reactnative.directory/?android=true. [Accessed 13 June 2022].
40. Dart_Flutter_Packages. n.d. *pub.dev*. [online]. Available at: https://pub.dev/packages?q=sdk%3Aflutter+platform%3Aandroid. [Accessed 13 June 2022].
41. Nuget.org. n.d. *NuGet Gallery*. [online]. Available at: https://www.nuget.org/profiles/Xamarin. [Accessed 13 June 2022].
42. GitHub. n.d. *Android*. [online]. Available at: https://github.com/android. [Accessed 10 June 2022].
43. Medium. n.d. *The most insightful stories about Android - Medium*. [online]. Available at: https://medium.com/tag/android. [Accessed 17 June 2022].
44. GitHub. n.d. *Code frequency facebook/react-native*. [online]. Available at: https://github.com/facebook/react-native/graphs/code-frequency. [Accessed 17 June 2022].
45. Medium. n.d. *The most insightful stories about React Native - Medium*. [online]. Available at: https://medium.com/tag/react-native. [Accessed 17 June 2022].
46. GitHub. n.d. *Code frequency · flutter/flutter*. [online]. Available at: https://github.com/flutter/flutter/graphs/code-frequency. [Accessed 17 June 2022].
47. Medium. n.d. *The most insightful stories about Flutter - Medium*. [online]. Available at: https://medium.com/tag/flutter. [Accessed 17 June 2022].
48. GitHub. n.d. *Pulse · xamarin/Xamarin.Forms*. [online]. Available at: https://github.com/xamarin/Xamarin.Forms/pulse. [Accessed 16 June 2022].
49. Medium. n.d. *The most insightful stories about Xamarin*. [online]. Available at: https://medium.com/tag/xamarin. [Accessed 17 June 2022].
50. Android Developers. 2022. *Meet Android Studio| Android Developers*. [online]. Available at: https://developer.android.com/studio/intro. [Accessed 16 June 2022].
51. Kaczorowski, M. (2021, October 11). *The best IDEs for React Native to use in 2022*. Ideamotive. Available at: https://www.ideamotive.co/blog/the-best-ides-for-react-native.
52. Docs.flutter.dev. n.d. *Set up an editor*. [online]. Available at: https://docs.flutter.dev/get-started/editor?tab=vscode. [Accessed 16 June 2022].
53. Visual Studio. n.d. *Xamarin App Development with Visual Studio | Visual Studio*. [online]. Available at: https://visualstudio.microsoft.com/xamarin/. [Accessed 15 June 2022].
54. www.javatpoint.com. n.d. *Ionic vs Cordova - javatpoint*. [online]. Available at: https://www.javatpoint.com/ionic-vs-cordova. [Accessed 30 June 2022].
55. Statista. 2022. *Cross-platform mobile frameworks used by global developers 2021| Statista*. [online]. Available at: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/. [Accessed 18 January 2022].
56. Malavolta, I., Ruberto, S., Soru, T. and Terragni, V. (2015). Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation. *2015 2nd ACM International Conference*

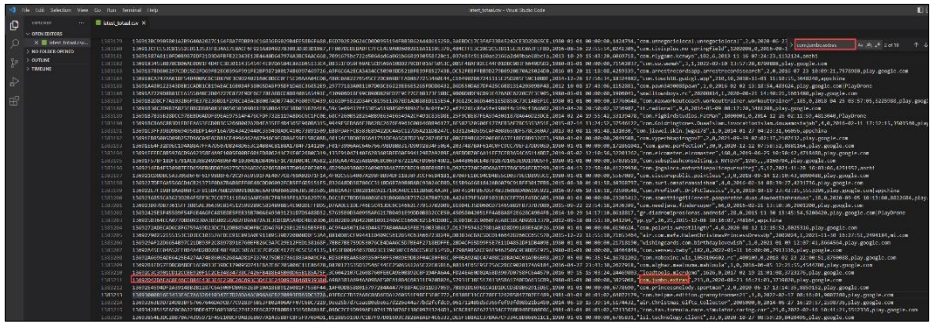
on Mobile Software Engineering and Systems, 2015, pp. 56-59, https://ieeexplore.ieee.org/document/7283028.
57. Ashwini, A., 2017. What is the difference between React.js and React Native? | CognitiveClouds Blog. [online]. Cognitiveclouds.com. Available at: https://www.cognitiveclouds.com/insights/what-is-the-difference-between-react-js-and-react-native#:~:text=performing%20UI%20layer.-,React.,to%20render%20components%20on%20mobile. [Accessed 1 July 2022].
58. Reactnative.dev. 2022. Testing · React Native. [online]. Available at: https://reactnative.dev/docs/testing-overview. [Accessed 19 June 2022].
59. Android Developers. 2021. What to test in Android | Android Developers. [online]. Available at: https://developer.android.com/training/testing/fundamentals/what-to-test. [Accessed 19 June 2022].
60. Docs.flutter.dev. n.d. Testing Flutter apps. [online]. Available at: https://docs.flutter.dev/testing. [Accessed 19 June 2022].
61. Docs.microsoft.com. 2022. Xamarin.UITest - Visual Studio App Center. [online]. Available at: https://docs.microsoft.com/en-us/appcenter/test-cloud/frameworks/uitest/. [Accessed 19 June 2022].

Appendix 1 Package name, SHA number, and APK download

- The package name of the app “Jumbo Extra’s” is com.jumbo.extras



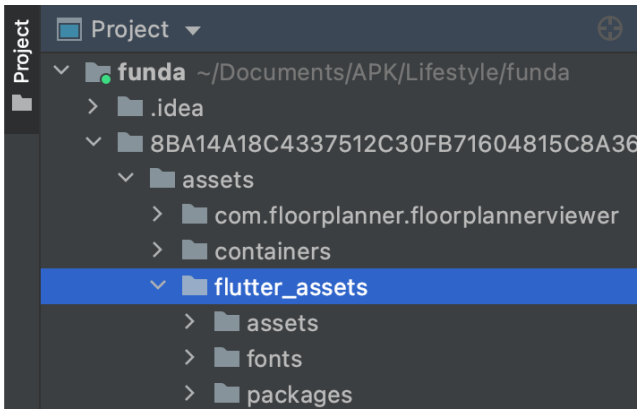
- SHA256 number of the package name “Jumbo Extra’s”

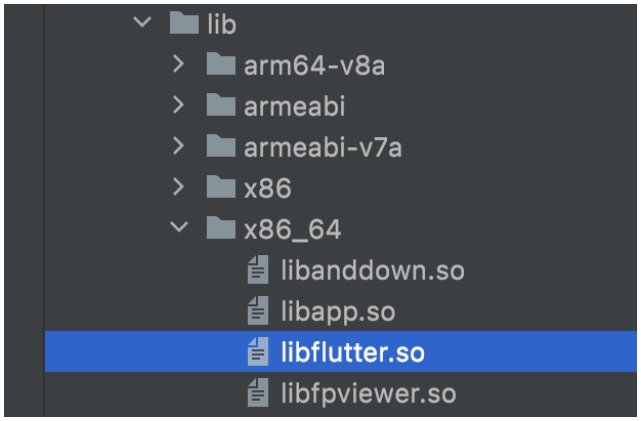


- Download of APK file. The APK Key is personal and may not be distributed or made publicly available. Data cannot be redistributed without consent of AndroZoo.

Appendix 2 Example of recognition method

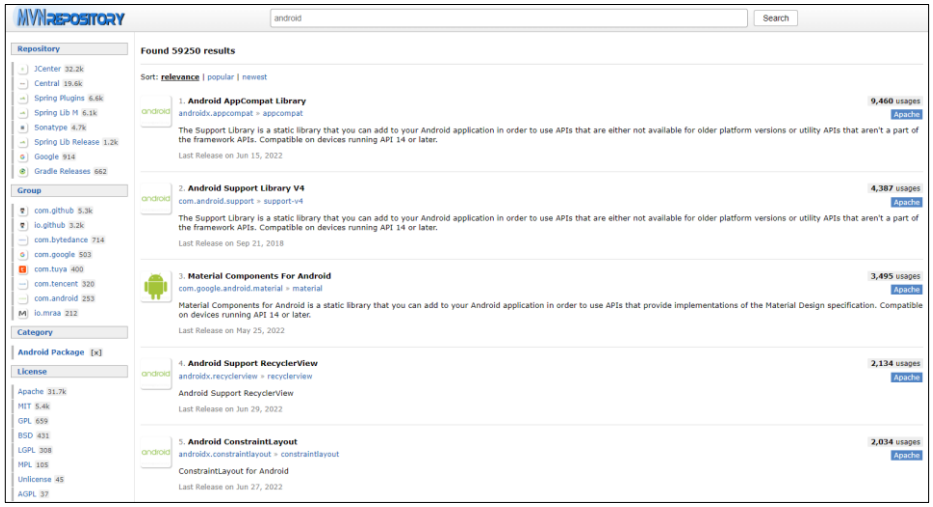
- If the *flutter_assets* folder in the project AND the file *"libflutter.so"* in *"/lib/x86_64"* is present, the app is developed using the Flutter framework.



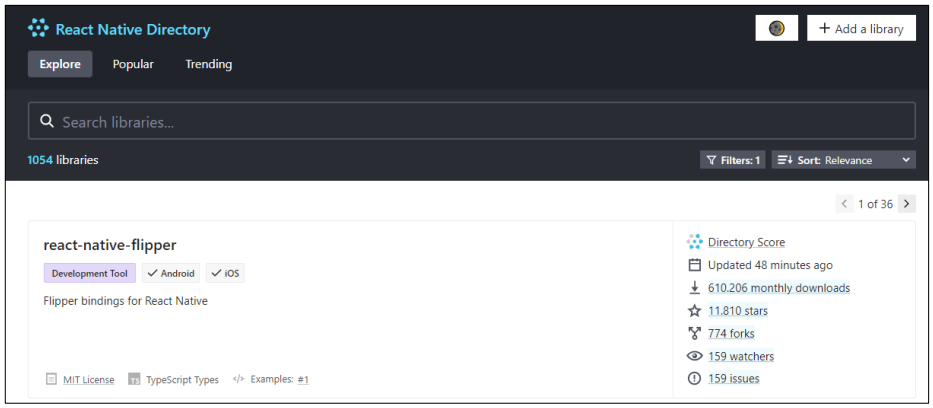


Appendix 3 Packages

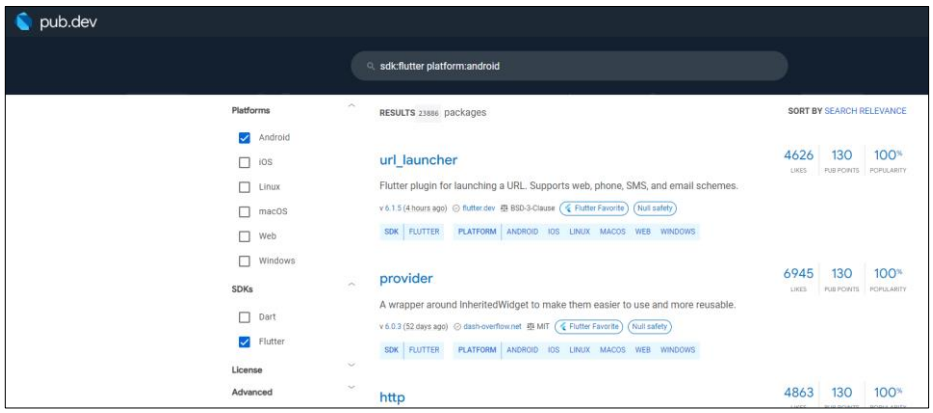
- Android packages



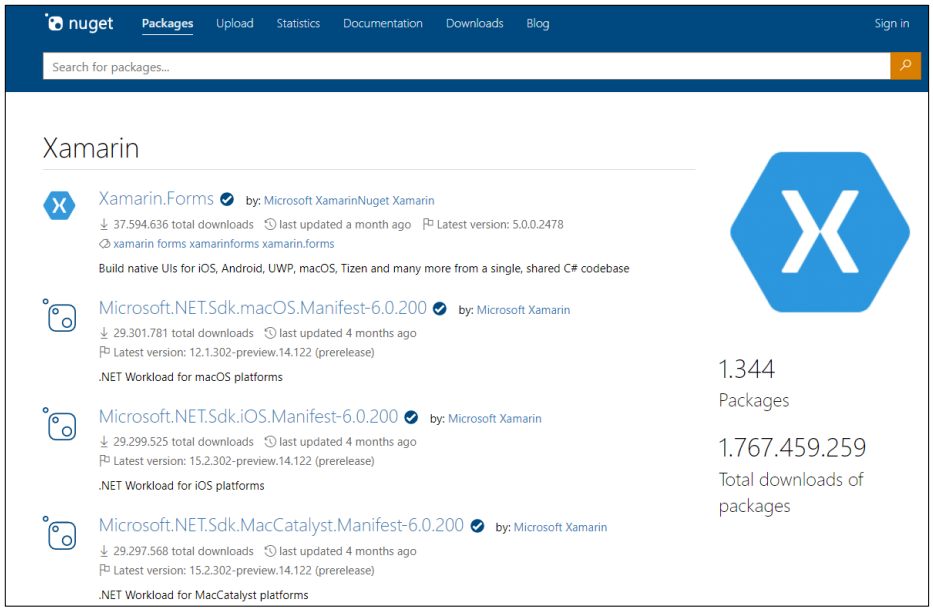
- React Native packages



- Flutter packages

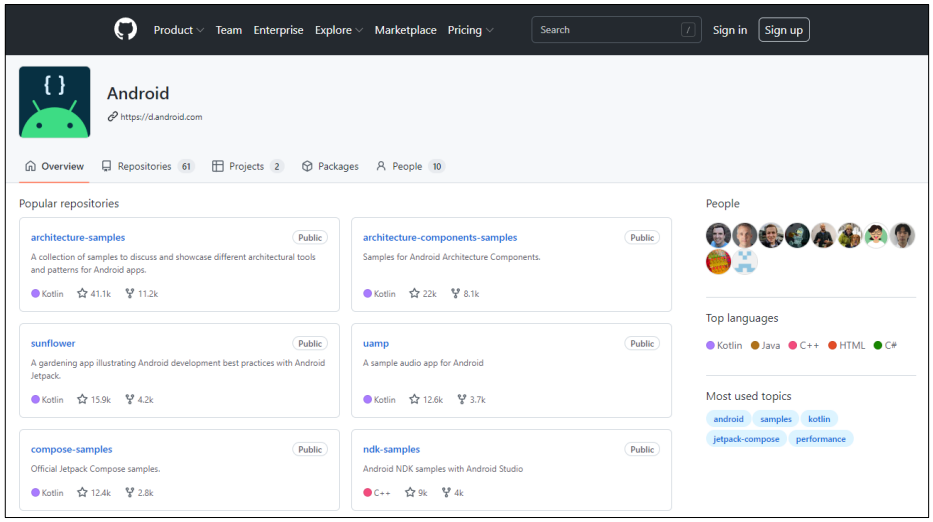


- Xamarin packages

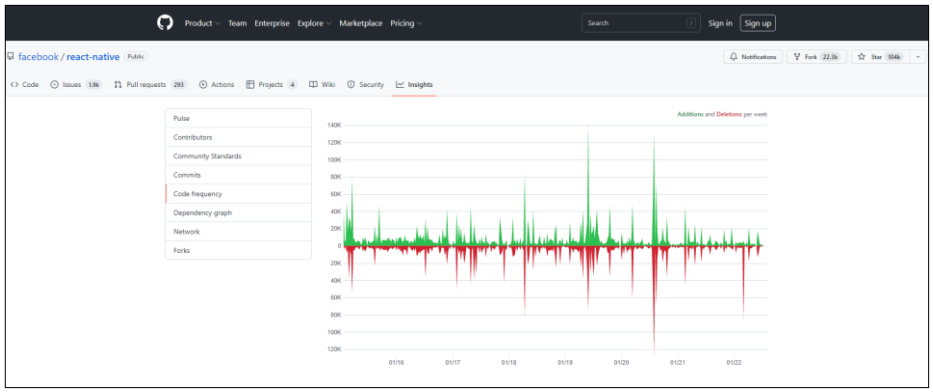


Appendix 4 GitHub stars

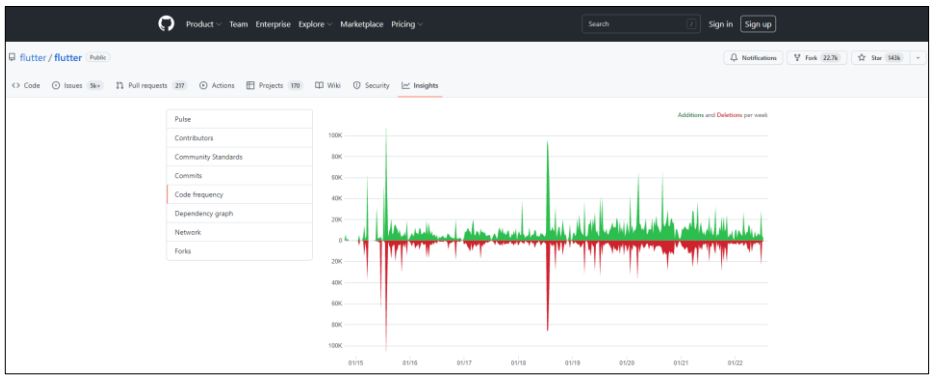
- GitHub-Android



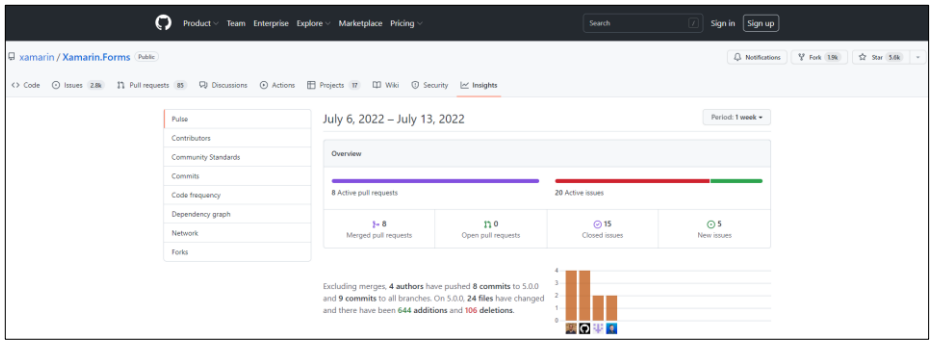
- GitHub-React Native



- GitHub-Flutter

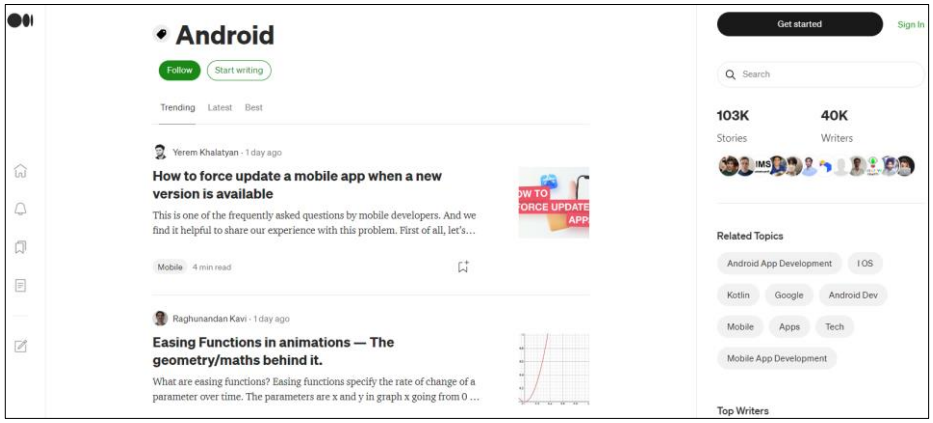


- GitHub-Xamarin

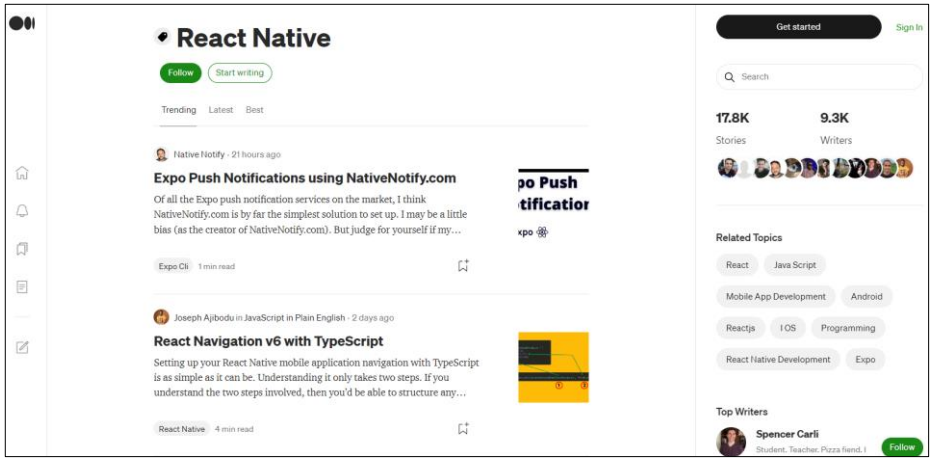


Appendix 5 Medium stories

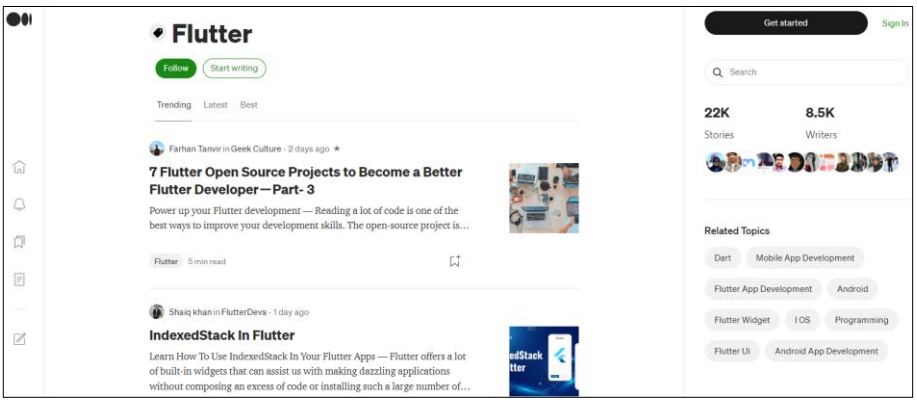
- Medium-Android



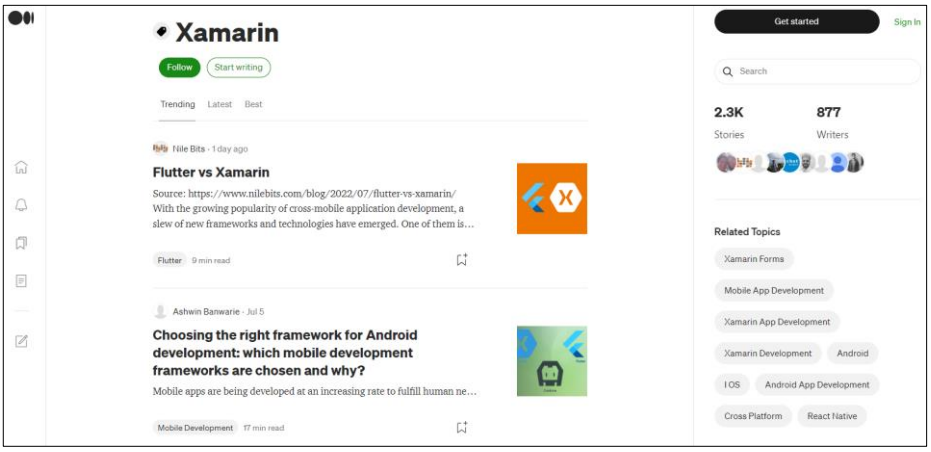
- Medium-React Native



- Medium-Flutter

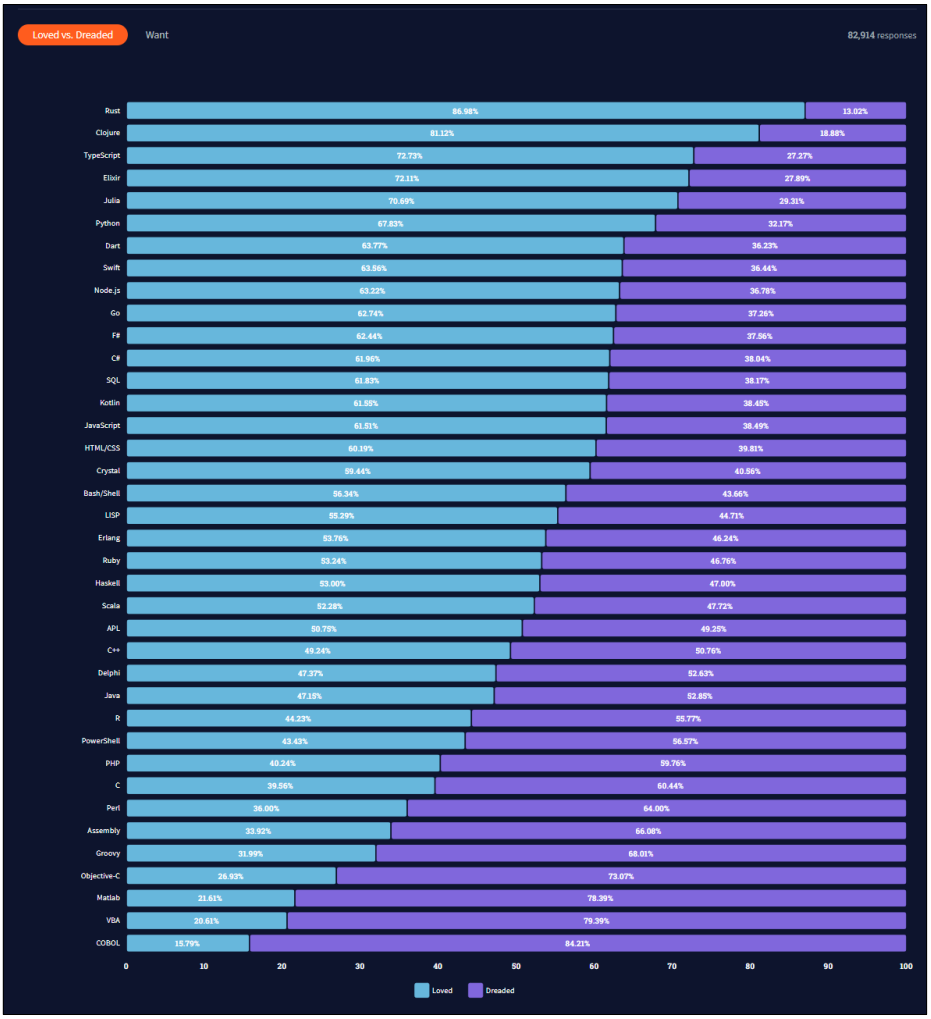


- Medium-Xamarin

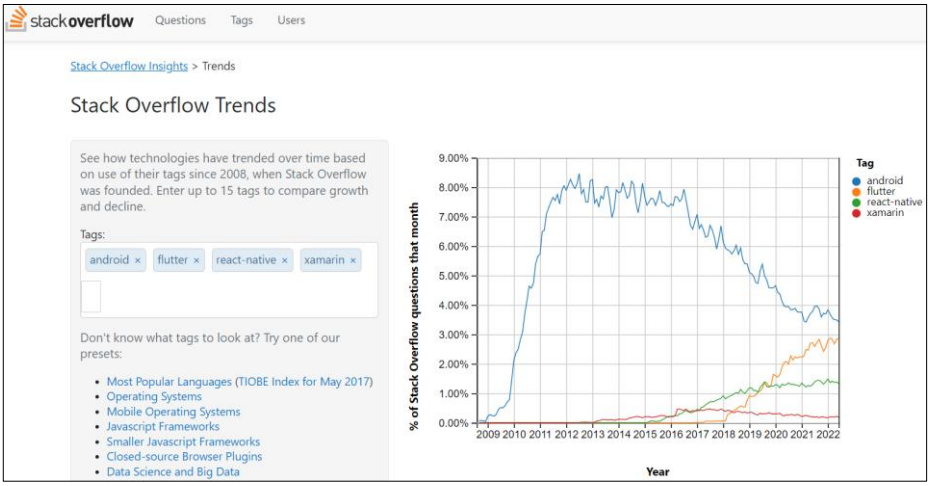


Appendix 6 Stackoverflow

- Stackoverflow-programming languages



- Stackoverflow-trends



Appendix 7 Semi-structured interview

Introduction

The interview will first start with an introduction of the interviewer and interviewee.

- I will tell the interviewee what the purpose of the research is and give a brief introduction of myself.
- The interviewee will also introduce them and explain their role within the organization.

General

- Which apps have you developed within the organization?
- For which target group you have developed these apps (B2C/B2B/Internal use)?
- How many users are using the apps?

Content

- Which processes do you go through before choosing a mobile development framework/architecture?
  Possible follow-up questions:
  - Research of similar projects
  - Research of frameworks
  - Feasibility studies
  - Use of developer's expertise
- With which mobile development frameworks have you developed these apps?
- Why did you choose this framework?

- Have you considered developing these apps in another framework (e.g., Flutter, Xamarin, etc.)?
- What are the deciding factors (criteria) that you considered when choosing the framework?
  Possible follow-up questions:
  - Performance
  - Portability of apps (supported platforms)
  - UX/UI
  - Development skills
  - Development cost (expected cost differences between frameworks)
  - Development time
  - Quality (difference in quality between the frameworks)
  - Code update
  - License cost
  - Maintenance cost
  - Security issues: Common bugs in frameworks, whether a framework is chosen based on security
- What impact does the framework have on the quality of the product and the development process?
- How long did it take to choose the most suitable framework?
- Do you have anything to add in terms of choosing a framework for app development?

Note: Possible probing questions can be asked during each question to clarify or to give an example.

Appendix 8 Survey

The full version of the survey can be found on the following pages.

Research in the field of mobile development frameworks

This survey is conducted at the Vrije Universiteit Amsterdam in the field of mobile development framework and takes approximately 5-10 minutes to complete. The title of my research is: "Choosing the right framework for Android development: which mobile development frameworks are chosen and why?". This survey is specially designed for mobile developers working with Android native or mobile development frameworks such as React Native, Flutter, Xamarin, Ionic, Cordova, or Unity.

The goal of this survey is to gain insight into the following topics:
- Experience with mobile development
- The processes before choosing a mobile development framework
- Rating of the various frameworks
- Advantages and disadvantages of mobile development frameworks

Responses from this survey will only be used for scientific purposes and will be anonymized.

Your response to this survey is highly appreciated! Any questions or comments regarding this research are welcome via the following email address: a.banwarie@student.vu.nl

*Vereist

Experience with mobile development

1. How much experience do you have with mobile development in general? *

Markeer slechts één ovaal.

- ◯ Less than 1 year
- ◯ Between 1 and 3 years
- ◯ Between 3 and 5 years
- ◯ More than 5 years

Processes before choosing a framework

2. Suppose you have to choose the most suitable framework. Which processes do you consider when choosing a mobile development framework? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Research of similar projects
- [ ] Research of frameworks
- [ ] Feasibility studies
- [ ] Use of developer's expertise
- [ ] Use of in-house guidelines
- [ ] Use of external guidelines
- [ ] Comparison analysis with previous projects
- [ ] Anders: _____

3. How long does it take to choose the most suitable framework? *

Markeer slechts één ovaal.

- ◯ Less than 2 weeks
- ◯ Between 2 and 4 weeks
- ◯ Between 4 and 6 weeks
- ◯ More than 6 weeks

4. Suppose you have to choose the most suitable framework. Which deciding factors do you consider the most when choosing a framework? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Performance
- [ ] Target platforms
- [ ] UX/UI
- [ ] Development skills
- [ ] Development cost
- [ ] Development time
- [ ] Documentation and resources
- [ ] Security
- [ ] App functionalities (features)
- [ ] Code usability
- [ ] Maintainability
- [ ] License cost
- [ ] Availability of libraries
- [ ] Anders: _____

Rating of the various frameworks

5. Suppose you would have to start a new mobile project next week. Would you consider using the following frameworks? Choose the most relevant option: *

Markeer slechts één ovaal per rij.

	Would use again	Would not use again	Interested	Not interested	Never heard
Android native	◯	◯	◯	◯	◯
React Native	◯	◯	◯	◯	◯
Flutter	◯	◯	◯	◯	◯
Xamarin	◯	◯	◯	◯	◯
Ionic	◯	◯	◯	◯	◯
Cordova	◯	◯	◯	◯	◯
Unity	◯	◯	◯	◯	◯

Experience in mobile development with Android native

This section is intended for mobile developers who have experience with Android native. If you don't have experience with Android native, choose the option "None" and you will be navigated to the following section.

6. How much experience do you have with Android native mobile development? *

Markeer slechts één ovaal.

- ◯ None *Ga naar vraag 10*
- ◯ Less than 1 year
- ◯ Between 1 and 3 years
- ◯ Between 3 and 5 years
- ◯ More than 5 years

Advantages and disadvantages of Android native

7. What are the advantages or the things that you like about Android native? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A nice development experience
- [ ] Always up to date with the latest version of Android
- [ ] Android native is a mature framework
- [ ] Android native apps work offline (no internet connection)
- [ ] Android native development is easy to learn for mobile developers
- [ ] Based on Kotlin or Java language, which is popular among mobile developers
- [ ] Documentation is excellent
- [ ] Easy to debug
- [ ] Excellent performance
- [ ] Full freedom to use features such as sensors (NFC), camera, GPS, microphone and bluetooth
- [ ] Integration with a cross-platform framework is possible
- [ ] Jetpack compose makes it much faster and easier to build android native UI
- [ ] Kotlin or Java are typed languages
- [ ] Open source and free
- [ ] Rich set of libraries
- [ ] Small file size in comparison with apps developed in cross-platform frameworks
- [ ] Testing is integrated and supported in Android
- [ ] The possibility to build excellent UX/UI
- [ ] The possibility to build fully native apps for the Android platform
- [ ] Very large (active) developer community
- [ ] Anders: _____

8. What are the disadvantages or the pain points of Android native? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Code reuse is not possible for other platforms
- [ ] Expensive as Android native development allows you to develop an app for one platform (Android platform)
- [ ] Live and hot reloading is not possible
- [ ] No flexibility as developers must code for one platform
- [ ] Time-consuming development process as Android native development allows you to develop an app for one platform (Android platform)
- [ ] Anders: _____

9. What are you using Android native for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Developing new apps from scratch 100% in Android native
- [ ] Developing proof of concept for apps
- [ ] Rebuilding apps from other frameworks to Android native
- [ ] Using Android native for a personal project
- [ ] Anders: _____

Experience in mobile development with React Native

This section is intended for mobile developers who have experience with React Native. If you don't have experience with React Native, choose the option "None" and you will be navigated to the following section.

10. How much experience do you have with React Native? *

Markeer slechts één ovaal.

- ◯ None *Ga naar vraag 14*
- ◯ Less than 1 year
- ◯ Between 1 and 3 years
- ◯ Between 3 and 5 years
- ◯ More than 5 years

Advantages and disadvantages of React Native

11. What are the advantages or the things that you like about React Native? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A nice development experience
- [ ] App maintenance and updating are simplified
- [ ] Based on JavaScript language, which is popular among web/mobile developers
- [ ] Code reuse and pre-developed components
- [ ] Cost-saving as React Native requires one team to develop apps for multiple platforms
- [ ] Documentation is good
- [ ] Relatively mature for apps that use sensors (NFC), camera, GPS, microphone and bluetooth
- [ ] Fast development process
- [ ] Good performance
- [ ] Integration with a native application is possible
- [ ] Large (active) developer community
- [ ] Live and hot reloading is possible
- [ ] Open source and free
- [ ] React Native is a mature framework
- [ ] React Native is easy to learn when you have a JavaScript background
- [ ] Rich set of third-party libraries and plugins
- [ ] Single codebase (time to market is fast)
- [ ] The possibility to build platform-specific apps with a native look and feel
- [ ] Anders: _____

12. What are the disadvantages or pain points of React Native? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Difficult to debug
- [ ] JavaScript is a dynamic language
- [ ] Large file size in comparison with native
- [ ] Slightly delayed support for the latest platform updates
- [ ] Some companies are reluctant to use React Native as it is supported by Meta (Facebook)
- [ ] Testing is not integrated into the framework. This is done by third-party tools and frameworks
- [ ] The native side of React Native can be challenging sometimes
- [ ] Anders: _____

13. What are you using React Native for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Developing new apps from scratch 100% in React Native
- [ ] Developing proof of concept for apps
- [ ] Rebuilding apps from native/other frameworks in React Native
- [ ] Using React Native for a personal project
- [ ] Anders: _____

Experience in mobile development with Flutter

This section is intended for mobile developers who have experience with Flutter. If you don't have experience with Flutter, choose the option "None" and you will be navigated to the following section.

14. How much experience do you have with Flutter? *

Markeer slechts één ovaal.

- ◯ None *Ga naar vraag 18*
- ◯ Less than 1 year
- ◯ Between 1 and 3 years
- ◯ Between 3 and 5 years
- ◯ More than 5 years

Advantages and disadvantages of Flutter

15. What are the advantages or the things that you like about Flutter? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A nice development experience
- [ ] App maintenance and updating are simplified
- [ ] Code reuse and a widget-based framework
- [ ] Cost-saving as Flutter requires one team to develop apps for multiple platforms
- [ ] Dart is a typed language
- [ ] Dart is easy to learn for developers
- [ ] Documentation is good
- [ ] Easy to debug
- [ ] Relatively mature for apps that use sensors (NFC), camera, GPS, microphone and bluetooth
- [ ] Fast development process
- [ ] Good performance
- [ ] Hot reloading is possible
- [ ] Integration with a native application is possible
- [ ] Large (active) developer community
- [ ] Open source and free
- [ ] Rich set of third-party libraries and plugins
- [ ] Single codebase (time to market is fast)
- [ ] Testing is integrated and supported in the framework
- [ ] The possibility to build platform-specific apps with a native look and feel
- [ ] Anders: _____

16. What are the disadvantages or pain points of Flutter? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Based on Dart language, which is not so popular among mobile developers
- [ ] Dart developers are limited
- [ ] Large file size in comparison with native
- [ ] Live reloading is not possible
- [ ] Slightly delayed support for the latest platform updates
- [ ] Some companies are reluctant to use Flutter because it is still new and not so mature yet
- [ ] The native side of Flutter can be challenging sometimes
- [ ] Anders: _____

17. What are you using Flutter for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Developing new apps from scratch 100% in Flutter
- [ ] Developing proof of concept for apps
- [ ] Rebuilding apps from native/other frameworks in Flutter
- [ ] Using Flutter for a personal project
- [ ] Anders: _____

Experience in mobile development with Xamarin

This section is intended for mobile developers who have experience with Xamarin. If you don't have experience with Xamarin, choose the option "None" and you will be navigated to the following section.

18. How much experience do you have with Xamarin? *

Markeer slechts één ovaal.

- ◯ None *Ga naar vraag 22*
- ◯ Less than 1 year
- ◯ Between 1 and 3 years
- ◯ Between 3 and 5 years
- ◯ More than 5 years

Advantages and disadvantages of Xamarin

19. What are the advantages or the things that you like about Xamarin? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A reasonable development experience
- [ ] App maintenance and updating are simplified
- [ ] C# is a typed language
- [ ] Code reuse and a shared .NET standard library and individual platform projects
- [ ] Cost-saving as Xamarin requires one team to develop apps for multiple platforms
- [ ] Documentation is good
- [ ] Easy to debug
- [ ] Relatively mature for apps that use sensors (NFC), camera, GPS, microphone and bluetooth
- [ ] Fast development process
- [ ] Good performance
- [ ] Hot reloading is possible with XAML
- [ ] Integration with a native application is possible
- [ ] Open source and free
- [ ] Single codebase (time to market is fast)
- [ ] Testing is integrated and supported in the framework
- [ ] The possibility to build platform-specific apps with a native look and feel
- [ ] Xamarin is easy to use if you have a C#/.NET background
- [ ] Anders: _____

20. What are the disadvantages or pain points of Xamarin? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Large file size in comparison with native
- [ ] Live reloading is not possible
- [ ] Slightly delayed support for the latest platform updates
- [ ] Small (active) developer community
- [ ] The native side of Xamarin can be challenging sometimes
- [ ] Xamarin developers are limited
- [ ] Xamarin has limited access to open-source libraries and does not support all available third-party libraries
- [ ] Xamarin is becoming less popular among mobile developers
- [ ] Xamarin is open source, but for commercial purposes, you still need to pay the license cost for Visual Studio
- [ ] Anders: _____

21. What are you using Xamarin for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Developing new apps from scratch 100% in Xamarin
- [ ] Developing proof of concept for apps
- [ ] Rebuilding apps from native/other frameworks in Xamarin
- [ ] Using Xamarin for a personal project
- [ ] Anders: _____

Experience in mobile development with Ionic

This section is intended for mobile developers who have experience with Ionic. If you don't have experience with Ionic, choose the option "None" and you will be navigated to the following section.

22. How much experience do you have with Ionic? *

Markeer slechts één ovaal.

- None *Ga naar vraag 26*
- Less than 1 year
- Between 1 and 3 years
- Between 3 and 5 years
- More than 5 years

Advantages and disadvantages of Ionic

23. What are the advantages or the things that you like about Ionic? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] App maintenance and updating are simplified
- [ ] Based on JavaScript language, which is popular among web/mobile developers
- [ ] Code reuse and offers a library of components and plugins
- [ ] Cost-saving as Ionic requires one team to develop apps for multiple platforms
- [ ] Documentation is good
- [ ] Fast development process
- [ ] Integration with Angular, React, Vue, Capacitor, or Cordova frameworks
- [ ] Integration with a native application is possible
- [ ] Ionic is easy to learn when you have a JavaScript, HTML, and CSS background
- [ ] Live reloading is possible
- [ ] Open source and free
- [ ] Rich set of third-party libraries and plugins
- [ ] Single codebase (time to market is fast)
- [ ] Anders: _____

24. What are the disadvantages or pain points of Ionic? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A poor to reasonable development experience
- [ ] Difficult to debug
- [ ] Hot reloading is not possible
- [ ] Ionic app is a mobile website rendered into a mobile app (no native look and feel)
- [ ] Ionic is becoming less popular among mobile developers
- [ ] JavaScript is a dynamic language
- [ ] Large file size in comparison with native
- [ ] Not so mature for apps that use sensors (NFC), camera, GPS, microphone and bluetooth
- [ ] Poor performance
- [ ] Slightly delayed support for the latest platform updates
- [ ] Small (active) developer community
- [ ] Testing is not integrated into the framework. This is done by third-party tools and frameworks
- [ ] Anders: _____

25. What are you using Ionic for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Developing new apps from scratch 100% in Ionic
- [ ] Developing proof of concept for apps
- [ ] Rebuilding apps from native/other frameworks in Ionic
- [ ] Using Ionic for a personal project
- [ ] Anders: _____

Experience in mobile development with Cordova

This section is intended for mobile developers who have experience with Cordova. If you don't have experience with Cordova, choose the option "None" and you will be navigated to the following section.

26. How much experience do you have with Cordova? *

Markeer slechts één ovaal.

- ◯ None *Ga naar vraag 30*
- ◯ Less than 1 year
- ◯ Between 1 and 3 years
- ◯ Between 3 and 5 years
- ◯ More than 5 years

Advantages and disadvantages of Cordova

27. What are the advantages or the things that you like about Cordova? Multiple options possible: *

Vink alle toepasselijke opties aan.

- ☐ App maintenance and updating are simplified
- ☐ Based on JavaScript language, which is popular among web/mobile developers
- ☐ Code reuse and offers a library of components and plugins
- ☐ Cordova is easy to learn when you have a JavaScript, HTML, and CSS background
- ☐ Cost-saving as Cordova requires one team to develop apps for multiple platforms
- ☐ Documentation is good
- ☐ Fast development process
- ☐ Integration with Angular, React, Vue, and Ionic frameworks
- ☐ Integration with a native application is possible
- ☐ Live and hot reloading is possible
- ☐ Open source and free
- ☐ Rich set of third-party libraries and plugins
- ☐ Single codebase (time to market is fast)
- ☐ Anders: _____

28. What are the disadvantages or pain points of Cordova? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A poor to reasonable development experience
- [ ] Cordova app is a mobile website rendered into a mobile app (no native look and feel)
- [ ] Difficult to debug
- [ ] JavaScript is a dynamic language
- [ ] Large file size in comparison with native
- [ ] Not so mature for apps that use sensors (NFC), camera, GPS, microphone and bluetooth
- [ ] Poor performance
- [ ] Slightly delayed support for the latest platform updates
- [ ] Small (active) developer community
- [ ] Some companies are reluctant to use Cordova as it is becoming less popular for mobile development
- [ ] Testing is not integrated into the framework. This is done by third-party tools and frameworks
- [ ] Anders: _____

29. What are you using Cordova for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Developing new apps from scratch 100% in Cordova
- [ ] Developing proof of concept for apps
- [ ] Rebuilding apps from native/other frameworks in Cordova
- [ ] Using Cordova for a personal project
- [ ] Anders: _____

Experience in mobile development with Unity

This section is intended for mobile developers who have experience with Unity. If you don't have experience with Unity, choose the option "None" and you can complete the survey.

30. How much experience do you have with Unity? *

Markeer slechts één ovaal.

- None
- Less than 1 year
- Between 1 and 3 years
- Between 3 and 5 years
- More than 5 years

Advantages and disadvantages of Unity

31. What are the advantages or the things that you like about Unity? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] A nice development experience
- [ ] App maintenance and updating are simplified
- [ ] Code reuse and offers an asset store for all developers to fulfill their app/game requirements
- [ ] Cost-saving as Unity requires one team to develop apps/games for multiple platforms
- [ ] Documentation is good
- [ ] Easy to debug
- [ ] Famous cross-platform framework for game development, virtual reality or augmented reality, etc.
- [ ] Fast development process
- [ ] Good performance
- [ ] Integration with a native application is possible
- [ ] Large (active) developer community
- [ ] Rich set of third-party libraries and plugins
- [ ] Single codebase (time to market is fast)
- [ ] Testing is integrated and supported in the framework
- [ ] Unity is easy to use if you have a C#/.NET and JavaScript background
- [ ] Unity supports high-quality audio and visual effects
- [ ] Anders: _____

32. What are the disadvantages or pain points of Unity? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Large file size in comparison with native
- [ ] Not open source for commercial purposes, license cost needs to be paid
- [ ] Slightly delayed support for the latest platform updates
- [ ] Unity is not ideal for non-visual apps
- [ ] Anders: _____

33. What are you using Unity for at the moment? Multiple options possible: *

Vink alle toepasselijke opties aan.

- [ ] Building new apps/games from scratch 100% in Unity
- [ ] Developing proof of concept for apps/games
- [ ] Rebuilding apps/games from native or other frameworks in Unity
- [ ] Using Unity for a personal project
- [ ] Anders: _____

34. The Unity framework has license cost (Plus $399/yr per seat, Pro $1800/yr per seat and Enterprise $4000/mo per 20 seats). Would this cost prevent you from recommending this framework for developing mobile apps/games? *

Markeer slechts één ovaal.

- ◯ Yes, these costs would prevent me from recommending Unity
- ◯ No, I would still recommend Unity
- ◯ Anders: _____