Mobilex: a generic framework for cross-platform mobile development based on web language

William Niemiec, Rafael F. Borges, Érika Cota
PPGC - Informatics Institute - Federal University of Rio Grande do Sul
Porto Alegre, Brazil
{williamniemiec,rafaeelfernandes}@hotmail.com,erika@inf.ufrgs.br

ABSTRACT

Several mobile development frameworks allow developers to program a single code to run on different types of devices and operating systems. However, the application code is dependent on the chosen framework, making it difficult to reuse it when there is a need to migrate to other frameworks. This paper proposes a generic framework for cross-platform mobile development, which uses web technologies (HTML, CSS, and JavaScript) as a basis for application development. The use of this technology to define the application and its automatic translation to a platform-specific language makes it easier to produce and maintain the application on multiple platforms. This paper discusses the proposed approach and shows, through case studies, that it allows flexibility in the choice of cross-platform development framework, not being necessary to change the code already developed in case of change in the development platform. Video presentation: https://youtu.be/ZS152tXUqBQ

CCS CONCEPTS

• **Software and its engineering** → **Software creation and management**.

KEYWORDS

Mobile application development, cross-platform application, Code Generation, Cross-Compilation

ACM Reference Format:
William Niemiec, Rafael F. Borges, Érika Cota. 2022. Mobilex: a generic framework for cross-platform mobile development based on web language. In *XXXVI Brazilian Symposium on Software Engineering (SBES 2022), October 5–7, 2022, Virtual Event, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3555228.3555274

1 INTRODUCTION

Mobile application market has grown a lot in recent years, with the expectation of reaching 520.7 billion dollars by 2030 [5]. However, developing mobile applications is not a trivial task, as it is necessary to produce specific code for each operating system. Major mobile devices use Android or iOS [8] operating system, and code generated for one cannot be reused in the other.

We can approach this problem in two ways: creating the same project for each platform in parallel or using a cross-platform mobile development framework (called CPMD hereafter). The first approach, called native mobile development, is costly because it requires specialized developers for each platform. The second approach, CPMD, requires the use of a development tool capable of generating native code for each execution platform from a single input code, which is specific to the development platform (for instance, Flutter Framework [1]). Therefore, the second approach provides savings in development and maintenance costs, as it reduces the time and cost required for developing mobile applications [2].

Since each CPMD framework has its positive and negative aspects, we may have the necessity to change one for another. This may occur either because we want to choose a CPMD that is more efficient or we want to offer the application to an operating system that the current CPMD does not has compatibility with [4]. However, changing one CPMD for another is challenging because they cause a dependency on the chosen CPMD so that, when replacing one with another, all source code will have to be redone.

This paper proposes Mobilex [2]: a generic framework for CPMD that aims at reducing/mitigating the rework required to migrate an application to a new development platform. In the proposed framework, developers write the application source code only once, using the web language as a basis, and can choose which CPMD will be used to generate mobile applications for various platforms. In this way, we make the application development code independent of a CPMD, allowing its exchange without having to change the code already produced. This facilitates the generation of applications for new platforms because it is enough to use a CPMD that produces applications on it.

In the proposed framework, we use as input a file written in the Mobilang language, created by us. It is an Extended Markup Language (XML) that uses web language to represent screens. We chose web language because it is a successful case due to its wide use in addition to being able to make the portability of web knowledge to the mobile world, thus trying to unify these two areas and expand the range of opportunities for various professionals.

This paper presents the following contributions:

- Proposal of a generic framework for CPMD that allows its exchange without changing the application code already produced;
- Use of web knowledge to develop mobile applications;
- It presents the limitations of the proposed framework and how to extend it to support new CPMD frameworks and target platforms.

[1] https://flutter.dev/
[2] https://github.com/wniemiec-mobilex

SBES 2022, October 5–7, 2022, Virtual Event, Brazil

ACM ISBN 978-1-4503-9735-3/22/10…$15.00
https://doi.org/10.1145/3555228.3555274

The paper is structure as follows: Section 2 discusses the related work. Section 3 presents the overall strategy of the framework. Section 4 details the tool developed, giving essential knowledge concerning the implementation and use of the framework. Sections 5 and 6 describe, respectively, the validation strategy and the experimental results. Finally, Section 7 concludes the paper.

2 RELATED WORK

There are several approaches in the literature to tackle the challenges of CPMD. These approaches consist of reducing development efforts and costs. Some approaches [1, 6] use Model-Driven Development (MDD) to develop mobile applications at a higher level of abstraction. Other works use other approaches, such as [7], which uses the concept of developing mobile applications based on widgets, and [3], which is compatible with more than one type of input. In [7], the developer chooses predetermined components that will be part of each screen of the application, being generated mobile applications for Android and iOS devices. On the other hand, [3] presents a framework for the generation of mobile applications which the developer can choose if he prefers to develop the application using a specific language of a platform, a set of requirements that the application must satisfy, or through an abstract template design.

Although these works reduce development time and bring several contributions, there are some challenges they do not address:

- Adding compatibility with new platforms in a easy way;
- Changing from one CPMD to another without changing the application source code.

In this work, we propose a very simple language (Mobilang) for representing mobile applications using web language as a basis while we delegate the responsibility for generating applications to CPMD frameworks. By using web language, we bring a technology that is widely known to the community to reduce the learning curve necessary for mobile development. Furthermore, by transferring the responsibility for generating mobile applications to a CPMD framework, we allow our solution to be flexible to generate applications for any platform, as long as there is a framework that performs this generation. Consequently, changing from one application type to another is simplified, as all one need to do is choose a CPMD framework that generates applications of the desired type.

3 TOOL STRATEGY

3.1 Input and Output

The developed framework receives as input an XML file written in a language - Mobilang - created along with the framework (Figure 1). This language (detailed in Section 3.2) is very simple and contains the application structure. This structure contains all application screens, which are developed in web language. Also, Mobilang is used to indicate the target platforms and the cross-platform framework to be used to generate the application. The type of application that will be generated is determined by the chosen mobile development framework. For example, if a native framework focused on generating native applications is used, a native application will be generated.

As output, mobile applications will be generated for the specified platforms (as long as they are supported by the specified framework). If they are not, a warning message informing about the non-compatibility with the specified platform is displayed and only the versions for the compatible platforms are generated.

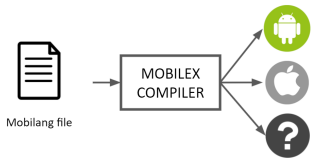


Figure 1: Framework general overview

3.2 Mobilang

Mobilang is a very simple (low learning curve) XML created to represent, by itself, the structure of screen-oriented mobile applications. Figure 2 presents the structure of a Mobilang file. Such structure is composed of screens and properties, which will be detailed below.

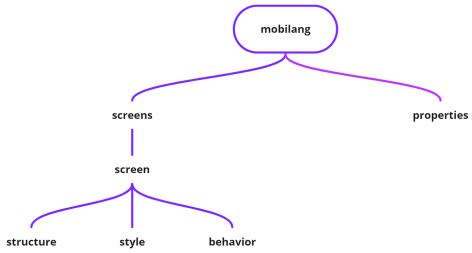


Figure 2: Mobilang structure

3.2.1 Representing a screen. The "screens" tag is where the business logic and layout of the application are. This tag will contain all the application screens. As the Mobilang language represents a screen-oriented application, it is through them that any application is represented. Inside this tag, there is a set of tags of the "screen" type, all of which must be identified by an identifier called "id". This unique identifier per screen is important for screen navigation and must start with a letter.

A screen is represented by its structure, styling, and behavior. These elements are represented using web language: HTML to define the structure, CSS to define the style, and JavaScript to define the behavior of the screen. We selected HTML, CSS, and JavaScript because they are widely known languages to describe, respectively, the structures, the style, and the behavior of a web screen and are used to provide expressiveness while taking advantage of the developer's common knowledge.

Structure defines the screen elements. These elements are the same as those used for website development, as they both use the same markup language: HTML. Also, properties of HTML tags, such as "class" and "id", can be used normally.

Styling defines how structure tag items should appear on the screen. Through CSS it is possible to create styling rules and use

selectors to indicate which tags will be affected by styling, just like the way it is used in website styling.

Finally, behavior indicates how the screen should behave throughout its existence. It is possible to use code to manipulate the Document Object Model (DOM), request data through a URL, and create events and notifications, among others. This is where one can redirect to another screen as well as retrieve received parameters. To navigate to another screen, the same code that would be used to redirect the user to another page on a website can be used to move to another screen in the application. For example, a command that can be used is "window.location.href", where the programmer informs the URL that he wants to redirect the user to. In this case, instead of being a URL or an HTML file, the screen directive must be used (Table 1), which follows the format "mobilang::screen::<name_screen>" (as shown in Figure 3). Finally, to pass parameters from one screen to another, the concept of "query" is used. It must indicate the name of the screen, a question mark, and, for each parameter: "name_param=value_param". If there is more than one parameter to be passed, separate them by "&" (Figure 3).

```
window.location.href = "/watch?v=foo&t=123"                  // Web
window.location.href = "mobilang::screen::watch?v=foo&t=123"  // Mobilang
```

Figure 3: Mobilang screen redirection

3.2.2 Defining properties. On the other hand, the "properties" tag contains the application definitions. These definitions are all set through a key-value pair inside the 'properties' tag, using the JavaScript Object Notation (JSON) format for this. The properties are diverse, such as the name of the application, its version, the framework to be used to generate the application, and the location of the assets. Several properties can be defined, like author information, assets location, application information, and framework's name to be used for generating applications.

3.3 Mobilang directives

In order to identify specific commands for mobile applications, such as screen switching or accessing device resources, directives were defined to be used within the structure and behavior tags. The structure of a directive is represented in the Figure 4. It is composed of two parts: a prefix, indicating that it is a directive, and a content. The prefix always starts with "mobilang::", and the content is made up of two parts: the type of the directive and its respective value. The complete list of directive types as well as accepted values can be found in the Table 1.

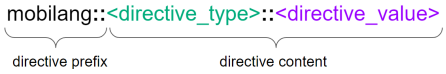


Figure 4: Anatomy of Mobilang language directives

4 TOOL IMPLEMENTATION

In this section, we present the Mobilex Compiler, the implementation of the approach described in Section 3. Bearing in mind that the compiler must be compatible with any mobile development framework, we created two internal compilers, one responsible for generating the Abstract Syntax Tree (AST) and the another one for generating mobile applications through a mobile development framework. Next, we will detail the main implementation decisions of the solution. Finally, a simple example (an application with one screen) is available in our public repository called "examples" [3]. This example contains only the minimum necessary to build an application.

Directive type	Directive value	Description
screen	[A-z]+[A-z0-9-_]*	Redirects to screen with id equal to the given value.
param	[A-z]+[A-z0-9-_]*	Gets the value of a parameter passed to the screen. The value must be the name of this parameter.
reload	screen	Reload current screen.
input	[A-z]+[A-z0-9-_]*	Get the current value of an input.

Table 1: Mobilang language directives

4.1 Mobilex Compiler

The Mobilex compiler [4] (Figure 5) is the main component of the framework. It was built with Java and is composed of two parts, the first being framework-independent, and the second framework-dependent. The first part of the process is composed of the MAST compiler (Section 4.1.1), which is responsible for generating the AST from the Mobilang file. The second part of the Mobilex compiler structure is formed by the AMA compiler (Section 4.1.2). This compiler receives the previously generated AST as input and generates mobile applications according to what is specified in the AST. For that, this compiler calls a third-party mobile development framework.

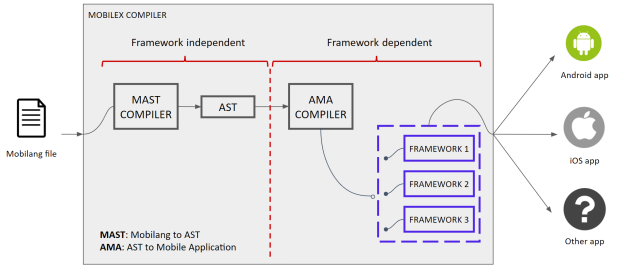


Figure 5: Mobilex compiler architecture

4.1.1 MAST Compiler. Mobilang to AST compiler [5] (Figure 6) is a framework-independent compiler whose objective is to generate an AST from a Mobilang file. In this compiler, the syntax rules of the Mobilang language is defined in addition to its semantics. Its

[3] https://github.com/wniemiec-mobilex/examples/tree/master/simple-app
[4] https://github.com/wniemiec-mobilex/mobilex-compiler
[5] https://github.com/wniemiec-mobilex/mast-compiler

architecture consists of four modules: syntax, semantics, export, and data.

The syntax module is responsible for defining the Mobilang syntax. It is built with Flex [6], using C language, and follows the structure detailed in Figure 2. The semantics module defines the Mobilang semantic, parsing the input file and generating its AST as output. For that, it is used Bison [7] and also uses C language. Also, we use JavaScript and Node [8] for generating AST of HTML, CSS, and JavaScript content from Mobilang. The export module exports AST generated in the semantics module to Graphviz format [9] and uses C++ language. Finally, the data module contains data structure and objects.

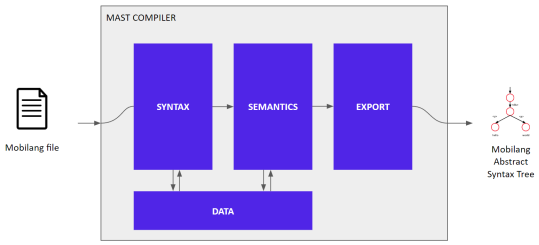


Figure 6: MAST compiler architecture

4.1.2 AMA Compiler. AST to Mobile Applications compiler [10] (Figure 7) is a framework-dependent compiler whose objective is to generate mobile applications from an AST generated from a Mobilang file. For this, it must use a CPMD framework. However, to use this framework, AMA compiler would need to know how to generate mobile applications from these frameworks. To circumvent this problem, we define an interface that must be implemented if the programmer wants to support a framework that the AMA compiler is unaware of. However, this is our responsibility (to provide compatibility with several CPMD), and from the developer's view, he/she usually does not need to worry about it (only if he/she wants to use a CPMD that AMA compiler does not have compatibility yet). This interface is described in Figure 8, where "Properties" and "Screen" represent data from tags with the same name in the Mobilang language, while "Project" contains the application project dependencies created by the CPMD framework along with application codes. Finally, it was built with Java and its architecture consists of six modules: models, reader, parser, coder, export, and framework.

The models module contains data structure and objects. The reader module receives the Mobilang AST file and converts it into JSON objects. The parser module is responsible for parsing JSON objects generated in the reader module and converting them into modules. The coder module is responsible for generating specific code to a CPMD framework from modules generated in the parser module. The export module generates mobile applications generated in the coder module using the same CPMD framework. Finally, the framework module is responsible for handling code generation and mobile applications using a specific CPMD framework. For that, it is necessary that the interface described in Figure 8 was already implemented for the chosen CPMD.

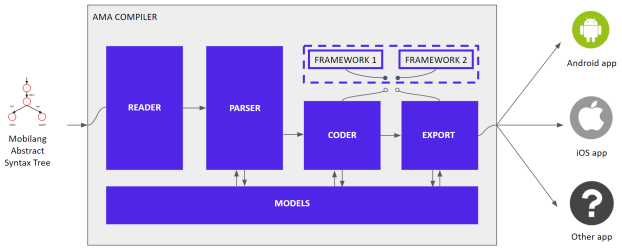


Figure 7: AMA compiler architecture

5 VALIDATION METHODOLOGY

Our goal is to provide a tool that makes it possible to generate mobile applications using web language, in addition to allowing any CPMD framework to be chosen without having to change the code already developed. Thus, in addition to the typical verification strategies used during implementation, we performed an experimental validation experiment to evaluate our tool and that it fulfills its objectives. To this end, we evaluated the following research questions (RQs):

> ***RQ1:*** *Is it possible to generate applications for different mobile platforms using Mobilang?*
>
> ***RQ2:*** *Is it possible to generate mobile applications using different mobile development frameworks without changing the language used to develop the application logic?*

To perform the analysis, we had to choose mobile applications to respond to RQ1 and choose mobile development frameworks to

```
1  public interface Framework {
2      void createProject(
3          Properties properties, Path location
4      ) throws IOException;
5
6      void addProjectDependency(
7          String dependency, Path location
8      ) throws IOException;
9
10     Project generateCode(List<Screen> screens)
11     throws CoderException;
12
13     void generateMobileApplicationFor(
14         String platform, Path source, Path output
15     ) throws AppGenerationException;
16 }
```

Figure 8: Interface that must be implemented so that the AMA compiler knows how to handle a CPMD framework

[6] https://www.gnu.org/software/flex
[7] https://www.gnu.org/software/bison
[8] https://nodejs.org/
[9] https://graphviz.org
[10] https://github.com/wniemiec-mobilex/ama-compiler

respond to RQ2. We have defined a strategy for both, each of which is described below.

To answer RQ1, we will work with two mobile applications. To choose which ones they will be, we conducted a survey on mobile applications and defined the following inclusion criteria:

(1) the application must be available for both Android and iOS;
(2) the application must perform some communication with the database;
(3) the application must be free;
(4) the application must have at least two screens.

Based on these criteria, we selected two mobile apps: WhatsApp [11] and Booking [12]. WhatsApp emerged as an alternative to the SMS system. It provides a simple, secure, and reliable messaging and calling service for cell phones all over the world. Booking is one of the largest travel e-commerce companies in the world. It allows you to book accommodation, flights, and car rentals.

On the other hand, to answer RQ2, we need to choose mobile development frameworks. Our goal is to work with two and, to make the choice, we define the following inclusion criteria:

(1) the framework must generate mobile applications that are accessible on both Android and iOS devices;
(2) the framework must be cross-platform, that is, be functional in different desktop operating systems;
(3) the framework must be among the five most used for mobile development.

With these criteria, we selected two frameworks: React Native [13] and Ionic [14]. Based on React (a JavaScript framework for web development) React Native enables the creation of native multi platform mobile applications. On the other hand, Ionic is an Open Source Framework used for the development of hybrid mobile applications.

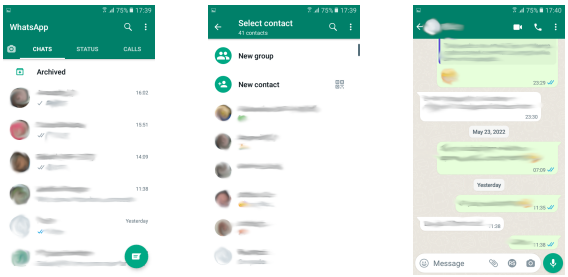
6 EXPERIMENTAL RESULTS

In this section, we present and analyze the experimental results with respect to the established research questions. The analyses are presented in two parts: 1) the results of the experiments conducted to answer RQ1 and 2) the results of the experiments conducted to answer RQ2.

6.1 Analyzing RQ1

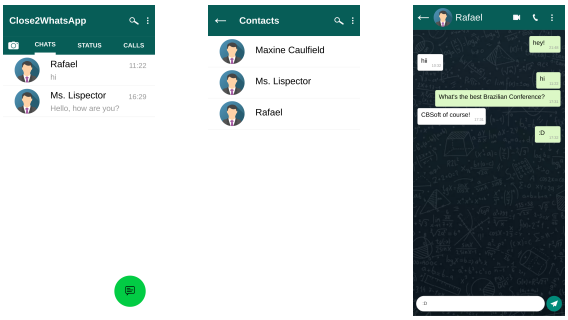
We developed code using web technologies to simulate the development of WhatsApp and Booking applications in our framework. We will call these apps "Close2WhatsApp" and "Close2Booking" respectively. We will then generate mobile apps for Android and iOS through our platform, using the cross-platform React Native mobile development framework. It is worth mentioning that not all the application's features were developed because this experiment does not aim to create a replica of the applications, but to show that it is possible to recreate them in the framework.

6.1.1 Close2WhatsApp. We have implemented three WhatsApp screens: the messages screen, the new conversation screen, and the chat screen (Figure 9). We developed an Application Programming



(a) messages screen (b) new chat screen (c) chat screen

Figure 9: WhatsApp



(a) messages screen (b) new chat screen (c) chat screen

Figure 10: Close2WhatsApp

Interface (API) to communicate the application with a database that we created to store the conversations. After developing the code using web language, we put it in a Mobilang file following the grammar defined in Section 3.2 and generate mobile applications. The results (Figure 10) were very satisfactory: both for Android and iOS, is possible to chat with another person by text in the same way that it is done in the original application.

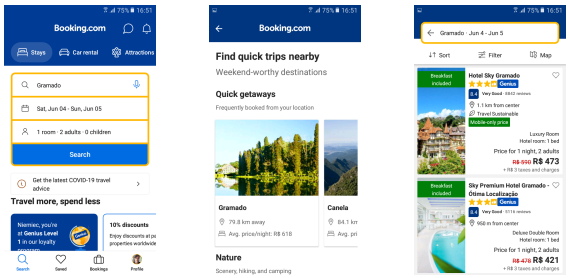
6.1.2 Close2Booking. We decided to implement three Booking screens: the home screen, quick trip, and stays screen (Figure 11). As in Section 6.1.1, we have also developed an API so that the application can obtain data from a database created to store the places and their stays. We managed to obtain excellent results (Figure 12), with the navigation being identical to the one that the original application has.

6.2 Analyzing RQ2

We carried out a study to understand how the React Native and Ionic frameworks work. Our goal was to get to know these frameworks better to implement the interface defined in Section 8 and make it compatible with our framework. Next, we will comment on the results of this experiment.

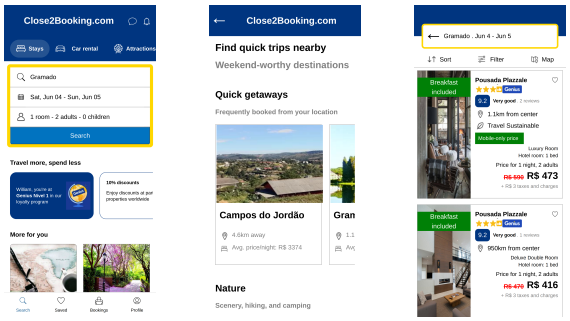
6.2.1 React Native. Supporting the React Native framework was easy with the approach we used. Instead of directly converting

[11] https://www.whatsapp.com
[12] https://www.booking.com/
[13] https://reactnative.dev
[14] https://ionicframework.com



(a) home screen (b) quick trip screen (c) stays screens

Figure 11: Booking



(a) home screen (b) quick trip screen (c) stays screen

Figure 12: Close2Booking

screen elements ("structure", "style" and "behavior" - Section 3.2) into framework elements, we chose to use a native container that accepts web language, called "WebView". After that, we produce the applications already shown in RQ1 (Section 6.1). The applications will work almost as expected, with some caveats. The first is that the transitions were abrupt from one screen to another. Furthermore, if the user clicks the back button on the cell phone, the application closes (as would be expected if the application had only one screen). We analyzed this and realized that the cause of them was due to our initial decision to use "WebView". This decision made the process of making the framework compatible with ours faster, but, as it does not use its elements of the framework, it ends up with these defects.

6.2.2 Ionic. The implementation of the Framework interface was relatively smooth. Some processing was necessary, such as converting functions to arrow functions and assigning tag events in a script of its own, among others. After implementing Ionic support, we generate the applications developed in RQ1 (Section 6.1) and analyze whether the generated applications work as expected. We verified that the generated applications, both for Android and iOS, worked exactly as expected. As with React Native (Section 6.2.2), some transitions and animations are not the same as the original application, but it is something that can be improved in future versions of our framework.

7 CONCLUSION AND FUTURE WORKS

Producing applications for each mobile platform is a costly task. Many approaches have been developed to reduce this cost, like CPMD frameworks. But they make the developed code dependent on their frameworks and, if at some point one chooses to migrate to another framework, all the work done must be redone in the other one. In this paper, we propose a framework that removes this dependency: the application code is developed once and mobile applications are generated through a CPMD framework. If one wants to change the framework, this can be done without having to change anything in the application code. In addition, the language used for application development is the same as those already used on the web, reducing the learning curve necessary to develop mobile applications. Finally, our solution is publicly available under a MIT license.

In future works, we intend to improve support for React Native and Ionic frameworks, as well as add support for other frameworks. Finally, we intend to carry out further optimizations and allow the input file to be modularized into other files to improve its maintainability.

ARTIFACT AVAILABILITY

Worked use cases to answer RQ1 are available in a public repository hosted on GitHub titled "examples" [15]. Furthermore, the results of RQ2 are available in the same repository, while the framework implementation code is located in the AMA compiler repository [16].

ACKNOWLEDGMENTS

This work is partially supported by Propesq-UFRGS, Prorext-UFRGS and CNPq, Brazil.

REFERENCES

[1] J Z Blanco and D Lucrédio. 2021. A holistic approach for cross-platform software development. *Journal of Systems and Software* 179 (2021), 110985. https://doi.org/10.1016/j.jss.2021.110985

[2] Clutch. 2017. How Much Does It Cost to Develop an App: 2017 Survey. https://clutch.co/app-developers/resources/cost-build-mobile-app-survey. Accessed: 2022-29-05.

[3] Wafaa S El-Kassas, Bassem A Abdullah, Ahmed H Yousef, and Ayman Wahba. 2014. ICPMD: Integrated cross-platform mobile development solution. *2014 9th International Conference on Computer Engineering Systems (ICCES)*, 307–317. https://doi.org/10.1109/ICCES.2014.7030977

[4] Lamia Gaouar, Abdelkrim Benamar, and Fethi Tarik Bendimerad. 2016. Desirable requirements of cross platform mobile development tools. *Electronic Devices* 5 (2016), 14–22.

[5] Globe News. 2022. The Worldwide Smartphones Industry is Expected to Reach $520.7 Billion by 2030. https://www.globenewswire.com/en/news-release/2022/04/04/2415593/28124/en/The-Worldwide-Smartphones-Industry-is-Expected-to-Reach-520-7-Billion-by-2030.html. Accessed: 2022-29-05.

[6] Christoph Rieger, Daniel Lucrédio, Renata Pontin M Fortes, Herbert Kuchen, Felipe Dias, and Lianna Duarte. 2020. A Model-Driven Approach to Cross-Platform Development of Accessible Business Apps. , 984-993 pages. https://doi.org/10.1145/3341105.3375765

[7] Zhaoning Wang, Bo Cheng, Ying Jin, Yimeng Feng, and Junliang Chen. 2017. Poster: EasyApp: A Widget-Based Cross-Platform Mobile Development Environment for End-Users. *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 591–593. https://doi.org/10.1145/3117811.3131242

[8] Murat Yesilyurt and Yildiray Yalman. 2016. Security threats on mobile devices and their effects: estimations for the future. *International Journal of Security and Its Applications* 10, 2 (2016), 13–26.

[15] https://github.com/wniemiec-mobilex/examples

[16] https://github.com/wniemiec-mobilex/ama-compiler