Development of mobile app through React Native hybrid framework

Shpetim Kadrija
Faculty of Natural Sciences and Mathematics
University of Tetova
Tetova, North Macedonia
kadrijashpetim@gmail.com

Agon Memeti
Faculty of Natural Sciences and Mathematics
University of Tetova
Tetova, North Macedonia
agon.memeti@unite.edu.mk

Shkurte Luma-Osmani
Faculty of Natural Sciences and Mathematics
University of Tetova
Tetova, North Macedonia
shkurte.luma@unite.edu.mk

Abstract—**Mobile apps often include methods and functions that collect and process sensitive user data. Anyway, when utilizing them, we usually encounter certain drawbacks. Therefore, the aim of this paper is to present a native app, as well as emphasize the necessary reasons for developing such application, it's main purpose, the technologies used to develop this application, the limitations and benefits, system capabilities and necessary conditions for manipulating this application. The Stellar Mobile App will be available to every user of mobile devices, regardless of the operating system in which it operates, or regardless of the device.**

Keywords-mobile app; android; iOS; react native; JavaScript; expo; REST API

I. INTRODUCTION

The main issue that needs to be considered when starting a project, no matter how voluminous, is identifying a key goal. Therefore, before starting to implement an idea to create an application that would deal with some of the daily obligations in human's life, firstly we started researching about the application market mainly in the territory of the Republic of North Macedonia. One thing that was noticed, is that apart from different banking applications which do not perform all the functions of electronic payments, there cannot be found a mobile application which includes PayPal or Sandbox payments. Consequently, this was also the main reason which motivated us to create a function within the application to perform this type of service. The other reason for creating this note and to-do list application, was because according to statistics in the top 10 most used mobile applications, there is a lack of apps for storing various notes, especially in our country.

The different horizon that this application offers allow us to show some of its main purposes:

- Maintaining the daily itinerary
- Facilitate the storage of sensitive data
- Quick access to data
- Electronical Payment
- Privacy in data storage

Considering how this app tries to keep some of the most sensitive elements of users under control, its non-functional requirements are also up to date. Its presentation is mostly fast, no-delays on callbacks and many features which will be elaborated further on the paper. All this makes this application desirable in the market, without implicating any shortcomings in its main functions.

II. STATE OF THE ART

Mobile technology has pushed the human race to develop and adapt. Several authors [6] have listed the key components of mobile usability models as regarding three main points of view that include: efficiency satisfaction, and effectiveness, whereas other researcher have specified the development approaches that are available when developing a business mobile app [7]. A comparison approach among native and hybrid mobile apps that are built on JavaScript is provided in [8]. The assessment is done based on seven pertinent principles to the mobile app developments. However, conclusions show that React Native exhibits the best outcomes. When taking in consideration the operating systems, [9] has figure out that users are much more likely to see the combination of Java and Eclipse than Objective-C and Xcode. Researchers [10] consider the data layer which is seen as a critical task for mobile apps that need constant access to remote data.

978-1-6654-6828-2/22/$31.00 ©2022 IEEE

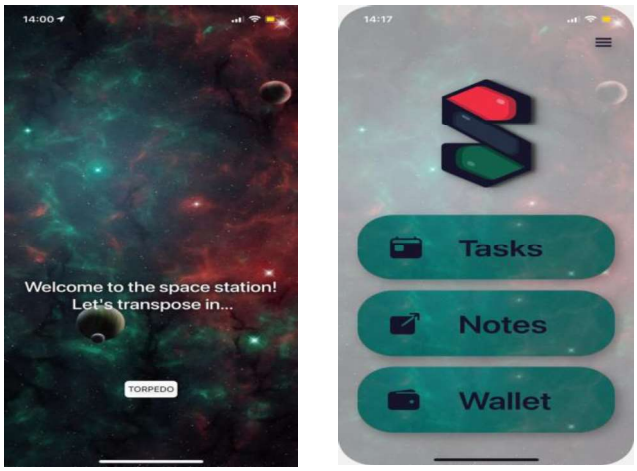

Figure 1. Splash and Profile change screen

III. TOOLS AND TECHNOLOGY USED

The introduction of multiple technologies that perform different tasks, is undoubtedly an indisputable development, but it makes even more complicated the creation of applications, when only one person comes to the fore. To create such a system, were utilized different technologies as noted below:

- Visual Studio Code
- Expo
- React Native
- MySQL
- Babel Javascript
- PHP

A trend in the development of applications that deal with scripting languages, such as JavaScript [1], is mostly used Visual Studio Code, which comes with a lot of space to install various extensions and allows us to write any code possible and compile it.

To create a native mobile app, it is necessary to be encoded in Android Studio for native users specifically created for Android, and in XCode to create apps for iOS, therefore, the app presented in the paper is created in a hybrid framework, such as React Native, which is a framework written by Facebook and has JavaScript [1] as its main language, and enables us to create mobile applications for both Android and iOS operating systems.

Expo is presented as a connecting bridge which enables us to test and see the errors during the debugging of the system in real time, connecting it to any simulator or even through QR Code, on real devices [2][3].

MySQL was used to store data and manipulate it, but this would not have been possible without the bridge between Back-End and Front-End with technologies like PHP. The interface of the mobile app is presented in figure 1.

IV. SYSTEM LIMITATIONS

If we were to consider even the most complex applications created by different teams, such as: Facebook, Instagram, Documents, etc., we would definitely encounter different limitations. Hence, we cannot lobby that there is any application which is not limited in terms of the methods it performs. This brings us back to the developed application, by mentioning some of the services which are limited but of course, the reasons are different.

A major limitation is the interconnection of the application with other applications, this cannot be real at the moment because all other applications use closed REST API's. Also considering that PayPal is not legal in our territory, and there is no point where we can register or even communicate about our services, to receive our remittances from PayPal again we have to be physically in one of the banks which has taken on the responsibility of being the link between customers and PayPal.

Other limitations that occur during the use of the system we can say that definitely during the registration of a client there must be a person who will perform some services in the Back-End of the software, such as:

• Automatic registration of transaction points

• Change of token - password, send by SMS

• Data manipulation in case of any deletion operation by mistake.

V. PROJECT DESCRIPTION

A. System Analysis

The first stage of software development is the project analysis process. The importance of the project analysis phase is the creation of requirements for the system to be developed, and installed by users. Analyzing the project to understand its complexity forms the vital part of studying the system. Problem areas are identified and information is collected. Finding facts or gathering information is essential to any claim analysis. System analysis involves investigating and possible changes to the existing system. At the end of the analysis, the system description and the set of requirements for a new system follow. Development begins with defining a new system model and continues with the advancement of this model into a working system. The system model shows what the system must do to meet these requirements. Finally, data forms are converted into a database and processed into user procedures and computer programs.

B. *Project Milestones*

The time it takes to develop an application all depends on the requirements of the clients, whereas it takes an average of about three months to develop a software application for smart phones, not counting application maintenance. The developed application has encountered some delays in forecasting milestones, because it was needed to include different libraries in order not to lose its quality and efficiency.

TABLE I. PROJECT MILESTONES

Milestones	Deliverables	Date
Project Proposal	Defining our ambitions, creating a document containing projects' achievements at the very end	One week
Requirement Analysis	Writing the SRS Document, this happened to be discussed between myself!	One week
Structure Design	Breaking the project in small pieces	Three to four weeks
Coding	Actually, writing the methods, functions, variables, constants of the system!	Three to four weeks
Demo	Creating a endpoint, where I came to ask some friends what they think about it, changing some parts	Two to three days
Review	Watching how the system reacts to different I/O	One week
Testing	Making willingly cases with a lot of mistakes when using the system	One to two weeks
Final Product	Installing the product, publishing it to the Expo Framework	One day

C. *Risk Management*

At this stage we take into account any risks that may threaten our system at the time of its development, analyze them and keep them under surveillance. We have also encountered risks that were not predictable in case of any "catastrophes" in different frameworks, ie those that we would not be able to control.

Each project has its own risks and their forecast is very normal, at least we can be aware of the unexpected and have basic information on how to find the right solution, in a word to adhere to the consequences that may come from the risks.

TABLE II. RISK MANAGEMENT

Risk	Effect	Solution
Bad Timing	Disrupts the flow of the process	Take care of the duration of processes from the beginning
Incorrect Budget Estimation	Cannot find new resources when needed	Try to find the basics, which won't let the software fail
Poor Code Quality and Technical Risks	Testing will fail, eventually the whole app itself	Test the programmers before hiring
Unpredictable External Risks	Emotions, poor management	Find creative solutions!

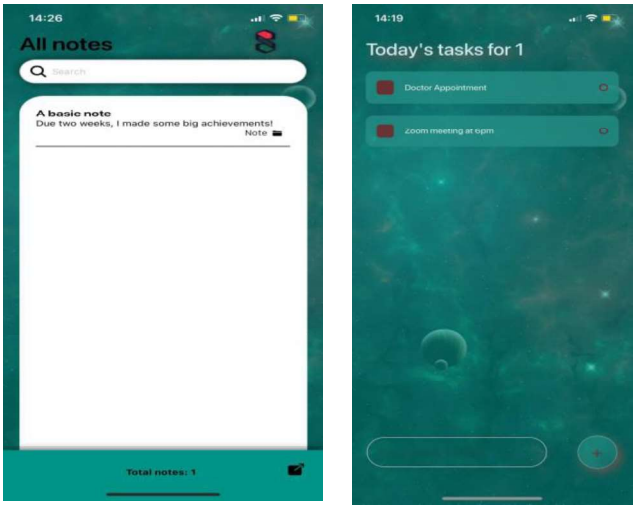

Figure 2. Notes and To Do List screen

VI. REQUIREMENTS SPECIFICATION

Analyzing or specifying system requirements is the first and very complex step in the project development and management process. The community of requirements is the prediction of user needs in our application that aims to understand the problems and how they will be solved.

Always, before we start implementing any project, the main point that needs to be completed is the phase of specifying the functional and non-functional requirements that constitute the complete specification of the functionalities that the application will contain.

Below we will present the functional and non-functional requirements that our application will contain, its structure and the flow of activities to perform specific operations in the application. To see what a successful project would look like before it was completed, we can look at the application documentation and analysis process. Requirements are some of the key details that describe the features of the software product.

A. *Functional Requirements*

Knowing that this application will be at the service of different users, we will present the functions that they will be able to perform.

- Downloads the app from the App Store or Play Store
- Install the application
- Login to the system, there is only one button, which sends us to the first subpage
- The first subpage is about logging in to the customer's private profile

- If we do not have an account, with a swipe, we can go to the registration section with personal account, entering data, such as: email, password, username
- Once we have completed these procedures, we log in to the system
- Presentation of the main menu where we have to-do lists, notes and wallets
- The to-do list presents us with the daily obligations that we write down, or delete if we have completed them
- Notes are intended to enable us to keep records regarding certain tasks, or different data
- We can delete notes, mark new and distribute certain ones
- Also presents a total number of notes contained in our application
- Finally, we have the wallet, which shows us the balance of our transaction account, the last transactions are presented in a chart, the total and detailed number of transactions.
- The rest of the wallet contains a subpage which allows us to perform online transactions through PayPal and Sandbox (for testing).

B. *Non-Functional Requirements*

The very name of this process represents the definition of the requirements required by our system, but they do not perform functions, so in a word they are requirements that do not require logical or mathematical performance. We will list them below:

- Network access
- The main themes of the application we used
- Different animations that appear on the subpage
- Icons used
- Security permits
- Site use permits
- Contacts permits
- The capacity of the system is not aggravated depending on the number of users at the same time
- Server with maximum capacity for downloads and requests up to API Calls.

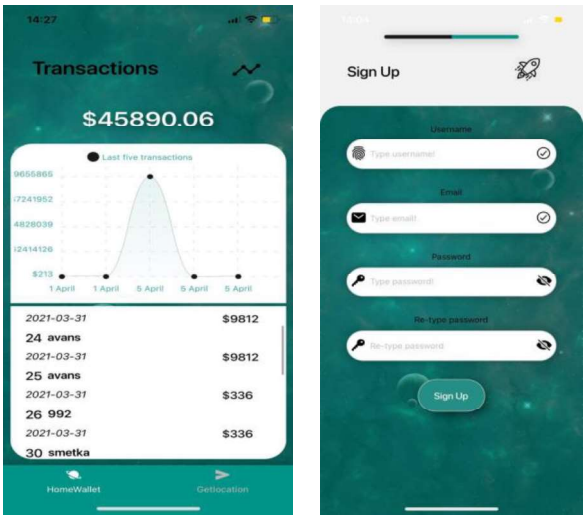

Figure 3. Balance and Sign-Up screen

C. *Logical Requirements*

To understand the logic of how the various operations and methods of the system are perceived, we will present the E-R Diagram of this system. This E-R Diagram is the starting point from which the database for our application is built, then the stage of creating the methods we used in the application has flowed from the way the data flows in this database.
The E-R Diagram presents a description of all the actors participating in this system, their relationship with different objects, the use of different methods, and the events in the system.

D. *Data Flow Diagram*

Also known as DFD, Data Flow Diagram is a diagram which is graphically represented to determine the flow of data in an information system. DFD describes the processes that are involved in a system to transfer data from different inputs to the database and to generate reports based on the required results.

Data Flow Diagram can be divided into logical and physical. Logical DFD is a diagram that describes the logical side of a data stream where certain system functions are performed. While physical DFD describes the implementation of data flow logic.

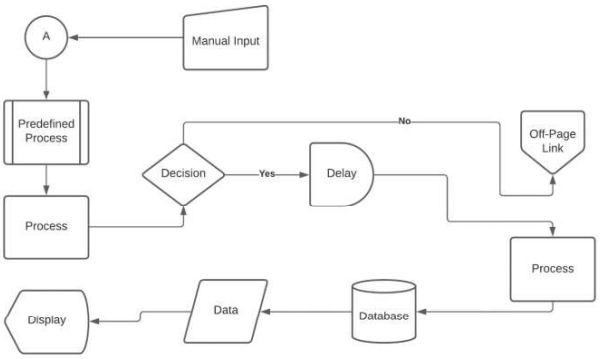

Figure 4. Data Flow Diagram

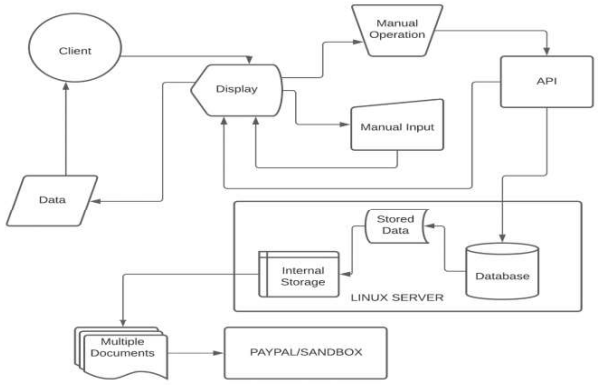

Figure 5. System Design

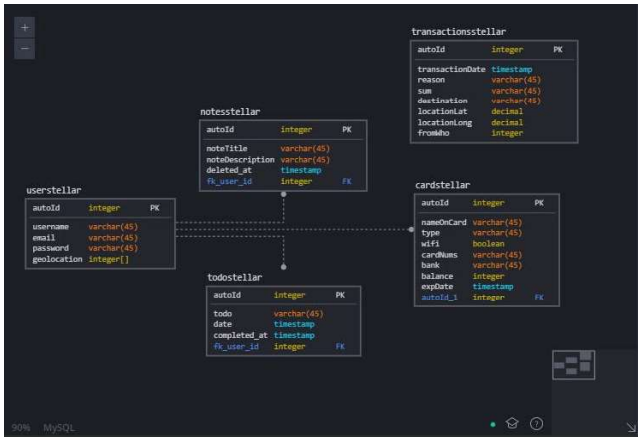

Figure 6. MySQL - DB design

VII. MAINTENANCE

There can be no software or system so perfect that there are no errors. In the maintenance phase the errors are just the beginning of tireless work. All new problems that will be detected while using our application, will be reported to bugs / technical support that is automatically provided by the Play Store or App Store as an option. Also, to attract other users, this system needs to be ready at any moment to incorporate new features, which will make it more attractive in the market.

A. *Maintenance Importance*

Software maintenance is presented as the most important part in developing perfect systems. This section is about all the modifications and updates that will be made to the system after it is published. There are several reasons why modifications are needed which we will present below:

• Market - market protocols, which will vary depending on the success of our application, we may have additional costs.

• User requirements - over time users may request additional features in our system

• Hosting Modifications - if in any case the platform protocols where we have located our application server change, we must definitely apply the changes to our software

• Change of organization - in case we would sell some rights of our software to specific organizations, in order to finance it, and we would be asked to make changes.

B. *Types of maintenance*

In the software development lifecycle, the types of maintenance differ from the nature of the system we create. There may be some small changes in order to correct the errors or even larger ones to change the application volume as a whole. Below we will list some of the most universal we can encounter [5]:

- Maintenance to improve - modifications and updates will occur in such cases where we are required to fix problems, which come from different reports from the behavior of the system with different users.
- Adaptive maintenance - here modifications will occur in case we want to update or add other parts to our system, to make it more attractive.
- Perfection maintenance - here as usual it is more about maintaining and enhancing performance. This part is more present when we want our system to be used for a long period of time.
- Preventive maintenance - this system maintenance is done in such a way as to foreseen the problems in the future, to remove some of the warnings that are displayed to us, and not to bypass them.

C. *Maintenance cost*

According to a study [5], the cost of maintaining a software system appears to be as much as 67% of all different software costs from scratch.

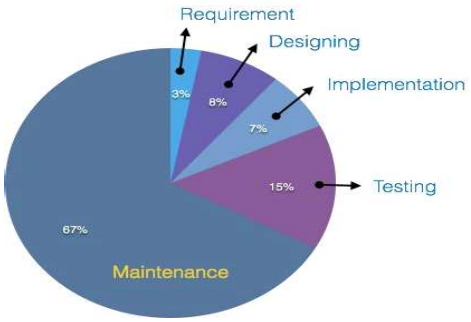

Figure 7. Software maintenance

Factors leading up to this cost are usually real-life factors like [5]:

- The older the application gets, the more problems we will encounter during maintenance because technologies evolve and the introduction of new forms of operating systems may not correspond to the codes we have written before.
- With the development of technology, it becomes more costly to maintain old software.
- Many maintainers are young and inexperienced in this regard
- Some of the changes may disrupt the overall structure of the application
- Undocumented changes may have consequences in the future
- Software program structure and Programming language
- Dependence on other environments or libraries which stop the pace of their development
- Dependence on irresponsible or inexperienced staff.

VIII. CONCLUSION

Concluding, we can note that Stellar as a completed application can be calculated as a contemporary and in line with the requirements of users who want to have as much control over their tasks or obligations. As an application that deals with simple users and that is for personal needs, it comprises of a user-friendly interface.

As for the requirements and functions that have been emphasized, they all are in line with the latest technology, their realization with maximum efficiency while guaranteeing speed and quality for users. Safety and maintenance are also on par with all technological advances that may occur in the future. As a software with the first version, it may appear simple, but in the future the claim is to connect to other parts, depending on the needs of users.

REFERENCES

[1] Eisenman, B. "Learning React Native: Building Native Mobile Apps with JavaScript", O'Reilly Media, 2016

[2] Learning Expo – Documentation - https://docs.expo.dev/tutorial/follow-up/ [Accessed 15 February 2022]

[3] Errors and debugging - https://docs.expo.dev/get-started/errors/ [Accessed 15 February 2022]

[4] Antani, V., Timms, S., Prusty, N., "JavaScript: Moving to ES2015", Packt Publishing, March 2017

[5] Software Maintenance Overview, https://sceweb.sce.uhcl.edu/helm/WEBPAGES-SoftwareEngineering/myfiles/TableContents/Module-13/software_maintenance_overview.html [Accessed 15 February 2022]

[6] Harrison, R., Flood, D. & Duce, D. "Usability of mobile applications: literature review and rationale for a new usability model". J Interact Sci 1, 1 (2013). https://doi.org/10.1186/2194-0827-1-1

[7] Saratchandran, V. "Choosing the Right Mobile App Development Approach for Your Business", October 2019, https://medium.com/techies-toolkit/choosing-the-right-mobile-app-development-approach-for-your-business-204dbcc34093[Accessed 14 January 2022]

[8] H. Brito, A. Gomes, Á. Santos and J. Bernardino, "JavaScript in mobile applications: React native vs ionic vs NativeScript vs native development," 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), 2018, pp. 1-6, doi: 10.23919/CISTI.2018.8399283

[9] Goadrich, Mark & Rogers, Michael. (2011). "Smart smartphone development: IOS versus Android". 607-612. 10.1145/1953163.1953330.

[10] Núñez, M., Bonhaure, D., González, M., Cernuzzi,L "A model-driven approach for the development of native mobile applications focusing on the data layer", Journal of Systems and Software, Volume 161, 2020, doi.org/10.1016/j.jss.2019.110489.