Developing a Mobile Application Using Open Source Parking Management System on Etherium Smart Contracts

Gerald Imbugwa, Manuel Mazzara & Salvatore Distefano
Universitetskaya St, 1, Innopolis, Republic of Tatarstan, 420500
E-mail: g.imbugwa@innopolis.university,m.mazzara@innopolis.ru,salvatdi@gmail.com

October 2020

Abstract. In this paper, we envision to illustrate the process to be used in developing a mobile application on the smart contract. We start by looking at what other researchers have accomplished and how we can improve on what already exists. The paper highlights the requirement gathering process, the methodology for data collection to streamline and validate the requirement, the architecture and implementation phase by analyzing the technological stack to achieve the business goal.

1. Introduction

Over the years, the dynamics of how we find free parking space for our cars have changed with the introduction of systems and 'Internet of Things' (IoT). Drivers can easily access applications on Google Playstore for Android users and App Store for Apple users. These applications allow drivers to find locations with empty slots, and they can choose and book those that suit their needs.

This project, dubbed Open Source Solution for Smart Contract-Based Parking Management [opensourceparking], seeks to provide a solution to parking in urban areas using the smart contract on Etherium. The research achieves its intended aim by coming up with four main actors in the system. The four actors are as shown in Figure 1:

- Administrator - responsible for authorising processes in the city [opensourceparking].
- Landlord - is a person or a company that poses a piece of land/space that is suitable for parking [opensourceparking].
- Tenant - is a company or a person to whom a landlord rents out a subset of their parking land based on some conditions [opensourceparking].
- Service provider - is a tenant that has rented a parking facility intending to rent the space to other users [opensourceparking].
Most of the work done by opensourceparking majorly focuses on the back-end, which provides smart contracts for the four actors in the system. Notwithstanding the brilliant work done, the end-users still are unable to use the system since there is no front end to help them interact with the system.

In this paper, we shall focus on enhancing an already existing solution by adding a front-end mobile application. We intend to separate the app into four separate application for the different actors. In each application will only focus on each specific actor and function they can perform in the system. In addition to that, the paper will discuss the process we envision to use from requirement gathering to development of the mobile applications.

2. Related Work

The population around the world keeps changing rapidly the statistics projection carried out by [Uni01] shows it will continue to rise [Bil98]. Despite the population increase, resources, such as land remain the same. People are migrating from one country to another in search of a better life [Bil98]. Researchers have worked on solutions to aid the urbanization process. The main aim is to build cities and spaces that will have the capacity to accommodate the population for better living. Below are some of the solutions that are currently available:

Smart Park is a company based in Italy which specializes in developing parking solutions geared towards urban mobility. Computer vision and deep learning used in the solutions aids in data analysis and making informed decisions. The solutions help citizens save time, reduce traffic jam congestion, and improve the quality of life [Sma].

IOTA Smart parking using tangle solution has been created, which allows Internet of Things (IoT) devices to communicate and exchange information. The cloud database
is updated asynchronously with new status for each slot by IoT devices installed in the parking facility. Drivers can locate for free parking slots using a mobile application that is connected to the cloud. Payment starts deducting from the car wallet once the car is in the parking slot at the moment it is out of the parking facility [KA16].

Gongjun et al proposes a smart parking system which uses a concept and architecture of NOTICE. NOTICE architecture aids in developing an intelligent with a novel privacy-aware infrastructure. The infrastructure allows vehicles to locate parking slots and book them. The paper goes further in discussing how the belt infrastructure is used to protect user-information [Yan+11].

3. Requirement Gathering

To develop a mobile application around the back-end created by [opensourceparking], we needed to come up with requirements that would help us realize the system. In this section, we illustrate how we came up with the system requirements.

We went through the smart contracts to test the functionality by running the source code. Running the source code allowed us to be able to know what each smart contract function was capable of and the parameters needed. These will aid during development when we create a form, for example, for registration we can know which information is required and how to validate the input. We also had a look at user stories used in developing the system [opensourceparking]. From the previous user stories, we adopted 32 user stories [opensourceparking] and added two more for user authentication and roles.

4. Architecture

![Mobile App Architecture](image)

Figure 2. Mobile App Architecture

The architecture only covers the mobile application side, which will be the main focus of this project. Model View View Model (MVVM) is the architecture that will be used in the front-end. We choose MVVM for three main reasons; it allows collaboration in a team each developer can focus on where they are good. It focuses on DRY code principal no need to have code duplication that can be expensive to maintain. Last, it is easy to test as MVVM breaks the coupling between the application logic and the UI.
Moreover, it aligns with our objective of achieving the 34 user stories we came up within the previous section.

For the mobile application to communicate with the local EVM node, Web3 package library will be used [Dan17]. The format of data, http response and request have to be in a format that can be easily consumed. Web3 library will help in making a connection to the EVM using HTTP protocol as illustrated in the mobile architecture above.

Commands invoked from the mobile application by different users in the system are sent to the local EVM, which runs the specific smart contract for making the necessary transaction. To protect the users from carrying out invalid transactions, all requests are validated. HTTP response which are in the form of JSON format, are parsed using the model to a format which can easily be configured by the view model to be used by different views [Gro+13].

5. Methodology

Coming up with software that is more centred towards the end-user needs, modern software approaches are required. Luqi in her research on ”Prototyping” [Luq97] shows the benefits accrued by understanding the requirements. Other research done by Dr Barry Boehm’s illustrates how the prototyping approach can help reduce the efforts of the programmer up to 40%, with a smaller binary size of the application, at the end of the process [Sel07]. We plan to use prototyping to streamline the product before the release to the market.

A prototype of the application based on the 34, user stories, will be used to create a mockup design of the application which mimics the functionality of the application. The first iteration of the developed application will be shared with a selected target of users. In addition, there will be some questions to help understand their experience.

The prototype is meant to observe the usability of the application. The end-user is presented with brief information about the mobile application, how it works, and its benefits without demonstrating the app. The user will be allowed to interact with the application while we observe. The data gathered during observation will allow uncovering the problems encountered during the usage of the application. The data collected will be used to improve the user experience during the development process for the second iteration.

The end-user will have the opportunity to make their recommendations of the issues they have experienced based on their usage. A set of questions will be presented to the users to validate the solution on the market.

Both the observation and questionnaire will be the main data collection techniques. These techniques will help in reducing the complexity of the problem and provide early validation of the system design.
6. Implementation

App Development

We understand the importance of this stage as it plays a vital role. The development phase is where our requirements become an actual product (Smart Parking app).

Based on the architecture mentioned above, we have to choose a technology stack and the patterns that will suit it. The technological stack chosen should focus on the quality attributes based on ISO25010, performance, usability, reliability, security, and maintainability [ISO].

To choose a technological stack that suits the business goals, we have to analyze the technology stack available at our disposal.

Technological Stack

Over recent years, there has been a sharp increase in mobile devices released to the market based on the statistics released by [ODe]. As at the time of writing, there were 1571.22 million units sold worldwide [ODe]. The major players are Apple and Google [HO11], who leads the industry currently. These devices rely on applications which are developed and submitted to different application stores for download by end-users. To better understand the technological stack, the three categories in which the stack falls are:

Platform-specific Native

An application written in a specified programming language. The applications written using this approach can only run on specific platforms, such as, Android or iOS. The main advantage of this approach is that the code can be optimized for each platform, it will run on. Moreover, the User Interface looks native since it utilizes API libraries that have designed for the platform [Mal16]. Some of the current trends that can be identified are the companies are embracing the declarative programming approach. Apple has introduced SwiftUI [Devb] while Google has developed Jetpack compose with the use of kotlin [Deva]. The declarative approach can help the developer to develop application more quickly by writing less amount of code.

Cross-Platform Native application

Cross-Platform Native application is an application which shares its codebase partially or entirely [Mal16]. This approach allows for application optimization and styling of the user interface (UI) to match each platform. The main advantage of this approach is that it is cheaper in comparison with to platform-specific native. Examples of cross-platform includes Flutter by Google which uses dart programming language [Devc] and xamarin which has been around for some time now which uses C# programming language.

Hybrid

A hybrid application, built using web technologies, HTML, CSS, and Javascript, along with a wrapper used to allow communication with hardware [Mal16]. Technologies used as Native Wrappers include Cordova, Phone Gap and Ionic. This option is the cheapest among the three since anyone with web experience can create mobile applications [GJS14]. Hybrid comes with its drawbacks which includes;

• The performance limitation of Hybrid application mobile applications is what makes
REFERENCES

the native popular option in most instances [BPN16].

- Lack of native User Interface and User Experience (UI/UX) [FRA]
- There are a lot frameworks out there it makes it hard to choose and uncertainty of its future [FRA]

Based on the technical stack analysis to support the architecture along with different quality attributes, the native approach is adopted at this stage.

7. Conclusion

In this vision paper, we examined the process we plan to be used to realize the mobile application for the system. The paper introduces the problem and the solution and how we intend to solve it. Moreover, we highlight the existing smart contract with Etherium which will act as a backend for the mobile application. In the literature section, we discuss the related work which tries to solve the parking problem in urban areas. One thing we can note is that they suffer from one point to another. A good example is the IoT smart parking application. When the wallet runs out of funds and the vehicle is still in the parking slot, what should happen? The paper also highlights the approach used in requirement gathering, architecture and methodology. In our methodology, we look at how the prototype can improve the product. In the process of requirement gathering, we were able to notice some limitations of [opensourceparking]. Authentication and role management is not possible with the system. These are some of the features that should be implemented in future work.

References

[Luq97] Luqi. Computer Aided Prototyping. 1997. URL: https://calhoun.nps.edu/handle/10945/36425.
[ Bil98] Richard E Bilsborrow. Migration, urbanization, and development: new directions and issues. Springer Science & Business Media, 1998.
[Uni01] New York (United States)(Germany); United Nations. World population prospects The 2000 revision. Vol. 1: comprehensive tables. 2001.
[Sel07] Richard W Selby. Software engineering: Barry W. Boehm’s lifetime contributions to software development, management, and research. Vol. 69. John Wiley & Sons, 2007.
[HO11] Adrian Holzer and Jan Ondrus. “Mobile application market: A developer’s perspective”. In: Telematics and informatics 28.1 (2011), pp. 22–31.
[Yan+11] Gongjun Yan et al. “SmartParking: A secure and intelligent parking system”. In: IEEE intelligent transportation systems magazine 3.1 (2011), pp. 18–30.
[ Gro+13] JSON-RPC Working Group et al. “Json-rpc 2.0 specification”. In: lineadisponible: https://www.jsonrpc.org/specification,[Último acceso 02 de Setiembre de 2019] (2013).
REFERENCES

[GJS14] Martin Georgiev, Suman Jana, and Vitaly Shmatikov. “Breaking and fixing origin-based access control in hybrid web/mobile application frameworks”. In: NDSS symposium. Vol. 2014. NIH Public Access. 2014, p. 1.

[BPN16] Stefan Bosnic, Istvan Papp, and Sebastian Novak. “The development of hybrid mobile applications with Apache Cordova”. In: 2016 24th Telecommunications Forum (TELFOR). IEEE. 2016, pp. 1–4.

[KA16] Abhirup Khanna and Rishi Anand. “IoT based smart parking system”. In: 2016 International Conference on Internet of Things and Applications (IOTA). IEEE. 2016, pp. 266–270.

[Mal16] Ivano Malavolta. “Beyond native apps: web technologies to the rescue!(keynote)”. In: Proceedings of the 1st International Workshop on Mobile Development. 2016, pp. 1–2.

[Dan17] Chris Dannen. Introducing Ethereum and solidity. Vol. 1. Springer, 2017.

[Deva] Android Developer. Android Apps: Kotlin. URL: https://developer.android.com/kotlin. (accessed: 08.10.2020).

[Devb] Apple Developer. SwiftUI: Better apps. Less code. URL: https://developer.apple.com/xcode/swiftui/. (accessed: 08.10.2020).

[Devc] Flutter Developer. Flutter. URL: https://flutter.dev. (accessed: 08.10.2020).

[FRA] PAUL FRANCIS. 5 Points to Consider Before Making a Hybrid Mobile App. URL: https://www.apptentive.com/blog/2015/08/25/5-points-to-consider-before-making-a-hybrid-mobile-app/. (accessed: 17.10.2020).

[ISO] ISO/IEC25010. ISO/IEC 25000: Software and Data quality. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. (accessed: 17.10.2020).

[Lac] Matt Lacey. MVVM pattern.

[ODe] S. O’Dea. Number of smartphones sold to end users worldwide from 2007 to 2021. URL: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/. (accessed: 08.10.2020).

[Sma] Park Smart. THE NEW FRONTIER OF URBAN MOBILITY. URL: https://www.parksmart.it/#fh5co-wedo. (accessed: 09.10.2020).