Computing and Information Technology
2022

Universitatea Politehnica Timişoara

Walman
A Virtual Wallet Management System

Candidate: David Daniel, Pava
Coordinator: Assoc. Prof. Razvan, Bogdan

ABSTRACT

CONTENTS

1 Introduction **4**
1.1 Context . . . 4
1.2 Motivation . . . 5
1.3 Similar Products Available on the Market . . . 5

2 Technology Stack **7**
2.1 Frontend . . . 7
2.1.1 Flutter . . . 7
2.1.2 Dart . . . 8
2.1.3 Code Generation . . . 9
2.1.4 State Management . . . 9
2.2 Firebase . . . 10
2.3 Blockchain . . . 11
2.3.1 What is a Blockchain? . . . 11
2.3.2 Ethereum . . . 12
2.3.3 Smart Contracts . . . 12
2.3.4 Solidity . . . 12
2.3.5 Test Networks . . . 12
2.4 Development Environment . . . 12

3 Implementation **13**
3.1 Use Cases . . . 13
3.2 System Architecture . . . 13
3.3 Password Management . . . 13
3.4 QR and Barcode Management . . . 13
3.5 OTP Authenticator . . . 13
3.5.1 HOTP . . . 13
3.5.2 TOTP . . . 13
3.6 Cryptocurrency Wallet . . . 13
3.7 Backup . . . 13
3.7.1 Cloud Backup . . . 13
3.7.2 Blockchain Backup . . . 13
3.8 Security . . . 13

4 Tests **14**
4.1 Test Pipeline . . . 14
4.2 Unit Tests . . . 14
4.3 Widget Tests . . . 14
4.4 Performance Statistics . . . 14

5 Conclusions **15**
5.1 Possible Improvements . . . 15

Bibliography **17**

1 INTRODUCTION

In this project I implemented a wallet manager in the form of a mobile application. The main features are password management, password generation, qr and barcode storage and management, crypto wallet and OTP token management. The user has the option to backup the data either in the cloud or on the blockchain. Once backed up, the data will be available to be restored.

1.1 CONTEXT

In the last 30 years, the number of tasks that are digitalized has increased exponentially. The most important part of the security systems of these tasks is user management and authentication. The password is the most widely spread form of user authentication and thus is often the prime target of attackers that want to impersonate someone else.

According to [6], a *"systematic literature review in the area of passwords and passwords security"*, there are many problems with password security and management ranging from weak passwords and password reuse, to users writing down passwords or sending them through unsecure channels. Most of these problems according to [6] are solved by using password recommendations. A good solution to most of these problems is a password management tool. A password manager is a piece of software designed for generating and managing passwords, in this way the user can have unique, complex and safely stored password without having to remember them.

Another great method to better secure you accounts is using a two factor authentication (2FA) method. These method vary from security questions, to one time passwords (OTP) sent from the server to the user via email or SMS, to OTPs generated using specialized algorithms such as: HMAC-based One Time Password (HOTP)[7] and Time Based One Time Password (TOTP)[8].

The cryptocurrency market is another area that has seen a considerable development lately. As of May 2022, the market cap of Bitcoin is around 565 billion USD, and the market cap of Ethereum is around 214 billion USD. In the case of Bitcoin, that is more than double of what it was in 2019 (around 211 billion USD), referenced in [1]. Cryptocurrencies also offer secure and long term storage capabilities thanks to the blockchain technology. Blockchain backups, thanks to the decentralized nature of the blockchain, are very hard to be tempered with. A traditional cloud backup could be lost or inaccessible in more than one situations. The most obvious one is data loss happening as result of a cyber attack or the company simply going bankrupt. There are also situations in which the company itself can refuse to serve you anymore. They can freeze your account or just refuse to serve an entire country all-together, we have the recent example of companies like Visa and Mastercard refusing to serve russian citizens as result of political tensions as described in [5] and [3]. All these scenarios cannot happen in a decentralized blockchain system.

Businesses that were traditionally not online like shopping also have inversely digitalized. Nowadays most of the hypermarkets offer fidelity cards. Usually these cards are built around a unique barcode or qr code. Often it's hard to manage all your cards, so a digital storage solution to solve this issue would help the end user better manage their cards.

Considered all mentioned above, a user has to remember and manage a lot of information in order to interact with the currently available online infrastructure. A tool that could help them manage all this data better is a wallet manager.

1.2 MOTIVATION

My personal motivation for creating a wallet manager is the fact that I want to use it myself. Also I wanted for a long time to explore the state of the art in smart contract development, so this was a great occasion to do so.

I chose to create this project in the form of a mobile application since people tend to have their smartphones with them most of the time, so having a virtual wallet on your mobile device makes sense.

Another factor that motivates me is the fact that currently in the mobile application market there are almost no free and open source password management applications available. The user needs to *trust* the creators of the application with their data, not knowing how the implementation of the product is made, they have no guarantee that the data is safe.

1.3 SIMILAR PRODUCTS AVAILABLE ON THE MARKET

There are a lot of password managers available on the market. In this section we are going to try to make a comparison between some of the most popular options available.

Property	**LastPass**	**RememBear**	**KeePass**	**PassMan**	**KeyBase**
Mobile Version	Yes	Yes	No	Yes	Yes
Blockchain Storage	No	No	No	No	Yes
Price	$3/month	$6/month	Free	Free	Free
License	Proprietary	Proprietary	GPL-2.0	AGPL-3.0	BSD-3

Table 1.1: A comparison between some of the most popular password managers.

First off we have LastPass[18] and RememBear[22], two very similar password managers, both having a free and a payed plan. Neither of these two is open source, so the most pressing issue regarding them is the guarantee that your data is safe. Without having the ability to see how your data is managed and stored you cannot be certain that it is secure. Also these applications do not have blockchain backups.

KeePass[16] is probably the most popular password manager for desktop. It is free and open source, and the code was analyzed and certified by specialized organizations such as the Open Source Initiative. The biggest drawback to KeePass is the aged user interface and the missing mobile application counterpart. Nowadays a lot of the situations where a user needs access to their credentials are happening while using smartphones. Also the features are limited, KeePass doing one thing and doing it well that being password management. There are no cloud or blockchain backups, so the user needs to manager backing up and storing their password database themselves.

Similar with KeePass, PassMan[19] is a free and open source password manager. They have a mobile version of the application, but blockchain backups are missing. Also, again, PassMan is just a password manager. It does not manage shopping cards or crypto wallets.

Last but not least there is KeyBase[17] which is not technically a password manager. KeyBase is a blockchain, decentralized, social media application. You can store password and secure notes inside the application but from the user experience point of view, KeyBase was never designed to be a password or wallet manager. The reason why it is mentioned, is because KeyBase is implemented on the blockchain, all user data is encrypted and it's free and open source.

2 TECHNOLOGY STACK

In this chapter I am going to describe the technologies used. The application has 3 main components. The first component is the frontend, a mobile application. The second one is the cloud storage backend. The last part are the smart contracts deployed on the blockchain.

2.1 FRONTEND

The most important aspects I considered when I chose the technologies used for the frontend was cross-platform capabilities, ease of testing, documentation availability and performance. This is why for this project I chose a Flutter stack.

2.1.1 Flutter

Flutter[12] is a mobile application development framework developed by Google in the Dart programming language. It was released in May 2017 and it currently is one of the most popular mobile development frameworks.

According to *"An empirical investigation of performance overhead in cross-platform mobile development frameworks"*[2], Flutter has one of the better resource management systems when compared with other popular mobile development frameworks.

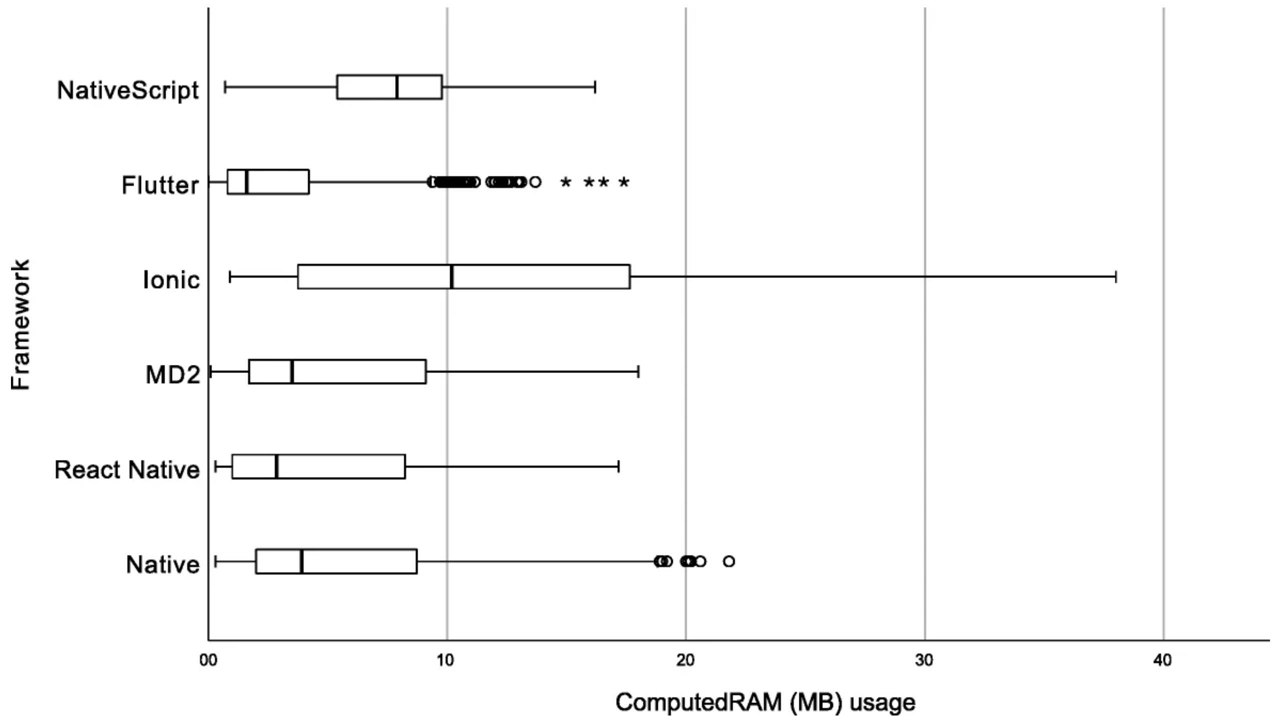


Figure 2.1: Boxplot from [2] of RAM usage across all tests done in [2]

As seen in 2.1, on average Flutter outperforms most of the other frameworks in memory management.

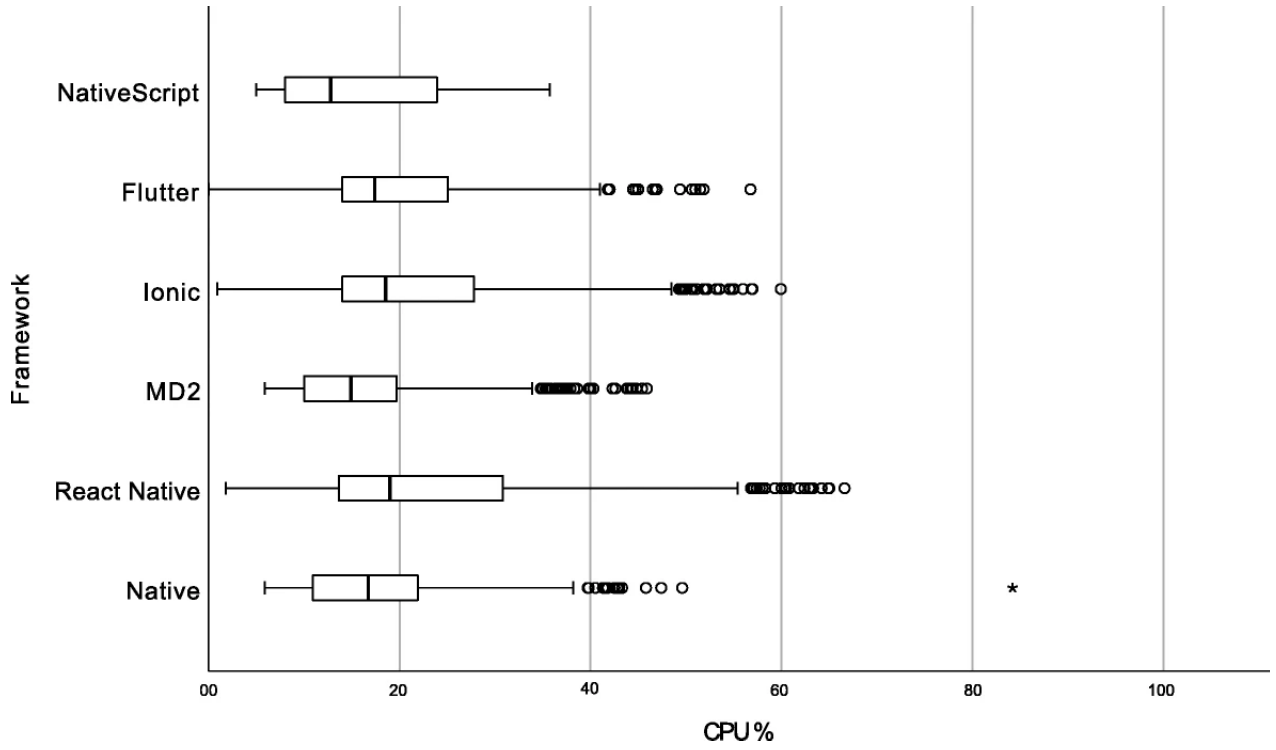


Figure 2.2: Boxplot from [2] of CPU usage across all tests done in [2]

In 2.2 we have a comparison between CPU usage of similar applications implemented in different frameworks in [2], where Flutter achieves a competitive result when compared to the other frameworks.

Another very important feature of Flutter is cross-platform compatibility. A mobile application developed in this framework can be build in native Android and IOS applications with minimal performance loss. There is also Flutter Web for web applications, offering the option to create a browser variant of the application in the future, reusing already developed and tested parts from the mobile application.

Flutter comes with a very rich and detailed documentations, the Flutter Docs[12] and a set of development plugins for the most used IDE and text editors such as Android Studio, Intellij or Visual Studio Code. During the development process the code is executed into a runtime environment, allowing almost instant compilation times speeding up the development, and in production, the code is compiled into a native application for performance enhancement.

The User Interface in Flutter is build based on a widget tree, similar to React. Every User Interface item inherits the Widget class.

2.1.2 Dart

Dart[9] is a general purpose programming language developed by Google starting with 2011. It was intended to replace JavaScript and TypeScript for frontend applications, but instead, later, it was used to create the Flutter framework.

Dart is a type safe, C-like programming language. It can be both interpreted by a runtime or compiled. The memory management is handled by a garbage collector similar with Python or JavaScript.

One of the strongest features of Dart is the compiler. It can be compiled in binary code, JavaScript or mobile native code such as Java and Kotlin for Android and Objective-C and Swift for IOS devices. Dart also performs tree shaking at compile time, discarding unused objects, methods and functions.

2.1.3 Code Generation

During the development of the project I used multiple packages (dart libraries) in the process. One very important package that needs to be mentioned is the *freezed* package[15]. This offers code generation for common model functionalities such as json encoders and decoders, copyWith methods, different constructors and access functions and more.

Classes generated by the package are annotated with the *@freezed* tag. The generated code is stored into files that contain the *.freezed.dart* and *.g.dart* extensions.

2.1.4 State Management

One of the most important aspects of frontend application development is state management architecture. There are a lot of different state management patterns available in Flutter[14] such as Provider, BLoC or the simple setState. These state management patterns tell the application when the state has changed and when certain components of the presentation layer (the UI of the application) needs to be updated as a result of that.

For this project I used the Redux state management architecture. This pattern is a very popular solution for managing the state of an application, and it is commonly used in web development. There is an implementation for it in Flutter in the packages: flutter_redux[13], redux[20] and redux_epics[21].

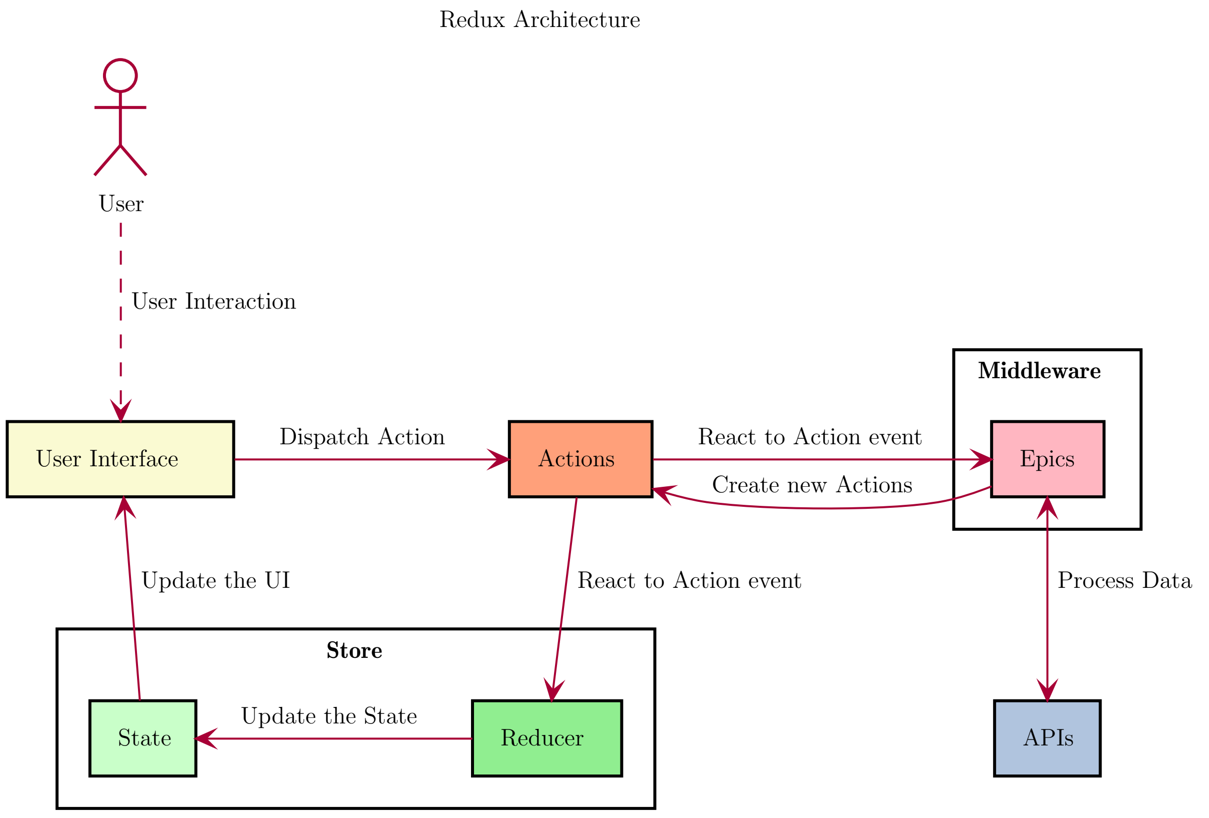


Figure 2.3: Basic structure of a Redux State Management System

In 2.3 we can see the core structure of the redux architecture I used in the project. A state change begins the dispatch of an action. They are usually triggered by the user interface, but in some cases they can be triggered by an API event. The Epics are a set of listeners which analyze the action stream. When an Epic recognizes an Action, it performs a series of operations which can process the data using the APIs (business logic) or dispatch new actions. Every action is also watched by the Reducer, which listens for Actions and changes the State accordingly. When the State changes, the Widgets (in case of Flutter) that depend on the elements updated in the State, are triggered to pe updated.

In order to access specific elements of the state, and not update every widget, every time the state changes. Containers can be used to access a specific part of the state. In order to access the State, the User Interface needs to request it from the Store, in this way, the Store knows when to redraw the object. The widgets that update when the state is modified and have access to the Store are part of the flutter_redux package[13].

Error handling in redux is made using special stream functions from the RxDart package[23]. RxDart offers and extension to the functional capabilities of dart. In redux the application is represented as a stream of actions. In order to make error handling efficiently, every action sequence spawns a new stream. In case of an exception, the stream will have an exception and we can dispatch a new Action with relevant information about the error. In this way the probability of total application wide runtime exceptions is dramatically lowered.

2.2 FIREBASE

In this section I am going to describe the technologies used for the creation of the cloud storage server portion of the application. In order to use the cloud storage, the user needs to create an account. After the account is created, the user needs to have the ability to create backup entries and restore previous backups.

Firebase[11] is an app development platform created by Google, designed as a backend for mobile and web applications. Firebase offers already implemented solutions for user management, artificial intelligence integration and databases. A firebase backend is hosted by Google in a “pay as you use” monetization scheme.

For this project I chose to use the services provided by Firebase to implement the cloud storage part of the project, mainly for ease of deployment, the good integration with Flutter throughout the dedicated packages and reduced costs (so far free).

The user management uses the email and password login service provided by Firebase. Once a user account is created, a database entry is also made, access to this database being restricted to the user base on their unique identifier.

For the storage, Firebase provides two different services. The first one is Real Time Database, which is a NoSQL type database, that resembles a Json type file. There are drawbacks to the Real Time Database such as low fragmentation which increases the data transfer size between the client and the server. A solution to this problem is the second service, Firestore, which is also a NoSQL type database similar with Real Time Database but it has some additional features. Firestore is structured into collections of documents which each have multiple fields of different types. This approach allows the client a more granular access to the data, reducing the size of files transferred between the client and the server and client side processing.

2.3 BLOCKCHAIN

For the Blockchain storage of the application I used the Ethereum smart contract development stack with Solidity. In this chapter I am going to explain the basic logic behind the blockchains, how smart contracts are built into them and the economy between smart contract deployment and usage.

2.3.1 What is a Blockchain?

The blockchain is the basic structure that sits at the base of most of the cryptocurrencies. At it's core, it is a decentralized, distributed system of data storage and processing.

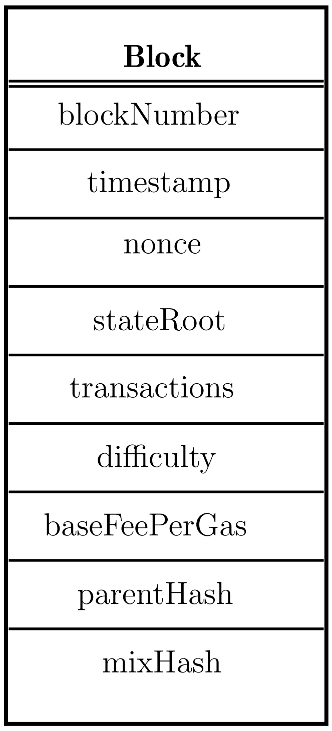


Figure 2.4: The structure of a block as described by the Ethereum documentation[10]

The blockchain is made up of a list of blocks connected to each other. Every block contains a set of information about itself and a reference to the previous block, the *parentHash* field in case of Ethereum, in the form of a hash as described in 2.4. The *parentHash* of the current block is the *mixHash* of the previous block.

The *mixHash* is the digest of a hashing algorithm of the entire content of the block. The *mixHash* needs to have the first two bytes 0, in order to be considered mined. This is achieved by incrementing the *nonce* field until the target *mixHash* is reached.

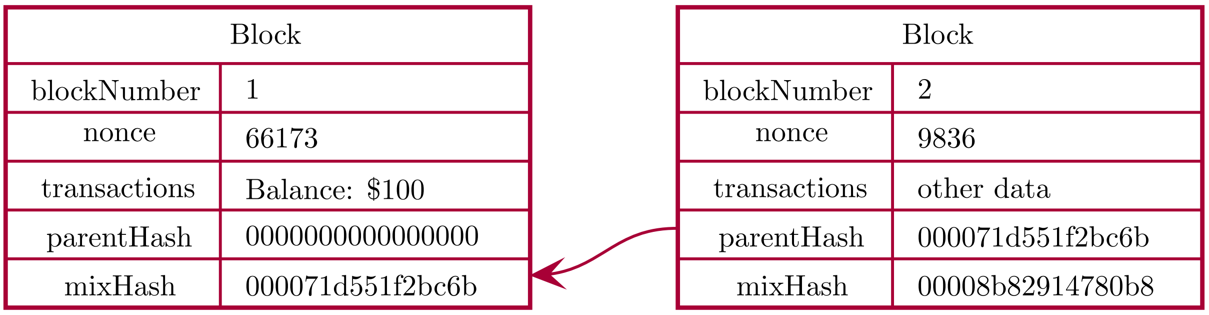
For the hashing algorithm, Ethereum was developed using the Keccak-256 hashing function, which was later standardized as SHA-3 in [4].

The block also has the *timestamp* when it was mined and the *stateRoot* which contains metadata about the current state of the system.

Information about the difficulty of the mining process and the respective price of the process are also stored in the block, in the *difficulty* and *baseFeePerGas* fields.

The *transactions* field contains information about the transactions associated with the block and the optional data exchanged. This is the place where the smart contract logic is deployed and later interacted with.

A successfully mined block is added to the blockchain and then the transaction is confirmed by every subsequent blocks.



Block	
blockNumber	1
nonce	66173
transactions	Balance: $100
parentHash	0000000000000000
mixHash	000071d551f2bc6b

Block	
blockNumber	2
nonce	9836
transactions	other data
parentHash	000071d551f2bc6b
mixHash	00008b82914780b8

Figure 2.5: High level relationship between two blocks

In 2.5 the relationship between two blocks is represented. As stated before, *block 2* has the mixHash of *block 1* in it's parentHash field. In this example, *block 1* being the first block in the blockchain has the null value as parentHash.

If the data in *block 1* was to be changed, it would invalidate the mixHash of *block 1.* The mixHash would have to be mined again, but then it will be different thant the parentHash of *block 2*, therefore invalidating the blockchain. This is the first mechanism of defense against malicious data manipulation. There cannot be changes in any previous blocks of the blockchain without invalidating every block starting from the change.

The blocks following a changed block can be mined again in order to re-validate the blockchain. Here is where the second line of defense comes in, distribution. Every chain is stored on multiple nodes, therefore a change in one has to be reflected in all of them. This makes data alteration nearly impossible.

2.3.2 Ethereum

2.3.3 Smart Contracts

2.3.4 Solidity

2.3.5 Test Networks

2.4 DEVELOPMENT ENVIRONMENT

3 IMPLEMENTATION

3.1 USE CASES

3.2 SYSTEM ARCHITECTURE

3.3 PASSWORD MANAGEMENT

3.4 QR AND BARCODE MANAGEMENT

3.5 OTP AUTHENTICATOR

3.5.1 HOTP

3.5.2 TOTP

3.6 CRYPTOCURRENCY WALLET

3.7 BACKUP

3.7.1 Cloud Backup

3.7.2 Blockchain Backup

3.8 SECURITY

4 TESTS

4.1 TEST PIPELINE

4.2 UNIT TESTS

4.3 WIDGET TESTS

4.4 PERFORMANCE STATISTICS

5 CONCLUSIONS

5.1 POSSIBLE IMPROVEMENTS

BIBLIOGRAPHY

[1] Brooks Allen and Sarah K Bryant. The market for cryptocurrency: How will it evolve? *Global Economy Journal*, 19(03):1950019, 2019.

[2] Andreas Biørn-Hansen, Christoph Rieger, Tor-Morten Grønli, Tim A. Majchrzak, and Gheorghita Ghinea. An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, 25(4):2997–3040, June 2020.

[3] Ryan Bourne. The huge scale—and implications—of the private sector boycott of russia. *Policy Commons*, 2022.

[4] Morris J Dworkin et al. Sha-3 standard: Permutation-based hash and extendable-output functions. *National Institute of Standards and Technology*, 2015.

[5] Rebecca M Nelson. *US sanctions on Russia: Economic implications.* Congressional Research Service Washington, DC, 2015.

[6] Viktor Taneski, Marjan Heričko, and Boštjan Brumen. Systematic overview of password security problems. *Acta Polytechnica Hungarica*, 16(3):143–165, 2019.

[7] Mountain View, David M'Raihi, Frank Hoornaert, David Naccache, Mihir Bellare, and Ohad Ranen. HOTP: An HMAC-Based One-Time Password Algorithm. RFC 4226, December 2005.

[8] Mountain View, Johan Rydell, Mingliang Pei, and Salah Machani. TOTP: Time-Based One-Time Password Algorithm. RFC 6238, May 2011.

[9] Dart documentation. Accessed: 2022-06-09,
https://dart.dev/.

[10] Firebase documentation. Accessed: 2022-06-11,
https://ethereum.org/en/developers/docs/.

[11] Firebase documentation. Accessed: 2022-06-09,
https://firebase.google.com/docs.

[12] Flutter documentation. Accessed: 2022-06-09,
https://docs.flutter.dev/.

[13] flutter_redux package documentation. Accessed: 2022-06-09,
https://pub.dev/packages/flutter_redux.

[14] Flutter state management documentation. Accessed: 2022-06-09,
https://docs.flutter.dev/development/data-and-backend/state-mgmt/options.

[15] freezed package documentation. Accessed: 2022-06-09, https://pub.dev/packages/freezed.

[16] KeePass product page. Accessed: 2022-06-09, https://www.keepass.info/.

[17] KeyBase product page. Accessed: 2022-06-09, https://www.keybase.io/.

[18] LastPass product page. Accessed: 2022-06-09, https://www.lastpass.com/.

[19] PassMan product page. Accessed: 2022-06-09, https://www.passman.cc/.

[20] redux package documentation. Accessed: 2022-06-09, https://pub.dev/packages/redux.

[21] redux_epics package documentation. Accessed: 2022-06-09, https://pub.dev/packages/redux_epics.

[22] RememBear product page. Accessed: 2022-06-09, https://www.remembear.com/.

[23] rxdart package documentation. Accessed: 2022-06-09, https://pub.dev/packages/rxdart.