Expertise
and insight
for the future

Henri Järvinen

Decentralized react native android application

Metropolia University of Applied Sciences

Bachelor of Engineering

Software engineering

Bachelor's Thesis

08 May 2019

Author Title Number of Pages Date	Henri Järvinen Decentralized react native android application 53 pages 08 May 2019
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Software Engineering
Instructors	Janne Salonen, Head of Department

Abstract

The purpose of this thesis was to examine theoretical principles of blockchain technology and conduct research on development into the field for creating android application using react native framework.

Thesis examines different types of blockchains and how they differentiate from one another. Blockchains examined more in depth are bitcoin blockchain and ethereum blockchain.

As a part of this thesis is study on developing environment and tools needed for enabling development for android application using react native to successfully build working android application utilizing smart contract deployed on Ethereum blockchain.

Thesis includes detailed instructions on how to replicate findings and final results. Thesis gives good starting point for further development. Research conducted in thesis was successful and working solution was found.

Keywords	Blockchain, ethereum, smart contract, android, react native

Tekijä Otsikko Sivumäärä Aika	Henri Järvinen Hajautettu react native android applikaatio 53 sivua 08.5.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto-ja viestintätekniikka
Ammatillinen pääaine	Ohjelmistotuotanto
Ohjaajat	Janne Salonen, Osaamisaluepäällikkö

Tiivistelmä

Opinnäytetyön tarkoituksena oli tutkia lohkoketjun teoreettista toimintaperiaatetta ja suorittaa tutkimus kehitystyöstä tälle alueelle kehittämällä android sovellus käyttäen react native sovelluskehystä.

Opinnäytetyö tutkii eri tyyppisiä lohkoketjuja ja kuinka ne eroavat toinen toisistaan. Lohkoketjut, joita tutkittiin tarkemmin ovat bitcoin lohkoketju ja ethereum lohkoketju.

Osana opinnäytetyötä on tutkimus kehitysympäristöstä ja työkaluista jotka ovat tarpeen Android-sovelluksen kehittämisen mahdollistamiseksi käyttäen react nativea, jotta voidaan onnistuneesti rakentaa toimiva android sovellus käyttämään ethereum lohkoketjuun lähetettyä älykästä sopimusta.

Opinnäytetyö sisältää yksityiskohtaiset ohjeet kuinka toistaa tulokset ja lopputuloksen. Opinnäytetyö antaa hyvän lähtökohdan jatkokehitykselle. Opinnäytetyössä suoritettu tutkimus oli onnistunut ja toimiva ratkaisu löydettiin.

Avainsanat	Lohkoketju, Ethereum, älykäs sopimus, android, react native

Contents

List of Abbreviations

POW	Proof of Work protocol
NVM	Node version manager
NPM	Node package manager
ES6	ECMAScript 6, also known as ES2015, is the sixth version of ECMA-262 standard
ES5	ECMAScript 5 is the fifth version of ECMA-262 standard
JDK	Java development kit
JRE	Java runtime environment
IDE	Integrated development environment
AVD	Android virtual device
VM	Virtual machine
EVM	Ethereum virtual machine
OS	Operating system
PC	Personal computer
SDK	Software Development kit

1 Introduction

The internet is great for communicating and collaborating online in various different scenarios, but the most important thing is that it connects billions of humans around the world. Internet being one valued asset in every day life finding and sharing information and even more so now with nearly all of mobile devices being connected to internet. Internet enables technologies such as blockchain. Blockchain technology can be utilized for various purposes, whether it is for transacting peer-to-peer manner, storing information, implementing specific logic on top of blockchain for specific use case. All data sent to blockchain is validated via specific procedure. [1] [2]

Blockchain technologies is a growing field, big companies are getting interested and adopting blockchain technology for their own purpose and use case. According Simanov [3] for example Walmart is aiming to switch to blockchain technology to trace its whole supply chain and according to Baydakova [4] New York Times had job posting siting "design a blockchain-based proof of concept for news publishing". These are specific use cases for blockchain technology, but most known blockchain purpose is involving currency as payment [3].

Blockchain technology did not just pop up suddenly, the technology had been researched before. Dr. Stornetta and Dr. Haber had been researching and publishing papers on this subject between 1991-1997. Their original idea was to create solution for having a way to store files in secure manner other than storing files on personal computers hard drivers. A way that the stored files cannot be tempered with. Dr. Stornetta came up with idea that the files could be dispersed widely, but interconnected copies of shared ledger other instead of trusting a single central authority. Dr. Stornetta and Dr. Haber had patent on this technology, but in 2004 the patent lapsed. [5]

Pseudonymous person called Satoshi Nakamoto had developed further Dr. Stornetta's and Dr. Haber's idea about shared ledger and storing information in secure way [5]. Nakamoto released paper in 2008 called "Bitcoin: A Peer-to-Peer Electronic Cash

System" [6]. Soon after blockchain technology was first introduced in form of Bitcoin by Satoshi Nakamoto in January 2009 [7]. Nakamotos vision was to allow direct transactions from one party to another party without going through a centralized banking service using peer-to-peer network [8]. Financial institution is centralized service and Bitcoin is run by decentralized system, meaning no single authority has control over the network [6].

Two examples of how blockchain based currency such as bitcoin is trying to overcome:

- The internet has changed how we can pay our bills by banks providing internet service to log in to pay the bills. Pay for groceries by inserting payment card into payment terminal and giving right pin code and then the payment terminal connects to the banks system via internet and checks if account has enough balance and the payment terminal gives output according to the response from the banks system. If response is that the balance is enough the amount is then transferred to groceries store bank account. These information's are run through the internet. Banks are basically services that hold information about bank accounts to which their owner has access to. There are thousands of different banks in the world and they need to communicate with each other to see bank to bank transaction through with their own validation process. Banks are middlemen when two or more parties want to transact and do business. With certain blockchain technologies transactions are purely peer-to-peer, without middlemen.
- Let us imagine a scenario that I am going to vacation to another country with different currency than I am normally using. I am going to need to prepare myself financially, because I am going to foreign country and in that foreign country is mandatory to use another currency. I am going to need to exchange my normally used currency to the destination country's currency in a third-party service and paying exchange fees according to the third-party service. Plus follow laws involving travelling with money. When everything is set, take that physical foreign currency with me to the vacation and be vary of thieves. When originally exchanged currencies run out, I need to exchange more in some other third-party currency exchange service with their fees. With blockchain technology and widely adopted virtual currency I would not need to do this, because I could use same currency all over the world, without needing any exchange services.

Ethereum blockchain is built on the same foundation as bitcoin blockchain [9]. Vitalik Buterin, the father of Ethereum blockchain idea, took bitcoins core idea and protocols and developed it further [10]. Ethereum blockchains protocol is to allow runnning programming code in decentralized system as well as sending and receiving Ethereum currency Ether. Bits of code are stored in Ethereum blockchain and the code is running on top of whole system wide computational network. These bits of codes that are running in the network can be broad variety of applications, for example financial contracts or voting based logic. [11]

2 Blockchain

In this chapter is explained how blockchain works.

2.1 General idea

Blockchain is a data structure of how data is stored in objects called “blocks”. Within this data there is hash from previous block forming traceable connection to the previous block. Due to this, blocks are “linked” together forming chain of blocks, thus the term blockchain. [12] [13]

Satoshi Nakamotos invention regarding blockchain is applying existing, working technologies and combine them to one, making it a whole new technology all together. Blockchain is essentially combined with three technologies [14]:

1. Public and Private Key Cryptography
   In blockchain use case public and private key cryptography is used to generate blockchain address and access to said address. When creating a blockchain address, with which user can send and receive currency, the procedure begins with creating a private key. The private key is then used to create public key with known algorithm. Public key and private key form a keypair and with private key user can access public keys data. The public key, which is the address generated, is then translated to more shorter and understandable version. [14] [15]
2. Peer to Peer networking

In bitcoins case millions of computers hold a copy of the same ledger. Every computer in the network has a copy of blockchains history, already done transactions and it is updated every time when new information is added to the blockchain. Every copy of blockchain is updated in the network. Computers called “nodes” and “miners” communicate with each other keeping the ledger identical to one another. [14] [16]

3. Blockchains own protocol

   Protocol varies from blockchain to blockchain [17]. For example, Bitcoin blockchain and Ethereum blockchain are both based on a protocol called Proof of Work, rewarding the computers which are calculating and verifying newest transactions, and both being public permissionless blockchains. [14] [17] [18]

2.2 Node and miner

Full node is a computer which is set to store whole blockchains history, and normally they do not get paid to do so. Users who run full node are just helping to preserve blockchain ledger distributed and thus decentralized. [19]

Computers that are set for doing computational work are called miners and some miners store whole blockchains history as well. What sets nodes and miners apart, in addition to some preserving whole blockchains history, is that they get workloads to validate transactions from which miner is trying to construct a “block” via specific procedure. Miner is after the reward which comes from Proof of Work protocol. [19]

2.3 Block

A block is a container for information, it has specific data structure. Each block consists of data specific to that block. [12] [20]

Block consists of seven major components, six of them being in block header [12]:

- Software version, a part of header, which describes the data structure of the block. Miners can determine when verifying block that which version to use. [12]
- Hash of previously added block to the blockchain, a part of header. Previous blocks hash is used to create this blocks hash. [12] [20]
- All current blocks transactions hashed with Merkle tree root hash, a part of header. Summarizes all the transactions within current block into single hash value. [12] [20]
- Timestamp in seconds since 1.1.1970 00:00UTC, a part of header. Part of when calculating the right hash for this block. [12] [20]
- Network difficulty on mining, also called the target, a part of header. Part of when calculating the right hash for this block. [12] [20]
- The nonce, a number which was suitable for this block, a part of header. Part of when calculating the right hash for this block. [12] [20]
- All the transactions included in the block as a list [12].

With these components each block is unique. Each block's unique hash, a so called identifier, is calculated by hashing block header two times with SHA256 algorithm. [20]

2.4 Proof of Work

Proof of Work (PoW) is calculating and solving complex compute intensive mathematical workloads, which then will be constructed as a block and reaching consensus between different computers that do the same calculations. In order to calculate these workloads a high amount of processing power is required, and these calculations produce specific kind of data. This specific data result is evidence that the computer has done "great amount of work", because of this the process is called "Proof of Work". PoW protocol is incentive and any of the computers doing these calculations might get done calculations with correct output and that will give reward to the computer who made the calculation, but the computer must be first to do so to be entitled to the reward. [21].

The correct output from the calculations per block vary because of variable called "nonce". The nonce is a 32-bit number, which is included in hashing the block header.

The nonce number is randomly chosen and hashed, with rest of the header, in bitcoins case, into 256-bit number. The miners are looking for a nonce number when hashed with rest of the header produces 256-bit number that is less than or equal to what network has set as difficulty. This means that the 256-bit number must have large number of zeros in front, being a very small value and hard to get with hashing. If the 256-bit number is not small enough, miner discards the nonce used and tries again with another nonce number. This is repeated until miner finds suitable nonce number. Once miner finds suitable nonce, called a “golden nonce”, the miner finally gains the opportunity to add that block to the blockchain and with that wins the reward. [22]

2.5 Consensus

Consensus protocol is set of rules that is followed between nodes and miners on a block and validating its contents. Nodes and miners reach same conclusion and agreeing that this specific block’s content is valid or not [23]. If nodes and miners reach consensus about specific blocks contents being valid, the block will then be added to blockchain [21].

2.6 Blockchain network speed

Blockchain network speed is measured in hashes done by miners together per second trying to find golden nonce. For bitcoin the difficulty is adjusted according to hash rate every 2016 blocks. The difficulty change is done so that golden nonce is found once per 10 minutes on average. [24]

2.7 Example of transaction

Here is an example of how one transaction is run through the process from sending to being added to a block and broadcasted to blockchain network [25]:

1. User is attempting to send currency from users address to another users address. The attempt is signed with senders private key [26]. This forms not yet confirmed transaction. [25]
2. The not yet confirmed transaction is then broadcasted in the blockchain network. The transaction is waiting to be selected in their own block by one or many miners in said blockchain network. [25]
3. Miners then select transactions to be included in their own block. The transaction can be selected by many miners. [25]
4. Miners then use Proof of Work protocol and trying to construct a block
5. Miner finds the suitable nonce for its own block and broadcasts the information regarding its findings to other miners. [25]
6. Other miners receiving this broadcast then verify this information. Verification process is done by hashing a specific string of the broadcasted block and comparing it to the signature included. If the hash and signature matches, then the miners receiving this broadcast agrees that the block is valid and reaching consensus. After miners reaching consensus that the block is valid the block is added to the blockchain. The information about new block being added to blockchain is then broadcasted to network. Nodes and miners update their ledger up to date. At this point the not yet confirmed transaction becomes confirmed and the currency sent is now on receiving end address. [25]

2.8 Different types of blockchains

Three different types of blockchains:

1. Public blockchains

   Public blockchains that are based on PoW protocol are permissionless, meaning anyone can participate, because the network does not require user to have permission to participate validating transactions. Anyone can send transactions as long as the transactions are valid. All data is public information, from the first constructed block to the newest constructed block, for example bitcoins whole history is available in website https://www.blockchain.com and here is first ever created block in bitcoin blockchain, block #1 https://www.blockchain.com/btc/block/00000000839a8e6886ab5951d76f411475428afc90947ee3

20161bbf18eb6048 . Examples of public permissionless blockchains are Bitcoin and Ethereum. [27] [17]

2. Federated blockchains

   Federated blockchains are more restricted in a way that not everyone can participate in validating transactions, miners are preselected. They operate still in a semi- decentralized fashion, meaning there are still many nodes and miners in the network. Federated blockchains are mainly used in banking sector. Advantages of restricted access is that blockchain works fast, better scalability and transaction privacy. Examples of federated blockchains are R3 and EWF. [27] [17]

3. Private blockchains

   Private blockchains are private and normally on organizations internal usage. Write permissions are centralized, meaning only one authority have the right to write in blockchain and validate blocks. Examples of private blockchains are MONAX and Multichain. [27] [17]

3 Ethereum blockchain

In this chapter is explained what is Ethereum blockchain and few examples on how it differentiates from bitcoin, ethereum blockchain currency ether and smart contracts.

3.1 Ethereum blockchain specified

Ethereum blockchain is designed with few Bitcoins core ideas. Bitcoin blockchain and Ethereum blockchain are both based on distributed ledger principle. Cryptography is being used in the same ideology and both being digital currency, but the two blockchains differentiate from one another in other ways. [28]

Examples on differences between bitcoin and Ethereum:

- Hashing the block header is done with ethash algorithm, unlike in Bitcoin the block header is hashed with SHA256 algorithm [28]. Ethash is a modified version of Dagger-Hashimoto algorithm [29].

- Bitcoin being designed as virtual currency, transacting peer-to-peer manner bypassing central authority in payment systems. Ethereum blockchain is designed for more use cases than that, while ether can be used for transacting peer-to-peer manner. Ethereum blockchain is designed to allow software's to be run on the network. [28]
- Average block time on bitcoin is 10 minutes while in Ethereum network average block time is between 10 to 19 seconds [30].

Ethereum blockchain specifically is designed for peer-to-peer transactions and to store programming code that is run in the Ethereum network. These programming codes are called smart contracts. Smart contracts can be utilized to be specific logic to many things, most obvious of them being payment. It really depends what kind of logic is implemented in smart contract and according to Coleman [9] smart contracts can be designed to, for example hold records, securities, trade finances. [11]

Ethereum network can be called "decentralized software platform" according to Bajpai [28] or "world computer" according to [31]. While in normal client-server model there is servers or cloud services running programs, in Ethereum network there are thousands of nodes and miners running the whole network in which developers can deploy their programs to be run at [31].Every node and miner connected to the Ethereum network runs Ethereum Virtual Machine (EVM) using same instructions. EVM enables code or program to be run on the network. Ethereum can be called "Turing complete". [32]

3.2 Smart Contract

A smart contract is a set of rules and conditions which are running on top of Ethereum blockchain [33]. Smart contracts are essentially logic written in computer language which is then stored in the blockchain. The smart contract is then run in the network and provide the logic to user using this specific smart contract [34].
Smart contracts deal with virtual currency and having smart contract logic secure and without errors is a must. A company called Zeppelin solutions has security auditing service for companies and also have created a set of smart contracts, called "OpenZeppelin", for community to use [35]. These smart contracts are open source [36] and

community reviewed [37]. Most of smart contracts handling token logic are built on ERC20 standard [38].

Smart contracts are written in Solidity programming language. Solidity is specifically designed for smart contracts. Solidity is high-level, object-oriented language, which has taken influences from JavaScript, Python and C++. [39]

Ethereum being public permissionless blockchain, all the data is public information including smart contracts. With website https://etherscan.io/ it is possible to read and go through every single created block in Ethereum blockchain. With these blocks reader can examine not only Ether transactions and through smart contract applied transactions, but deployed smart contracts as well. Although smart contract code is shown as bytecode, but there are translator services such as https://ethervm.io/decompile to translate bytecode into more readable form. With this or other translator services user can read smart contract to make sure that there are no malicious intentions.

3.3 Ether denomination

Ether is the currency used in Ethereum network [40]. Ether can be divided into many pieces and denominations used as units of ether [41].

Table 1 Table of Ether denomination

International system	Usual name	Effigy
10^{-18} – attoether	wei	Wei Dai
10^{-15} – femtoether	kwei or ada	Ada Lovelace
10^{-12} – picoether	mwei or babbage	Charles Babbage
10^{-9} – nanoether	gwei or shannon	Claude Shannon
10^{-6} – microether	szabo or micro	Nick Szabo
10^{-3} – milliether	finney or milli	Harold Finney
1 – ether	ether	
10^{3} – kiloether	kether, grand or einstein	Albert Einstein
10^{6} – megaether	mether	
10^{9} – gigaether	gether	
10^{12} – teraether	tether	

Table 1 shows that some of the denominations have a nickname based on famous persons [41].

3.4 Gas

Gas in Ethereum network is a unit to measure how much computational power will it take to run operations and every action has a gas price according to the computational power needed. [42] [43]

Gas price is calculated according to price of ether. Transaction fee must always be the same, in case Ether price goes up the gas price relative to ether goes down, keeping the relative cost of transaction the same. [44]

4 Development environment

This chapter is dedicated to explaining every part needed for development environment, enabling building android application to utilize smart contract in Ethereum blockchain.

4.1 Android

Android is a mobile operating system developed by Google. Android was invented by Android Inc and Google acquired Android Inc back in 2005 to start developing Android operating system we know now as Android. Android operating system is designed for Smartphones and tablets. First version of Android operating system was released in September 23, 2008 and it did not have name other than version number 1.0. Google later introduced sweet candy and dessert -like names for different iterations of operating systems. [45] Ninth and the newest version of Android operating system was released in August 6, 2018 and it is called “Pie” [46]. Android became most used mobile operating system soon after iterating few versions from Android 1.0. According to Chau and Reith [47] in third quarter of 2018 Android held 86,8% operating system market share.

Android OS comes with pre-installed Play store application [48]. Play store is a digital distribution platform for application, an application market place [49]. With play store user can access many of different applications according to Sage [50] and be able to download and install applications. According to research done by Statista [51] google play store had 3.7 million applications in December 2018.

Android Studio

Android Studio is an integrated development environment (IDE). Android Studio is the official IDE for Android application development, which is built on foundation of IntelliJ’ IDE. Android Studio is built on top of IntelliJ’s development tools and code editor. [52]

To help application development Android studio uses gradle-based system, Android Virtual Device and code templates [53]. Android studio gives access to Android Software Development kit (SDK), which is needed to enable the application run on android when building it [54].

SDK manager

SDK manager allows user to download SDK platforms, meaning different versions of Android OS development kits. SDK tools and option to select and add more SDK update

sites from where to check if new update is available. SDK manager checks every time when Android Studio is launched that is there new updates available. [55]

Android Virtual Device

In Android studio it is possible to create an Android Virtual Device (AVD) using Android Emulator. This enables possibility to test applications without having actual physical Android device. AVD may be configured to users will and be given specific specifications upon creation. AVDs can be created through Android Virtual Device Manager, with which user can edit and modify existing AVDs as well as perform actions such as performing a cold boot to selected AVD. There are ready made emulator templates with different Android versions and device specifications for quick selection and it is possible to create custom configuration to be identical to any physical device running Android. [56]

Virtualization

Virtualization means that it is possible to create virtual machine (VM) that is equivalent with given specifications to real physical machine with same specifications [57]. Virtualization points to creation of virtual resource of, for example server, desktop, operating system [58].

4.2 Environment tools

This subheading is dedicated to environment tools.

NodeJS

Node.js is JavaScript runtime environment. It is cross-platform and open-source development tool. Node.js utilizes V8 JavaScript engine [59]. A JavaScript engine is a program that understands JavaScript code and can execute in JavaScript coding language written code [60]. V8 JavaScript engine is developed by Google and V8 engine is open-source, which means everyone can access and view the code, which V8 engine is built on. Being open source gives developers a level of security as they can view the code and check

that there are no malicious intentions involved [61]. V8 is high-performance and Webassembly engine written in middle level programming language called C++ [62].

A small list of benefits of Node.js according to nodejs.dev [59]:

- Node.js one main advantage is that developer can write client side code and server side code using JavaScript [59].
- Node.js can handle thousands of concurrent connections with just one server without developer needing to manage thread concurrency. [59]
- Node.js supports new ECMAScript standards. With Node.js user is in charge of which ECMAScript user want to use in development environment by changing Node.js version. Website https://node.green/ gives full list of supported ECMAScript features in different versions of Node.js. [63]

Node Version Manager

With Node Version Manager (nvm) it is possible to change which Node.js version and which npm version is in use at the moment. Not all Node.js and npm users use the same versions and nvm comes handy in situations when you want to test your application on different versions of npm to be certain that your application works. It is recommended to use nvm for managing Node.js versions. [64]

Node.js Package Manager

Node.js package manager (npm) is a separate project from Node.js [65] and it is a command line tool for managing Node.js packages. With npm user can install, update and uninstall Node.js packages [66].

In the early stages of development of npm was meant to be tool for developers who write server side of applications. Npm has become very popular among front-end developers writing web applications. According to npm, Inc [67] over 80% of npm users are writing front-end code and over 11,000,000 JavaScript developers use it claims npm, Inc [68]. Statistics can be seen from npm package's own package information page from website https://www.npmjs.com/package/npm, it has well over 1,000,000 weekly downloads.

Npm has another meaning to it as well other than being command line tool for managing packages, it is the whole registry of available JavaScript packages as well. Developers, whether they are open source or from companies, use npm registry to download packages to use in their projects. It goes another way around too, developers from different origins contribute to npm registry packages for entire npm community to use or to within their own organization and their members. [69] According to npm, Inc [70] npm's own website they are the largest software registry. With their own website it is possible to discover packages, manage access to public or private packages via setting up organization, create profiles [70].

Node.js packages

Node.js packages are JavaScript packages that can be downloaded and installed via npm [71]. Node.js packages can be installed locally and globally with npm [72].

Locally installed packages are installed in node_modules folder inside the current package root folder where user ran command to install said package. Locally installed packages can be accessed only from within the same project folder project structure where package is installed [73]. To install locally user don not need to any flags to command line command [72].

Globally installed packages are installed a global location. Globally installed packages can be accessed from anywhere. [74] [75]. To install globally user must give a flag "—global" or in short "-g" to command line command [72].

Too long did not read -section by npm, Inc [76] summarizes package installation information about local installation and globally installation:

- User should install package locally if the user is going to require() the to be installed package in project files.
- User should install package globally if the user is going to run it on the command line.

Java Development Kit

Java Development kit (JDK) is a software package that includes many kinds of tools for developing in Java environment. Utilizing said tools it is possible to develop, package, monitor and deploy any application that uses Java platform standards. JDK is a kit that includes all necessary application building tools for Java platform. [77]. JDK allows to compile java programs [78].

JDK is developed by a company called Oracle [79]. JDK is development environment tool for building applications and components using the Java programming language [80].

Java Runtime Environment

Java Runtime Environment (JRE) is a runtime environment that allows running Java based software's. JRE includes Java Virtual Machine, Java class libraries and Java class loader. JRE is needed for Android Studio to run java tasks. [81]

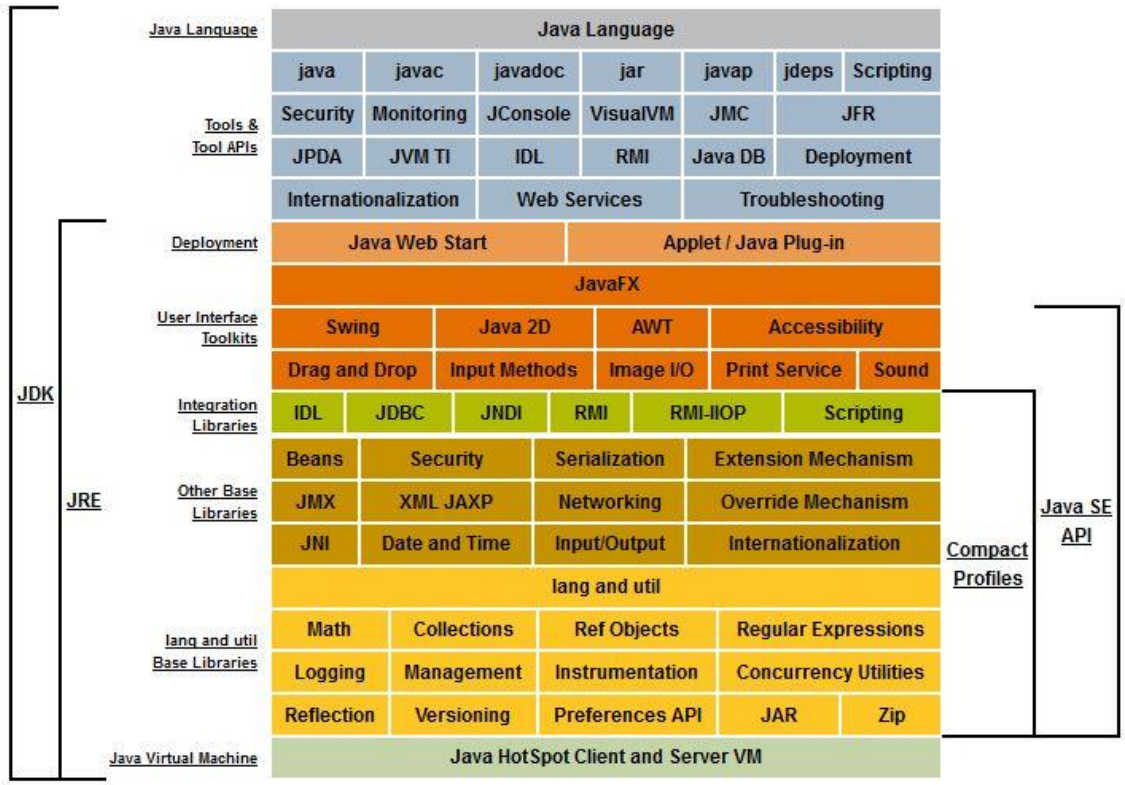

Figure 1. What is included in Java packages [82]

Figure 1 shows list of what is included in which package [82]. JRE is included in JDK package as the figure shows, no need to download it separately. Main difference between JDK and JRE is that the tools and APIs are not included with the JRE package (http://javacodedepot.com/java/java-jre-vs-jdk). JRE allows to run java programs, but with JDK is possible to compile java programs as well [78].

4.3 Frameworks

This subheading is dedicated to react native framework and two things that are in most cases required in android application development with react native [83].

React Native

React Native is a JavaScript framework, which enables developer to write and build native applications on mobile, both IOS and Android platforms [84]. Building native applications written on React Native is possible because React Native compiles to native application components [85].

In order to understand how React Native came to fruition one needs to understand its origins. React Natives origins come from React JS, which is a JavaScript library for building user interfaces. Reach JS is developed by Facebook [84]. This React JS library tangled up together a new way of rendering pages and agility of JavaScript code ending up with fast and dynamically reacting and rendering webpages to user input. After two years of React.js library being open sourced, the same team that developed React JS library in the first place released React Native, which is meant for mobile application development instead of web platforms. [85]. The Facebooks developer team wanted to take all advantages from web development and bring the advantages over to mobile development side with which Facebook then developed Facebooks own mobile application. According to Ideamotive [86] React Native framework has benefitted companies accelerate development time up to 300%.

State and props

If component has been given a state in constructor. Components state is mutable and can be changed anytime calling function ‘setState’. Props are immutable and are used to pass data to different components. [87]

ECMAScript

ECMAScript is developed by Ecma International. Ecma international is an organization that creates standards for technologies. Ecma International created standard ECMA-262 for scripting language and it is called as ECMAScript. ECMAScript gives rules, details and guidelines. These rules and guidelines must be obeyed in order to be considered ECMAScript compliant. For example JavaScript mostly implements ECMA-262 specifications. [60]

ECMAScript 6 (ES6) is the sixth version of ECMA-262 standard. It brings changes and improvements to the ECMAScript specification. ES6 is also known as ES2015.

According to Congleton [88] here is few selected best JavaScript features which came along with ES6 implementation:

- Default Parameters for automatically filling parameters
- Template literals for entering variables directly into strings
- Multi-line Strings means that strings can go over multiple lines and it only needs apostrophes in both ends
- Classes for object-oriented class design, which was not possible before
- Array functions for more simplified function creation and for simpler viewing experience
- Promises for handling asynchronous code, a promise is a container for value which is set when asynchronous code has finished. [88] [89]

Full list of what improvements and features ES6 brings are found in this website: http://es6-features.org/ .

Babel

Babel is a translator that converts ES6 code to earlier version of ECMAScript standards. In this case from ES6 to ECMAScript 5 (ES5). This is needed because not all browsers

support fully new functions or new improvements of ES6, but all of most used browsers do support ES5. Developers can use Babel to translate their code into version that browsers surely understands from ES6 to earlier versions of ECMAScript. [60]

React-native-Cli

React Native Command line interface (React Native Cli) is a cli interpreting commands. With react native cli user can initialize project structure. Initialization sets the project as react native project. [90]

4.4 Ethereum development tools

This subheading is dedicated to Ethereum development tools.

Ganache

Ganache is a software that runs blockchain for Ethereum development, which user can run locally on computer. Ganache is real Ethereum blockchain, but only for the user. User can use it to deploy contracts, develop your applications, and run tests easily because it mimics real blockchain. Ganache can be installed as command line [92] or with graphical user interface. [93]

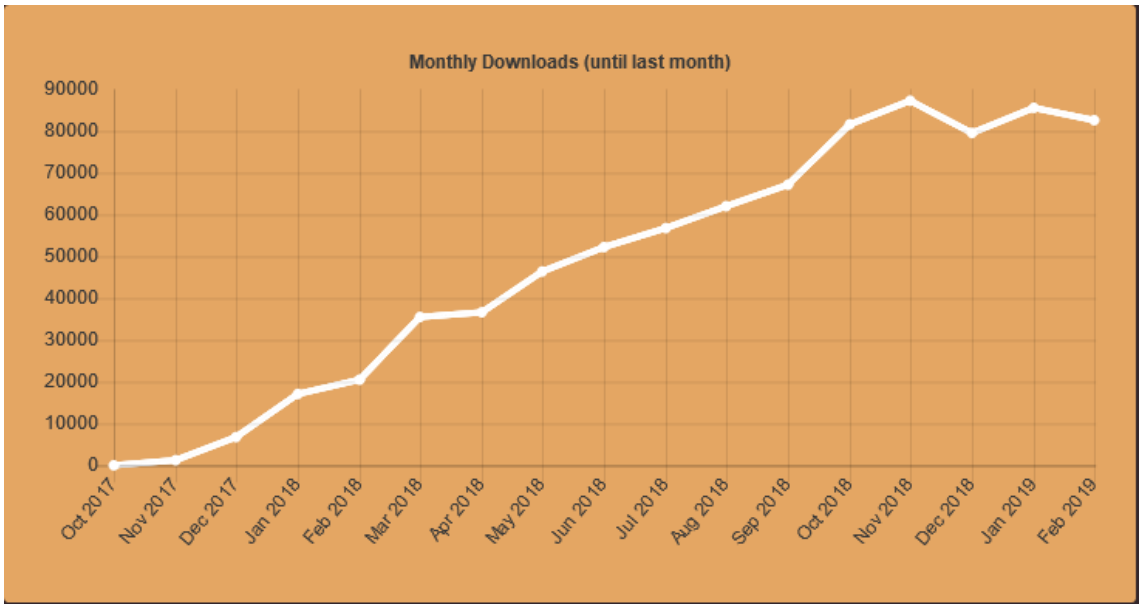

Figure 2. Ganache monthly downloads [91]

Figure 2 shows Ganache's monthly downloads and Ganache is becoming known and popular among developers as the trend shows. Ganache has 873777 lifetime downloads as of February 2019 and 82576 monthly downloads. Data is from https://trufflework.com/dashboard

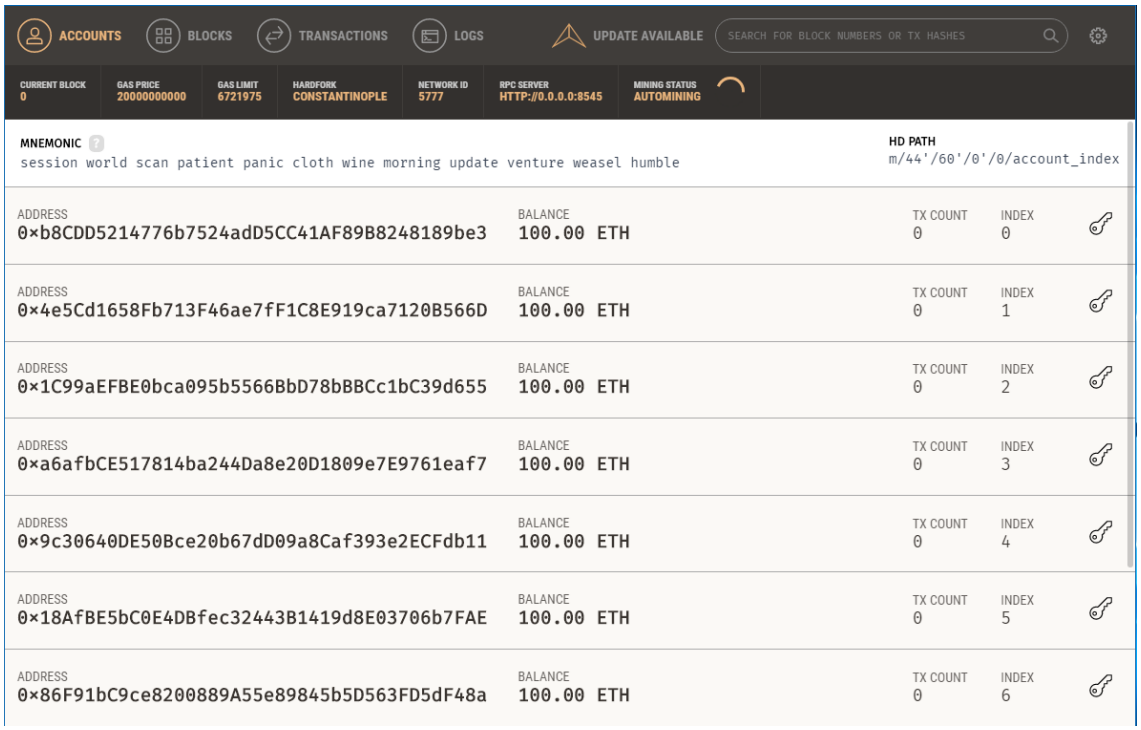

Figure 3. Ganache graphical user interface

Figure 3 shows user interface of Ganache. From accounts tab can be seen mnemonic, a 12 word seed phrase. Ganache graphical user interface has four tabs at the top:

- Accounts tab shows accounts and their balances, which are created for development purposes. Ganache creates ten accounts automatically for this purpose [94]. [93]
- Blocks tab shows all blocks mined, with transactions and gas used. [94]
- Transactions tab shows all transactions. [94]
- Logs tab shows server logs. [94]

Truffle

Truffle is development environment for Ethereum blockchain environment. With Truffle it is possible to run test framework and procedure to handle assets for blockchains. This is done by using the Ethereum Virtual Machine (EVM). It aims to make Ethereum decentralized application developers' life a bit easier. [95] [96]

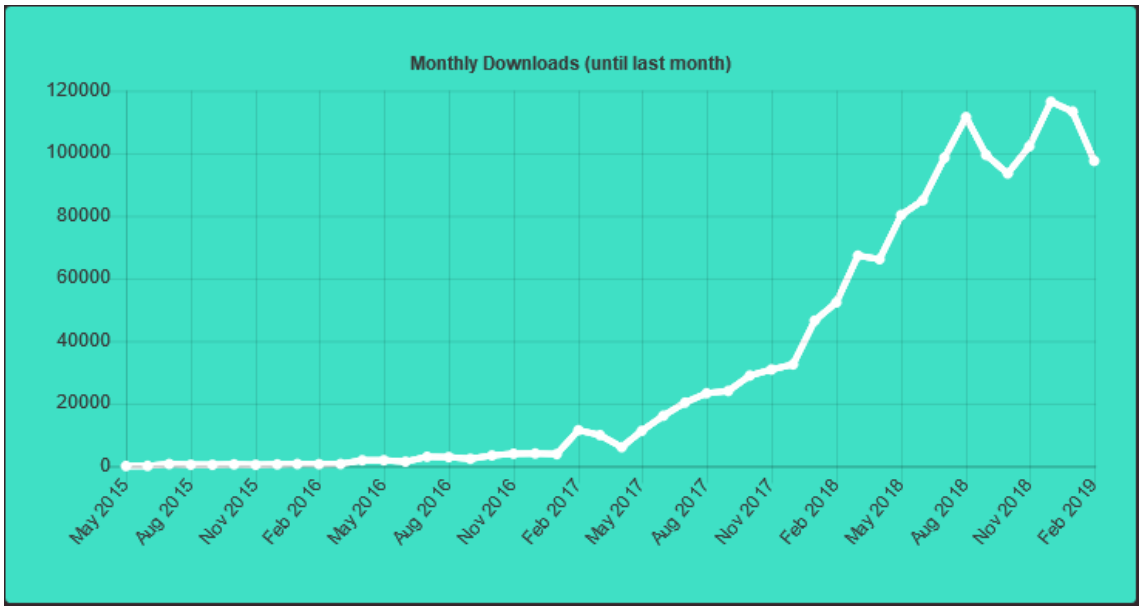

Figure 4. Truffle monthly downloads [91]

Figure 3 shows Truffle's monthly downloads and Truffle is very popular among developers who develop applications to Ethereum blockchain environment as the trend shows. Truffle has 1,550,959 lifetime downloads as of February 2019 and 97527 monthly downloads. Data is from https://truffleframework.com/dashboard.

The reason it is becoming so popular is mostly due it's abilities and according to Truffle Suite [95] Truffles prominent features are:

- Truffle has Built-in smart features such as contract compilation, smart contract deployment and binary management.
- Truffle has a framework that can be used to deploy and migrate.
- Truffles has a network management config for deploying. With it is possible to deploy many different networks deploy. The networks can be public or private.
- Truffle has implemented an package management system with EthPM & NPM

With Truffles own package management system it is possible to download pre-built boilerplate projects of many kinds [96]. In Truffle boilerplate projects are called "Truffle boxes" [97].

5 Installing development environment

In this chapter I explain how to get this development environment up and running. Note that I work on Windows operating system (OS) therefore the following information will apply on windows OS and is not guaranteed to work on MacOS or any other OS. There are many small quirks that need to be in place in order everything to run smoothly.

This chapter contains software's that were used, few development environment tools and a couple of globally installed npm packages.

5.1 Android application development tools

Java Development Kit

Android applications core are written in Java or Kotlin, because of this the Oracles Java-compiler and Java libraries are needed to compile and build applications and therefore Java Development kit is needed. JDK is needed for Android Studio. Android Studio installer comes bundled with JDK, but it is advisable to install JDK separately. JDK can be downloaded from the following website: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Figure 5. Java version

Figure 5 shows used java version JDK version 8 update 181, build 1.8.0_181-b13.

After installing JDK it is important to set entry of JDK's install location in windows environmental variables otherwise Android Studio will not know where Java compiler is located and will not be able to run tasks that involve java compiler. Information on how to set environment variables are found in the following website: https://docs.oracle.com/javase/tutorial/essential/environment/paths.html. Easiest way to locate your own java install path is to type "where java" in cmd.

I did try the newest JDK, but when ever I tried to run my developing environment many tasks failed or did not work at all, which is weird as I had environmental variables in place just like for this older version. After few hours of trying I reverted back to 8 update 181, build 1.8.0_181-b13.

Android Studio

I used during thesis application development Android studio for code editing and Android emulator mainly because convenience of code editor and emulator being in same application. Android studio can be downloaded from the following website: https://developer.android.com/studio/

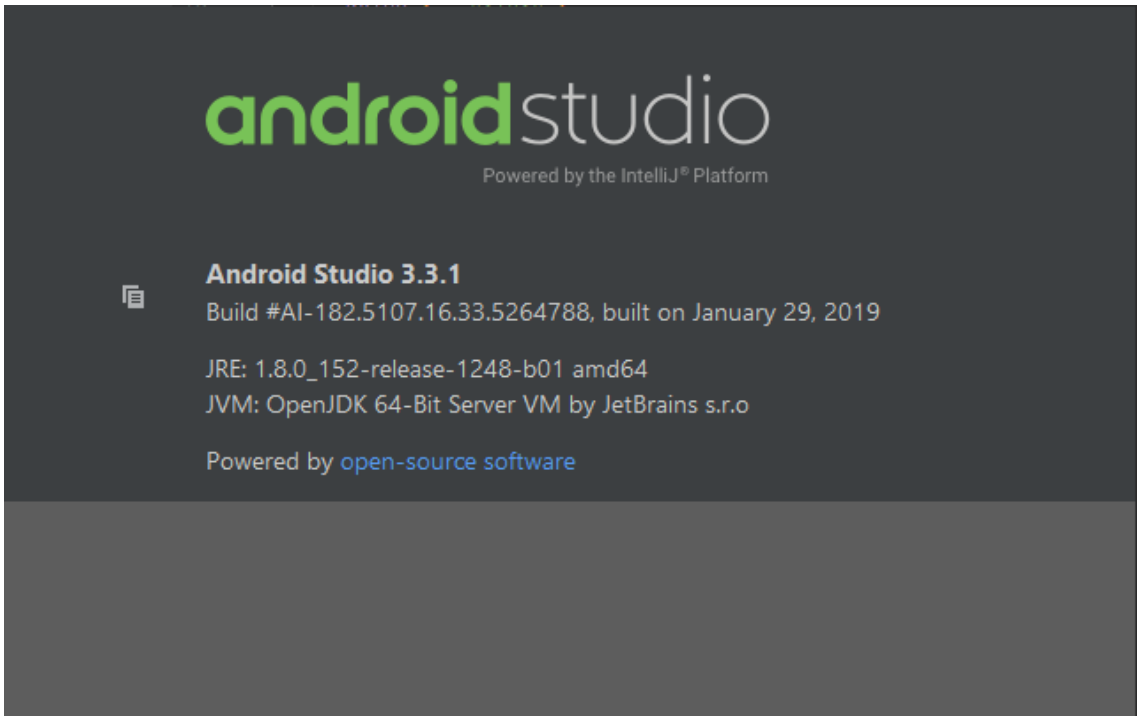

Figure 6. Android studio version

Figure 6 shows Android Studio version used during thesis application development. Android Studio was 3.3.1, build #AI-182.5107.16.5264788, which was built on January 29.2019.

Android studio needs environment variable for SDK location [100]. Information on how to set environment variable for Android Studio to locate SDK installation folder is in the following website: https://developer.android.com/studio/command-line/variables.

Android Virtual Device

I used Nexus 5X as the Android Virtual Device (AVD). Operating system was Android 8.1 API 27 on my AVD.

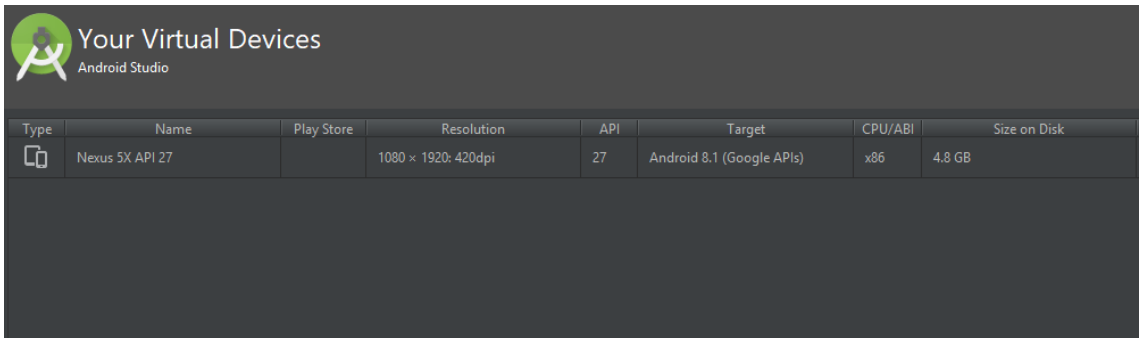

Figure 7. Android Virtual Device manager

Figure 7 shows screen capture of AVD manager, AVD being Nexus 5X, API 27, Android 8.1.

Virtualization

I have my development environment installed on my desktop personal computer (PC) at home, which has AMD Ryzen processor. In order to get Android Studio Virtual Device to run on my PC I had to do these steps:

- Install Android Studio 3.2 Beta or higher. [98]
- Download Android Emulator v27.3.8+ via Android Studio SDK manager. [98]
- Create x86 Android Virtual Device (AVD) with downloaded SDK. [98]
- Update Windows 10 with April 2018 Update. [98]
- Enable Windows Feature called "Windows Hypervisor Platform" from Windows Features. [98]

In addition to all these, it is mandatory to enable Virtualization from motherboards bios, otherwise Android Virtual Devices or any virtualization will not run.

React Native Command line interface

React native cli was used to initialize project structure and creating it as react native project. React native cli can be installed via following command:

```
npm install -g react-native-cli
```

Listing 1. Command to install react-native-cli

Listing 1 shows command to install react-native-cli

```
$ react-native -v
react-native-cli: 2.0.1
```

Figure 8. React-native version

Figure 11 shows react-native-cli version used.

5.2 Node.js

Download latest release of NVM from website https://github.com/coreybutler/nvm-windows/releases and install.

Node.js

Installing Node.js using nvm is done with command:

```
nvm install version
```

Listing 2. nvm install command

Using Listing 2 command and giving version in place of “version”, nvm installs that version.

```
$ node -v
v11.6.0
```

Figure 9. Node.js version

Figure 10 shows node.js version used in thesis.

On windows platform Node.js requires windows-build-tools to be installed for compiling native Node modules. [99]

To install windows-build-tools use command:

```
npm install --global windows-build-tools
```

Listing 3. install windows-build-tools command

Listing 3 shows command to install windows-build-tools.

Node package manager

Npm is bundled in Node.js installer as it states in npms own package information on website: https://www.npmjs.com/package/npm. It is advisable to install npm separately, otherwise Node.js installer will install npm locally and that can cause some permission access errors with running packages globally [64]. After installing Node.js via NVM, run command:

```
npm install npm -g
```

Listing 4. Command to install npm globally

Listing 4 shows command for installing npm globally.

5.3 Ethereum environment development tools

Ganache

I used Ganache in this thesis to test deploy Smart Contracts and to see if transactions went through. Ganache gives you ten test accounts to work with. Download installer from https://trufflframework.com/ganache and install Ganache.

Figure 10. Ganache version

Figure 8 shows version used during thesis development. During thesis development Ganache version 1.3.0 was used. Running blockchain locally is nice and easy with Ganache. It is a relief knowing that you can test deploy smart contracts as many times you want and knowing that no harm can be done.

Truffle

I used Truffle to compile smart contracts and migrate smart contracts. I installed truffle globally via npm using command

```
npm install -g truffle
```

```
$ truffle version
Truffle v5.0.4 (core: 5.0.4)
Solidity v0.5.0 (solc-js)
```

Figure 11. Truffle version

Figure 9 shows Truffle version used during development. Truffle version was v.5.0.4.

Cmd will not recognize truffle commands, I used git bash to run truffle commands. Git can be downloaded from the following website: https://git-scm.com/downloads.

Git

I used git command line interface and downloaded it from (https://git-scm.com/downloads). Few times I used Git bash, because git cli or cmd for the same manner, could not interpret truffle commands.

6 Building application

In this chapter I will explain in more code-oriented manner about what commands and how the project was set up.

6.1 Skeleton and settings

I used react-native cli, which is globally installed node package, to create project skeleton using command in command line interface:

```
react-native init Thesis
```

Listing 5. Skeleton initialization command

This command created folder and necessary files to run bare bones react-native application. Note that if initialized project is not capitalized, it will cause build errors later. I made this mistake by initializing folder not capitalized first.

Opening project in Android Studio and opening SDK manager. Downloading necessary SDK, for my case it was Android 8.1 API27. Different SDK Tools for building android application, in my case they were:

- Android SDK Build-Tools 29-rc2
- Android Emulator

- Android SDK Platform-Tools
- Android SDK Tools

Then configuring Android virtual device and after that build gradle settings in “build.gradle” file from project folder “/android/” and “/android/app/” to match Android virtual device API level, and set wrapper in folder “/android/gradle/wrapper/” in a file called “gradle-wrapper.properties”:

```
distributionUrl=https\://services.gradle.org/distributions/gradle-4.6-all.zip
```

Listing 6. Gradle wrapper version

Listing 6 shows gradle wrapper version used in thesis, gradle-4.6-all.zip.

6.2 Truffle structure

Creating necessary structure for truffle inside project folder using command line and navigating to Thesis folder and initializing truffle using command:

```
Truffle init
```

Listing 7. Truffle initialization command

Listing 7 command will create folder structures for truffle to understand and one config file for truffle to read called truffle-config.js. Truffle-config.js has information about what network will truffle connect and what solidity compiler is used. Default network is development, connecting to loopback address 127.0.0.1, which is where Ganache will be found at this point. By changing network, it is possible to deploy smart contract to main Ethereum network, but in thesis I only deployed to local Ethereum blockchain.

6.3 Installing node.js packages

In this subheading I will explain what project specific node packages I used to build application. To determine which version of specific package I would have to use was hard, because this field is still highly experimental and still in early phases of being mature

field. Although many different node packages are being developed, they do not necessary work in my setup. I could not be certain which packages would work and would not work with packages which was necessary for the application. I ended up rolling different version many times and testing what would work.

Project dependencies and their versions from package.json file, not listing packages that were included in initialization of the project with react-native cli and devDependencies:

- @babel/plugin-proposal-decorators: “^7.2.3”,
  Enables react native to understand annotations.
- babel-plugin-transform-builtin-extend: “^1.1.2”,
  This package was added to fix extendablebuiltin-error, which caused trouble.
- es6-promise: "^4.2.5",
  Enables react native to understand promises
- jsc-android: "^236355.1.1",
  Build scripts for JavaScript engine. This package was added to fix build errors.

```
configurations.all {
         resolutionStrategy {
         force 'org.webkit:android-jsc:r236355'
    }
}
```

Listing 8. configuration to use jsc

  Fixing build error was done by adding code in Listing 8 into file called “build.gradle” from path “/android/app/”.
- node-libs-react-native: "^1.0.3",
  Enables react native to understand Node core modules like http and helped in fixing not able to access crypto module -error.

```
const nodeLibs = require('node-libs-react-native');

module.exports = {
  resolver: {
    extraNodeModules: nodeLibs
  },
};
```

Listing 9. configuration to use nodeLibs

  Fixing not able to find node certain node modules error by adding code in Listing 9 into file called “rn-cli-config.js”.
- openzeppelin-solidity: "^2.2.0",

Enables implementation of standardized smart contracts, importing and extending.

- react-native-randombytes: "^3.5.1",

  Enables access to randombytes, which react native could not access from node modules and helped in fixing not able to access crypto module -error. It is mandatory to run command:

```
react-native link
```

Listing 10. React-native link command

With Listing 10 command this package is imported to MainApplication.java, added to dependencies in *build.grade* file, added in settings.gradle file and added to list of packages used (and IOS side, but that I have not tested).

- truffle-contract: "^4.0.5",

  Truffle handles contract abstraction via this module.

- truffle-hdwallet-provider: "0.0.3",

  Use it to sign transactions for addresses derived from a 12-word mnemonic.

  Truffle-hdwallet-provider is the wallet, which holds mnemonic to an account and address where to connect. In my case Ganache provided me the test accounts and gives mnemonics to all these accounts as well. The address where to connect I set my local Internet Protocol (IP) address. Nearly all google results I encountered suggested I should be connecting to loopback address, which is 127.0.0.1, but I could not get it to work. Instead I tried with my local IP address and it worked.

- vm-browserify: "^1.1.0",

  This package is needed to emulate node's vm module for browser and helped in fixing not able to access crypto module -error.

```
nodeLibs.vm  = require.resolve('vm-browserify');
```

Listing 11. configuration to vm

Adding Listing 11 code into rn-cli-config.js helped in fixing not able to access crypto module -error.

- web3: "^1.0.0-beta.34"

  Web3 library is essential for this project. Web3 library contains specific functionality for Ethereum network. Web3 is in charge of making the connection to the Ethereum network. In thesis was used Ganache, which is the locally run Ethereum blockchain.

With truffle-hdwallet-provider I ended up rolling all the versions available to the package due to not being able to receive response from localhost, until I installed the very first version of said package. The very first version managed to give me error about not able to connect to localhost and any other version could not provide this error. With all

versions other than the newest, which was 1.0.6 at the time, the application was built successfully. With newest version the application build stopped at 99,4%. Reasons for that still being unknown. Tried to install many versions of different packages.

One package was added to the devDependencies:

devDependencies:

- babel-preset-es2015: "^6.24.1"
  Enables babel to understand EcmaScript 2015 standard.

Due to some packages, that were necessary for the application, not being supported in much newer version of react, react-native, babel translator and the build structure, which came along with initializing the project structure. I had to install previous versions of them all and set previously used build structure.

After many trial and error iterations, I got the application running with the following versions:

Dependencies:

- react": "16.7.0",
- react-native": "0.57.7"

devDependencies:

- babel-cli: "^6.26.0",
- babel-jest: "23.6.0",
- metro-react-native-babel-preset: "^0.51.1",
- react-test-renderer: "16.7.0"

One more addition to all packages is adding a file globals.js to root folder. It is necessary, because used modules do not have native base64 converter. This file is created to fix not able to find “btoa or “atob”.

```
import url from "url";
```

```
global.URL = class URL {
  constructor(inputUrl) {
    return url.parse(inputUrl);
  }
};

if (typeof btoa === 'undefined') {
  global.btoa = function (str) {
    return new Buffer(str, 'binary').toString('base64');
  };
}

if (typeof atob === 'undefined') {
  global.atob = function (b64Encoded) {
    return new Buffer(b64Encoded, 'base64').toString('binary');
  };
}
```

Listing 12. globals.js

Listing 12 shows code in globals.js file. The file must be imported to App.js file and index.js file.

With these packages and global.js file I managed to get the application running, without producing any critical errors. With these node packages it was trial and error after another, but now I have found a working solution to successfully launch application with needed packages for the project.

6.4 Configuring connection to Ganache

Code from App.js required for android application to connect to Ganache

```
import HDWalletProvider from 'truffle-hdwallet-provider';
const Web3Library = require('web3');
const mnemonic = 'session world scan patient panic cloth wine morning update
venture weasel humble'; // 12 word mnemonic
const Provider = new HDWalletProvider(mnemonic, "http://192.168.10.62:8545");
const web3 = new Web3Library(Provider);
```

Listing 13. Configuration of connection to Ganache

Listing 13 shows code required for android application to connect to Ganache. In normal situation mnemonic would not be hard coded, because with mnemonic user gains access to account and its balance.

6.5 Configuring truffle

Set solidity compiler version to same as in smart contract created in file truffle-config.js.

Configuring application to access smart contract is done as follows:

```
import MetropoliaToken from './build/contracts/MetropoliaToken.json'
var contract = require("truffle-contract");
let tokenContract = contract(MetropoliaToken);
tokenContract.setProvider(Provider);
```

Listing 14. Configurations for truffle and access to smart contract

Listing 14 code shows how android application accesses smart contract using truffle. Provider being the same Provider as in Listing 11.

6.6 User interface and functionality

Simplistic user interface with four different buttons designed to test functions.

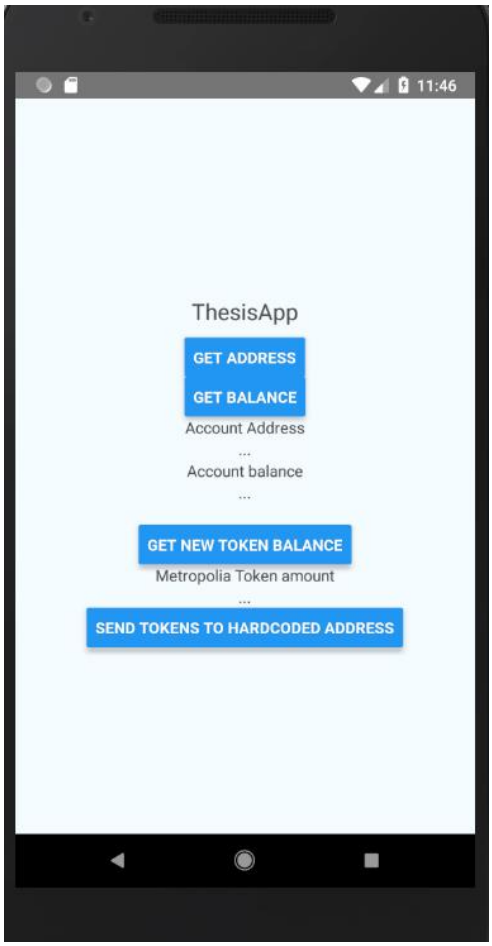

Figure 12. Screencapture of ThesisApp

Figure 12 shows user interface with four buttons components and three text components.

1. First button will change text state and change it to address being used.

```
<Text>{this.state.accountAddress}</Text>
```

Listing 15. text component for accountAddress

Listing 15 shows text component, which state will be changed once button is pressed.

```
//Get account function
GetAccount() {
    web3.eth.getAccounts().then((accounts) => {
        this.setState({accountAddress: accounts[0].toLowerCase()})
    });
```

```
};
```

Listing 16. Get account function

Listing 16 shows function that will change text component accountAddress state into account address being used.

2. Second button will change text state and change it to accounts Ethereum balance.

```
<Text>{this.state.accountBalance}</Text>
```

Listing 17. Text component accountBalance

Listing 17 shows text component, which state will be changed once button is pressed.

Function that will change accountBalance state:

```
//Get balance function
GetBalance () {
    web3.eth.getAccounts().then((accounts) => {
          web3.eth.getBalance(accounts[0]).then((balance) => {
               this.setState({accountBalance: balance})
          });
    });
};
```

Listing 18. GetBalance function

Listing 18 shows function that will change text component accountBalance state into account balance.

3. Third button will change text state and change it to accounts MetropoliaToken balance.

   Text, which state will be changed once button is pressed

```
<Text>{this.state.accountTokenBalance}</Text>
```

Listing 19. Text component accountTokenBalance

Listing 19 shows text component accountTokenBalance, which state will be changed once button is pressed.

```
//Get token balance function
GetTokenBalance () {
```

```
    return new Promise((resolve, reject) => {
          web3.eth.getAccounts((error, accounts) => {
                tokenContract.deployed().then((instance) => {
                      return instance.balanceOf(accounts[0])
                }).then((result) => {
                      let balance = result.words[0];
                      resolve(balance);
                      this.setState({accountTokenBalance: balance});
                }).catch((error) => {
                      console.error("Error" + error);
                      reject(error);
                });
          });
    });
};
```

Listing 20. Get token balance function

Listing 20 shows function that will get accounts token balance and change accountTokenBalance state into accounts token balance.

4. Fourth button will send 50 MetropoliaTokens to another account, which is a hardcoded address into the function.

```
//Send tokens function
//This function will send 50 tokens to hardcoded address
SendTokens() {
    return new Promise((resolve, reject) => {
          web3.eth.getAccounts((error, accounts) => {
                tokenContract.deployed().then((instance) => {
instance.transfer("0x4e5Cd1658Fb713F46ae7fF1C8E919ca7120B566D", 50,
{from: accounts[0]})
                }).catch((error) => {
                      console.error("Error" + error);
                      reject(error);
                });
          });
    });
};
```

Listing 21. Send tokens to hardcoded address function

Listing 21 shows code for fourth button component for sending 50 MetropoliaTokens to hardcoded address.

6.7 Smart contract, compile and migrate

I created very simplistic smart contract for thesis. The purpose of this smart contract is to show that it works and functions as it should.

```
pragma solidity ^0.5.2;
```

```
import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";

contract MetropoliaToken is ERC20 {

      string public name = "MetropoliaToken";
      string public symbol = "MPT";
      uint8 public decimals = 2;
      uint public INITIAL_SUPPLY = 25000;

      constructor() public {
        _mint(msg.sender, INITIAL_SUPPLY);
      }
}
```

Listing 22. Contents of MetropoliaToken.sol file

Listing 22 shows contents of MetropoliaToken.sol file. Pragma solidity ^0.5.2 line is declaring which version of Solidity compiler this smart contract should use, compile version is also declared in truffle-config file. Next importing foundation of standardized token functions which are all included in ERC20.sol, which is already implementing safemath logic. Then declaring values for this thesis token:

- Contract name: MetropoliaToken,
- Token name: MetropoliaToken,
- Token symbol: MPT,
- Token decimals: 2,
- Initial supply: 25000

Constructor calling function _mint, which is function in ERC20 and the purpose this function is to set created tokens to an account. The function takes two parameters and they are account and tokens amount. In this case I sent initial supply amount to the sender.

Compiling and deploying smart contract

Before compiling and deploying one file must be created for truffle to understand the smart contract filename. A file called “2_deploy_contracts.js” must be created in “/migrations/” folder.

```
var MetropoliaToken = artifacts.require("MetropoliaToken");

module.exports = function(deployer) {
  deployer.deploy(MetropoliaToken);
```

```
};
```

Listing 23. 2_deploy_contracts.js file

Listing 23 shows code from 2_deploy_contracts.js file.

Compiling smart contracts to be ready for deployment using command:

```
Truffle compile
```

Listing 24. Tuffle compile command, must be given in bash or powershell

Listing 24 shows command with which smart contract is compiled.

Ganache must be running for successful migration. Migrating and deploying compiled smart contracts using command:

```
Truffle migrate
```

Listing 25. Truffle migrate command, must be given in bash or powershell

Listing 25 shows command for migrating and deploying smart contract.

Output of migrate command:

```
Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:
0x21ea5e9d99a5372a168c6e5c7085b80cdaa9946beae6635fd1974f9a552b573b
- Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0x3379cf21C79D5f8EdBdE847fCAFf0a2d743644d6
   > account:             0xb8CDD5214776b7524adD5CC41AF89B8248189be3
   > balance:             99.99453804
   > gas used:            273098
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00546196 ETH
```

```
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00546196 ETH


2_deploy_contracts.js
=====================

   Replacing 'MetropoliaToken'
   ---------------------------
   > transaction hash:
0x053ed4eb9fbd10adf86c74c72303f05dd37a21d145f30bc80dd7648601a9d1f6
- Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0xF9AA2000b2964d8c51Bb878213dD9A65e998D877
   > account:             0xb8CDD5214776b7524adD5CC41AF89B8248189be3
   > balance:             99.97218292
   > gas used:            1117756
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.02235512 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.02235512 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.02781708 ETH
```

Listing 26. Output of truffle migrate command

Listing 26 shows output of truffle migrate command, and that the smart contract was successfully migrated. It shows how much processing power it took from network in form of gas being used and final cost of deploying into the network in form of Ether.

These events can be seen from Ganache blocks tab now that contract is deployed and mined.

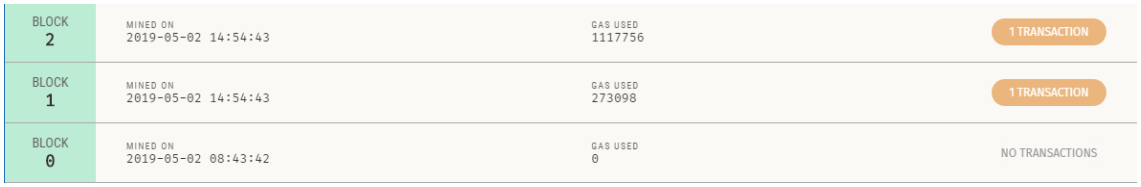

Figure 13. Migrations being shown in blocks tab

Figure 13 shows that migrations are visible in blockchain, in Ganache blocks tab.

6.8 Result

This subheading is showing the results of successful built, functioning thesis android application.

Run ganache, compile and deploy smart contract using truffle, run AVD and run application by entering command in cmd while being in project root folder:

```
react-native run-android
```

Listing 27. Command to run android application

Listing 27 shows command to run application.

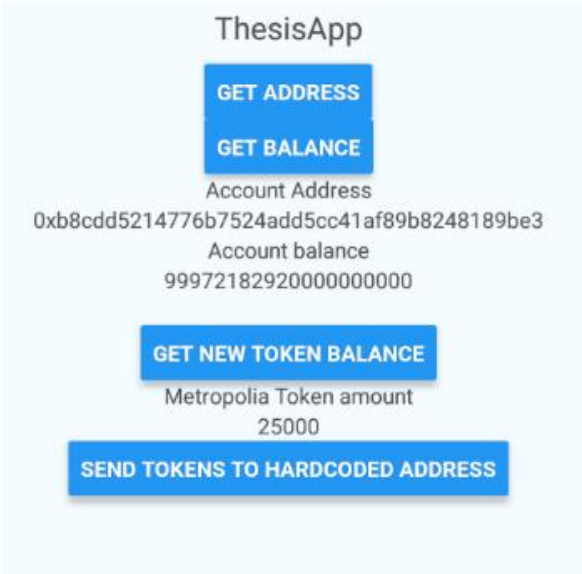

Figure 14. Result after migrating

Figure 14 shows account balance after migrating and Metropolia Token amount after initial migration. As previously stated, the MetropoliaTokens were set to sender, here it shows that all 25000 is in used accounts balances.

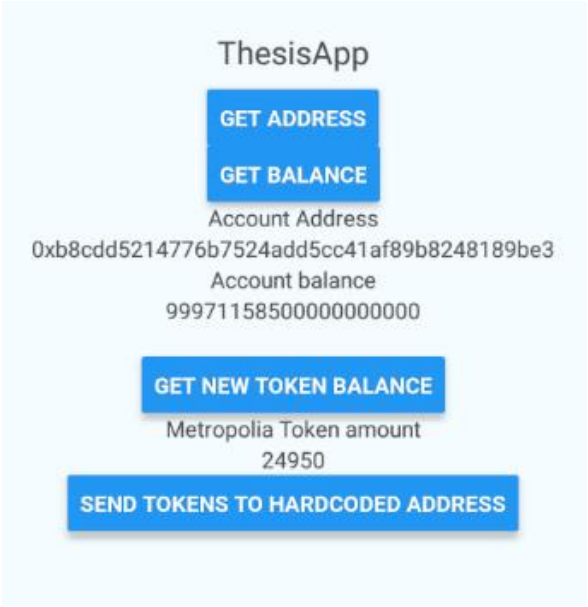

Figure 15. Result after sending tokens

Figure 15 shows that after sending 50 tokens to hardcoded address, the values of account balance and Metropolia Token amount have changed. The values are updated by pressing “get balance” button and “get new token balance” button.

Figure 16. Mined block for sending 50 MetropoliaTokens

Figure 16 shows mined block for sending 50 MetropoliaTokens to hardcoded address. Gas amount used is 51221 as it states in Figure 16.

7 Conclusion

Conducted research was a success. Study resulted in successfully built application that utilizes smart contract deployed to Ethereum network. Even thought the smart contract with standardized functions deployed for ThesisApp was a simple one, it enables further development as the whole system works.

Building mobile application to utilize blockchain technologies is not common. Even less common is building mobile application with react native in this field. Leading to finding solutions for many of problems encountered during application development being very hard. There is no clear guidelines or tutorials on how to build application for same purpose as ThesisApp.

Android application development was mostly trial and error due to this field being far from mature. Most troublesome packages were web3 library, truffle-hdwallet-provider. Finding node.js packages that would work with each other took several weeks, well over a month.

This was a great chance on learning new and hopefully thesis findings will help others developing in this field with react native. ThesisApp gives good starting point for further development.

Here is link to source code of ThesisApp: https://gitlab.com/fourtti/metropoliathesis.

References

1 Chantelle Lafaille. What is Blockchain Technology? A Beginner's Guide. https://www.investinblockchain.com/what-is-blockchain-technology/

2 BehaviourExchange (BEX). 10+ Uses for Blockchain that will Change the World. https://hackernoon.com/10-uses-for-blockchain-that-will-change-the-world-c5b96cf7c976

3 Nate Simantov. 6 major companies that went blockchain in 2018. https://medium.com/orbs-network/6-major-companies-that-went-blockchain-in-2018-712666afba0c

4 Anna Baydakova. The New York Times Is Planning to Experiment With Blockchain Publishing. https://www.coindesk.com/the-new-york-times-is-planning-to-experiment-with-blockchain-publishing

5 Sead Fadilpašić. A Key Insight For the Blockchain Came in 1991. https://cryptonews.com/news/a-key-insight-for-the-blockchain-came-in-1991-1895.htm

6 Economist. The great chain of being sure about things. https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things

7 Zoë Bernard. Everything you need to know about Bitcoin, its mysterious origins, and the many alleged identities of its creator. https://nordic.businessinsider.com/bitcoin-history-cryptocurrency-satoshi-nakamoto-2017-12?r=US&IR=T

8 Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

9 Futurethinkers. FTP016: Vitalik Buterin – What is Ethereum and How to Build a Decentralized Future. https://futurethinkers.org/vitalik-buterin-ethereum-decentralized-future/

10 Cointelegraph. Who is Vitalik Buterin. https://cointelegraph.com/ethereum-for-beginners/who-is-vitalik-buterin

11 Alyson. "What is Ethereum?" and other answers to basic questions about the Ethereum network.

https://support.blockchain.com/hc/en-us/articles/115003651143--What-is-Ethereum-and-other-answers-to-basic-questions-about-the-Ethereum-network

12 Marko Vidrih. What Is a Block in the Blockchain?. https://medium.com/datadriveninvestor/what-is-a-block-in-the-blockchain-c7a420270373

13 ANTONYLEWIS2015. A Gentle Introduction to Blockchain Technology. https://bitsonblocks.net/2015/09/09/gentle-introduction-blockchain-technology/

14 Nolan Bauerle. What is Blockchain Technology?. https://www.coindesk.com/information/what-is-blockchain-technology

15 Tanya. Public and private keys. https://support.blockchain.com/hc/en-us/articles/360000951966-Public-and-private-keys

16 Luke Fortney. Blockchain, Explained. https://www.investopedia.com/terms/b/blockchain.asp

17 Blockchainhub. Blockchains & Distributed Ledger Technologies. https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/

18 Tanya. Transaction fees. https://support.blockchain.com/hc/en-us/articles/360000939903-Transaction-fees

19 World Crypto Index. How Nodes Work on the Blockchain. https://www.worldcryptoindex.com/how-nodes-work/

20 Damien Cosset. Blockchain: what is in a block?. https://dev.to/damcosset/blockchain-what-is-in-a-block-48jo

21 Bitcoin Exchange Guide News Team. Proof Of Work Vs Proof Of Stake – What Is POW & POS Mining?. https://bitcoinexchangeguide.com/proof-of-work-vs-proof-of-stake-mining/

22 Ben Whittle. What Is a Nonce? A No-Nonsense Dive into Proof of Work. https://coincentral.com/what-is-a-nonce-proof-of-work/

23 Lisk. Consensus Protocols. https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/consensus-protocols

24 Sudhir Khatwani. Explaining Hash Rate Or Hash Power In Cryptocurrencies. https://coinsutra.com/hash-rate-or-hash-power/

25 Jimi S.. Blockchain: how mining works and transactions are processed in seven steps. https://blog.goodaudience.com/how-a-miner-adds-transactions-to-the-blockchain-in-seven-steps-856053271476

26 Haseebrabbani. What is Hashing & Digital Signature in The Blockchain?. https://blockgeeks.com/what-is-hashing-digital-signature-in-the-blockchain/

27 Blockchainexpert. Types of Blockchain — Public, Private, and Consortium Block-chain. https://www.blockchainexpert.uk/blog/types-of-blockchain

28 Prableen Bajpai. Bitcoin Vs Ethereum: Driven by Different Purposes. https://www.investopedia.com/articles/investing/031416/bitcoin-vs-ethereum-driven-different-purposes.asp

29 Ethereum community. Mining. http://ethdocs.org/en/latest/mining.html

30 Prabath Siriwardena. The Mystery Behind Block Time. https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a?gi=6d8ffb3d4470

31 Alyssa Hertig. What is Ethereum?. https://www.coindesk.com/information/what-is-ethereum

32 Ethereum community. What is Ethereum?. http://ethdocs.org/en/latest/introduction/what-is-ethereum.html

33 BlockchainHub. Smart Contracts. https://blockchainhub.net/smart-contracts/

34 Anthonylewis2015. A gentle introduction to smart contracts. https://bitsonblocks.net/2016/02/01/gentle-introduction-smart-contracts/

35 Josh Quintal. Building Robust Smart Contracts with OpenZeppelin. https://truffleframework.com/tutorials/robust-smart-contracts-with-openzeppelin

36 OpenZeppelin. openzeppelin-solidity. https://github.com/OpenZeppelin/openzeppelin-solidity

37 OpenZeppelin. Build Secure Smart Contracts in Solidity.
https://openzeppelin.org/

38 EverestCrypto. Ethereum Essentials: Smart Contracts and ERC20 Standard.
https://everestcrypto.com/ethereum-essentials-smart-contracts-and-erc20-standard/

39 Solidity. Solidity.
https://solidity.readthedocs.io/en/v0.5.7/

40 Ethereum community. Ether.
http://ethdocs.org/en/latest/ether.html

41 Alexis Ulrich. The ether units of Ethereum.
https://www.languagesandnumbers.com/articles/en/ethereum-ether-units/

42 Blockgeeks. What is Ethereum Gas? The Best Step-By-Step Guide Ever!.
https://blockgeeks.com/guides/ethereum-gas-step-by-step-guide/

43 Walking Tree Technologies. Understanding Gas in Ethereum.
https://medium.com/coinmonks/understanding-gas-in-ethereum-53ad816f79ae

44 Ethereum community. Ether.
http://ethdocs.org/en/latest/ether.html#gas-and-ether

45 John Callaham.The history of Android OS: its name, origin and more.
https://www.androidauthority.com/history-android-os-name-789433/

46 Team AA. Android 9 Pie update tracker: When will your phone get it? (Updated April 30).
https://www.androidauthority.com/android-9-0-update-880718/

47 Melissa Chau, Ryan Reith. Smartphone Market Share.
https://www.idc.com/promo/smartphone-market-share/os

48 Google. Find the Google Play Store app.
https://support.google.com/googleplay/answer/190860?hl=en

49 Artyom Dogtiev. App Download and Usage Statistics (2018).
http://www.businessofapps.com/data/app-statistics/

50 Simon Sage. How to install Android apps.
https://www.androidcentral.com/android-apps-install

51 Statista. Number of apps available in leading app stores as of 3rd quarter 2018. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

52 Google Developers. Meet Android Studio. https://developer.android.com/studio/intro

53 Margaret Rouse. Android Studio. https://searchmobilecomputing.techtarget.com/definition/Android-Studio

54 Alex Mullis. Android Studio tutorial for beginners. https://www.androidauthority.com/android-studio-tutorial-beginners-637572/

55 Adam Sinicki. Android SDK tutorial for beginners. https://www.androidauthority.com/android-sdk-tutorial-beginners-634376/

56 Techotopia. Creating an Android Virtual Device (AVD) in Android Studio. https://www.techotopia.com/index.php/Creating_an_Android_Virtual_Device_(AVD)_in_Android_Studio

57 Antonio Angelino. Introduction to Virtualization Technologies. https://cloudacademy.com/course/introduction-to-virtualization-technologies/

58 Techopedia. Virtualization. https://www.techopedia.com/definition/719/virtualization

59 Node.js.dev. Introduction to Node.js. https://nodejs.dev/

60 Michael Aranda. What's the difference between JavaScript and ECMAScript?. https://medium.freecodecamp.org/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5?gi=466f8628c206

61 Red Hat, Inc. What is open source?. https://opensource.com/resources/what-open-source

62 V8.dev. What is V8?. https://v8.dev/

63 Node.js Foundation. ECMAScript 2015 (ES6) and beyond. https://nodejs.org/en/docs/es6/

64 Npm, Inc.. Downloading and installing Node.js and npm. https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

65 Npm, Inc.. Get npm!.
https://www.npmjs.com/get-npm

66 TutorialsTeacher, Node Package Manager.
https://www.tutorialsteacher.com/nodejs/what-is-node-package-manager

67 Npm, Inc.. Resources.
https://www.npmjs.com/resources

68 Npm, Inc.. Build amazing things.
https://www.npmjs.com/

69 Npm, Inc.. About the public npm registry.
https://docs.npmjs.com/about-the-public-npm-registry

70 Npm, Inc.. About npm.
https://docs.npmjs.com/about-npm/

71 Michael Wanyoike, Peter Dierx. A Beginner's Guide to npm — the Node Package Manager.
https://www.sitepoint.com/beginners-guide-node-package-manager/

72 Flaviocopes. npm global or local packages.
https://flaviocopes.com/npm-packages-local-global/

73 W3schools. What is npm?.
https://www.w3schools.com/whatis/whatis_npm.asp

74 Arui.tech. [Node.js] Why installing packages locally is much better than globally in npm.
https://arui.tech/en/why-locally-install-is-much-better-than-globally-in-npm/

75 Eric Lathrop. The Problem with `npm install --global`.
http://ericlathrop.com/2017/05/the-problem-with-npm-install-global/

76 Npm, Inc.. npm-folders.
https://docs.npmjs.com/files/folders.html

77 Margaret Rouse. Java Development Kit (JDK).
https://www.theserverside.com/definition/Java-Development-Kit-JDK

78 Stanford.edu. CS 193A: Android App Development, Winter 2019.
https://web.stanford.edu/class/cs193a/android-studio.shtml

79 Oracle. Website.
https://www.oracle.com/index.html

80 Oracle. Java SE Development Kit 8 Downloads.
https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

81 Matthew Tyson. What is the JRE? Introduction to the Java Runtime Environment.
https://www.javaworld.com/article/3304858/what-is-the-jre-introduction-to-the-java-runtime-environment.html

82 Seymour. Java JRE vs JDK.
http://javacodedepot.com/java/java-jre-vs-jdk

83 Facebook Inc.. JavaScript Environment.
https://facebook.github.io/react-native/docs/javascript-environment

84 Bonnie Eisenman. Chapter 1. What Is React Native?.
https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

85 Amit Ashwini. What Is The Difference Between React.js and React Native?.
https://www.cognitiveclouds.com/insights/what-is-the-difference-between-react-js-and-react-native/

86 Ideamotive. CHOOSING REACT NATIVE FOR YOUR MOBILE TECH STACK.
https://ideamotive.co/react-native-development-guide/

87 Madesh Venkatraman. Understanding the basic components of React native.
https://habiletechnologies.com/blog/understanding-the-basic-components-of-react-native/

88 Nicholas Congleton. The 10 Best JavaScript ES6 Features.
https://www.lifewire.com/best-javascript-es6-features-4579821

89 Brandon Morelli. A Simple Guide to ES6 Promises.
https://codeburst.io/a-simple-guide-to-es6-promises-d71bacd2e13a

90 Npm, Inc.. react-native-cli.
https://www.npmjs.com/package/react-native-cli

91 Truffle Suite. Dashboard.
https://truffleframework.com/dashboard

92 Truffle Suite. Welcome to Ganache CLI.
https://github.com/trufflesuite/ganache-cli/blob/master/README.md

93 Truffle Suite. Ganache Quickstart.
https://trufflframework.com/docs/ganache/quickstart

94 Damien Cosset. Ethereum Development: Getting started.
https://dev.to/damcosset/ethereum-development-getting-started-m09

95 Truffle Suite. Truffle Overview.
https://trufflframework.com/docs/truffle/overview

96 Ross Bulat. Introduction to the Truffle Suite and Dapp Development Pipeline.
https://medium.com/@rossbulat/introduction-to-the-truffle-suite-and-dapp-development-pipeline-1b33bb8228d4

97 Truffle Suite. TRUFFLE BOXES.
https://trufflframework.com/boxes

98 Jamal Eason. Android Emulator - AMD Processor & Hyper-V Support.
https://android-developers.googleblog.com/2018/07/android-emulator-amd-processor-hyper-v.html

99 Npm, inc.. Windows-Build-Tools.
https://www.npmjs.com/package/windows-build-tools

100 Google Developers. Environment variables.
https://developer.android.com/studio/command-line/variables