



UNIVERSITY OF TUNIS
TUNIS BUSINESS SCHOOL



GRADUATION FINAL PROJECT REPORT

As a partial fulfillment of the degree of
BACHELOR DEGREE IN BUSINESS ADMINISTRATION

MAJOR : INFORMATION TECHNOLOGY

MINOR : BUSINESS ANALYTICS

Elaborated by

Mechergui Rami

DECENTRALIZED FUNDRAIDING PLATFORM

Professional Advisor: **DR. BEN MESSOUAD MONTASSAR**

Academic Advisor: **MR. BEN MILED MONDHER**

Hosted by: INFINITY MANAGEMENT



Tunis, September 2022

I hereby grant permission to the student **MECHERGUI RAMI** to submit his graduation internship report for the purpose of project defense

Professional Advisor, **MR. BEN MILED MONDHER**

signature and stamp

I hereby grant permission to the student **MECHERGUI RAMI** to submit his graduation internship report for the purpose of project defense

Academic advisor, **DR. BEN MESSOUAD MONTASSAR**

Signature

Dedicated,
To you, my inspiring parents,
To my Brother and Sister,
To all those I love,
for being the pillows, role models,
cheer-leading squad and sounding boards
I have needed

ACKNOWLEDGEMENTS

This report is the result of a rewarding internship at Infinity Company during my graduating year. That Pleasure I owe to the interaction I have Had with my supervisors, colleagues, and coworkers.

A special gratitude I would like to give to my supervisors Mr. Mondher Ben Miled System Architect for the patient guidance, encouragements and advice they have provided me during my internship. They always listen carefully to my questions and they guided my resolution to any faced problem.

A special Thanks goes to my academic advisor Dr. Montassar Ben Massaoud on his unwavering support, guidance and encouragements. Thank you for pushing me to expand my limits. I owe a huge debt of gratitude to my friends and family for their constant support and continuous encouragement during my years of study and for believing in my ability to thrive.

Without them, this achievement would not have been possible.

Finally, I want to express my sincere gratitude to the jury board members, in the hopes that they will find the clarity and commitment they seek in this report.

Contents

General Introduction	1
1 General Presentation	2
1.1 Introduction	3
1.1.1 Academic Context	3
1.1.2 Host Organization	3
1.2 Study of the existing	4
1.2.1 Description and criticism of the existing	4
1.2.2 Proposed solution	4
1.3 conclusion	5
2 study and project planning	6
2.1 Introduction	7
2.2 Identification of actors	7
2.3 Identification of needs	7
2.3.1 Funcational needs	7
2.3.2 Non-Funcational needs	9
2.4 Global use case diagram	10
2.5 Software Architecture	10
2.5.1 Physical Architecture	10
2.5.2 Logical Architecture	11
2.6 Development environment	12
2.6.1 Software environment	12
2.6.2 Used technologies	13
2.6.3 Version Control Tool :	15
2.6.4 Testing Tool:	16
2.6.5 Documentation tool :	16
2.7 Conclusion	17
3 Sprint 01 : Setting up Private Ethereum Blockchain	18
3.1 A first look at Web3	19
3.1.1 web2.0	19

3.1.2	web3.0	19
3.2	Decentralized Application	20
3.2.1	Ethereum as platform	20
3.2.2	Go-Ethereum	20
3.3	Consensus mechanism	22
3.4	Technical Steps to setup private blockchain	24
3.5	Choosing the right provider	30
3.5.1	What is a provider ?	30
3.6	Monitoring private blockchain	32
3.6.1	What is Ethstats ?	32
	Conclusion générale	34

List of Figures

1.1	Infinity Management Logo	3
2.1	Physical Architecture	11
2.2	Application Logical Architecture	11
2.3	Docker Logo	12
2.4	Visual Studio Logo	13
2.5	Go-Ethereum Logo	13
2.6	ReactJS Logo	14
2.7	Nodejs Logo	14
2.8	MongoDB Logo	14
2.9	Traefik Logo	15
2.10	Mailjet Logo	15
2.11	Github Logo	16
2.12	Truffle Logo	16
2.13	Gitbook Logo	17
3.1	Different Geth versions released by Ethereum foundation	21
3.2	Screenshot of the first node creation showing its public address	25
3.3	The genesis Block	26
3.4	Screenshot of the first node creation showing its public address	28
3.5	Screenshot showing 3 nodes connected to the blockchain through bootnode	29
3.6	Screenshot showing Nodes Peers	30
3.7	Screenshot showing Nodes Peers	30
3.8	Web3js	32
3.9	Screenshot of Ethstat Dashboard	33

List of Tables

- 3.1 comparative table highlighting the main differences between Web 2.0 and Web 3.0 . . . 20
- 3.2 comparative table highlighting differences between Geth with Tools and Geth without
Tools 22
- 3.3 comparative table highlighting differences between different types of consensus mechanism 23
- 3.4 comparative table highlighting differences between different types of consensus mechanism 31

Liste des abréviations

- **CV** = Curriculum Vitae
- **HR** = Human resources
- **J2EE** = Java 2 Enterprise Edition
- **JPA** = Java Persistence API
- **SQL** = Structured Query Language
- **UML** = Langage de Modélisation Unifié

General Introduction

In the dynamic and ever-evolving landscape of [relevant field/industry], the emergence of blockchain technology has sparked profound interest and transformative potential. Decentralized systems built on blockchain, such as Ethereum, have paved the way for innovative applications and solutions across various sectors. Among these, one area of significant interest is the utilization of blockchain in the context of startup fundraising and investment.

This thesis delves into the exploration and analysis of a groundbreaking decentralized application (dApp) based on the Ethereum blockchain. The primary focus is to revolutionize the traditional startup fundraising model, enabling startups to connect with a diverse pool of investors and allowing investors to discover promising opportunities in an efficient, secure, and transparent manner.

The central objective of this thesis is to design, develop, and evaluate the effectiveness of a custom-built dApp tailored specifically for startups and investors. The dApp will leverage Ethereum's smart contract capabilities to ensure security and trust between both parties. Through this research, we seek to understand the potential advantages and challenges of deploying blockchain technology in the context of startup fundraising, while exploring the impact on traditional fundraising practices.

GENERAL PRESENTATION

Plan

1	Introduction	3
2	Study of the existing	4
3	conclusion	5

Introduction

This chapter aims to provide a comprehensive overview of our project, encompassing various aspects relevant to its scope. It will delve into the significance of different contexts.

1.1 Introduction

1.1.1 Academic Context

This project represents the culmination of our Bachelor of Science degree in Business Administration, taking the form of an end-of-studies internship. The internship spanned from February 8th to July 8th, 2023, during which we had the privilege of joining the esteemed Infinity Management team.

En 2015, la société fusionne avec Steria, formant le groupe Sopra Steria. Elle est implantée dans plus de 20 pays avec environ 42000 salariés.[1]

1.1.2 Host Organization

Infinity Management is a player in the fields of consulting and IT engineering services. It assists clients in conceptualizing and implementing the digital era from subsidiaries in Tunisia, France, and Spain, understanding their needs and offering them solutions tailored to their strategic development. Its objective is to help clients create business-oriented solutions through our experienced and highly qualified analysts and architects. Its identity is based on three main pillars: Consulting, Human, and Innovation.



Figure 1.1: Infinity Management Logo

1.1.2.1 Activities and areas of expertise

Our hosting company operates across various sectors, offering a diverse range of solutions in different sectors : industry, automotive, banking, finance, insurance, telecommunications, and more...

1.2 Study of the existing

In this section, we conduct a comprehensive study of the current startup fundraising landscape, highlighting the challenges faced by emerging startups. Many startups encounter significant liquidity issues and struggle to find potential investors willing to support their projects. Conversely, a substantial number of investors actively seek new and promising investment opportunities to diversify their portfolios. The traditional fundraising model, while established, often poses obstacles for startups, impeding their growth potential, and restricting access to a diverse pool of investors. Moreover, such limitations can have significant implications on the economy of each country, as potential innovative projects may be stifled, affecting job creation and overall economic growth.

1.2.1 Description and criticism of the existing

We delve deeper into the shortcomings of the traditional fundraising model, pointing out the time-consuming and resource-intensive processes that startups must navigate. Traditional fundraising methods often involve substantial administrative burdens, complex legal frameworks, and high transaction costs, making it challenging for startups, particularly those in their early stages, to attract investors effectively. Furthermore, the lack of accessibility to a global network of investors further restricts the opportunities available to startups. These limitations have a considerable impact on economic growth, as potentially transformative projects may face difficulty in securing the necessary funds to materialize their innovations.

1.2.2 Proposed solution

To address the liquidity challenges faced by startups and the demand from investors for innovative projects, we propose a revolutionary solution based on blockchain technology. Our solution involves the development and implementation of a decentralized application (dApp) built on the Ethereum blockchain. By harnessing the power of blockchain and smart contracts, our dApp provides a transparent, secure, and efficient platform for startups to initiate fundraising campaigns. This platform also enables investors from across the globe to discover and invest in startups that align with their investment preferences.

Through our Ethereum-based dApp, startups gain access to a broader pool of potential investors, allowing them to attract funding more efficiently and with reduced administrative overhead. This increased accessibility and inclusivity foster a thriving ecosystem for entrepreneurship, encouraging the growth of innovative projects and promoting economic development in various regions.

1.3 conclusion

In conclusion, this thesis addresses the critical liquidity challenges faced by startups and the quest of investors for promising investment opportunities. The traditional fundraising model's limitations can hinder economic growth by impeding innovation and job creation. To overcome these obstacles, we propose a transformative solution through the development and deployment of a decentralized application on the Ethereum blockchain. By leveraging blockchain technology, our dApp aims to revolutionize startup fundraising, providing startups with improved access to capital while offering investors diverse and transparent investment opportunities. By fostering a conducive environment for entrepreneurial ventures, we envision a future where startups flourish, investors thrive, and economic growth is fueled by innovation and collaboration. Through this research, we contribute to the evolving landscape of blockchain and its potential to shape a more inclusive and dynamic economy.

STUDY AND PROJECT PLANNING

Plan

1	Introduction	7
2	Identification of actors	7
3	Identification of needs	7
4	Global use case diagram	10
5	Software Architecture	10
6	Development environment	12
7	Conclusion	17

2.1 Introduction

The study and planing stage is designed to determine the different functionalities expected from the system. Indeed, in this chapter we first present the actors involved in our system. Then, we begin the study of functional and non-functional needs. These needs will be expressed in the form of use case diagrams that detail the possible scenarios that the different actors can carry out. and we will end with the global class diagram.

2.2 Identification of actors

Any interactive system must ensure and facilitate interaction with its users. An actor represents the role of an external entity operating the system through its various interfaces. We specify the actors of the system that are relevant to this project, which are divided as follows:

Main Actors :

- Token Issuer
- Investor
- Administrator

Secondary Actors :

- System

2.3 Identification of needs

In this section, we will expose the functional and non-functional needs of our application.

2.3.1 Funcational needs

1.Token Issuer functional needs :

1. **User Registration :** Token Issuer should be able to register on the platform by providing necessary details such as name, contact information, and verifying their identity through a secure authentication process.
2. **Token Creation:** Token Issuer should have the ability to create and issue their own tokens on the platform. They should be able to define parameters such as token name, symbol, total supply, and any additional metadata or attributes associated with the tokens.

3. **Token Sale Configuration :** The Token Issuer should be able to configure the token sale event. They should be able to set the fundraising amount they aim to achieve and define the number of tokens that will be available for sale during the fundraising event.
4. **Concept Presentation :** The Token Issuer should be able to present their startup concept on the platform. They should have the capability to provide a detailed description of their startup, including their mission, vision, target market, and unique selling points. Additionally, they should be able to share presentations, videos, and any other relevant media that can help investors understand their concept better.

2.Investor Functional Needs:

1. **User Registration:** Investors should have the ability to register on the platform by providing necessary details such as name, contact information, and verifying their identity through a secure authentication process.
2. **Token Discovery :** Investors should be able to explore and discover various startups and their associated tokens on the platform. They should have access to a search feature, categories, or filters to narrow down their search based on their investment preferences.
3. **Concept Review :** Investors should be able to view detailed information about each startup and their concept on the platform. They should have access to the description, presentations, videos, and any other media shared by the Token Issuers to evaluate the viability and potential of the startup.
4. **Investment Process :** Investors should have the capability to participate in the token sale event. They should be able to purchase tokens from the startups they are interested in by following the provided instructions and using the supported payment methods.

3.Administrator Functional Needs:

1. **User Registration and Management** Administrators should have the ability to register on the platform with appropriate permissions and access levels. They should also be responsible for managing user accounts, including approving new registrations, suspending or banning accounts when necessary, and handling account deletion requests.
2. **Token Issuer Verification** Administrators should have the capability to verify the identity and authenticity of token issuers before their concepts and token sales are listed on the platform. They should perform necessary checks, such as reviewing documents, conducting background checks, and ensuring compliance with platform policies.

3. **Dispute Resolution** Administrators should handle and resolve any disputes or conflicts that may arise between token issuers and investors.
4. **Communication and Support** Administrators serve as points of contact for users, addressing their queries, concerns, and providing support as needed.
5. **User Management** Ability for the administrator to manage user accounts, including registration approval, suspension, and deletion.

2.3.2 Non-Functional needs

The non-functional needs explain all of the constraints to which the system is subjected in order for it to be realized and function properly.

1. **Security** : The DApp should implement robust security measures to protect user data, transactions, and sensitive information. This includes encryption of data, secure user authentication, and protection against common security threats such as unauthorized access, data breaches, and hacking attempt
2. **Scalability** : The DApp should be designed to handle a growing number of token issuers, startups, and investors without compromising performance or user experience. It should scale efficiently to accommodate increased transaction volumes and user activity.
3. **Reliability** : The DApp should be reliable and available for users at all times. It should minimize downtime, implement backup and recovery mechanisms, and handle system failures gracefully.
4. **Usability** : The user interface (UI) and user experience (UX) of the DApp should be intuitive and user-friendly. It should require minimal training for users to understand and navigate the platform effectively. The DApp should be accessible to users with different levels of technical expertise.
5. **Interoperability** : The DApp should be designed to integrate and interact seamlessly with external systems or blockchain platforms, ensuring compatibility and smooth data exchange.
6. **Compliance** : The DApp should adhere to relevant legal and regulatory requirements, such as data privacy regulations, financial regulations, and industry-specific regulations. It should implement necessary compliance measures, including KYC and AML procedures, to meet regulatory standards.

7. **Performance** : The DApp should be optimized for high performance, with fast response times and minimal latency. It should be capable of handling concurrent user requests, large transaction volumes, and complex computations efficiently
8. **Maintainability** : The DApp should be designed and developed using modular and well-documented code to facilitate ease of maintenance and future enhancements. It should follow best practices, coding standards, and version control systems for efficient collaboration among developers.
9. **Data Privacy** : The DApp should protect user privacy and ensure that personal information is collected, stored, and processed in accordance with data privacy regulations. It should provide users with control over their personal data and enable them to manage their privacy settings.
10. **Auditability** : The DApp should provide transparent tracking and auditing of token transactions, fundraising amounts, and user interactions. It should allow for the verification and validation of data to ensure the integrity and accuracy of information.

These non-functional requirements are essential to ensure the overall quality, performance, security, and user satisfaction of the DApp.

2.4 Global use case diagram

2.5 Software Architecture

The underlying structures of a software system, as well as the discipline of building such structures and systems, are referred to as software architecture. Each structure is made up of software elements, relationships between them, and attributes of each elements and relationships.

2.5.1 Physical Architecture

A physical architecture is a collection of physical pieces (system components and physical The underlying structures of a software system, as well as the discipline of building such structures and systems, are referred to as software architecture. Each structure is made up of software elements, relationships between them, and attributes of each elements and relationships. The underlying structures of a software system, as well as the discipline of building such structures and systems, are referred to as software architecture. Each structure is made up of software elements, relationships between them, and attributes of each elements and relationships.) that provide an architectural solutions for a product, service, or firm while matching logical architectural elements and technical criteria.

The figure n below represent the physical architecture of our application.

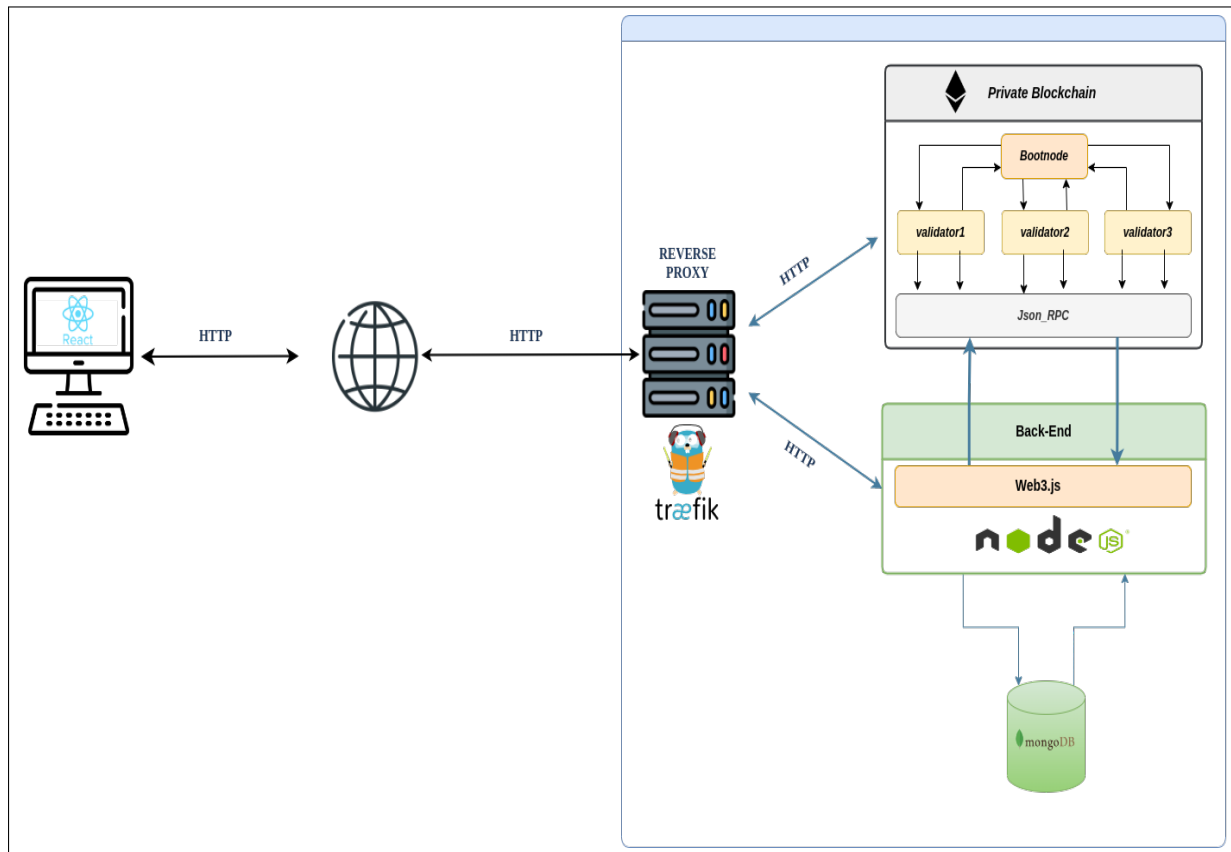


Figure 2.1: Physical Architecture

2.5.2 Logical Architecture

The identification of the System components (and their dependencies) that deliver the software services required to achieve the business goals criteria for deployment is referred to as logical architecture. Logical Architecture is the technique and documentation used to provide a more accurate, comprehensive, and clear description of system components by providing well-defined interfaces and component specifications, as well as essential architectural procedures. The figure 2.4 below represent our application logical architecture that is based on MVC concept.

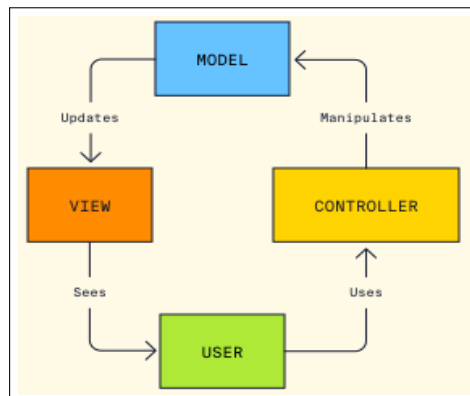


Figure 2.2: Application Logical Architecture

MVC (Model-View-Controller) is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance. The three parts of the MVC software-design pattern can be described as follows:

- **Model** : Manages data and business logic.
- **View** : Handles layout and display.
- **Controller** : Routes commands to the model and view parts.

2.6 Development environment

2.6.1 Software environment

In this part, we will list the different environment characteristics(Software,Technologies) that has been used to develop this project.

- **Docker** A platform that enables developers to build, deploy, and run applications using containerization. Docker allows applications and their dependencies to be packaged into containers, providing a consistent and isolated environment across different systems.

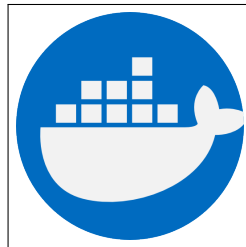


Figure 2.3: Docker Logo

[2].¹

- **VS Code** Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

¹<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.



Figure 2.4: Visual Studio Logo

[2]. ²

2.6.2 Used technologies

We will list in this section the different software solutions that were used during the implementation of our mobile application.

- **Go-ethereum (geth)** : Geth (go-ethereum) is a Go implementation of Ethereum - a gateway into the decentralized web. Geth has been a core part of Ethereum since the very beginning. Geth was one of the original Ethereum implementations making it the most battle-hardened and tested client. Geth is an Ethereum execution client meaning it handles transactions, deployment and execution of smart contracts and contains an embedded computer known as the Ethereum Virtual Machine. Running Geth alongside a consensus client turns a computer into an Ethereum node.

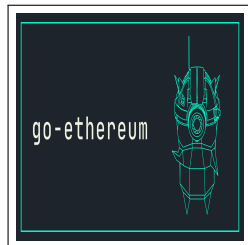


Figure 2.5: Go-Ethereum Logo

[2]. ³

- **React js** : React is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta and a community of individual developers and companies. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js.

²<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

³<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

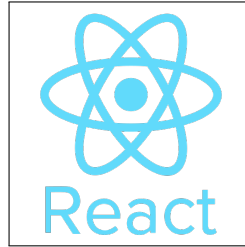


Figure 2.6: ReactJS Logo

[2].⁴

- **Node.js :** Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser

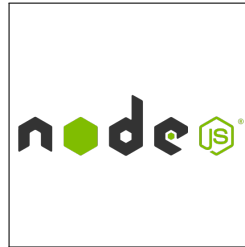


Figure 2.7: Nodejs Logo

[2].⁵

- **Mongo-db :** MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License which is deemed non-free by several distributions.



Figure 2.8: MongoDB Logo

[2].⁶

⁴<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

⁵<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

⁶<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

- **Traefik :** Traefik is a modern, cloud-native HTTP reverse proxy and load balancer. It provides strong support for microservices and integrates with Docker, Kubernetes, Rancher, and Consul. Traefik requires minimal configuration and uses automated service discovery to inject routes to backend services. It accomplishes this by monitoring the API of the underlying orchestration or registry service without the need for manual configuration.

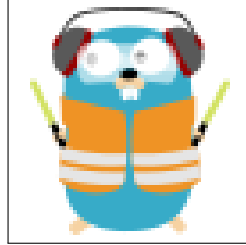


Figure 2.9: Traefik Logo

[2]. ⁷

- **Mailjet :** When searching for an email marketing solution, going with a provider that scales and offers advanced features and ease of use is crucial. Mailjet lets you build email templates and email campaigns, automate, send, and monitor your email marketing campaigns. Ensure your emails hit the inbox with top-notch deliverability. (Source : <https://www.mailgun.com/email-marketing/mailjet/>)



Figure 2.10: Mailjet Logo

[2]. ⁸

2.6.3 Version Control Tool :

In this part, we will list the different environment characteristics(Software,Technologies) that has been used to develop this project.

- **GitHub :** A web-based hosting service for version control using Git. GitHub allows developers to collaborate on projects, track changes, and manage code repositories. It provides features like pull requests, issue tracking, and branching for efficient collaboration within development teams.

⁷<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

⁸<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.



Figure 2.11: Github Logo

[2].⁹

2.6.4 Testing Tool:

In this part, we will list the different environment characteristics(Software,Technologies) that has been used to develop this project.

- **Truffle** A development framework for Ethereum that simplifies the process of building, testing, and deploying smart contracts. Truffle provides a suite of tools and utilities for compiling contracts, managing migrations, and running automated tests.



Figure 2.12: Truffle Logo

[2].¹⁰

2.6.5 Documentation tool :

- **Gitbook** : GitBook helps you help your users with easy-to-publish, intuitive to use, highly searchable docs for your documentation. Source Wikipedia

⁹<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

¹⁰<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.



Figure 2.13: Gitbook Logo

[2].¹¹

2.7 Conclusion

¹¹<https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>.

SPRINT 01 : SETTING UP PRIVATE ETHEREUM BLOCKCHAIN

Plan

1	A first look at Web3	19
2	Decentralized Application	20
3	Consensus mechanism	22
4	Technical Steps to setup private blockchain	24
5	Choosing the right provider	30
6	Monitoring private blockchain	32

3.1 A first look at Web3

In this section, I will discuss the the main differences between Web 2.0 and Web 3.0

3.1.1 web2.0

Web2 is the second generation of the World Wide Web, also known as the “read/write “web. It is a move to interactive functionality and compatibility through the following features :

3.1.1.1 features of web2.0

- User-generated content
- Transparency in data and integrations
- The web is a platform and not a network
- Software as a Service (SaaS) via API implementation technolog

Failure to remove outdated information

3.1.1.2 Limitation of web2.0

- Dependency on the internet to network with people
- Keyword-based
- The web is a platform and not a network
- A high number of hackers and frauds
- Lack of intelligence

3.1.2 web3.0

Web3 is the “**read-write-own**” mode of the internet that uses blockchains, NFTs and cryptocurrencies to empower the users in the form of ownership.

3.1.2.1 Features of web3

- **Decentralized** : Unlike web2, where the internet is controlled and owned by centralized entities, web3 allows ownership to be distributed amongst its users and builders.
- **Permissionless** : Everybody has equal access to web3, and nobody is excluded.

- **Native payments** : Web3 uses cryptocurrencies to send and spend money online instead of relying upon the old infrastructure of banks and payment processors.
- **Trustless** : It uses incentives and economic mechanisms to operate rather than relying upon trusted third parties.

	Web2.0	web3.0
Data	- Centralized	- Decentralized
Control	- Control by centralized entities	- Shared control among participants
Trust	- Trust in centralized authorities	- Trust in cryptography and consensus
Interactions	- Limited user-to-user interactions	- Rich user-to-user interactions
Content Ownership	- Platform-owned	- User-owned
Monetization	- Ad-based revenue models	- Token-based economies
Smart Contracts	- Not native to the platform	- Native support for smart contracts
Privacy	- Relies on trust in platform providers	- Enhanced privacy through encryption
Scalability	- Control by centralized entities	- Scalability through blockchain
Innovation	- Centralized decision-making	- Open and permissionless innovation

Tableau 3.1: comparative table highlighting the main differences between Web 2.0 and Web 3.0

3.2 Decentralized Application

A decentralized application is an application built on a decentralized network that combines a smart contract and a front-end user interface. It is a type of distributed open-source software application that runs on a peer-to-peer blockchain network. The use of blockchain in dApps allows them to process data through distributed networks and execute transactions.

3.2.1 Ethereum as platform

Ethereum is a technology for building apps and organizations, holding assets, transacting and communicating without being controlled by a central authority. It is the base of a new, decentralized internet.

3.2.2 Go-Ethereum

Geth (go-ethereum) is a Go implementation of Ethereum - a gateway into the decentralized web. Geth has been a core part of Ethereum since the very beginning. Geth was one of the original Ethereum implementations making it the most battle-hardened and tested client. Geth is an Ethereum execution

client meaning it handles transactions, deployment and execution of smart contracts and contains an embedded computer known as the Ethereum Virtual Machine. Running Geth alongside a consensus client turns a computer into an Ethereum node.

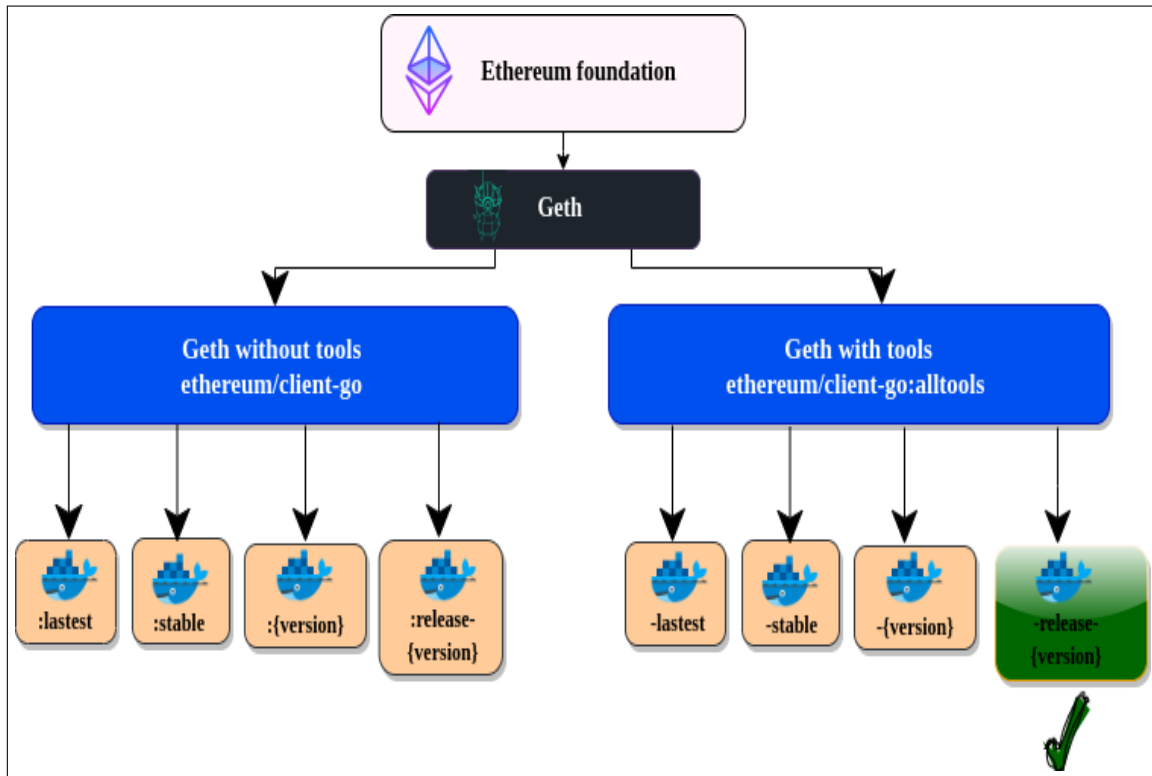


Figure 3.1: Different Geth versions released by Ethereum foundation

The Ethereum Foundation’s mission is to promote and support the development of decentralized applications on the Ethereum platform, and Geth is one of the core components of the Ethereum network. There are two versions of Geth available for download: Geth without tools and Geth with tools : Geth without tools is a lightweight version of the software. Geth with tools, on the other hand, includes additional tools and features such as a JavaScript console, a JavaScript API, and other command-line tools.

	Geth without tools	Geth with tools
latest	- the latest development version of Geth (default)	- the latest development version of the Ethereum tools
stable	- the stable version of Geth at a specific version number	- the stable version of the Ethereum tools at a specific version number
version	- the stable version of Geth at a specific version number	- the stable version of the Ethereum tools at a specific version number

release (version)	- the latest stable version of Geth at a specific version family	- the latest stable version of the Ethereum tools at a specific version family
------------------------------	--	--

Tableau 3.2: comparative table highlighting differences between Geth with Tools and Geth without Tools

3.3 Consensus mechanism

A consensus algorithm is a mechanism that allows users or machines to coordinate in a distributed setting. It needs to ensure that all agents in the system can agree on a single source of truth, even if some agents fail

	Description	Scalable	Efficiency	security	decentralized	Fast
Proof of Work (PoW)	- Requires miners to solve complex mathematical problems to add blocks to the chain. The first miner to solve the problem and add the block is rewarded with cryptocurrency.	- No	- No	- High	- Yes	- No
Proof of Stake (PoS)	- Validators are chosen based on the amount of cryptocurrency they hold and are willing to "stake" as collateral. Validators are rewarded with transaction fees.	- Yes	- Yes	- High	- Yes	- Yes

Proof of Authority (PoA)	-Validators are known entities that are authorized to add blocks to the chain. Validators are typically large organizations or institutions.	- Yes	- Yes	- High	- No	- No
Delegated Proof of Stake (DPoS)	- Similar to PoS, but instead of all validators having equal voting power, token holders elect a smaller group of delegates to validate transactions on their behalf.	- Yes	- Yes	- Medium	- No	- No

Tableau 3.3: comparative table highlighting differences between different types of consensus mechanism

Proof of Work (PoW) is the superior consensus mechanism when compared to others like Proof of Authority (PoA). While PoA may have a small group of known and authorized validators, it inherently lacks the robustness and decentralization provided by PoW. PoW's energy-intensive computations and extensive network of validators contribute to its unparalleled security, making it highly resistant to attacks and manipulation. PoW's decentralized nature ensures that no single entity has control over the network, reducing the risk of potential central points of failure or censorship. Although PoA may prioritize efficiency, it comes at the cost of sacrificing the principles of decentralization and opens up possibilities of collusion and manipulation within the small validator group. In contrast, PoW remains the preferred consensus mechanism for blockchain systems seeking true immutability, trustlessness, and a highly secure foundation.

3.4 Technical Steps to setup private blockchain

1. Pull the specified docker image from Dockerhub and run a container out of this image:

```
ramimechergui@TravelMate-P259-M:~$ docker pull ethereum/client-go:alltools-release-1.9
```

2. Create a directory dedicated to your private network

```
1 mkdir home/Private_Ethereum_Blockchain
2 cd home/Private_Ethereum_Blockchain
```

3. Create Nodes:

Nodes are considered as wallets . They hold a **public** and **private key** that are required to interact with the whole blockchain . Each Node should be able to identify itself in the blockchain with its public key and sign the transaction with its private key

why running our nodes ?

Running your own node enables you to use Ethereum in a private, self-sufficient and trustless manner. You don't need to trust the network because you can verify the data yourself with your client. "**Don't trust, verify**" is a popular blockchain mantra.

4. **Create Accounts (Wallets)** : Now, you have to create an account for each node. So, you have to repeat the following step for each Node

```
1 /home/Private_Ethereum_Blockchain # geth --datadir Node1 account new
```

- **geth** is a command-line interface program for running Ethereum nodes on your computer.
- **--datadir Node2** specifies the data directory for the Ethereum node that you are creating. In this case, the data directory is called "Node2".
- **account new** is a command that you can use with geth to create a new Ethereum account.

Give **password** to your Node : **pwdNode1**

```
1 // Saving the password of your node in Node1
2 // as we are gonna use it later to unlock the account
3 /home/Private_Ethereum_Network # echo 'pwdNode1' > Node1/password.txt
```

save the public keys of your nodes inside a file accounts.txt

```
1 /home/Private_Ethereum_Network # echo 'public address of 1st Node'>> accounts.txt
```

```

/home/Private_Ethereum_Blockchain # geth --datadir Node3 account new
INFO [03-24|22:44:46.007] Maximum peer count          ETH=50 LES=0 total=50
INFO [03-24|22:44:46.008] Smartcard socket not found, disabling  err=stat /run/pcscd/pcscd.comm: no such file or directory
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:  0x50efBc9CAAE5eD53F4E2676BB0B7c985d3ac0d97
Path of the secret key file: Node3/keystore/UTC--2023-03-24T22-44-58.392330465Z--50efbc9cae5ed53f4e2676bb0b7c985d3ac0d97

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!

```

Figure 3.2: Screenshot of the first node creation showing its public address

5. **Create the genesis block:** A genesis file is the file used to initialize the blockchain. The very first block, called the genesis block, is crafted based on the parameters in the genesis.json file.

```
/home/private_ethereum_network # touch genesis.json
```

```

1  {
2      "config": {
3          "chainId": 12345,
4          "homesteadBlock": 0,
5          "eip150Block": 0,
6          "eip155Block": 0,
7          "eip158Block": 0,
8          "byzantiumBlock": 0,
9          "constantinopleBlock": 0,
10         "petersburgBlock": 0,
11         "istanbulBlock": 0,
12         "berlinBlock": 0,
13         "ethash": {}
14     },
15     "difficulty": "1",
16     "gasLimit": "8000000",
17     "alloc": {
18         "public address of the first node  ": {"balance": "10000000000000000000"},
19         "public address of the second node ": {"balance": "20000000000000000000"},
20         "public address of the third node  ": {"balance": "30000000000000000000"}
21     }

```

```
22 }
```

```
23 }
```

The JSON code you provided is a basic configuration file for setting up a private Ethereum network. Here is a breakdown of what each field does:

- **config** : This object defines the network's configuration parameters, including:
- **chainId** : This sets the chain ID for the network. `homesteadBlock`, `eip150Block`, `eip155Block`, `eip158Block`, `byzantiumBlock`, `constantinopleBlock`, `petersburgBlock`, `istanbulBlock`, `berlinBlock`: These fields define the block numbers at which various network upgrades were introduced.
- **ethash** : This object defines the Ethash mining algorithm's parameters. `difficulty`: This sets the difficulty level for mining blocks on the network. In this case, it is set to a low value of 1 for testing purposes.
- **gasLimit** : This sets the maximum amount of gas that can be used in a block. In this case, it is set to 8 million.
- **alloc** : This object specifies the initial account balances for the nodes in the network. Each account is identified by its public address, and its initial balance is specified in wei (the smallest unit of Ether). In this example, there are three nodes with different initial balances. Allocated accounts should be the address of the previously created accounts as they are going to pre-funded with free ether

Allocated accounts should be the address of the previously created accounts as they are going to pre-funded with free ether

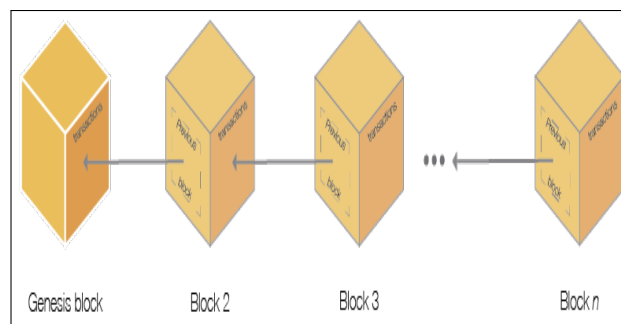


Figure 3.3: The genesis Block

6. **Initialize nodes:** Each node should be initialize with the same genesis file standards.

```
1 /home/Private_Ethereum_Blockchain # geth --datadir Node1/ init genesis.json
```

- `-datadir Node1/` : This specifies the data directory for the Ethereum node that you are initializing. In this case, the data directory is called `Node1/`.
- `init` This is a command that you can use with `geth` to initialize a new Ethereum blockchain.
- `genesis.json`] This is the path to the JSON file that contains the configuration for the genesis block

7. **Create a bootnode and Starting its service** : A bootnode only purpose is to helping nodes discovering each others (remember, the Ethereum blockchain is a peer-to-peer network). Nodes could have dynamic IP, being turned off, and on again. The bootnode is usually ran on a static IP

```
1 /home/private_ethereum_network # bootnode -genkey boot.key
```

`bootnode` is the name of the executable file that runs the bootnode tool. `-genkey boot.key` generates a new private key file with the name `boot.key`. This key will be used by the bootnode to sign its identity and allow other nodes to connect to it.

```
1 // Starting the bootnode
```

```
2 /home/private_ethereum_network # bootnode -nodekey boot.key -verbosity 9 -addr :30310
```

`bootnode` is the name of the executable file that runs the bootnode tool.

- **`-nodekey boot.key`** : specifies the private key file to use for the bootnode. This key is used to sign the node's identity and allow other nodes to connect to it.
- **`-verbosity 9`** : sets the verbosity level for the log output of the bootnode. In this case, it's set to the highest level, which means that the bootnode will output detailed debug information about its operation.
- **`-addr :30310`** : specifies the UDP listening address for the bootnode. In this case, it's set to listen on all available network interfaces on port 30310.

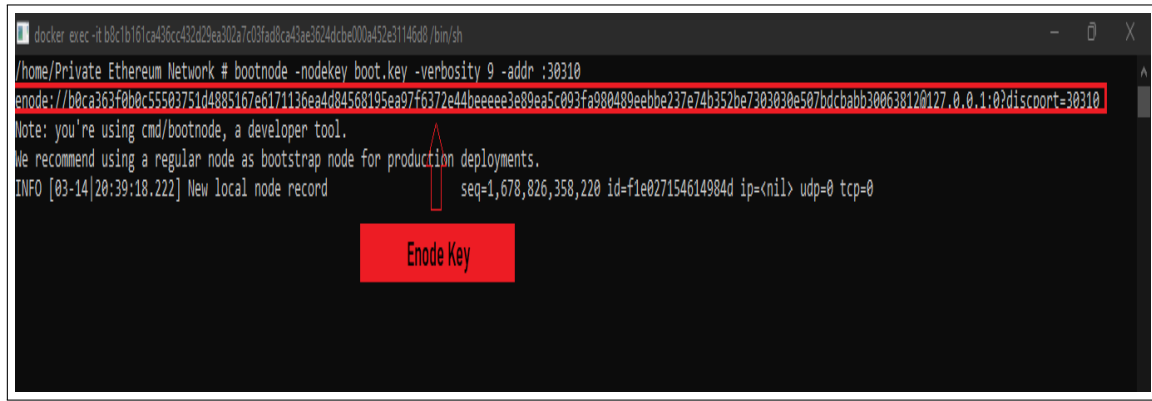


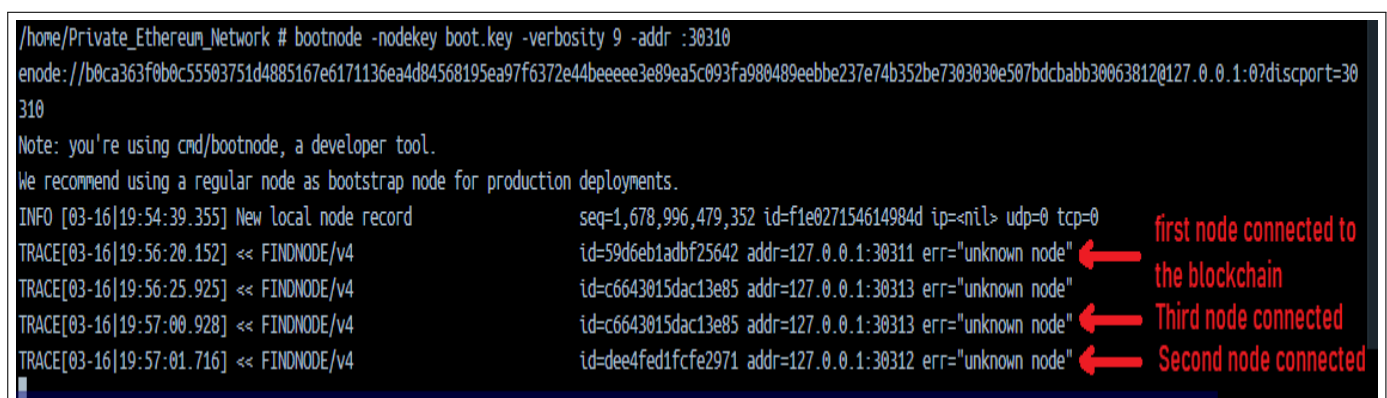
Figure 3.4: Screenshot of the first node creation showing its public address

8. **Starting your services :** The DApp should be designed and developed using modular and well-documented code to facilitate ease of maintenance and future enhancements. It should follow best practices, coding standards, and version control systems for efficient collaboration among developers.

```
1 // Starting the bootnode
2 /home/Private_Ethereum_Blockchain # geth --datadir Node1/ --syncmode 'fast'
3 --port 30311 --http --http.addr 0.0.0.0 --http.corsdomain '*' --http.port 8545
4 --http.api 'eth,net,web3,txpool,miner' --ws --ws.port 3334
5 --ws.api eth,net,web3 --ws.origins '*' --bootnodes 'enodekey'
6 --networkid 12345 --unlock 'public address of the first node'
7 --allow-insecure-unlock --password Node1/password.txt
8 --miner.etherbase 'public address of the first node'
9 --preload './script.js' --miner.gasprice 0 console
```

- **—datadir Node1/**: This sets the directory where the node’s data will be stored.
- **—syncmode 'full'**: This sets the synchronization mode to full, which means the node will download and process all blocks from the blockchain.
- **—port 30311**: This sets the TCP listening port for the P2P protocol.
- **—http**: This enables the HTTP-RPC server.
- **—http.addr 0.0.0.0**: This sets the HTTP-RPC server’s listening interface to all available network interfaces.
- **—http.corsdomain '*'**: This sets the CORS header to allow all domains to access the HTTP-RPC server.

- `-http.port 8545`: This sets the HTTP-RPC server's listening port.
- `-authrpc.port 8601`: This sets the authentication port for the HTTP-RPC server.
- `-http.api 'eth,net,web3,txpool,miner'`: This sets the API modules enabled on the HTTP-RPC server.
- `-ws`: This enables the WebSocket server.
- `-ws.port 3334`: This sets the WebSocket server's listening port.
- `-ws.api eth,net,web3`: This sets the API modules enabled on the WebSocket server.
- `-ws.origins '*'`: This sets the allowed WebSocket origins to all domains.
- `-bootnodes 'enode://...'`: This sets the bootnode for the node to connect to when starting up.
- `-networkid 12345`: This sets the network identifier for the node.
- `-miner.gasprice '0'`: This sets the gas price for transactions created by the node's miner.
- `-unlock 'public address of first node'`: This unlocks the account with the given address for mining.
- `-allow-insecure-unlock`: This flag allows the node to unlock accounts using HTTP or insecure connections.
- `-password Node1/password.txt`: This sets the file containing the password for the unlocked account.
- `-miner.etherbase 'public key'`: This sets the account address where mining rewards will be sent.
- `console`: This starts the node with an interactive console.



```

/home/Private_Ethereum_Network # bootnode -nodekey boot.key -verbosity 9 -addr :30310
enode://b0ca363f0b0c55503751d4885167e6171136ea4d84568195ea97f6372e44beeeee3e89ea5c093fa980489eebbe237e74b352be7303030e507bdcabb300638120127.0.0.1:0?discport=30310
Note: you're using cmd/bootnode, a developer tool.
We recommend using a regular node as bootstrap node for production deployments.
INFO [03-16|19:54:39.355] New local node record      seq=1,678,996,479,352 id=f1e027154614984d ip=<nil> udp=0 tcp=0
TRACE[03-16|19:56:20.152] << FINDNODE/v4           id=59d6eb1adb25642 addr=127.0.0.1:30311 err="unknown node"
TRACE[03-16|19:56:25.925] << FINDNODE/v4           id=c6643015dac13e85 addr=127.0.0.1:30313 err="unknown node"
TRACE[03-16|19:57:00.928] << FINDNODE/v4           id=c6643015dac13e85 addr=127.0.0.1:30313 err="unknown node"
TRACE[03-16|19:57:01.716] << FINDNODE/v4           id=dee4fed1fcfe2971 addr=127.0.0.1:30312 err="unknown node"

```

first node connected to the blockchain

Third node connected

Second node connected

Figure 3.5: Screenshot showing 3 nodes connected to the blockchain through bootnode

```

> net
{
  listening: true,
  peerCount: 2,
  version: "12345",
  getListening: function(callback),
  getPeerCount: function(callback),
  getVersion: function(callback)
}
> net.peerCount
2
>

```

Figure 3.6: Screenshot showing Nodes Peers

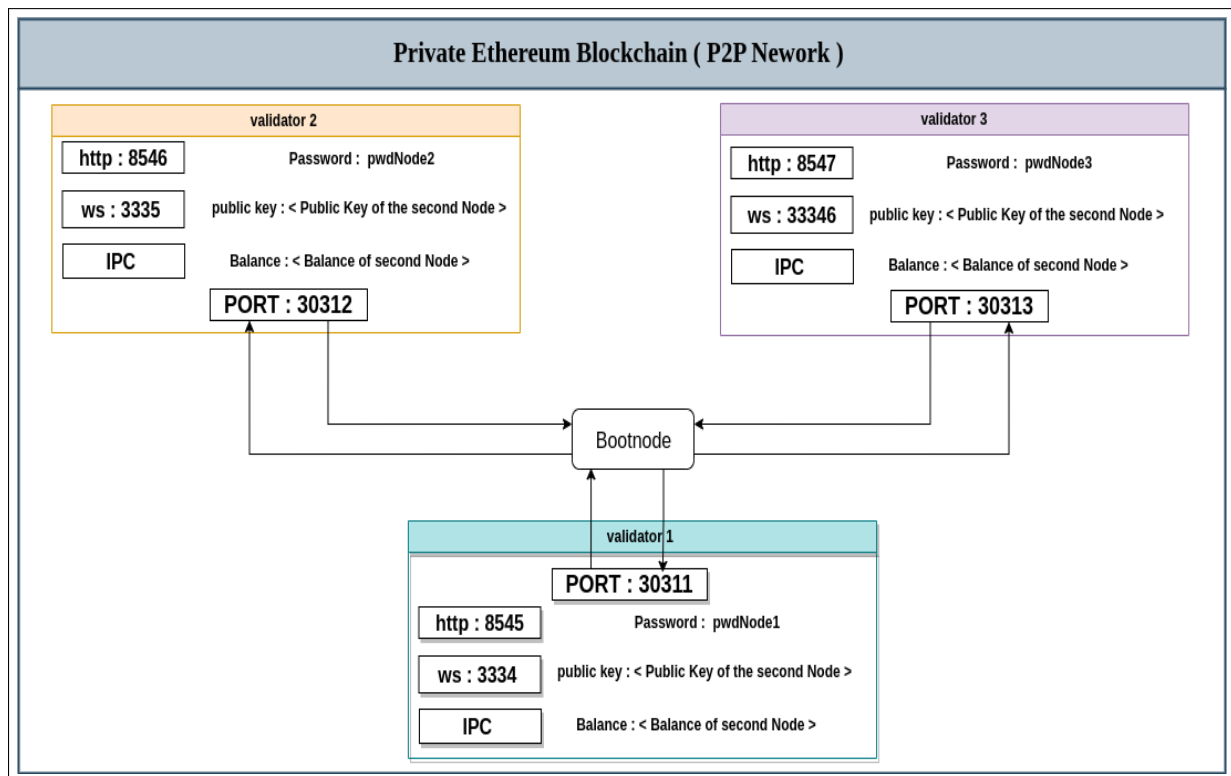


Figure 3.7: Screenshot showing Nodes Peers

3.5 Choosing the right provider

In the following section, we will discuss various providers for blockchain development and analyze the advantages and disadvantages of each one.

3.5.1 What is a provider ?

In the context of blockchain development, a provider refers to a software module or library that facilitates the interaction between a decentralized application (DApp) and a blockchain network. Providers typically provide a set of application programming interfaces (APIs) that abstract the

underlying complexity of the blockchain and allow developers to interact with the network in a simpler and more user-friendly manner.

	Infura	Moralis	Web3.js
Cost	- Free to use up to certain limits, paid plans available	- Free to use up to certain limits, paid plans available	- Free and open source
Availability	- Ethereum, IPFS	- Ethereum, Binance Smart Chain, Polygon, xDai	- Ethereum
Transparency	- Publicly auditable nodes	- Open source, publicly auditable	- Publicly auditable nodes
Scalability	- Can handle high volumes of requests	- Highly scalable infrastructure	- Limited by Ethereum's scalability
Speed	- Fast response times	- Fast response times	- Can be slow due to reliance on Ethereum network
Security	- High reliability and security	- High reliability and security	- High reliability and security

Tableau 3.4: comparative table highlighting differences between different types of consensus mechanism

While there are many blockchain providers available, each with its own set of strengths and weaknesses, web3.js stands out as a suitable library for interacting with Ethereum blockchain networks

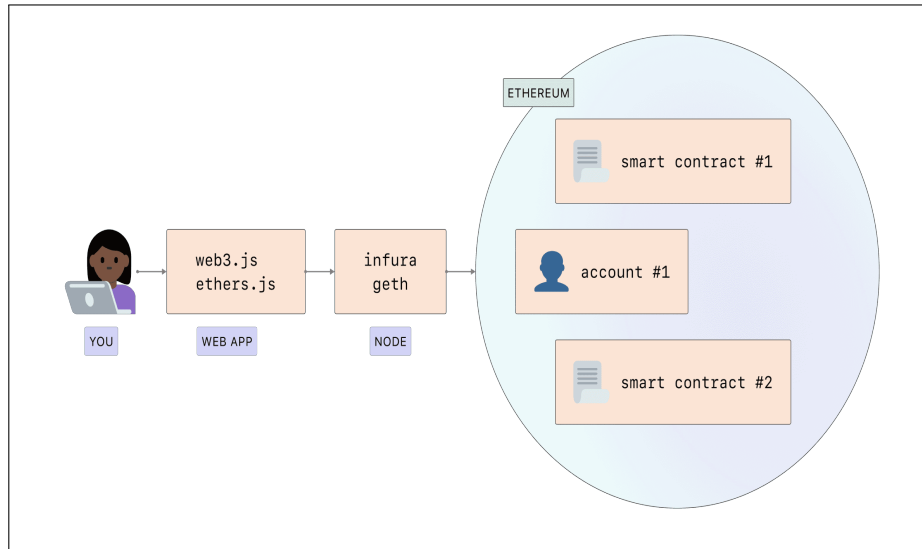


Figure 3.8: Web3js

3.6 Monitoring private blockchain

This section is dedicated to setting up Ethstats in go-ethereum and configuring it to monitor our private blockchain

3.6.1 What is Ethstats ?

Ethstats is a service that displays real time and historical statistics about individual nodes connected to a network and about the network itself. Individual node statistics include the last received block, block time, propagation time, connected peers, latency etc. Network metrics include the number of nodes, average block times, node geolocation, transaction counts etc.

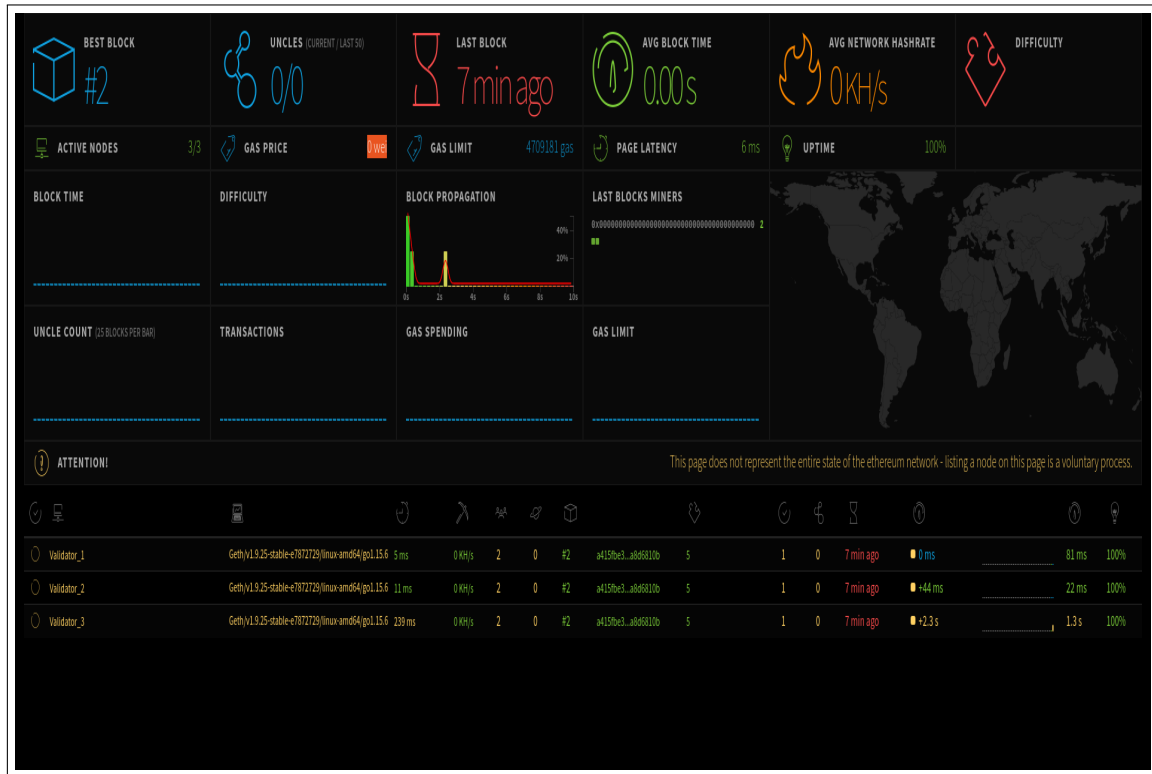


Figure 3.9: Screenshot of Ethstat Dashboard

Conclusion et Perspective

Le présent rapport est une présentation du travail réalisé durant notre stage de fin d'étude au sein de l'entreprise Sopra HR Software Tunisie afin d'obtenir le diplôme national d'ingénieur en informatique nous avons pu améliorer et mettre en pratique les notions théoriques acquises au cours de notre formation à TEK-UP et nous sommes introduit au monde professionnel.

Nous avons été en confrontation réelle avec l'intégration au sein des équipes du travail. La gestion de recrutement semble être une tâche délicate qui représente le premier contact du candidat avec le recruteur. Nous avons élaboré une application qui combine à la fois l'intérêt de processus de recrutement et un outil de sélection automatique des CV. L'objectif de de notre travail est de la correspondance entre une offre d'emploi et plusieurs CV proposés.

Cette correspondance consiste à chercher le profil le plus proche parmi l'ensemble des CV à l'offre d'emploi. Ceci s'inscrit dans le but de faciliter l'interaction entre le recruteur et le candidat sans l'intervention d'autres acteurs construisant ainsi une entente potentielle entre eux.

Sur le plan technique, ce stage nous a été une bonne opportunité pour développer nos compétences en développement des projets web.

Loin d'être complètement achevé ce projet est toujours en cours d'avancement. Ce projet est apte à l'ajout d'autre enrichissement tels que les entretiens en ligne via une connexion vidéo pair à pair entre le recruteur et le candidat.

Bibliography

- [1] Sopra-Steria. [Accès le 5-Mars-2019]. (), [Online]. Available: <http://www.soprasteria.com/carrieres/consultation-offre?ref=SE/AIX/IED/1001448>.
- [2] J.-M. D. JavaEE. [Consulté le 22-Avril-2019]. (), [Online]. Available: <https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm..>

Résumé

Résumé en français.

Mots clés : SpringBoot, React js, MySQL, UML

Abstract

En anglais

Keywords : SpringBoot, React js, MySQL, UML