Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

UNIVERSITY OF DJILLALI LIABES

FACULTY OF EXACT SCIENCES

SIDI BEL-ABBES

Thesis

Written by

Merbah Mohamed Amine

Master's degree final year project report

Specialty : Computer Science

Major : Information Systems Security and Networks

Titled

Building a private blockchain network for students records

Exploring blockchain in the education sector

Thesis will be presented

In front of the jury composed of

President :	*Pr.ABDERRAHMAN YOUSFATE*
Jury :	*Pr.ADIL TOUMOUH*
Supervisor :	*Pr.KAMEL MOHAMMED FARAOUN*

2021-2022

Acknowledgments

I would like to express my sincere gratitude to several individuals for supporting me throughout my Graduate study. First, I am grateful to my parents for their consistent support and assistance. I also wish to express my sincere gratitude to my supervisor, Professor Kamel Mohammed Faraoun, for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in my research and writing of this report.Finally, last but by no means least; also to everyone in the Department of computer science, it was great sharing premises with all of you during the last Two years. Thanks for all your encouragement!

Abstract

Blockchain, the technology that underpins the widely used cryptocurrency bitcoin, is transforming the world and bringing huge changes to the technological and corporate environments. Blockchain is a tamper-proof, distributed, and decentralized peer-to-peer technology for tracking and verifying digital transactions. Its goal is to increase data transparency, security, and integrity. Almost every domain, including finance, education, energy, supply chain, health care, and many more, uses this most trustworthy, decentralized ledger. This paper delves into the various components that contribute to the technology's trustworthiness and dependability. It explores the various uses of blockchain technology in the field of education. It will also show what technologies are required to build the network on which the customized blockchain that the project is about operates in order to provide solutions that students, teachers, and administrators may benefit from, making this paper useful as a guide for anyone interested in learning more about blockchain and blockchain-based solutions for any application in the education field.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

General Introduction **12**

1 Preliminary study **15**

1.1 Basic concepts 15

1.1.1 The components and the functioning of the Blockchain 16

1.1.2 Blockchain network layers 23

1.2 Problem Analysis and Specific requirements 25

1.2.1 Applications and Challenges of the Blockchain in education . . 25

1.2.2 Identifying the functional and non-functional requirements . . 33

1.3 Conclusion 39

2 Technical conceptual design **41**

2.1 Hyperledger Fabric Blockchain 41

2.1.1 A Technical Viewpoint on Fabric 44

2.1.2 Transaction flow and chaincode lifecycle 52

2.2 Detailed conception of the system 57

2.2.1 The conception of the network 57

2.2.2 The conception of the chaincode 64

2.3 Conclusion 69

3 Production and realization of test **71**

3.1 Environment description 71

3.1.1 Software Environment 71

3.1.2 The implementation repository description 74
3.2 Testing realization 82
3.2.1 Executing the network 83
3.2.2 Interacting with the network 92
3.3 Conclusion 95

General Conclusion **96**

References **98**

List of Figures

1-1 Chain of blocks 17
1-2 a Block 18
1-3 Blockchain-P2P-Network 19
1-4 Digital Signature 20
1-5 Blockchain Platforms Comparison Table[5] 38

2-1 Hyperledger Greenhouse 42
2-2 Hyperledger Fabric 42
2-3 The Fabric Ledger 45
2-4 Example of each node role 46
2-5 A simple Fabric network 48
2-6 Overview of the RAFT consensus protocol and block creation[18] . . 50
2-7 Transaction Flow[46] 52
2-8 Transaction Flow 55
2-9 chaincode lifecycle[10] 56
2-10 network infrastructure design 58
2-11 network channels design 60
2-12 blockchain design 61
2-13 Transactions design 62
2-14 Application call overview 63
2-15 Student asset 64
2-16 Certificate asset 66
2-17 Project asset 67

2-18 Event asset 68

3-1 compose folder 75
3-2 organizations folder 76
3-3 scripts folder 77
3-4 config directory 80
3-5 bin directory 81
3-6 Help page 83
3-7 CAs containers 83
3-8 CA admin enroll 84
3-9 register peer, user and admin 84
3-10 generates peer's MSP and certeficats 84
3-11 generates admin's and user's MSPs and certificates 85
3-12 finish the set-up 85
3-13 containers running 86
3-14 Generating the genesis block 86
3-15 The creation of the channel 87
3-16 Joining the peers 87
3-17 set the anchor peer 87
3-18 config file 88
3-19 list of all the channels 88
3-20 stopping the containers 89
3-21 removing the containers 89
3-22 packaging the chaincode 90
3-23 installing the chaincode 90
3-24 querying the chaincode 90
3-25 approving the chaincode 91
3-26 the chaincode approved 91
3-27 the chaincode committed 91
3-28 querying the chaincode definitions 92

3-29 chaincode containers 92
3-30 initiating the state database 93
3-31 query all students 93
3-32 query all students using Fauxton 93
3-33 Create new student 93
3-34 New student created 94
3-35 Read student7 94
3-36 Transfer student7 94
3-37 Read Updated student7 94
3-38 Delete student3 94
3-39 student3 Deleted 95

THIS PAGE INTENTIONALLY LEFT BLANK

General Introduction

When Bitcoin first appeared in 2008, it changed the financial world perspective. It was the first alternative to paper money, which was based on debt. Bitcoin is driven by a technology that has surprised the technical world, which is the Blockchain. It is an immutable evidence of records linked together in a chain and transmitted to every peer participating in the network. This powerful concept gives rise to a new discipline of computing. It's being developed with applications in supply chain, medical records, property records, and student records in mind.

In partial Fulfillment of the Requirement for Master's Degree of Networks and Security of Information Systems in the university of Djillali Liabes Sidi Bel Abbes, I am delighted to present my Final year project report. The project will be devoted to building a fully custom, private, and permissioned blockchain network to be used as platform for students information systems that could be developed in the future in our university. The main objective of this paper is to explain the components behind this technology and the platforms that relays on and explore the applications of it in the educational field and the challenges that face it as well and not only by talking but also by doing where we will see the greatness and the difficulties of architecting and implementing solutions for certain cases on the top of this blockchain network.

The project report will be composed of three chapters.In the first chapter, we will introduce this new technology, define it and define its components, its characteristics and its way of functioning. we will also discover the different types of it, where the implementation of blockchain based solutions will differ. we will also get into the topic of its applications and challenges, especially when it comes to the education sector where our focus will be. And finally we will see in close the main core of this

technology which is the Blockchain Peer-To-Peer network and the platform that runs it, examining the existing platforms and proposing our methodology and our way of implementing a platform that will enable a smart education system in our university. For the second chapter, we will give a detailed description and conception for our proposed solution. The chapter three will be consecrated to testing our final product, with mentioning the work and the test environment.

Chapter 1

Preliminary study

The blockchain is the newest and most talked-about technology now days. it is the technology that underpins Bitcoin, a well known disrupting cryptocurrency. According to experts, the blockchain will spark a revolution similar to what the Internet did. But what exactly is it, how can it be used in education, and what is holding this technology back? This is the topic of this chapter.

1.1 Basic concepts

In 1991, the term "blockchain" was first coined. A group of researchers intended to develop a technique for timestamping digital documents, so they couldn't be modified or backdated. In addition, Satoshi Nakamoto altered and redesigned the technique and established the first cryptocurrency, Bitcoin, a blockchain-based initiative, in 2008.

Blockchain, sometimes referred to as Distributed Ledger, it is a decentralized data structure that can be defined by a list of records, also called blocks, that can contain data from any type and that are linked by means of cryptographic references[39], therefore, we can list the characteristics of it as follows:

- Immutability: Each block in the blockchain is connected to other blocks with cryptographic hash functions. Transactions are recorded in a chronological order that makes blockchain tamper-proof.

- Transparency: Changes in the network are publicly available. Transactions are validated by authorized nodes on the network, so that any change could be detected at any instance of time, making Blockchain a compatible platform to work with.

- Traceability: Timestamp feature in Blockchain helps in recording transactions at each point of time by tracking every movement with hash functions adding up more functionality to their succession, and making the process efficient and secure.

- Decentralization: A distributed database that allows multiple platforms to have authoritative access to the database. It helps in reducing mediators expenses and make sure that the data is stored inside a secure environment.

- Trust factor: Trust is the principal factor that plays a major role while performing a transaction. Blockchain allows unknown entities to have secure transactions without any externalities and interruptions, offers a trustworthy environment for transferring assets to build trust.

[44]

1.1.1 The components and the functioning of the Blockchain

After we saw what the Blockchain is, and what the characteristics that distinguish it from any traditional Database.Now we can look into the core components that form it. we have to know that some blockchain networks or platforms have different architectures and functionalities that depend on the goal and the use case of the blockchain platform and some of them are just a fork of other platforms, but in general we can find the same core units even if themselves have different implementations depending of course on the blockchain platform used.

the Block and the Chain

Transactions are batched into blocks to guarantee that all network participants maintain a synchronized state and agree on the precise history of transactions [13]. A block is a data structure formed by special actors on the network, commonly termed miners in certain platforms, that consists of the most current and previously never occurring or contained entries. Blocks are batches of transactions with a hash of the previous block in the chain [32]. This links blocks together (in a chain) because hashes are cryptographically derived from the block data. This prevents fraud, because one change in any block in history would invalidate all the following blocks, as all subsequent hashes would change and everyone running the blockchain would notice [13].

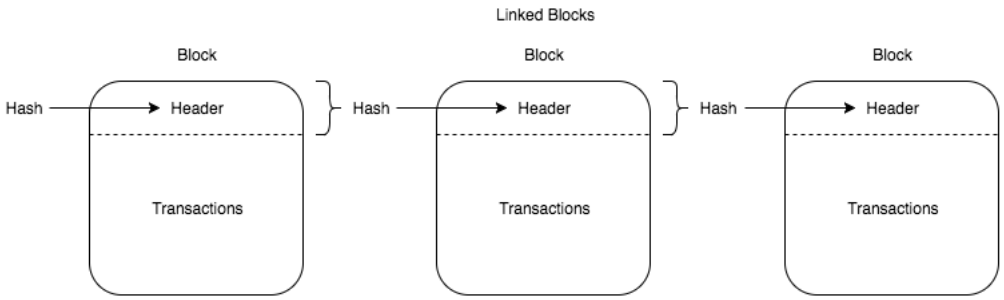


Figure 1-1: Chain of blocks

A block includes a block header where the metadata of the block is available and a block body where all valid transactions will be included. The metadata of the block varies based on the blockchain implementation[3], typical block metadata contains:

- version - the current version of the block structure
- previous block header hash - the reference to this block's parent block
- Merkle root hash - a cryptographic hash of all the transactions included in this block
- time - the time that this block was created

- nBits - the current difficulty that was used to create this block
- nonce ("number used once") - a random value that the creator of a block is allowed to manipulate however they so choose

[37]

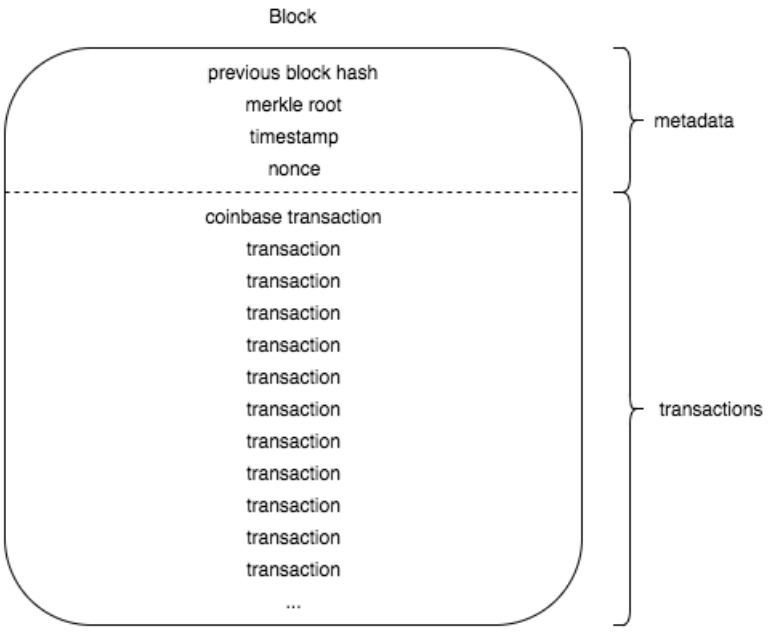


Figure 1-2: a Block

the Node and the network

Blockchain works on the top of a peer-to-peer network. Nodes in this network are every single entity or computer that processes and verifies transactions in each block[11]. We should understand that there are different types of nodes that consume data differently, and their names differ depending on the blockchain platform used, for example in Ethereum where we can find three types of nodes : Full nodes, Light nodes and Archive nodes. We can find other types or names in other platforms, like the Mining nodes, Lightning Nodes, Master nodes, Authority Nodes and Pruned Nodes[15]. The function of a node in blockchain can be further explained as when a new transaction

or a modification to an existing transaction comes, certain algorithms that are proposed for that transaction are evaluated in every node. If a majority of nodes of these algorithms come to a consensus that it is a valid transaction, then the transaction is accepted into the ledger, and a new block is added to the chain of transactions; otherwise, it is denied [47].

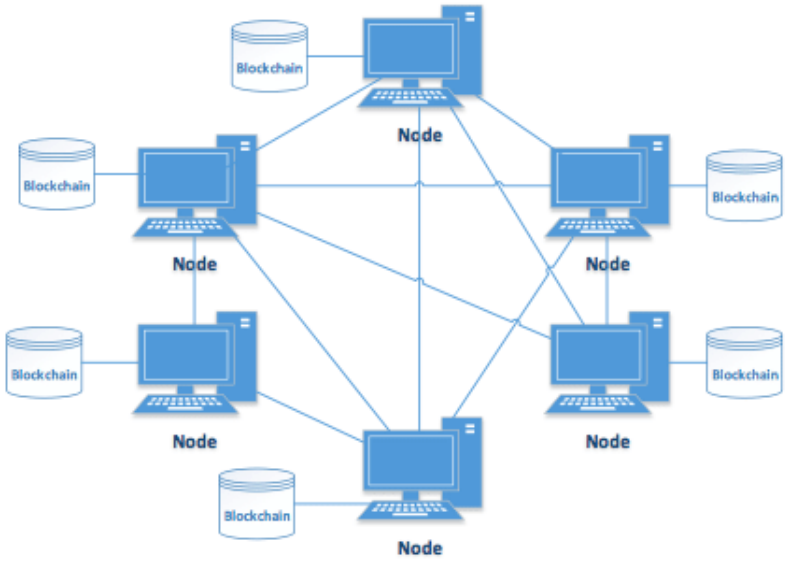


Figure 1-3: Blockchain-P2P-Network

Cryptography

The two types of cryptographic algorithms utilized in blockchains are asymmetric-key algorithms and hash functions. Thanks to hash functions, each participant has access to the features of a single image of the blockchain.Blockchains commonly employ the SHA-256 hashing algorithm as their hash function.The blocks are connected to one another, and hash algorithms play a significant role in preserving the integrity of the data kept inside each block. Any alteration in the block data can lead to inconsistency and break the blockchain, making it invalid. This requirement is achieved by the property of the hash functions, called the 'avalanche effect'.According to this, if we make even a slight change in the input to the hash function, we will end up getting a totally unrelated output as compared to the original output [43]. Asymmetric-key encryption is the second encryption technique that plays a significant part in

the applications of cryptography on the blockchain. Public-key cryptography, commonly referred to as asymmetric-key encryption, uses distinct keys for encryption and decryption operations. The tasks of the encryption key and decryption key can be performed by the public key and private key, respectively. The key-pair is created by asymmetric-key cryptography techniques, in which the public key is disclosed publicly, and the private key is kept private. Public-key cryptography can help two completely unknown parties for exchanging information securely. Digital signatures leverage public-key cryptography and help individuals to prove ownership of their private keys. Interestingly, users don't have to reveal their private keys to the other parties, as they can prove it by decrypting messages. So, the applications of cryptography in blockchain with digital signatures focus on the transaction process for ownership verification. Hashing, public-private key pairs, and the digital signatures together constitute the foundation for the blockchain. These cryptographic features make it possible for blocks to get securely linked by other blocks, and also ensure the reliability and immutability of the data stored on the blockchain [45].

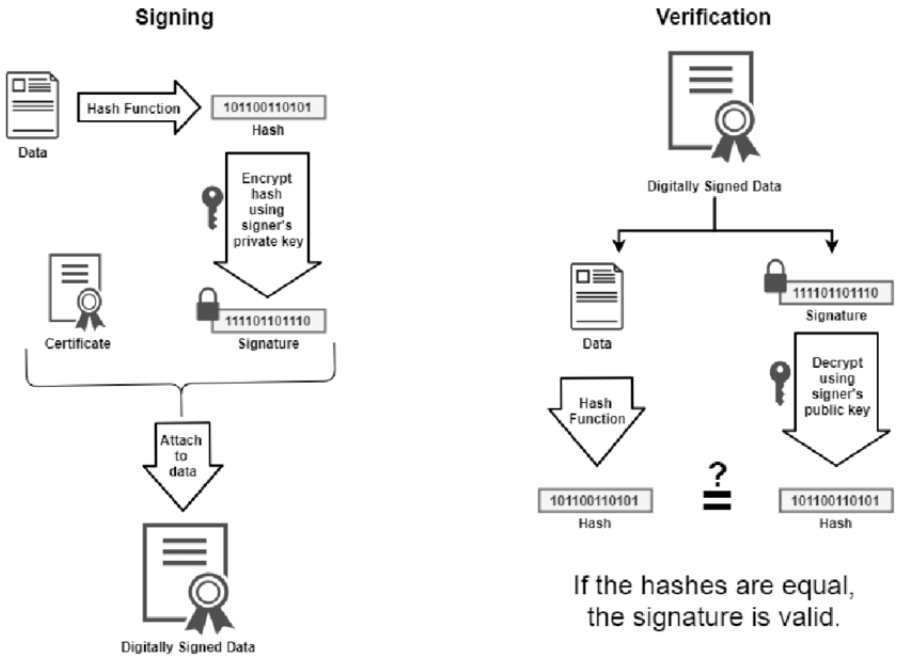


Figure 1-4: Digital Signature

the Consensus

When it comes to blockchains, which are, in essence, distributed databases, the network's nodes must reach an agreement on the network's current state. This agreement is achieved using consensus mechanisms.By consensus, we mean that a general agreement has been reached. Consider a group of people going to the cinema. If there is not a disagreement on a proposed choice of film, then a consensus is achieved. In the extreme case, the group will eventually split.In regard to blockchain, the process is formalized, and reaching consensus means that at least 51% of the nodes on the network agree on the next global state of the network.The consensus mechanisms (also known as consensus protocols or consensus algorithms) allow distributed systems (networks of computers) to work together and stay secure, and is realized in a codified set of rules that everyone is playing by [14]. These rules are entirely self-enforced. As the blockchain network grows larger, and more nodes and miners participate in it, the overall consensus grows stronger due to the sheer number of actors that are enforcing their own rules (that everyone else is enforcing, as well) [41].

Types of Blockchain

Based on their intended uses, the various blockchain types are grouped. Public and private blockchains are the two main forms of blockchain. Additionally, there are two other types, including the Consortium and Hybrid Blockchain. The types of Blockchain in a nutshell are:

- *Public Blockchain*

  are the simplest and publicly accessible blockchains. They are open source, non-restrictive, fully distributed, decentralized and permission less. Any entity with internet access can sign in to a Public Blockchain to become an authorized participant and become a user, miner or developer. The contents of the blockchain are readily available to every node with complete transparency. The major benefit of Public Blockchain is its uncontrollability or not granting full authority to any particular node. All the nodes adhere to the consensus mech-

anisms to ensure the security of the public blockchain. The most common use of Public Blockchains are for mining activities and cryptocurrency interchange. Thus, the most common public blockchains are Bitcoin, Ethereum and Litecoin blockchains [42].

- *Private blockchains*

  Referred to as managed blockchains, They are permissioned blockchains controlled by a single organization. In a private blockchain, the central authority determines who can be a node. The central authority also does not necessarily grant each node with equal rights to perform functions. Private blockchains are only partially decentralized because public access to these blockchains is restricted. Some examples of private blockchains are the business-to-business virtual currency exchange network Ripple and Hyperledger, an umbrella project of open-source blockchain applications.Private blockchains are not truly decentralized [48]. This is one of the biggest disadvantages of private blockchain and goes against the core philosophy of distributed ledger technology or blockchain in general. But they are fast. This is because there are few participants compared to the public blockchain. In short, it takes less time for the network to reach consensus, resulting in faster transactions [33].

- *Consortium Blockchains*

  Consortia blockchains, also known as federated blockchains, are permissioned blockchains that are managed by a number of organizations, in contrast to private blockchains, which are managed by a single organization. Since consortium blockchains are more decentralized than private blockchains as a result, their security is increased. Because it is a collaborative network, it is more effective and productive both when used collectively and when used individually. Consortium blockchains are often used by banks, governments, and other institutions.

- *Hybrid blockchains*

  The hybrid blockchain is best defined as the blockchain that attempts to use the best part of both private and public blockchain solutions. In an ideal world, a

hybrid blockchain will mean controlled access and freedom at the same time.The hybrid blockchain architecture is distinguishable from the fact that they are not open to everyone but still offers blockchain features such as integrity, transparency, and security [16].Hybrid blockchain architecture, as usual, is completely customizable. Who can use the hybrid blockchain and which transactions are made public are both decisions made by the members of the hybrid blockchain. This combines the best aspects of both worlds and makes it feasible for a business to collaborate with its stakeholders in the best way possible.

1.1.2 Blockchain network layers

[40]
Blockchain technology takes years to develop, and it is quite sophisticated process. It is useful to divide the various components that make up a blockchain into technical layers in order to make it clearer and easier to comprehend. The layers making up blockchain technology include:

Hardware infrastructure layer

The blockchain's content is housed on a server that is a component of the P2P computer network that computes, verifies, and saves transactions in an organized manner in a shared ledger. As an outcome, a distributed database is created, which logs all data, transactions, and other pertinent information. Each computer in a P2P network is known as a node. Nodes are accountable for validating transactions, organizing them into blocks, broadcasting them to the blockchain network, and it keeps on. While reaching consensus, the nodes commit the block to the blockchain network and update their local ledger copy [50]. Notably, the center of this layer is its nodes. A device is referred to and utilized as a node when it connects to a blockchain network.

Data layer

A blockchain's data structure is expressed as a linked list of blocks in which transactions are ordered. The data structure of the blockchain consists of two fundamental elements: pointers and a linked list. A linked list is a list of chained blocks with data and pointers to the previous block [9]. Transactions are digitally signed as part of the data layer's concern for the security and integrity of the data present in the blockchain. Transactions are signed with a private key, and the signer can be verified by anybody who has the public key. Information manipulation is recognized by the digital signature. Digital signatures guarantee unity since the data that is encrypted signed as well. Therefore, any tampering will invalidate the signature. A digital signature also conceals the sender's or owner's identity. A signature is therefore irrevocably connected to its owner and cannot be ignored.

Network layer

The network layer, commonly referred to as the P2P layer, is responsible for internode communication. Discovery, transactions and block propagation are all handled by the network layer. Propagation layer is another name for this layer. This P2P layer ensures that nodes can find one other and interact, disseminate and synchronize to keep the blockchain network in a legitimate state. A P2P network is a computer network in which nodes are distributed and share the workload of the network to achieve a common purpose. The blockchain's transactions are carried out by nodes [9].

Consensus layer

Blockchain platforms cannot survive without the present of the consensus layer. Whether a blockchain is built on Ethereum, Hyperledger, or another platform, the consensus layer is the most important and crucial one. The blocks are ordered, validated, and guaranteed to be in the correct sequence by the consensus layer. The distributed P2P network's consensus layer establishes a clear set of agreements be-

tween nodes. It guarantees that power remains decentralized and diffused. As a result, no one organization can rule the entire blockchain network. It helps a P2P network become reliable by ensuring that a single chain is followed and that it contains the truth.

Application layer

This layer takes care of deploying applications on top of the blockchain. For instance, decentralized applications (DApps), smart contracts, exchanges, and sites that provide information about a blockchain are applications built on top of blockchains. For the application layer, the blockchain needs to expose APIs. Different blockchains are similar, as they all provide a way for a client to communicate with the network [12].

1.2 Problem Analysis and Specific requirements

1.2.1 Applications and Challenges of the Blockchain in education

Due to its salient features, blockchain is applied not only in decentralized cryptocurrencies, but much beyond that. Blockchain can change the business transactions models and protocols of managing assets, E-voting, renting a car, watching a movie and many more. It widens its applications in major sectors like FinTech, Healthcare, Governance, Supply chain, Manufacturing Industries, Insurance, Education, IoT, Big Data systems and Machine Learning etc. [4].

The Use of Blockchain in education

Higher education has had a lengthy era of stability and consistent revenues as an enterprise. With the introduction of digital technologies such as social, mobile, analytics, the cloud, artificial intelligence, and the Internet of Things, that utopia came crumbling down. Disruptive forces are wreaking havoc on higher education, the most prominent of which being online learning. Students have become more open to and

trusting of new means of accessing learning as a result of economic necessity. Consumer trust in the peer-to-peer model has grown as a result of increased engagement in the sharing economy. Peer-to-peer collaboration in education is rapidly evolving, resulting in new sharing models and financial potential. With the rise of massive open online courses (MOOCs), there's a good chance that more students will choose free education options. Simultaneously, the education industry must deal with a significant regulatory load. When compared to analogous transactions in the digital era, seemingly simple tasks like sending a school transcript now take an inordinate amount of time and money. Consumers are accustomed to having their needs handled swiftly and with individualized attention, therefore this creates an expectation gap. Higher education institutions are striving to stay up with all of these changes. Furthermore, the majority of educational institutions follow a model in which each institution maintains control over its own student data and credentials. Because each organization is in charge of its own data, it can be changed or removed at any time, and there is no fail-safe or recourse if the data becomes corrupted. Institutions can restrict data access or impose constraints on it at their discretion, or data can be misused and disseminated without permission. This methodology also puts data at danger of being tampered with or deleted as a result of global events like war or natural disasters like floods or earthquakes. The war in Syria, for example, obliterated credential records all over the country, exacerbating the already chaotic situation. As students seek more mobile learning styles and study abroad, the need to transmit data in a secure and seamless manner will only grow. Institutions that want to recruit students from all over the world must update their systems and processes to meet the needs of today's students. The distributed ledger technology or the blockchain can help to address all of these problems. Through public key infrastructure (PKI) encryption, anonymity, longevity, integrity, transparency, and immutability, its decentralized design provides better privacy and security. Blockchain can enable essential changes in higher education's business, operating, and technological models by drastically reducing data management costs by removing many manual processes, including credential verification. [34]

In 2016, a distributed blockchain-based ecosystem was developed for the storage of educational records. Educationalists can save student learning achievements, educational credentials, credit management, and other information using blockchain-based systems. Any information pertaining to a learning institution can be saved and later used for analysis and decision-making. Blockchain technology can be used to expedite the publication of research articles. This submission allows for timely assessment and verification of research articles, as well as the avoidance of research article forgeries. The University of Nicosia is the first institution to keep academic degrees in blockchain. Block certs, an open source platform for developing, publishing, and validating blockchain-based educational certifications, was later developed by MIT Media Lab and Learning Machine, a software company. In terms of education, smart contracts, asset transactions, digital signatures, and certificates can all be maintained in blockchain. Smart contracts are a set of programs that are automatically performed when certain circumstances are met. Asset transactions, on the other hand, are ownership proof documentation for any tangible or intangible assets. Certificates are certificates of accomplishments, and signatures prove that the certificate was issued by and to authorized individuals [1]. The following are some examples of how blockchain can be used in the educational sector:

- **Keeping track of student credentials :**

  The completed course records of most higher education institutions are kept in proprietary forms. These databases are created with little to no interoperability for staff-only access in dedicated online platforms and within an institution. Furthermore, most schools have their own specialized method for storing students' finished course information, which protects the database's private data structure. Contrarily, blockchain records are permanently stored, making it possible to safeguard and verify documents like diplomas and course certificates whether or not a user has access to an institution's record-keeping system. Even if the institution that produced the credentials closed or the entire education system failed, those certifications could still be verified against blockchain data. Additionally, once an institution issues a certificate, no additional work is required

to verify that certificate's authenticity to third parties, because the certificate may verify itself directly on the blockchain.

- **Identity verification:**

  Identity verification is a recurrent issue for educational institutions, involving a great deal of manual involvement and exposing several vulnerabilities to data tampering. Validation of a student's identity happens only once through a digital process. The blockchain network holds information about the student identity document rather than the document itself. Students and job applicants can use blockchain to identify themselves online while preserving control over the storage and management of their personal information. Students in larger institutions are required to identify themselves to various areas of the organization on a regular basis. In such circumstances, either each section of the institution collects student data for itself, or the organization uses single sign-on, which allows all parties within the organization to access a single shared copy of the student data. Hundreds, if not thousands, of persons could have access to a student's personal information under both of these scenarios. Keeping that data safe necessitates monitoring all of those people's access rights and ensuring that their devices are similarly secure. Only a limited few – notably the parties in charge of validating a student's identification – have access to the data thanks to blockchain. Aside than that, it's in the hands of the learner. This means that the organization just needs to safeguard the device or network where the initial verification takes place, rather than managing complex systems for access privileges. This eliminates the need for major investment in hardening the network against data breaches, data protection training for employees, and access rights management.

- **Protection of intellectual property :**

  Professors publish research and publications on a regular basis as part of their jobs. When a professor begins his or her research, there is a little way to tell if a similar academic study is underway under the traditional system. Also, there

is a great deal of research piracy. The usage of blockchain technology aids in the resolution of these issues. Educators can utilize a blockchain to document and mark the publication of free educational resources. For copyright purposes, this would necessitate a notary of the publication date, as well as the ability to track the extent of re-use of any specific work. Journal articles are in the same boat. Journal citation tracking is a time-consuming operation that requires the assistance of a third party. We can avoid using a third party and allow anyone to write articles and track citations without restricting access to the source articles using blockchain. An incentive system depending on the quantity of resource re-used might potentially be implemented for the authors.

- **ownership of learning credentials:**

  Blockchain technology may be used to provide a more robust and flexible system for maintaining student credentials as they switch between courses throughout their secondary education and professional careers. It is more reliable to store credentials for a lifetime of study on the blockchain, since they cannot be altered. Personal data can be kept private by using blockchain technology. Students receive control and ownership of all of their educational data, including accreditation and work portfolios, in a secure location that can be accessed by anybody who needs to verify it. Self-sovereignty is facilitated by private blockchains, which allow individuals to decide who has access to and uses their data and personal information. Within the educational setting, the phrase is quickly becoming synonymous with the ability of individual learners to own, control, and share aspects of their credentials without relying on a trusted middleman like the educational institution. As the business moves toward skills-based education, this will become increasingly crucial. Learners might use blockchain to save their own evidence of formal or informal learning, distribute it with a targeted audience, and verify it instantly. Students may therefore have a self-updating curriculum vitae that they may immediately share with prospective employers. Employers, on the other hand, may minimize their workload by not

having to validate CVs and instead searching to see if individuals possess the necessary abilities.

- **Transfer of credits:**

  For many years, schools have struggled with the transfer of credits, which puts students at a disadvantage when they learn, for instance, that they must repeat courses to match the requirements of a new university. Additionally, maintaining records of and presenting coursework from previous universities might be challenging for students who wish to transfer to another school of higher learning. This issue is exacerbated when a student wishes to transfer to a school in another nation, where language obstacles and different protocols are likely to be encountered. Additionally, record storage standards differ, making inter-institutional record interchange challenging. Credit transfers currently rely on institutions negotiating agreements to recognize one other's credits under specified situations, but students claim that these agreements are frequently ignored. With a blockchain approach, these agreements might be formalized as blockchain-based smart contracts, with the credits being transferred automatically whenever the contract's requirements are met.

[35]

A study of a use case[2]

Certificates are proof of statements issued from an authorized entity to another because of certain achievements made by the recipient are true. The key components of a certificate are the achievement—the fact for which the statement has been issued. For example, the student has completed a course successfully, the issuer—the authorized source who has verified and validated the achievements of the learner, the receiver—the learner who has achieved the fact mentioned by the issuer, the certificate—a document of proof that includes the achievement, the issuer, the receiver, and the evidence if there are any. Once the learner has received a certificate based on the blockchain, it is possible to share it by the learner on demand.

Traditionally, the certificate issuing process includes issuing, verifying and sharing of certificates. The issuing process includes the activities involved in including the certificate information like the issuer, learner, achievements, logos, signature, etc. this information will usually be stored in a central repository. The verification process involves a third party to check the authenticity of the certificate issued. This can be done in various ways like checking the in-built security features of the paper certificates or checking with the issuer itself asking about the details. Sharing means the receiver on receiving the certificates can share it with a third party via post or email or in-person in case of higher education or employment. The whole process needs a centralized system for storage or a trusted third party for verifying the certificates. It is time-consuming and needs lots of security mechanisms and regulations for issuing and verification process. However, digital certificates have certain advantages over paper certificates like fewer resources needed for issuing, verifying, maintaining and using. But there are no proper global standards available for digital signatures. Also, it needs third-party verification for the digital signatures, and digital certificates are easy to counterfeit without signatures.

About Blockchain-Based Digital Certificates, the Blockchain resolves the problems of paper and digital certificates and provides an infrastructure to access for verifying and sharing the certificates in a secured way. By using hashing techniques, it keeps the document containing issuer, receiver and achievement details and whatever data need to be incorporated in the certificate in the blockchain which is spread across the network. Therefore, forging the certificates is almost impossible in the blockchain. Since the certificates are available over the network, it is possible for anyone who has access to the blockchain, thereby eliminating the need for third-party verification. Only the hash of the document is stored in the blockchain and not the document itself. Therefore, the privacy of the user is maintained. Thus, the benefits of the receiver of this blockchain based digital certificates are data independence and ownership. Recipient has the control over the achievements and can share wherever he wants to share. At the same time, the data record is permanent and cannot be destroyed.

The Challenge of Blockchain adoption[36]

Even if Blockchain Technology has a lot of potential, there are still obstacles that prevent it from being used in a wider range of applications.Blockchain deployment necessitates a wide range of business process changes, with the technological component accounting for only a small portion of the overall effort. For blockchain to have a substantial influence on education, it will require grassroots transformation as well as cross-border collaboration from all stakeholders. Governmental agencies will be crucial in developing and enforcing blockchain legislation, as well as constructing the infrastructure required to spread blockchain use. This project has already started. The European Commission recently set aside €300 million to invest in the European blockchain area as part of the European Union's aim to capitalize on the many benefits of blockchain while avoiding fragmentation. To promote and facilitate the integration, such public investments in blockchain infrastructure are vital. The removal of legal obstacles to blockchain is also being pioneered by regions outside of North America and Europe, including Australia, Singapore, and other nations in the Asia/Pacific region. A number of issues, including the creation of data standards, the high cost of processing and storing data, where to store the data, and how to filter, analyze, and securely share data, remain unanswered.

Standards are a particularly challenging issue in the realm of education. Every university has its own system for storing and managing student information. New standards are being developed every day as blockchain usage grows, posing the risk of chaos. IMS Global, W3C, and IEEE are all potential catalysts for this type of engagement. An education-industry taxonomy, metadata, data privacy, accessibility, geo-specific data storage policies (as applicable in Europe), data-exchange frameworks for credits, qualifications, and skills, what data can be made transparent to others vs. what needs to be hidden, data-access governance and rules, and so on are among the standards that will be developed. Another noteworthy issue is the data storage change. What happens to the existing data while the new entries are available on blockchain? How do organizations migrate all of their legacy data to modern systems

or make it accessible over new networks (while ensuring that standards are followed as they emerge)? Another unanswered question is whether and to what extent institutions will realize and embrace the necessity to relinquish data control. Organizational change management and governance are also important considerations. Overcoming governance difficulties will necessitate a deliberate effort to ensure that digital credentialing system standards are open and take into account the demands of all parties involved, including learners, educational institutions, businesses, and governments.

1.2.2 Identifying the functional and non-functional requirements

[49]

the functional requirements

The behavior of a system is described by functional requirements. In other words, the functional requirements describe the system's properties and concentrate on the output from the provided input. The following points give an overview of the functional requirements for the reference system architecture that were judged necessary.

- **node-to-ledger communications:** Nodes continuously interact and synchronize (read and write) data across a peer-to-peer (P2P) network to build the distributed ledger in any blockchain ecosystem. Using this typical blockchain use-case in conjunction with smart contracts, machine-to-machine (M2M) communication can be facilitated. The proposed architecture should support autonomous internet of things (IoT), node-to-ledger connections, and intelligent student record management ideas through M2M capabilities.

- **ledger-to-ledger communication:** This use-case occurs when many permissioned ledgers need to share data with one another. To preserve internal ledger consistency, the suggested design should permit read-only data access between ledgers.

- **ledger-to-a distributed storage communication:** Distributed ledgers allow participants in the blockchain to exchange data. However, transmitting data via the ledger may become challenging when the size of the sent data exceeds certain constraints. Ledger platforms can transport vast volumes of data at a reasonable cost using distributed storage managed by protocols like IPFS. The reference architecture should be able to support the distributed storage communication pattern to make it easier to disseminate huge data throughout the ledger.

- **ledger-to-external data sources (Oracles) communication:** On occasion, data that isn't present on the blockchain may be needed by smart-contract and blockchain applications. When accessing data from sources outside the ecosystem, such as oracles, the reference design should support a range of Oracle data interface techniques.

- **Regulatory compliance:**Standards, laws, rules, and policies must all be complied with. More than merely a corporate requirement, regulatory compliance is important. In many cases, it's a mandate of the law. In order to ensure regulatory compliance across the whole blockchain ecosystem, the reference architecture must contain technical processes and capabilities.

- **Tokenization:**One of the most important functions that our system must provide us. Even though the majority of us associate tokens with speculative coins, initial coin offers (ICOs), and other related activities, tokens are much more than that. The production of digital representations for goods, services, and rights is made possible through tokens. They may be fungible or non-fungible, depending on the good or service. Tokens can be used to symbolize ownership of goods, to grant you specific usage rights, or to obtain access to information about the thing they stand for.

Identifying the actors and their behaviors

the actors are the external entities that will interact with our system, whether directly by accessing direct control to the infrastructure or indirectly by client application with the help of APIs.

- *university, school or institution members:*
  these members could be the workers at the admission, teachers or the professors of the faculties, universities, schools or the institutions. they are the ones that able to write student records into the system and record his scores, courses, specialties, attendance, punishment and issue his certificates and approve his achievements, projects, papers, writings ... etc. and they will be able to read and track the entire history of any current student.

- *Students:*
  With this system, the student will own his data. it will enable him to read, query and receive their official records in a digital format that is naturally tamperproof without going back to his educational institution. Learners will be able to access the proof of their learning at any time (even if he is no longer a student) and any place (even if he changes the country) — proof that remains verifiable even if the educational institutions which issued the records cease to function.Also, he will be able to share his records with other external entities like employers and submit his projects, papers, writings to the system as well.

- *Government:*
  in order to handle and use student records in different sectors of the government (other ministries for example ministry of defense where students will not have to submit their files every year to prove that they are still students to not go to the obligatory military service) or other governments of other countries (for example doing academic equivalency for another university in other country), the higher authorities need to be able to read and query student data.

- *External entities:*

they could represent employers, companies, or enterprises that want to hire either current students or past students that apply to them to know the truth about their learning journey and verify their credentials with students permission of course. They also could represent other entities like e-learning platforms (Coursera, Udacity...) or private institutions(institution that provide highly reputed certification like CCNA or IELTS...) these entities needs to be able to write to the system with the permission of all other actors in the network.

the non-functional requirements

Non-functional requirements include a wide spectrum of a system's operational behavior and are concerned with how software performs its functions. Non-functional needs aren't always linked to specific business use cases. They are, nonetheless, essential system capabilities that ensure the system runs as intended. Although there is a wide range of non-functional needs, we've determined the following as the most important system capabilities that the reference architecture must include.

- **Scalability:**Scalability, when combined with elasticity, ensures that a heavily loaded system can grow up to handle the increased load and then shrink down to return to its original state. This ecosystem could include a wide range of devices and systems with variable processing capabilities, making scalability a critical design consideration. To achieve the performance requirements, the critical components of the reference design should be able to scale horizontally and vertically.

- **Data protection:** Data security is even more important for blockchain-based systems that replicate the ledger content across several nodes. The distributed design necessitates sophisticated data protection methods due to the large amount of data replication. As a result, the reference design should include technical safeguards to prevent unwanted access to data at rest, in transit, and during processing.

- **Logging, traceability, and auditability:**Security features such as logging,

traceability, and auditability are critical for ensuring openness and accountability. Legitimate parties can track the history of a user, system, and transactional events using auditability and traceability. Auditability and traceability are ensured by extensive logging mechanisms operating at multiple system levels. As a result, the blockchain-based reference should be able to provide traceability and auditability, as well as logging services that are strong, secure, and scalable.

- **Availability:**Availability assures that the system is available whenever and wherever it is needed, regardless of the conditions or time. Highly available systems, especially when Blockchain is used to represent real students and their data, are fault-tolerant and robust in order to sustain the operational state for a long period. The reference architecture should be able to meet high and continuous availability requirements.

- **Identity and access management:**Identity and access management systems are technologies that allow a wide range of system players to be identified and ensure that only privileged actors have access to network resources. As a result, the reference architecture should have capabilities for identifying, authenticating, and authorizing a wide range of distinct actors across all system automation tiers and across many organizational structures.

The blockchain platform

Numerous reports suggest that there is a lower success rate with blockchain implementation when it comes to the development of complicated systems. If a company wants to develop its own blockchain network and infrastructure, the entire implementation process becomes more challenging. In other words, a blockchain end-to-end solution cannot be created from the start. The blockchain platform can help with that. It lets us take advantage of existing infrastructure, solution, or service related to blockchain in order to build our custom blockchain network that is customized for a certain thing. We also need to factor in blockchain growth. If we adopt our own blockchain solution, it will simply be impossible for us to maintain our system.

Blockchain is developing quickly, so it is always advisable to let blockchain platforms handle the updating[7]. Blockchain platforms are widely available. To ensure that we choose the best one out there, we need to get to know some of them and choose the one that is aligned with our functional and non-functional needs, which are listed down in the figure 1-5.

	Ethereum	Hyperledger-Fabric	Hyperledger-Sawtooth	Hyperledger-Iroha	Corda	Ripple	Quorum	Hedera Hashgraph
Ledger Type	Per-missionless	Per-missioned	Per-missionless and Per-missioned	Fully Per-missioned	Per-missioned	Per-missioned	Fully Per-missioned	Per-missioned
Consensus Algorithm	Proof of Work	Solo, Kafka, Raft	PBFT, PoET, Raft	YAC Algorithm	Pluggable Consensus	Probabilistic Voting	Raft, Istanbul BFT	Asyn-chronous BFT
Smart Contract	✓	✓	✓	Yes, but mostly pre-defined	✓	X	✓	Yes (Using API)
Industry Focus	Cross-Industry	Cross-Industry	Wide ranging-Industry	Cross-Industry	Financial/ Cross-Industry	Financial Industry	Cross-Industry	Cross-Industry
Transaction Speed	~20 TPS	>2000 TPS	>1000 TPS	≤1000 TPS	~170 TPS	~1500 TPS	~100 TPS	~50k+ TPS
Smart contract Language	Solidity	Go, Java, JavaScript	Python, JavaScript, Go, Java, Rust, C++	C++	Kotlin, Java	-	Solidity	Solidity (using API)

Figure 1-5: Blockchain Platforms Comparison Table[5]

The first thing to consider in choosing the right platform is the ability to identify and permissioned each unit that is participating in the network because we are building a private network therefore we need a platform that allows us to create a permissioned network which means that the Ethereum platform is not considered for our solution, another thing we need to consider is the ability to communicate withe ledger and the ability to store different assets structures and this will be achieved only by smart contracts, so we need a platform that allow us to deploy smart contracts before and when the network run and can execute them inside the network which means that we will leave the ripple, Hyperledger Iroha and Hadeera out of our consideration. the last thing to consider is the performance which is referred in the

table as the transaction speed that can be achieved in a second and like it is shown in the table, Hyperledger fabric is better than the rest which means that we will work on the network using this platform.

1.3 Conclusion

In this chapter, I laid the foundations and explained basic concepts regarding blockchain; I explained the characteristics that this tech has to offer such as immutability, transparency, Traceability, Decentralization and Trust. I covered the pieces that make up a blockchain, including blocks, nodes, Cryptography, Consensus and P2P network. Lastly, I covered the blockchain P2P network and the different layers that make up the network: Hardware infrastructure, Data layer, Network layer, consensus layer and application layer. we also learned about the types of blockchain networks like the Public, Private, Hybrid and Consortium Blockchains. We also tackled the subject of deploying blockchain solutions in the education field and the challenges that are against its adoption, and we saw the functional and the non-functional needs that our system have to offer, and we chose the platform that will help us in creating our proposed system.

In the next chapter, will dive into our project, where we will design our solution in order to reach our goal, and we will talk about Hyperledger fabric the platform that will aid us in creating the network.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Technical conceptual design

The initial stage of this project is conceptual design, during which drawings or solid models enable us to give more specific feedback at an early stage of the building process and ensure that we deliver an excellent final product. So this chapter will go in details about the chosen blockchain platform, and it will be dedicated to present the clear plan to achieve our goal by designing the network and the business logic around it.

2.1 Hyperledger Fabric Blockchain

Hyperledger is an open source project created to support the development of blockchain based distributed ledgers. Hyperledger consists of a collaborative effort to create the needed frameworks, standards, tools and libraries to build blockchains and related applications. Since Hyperledger's creation by the Linux Foundation in 2015, the project has had contributions from organizations such as IBM and Intel, Samsung, Microsoft, Visa, American Express and blockchain startups such as Blockforce. In all, the collaboration includes banking, supply chain management, internet of things (IoT), manufacturing and production-based fields[17]. Hyperledger incubates and promotes a range of business blockchain technologies, including distributed ledger frameworks (like Hyperledger Fabric, Sawtooth, Indy, Besu), smart contract engines (Hyperledger Burrow), client libraries(Hyperledger Ursa, Aires), graphical interfaces

(Hyperledger Explorer), utility libraries (Hyperledger Caliper), and sample applications. it provides an alternative to the cryptocurrency-based blockchain model, and focuses on developing blockchain frameworks, tools and libraries to support global enterprise solutions. The focus of it is to provide a transparent and collaborative approach to blockchain development[8].Hyperledger Fabric is one of the projects that born out of it, which we will talk about in this section.

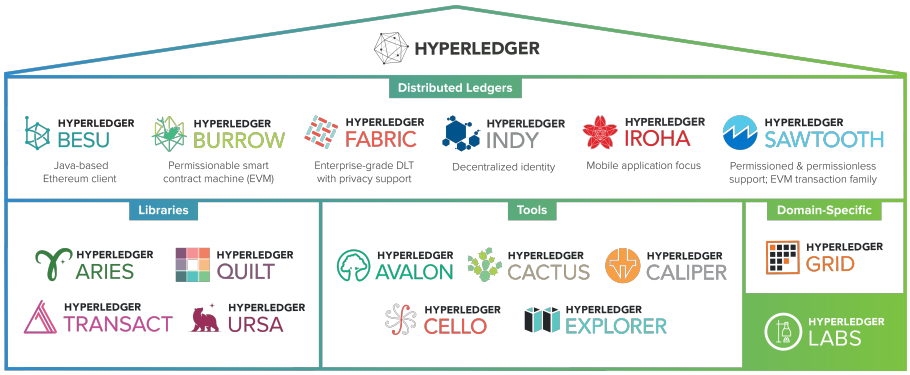


Figure 2-1: Hyperledger Greenhouse

Hyperledger Fabric is an enterprise-grade, distributed ledger platform that offers modularity and versatility for a broad set of industry use cases[19]. The modular architecture for Hyperledger Fabric accommodates the diversity of enterprise use cases through plug and play components, such as consensus, privacy and membership services. Rather than an open, permission-less system, Fabric offers a scalable and secure platform that supports private transactions and confidential contracts. This architecture allows for solutions developed with Fabric to be adapted for any industry, thus ushering in a new era of trust, transparency, and accountability for businesses and the actors in the education sector are not an exception. It also offers a unique approach to consensus that enables performance at scale while preserving privacy.

Figure 2-2: Hyperledger Fabric

Below are some key features of Hyperledger Fabric and what differentiates it from other distributed ledger technologies that led us to choose it as our backbone blockchain platform for our network:

- Permissioned architecture — meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted.

- Highly modular — it implements a modular architecture to provide functional choice to network designers. Specific algorithms for identity, ordering (consensus) and encryption, enabling innovation, versatility and optimization for a broad range of industry use cases

- Pluggable consensus — enable the platform to be more effectively customized to fit particular use cases and trust models. For instance, when deployed within a single enterprise, or operated by a trusted authority, fully byzantine fault tolerant consensus might be considered unnecessary and an excessive drag on performance and throughput. In situations such as that, a crash fault-tolerant (CFT) consensus protocol might be more than adequate whereas, in a multi-party, decentralized use case, a more traditional byzantine fault tolerant (BFT) consensus protocol might be required.

- Open smart contract model — flexibility to implement any desired solution model(account model, UTXO model, structured data, unstructured data, etc)

- Low latency of finality/confirmation

- Flexible approach to data privacy — data isolation using 'channels', or share private data on a need-to-know basis using private data 'collections'

- Multi-language smart contract support: Go, Java, Javascript , rather than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed.

- Support for EVM and Solidity — means that the other blockchain platforms smart contracts can run in this platform
- Designed for continuous operations, including rolling upgrades and asymmetric version support
- Governance and versioning of smart contracts
- Flexible endorsement model for achieving consensus across required organizations
- Queryable data (key-based queries and JSON queries)

[23]

Finally, a vibrant community is what drives every successful open source technology. The security, usability, robustness, performance, and feature set of Hyperledger Fabric have all been continuously enhanced by the community. These attributes are crucial to the success of our network. No other distributed ledger technology frameworks have, to yet, been adopted as widely by Cloud Service Providers like AWS, Azure, IBM, Google, and Oracle. If these well-known corporations believe in it, then why shouldn't we? We must choose Fabric to construct our blockchain network solution for student data because of the combination of these distinguishing traits that make Fabric one of the best platforms currently available.

2.1.1 A Technical Viewpoint on Fabric

In order to better understand some of the basic concepts that we will be using in designing and implementing the blockchain network, we need to introduce the key components of Fabric and their relations to each other and how they work as a whole in this section and the next one.

the ledger[24]

In the Hyperledger Fabric network, ledgers serve the dual functions of displaying the present value of a set of ledger states and preserving the transactional history that

produced those values. The ledger does not record business objects; rather, it records facts about an object's present state and the transactions that lead to that state. The ledger consists of two related components:

- *World state:* Database that holds a cache of the current values of a set of ledger states. This world state allows fast and easy access to the current value of a state, instead of having to calculate it through the entire transaction log. Hyperledger lets you choose among many, although one common used component for this is CouchDB, as it provides states structured as JSON documents.
- *The transaction log(the Blockchain):* it records all the changes that have resulted in the current world state. Transactions are collected inside blocks that are appended to the blockchain, enabling us to understand the history of changes that have resulted in the current world state. The blockchain data structure is very different to the world state because once written, it cannot be modified; it is immutable.

[38]

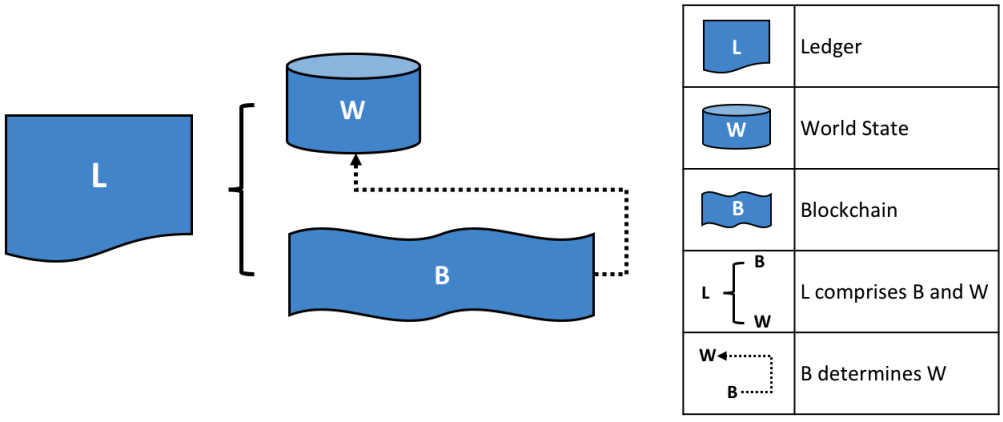


Figure 2-3: The Fabric Ledger

Peer nodes[27]

The nodes that host the ledgers and smart contracts are known as peers, and they constitute the core component of the blockchain network. Every application that

interacts with the network uses peers as its point of entry. Through the peers, apps can run chaincodes (smart contracts) that either query or change the ledger. Each peer on the network is given a digital certificate that serves as his identification in the various channels to which he connects. A peer could have varied permissions on each channel using this technique. Fabric defines various peer node types on its model, assigning each peer of the network a unique role:

- *Committing peer (or simply peer).* Each peer maintains the current snapshot of the current state of the ledger as a key-value store. Such peers cannot invoke chaincode functions.

- *Endorser peer.* Endorser peers have chaincode installed. When they receive a transaction proposal, they simulate the transaction execution on isolated containers. Based on that simulation, such peers prepare a transaction proposal sent to the orderer peer. The existence of endorser peers avoids sequential execution of transactions by all peers.

- *Orderer peer.* Orderers receive endorsed transactions and assemble them into blocks. After grouping transactions, orderers assure consensus by propagating such blocks to committing peers. They are validated and then committed to the shared ledger. Orderer peers record valid and invalid transactions, while other peers only contain valid transactions.

[6]

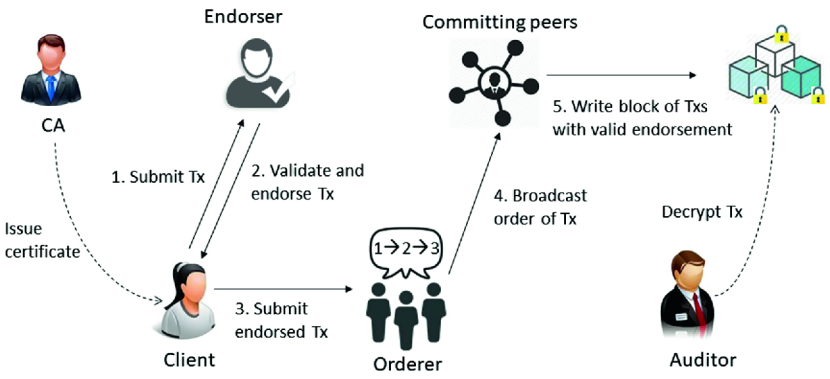


Figure 2-4: Example of each node role

Fabric also distinguishes between *anchor peers* and *leader peers*. Between peers from its organization and those from an outside one, anchor peers act as a bridge. Distribution of the transactions from the orderer to committing peers within the same organization is the job of leader peers. Keep in mind that a peer can be a leading peer, endorsing peer, committing peer, and anchor peer all at once! There will always be a leading peer, at least one endorsing peer, and at least one committing peer; the anchor peer is the only one who is optional.

Channels[20]

A communication path between the subset of participants who can see a subset of transactions can be established between participants via channels. For example, a subset of peers in the same network may only have access to a particular type of transaction.

Members (orgs), the shared ledger, chaincode apps, and the ordering service node define any channel in the network (s). Each party must be authenticated and approved in order to transact on a given channel, where every transaction on the network is carried out. A Membership Service Provider (MSP), which authenticates each peer to its channel peers and services, assigns a unique identity to each peer when they join a channel.

Each channel has its own unique ledger. This calls for a wholly distinct blockchain and wholly distinct world states. Applications and smart contracts can talk to each other through different channels in order to access ledger data.

The client SDK calls configuration system chaincode and specifies properties like anchor peers and members (organizations) when creating a new channel. The channel ledger, which maintains configuration data about the channel policies, members, and anchor peers, receives this request and constructs a genesis block for it. This genesis block or, if appropriate, a more recent reconfiguration block is shared with the new member when they are added to an existing channel.

A selected group of businesses on a channel can isolate their data from other organizations using Fabric's private data feature, which is available in addition to channels.

For instance, it can be advantageous for a seller to conceal the price of their goods from purchasers when numerous dealers participate in a blockchain ecosystem.
Private data is logically separated from channel ledger data, and organizations (network participants) with permissions can endorse, commit, or query this data. Private information may be disclosed and exhibited in the event of a dispute.
Private data is hashed and sent through the orderer rather than the actual data in order to increase privacy. To arrange the transactional data into blocks, the orderer just accesses its hash. Instead of using blocks, it is distributed peer-to-peer. Instead of channels, private data collections can be used when transaction data needs to be kept private from ordering service nodes.
For example, Figure 2-5 depicts a simple Fabric network compressed on two orgs with one peer each, connected by one channel. One orderer orders the transactions approved by both orgs.

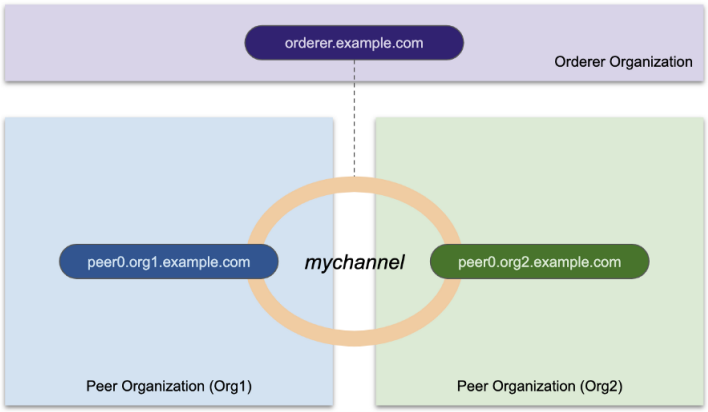


Figure 2-5: A simple Fabric network

Smart Contracts and Chaincode[30]

Smart contracts can be created by blockchain developers using a common programming language like Java, GoLang, or Javascript. A Hyperledger Fabric blockchain system's core consists of a smart contract and a ledger. A smart contract specifies the executable logic that creates new facts that are added to the ledger, as opposed

to a ledger which contains facts about the present and past states of a collection of objects. Although it may also be used for low level Fabric system programming, chaincodes are primarily utilized by administrators to group relevant smart contracts for deployment.

All of a chaincode's smart contracts are made accessible to the network's organizations after it has been deployed. So only administrators should be concerned with chaincode; everyone else should consider using smart contracts instead. Implementing several use cases is possible with chaincode. Although Fabric does not come with a built-in cryptocurrency, it is still feasible to use chaincode to create an underlying token that represents assets or the authority to carry out specified actions. Transactions between network users can be used to swap these assets. Chaincode is a crucial component of a Fabric network since it specifies the guidelines that member participants must adhere to. It is segregated from the shared ledger because it is run in Docker containers.

There are two types of chaincode: *application chaincode*, that executes the application logic and communicates with the peers using *gRPC messages*, and *system chaincode*, ran on the *configuration channel*. System chaincode assures chaincode execution correction (for example, that the endorsement policies are respected). Chaincode can be deployed dynamically, and it is usually running concurrently on the network. It runs directly on the peers' processes. The configuration channel stores the definition of MSPs (membership service providers), configuration about the consensus, ordering service parameters, and rules on how the channel configuration is tweakable. it executes transaction proposals against world state data, which we will see how in details in the next section[6].

A smart contract can call other smart contracts both within the same channel and across multiple channels, which is another issue that needs to be discussed. This allows them to access global state data that would otherwise be inaccessible due to smart contract namespaces by reading and writing it.

Consensus[26]

Recent developments in distributed ledger technology have made consensus synonymous with a certain algorithm, within a particular function. The fundamental role that consensus plays throughout the entire transaction flow, from proposal and endorsement to ordering, validation, and commitment, highlights the fact that consensus encompasses more than just agreeing on the order of transactions. This distinction is highlighted in Hyperledger Fabric. Consensus is essentially the full-circle confirmation of the accuracy of a group of transactions making up a block.

Fabric allows for different kinds of participants in the network, which facilitates the *execute-order-validate* paradigm for distributed execution of chaincode. Endorsement peers (*endorsers*) execute (*endorse*) the smart contracts (*chaincode*) and return blockchain clients the validation output of submitted transactions, which contain the endorsement peers' signatures. This process allows parallel execution and addresses non-deterministic code.

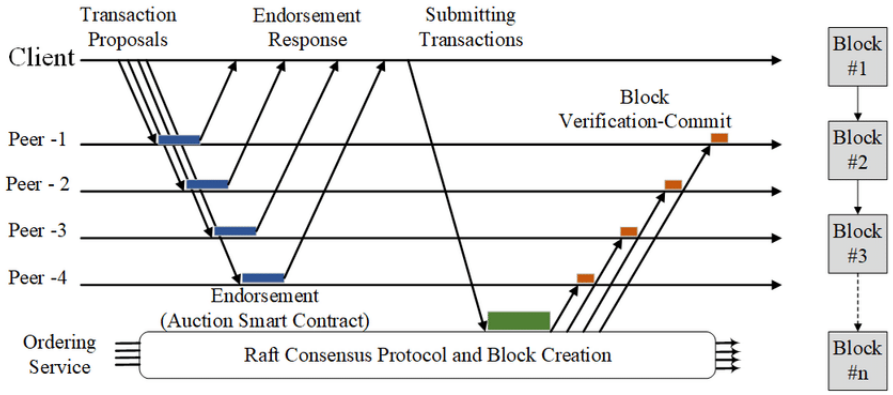


Figure 2-6: Overview of the RAFT consensus protocol and block creation[18]

To achieve consensus, Fabric uses a permissioned voting-based scheme, which achieves low-latency. The *endorsement policy* defines the voting-based scheme to be used by peers and, consequently, the weight of each peer regarding the validity of a transaction. In essence, the endorsement policies gather approvals for the execution of each transaction.

Apart from the consensus, Fabric needs to order blocks of transactions proposed by different peers. The consensus on the block order is pluggable, i.e., using Kafka's

Zookeeper (in previous 1.x.x Fabric versions) or RAFT (as in 2.x.x), both crash fault-tolerant consensus mechanisms. Plugability allows the developer to adapt the consensus to the requirements of the environment. The consensus algorithms are currently implemented to achieve consensus deterministically. This way, forks as in the Bitcoin network are prevented. Fabric can utilize different consensus protocols that do not require a native cryptocurrency, which reduces attack vectors and performance issues due to expensive operations, for instance, the ones required by proof of work.
In conclusion, the consensus in Fabric is not limited to the agreed-upon order of a batch of transactions between peers. Rather, it is also the result of the execute-order-validate process during a transaction's flow from its proposal to its commitment to each peer's ledger.

Identities, MSPs and Policies[29]

Peers, orderers, client applications, administrators, and more are some of the various actors in a blockchain network. These actors each have a digital identity contained in an X.509 digital certificate issued by a Certificate Authority (CA), which identifies them as active elements inside or outside of a network that can use services. These identities are crucial because they specify the precise access to information and resource permissions that actors in a blockchain network have. Since CAs are crucial, Fabric includes a built-in CA component that enables us to establish CAs on the blockchain network we will establish. This portion, referred to as Fabric CA, is a private root CA provider with the ability to manage participants in Fabric whose digital identities take the form of X.509 certificates[22].
Although certificates issued by any CA are valid in the network, they are not all used to verify that network entities are authenticated. Membership Service Providers (MSP), who are accustomed to establishing which identities are permitted to operate, add an additional layer of authentication. In more detail, an MSP is a part that establishes the guidelines for legal identities within an organization. The Membership Service Provider, despite its name, does not actually offer any services. Instead, the MSP requirement is implemented by adding a set of folders to the network's config-

uration where it exists in two realms. Locally(local MSP) (defined for clients and for nodes (peers and orderers)) and in channel configuration (channel MSP) (define administrative and participatory rights at the channel level)[25].

Last but not least, we must point out that every action we take on a Fabric network is governed by a policy; in this case, policies serve as the infrastructure management mechanism. They represent the process by which members decide whether to approve or disapprove changes to the network, a channel, or a smart contract. Access Control Lists, Chaincode Lifecycle Policies, Chaincode Endorsement Policies, Channel Policies, Channel Modification Policies, and so forth can all make use of them[28].

2.1.2 Transaction flow and chaincode lifecycle

Transaction flow[31]

The transaction flow, which follows the execute-order-validate paradigm, is depicted in Figure 2-8, is as it follows:

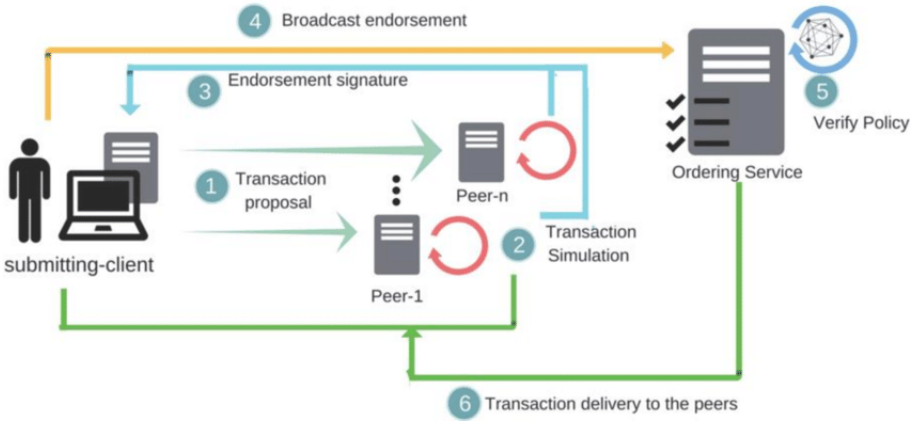


Figure 2-7: Transaction Flow[46]

- *Transaction proposal.* A blockchain client, which represents an organization, creates a transaction proposal and sends it to endorsement peers, as defined in the endorsement policy. The proposal contains information regarding the proposer's identity, the transaction payload, a nonce, and a transaction identifier.

- *Execute (endorsement)*: the endorsement consists in the simulation of the trans-

action. The endorsers produce a write-set, containing the keys and their modified values, and a read-set. The endorsement peers execute the transactions in an isolated environment. The endorsement is sent as the proposal response. It contains the write-set, read-set, the transaction ID, endorser's ID, and the endorser's signature. When the client collects enough endorsements (which need to have the same execution result), it creates the transaction and sends it to the ordering service. The endorsement phase eliminates any eventual nondeterminism.

- *Order*: after the endorsement, there is the ordering phase, performed by orderers. The ordering service checks if the blockchain client that submitted the transaction proposal has appropriate permissions (broadcast and receiving permissions) on a given channel. Ordering produces blocks containing endorsed transactions, in an ordered sequence, per channel. The ordering allows the network to achieve consensus. The orderer broadcasts the transaction's outputs to all the peers. For correct ordering, there are some properties that the system must comply with, defined as:

  Definition 1 *Hash chain integrity: For any two blocks, B delivered with sequence number s, and B' delivered with s' at correct peers such that s = s', it holds B = B'.*

  Definition 2 *Hash chain integrity: If some correct peer delivers a block B with number s and another correct peer delivers block B' = ([tx 1 , . . . , tx k], h') with number s+1, then it holds h' = H (B), where H (·) denotes the cryptographic hash function.*

  Definition 3 *No skipping: if a correct peer p delivers a block with number s > 0 then peer p has already delivered all blocks with number < s.*

  Definition 4 *No creation: When a correct peer delivers block B with number s, then for every tx ∈ B, some client has already broadcast tx.*

Definition 5 *Validity: If a correct client invokes broadcast(tx), then every correct peer eventually delivers a block B that includes tx, with some sequence number.*

The ordering step assures all the above properties for each channel. The ordering service broadcasts blocks to the peers that maintain the ledgers' state via the ordering service or gossip protocol.

There is the need to note that the ordering service aims to achieve consensus and prevents the network from forks. Consensus in Fabric encompasses the whole transaction flow, from the transaction proposal to the committing. It happens at the transaction level, where not all nodes need to engage in the consensus mechanism. Channels ensure that messages are delivered in the same logical order to committing peers. The ordering service achieves consensus in a deterministic way. As the ordering is deterministic and not probabilistic, as in Bitcoin, forks do not occur. Permissioned blockchains that rely on the BFT replication protocols to achieve consensus can only support f faulty nodes out of $3f+1$ nodes. This assumption may not match the trust model idealized for a specific application. The endorsement policy establishes the trust model, decoupled from the consensus mechanism, allowing developers to reason about the trust model independently of the consensus algorithm.

- *Validate*. Firstly, each peer validates the received transactions by checking if a transaction follows the corresponding endorsement policy. After that, a read-write conflict check is run against all transactions in the block sequentially. It compares the versions of the keys in the read-set with those currently on the ledger for each transaction. In case they do not match, the peers discard the transaction. Finally, the ledger is updated, in which the ledger appends the created block to its head. The ledger appends the results of the validity checks, including the invalid transactions.

[6]

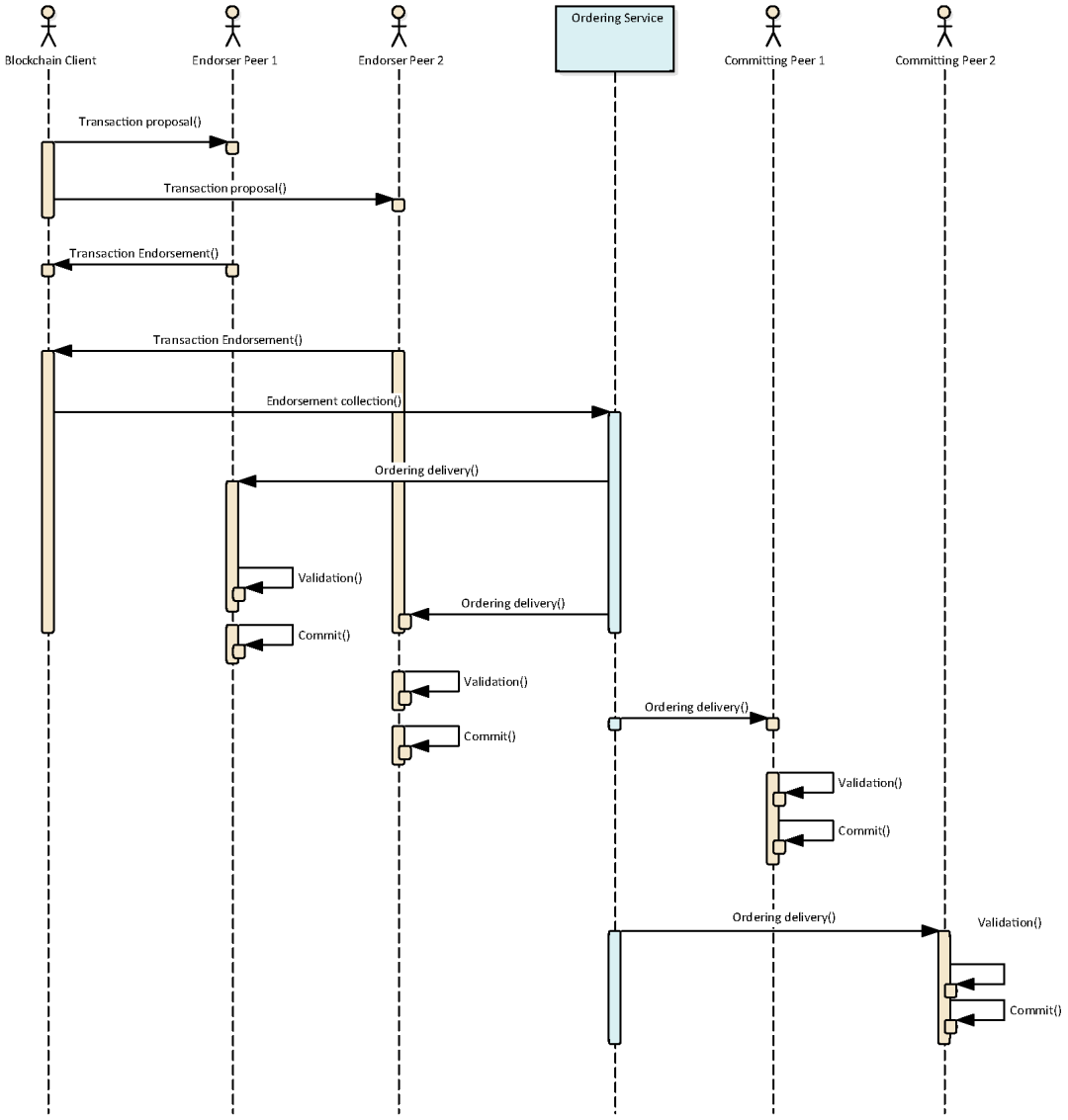


Figure 2-8: Transaction Flow

chaincode lifecycle[21]

The Fabric chaincode lifecycle is a process that allows multiple organizations to agree on how a chaincode will be operated before it can be used on a channel. we would use the Fabric lifecycle to install, define, deploy and upgrade a chaincode.
Fabric chaincode lifecycle requires that organizations agree to the parameters that define a chaincode, such as name, version, and the chaincode endorsement policy. Channel members come to agreement using the following four steps. Not every organization on a channel needs to complete each step.

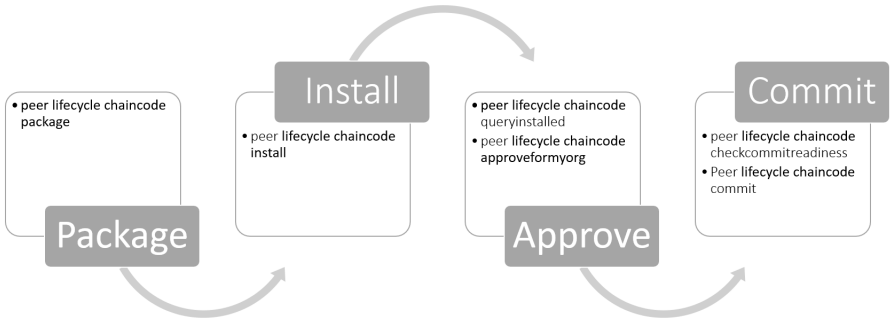


Figure 2-9: chaincode lifecycle[10]

- *Package the chaincode:*the chaincode is packaged into a deliverable on a build machine, in which the Chaincode needs to be packaged in a tar file.This step can be completed by one organization or by each organization.

- *Install the chaincode on the peers:*The deliverable is copied onto the peers (Docker containers) in which it needed to install the chaincode package on every peer that will execute and endorse transactions. The peer will build the chaincode after the chaincode is installed, and return a build error if there is a problem with the chaincode which requires reinstalling.Every organization that will use the chaincode to endorse a transaction or query the ledger needs to complete this step.

- *Approve a chaincode definition for the organizations:*The chaincode is governed by a chaincode definition. When channel members approve a chaincode defini-

tion, the approval acts as a vote by an organization on the chaincode parameters it accepts. These approved organization definitions allow channel members to agree on a chaincode before it can be used on a channel.Every organization that will use the chaincode needs to complete this step. The chaincode definition needs to be approved by a sufficient number of organizations to satisfy the channel's Lifecycle Endorsement policy (a majority, by default) before the chaincode can be started on the channel.

- *Commit the chaincode definition to the channel:*The commit transaction needs to be submitted by one organization once the required number of organizations on the channel have approved. The submitter first collects endorsements from enough peers of the organizations that have approved, and then submits the transaction to commit the chaincode definition.

2.2 Detailed conception of the system

A blockchain network is a technical infrastructure that provides ledger and smart contract (which are packaged as part of a "chaincode") services to applications. Primarily, smart contracts are used to generate transactions which are subsequently distributed to every peer node in the network where they are immutably recorded on their copy of the ledger. The users of applications might be end users using client applications or blockchain network administrators.In this section, we will go through designing our prototype network, and we will see our proposed chaincode solution that built in top of this network.

2.2.1 The conception of the network

like we mentioned before, in order to ease the conception of the network, the blockchain network has been divided to multiple layers (infrastructure, network, consensus, data and application layer).Therefore, we are going to design our prototype network and go through it layer by layer.

Infrastructure layer

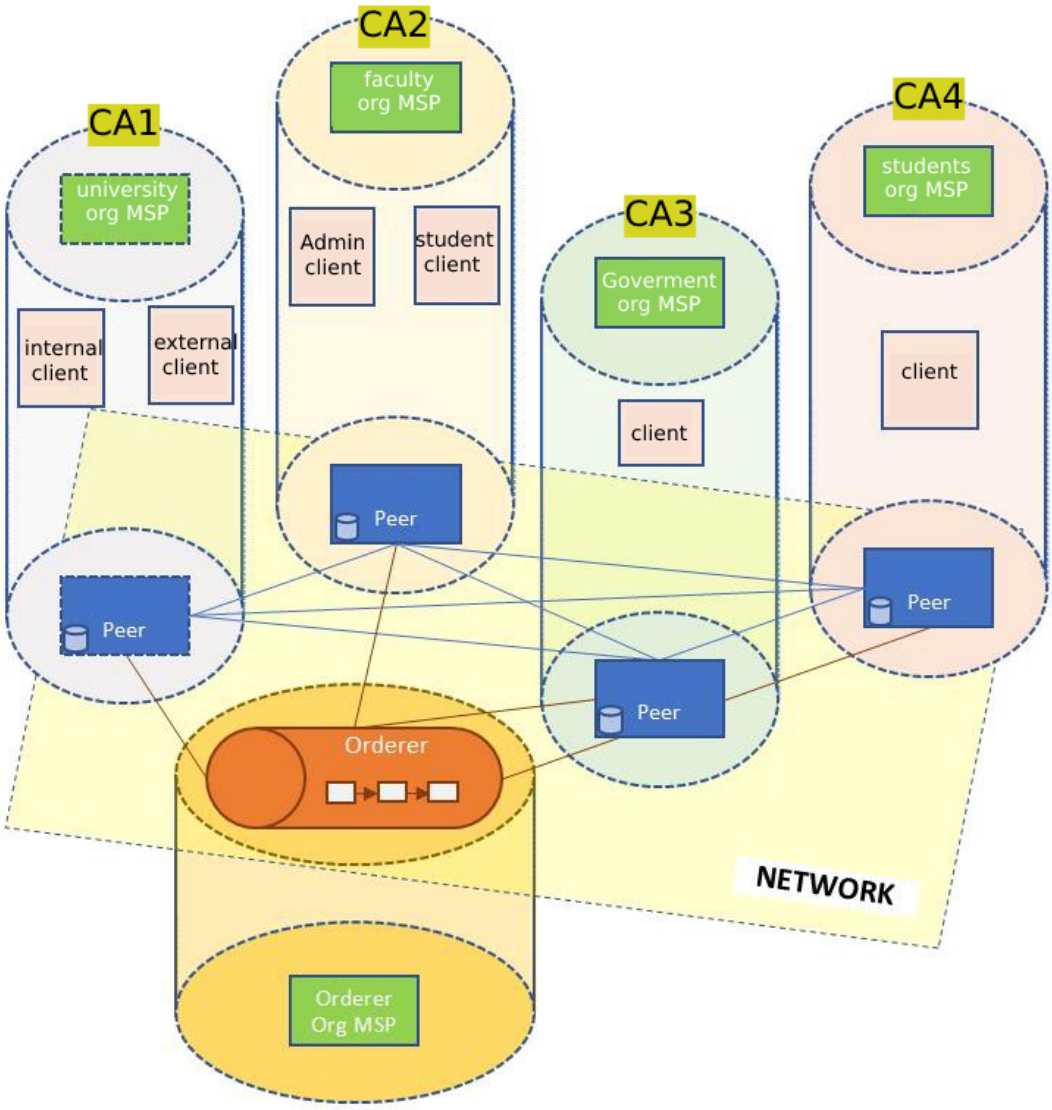


Figure 2-10: network infrastructure design

the figure 2-10 shows our final proposed prototype infrastructure P2P network. This network composed of four organizations and one ordered organization. The first organization represented by a university with one peer, The second one represented by a faculty that have one peer, the third is the government that also have one peer, the fourth one is not that necessary, and it does not represent each student but could represent a club or a committee, it allows the students to be a part of the system for the goal of democratizing this system (by holding a copy of the truth).

each organization could have its own CA or could have the same CA which will give an identity to its members (admins, peers, clients) and register their roles in their respective MSPs. They also could have multiple peers and client applications.

Network layer

like we mentioned, the network layer or the propagation layer deals with the peer-to-peer communications between the nodes that allow them to discover each other and get synced with another node in a network. Hyperledger fabric do this through channels between the joining organizations to it where each channel here considered a logical network and maintains its own ledger.
And because we have different records for students we need different ledgers therefore we need to create multiple channels which it will go like this :

- *Channel 01:*

  For storing general student information that don't change much or don't change at all like his full name, date of birth, place of birth, address, university, major, specialty, year of enrollment ... all the organizations mentioned above will be a part of this channel

- *Channel 02:*

  For recording students, certificates that they got with the full transcript and stores their year and semesters scores. Here also all the organizations will be a part of this channel

- *Channel 03:*

  For recording the projects, papers and the work that student did through the years, like final year projects, essays or presentations... all the organizations will be a part of this channel as well.

- *Channel 04:*

  For recording students clubs, events, competitions enrolled ... all the members will be a part of this channel also.

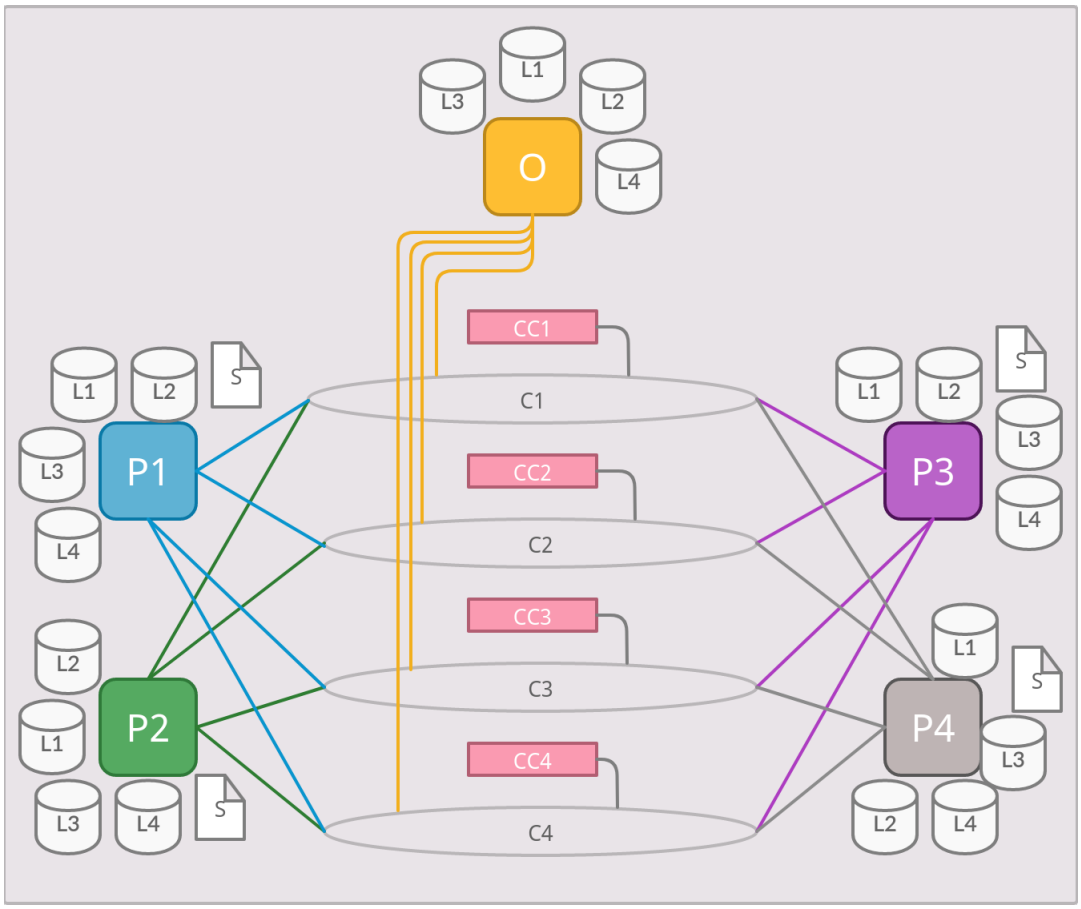


Figure 2-11: network channels design

Data layer

We did mention that the data layer is concerned with data structure of the blocks in the blockchain. In case of Hyperledger fabric like we also mention, it offers us a fertile ground for this where we have the ledger that is composed of world state database that hold the current state of the asset and the actual blockchain the transaction log that records the facts about how these assets arrived at their current states. The blockchain is always implemented as a file, in contrast to the world state, which uses a database. This is a sensible design choice, as the blockchain data structure is heavily biased towards a very small set of simple operations. Appending to the end of the blockchain is the primary operation, and query is currently a relatively infrequent operation.And because of this, we are not going to need to change the structure of every transaction log (blockchain) of all the channels created here, and we will leave

it to the default norm.

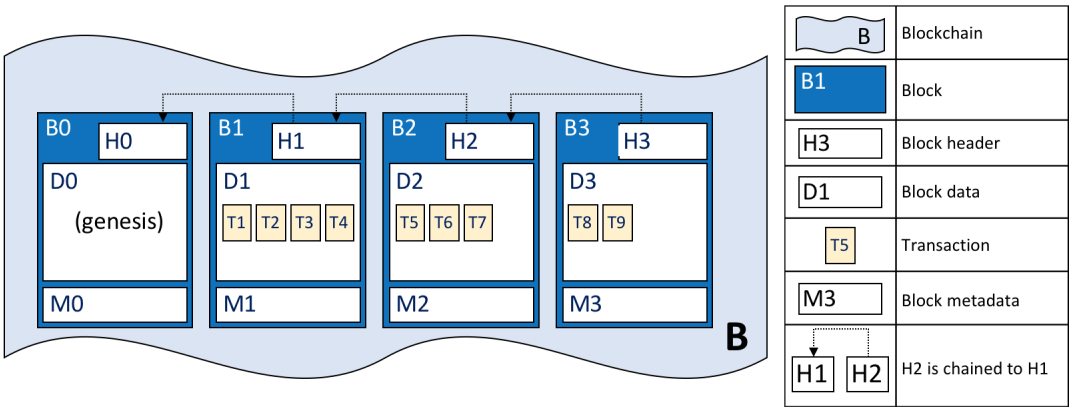


Figure 2-12: blockchain design

Let's have a closer look at the structure of a block. It consists of three sections:

- *Block Header:* This section comprises three fields, written when a block is created.These fields are internally derived by cryptographically hashing the block data. They ensure that each and every block is inextricably linked to its neighbor, leading to an immutable ledger.
  - Block number: An integer starting at 0 (the genesis block), and increased by 1 for every new block appended to the blockchain.
  - Current Block Hash: The hash of all the transactions contained in the current block
  - Previous Block Header Hash: The hash from the previous block header.
- *Block Data:* This section contains a list of transactions arranged in order. It is written when the block is created by the ordering service. These transactions have a rich but straightforward structure, which we will describe in a moment.
- *Block Metadata:* This section contains the certificate and signature of the block creator, which is used to verify the block by network nodes. Unlike the block data and header fields, this section is not an input to the block hash computation.

As we've seen, a transaction captures changes to the world state. Let's have a look at the detailed block data structure, which contains the transactions in a block.

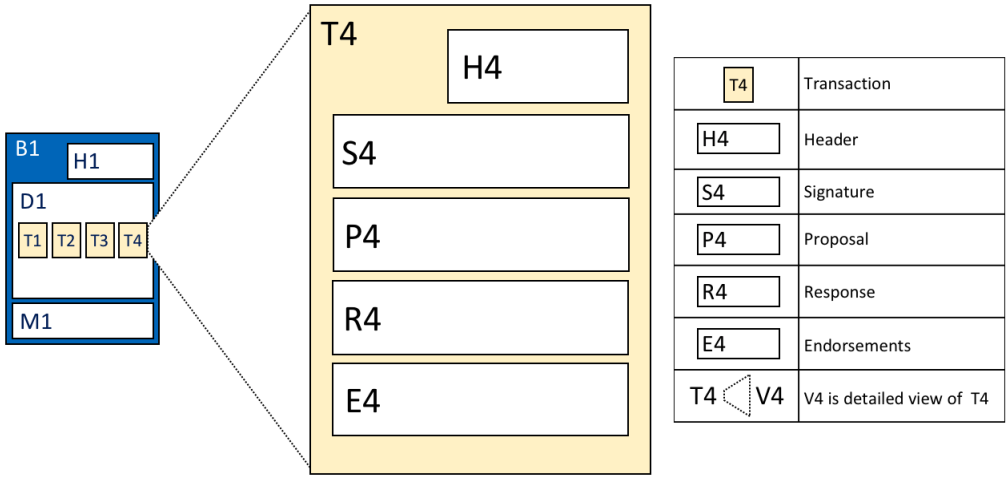


Figure 2-13: Transactions design

in the above figure, we can see the following fields:

- *Header:* captures some essential metadata about the transaction – for example, the name of the relevant chaincode, and its version.
- *Signature:* contains a cryptographic signature, created by the client application.
- *Proposal:* encodes the input parameters supplied by an application to the smart contract, which creates the proposed ledger update.
- *Response:* captures the before and after values of the world state, as a Read Write set (RW-set). It's the output of a smart contract, and if the transaction is successfully validated, it will be applied to the ledger to update the world state.
- *Endorsements:* this is a list of signed transaction responses from each required organization sufficient to satisfy the endorsement policy.

Now we have to talk about the world state in the ledger.The world state is physically implemented as a database, to provide simple and efficient storage and retrieval of ledger states.it could be a relational data store, or a graph store, or a temporal

database. The current option for the peer state database is LevelDB which is the default key-value state database embedded in the peer process but we are also going to use CouchDB. CouchDB is an alternative external state database which allows us to store data in JSON format and runs as a separate database process alongside the peer(therefore there are additional considerations in terms of setup, management, and operations) and using it can help us meet auditing and reporting requirements that are not supported by LevelDB.unlike the transaction log, each channel here will have different world state data that is modeled with its respective chaincode using JSON therefore we will leave the conception of it in the next section.

Consensus and Application layer

In Hyperledger Fabric, consensus is made up of three distinct steps Transaction endorsement, Ordering, and Validation and commitment which means it does not follow a single algorithm and this have been made clear in the previous sections and in the figure 2-6.Every network that uses the Hyperledger fabric platform will follow the same process and left the customizability of the consensus at the ordering service part, where the default is the RAFT consensus which it will be kept as it is because we can't see any point of changing it especially that we are using only one ordering node.

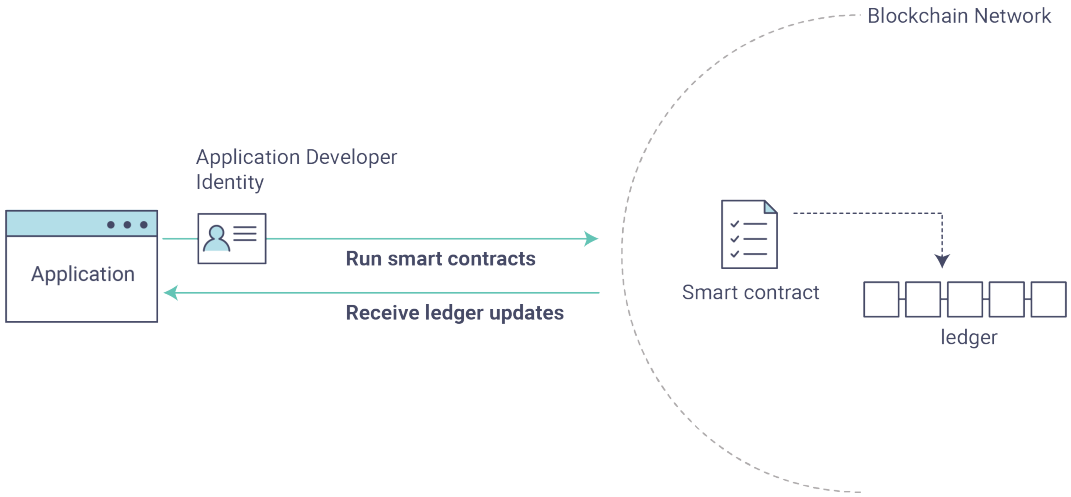


Figure 2-14: Application call overview

About the application layer, it will be responsible for exposing APIs to client applications like it shown in the figure 2-14 and this will only be possible by smart contracts (chaincode) therefore we will leave the conception of it to the next section.

2.2.2 The conception of the chaincode

For the purpose of building complete blockchain-based applications, the application layer needs to be implemented and in order to do that we need to provide two sub-layers of code which are the chaincode (For Interaction with the Blockchain and Ledger) and the API (For Interaction between Applications and Chaincode), fortunately Hyperledger fabric provides an SDK that offers a structured environment of libraries for us to write and test chaincode and APIs with ease.
In this part, we will get into chaincode implementation in our network, so our network becomes specific to students records. we have created four channels for the reason of separating the data, therefore we will deploy four chaincodes for each channel and each chaincode will be concerned with one asset to manage.

Channel one

Looking at the role that we gave to this channel, the asset here will be the student itself.

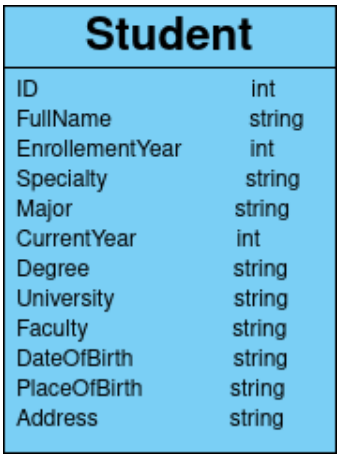


Figure 2-15: Student asset

We have to note that world state database will store the student asset as key

and value, where the key here will be the ID attribute of the student and the rest of attributes will be as value.We can see how it is going to shape:

```
{
   "ID":{
      "FullName":"fullname",
      "CurrentYear":"currentyear",
      "Speciality":"speciality",
      "EnrollementYear":"enrollementyear",
      "Major":"major",
      "Degree":"degree",
      "University":"university",
      "Faculty":"faculty",
      "DateOfBirth":"dateofbirth",
      "PlaceOfBirth":"placeofbirth",
      "Address":"adress"
   }
}
```

About the transactions log or the actual blockchain, and like we mentioned before, it will be responsible for storing all the operations that alter the state of the asset and in the most of the time it comes from client application (transaction proposals), so we will list all the important methods that we need to create to manage the student asset:

- *InitLedger():* supposed to add a base set of students to the ledger, it returns nothing in our case, but we can use it for testing purposes.
- *CreateStudent():* issues a new student to the world state with given details in its arguments.
- *ReadStudent():* returns the student stored in the world state with given ID.
- *UpdateStudent():* updates an existing student in the world state with provided parameters.

- *DeleteStudent():* deletes a given student from the world state.
- *StudentExists():* returns true when student with given ID exists in world state.
- *TransferStudent():* updates the University field of student with given ID in world state, and returns the old University.
- *ChangeMajor():* updates the Major field of student with given id in world state, and returns the old Major.
- *ChangeDegree():* updates the Degree field of student with given id in world state, and returns the old Degree.
- *ChangeCurrentYear:* updates the CurrentYear field of student with given id in world state, and returns the old CurrentYear.
- *GetAllStudents:* returns all students found in world state.

Channel two

The achievement certificates here will be the asset, where we can see that the ID of the certificate is the key and the rest will be considered as value.

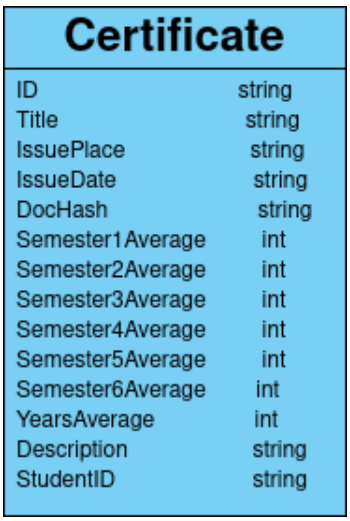


Figure 2-16: Certificate asset

And here's all the important methods that can be used at this asset:

- *InitLedger():* it does the same thing as the methode found in the students channel.
- *CreateCerteficat():* issues a new certificate to the world state with given details in its arguments.
- *ReadCerteficat():* returns the certificate stored in the world state with given ID.
- *UpdateCerteficat():* updates an existing certificate in the world state with provided parameters.
- *DeleteCerteficat():* deletes a given certificate from the world state.
- *CerteficatExists():* returns true when certificate with given ID exists in world state.
- *GetAllCerteficats:* returns all certificate found in world state.

Channel three

Like we state before, this channel will be responsible for storing student projects and work, so we can say that the asset is a project.

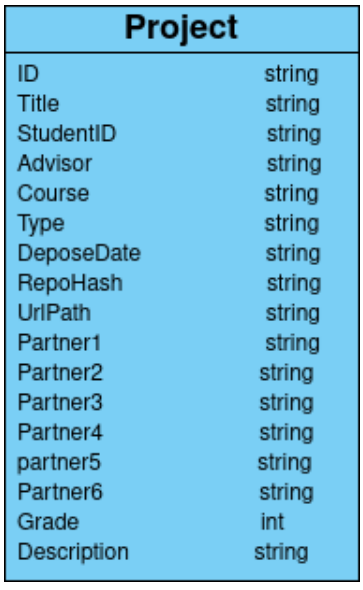


Figure 2-17: Project asset

And here's all the important methods that can be used at this asset:

- *InitLedger():* it does the same thing as the methode found in the students channel.
- *CreateProject():* issues a new Project to the world state with given details in its arguments.
- *ReadProject():* returns the Project stored in the world state with given ID.
- *UpdateProject():* updates an existing Project in the world state with provided parameters.
- *DeleteProject():* deletes a given Project from the world state.
- *GetAllProject:* returns all Project found in world state.

Channel four

we did mention that this channel will be responsible for storing student Events and clubs that he joined, so we can say that the asset here is an event.

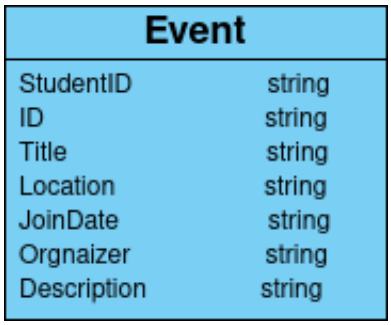


Figure 2-18: Event asset

And here's all the important methods that can be used at this asset:

- *InitLedger():* it does the same thing as the methode found in the students channel.
- *CreateEvent():* issues a new Event to the world state with given details in its arguments.

- *ReadEvent():* returns the Event stored in the world state with given ID.
- *UpdateEvent():* updates an existing Event in the world state with provided parameters.
- *DeleteEvent():* deletes a given Event from the world state.
- *EventExists():* returns true when an Event with given ID exists in world state.
- *GetAllEvent:* returns all Event found in world state.

We ought to note that all of these functions are the ones that are necessary to be created, we can add more specific function to the chaincode that run this system.

2.3 Conclusion

In this chapter, we got introduced to the blockchain framework Hyperledger Fabric, where we saw how it functions and how it will help us in applying our project goals. we also get into planning and designing the overall network, where it got done from the infrastructure layer to the application layer. In the next chapter, we will run and execute the project, and we will describe every aspect of it.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Production and realization of test

In this chapter, we will go through the last phase of our project development, which is the testing stage.Testing should ensure that each function works correctly and helps us to know if we reached our goals. We will go into every functionality, and we will describe every aspect of the project here.

3.1 Environment description

Environment here means all necessary files and software environments relating to our project, including, without limitation, separate and distinct software environments for each of development, testing and production, of our project product.

3.1.1 Software Environment

Software used

- *Atom IDE:* Atom is a free and open-source text and source code editor with support for plug-ins written in JavaScript, and embedded Git Control. Developed by GitHub, Atom is a desktop application built using web technologies. We used it for creating all of our shell scripts and smart contracts.
- *Bash (Unix shell):* Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne

shell.it is a command processor that typically runs in a text window, where the user types commands that cause actions. Bash can also read and execute commands from a file, called a shell script. Like most Unix shells, it supports filename globbing (wildcard matching), piping, here documents, command substitution, variables, and control structures for condition-testing and iteration. The keywords, syntax, dynamically scoped variables and other basic features of the language are all copied from sh. it is important to be included here because the entire project relies heavenly on it, which means that the project requires a Linux environment to be able to run.

- *Docker:* Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers are made possible by process isolation and virtualization capabilities built into the Linux kernel. These capabilities - such as control groups (Cgroups) for allocating resources among processes, and namespaces for restricting a processes access or visibility into other resources or areas of the system - enable multiple application components to share the resources of a single instance of the host operating system in much the same way that a hypervisor enables multiple virtual machines (VMs) to share the CPU, memory and other resources of a single hardware server. And it was our key component that let us execute our test network in a single host with the help of its built in tools like docker composer and docker volumes, and enables the smart contract to be reusable in other environments.

- *Git:* is a software for tracking changes in any set of files. every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities. it helps us tremendously from the beginning to the end of the project.

- *Creately:* is a SaaS visual collaboration tool with diagramming and design ca-

pabilities designed by Cinergix. we used to visualize our conception of the project.

- *Overleaf:* is a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents.we used it to write this paper.

Frameworks and libraries used

- *Hyperledger Fabric:* is a platform for distributed ledger solutions, underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility and scalability. It is designed to support pluggable implementations of different components, and accommodate the complexity and intricacies that exist across the economic ecosystem. Hyperledger Fabric delivers a uniquely elastic and extensible architecture, distinguishing it from alternative blockchain solutions.it was our starting point.it is the blockchain platform that we used in our project and facilitate the creating of the blockchain network and allows us to build our solution around it.

- *Yacht:* it is a web interface for managing docker containers with an emphasis on templating to provide one click deployments of dockerized applications.

- *CouchDB:* Apache CouchDB is an open-source document-oriented NoSQL database, implemented in Erlang. it uses multiple formats and protocols to store, transfer, and process its data. It uses JSON to store data, JavaScript as its query language using MapReduce, and HTTP for an API.Unlike a relational database, a CouchDB database does not store data and relationships in tables. Instead, each database is a collection of independent documents. Each document maintains its own data and self-contained schema.

Languages used

- *JSON:* JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

- *Golang:* Go is a statically typed, compiled programming language designed at Google.it is expressive, concise, clean, and efficient. Its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines, while its novel type system enables flexible and modular program construction. Go compiles quickly to machine code yet has the convenience of garbage collection and the power of run-time reflection. It's a fast, statically typed, compiled language that feels like a dynamically typed, interpreted language. we used it here to write our smart contract.

- *Bash scripting:* A Bash script is a plain text file which contains a series of commands. Bash (or shell) scripting is a great way to automate repetitive tasks and can save us a ton of time.

- *YAML:* it is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted. YAML targets many of the same communications applications as Extensible Markup Language but has a minimal syntax which intentionally differs from SGML.we used it for configuration files for docker.

3.1.2 The implementation repository description

in this part, we are going to describe the project implementation repository and give a brief explanation of the directories structure. there are four folders which are:

/hypmbh

It contains all the scripts and the config files needed to launch our network, create the channels and deploy our smart contracts.

- **/compose:** this folder contains all the compose files that docker compose will use to launch our network.compose files are used to define all the services that we need to run along with their configurations and environments variables.we can see here that compose-test-net.yaml defines all the peers needed and refers

to the core.yaml file in the peercfg folder so we can run the peers, compose-couch.yaml defines the CouchDB databases on each peer and compose-ca.yaml defines the CA servers of each organization. The purpose of thees files is to start all the services needed at once, instead of running them one by one. it will be used by the start.sh script that will use the docker-compose command.

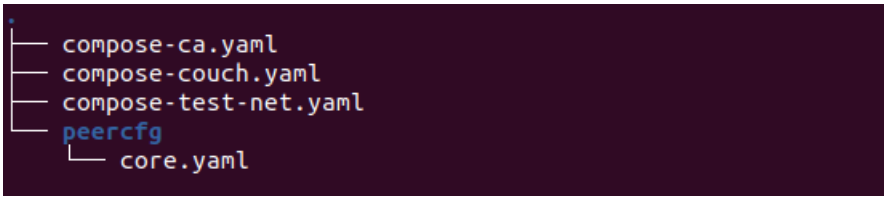


Figure 3-1: compose folder

- **/configtx:** the folder contains only the configtx.yaml file that contains the definitions for our channels for our sample network.it extends the configtx.yaml that found in the config directory.it is used by the configtxgen tool found in the bin directory to create all our channels' genesis blocks.The genesis block is defined using the "FourOrgsApplicationGenesis" profile at the bottom of the file. This profile defines an application channel consisting of our four Peer Orgs. The peer and ordering organizations are defined in the "Profiles" section at the top of the file. As part of each organization profile, the file points to the location of the MSP directory for each member. This MSP is used to create the channel MSP that defines the root of trust for each organization. In essence, the channel MSP allows the nodes and users to be recognized as network members.

- **/organizations:** Each organization needs to generate the crypto material that will define that organization on the network. Because Hyperledger Fabric is a permissioned blockchain, each node and user on the network needs to use certificates and keys to sign and verify its actions. In addition, each user needs to belong to an organization that is recognized as a member of the network.By default, the sample network uses cryptogen tool found in the bin directory.The cryptogen tool consumes a series of configuration files for each organization in the "/cryptogen" folder. it uses the files to generate the crypto material for each

org and put the output in this folder, the "organizations" folder. we can also use Fabric CAs to generate the crypto material. CAs sign the certificates and keys that they generate to create a valid root of trust for each organization. The start.sh script uses Docker Compose to bring up five CAs, one for each peer organization and the ordering organization. The configuration file for creating the Fabric CA servers are in the "/fabric-ca" folder.There are also :

- *ccp-generate.sh:* a script that uses the ccp-template.yaml and also uses ccp-template.json to generate connection description files for each organization that can be used for discovery service and applications.
- *registerEnroll.sh:* this script uses the Fabric CA client command that found in the bin directory to create the identities, certificates, and MSP folders that are needed to create the network and put the output inside this folder the "organizations" folder.

Figure 3-2: organizations folder

- **/scripts:** it contains a set of scripts that needed by start.sh script to run our network. we can see here the full list and the description of each of the scripts created:

Figure 3-3: scripts folder

- *utils.sh:* it is a helper script that help the users to use thees scripts. it contains the printHelp() function that shows the Help page of all the possible commands that we can run. it also contains other functions that help in showing update messages when running the commands, like error messages gets colored with red using errorln() or info messages that get colored with blue using infoln(). the other functions are println() that simply prints a string, successln() shows a success message colored in green, warnln() shows a warning message colored in yellow and fatalln() that shows an error message but exits the CLI for indicating a fatal error.
- *envVar.sh:* This is a collection of bash functions used by different scripts. We can find the function setGlobals() that sets environment variables for the peer orgs, setGlobalsCLI() that sets environment variables for use in the CLI container and the script itself sets the environment variables of the orderer and the CAs existed. The reason for this is that we are running the network in a single host, therefore when we want to use one of the registered entities in the network like an admin of a peer or a peer of an organization we need to point to its identity related files and configuration files. We can also find the helper function that sets the peer connection parameters for a chaincode operation parsePeerConnectionParameters() and the function verifyResult() that execute the fatalln() from util.sh script in case of a fatal error that caused by certain commands.
- *createChannel.sh:* as the name implies, this script is used to create channels but not only that, it also joins the peers to the created channels and sets the anchor peer of each organization (anchor peer is a peer node on

a channel that all other peers can discover and communicate with. Each Member on a channel has an anchor peer (or multiple anchor peers to prevent single point of failure), allowing for peers belonging to different Members to discover all existing peers on a channel).at first, the function createChannelTx() generates the genesis block using the configtxgen tool, than createChannel() function inside this script will create the actual channel and list all the created channels using the osnadmin tool, after that the script uses the joinChannel() to join each peer to the created channel using the peer tool, and at the end the anchor peer of each organization will be set using setAnchorPeer() that will run the setAnchorPeer.sh script inside the cli container.

- *setAnchorPeer.sh:* After setting the global environments of an organization, createAnchorPeerUpdate() will use the function fetchChannelConfig() from the configUpdate.sh script to get the current channel configurations then it modifies it using jq command and creates a new config file using the createConfigUpdate() found also in the configUpdate.sh script. after all this, updateAnchorPeer() will use the peer tool to submit the modified configuration file and sets the intended peer as anchor.

- *configUpdate.sh:* the script used for updating the configuration files when the network is running. it contains two functions. the first is fetchChannelConfig() that fetches the most recent configuration block for a channel using the peer commmand and then decods the config block to JSON using the configtxlator tool. The second function createConfigUpdate() takes an original and modified config, and produces the config update.

- *deployCC.sh:* this is the script that is responsible for deploying the chaincode on the network.at first it will check if all the needed arguments are present then it will build and compile the code source of the chaincode depending on the language used, after that it will package the chaincode using the packageChaincode() function that uses the peer tool. Now after

having a packaged chaincode, it will complete the rest of the lifecycle of a chaincode deployment by using functions from the ccutils.sh script where it will install chaincode on all the peers using installChaincode() then it will query whether the chaincode is installed on all peers using queryInstalled() then it will approve the definition of the chaincode and check whether the chaincode definition is ready to be committed on each organization using approveForMyOrg() and checkCommitReadiness() respectively now that we know for sure all orgs have approved, it will commit the definitions using commitChaincodeDefinition() on all peers then it will query on all orgs to see if the definitions committed successfully using queryCommitted(). the last step which is not mandatory unless it specified to be necessary using the chaincodeInvokeInit() the script will invoke the init function of the chaincode that will initialize the ledger.

  - *ccutils.sh:* the script contains a set of functions that the deployCC.sh script will need. we can find eight functions in it which are installChaincode(), queryInstalled(), approveForMyOrg(), checkCommitReadiness(), commitChaincodeDefinition(), queryCommitted() and lastly chaincodeInvokeInit(). we already saw how each function will be used, so it is not necessary to repeat this here, although we have to note that each of thees functions will use the peer tool to do its job.

- **start.sh:** it is our main scripts that launches our network and lets us create the channels and deploy all the smart contracts with the help of the mentioned scripts above. "./start.sh" the main command that will run the script with specifying the right arguments to run the intended function. the script contains the function networkUp() that call the createOrgs() function to create organizations crypto materials using cryptogen by default or CAs by calling the registerEnroll.sh script functions found in "/organizations/fabric-ca". then createOrgs() creates connection description files using the ccp-generate.sh found in "/organizations", finally the networkUp() brings up the peer and orderer nodes using

docker compose.it also contains the networkDown() function that tear down the running network, if it is run under the mode restart, it will stop all the running containers but if it is not, it will also remove all the containers, created images, the generated artifacts and all the generated data by calling the clearContainers() and removeUnwantedImages() functions. we can also find two other main functions which are the createChannel() that checks if the network is running and all the containers are present to run the createChannel.sh script for creating a channel. and we can find the deployCC() that calls the deployCC.sh script for deploying a chaincode.

/chanicode

All our business logic is here, where we can find all the smart contracts written for the purpose of storing student records. Each channel has its own set of smart contracts, therefore we created four folders, *channel1, channel2, channel3 and channel4*, each one of them contains a Go file that has the logic written on.

/config

Base configuration files that are needed to set up our peers, orderers, and channels.in the figure 3-4 we can see that there are three files. The core.yaml is for configuring the peers processes, the orderer.yaml is for configuring the orderers processes and the configtx.yaml is for channels related configurations. The binaries in the bin directory will heavenly rely on thees config files so as well as our network.

Figure 3-4: config directory

/bin

platform specific Hyperledger Fabric CLI tool binaries.it will help us to interact with the network where the scripts from the hypmbh directory will use it.

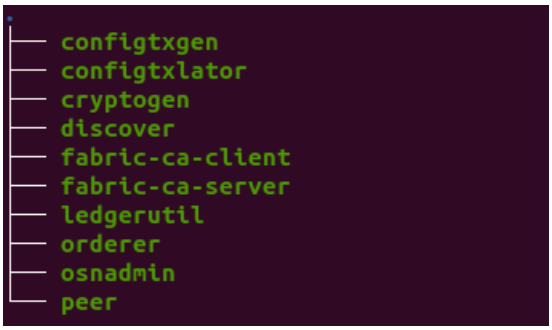


Figure 3-5: bin directory

- *configtxgen:* The configtxgen command allows users to create and inspect channel config related artifacts.The tool's output is largely controlled by the content of configtx.yaml. This file is searched for at FABRIC_CFG_PATH and must be present for configtxgen to operate.

- *configtxlator:* The configtxlator command allows users to translate between protobuf and JSON versions of fabric data structures and create config updates. The command may either start a REST server to expose its functions over HTTP or may be utilized directly as a command line tool.

- *cryptogen:* cryptogen is a utility for generating Hyperledger Fabric key material. It is provided as a means of preconfiguring a network for testing purposes. It would normally not be used in the operation of a production network.

- *discover:* the SDK needs a lot of information in order to allow applications to connect to the relevant network nodes, the discovery service have the peers to compute the needed information dynamically and present it to the SDK in a consumable manner.The discover command will launch its own Command Line Interface (CLI) which uses a YAML configuration file to persist properties such as certificate and private key paths, as well as MSP ID althoug this command is no longer that much needed in the new version of the Fabric SDK.

- *ledgerutil:* it mainly used for troubleshooting, like The ledgerutil compare command allows administrators to compare channel snapshots from two different peers. Although channel snapshots from the same height should be identical

across peers, if the snapshot files indicate different state hashes (as seen in _snapshot_signable_metadata.json file from each snapshot) then it indicates that at least one of the peers state databases is not correct relative to the blockchain and may have gotten corrupted.

- *orderer:* it is simply used to run an orderer node

- *osnadmin:* The osnadmin allows administrators to perform channel-related operations on an orderer, such as joining a channel, listing the channels an orderer has joined, and removing a channel. The channel participation API must be enabled, and the Admin endpoint must be configured in the orderer.yaml for each orderer.

- *fabric-ca-client:* The fabric-ca-client command allows us to manage identities (including attribute management) and certificates (including renewal and revocation).The Hyperledger Fabric CA is a Certificate Authority (CA) for Hyperledger Fabric.

- *fabric-ca-server:* The fabric-ca-server command allows you to initialize and start a server process which may host one or more certificate authorities.

- *peer:* The peer command has different subcommands, each of which allows administrators to perform a specific set of tasks related to a peer. For example, you can use the peer channel subcommand to join a peer to a channel, or the peer chaincode command to deploy a smart contract chaincode to a peer.

3.2 Testing realization

In this section, we are going to execute and run our project in order to examine the artifacts and the behavior of the project, test the functionalities built in it and review the deployment infrastructure and associated scripts.

3.2.1 Executing the network

now we get to the practical side of our project, where we will see how things will work out.in this part we will launch our whole network like it has been set in the design face, including the creation of the channels of course.

Any operation that we are able to do here will only rely on the start.sh script by running the ./start.sh command into the CLI. we can see down in the figure 3-6 the help page that show us all the different arguments that we can use.

```
Usage:
  start.sh <Mode> [Flags]
    Modes:
      up - Bring up Fabric orderer and peer nodes. No channel is created
      up createChannel - Bring up fabric network with one channel
      createChannel - Create and join a channel after the network is created
      deployCC - Deploy a chaincode to a channel (defaults to asset-transfer-basic)
      down - Bring down the network

    Flags:
    Used with start.sh up, start.sh createChannel:
    -ca <use CAs> -  Use Certificate Authorities to generate network crypto material
    -c <channel name> - Name of channel to create (defaults to "mychannel")
    -s <dbtype> - Peer state database to deploy: goleveldb (default) or couchdb
    -r <max retry> - CLI times out after certain number of attempts (defaults to 5)
    -d <delay> - CLI delays for a certain number of seconds (defaults to 3)
    -verbose - Verbose mode

    Used with start.sh deployCC
    -c <channel name> - Name of channel to deploy chaincode to
    -ccn <name> - Chaincode name.
    -ccl <language> - Programming language of the chaincode to deploy: go, java, javascript, typescript
    -ccv <version>  - Chaincode version. 1.0 (default), v2, version3.x, etc
    -ccs <sequence>  - Chaincode definition sequence. Must be an integer, 1 (default), 2, 3, etc
    -ccp <path>  - File path to the chaincode.
    -ccep <policy>  - (Optional) Chaincode endorsement policy using signature policy syntax. The default policy requires an endorsement all orgs
    -cccg <collection-config>  - (Optional) File path to private data collections configuration file
    -cci <fcn name>  - (Optional) Name of chaincode initialization function. When a function is provided, the execution of init will be requested and
the function will be invoked.

    -h - Print this message

 Possible Mode and flag combinations
   up -ca -r -d -s -verbose
   up createChannel -ca -c -r -d -s -verbose
   createChannel -c -r -d -verbose
   deployCC -ccn -ccl -ccv -ccs -ccp -cci -r -d -verbose
```

Figure 3-6: Help page

The first thing that we have to do is to run the command "./start.sh up" to bring all the nodes up along with -ca flag to use Fabric CA instead of the default cryptogen tool and -s couchdb flag to use CouchDB as a stat database instead of the default LevelDB.

At first, CAs orphan containers will start.

```
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'couchdb' with crypto from 'Certificate Authorities'
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_org3    ... done
Creating ca_org2    ... done
Creating ca_orderer ... done
Creating ca_org1    ... done
Creating ca_org4    ... done
```

Figure 3-7: CAs containers

then it will create the organization identities starting with enrolling the CA admins.

```
Creating Org1 Identities
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-org1 --tls.certfiles /home/mbh/Desktop/project/hypmbh/organizations/fabr
ic-ca/org1/ca-cert.pem
2022/06/01 08:57:11 [INFO] Created a default configuration file at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/f
abric-ca-client-config.yaml
2022/06/01 08:57:11 [INFO] TLS Enabled
2022/06/01 08:57:11 [INFO] generating key: &{A:ecdsa S:256}
2022/06/01 08:57:11 [INFO] encoded CSR
2022/06/01 08:57:11 [INFO] Stored client certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/msp/signcert
s/cert.pem
2022/06/01 08:57:11 [INFO] Stored root CA certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/msp/cacerts
/localhost-7054-ca-org1.pem
2022/06/01 08:57:11 [INFO] Stored Issuer public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/msp/IssuerPub
licKey
2022/06/01 08:57:11 [INFO] Stored Issuer revocation public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/ms
p/IssuerRevocationPublicKey
```

Figure 3-8: CA admin enroll

then it will register all the peers, users and admins.

```
Registering peer0
+ fabric-ca-client register --caname ca-org1 --id.name peer0 --id.secret peer0pw --id.type peer --tls.certfiles /home/mbh/Desktop/project/hypmbh/organ
izations/fabric-ca/org1/ca-cert.pem
2022/06/01 08:57:11 [INFO] Configuration file location: /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/fabric-ca-cl
ient-config.yaml
2022/06/01 08:57:11 [INFO] TLS Enabled
2022/06/01 08:57:11 [INFO] TLS Enabled
Password: peer0pw
Registering user
+ fabric-ca-client register --caname ca-org1 --id.name user1 --id.secret user1pw --id.type client --tls.certfiles /home/mbh/Desktop/project/hypmbh/org
anizations/fabric-ca/org1/ca-cert.pem
2022/06/01 08:57:12 [INFO] Configuration file location: /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/fabric-ca-cl
ient-config.yaml
2022/06/01 08:57:12 [INFO] TLS Enabled
2022/06/01 08:57:12 [INFO] TLS Enabled
Password: user1pw
Registering the org admin
+ fabric-ca-client register --caname ca-org1 --id.name org1admin --id.secret org1adminpw --id.type admin --tls.certfiles /home/mbh/Desktop/project/hyp
mbh/organizations/fabric-ca/org1/ca-cert.pem
2022/06/01 08:57:12 [INFO] Configuration file location: /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/fabric-ca-cl
ient-config.yaml
2022/06/01 08:57:12 [INFO] TLS Enabled
2022/06/01 08:57:12 [INFO] TLS Enabled
Password: org1adminpw
```

Figure 3-9: register peer, user and admin

then it will generate all the peer's, admin's and user's MSPs and certificates.

```
Generating the peer0 msp
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations
/org1.example.com/peers/peer0.org1.example.com/msp --csr.hosts peer0.org1.example.com --tls.certfiles /home/mbh/Desktop/project/hypmbh/organizations/f
abric-ca/org1/ca-cert.pem
2022/06/01 08:57:12 [INFO] TLS Enabled
2022/06/01 08:57:12 [INFO] generating key: &{A:ecdsa S:256}
2022/06/01 08:57:12 [INFO] encoded CSR
2022/06/01 08:57:12 [INFO] Stored client certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/peers/peer0.
org1.example.com/msp/signcerts/cert.pem
2022/06/01 08:57:12 [INFO] Stored root CA certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/peers/peer0
.org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2022/06/01 08:57:12 [INFO] Stored Issuer public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/peers/peer0.o
rg1.example.com/msp/IssuerPublicKey
2022/06/01 08:57:12 [INFO] Stored Issuer revocation public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/pe
ers/peer0.org1.example.com/msp/IssuerRevocationPublicKey
Generating the peer0-tls certificates
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations
/org1.example.com/peers/peer0.org1.example.com/tls --enrollment.profile tls --csr.hosts peer0.org1.example.com --csr.hosts localhost --tls.certfiles /
home/mbh/Desktop/project/hypmbh/organizations/fabric-ca/org1/ca-cert.pem
2022/06/01 08:57:12 [INFO] TLS Enabled
2022/06/01 08:57:12 [INFO] generating key: &{A:ecdsa S:256}
2022/06/01 08:57:12 [INFO] encoded CSR
2022/06/01 08:57:12 [INFO] Stored client certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/peers/peer0.
org1.example.com/tls/signcerts/cert.pem
2022/06/01 08:57:12 [INFO] Stored TLS root CA certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/peers/p
eer0.org1.example.com/tls/tlscacerts/tls-localhost-7054-ca-org1.pem
2022/06/01 08:57:12 [INFO] Stored Issuer public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/peers/peer0.o
rg1.example.com/tls/IssuerPublicKey
2022/06/01 08:57:12 [INFO] Stored Issuer revocation public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/pe
ers/peer0.org1.example.com/tls/IssuerRevocationPublicKey
```

Figure 3-10: generates peer's MSP and certeficats

```
Generating the user msp
+ fabric-ca-client enroll -u https://user1:user1pw@localhost:7054 --caname ca-org1 -M /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations
/org1.example.com/users/User1@org1.example.com/msp --tls.certfiles /home/mbh/Desktop/project/hypmbh/organizations/fabric-ca/org1/ca-cert.pem
2022/06/01 08:57:12 [INFO] TLS Enabled
2022/06/01 08:57:12 [INFO] generating key: &{A:ecdsa S:256}
2022/06/01 08:57:12 [INFO] encoded CSR
2022/06/01 08:57:12 [INFO] Stored client certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/users/User1@
org1.example.com/msp/signcerts/cert.pem
2022/06/01 08:57:12 [INFO] Stored root CA certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/users/User1
@org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2022/06/01 08:57:12 [INFO] Stored Issuer public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/users/User1@o
rg1.example.com/msp/IssuerPublicKey
2022/06/01 08:57:12 [INFO] Stored Issuer revocation public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/us
ers/User1@org1.example.com/msp/IssuerRevocationPublicKey
Generating the org admin msp
+ fabric-ca-client enroll -u https://org1admin:org1adminpw@localhost:7054 --caname ca-org1 -M /home/mbh/Desktop/project/hypmbh/organizations/peerOrgan
izations/org1.example.com/users/Admin@org1.example.com/msp --tls.certfiles /home/mbh/Desktop/project/hypmbh/organizations/fabric-ca/org1/ca-cert.pem
2022/06/01 08:57:12 [INFO] TLS Enabled
2022/06/01 08:57:12 [INFO] generating key: &{A:ecdsa S:256}
2022/06/01 08:57:12 [INFO] encoded CSR
2022/06/01 08:57:13 [INFO] Stored client certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/users/Admin@
org1.example.com/msp/signcerts/cert.pem
2022/06/01 08:57:13 [INFO] Stored root CA certificate at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/users/Admin
@org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2022/06/01 08:57:13 [INFO] Stored Issuer public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/users/Admin@o
rg1.example.com/msp/IssuerPublicKey
2022/06/01 08:57:13 [INFO] Stored Issuer revocation public key at /home/mbh/Desktop/project/hypmbh/organizations/peerOrganizations/org1.example.com/us
ers/Admin@org1.example.com/msp/IssuerRevocationPublicKey
```

Figure 3-11: generates admin's and user's MSPs and certificates

It will repeat the process on all organizations, then it generates their respective CCP files and at the end it will create the rest of the network services and the start our peers.

```
Generating CCP files for Org1 Org2 Org3 and Org4
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating volume "compose_peer0.org3.example.com" with default driver
Creating volume "compose_peer0.org4.example.com" with default driver
WARNING: Found orphan containers (ca_orderer, ca_org3, ca_org2, ca_org1, ca_org4) for this project. If you removed or renamed this service in your com
pose file, you can run this command with the --remove-orphans flag to clean it up.
Creating couchdb3            ... done
Creating couchdb2            ... done
Creating couchdb0               ... done
Creating couchdb1            ... done
Creating orderer.example.com ... done
Creating peer0.org3.example.com ... done
Creating peer0.org4.example.com ... done
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli                    ... done
```

Figure 3-12: finish the set-up

We can use Yacht to monitor all the functioning containers. we can see here all the containers that are up and running.

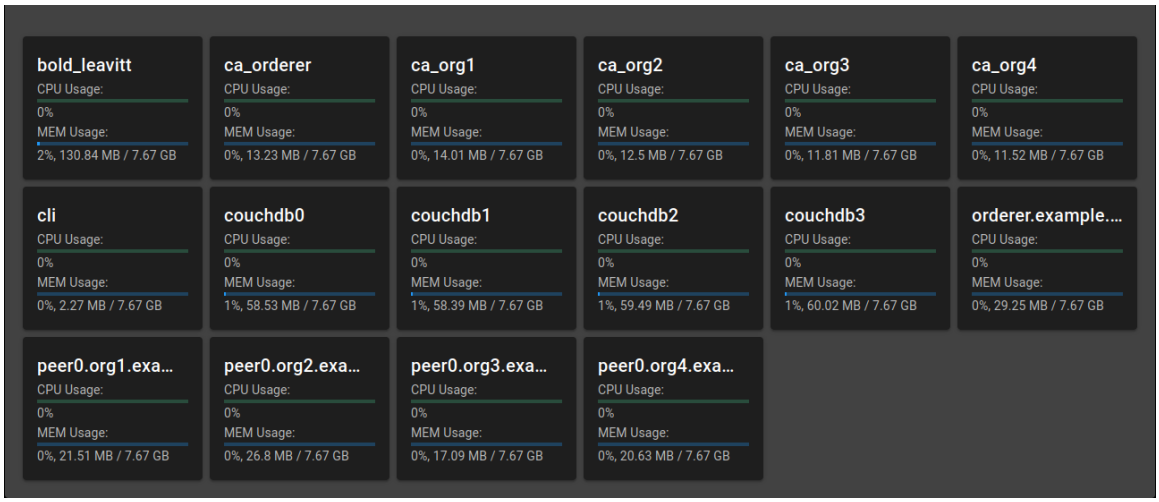


Figure 3-13: containers running

After this, we can now create all the channels needed. And this will be by typing "./start.sh createChannel" command with -c flag to name it. The first thing that happen is the generation of the genesis block that contains the configuration choices of that channel.

```
Using docker and docker-compose
Creating channel 'channel1'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'couchdb
Network Running Already
Generating channel create transaction 'channel1.tx'
+ configtxgen -profile FourOrgsApplicationGenesis -outputBlock ./channel-artifacts/channel1.block -channelID channel1
2022-06-01 10:31:08.439 CET 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-06-01 10:31:08.447 CET 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2022-06-01 10:31:08.447 CET 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tick
_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2022-06-01 10:31:08.447 CET 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /home/mbh/Desktop/project/hypmbh/configtx/c
onfigtx.yaml
2022-06-01 10:31:08.471 CET 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2022-06-01 10:31:08.471 CET 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2022-06-01 10:31:08.471 CET 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
```

Figure 3-14: Generating the genesis block

After that the channel will be actually created, and we can see that after listing the existing channels down the figure 3-15

```
Creating channel channel1
Using organization 1
+ osnadmin channel join --channelID channel1 --config-block ./channel-artifacts/channel1.block -o localhost:7053 --ca-file /home/mbh/Desktop/project/h
ypmbh/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --client-cert /home/mbh/Desktop/project/hypmbh/organizations/ord
ererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt --client-key /home/mbh/Desktop/project/hypmbh/organizations/ordererOrganizat
ions/example.com/orderers/orderer.example.com/tls/server.key
+ osnadmin channel list -o localhost:7053 --ca-file /home/mbh/Desktop/project/hypmbh/organizations/ordererOrganizations/example.com/tlsca/tlsca.exampl
e.com-cert.pem --client-cert /home/mbh/Desktop/project/hypmbh/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.c
rt --client-key /home/mbh/Desktop/project/hypmbh/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.key
Status: 200
{
        "systemChannel": null,
        "channels": [
                {
                        "name": "channel1",
                        "url": "/participation/v1/channels/channel1"
                }
        ]
}

+ res=0
Status: 201
{
        "name": "channel1",
        "url": "/participation/v1/channels/channel1",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}

Channel 'channel1' created
```

Figure 3-15: The creation of the channel

Now the time to join all the peers' organizations to the channel and repeat that's for every organization.

```
Joining org1 peer to the channel...
Using organization 1
+ peer channel join -b ./channel-artifacts/channel1.block
+ res=0
2022-06-01 10:17:45.751 CET 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-01 10:17:46.797 CET 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Joining org2 peer to the channel...
Using organization 2
+ peer channel join -b ./channel-artifacts/channel1.block
+ res=0
2022-06-01 10:17:49.850 CET 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-01 10:17:50.807 CET 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Joining org3 peer to the channel...
Using organization 3
+ peer channel join -b ./channel-artifacts/channel1.block
+ res=0
2022-06-01 10:17:53.864 CET 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-01 10:17:54.920 CET 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Joining org4 peer to the channel...
Using organization 4
+ peer channel join -b ./channel-artifacts/channel1.block
+ res=0
2022-06-01 10:17:57.997 CET 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-01 10:17:58.911 CET 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
```

Figure 3-16: Joining the peers

then the anchor peers will be set for each organization.

```
Setting anchor peer for org1...
Using organization 1
Fetching channel config for channel channel1
Using organization 1
Fetching the most recent configuration block for the channel
+ peer channel fetch config config_block.pb -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com -c channel1 --tls --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem
2022-06-01 09:18:00.201 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-01 09:18:00.218 UTC 0002 INFO [cli.common] readBlock -> Received block: 0
2022-06-01 09:18:00.218 UTC 0003 INFO [channelCmd] fetch -> Retrieving last config block: 0
2022-06-01 09:18:00.220 UTC 0004 INFO [cli.common] readBlock -> Received block: 0
Decoding config block to JSON and isolating config to Org1MSPconfig.json
+ configtxlator proto_decode --input config_block.pb --type common.Block --output config_block.json
+ jq '.data.data[0].payload.data.config' config_block.json
Generating anchor peer update transaction for Org1 on channel channel1
+ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1
.example.com","port": 7051}]},"version": "0"}}' Org1MSPconfig.json
+ configtxlator proto_encode --input Org1MSPconfig.json --type common.Config --output original_config.pb
+ configtxlator proto_encode --input Org1MSPmodified_config.json --type common.Config --output modified_config.pb
+ configtxlator compute_update --channel_id channel1 --original original_config.pb --updated modified_config.pb --output config_update.pb
+ configtxlator proto_decode --input config_update.pb --type common.ConfigUpdate --output config_update.json
```

Figure 3-17: set the anchor peer

we can see here the updated config file.

Figure 3-18: config file

we repeat the same process for every channel that we want to create. we can see here all the channels created.

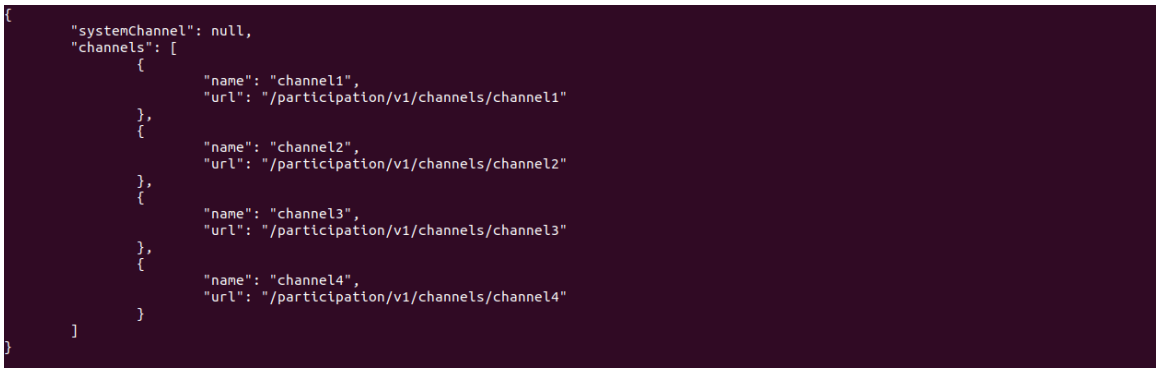


Figure 3-19: list of all the channels

Now our network is complete following what was on the design face and finally the network is up and running. Just want to note, in case we want to tear the network down, we will need to type the command "./start.sh down". at first, it will stop all the running containers.

```
Stopping network
Stopping cli                    ... done
Stopping peer0.org1.example.com ... done
Stopping peer0.org2.example.com ... done
Stopping peer0.org4.example.com ... done
Stopping peer0.org3.example.com ... done
Stopping couchdb0               ... done
Stopping couchdb1               ... done
Stopping couchdb2               ... done
Stopping couchdb3               ... done
Stopping ca_org4                ... done
Stopping ca_org1                ... done
Stopping ca_org2                ... done
Stopping ca_orderer             ... done
Stopping ca_org3                ... done
```

Figure 3-20: stopping the containers

then it will remove them and remove all the generated data and artifacts.

```
Removing cli                    ... done
Removing peer0.org1.example.com ... done
Removing peer0.org2.example.com ... done
Removing peer0.org4.example.com ... done
Removing peer0.org3.example.com ... done
Removing couchdb0               ... done
Removing orderer.example.com    ... done
Removing couchdb1               ... done
Removing couchdb2               ... done
Removing couchdb3               ... done
Removing ca_org4                ... done
Removing ca_org1                ... done
Removing ca_org2                ... done
Removing ca_orderer             ... done
Removing ca_org3                ... done
Removing network fabric_test
Removing volume compose_orderer.example.com
Removing volume compose_peer0.org1.example.com
Removing volume compose_peer0.org2.example.com
Removing volume compose_peer0.org3.example.com
Removing volume compose_peer0.org4.example.com
```

Figure 3-21: removing the containers

the final step, it will be deploying our chaincodes to each channel, so this network becomes specific to storing student records. starting with the first channel, we will use the command "./start.sh deployCC" with thees flags: the label of the chaincode, the version of the chaincode, the chaincode language, the name of the channel and the chaincode path.So we will be able to launch the deploying process. the first thing that will be done is packaging the chaincode like it is presented in the figure 3-22.

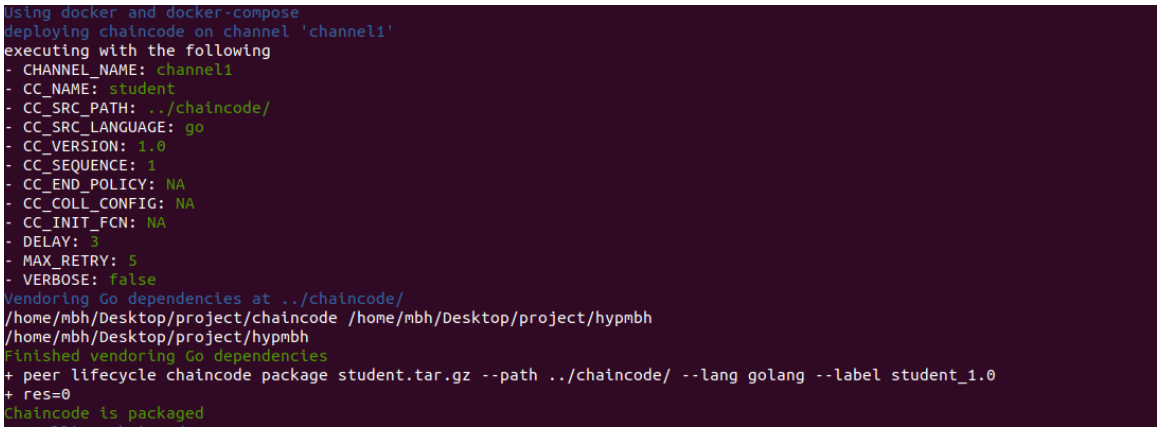


Figure 3-22: packaging the chaincode

the script will start installing the package on each peer, and then it will check if the package has been installed successfully by querying the chaincode.

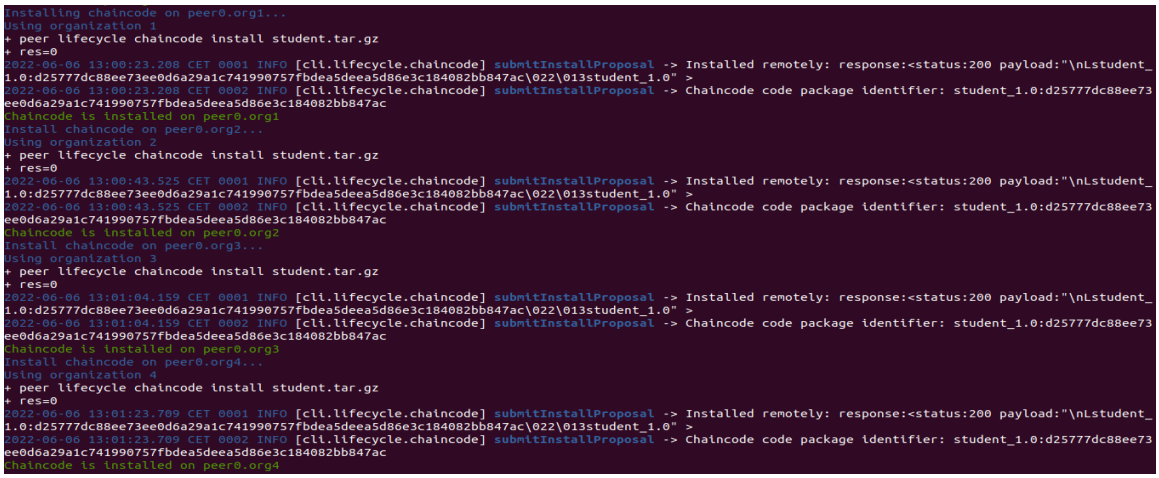


Figure 3-23: installing the chaincode

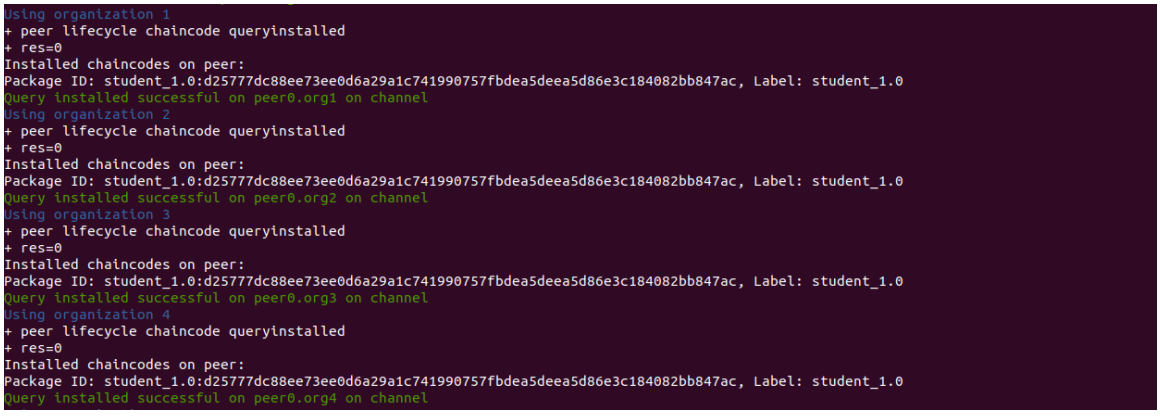


Figure 3-24: querying the chaincode

Now the chaincode needs to be approved of by every organization, so it can be committed to the channel.

each of the endorsing peers will start approving the chaincode and then checks whether all the other peers have approved.

The figure 3-25 shows that only org one has approved.

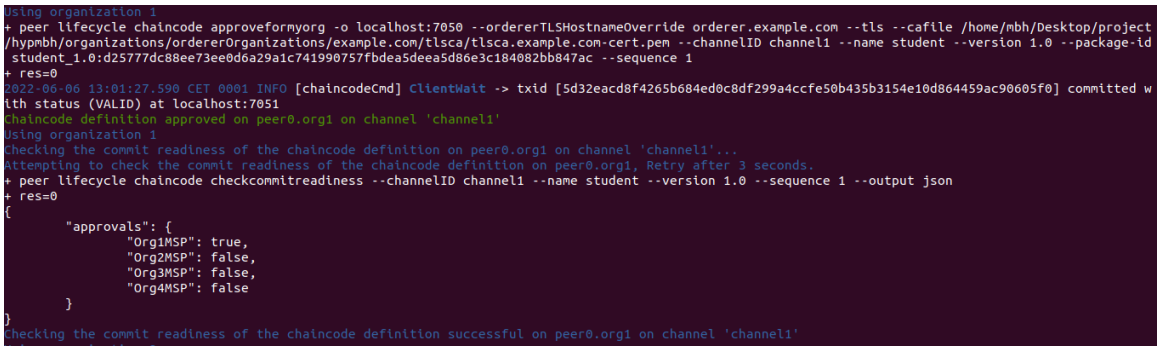


Figure 3-25: approving the chaincode

The figure 3-26 shows that all the orgs have approved and prepared to commit the chaincode definitions to the channel.

Figure 3-26: the chaincode approved

finally, the chaincode is being committed and ready to be used.

Figure 3-27: the chaincode committed

we can now check that by querying the chaincode definitions.

```
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'channel1'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID channel1 --name student
+ res=0
Committed chaincode definition for chaincode 'student' on channel 'channel1':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true, Org4MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'channel1'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'channel1'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID channel1 --name student
+ res=0
Committed chaincode definition for chaincode 'student' on channel 'channel1':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true, Org4MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'channel1'
Using organization 3
Querying chaincode definition on peer0.org3 on channel 'channel1'...
Attempting to Query committed status on peer0.org3, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID channel1 --name student
+ res=0
Committed chaincode definition for chaincode 'student' on channel 'channel1':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true, Org4MSP: true]
Query chaincode definition successful on peer0.org3 on channel 'channel1'
Using organization 4
Querying chaincode definition on peer0.org4 on channel 'channel1'...
Attempting to Query committed status on peer0.org4, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID channel1 --name student
+ res=0
Committed chaincode definition for chaincode 'student' on channel 'channel1':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true, Org4MSP: true]
Query chaincode definition successful on peer0.org4 on channel 'channel1'
```

Figure 3-28: querying the chaincode definitions

we can see that there are additional containers have been created in the figure 3-29

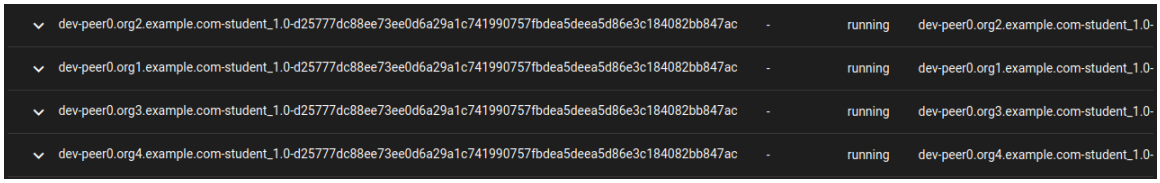


Figure 3-29: chaincode containers

Now, the first chaincode has been fully deployed to the first channel. we need to repeat the same process for every chaincode in every channel, so we can say that we finished our network.

3.2.2 Interacting with the network

Hyperledger Fabric provides us with a rich set of commands that let us interact with the elements of the network (peers, channels, orderers...), but here we need to focus on the commands that invoke the chaincode, so we can at least simulate what a client applications can do with our system and test our chaincode functionalities.
We are going to do simple CRUD operations on the student asset, starting with initiating the state database by invoking the InitLedger() function using "peer chaincode invoke" command with specifying the channel, the orderers and every endorsing peer.

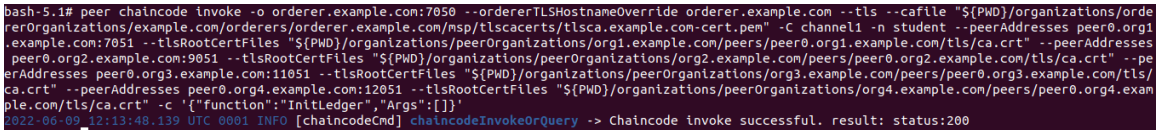


Figure 3-30: initiating the state database

using "peer chaincode query" that invokes the GetAllStudents() function will show all the students created

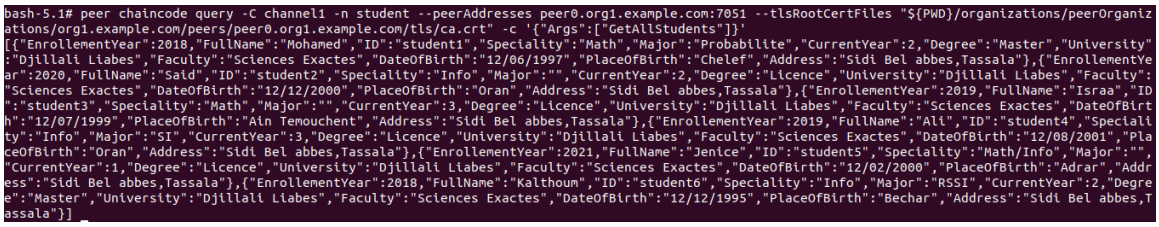
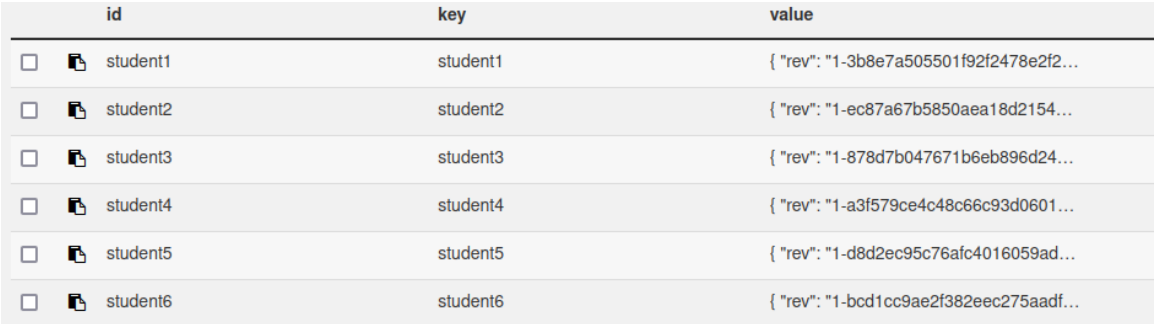


Figure 3-31: query all students

the figure 3-31 above shows all the students as raw data, but we can see the results better from the Fauxton WEB GUI interface of CouchDB.



	id	key	value
☐	student1	student1	{ "rev": "1-3b8e7a505501f92f2478e2f2...
☐	student2	student2	{ "rev": "1-ec87a67b5850aea18d2154...
☐	student3	student3	{ "rev": "1-878d7b047671b6eb896d24...
☐	student4	student4	{ "rev": "1-a3f579ce4c48c66c93d0601...
☐	student5	student5	{ "rev": "1-d8d2ec95c76afc4016059ad...
☐	student6	student6	{ "rev": "1-bcd1cc9ae2f382eec275aadf...

Figure 3-32: query all students using Fauxton

we can create a new student like it shown in the figure 3-33

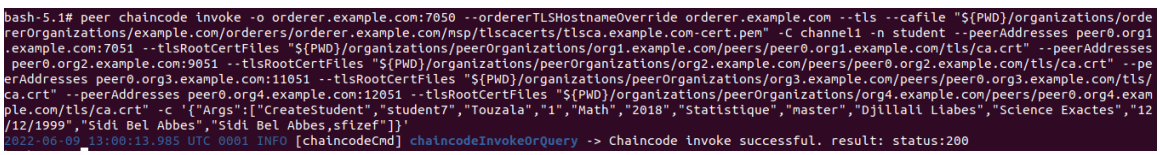


Figure 3-33: Create new student

we can see in the state database that the student has been successfully created.

	id	key	value
☐	student1	student1	{ "rev": "1-3b8e7a505501f92f2478e2f2…
☐	student2	student2	{ "rev": "1-ec87a67b5850aea18d2154…
☐	student3	student3	{ "rev": "1-878d7b047671b6eb896d24…
☐	student4	student4	{ "rev": "1-a3f579ce4c48c66c93d0601…
☐	student5	student5	{ "rev": "1-d8d2ec95c76afc4016059ad…
☐	student6	student6	{ "rev": "1-bcd1cc9ae2f382eec275aadf…
☐	student7	student7	{ "rev": "1-5ebd88a6ff8806828433797…

Figure 3-34: New student created

we can read that student information like it shown in the figure 3-35

```
bash-5.1# peer chaincode query -C channel1 -n student --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganiz
ations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" -c '{"Args":["ReadStudent","student7"]}'
{"EnrollementYear":2018,"FullName":"Touzala","ID":"student7","Speciality":"Math","Major":"Statistique","CurrentYear":1,"Degree":"master","University":
"Djillali Liabes","Faculty":"Science Exactes","DateOfBirth":"12/12/1999","PlaceOfBirth":"Sidi Bel Abbes","Address":"Sidi Bel Abbes,sfizef"}
```

Figure 3-35: Read student7

There are various functions to update the student information, but let us only try the TransferStudent() function to change the university of the student7

```
bash-5.1# peer chaincode invoke -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/orde
rerOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C channel1 -n student --peerAddresses peer0.org1
.example.com:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses
 peer0.org2.example.com:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" --pe
erAddresses peer0.org3.example.com:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/
ca.crt" --peerAddresses peer0.org4.example.com:12051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org4.example.com/peers/peer0.org4.exam
ple.com/tls/ca.crt" -c '{"Args":["TransferStudent","student7","Abu Bekr Belkaid"]}'
2022-06-09 14:46:05.814 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 payload:"Djillali Liabe
s"
```

Figure 3-36: Transfer student7

we can see the result by querying the student7 again.

```
bash-5.1# peer chaincode query -C channel1 -n student --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganiz
ations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" -c '{"Args":["ReadStudent","student7"]}'
{"EnrollementYear":2018,"FullName":"Touzala","ID":"student7","Speciality":"Math","Major":"Statistique","CurrentYear":1,"Degree":"master","University":
"Abu Bekr Belkaid","Faculty":"Science Exactes","DateOfBirth":"12/12/1999","PlaceOfBirth":"Sidi Bel Abbes","Address":"Sidi Bel Abbes,sfizef"}
```

Figure 3-37: Read Updated student7

if we want to delete a student we can do that also let us try it on the student3.

```
bash-5.1# peer chaincode invoke -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/orde
rerOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C channel1 -n student --peerAddresses peer0.org1
.example.com:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses
 peer0.org2.example.com:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" --pe
erAddresses peer0.org3.example.com:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/
ca.crt" --peerAddresses peer0.org4.example.com:12051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org4.example.com/peers/peer0.org4.exam
ple.com/tls/ca.crt" -c '{"Args":["DeleteStudent","student3"]}'
2022-06-09 16:37:00.013 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
```

Figure 3-38: Delete student3

	id	key	value
☐	student1	student1	{ "rev": "1-3b8e7a505501f92f2478e2f2...
☐	student2	student2	{ "rev": "1-ec87a67b5850aea18d2154...
☐	student4	student4	{ "rev": "1-a3f579ce4c48c66c93d0601...
☐	student5	student5	{ "rev": "1-d8d2ec95c76afc4016059ad...
☐	student6	student6	{ "rev": "1-bcd1cc9ae2f382eec275aadf...
☐	student7	student7	{ "rev": "2-87382ea2e32f9322531a77a...

Figure 3-39: student3 Deleted

At the end, we can say that almost the same operations can go for the other chaincodes so no need to test all of them here, and that we can trigger them using commands or by client applications that needs to use the Hyperledger Fabric SDK in order to be able to call our chaincode.

3.3 Conclusion

In this chapter we get through our project implementation where we saw all the scripts and files created, and we saw all the tools and languages that helped us through our entire journey. we also ran and executed our project where we tested every functionality built in. we created the network that contains the four organizations and one orderer. each organization had one peer and its own CA server. All the organization were associated with four channels. each channel has its own set of smart contracts that govern it own set of data.

THIS PAGE INTENTIONALLY LEFT BLANK

General Conclusion

By avoiding human verification of transcripts and other papers, we can save money and prevent fraud, we can increase innovation if there is a space for business and educational institutions to collaborate, we can fully harness the code ability to automate things, so time can be saved and costs will be less, we can give the students the full ownership of their lifelong learning credentials and achievements and protect their IPs and full records, all this is not things that we are dreaming of, but things that the blockchain technology can actually achieve therefore we can see that blockchain technology has huge potential to assist education institutions.And because of this, we decided to explore the potential of this technology, and successfully we built a project that centered around it that helpfully will help our university to over all the problems that this technology could solve even thought it is challenging to attain. Certainly, there is plenty of room for improvement in our implementations.We can scale it to include more institutions, which means a more secure system. We can also store more specific data like the attendee record of a student. also we have to note that the system is not ready for the end users therefore more developers needs to create application and build solutions in top of our implementation. At the end, we can say that our project could be much more of assist and help and that we see that Blockchain has the potential to reshape the education industry, the leading members must start learning about and experimenting with the technology now, both individually and as part of a collaboration, so our educational institution can be ready for the future.

THIS PAGE INTENTIONALLY LEFT BLANK

References

[1] Sudheer Hanumanthakari A. R. Sathya, Sandeep Kumar Panda. *Blockchain Technology: Applications and Challenges*, volume 203 of *Intelligent Systems Reference Library*, chapter 10, pages 170,171. Springer, first edition, 2021.

[2] Sudheer Hanumanthakari A. R. Sathya, Sandeep Kumar Panda. *Blockchain Technology: Applications and Challenges*, volume 203 of *Intelligent Systems Reference Library*, chapter 10, pages 173,174,175. Springer, first edition, 2021.

[3] Sujata Priyambada Dash Ajay Kumar Jena. *Blockchain Technology: Applications and Challenges*, volume 203 of *Intelligent Systems Reference Library*, chapter 1, page 6. Springer, first edition, 2021.

[4] Sujata Priyambada Dash Ajay Kumar Jena. *Blockchain Technology: Applications and Challenges*, volume 203 of *Intelligent Systems Reference Library*, chapter 1, page 8. Springer, first edition, 2021.

[5] Hasib Anwar. Best blockchain platforms for enterprises. https://101blockchains.com/best-blockchain-platforms/, JUN 2020.

[6] Rafael Belchior. Hyperledger fabric : Technical overview. https://towardsdatascience.com/hyperledger-fabric-technical-overview-a63046c2a430, NOV 2019.

[7] 101 Blockchains. Top blockchain platforms and enterprise solutions to choose from. https://101blockchains.com/blockchain-platforms, JLY 2021.

[8] Guillaume Bonnot. [blockchain] introduction to the hyperledger ecosystem. https://medium.com/@bonnotguillaume/blockchain-introduction-to-the-hyperledger-ecosystem-23279a090c4f, JUN 2021.

[9] cointelegraph. A beginner's guide to understanding the layers of blockchain technology. https://cointelegraph.com/blockchain-for-beginners/a-beginners-guide-to-understanding-the-layers-of-blockchain-technology, jan 2021.

[10] Forrest Colyer. Introducing hyperledger fabric 2.2 lts support on amazon managed blockchain. https://aws.amazon.com/blogs/database/introducing-hyperledger-fabric-2-2-lts-support-on-amazon-managed-blockchain/, MAR 2022.

[11] Constantine Constantinides. introduction of blockchain technology to the health tourism sector. https://www.linkedin.com/pulse/introduction-blockchain-technology-health-tourism-constantine/, DEC 2021.

[12] Elad Elrom. *The Blockchain Developer: A Practical Guide for Designing, Implementing, Publishing, Testing, and Securing Distributed Blockchain-based Projects*, chapter 1, page 29. Apress, first edition, 2019.

[13] Ethereum. *Blocks*, JUN 2022. Accessd on: https://ethereum.org/en/developers/docs/blocks/.

[14] Ethereum. *Consensus mechanisms*, JUN 2022. Accessd on: https://ethereum.org/en/developers/docs/consensus-mechanisms/.

[15] John Evans. Blockchain nodes: An in-depth guide. https://nodes.com/, 2021.

[16] Diego Geroni. Hybrid blockchain: The best of both worlds. https://101blockchains.com/hybrid-blockchain/, JAN 2021.

[17] Alexander S. Gillis. Hyperledger. https://www.techtarget.com /searchcio/definition/Hyperledger, MAY 2021.

[18] Houshyar Honar Pajooh, Mohammad Rashid, Fakhrul Alam, and S.N. Demidenko. Hyperledger fabric blockchain for securing the edge internet of things. *Sensors*, 21, 01 2021.

[19] Hyperledger Fabric. *A Blockchain Platform for the Enterprise*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/.

[20] Hyperledger Fabric. *Channels*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html.

[21] Hyperledger Fabric. *Fabric chaincode lifecycle*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/chaincode$_l ifecycle.html$.

[22] Hyperledger Fabric. *Identity*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/identity/identity.html.

[23] Hyperledger Fabric. *Introduction*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html.

[24] Hyperledger Fabric. *Ledger*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html.

[25] Hyperledger Fabric. *Membership Service Provider (MSP)*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/membership/membership.html.

[26] Hyperledger Fabric. *The Ordering Service*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering$_s ervice.html$.

[27] Hyperledger Fabric. *Peers*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/peers/peers.html.

[28] Hyperledger Fabric. *Policies*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/policies/policies.html.

[29] Hyperledger Fabric. *Security Model*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/security$_m$*odel.html*.

[30] Hyperledger Fabric. *Smart Contracts and Chaincode*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html.

[31] Hyperledger Fabric. *Transaction Flow*, 2022. Accessd on: https://hyperledger-fabric.readthedocs.io/en/latest/txflow.html.

[32] Gwyneth Iredale. History of blockchain technology: A detailed guide. https://101blockchains.com/history-of-blockchain-timeline, NOV 2020.

[33] Gwyneth Iredale. What are the different types of blockchain technology? https://101blockchains.com/types-of-blockchain/, jan 2021.

[34] Alok Kumar Jain. Blockchain goes to school. *Cognizant 20-20 Insights*, page 3, mar 2019.

[35] Alok Kumar Jain. Blockchain goes to school. *Cognizant 20-20 Insights*, pages 6,7,8,9, mar 2019.

[36] Alok Kumar Jain. Blockchain goes to school. *Cognizant 20-20 Insights*, pages 10,11, mar 2019.

[37] Naveen Joshi. Breaking down the blockchain architecture. https://www.allerin.com/blog/breaking-down-the-blockchain-architecture, jan 2020.

[38] Mauro Krikorian. Short intro to hyperledger fabric service. https://medium.com/southworks/short-intro-to-hyperledger-fabric-service-351f8e0040c8, MAY 2021.

[39] Yohenba Kshetrimayum. Blocks not generating while mining on a private ethereum. https://medium.com/data-science-community-srm/energy-blockchain-f51d558c5da2, FEB 2021.

[40] Anubhab Paul. Layered structure of the blockchain architecture. https://blog.ineuron.ai/Layered-structure-of-the-blockchain-architecture-b2DITCkF7f, FEB 2021.

[41] Pluralsight. Blockchain architecture. https://www.pluralsight.com/guides/blockchain-architecture, JAN 2019.

[42] Pratyusa Mukherjee Chittaranjan Pradhan. *Blockchain Technology: Applications and Challenges*, volume 203 of *Intelligent Systems Reference Library*, chapter 3, page 35. Springer, first edition, 2021.

[43] Mayank Sahu. Cryptography in blockchain: Types applications. https://www.upgrad.com/blog/cryptography-in-blockchain/, JAN 2021.

[44] Seethamsetty Uday Kumar Vijaykumar R. Urkude Shubhangi V. Urkude, Himanshu Sharma. *Blockchain Technology: Applications and Challenges*, volume 203 of *Intelligent Systems Reference Library*, chapter 4. Springer, first edition, 2021.

[45] Amarpreet Singh. Cryptography in blockchain explained. https://medium.com/brandlitic/cryptography-in-blockchain-explained-df11fe1bd0f7, apr 2021.

[46] Eugene Tarasenko. Benefits of blockchain hyperledger fabric. https://merehead.com/blog/benefits-of-blockchain-hyperledger-fabric/, MAR 2020.

[47] Gopi Manoj Vuyyuru. Ranchimall digest 3. https://medium.com/@manojvolk/ranchimall-digest-2-83cdb9c67bd2, MAR 2018.

[48] Kathleen E. Wegrzyn Eugenia Wang. Types of blockchain: Public, private, or something in between. https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between, AUG 2021.

[49] Erkan Yalcinkaya, Antonio Maffei, and Mauro Onori. Blockchain reference system architecture description for the isa95 compliant traditional and smart manufacturing systems. *Sensors*, 20(22), 2020.

[50] Linchao Zhang and Dohyeun Kim. A peer-to-peer smart food delivery platform based on smart contract. *Electronics*, 11(12), 2022.