|  |
| --- |
| **Nordic Blockchain Semi-Centralized Distributed Ledger Technology for banks** |

**Luca Francioni (240068)**

**Supervisor: Michael Viuff**

**VIA University College - Horsens**

**[Number of characters]**

**ICT Engineering**

**7th Semester**

**Table of content**

[Abstract iv](#_Toc9871722)

[1 Introduction 1](#_Toc9871723)

[2 Domain Model 2](#_Toc9871724)

[3 Analysis 6](#_Toc9871725)

[3.1 Requirements 6](#_Toc9871726)

[3.2 Functional Requirements 7](#_Toc9871727)

[3.3 Non-Functional Requirements 9](#_Toc9871728)

[4 Design 11](#_Toc9871729)

[4.1 Entity structures 11](#_Toc9871730)

[4.1.1 Transaction 11](#_Toc9871731)

[4.1.2 Block 12](#_Toc9871732)

[4.1.3 Blockchain 13](#_Toc9871733)

[4.1.4 Node 14](#_Toc9871734)

[4.1.5 Encapsulation (Crypto Layered Message) 14](#_Toc9871735)

[4.1.6 Transaction Request Path 16](#_Toc9871736)

[4.1.7 Miner Confirmation Process 17](#_Toc9871737)

[4.1.8 Operator Statistics Request 18](#_Toc9871738)

[4.1.9 Genesis Block 19](#_Toc9871739)

[4.1.10 Serialization 20](#_Toc9871740)

[4.1.11 Overall architecture 21](#_Toc9871741)

[4.2 Choice of technologies 23](#_Toc9871742)

[4.2.1 Asymmetric Cryptography 23](#_Toc9871743)

[4.2.2 WebSocket 23](#_Toc9871744)

[4.2.3 .NET Core 24](#_Toc9871745)

[4.2.4 NuGet Package Manager 24](#_Toc9871746)

[5 Implementation 25](#_Toc9871747)

[5.1 Block 25](#_Toc9871748)

[5.2 Blockchain 26](#_Toc9871749)

[5.3 Operations 28](#_Toc9871750)

[5.3.1 Transaction Operation 29](#_Toc9871751)

[5.4 Network 30](#_Toc9871752)

[5.5 Test Miner 31](#_Toc9871753)

[6 Test 31](#_Toc9871754)

[6.1.1 Testing practices 31](#_Toc9871755)

[6.1.2 Test Results 33](#_Toc9871756)

[6.2 Test Specifications 33](#_Toc9871757)

[6.2.1 Specification 1: User’s transaction placement, confirmation and verification. 33](#_Toc9871758)

[6.2.2 Specification 2: Administrator’s request of statistics. 34](#_Toc9871759)

[6.2.3 Specification 3: Miner’s fetch of pending operation (transaction) and confirmation. 34](#_Toc9871760)

[7 Results and Discussion 34](#_Toc9871761)

[8 Conclusions 37](#_Toc9871762)

[9 Sources of information 38](#_Toc9871763)

[10 Appendices 1](#_Toc9871764)

# Abstract

Nordic Blockchain is a software service for banks, its purpose is to replace previous mechanisms and systems used in banks for transactions with a modern blockchain based one, reducing costs of operations and improving inter-banks data synchronization and trust; it offers a backend oriented only plan, as the project aims to become a starting point for future projects based on the same logic of this.

The following report describes the analysis, design and implementation of such service but properly distinguishing the intended design and the resulting prototype. The project also raises technical and security aspects that can be subject to academic scrutiny and further development.

The appendix section contains the full code of the service; documentation towards the modular frameworks and components being used are described in the document.

The service is structured with the intent of being easily understandable, replicable and extendable.

# Introduction

The banks’ services have been alive and used for millennia, being the core structure of the worldwide economy development and actively participating in assets exchange, monetary value fluctuation, investments and everyday money exchange for services.

The transfer of assets of a bank’s customer to another bank’s customer is a process that involves several systems – either human or technology based - that must be properly configured and synchronized, offering as well adequate security and control for the bank’s administration, transparency and certainty for both the banks involved that the transactions are valid, secure and effectively requested by the originating customer: this may lead to slow downs and the overall system must always be available, scalable and stable.

The current systems involve years of software and hardware development and research, making a user operation fee based his/her desire of guaranteed speed of processing for the operation to be completed and the complexity of the transaction request path (e.g. international transactions), yet faster than physical writing and human elaboration, but still exposing the mechanism to exploits by malicious individuals (e.g. frauds, fake signatures, system penetrations, phishing, etc.).

Nordic Blockchain changes the way the whole underlying structure of a bank transaction management system work, unlocking unprecedented levels of scalability, speed of the operation and its costs, including security and transparency between multiple banks with absolute certainty of validity of the data by replacing the traditional systems with a modern blockchain design. Aiming to be the most independent and collaborative middle-ware system for transactions of assets and multiple possible operations that need to be synchronized between multiple banks in real time with least effort possible and maximum speed of transmission available; putting the storage control system software in each bank internal system and allowing banks to run undefined number of elaboration software components to process the data with fraud and intrusion detection algorithms, including already adopted ones.

The blockchain technology has a strong backward data validity, allowing intrusion or tampering of transactions almost impossible and immediately reported to the administration party, also allowing the banks to apply real-time monitoring for the operations inside the system and extending it with fewer and simpler workload and simplified testing, given the modularity nature of a blockchain; completely removing certain attack vectors, giving the chances only to phishing or physical danger.

The Nordic Blockchain is designed in a way that can be integrated, or be the replacement, to the current software systems without the bank’s customers notice as it mimics the same current functionalities while radically changing the mechanism behind it and the quality of life.

# Domain Model

The current monitoring and traceability system used by bank is SWIFT, it is a global cooperative owned software used for secure financial messaging services.

The software is from 1974 and is subject to millions of transactions requests to elaborate worldwide, considered the current best due to high security standards, better bandwidth usage and highly monitoring capabilities.

Such technology is, however, not using publicly known structure but is instead using RESTful APIs architecture, making the system especially centralized and closed and/or slowing the process of access another bank’s transactions logs.

The usage of a blockchain, a chain of “blocks”, containing transactions and being synchronized in real time by all the Nordic Blockchain’s network nodes (Figure 1), completely removes slowdowns of accessing certain essential information from the monitoring systems and significantly speeds up the process of a transaction request, as well as offering flexibility for temporary offline nodes, that only have to download latest version of the blockchain to get back to operative status.

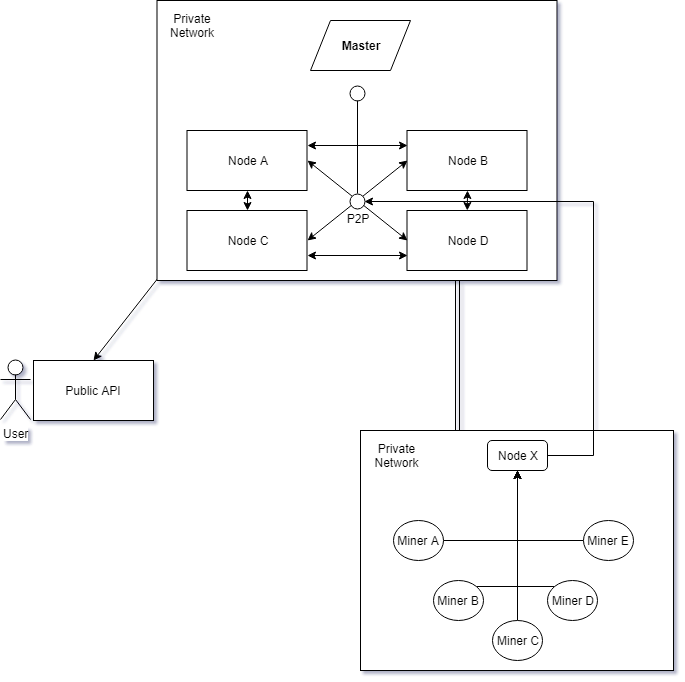


Figure 1 - The overview of a Nordic Blockchain network

In a blockchain, the key role of the bank becomes the node, the node is a copy of the blockchain that continuously synchronizes its content over the time, becomes the single bank’s internal transaction management system to which its customers transactions pass through and queued in the miner’s pending operations.

The node is also the internal monitor system, as it checks the validity of the transaction requests’ origin before being inserted into the queue.

The nodes are not able to communicate to each other directly and transactions aren’t immediately sent to the destination bank specified by the user.

This introduces the known figure of the miner: its original schedule – using Bitcoin as comparison reference – is to discover new block trough specific cryptographic operations, where new transactions are going to be stored between the nodes; and the verification process that the transaction is valid and trustworthy.

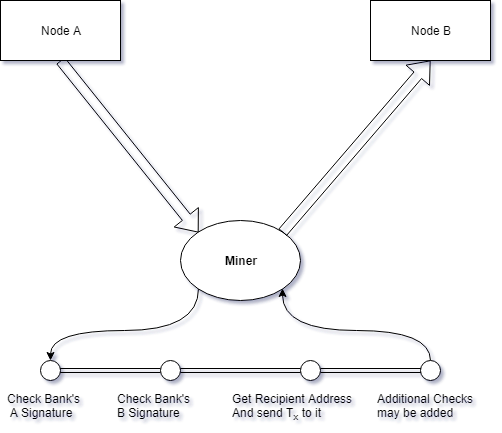


Figure 2 - The miner's process of verification of a transaction

In Nordic Blockchain, the miner’s role remains similar, yet different, as the miner’s objective is to verify the validity of the origin and destination node of the transaction and verify the validity of the transaction content itself without knowing its content (Figure 2), as the miner is being deployed by the bank, and not independent users.

Trough this process, secrecy of the content (such as asset value) is not disclosed but its trustworthy is verified.

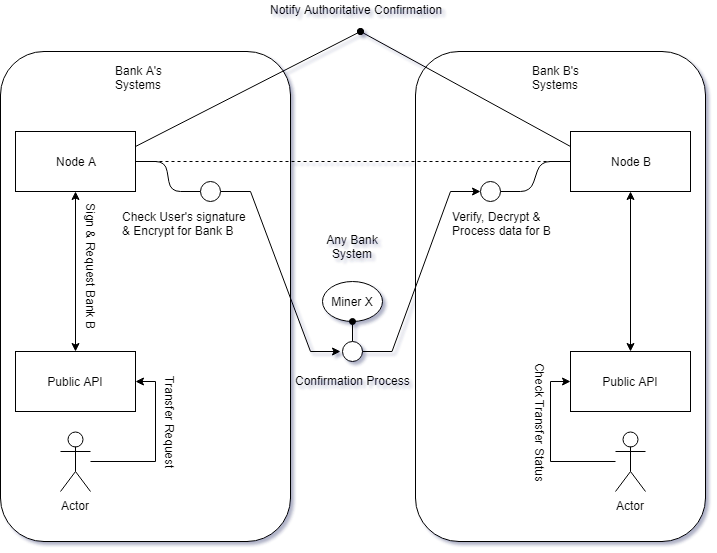


Figure 3 - Miner's macro-role in the Nordic Blockchain network

When a miner ends its verification process, a verdict is sent to the Nordic Blockchain network by increasing the transaction’s synchronized counter by 1.

Once the confirmation counter of a transaction reaches a fair amount, the transaction is to be considered trustworthy, hence, confirmed and concluded (Figure 3).

The process above described may appear long, but given the correct setup of the nodes and resources allocated to the software and enough miner being run by the collaborative network, the transaction time can be drastically cut to minutes (internationally) and the costs of the transaction fees is reduced as well, due to a significant cut of bureaucracy processes that can now be slightly adapted to a natively semi-transparent system; therefore the more miner are deployed in the internal network the faster the transactions are processed, unlocking a significant scalability solution by simply deploying more instances on running systems.

As of the monitoring, the miner’s process can be easily customized by adding custom steps, these can be also the current existing monitoring platforms, allowing perfect modularity and extension of the whole Nordic Blockchain platform with little to no risk of unintentional sabotage/misconfiguration and an easier migration from previous platforms to Nordic Blockchain.

# Analysis

## Requirements

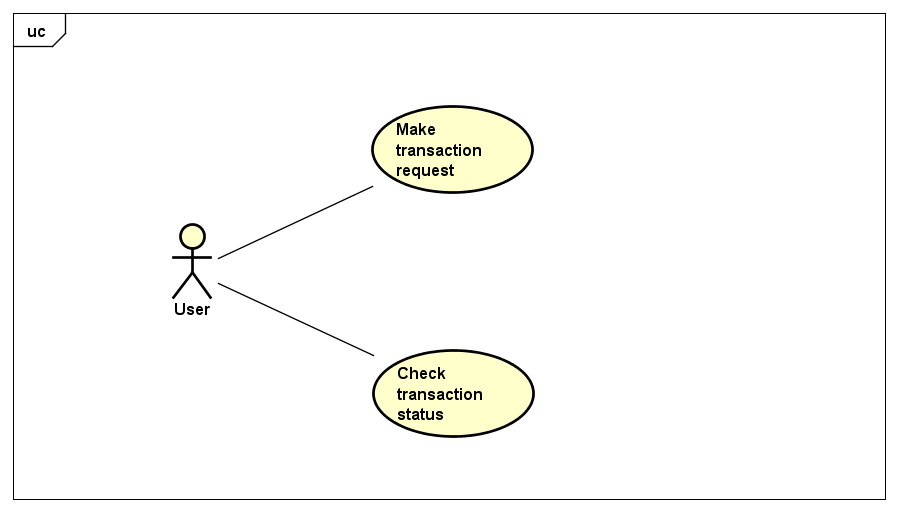
The following are the requirements adopted for this project.

The involved users in the various requirements are mainly of three types:

* **User** – A bank customer, average user that has no knowledge of what is happening behind the scene.
* **Bank Operator** – A bank employee dedicated to the project, depending on its goal.
* **Miner** – An automated system for transactions confirmation, that ensures security and validity.

## Functional Requirements

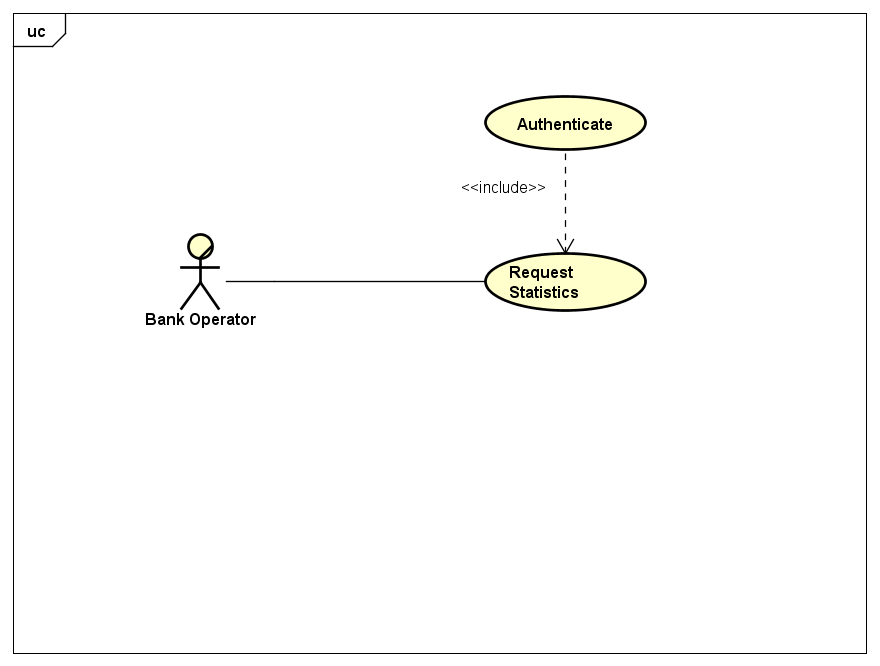
1. User must be able to interact with transactions.



A user must be able to create, show and gather information of his/her transactions or the ones he/she is receiving.

* A transaction can be created by the user:
  + The transaction must be secured before transmission to the other nodes:
    - The information must be properly encoded as the protocol dictates and must be checked for tampering.
* The transactions status can be queried any time by the user:
  + The query must contain the transaction ID as reference.
  + The request must be secured using the protocol’s specifications.
  + The response must not contain the asset value.

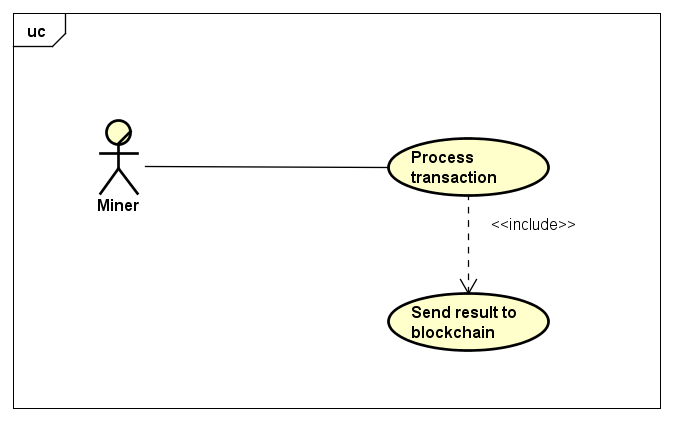
1. Blockchain statistics must be available to operator



A bank operator can query statistics of its bank node:

* The response must respect the protocol’s specifications.

1. Transaction confirm



The miner must be able to confirm the transaction and increase the confirmation status counter by one:

* The miner must be able to retrieve the latest transactions by asking to the node.
* The transaction request must be elaborated, and a verdict must be chosen.
* The verdict must be transmitted to the bank node.

## Non-Functional Requirements

* The transaction request transmission should happen within 2 seconds after compilation and confirmation of the operation.
* The resulting data from transaction status query must be transmitted to the requesting user within 10 seconds from the request.
* The response of the statistics query must happen within 10 seconds.

1. **Performance**

The Quality of Service offered by the software strictly depends on the performance of the system the bank decides to install.

* 1. Network – It is heavily encouraged to monitor bandwidth usage as a blockchain may have lots of network operations.
  2. Virtual space – The fork of a blockchain may become bigger and bigger by the time, virtual space is the main resource that should be scalable.

1. **Operating constraints**

It is required by the whole bank systems to be connected to a common network for communication to work properly preferably using VPNs, ensuring also additional security and layer of isolation.

1. **Security**

The system must use known and approved cryptography mechanisms to guarantee security of the blockchain, the current asymmetric cryptography algorithm is RSA, can be upgraded to ECDH for better security measure.  
It is also necessary for production usage a layer of security for network communication protocol.

# Design

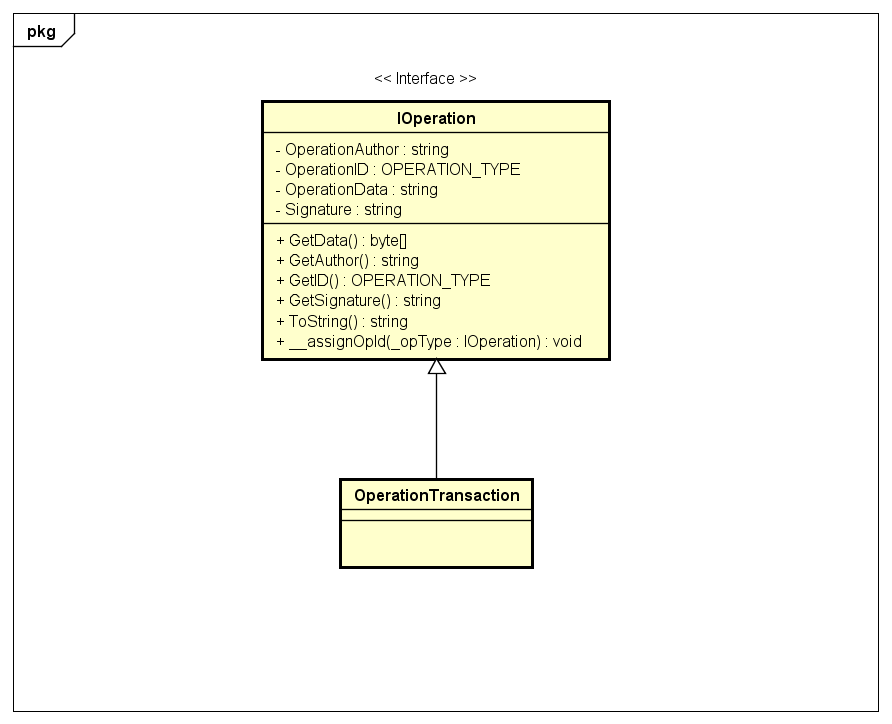
The following section describes the main components of the software, highlighting relevant information for comprehension of the subject and the mechanism of functioning.

## Entity structures

### Transaction

The transaction is the entity containing vital information for the asset transfer from a user to another, containing:

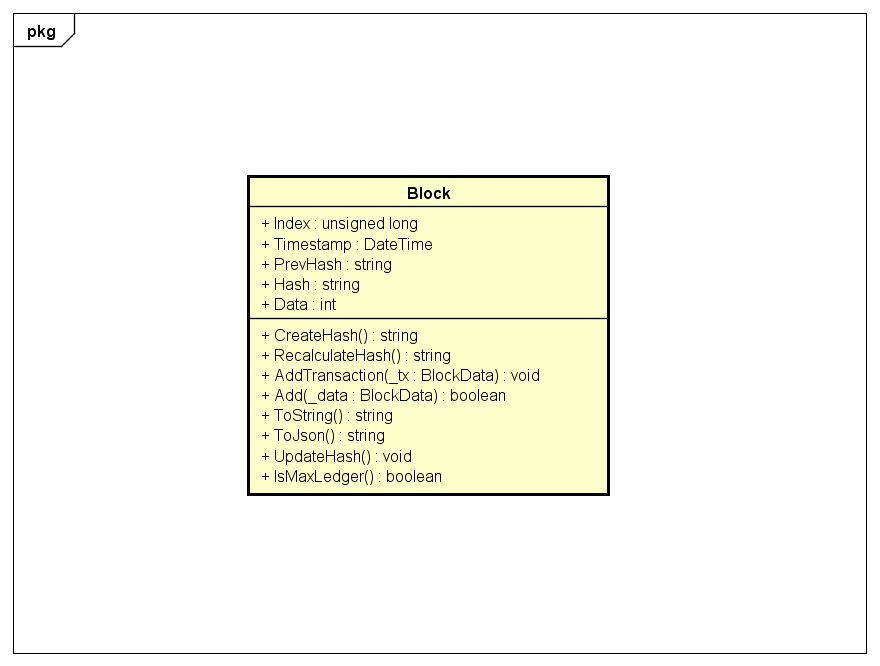
* **Sender Address** – The sender’s address (identifier).
* **Receiver Address** – The receiving user’s address (identifier).
* **Asset Value** – The value of the asset being transferred (e.g. money).



### Block

The block is the container of transactions being synchronized between multiple nodes, the block can be of two types:

* **Pending Block** – The current block being filled with confirmed transactions, still not synchronized between all the nodes and it is still editable.
* **Stored Block** – The block is stored into the blockchain and locked for modifications, it is now ready to be synchronized between multiple nodes and become part of the blockchain.

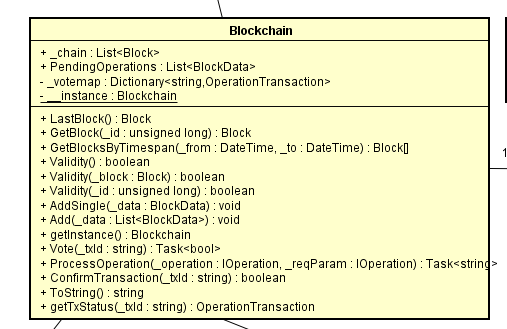


### Blockchain

The blockchain is the combination of stored blocks linked together and it’s the whole tree that composes the history of the confirmed operations inside the network.

A blockchain may be reset (wiped clear) to a completely new one for virtual space usage optimization (and archiving backups), the version of a blockchain is called “Fork”.

The blockchain is structured in the following way:



### Node

The node is the central unit a bank uses for synchronization and storage of pending transactions originated by its own customers (users).

Every bank can have one or more nodes.

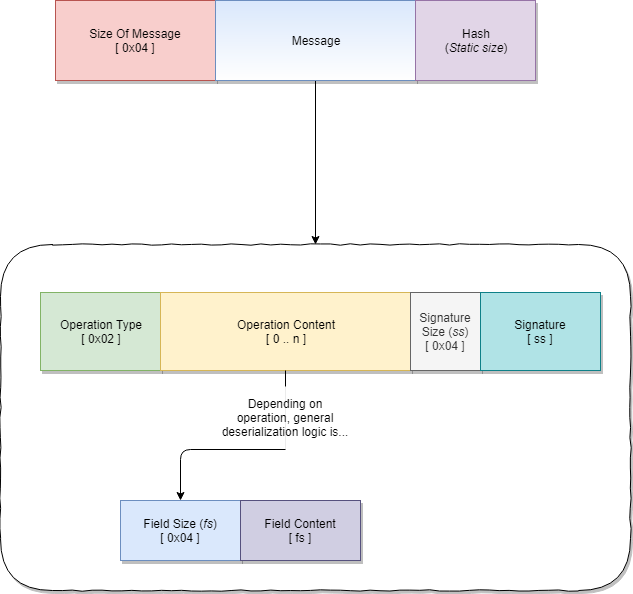
The node is structured by:

* Its own copy of the blockchain, that can be synchronized with other nodes.
* Its own customers pending transactions (awaiting miner’s confirmation).
* Transactions confirmation counters.

### Encapsulation (Crypto Layered Message)

To ensure security and obscurity of vital information, the data being sent would be properly encrypted and signed through asymmetric encryption algorithms.

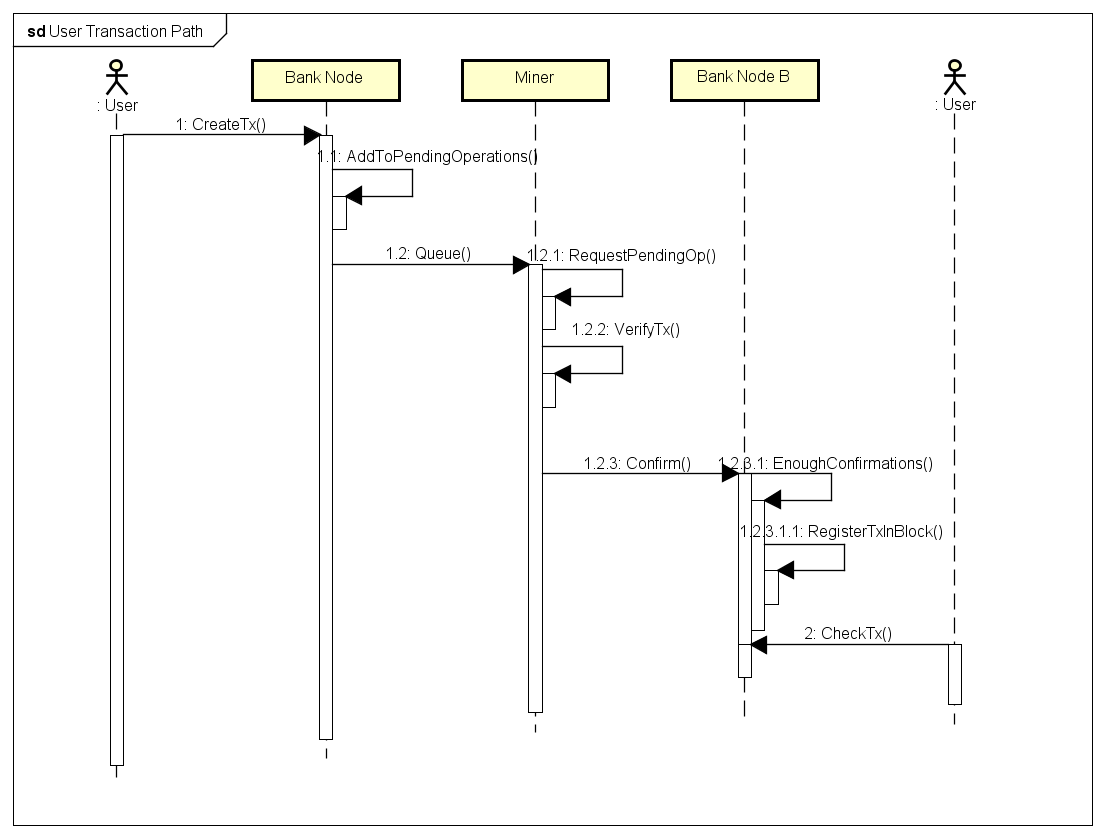
The encapsulation message is the container of the information being sent and the protocol to follow for correct requests and responses; structured in the following way:



The message’s buffer is *not* to be compressed if it contains sensible data, but only encrypted, for security reasons (refer to “CRIME” or “Compression Oracle”).

Nordic Blockchain already implements AES as block cipher for communication encryption, but the class is not being used for this version of the project.

### Transaction Request Path



A transaction follows a long path of processing before being registered permanently for its scope.

The process can be divided into 3 main phases:

1. **Acceptance Phase**

The transaction request must be created on first place, its origin must be verified to avoid fraud; once authenticated, the request is to be marked valid, encrypted and enqueued into the pending operations, awaiting miner’s notice.

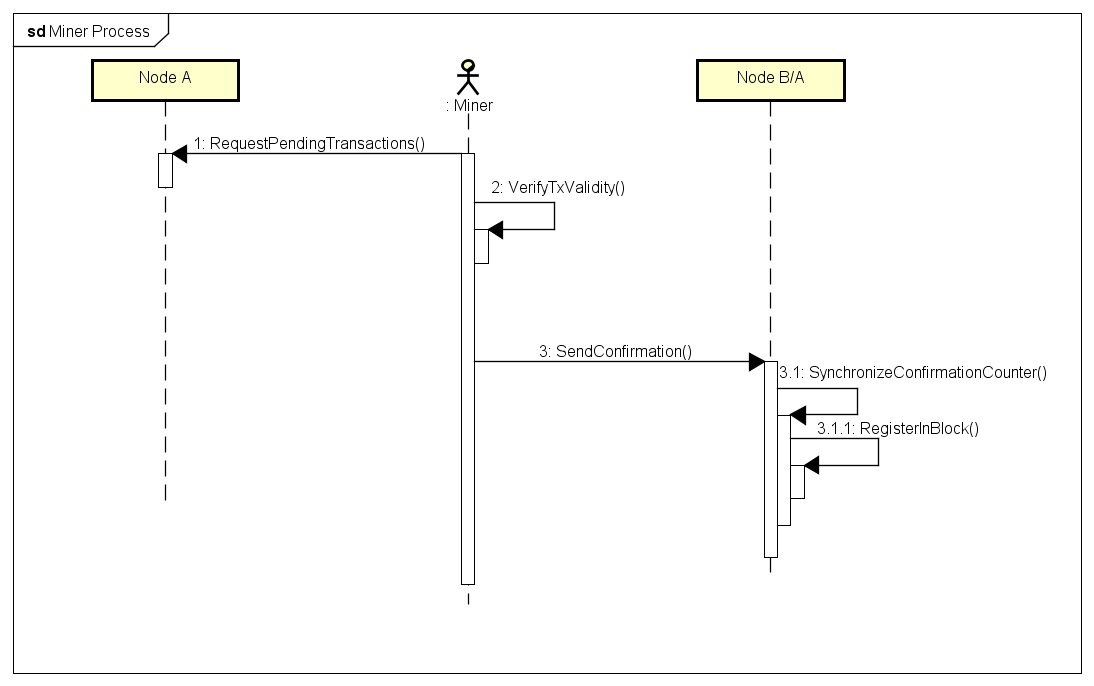
1. **Transmission Phase**

Once the transaction is retrieved from the queue by the miner, the miner has to verify certain aspects of the transaction to be marked definitively acceptable (and therefore “voting” for valid) (Refer to “Section 4.3”) by the miner and finally trace the receiver’s node or the originating one, to which the vote is transmitted.

1. **Processing Phase**

If a pending operation reached enough confirmations, the node can process its content and apply its actions permanently, registering the transaction into the blockchain’s latest block available if not full, or create a new block where the transaction is put.

### Miner Confirmation Process



The miner’s duty is to be considered fundamental for the verification process of a transaction and its actuation.

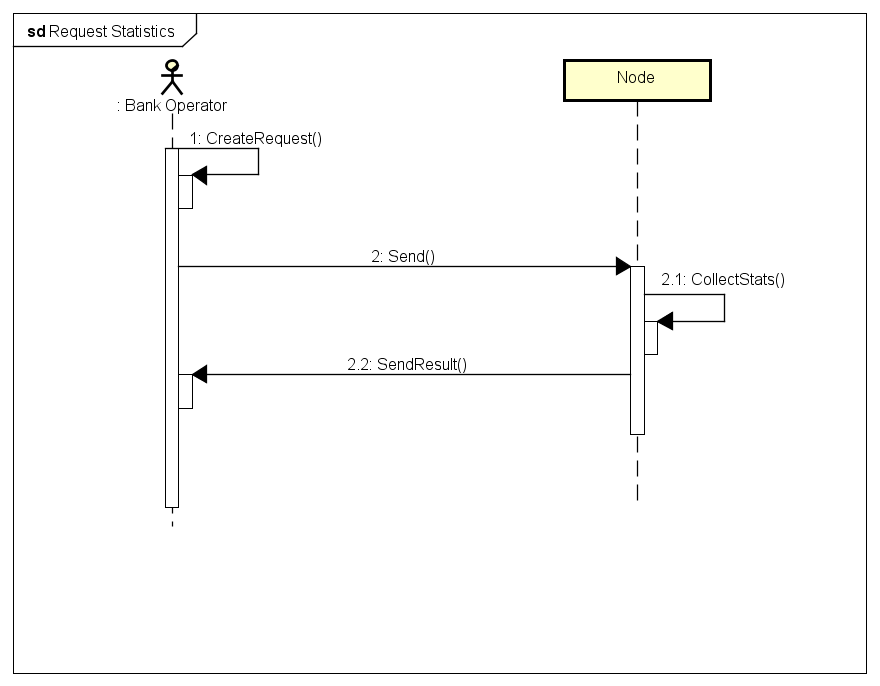
Once a transaction is retrieved by the pending operations, the transaction’s validity comes from the signatures included in the transaction request.

If the transaction signature is by the originating bank (and therefore the bank certified the origin of the transaction is by an authorized user), the miner must recover the receiver’s address and appropriate node(s) to send the vote to (in this project execution, only the one node is online).

The vote is also registered as an operation by implementing “IOperation” interface and will be registered inside the transaction “confirms”, in the block.

Whenever the transaction’s confirms reach the maximum necessary to be considered definitive, the transaction is automatically put into the latest available block and registered permanently.

### Operator Statistics Request



While the previously described operations may appear complicated, the statistics request is a simpler function used as a debug feature for the communication of the node and the content of the blockchain.

If the node recognizes authorization by the requesting’s signature, the following statistics are sent back:

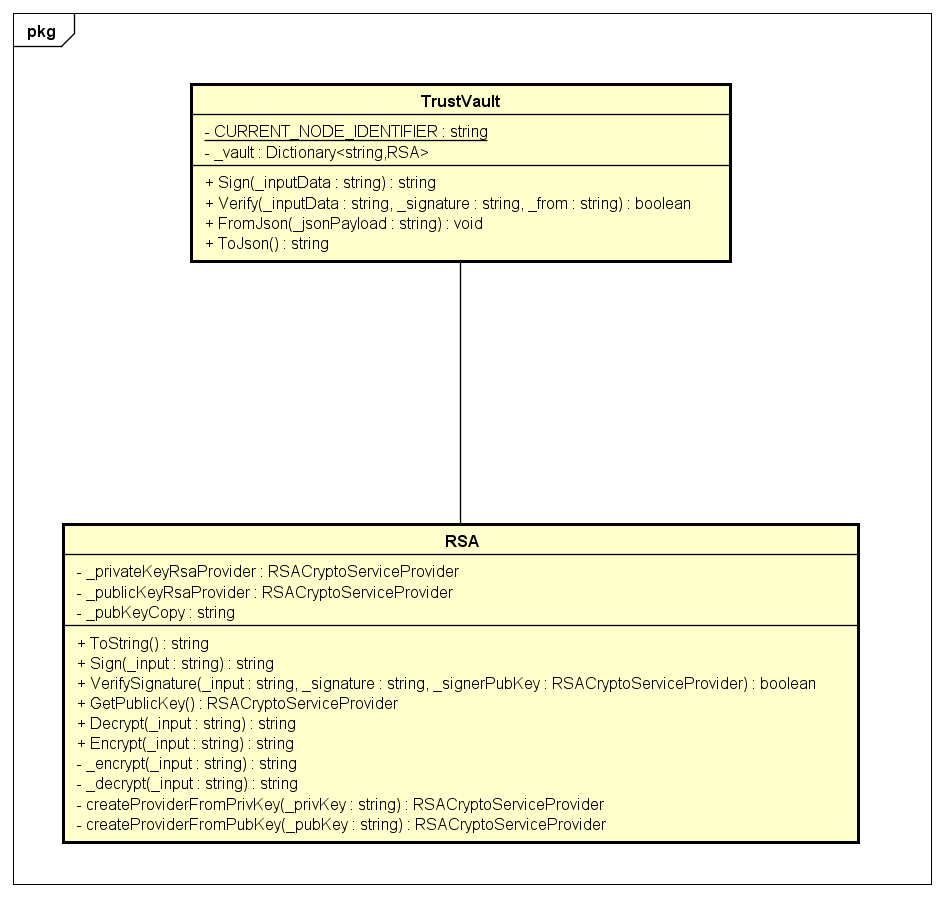
* Pending operations remaining
* Latest block information and content
  + Creation date
  + Containing transactions
  + Containing operations in general

### Genesis Block

The genesis block is the first block created in the blockchain that will start the succession of the other blocks.

Its presence is essential as it ensure the second block (first real-usage block) has the necessary security measures (Hash) coherent from the start.

In this implementation of the Nordic Blockchain, the genesis block is automatically created with non-significant content.

In the project, the genesis block is the deposit of all the public keys of the nodes with relative owner, represented by a plain Block containing a single transaction, of which content is the “Trust Vault” structure.

### Serialization

As the Crypto Layered Message (CLM) is a binary formed information exchange, a proper serialization and deserialization class is required for transmission over the network.

The “ClmManager” class offers such support using templates (Generic) for auto-determination of serialization or deserialization process to employ for the designed class structure.

### Overall architecture

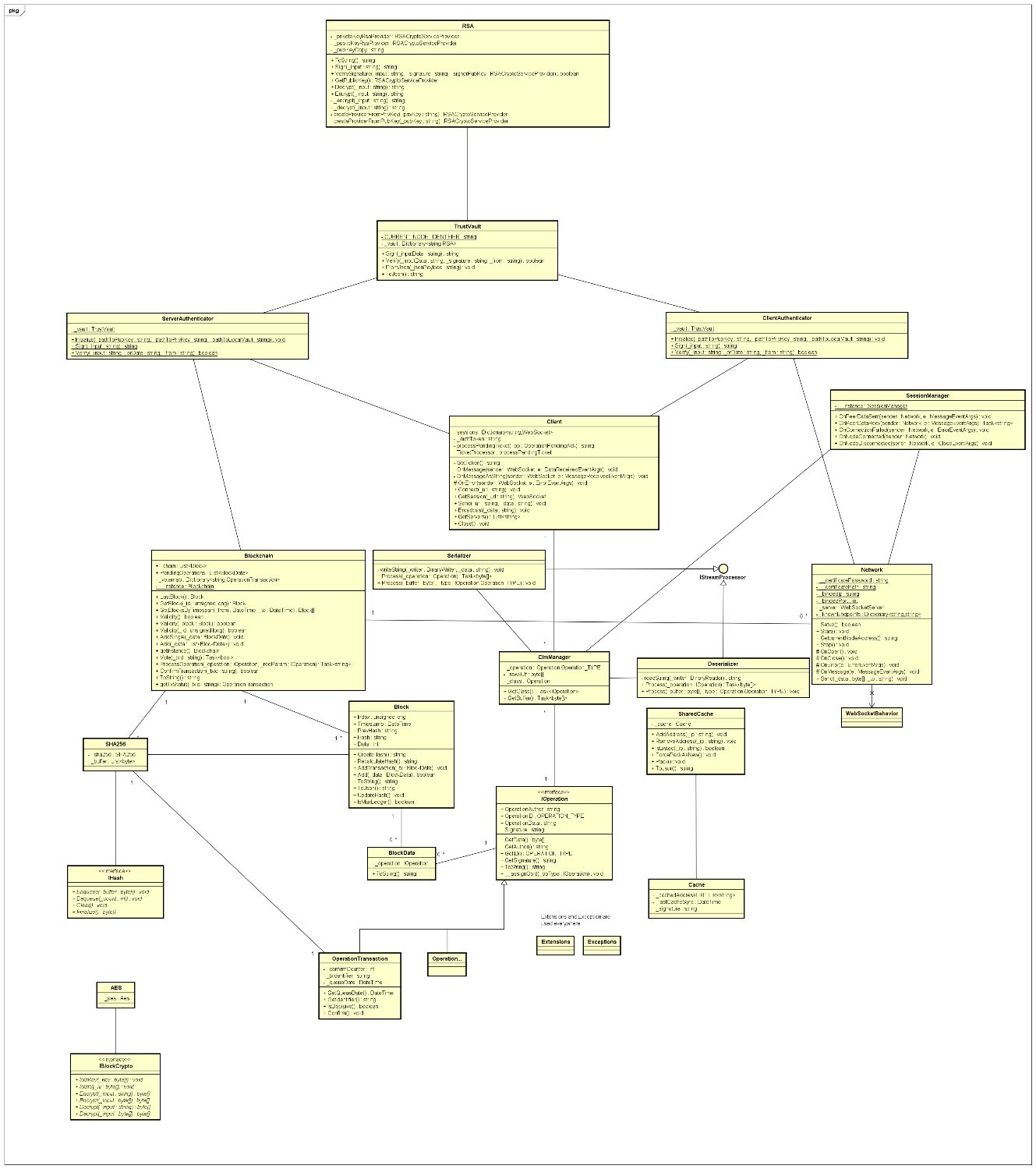
The overall architecture shows the relationship between the components of the Nordic Blockchain project, both by “packages” (contextual sections) and the full version.

#### Simplified Version

Immagine che contiene mappa, testo

Descrizione generata automaticamente

#### Full Version



## Choice of technologies

### Asymmetric Cryptography

The project and its security heavily rely on asymmetric cryptography to authenticate and encrypt the data received and sent, as well for authorization levels determining and origin verification.

While an alternative to RSA is ECDSA and is even newer technology, the choice of RSA in this project is purely for simplicity, compatibility coverage and accessibility as well as its maturity.

RSA is a well-known asymmetric cryptography algorithm, created in 1977 compared to ECDSA that has been proposed in 1992; thus, ECDSA is newer and its performance and time-complexity is supposedly better than RSA, despite that, RSA has better performance due to longer time of improvements.

While ECDSA offers better scaling capabilities and bigger key sizes, its vulnerability to Shor’s algorithm is considered weaker than RSA, that offers better resistance to quantum-based attacks, features longer maturity and a long list of adoptions in a huge variety of software, both FOSS and private.

For the above-mentioned reasons, the Nordic Blockchain’s team decided to adopt RSA for simplicity and extended documentation coverage compared to ECSA and a multitude of available APIs covered by different licenses, mainly FOSS-oriented.

### WebSocket

The communication classes (Network, Client, SessionManager) use WebSockets for communication protocol instead of plain sockets.

The choice of WebSockets is to put a baseline for compatibility with multiple applications, including external ones, that desire to use such service, since the WebSocket’s transportation protocol obeys under the HTTP standards and supports SSL.

Also, the usage of standardized protocol and technologies is always preferred over custom implementations that might deteriorate over time and decrease maintainability.

### .NET Core

The choice of .NET Core technology instead of .NET Framework is simply for simplicity of the C# language and, majorly, for cross-compilation and cross-compatibility towards different operative systems.

C# is becoming an increasingly influential programming language every year and the recent porting of the .NET technology into Linux/Unix systems is nonetheless speeding up the process of propagation, the author considered the possibility of keeping a project like this on multiple systems, considering the banks networks may have a bias towards a specific OS.

### NuGet Package Manager

The installation and usage of external libraries is possible thanks to the NuGet package manager.

The software is a one-click package installer for libraries developed by any entity (private or corporate) and it is included into Visual Studio IDE, speeding up the development times for the author of the project.

NuGet may not be suitable for certified-only production development – concerning security - and may retain further investigation for production quality and critical security based projects.

# Implementation

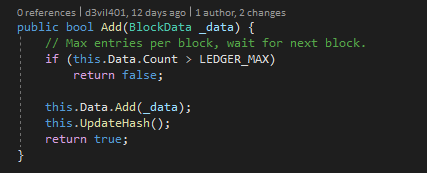
## Block

Immagine che contiene screenshot

Descrizione generata automaticamente

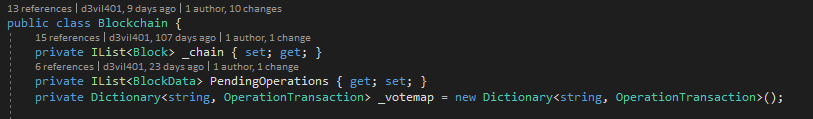
The block structure and calculation of hash.

The hash calculation is a merging of the previous block hash, the creation time and the content, which ensures the concatenation of blocks that are recursively verified for compromise verification.

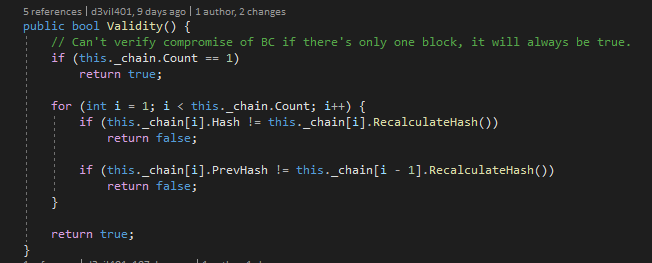


The function that registers the new data if not full and updates the current block hash whenever it changes.

## Blockchain



The blockchain contains 3 complex structures, the “\_chain” is the current chain of blocks that contains permanently registered data, the “PendingOperations” holds the list of pending operations that the miners will request, this is updated automatically by removing newly confirmed operations and only adding newly created ones, and, the “\_votemap” that holds the reference of transactions based on their Transaction Identifier, used for miner’s vote casting.



The “Validity” function is for verification of compromised of a blockchain, it compares the registered hashes between concatenated blocks and the latest hash version.

Immagine che contiene screenshot

Descrizione generata automaticamente

The “ProcessOperation” is the call-back function used by the network for operations that should interact with the blockchain or its content.

It functions as a bridge of communication between networking and blockchain manipulation that is under supervision of the developer’s decision of handling.

## Operations

Immagine che contiene testo

Descrizione generata automaticamente

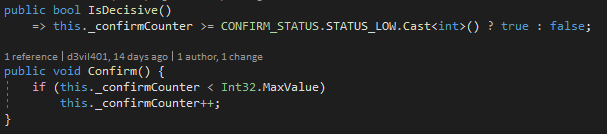
The registered IDs for data transmission, differentiated to ensure uniqueness and proper identification of the operation request.

### Transaction Operation

Immagine che contiene screenshot

Descrizione generata automaticamente

The transaction structure and assignation of unique transaction ID.



Decisive status of transaction function and vote casting function, respectively.

## Network

Immagine che contiene screenshot

Descrizione generata automaticamente

The first entry point of received data from the WebSocket, being parsed trough the “CLM Manager” for protocol handling and then processed.

Immagine che contiene screenshot

Descrizione generata automaticamente

The encapsulation processing function, verifying the tampering as well.

## Test Miner

Immagine che contiene screenshot

Descrizione generata automaticamente

The testing miner, impersonating respectively:

1. The miner role, asking and processing a pending operation.
2. The admin role, asking and printing the node’s statistics.
3. The user role, asking for the previously confirmed and registered transaction by using the already recorded TxID.

# Test

This section describes employed testing methodologies, purpose and results.

### Testing practices

To verify implementation of the functional requirements previously described and to ensure stability of the proof of concept project, the system must be tested.

The entry points in this project aren’t many, the main components that should be tested are

* Network functionalities: the WebSocket server creation and client creation, including connection, transmission and interoperability of the operations towards the network stack.
* Blockchain functionalities: the blockchain should not be exposed to intrusions and must identify when one happens, the creation and registration of blocks and pending operations, the network’s interoperation with the blockchain’s functionalities.
* Cryptographic functionalities: the used cryptography should be tested and checked for proper implementation.

Since the project is mostly back-end type, the White Box testing methodology has been chosen, by implementing and using the results of Unit Testing.

For Unit Testing, the X-Unit testing framework was the only compatible one with .NET Core that was also by default included in the installation of the .NET subsystem.

A brief description of the unit tests developed:

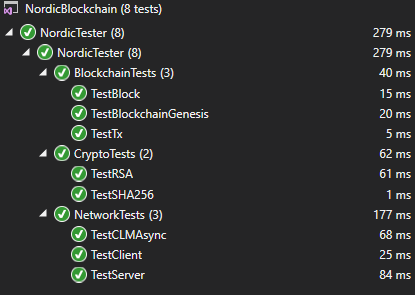
* Network functionalities:
  + Test server creation.
  + Test client creation and connection with the server, using an IOperation for transmission test.
  + Test CLM Manager, by creating a request and expecting a correct response.
* Blockchain functionalities:
  + Test blockchain creation and genesis block.
  + Test transaction requests, vote casting, confirming status and status request.
  + Test block creation, block registration, block querying and blockchain compromise.
* Cryptographic functionalities:
  + Test RSA private and public key importation, signing and signature verification.
  + Test SHA-256 creation, hash calculation and hash comparison including expected hash result comparison (hardcoded).

Testing the functional requirement 3.2 B and 3.2 C required a different approach than Unit Testing and an ad-hoc project has been created for such tasks (NordicMiner), containing the required code using the functional requirements above mentioned.

These should be verified trough debugging due to the structure of the dedicated classes (CLM Manager) since the exceptions thrown are designed not to expose too many information of traveling data.

### Test Results

The results of the test are all positive.



## Test Specifications

### Specification 1: User’s transaction placement, confirmation and verification.

The test should emulate the user’s placement of an OperationTransaction in the blockchain, verify that the OperationTransaction has been placed in the pending operations and has not been confirmed yet.

If the transaction has been put into the pending operations, cast a vote and expect it to be confirmed definitively and verify that it has been placed in the latest block.

### Specification 2: Administrator’s request of statistics.

The test must use the Client class, after a server instance has been created, to send a request for statistics and expect a response with a compressed string corresponding to the JSON representation of the blockchain’s last block and a number representing the pending operations stored in the node.

### Specification 3: Miner’s fetch of pending operation (transaction) and confirmation.

The test must use the Client class, after a server has been created and a transaction request has been placed, to request a pending operation to be processed and verify that the pending operation received is the same as the one registered (by identifier comparison).

Once the operation is received, the miner must cast a vote and a transaction status request must be sent to the server to verify that the operation has been confirmed and placed in a block.

# Results and Discussion

The development process for Nordic Blockchain concluded into a creation of a functional proof of concept prototype that satisfies the functional requirements, validating its goal.

The blockchain structure has proven to be ready for usage in a production quality environment as it is not a complex structure and has been properly tested.

Certain operations, such as the cryptographic ones, are not reliable due to the lack of time and resources allocated to the project development but are not necessary to justify the existence of the project itself.

Certain aspects of the project’s structure and design, during development, had to be hardcoded or simplified due to the lack of funding, personnel and time till the deadline.

A few discussion points are listed below:

* The blockchain structure is a solid design but may be improved over the time by adding additional dependency for validity such as double-sided hash storage (store both the previous block’s hash and the next block’s hash), albeit enforcing the Block security trough additional dependency factors.
* The miner’s purpose, on the contrary of traditional blockchain systems, is not to keep decentralization focus but to only automate the process of transactions verification; decentralization in a context of full-control and monitorization is weak.  
  Despite the limitations, the Node role changes from point of storage to authoritative broadcasting of operations.
* A strength point for eventual rogue miners is to target and confirm specific malicious transactions; the design approached here does not in fact solve or limit the 51% attack but the usage of asymmetric keys allows the nodes to only recognize and trust miners that have been previously put into the trust list (ClientAuthenticator).
* The implementation of a wallet system will allow user identification and perhaps the registration of assets directly on the blockchain.
* The CLM Manager is a very specific implementation-oriented class and requires lots of care when implementing new operation types; a better and more generic implementation can be achieved if slightly reworked.
* The SessionHandler class is not properly implemented, no unique identifier is associated with existing sessions and no server-side session control exists due to limitations of used APIs; development of custom WebSockets API will unlock another area of server-controls.
* Persistence of blockchain hasn’t been planned in this project, but the usage of JSON will speed up future implementations.
* The signing and verification trough RSA should be done over the SHA256 hash of the content, not on the content directly.
* A fundamental safer mechanism for transaction would be the usage of Smart Contracts, that would bring even a more modern approach to the original problem.

# Conclusions

In conclusion, NordicBlockchain’s development was satisfactory, despite the lack of manpower. The blockchain is proven to be a scalable and secure storage system and easy to backup, provided the stakeholder will implement its persistence.

The author had different additional ideas over the networking and synchronization functionalities that had to be sacrificed, but traces of implementation are left in the code for future development.

Further research should be made into the security side of the project and the front-end part of the project for user’s interaction.

A test-driven development should be employed rigorously due to criticality of safety nature of the project itself.

The system’s structure offered interesting results for a baseline of development for future implementation of verify-and-record mechanisms that are very common in government and administration areas.

Perhaps, stakeholders can extend the interest area of this project baseline to a multiplicity of implementation possibilities and experiments, offering limitless possibilities and provoking discussion or ideas for different purposes.

# Sources of information

Henry He, 2018. *Building A Blockchain In .NET Core*. Available at:

<https://www.c-sharpcorner.com/article/building-a-blockchain-in-net-core-proof-of-work/>

Ana Berman, 2018. *Private Blockchains Could Be Compatible with EU Privacy Rules, Research Shows*. Available at:

<https://cointelegraph.com/news/private-blockchains-could-be-compatible-with-eu-privacy-rules-research-shows>

Jake Frankenfield, 2019. *51% Attack*. Available at:

<https://www.investopedia.com/terms/1/51-attack.asp>

Ivan Ristic, 2012. *CRIME: Information Leakage Attack against SSL/TLS*. Available at:

<https://blog.qualys.com/ssllabs/2012/09/14/crime-information-leakage-attack-against-ssltls>

Software Testing Fundamentals, 2019*. Unit Testing*. Available at:

<http://softwaretestingfundamentals.com/unit-testing/>

Stack Overflow, 2019. *Most Popular Technologies*. Available at:

<https://insights.stackoverflow.com/survey/2019#technology> [Accessed 28 May, 2019].

GeeksforGeeks, 2019. *RSA Algorithm in Cryptography*. Available at:

<https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

Wikipedia, 2019. *Secure Hash Algorithms*. Available at:

<https://en.wikipedia.org/wiki/Secure_Hash_Algorithms>

Accenture, 2019*. The future of blockchain banking*. Available at:

<https://www.accenture.com/us-en/insight-blockchain-technology-how-banks-building-real-time>

IETF, 2011. *RFC 6455: The WebSocket Protocol*. Available at:

<https://tools.ietf.org/html/rfc6455>

Wikipedia, 2019. *Transport Layer Security*. Available at:

<https://it.wikipedia.org/wiki/Transport_Layer_Security>

Ameer Rosic, 2016. *Smart Contracts: The Blockchain Technology That Will Replace Lawyers*. Available at:

<https://blockgeeks.com/guides/smart-contracts/>

Wikipedia, 2019. JavaScript Object Notation. Available at:

<https://it.wikipedia.org/wiki/JavaScript_Object_Notation>

IEEE Computer Society, 2008. *IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation*.

Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*.

Wikipedia, 2019. *Kanabn (development)*. Available at:

<https://en.wikipedia.org/wiki/Kanban_(development)>

**Note: Use the standard reference method: Harvard Anglia. A very good reference tool is Mendeley** (Mendeley.com 2016), **ask VIA Library if you need help.**

# Appendices

The purpose of your appendices is to provide extra information to the expert reader. List the appendices in order of mention.

Examples of appendices

* Project Description
* User Guide
* Source code – source documentation
* Diagrams
* Data sheets
* Etc.

**Appendix A Project Description**

Insert the original Project Description here