МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

**ЗВІТ**

про проходження технологічної практики

в умовах кафедри програмних систем і технологій

Індивідуальне завдання: Software Architecture Document (SAD)

Виконав: ст. гр. ІПЗ-33

Гоша Давід

Керівник практики:

доцент Зубик Людмила Володимирівна

Київ – 2023

Contents

[1. Introduction 2](#_Toc136117654)

[1.1 Objective 2](#_Toc136117655)

[1.2. Scope of application 2](#_Toc136117656)

[1.3. Definitions, acronyms and abbreviations 3](#_Toc136117657)

[1.4 List of references 5](#_Toc136117658)

[1.5 Overview 7](#_Toc136117659)

[2. General description 8](#_Toc136117660)

[2.1 Product perspective 8](#_Toc136117661)

[2.2 Product features 9](#_Toc136117662)

[2.3 User classes and characteristics 10](#_Toc136117663)

[2.4 Operating environment 10](#_Toc136117664)

[2.5 Design and implementation constraints 11](#_Toc136117665)

[2.6 User documentation 11](#_Toc136117666)

[2.7 Assumptions and dependencies 11](#_Toc136117667)

[3. System architecture 12](#_Toc136117668)

[3.1.1 Block Structure 12](#_Toc136117669)

[3.1.2 Transaction Structure 13](#_Toc136117670)

[3.1.3 Transaction and Block Verification 13](#_Toc136117671)

[3.1.4 Blockchain Storage and Distribution 13](#_Toc136117672)

[3.1.5 Consensus Mechanism 13](#_Toc136117673)

[3.2 User Interface and Experience 13](#_Toc136117674)

[3.3 Networking and Communication 14](#_Toc136117675)

[3.4 Architectural diagram 15](#_Toc136117676)

[4 Detailed system design 16](#_Toc136117677)

[4.1 User Interface Design 16](#_Toc136117678)

[4.2 Data Structures 16](#_Toc136117679)

[4.2 Data Structures 17](#_Toc136117680)

[4.3 Structure Design 17](#_Toc136117681)

[4.4 Implementation Details 17](#_Toc136117682)

[5. Conclusions and further work 18](#_Toc136117683)

[5.1 Summary of achievements 18](#_Toc136117684)

[5.2 Lessons learnt 18](#_Toc136117685)

[5.3 Future work 19](#_Toc136117686)

# 1. Introduction

This document provides a comprehensive architectural overview of a blockchain application developed in Go. The blockchain application provides a secure and decentralized system for storing and transferring tokens with a focus on peer-to-peer transactions. It uses a hybrid consensus mechanism that combines proof-of-eternity (PoET) and proof-of-work (PoW) to ensure security and scalability.

## 1.1 Objective

This Software Architecture Document (SAD) provides an architectural plan for a blockchain application, designed for stakeholders like developers, testers, project managers, and business analysts. The application emphasizes security, performance, and scalability through a hybrid PoET and PoW consensus mechanism. This SAD acts as a reference for system understanding, evaluation, and future enhancements. It promotes clear communication, effective decision-making, and enhances the application's quality and usability. Upcoming sections will explore the system architecture, main components (blockchain core, web wallet, console interface), their interaction, the hybrid consensus mechanism, and the system's security, performance, and scalability approaches.

## 1.2. Scope of application

This Software Architecture Document (SAD) outlines the components of a Go-developed blockchain application:

1. Blockchain core: The main component ensuring secure, decentralized transactions. It uses a hybrid consensus model combining proof-of-eternity (PoET) and proof-of-work (PoW) for transaction validation and block addition.
2. Web wallet: The user interface for conducting transactions and checking token balances. It handles private keys crucial for transaction signing and user identity verification.
3. Console interface: A lower-level interaction toolset for technical stakeholders, offering direct interaction with the blockchain core.

These components create a secure, user-friendly system for decentralized token storage and transfer. The SAD describes the software architecture, compliance with GOST standards, and best practices. It serves as a guide for a wide audience, offering technical stakeholders an in-depth look at the system architecture and non-technical stakeholders an overview of the design and functionality. The architectural representation, key decisions, and how the system meets requirements will be discussed later.

## 1.3. Definitions, acronyms and abbreviations

This section provides a list and brief explanation of key terms used in this Software Architecture Document (SAD).

1. Blockchain: A digital ledger of transactions duplicated and distributed across a network of computers.
2. Kernel: The central component of a computer system controlling operation.
3. Web wallet: Allows users to manage cryptocurrencies through a web-based interface.
4. Private key: A cryptographic element that allows a user to access their cryptocurrency.
5. Consensus mechanism: A fault-tolerant mechanism achieving agreement on a single data value or network state.
6. PoET and PoW: Proof of Elapsed Time and Proof of Work, consensus algorithms in blockchain networks.
7. SAD: Software Architecture Document, a document that covers a software system's architecture.
8. GO: A programming language used to develop the blockchain core, web wallet, and console interface.
9. Hashing: A process that returns a fixed-size string of bytes from input data.
10. Nonce: A number added to a hashed or encrypted block in a blockchain that meets complexity limitations.
11. Miner: An individual or entity that verifies new transactions and adds them to the blockchain.
12. Cryptocurrency: Digital or virtual currency using cryptography for security.
13. Bitcoin: The first decentralized cryptocurrency.
14. Gossip Protocol: A process of computer peer-to-peer communication.
15. Overlay network: A computer network overlaid on another network.
16. Peer-to-peer network (P2P): A network where each computer acts as a server for others.
17. Node: In blockchain, any computer that is connected to the network.
18. Wallet: A digital tool allowing users to interact with the blockchain network.
19. CLI (Command Line Interface): A text-based user interface used to enter commands directly into a computer system.
20. GUI (Graphical User Interface): A user interface allowing users to interact with electronic devices through graphical icons and visual indicators.
21. dAPP (decentralized application): An application running on a decentralized network, avoiding a single point of failure.
22. IP (Internet Protocol): The basic communication protocol for transmitting datagrams across network boundaries.
23. TCP (Transmission Control Protocol): One of the main protocols in the Internet protocol suite, used by applications requiring guaranteed delivery.
24. P2P (Peer-to-Peer): A decentralized network architecture where individual nodes can communicate directly.
25. API (Application Programming Interface): A computing interface defining the interaction between multiple software intermediaries.
26. JSON (JavaScript Object Notation): A lightweight data exchange format.
27. These terms form the basis of the language used in this document's description and discussion of software architecture.

## 1.4 List of references

This section contains a list of the main references that were used in the development of this project, as well as sources that can provide additional in-depth knowledge of the technologies and methodologies used:

1. Nakamoto, S. (2008). "Bitcoin: a peer-to-peer electronic money system". Available at: https://bitcoin.org/bitcoin.pdf.
2. Wood, G. (2014). "Ethereum: A secure decentralized generalized ledger of transactions". Available at: https://ethereum.github.io/yellowpaper/paper.pdf.
3. Antonopoulos, A. M. (2014) (2014). "Mastering Bitcoin: Unlocking digital cryptocurrencies". O'Reilly Media, Inc.
4. "Go Programming Language." (2023). Available at: https://golang.org/doc/.
5. Buterin, V. (2013). "Next generation smart contract and decentralised application platform". Available at: https://ethereum.org/en/whitepaper/.
6. "Standard for Verification and Validation of Systems and Software - IEEE Std 1012-2016" (2016). IEEE Standards.
7. "ISO/IEC 27001-2016: Information technology - Security methods - Information security management systems - Requirements" (2016). National Standard of Ukraine
8. Montresor, A., & Jelasity, M. (2013). "PeerSim: A scalable P2P simulator". In Proceedings of the Ninth IEEE International Conference on Peer-to-Peer Computing (P2P'09). https://www.gsd.inesc-id.pt/~ler/docencia/rcs1314/papers/P2P2013\_041.pdf
9. SAFE Network (2020). "The evolution of terminology with the development of technology: Decentralised versus distributed. https://medium.com/safenetwork/evolving-terminology-with-evolved-technology-decentralized-versus-distributed-7f8b4c9eacb.
10. The Go programming language. (2023). "Go programming language playlist". https://www.youtube.com/playlist?list=PL4\_hYwCyhAvZmzpIjwewZOdBMFJooHIHx.
11. Go to the standard project layout. (2023). "Standard Go project layout". https://github.com/golang-standards/project-layout.
12. Walter, K. (2023). "Overview of consensus algorithms in blockchain". https://github.com/cedricwalter/blockchain-consensus.
13. Antonopoulos, A. M., & Dhillon, V. (2017). "Mastering bitcoin: programming an open blockchain". https://github.com/bitcoinbook/bitcoinbook.
14. Superstas. (2023). "Gcoin Mempool Implementation". https://github.com/superstas/gcoin/blob/master/gcoin/mempool/mempool.go.
15. Kiayias, A., Russell, A., David, B., & Oliynykov, R. (2017). "Ouroboros: A trusted secure blockchain protocol with proof of stake". https://pdfs.semanticscholar.org/7dce/801b2b13001d0d3b0319c550ee1977e456df.pdf
16. Liao, K., Katz, J., & Zikas, V. (2018). "BFT protocols under fire". https://arxiv.org/pdf/1801.07447.pdf.
17. Berini, M. (2015). "Development and implementation of a bitcoin wallet application".https://openaccess.uoc.edu/bitstream/10609/45861/6/mberiniTFM1215memoria.pdf.

These references cover both practical aspects of blockchain technology and the Go programming language, as well as the theoretical foundations and standards of cryptographic systems and information security management systems.

## 1.5 Overview

This document presents the technical specifications, design principles, and architectural blueprint for the proposed blockchain system. It includes:

1. General Description: Provides a high-level overview of the system, including functions, user classes, environment, and constraints.
2. System Architecture: Explores the technological core of the system, detailing the structural components, user interface, network protocols, and blockchain interactions.
3. Detailed System Design: Discusses specifics of system components, user interface design, data structures, class design, and critical implementation details.
4. Implementation and Testing: Describes the execution process, testing procedures, challenges faced, solutions, and quality measures.
5. Conclusions and Future Work: Reviews the project's accomplishments, lessons learned, and areas for improvement.

This document serves as a detailed guide for technical and non-technical stakeholders.

# 2. General description

## 2.1 Product perspective

The blockchain application we're developing is a standalone system designed to provide a secure, decentralized platform for the peer-to-peer transfer of tokens. It comprises three main components: the blockchain core, a web wallet, and a console interface.

The blockchain core is the foundation of the system, managing transactions, blocks, and consensus algorithms.

The web wallet provides a user-friendly interface, facilitating the secure storage and management of tokens.

The console interface is a command line interface (CLI) that simplifies user interaction with the blockchain, enabling token transfer and balance checking.

The system employs a hybrid consensus mechanism combining Proof of Elapsed Time (PoET) and Proof of Work (PoW), ensuring secure, efficient transaction validation. This system is intended for a wide range of users and has no direct analogues, being a unique blend of blockchain core, web wallet, and console interface functionalities.

# 2.2 Product features

Key features of the system include:

* **Blockchain core**: This component manages the fundamental functionalities of the blockchain, ensuring the integrity and security of the decentralized ledger.
* **Web wallet**: A secure environment for token storage and management. Users can check balance, transfer tokens, and upload private keys for enhanced security.
* **Console interface**: This CLI allows users to interact easily with the blockchain system, facilitating operations like transaction creation and balance checking.
* **Hybrid PoET and PoW consensus**: This unique approach ensures fair and efficient transaction validation, enhancing system security, performance, and scalability.
* **Security features**: Compliant with GTSU R standards, the system prioritizes security to protect user tokens and transactions.
* **Scalability and performance**: The system, developed with Go, is capable of handling a significant number of transactions and users while maintaining fast processing speed.
* **Customizability**: The system enables users to create their own tokens, broadening its potential use cases.

The system, therefore, provides a comprehensive, secure, and user-friendly platform for peer-to-peer token storage and transfer.

# 2.3 User classes and characteristics

The Go app is designed for two main user categories:

1. End users: Ranging from beginners to advanced users, these individuals interact with the blockchain primarily via the web wallet and console interface.
2. Developers and administrators: Interacting on a technical level with the blockchain core and source code, these users have in-depth knowledge of blockchain technology.
3. Miners: These users provide computational power to verify and add transactions to the blockchain, contributing to the security and reliability of the network.
4. Novice users: Majority of end users, they interact primarily with the web wallet to manage their cryptocurrency assets.

# 2.4 Operating environment

The blockchain core, web wallet, console interface, and mining nodes operate on modest hardware requirements. The network is designed to handle approximately two transactions every two minutes to balance user needs and avoid network overload.

# 2.5 Design and implementation constraints

Constraints include programming language limitations, compliance with legal and regulatory requirements, platform restrictions, encryption standards, and implementation of industry best practices.

# 2.6 User documentation

User documentation is divided into end user and developer/administrator documentation. For end users, a comprehensive User Manual, an online help system, a FAQ section, and video tutorials are provided. For developers and administrators, a detailed Developer Guide, API documentation, and Administrator's Guide are available. All documentation is updated regularly to match system updates.

# 2.7 Assumptions and dependencies

**Assumptions:**

1. User knowledge: Users have basic web application skills, and miners possess technical blockchain knowledge.
2. Internet access: Users have reliable, high-speed internet for real-time updates of transactions and blocks.
3. Regulatory environment: The application adheres to Ukrainian laws and regulations related to blockchain and cryptocurrencies.
4. Maintenance and support: Continuous maintenance and support for the application are expected.

**Dependencies:**

1. Go programming language: The application's functionality and development depend on Go's continued support.
2. Web technologies: HTML, SASS, and JavaScript updates can impact the web wallet.
3. PoET consensus mechanism: Changes to PoET may affect the operation of the blockchain.
4. TEE (Trusted Execution Environment): Application's performance is tied to TEE technology for secure transaction processing.
5. Network infrastructure: Reliable network infrastructure is crucial for connecting miners and nodes for transaction verification and block creation.

# 3. System architecture

This public blockchain application developed in Golang maintains an immutable transaction record on the network. This section highlights the main components like block structure, transaction structure, verification mechanisms, storage, and consensus mechanisms.

# 3.1.1 Block Structure

Each block includes fields like CurrHash (hash of current block), PrevHash (hash of previous block), Nonce (unique number for mining), Difficulty (mining complexity), Miner (public key of miner), Signature (digital signature for block integrity), TimeStamp (time of block addition), Transactions (array of transactions), and Mapping (tracks all transactions).

# 3.1.2 Transaction Structure

Transactions, which drive blockchain actions, include RandBytes (random bytes for entropy), PrevBlock (hash of previous block), Sender (public key of sender), Reciver (public key of recipient), Sum (amount of transferred cryptocurrency), ToStorage (amount transferred to storage), CurrHash (hash of current transaction), and Sign (digital signature for transaction integrity).

# 3.1.3 Transaction and Block Verification

Security and integrity are maintained through the IsValid() function (validates transactions) and the IsBlockValid() function (validates blocks).

# 3.1.4 Blockchain Storage and Distribution

The blockchain data is stored via a SQLite database, each block serving as a record, enabling efficient storage, retrieval, and database replication between nodes for decentralization.

# 3.1.5 Consensus Mechanism

A hybrid model combining proof-of-elapsed time (PoET) and proof-of-work (PoW) ensures fairness in a decentralized environment and maintains system security.

Overall, the blockchain application structure ensures transaction security and integrity, fitting various applications like cryptocurrencies and decentralized applications (dApps).

# 3.2 User Interface and Experience

The blockchain application has both a GUI and CLI. The GUI, based on web technologies, offers an intuitive, user-friendly interface with a web wallet for browsing the blockchain, initiating transactions, and viewing transaction history. Key features include a Home page, Login/Registration, Wallet page, Explorer page, and secure Logout. For advanced users, a CLI offers additional control.

Feedback is given through notifications, clear error messages when problems occur, and robust security measures protect users' private keys and safely terminate user sessions. The interface balances usability and functionality for all user levels.

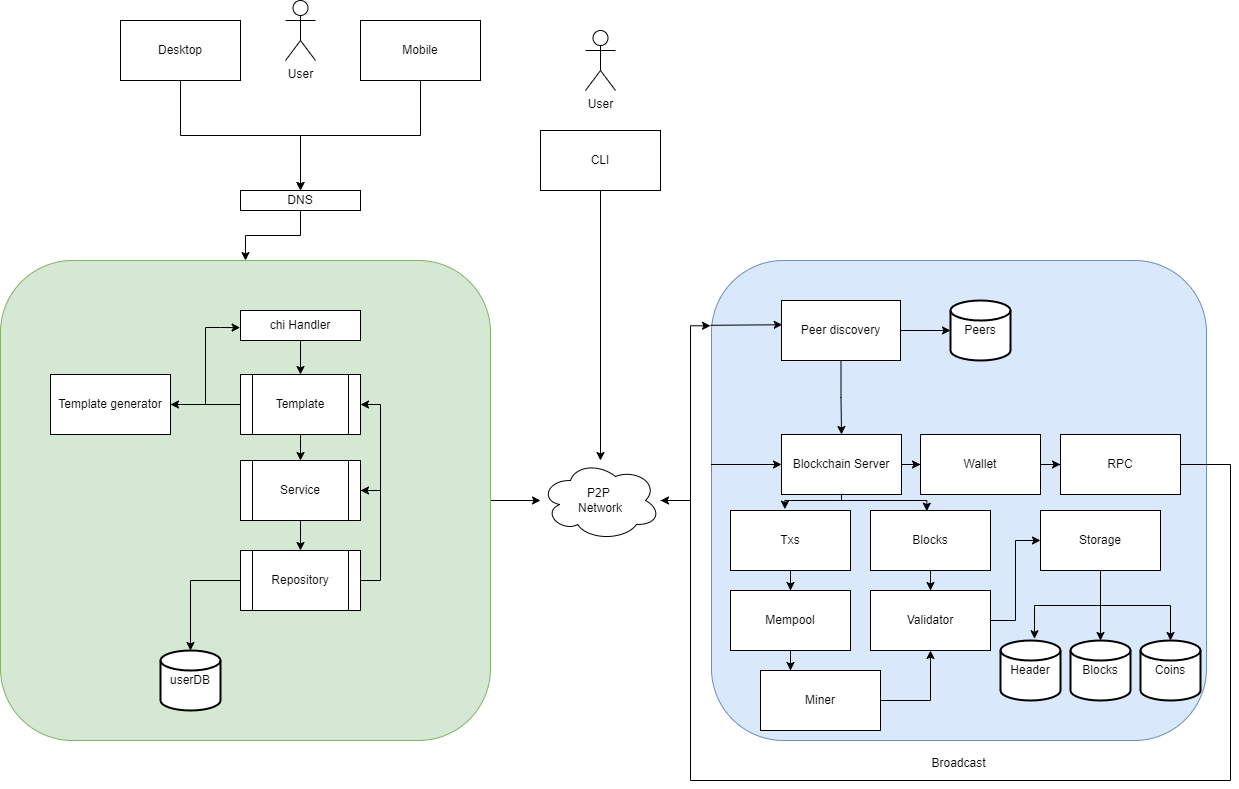
# 3.3 Networking and Communication

This blockchain application's hybrid network architecture combines client-server and peer-to-peer traits. The network communication flow involves clients and nodes interacting for balance and block information, and to record transactions. Nodes request to add new blocks to the blockchain and request specific mining ranges or the current time state.

The handleServer function allows communication between nodes using TCP protocol, offering decentralization, high fault tolerance, and resistance to network partitioning. Transactions are distributed across the network using the makeTransaction function, and new blocks are distributed with the pushBlockToNet function. The "longest chain wins" rule maintains consensus in the network

Nodes require a file with IP addresses of trusted nodes, and the encryption standards used ensure secure communication between nodes. The presence of nodes accessing pool and time servers suggest support for a collaborative mining strategy and synchronized time on all nodes, boosting reliability and accuracy. This architecture provides a decentralized, resilient, and secure blockchain network enabling smooth, reliable transactions.

## 3.4 Architectural diagram



Architectural diagram

# 4 Detailed system design

# 4.1 User Interface Design

The application's user interface is designed to be intuitive for all users. Key elements include:

1. Landing page: Provides an overview and prompts for registration or login.
2. User dashboard: Displays balance, transaction history, and allows for transaction initiation and blockchain exploration.
3. Transaction process: Users input the recipient's address and transfer amount, with the system confirming these details.
4. Blockchain data: Presented transparently, showing transaction details for each block.
5. Registration and login: Simple and secure pages where users provide details to register and login.

# 4.2 Data Structures

The system's data structures, based on Go, mimic typical blockchain structures:

1. Blockchain: A chain of blocks, each containing a list of transactions and linked by storing the previous block's hash.
2. Block: Contains a list of transactions, a timestamp, previous block's hash, and its own hash.
3. Transaction: A cryptocurrency transfer, containing sender and recipient addresses, transfer amount, and timestamp.
4. User: A network member with unique credentials and a wallet.
5. MemPool: A collection of transmitted, yet unconfirmed, transactions.

The blockchain is stored on network nodes, not in the wallet, with the wallet interacting with it.

# 4.2 Data Structures

The system's data structures, based in Go, include:

1. Blockchain: A chain of blocks, each housing a list of transactions. Blocks are linked by their hashes.
2. Block: Contains transactions, a timestamp, the previous block's hash, and its own hash.
3. Transaction: Represents a cryptocurrency transfer, containing sender and recipient addresses, transfer amount, and timestamp.
4. User: A network member with unique credentials and a wallet.
5. MemPool: A collection of transactions that are not yet included in the block.

The blockchain interacts with the wallet, but is stored on network nodes.

# 4.3 Structure Design

Key structures include BlockChain, Block, Transaction, User, MemPool, and Package. They represent the fundamental components of the blockchain and provide methods for operations such as adding transactions and verifying blocks.

# 4.4 Implementation Details

The application, implemented in Go, uses standard Go library along with third-party libraries like Chi, Logrus, and SQLite. It features a multi-level architecture that promotes maintainability and scalability. Error handling is in place to manage issues during transactions or network communications. Security is ensured through strong encryption methods.

# 5. Conclusions and further work

## 5.1 Summary of achievements

The development and completion of this blockchain-based cryptocurrency system is a significant achievement. We have successfully implemented a system that includes the basic functions of a cryptocurrency network, including the ability to conduct transactions, mine new blocks and maintain a distributed ledger. The hybrid architecture of the system, which combines peer-to-peer and client-server network models, ensures reliable and efficient communication between different network nodes.

In addition, a notable achievement was the creation of two types of wallets - a web wallet for new users and a command line interface (CLI) for advanced users and miners. This dual approach makes the system accessible to a wide range of users, while offering advanced tools for more experienced users.

## 5.2 Lessons learnt

During the project implementation, we faced numerous challenges and gained invaluable experience. Implementing a peer-to-peer network for blockchain distribution and transaction verification was a significant experience. In addition, securing the keys in the database and ensuring the security of transactions was a complex task that required strict attention to detail.

The development process also highlighted the importance of thorough testing and debugging. Given the complexity of blockchain technology, thorough testing was vital to identify and correct potential issues. If the project were to be started again, paying more attention to the architecture and design of the system would help to anticipate and avoid potential problems.

## 5.3 Future work

While the current state of the project is working and meets its original goals, there is still room for expansion and improvement. Future work may include integrating more advanced cryptographic methods to increase the security of the system and implementing more user-friendly features in the web wallet interface to improve the user experience.

In addition, scaling the network to support more transactions and users is a potential area of future work. Research into more efficient consensus algorithms could also be conducted to reduce the computing power required for mining and make the system more energy efficient.

In addition, creating an API for third-party applications to interact with the system could be a useful feature to consider in the future. This would allow for the development of additional services and applications around the cryptocurrency system, thereby expanding its potential uses.