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:
д.т.н. с.н.с Порєв Геннадій Володимирович

Київ – 2023

ЗМІСТ

Taras Shevchenko National University of Kyiv

Blockchain-based peer-to-peer network for an automated payment system

Software Architecture Document (SAD)

CONTENT OWNER: Hosha David

Table of Contents

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

The purpose of a Software Architecture Document (SAD) is to provide a detailed architectural plan for a blockchain application. This document is intended for a wide range of stakeholders, including developers, testers, project managers, and business analysts, to help them understand the architectural design and functionality of the system.

The blockchain application meets key system requirements such as security, performance and scalability. Priority is given to secure storage and transfer of tokens, as well as high performance and scalability through a hybrid PoET and PoW consensus mechanism.

This document serves as a reference for maintaining and further developing the system. It can be used to understand the system structure, evaluate architectural trade-offs, verify that the architecture meets the system requirements, and plan future enhancements.

The SAD also aims to ensure that the application architecture is designed and documented in a clear, concise and consistent manner that facilitates better communication between stakeholders, eases decision-making processes and improves the overall quality and usability of the application.

In the following sections, we will discuss the system architecture, including the main components of the system (blockchain core, web wallet and console interface) and how they interact, the use of a hybrid PoET and PoW consensus mechanism, and the system's approach to security, performance and scalability

1.2. Scope of application

This Software Architecture Document (SAD) covers the three main components of a blockchain application developed with Go. These include.

1. **The blockchain core:** This component is the heart of the system. The core is the underlying blockchain infrastructure that enables secure, decentralized transactions. It uses a hybrid consensus model that combines proof-of-eternity (PoET) and proof-of-work (PoW) to validate transactions and add blocks, offering a balance between security, performance and scalability. The blockchain core was designed and implemented in accordance with GOST standards to ensure reliability and resilience.
2. **Web wallet:** This component provides users with an interface to interact with the blockchain. It allows users to conduct transactions and check the balance of their tokens. The key feature of the web wallet is the work with private keys. After a user registers, the system downloads a private key that is linked to their account. This key is crucial for signing transactions and verifying the user's identity. It is important to note that the private key management mechanism strictly adheres to cybersecurity best practices and relevant GOST standards to maintain the highest level of security.

3. **Console interface:** The console interface is a lower-level interaction system designed primarily for developers, testers, and other technical stakeholders. It provides a set of command line tools for direct interaction with the blockchain core, offering more detailed control and feedback than a web wallet. This makes it an invaluable tool for debugging, testing, monitoring, and optimizing the system.

These components work together to create a secure, user-friendly system for peer-to-peer decentralized token storage and transfer. This SAD provides a comprehensive description of the software architecture of these components, explaining how they meet the project requirements while adhering to relevant GOST standards and software development best practices.

This document is intended for a wide audience, including but not limited to software developers, testers, project managers, and business analysts. It is structured to be accessible to both technical stakeholders who can drill down into the fine details of the system architecture and non-technical stakeholders who can gain insight into the overall design and functionality of the system.

In the following sections of this RFI, the architectural representation of the system, important architectural decisions and the rationale for these decisions will be discussed. It will also show how the system meets performance, security and scalability requirements. As part of our commitment to transparency and credibility, this document will use reputable academic and industry sources.

1.3. Definitions, acronyms and abbreviations

To facilitate comprehension, this section delivers an overview and elucidation of the definitions, acronyms, and abbreviations employed within this Software

Architecture Document (SAD). These terminologies and abbreviations serve as the foundation for the language adopted in the subsequent sections of this document, aiding the description and exploration of software architecture.

This clarification has been positioned at the conclusion of the paper to provide a ready reference, enabling clear understanding of the specialist terms used throughout the document.

1.5 Overview

The following sections of this document systematically outline the technical and functional specifications, design principles, architectural plan, and other important aspects of the proposed blockchain-based system.

The General Description section provides a general summary of the system, outlining its main functions, key user classes and their characteristics, operating environment, design and implementation constraints, user documentation, and key dependencies.

"System Architecture delves into the technological basis of the system, looking at the blockchain's structural components, user interface design, network protocols, and various blockchain interactions.

"Detailed System Design describes the specifics of the system components, looking in detail at user interface design, data structures and their interactions, class design, and specific implementation details that are critical to actually building the system.

The Implementation and Testing section describes the process of bringing the project to life, providing insight into the implementation strategy, testing procedures, problems encountered and their solutions, and the measures taken to ensure quality and code review.

Finally, the Conclusions and Future Work section discusses the project's achievements, lessons learnt and opportunities for further improvement. This forward-looking section emphasizes the iterative nature of the software development process and highlights the potential for further improvements and updates in the future.

This document aims to serve as a comprehensive guide to the project, offering information for both technical (developers, testers) and non-technical (project managers, business analysts) 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 Operating Environment section describes the necessary hardware and software environments in which the software product operates. Here is a general explanation of what the operating environment might look like for your GO blockchain application:

Hardware:

- Server Side: The application server may run on a modern server-grade machine, with a multicore processor and a generous amount of RAM to handle multiple concurrent requests efficiently. The actual hardware requirements will depend on the anticipated transaction load and the size of the blockchain. Given that blockchains can be quite large, significant storage space will be required.
- Client Side: The client-side application, specifically the web wallet, should be accessible on any device with web access. This includes desktop computers, laptops, tablets, and smartphones.

Software:

- Server Side: The server side of the application is written in the Go language, so the server would need to have a compatible Go runtime installed. If the server is using any database for storing data off-chain, an appropriate database management system would be needed. As for the operating system, it could be a Unix/Linux-based system which is commonly used for servers due to their stability and security features.

- Client Side: The client-side application is web-based, so it can be accessed through any modern web browser (like Google Chrome, Mozilla Firefox, Safari, etc.) without any additional software requirements. The JavaScript runtime integrated into these browsers would handle the execution of any client-side scripts.

Network:

- As a decentralized application, peers will need to be interconnected. This requires a stable internet connection. The specifics of the networking requirements, like bandwidth and latency, would depend on the size and frequency of the transactions.

Please note that the above description is quite generic and the actual requirements could vary depending on specifics of your application like its scale, the number of concurrent users it needs to support, the size and rate of growth of the blockchain, etc.

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

The system under study is a public blockchain application developed in Golang that implements a blockchain structure that keeps an immutable record of all transactions that occur on the network. This section describes the structure and key components of a blockchain application, with a focus on block structure, transaction structure, block and transaction verification mechanisms, storage, and consensus mechanisms.

3.1.1 Block structure

The basic component of a blockchain application is a block. A block serves as the fundamental unit of the blockchain, containing a record of multiple transactions, and is linked to other blocks to form a chain-like structure.

A block in the application consists of several fields. The structure of each block is defined in the Block struct block, which includes the following:

- CurrHash: Stores the hash of the current block.
- PrevHash: Contains the hash of the previous block, which binds the blocks together to form the blockchain.
- Nonce: A unique number used in the mining process.
- Difficulty: Indicates the complexity of the mining problem.
- Miner: Contains the public key or identifier of the miner who added the block to the chain.
- Signature: Contains a digital signature to ensure the integrity of the block.
- TimeStamp: Stores the time when the block was added to the chain.
- Transactions: An array of transactions contained in a block.

- Mapping: A map that tracks all transactions, such as how much cryptocurrency was transferred from one address to another.

3.1.2 Transaction structure

Transactions are the driving force behind the blockchain as they represent actions that take place on the network. The structure of each transaction is defined in the Transaction Structure, which includes:

- RandBytes: Random bytes for entropy.
- PrevBlock: The hash of the previous block.
- Sender: The public key of the transaction sender.
- Reciver: The public key of the recipient of the transaction.
- Sum: The amount of the cryptocurrency transfer.
- ToStorage: The amount of cryptocurrency transferred to the storage.
- CurrHash: The hash of the current transaction.
- Sign: A digital signature to confirm the integrity of the transaction.

3.1.3 Checking transactions and blocks

Verification mechanisms are an integral part of maintaining the security and integrity of the blockchain. The system uses special functions to verify transactions and blocks:

- IsValid(): This function validates transactions by checking the transaction hash and the sender's digital signature.
- IsBlockValid(): This function validates blocks by checking various elements such as hash, signature, proof, timestamp, and transaction validity.

3.1.4 Blockchain storage and distribution

Blockchain data is stored using a SQLite database, where each block is stored as a record. This method ensures efficient storage and retrieval of blocks, allowing for easy replication of the database between different nodes, thus providing decentralization and resistance to data loss.

3.1.5 Consensus mechanism

The consensus mechanism used in the application is a hybrid model that combines proof-of-elapsed time (PoET) and proof-of-work (PoW). This mechanism ensures fairness by maintaining a decentralized environment where each participating node has a fair opportunity to mine a block while maintaining system security.

The structure and components of this blockchain application provide a reliable, decentralized system, guaranteeing the security and integrity of transactions. Its design makes it suitable for a variety of applications, including cryptocurrencies and decentralized applications (dApps), offering a promising prospect for future research and development in blockchain technology.

3.2 User interface and experience

A blockchain application has both a graphical user interface (GUI) and a command line interface (CLI) to cater to a diverse range of users. The GUI is mainly based on web technologies, which makes it accessible to users with different levels of technical expertise. It is designed to be intuitive and user-friendly, especially for those who are new to the world of cryptocurrencies.

The GUI provides a web wallet that allows users to browse the blockchain, initiate transactions, and view their transaction history. There are several features that enhance the user experience, including:

1. Home page: Users are greeted with an introductory homepage that offers basic functionality and directs them to further features.
2. Login/Registration: For enhanced security and personalization, users are required to log in to access their wallets. There is also a registration option for new users.
3. The wallet page: This is where users manage their funds. They can view their balance, initiate new transactions, and view their transaction history.
4. Explorer page: This page allows users to explore the blockchain, including viewing all blocks and transactions.
5. Logout: Users can log out of their wallet securely, ensuring that their information remains safe.

In addition to the web wallet, there is a CLI designed for more advanced users and miners. This allows more experienced users to interact with the system at a lower level, offering them additional control and options.

In terms of feedback, the app is designed to keep users informed of their actions. Notifications are sent when transactions are initiated, confirmed or completed. Additionally, the system offers clear and informative error messages when problems occur to help users resolve the issue.

The application places a high priority on security. Users' private keys are encrypted and securely stored in the database, which protects them from unauthorised access. In addition, the application includes a well-structured logout mechanism to ensure that user sessions are safely terminated.

In summary, this blockchain app strikes a balance between usability and functionality. It provides a comprehensive yet easy-to-navigate interface for beginners, while offering in-depth controls that more advanced users may need. The app is dedicated to providing a positive user experience, which is achieved through thoughtful design, clear communication and robust security measures.

3.3 Networking and communication

This blockchain application uses a hybrid network architecture that combines both client-server and peer-to-peer characteristics.

The network communication flow is as follows:

- Client -> Address server (via broadcast)
- Node -> Address server (via broadcast)
- Client -> Node (peer-to-peer)
- Node -> Node (via peer-to-peer network)
- Node -> Pool server (via a peer-to-peer network)
- Node -> Time server (via peer-to-peer network)

Client-server interactions mainly revolve around clients interacting with nodes to obtain balance, block information, or to record a transaction in a block. Nodes send requests to other nodes in the peer-to-peer network to add a new block to the blockchain, and can also request a specific mining range from a pool server or request the current time state from a time server.

Communication between nodes is carried out using the handleServer function, which listens for various incoming requests (ADD_BLOCK, ADD_TRNSX, GET_BLOCK, GET_LHASH, GET_BLNCE, WAKEUP_MSG) and responds accordingly. The nodes communicate with each other using the TCP protocol, with each node acting as a client (initiating the connection) and as a server (receiving the connection). Such peer-to-peer communication provides decentralization, high fault tolerance and resistance to network partitioning.

Transactions are distributed across the network using the makeTransaction function in the client, where each transaction is sent to all connected nodes. The nodes then add the transaction to their memory pool (the pool of transaction data waiting for confirmation) using the handleTransaction function.

When a new block is mined, it is distributed across the network using the pushBlockToNet function. To resolve conflicts and maintain consensus in the network, nodes follow the "longest chain wins" rule.

To establish connections, nodes must have a file containing the IP addresses of trusted nodes. The protocol does not offer automatic node discovery.

A blockchain application follows the same encryption standards and security measures used in Bitcoin for communication between nodes. Private keys are encrypted for security and stored in a database.

It is worth noting that the presence of nodes that access the pool server and the time server indicates that this blockchain application supports a collaborative mining strategy and depends on synchronized time on all nodes, which increases its reliability and accuracy.

In general, this architecture results in a highly decentralized, resilient and secure blockchain network that allows for smooth, transparent and reliable transactions.

3.4 Architectural diagram

This system is a blockchain-based electronic payment system implementing a hybrid consensus of PoET and PoW. The cryptocurrency targets a wide range of users, including individuals who are advanced in cryptography and those who are not. The system incorporates a web wallet and a console interface, improving accessibility and usability.

System Components and Interaction

The core components of the system include:

1. Blockchain Nodes: These are the fundamental building blocks of the blockchain network. Nodes maintain the blockchain and provide transaction verification services.
2. Blockchain Ledger: This distributed database holds the transaction history and state of the blockchain network.
3. Mempool: This component stores unconfirmed transactions before they are added to the blockchain.
4. Transaction Verification Process: This component ensures that only valid transactions are included in the blockchain.
5. Web Wallet: This client-facing application allows users to interact with the blockchain. It connects to some of the nodes and stores user wallet data.
6. Console Interface: This allows more sophisticated users and system administrators to interact with the blockchain network in a text-based environment.

Data flows through these components following standard protocols for blockchain networks. In the web wallet, all web data is encrypted. It processes authorization data and stores keys in a database. In the blockchain core, data packets, including block and transaction notifications, are exchanged among nodes.

Hardware/Software Mapping

This system can be deployed on various hardware platforms or cloud-based environments. Software components (nodes, web wallet, console interface) can be installed on user devices or servers.

Non-Functional Characteristics

The system's architecture ensures robust security and scalability. Blockchain's inherent design offers data security, while the hybrid PoET and PoW consensus mechanism enhances reliability and scalability.

External Interfaces

The web wallet and console interface serve as the primary external interfaces, allowing users to interact with the blockchain network.

Architectural Styles and Patterns

The system adopts the decentralized nature of blockchain technology, leading to a distributed architecture style. It adheres to standard blockchain patterns, with a hybrid consensus mechanism that combines elements of PoET and PoW.

4 Detailed system design

4.1 User interface design

The user interface for the blockchain-based application is designed to be intuitive and simple to accommodate both beginners and advanced users. The layout includes basic components similar to popular cryptocurrency wallets. The key elements of the user interface include

Landing page: The application landing page is designed to provide an overview of the application's functionality and a call to action for registration or login.

User dashboard: After logging in, users are taken to the dashboard where they can view their current balance, transaction history, initiate transactions, and access the blockchain explorer.

Transaction process: The transaction process is designed to be simple. Users need to enter the recipient's address and the amount they want to transfer. The system then confirms these details before completing the transaction.

Blockchain data: Blockchain data is presented in a transparent and understandable format, allowing users to browse the blockchain and see the details of transactions for each block.

Registration and login: The registration and login pages are simple and secure. Users must provide their details to register, and then they can use those details to log in.

4.2 Data structures

The system's data structures are mainly based on Go and are designed to mimic those in a typical blockchain. The main data structures include

Blockchain: This is a linear chain of blocks, each containing a list of transactions. Each block is linked to the previous block by storing its hash.

Block: Each block contains a list of transactions, a timestamp, a hash of the previous block, and its own hash.

Transaction: This is a transfer of cryptocurrency from one user to another. It contains the addresses of the sender and recipient, the amount of the transfer, and a timestamp.

User: represents a member of the blockchain network with unique credentials and a wallet.

MemPool: This is a collection of transactions that have been transmitted to the network but not yet included in the block.

The blockchain is not stored in the wallet; rather, the wallet interacts with the blockchain stored on the network nodes.

4.3 Classroom design

Since this application is developed in Go, it mainly uses structures rather than classes. Key structures include:

BlockChain, Block, Transaction, User, MemPool: These structures represent the fundamental components of the blockchain. They have corresponding methods for operations such as adding transactions, verifying blocks, and so on.

Package: This structure represents a packet sent over the network that contains an option (representing the type of message) and data (the message payload).

Each of these structures and their methods function together to support the blockchain and process transactions.

4.4 Implementation details

The application is mainly implemented in Go, a statically typed compiled language known for its simplicity and efficiency. The standard Go library is heavily used, as well as additional third-party libraries such as Chi for routing, Logrus for logging, and SQLite for database management.

The application architecture has a multi-level structure, including a service repository template. The application structure divides the problems into separate layers, which contributes to the maintenance and scalability of the code.

Error handling is an integral part of the system. Any problems during transactions, block creation or network communications are properly caught and handled, ensuring system reliability.

Security is a top priority, especially given the nature of blockchain-based applications. The system uses strong encryption methods to protect transactions and user data. For example, keys are encrypted in the database, which increases the security of user credentials

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.

The system successfully adheres to consensus rules borrowed from Bitcoin, such as the longest chain rule for resolving conflicts and the proof-of-work concept for mining blocks. In addition, the system provides secure storage of users' keys in an encrypted form in a database.

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.

ВИСНОВКИ

В рамках цього проекту ми провели глибоке дослідження технології блокчейн, успішно реалізувавши інноваційну та функціональну криптовалютну систему на основі блокчейну. Ця система використовує ключові атрибути технології блокчейн - децентралізацію, незмінність і прозорість - та узгоджує їх з потребами сучасних користувачів.

Наш гібридний підхід до мережевої моделі, який поєднує в собі аспекти однорангових та клієнт-серверних мереж, є значним досягненням, оскільки він забезпечує баланс між надійністю системи та ефективністю комунікації між різними вузлами мережі.

Ми ще більше диверсифікували нашу базу користувачів, представивши два типи гаманців: зручний веб-гаманець і гаманець з інтерфейсом командного рядка (CLI) для більш технологічно підкованих користувачів і майнерів. Це дозволяє нам обслуговувати широкий спектр користувачів, кожен з яких має різний рівень розуміння та знайомства з криптовалютою.

Система також відповідає критично важливим правилам консенсусу, таким як правило найдовшого ланцюжка і концепція доказу роботи, таким чином гарантуючи, що наш блокчейн підтримує цілісність і достовірність своїх даних. Нарешті, ми подбали про безпеку ключів користувачів, зберігаючи їх у зашифрованому вигляді в базі даних, щоб запобігти несанкціонованому доступу.

ЛІТЕРАТУРА

1. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [Electronic resource] Mode of access: https://bitcoin.org/bitcoin.pdf (Accessed: 28.05.2023).
2. Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. [Electronic resource] Mode of access: https://ethereum.github.io/yellowpaper/paper.pdf (Accessed: 28.05.2023).
3. Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc.
4. Go Programming Language. (2023). [Electronic resource] Mode of access: https://golang.org/doc/ (Accessed: 28.05.2023).
5. Buterin, V. (2013). A Next Generation Smart Contract & Decentralized Application Platform. [Electronic resource] Mode of access: https://ethereum.org/en/whitepaper/ (Accessed: 28.05.2023).
6. IEEE. (2016). Standard for Verification and Validation of Systems and Software - IEEE Std 1012-2016. IEEE Standards.
7. GSTC. (2018). GSTC R 34.10-2018: Information Technology - Cryptographic Protection of Information - Processes of Signing and Verification of Electronic Digital Signature. National Standard of Ukraine.
8. GSTC. (2012). GSTC R 34.11-2012: Information Technology - Cryptographic Protection of Information - Hash Functions. National Standard of Ukraine.
9. ISO/IEC. (2016). ISO/IEC 27001-2016: Information Technology - Security Methods - Information Security Management Systems - Requirements. National Standard of Ukraine.
10. 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). [Electronic resource] Mode of access: https://www.gsd.inesc-id.pt/~ler/docencia/rcs1314/papers/P2P2013_041.pdf (Accessed: 28.05.2023).

11. SAFE Network. (2020). The Evolution of Terminology with the Development of Technology: Decentralised versus Distributed. [Electronic resource] Mode of access: https://medium.com/safenetwork/evolving-terminology-with-evolved-technology-decentralized-versus-distributed-7f8b4c9eacb (Accessed: 28.05.2023).
12. Go Programming Language. (2023). Go Programming Language Playlist. [Electronic resource] Mode of access: https://www.youtube.com/playlist?list=PL4_hYwCyhAvZmzpIjwewZOdBMFJooHIHx (Accessed: 28.05.2023).
13. Standard Go Project Layout. (2023). Standard Go Project Layout. [Electronic resource] Mode of access: https://github.com/golang-standards/project-layout (Accessed: 28.05.2023).
14. Walter, K. (2023). Overview of Consensus Algorithms in Blockchain. [Electronic resource] Mode of access: https://github.com/cedricwalter/blockchain-consensus (Accessed: 28.05.2023).
15. Antonopoulos, A. M., Dhillon, V. (2017). Mastering Bitcoin: Programming the Open Blockchain. [Electronic resource] Mode of access: https://github.com/bitcoinbook/bitcoinbook (Accessed: 28.05.2023).
16. Superstas. Gcoin Mempool Implementation [Electronic resource]. — Mode of access : https://github.com/superstas/gcoin/blob/master/gcoin/mempool/mempool.go. — Title from the screen. — (2023).
17. Kiayias, A., Russell, A., David, B., Oliynykov, R. Ouroboros: A trusted secure blockchain protocol with proof of stake [Electronic resource]. — Mode of

access: https://pdfs.semanticscholar.org/7dce/801b2b13001d0d3b0319c550ee1977e456df.pdf. — Title from the screen. — (2017).

18. Liao, K., Katz, J., Zikas, V. BFT protocols under fire [Electronic resource]. — Mode of access: https://arxiv.org/pdf/1801.07447.pdf. — Title from the screen. — (2018).

19. Berini, M. Development and implementation of a bitcoin wallet application [Electronic resource]. — Mode of access: https://openaccess.uoc.edu/bitstream/10609/45861/6/mberiniTFM1215memoria.pdf. — Title from the screen. — (2015).

АКРОНІМИ ТА СКОРОЧЕННЯ

1. Blockchain: A digital register of transactions that is duplicated and distributed across a network of computer systems on a blockchain.
2. Kernel: The central component of a computing system that controls the operation of the computer and hardware.
3. Web wallet: A type of wallet that allows users to manage their cryptocurrencies through a web-based interface, making it accessible from any computer device with an internet connection.
4. Private key: A complex form of cryptography that allows a user to access their cryptocurrency.
5. Consensus mechanism: A fault-tolerant mechanism used in computer and blockchain systems to achieve the necessary agreement on a single data value or a single network state between distributed processes or multi-agent systems.
6. PoET: An acronym for Proof of Elapsed Time. This is a consensus algorithm that uses a fair lottery system where every node in the network has an equal chance of winning.
7. PoW: An abbreviation for Proof of Work. It is a consensus mechanism in the blockchain network that is used to confirm transactions and create new blocks in the chain.
8. SAD: An acronym for Software Architecture Document. A comprehensive document that covers the architecture of a software system, including behaviour, structure and other views of the system.
9. GO: A programming language used to develop the blockchain core, web wallet, and console interface.

10. Hashing: A hash function is a process that takes input data and returns a fixed-size string of bytes, typically a "digest". Hashing is the cornerstone of blockchain technology. In the context of cryptocurrencies, a cryptographic hash function is a special class of hash function that is particularly well suited to securely processing large amounts of data.
11. Nonce: A nonce ("number used only once") is a number added to a hashed or encrypted block in a blockchain that, when re-hashed, meets the complexity limitations. It is used in blockchain proof-of-work systems to make the process of mining a new block (or a ledger entry) for the blockchain computationally expensive.
12. Miner: In the context of cryptocurrencies, a miner is an individual or legal entity that confirms and verifies new transactions and adds them to the blockchain. This process involves solving complex mathematical problems and requires significant computing power.
13. Cryptocurrency: This is a common abbreviation for "cryptocurrency", a type of digital or virtual currency that uses cryptography for security. Examples of cryptocurrencies include Bitcoin, Ethereum and Ripple.
14. Bitcoin: Bitcoin is the first decentralised cryptocurrency and remains the most famous and valuable cryptocurrency. It was created in 2008 by an unknown person under the pseudonym Satoshi Nakamoto. Bitcoins are created as a reward for a process known as mining.
15. Gossip Protocol: Also known as the "epidemic protocol", it is a procedure or process of computer peer-to-peer communication that is based on the way epidemics spread. Some distributed systems use the gossip protocol for communication because of its high reliability.
16. Overlay network: An overlay network is a computer network that is overlaid on another network. The nodes in an overlay network can be thought of as

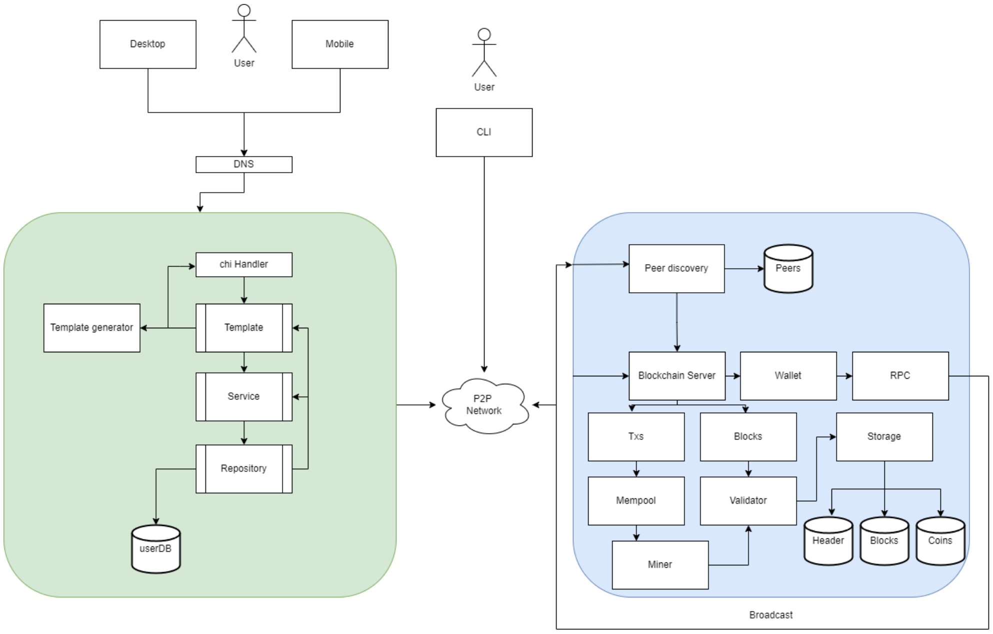
connected by virtual or logical links, each of which corresponds to a path, possibly through many physical links, in the underlying network.

17. Peer-to-peer network (P2P): A P2P network is a network in which each computer (or "peer") acts as a server for others, allowing them to share files and peripherals without the need for a central server. P2P networks can be set up at home, in a business or on the Internet.
18. Node: A device or data point in a large network. In the context of blockchain, a node is any computer that is connected to the blockchain network.
19. Wallet: In the context of cryptocurrency, a wallet is a digital tool that allows users to interact with the blockchain network. It allows users to store and manage their cryptocurrencies.
20. CLI (Command Line Interface): A text-based user interface used to enter commands directly into a computer system. In the context of this project, the CLI is intended for advanced users and miners.
21. Graphical User Interface (GUI): A type of user interface that allows users to interact with electronic devices using graphical icons and visual indicators, such as secondary labels, instead of text interfaces, typed command labels, or text navigation.
22. dAPP (decentralized application): This is an application that runs on a decentralized network, avoiding a single point of failure. dApps have back-end code that runs on a decentralized peer-to-peer network, unlike traditional applications where back-end code runs on centralized servers.
23. IP (Internet Protocol): The basic communication protocol in the Internet protocol suite for transmitting datagrams across network boundaries.
24. TCP (Transmission Control Protocol): One of the main protocols in the Internet protocol suite, TCP is used by applications that require guaranteed delivery.

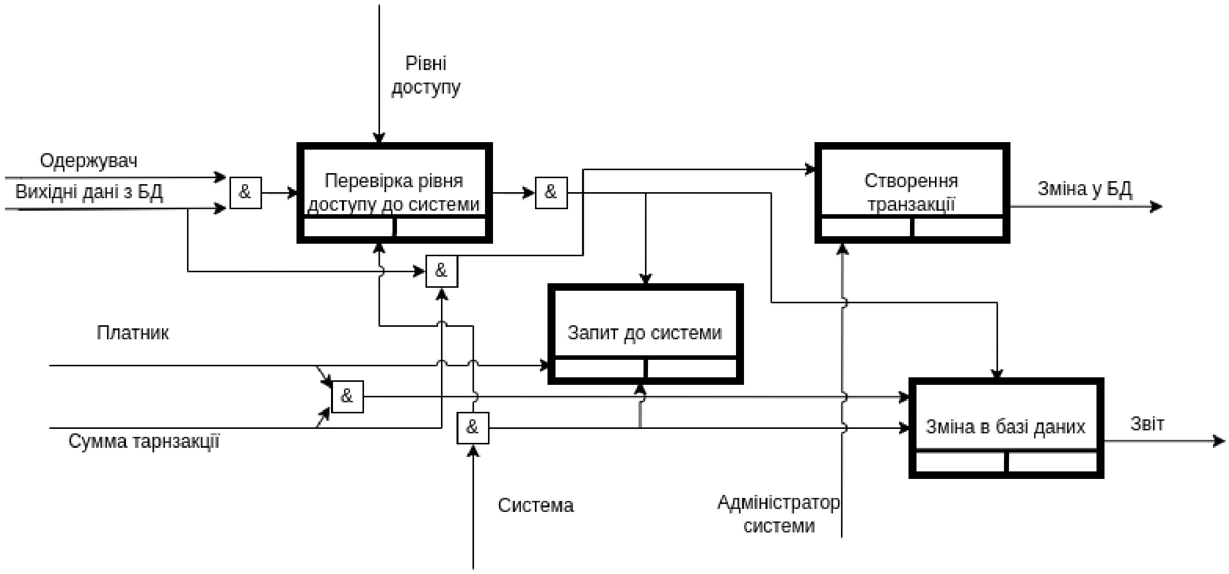
25. P2P (Peer-to-Peer): A decentralised form of network architecture in which individual network nodes ("peers") act as an independent network capable of communicating directly with other nodes using P2P communication protocols.
26. API (Application Programming Interface): A computing interface that defines the interaction between multiple software intermediaries. It defines the types of calls or queries that can be made, how they are made, the data formats that must be used, and so on.
27. JSON (JavaScript Object Notation): A lightweight data exchange format that is easy for humans to read and write, and easy for machines to analyse and generate. JSON is often used when data is sent from a server to a web page.

ДОДАТОК А

А.1 Діаграмма Архітектури



А.2 Дочірня діаграма 1-го рівня в представленні IDEF3


Рівні доступу
Одержувач
Вихідні дані з БД
&
Перевірка рівня доступу до системи
&
Створення транзакції
Зміна у БД
&
Запит до системи
Платник
&
&
Сумма тарнзакції
Зміна в базі даних
Звіт
Система
Адміністратор системи