IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 13, Issue 9, September 2023, ||Series -I|| PP 01-09

Digital user avatar on the blockchain

[1](Sudbin Danila Sergeevich)
[2](Dvornikov Dmitry Alexandrovich), [3](Zuev Dmitry Alexeyevich)
[1]Senior Software Engineer - Moneycat Financing Inc. Unit 2202D West Tower Tektite Building Exchange Road Barangay San Antonio Ortigas Pasig City Philippines 1600
[2]Institute for Cybersecurity and Digital Technologies, MIREA - Russian Technological University, Moscow, Russia
[3]Digital Technologies and Information Systems, MAI - Moscow Aviation University, Moscow, Russia
Received 26 August 2023; Accepted 07 September2023

Abstract:. The rapid growth in the number of users in the field of web2.0 and web3.0 increases the need for the research of systems that provide access to transparent and secure person's identity. The lack of a unified system of this type forces services and governmental bodies to double-check the same information several times. This drives down user loyalty, requires additional labor costs of qualified specialists, and deteriorates security. A possible solution to this problem can be envisaged as the creation of the user's avatar in the blockchain. This research offers a solution with open-source code that decreases the time of deploying digital avatar management software. Its estimated reach is 1,595 billion users.
Index terms: blockchain, authentication, digital avatar, web3.0.

I. INTRODUCTION

Due to a significant increase in the number of web2.0 and web3.0 users, the number of systems that provide their identification has grown. The lack of a unified system forces services and governmental bodies to check the same information several times. This reduces user loyalty, requires additional labor costs of qualified specialists, and deteriorates security.

An important step in this area is the creation of a decentralized digital avatar (the result of collecting such data) to simplify access to user data as well as ensure storage security and immutability.

Let's say, the client has already passed the KYC (know your client) and AML (anti-money laundering) procedures in a service. Then, with the help of this development, a third-party service can trust the quality of this verification since it is tied to a digital avatar. That third-party service will not need to conduct its own verification in the case of classical identification.

The aim of the study is to develop a framework for working with a digital avatar that allows you to quickly implement a web application (an option for interacting with a user's digital avatar) and covers the maximum number of users of third-party services (employing a minimum number of authentication systems).

To achieve the goal, it is necessary to:
(1) Study the existing approaches to solving this problem
(2) Select the minimum scope of functions for managing a digital avatar
(3) Determine the minimum number of identification systems to obtain an acceptable level of coverage
(4) Develop the code base
(5) Get an evaluation of system performance
(6) Compare services and approaches with the resulting development

II. METHODS AND MATERIALS

1. Technologies used

Blockchain standards — ERC725, ERC735;
Technology stack: frontend (React, Redux, Babel, Prettier), blockchain (IPFS, web3, Solc, Solidity), encryption (ECDSA, SHA-3 (keccak256), ABI);
Why exactly these technologies: blockchain as a distributed data storage, since this approach reduces the risk of data loss and ensures its integrity. React as it is one of the most popular front-end frameworks.

2. Related works
Alternative Properties (Physical Data Collection)

Physical data that is collected in the AR/VR metaverses can serve as alternative properties of an avatar, including for identification purposes. This data can be collected through headsets, smart glasses, character control gloves, and other IoT devices. It is suggested [2] that this data should be stored and exchanged within the blockchain. Thus, companies that rely on physical devices to gather data use the ERC725 and ERC735 standards indirectly in terms of collecting data and linking it to a digital avatar.

Other identification methods[4]

The list of some common authentication methods includes: (1) Identifier (identification) / password: to open a session on a computer or to authenticate on the Internet. (2) PIN code (personal identification number) to unlock the smart card. (3) RFID card: for accessing the building. (4) Fingerprint: for unlocking the door. (5) Facial recognition system with webcam: for starting an online session. (6) USB token. (7) One-time password token, etc.

SAuth protocol[5]

To access the system, you need to log in to service S as well as to system V, which acts as a guarantor. Both services S and V are normal websites that the user visits every day (e.g. Twitter, Facebook or Gmail). To steal an account, an attacker needs to gain access to both services S and V at once. The number of these services may increase during implementation. SAuth extends rather than replaces the existing authentication methods.

III. DETERMINING THE WAYS OF WORKING WITH A DIGITAL AVATAR

Let's define the terms. **Identity** is the owner of a digital avatar. **Claim** is a property of this **Identity. Claim issuer** is a third-party service that issues confirmations in the role of a verifier. **Smart contract** is a set of code and data located in the blockchain, which will be executed if predetermined conditions are met.
In this work, a digital avatar means uploading data about a person to the web3.0 system. The properties of the avatar must be immutable and available for addition.
Let’s define the ways of interacting with the properties of a digital avatar:
(1)The way to interact with **properties without confirmation**: in this case, **Identity** itself signs the property with its private key;
(2)The way to interact with **properties with a mandatory confirmation**: in this case, **Identity** requests **claim** from an external service. The property will be added to the user automatically after this request. In addition, the external service needs to get not just the standard private key but the **claim issuer** key in order to sign the property not as an **Identity**, but as a **claim issuer**;
(3)The way to interact with **reputation management properties**: other **Identities** (with a user key, not a verifier key) add **claim** for this **Identity**. Next, it will wait for confirmation from **Identity**. Thus, **Identity** can dispute the reputation property but cannot dispute the property from a **claim issuer**;
(4)The way **property existence check** acts: one **Identity** must be able to know if another **Identity** has a particular property.
Defining properties
When choosing platforms, preference is given to FAANG member companies (Facebook, Amazon, Apple, Netflix, Google) as they have the largest audience of all companies.
Selected properties without confirmation: full name, email.
Selected properties with mandatory claim issuer confirmation: (1)**Platform confirmation**; (2)**Alternative authorization**: Google is a FAANG member with around 3.258 billion users[10]; (3)**Social network profile**: Facebook is a FAANG member with approximately 2.95 billion monthly active users[11]; (4)**Education**: employers in most cases are interested in workers with a confirmed diploma — around 3.29 billion users, based on the number of people employed in the global labor market[15]; (5)**Professional profile**: we consider Github as an authorization service for early adopters — it features about 83 million developers [12,13]. (6)**Reputation system, credit rating, KYC, AML**: a high reputation rating allows you to improve the contractual process (loan rate, counterparty verification, etc.). The digital avatar solves the problem of those users of the financial system (about 6.4 billion [9]) who have to gather all proofs of their integrity every time the organization lacks trust in other parties. (7)**Crypto Wallet**: Metamask — used as a crypto wallet to interact with the Ethereum blockchain.
Coverage efficiency: we calculate it based on the average number of potentially active users and take the migration conversion to the general system as 10%. We get: (3.258 + 0.094 + 3.29 + 6.4 + 2.91)*0.1 = 1.5952 billion users.

The final properties of a digital avatar: profiles in systems (Google, Facebook, Github, Metamask), reputation rating (consists of a list of fines/complaints/reprimands and rewards/bonuses, etc. — confirmations are stored in the IPFS storage).
Digital avatar personal data: full name, email address, phone number, short bio and full profile bio, date of birth, medical card, driving license, citizenship, visa history, administrative offenses, credit rating, addresses of other crypto wallets.

IV. Development of a functional matrix

The Digital Avatar Journey:
(1)Authorization via Metamask: a digital avatar is tied to the web3.0 system (any of the blockchains) in which all data about it will be stored; (2)Self-adding properties required to apply for KYC and AML processing; (3) Passing KYC and AML (after this stage, we get a fully verified user whose digital avatar will not raise doubts about its authenticity); (4)At this stage, the account can accept new properties from other verification centers and act as a verifier on its own; (5)Additional properties that do not require (but imply, if necessary) a user request are formed and added by third-party system participants.

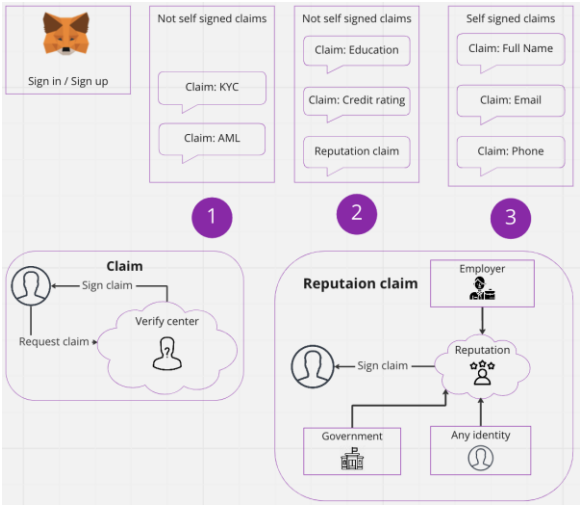

Figure 1. **Items 1, 2, 3** are the options for interacting with the smart contract.

Blocks with different types of properties (marked as claims in the diagram) display the user's journey when interacting with them.
*Properties and services are subject to change according to implementation. In this solution, those are selected that, with the smallest implementation, will deliver the greatest benefit from using the framework.

V. RESULTS

Code base:
Functional requirements: simplify assembly with the ERC725 and ERC735 identification standards. Take into account all modern development standards. Develop a front-end interface to make it easier to understand how the process functions.
1. Development of smart contracts:
1. The ERC725 standard is used as the basis. It describes proxy smart contracts that can be controlled with the help of multiple keys and other smart contracts. In this case, all the control, like all the keys, belongs to the users. Such an approach prevents data leaks. A public smart contract allows the use of a digital avatar in a decentralized manner, which in turn gives access to Dapps platforms.
Accounts based on smart contracts that comply with this standard have the following benefits: (1)can contain any asset (ERC20-like tokens etc.); (2)can execute any smart contract and deploy smart contracts; (3)have upgradeable security (via change of ownership, e.g. to gnosis-safe); (4)are simple enough to work for a long time; (5)offer a possibility of extension due to additional standardization of key/value data; (6)can act as owner/controller or trustee of other smart contracts.
Accounts can also be based on the ERC735 standard. This is a related standard for adding and removing statements to the ERC 725 identification smart contract. Next, we'll specify the set of properties required for implementation: (1) **claim issuer**: another smart contract or obtained account that issues a statement about that person. The statement publisher can be the identity contract itself; (2) **claim checker**: a smart contract or derived account that

returns a statement about the **Identity** properties. For example, whether they have an education or whether their medical record is linked to their digital avatar;
(3) **claim**: a claim is the information that the issuer has about the owner of an identity. This contains the following:

- (3.1)**topic**: uint256 is the number that closes the subject of the sentence. (e.g. 1 biometric, 2 residences. Numerical schemes and subtopics based on number ranges are to be defined);
- (3.2)**scheme**: The scheme against which this statement should be tested or how it should be processed. It's uint256 for different schemes. For example, this could trigger a contract validation, where the data would contain information about a function call and the contract address of the issuer for the function call (must be defined). This can also denote different types of keys, such as 1=ECDSA, 2=RSA, etc.;
- (3.3)**issuer**: address of the contract with the issuer's identities or the address used for placing the above-mentioned signature. If it is an identification contract, it must have the key with which the message was signed. If the key is no longer available, the statement should be regarded as invalid. The issuer can also be the address of the contract itself. The requirement for the contract can be checked with a data call;
- (3.4)**signature**: A signature that proves that the issuer of the application issued the application topic for this identity. This must be a signed message with the following structure: keccak256[] (address identityHolder_address, uint256 _ topic, bytes data) or keccak256(abi.encode(identityHolder_address, topic, data));
- (3.5)**data**: A hash of complaint data located elsewhere, a bitmask, call data or actual data based on the complaint scheme;
- (3.6)**uri**: location of the complaint: it can be HTTP links, swarm hashes, IPFS hashes, etc.

Build a smart contract with the random name KeyHolder, which extends the ERC725 standard and facilitates working with key storage for smart contracts or record discovery:
(1)When creating the contract add a control key for KeyHolder with the **ECDSA encryption algorithm**[18]; (2)Add the ability to store information: address; uint256 value; bytes data; bool approved; bool executed; (3)Add arrays for storing: keys, details about keys, data about key schemes; (4)Implement the functionality: reading (from arrays for storage), adding and deleting data.
Develop a smart contract called ClaimVerifier, which implements the functionality of checking the authenticity of a signature proposed by Claim:
(1)Implement the functionality: checking for validity, checking for validity by decrypting a signature using the **SHA-3** algorithm (**keccak256**). The data specified in the description of the ERC735 functioning takes part in the validity check; (2)Thus, expand the functionality of KeyHolder.
Build a smart contract called ClaimHolder, which extends the functionality of KeyHolder and ERC735:
(1)The data format for working with the property is specified in the description of the ERC735;
(2)Deletion and viewing occurs by property id.
Develop a smart contract called Identity, which extends the functionality of ClaimHolder and simplifies the functionality of reading **Identity**.
Work with properties and basic services occurs through ABI:

```
const contract = new web3.eth.Contract(ClaimHolder.abi, identity)
var tx  = contract.methods.removeClaim(claim).send({
   from: (wallet address), gas: 3000000
})
```

2. Development of the front-end part:
On the start screen after authorization, we have the opportunity to create 3 entities: **identity, claim issuer** and **claim checker**. All actions with adding properties are carried out by inserting data into the pop-up form.
Let's add a new one or import **Identity**. When creating a new one, add the ECDSA management key (where the private key is stored by the user, and through public access, you can check whether they have access to this account). We examined the personal account, which is shown in fig.1.

Figure 1. User's personal account after creating their digital avatar.

Besides, all transactions of interacting with the blockchain are stored in the Redux data storage. This way, we can display all the events that were recorded on the blockchain.

Properties without confirmation

We fill in the digital avatar: full name, email address, phone number, short and full profile description, date of birth, emergency medical card (medical notes), driving license, citizenship. As a result, we get a digital avatar as in fig.2.

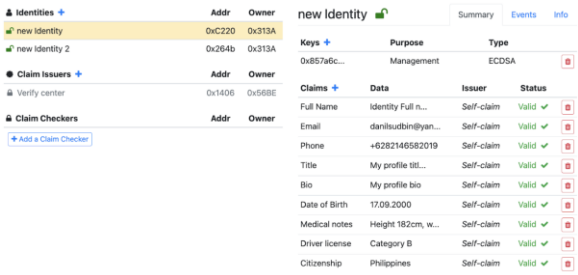

Figure 2. Personal account after adding all the properties that a digital avatar can add on one's own

At the same time, the functionality of reviewing transaction data is provided.
Subject, Issuer, Claim ID, Type, Scheme, Data, URI, Combined, Signature, Recovered, Hashed, property validated or not. In the case of adding properties on one's own, they will be confirmed automatically.

Properties with mandatory confirmation

To work with this property format, it's necessary to create a Claim Issuer.
To do this, you need to follow several steps. The interface for performing these steps is shown in fig.4:
(1)Create a user as part of Claim Issuers. (2)Add the service that will confirm this Claim Issuer. For this purpose, it's necessary to specify the URL address (to which the user will need to be transferred to confirm the property) and the property itself. As an example, consider the creation of a digital avatar verification service on this platform. (3)Create in the interface and add a private key to the Claim issuers confirmation server configuration file. (4)Restart issuer service.

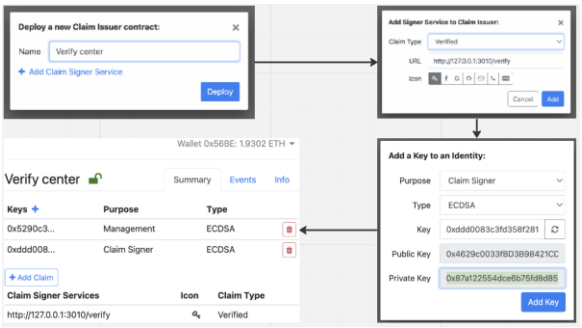

Figure 3. Personal account with functionality for adding a Claim Issuer

It should be noted that verification services must be implemented by an external service independently. When adding a property, it is necessary to sign it with the above-mentioned verifier key.

Next, we will demonstrate how you can get property confirmation from an external service. To do this, click on the "Claim +" button and select (in the pop-up window) the service that was created above. The user will be redirected to the service URL that will sign the user property. After that, a pop-up window will be created with the data received from the external service, and the user will be able to add this property to themselves. After that, it will appear in the list of properties of this digital avatar. The front end is shown in fig.4 below.

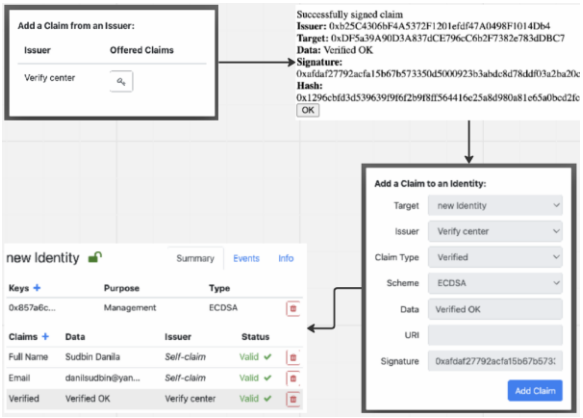

Figure 4. Algorithm for adding a property from a third-party service (Claim Issuer)

Reputation management properties

To work with the **reputation management** functionality, the following has been implemented: adding a property that affects reputation, as well as confirming this property from the digital avatar itself. The front end is shown in fig 5. below.

To demonstrate the functionality, you need to create a "new Identity 2" user and add a reputation property to it via a pop-up window. After that, this property will appear for the user in the properties of the digital avatar. To confirm this property, the "new Identity 2" digital avatar must log into their personal account and click on the "Approve" button.

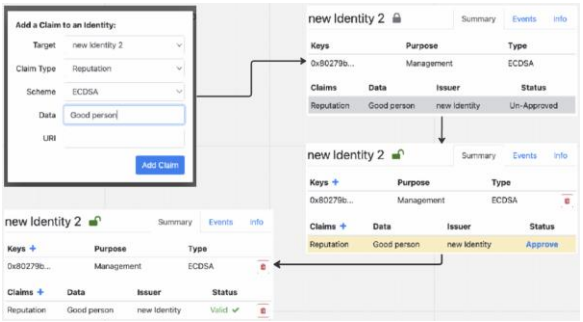

Figure 5. Ways to work with reputation properties

Checking for the presence of properties

To work with the functionality of **checking the presence of properties**, we need to add a Claim Checker. You must specify the name of the trusted claim issuer, which is responsible for adding the property to be checked (Trusted Issuer), the property (ClaimType) for checking, and the button call method to be checked (Method Name). The verification algorithm is shown in Fig. 6.

In the current demo, the Digital Avatar named 'new Identity' has passed verification, while the avatar named 'new Identity 2' has not.

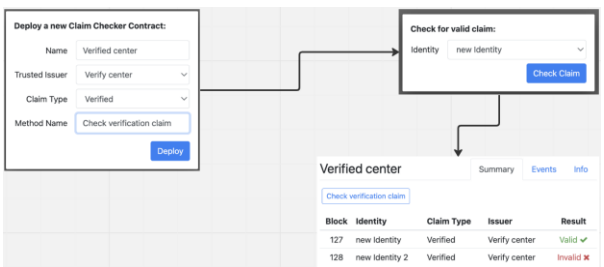

Figure 6. Checking the digital avatar property

Performance Evaluation

Experiments were performed on a local server with the following parameters: Intel Core i7, 2.4 GHz, 8 GB of RAM, and 1 TB of SSD storage. The experiments were carried out on a blockchain of size N ∈ {100, 500, 2000}. For each test, records and 50 measurements are collected.

Testing was done in the Chrome web browser using 10 concurrent requests. The results are shown in the table below.

Evaluation №1

*DA — digital avatar

*DA-{S}-{V}-{T}-{A} — where S is the number of the properties that the user signed themselves, V is the number of validated properties from a third-party verification authority, T is the number of reputation properties and A is the number of accepted reputation claims.

The results of this evaluation are shown in Table 1.

Table 1. Frontend interface performance evaluation depending on the number of digital avatar properties

Options	Loading time Fetch/XHR [s]	Data volume [kB]
DA-0-0-0-0	1.06	1.5
DA-1-0-0-0	1.24	6.7
DA-9-0-0-0	1.46	48.7
DA-9-1-0-0	1.46	54
DA-9-1-1-0	1.4	55.6
DA-9-1-1-1	1.51	59.3

Evaluation №2

The initial blockchain (N = 1) was found to be 19 KB in size. As the number of records in the blockchain increases, so does the total size of the blockchain. The average response time with 1 entry on the blockchain takes 54ms, while the call time with 1 entry takes an average of 2196ms. The scaling of the experiments was limited by the memory resources of the local computer. Although the initial size of the blockchain was small, the limiting factor was the amount of RAM needed to run system resources simultaneously.

The blockchain response time is comparable to the response time of the login function and the system data update function. The response time of the functional matrix is closely related to the call time.

The reason is that the implementation is related to the interaction with the blockchain network. Each call to the smart contract function leads to mandatory communication in the form of reading or writing data. Another thing to note is that as the number of records in the blockchain increases, the response time to requests or calls to the blockchain, as well as the functions of the system, increase slightly.

The results of this evaluation are shown in Table 2.

Table 2. Blockchain performance evaluation depending on its size

Smart contract features	N=100	N=500	N=2000
DA registration	2560±75	2620±90	2657±76
Add claim	2273±14	2289±15	2305±19
Remove claim	2251±18	2267±22	2271±21
Accept claim	2202±15	2205±14	2218±12
Request for reading claim	54±1.9	56±1.6	59±2.4
Blockchain size [kB]	38	112	392

Comparison of services and approaches with this solution

Let's introduce the term **DAF** to denote this solution (**D**igital **A**vatar **F**ramework). The results of this comparison are shown in Table 3.

Table 3. Results of comparison of this solution with alternatives

Solutions	SAuth	Firebase	Moralis	DAF
Authorization support in many web2.0 systems	+	+	-	+
web3.0 support	-	-	+	+
Property validation	-	-	-	+
Decentralization	-	-	-	+
Reducing the risk of data loss thanks to authorization through additional channels	+	-	-	+

VI. Discussion

From a scientific point of view, scalability is a typical problem of blockchains [14]. Assuming linear scaling to 20k users, we estimate that the response time to a request on the blockchain will increase by about 46% (86ms). In particular, the call time will increase by about 7.5% (2.4 seconds).

The results of the development of the functional matrix indicate that all parties of the smart contract should equally participate in the system in different roles: digital avatar, verifier, digital avatar reputation applicant (the digital avatar itself acts as the applicant).

From a managerial point of view, the most effective way to ensure the broadest possible identification is to increase the number of participants in the network of Digital Avatars. In this case, compiling a complete picture of a user on the network will be quite a feasible task.

Thanks to this solution, the user does not need to worry about several problems: (1)stability of the server and data forgery, since in web3.0, code execution and data storage are distributed between network nodes, which means the network cannot refuse to work and the data cannot be forged, since this requires the consent of all nodes. (2)Data access — access as well as data control is only on the user's side, as well as their private access keys to the system.

VII. Conclusion

During the research, each of these tasks was completed:

(1)The existing approaches to solving the problem were studied: the collection of alternative properties (physical data), alternative identification methods (such as physical media: RFID, etc.), and the SAuth authorization protocol;

(2)The minimum functionality was selected for this solution to ensure rapid implementation of the digital avatars system and maximize its effectiveness;

(3)It’s enough to use the sample of the properties indicated in this study to achieve such a remarkable result;

(4)Based on the ERC725 and ERC735 identification standards, an open-source library was launched that takes into account all modern development standards. A frontend interface was built to make it easier to understand how the process functions;

(5)An evaluation of the performance of the developed system and the effectiveness of the coverage was obtained;

(6)A comparative analysis of DAF with such services and approaches as Firebase, Moralis and SAuth was performed.

It was established that the minimum set of information (self-filled profile, one property from the verification center and a confirmed reputation property) have a load time of Fetch / XHR requests in the frontend interface of 1.51ms and the result weighs 59.3kB.

It was found that the size of the blockchain for storing 2000 accounts is 392 blocks and the coverage is approximately equal to 1.5952 billion users.

The goal of the study was achieved: an open-source code was developed to reduce the deployment time of software for managing digital avatars.

In the future, this framework can be supplemented with its own services (such as text recognition of submitted documents, searching for information about a digital avatar on the global Internet, etc.) that simplify the work of external services. Plus, access rights will be granted to those who require information about a digital avatar. In the future, it is planned to study the properties and integrations of services to increase the coverage of the system.

From a managerial point of view, this solution allows you to safely store and manage your data. It remains immutable because it is in the blockchain network, which features many copies of this information. Management is simplified when the network welcomes a large number of participants since they can partly take over the functions of managing this data (verification and confirmation).
From a scientific point of view, this solution, when compared to others, shows greater adaptability to solve the problem of secure identification and simplification of data processing and storage, which was established during the study.

References

[1]. **DAF** - this development (**D**igital **A**vatar **F**ramework) https://gitlab.com/danilsudbin/daf
[2]. Gadekallu, T. R., Huynh-The, T., Wang, W., Yenduri, G., Ranaweera, P., Pham, Q. V., ... & Liyanage, M. (2022). Blockchain for the Metaverse: A Review. arXiv preprint arXiv:2203.09738.
[3]. Bakre, A., Patil, N., & Gupta, S. (2017). Implementing decentralized digital identity using blockchain. International Journal of Engineering Technology Science and Research, 4(10), 379-385.
[4]. Idrus, S. Z. S., Cherrier, E., Rosenberger, C., & Schwartzmann, J. J. (2013). A review on authentication methods. Australian Journal of Basic and Applied Sciences, 7(5), 95-107.
[5]. Kontaxis, G., Athanasopoulos, E., Portokalidis, G., & Keromytis, A. D. (2013, November). Sauth: Protecting user accounts from password database leaks. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 187-198).
[6]. Mikula, T., & Jacobsen, R. H. (2018, August). Identity and access management with blockchain in electronic healthcare records. In 2018 21st Euromicro conference on digital system design (DSD) (pp. 699-706). IEEE.
[7]. Number of employees worldwide 1991-2022 Published by D. Clark, May 16, 2022 https://www.statista.com/statistics/1258612/global-employment-figures/
[8]. Global social networks ranked by number of users 2022 Published by S. Dixon, Jul 26, 2022 https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users
[9]. International Debt Statistics 2021.October 2020 International Bank for Reconstruction and Development / The World Bank. https://openknowledge.worldbank.org/bitstream/handle/10986/34588/9781464816109.pdf
[10]. More than 3 billion internet users now use the Google Chrome browser Anton P. | May 25, 2021 https://atlasvpn.com/blog/more-than-3-billion-internet-users-now-use-the-google-chrome-browser
[11]. Facebook: quarterly number of MAU (monthly active users) worldwide 2008-2022 Published by S. Dixon, Oct 27, 2022 https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
[12]. Number of accounts. June 18, 2020. https://en.wikipedia.org/wiki/GitHub#cite_note-23
[13]. "User search". GitHub. Retrieved January 29, 2019. https://github.com/search?q=type:user&type=Users
[14]. Li, W., Sforzin, A., Fedorov, S., & Karame, G. O. (2017, April). Towards scalable and private industrial blockchains. In Proceedings of the ACM workshop on blockchain, cryptocurrencies and contracts (pp. 9-14).
[15]. Number of employees worldwide 1991-2022 Published by D. Clark, May 16, 2022 https://www.statista.com/statistics/1258612/global-employment-figures/