All Sciences Proceedings
http://as-proceeding.com/

1st International Conference on Pioneer and Innovative Studies

June 5-7, 2023 : Konya, Turkey

© 2023 Published by All Sciences Proceedings

https://as-proceeding.com/index.php/icpis

Integrating Technologies for a Seamless Play-to-Earn Experience: A Case Study on a Hyper-Casual Mobile Game with a Decentralized Ecosystem

Recep AKKAYA[*] , Mert ÜNAL[*] and Rayan ABRİ[2]

[*]*Computer Engineering, Ostim Technical University, Turkey*

[*]*(mertunal.education@gmail.com)*

Abstract – This project integrates Web 3.0, web development, and mobile game development to create a play-to-earn (P2E) mobile game with a web component. The objective of this project is to seamlessly combine a mobile game with a website and a distinctive token ecosystem to provide players with an enjoyable experience. For the purposes of creating mobile games, websites, and smart contracts, respectively, the Unity Game Engine and the "Next.js' framework, Solidity, are employed. The project makes use of Web3 Modal, the MetaMask extension, Microsoft Azure PlayFab for player management, and Web3 Modal for a seamless interface with blockchain networks. The main outcomes of the project include successful implementations like a mobile game with simple gameplay that players can quickly understand and a website with an interface that will allow you to withdraw crypto tokens securely using smart contracts. By offering real benefits like personalized crypto tokens, the idea increases player participation. The primary conclusion to be drawn from this project is that combining gaming with blockchain integration results in a unique and engaging experience for gamers. In conclusion, our study shows that integrating several technologies to improve user experiences in the gaming and blockchain areas is both feasible and potentially rewarding.

Keywords – Play-to-Earn, Hyper-casual, Web3.0, Blockchain, dApp, Mobile Game, Cryptocurrency, Smart Contract

I. INTRODUCTION

The combination of technology, gaming, and blockchain has had a transformative impact on various industries. In this article, we want to share an exciting project that merges mobile game development, web development, and Web 3.0 integration. Our goal is to create an immersive experience that captivates players.

In recent years, mobile gaming has become incredibly popular among people of all ages and backgrounds. Our project focuses on the hyper-casual genre, known for its simple gameplay and minimalist design, making it accessible to a wide range of players.

Web development plays a crucial role in connecting the mobile game to a larger token ecosystem. We have developed a user-friendly website where players can easily view their in-game scores, claim custom tokens, and engage with the project's ecosystem. The website has a responsive and interactive interface, enhancing the user experience and encouraging exploration of its features.

Our project heavily relies on Web 3.0 technologies, which securely connect users' crypto wallets to the website. This integration allows players to manage their crypto wallet accounts and interact with decentralized applications effortlessly. Smart contracts are implemented to reward players based on their in-game achievements, ensuring transparent and secure token transfers.

The methods, resources, and technology used in the project will all be covered in detail in the sections that follow. Our goal is to establish a unique methodology that fuses digital currency, technology, and entertainment. We hope to give our audience a distinctive and engrossing experience through the seamless combination of mobile game development, web development, and Web 3.0 modules.

II. Materials and Method

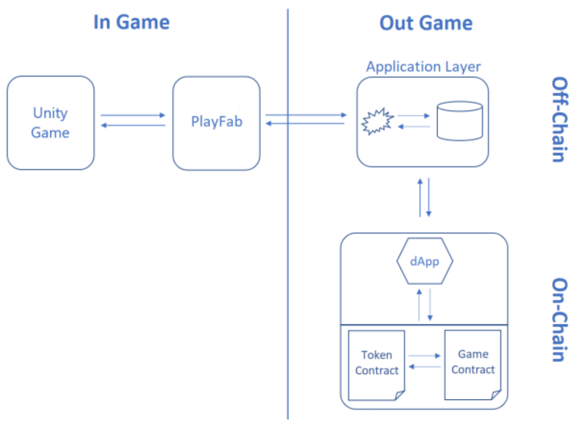


Fig. 1 System Architecture

This section provides a thorough analysis of the tools and methods applied as this project developed. It digs into the particular steps needed to develop mobile games, build websites, and implement Web 3.0 technology. This provides a description of the procedures, equipment, and materials used in the creation of the mobile game. We'll also look into the frameworks and construction methods employed for the project's website. The integration of Web 3.0 will then be examined, with a focus on how the project is using blockchain technology, including the use of tools, libraries, and smart contract execution. Together, these subsections offer a detailed overview of the tools and methods employed to complete this imaginative undertaking.

A. Mobile Game Development Process

Game Concept and Design: Game concept and design were key early steps in the game development process. We considered a number of ideas before deciding that the hyper-casual genre was most suited for our project. The short playing sessions and straightforward gameplay features that characterize hyper-casual games make them widely accessible to players of all ages and skill levels. These games typically include simple controls and objectives that make it easy for players to grasp and participate in the gameplay. We outlined the main gameplay mechanisms and game objectives during the design stage. Our goal was to create a simple game that nevertheless provided a fun and pleasurable experience.

We used Panteon Academy's resources to improve the aesthetic appeal of our game [4]. We used these resources to create a visually appealing game environment after receiving the required authorizations and acknowledgments. The overall visual design of the game was greatly improved by the incorporation of visuals, animations, and visual effects from Panteon Academy's assets. We were able to develop a sophisticated and compelling visual experience that could engage and entertain gamers thanks to the use of these resources.

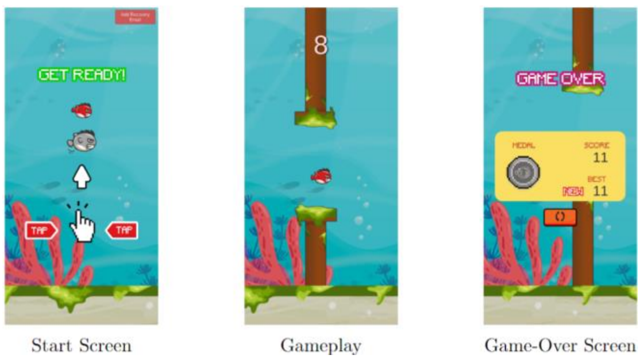


Fig. 2 Tappy Fish Mobile Game

Game Engine: We choose to use the Unity Game Engine [5] as our main development platform in order to guarantee versatility and universal usability. For game production, Unity provides a very effective and user-friendly platform. We were able to create game logic, include audio and visual assets, and provide engaging gameplay experiences due to its rich features and tools.

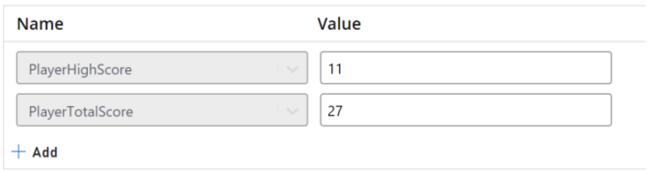
We used the C# programming language to create and implement the game mechanics. We were able to develop the game's mechanics, including game dynamics, player interactions, and PlayFab services, using C#. We were able to manage these factors with simplicity because of C#'s dependability and efficacy, which also made it easier to create modular, reusable code structures that corresponded to object-oriented programming standards.

Utilizing Unity's scripting features, unique C# scripts were created to manage user input and regulate the behavior of game objects. We were able to effortlessly add important elements like obstacle generation, player movement, collision detection, and score management through the use of these scripts. The installation of these essential elements was simple and effective thanks to Unity's scripting capabilities.

Gameplay Mechanics and User Experience: We placed a lot of effort on creating simple-to-understand game mechanics during the whole development process. We took inspiration from the well-known hyper-casual game Flappy Bird [6] in order to achieve this goal. To ensure accessibility and promote recurrent gameplay, we added features from the hyper-casual genre, such as its defining simplicity, minimalistic style, and brief play sessions. We intended to create an experience that would be approachable and interesting for players by including these elements in our game.

To encourage players to achieve greater scores and advance their talents, a reward system was put in place. To increase player engagement and pleasure, various reward mechanisms were included, such as the ability to earn cryptocurrency tokens [7], depending on in-game scores.

Microsoft Azure PlayFab: We integrated database services from Microsoft Azure PlayFab [8], [9] to manage player statistics, account information, and authentication. Secure player management, made possible by PlayFab, ensured trustworthy authentication and consistent player experiences across many devices. To handle and save game scores and other important player data, PlayFab's data storage features were used. As a result, player progress and accomplishments from many gaming sessions could be retrieved and synchronized.



Name	Value
PlayerHighScore	11
PlayerTotalScore	27

Fig. 3 PlayFab Player Statistics

Testing and Iteration: To find and fix any faults or problems, extensive testing was done throughout the development process. To ensure compatibility and maximize performance, the game was tested across a range of platforms. To learn more about the tastes and behaviors of players, player feedback was examined. The overall player experience was improved using this data in iterations.

B. Mobile Game Development Process

In this section, we'll examine the technical details of our website. We will talk about the precise methodology, tools, designs, and technologies used to make it come to life and make it possible for smooth integration with the unique token ecosystem.

The popular Next.js React framework [10], known for its efficiency and scalability, was used throughout the web development process. A seamless user experience was made possible because of Next.js's ability to create interactive and responsive user interfaces. The Microsoft Azure PlayFab database was also integrated in order to handle player authentication, store user information, and process transaction requests.

The three primary pages of our website, the login page, the main page, and the admin page, were created during the web construction process to make it more user-friendly. Players used the website as a platform to log into their gaming accounts, check their in-game standings, link their cryptocurrency wallets, and engage with the unique token ecosystem.

The login page acted as the entry point for users, requiring them to authenticate with their PlayFab email and password, which are added by players in our mobile game’s ‘Add Recovery Email’ panel. This authentication process ensured that only registered players with registered PlayFab accounts could access the main features of the website. Upon successful authentication, users were directed to the main page.

Users had to enter the site through the login page and authenticate using their PlayFab email and password, which were added by players under the Add Recovery Email panel of the mobile game. Only registered gamers with active PlayFab accounts were able to access the website's core features thanks to this authentication mechanism.

Users who successfully authenticated were taken to the home page.

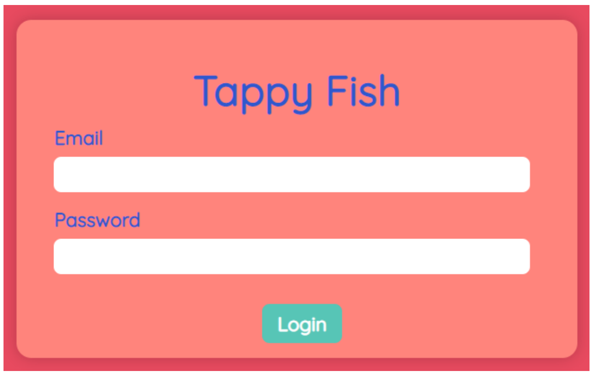
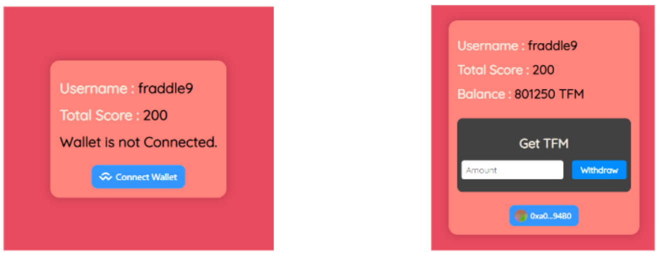


Fig. 4 Login Page

Players could display their usernames and in-game scores on the website's home page, which had an easy-to-use user panel. Users may connect their crypto-wallets to the website by clicking the "Connect Wallet" button, which is also available. As soon as the connection was made, the button vanished, and a fresh panel was added to the user panel. This new panel showed users' token balances and had a "Withdraw" button and an amount label that they could use to ask for customized tokens in exchange for a certain amount.


Username : fraddle9
Total Score : 200
Wallet is not Connected.

Username : fraddle9
Total Score : 200
Balance : 801250 TFM

Before Wallet Connection After Wallet Connection

Fig. 5 User Panel

Notably, the admin could connect their wallet to the website using the "Connect Wallet" button, and this caused an invisible admin panel button on the home page to appear. Following a successful connection, the admin could access the admin panel by selecting the admin panel button and providing the necessary password. Users withdrawal requests, together with their wallet addresses and token amounts requested, were accessible through the admin panel. Also, the admin might start the token transfers by pressing the "Confirm Transactions" button.

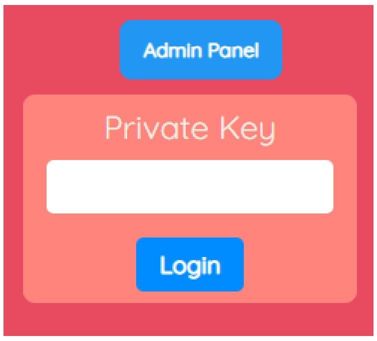


Fig. 6 Admin Panel

C. Web 3.0 Integration

The website's access to blockchain networks, ease of wallet connections, and smooth interactions with smart contracts were all made possible thanks to Web 3.0. The integration of wallet connection with Web3Modal [11], generation and withdrawal of custom tokens using the Web3.js library, and Solidity-written smart contracts are the main topics of this section.

Wallet Connection: With the help of the Web 3.0 integration, users were able to safely manage their accounts and engage with decentralized applications by linking their crypto-wallets to the website. We used the Web3Modal library, an open protocol to securely communicate between wallets and Dapps, for wallet connections. This library offered a unified interface for connecting to different blockchain networks and supported multiple wallet providers, including Metamask [12], Coinbase [13], Bitski [14], Venly [15], and others. Users were able to access their accounts on the website by connecting their wallets via Web3Modal.

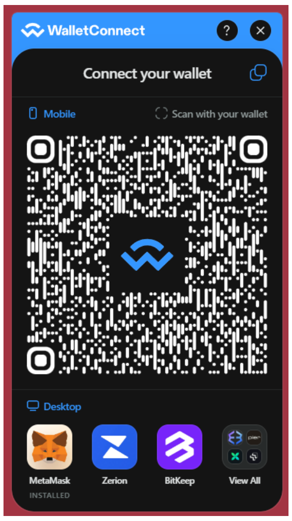
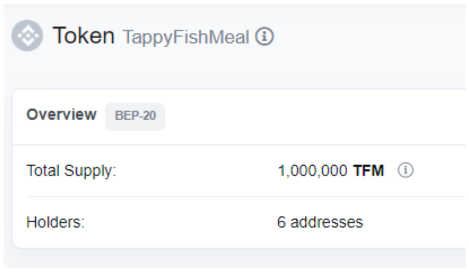


Fig. 7 Wallet Connection

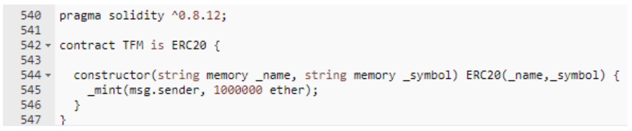
Smart Contracts and Solidity: Solidity [16] based smart contracts served as the framework for our token ecosystem. They made sure that token transactions between users and the website were safe and clear, enabling users to get tokens based on their performance in the game. The TokenContract, our main smart contract, was created in Solidity and is based on the ERC-20 [17] standard. It offered crucial token management features, such as creating a supply of tokens (minting) upon deployment and facilitating withdrawals.

The token contract is implemented as a TFM (Tappy Fish Meal) contract. Version 0.8.12 of Solidity was used to write it. The ERC-20 contract, for implementing fungible coins on the Ethereum blockchain, is extended by this contract. Upon deployment, the constructor function of the TFM contract calls the constructor function of the ERC-20 contract with the supplied name and symbol to initialize the token. Within the ecosystem of the project, this contract forms the foundation for the unique token. Transfers, permissions, and balance inquiries are among the tools and facilities it offers for managing tokens.



Total Supply:	1,000,000 TFM
Holders:	6 addresses

Fig. 8 Custom Token TFM

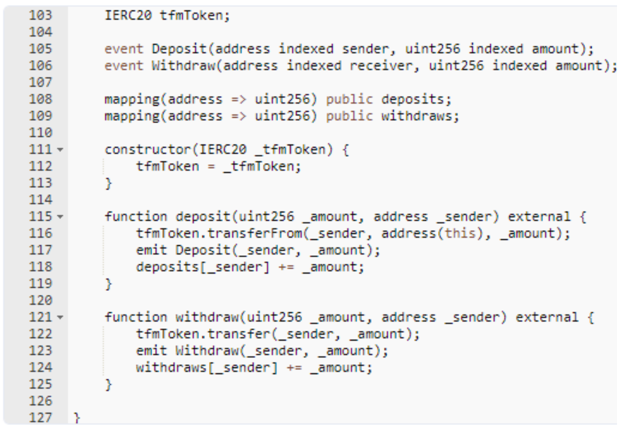


```
pragma solidity ^0.8.12;

contract TFM is ERC20 {

  constructor(string memory _name, string memory _symbol) ERC20(_name,_symbol) {
    _mint(msg.sender, 1000000 ether);
  }
}
```

Fig. 9 Token Contract

The game contract is tasked with making it easier for tokens to be withdrawn from the project's environment. The TFM token contract, which is supplied as a parameter in its constructor, is interacted with by it. The TFM token contract address is passed to the game contract's constructor as an instance of the IERC-20 interface. A standardized interface for communicating with ERC-20 tokens is provided through the IERC-20 interface.

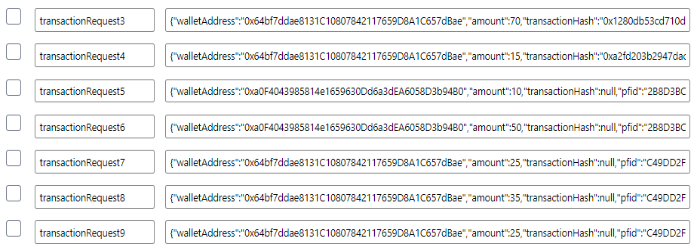
Users are able to get TFM tokens from the 'Game Contract' using the 'Withdraw' function call. Using the transfer function of the TFM token contract, it sends the requested number of tokens from the contract address to the sender's address. The mapping variable "withdraws" is our "game contract". This mapping records the addresses and token withdrawal amounts for every user.



```
    IERC20 tfmToken;

    event Deposit(address indexed sender, uint256 indexed amount);
    event Withdraw(address indexed receiver, uint256 indexed amount);

    mapping(address => uint256) public deposits;
    mapping(address => uint256) public withdraws;

    constructor(IERC20 _tfmToken) {
        tfmToken = _tfmToken;
    }

    function deposit(uint256 _amount, address _sender) external {
        tfmToken.transferFrom(_sender, address(this), _amount);
        emit Deposit(_sender, _amount);
        deposits[_sender] += _amount;
    }

    function withdraw(uint256 _amount, address _sender) external {
        tfmToken.transfer(_sender, _amount);
        emit Withdraw(_sender, _amount);
        withdraws[_sender] += _amount;
    }

}
```

Fig. 10 Game Contract

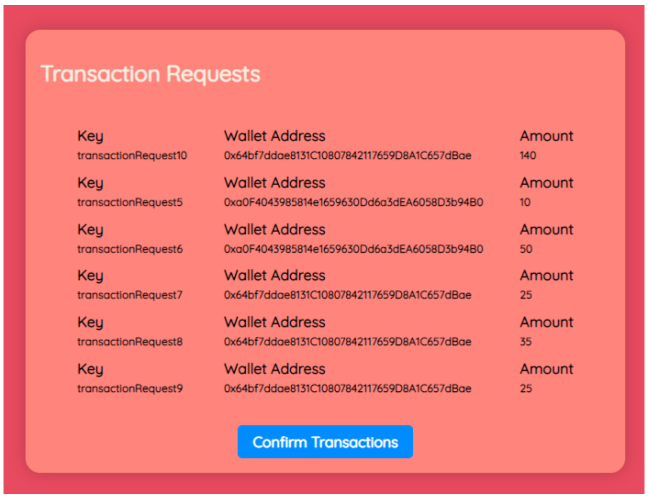
Withdrawal Process: Users can decide to request a specified quantity of TFM custom tokens. Through the user panel on the website, the user starts this process. The requested amount is verified when a user requests a withdrawal. The withdrawal is declined if it exceeds the player's score or is below zero. The requested amount is subtracted from the player's score and placed in the PlayFab database as a transaction request along with the user's wallet address, PlayFab ID, and transaction hash, which returns after a successful transaction. In our case, after a successful withdrawal.



☐	transactionRequest3	{"walletAddress":"0x64bf7ddae8131C10807842117659D8A1C657dBae","amount":70,"transactionHash":"0x1280db53cd710d
☐	transactionRequest4	{"walletAddress":"0x64bf7ddae8131C10807842117659D8A1C657dBae","amount":15,"transactionHash":"0xa2fd203b2947dac
☐	transactionRequest5	{"walletAddress":"0xa0F4043985814e1659630Dd6a3dEA6058D3b94B0","amount":10,"transactionHash":null,"pfid":"2B8D3BC
☐	transactionRequest6	{"walletAddress":"0xa0F4043985814e1659630Dd6a3dEA6058D3b94B0","amount":50,"transactionHash":null,"pfid":"2B8D3BC
☐	transactionRequest7	{"walletAddress":"0x64bf7ddae8131C10807842117659D8A1C657dBae","amount":25,"transactionHash":null,"pfid":"C49DD2F
☐	transactionRequest8	{"walletAddress":"0x64bf7ddae8131C10807842117659D8A1C657dBae","amount":35,"transactionHash":null,"pfid":"C49DD2F
☐	transactionRequest9	{"walletAddress":"0x64bf7ddae8131C10807842117659D8A1C657dBae","amount":25,"transactionHash":null,"pfid":"C49DD2F

Fig. 11 Transaction Requests

The admin page displays all transaction requests, giving the admin a complete list of all waiting withdrawals. The confirm transactions button is used by the admin to verify the requests and confirm the transactions, which activates the withdraw feature of the game contract. Amounts of TFM custom tokens are transferred from the game contract to the user's wallet address via the withdraw function after confirmation. The withdrawal function of the game contract is used to carry out the transaction, which also activates the transfer function of the token contract. The transaction hash of the successfully completed transfer is appended to the transaction request data in PlayFab, marking the request as fulfilled after a successful transfer.



Transaction Requests

Key	Wallet Address	Amount
transactionRequest10	0x64bf7ddae8131C10807842117659D8A1C657dBae	140
transactionRequest5	0xa0F4043985814e1659630Dd6a3dEA6058D3b94B0	10
transactionRequest6	0xa0F4043985814e1659630Dd6a3dEA6058D3b94B0	50
transactionRequest7	0x64bf7ddae8131C10807842117659D8A1C657dBae	25
transactionRequest8	0x64bf7ddae8131C10807842117659D8A1C657dBae	35
transactionRequest9	0x64bf7ddae8131C10807842117659D8A1C657dBae	25

Fig. 12 Admin Page

III. Results

Our play-to-earn [18] (P2E) mobile game adopted the hyper-casual genre's simple gameplay. Also, players could connect their cryptocurrency wallets to the website's user-friendly interface to monitor their in-game scores and take part in our custom decentralized token ecosystem.

IV. DISCUSSION

Our website, which has a user-friendly layout, enhances the mobile game. Players may effortlessly connect their crypto wallets, get their in-game scores, and take part in the exclusive token ecosystem.

The Solidity smart contracts have made it easier to withdraw tokens and make secure token transfers. As a result, the ecosystem's transactions are guaranteed to be reliable and honest, which gives players more trust. The security of the entire system and user-based data tables is further enhanced by the use of Microsoft Azure PlayFab for player management and authentication.

This project also shows how crucial it is to have user-friendly interfaces and intuitive gameplay to draw in and keep gamers. The game can reach a wider audience by concentrating on the hyper-casual genre, encouraging increased adoption and involvement.

Although the project's outcomes are encouraging, there are a number of areas that call for additional study and expansion. Scalability is an important factor since the system must be able to manage increased user activity and token transactions. The play-to-earn model's long-term profitability and its required time for development also depend on user uptake and retention.

V. CONCLUSION

In conclusion, the project introduced a "play-to-earn" model, allowing players to earn rewards through gameplay. This shows the potential of the newly trending blockchain technology in games to provide a more engaging and rewarding gaming experience. The results show us that additional research in this area has the potential to transform the gaming business and open up new doors for both game makers and users.

ACKNOWLEDGEMENT

Versiyonbir Blockchain Teknolojileri (info@versiyonbir.com) and Panteon Yazılım Oyun Eğitim Teknolojileri ve Hizmetleri San. Tic. A. Ş. (info@panteon.games) provided support for this work.

REFERENCES

[1] M. Di Pierro, *"What Is the Blockchain?"* in Computing in Science & Engineering, vol. 19, no. 5, pp. 92-95, 2017, doi: 10.1109/MCSE.2017.3421554.

[2] RUDMAN, Riaan; BRUWER, Rikus. *Defining Web 3.0: opportunities and challenges*. The electronic library, 2016.

[3] Z. Yang and B. Sun, "*Hyper-Casual Endless Game Based Dynamic Difficulty Adjustment System For Players Replay Ability*," 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Exeter, United Kingdom, 2020, pp. 860-866, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00133.

[4] (2022) The Panteon Academy website. [Online]. Available: https://www.panteon.games/akademi/

[5] Nicoll, B., Keogh, B. (2019). *The Unity Game Engine and the Circuits of Cultural Software*. In: The Unity Game Engine and the Circuits of Cultural Software. Palgrave Pivot, Cham. https://doi.org/10.1007/978-3-030-25012-6_1

[6] Flappy Bird, Dong Nyguyen, GEARS Studios, 2013.

[7] Liu, C., Wang, H. (2019). *Crypto Tokens and Token Offerings: An Introduction*. In: Goutte, S., Guesmi, K., Saadi, S. (eds) Cryptofinance and Mechanisms of Exchange. Contributions to Management Science. Springer, Cham. https://doi.org/10.1007/978-3-030-30738-7_8

[8] Bors, B. (2023). *Game Analytics*. In: Game Backend Development. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8910-5_7

[9] Nicoll, B., Keogh, B. (2019). *The Unity Game Engine and the Circuits of Cultural Software*. In: The Unity Game Engine and the Circuits of Cultural Software. Palgrave Pivot, Cham. https://doi.org/10.1007/978-3-030-25012-6_1

[10] Thakkar, M. (2020). *Next.js*. In: Building React Apps with Server-Side Rendering. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5869-9_3

[11] (2022) Web3Modal website. [Online]. Available: https://web3modal.com

[12] (2016) Metamask website. [Online]. Available: https://metamask.io

[13] (2012) Coinbase website. [Online]. Available: https://www.coinbase.com

[14] (2018) Bitski website. [Online]. Available: https://www.bitski.com

[15] (2018) Venly website. [Online]. Available: https://www.venly.io

[16] M. Wohrer and U. Zdun, "*Smart contracts: security patterns in the ethereum ecosystem and solidity*," 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Campobasso, Italy, 2018, pp. 2-8, doi: 10.1109/IWBOSE.2018.8327565.

[17] Bauer, D.P. (2022). ERC-20: *Fungible Tokens*. In: Getting Started with Ethereum . Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8045-4_3

[18] VIDAL-TOMÁS, David. *The new crypto niche: NFTs, play-to-earn, and metaverse tokens*. Finance research letters, 2022, 47: 102742.