Cross-platform application for NFT tokens

https://doi.org/10.31713/MCIT.2023.052

O. Hladka
Department of Computer Technology and Economic Cybernetics
National University of Water and Environmental Engineering
Rivne, Ukraine
o.m.hladka@nuwm.edu.ua

I. Karpovich
Department of Computer Technology and Economic Cybernetics
National University of Water and Environmental Engineering
Rivne, Ukraine
i.m.karpovich@nuwm.edu.ua

M. Opanasiuk
Master's student
National University of Water and Environmental Engineering
Rivne, Ukraine
opanasiuk_ak19@nuwm.edu.ua

Abstract—**A software application has been developed, which is a platform for creating, storing, viewing and selling NFT tokens. The object of research is the process of managing NFT tokens, the subject of research is the development and implementation of the NFT token marketplace. The developed application is a micro service that includes an API for working with NFT tokens and a client part that provides a convenient way to use the marketplace for users. The backend part was developed in the C# programming language, and the frontend part uses React and Redux. OOP methods, the Git version control system, namely Github, work with multithread, asynchrony and implemented SOLID principles were used during development. The application allows users to access NFT tokens, perform various operations with them and manage tokens. The libraries used in the application ensure stability and security in the process of working with tokens. This application allows for efficient and secure exchange of NFT tokens between different systems and users. The developed software product is built according to the principle of Clean Architecture, which will provide easy further expansion of the scope of the project**

Keywords—NFT tokens; blockchain technologies; cross-platform application; marketplace; software; Clean Architecture.

I. INTRODUCTION

Computer technology today plays an important role in the development of new products and services that help simplify and improve many aspects of our lives. One such innovation is the application of NFT tokens, which is a hot topic in today's world, especially in the field of cryptocurrencies and blockchain technologies. Recently, NFT tokens have become popular among artists, musicians, and other creative people, as they allow digital creations to be stored and sold as unique art objects. NFT tokens can be used to store digital assets such as music, videos, photos and other digital materials.

Non-Fungible Token (NFT) is a blockchain technology that allows the creation of digital tokens that are not fungible one-to-one. Each NFT token has its own unique identifier that allows it to be identified as a unique object. It allows you to create digital assets that can be sold as unique pieces of art, as well as be used to store other digital assets such as music, videos, photos and other materials. NFTs solve provenance and ownership issues. In traditional art or collectibles markets, proving a work's authenticity or ownership can be a cumbersome process. However, with NFTs, the authenticity of a digital asset can be verified using blockchain technology and ownership can be easily transferred with the click of a button.

There are several types of NFT tokens used in different industries: ERC-721 tokens, which are used on the Ethereum blockchain for to create unique art objects and other digitals assets; ERC-1155 tokens for to create both unique and replaceable tokens that can be used in games and other online platforms, etc.

To facilitate the creation, management and sale of NFTs, various applications and platforms have emerged in recent years. These platforms enable users to easily create, buy, sell and exchange NFTs. In addition, many of these platforms offer features such as digital wallets, marketplaces, and artist profiles to further enhance the user experience. One such platform is OpenSea, a popular marketplace for buying and selling NFTs that allows users to create and sell their own NFTs, as well as view and buy NFTs created by others [1].

Despite the presence of such applications, the demand for platforms that allow working with NFTs is only growing. Because NFT is a revolutionary technology that has changed the art and collectibles market by enabling the creation and trading of unique, verifiable digitals assets. With the increasing popularity of NFTs, it is important for individuals and businesses to have a comprehensive understanding of their use and potential applications.

The purpose of this research is to create a client-server application for managing NFTs, and to explore how this technology is changing the way digitals assets are managed and traded. The object of the research is the microservice architecture for managing NFT tokens, and the subject of the research is the development of the NFT token management system.

II. Analysis of Technologies of the Application

A. Server technology stack

The technology stack used in the NFT token application consists of several popular and widely used web application development tools and frameworks. Since this is a client-server application, two parts were developed: backend and frontend. The backend stack includes Entity Framework Core, MS SQL Server, UnitOfWork and Repository patterns, MediatR pattern, JWT authentication, NLog logging, Swagger and the Cors engine.

Entity Framework Core is an object-relational mapping (ORM) framework that allows developers to work with databases using .NET objects, including MS SQL Server, and provides a powerful set of tools for querying, updating, and manipulating data [2].

MS SQL Server is a relational database management system that provides a highly scalable, reliable and secure platform for data storage and management. MS SQL Server supports advanced security features including role-based security, encryption and secure communication protocols, which is especially important in the context of an NFT token application that deals with sensitive data and requires a high level of security. MS SQL Server supports transactional data processing, which is crucial for the application of NFT tokens, where a transaction represents the transfer of a unique digital asset from one owner to another. MS SQL Server provides a number of advanced functionalities, including advanced analytics, machine learning and business intelligence, which were used to analyze the performance of the developed application, as well as to perform advanced analytics on the data stored in the database.

The UnitOfWork and Repository design patterns are implemented in our NFT token application using Entity Framework Core to ensure data persistence. The UnitOfWork pattern is used to manage transactions and ensure data integrity and consistency of all changes made to the database. The Repository pattern provides a way of encapsulating the logic of requests and data manipulation, that is, it is responsible for abstracting the application's data access level [3].

The MediatR pattern is a design pattern that allows you to separate communication between system components [4]. MediatR is used to implement the Command Query Responsibility Segregation (CQRS) architectural pattern in our NFT token application. The MediatR pattern is responsible for separating the sender of the request (i.e. command or request) from the recipient of the request (i.e. the handler). This pattern ensures that commands and requests are handled by separate classes, each of which has a separate responsibility. In our application, the MediatR pattern is used to implement the business logic of the application and ensures that the handler is decoupled from the sender, and allows us to change or extend the behavior of the handler without affecting the rest of the application.

Together, the UnitOfWork, Repository, and MediatR patterns provide a robust and scalable architecture for our NFT token application, and ensure that the data access layer is abstracted, business logic is decoupled, and transactions are atomic and consistent.

JWT authentication is used in the application to securely transfer user authentication data between the client and the server. The security method uses JSON tags (JWT) to authenticate and authorize users and provides a way to securely transfer information between the client and the server. This authentication method is more secure than traditional methods such as cookies and session IDs [5].

For logging, the platform uses a flexible and extensible logging framework, NLog logging, which provides the ability to collect and store data about errors, warnings, and other events that occur during application execution, and can help diagnose and troubleshoot problems. NLog allows developers to log different types of messages with different severity levels, such as debug, info, warning, error, and fatal [6].

Swagger and Cross-Origin Resource Sharing (CORS) are two important components in our NFT token API application. Swagger is a tool that helps create API documentation, allows us to describe the structure of our API using YAML or JSON format, and automatically generates an interactive interface that developers can use to explore and test the API. This documentation is important for developers who need to integrate with our API because it provides a clear and concise overview of the endpoints, their inputs and outputs, and any authentication or authorization required [7].

One of the key benefits of using Swagger in our NFT token API application is that it helps standardize our API documentation. By defining the structure of our API using the OpenAPI specification, we can ensure that all of our endpoints are documented in a consistent manner. This makes it easier for developers to understand our API and create integrations with it.

CORS is a security mechanism used to restrict web applications from accessing resources on another domain. In our NFT token API application, we use CORS to determine which domains are allowed to access our API. This is important for security reasons as it helps prevent malicious attacks that could compromise our API or its data. In addition, CORS provides a standard cross-request control mechanism that makes it easier to implement and maintain security policies across different parts of our application.

The backend of the application is implemented in the C# programming language, which supports the multithreading required to create high-performance applications such as an NFT token application that requires real-time updates and large amounts of data processing. C# supports asynchronous programming, which allows you to write code that can be executed in parallel without blocking the main thread, resulting in faster and more efficient code [8].

The advantage of using C# for an NFT token application is its compatibility with other programming languages and frameworks. C# is designed to work seamlessly with other Microsoft technologies, such as .NET Core, which is a cross-platform, open-source framework that can be used to develop applications for Windows, Linux, and macOS. This means that the

developed NFT token application can be deployed on any platform without any problems.

The backend API part of the application is implemented according to the principle of Clean Architecture and contains 5 application levels:

1) allows you to save the entities that the ORM system converts into the corresponding database tables;

2) saves DB migrations if it is necessary to change the parameters of one of the DB tables, so that it can be done safely, without data loss; Fluent API configurations for objects will also be stored here to maintain database integrity as much as possible;

3) stores services, manually created exception handling classes and constants necessary for the application in its various parts, which will allow you to easily change the value of the constants only in one place of the application, without changing them throughout the project;

4) is a level where requests and commands implemented by the MediatR pattern are stored;

5) is the level of processing incoming HTTP requests, where the configuration of all application services, JWT authentication and actual controllers for processing all user requests provided by the system will take place.

B. Stack of interface technologies

The front-end part is implemented in the TypeScript programming language using JavaScript React JS libraries in connection with Redux. React JS was used to build the user interface because it is based on the concept of a virtual Document Object Model (DOM), which breaks down a complex interface into independent, reusable elements and provides fast and efficient interface updates. The Redux library was used to manage the state of the application, since the React project itself is stateless [9]. Redux offers centralized storage of application state in one place "store", which simplifies access and update of state, makes it more predictable and manageable. Redux also makes it easy to add new functionality to an application using middleware functions that can intercept and modify actions before they reach reducers that change state. This allowed us to add logging, asynchronous requests, routing, and other features to our application.

III. CONCLUSIONS

The implementation of the NFT token marketplace application involved a combination of different technologies, including Entity Framework Core, MS SQL Server, UnitOfWork and Repository patterns, MediatR pattern, JWT authentication, NLog logging, Swagger, the CORS engine on the backend, and React JS, Redux, and Typescript on the frontend.

One of the main problems during implementation were working with asynchrony and multithreaded processes. We solved this problem using the MediatR pattern, which helped us simplify the implementation of asynchronous requests by using a mediator class to manage communication between different parts of the application.

The MediatR pattern allowed us to use of request and response objects, which simplified us to management and control of data flow. UnitOfWork and Repository helped us to simplify work with the database by providing a standard way to access the DB. These patterns have helped us ensure data consistency and reliability by providing a single point of access and control for all database operations.

Using Swagger was critical to the implementation of the NFT token API application, as Swagger provided a user-friendly interface for the application, which made it easy for us to test and interact with the API. Swagger helped us automate the generation of documentation for the API, which made it easier to maintain and update the documentation as the application evolved. The CORS mechanism was used to provide access to the API for clients from different domains. All this was achieved by setting appropriate headers that allowed cross-requests to be made to the API. The mechanism helped improve the usability of the application by allowing clients from different domains to easily access and use the API.

Thus, we have combined various technologies and skills, in particular the technology stack listed above, to create an NFT token application. We have worked for the coherence of the technology and solving problems related to asynchrony and multi-threaded processes, as a result, we have implemented an application marketplace.

REFERENCES

[1] M.J. Price, “Apps and Services with .NET 7: Build practical projects with Blazor, .NET MAUI, gRPC, GraphQL, and other enterprise technologies,” Packt Publishing, 2022.

[2] J.P. Smith, “Entity Framework Core in Action,” Manning, 2018.

[3] “Unit of Work in Repository Pattern,” [Online]. Available: https://www.c-sharpcorner.com/UploadFile/b1df45/unit-of-work-in-repository-pattern. [Accessed: October 16, 2023].

[4] A. Shvets, *Dive Into Design Patterns: An ebook on design patterns and the principles behind them*. [Online]. Available: https://refactoring.guru/design-patterns/book. [Accessed: October 16, 2023].

[5] I. Gregorchenko, *Pros and cons of JWT: a brief overview of the intricacies of this technology*. [Online]. Available: https://highload.today/jwt-auth-json. [Accessed: October 16, 2023].

[6] J. Trivedi, *Introduction To NLog With ASP.NET Core*. [Online]. Available: https://www.c-sharpcorner.com/article/introduction-to-nlog-with-asp-net-core2. [Accessed: October 16, 2023].

[7] S. Pochekutov, *Swagger: what is it and how to work with it?* [Online]. Available: https://highload.today/swagger-api. [Accessed: October 16, 2023].

[8] J. Albahari, “C# 10 in a Nutshell: The Definitive Reference,” O'Reilly, 2022.

[9] “.NET Technology Stack,” [Online]. Available: https://www.qat.com/net-technology-stack/. [Accessed: October 16, 2023].