info@apriorit.com
www.apriorit.com

Case study

Developing a Decentralized Asset Market on the Tezos Blockchain

A US-based company delivering tools and services for blockchain-based financial products approached Apriorit with a challenging task — to develop a Tezos-based market for crypto assets. Our client wanted to validate the idea of a crypto asset market and finalize their product vision. We created and tested several smart contracts to help our client determine what capabilities to implement next.

We also created automation scripts enabling our client to quickly deploy new asset markets and check the impact of minor platform changes.

The customer

Our client is a US-based company delivering smart contracts, crypto wallets, and other tools for blockchain-based financial solutions. Previously, Apriorit blockchain experts performed a security audit of one of their blockchain solutions. Satisfied with our level of expertise and the quality of results delivered, the client entrusted Apriorit with building a Tezos-based platform for their new project.

The challenge

The client asked us to create a Tezos-based marketplace with decentralized governance on which users could freely and securely borrow and lend crypto assets.

The client had a general idea of the market's concept and wanted to know more about the technical requirements. In contrast to legacy financial services, this market was supposed to be operated not by a designated entity but by a set of automatically executed smart contracts. These smart contracts would contain key platform rules, interest rates, and the mechanics for all crypto asset–related operations.

As a reference for their new project, the client suggested using the Compound project — a popular asset market based on the Ethereum blockchain.

The result

The Apriorit blockchain development team analyzed different options for crypto asset market implementation to finalize a concept suitable for the Tezos blockchain. Then, we designed the platform architecture and developed a blockchain asset market's test version containing only core functionality and contracts.

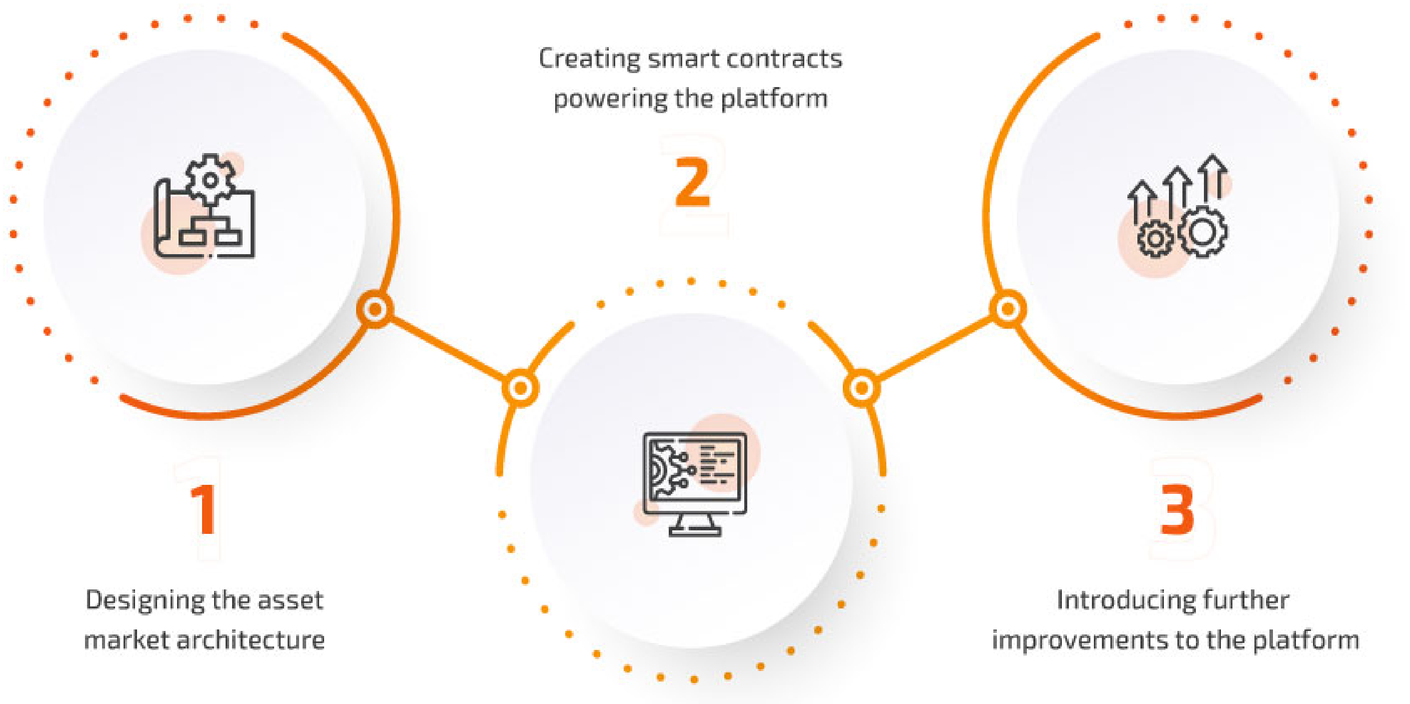
This test version of the Tezos-based asset market will help our client evaluate their overall idea and determine what features and capabilities to implement next.

Our approach

Apriorit formed a team of blockchain developers experienced in building blockchain solutions and writing Tezos smart contracts. Our team started with analyzing the client's requirements and suggested reference projects. Through ongoing communication with the client, we determined the project's initial technical requirements and began creating a crypto marketplace from scratch.

This project went through three major stages:

MAIN STAGES OF ASSET MARKET DEVELOPMENT


Creating smart contracts powering the platform
2
1
Designing the asset market architecture
3
Introducing further improvements to the platform

Let's see how Apriorit's blockchain specialists worked at each of these stages.

1. Designing the asset market architecture

We decided to start with creating core smart contracts for markets operating with different tokens, coins, and interest rate models. Specifically, we implemented support for Tezos coins (XTZ) and two types of token standards:

- FA1.2 — Approvable Ledger Token
- FA2 — Multi-Asset Token

This will later enable our client's asset market to efficiently work with a rich variety of tokens.

However, due to some workflow peculiarities and limitations of Tezos, implementing the core functionality set turned out to be challenging.

2. Creating smart contracts powering the platform

Since our client wanted to develop a decentralized asset market, all business logic and operating rules for this platform were encoded in smart contracts. All smart contracts needed to implement the core functionality of the crypto asset market were created in five steps:

IMPLEMENTING ASSET MARKET SMART CONTRACTS IN 5 STEPS

1. Write the code for smart contracts powering the platform
2. Cover smart contracts with unit tests
3. Prepare smart contract documentation
4. Deploy ready smart contracts to the testnet
5. Test the deployed smart contracts

While Tezos supports several smart contract languages, the client wanted their smart contracts to be written in SmartPy.

PROJECT'S TECHNOLOGY STACK

IDE	Smart contract language	Smart contract deployment library
VS Code	SmartPy	ConseiUS

Also, compared to Ethereum, Tezos has a number of technical limitations and peculiarities affecting the development and operation of smart contracts. For example, when working with Tezos, you can't simply get the return value from a smart contract you send a request to. To get a response from a smart contract, you need to use a callback mechanism — call the target contract with a request and that contract will call you back with an answer.

Another network-related challenge was the limited operation size. In Tezos, you can't submit an operation larger than 32 kB. To address this issue, we had to split large operations into sequences of smaller operations and execute them in a batch.

Once we had prepared the core set of smart contracts, we moved to the testing stage. Full coverage of smart contracts with unit tests allowed us to detect and fix minor code issues. It also helped us emulate different scenarios of user interaction with the entire platform despite having a limited set of smart contracts ready.

After that, we successfully deployed ready smart contracts to the test network and tested them again. For each smart contract, our specialists prepared detailed documentation: a description of smart contact methods and error codes along with guides and manuals for contract deployment and maintenance.

With these smart contracts developed, our client could now outline possible improvements to core functionality and decide on new capabilities to be implemented in their asset market.

3. Introducing further platform improvements

As the created asset market contains only basic functionality, our client will surely need to improve its performance and implement new features. This requires being able to quickly introduce minor changes to the platform and check their impact on existing functionality.

To simplify platform maintenance for the client, Apriorit's blockchain developers created several scripts to speed up and automate such tasks as:

- Deploying new smart contracts to the platform
- Testing changes introduced to the platform

Using these scripts, the client can significantly speed up the deployment of new features and make sure they don't interrupt the operation of previously implemented functionality.

The impact

Lack of a clear vision and finalized technical requirements shouldn't prevent ambitious projects from happening. With a test version of their crypto asset market, our client can now evaluate core functionality and better understand their vision for further platform improvements to build a useful and competitive product.

Want to build a blockchain-based marketplace or check the security of your smart contracts? Delegate these tricky tasks to Apriorit's blockchain experts!