info@apriorit.com
www.apriorit.com

Case study

Decentralized Blockchain Application for International Remittance Payments

Our client is a financial services provider. They were launching an international remittance payments project based on blockchain. Before contacting Apriorit, they created an Ethereum-based solution, but it wasn't fast and flexible enough to meet clients' needs.

To bring clients' vision to life, the Apriorit team designed a smart economy, created an Ontology-based network, and developed a DApp, web wallet, and Android application.

Client: <NDA protected name>

Our client is a financial services provider with strong demand internationally. They wanted to create an innovative solution to facilitate global digital payments using blockchain technology.

The challenge

Our client wanted to start a blockchain-powered remittance service that could support a high volume of transactions.

Originally, they had a token based on Ethereum, but it had a number of issues:

- **Lack of flexibility.** The smart contract was difficult to update once it was deployed to the Ethereum network. Our client couldn't adjust the solution based on changing requirements.
- **Slow operations.** The Ethereum network has a capacity of 10 to 30 transactions per second because of the network workflow and inefficient Proof of Work consensus. Confirming transactions could take up to several minutes.
- **High transaction price.** Executing a smart contract cost our client around $0.02 to 0.05. But the cost could get as high as $0.10 if the network was particularly busy.

They needed to build a brand-new blockchain-based cloud solution to replace this system and entrusted the task to Apriorit.

Our solution

To meet our client's needs, we designed a smart economy and developed a new Ontology-based network, web wallet, decentralized application (DApp), and Android application.

The result

We created a solution with a maximum capacity of around 3,000 transactions per second. Transactions in our solution cost less than $0.0001 each.

Moreover, fees are compensated for transactions generated by the DApp, meaning DApp users don't pay anything. A fee is charged only for personal transactions between users.

Project summary

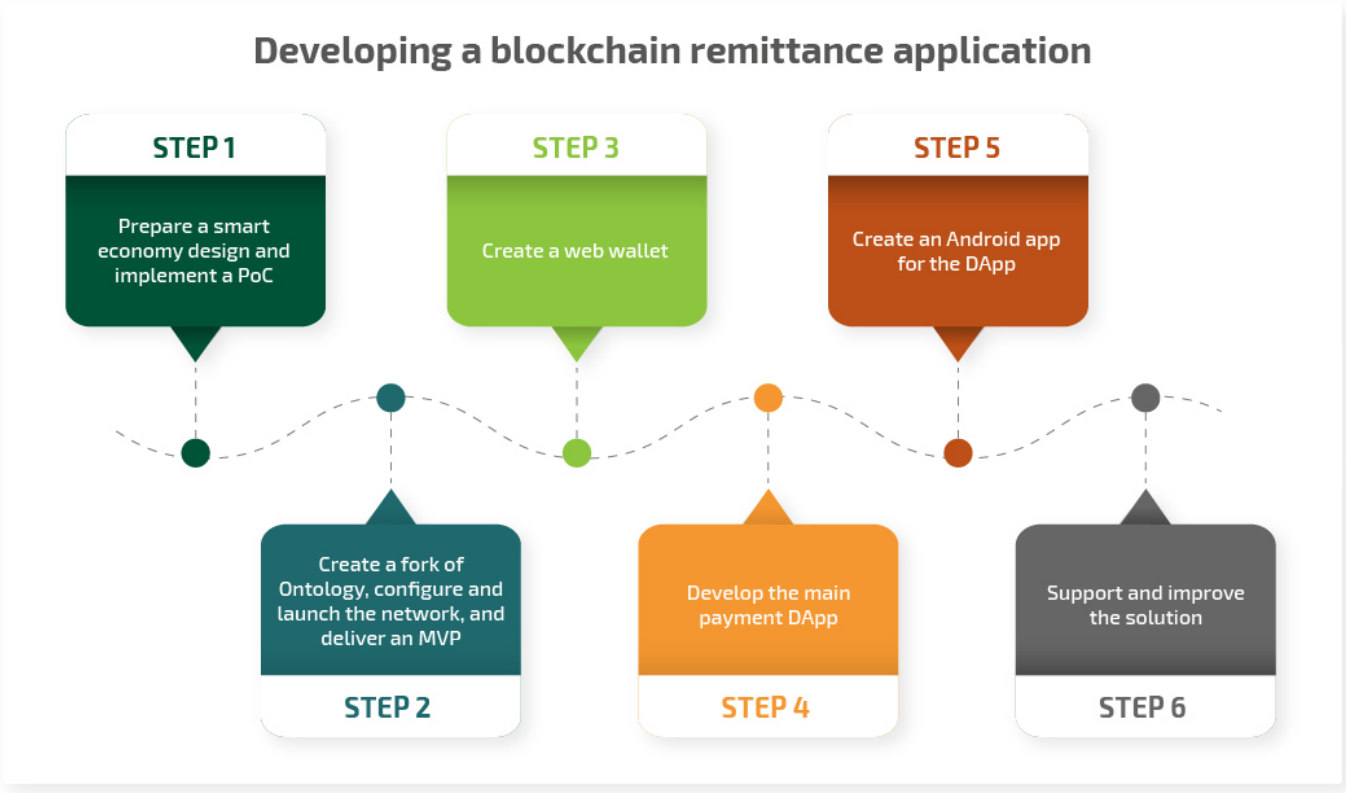
Skills we used	Technology stack	Project team
Project management	Ontology blockchain network	Project manager
Business analytics	Go	Business analysts
Blockchain development	Node.js	Blockchain developers
Mobile development	React	DevOps engineer
Web app development	Python	QA specialists
DevOps	Amazon Web Services	
Quality assurance		

Scope of work

At the early stages of the project, Apriorit experts prepared and discussed with the client several possible scenarios for implementing this blockchain-based remittance service. Instead of developing a new network from scratch, we offered to create a fork of an existing open-source solution. This allowed us to speed up work on the project.

We decided between the Ripple and Ontology networks as a starting point for development. Ripple has a lot of built-in functionality to facilitate large-scale payments and features a robust B2B payment solution. However, the Ripple network lacks functionality for currency transfers and has a high transaction throughput.
That's why we decided to use Ontology. This network is more flexible, allowing you to add custom business logic to your solution using smart contracts. As a result, our developers could easily expand the network's functionality. For example, as per customer demand, we added a complex reconfigurable fee structure, different user roles, and complex user interactions.

After discussions, we divided the scope of work on the project into six phases.

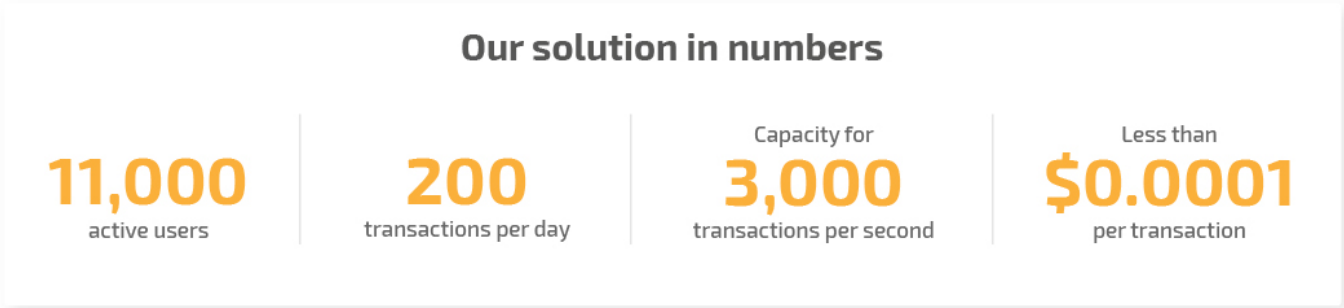


All of the network components we used during development were either based on open-source projects that we modified to fit our client's requirements or developed from scratch by our team. To deploy our solution, we used the Amazon Web Services (AWS) cloud infrastructure.

Our approach

We consistently communicated with the client during the project, informing them of progress, the project status, and planned tasks. Assigning a dedicated project manager is part of the Apriorit delivery process. It makes our projects more transparent and manageable.
To keep our client up to date, during each phase we:

- prepared the scope of work and discussed it with the client to identify new requirements and set priorities
- prepared detailed specifications and technical solutions for each new component and feature
- estimated the scope of work based on these specifications
- divided large development cycles into sprints, allowing us and the client to review progress and make changes as necessary
- performed confirmation and regression testing after each phase of development to ensure that the system and all its components worked as expected
- provided weekly reports on the status of our work



Phase 1: Create a smart economy and PoC

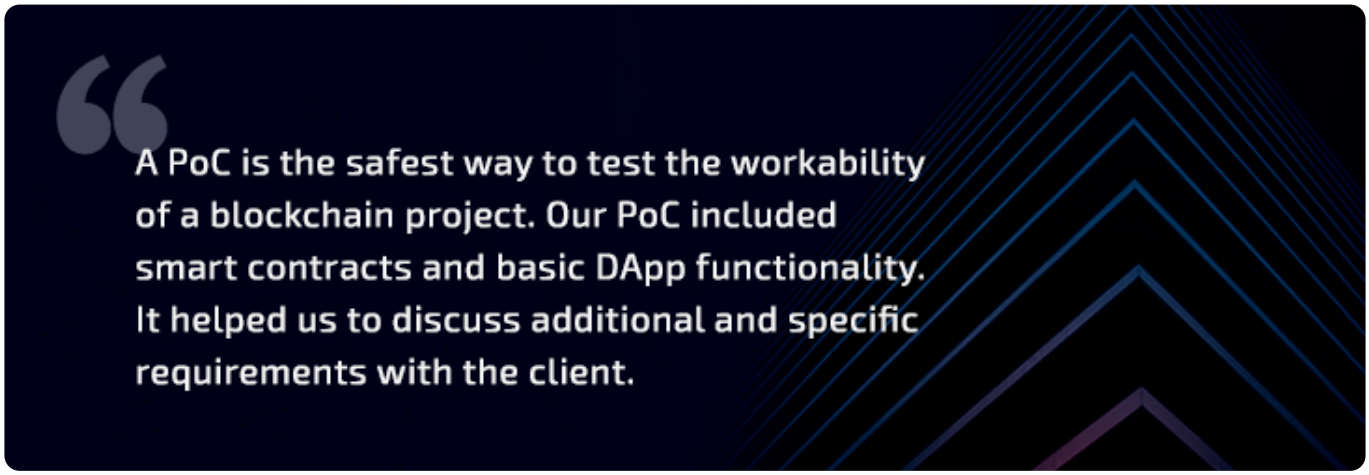
We started with our lead blockchain expert preparing a **smart economy design** for a blockchain application for remittance. The term smart economy has a lot of definitions, but in terms of our project, we use it to describe interactions between users and the flow of currency through the system. Our smart economy design allowed us to go through the application workflow in detail before development.

The smart economy provided a good starting point for this project. During several discussions with the client, we refined the design and established additional requirements such as fees and referral rewards.

Next, we created a **proof of concept (PoC) implementation** of the smart economy payment system. Creating a PoC is common practice in blockchain projects, as it allows you to verify

the efficiency of the app's core idea. Our PoC included smart contracts and a basic DApp that we demonstrated to the client.

This useful practice helped us to identify new requirements that weren't discussed before and to be on the same page with the client. Also, the PoC allowed us to discover potential flaws in the system at an early stage. These flaws were fixed during product implementation.



Our team prepared smart economy specifications simultaneously with other development tasks, as the discussion of the details didn't affect more general tasks like creating a blockchain network.

Phase 2: configure and launch the network

After finishing the smart economy specification, we focused on delivering a minimum viable product (MVP). An MVP is an early version of an application that contains enough features to satisfy the first users. It helps us get real feedback from customers before development is finished.

Our MVP contained cryptocurrency that could be created and distributed to investors. Later, this cryptocurrency was used to power the blockchain-based application, including the main payment DApp.

A large part of the second phase was dedicated to **creating the core of our system — the blockchain network**. At this stage, the team created a fork of the Ontology network. Specifically, they created a fork of the master node implementation written in Go. Ontology network cryptocurrencies (ONT and ONG) were renamed appropriately in the source code and total supply.

We also adjusted the rewards and precision of the coin at our client's request. The network was configured and extensively stress tested. Stress testing proved that the network was stable under heavy loads. Also, this type of testing determines network limits and possible models of failure.

Phase 3: develop the web wallet

The third phase included creating a wallet that allows users to send and receive tokens as well as claim investments. As our team did when implementing the blockchain network, they used an existing open-source implementation — in this case, the Ontology web wallet — instead of starting from scratch. However, we made **extensive modifications to the wallet** to bring it up to our client's requirements:

- Converted the wallet from a browser extension to a web application
- Updated the user interface to correspond with design guidelines
- Added more file export and import options
- Changed the wallet creation procedure to ensure that users save their private keys and backup phrases

Also, we added **investment claiming functionality**. Investors can purchase tokens using fiat money (e.g. USD) and claim an equivalent amount of crypto in return. This functionality allowed our client to launch and complete their initial coin offering (ICO).

Additionally, we added the ability to purchase utility tokens used for paying transaction fees.

Phase 4: develop the payment DApp

During the fourth phase, our goal was to launch the main DApp that features blockchain-backed currency transfers, even with fiat currencies. We achieved this goal by using stablecoins — a kind of cryptocurrency tied at an approximately 1:1 ratio to a certain fiat currency. Users can receive stablecoins by exchanging fiat money through agents.

Agents act as gateways for DApp users who want to deposit or withdraw fiat money into the system. In exchange for fiat currency, users receive stablecoins, which can be used for digital transfers.

At this stage, most of the smart contracts were already in place thanks to the PoC. Our team **improved smart contracts** to ensure the security and stability of the system.

Security is the key reason people trust blockchains, so we needed to make sure our smart contracts held up to the challenge. To ensure that, we developed smart contracts keeping in mind our own vulnerability database. Also, it's good practice to perform penetration testing before implementing smart contracts.

In this project, we added an escrow feature to increase trust between agents and users. This feature allows both parties to deposit and withdraw currency without requiring mutual trust. Also, we used smart contracts as the on-chain backend for the web-based DApp.

The DApp was written using **Node.js and React**. Our application is compatible with wallet files exported from the web wallet, so users can create a single account for the web wallet and DApp.

In order to use our client's solution, a user needs two accounts:

1. An anonymous blockchain account to perform transactions inside the network. This account is basically a private key.
2. A DApp account to interact with other users. When a user registers in the DApp (which can be done via a Telegram bot), this account is linked with the relevant blockchain account.

Registered users can transfer or exchange stable fiat-backed currencies and withdraw or deposit these currencies with the help of local agents.

When we developed the PoC, our team noticed performance issues with the DApp. For instance, accounts with a large number of transactions took a long time to load because of

constant communication with the blockchain network. This could be a serious issue for an international financial project.

To improve DApp performance, our software engineers cached blockchain data in a database. All transactions are still performed on the blockchain, but read operations are performed using a database that mirrors the blockchain's state. This solution allowed us to **decrease the load on the blockchain network** while greatly improving the responsiveness of the wallet.

Another important activity at this stage was the **expansion of cloud infrastructure** on AWS. By this point, the project required over 40 servers and two Amazon Relational Databases. A DevOps engineer orchestrated AWS resources and set up continuous deployment. We also improved security by adding policies and access management.

Phase 5: create an Android app

By this stage, users were actively using the application, and we started to receive feedback from both end users and our client. This feedback led us to create an Android app to provide convenient access to payments.

The **Android app** is a lightweight wrapper of the web version of the DApp. It behaves like a native Android application and can be downloaded from the Google Play Store. Under the hood, it presents the web version of the DApp.
This approach allowed our team to add a convenient app without spending too many resources building it from scratch. For the client, having an Android app makes their application more comfortable for end users.

Phase 6: support and improve the solution

The final phase was dedicated to **updating and improving the platform**. In particular, we:

- simplified account management in the DApp
- updated the fee structure of the DApp
- added a block explorer for blockchain history analysis and security monitoring

The block explorer helps users and the client analyze the blockchain transaction history and check for any suspicious activity in the network.

Apart from the main development process, our team performed some additional tasks requested by the client:

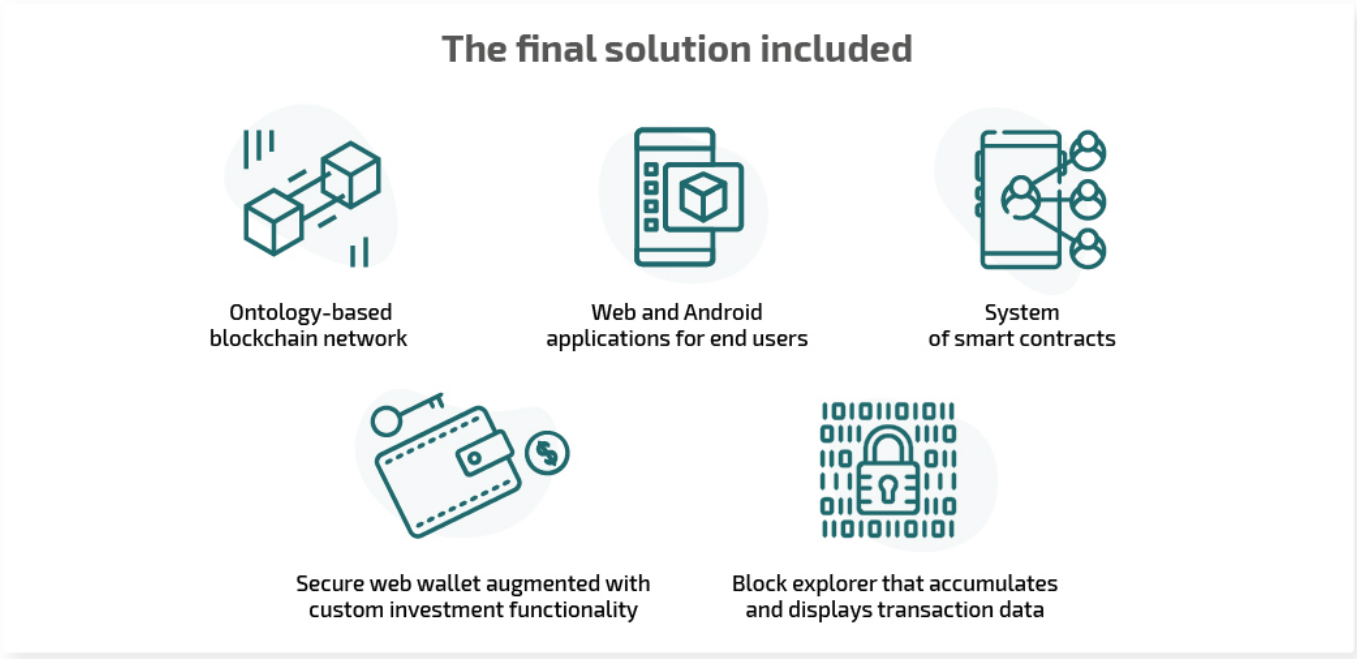
- Supporting and maintaining the system and its components, fixing bugs, and making small adjustments

- Writing support materials such as user documentation, community blog posts, white papers, and guides for third-party developers
- Providing investment reports
- Adding component monitoring
- Performing an in-house security audit of blockchain components

These tasks weren't included in the overall scope of the project, but they were necessary to provide a great service for end users. They also helped us ensure the security and stability of the system.

Project results

Our client acquired a complex remittance blockchain platform that corresponded with their initial vision of the project. The proof of concept helped us identify potential issues in primary requirements. Then we proposed a smart economy and discussed it prior to development.



Throughout the whole process, our client was closely involved in development. Thanks to our flexible delivery process, we easily made adjustments to project requirements on the fly. Our team was not only responsible for developing the system but for writing technical and user documentation.

Our QA engineers ensured the system was stable. Constant monitoring and support guaranteed that it's robust and secure. As a result, our client acquired a solution that meets their needs in terms of flexibility, functionality, operating speed, and transaction cost.