Anatomy of a Token Platform

Justus Perlwitz

2018-03-20

Contents

I had the tremendous privilege of speaking about creating sustainable token platforms at TokenSky 2018 in Seoul, Korea. Not only did I have a great time presenting some of the ideas that I've come across, I also encountered a lot of stimulating questions from individuals and companies all interested in enhancing their platform using some kind of blockchain token.

This post builds on those questions and experiences to identify some of the things you should consider when tokenizing your application. There are roughly six topics.

- **The Token**: What kind of token do you need?
- **The Purchase**: How do users get access to your tokenized platform?
- **The Wallet**: Where are your user tokens stored?
- **The Smart Contract**: Which parts of your application can run on the blockchain?
- **The Backend**: How will your blockchain application integrate with the rest of your platform?
- **The Exchange**: How can users trade your tokens?

1 1. The Token

First things first. When you create your token, you should be very clear on what it represents. Is it a currency users can exchange for goods and services? Or does it represent unique assets users can purchase?

For example, you can replace your in-game currency with a token. A user's gold coins, formerly stored in a central SQL database, now become something they associate with a sense of ownership.

Customer loyalty programs — such as bonus miles — are similar. Data you used to store on your backend can now be in the hands of your customers. A bonus mile becomes a token mile.

On the other hand, if your user assets are indivisible — representing individual assets in the real world or in the virtual world — then you should consider representing them as "non-fungible tokens". A non-fungible token assigns a unique ID to assets. Users can exchange these IDs for other tokens on an open market just like ERC20 tokens. A good example for this is the CryptoKitties marketplace, where users can trade unique, virtual cat avatars.

The advantage with non-fungible tokens is that a customer who owns assets — such as real estate — gets a one-to-one token equivalent, stored securely on the blockchain. And any user who owns an in-game asset — such as potions in a role-playing game now owns their hard-earned treasures and trade and barter with other players.

2 2. The Purchase

As soon as a user acquires or purchases your tokens, they can start taking full advantage of your platform. So you need to carefully consider how to perfect the user experience. After all, this is the gateway to your blockchain solution. It can define every single interaction that follows.

There are a few questions to consider here:

- How does a user acquire your token?
- What payment methods are available?
- Can the token be purchased as a stand-alone? Or is it tied to acquiring something else on your platform?

Depending on how your platform is set up and whether or not tokens can be directly purchased, there are lots of options available. Among them are three possible token acquisition scenarios:

2.1 2.1. Fiat Payment

Users can acquire your token using a credit card or similar electronic payment services. Tokens will be offered at a fixed exchange rate and users can freely spend them on your platform.

Since users could initiate get a chargeback on credit card transactions, it is important to fully evaluate your risk here.

2.2 2.2. Coupled Purchase

While on your platform, a user purchases an item or a service. Depending on how much they spend, they receive a proportional amount of tokens. As an example, think about tokenizing a music streaming service: Users buy a monthly subscription for $10. In return, they receive 100 Streaming Tokens to use in future transactions. The next time they want to subscribe to the service, they can now use their 100 Streaming Tokens to buy another monthly subscription.

You can even enhance this system by ensuring that the original transaction clears and only giving tokens to the user after a set grace period. That way, you, you lower some of the risk of credit card chargebacks.

2.3 2.3. Cryptocurrency Exchange

The third and most platform-appropriate way of acquiring tokens is by using a currency or token that is native to your platform. In the case of Ethereum, this means users buy your token in exchange for Ether. This can also happen on secondary markets, where users buy your tokens on a decentralized exchange.

Since transactions have to be initiated securely in a blockchain transaction, users have to have access to a wallet and already possess some form of cryptocurrency before they can make purchases.

3 3. The Wallet

Once a user acquires your token, the next question should be where they'll be stored. Technically, there should be no difference between a web wallet and a desktop wallet, as long as the wallet understands the particular type of token you're using and that is sufficiently secure. only the user should be able to control the private key to access the wallet. If you store private keys on the user's behalf, you will make your platform vulnerable to theft of user assets.

You can also simply track how many tokens a user owns and make them available once they want to spend them. In this case you need to maintain your own wallet and ledger and make sure that you're able to fulfill user withdrawal requests.

So the trade-off is making the platform safer at the cost of user convenience. Storing tokens for users makes the single, central wallet susceptible to theft. On the other hand, a central wallet makes it easier to trade tokens. Coinbase is one example of prioritizing trade of perfect user security, since they provide an exchange rather than a wallet.

It's worth considering going with the first option: handing users the keys to their tokens and giving them full ownership. That's the only way you can truly implement all the benefits of a decentralized application on the blockchain.

To sum up, the questions to ask yourself at this point are:

- Where do your users store their tokens?
- Do users own the private key to their wallets?
- How will you secure your central wallet?
- What will the user experience of your wallet look like?

4 4. The Smart Contract

While tokens are useful as proof of ownership, they only reach their full potential when coupled with decentralized applications (DApps). DApps are implemented with smart contracts, which are validated independently in a distributed network. This means that even in an low-trust environment, the smart contract itself can always be trusted. Part of the network's strength here is the fact that anyone can participate and create a network node that verifies all transactions.

If you want to figure out how to use decentralized applications for your tokens, it helps to identify the problems DApps can solve. Typical scenarios include:

- Distributed governance — helping token owners shape the development and functionality of a platform;
- Exchanges or auction platforms — implemented on top of the token, where some sort of automatic fulfillment of the contract is desired;
- Decentralized games where using tokens can minimize cheating.

What the blockchain provides here is an immutable, irrefutable history of all user actions. Everything can be proved with mathematical precision.

Once you've come up with ways that parts of your platform can be implemented or improved with DApps, you should then consider usability. Typical smart contract interactions make use of some osrt of SDK or API — like web3 — that facilities easy interaction between your app developers and smart contracts, using a cryptocurrency wallet, to query to blockchain for values and sending transactions.

Then, once that your smart contract is up and running, you need to evaluate how to continue supporting. A contract will need to be updated eventually, since user requirements change from time to time. Typical solutions for contract migrations involve some kind of forwarder or contract proxy. A sophisticated, efficient migration can be a lot of work.

Some of the questions you need to consider when developing smart contracts on your platform are:

- Should the token allow access to a decentralized application?
- How will users interact with the smart contract?
- Does someone own the smart contract?
- How do you manage smart contract migrations?

5 5. The Backend

In many scenarios, users can access a smart contract or decentralized application directly. Often, it also makes sense to access the application from the backend. An auction can allow users to see bids without having to connect directly to the blockchain in their web browser. Or maybe you want to send out email reminders as soon as an auction is about to time out.

Integrating application backends and decentralized applications requires a certain shift in perspective regarding who *has the final say*. Data on a blockchain is stored with hard guarantees on *correctness*, *immutability*, and *reliability*.

- *Correctness*: If your smart contract operates according to specification, the results stored on the blockchain will always be correct.
- *Immutability*: Changing the data on the blockchain, including transaction data, is almost impossible.
- *Reliability*: Storing the results of blockchain transactions independently on a broad network of equal peer-to-peer nodes guarantees authenticity, thanks to cryptographic hash signatures.

In contrast, data stored on a centralized server instead is fungible. It's easy to amend and delete records. So you need to consider the needs of each type of data you store Will it benefit from being stored in several places at once? Will it need to be changed or deleted?

Typical data that you will want to make easy to change and delete on a decentralized blockchain could include:

- Personal information, including payment data;
- User-submitted content that could be illegal in some of the countries you operate in;
- Data that doesn't require a transaction transaction history, such as caches.

Finally, once you have a clear idea of what data needs to be stored where, you also need to have a clea concept of how the two can interact. If data changes on the blockchain are relevant to your bakckend operations, then the changes need to be properly propagated. Some of the questions you should ask yourself are:

- How will the rest of your platform interact with the token application?
- Should the backend create blockchain transactions that interact with user data?
- How will you keep your backend and blockchain data in sync?

6 6. The Exchange

The final piece of the puzzle is allowing your users to buy and sell your tokens on an open market. Exposing your token to a broader market allows it to arrive at a fair price point. There are many exchanges exist out there, some of which specialize in the exchange of ERC20 tokens on the Ethereum blockchain.

Your users appreciate the transparency and openness that open market exchanges offer. And you don't want your users to feel taken advantage of. Making your token *public* also allows you to acquire more users. Instead of going through your platform directly, users can purchase platform assets elsewhere and start using them immediately.

I will leave you with a few final questions:

- Where will your users be able to trade their tokens?
- Is interoperability with other tokens important to you?
- What are the potential opportunities and risks of exposing your token to a wide market?