DARA: Decentralized Agnostic Recording Assistant

Icewave and Gigamesh
www.icewave.io and www.theimmutable.net
June 28, 2021

Contents

The browser extension

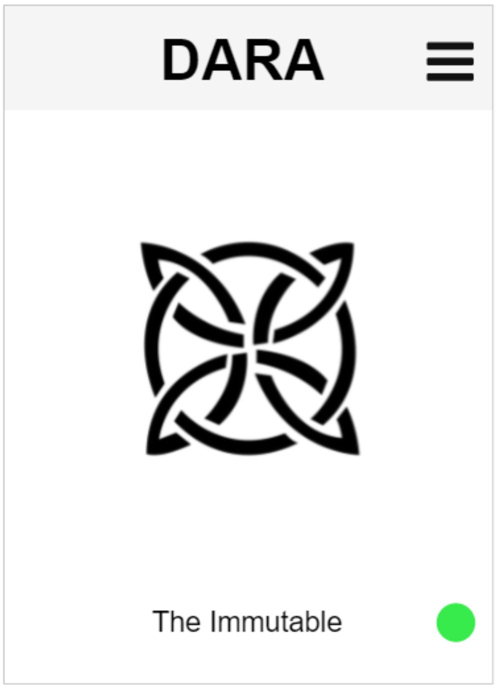
Design

The browser extension will support only Google Chrome for now, more browsers can be supported in the future. The extension will be developed using standard web technologies such as HTML, CSS, JavaScript and JQuery.

A simple client-server model will be used for the front-end communication with the back-end, using a custom-developed REST API.

The extension will only be available when browsing Medium website and to save any blog post content and data into IPFS. More content applications could be supported in the future.

User Interface



The extension will provide the following features:

1. Options Menu
    - Records
        - A list of posts and their IPFS hashes previously uploaded
    - Tutorials
        - Link to The Immutable website
    - Advanced
        - (checkbox) Record the IPFS hashes and data (author, title, etc.) to the Immutable DAO ledger.

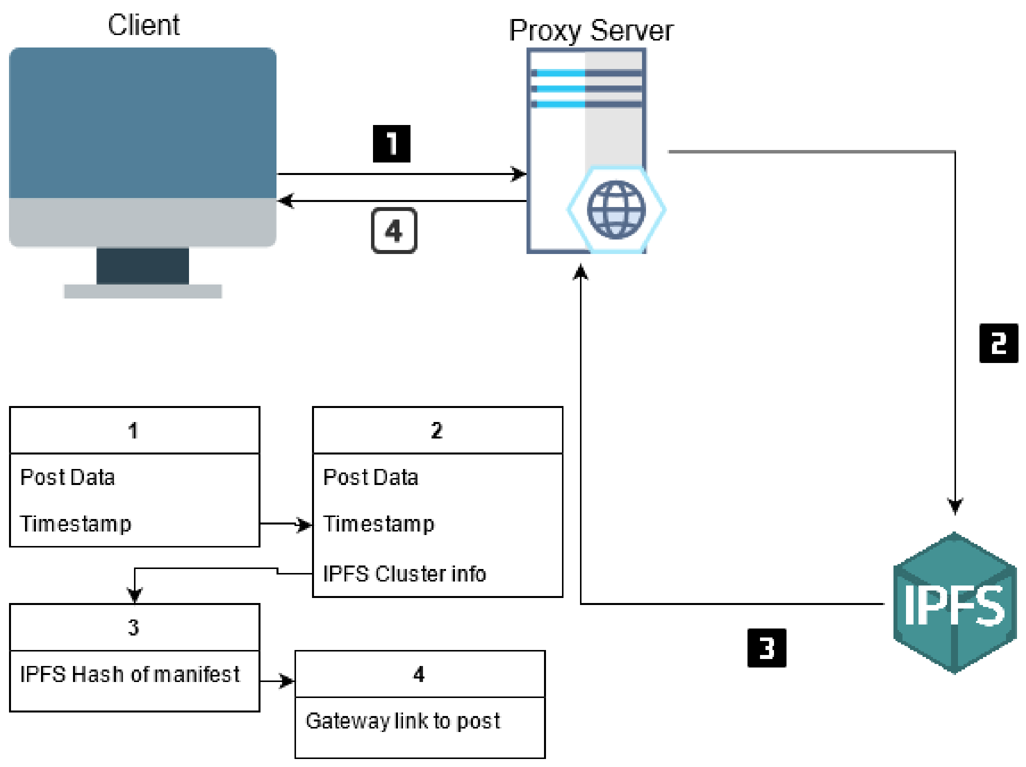
2. Save Button (logo) The main logo will be clickable and used to save the current Medium blog post content to IPFS.

3. Status Indicator
    - Green: Ready to record
    - Yellow: Recording
    - Red: Offline (Server is offline or the Medium website is not open)

API layer communication

In this section there are 3 steps of communications that will happen from within the extension

1. Data is sent to the proxy server regarding storage of data onto IPFS
2. The communication of said data to be a part of IPFS
3. The stored hash of data is communicated back to the proxy server and then this is relayed as a gateway URL back to the client



Backend proxy layer API

API Design

As far as the design goes we will be using a simple server-database model to make the transfer from web2 to web3 as easy as possible.

This will also provide us the ability to help distinguish between users and have a more natural feel to the whole application by letting it be more of a social network that integrates into the old web.

Routes

We will have very few exposed routes

```
/storeData -> For posting data onto IPFS
/fetchData -> For getting data from IPFS for the client
/login -> If the user chooses to have an account with immutable
/signup -> For the user to sign up
/daoSignal -> Signalling for DAO peers in the IPFS cluster
```

Communication layer with IPFS

Since on our server we will be using node-js and mongoDB combo to act as the proxy layer, we will resort to using a go-ipfs daemon with it's HTTP API exposed locally and communicate with it.

Once the data is locally pinned, the manifest hash along with the contents of it are supplied to other nodes in an IPFS cluster, this will be constitued by the people who are members of the DAO.

Ethereum-Binance Smart chain contract

We are comfortable with both Ethereum and Binance smart chain contract ecosystems, so we will be providing the choice to our partner in terms of where they want their application to be hosted.

There are negatives and positives to both case scenarios

- Ethereum has a very low overhead or requirements as far as point of entry goes, where as setting up BSC will cost the user a few minutes if they already haven't something set up.
- Ethereum has much higher fees in comparison to BSC due to how it works.

ERC-20 Features

The smart contract will have an ERC-20 token built into it, which can have all the functions that the client desires including setting the tokenomics as need be. The ERC-20 token can be listed on any of the new generation dex including uniswap/cakeswap/etc etc.

IPFS Hash get-set features

This is the heart of the project, the key value get and set storage function, every IPFS hash will be given a unique UUID, and can be used to be shared or searched from our frontend blogger application.

Fee routing and monetization functions

There could be a small fee that we could take from the users for storage of data, this doesn't have to be the case in the beginning but we can set the functions up in case we want to enable the features later on. This fee can be distributed to the members of the DAO who will be running nodes in the IPFS cluster.

IPFS schema

IPFS Cluster set up for DAO

The DAO members can be elected according to how the client thinks is fair. Setting up an IPFS cluster for them would mean running a specific client on a server remotely, by each of it's members, this will ensure the storage of data in an incentivized manner.

Incentivization for storage of IPFS data for the DAO

This could be active incentivization or passive incentivization (from fees), active incentivization would mean we mint tokens to the DAO members wallets, this could have an emission curve or some other kind of algorithm. Passive incentivization would mean they'd only get money when someone chooses to pay for storage on the DAO clusters.

Conclusion

Immutable's DARA poses some challenges to implement with a lot of moving parts. We will be implementing all of the above features in a proper manner with each step given it's own time. That being said the most important part of this would be the smart contract with the ERC-20 features. Then comes the tools based around it, we reckon we should do the smart contract first then work on the features surrounding it as it makes the most sense and we would have a better idea of where to go with things, development wise.