Uniflux

Technical Deck

Overview

A revolutionary back-end infrastructure to power the next generation of the web. This solution is powered by the Ceramic protocol - a decentralised data warehouse that allows mutability.
We intend to power our Dapp using decentralised identifiers: Here's why.

DIDs as a Solution

- Removes the cost of bias from hiring - a 1% gender bias in a Fortune 500 company is expected to cost upwards of 2.8 million a year - we need a system that promotes meritocracy
- Allows students to prove that they are a student through zk proofs and avoid revealing data such as name and email
- Enable users to own their own data
- Allows their identities to be composable across chain
- Allows us to implement attestations for a particular identity in a way that makes sense

Agreed Standard

DID Technical Stack - Ethereum

Data Registry

The Ethereum Blockchain

DID

Cermaic can be used to store the data on-chain in a ledger style format. These can include profiles, reputation scores and social account links

Agent & Authentication

Metamask can be used with the did:ethr:<your ethereum wallet address> . The agent can prove access to this DID by signing messages with Metamask

Access & Authorisation

Access conditions will require the msg.sender of the original transaction be the one to update the profile section of the document.

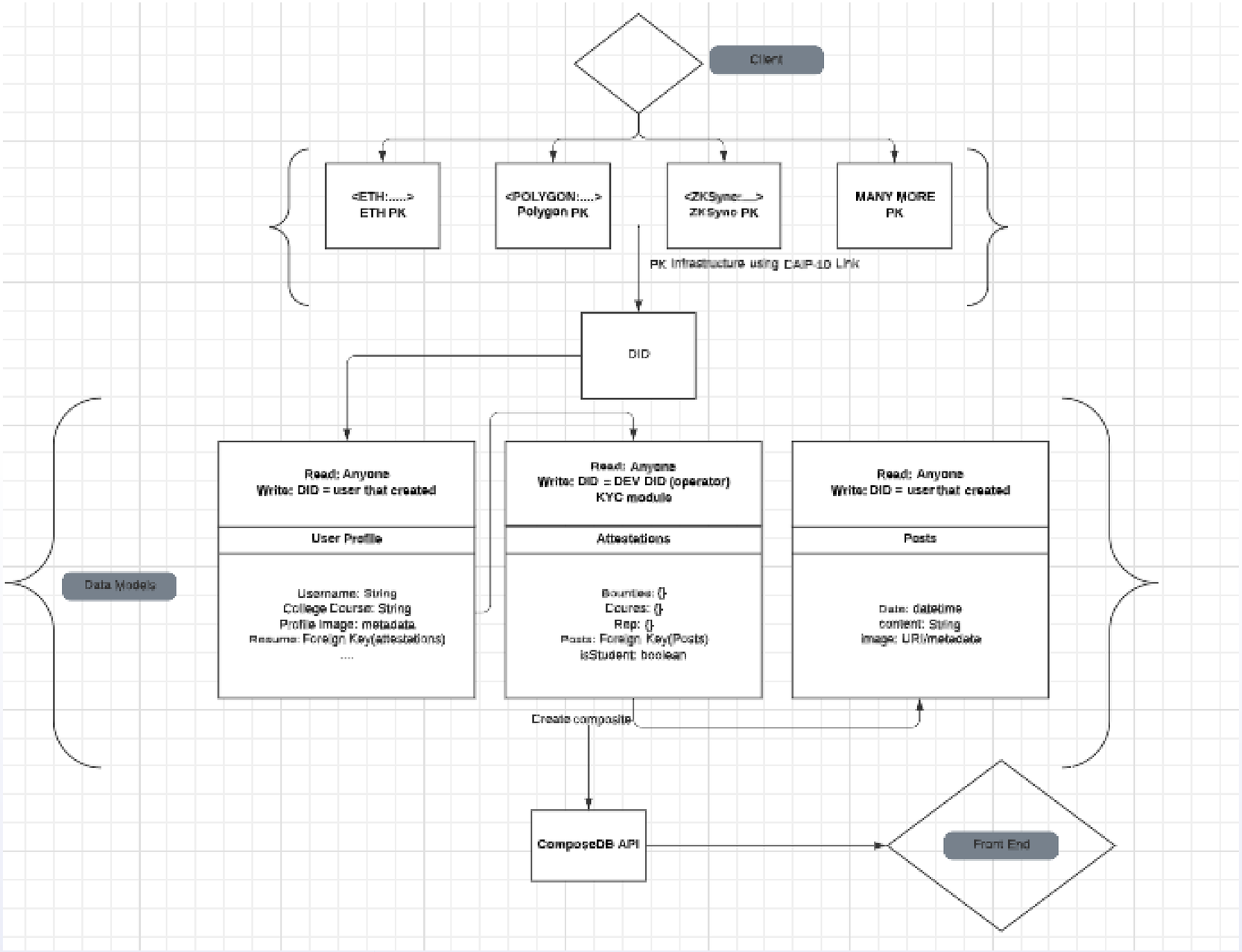
```
{
name: 'Oisin',
age: 19
};
```

However, there will encryption and further access conditions placed on everything else such as attestations and isStudent parameters to prevent malicious activities.

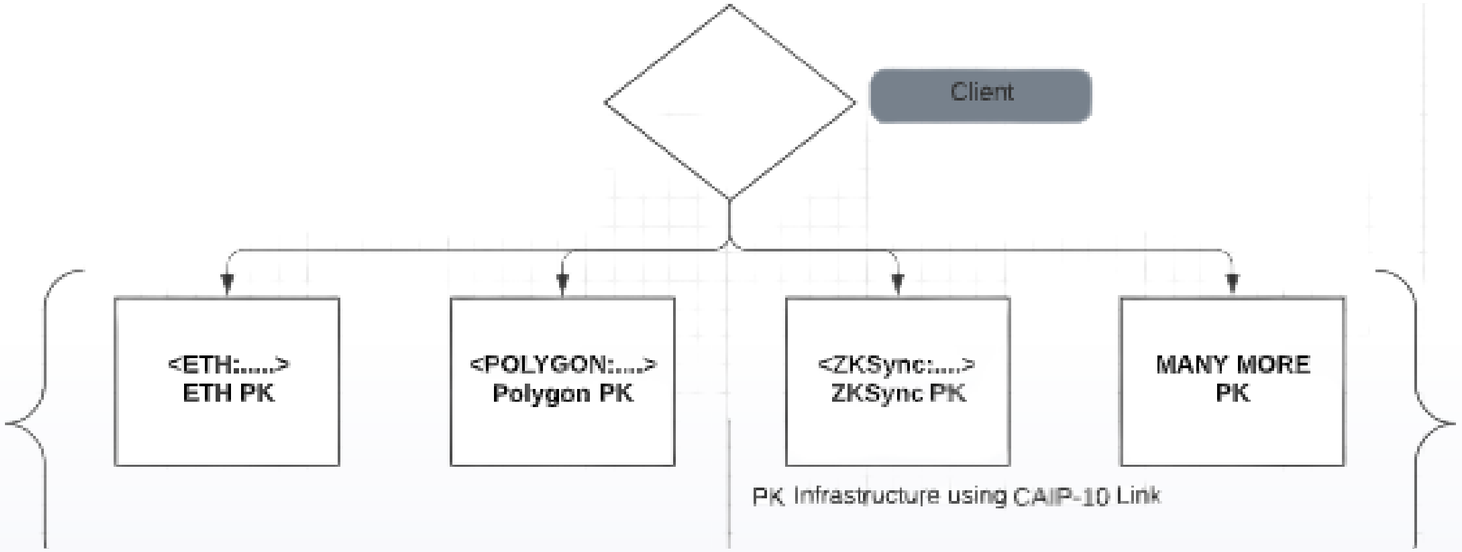
Attestations

These will include the ntNFTs obtained from bounty and course completion. They may also include any posts submitted by the user using the DaoLens API once that has been built.

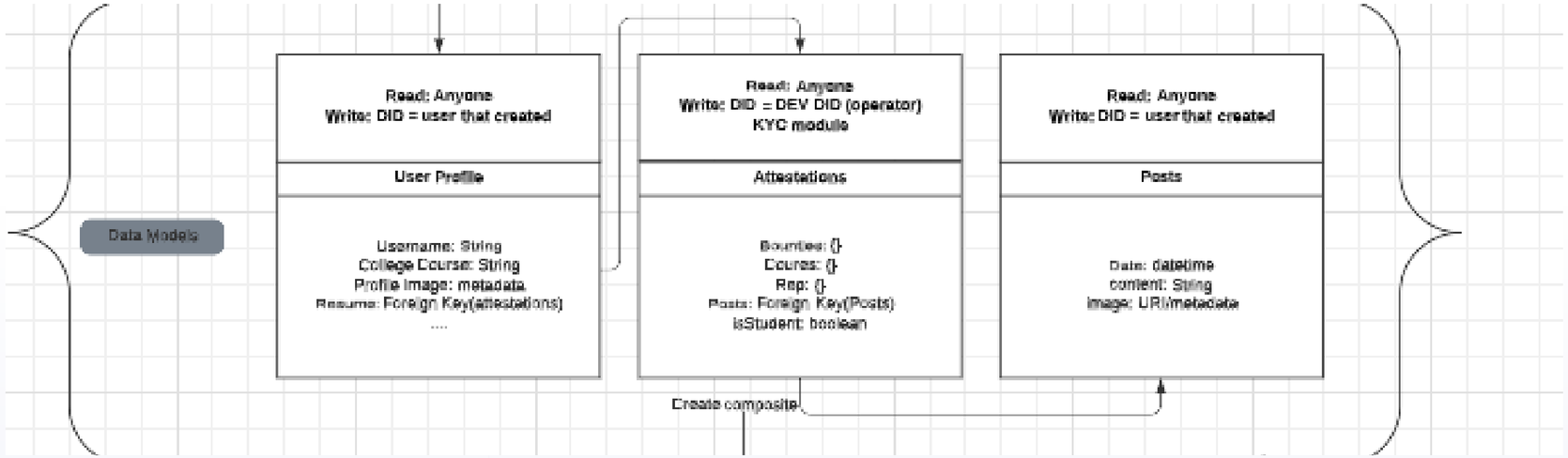
Oveview


Client
<ETH:.....>
ETH PK
<POLYGON:.....>
Polygon PK
<ZKSync:....>
ZKSync PK
MANY MORE
PK
PK Infrastructure using CAIP-10 Link
DID
Data Models
Read: Anyone
Write: DID = user that created
User Profile
Username: String
College Course: String
Profile Image: metadata
Resume: Foreign Key(attestations)
....
Read: Anyone
Write: DID = DEV DID (operator)
KYC module
Attestations
Bounties: {}
Courses: {}
Rep: {}
Posts: Foreign Key(Posts)
isStudent: boolean
Read: Anyone
Write: DID = user that created
Posts
Date: datetime
content: String
image: URI/metadata
Create composite
ComposeDB API
Front End

1. Caip-10 Links



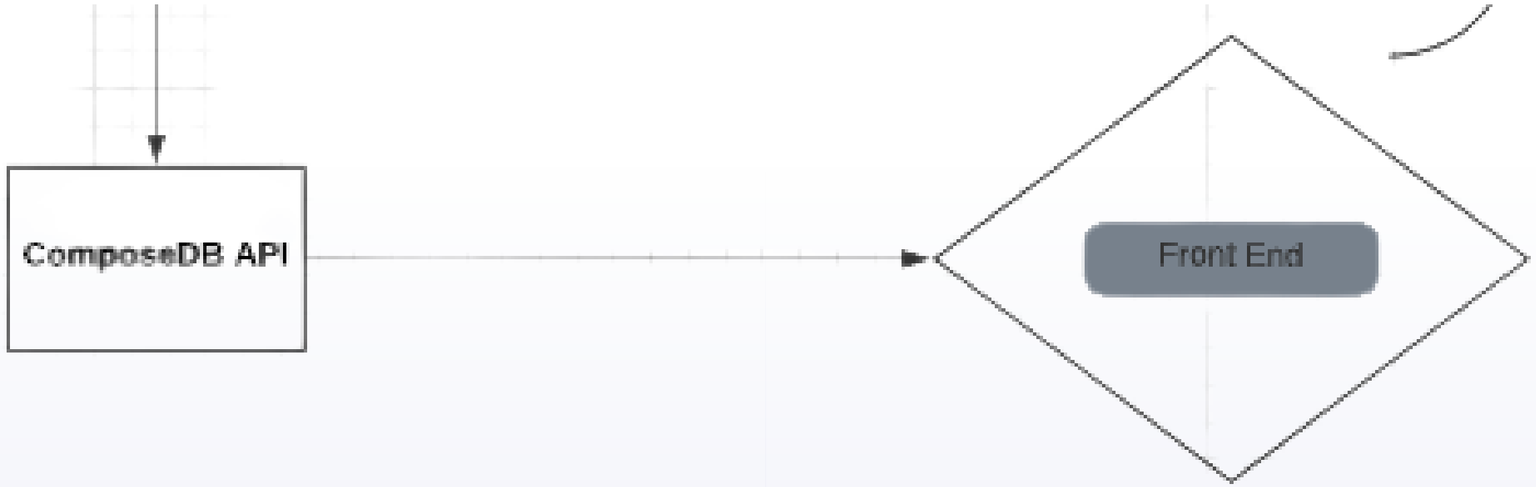
2. Data Models



User Profile Data model: https://github.com/Odhran7/Back-End-simualtion

KYC: https://github.com/Odhran7/KYC-App

3. Querying Composites



Thank you!