Diagram

Description automatically generated

Components

Postgres Database

Redis (Celery message broker and result backend)

Celery (Asynchronous task queue)

Celery beat (Scheduler)

Flower (Celery monitoring tool)

Backend API (Django rest framework)

Frontend Client (React)

Blockchain Data:

Used Blockchain Explorer free API to access balance and transaction data

Database Model:

Resources:

Address and Transaction

Address keeps track of Bitcoin addresses associated with users

Transaction keeps stores transactions associated with

Data Models

Address:

address

transaction\_count

final\_balance

user –> Related Field

address + user is considered unique

Transaction

address -> Related field

hash -> transaction hash

Inputs -> transaction inputs

Out -> transaction outputs

fee -> tx fee

date -> tx timestamp

address + hash is considered unique

Assumptions made to go with this data model

One of the primary assumptions made was that multiple users can add the same address

Which means we store the same transaction twice.

With the above assumption the transactions are belong to an address are accessing using

user + address + hash

API Functionality

Add address

Remove Address

Get All addresses

Sync Transaction for an address

Get All the transactions for a single address which returns paginated response

Limits to 10 transactions / page

Used limit & offset to access paginated results

Tasks performed using celery:

sync\_transactions task fetched all the transactions from blockchain explorer and stores in the database

sync\_transactions\_after\_most\_recent\_tx\_date syncs all the transactions after a certain data.

This is not used in the app. Want to avoid fetching all the transactions during each sync as this is a waste of resource. But ultimately decided not to spend too much time on this.

schedule\_sync\_for\_all\_addresses which is ran every day using celery beat scheduler which schedules sync\_transactions tasks for all the addresses once every day.