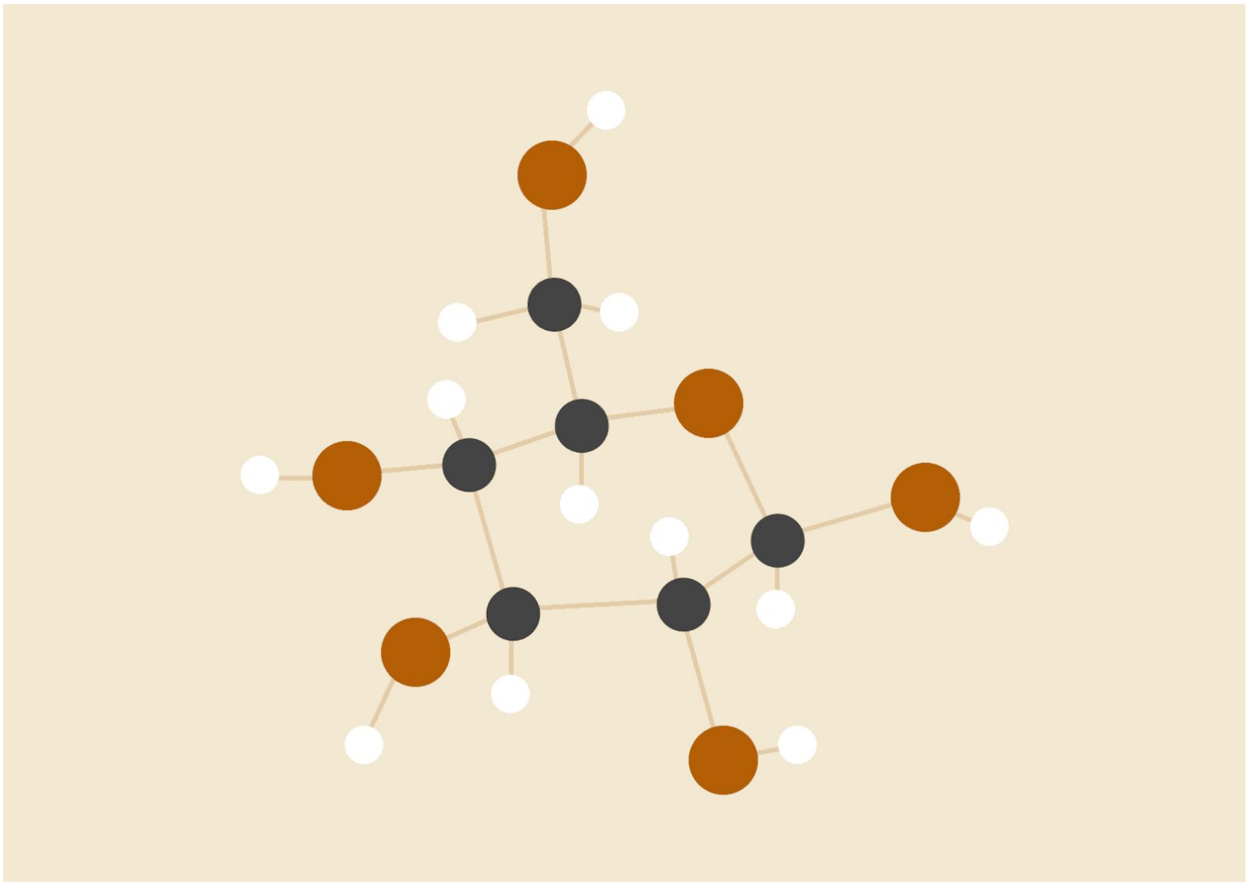
Everse

Project Report



Michele Biondi

21/06/2020

ENTERPRISE SOFTWARE INFRASTRUCTURES

Project Code

The code of the project can be found here: https://github.com/0nism/everse-esi

Introduction

The main goal of this project is to create a new way of financing public projects that have potential revenue streams. Public entities can post new funding campaigns and private investors can purchase tokens that represents a share of ownership of the asset funded.

Each token guarantees a share in the future revenue streams generated directly or indirectly by the asset.

The business logic of the whole process of funding is modeled as an executable business process collaboration.

To improve transparency and trust in the system, Ethereum blockchain is used to track creation and purchase of the various tokens.

Project overview

The project is composed of various web services communicating with each other through a REST interface.

The main reason of this architectural choice is that of flexibility of the system for future developments as monolithic systems pose a competitive disadvantage in the modern world.

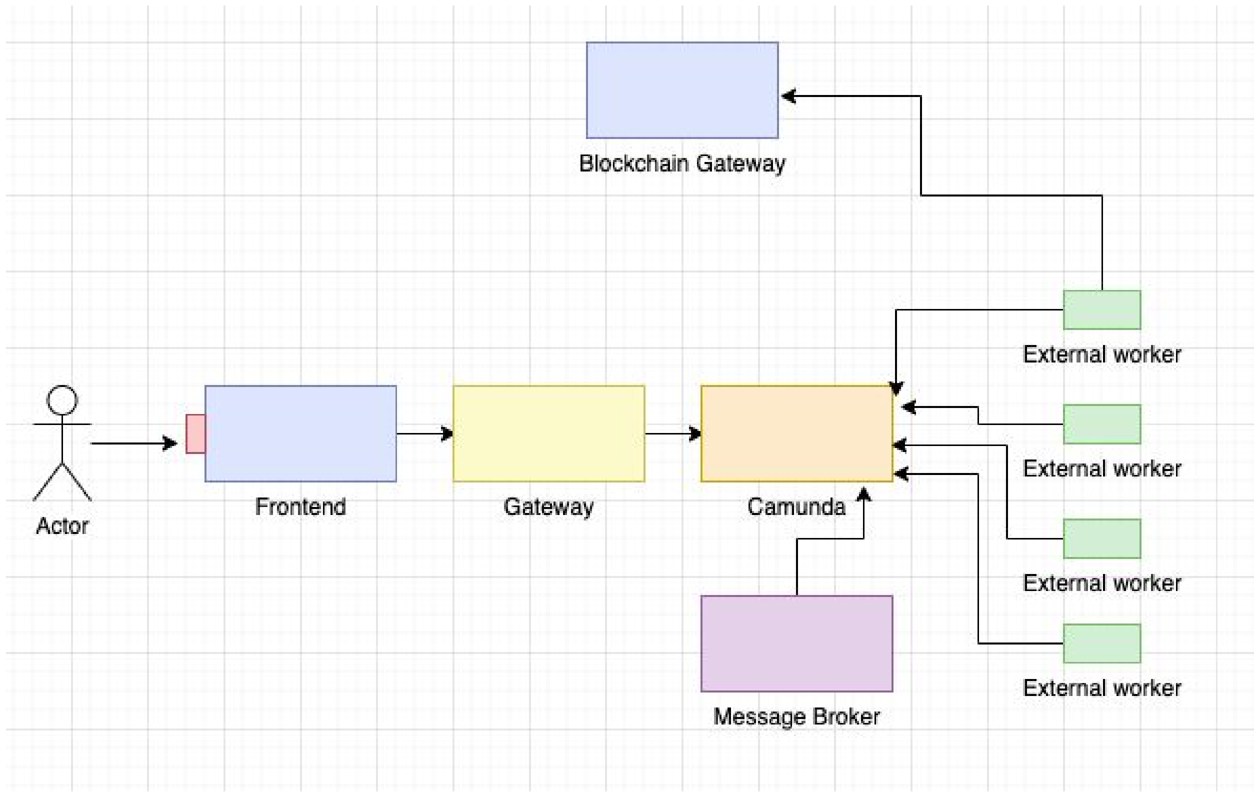
Also the architecture makes it easy to scale the system horizontally with an infinite scalability potential.

The main components are the following:

- **frontend**: SPA built using React, the graphical user interface of everse

- **gateway**: An intermediary between the frontend and camunda that abstracts the camunda REST API details from the frontend and can be used to further improve security.
- **camunda**: the workflow automation and decision platform that implements the man business logic through modeled processes.
- **message-broker**: software portion that deals with the dispatch of messages between camunda processes
- **external workers**: they implement some specific code logic of the processes
- **blockchain gateway**: abstracts the tracking of activities on the blockchain

The following are the relationships between components on the final system:



To store data and contract address for the purpose of the demo it was used a file-based db called NeDb that is 100% compatible with MongoDb and ready to be swapped with it for production.

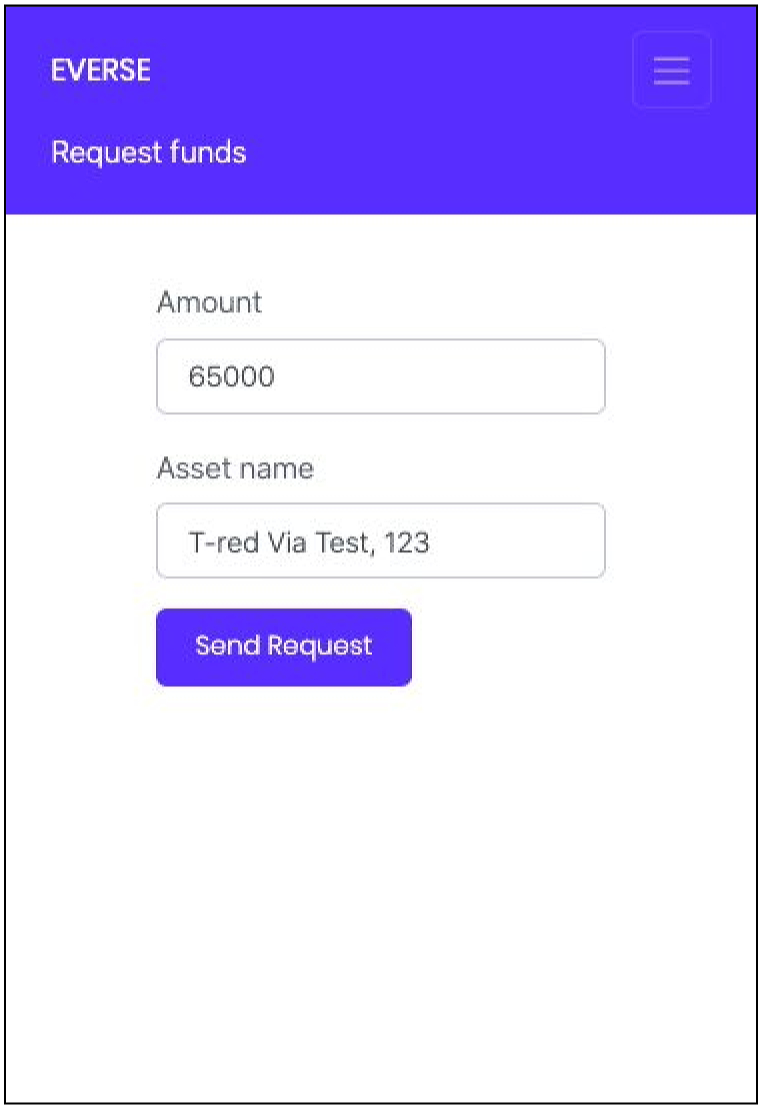
Now we will explore each component in detail.

Frontend

The frontend is a simple Single Page Application written in React that features two main pages.

The home page lists all active campaign and makes possible to buy tokens of the various assets.

The request refund page at /request is a simple form to send a funding request.

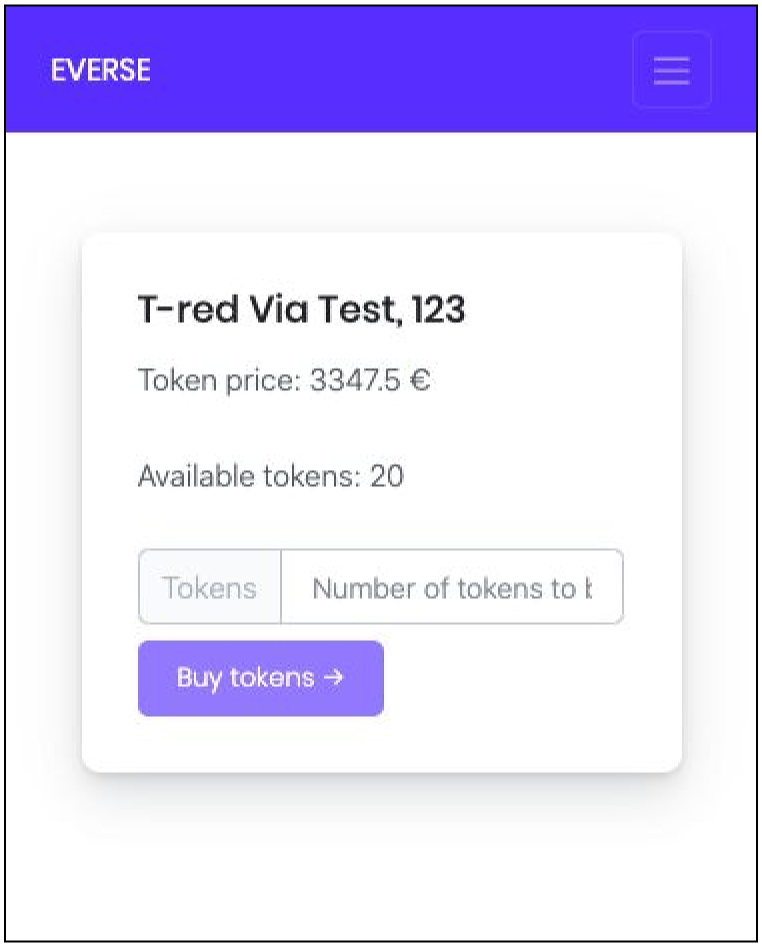


Amount

65000

Asset name

T-red Via Test, 123



Gateway

The gateway is the portion of the project that communicates with the frontend and abstracts away the logic details of communication with Camunda. It is also responsible to fetch data like the current number of available tokens or campaigns available and return them in JSON format to the fronted so that they can be displayed.

The available endpoints are:

- GET /assets : used to retrieve all assets currently available in the system
- POST /tokens : to send token purchase requests
- POST /request: used to send an asset funding request and to start the collaboration

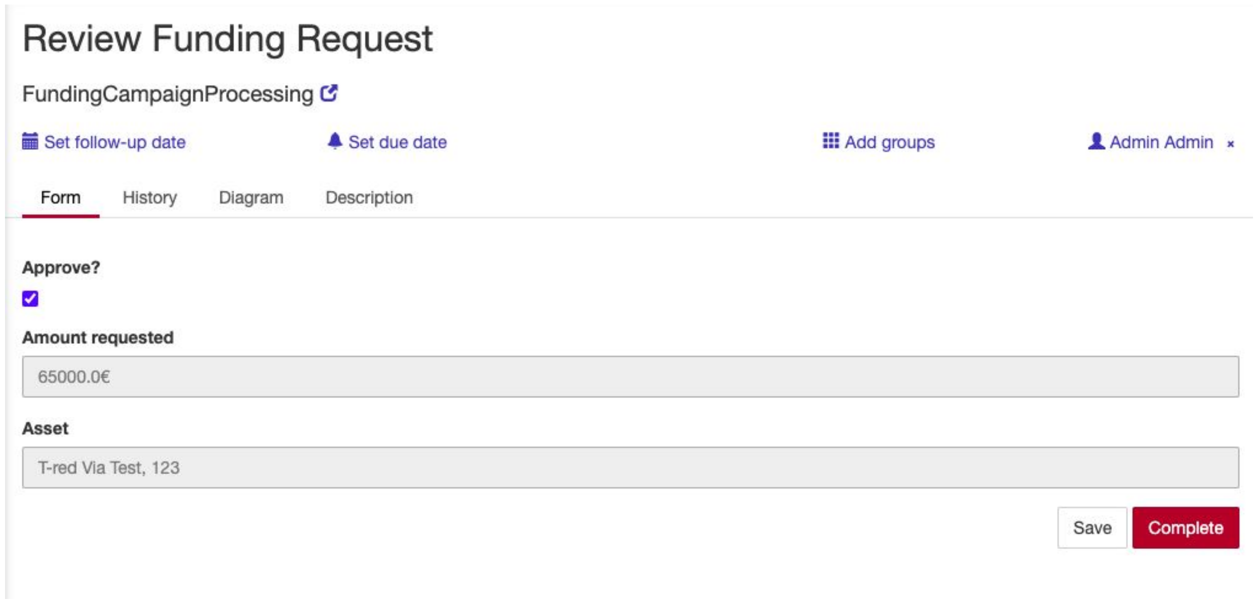
Both POST requests use the Camunda REST API under to hood to forward data to the processes. The gateway is written in NodeJs.

Camunda

The camunda part is implemented using the official camunda spring boot starter (from https://start.camunda.com/) to make it easy to modify camunda platform options.

It also auto-deploys modeled Processes and DMN.

The Camunda tasklist is also used when accepting or rejecting a funding request.



Review Funding Request

FundingCampaignProcessing

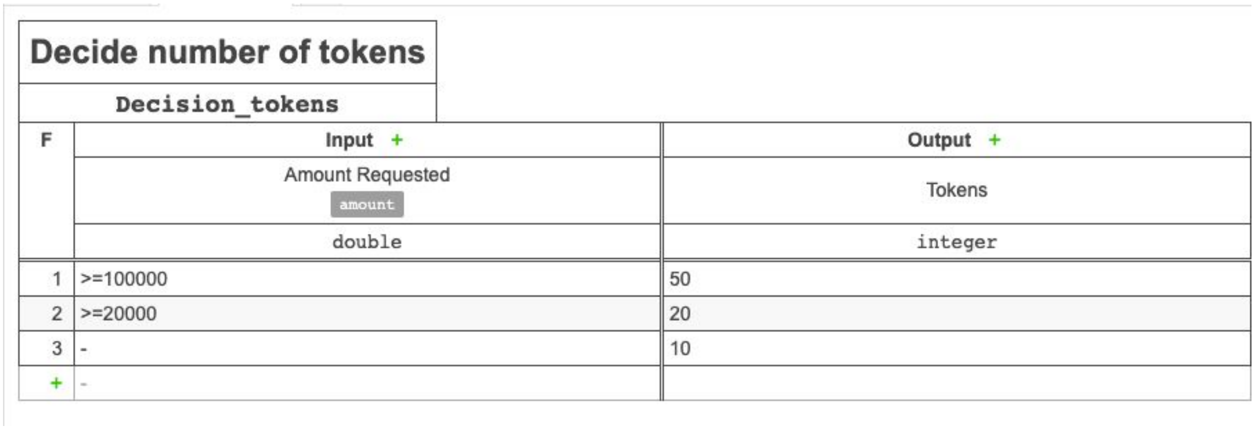
Approve? ✓

Amount requested: 65000.0€

Asset: T-red Via Test, 123

The business processes used in the collaboration can be seen at the following link(image is big): https://imgur.com/a/765y9Zj

The following is the DMN that rules how many tokens will be created for the asset:



Decide number of tokens

Decision_tokens

F	Input +	Output +
	Amount Requested amount	Tokens
	double	integer
1	>=100000	50
2	>=20000	20
3	-	10
+	-	

All the processes regarding the same asset use the same business key, to easily be identified. The business key is also the identifier used in the smart contract that keeps track of the collaboration.

Message Broker

The message broker is the implementation of camunda's messages sent between the business processes. Camunda has various options to implement messaging, the one i chose is by using an External implementation in NodeJs.

External Workers

External workers are used as implementation of services inside business processes. There are a total of 3 external workers used:

- A service to split the asset in tokens, it calculates the price and adds a fee for the platform
- A service to publish and run the funding campaign, it basically stores the asset data inside the database so it is ready to be used.
- A service to register a successful purchase of token(s)

All the services log their activity to the blockchain using the blockchain gateway we will discuss next.

Blockchain Gateway

The blockchain gateway abstract away the interactions with the blockchain by exposing a REST API that can be easily used. The endpoint available are:

- GET /collaborationId/:key : used to generate a collaboration Id to be used as business key
- POST /:id : Used to write activities onto the blockchain of collaboration :id
- GET /:id : Returns all the activities written in the blockchain that are part of collaboration :id

Under the hood the blockchain gateway is written using NodeJs and uses Web3Js to interact with the ethereum blockchain.

As a provider to connect to the Rinkeby network i used Infura and their free API quota because of limited space in the hard drive that made it impossible to host a local Ethereum node.

Future developments

The project is now in a really early state and it's more a demo / proof of concept to showcase the interaction between the various technologies used.

The first thing that it needs is to link a real payment system or maybe to pay directly using Ether or stable cryptocurrencies like DAI. Also when an asset is fully funded it could deploy a smart contract that has methods to receive direct payments and that it instantly distributes the share of profit to the token owners relatively to their quota.

Also the blockchain tracking could be extended to include more activities, even rejected fundings to increase transparency further.