Deliverable 3.3

Services architecture

DEFRAUDify - Detecting Fraudulent activities on the internet

Author: Mathijs Homminga [mathijs@web-iq.com]

Date: 09.06.2021

Status: Final

Version: 1.0

## Introduction

One of the strengths of DEFRAUDify is that the technology partners (TNO, TU/e, Sentinels.ai, CFLW, Almende, Web-IQ and BEIA) leverage their existing tools and technologies and combine their experience and expertise to create new solutions that support the challenging use cases of the end-user partners (Bunq and Hoffman).

The focus in the project is therefor to adapt, innovate and integrate the existing technologies into a collection of interoperable and reusable tools.

The coherence of the DEFRAUDify tools is not achieved by creating a monolithic application, running them in a central place or by putting the code in a central repository. Instead, tools are developed and operated by the partners that already have the expertise and optimal infrastructure in place and synergy is achieved by building on a central and shared ontology [D3.2] that relates all the data in the DEFRAUDify ecosystem and by selecting and shaping the tools to complement and build on each other’s features using APIs.

## Services architecture

### Integration via APIs

The preferred way of connecting tools in DEFRAUDify is by running the tools as a service and integrate via standard RESTful APIs.

Typically following the OpenAPI format:

- Using JSON as request and response payload.

- Providing generated documentation

- Taking into account versioning.

- Transport over SSL.

- Using authentication, like basic auth, OAuth 2 or jsonp.

Each tool provider can decide himself how the API is implemented. This can be as a collection of microservices running in Docker containers, or as a monolithic application running on a main frame. The idea is that each of the tools requires very specific technology. E.g., providing intelligence on a cryptocurrency address, by traversing blockchain data, is different than analysing a single piece of text, using a trained NLP model.

Example of an API integration:



### DEFRAUDify ontology

Note that creating a central ontology means creating a shared definition that describes the data in DEFRAUDify. It does not entail creating and using a shared central database.

This will be elaborated in Deliverable 3.2.



## Annex: functional architecture

As an illustration for the type of services that will link together within DEFRAUDify, the functional architecture of the DEFRAUDify toolset is sketched below. The arrows between the tools indicate data transfer through APIs. This diagram only shows the MVP (Minimum Viable Product) pipelines. More tools and connections may be added when the research indicates that this is possible.

