Funded by the Horizon 2020 Framework
Programme of the European Union
SocialTruth - Grant Agreement 825477

D2.2 Distributed System Architecture, Data Modelling and Interfaces

Dissemination Level:	PU
Nature of the Deliverable:	R
Date:	31/05/2019
Distribution:	Internal
Editors:	UTP
Contributors:	UTP, Thales, ESF, QWANT, Z&P, TECOMS
Reviewers:	ICCS, ESF

*** Dissemination Level:***	*PU= Public, RE= Restricted to a group specified by the Consortium, PP= Restricted to other program participants (including the Commission services), CO= Confidential, only for members of the Consortium (including the Commission services)*
***** Nature of the Deliverable:***	*P= Prototype, R= Report, S= Specification, T= Tool, O= Other*

Disclaimer

Revision History

Date	Rev.	Description	Partner
24/01/2019	1.0	First version of DDP	UTP
28/01/2019	2.0	Final version of DDP	UTP
20/02/2019	3.0	Initial deliverable content (sections 1 and 2)	UTP
08/05/2019	4.0	Pre final version with sections 2 and 5 completed	UTP
15/05/2019	5.0	Version after technical telco	UTP
24/05/2019	6.0	Consolidated version	UTP
28/05/2019	7.0	Pre-final version	UTP
28/05/2019	8.0	Version after review	UTP
30/05/2019	9.0	Final version	UTP

List of Authors

Partner	Author
Z&P	Giulia Venturi
ADNK	Giovanni Parise, Eugenio Lauro
LSBU	Manik Gupta
ICCS	Konstantinos Demestichas, Ioannis Loumiotis, George Koutalieris, Pavlos Kosmides, Evgenia Adamopoulou
ADNK	Giovanni Parise, Eugenio Lauro
UTP	Rafał Renk, Michał Choraś, Rafał Kozik, Marek Pawlicki, Krzysztof Samp, Paweł Ksieniewicz, Michał Woźniak
TECOMS	Guido Villa
QWANT	Stan ASSIER, Noel Martin
ESF	Charles Huot, Sonia Collada
DEASC	Sebastiano Bagarotto

Table of Contents

Index of figures

Index of graphs

Index of tables

Glossary

WP	Work Package
RAG	Red Green Amber
HMI	Human-Machine Interface

Executive Summary

This document (D2.2) exists to provide detailed, functional and non-functional specifications of the distributed SocialTruth Architecture and the related interfaces, taking into account the use-cases and scenarios of T2.1. The document also tackles the technical, security and social & human aspects of SocialTruth. It is intended for the end-user partners, who will deploy the platform, the project researchers and developers, who will be providing the technical solutions, and to the platform integrators. D2.2. produces the outcomes to deliverables D3.1-3, D4.1-3, and D5.2.1. The motivation, scope, relation to other deliverables and the intended audience is fully explained in Section 1.

Section 2 addresses the architectural questions of the SocialTruth platform. The main challenges include scalability, availability, independent and autonomous management, decentralised governance and failure isolation. After careful consideration, a microservices-based solution is selected.

In this section, a general specification of the key components of the SocialTruth platform is also given. Most notably, the description of the Digital Companion, the Distributed Verification Services, the Expert Media-Verification, and the SocialTruth blockchain - the key pillars of the platform - is put forward.

To address the scalability issue, Docker alongside DockerSwarm is proposed as the operational ecosystem. Besides scalability, other advantages of the container technology include simple dependency management and application versioning, lightweight deployment and solving challenges connected with isolation and portability, as well as with runtime environment configuration.

In accordance with the SocialTruth DoA, the deployment of project outputs will be characterised by decentralization. For this reason, four possible, cloud-based deployment options are considered and evaluated - Public-, Private-, Community- and Hybrid-Cloud.

Section 3 is dealing with the integration aspects. Various interfaces to existing partner solutions are outlined, ensuring proper communications between the solutions and the SocialTruth platform. API's to QWANT solutions, ESF solutions (Cogito 14) and Thales solutions are considered. The SocialTruth Blockchain implementation is investigated and some preliminary options are presented: Multichain, Hyperledger fabric or Ethereum. The interface to the blockchain is also delineated.

Section 4 details the ways SocialTruth will comply with the GDPR and the OWASP Dev Guide Principles of Security Engineering, ensuring both Privacy by Design and Security by Design. The challenges of authentication in the selected architecture model are also listed.

Section 5 covers the socio-technical and human factors of the project. The section evaluates how the SocialTruth platform interface will achieve clearness, conciseness, familiarity, responsiveness, consistency, aesthetics, effectiveness and recovery, which are established guidelines for usability, providing a solid foundation for exceptional user experience. The platform is going for a natural and discreet look, fostering a habits-driven attitude of the user towards it. The SocialTruth branding, as well as the recommendations for each use case are exemplified. Chrome and Safari are chosen as the web-browsers the plugin is going to cover, and Android is selected for the mobile application.

Table of Contents

1 Introduction

This document combines outcomes from the following tasks:

- Task 2.2: Distributed Architecture Design – led by UTP,
- Task 2.3: Security & Privacy by Design – led by TECOMS, and
- Task 2.4: Socio-technical and Human Aspects – led by Z&P.

D2.3 document (Refined Distributed System Architecture) is considered as the follow-up of the current deliverable and expected in M15. Therefore, the current document is the initial version of the SocialTruth architecture and will be updated until M15.

1.1 Motivation

Task 2.2 is responsible for the definition of the modular, flexible and open distributed architecture to support the operation of the SocialTruth solution and its uninterrupted service provision. It will identify the specifications for all software components as well as the interfaces and interactions among different components of the architecture. The specifications of the interfaces with the external services interoperating with the SocialTruth components will be produced as well. The structural and behavioural description of the proposed functionalities and services, as well as the deployment options for each introduced component, will be provided using a well-defined Architecture Description Language (ADL). A (micro) service-oriented approach will be adopted, wherever applicable, while other important attributes, such as modularity, openness and scalability should be taken into account at the same time.

1.2 Intended audience

This deliverable is a report produced for all the members of the SocialTruth project. Specifically, the results of this report are addressed to the following audience:

- End-user partners, who will deploy elements of the SocialTruth platform and its particular components,
- The SocialTruth project researchers and developers, who will provide technical solutions,
- The platform integrators.

1.3 Scope

In general, the purpose of D2.2 document reporting this task is to provide detailed, functional and non-functional specifications of the distributed SocialTruth Architecture and the associated interfaces with the outputs of Task 2.1 (scenarios specification, use-cases and requirements) taken into account.

In details, D2.2 will be divided into three parts, which will focus on:

- Technological aspects: specification of the SocialTruth components and interfaces with the focus on modularity, flexibility and openness (microservice-oriented approach is considered),
- Security aspects: addressing security and privacy needs in technical specification, including such techniques as data encryption, privacy keys, digital signatures, security of the communication (protocols), authentication, anonymization, etc.

- Social and human aspects affecting system design, in particular design and optimization of HMIs for SocialTruth addressing needs of different user categories.

1.4 Relation to other deliverables

D2.2 is the first iteration of the SocialTruth architecture, laying the foundation for D2.3, where a fully detailed top down and for each of modules description, design and functionalities will be exposed. D2.2 will also include the scout of the State-of-the-Art technologies that will be used in the project, as well as the potential interfaces that will be incorporated into the system, both external for the overall platform (Common Interfaces) as well as internal for each of the composing modules.

This deliverable is linked with other deliverables produced within the SocialTruth project. The relations have been shown in Figure 1.

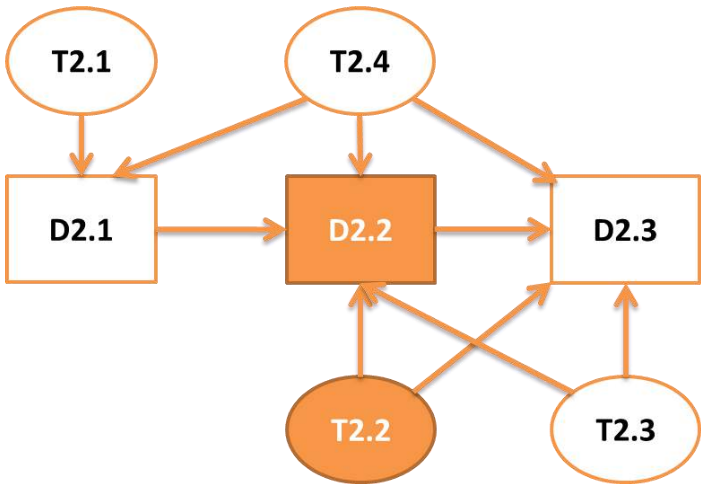


Figure 1 - Relation of D2.2 with other deliverables and project tasks.

The D2.2 deliverable produces outcomes to the following deliverables:

- D3.1 Social Mining Descriptors
- D3.2 SocialTruth Semantic Analyzer
- D3.3 SocialTruth Deep Learning Multimedia Verification
- D4.1 SocialTruth Blockchain
- D4.2 SocialTruth Lifelong Learning Expert System
- D4.3 SocialTruth Digital Companion
- D5.2.1 SocialTruth Integrated Prototype R1.0

2 SocialTruth Platform Architecture

2.1 Analysis of input from D2.1

SocialTruth D2.1 The requirements and use-cases deliverable finalized in M3, was focused on identification and detailed description of the project scenarios and presentation of the user requirements gathering and formalization process. It also provided common understanding of the SocialTruth research and development goals, guiding the further consortium work, also in terms of D2.2 architectural design.

In detail, D2.1 provided:

Pilots technological needs assessment – from the architectural viewpoint it refers to the verification services layer of the SocialTruth architecture. From the initial architecture perspective provided in the current document, the verification services are transparent, i.e. the integration mechanisms and API definitions provided in Section 3, as well as the microservice-based approach ensure scalability and extendibility of the platform without complex adaptation of the architecture to the existing or new services to be plugged into the system. Details of the verification services and mechanisms hidden behind APIs are not significant for the architecture design.

Details on four SocialTruth scenarios/use-cases, namely: (1) Checking sources in the production process, (2) Digital companion for content verification, (3) Search engine rankings & advertising prevention for fraudulent sites, and (4) External sources reliability check in the educational domain. Although these use-cases are put into different contexts and have different actors/end-users, they can be characterized with some commonalities (the same or very similar functionalities), such as: launching verification process using SocialTruth front-end (Digital Companion), requesting verification services to calculate scores separately for each verification criterion, fine-tuning or specification of thresholds for verification criteria, combination of verification results into the single credibility score and displaying it to a user, storing results in a blockchain. Therefore, the current release of the SocialTruth architecture specifies general building blocks and relations between them, covering the main, common features of the system relevant to each of the use-cases. They enable execution of the verification process for each of end-user categories and in each specified scenario.

Formalization of the SocialTruth requirements with the end-user assessment of their criticality and the UML use case diagrams – similarly to the detailed use-cases/scenarios, the functional requirements show priorities of the main/common system functionalities, covered by the current architecture release. The detailed functional requirements expressed by the end-users refer to the level of particular verification services (therefore they are non-significant in the general architecture context). The non-functional requirements, including scalability, integration capabilities, security and privacy, and usability are covered in the current document in sections 3, 4 and 5 respectively.

2.2 Architectural challenges

The distributed environment imposes many technical challenges that have to be addressed. These are commonly related to the fact that the Microservices-oriented architecture style promotes loosely coupling and decomposition of application into smaller and independent entities that implement specific business logic. Some of the challenges include:

- Scalability
- Availability
- Independent and autonomous management
- Decentralized governance
- Failure isolation

2.3 Microservices approach overview

2.3.1 Popularity of micro-services

The growth in container solutions has resulted in the development of micro-service solutions for deploying dedicated applications which can be integrated in the main platform using standards protocols. One of the critical characteristics of micro-service deployments is the reliance on carrying out unit testing on individual components without any external dependencies. This feature has also contributed to the creation of distributed and flexible workflow architecture that partly enables/disables component instantiation without affecting the overall performance of the platform. The distribution of micro-services packaged in containers has also resulted in achieving scalability as several instances of a single component can be deployed in run-time. Service Architecture Models

The industry-best practice allows for two modes of integration for sharing information between software components. The first approach is based on webservices, which uses the HTTP protocol for transporting information between components. The implementation is light weight and supports flexibility in the design of the APIs. It also supports several forms of MIME data payload between different components. On the other hand, the popularity of messaging solutions based on queues has also been a popular solution for the integration of complex systems over the last few years. The adaptation of messaging protocols for enterprise integration has been considered to provide extended reliability. Since both integration models provide an added advantage to the platform, the approach in the SocialTruth architecture design will be to adopt a hybrid solution that supports both forms of integration.

2.3.1.1 RESTful Webservices

The microservice architectural style is an approach which allows for the development of an application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, like HTTP, as aforesaid. This enables each microservice to be independently deployable and scalable. This provides a firm module boundary, therefore facilitates easier management and empowers being written by different teams, and thus making it worth to be considered in the SocialTruth architecture.

We can define the microservices architecture as a distributed architectural system that organically evolves into scalable, loosely coupled, independently managed sets of services that work together to deliver business value with acceptable trade-offs. We can name the following key features of microservices:

flexible technology choices; “smart endpoints” “dumb pipes”; Independently scalable; Decentralized, choreographed interactions; Testable; Automation; Designed for failure; Evolving design.

In fact, in current architectures we can see a transition from the current (still) governing one: Services, SOA, ESB, Adapters, Protocol Mediation, Enterprise Integration patterns, Canonical Messages, to the Upcoming one: API, RESTful Microservices, Gateways, HTTP(s) Only, Service Discovery, Circuit Breakers. We are moving from a hub architecture to a peer to peer one. The microservice architecture is very well suited for state of the art and particularly adapts well to the agile methodology developments, REST, domain (modular) driven designs, etc.

2.3.1.2 Messaging platforms

The Advanced Message Queuing Protocol (AMQP) is an open standard for passing business messages between applications or organizations. It connects systems, feeds business processes with the information they need and reliably transmits onward the instructions that achieve their goals. RabbitMQ implements the AMQP protocol, providing thus a common platform to services for their message driven communication. It is open-source and is packaged as a server tool. Among other alternative solutions, it is the most mature and offers client libraries for almost any programming language. RabbitMQ facilitates the work queues required for the task dispatching in the context of the SocialTruth architecture integration.

- *Reliability*: Offers message delivery acknowledgments, a feature that is not supported by other similar solutions
- *Flexible routing*: Messages are routed from exchanges offering advanced routing (e.g., automatic publish of one message to multiple queues)
- *Many clients*: Client libraries are available for a large number of programming languages
- *Highly available queues*: Queues can be mirrored across several queues in a cluster
- *Management UI*: Offers an easy-to-use management UI for monitoring the message broker
- *Large Community*: There is a large community supporting RabbitMQ, so as all sorts of clients, plugins, guides, etc. can be easily obtained.

2.3.2 Standardisation of API models

The SocialTruth architecture will adopt a holistic approach for achieving optimal integration of system components. The webservice based integration will be adopted for those components which share the state of the system without too much computational overhead while the messaging queues will be used for instantiating computationally expensive services. Each component integrated in the architecture will specify the APIs for integration.

Other considerations like loosely/tightly coupled components or relying on external tools (and thus hybrid clouds) will be also considered, making it more appealing in this case to rely on a webservices approach.

2.4 Service-oriented architecture vs. Microservices architecture styles

The simple definition of the service-oriented architecture (SOA) is an architecture allowing interconnection, communication and coordination of services. Service in such an architecture is well-defined, self-contained, and independent from the other services function, designed to realize some functionality e.g. by software component. In SOA, services have to be accessible remotely (by the user or other services) and can be updated, extended or unplugged independently and with no impact on the status of other services. Service providers provide one or more services through published interfaces for service consumers, which invoke services directly or through an intermediary.

In addition, services can be considered as a black box for its consumers, i.e. provide specific result based on the provided input after the consumer request. Also, service in SOA architecture may consist of other underlying services. Different services can be used in conjunction to provide the functionality of a large software application. What is important in this style, services are also technology independent - service provider components and service consumer components can use different implementation languages and platforms.

Service-oriented architectural style can be characterized by the combination of the following principles:

- Loose coupling – relations between services that that minimizes their mutual dependencies and only requires that they maintain an awareness of each other
- Standardized Service Contract - services adhere to a communications agreement (a service-description), as defined collectively by one or more service description documents
- Abstraction - services hide logic from the outside world (beyond what is described in the service description)
- Reusability - logic is divided into services with the intention of promoting reuse
- Composability – services can be reused in multiple solutions that are themselves made up of composed services (e.g. complex functionalities are break down into simple tasks)
- Autonomy - services have control over the logic they encapsulate
- Statelessness - services minimize retaining information specific to an activity
- Discoverability - services are designed to be found and discovered, usually using a service registry or via other available discovery mechanisms

The practical implementation of SOA is Enterprise Service Bus (ESB) serving as a hub for connected services as well as centralizing and simplifying communication between them.

On the other hand, the recent advancements in cloud solutions have changed the way of building middleware components for the Platform as a Service (PaaS) architectures together with the container-based technologies that gain popularity in development and management of architectures. One of the architecture styles employed to reduce the investment of effort in rewriting the legacy monolithic systems (e.g. based on ESB) to provide new functionalities is the microservices approach. Microservices are used to build a system as a set of autonomous, self-contained, and loosely coupled capabilities. Usually, the application build in a microservices architecture style is assembled as an artefact embedding all dependencies (libraries, HTTP server etc.) and run as standalone process (Figure 2).

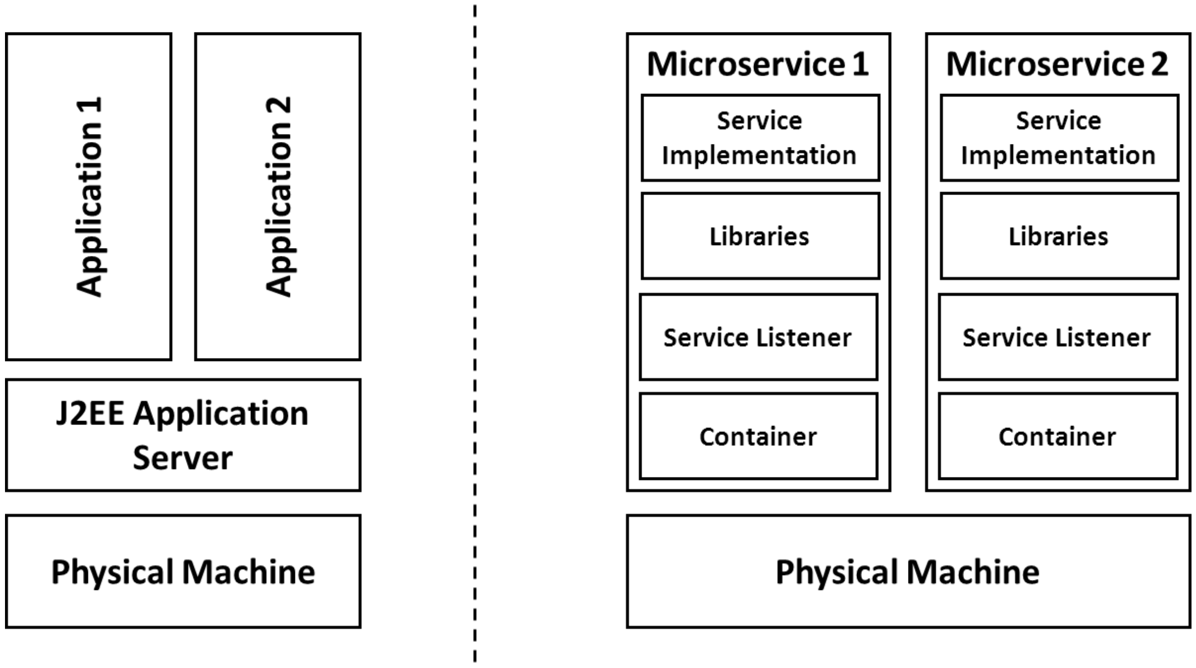


Figure 2 - Comparison of monolithic application (left) vs. microservices approach (right).

The potential use of microservices architectural style in the context of SocialTruth could be beneficial due to a number of capabilities it offers, e.g.: ease of integration of new services/functionalities into the framework, simplification of integration tests, simplification of development updates (only the relevant parts of an application), better resiliency and stability of the application by eliminating a susceptibility to a single point of failure. Microservices/API-based approaches also have some drawbacks, namely the complexity of development distributed systems, multiple databases and the necessary transaction management, container-based deployment.

Taking into account the general, high-level architecture of the SociaTruth (please refer to the next subsection) and the nature of the research and development project, the recommendation is to adapt microservice architecture style – as initially suggested in the SocialTruth DoA. The rationale behind this choice is:

- Easiest implementation of new services, changes and updates – particularly important for frequent releases of the system that is developed in collaborative project,
- Better fitting to cloud-based approach and SocialTruth requirement of services decentralization,
- More suitable to use containers (e.g. Docker),
- Better suited for smaller (than large enterprise solutions) and well-partitioned, web-based systems,
- Better addressing requirement of the SocialTruth extendibility.

2.5 Architectural design patterns

Analysing various microservices based solutions, several architectural design patterns can be identified. These could be clustered according the challenge they need to address, namely:

- Decomposition patterns addressing the problems of breaking application into smaller, autonomous and independent pieces.
- Integration patterns addressing the problems of communicating the services with each other and presenting the scattered data to the consumers.
- Database-related patterns aiming at addressing the problems of database architecture in the microservices environment.
- Monitoring and observability patterns tackling the challenge of monitoring the distributed systems.

2.5.1 Decomposition

Microservices adapt the single responsibility paradigm and promote loosely coupling. There are different ways as to how the monolithic application can be divided into several smaller autonomous components. The most obvious strategy is to use decomposition that is **based on business capability**. For example, a system supporting sales would be decomposed in to services responsible for customer, orders, invoices, etc. In general, the process of decomposition produces smaller entities that can be developed individually by separate teams. This allows the teams to sustain autonomy in terms of architectural patterns and technologies selected to develop a specific service.

2.5.2 Integration

When the application is broken down into a set of separate services, eventually it happens that these need to communicate in order to provide complex business capabilities. That capability usually needs to assemble the results obtained from multiple services.

Many challenges may appear depending on the application. For example, some services need to be orchestrated to produce the final result. It means that a specific chain of actions needs to happen and these need to be sequenced in a time manner. In that case an aggregator pattern could be adapted. In general, such a pattern defines different strategies to aggregate data from services and then provides them to the consumer.

The aggregation could be performed in different ways and one of the most popular and widely used is the **API Gateway pattern**. The pattern appears in many microservice frameworks such as Java Spring Cloud [1]. In general, the gateway can be seen as a reverse proxy, which is used by services that reside in the backend (are hidden behind the reverse proxy). It takes requests from the client and forwards these requests to one of the backend services. There are several advantages of using the API Gateway pattern. Firstly, it constitutes a single entry point for any call. This, for instance, allows for implementing the authorisation functionalities at the gateway. Secondly, the gateway can translate the request protocol to something

[1] https://spring.io/projects/spring-cloud

else such as AMQP (Advanced Message Queuing Protocol). Thirdly, the gateway can proxy request from client to multiple services and aggregate results.

2.5.3 Database

The microservices need to store data. Depending on the context, this can also be a challenging task to implement. Defining the appropriate database architecture is problematic due to following aspects:

- Sharing single database between microservice impacts the system scalability and services autonomy
- Some business capabilities need data that is owned by several microservices
- Implementing transaction mechanism in a distributed environment is problematic and requires coordination and extensive communication of participating microservices

The good practice says that single database per service should be used. This means that the specific service has its own database that is isolated and is not shared with others directly. This avoids the situations where the development of one service and its data model influences the development of another service. However, it happens that the isolated service eventually needs to reach out for the data maintained by another service. One of the options solving this would be a **CQRS pattern** (Command Query Responsibility Segregation. It promotes splitting the command and query parts, so that typical CRUD (Create Read Update Delete) command-based operations are handled by one system while data querying capabilities are served by other. In order to provide query results that join data from multiple services materialised views are used. The views are updated whenever any part of the data changes. These are communicated using event busses (e.g. Apache Kafka or RabbitMQ). The service maintaining the materialised view listens to the event bus for notifications and updates the view accordingly.

2.5.4 Monitoring and Observability

In a microservice architecture a single request often spans multiple services that are hosted on separate physical servers. Each service generates a log file that is stored locally. In such case, reconstructing the original information flow (from client request to the returned result) could be time consuming if done manually. This is also an important aspect from the system auditing or user accounting point of view. If one needs to trace the request end-to-end a dedicated centralized service aggregating logs is needed.

An effective approach increasing traceability would be to assign each request with a unique identifier. In case of HTTP protocols additional header parameter could be added. Then, the identifier passed to the services can be used by them to annotate each operation stored in a log file. The log files can be efficiently shipped for central analysis and inspection using such frameworks as Elastic Stack together with Apache Kafka.

2.6 General specification of SocialTruth components

This section provides general specification of the key components that compose the SocialTruth platform. As the starting point for the SocialTruth platform architecture design specification we have used the general model depicted in the Description of Action (see Figure 3).

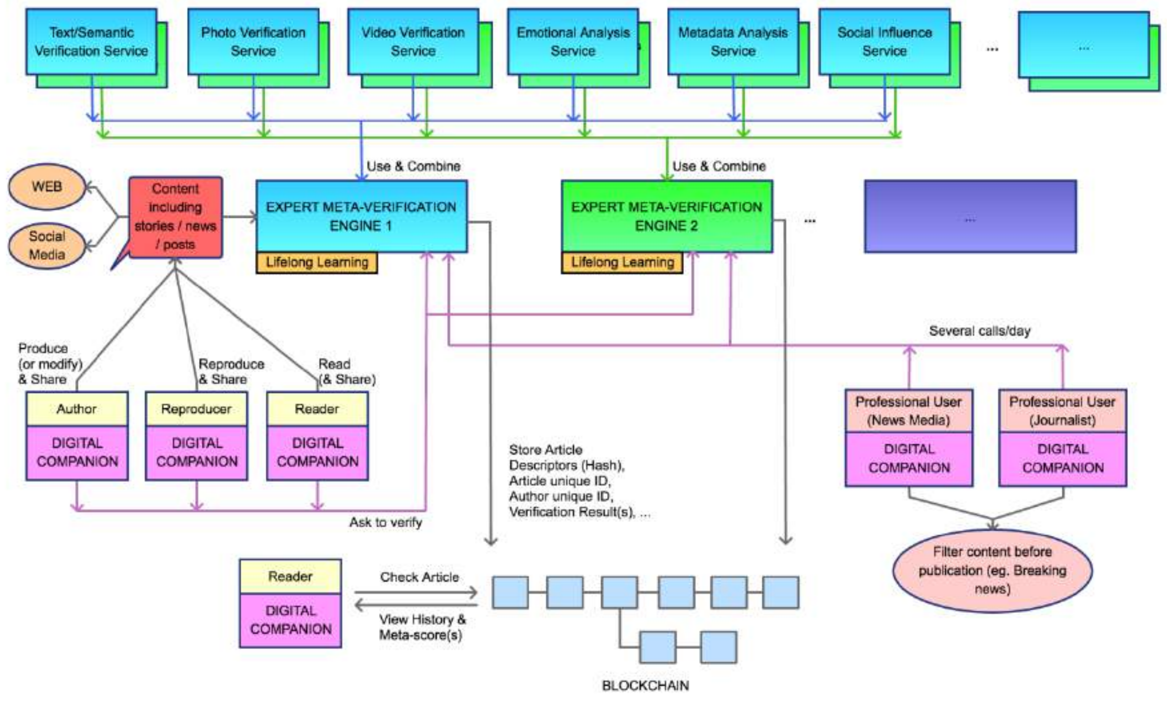

Text/Semantic Verification Service
Photo Verification Service
Video Verification Service
Emotional Analysis Service
Metadata Analysis Service
Social Influence Service
...
Use & Combine
Use & Combine
WEB
Social Media
Content including stories / news / posts
EXPERT META-VERIFICATION ENGINE 1
Lifelong Learning
EXPERT META-VERIFICATION ENGINE 2
Lifelong Learning
Several calls/day
Produce (or modify) & Share
Reproduce & Share
Read (& Share)
Author
DIGITAL COMPANION
Reproducer
DIGITAL COMPANION
Reader
DIGITAL COMPANION
Professional User (News Media)
DIGITAL COMPANION
Professional User (Journalist)
DIGITAL COMPANION
Store Article Descriptors (Hash), Article unique ID, Author unique ID, Verification Result(s), ...
Ask to verify
Filter content before publication (eg. Breaking news)
Reader
DIGITAL COMPANION
Check Article
View History & Meta-score(s)
BLOCKCHAIN

Figure 3 - SocialTruth Platform Architecture [source: DoA]

The key pillars of SocialTruth platform constitute:

- **Digital Companion**: considered as a browser plugin that allows a non-professional user to invoke a meta-verification process upon some form of digital content (e.g. an article), passing its URI as an input to a meta-verification engine. In case of a non-professional use, the Digital Companion can be used by the author of the digital article, by a reproducer (who shares the article in the Social Media) or even by a simple reader of the article, who wishes to get an estimation of the credibility of the content before or after reading it. In case of a professional use, the Digital Companion is a web front-end for medium/large organisations (e.g. news agencies, search engines, etc.) that allows several calls per day to the APIs of the meta-verification engine(s). SocialTruth will follow a user-centred design approach to product for the Digital Companion.
- **Distributed Verification Services**: a set of heterogeneous verification services providing a specific type of content analytics (e.g. for text, image, video) or verification-relevant functionality (e.g. emotional descriptors, social influence mapping). Some of these services are made available and deployed by the SocialTruth consortium partners, while others are coming (either in open source or not) from third-party service providers. Each service can be deployed at a different hosting facility (e.g. different servers or clouds), hence there is no imposed centralization. All of them use the same standard SocialTruth interfaces to allow them to be easily accessible, reusable and interchangeable. The registrar of service providers and the services they offer is stored and maintained in the blockchain.
- **Expert Meta-Verification Engine(s):** to combine verification results from various verification services to compute a meta-score that reflects the credibility of the digital content under consideration. It follows an open design, open algorithms and an expert-systems approach. It uses

open algorithms while most of its settings and weights (e.g. which verification services to prefer or to avoid, with what priority, etc.) can optionally be configured through its standard web-service interfaces.

- **The SocialTruth blockchain**: a distributed system of records with respect to the digital content verification history. Since the complex web and social media landscape is characterised by several competing content creators and distributors, each with their own motives, interests, strategies and practices, the blockchain is an ideal tool to establish reputation and trust without the need of a central authority or intermediary (thus also avoiding to centralize even more regulatory power to the US Internet giants, such as Facebook or Google). Hence, a public distributed ledger provides an auditable and immutable trail of verification actions and reputation scores. The blockchain will store article identification information, article descriptors (e.g. hash codes for digital content integrity), author identification information, verification and meta-verification scores, as well as identification information for the verification services that have been used to calculate them. It will also hold the registrar of verification service providers and the services they offer.

Functionally, these elements depend on one another and are logically pile-up as classical **N-tier architectural** model that is comprised of a data layer, a business layer, and a presentation layer. The bottom data layer of SocialTruth N-tier architecture model constitutes Distributed Verification Services together with common interfaces providing access to the data. The middle layer, providing business logic is the Expert Meta-Verification Engine. Finally, the presentation layer capabilities are provided by the Digital Companion component. Each of these functional components is further composed of dedicated modules that provide or facilitate the dedicated functionalities the specific component is intended to provide.

2.7 The operational ecosystem

In order to address the scalability and the platform orchestration we recommend using Docker together with Docker Swarm to maintain the ecosystem. The concept of using Docker Swarm in the SocialTruth architecture is presented in Figure 4.

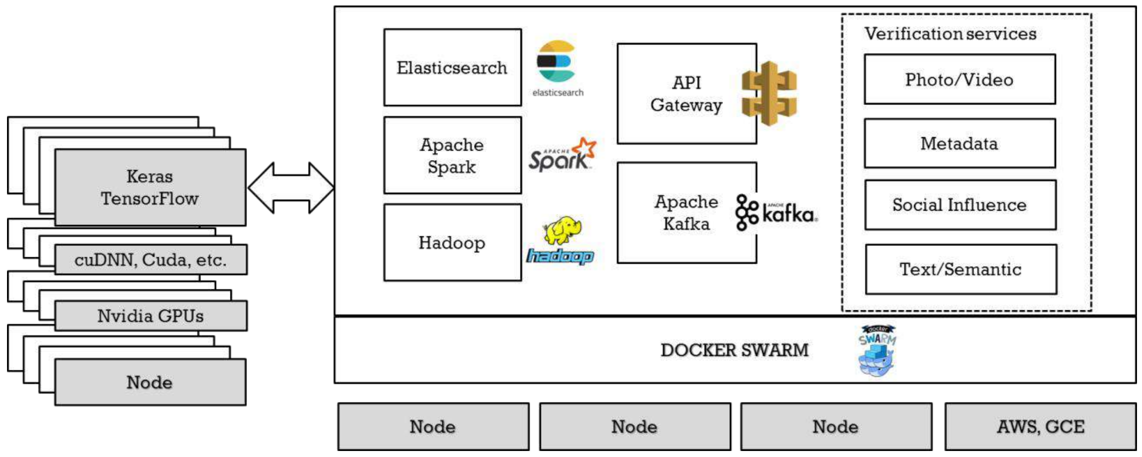


Figure 4 - Use of Docker Swarm in the SocialTruth architecture

The Docker Community is a forum for enthusiasts that use the virtual containers, micro services and distributed applications. Moby is an open framework created by Docker to assemble specialized container systems. It provides a “lego set” of dozens of standard components and a framework for assembling them into custom platforms. Docker is an open source project that is aimed at simplifying deployment of an allocation by means of containers. It allows for building an image with the application deployed inside. A running instance of image is called a container. The image contains all the dependencies that the applications need to run (e.g. operating system, runtime environment, specific system libraries, etc.). The image can be easily run anywhere (on the variety of host operating systems) executing the application in an isolated environment. Swarm is a Docker-native container orchestrator used to manage Docker containers as a cluster of machines. Docker Swarm eases the deployment, organization, management and scaling of Docker containers.

The containerisation differs from hardware virtualization in the way that it has higher performance (containers do not emulate the entire computer architecture), lower resource consumption, and smaller images (containers do not require a full operating system). Containers solve many problems of software delivery such as runtime environment configuration, isolation, application management, and portability. Using a single image one can run many containers (copies of the same application). At the same time Docker enables rapid “diff” changes within the various software builds to verify the consistency of the solution over the versions. An image is a stateless building block of the Docker system. From the functional point of view an image has a layered structure. It means that images are easy to extend by adding additional layers. For example, a J2SE application would have a basic image of the operating system (e.g. Ubuntu). A JDK can be easily installed on top of it. Optionally the user can also add additional programs such as Git. All modifications can be persisted afterwards as a new image. In case of Docker, it provides a dedicated configuration language that allows for quick image definition.

2.8 The technical architecture

There are two disjoint and fundamentally different subsystems within the SocialTruth platform. The first one is the p2p network composing the distributed ledger based on the blockchain technology. The second is the distributed verification system following the microservice architecture style.

The EMVEs (Expert Meta-verification Engines) will act as clients composing the p2p network. The technology stack of a single EMVE will include native libraries or software components that will facilitate p2p network matters, such as nodes discovery, routing, node management, and bilateral communications. Moreover, the EMVE will be facilitated with client component enabling communication with the distributed verification system (Figure 5).

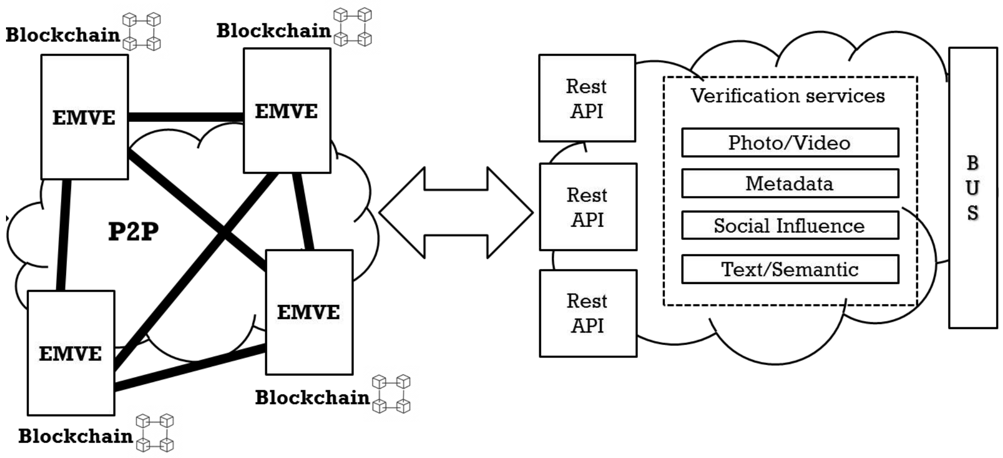


Figure 5 - SocialTruth technical architecture

P2P network of EMVEs

In order to implement the p2p network the EMVEs need to be facilitated with a dedicated software component. This needs to handle such low-level networking aspects as routing, network elements discovery, autonomous network management, etc. Apart from that mattes, also various elements facilitating blockchain and distribute ledger need also be in place.

The EMVEs will also be facilitated with a client library or a dedicated API allowing communication with the distributed verification system (Figure 6). This will need to transform the user request into a specific backend service call. For example, whenever the user requests document verification, the EMVE will need to call appropriate services that will verify images (if present in the document), text, run sentimental analysis, and check existing databases for any information that will allow for discovering any negative or positive symptoms judging the content of the document.

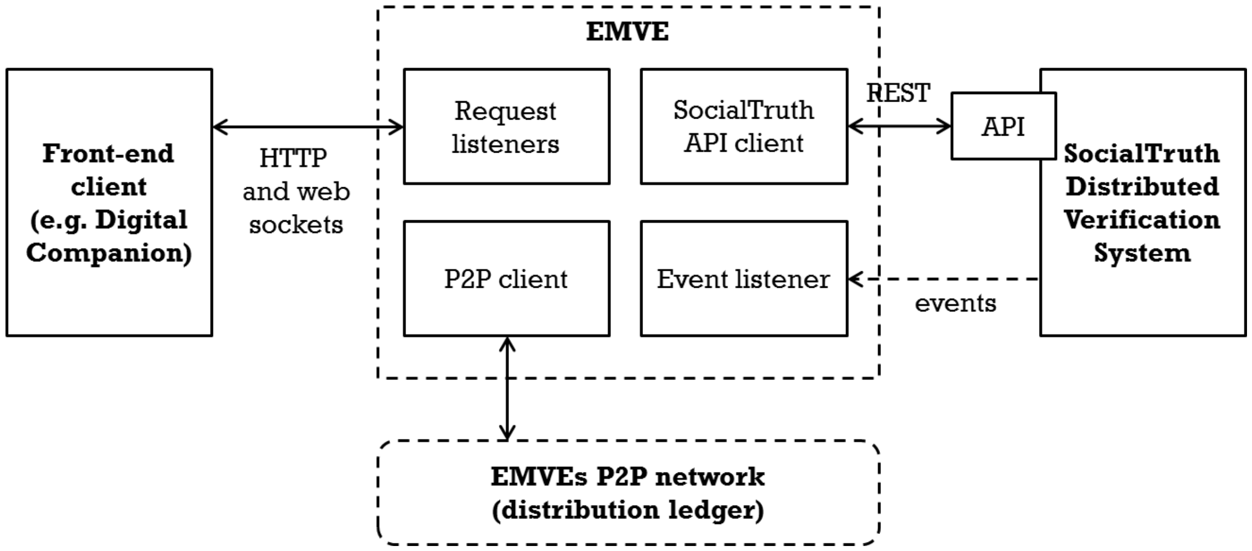

EMVE
Front-end client (e.g. Digital Companion)
HTTP and web sockets
Request listeners
SocialTruth API client
REST
API
SocialTruth Distributed Verification System
P2P client
Event listener
events
EMVEs P2P network (distribution ledger)

Figure 6 - SocialTruth technical architecture - single EMVE view

The EMVEs will need to implement the CQRS pattern (Command Query Responsibility Segregation) as the operation of querying and indexing of the document will take considerably varied amounts of time. Querying (or reading) the information of an already indexed document will be significantly faster, in particular if the document verification will be based on the URL address (e.g. we may already know that news published on a specific website is fake and results of analyses are already there in the SocialTruth database). In that case the results could be returned in a simple request-response manner via the RESTful API. On the other hand, the process of indexation (document ingestion) will require a different approach. The uploaded documents need to be stored and analysed asynchronously by the services. In that case an orchestration coordinated by the EMVE will be required. For example, it will need to start the analyses by pointing the services to the data to be analysed, wait for the results, consolidate them into a single information piece that will be consumed by the requestor. In the following scenario, a publish-subscribe messaging system would be a better fit than the request-response approach.

The EMVEs will also be the elements which are closely interacting with the end-user. There are two main cases where the user will be engaged. First, the operation when user queries the EMVE to verify a particular document, news, post, etc., second, the situation where the user wants to add a new document with the information about its credibility. Whenever the document has already been indexed and annotated by someone else, the user may also express her or his opinion regarding it.

2.8.1 Distributed content verification system

The distributed verification system will be composed of several heterogeneous services that will be functionally focused on a specific kind of context analysis, e.g. images, text, etc. (Figure 7). From the end-user point of view these specific services should be visible as a monolithic system providing various capabilities.

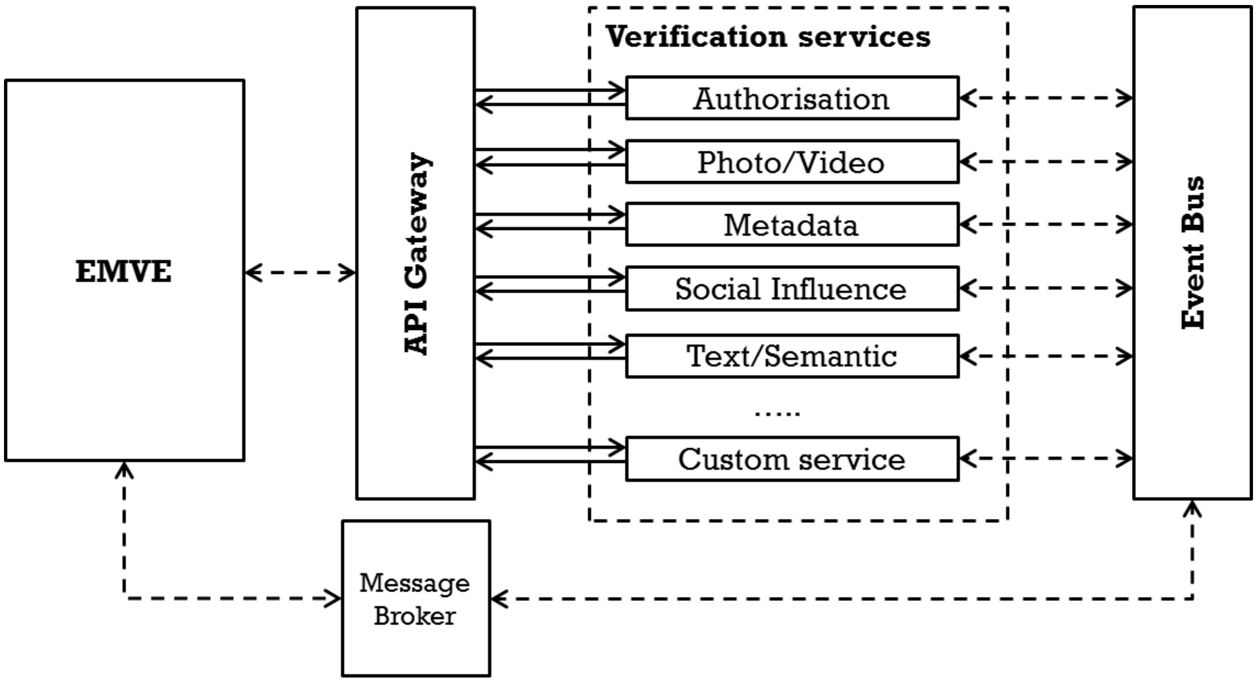


Figure 7 - SocialTruth technical architecture - distributed verification services view

API Gateway

In that sense an API Gateway pattern could be used. It would allow for hiding the microservices behind the middle-layer that would be acting as a reverse proxy, handing the client requests, passing them to the specific services, and returning the received results to the client.

Another benefit of adapting this pattern is the fact that we can offload the microservices authentication burden directly onto a gateway. Moreover, we can abstract the microservice details (e.g. IP address) making the gateway work as a reverse proxy, mapping the user request into a specific backend service call. This will also be beneficial from the EMVE perspective, since it will decrease coupling – the EMVEs will not have to know the location of each service. Instead, the EMVEs will use a single entry point. This will also encourage development of a consistent API.

The services behind the gateway will be stateless entities that take the data in a predefined format and return the result. The interaction between them will also be limited. Nonetheless, the services will need several elements to facilitate their work. These include data storage, search engine, data processing, publish-subscribe systems, etc.

Data storage

There will be a certain amount of data that will need to be processed, analysed, stored, and indexed. Rather than sending back and forth the entire documents for verification, the data should be uploaded once and later referenced using pointers (e.g. URL address). Therefore, adequate data storage is an important element of the architecture.

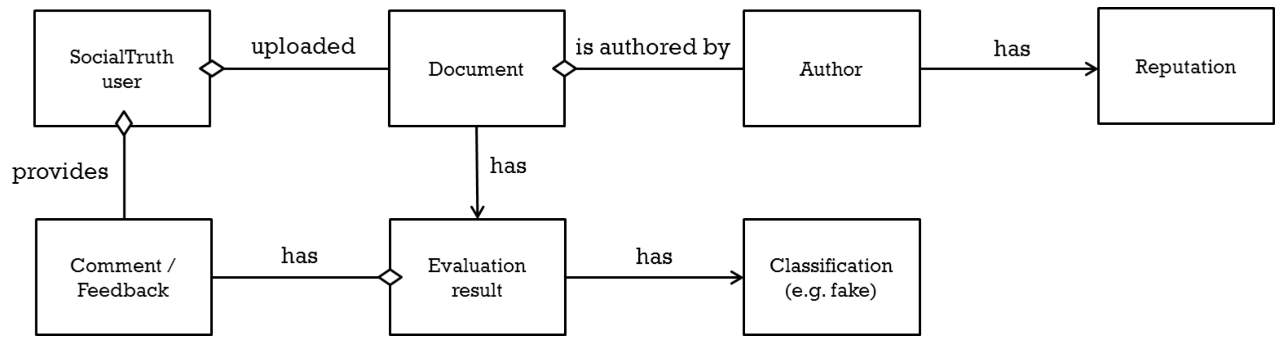


Figure 8 - Concept of data storage in SocialTruth

From the overall description of the SocialTruth proposal emerges a general data model to be shared among various modules comprising the platform. The first data entity is the document to be verified, the second one is the author of this document. The third one is the reputation (verification) score. Finally, the evaluation report also constitutes an important element (Figure 17).

The system should also provide capabilities to look up the previous verifications. These should be indexed in various ways in order to avoid double checking of the same document. The most straightforward way would be indexing the results by source (e.g. URL address).

Events-based communication

Once the data is uploaded the services can process it. Usually, it will take some time. Therefore, instead of periodically pooling the services for current state, an event-based publish-subscribe system would be a better fit in such scenarios. For example, a client requesting the verification of a specific image can submit the request via standard API and subscribe to the event bus for updates. Once the service finishes the processing it sends notification event via the event bus. Using such platforms as Apache Kafka it is possible to guarantee fault tolerance and scalability of such mechanisms. For instance, if the network connection fails or the client is down when a notification event is sent, the client is always capable of receiving the event once it is back to normal state.

2.9 SocialTruth deployment options

2.9.1 Deployment Models

According to the SocialTruth Description of Action, deployment of the project outputs will be characterized by decentralization. Each of verification services will be deployed at a different hosting facility (e.g. different servers or clouds). All of them use the same standard SocialTruth interfaces to allow them to be easily accessible, reusable and interchangeable.

For the deployment purposes of SocialTruth, the services will be hosted at the appropriate infrastructure provided or on the cloud. Both solutions have their own advantages and disadvantages. The on-premises solution requires a substantial investment to implement and dedicated people (IT experts) to maintain and upgrade the infrastructure according to the needs of the SocialTruth platform. On the other hand, the cloud-based solutions are generally considered more cost effective as the initial costs are much lower and they usually implement a pay-as-you-use model. Using this model, the users of the cloud-based solution are also relieved from the maintenance cost of the infrastructure. Apart from that, the cloud-based solutions offer scalability as they can easily increase the required storage capacity and processing power. In the long run, a cloud-based solution can be considered more cost-effective compared to on-premises as it reduces the overall capital expenditure while maximizing efficiency and productivity. Although the security of the cloud-based services has increased during the last years along with their wider adoption, a careful selection of the appropriate deployment model is required in order to guarantee the security concerns of the SocialTruth platform. The most common deployment models that can be used for the SocialTruth platform are presented below.

2.9.2 Public cloud

A public cloud is a publicly accessible cloud environment owned by a third-party cloud provider. The IT resources of a public cloud are generally offered to cloud consumers at a cost, usually by a pay-per-use model or are commercialized via other avenues (such as advertisement). Public clouds provide a convenient way to scale the required resources and the cloud provider is responsible for the creation and on-going maintenance of the public cloud and its IT resources. Their main advantages and disadvantages are presented below.

Advantages of public clouds:

- Scalability/Flexibility/Bursting
- Cost effective
- Easy to implement
- Continuous operation time
- 24/7 upkeep

Disadvantages of public clouds:

- Shared resources
- Operated by third party
- Unreliability
- Data Security and privacy
- Compromised reliability

2.9.3 Private cloud

A private cloud is owned by a single organization and it can be hosted either externally or on premises of the user company. This deployment model is best suited for organizations that deal with sensitive data and/or are required to uphold certain security standards by various regulations. Private clouds enable an organization to use cloud computing technology as a means of centralizing access to IT resources by different parts, locations, or departments of the organization. Their main advantages and disadvantages are presented below.

Advantages of private clouds:

- High degree of security and level of control

- Ability to choose your resources (i.e. specialized hardware)
- High reliability and scalability
- Greater control over cloud infrastructure

Disadvantages of private clouds:

- Lack of elasticity and capacity to scale (bursts)
- Higher cost
- Requires a significant amount of engineering effort

2.9.4 Community cloud

A community cloud deployment model resembles a private one to a large extent; the only difference is the set of users. While a private type implies that only one company owns the server, in the case of a community one, several organizations with common interests share the infrastructure and related resources. As the organizations have uniform security, privacy and performance requirements, this multi-tenant data centre architecture helps companies achieve their business-specific objectives. That is why a community model is particularly suited for organizations that work on joint projects. In that case, a centralized cloud facilitates project development, management and implementation. Also, the costs are shared across all users. Community cloud can exist either on-premise or off-premise and can be owned either by a third-party provider or shared by the organizations themselves. Their main advantages and disadvantages are presented below.

Advantages of Community cloud:

- Cost reduction
- Improved security, privacy and reliability
- Ease of data sharing and collaboration

Disadvantages of Community cloud:

- Higher cost than that of a public one
- Sharing of fixed storage and bandwidth capacity
- It is not widespread so far

2.9.5 Hybrid cloud

A hybrid cloud is a cloud environment comprised of two or more different cloud deployment models.

Advantages of Hybrid cloud:

- Cost effective
- Scalability/Flexibility
- Balance of convenience and security

Disadvantages:

- Same disadvantages as the public cloud

The comparison of the above cloud solutions can be summarized in the Table 1.

Table 1 - Summary of cloud deployment parameters

	Public	Private	Community	Hybrid
Ease of setup and use	Easy	Requires IT proficiency	Requires IT proficiency	Requires IT proficiency
Data security and privacy	Low	High	Comparatively high	High
Data control	Little to none	High	Comparatively high	Comparatively high
Reliability	Vulnerable	High	Comparatively high	High
Scalability and flexibility	High	High	Fixed capacity	High
Cost-effectiveness	The cheapest one	Cost-intensive, the most expensive one	Cost is shared among community members	Cheaper than a private model but more costly than a public one
Demand for in-house hardware	No	Depends	Depends	Depends

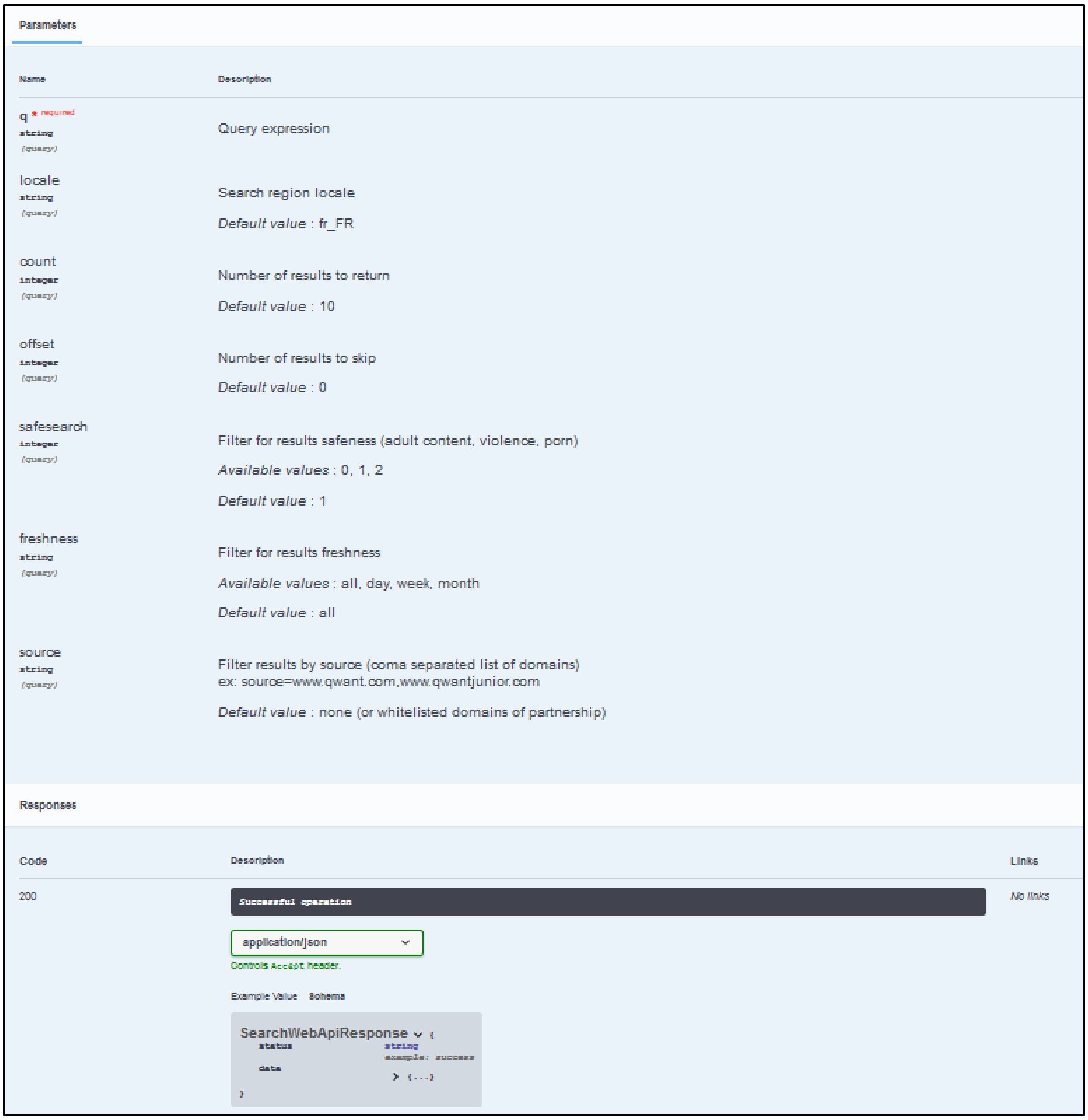
3 Integration aspects

Today, the majority of modern apps or services are RESTful APIs and use API definitions to facilitate communications between them. APIs are especially suited for microservices or serverless architectures with dozens or hundreds of mutually interacting microservices/functions. However there still exist some legacy services using SOAP.

3.1 API to QWANT solutions

APIs and result formats from QWANT are presented in Figures 9-13.

Qwant API to get Web results



Parameters

Name	Description
q * required string *(query)*	Query expression
locale string *(query)*	Search region locale *Default value* : fr_FR
count integer *(query)*	Number of results to return *Default value* : 10
offset integer *(query)*	Number of results to skip *Default value* : 0
safesearch integer *(query)*	Filter for results safeness (adult content, violence, porn) *Available values* : 0, 1, 2 *Default value* : 1
freshness string *(query)*	Filter for results freshness *Available values* : all, day, week, month *Default value* : all
source string *(query)*	Filter results by source (coma separated list of domains) ex: source=www.qwant.com,www.qwantjunior.com *Default value* : none (or whitelisted domains of partnership)

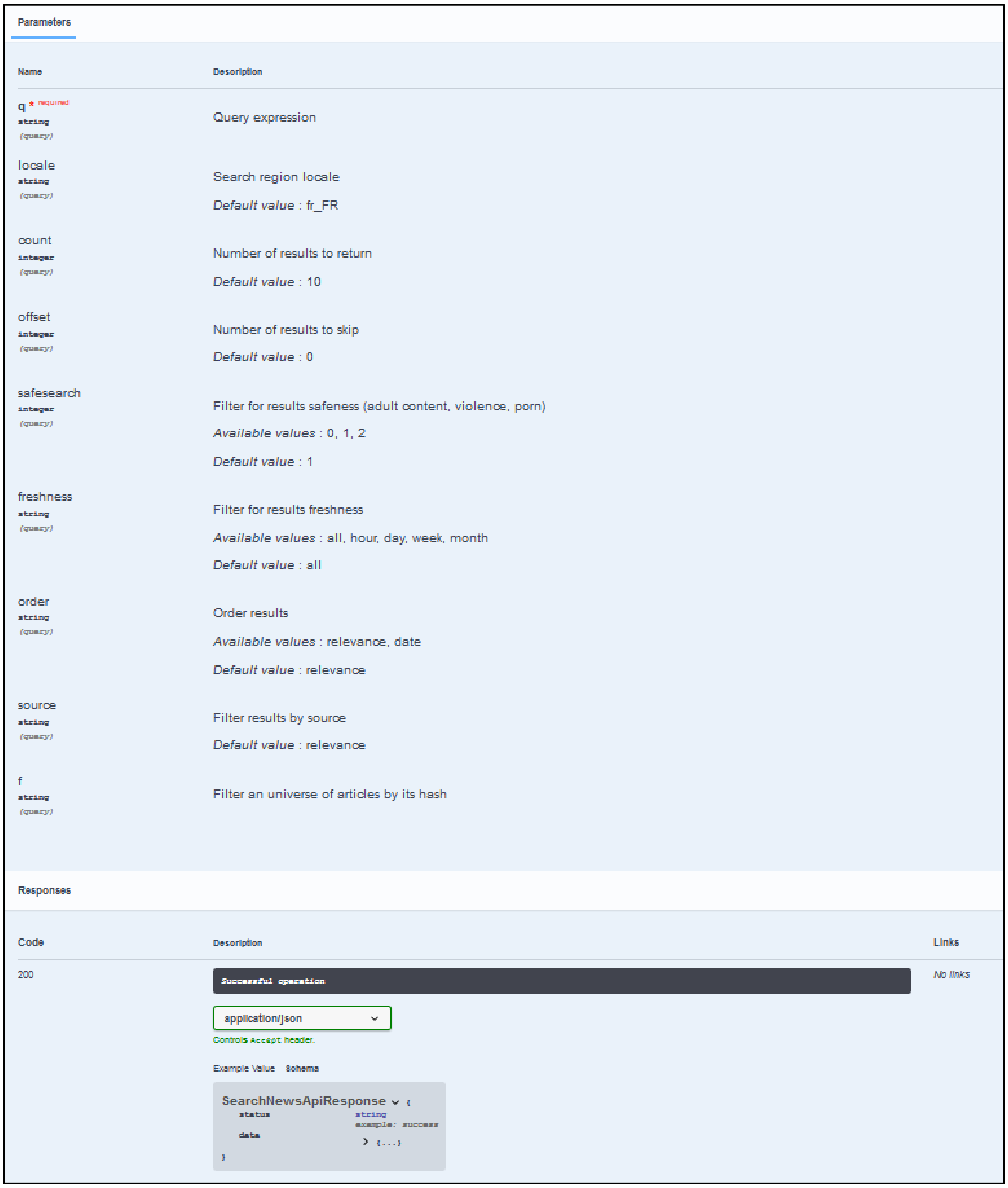
Responses

Code	Description	Links
200	*Successful operation* application/json Controls Accept header. Example Value Schema	*No links*

```
SearchWebApiResponse {
    status              string
                        example: success
    data                > {...}
}
```

Figure 9 - Qwant API to get Web results

Qwant API to get news results



Parameters

Name	Description
q * required string (query)	Query expression
locale string (query)	Search region locale *Default value* : fr_FR
count integer (query)	Number of results to return *Default value* : 10
offset integer (query)	Number of results to skip *Default value* : 0
safesearch integer (query)	Filter for results safeness (adult content, violence, porn) *Available values* : 0, 1, 2 *Default value* : 1
freshness string (query)	Filter for results freshness *Available values* : all, hour, day, week, month *Default value* : all
order string (query)	Order results *Available values* : relevance, date *Default value* : relevance
source string (query)	Filter results by source *Default value* : relevance
f string (query)	Filter an universe of articles by its hash

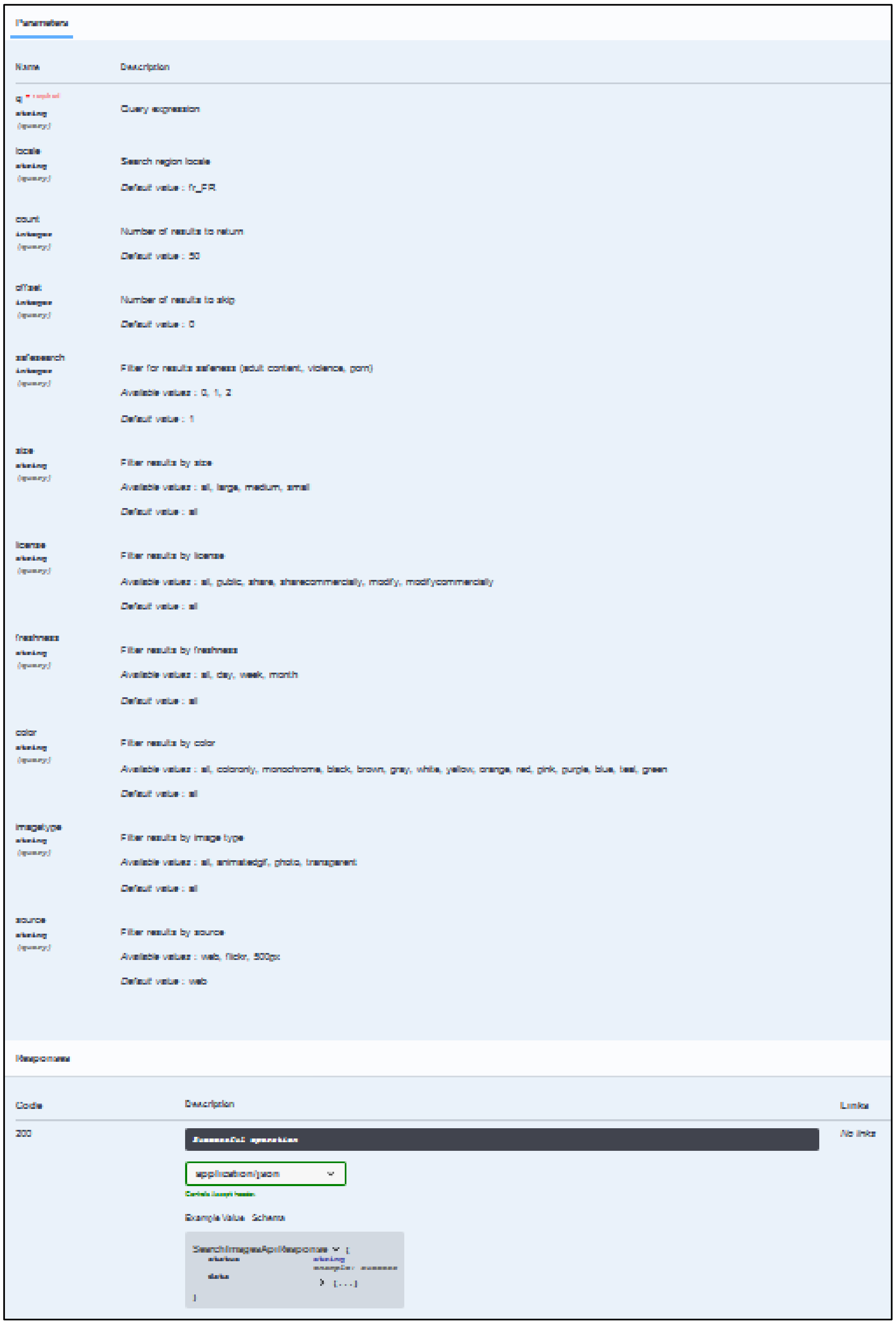
Responses

Code	Description	Links
200	Successful operation application/json Controls Accept header. Example Value Schema	*No links*

```
SearchNewsApiResponse {
    status                string
                          example: success
    data                  > {...}
}
```

Figure 10- Qwant API to get news results

Qwant API to get images results



Parameters

Name	Description
q * required string (query)	Query expression
locale string (query)	Search region locale *Default value* : fr_FR
count integer (query)	Number of results to return *Default value* : 50
offset integer (query)	Number of results to skip *Default value* : 0
safesearch integer (query)	Filter for results safeness (adult content, violence, porn) *Available values* : 0, 1, 2 *Default value* : 1
size string (query)	Filter results by size *Available values* : all, large, medium, small *Default value* : all
license string (query)	Filter results by license *Available values* : all, public, share, sharecommercially, modify, modifycommercially *Default value* : all
freshness string (query)	Filter results by freshness *Available values* : all, day, week, month *Default value* : all
color string (query)	Filter results by color *Available values* : all, coloronly, monochrome, black, brown, gray, white, yellow, orange, red, pink, purple, blue, teal, green *Default value* : all
imagetype string (query)	Filter results by image type *Available values* : all, animatedgif, photo, transparent *Default value* : all
source string (query)	Filter results by source *Available values* : web, flickr, 500px *Default value* : web

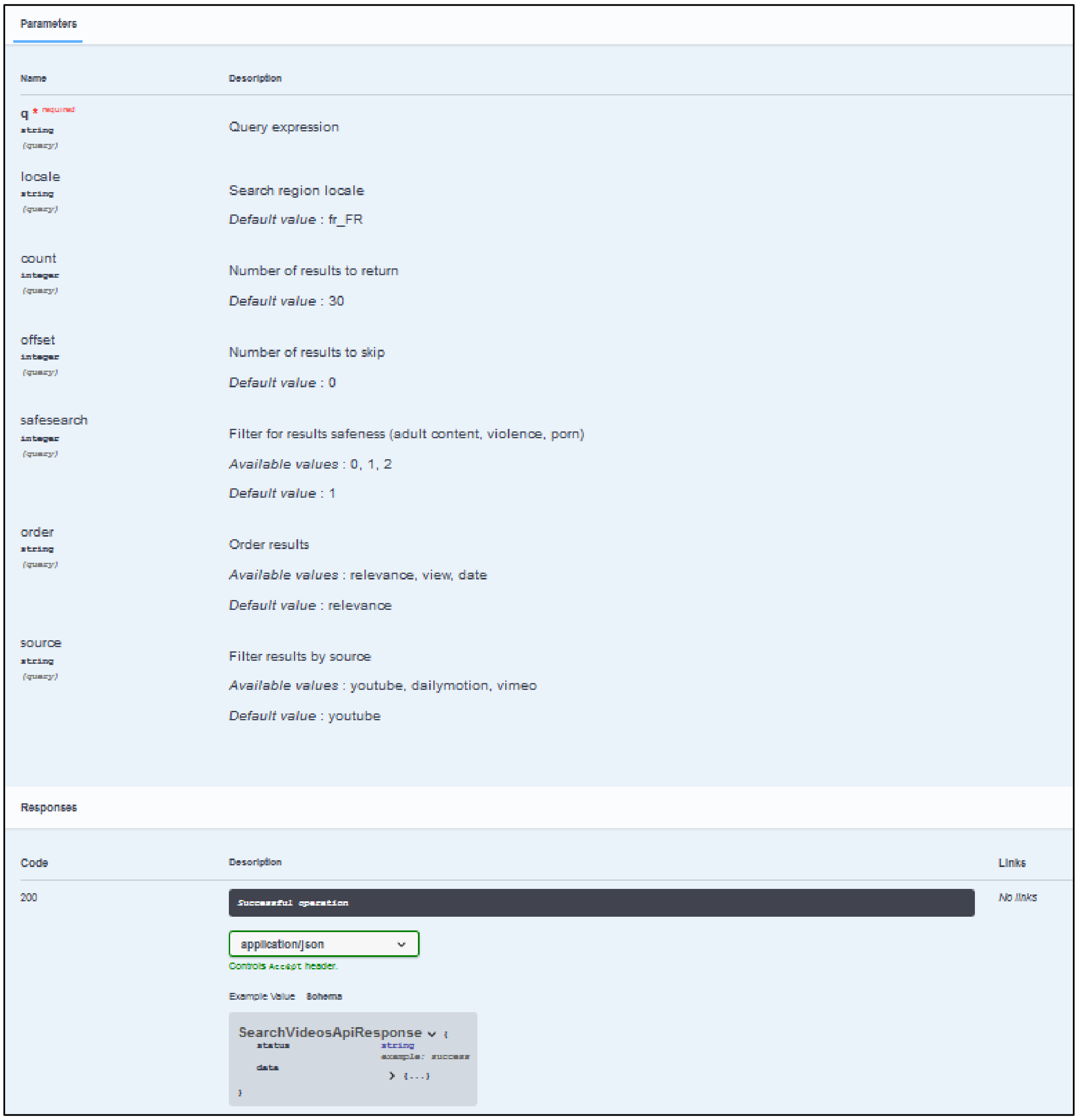
Responses

Code	Description	Links
200	Successful operation application/json Controls Accept header. Example Value Schema	*No links*

```
SearchImagesApiResponse ⌄ {
    status              string
                        example: success
    data                > [...]
}
```

Figure 11 - Qwant API to get images results

Qwant API to get video results



Parameters

Name	Description
q * required string (query)	Query expression
locale string (query)	Search region locale *Default value* : fr_FR
count integer (query)	Number of results to return *Default value* : 30
offset integer (query)	Number of results to skip *Default value* : 0
safesearch integer (query)	Filter for results safeness (adult content, violence, porn) *Available values* : 0, 1, 2 *Default value* : 1
order string (query)	Order results *Available values* : relevance, view, date *Default value* : relevance
source string (query)	Filter results by source *Available values* : youtube, dailymotion, vimeo *Default value* : youtube

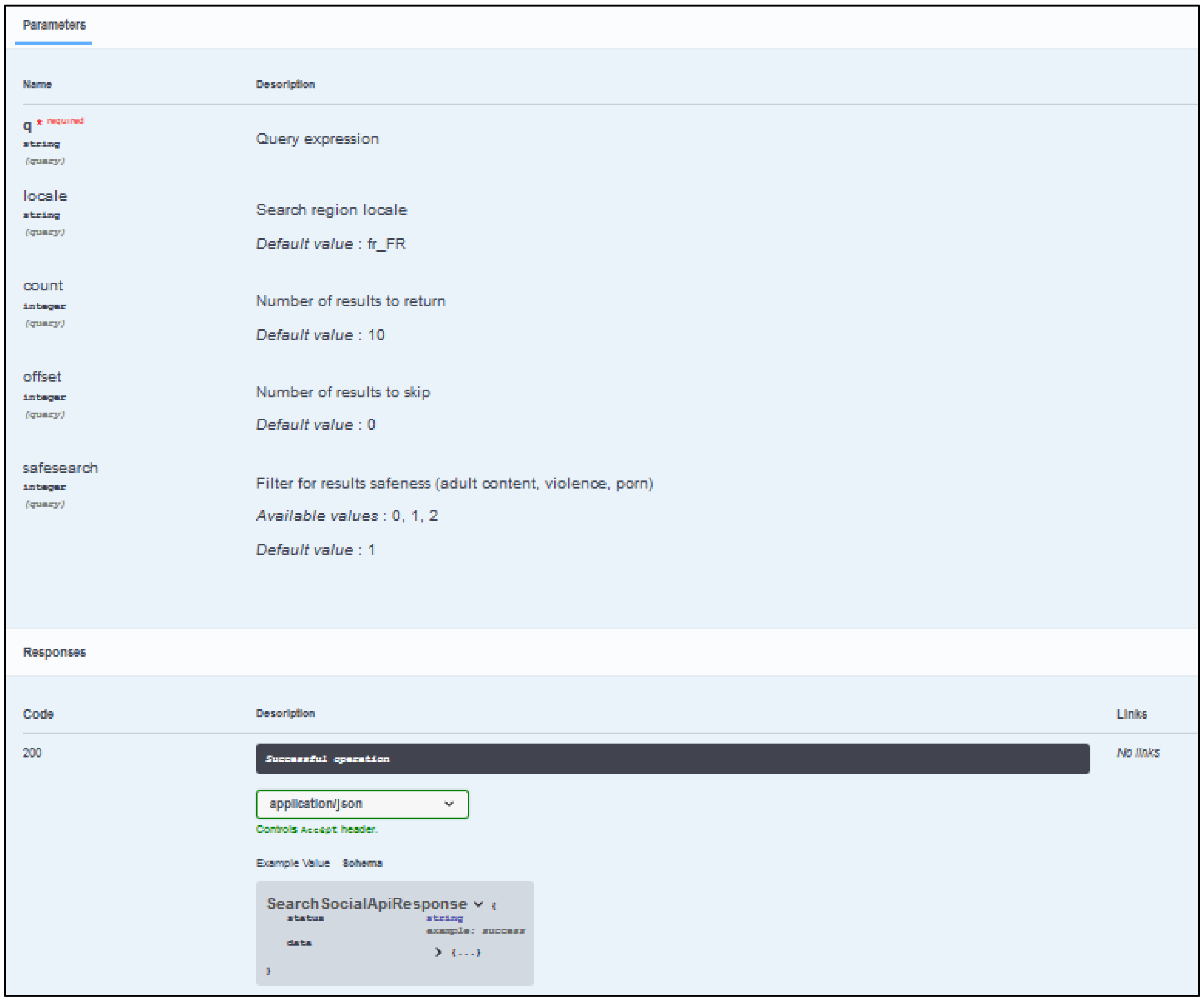
Responses

Code	Description	Links
200	*Successful operation* application/json Controls Accept header. Example Value Schema	*No links*

```
SearchVideosApiResponse {
    status          string
                    example: success
    data
                     > {...}
}
```

Figure 12 - Qwant API to get video results

Qwant API to get social (tweets) results



Parameters

Name	Description
q * required string *(query)*	Query expression
locale string *(query)*	Search region locale *Default value* : fr_FR
count integer *(query)*	Number of results to return *Default value* : 10
offset integer *(query)*	Number of results to skip *Default value* : 0
safesearch integer *(query)*	Filter for results safeness (adult content, violence, porn) *Available values* : 0, 1, 2 *Default value* : 1

Responses

Code	Description	Links
200	*Successful operation* application/json Controls Accept header. Example Value Schema	*No links*

```
SearchSocialApiResponse {
    status              string
                        example: success
    data                 > {...}
}
```

Figure 13 - Qwant API to get social (tweets) results

3.2 API to ESF solutions

Cogito® 14 is the last version of Cogito published by Cogito Labs which is a software that bases its cognitive capabilities on artificial intelligence algorithms that mimic the human ability to think at the speed of current technologies. Cogito® is made up of the following components presented in Figure 14:

- **Cogito® Discover**, is a powerful and scalable semantic content enrichment, categorization, entity extraction and text analytics software that recognizes and identifies relevant items of information hidden in text and enriches document metadata.
  It embeds syntactical, statistical, taxonomy-based and information extraction engines and supports many languages, enabling it to power high-throughput extraction applications across a wide range of use cases and geographies.
- **Cogito® Studio Express,** is a development environment that allows creating vertical thesaurus or an ontology to perform extraction and test the quality of a semantic model.
- **Cogito® Studio**, is a fully integrated development environment for building and deploying custom cognitive computing and AI-based applications, based on the Cogito Technology. As the command centre for the Cogito-powered deployments, Cogito Studio helps organizations and developers assert their business priorities while creating unique semantic-powered solutions for robotic process automation and information intelligence.

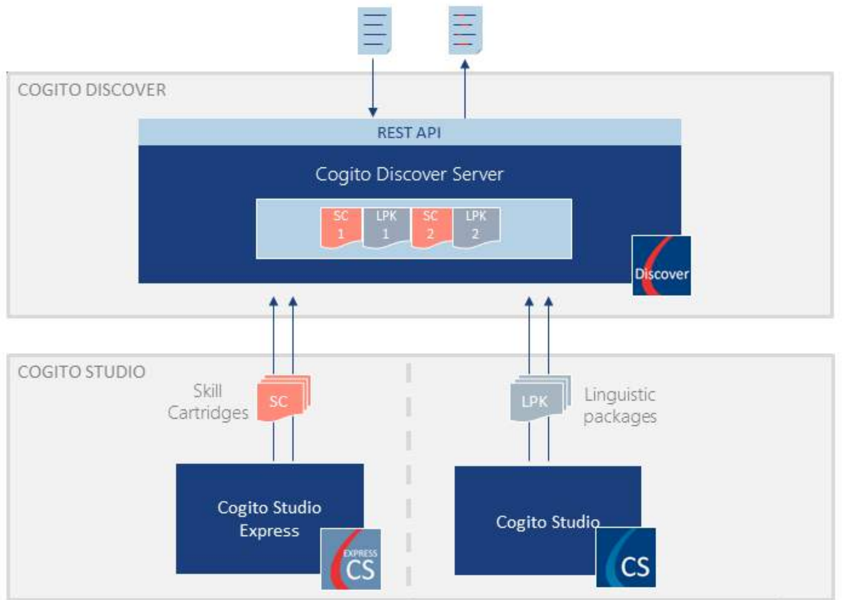


Figure 14 - Cogito components

- **Cogito® Intelligence API** Cogito Intelligence API is a server-side web application which performs semantic analyses of unstructured text documents written in English. Thanks to its underlying semantic engine, which runs on Expert System Cogito technology and is enriched with Intelligence domain-specific contents, the application provides full semantic processing features.

The service is accessible as an Application Programming Interface (API) and is distributed both as an "on-premises" web service installation and as a Java library (jar file) which can be integrated with third-party client applications (Figure 15). The system receives plain text (up to 100KB in size) containing unstructured information in input and returns text documents enriched with semantic structured metadata in order to give value and emphasis to the contents.

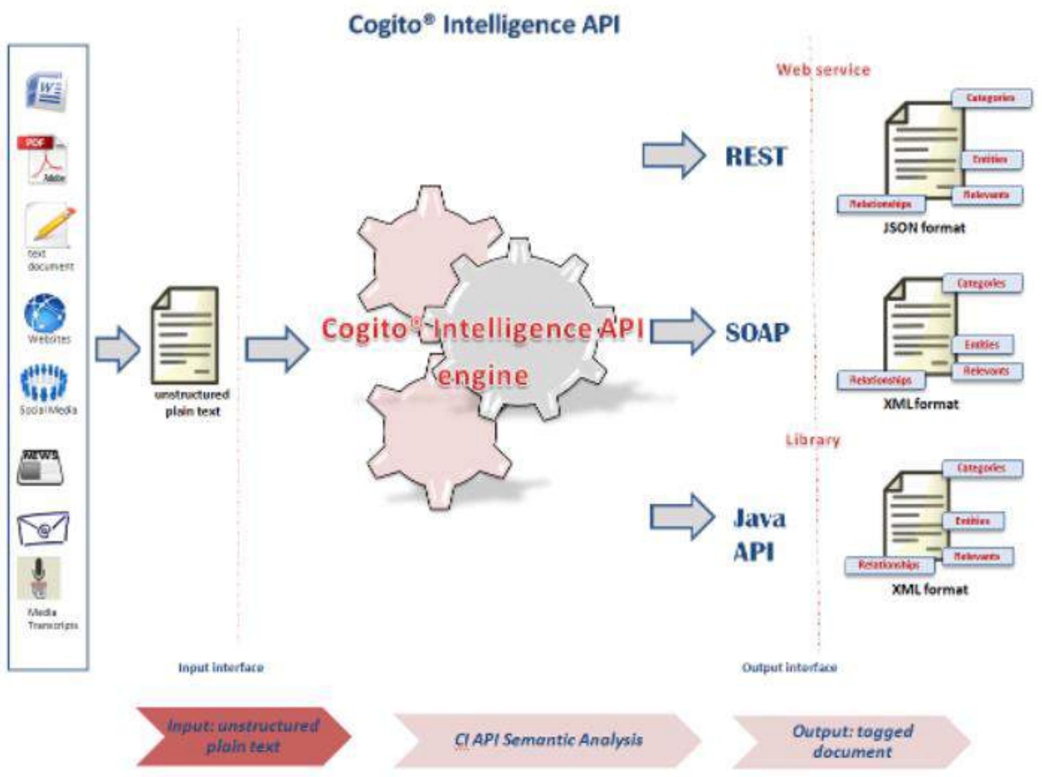


Figure 15 - Cogito Intelligence API

Currently, the application offers several analytic features:

- Content categorization with 5 vertical taxonomies (Intelligence, Crime, Cyber Crime, Geo, Emotions)
- Named Entity extraction (people, organizations and places)
- Text mining of entities (domain-specific, semantic reasoning and inferential entities)
- Relationships extraction
- Document tagging (summary and main elements)
- Fact mining
- Writeprint

An optional metadata section allows users to tag document analysis with extra data, which is forwarded "as is" from the analysis request to the response. Metadata can be used to tag analysis with supplementary data, which may be useful for subsequent client-side response processing (i.e. content author, source, publishing or acquisition date, etc.).

3.2.1 Integration

3.2.1.1. Cogito Discover

Cogito Discover is easily scalable for the number of documents that must be analysed daily and for the estimated maximum size of the global archive to ensure optimal performance.

The entry level configuration provides analysis for up to 10,000 documents per day, including the process of semantic analysis and information extraction (ETL) for each.

Entry Level Configuration	
Hardware requirements	**Software requirements**
Hardware requirements depend on the distribution of the daily load of documents to be analysed; the base configuration (10,000 documents analysed daily) requires a quad-core server with the latest generation 8 GB of RAM.	Available in both Windows and Linux environments, both for 32 and 64 bits. Developed in C++ and may be integrated into most of the newest production environments. Cogito Intelligence API

The Cogito Discover architecture is:

- Adaptable for different operational contexts.
- Reliable and scalable, both in number of users and for the number of documents.
- Easily integrated into existing systems.
- Expanded with additional modules or third-party software.

Scalability

If you require more execution power than the one defined as entry level, you can create any number of servers with the same configuration of Cogito Discover (we can call each of them Discover Analysis Unit) and then use a proxy in front of them to load balance the incoming requests (we can call this server Discover Control Unit). All client requests are sent to the Control Unit which will in turn dispatch them to a free server. When there are no "free slots" left on any server managed by the proxy (the proxy knows the number of CPU cores of each server), a round-robin policy is applied.

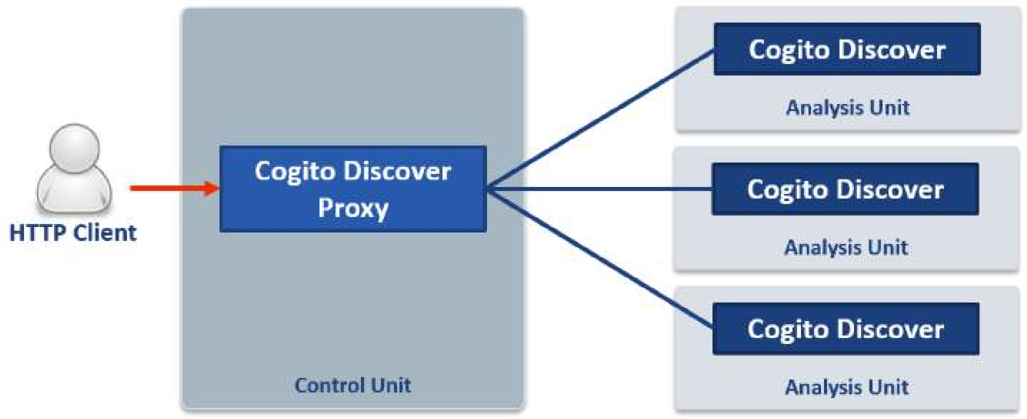


Figure 16 - Relation between Cogito Discover Proxy and other Cogito components

As presented in Figure 16, Cogito Discover comes with a Cogito Discover Proxy component you can use for this purpose (the Windows service is called "Cogito Discover Essex Proxy"). The Discover Admin web page allows you to configure network addresses and ports for your analysis nodes.

Connectors

The inherent format used by Cogito Discover is an XML representation called: Cogito eXchange format (COGX). The Cogito eXchange format is an XML-based language representation of the new Expert System document data model. The details of this format can be found in Expert System's documentation "Cogito COGX Format.pdf".

Cogito Discover manages also the main text formats (Microsoft Office documents such as DOC, DOCX, XLS, XLSX, PPT, PPTX, PDF, HTML, TXT, XML, etc.), as well as major source types (file system folders, Internet, Intranet, RSS). It is also integrated with specific connectors to acquire data from sources such as social media, news providers and document management systems already in use by the customer.

Through the use of dedicated connectors, Discover also enables analysis of text from the most common relational databases or other applications.

API

Cogito® 14 provides a new Web Service API based on the REST (Representational State Transfer) design model.

By default the REST API is available at the following URL address:

- http://localhost:8091/rest-api for Cogito® Discover
- http://localhost:8090/rest-api for Cogito® Studio Express

REST defines a set of architectural principles that can be used to design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP through a wide range of clients written in different languages. If assessed by the number of Web services that use it, in the last few years alone REST has emerged as a predominant Web service design model. In fact, REST has had such an immense impact on the Web that it has mostly displaced SOAP- and WSDL-based interface design because it is a much simpler style to use.

A specific implementation of a REST Web service follows four basic design principles:

- Uses HTTP methods explicitly
- Be stateless
- Exposes directory structure-like URIs
- Transfers XML, JavaScript Object Notation (JSON), or both

REST is an architectural style based on resources and representations of these resources. As our focus here is in a Web API, the resources are actually called "Web resources". In the REST APIs, Web resources are mainly either collection resources (e.g. list of users), collection member resources (e.g. one of the users in the list) or processing resources (e.g. resource called by an online calculator when form data is submitted in order to get results). For example:

- Collection resource: http://www.mywebsite.com/users
- Collection member resource: http://www.mywebsite.com/users/johndoe
- Processing resource: http://www.myonlinecalc.com/calculate

A REST API uses HTTP verbs to create, modify, use, or delete these resources. The HTTP verbs are GET, POST, PUT, and DELETE.

For more details on Cogito Discover API please refer to Expert System’s documentation “Cogito REST API.pdf” or use the Postman collection “Cogito Discover.postman_environment.json”.

3.2.1.1. Cogito Intelligence API

The web service version of Cogito Intelligence API can be accessed by either SOAP or REST interface. The two service interfaces coexist under the same web service distribution and differ as outlined in the table below:

Protocol	**SOAP**	**REST**
Serialization format	XML	XML/JSON
Functionalities selection time	None	Request phase
Functionalities per analysis	Always complete	one per request (“Full” feature also available)

For the SOAP service, the all functionalities are always enabled. All functionalities are included within the same analysis response to the client, for each analysis request; in this way the customers get a complete analysis of the text with just one single client application request.

For the REST service, a feature analysis corresponds to each exposed endpoint. For each text sent by the client, an analytical functionality is implicitly enabled, and the result of the analysis is delivered to the customer, thus providing a greater modularity in the selection of functionalities. A "full" analysis feature is also available, thus allowing customers to obtain a fully featured analysis of documents with a single analysis request.

As for the library distribution for integration into third-party Java applications, the output format can be either a data-structure JSON or XML String serialization, and the selection of analytical functionalities is allowed programmatically. The library consists of a "jar" archive that can be imported into applications together with other libraries from which the same Cogito Intelligence API library depends.

The Installation Guide is distributed along with Cogito Intelligence API and a Postman collection "Cogito Intelligence API.postman_collection.json" is available to help users requesting the API.

3.3 API to Thales solutions

3.3.1 Attribute-based Access Control

Thales has developed a unique solution of fine grained access control (AC) via Attribute Based Access Control (ABAC) using Blockchain (BC). ABAC is a logical AC model that controls access to objects by evaluating rules against the attributes of entities (subject and object), operations, and the environment relevant to a request. It enables more precise AC by allowing for a higher number of discrete inputs into an AC decision and thereby providing a larger set of possible combinations of those variables to reflect a larger and more definitive set of possible rules to express policies, which are limited only by the computational language and the richness of the available attributes. Using ABAC, the access decisions can change between requests simply by altering attribute values, without requiring changes to the subject/object relationships defining the underlying rule sets.

Thales Attribute based Access Control via BC is performed following two steps. The first step consists of the registration of all devices in the blockchain via smart contracts. The second step consists of the request to access to a resource managed by the blockchain. In the first step, every entity managed by the blockchain must create a smart contract (SC) to register. In this smart contract, information will be specified namely the identity of the entity, its attributes (in the ABAC) and access rules, address of its TrustStore, as well as a digital signature to ensure the authenticity of the data in this smart contract. This step requires the creation of a transaction each time an entity joins the network or updates its identity information. In the second step, the entity is granted access once its attributes and access rules have been checked. In terms of transaction cost, only the first step requires the creation of a transaction; while the second step does not require a new transaction.

In the following, we illustrate the main functionalities of the ABAC system in Figure 17. At the step (1) the user requests access to a resource through a "zero-knowledge" proof authentication mechanism based on Challenge/Response. After receiving the request, the resource will send a challenge (2). In step (3), the user will sign the challenge. Then, it will send the signed challenge and the address of the SC (step 4). Steps 1, 2, 3, and 4 are also the result of identity and authentication check done in the previous check of authentication.

The resource will request the address of the owner of the SC (5, 6, 7) in the register. This register consists of a new smart contract with an associative table having as keys the Ethereum addresses of entities, and as variables Ethereum addresses of associated SCs.

At step (8), the resource will compare between the address of the signed challenge and the address of the owner of the SC. Then there should be verification of the smart contract of the resource whether it has been certified by the authority (9, 10, 11, 12) as the consensus here is Proof of Authority. After that, the resource will verify whether the UE or User is authorized or not. During an access request, the function isAuthorized verifies at each time if the requester has an ISC certified by a trusted entity.

During the creation of the Smart Contract, the owner will define the Ethereum address of the TrustStore including all the trusted entities authorized to sign the SC. Concretely, this TrustStore is a smart contract containing a table of Ethereum addresses and offer the possibility, via the developed functions, to add, to delete and to verify an address.

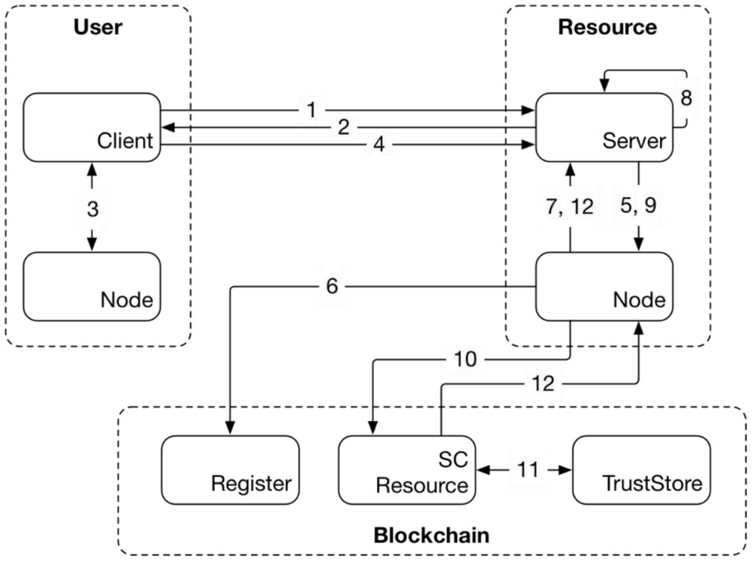


Figure 17 - Architecture of Access Request between User/Resource via Blockchain Network.

The ABAC system offers to every entity to publish or modify attributes of their identities and/or its access rules in the blockchain. Moreover, entities could access/retrieve access rules of other entities from the blockchain; and could retrieve signatures associated to an identity. A user or client could sign a message with its private key of its Ethereum account. Moreover, all data stored in the Ethereum BC could not be modified without authorization. Also, attributes and access rules of a smart contract could only be modified by its owner (owner of the smart contract).

3.3.2 Description of the API and the interface

We present the following requirements and specifications related to the smart contract.

- The identity of an entity is represented by a smart contract. Every SC has a function addAttribute.
- Every smart contract (SC) has a function sign.
- A function sign has a parameter to specify its expiration date.
- Every SC has a function isAuthorized. This function takes into parameter the address of the SC and returns a Boolean in order to indicate whether the owner of that SC is authorized to access or not.
- Every SC has a function addRule in order to publish/modify their access rules in the blockchain.

The resource could add attributes, add access rules and also sign the SC. The client also could sign a message with a private key of its Ethereum account. Attributes and access rules of a SC could be modified only by its owner. The ABAC system allows to pay only the registration and the modification of the SC. The consultation of the attributes, access rules or the call to the function isAuthorized does not incur a new transaction.

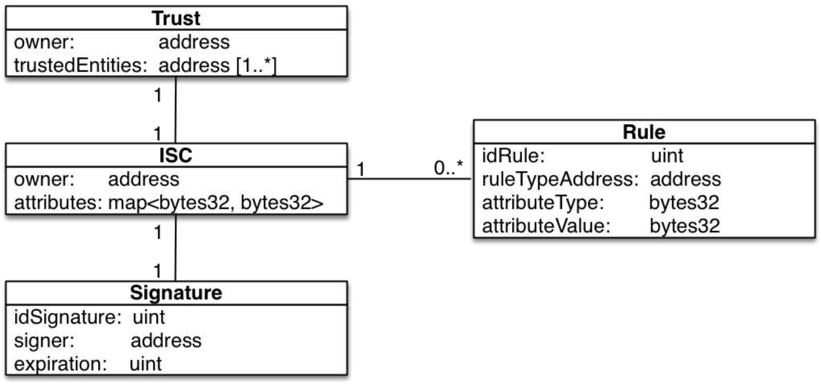


Figure 18: Simplified entity-relation diagram of an identity SC.

In Figure 18, we present a simple model (Entity-Relation Diagram) of an identity SC (ISC). Since every ISC has one owner, this model proposes a variable owner in the smart contract ISC. This variable refers to the Ethereum address of the owner. In order to have a valid transaction, this later should be signed with the private key associated to the Ethereum address of the owner.

Regarding the access rules, our model defines a new data structure rule with the following variables:

- idRule: an integer not signed that attributes to each rule a unique identifier.
- ruleTypeAddress: an Ehereum address corresponding to a type of rule.
- attributeType: the type of attribute.
- attributeValue: value of the attribute.

A modular architecture has been established in order to add type of rules easily. For the sake of simplicity, we present here three examples of the types of rules:

- MatchRule: type of rule returning true only when two given attributes are equal.

- TimeGreaterThanRule: type of rule returning true only when the actual time is superior than a given time in parameter.
- TimeLess ThanRule: type of rule returning true only when the actual time is less than the given time in parameter.

When a SC is published in the blockchain, it is necessary to guarantee to the other entities present in the network that all the information contained in the SC is correct. Hence, a signature mechanism is used.

3.4 Interoperability/Integration aspects

Due to the early stage in the project and because not all tasks involved in development have started yet, this section is not complete at the moment. This section will be more detailed in D2.3.

In order to deliver, in WP5, the different elements to be integrated for building the different use cases, we have to detail some integration principles in order to anticipate the usual integration issues when different actors are involved.

The first principle will be about the software integration, it means to ensure code quality, test the developed product and to track the changes made by different developers. The consortium has to define, for example, the coding standards to follow, to ensure that the code is readable and understandable by different developers involved in the process. Repositories, where the different versions committed by the developers are stored, are used to track the changes in software.

The second principle will be about components integration. In this case, this aspect will detail how to choose the standards for the new components and how to take into account the legacy components.

3.5 Blockchain aspects

3.5.1 SocialTruth Blockchain implementation

To answer trust requirements for SocialTruth results storage, SocialTruth has to select a specific Blockchain implementation among existing ones. This implementation will be reused as-is or with some modifications if needed. This implementation will also be used by the specific Thales access control besolution (based on ABAC approach) using the Blockchain described in this chapter. As of this day, the solution uses Ethereum implementation but will have to use SocialTruth Blockchain.

The Blockchain integration will depend on the selected blockchain. The main aspects to select one are:

- The deployment model,
- Smartcontracts or not,
- Database for storing the transaction contents,
- Data Confidentiality,
- Mode de deployment,
- Nodes deployment.

The justification for the selection of the Blockchain implementation will be described in SocialTruth deliverable D4.1 (M17). For now, various choices were already made:

1. The deployment model, the most appropriate solution seems to be a consortium blockchain (a deployment limited to the consortium).
2. Due to the transaction complexity, a Smartcontracts is mandatory.
3. Due to the volume of data and the GDPR rules, an external database has to be used but with access control by the Blockchain.
4. For the Data Confidentiality, we have not identified specific constraints.
5. The nodes (access points for the Blockchain) deployment will depend on the Blockchain implementation and, depending on the final architecture, a cloud deployment could be envisaged (all Blockchain implementations have this possibility)

As of this day, Thales is investigating various implementations, which are:

- Multichain[2]
- Hyperledger fabric (mature solution without risks)[3]
- Ethereum (fallback solution)[4]

3.5.2 Blockchain interface

The interface with the Blockchain will be based on smartcontract allowing the following actions:

Read:

Input:

- Content signature and type of content: identifying the content itself (to be defined).
- URL of the content, the declared author(s),

Return:

- SocialTruth score (global result).
- SocialTruth details score (details about the different verification services applied).
- Score history (list of different score resulting of different evaluations).
- Context of the content(s) (training, journalism, advertisement, …).

Write:

Input:

- Requester identifier (who want to perform an evaluation)
- Content signature and type of content: identifying the content itself (to be defined).
- URL of the content, the declared author(s),
- Context of the content(s) (training, journalism, advertisement, …)
- SocialTruth score (global result).

[2] https://www.multichain.com/
[3] https://www.hyperledger.org/projects/fabric
[4] https://www.ethereum.org/

- SocialTruth details score (details about the different verification services applied).
- Score history (list of different score resulting of different evaluations).

Return:

- Error code if error

Update:

Input:

- Requester identifier (who want to perform an evaluation)
- Content signature and type of content: identifying the content itself (to be defined).
- The URL of content, the declared author(s),
- Context of the content(s) (training, journalism, advertisement, …)
- SocialTruth score (global result).
- SocialTruth details score (details about the different verification services applied).
- Score history (list of different score resulting of different evaluations).

Return:

- Error code if error

This interface description is a very preliminary definition of the interface and will be update as soon as the final architecture will be consolidated.

3.6 Containers/dockerization

The increasing popularity of the container technology can be attributed to the release of Docker in 2013 along with offering a high-level abstraction of the isolated execution environments.

Docker is an open source project that is aimed at simplifying deployment of an allocation by means of containers. It allows for building an image with the application deployed inside. A running instance of an image is called a container. The image contains all the dependencies that the application needs to run (e.g. operating system, runtime environment, specific system libraries, etc.). The image can be easily run anywhere (on the variety of host operating systems) executing the application in an isolated environment. The containerisation differs from hardware virtualization in the way that it has higher performance (containers do not emulate the entire computer architecture), lower resource consumption, and smaller images (containers do not require a full operating system).

From the SocialTruth perspective, in particular in the light of the architecture-related challenges listed in section 2.2, use of containers will bring significant benefits into the process of SocialTruth platform development and deployment. Service-oriented or microservices approach to the architecture design combined with containerized applications eases scalability. Other advantages of the container technology also include simple dependency management and application versioning, lightweight deployment, and solving such challenges as runtime environment configuration, isolation, and portability. In addition, the

Docker images can be easily run anywhere (on the variety of host operating systems) executing the application in an isolated environment. This enhances the capabilities to ease the adoption of the final solution by the future end-users.

4 Security and privacy aspects

4.1 Authentication and authorization

The authentication and authorization in SoA or a microservices-based architecture imposes several challenges in comparison to the monolithic architecture, where implementation of the authentication and authorization mechanisms is straightforward. Usually in such architectures a dedicated security component provides AA implementation. Authenticating user access and authorizing their actions requested to application is considered as a single process. Designing an architecture based on microservices imposes additional challenges, from which one of the most important is implementation of secure and efficient authentication and authorization mechanisms with preservation of an overall flexibility of microservices:

- Each service in a microservice-based approach is considered as separate process and requires separate implementation of business logic, therefore access to each microservice needs to be authenticated and authorized independently,
- Global logic implemented independently in each microservice (repeated in each micro-application) brings challenges in the system maintenance, implementation of changes and updates,
- Challenging AA implementation in more complex scenarios, when the microservice-based application is connected with 3rd party solutions, extended by adding new services or in the case where multiple microservice applications start accessing one another.

Taking into account the abovementioned challenges and the overall architecture of the SocialTruth platform, at this stage of the project we consider three main options for the implementation of AA mechanisms in the microservices environment:

1) Distributed session management – implementation of a basic distributed session management in the microservice context may violate its fundamental assumption about statelessness. However, there are several approaches that can be effectively used to manage a distributed session:
    a. “sticky” authentication session (session affinity) - all user requests are sent and managed by an authentication server that handles the first request corresponding to a given user and ensures that session data is always correct for this user, or
    b. session replication - each instance saves all session data and synchronizes through the network,
    c. centralized session storage – when user accesses a microservice, user data can be obtained by other microservices from shared session storage

   However, each of those approaches has drawbacks - sticky session requires a load balancer and the actual session data can be lost if the user was to be shifted to another server, session replication brings issues related to the network bandwidth usage (continuous synchronization), while centralized session storage requires additional security mechanisms to protect the shared sessions.

2) Client token and token with the API gateway - in a basic implementation, the token holds the information on user identity and is sent to the server at each user request. This allows server to determine the identity of a visitor and the legitimacy of his request. Tokens are held by the users themselves in the form of browser cookies, therefore the server does not save the status of the users. The content of the token is based on the predefined schema/template and needs to be encrypted to avoid falsification by the requester or the third party. The extension of the basic client token approach is hiding microservices behind the API gateway providing entrance of the external requests to the system, translating basic transparent client tokens into opaque tokens.
3) SSO – Single Sign-On - with the use of a single sign-on server that mediates between end-users and the accessed resource server. To implement a single sign-on in a microservice-based architecture, additional identity and access management microservice needs to be used. This microservice is responsible for the generation of SSO token for the user, and for validating this token received from the resource server after the user's request. The drawback of this solution is the excessive network traffic related to the communication between the user and the resource server, the user and the SSO server, and the resource server and the SSO server. In addition, access related functionalities in microservices environment can be realized through mutual SSL/TLS certificate-based authentication, while in the scenarios requiring an access of 3rd party applications to the microservices - using an API token approach or an Oath open reference architecture based on open standards.

4.2 Privacy by design and by default (GDPR), security by design, link to D7.1

One of the objectives of this document is to ensure, in the design phase of the architecture, the respect of the "security by design" and the "privacy by design" principles. Article 25 of EU General Data Protection Regulation ("Data protection by design and by default") provides that *"the controller shall, both at the time of the determination of the means for processing and at the time of the processing itself, implement appropriate technical and organisational measures, such as pseudonymisation, which are designed to implement data-protection principles, such as data minimisation, in an effective manner and to integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation and protect the rights of data subjects."*

The provisions contained in article 25 make it clear that the approach of the GDPR is centered on risk assessment (risk based approach), which determines the extent of responsibility of the data controller taking into account the nature, of the scope, the context and the purpose of the processing, as well as the likelihood and severity of the risks for the rights and freedoms of users. That said, to better understand these principles it is appropriate to also consider the content of clauses 75 and 76 of the GDPR, which include also the definition of risk. In particular clause 76 reports that "The likelihood and severity of the risk to the rights of the data subject should be determined by reference to the nature, scope, context and purposes of the processing. Risk should be evaluated on the basis of an objective assessment, which is established whether the data processing operations involve a risk or a high risk "

The privacy by design obligation is based on risk assessment, as well as other obligations (e.g. a notification to the national guarantors), for which the risk inherent in data processing must be assessed. This evaluation should be done at the time of system design, so before the treatment begins. Clearly, the type of data processed must also be considered, so that in the presence of a treatment involving child data the obligations must be more stringent, in consideration of the fact that the risk is greater.

The risk-based approach means that the state of the technology must be taken into account, so the treatment must be adapted over time.

The cardinal principles of privacy by design are the following:

- To prevent and not to correct - problems must be assessed in the design phase, and the architecture must prevent the occurrence of risks;
- privacy as the default setting (for example, it must not be mandatory to fill in a field of a form in which the provision of data is optional);
- privacy must be embedded in the project (for example, the use of pseudonymisation techniques or data minimization);
- maximum functionality, in order to respect all the needs (rejecting false dichotomies such as more privacy = less security);
- security throughout the product or service cycle;
- visibility and transparency of data processing, i.e. all the operational phases must be transparent to ensure that the data protection is verifiable;
- user-centered, therefore assuming respect for rights, timely and clear responses to the user's requests for access.

Moreover, Article 32 of the GDPR (Security of processing) establishes some fundamental principles. In particular, the security measures must be prepared "taking into account the state of the art and the costs of implementation, as well as the nature, object, context and purpose of the processing, as well as the risk of various probabilities and severity for the rights and freedoms of individuals ".

The security measures, therefore, must be adequate, imposing not an obligation of result, but an obligation of means, so that the measures are reasonably satisfactory in the light of knowledge and practices.

In fact, security does not only concern the IT aspect of data processing, but also the organizational aspect to cover events such as the removal or loss of documents. The security measures, therefore, must guarantee that:

- the data can be consulted, modified, disclosed or deleted only by the persons authorized to do so (and that such persons act only within the authority granted to them);
- the processed data are accurate and complete in relation to the reason why you are processing it;
- the data remain accessible and usable, that is, in the event of accidental loss, modification or destruction, you must be able to recover them and prevent damage to the persons concerned, preparing an appropriate business continuity plan.

The security principle, therefore, provides for the obligation of confidentiality, integrity and availability of data.

In deliverable D7.1, besides providing detailed information on the application of ethical standards and guidelines of Horizon2020 within the SocialTruth consortium, the general principles and procedures related to data protection have been outlined, so that SocialTruth can guarantee that all legal provisions of personal data and location-aware services will be respected during all the phases of the project:

- SocialTruth will minimize the collection of and processing of personal data and will use anonymization techniques to remove the ability to identify individuals wherever possible, depending on the nature of experiment.
- Where de-identifying is not possible or desired, SocialTruth will protect all data collected that can be attributed or traced to an individual. It will store such data only with consent.
- Based on consent, personalized data can be transferred into anonymized data records for wider dissemination after the project lifetime and for further exploitation.
- Use of secure data storage, encrypted transfer of data over the capturing channels, controlled and auditable access for different classes of data.
- Obscuring/removing user identities at the source of experimental data generation to prevent direct user tracing.
- Obscure the location as much as possible and limit user tracking through correlation of de-personalized data based on its location.
- Personal data will be processed in compliance with the relevant legal regulations. Personal data will be collected on a strictly need-to-know basis, solely for the purposes of the SocialTruth project and will be destroyed when no longer needed for that purpose. Technical and operational measures will be implemented to ensure that users will be able to access, rectify, cancel and oppose processing and storage of their personal data.

It is anyway evident that one of the guiding principles of article 32 of the GDPR is the demanding that IT security be brought up to the level of "state of the art". It is then necessary to understand what exactly can be considered “state of the art” in IT security. In this matter, ENISA and TeleTrusT - IT Security Association Germany, have recently published a guideline document [5] intended to provide companies, providers (manufacturers, service providers) alike with assistance in determining the "state of the art" within the meaning of the IT security legislation. Based on ENISA and TeleTrusT document, the “state of the art" technology level is situated between the more innovative “existing scientific knowledge and research" technology level and the more established “generally accepted rules of technology" level.

These three states of technology are flanked by the categories “general recognition" and “proven in practice." The classification of the laws requires a clear distinction between subjective and objective criteria. The “state of the art" criterion is purely objective. The subjective aspects consider the laws in the event of an offence; however, they do not concern the definition of the “state of the art" itself. As a result, the “state of the art" can be described as the procedures, equipment or operating methods available in

[5]https://www.teletrust.de/fileadmin/docs/fachgruppen/2019-04_TeleTrusT_Guideline_State_of_the_art_in_IT_security_ENG.pdf

the trade in goods and services for which the application thereof is most effective in achieving the respective legal protection objectives. In short it can be said that the “state of the art" describes a subject’s best performance available on the market to achieve an object. The subject is the IT security measure; the object is the statutory IT security objective.

With regard to the “state of the art” guidelines for the implementation of the security measures in the development of applications, there is no doubt that one of the most authoritative reference points is represented by OWASP: The Open Web Application Security Project is an open-project source for Web application security. The OWASP also offers guides with tips on creating secure Internet applications, and directions for the tests they should be submitted to. That said, to ensure the state of the art IT security principles and techniques are adopted, during the design and the implementation of the SocialTruth platform, Security-by-Design principles defined by OWASP (Open Web Application Security Project) will be followed.[6]

The secure design principles contained in OWASP Dev Guide Principles of Security Engineering include:

- **Defense in Depth**: a security principle where single points of complete compromise are eliminated or mitigated by the incorporation of a series or multiple layers of security safeguards and risk-mitigation countermeasures. Have diverse defensive strategies, so that if one layer of defense turns out to be inadequate, another layer of defense will hopefully prevent a full breach;
- **Fail-Safe**: aims to maintain confidentiality, integrity and availability by defaulting to a secure state, rapid recovery of software resiliency upon design or implementation failure. In the context of software security, fail-secure is commonly used interchangeably with fail-safe, which comes from physical security terminology;
- **Least Privilege**: a person or process is given only the minimum level of access rights (privileges) that is necessary for that person or process to complete an assigned operation. This right must be given only for a minimum amount of time that is necessary to complete the operation: proper granularity of privileges and permissions should be established;
- **Separation of Duties**: if the successful completion of a single task is dependent upon two or more conditions that need to be met, just one of the conditions will not be sufficient for the completion of the task;
- **Economy of Mechanism**: the number of vulnerabilities increases with the complexity of the software architectural design and the number of lines of code. By keeping the software design and implementation details simple, the attackability or the attack surface of the software is reduced;
- **Complete Mediation**: ensures that authority is not circumvented in subsequent requests of an object by a subject, by checking for authorization (rights and privileges) upon every request for the object;
- **Open Design**: the implementation details of the design should be independent of the design itself, which can remain open, unlike in the case of security by obscurity wherein the security of the software is dependent upon the obscuring of the design itself;

[6] https://www.owasp.org/index.php/OWASP_Secure_Application_Design_Project

- **Least Common Mechanism**: the least common mechanisms principle disallows the sharing of mechanisms that are common to more than one user or process, if the users and processes are at a different levels of privilege;
- **Psychological acceptability:** ensures that the security functionality is easy to use and at the same time transparent to the user. Ease-of-use and transparency are essential requirements for this security principle to be effective;
- **Weakest Link:** resiliency of the software against hacking attempts will depend heavily on the protection of its weakest components, be it the code, service or the interface;

- **Leveraging Existing Components:** ensures that the attack surface is not increased and no new vulnerabilities are introduced by promoting the reuse of existing software components, code and functionality.

Moreover, it will be appropriate to follow the "OWASP Checklist for Securing Application Design[7]" as a guideline for the security aspects of the design. The checklist contains provisions regarding the following aspects:

- Design:
  - code Flow;
  - authentication and access control mechanism;
  - data access mechanism;
  - centralized validation and interceptors;
- Architecture:
  - Entry points;
  - External integration;
- Configuration;
  - External API's used;
  - Inbuilt security controls.

It will also be appropriate to use "**Owasp Application Security Verification Standard v4.0.1 (March 2019)[8] "** to validate the implementation of the security requirements: the Application Security Verification Standard is a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, tool vendors, and consumers to define, build, test and verify secure applications, structured as following:

- Architecture, Design and Threat Modelling Requirements;
- Authentication Verification Requirements;
- Session Management Verification Requirements;
- Access Control Verification Requirements;

[7] https://www.owasp.org/images/f/f7/Checklist_For_Design.pdf
[8]https://github.com/OWASP/ASVS/raw/master/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0-en.pdf

- Validation, Sanitization and Encoding Verification Requirements;
- Stored Cryptography Verification Requirements;
- Error Handling and Logging Verification Requirements;
- Data Protection Verification Requirements;
- Communications Verification Requirements;
- Malicious Code Verification Requirements;
- Business Logic Verification Requirements;
- File and Resources Verification Requirements;
- API and Web Service Verification Requirements;
- Configuration Verification Requirements.

5 Socio-technical and human factors

In the frame of the social and human aspects described in detail in deliverable D2.1, with the purpose of building engaging and user-centred services, this chapter aims at investigating the optimized Socialtruth's human-machine interfaces for targeted end-users, based on the use cases identified in Task 2.1, and on current industrial and commercial best practices. Special focus is given on the psychological aspects of the design, or in other words the way that these interfaces encourage and motivate specific actions or behaviours.

This will represent the base for the future definition of the interface specifications; In fact, the proposed HMIs will be proposed to test groups of end-users from the organizations of the SocialTruth partners (ADNK, INFOC, DEASC), to assess alternative "look and feel" style options and provide guidelines to ensure not only the aesthetic part but also usability, learnability, efficiency, intuitiveness, and user satisfaction aspects of the provided solutions, to be included in deliverable D2.3.

5.1 Introduction

In the following sections of this introductory chapter, the basic recommendations for effective web interface design are recalled, based on psychology-driven concepts found in literature. The Gestalt principles used to establish visual hierarchy within web interfaces are also presented. Finally, guidelines regarding the brand cultivation and the use of colours are proposed.

After this, the document proceeds with investigating possible approaches to the development of human-machine interfaces for SocialTruth, tailored to the use-cases identified in the project proposal and further defined in D2.1 Chapter 6 "Socio-technical & Human Aspects", namely:

- use case 1: "Checking sources in the production process";
- use case 2: "Digital companion for content verification";
- use case 3: "Search engine rankings & advertising prevention for fraudulent sites";
- use case 4: External sources reliability check in the educational domain".

For each use case, guidelines regarding the interface's look&feel are proposed, and the interface design affordances are translated into preliminary functionality requirements, to be further developed, validated and adjusted as the project proceeds.

5.1.1 Basic usability recommendations

As reported in the relation "The Psychology Behind Web Design" by Sarosha Imtiaz, expert in data-driven design & marketing strategies at the Department of Life Sciences of the McMaster University (Hamilton, Ontario, Canada) (Imtiaz, 2016), choosing the right features for interfaces and websites is a key point for their success and for assuring a pleasant and satisfying user experience. Well-known and general usability recommendations and guidelines represent **solid foundations to base the development of an effective interface on**. As reported by Imtiaz, by adopting psychology-based design tactics, website owners can provide an engaging experience for their users (Elder & Krishna, 2012). A website should accurately depict its products to encourage mental interaction from their visitors (Verhagen, Boter, & Adelaar, 2010). Once users can envision themselves interacting with a product, they become more likely to purchase it (Elder & Krishna, 2012).

The afore-mentioned psychology-based design tactics are mainly reported to be the following (Fadeyev, s.d.):

Table 2 - Psychology-based design tactics

FEATURE	DESCRIPTION
clearness	The main purpose of a user interface design is to enable people to interact with the system or tool by communicating its meaning and functions. It is essential that people understand how the application works and where to click. It is mandatory to prevent people from getting frustrated or confused. example: provide tooltips on buttons, popping up to explain the button's function
conciseness	Clearness requires explanations, but too much explanation makes the interface "heavy". People will not bother spending too much time trying to understand how the application works and reading instructions. Thus, clearness must go hand in hand with conciseness. example: label buttons with only one word
familiarity	The application has to appear familiar to the user, so that he or she intuitively understands how it works. This can be achieved by making the application look like something that users already know. example: tabs on folders are something every digital user can recognise. People immediately understand that clicking on a tab will allow navigating that particular section and that the rest of the tabs will remain there.
responsiveness	The interface should work fast: people get easily frustrated when waiting for content to load. If the content needs time to load, at least it is essential that the interface loads quickly (even if the content is yet to catch up) and provides some form of feedback about what is happening. example: buttons should display a 'pressed' state when users click on them, a "loading…" pop up should show with a spinning wheel or a progress bar to keep the user in the loop while the content loads
consistency	Interfaces should be internally consistent, so that the user can develop usage patterns and recognize at a glance what the different buttons, tabs, icons and other interface elements look like and what they do.

	example: different versions of the same application, for example desktop version vs. mobile version, should be consistent with each other
aesthetics	A good interface should be aesthetically pleasant and enjoyable, providing a more satisfying user experience. The fashion and the look&feel should be tailored to the specific audience. example: tabs with rounded corners and buttons with subtle gradients and pixel thin highlights look just fine
effectiveness	Interfaces are the thing that allows people to perform the functions of the application. The key point here is to understand what the user is trying to achieve and let him/her easily accomplish the task instead of simply implementing access to a list of features. example: if a lot of users want to share some content from the application to social media, provide an easy way (e.g. button) that immediately allows them to do so
recovery	A good interface should allow users to handle mistakes, undo actions and retrieve past information. Also, it should provide help to the user, rather that cryptic error messages. example: when someone navigates to a broken or nonexistent page, they should get a helpful list of alternative destinations

5.1.2 Gestalt principles

The Gestalt principles are a set of principles in psychology to account for the observation that humans naturally perceive objects as organized patterns. Human-machine interface designers can leverage human psychology when designing layouts by using these principles to establish visual hierarchy (Demangeot & Broderick, 2007).

As reported by Djamasbi *et al.* (Djamasbi, Siegel, & Tullis, 2011), visual hierarchy revolves around **eight aspects**: **size**, **contrast**, **similarity**, **symmetry**, **unity**, **grouping**, **style** and **colour**:

Table 3 - Gestalt principles

GESTALT PRINCIPLE	DESCRIPTION
size	Size is a primary technique to show dominance on a screen: bigger and bolder objects are perceived as more prominent.

contrast	Contrast refers to the comparison between colours: high contrast refers to colours that easily stand apart from each other while low contrast refers to colours that do not.
similarity	Similarity refers to the perception of objects in relation to their size, colour, proximity and shape compared to others.
symmetry	Visual symmetry is reported to help users remember content better, while asymmetrical designs can focus interest and on a certain section or object.
unity	Unity refers to holding designs together both visually and conceptually. This may include the overall colour and layout, as well as the arrangement and content of information displayed on the screen.
grouping	Grouping refers to reducing the cognitive load on the user by providing immediate separation of a group of elements from the rest of the page.
style	Additional emphasis can be placed on one piece of information over another by changing its style.
colours	Colours are a powerful communication tool and can be used to invoke action, influence one's mood, or even cause physiological reactions.

The SocialTruth application is intended not as a stand-alone product, but rather as an integrated part of existing environments (editorial board for journalists, web-browser for regular citizens, etc. based on the identified use-cases). This puts additional requirements for the effectiveness of the proposed solution(s), regarding the coherence between the application and the hosting environment. **The integration should look natural and discreet, fostering a habits-driven attitude of the user towards it**, just as if the new tools and services had always been there. Thus, the solution should leverage the habits already established by the user within the hosting environment and **retrace the existing Gestalt principles-driven patterns**.

5.1.3 Brand emphasis and use of colours

Every SocialTruth product, website, brochure, application etc. should be consistent with the others and **recognizable as part of the SocialTruth environment**, as part of an overall strategy of **brand cultivation**.

The SocialTruth's Logo has already been established with an agreed-upon shape, font and colour (as presented in Figure 20):

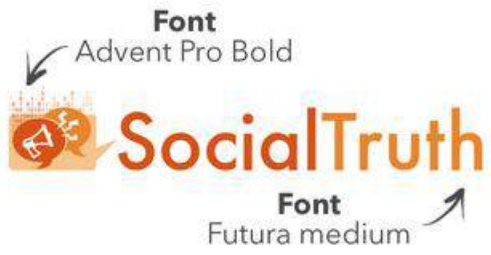
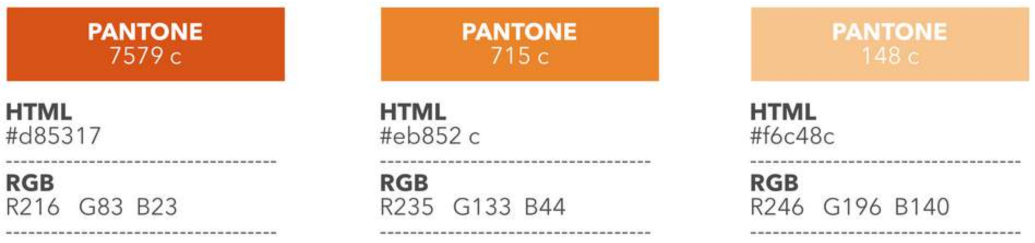


Figure 19 - Socialtruth Logo



PANTONE 7579 c	PANTONE 715 c	PANTONE 148 c
HTML #d85317	HTML #eb852 c	HTML #f6c48c
RGB R216 G83 B23	RGB R235 G133 B44	RGB R246 G196 B140

Figure 20 - Socialtruth colours

In order to fulfil the afore-mentioned aspects of **consistency and unity**, the interface layout should contain elements coherent with the logo style and colours, but with parsimony, since orange-like colours are warm colours known to invoke warmth and comfort, but may to the contrary lead to anger and hostility if used too much. Moreover, orange has been reported to be the most disliked colour by women (Hallock, 2003).

In any case, it is recommended that the hues of the logo are re-proposed and repeated in selected parts of the interfaces (example: see SocialTruth website icons).

For the use of colours, in general, it is recommended to always take into consideration the colour psychology applied to interface design, that is the logical and methodical approach to the use of colours based on the psychological properties of each hue and on the combination of specific shades, tones and tints that achieve a balance of good design and the desired psychological effect (A., 2009).

In any case, it is recommended that ADNkronos takes care of the management of colour- and brand-related issues for the interface, since they have major expertise in the field of graphics and already provided dissemination material for the project, including logo, brochures, flyers, website, social media profiles etc.

5.2 Interface recommendations for use case 1

For the use case 1 “Checking sources in the production process”, the application will be an independent application or a browser plugin. Based on updated pre-requirements described during plenary meeting in Bydgoszcz, there is no need to integrate Socialtruth’s functionalities in the editorial software already used by ADNKronos’ journalists, since the verification process happens before the journalists accesses it.

For this use case, it is **mandatory** that the whole interface is developed in the **italian language**, to assure the major usability issues related to language barriers and in accordance to the language used within the existing editorial software.

For the development of the HMI for this use case, the features reported in D2.1 Chapter 6 “Socio-technical & Human Aspects” must be taken into consideration and translated into interface design affordances, defined as actions that are made physically possible by the properties of an object or an environment within the interface.

The features reported in D2.1 for this use case are the following (Backholm, et al., 2017) (Brandtzaeg, Lüders, Spangenberg, Rath-Wiggins, & Følstad, 2016) (Schifferes, et al., 2014) (Schwartz, Naaman, & Teodoro, 2015) (Diakopoulos, De Choudhury, & Naaman, 2012):

Table 4 - Major functions of a web-based online content validation toolset for journalists

Major functions of a web-based online content validation toolset for journalists	
automatic features	• identify the source and its trustworthiness • find new trending content and developments within the already identified content areas
manual tasks	• users or teams should be able to modify the tool settings according to their needs, commonly mentioned settings include: ○ basic search through search engines ○ sorting functions (fresh, top, trending) ○ filter functions (geolocation, language of content, timeframe limitations, format – e.g. video only)
visualisation of results	• visualisation tools used as a bridge between the automated functions and the users, allowing to provide information focusing on areas of interest relevant to the current assignment, including: ○ summaries of how the content has been automatically compared and cross-referenced ○ visual chains of automatically identified steps between reposts and the original source ○ summaries of content listed according to the advanced search parameters, such as geolocated posts placed on a map
technical requirements	• tool functionality across screen sizes and equipment • automatic and frequent updates of content feeds to enable rapid inclusion of the latest information • easily accessible content saving function in a format compatible with existing publication formats
team-level	• the possibility to communicate within the toolset with colleagues, in order to avoid unnecessary repetitions of tasks already carried out by the other team members.

5.2.1 Content verification request input

The journalist should be able to give some input for the content verification request, such as an URL or text or image or video, and receive a feedback from the system regarding type of content, source, trustworthiness etc.

First of all, it is essential that the user is able to provide the input in the easiest possible way. Therefore, the interface should allow the user to copy&paste the URL of the content or website, but also to drag&drop it as presented in Figure 21:

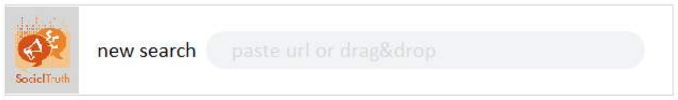
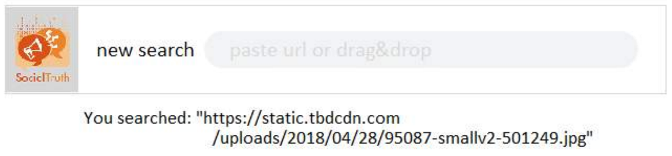


Figure 21 - Search bar (1)

Immediately after giving input to the system and pressing "Enter" or the search icon, he or she should receive a feedback from the system that the operation was successful (Figure 22):



You searched: "https://static.tbdcdn.com
/uploads/2018/04/28/95087-smallv2-501249.jpg"

Figure 22 - Search bar (2)

While the system collects data and information on the content, a pop up should show with a spinning wheel or a progress bar to keep the user in the loop during the loading phase.

It is essential that the interface prevents the user from having to specify what type of content he or she is giving as input to the system, meaning that the system has to automatically understand if the content is a text, an image, a video, a tweet, a website, an article and so on, and provide adequate and differentiated results for each type of content.

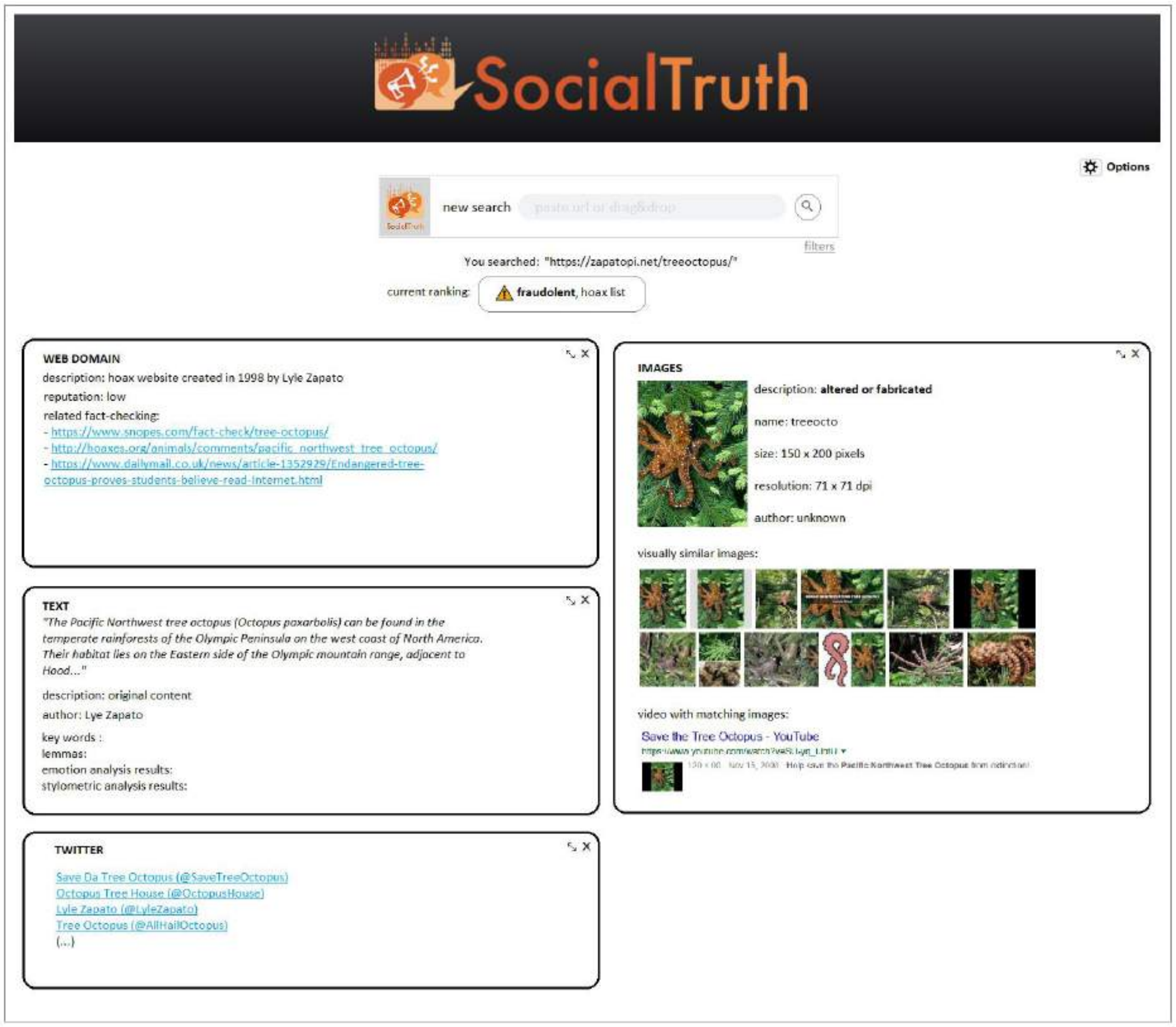
5.2.2 Content verification results

The system should allow the user to perform a content verification request on a certain piece of content. For example, if the user gives a news article as input to the system, the system should be able to provide information about the verification of the domain, text and image, including:

- the current ranking based on the verification algorithm;
- information about the URL domain (name, description, owner, location, reputation etc.)
- original author/ source of the text (name, date, location, etc.);
- results of the semantic and emotional analysis on the text (keywords, common lemmas, emotion, style etc.);
- information on the image and indication about whether it is genuine, fabricated, tampered or altered;
- original author/ source of the image (name, date, location etc.);

- a list of websites or social media posts where the news was shared after the original publication;
- optional: related images, visually similar images, videos with matching images.

Since the results to be displayed are many, the interface may for example be organized in blocks as presented in Figure 23.



You searched: "https://zapatopi.net/treeoctopus/"

current ranking: **fraudolent**, hoax list

WEB DOMAIN
description: hoax website created in 1998 by Lyle Zapato
reputation: low
related fact-checking:
- https://www.snopes.com/fact-check/tree-octopus/
- http://hoaxes.org/animals/comments/pacific_northwest_tree_octopus/
- https://www.dailymail.co.uk/news/article-1352929/Endangered-tree-octopus-proves-students-believe-read-Internet.html

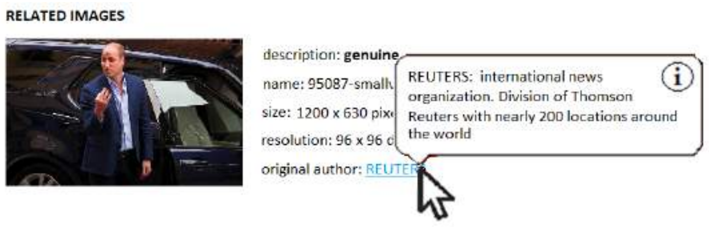
TEXT
"The Pacific Northwest tree octopus (Octopus paxarbolis) can be found in the temperate rainforests of the Olympic Peninsula on the west coast of North America. Their habitat lies on the Eastern side of the Olympic mountain range, adjacent to Hood..."
description: original content
author: Lye Zapato
key words :
lemmas:
emotion analysis results:
stylometric analysis results:

TWITTER
Save Da Tree Octopus (@SaveTreeOctopus)
Octopus Tree House (@OctopusHouse)
Lyle Zapato (@LyleZapato)
Tree Octopus (@AllHailOctopus)
(...)

IMAGES
description: **altered or fabricated**
name: treeocto
size: 150 x 200 pixels
resolution: 71 x 71 dpi
author: unknown
visually similar images:
video with matching images:
Save the Tree Octopus - YouTube
https://www.youtube.com/watch?v=[illegible]
[illegible] Nov 15, 2008 · Help save the Pacific Northwest Tree Octopus from extinction!

Figure 23 - Example of user interface for use case 1

Most of the provided information must be explorable, meaning that they must have a hyperlink that the user can click to obtain more info. Also, it would be useful to have a sort of a summary of what the hyperlink contains in terms of tooltips, in particular on names and sources:

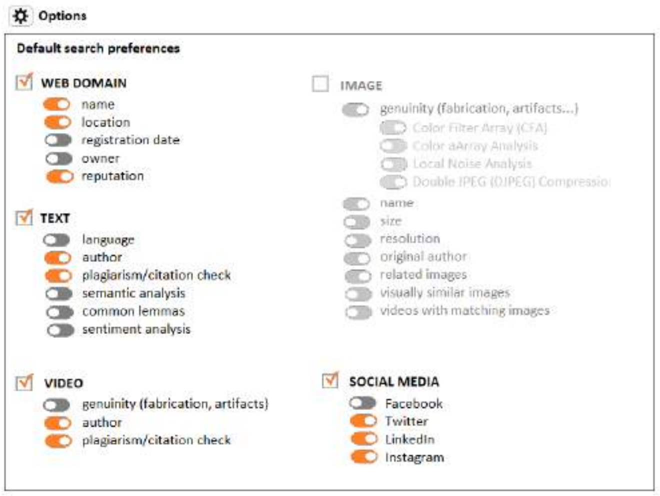

RELATED IMAGES

description: **genuine**
name: 95087-small
size: 1200 x 630 pix
resolution: 96 x 96 d
original author: REUTER

REUTERS: international news organization. Division of Thomson Reuters with nearly 200 locations around the world

Figure 24 - Tooltip visualisation

The default fields of search should be defined *a priori* by the system but should also be modifiable and customizable by the user, based on his or her preferences, habits and needs. This may be achieved by contemplating an “option” icon giving access to a control panel for the default search preferences (Figure 25).


Options

Default search preferences

☑ WEB DOMAIN
- ● name
- ● location
- ○ registration date
- ○ owner
- ● reputation

☐ IMAGE
- ○ genuinity (fabrication, artifacts...)
  - ○ Color Filter Array (CFA)
  - ○ Color aArray Analysis
  - ○ Local Noise Analysis
  - ○ Double JPEG (DJPEG) Compressio
- ○ name
- ○ size
- ○ resolution
- ○ original author
- ○ related images
- ○ visually similar images
- ○ videos with matching images

☑ TEXT
- ○ language
- ● author
- ● plagiarism/citation check
- ○ semantic analysis
- ○ common lemmas
- ○ sentiment analysis

☑ VIDEO
- ○ genuinity (fabrication, artifacts)
- ● author
- ● plagiarism/citation check

☑ SOCIAL MEDIA
- ○ Facebook
- ● Twitter
- ● LinkedIn
- ● Instagram

Figure 25 - Interface options

Also, the system should allow the user to arrange blocks to a different order, expand or reduce them, change the background, etc.

5.2.3 Search filters

The user should be able to select certain and specific areas of search. For example, if the user gives an image as input to the system, he or she should be able to require a search only from a specific geographic area or within a defined time window, thus applying filters to the search. Example is provided in Figure 26.

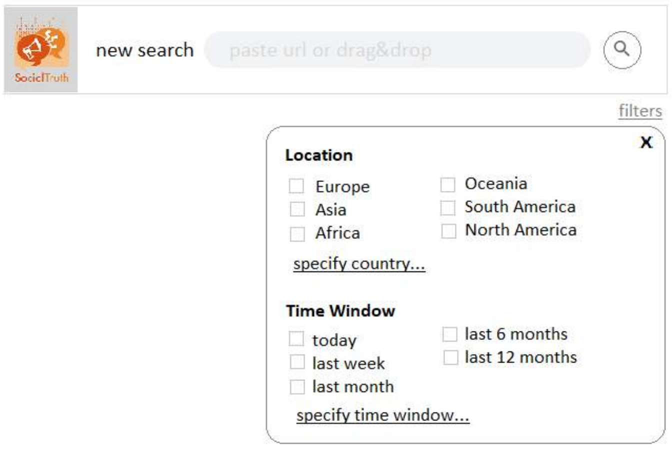


Figure 26 - Filters

This is particularly important when searching for social media posts and reposts.

5.2.3.1. Geographic provenance

The user should also be able to visualise the results of the search highlighting the geographic provenance of the online content publisher. For example, this may be useful when someone claims that is directly witnessing a natural disaster somewhere, but is actually publishing the news from a different position, or to the contrary when the journalist wants to read posts published by inhabitants of a place where a natural disaster was claimed to be happening.

5.2.4 Content verification history (blockchain)

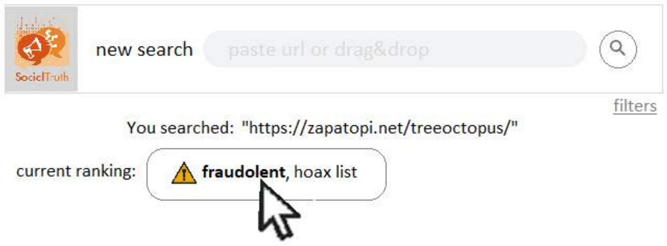


Figure 27 - Content verification history button

By clicking on the ranking button, the user should be able to visualize the verification history of the content (stored in the blockchain) as presented in Figure 27, or eventually to write in the blockchain an update on the trustworthiness of the content (as authorized user).

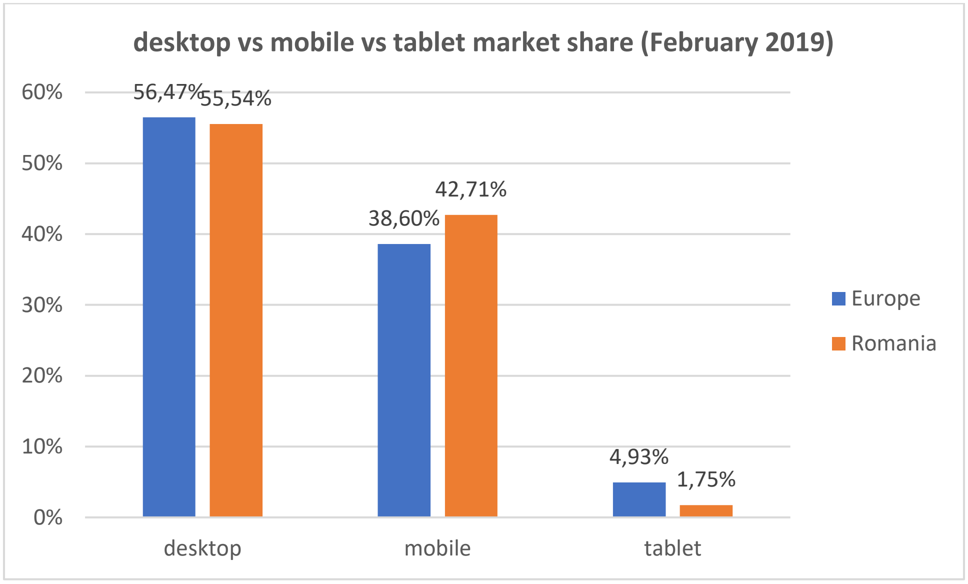
5.3 Interface recommendations for use case 2

For the use case no. 2 "Digital companion for content verification", the application will be a web browser plugin on the desktop side, and a smartphone/tablet application for on the mobile side.

For this use case, it is **mandatory** that the whole interface is developed in the **romanian language**, to assure the major usability issues related to language barriers.

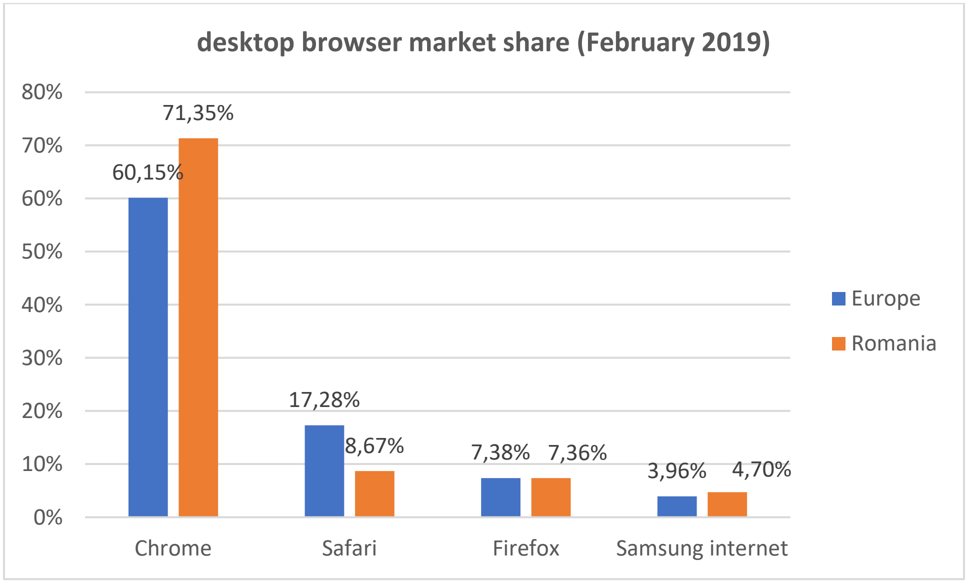
5.3.1 Target platforms

Based on the results of the statistical surveys conducted by StatCounter (StatCounter, s.d.) in the last year (from February 2018 to February 2019), the total desktop vs. mobile vs. tablet market share in Europe and Romania is presented in Graph 1.



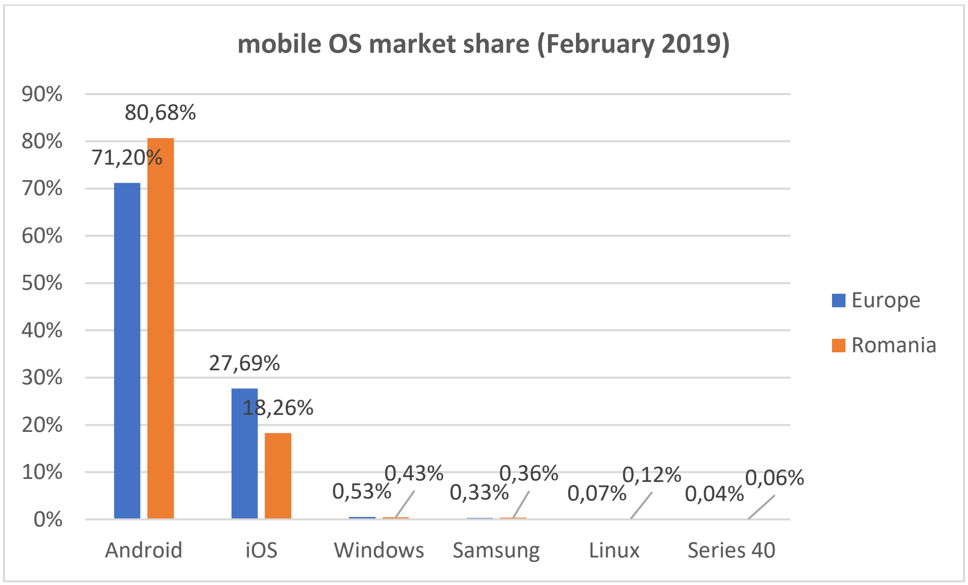
Graph 1 - Desktop vs. mobile vs. tablet market share (February 2019, Europe and Romania)

The browser market share in Europe and Romania is shown in Graph 2.



Graph 2 - Desktop browser market share (February 2019, Europe and Romania)

Graph 3 presents the mobile OS market share in Europe and Romania.



Graph 3 - Mobile OS market share (February 2019, Europe and Romania)

Based on these data and percentages, it is recommended that the Socialtruth desktop application is developed as a plugin or a search provider for **Chrome** and **Safari** at least, to cover 75-80% of the web browser market share.

For the mobile application, the main target platform should be **Android**, to cover 70-80% of the mobile OS market share.

5.3.2 Users attraction

The major issue about this use case is to assess why regular web users should want to install the SocialTruth Digital Companion.

Based on the findings reported in D2.1 Chapter 6 “Socio-technical & Human Aspects”, half of the European citizens say they come across fake news at least once a week and a large majority think that the existence of fake news is a problem in their country and for democracy in general.

One of the main concerns of people regarding news and media was reported to be the uncertainty over the truthfulness of the contents: 63% of the people world-wide agree that “the average person does not know how to tell good journalism from rumour or falsehoods” and 59% agree that “it is becoming harder to tell if a piece of news was produced by a respected media organisation”.

Given this encouraging data about the perception and awareness of people regarding content verification, it is mandatory to assess what is the main psychologic mechanism that may drive people to search for and install the SocialTruth application, and how to leverage that mechanism in order to foster the diffusion of SocialTruth.

As stated in D2.1, the Uses and Gratifications (U&G) theory appears to be useful in attempting to explain what social and psychological needs motivate audiences to select particular media channels and content choices, as well as the subsequent attitudinal and behavioural effects. From the U&G perspective, with regard to news reading, it is assumed that people actively choose among news sources owing to the sources ability to gratify their different needs. As reported by Lee *et al.*, the perceived gratifications of online news appear to be entertainment, information search, peer acceptance, relationship maintenance, socialising and self-status seeking.

Let us now deepen the meaning and consequences of the afore-mentioned data.

63% of the people world-wide agree that “the average person does not know how to tell good journalism from rumour or falsehoods”. When saying so, the respondents reasonably put themselves in the fraction of people who are better than average and can tell good journalism from rumour or falsehoods. Indeed, people tend to have a high consideration of themselves (Alicke, Vredenburg, Hiatt, & Govorun, 2001). Therefore, it is not recommended to leverage on the fact that they are not able to recognise hoaxes and false news and that they need an application for helping them.

SocialTruth should to the contrary enphasise the problem of false news spread, the consequence of such spread, and the importance of installing the SocialTruth application as part of the solution to the problem, thus leveraging people’s self-enhancement, that is the of motivation that works to make people feel good about themselves and to maintain self-esteem. They should feel that they are doing great, that they are important, that they are contributing to solve a severe problem affecting society and threatening democracy. Installing and using SocialTruth should induce in them the emotions of contentment and satisfaction.

Also, the main motivational affordances investigated in D2.1 Chapter 6 should be taken into consideration, namely:

The online environments allow users to keep up with friends, network with colleagues, and share their

- hypertextuality;
- multimediality;
- interactivity;
- portability;
- availability;
- locatability.

5.3.3 Content verification request input

The user should be able to give some input for the content verification request, such as an URL or text or image or video, and receive a feedback from the system regarding type of content, source, trustworthiness etc.

First of all, it is essential that the user is able to provide the input in the easiest possible way. Therefore, the interface should allow the user to copy&paste the **URL** of the content or website, but also to **drag&drop** as presented in Figure 28.

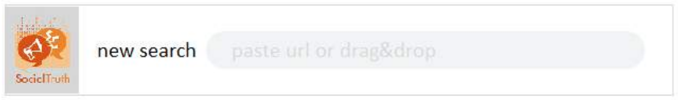


Figure 28 - Search bar (1)

Another possibility is to allow users to call the verification process by the **right click menu** on a link within the browser (see Figure 29).

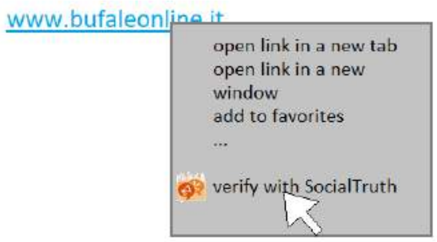
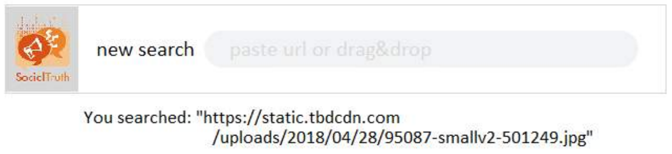


Figure 29 - Right-click menu

Immediately after giving input to the system and pressing “Enter” or the search icon, he or she should receive a feedback from the system that the operation was successful:



You searched: "https://static.tbdcdn.com /uploads/2018/04/28/95087-smallv2-501249.jpg"

Figure 30 - Search bar (2)

While the system collects data and information on the content, a pop up should show with a spinning wheel or a progress bar to keep the user in the loop during the loading phase.

It is essential that the interface prevents the user from having to specify what type of content he or she is giving as input to the system, meaning that the system has to automatically understand if the content is a text, an image, a video, a tweet, a website, an article and so on, and provide adequate and differentiated results for each type of content.

5.3.4 Content verification results

The system should allow the user to perform a content verification request on a certain piece of content. With respect to the use case 1, the current use case requires **more aggregated and summarized results**.

Moreover, the key-point for this use case is to provide new and credible information regarding de-bunking and fact-checking about the content. Giving the audience new and credible information is especially effective in thoroughly unseating misinformation, since new information allows people to update their understanding of events, justifying why they fell for the falsehood in the first place.

The search engine interface should be familiar and easy to use, looking just like a regular search engine. Figure 31 present an example.

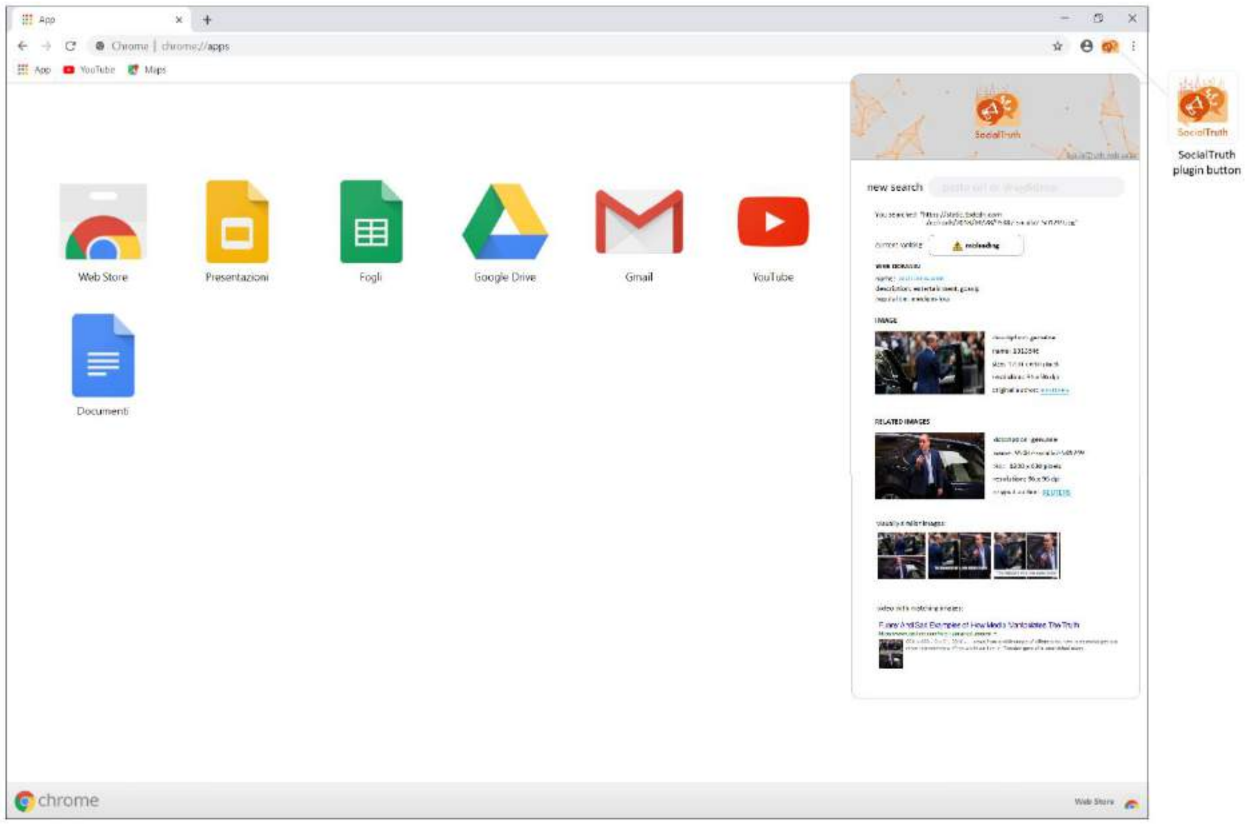


Figure 31 - Example of user interface for use case 2 (1)

If the user searches for some free text instead of an URL, the search results should be based on QWANT search engine + ranking as presented in Figure 32.

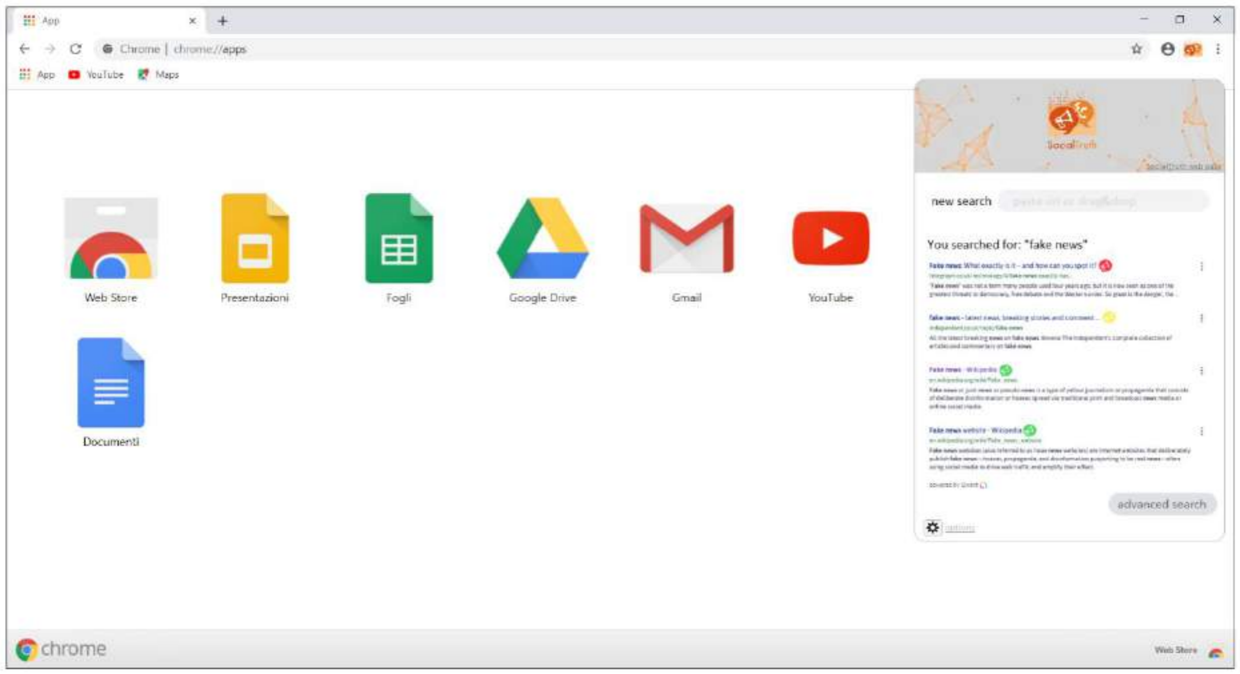


Figure 32 - Example of user interface for use case 2 (2)

5.3.5 Automatic alert

An optional functionaity may be investigated, allowing users to set automatic alerts to pop up if the encountering hoaxes online or visiting fraudulent websites.

5.3.6 Sharing

Based on the fact that people feel satisfied when they can share with friends, relatives and colleagues what they believe is important, the SocialTruth application should allow user to share findings through the social media (facebook, instagram, twitter) and also by e-mail.

The share could be in principle be based on the following: “Did you know this content is not trustable?” (or similar).

5.4 Interface recommendations for use case 3

For the use case 3 “Search engine rankings & advertising prevention for fraudulent sites”, the application will be embedded in the search engine result page, and the user interface in this case will mainly be an icon communicating the level of trustworthiness of the content shown.

For this use case, it is **mandatory** that the whole interface is developed in **the target geographic area(s) language**, to assure the major usability issues related to language barriers.

5.4.1 Colour-based rating

A colour-based rating method appears to be effective for this use-case.

Figure 33 shows the familiar RAG system is a popular method of rating for issues or status reports, based on **Red, Amber** (yellow), and **Green** colours used in a traffic light rating system (Veromann)/

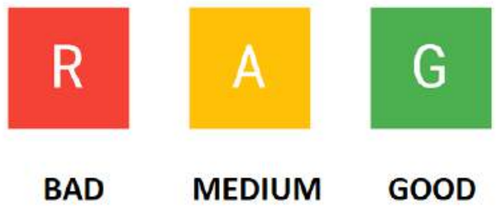


Figure 33 - RAG system

The choice of the RAG rating method is not the only one possible, other methods may be contemplated, such as the RAG+B (Veromann) presented in Figure 34.

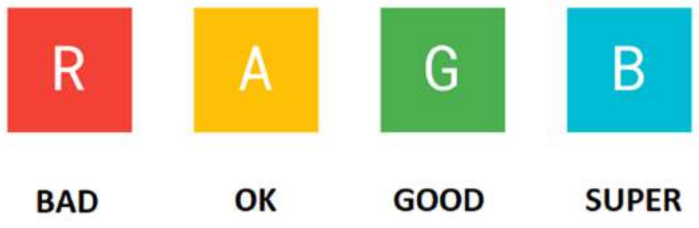


Figure 34 - RAG + B system

Here, the "super" category may contain ADNkronos, ANSA, REUTERS and other official/certified international press agencies.

Also other combinations presented in Figure 35 may be considered.

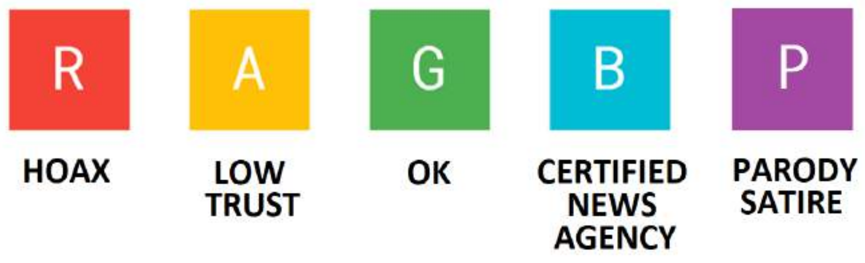
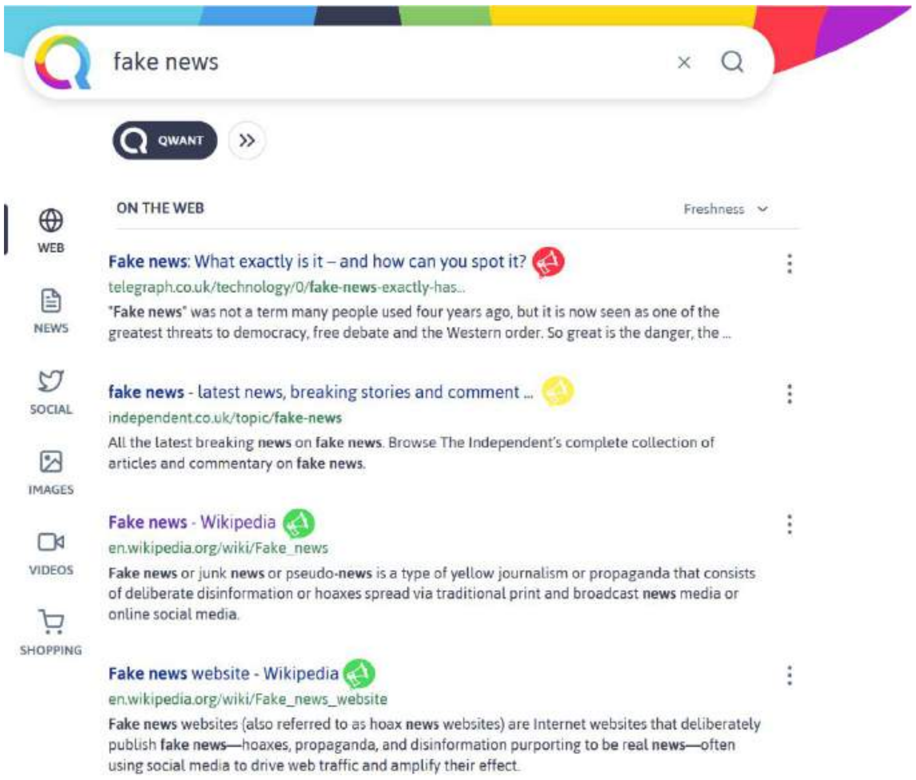


Figure 35 - RAG + B + extra system

In any case, once the type of colour-based rating method is assessed, it could be easily implemented within the existing QWANT search engine interface. A possible choice may be represented by the use of a coloured icon strictly connected to the SocialTruth application, for example a section of the logo such as the megaphone inside the balloon (Figure 36).



Fake news: What exactly is it – and how can you spot it?
telegraph.co.uk/technology/0/fake-news-exactly-has...
"Fake news" was not a term many people used four years ago, but it is now seen as one of the greatest threats to democracy, free debate and the Western order. So great is the danger, the ...

fake news - latest news, breaking stories and comment ...
independent.co.uk/topic/fake-news
All the latest breaking news on fake news. Browse The Independent's complete collection of articles and commentary on fake news.

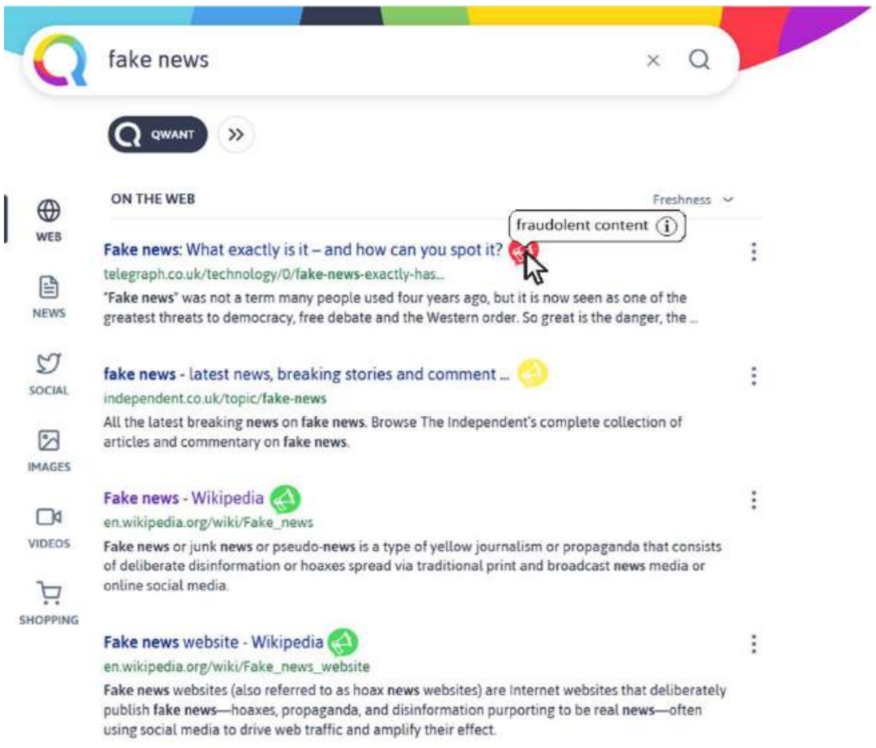
Fake news - Wikipedia
en.wikipedia.org/wiki/Fake_news
Fake news or junk news or pseudo-news is a type of yellow journalism or propaganda that consists of deliberate disinformation or hoaxes spread via traditional print and broadcast news media or online social media.

Fake news website - Wikipedia
en.wikipedia.org/wiki/Fake_news_website
Fake news websites (also referred to as hoax news websites) are Internet websites that deliberately publish fake news—hoaxes, propaganda, and disinformation purporting to be real news—often using social media to drive web traffic and amplify their effect.

Figure 36 - Example of user interface for use case 3

This sort of solution may help making the SocialTruth's brand (logo) easily recognisable, also in other contexts, thus contributing to brand cultivation.

Based on the requirements defined in D2.1 and during the plenary meeting in Bydgoszcz, for this use case QWANT does not need to provide additional information to its users regarding how the ranking was computed. Nonetheless, when passing the mouse on the icon, a brief tooltip should appear stating what the icon's colour means, with an info icon containing a hyperlink, so that if the user wants to know more, he or she can directly follow that link. Indeed, this functionality is mainly psychology-driven.



Fake news: What exactly is it – and how can you spot it?
telegraph.co.uk/technology/0/fake-news-exactly-has...
"Fake news" was not a term many people used four years ago, but it is now seen as one of the greatest threats to democracy, free debate and the Western order. So great is the danger, the ...

fake news - latest news, breaking stories and comment ...
independent.co.uk/topic/fake-news
All the latest breaking news on fake news. Browse The Independent's complete collection of articles and commentary on fake news.

Fake news - Wikipedia
en.wikipedia.org/wiki/Fake_news
Fake news or junk news or pseudo-news is a type of yellow journalism or propaganda that consists of deliberate disinformation or hoaxes spread via traditional print and broadcast news media or online social media.

Fake news website - Wikipedia
en.wikipedia.org/wiki/Fake_news_website
Fake news websites (also referred to as hoax news websites) are Internet websites that deliberately publish fake news—hoaxes, propaganda, and disinformation purporting to be real news—often using social media to drive web traffic and amplify their effect.

Figure 37 - Example of user interface for use case 3 with tooltip

The hyperlink should redirect the user to a web page containing an explanation of the rating result. If the rating is low, the page should also provide information regarding the de-bunking and the fact-checking: giving the audience some new and credible information is especially effective in thoroughly unseating misinformation, since new information allows people to update their understanding of events, justifying why they fell for the falsehood in the first place.

Also, by following further hyperlinks the user should be allowed to obtain information about the nature and purposes of the SocialTruth project and instructions for installing the SocialTruth browser plugin and/or mobile application developed for use case 2.

5.5 Interface recommendations for use case 4

For the use case 4 "External sources reliability check in the educational domain", the application will be a sort of add-on to the existing lesson plan maker platform "CreaLezioni". Therefore, the look&feel of the application will be mainly driven by the appearance of the existing software, in terms of buttons, tabs, etc.

For this use case, it is **mandatory** that the whole interface is developed in the **italian language**, to assure the major usability issues related to language barriers and in accordance to the language used within the existing teaching-support software.

Based on the requirements defined in D2.1 and during the plenary meeting in Bydgoszcz, the Human-Machine Interface for this use case will allow teachers to differentiate online contents based on its rank of trustworthiness, similarly to use case 3 by QWANT, but in addition to this DeAgostini Scuola feels it is necessary to provide users with an exhaustive, although **aggregated/summarised**, reasoning for the computed ranking. This functionality would be especially needed when the SocialTruth tool is used for pupils' education about recognising trustworthy and not trustable online content.

The main targets for the content verification service are reported to be web pages and YouTube videos.

Regarding the videos, a good approach may be to provide the users with the results of the tampering analysis, stating if the video is genuine, fabricated, altered etc. taking into consideration also the audio part, and highlighting cases of re-dubbing or similar.

If possible, the user should be allowed to visualise the ranking reasoning in terms of a "story" or timeline, preferably in the form of an **infographic**.

6 Conclusions

This document is the first release of the SocialTruth architecture design prepared in the early stage of the project. It will be updated later in Month 15, including ADL description of the architecture and such aspects as data adaptation models, big data management, annotation semantics, etc.

This report will help particular work packages and technology/components creators to follow the architectural principles and guidelines as well as common understanding of the SocialTruth ecosystem. As the input, we have used the requirements coming from D2.1 as well as the best practices and knowledge of current technologies and architecture design principles.

This deliverable provides the analysis behind several choices that have to be made by the consortium. It discusses the possible approaches to architecture and integration, such as options for 'heavy service bus (ESB)' and microservices. The document also presents the possible cloud solutions that can be used for SocialTruth deployment in a real environment. The consortium tends towards the use of microservices and the private cloud deployment model.

The report also displays the modules and components of the platform, lists the needed functionalities, discusses the interoperability aspects, as well as the security, privacy, social and human aspects.

7 References

A., W. (2009). *The Colour Affects System of Colour Psychology.* AIC Quadrennial Congress.

Alicke, M. D., Vredenburg, D. S., Hiatt, M., & Govorun, O. (2001). The "better than myself effect". *Motivation and Emotion, 25*, 7-22.

Backholm, K., Ausserhofer, J., Frey, E., Grondhal Larsen, A., Hornmoen, H., Hogvag, J., & Reimerth, G. (2017). Crises, Rumours and Reposts: Journalists' Social Media COntent Gathering and Verification Practices in Breacking News Situations. *Media and COmmunication, 5*(2), 67-76.

Brandtzaeg, P. B., Lüders, M., Spangenberg, J., Rath-Wiggins, L., & Følstad, A. (2016). Emerging Journalistic Verification Practices Concerning Social Media. *Journalism Practice, 10*(3), 323-342.

Demangeot, C., & Broderick, A. J. (2007). Conceptualising consumer behaviour in online shopping environments. *International Journal of Retail & Distribution Management, 35*(11), 878-894.

Diakopoulos, N., De Choudhury, M., & Naaman, M. (2012). Finding and assessing soccial media information sources in the context of journalism. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* Austin, TX.

Djamasbi, S., Siegel, M., & Tullis, T. (2011). Visual Hierarchy and Viewing Behavior: An Eye Tracking Study. Human-Computer Interaction. *Design and Development Approaches in Computer Science*, 331-340.

Elder, R. S., & Krishna, A. (2012). The "Visual Depiction Effect" in Advertising: Facilitating Embodied Mental Simulation through Product Orientation. *Journal of Consumer Research, 38*(6), 988-1003.

Fadeyev, D. (n.d.). *The Usability Post - Thoughts on design and user experience*. Retrieved from http://usabilitypost.com

Gorini M., C. V. (2013). *EMERALD deliverable 'D2.3 - EMERALD System Functional Architecture'.*

Hallock, J. (2003). *Colour Assignment - Preferences and Associations.*

Imtiaz, S. (2016). *The Psychology Behind Web Design.* McMaster University.

Lindgaard, G., Fernandes, G., Dudek, C., & Brown, J. (2006). Attention web designers: You have 50 milliseconds to make a good first impression! *Behaviour & Information Technology, 25*(2), 115-126.

Schifferes, S., Newman, N., Thurman, N., Corney, D., Goker, A., & Martin, C. (2014). Identifying and verifying news through social media: developing a user-centered tool for professional journalists. *Digital Journalism, 2*(3), 406-418.

Schwartz, R., Naaman, M., & Teodoro, R. (2015). Editorial algorithms: using social media to discover and report local news. *Ninth International AAAI Conference on Web and Social Media.* Oxford, UK.

Sillence, E., Briggs, P., Fishwick, L., & Harris, P. (2004). Trust and mistrust of online health sites. *Proceedings of the 2004 Conference on Human Factors in Computing Systems*, 663-670.

StatCounter. (n.d.). Retrieved from http://gs.statcounter.com/

The Open Group. (n.d.). *ArchiMate®, 2.1 specification*. Retrieved December 2013, from The Open Group: http://pubs.opengroup.org/architecture/archimate2-doc/

Tuch, A. N., Presslaber, E. E., Stöcklin, M., Opwis, K., & Bargas-Avila, J. A. (2012). The role of visual complexity and prototypicality regarding first impression of websites: Working towards understanding aesthetic judgments. *International Journal of Human-Computer Studies, 70*(11), 794-811.

Verhagen, T., Boter, J., & Adelaar, T. (2010). The Effect of Product Type on Consumer Preferences for Website Content Elements: An Empirical Study. *Journal of Computer-Mediated Communication, 16*(1), 139-170.

Veromann, V.-J. (n.d.). *RAG+B traffic light rating system – expanding established design patterns.* https://weekdone.com.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering* (pp. 195-216). Springer, Cham.

Dragoni, N., Lanese, I., Larsen, S. T., Mazzara, M., Mustafin, R., & Safina, L. (2017, June). Microservices: How to make your application scale. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics* (pp. 95-104). Springer, Cham.

https://medium.com/tech-tajawal/microservice-authentication-and-authorization-solutions-e0e5e74b248a (accessed May 23, 2018).