Structured Approaches for Forest fire Emergencies in Resilient Societies

SAFERS Platform Core Components Report

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 869353.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS LIST

AOI	Area of Interest
API	Application Programming Interface
CIAM	Customers Identity and Authentication Management
CIG	Crowdsourcing Intelligent Gateway
CRUD	Create Read Update Delete
dNBR	Differential Normalized Burned Ratio
EO	Earth Observation
ETL	Extract, Transform and Load
FOV	Field Of View
GIS	Geographic Information System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoU	Intersection over Union
IR	Infrared
JWT	JSON Web Token
ML	Machine Learning
NBR	Normalized Burned Ratio
OGC	Open Geospatial Consortium
PTZ	Pan Tilt Zoom
REST	Representational State Transfer
RMSE	Root Mean Squared Error
RTSP	Real Time Streaming Protocol
SIS	SAFERS Intelligent Services
SWIR	Short-wave infrared
TR	Technical Requirements
UAV	Unmanned Aerial Vehicle
UR	User Requirements

1. INTRODUCTION

1.1. SAFERS project overview

Forest fires are exacerbated by extreme weather conditions, which are increasing both in frequency and in magnitude due to climate change. Globally, massive fires have swept through forests and other landscapes in an alarming rate, resulting in the loss of human lives, destruction of homes and biodiversity, and emitting millions of tons of CO_2 and other pollutants in addition to various destructive impacts. Therefore, the need of effective management of forest fire emergencies has become very crucial.

To respond to this global challenge, the project SAFERS 'Structured Approaches for Forest fire Emergencies in Resilient Societies' is in a mission to support societies becoming more resilient across the key phases of the forests fires emergency management cycle. SAFERS is an Innovation Action project funded by the EU under H2020 programme (topic: SC5-16-2019 - Development of commercial activities and services through the use of GEOSS and Copernicus data) for a period of 3 years (2020-2023) and with a budget of 3.25 million euros. It is coordinated by LINKS foundation and brings together 14 partners coming from 7 European countries: Italy, Greece, Finland, Germany, United Kingdom, France and Spain.

SAFERS will realise a comprehensive Emergency Management System (EMS) which will act along the key phases of the emergency management cycle: a big data information system integrating Earth Observation data and services offered by the EU Copernicus programme and GEOSS, crowdsourced data from social media and from other applications used by citizens as well as emergency staff and real-time data provided by accurate sensors to detect smoke or fires. Advanced algorithm based on Artificial Intelligence will be used to generate risk maps and early warnings in the preparedness phase, it will also be used to estimate the forest fire extension and its propagation in function of the forecasted weather and soil conditions in the response phase, compute the impacts of an extinguished fire in terms of economic losses and monitor the soil recovery in the post-event phase. The project will gather all these data and services in one open platform designed for forest fire management and demonstrated in 4 pilots in Italy, France, Spain and Greece to validate SAFERS services.

1.1. General architecture

The architecture of ERMES Platform is showed in Figure 1. The architecture is designed as an open-source platform employing a microservice architecture pattern. This system integrates various components to deliver intelligent services for emergency management. The architecture ensures seamless interaction between different services and the efficient handling of data through a combination of frontend and backend components, identity management, and big data repositories.

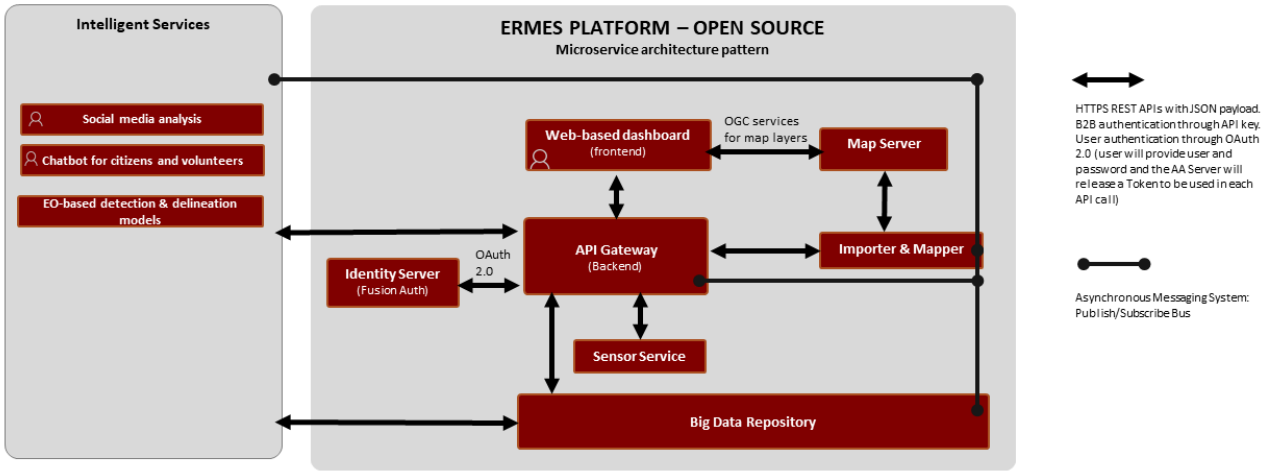


Figure 1 ERMES architecture

1.2. Structure of the document

The document is structured/organised as it follows:

- **Chapter 1** Introduction and description of the document itself
- **Chapter 2** API gateway backend module
- **Chapter 3** Identity Server
- **Chapter 4** Web-Based Dashboard for operational management and decision support
- **Chapter 5** Geodata Repository management
- **Chapter 6** Importer and Mapper
- **Chapter 7** Conclusions
- **Chapter 8** References

2. API GATEWAY BACKEND MODULE

This module represents the central-star element and it is interfaced with all the other components of this part of the SAFERS solution. It is closely linked to the CIAM module, as it must verify the validity and the clearance of the JWT token associated with every HTTP request. While the CIAM component takes care of managing the tasks related to user registry and user roles, the backend defines and manages the organization registry and the user–organization relationship. This information, together with all the other entities of the project, are stored in a PostgreSQL database. Data storage is complemented by the Azure Blob Storage component, on which multimedia content, such as images, videos and audio files, are saved. Once collected and processed, this information must be made available to the all the clients, including the Data Resilience Dashboard, which will take care of the visualization.

All the interactions occur through consumption of authenticated REST services, that allow to handle the lifecycle of reports, missions, communications and actions.

2.1. Architecture

The Crowdsourcing API Gateway is a web application that must manage and store the entities created or updated by the Chatbot. These entities will be made available for other external clients, like the Data Resilience Dashboard. Figure 2 shows the general architecture of the module.

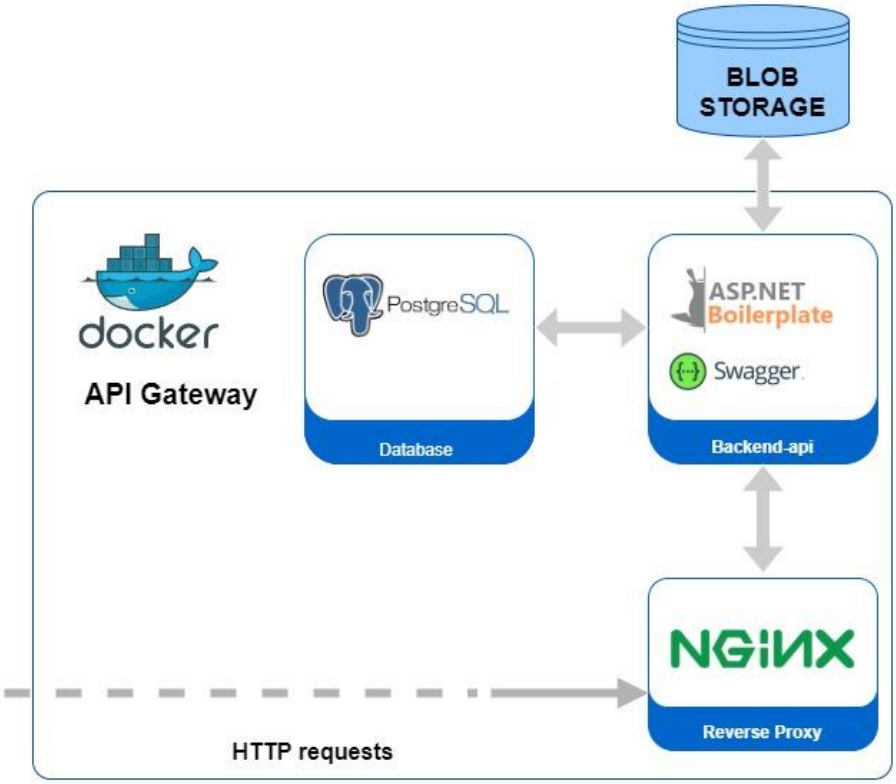


Figure 2: Crowdsourcing API Gateway architecture

This component is based on ASP.NET Boilerplate [1], that is an open source and general-purpose application framework that provides a layered architectural model, based on Domain Driven Design. This framework allows to speed up the starting phase by providing a complete set of built-in features like:

- **Logging**: possibility to write logs using the Logger object defined in the base class.
- **Unit of Work**: In ABP, each application service method is by default unit of work. It automatically creates a connection and begins a transaction at the beginning of the method. If the method successfully completes without an exception, then the transaction is committed, and the connection is disposed. All changes on entities are automatically saved when a transaction is committed.
- **Dependency Injection**: ABP uses and provides a conventional DI infrastructure.

- **Validation**: ABP automatically checks if the input is null. It also validates all the properties of an input based on standard data annotation attributes and custom validation rules. If a request is not valid, it throws a proper validation exception.
- **Auto Mapping**: AutoMapper library maps properties from one object to another based on naming conventions.

The layering of an application's codebase is a widely accepted technique to help reduce complexity and to improve code reusability. To achieve a layered architecture, ASP.NET Boilerplate follows the principles of **Domain Driven Design** [2]. There are four fundamental layers in Domain Driven Design (DDD):

- **Infrastructure Layer**: Provides generic technical capabilities that support higher layers mostly using 3rd-party libraries.
- **Domain Layer**: Includes business objects and their rules. This is the heart of the application.
- **Application Layer**: Mediates between the Presentation and Domain Layers. Orchestrates business objects to perform specific application tasks.
- **Presentation Layer**: Provides an interface to the user. Uses the Application Layer to achieve user interactions.

In addition to DDD, there are also other logical and physical layers in a modern architected application.

Figure 3 shows how the model below is suggested and implemented for ASP.NET Boilerplate applications.

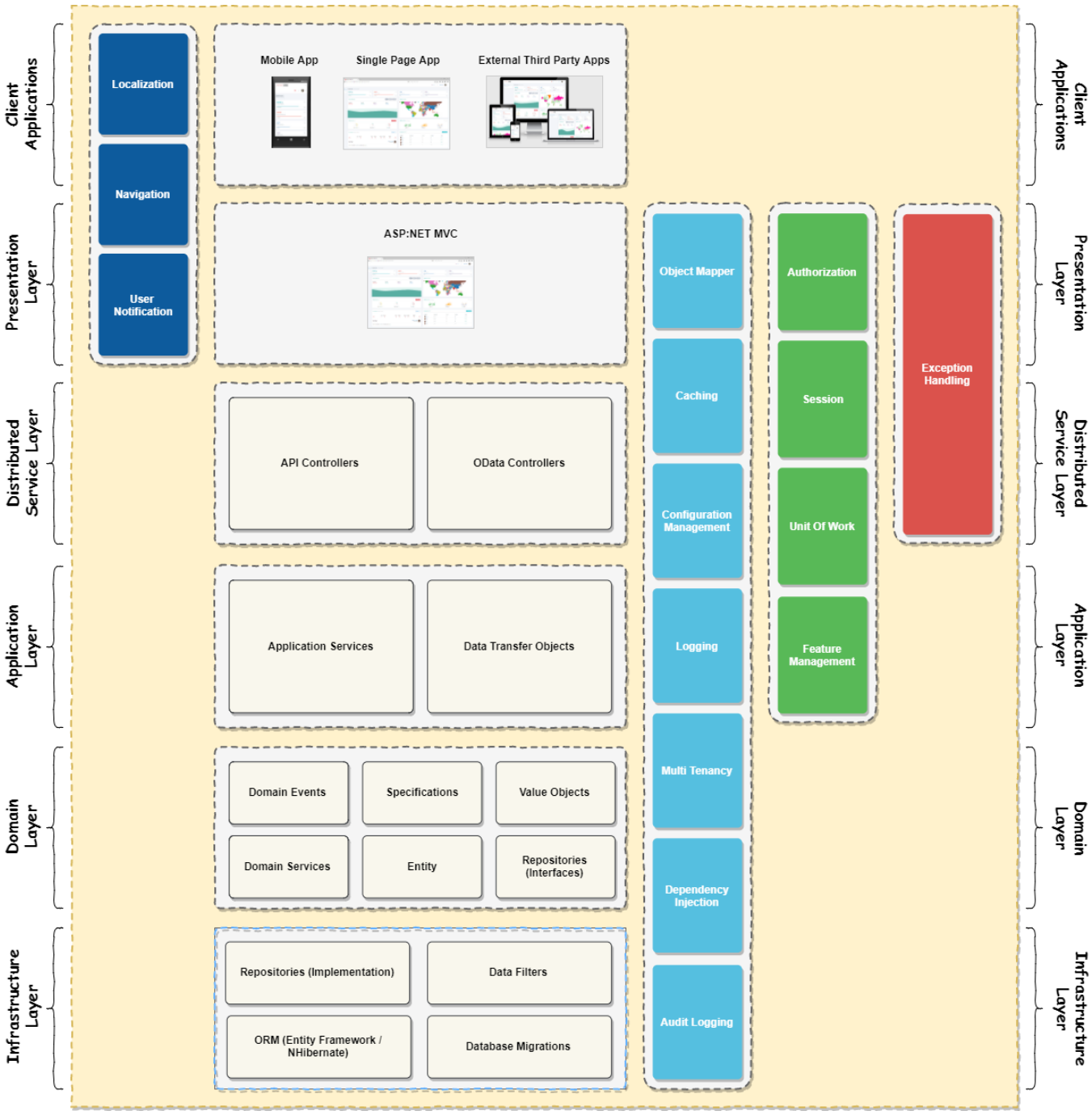


Figure 3: DDD model implemented in ABP application

- **Client Application**: these are remote clients that use the application as a service via HTTP APIs. For this specific project, an example of remote client is the chatbot.

- **Presentation Layer**: ASP.NET Core [3] MVC (Model-View-Controller) can be the presentation layer. It can be a physical layer (uses application via HTTP APIs) or a logical layer (directly injects and uses application services)
- **Distributed Service Layer**: this layer is used to serve application/domain functionality via remote APIs like REST. They do not contain business logic but only translate HTTP requests to domain interactions or can use application services to delegate the operation.
- **Application Layer**: the application layer mainly includes *Application Services* that use domain layer and domain objects to perform requested application functionalities. It uses *Data Transfer Objects* to get data from and return data to the presentation or distributed service layer.
- **Domain Layer**: this is the main layer that implements our domain logic. It includes *Entities, Values, Objects* and *Domain Services* to perform business/domain logic. It can also include *Specifications* and trigger *Domain Events.* It defines Repository Interfaces to read and persist entities from the data source.
- **Infrastructure Layer**: the infrastructure layer makes other layers work: It implements the repository interfaces (using *Entity Framework Core*) to work with a real database. This is not a strict layer below all layers but supports other layers by implementing the abstract concepts of them.

Entity Framework Core [4] is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology. EF Core can serve as an object-relational mapper (O/RM), enabling .NET developers to work with a database using .NET objects, and eliminating the need for most of the data-access code they usually need to write.

3. IDENTITY SERVER AND *MESSAGE* BUS

In modern cloud-based software architecture it has become more and more common to include a separate component for Customers Identity and Authentication Management (CIAM). This category of software includes both the features of an Authentication and Authorization (AA) Server as well as user profile management features [5].

Supporting several well-known standards such as OAuth 2.0 [6], OIDC [7] and Security Assertion Markup Language (SAML), a CIAM makes it easier to integrate any component of the architecture or even external clients. Nowadays, there are several CIAM services available, which can be either consumed as SaaS or deployed on-premises.

With a CIAM, all the user registration, identification and authentication procedures are decoupled from the services that consume it, saving a lot of development effort and gaining more efficiency and robustness. Furthermore, CIAMs usually implement state-of-the-art authentication protocols such as OAuth 2.0, with several authentication flows that can be adapted to several categories of clients, from static or dynamic web pages to IoT or wearable devices, enabling safe integration of diverse components. Most of the time, a CIAM is also chosen because it makes it straightforward to implement Single sign-on (SSO), as well as a federation with one or more external Identity Providers.

3.1. CIAM Module

The most important software requirement for the CIAM component to comply with is OAuth 2.0 support. Another relevant criterion for choosing a CIAM is that, having a separate user database from the main application server database helps anonymize user data and makes the whole system more robust for sensitive data protection.

Last, but not least, adopting a CIAM is also an advantage because the development team can focus on the design and implementation of the core value proposition without having to completely implement and handle the task of user management, which is also more robustly and flexibly implemented according to popular standards by a well-tested software.

In addition, the module provides security, scalability, reliability, low usage costs, customization, and ease of deployment and configuration. Table 1 summarizes the comparison made among the following four CIAMs, some of which are SaaS, and some are not, thus also allowing on-premises deployment. The deployment with OS-level virtualization (Docker or Kubernetes) allows to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files facilitating the updating and the managing of it.

Table 1 Comparison of different CIAM Solutions

Feature	FusionAuth	KeyCloak	Azure AD	Auth0
Homepage	https://fusionauth.io/	https://www.keycloak.org/	https://azure.microsoft.com/en-gb/services/active-directory/	https://auth0.com/
On-Premises	YES (Docker and Kubernetes support)	YES (Docker and Kubernetes support)	NO (SaaS)	NO (SaaS)
Unlimited users	YES	YES	NO	NO
Licence/Fee costs	Free or Commercial. No costs for free version	Open Source - No costs	Depend on volume	Depend on volume
OAuth 2.0 Protocol	YES	YES	YES	YES
OIDC	YES	YES	YES	YES
Themes/Templates	YES	YES	Limited	Limited

Support	Community or Commercial	Community	Commercial	Commercial
Official SDKs	YES	NO	YES	YES

A key feature for the selection was cost reduction, therefore the choice has been soon restricted to FusionAuth and KeyCloak, although often SaaS offers easier configuration. For FusionAuth and KeyCloak, the number of relevant features for the purposes of SAFERS is basically equivalent and the two software are interchangeable.

In SAFERS, it was chosen to adopt a Docker containers-based solution for the CIAM deployment, since it can scale well enough. Kubernetes was also considered, but given its higher dev-ops costs, it was left apart.

The CIAM also offers several REST API to control basically everything. Being multi-tenant by design, it allows to easily separate user bases and to provide per-tenant customization for the web pages, like, for instance, the login page, and email templates. Themes can be uploaded on the CIAM administration dashboard and applied to all the webpages used by the clients belonging to a tenant, and this helps provide a smoother User Experience (UX).

SAFERS clients use the following OAuth 2.0 flows supported by FusionAuth and KeyCloak: Authorization Grant Flow with JSON Web Token (JWT) and Refresh Token for session renewal. The CIAM supports OpenID and SAML for SSO among different identity provider (IdP), including popular social media such as Google, Twitter, and Facebook.

The selected CIAM can propagate events to authorized services, such as user creation, editing, or deletion. There are two main mechanisms: configurable webhooks or Kafka integration. Server authorization relies on API Keys. Event propagation is useful for keeping the data synced.

Another important CIAM feature is a better General Data Protection Regulation (GDPR) support [8], which follows the rules of GDPR which applies Europe-wide. A standalone CIAM server can isolate all the user data in a single database, separated by the one used in the API Gateway and other SAFERS modules, provide API to pseudonymize the data in the database, and supports the deletion of all the user data and activity. This has the benefit of making it easier for the whole module to be compliant with the GDPR and respect user privacy under the SAFERS terms of use, since there is no need to store personally identifying information outside the CIAM.

3.1.1. Authentication

Authentication flows are needed by each component in order to retrieve an access token from the CIAM, that is a string in JWT format that grants authorization and identification when consuming the REST API, and sometimes a refresh token, which is another string that allows obtaining a new access token upon expiration, without asking the user to perform again the authentication procedure.

All components will validate the JWT using a CIAM API, or locally using the signatures exposed by the CIAM itself, then handle user authorization and role-based access. If the access token is not valid, the user authentication will be rejected. The OAuth 2.0 standard authentication workflow adopted in SAFERS is illustrated in the following subsection.

3.1.2. Authorization Code Grant flow

The Authorization Code Grant Flow, described in RFC 6749 [9] is used for robust client authentication and authorization. This is the mechanism proposed for client SAFERS tools to use in SAFERS Authentication, and it is perhaps the most well-known OAuth 2.0 authentication workflow. This workflow is well supported by the selected CIAM.

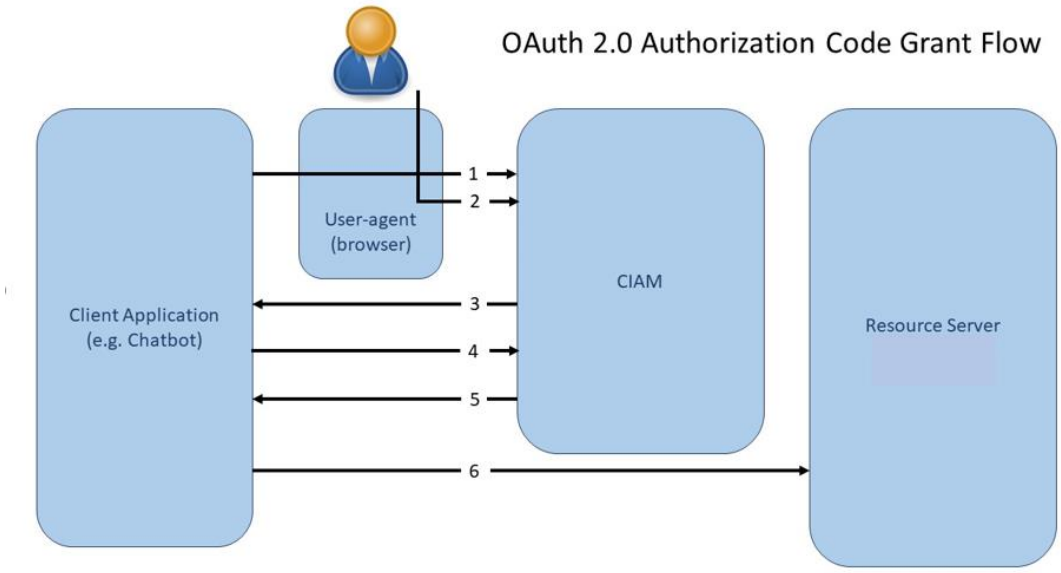


Figure 4 OAuth 2.0 Authorization Code Grant Flow

Figure 4 shows a summary diagram of the OAuth 2.0 Authorization Code Grant Flow as it is used, for instance, in the API Gateway, although this scheme is general: in step 1 of the figure, the Client Application (e.g., a server such as the Chatbot logic) requests the user to grant authorization, redirecting them to the CIAM authorization endpoint. Then, in step 2, the user grants the authorization by entering their credentials through a user-agent, usually a web browser or a web view in an application. At step 3, the CIAM redirects the user to a configured endpoint on the client application where it sends the authorization code, a 1-time only code that identifies this transaction. The client application sends the authorization code to the CIAM in step 4, using a preconfigured credentials set (client ID and client secret) provided by the CIAM itself and stored in the Client Application configuration. Upon validation, in step 5 the CIAM sends the authorization token (JWT) to the Client Application, which is used to consume the API provided by the Resource Server. The Resource Server shall validate the authorization token at each request, either

calling a CIAM endpoint, or locally using a JSON Web Key Set (JWKS) statically provided by the CIAM. Notice how the only component that has direct access to the user data and credentials is the CIAM.

If the refresh token is enabled, this is returned at step 5 together with the authorization token. Then, when the JWT expires, the Client Application can use the refresh token to obtain a new authorization token. The refresh token is usually long lived, and it is never sent to the Resource Server. Conversely, the JWT has usually a shorter lifetime compared to the Refresh Token. All these settings can be configured on the CIAM administration panel per each configured Client Application and tenant.

3.2. SAFERS Message Bus

The SAFERS architecture consists of multiple cloud-based services which must communicate with one another. This communication can happen in multiple ways: For quick synchronous calls, where the time factor is negligible, those services with REST API access can simply expose a set of HTTP endpoints that can be queried by other services, provided valid authentication tokens are included in the request. For services with long-running processes that cannot be done synchronously and/or with periodic tasks and ad hoc requests an asynchronous broker-based network infrastructure, or message bus, is used. This negates the need for services to explicitly query anything; once a message is produced it will be consumed by the appropriate services.

Nowadays, broker-based network infrastructures have drastically increased in popularity, thanks to distributed and modular architecture based on microservices, such as those used by SAFERS. Despite the added complexity of multiple components to handle, such systems have countless advantages, such as the possibility to spread computational resources on different machines and avoid a single point of failure. Among the most popular brokers, we investigated Kafka [10] and RabbitMQ [11]. While the former is certainly the most popular for scalable and real-time applications, the latter represents one of the most robust and versatile brokers available at this date, offering a mature platform available for practically any diffused programming language. In the following sections, we first provide a thorough comparison between the two aforementioned frameworks, highlighting the crucial points that led to the choice of RabbitMQ. Subsequently, the inner mechanisms of this tool are briefly presented and described, with a focus on the features employed in this project. Last, our designed solution is described, providing details on the framework configuration and choices made.

3.3. Related tools

As briefly mentioned before, we mainly investigated two tools, namely Apache Kafka and RabbitMQ. Both solutions share many interesting features: they comprise a broker-based messaging system using multiple queues, they allow different connection methods, such as direct access to queues or topic subscriptions, they can both handle message durability on disk and provide high availability setups with distributed deployments and shared queues. Lastly, despite the misconceptions, they can both handle thousands of messages every second.

Kafka is an open-source software that provides a framework focused on storing, reading and analyzing streaming data. Given it belongs to the group of Free and Open Source Software (FOSS), Kafka has a wide network of users and developers contributing daily towards updates and new features, offering long-term support for both seasoned and new users. It is also designed to be distributed by nature, meaning that rather than starting from a lightweight single-machine deployment, it inherently starts as a distributed system, potentially running across several servers and thus exploiting their storage capacity and processing power. In terms of messaging, Kafka implements its own protocol, which focuses on the optimization of huge amounts of data with extremely low latency. The most common communication protocol is based on Kafka topics, meaning the exchange of messages with particular routing keys, that are forwarded to a specific queue following predefined routing policies.

RabbitMQ is also part of the FOSS ecosystem, boasting several years of updates for thousands of new and experienced contributors. For this reason, while being an older alternative to the former framework, it is still one of the most popular message brokers, adopted by small and large enterprises alike. While being a bit less user-friendly from a configuration point of view, RabbitMQ remains extremely lightweight and easy to deploy both as a single on-premise instance or in the cloud. Its versatility also allows for more complex use cases such as high-availability, high-scale or federated deployments. From the message exchange standpoint, RabbitMQ supports many standard protocols, both natively and through plugin support, such as Asynchronous Message Queue Protocol (AMQP), which is the default one, Message Queue Telemetry Transport (MQTT) for more lightweight communication, e.g., IoT devices, or Simple Text Orientated Messaging Protocol (STOMP). Given the approach focusing on versatility rather than performance, RabbitMQ requires more tuning than its counterpart to reach a comparable level of efficiency, however it can comfortably support thousands of messages on many different queues with minimal latency. Moreover, the increased versatility allows for more complex communication, allowing extreme freedom in terms of routing policies and other configurations. RabbitMQ also allows for a large set of additional utilities, available out of the box, such as queue Time To Live (TTL), disk persistency, lazy modes, dead letter exchanges for error management, single active consumers to avoid overcommitments and many more.

Last, RabbitMQ is extremely lightweight: while Kafka requires at the very least one broker and one Zookeeper instance for registry purposes, the former supports stand-alone deployments in a single container. The container image itself only occupies around 50MB and does not require additional tools, while Kafka requires around 350MB of raw data, without taking into consideration Zookeeper. RabbitMQ allows for extremely intuitive configurations, both through Command Line Interface (CLI) or static JSON files and the Management User Interface. This allowed for an easier prototyping phase, quickly realizing proofs of concept to test temporary solutions.

3.4. Messaging system

RabbitMQ can implement a wide variety of communication paradigms, from direct messaging between two services to a more complex multi-publisher/multi-subscriber mechanism. Regardless of the pattern,

the communication revolves around three main blocks: queues, exchanges and messages, which can be defined as follows:

- **Queue**: A queue where messages can be published and read. Queues represent the main infrastructure; it can be imagined as a pipe network.
- **Message**: A single entity in a queue, it contains a custom payload with some additional attributes, called properties. Its main property is the *routing key* which can be seen as the destination address.
- **Exchange**: exchanges act as a sort of "router". Any publisher writes to an exchange, which in turn decides where to send it, based on its type and the routing key of the message. Exchanges can belong to four different categories:
  - **direct**: the routing key has to exactly match the binding key of the exchange.
  - **fanout**: the message is sent to each queue bound to the current exchange, regardless of the binding key.
  - **topic**: the routing key can partially match the binding key; the message is sent to every queue whose binding matches a given pattern.
  - **header**: based on the message header, rarely used.

RabbitMQ also comprises other naming conventions, which are required to better understand its internal mechanisms, such as:

- **Binding**: distribution policy assigned to an exchange. A Binding literally binds a queue to a given pattern of routing keys, providing a destination endpoint for every message respecting the defined policy.
- **Acknowledgement** (ack): every message sent via AMQP protocol requires a read receipt, which can be automatic or deferred/controlled by the client libraries.
- **Dead Letter Queue (DLQ)**: an additional queue where messages that cannot be sent are placed, it usually needs to be created by hand. The DLQ does not automatically handle such messages, but it's typically exploited to handle errors and exceptions without losing the payloads.
- **Channel**: on its core represents a multiplex Transmission Control Protocol (TCP) stream in a single TCP connection, abstracted in the form of a channel. A channel represents a communication link between client and broker.
- **Virtual Host**: a virtual host represents an isolated group of users, exchanges, bindings and queues. It appears in the form of URL-like names, such as '/production', and allows for different work environments in the same broker instance.

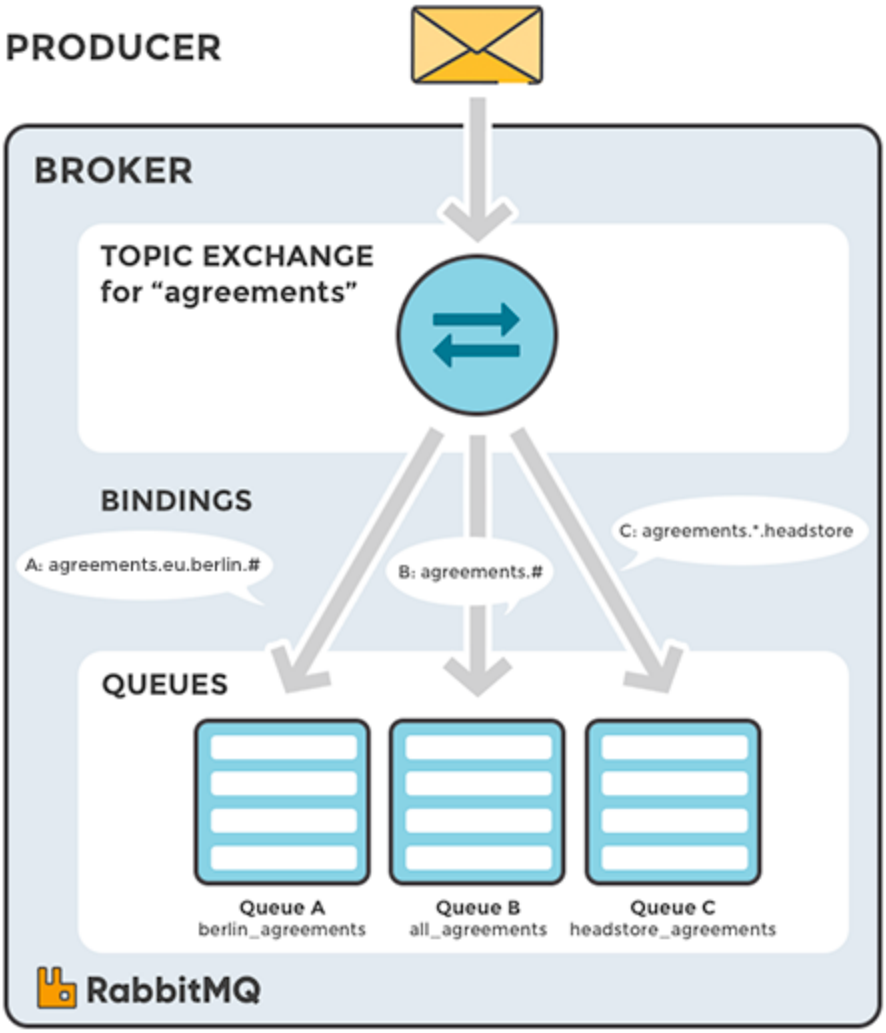


Figure 5 Simple example of topic exchange

Given that the whole implementation exploits the topic exchange for every communication, we provide here a brief description of its workings. The topic exchange is the most generic one, allowing complex interactions between senders and receivers. It can be compared to Kafka topics, where the routing key of the message can be potentially split into many components, representing an increasingly specific context. In RabbitMQ, senders and receivers work in slightly different ways: senders write to exchanges, while receivers read directly from their specific queue. This means that the publisher does not know where the message will be sent, it just provides a routing key and together with the message payload over to the exchange. The exchange matches the routing key with its binding keys, determining which queues should receive a message. A single message is broadcasted to every queue with a matching binding: for instance, in the example provided in Figure 5, a message with a routing key "agreement.eu.berlin.123" will be dispatched to both queue A and queue B, thus informing both consumers. On the consumer side instead,

there are no a priori filtering mechanisms. Therefore, bindings should be carefully designed so that the former only receives messages he's interested in.

Conventionally, exchanges, queue names and routing keys are provided as a list of lowercase words separated by a point (.). This allows to define arbitrarily complex routing keys that can subsequently be matched with simpler or more complex bindings. The hash symbol (#) allows for a match of zero or more words. For example, exchange B in the figure, "agreements.#" allows for routing keys such as "agreements", "agreements.eu.paris" or "agreements.eu.berlin.123". The asterisk symbol (*) allows instead for a single match in the specified position. In the specific case of exchange C, the binding "agreements.*.headstore" will bind "agreements.eu.headstore" but will not match "agreements.eu.paris.headstore". Thus, the routing keys used by providers are composed of increasingly specific categories allowing the binding keys used by subscribers to be as specific as possible.

3.4.1. Implementation details

Currently, the SAFERS message bus comprises a single RabbitMQ instance deployed at https://bus.safers-project.cloud. To guarantee a certain degree of security without burdening clients with specific certificates, our current configuration exploits nginx, a popular lightweight web server, as both HTTP proxy for the administration interface and TCP proxy for the AMQP protocol. In this way, the domain certificates are also exploited for SSL communication, without ad-hoc TLS support. Each client can therefore connect to the given hostname, using the standard port 5674, by providing only plain credentials in a standard SSL context. To ease the development process, three different virtual hosts have been provided: "/example," "/safers-test," and "/safers". The first one is dedicated to the development of each service and as initial testing ground, the second serves for integration between services, while the latter will provide the production-ready environment for the final components.

To avoid creating an exceeding number of credentials, each technical partner has a single account. To handle most of the use cases, each service has its own queue, whose name is defined as *q<partner name>.<service name>*. Whenever the partner has only one service, then only the partner's identifier is adopted as the queue name.

Every queue includes a set of configuration parameters to minimize resource consumption and avoid congestions. Specifically, each structure includes a TTL of 24 hours, meaning that after one day any message from the day before will be dropped if not read, lazy mode, where messages are kept on disk as much as possible instead of using RAM, and Single Active Consumer (SAC), meaning that only one consumer can connect at the same time, on the same queue. In order to ensure persistence of the content of these messages, the Dashboard copies read messages' content into its local database so that end-users can track historical data (the is described in greater detail in Section 4 below).

To allows for a complete pub/sub setup, messages should be published on a single exchange, named *safers.b2b*, unless required otherwise. Since RabbitMQ supports a list of special fields, messages sent through the message bus should provide the following properties:

1. message_id: an identifier for the given message (e.g. an ID generated by the sender to track results). Uniqueness is not guaranteed at bus level for this key: the only purpose of this field is to provide a tracker for the original sender of the message, or "dumb" services that replicate the ID elsewhere.
2. user_id: username of the partner that created the message. This must coincide with the RMQ username and can be exploited to identify the sender.
3. app_id: string without any kind of intrinsic meaning, but useful to identify the service that created the message. The code of each Intelligent Service is defined in the End Users Requirements.
4. delivery_mode: must be set to “persistent”; this is extremely important to make sure that the message will persist even after a broker restart.
5. timestamp: useful, albeit not required, as it allows for an easier tracking of messages through time.

Each message is automatically dispatched to the right queue by the exchange. A non-exhaustive list of the currently active bindings is summarized in Table 2. It is important to note that the definition of these bindings represents a proposal that is subject to change in the following months, due to test iterations and other implementation requirements.

Table 2 RabbitMQ bindings proposed for the implementation and currently being tested.

Topic	Example bindings	Description
newexternaldata.<wp>.<task>.<datatype_id>	newexternaldata.#, newexternaldata.3.# newexternaldata.3.1.* newexternaldata.3.2.32001	Notifies about the insertion, update, delete of data into the GeoData Repository. This is issued by the GDR.
status.<service_id>.<datatype_id>.<request_code>	status.# status.imp.# status.*.32001.*	Notifies about the outcome of a data import procedure. This is issued by the importer
event.<domain>.<type>	event.social.# event.# event.detected	Creation/Detection of an emergency event. This can be issued by event-based services or by

		the social media module
request.<datatype_id>.<request_code>	request.# request..32001.astro.1234	Request a mapping activity. This can be issued by the dashboard to trigger EO-based services.
propagator.<action/output>	propagator.start propagator.isochrone	Starts the fire propagation simulation.
mm.report.<id>	mm.report.created mm.report.updated	Notification that is triggered when a report is created or modified.
mm.mission.<status>	mm.mission.created mm.mission.closed	Notification that is triggered when a mission is created, deleted or modified. Issued by the crowdsourcing solutions.
mm.communication.<status>	mm.communication.created	Notification triggered when a new communication is sent or updated. Issued by the crowsourcing module.

The most important parameters are the *datatype ID* and the *request code*. The former uniquely identifies the output produced by a given service, in compliance with the Data Mapping Form. The latter refers instead to a composite key, necessary to discern between messages. This key is a combination of the partner's name and the message ID property. RabbitMQ does not guarantee the uniqueness of a value across the virtual host, thus with this method, provided that the message ID adopted is unique on the client's end, it is possible to unequivocally identify each message across the platform.

All data passed to and from the queue will be JSON. If geographical data is being posted to the GeoJSON format will be used. However, if the geographical data is very large or is not available as JSON or requires post-processing then the message can include a point (ie: a URL) to the actual data.

3.4.2. Data Flows

Messages can flow between SAFERS services in multiple ways. A SAFERS service might periodically publish messages to the queue as shown in Figure 6. In this example, no geographic data is being referenced. Therefore, neither the Geodata Repository nor the Importer are involved in the data flow. Instead, a simple status message is sent to the queue.

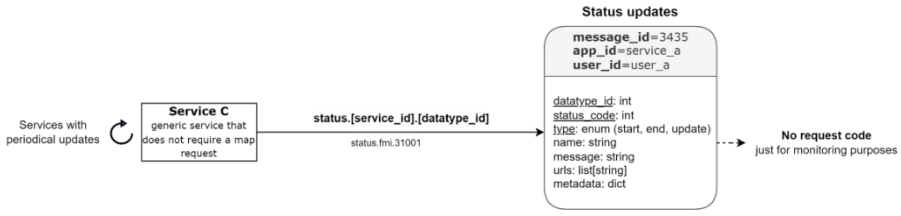


Figure 6 Periodic Notifications

Alternatively, a message might be published because of new data being sent to the Geodata Repository as shown in Figure 7. In this example a SAFERS service triggers a “newexternaldata” message to be sent to the bus. The exchange routes this message to a queue that the Importer is subscribed to. The Importer converts that data to an appropriate format and sends a status message to the bus. That status message is routed by the exchange to the Dashboard which can now make a map request to retrieve the converted data.

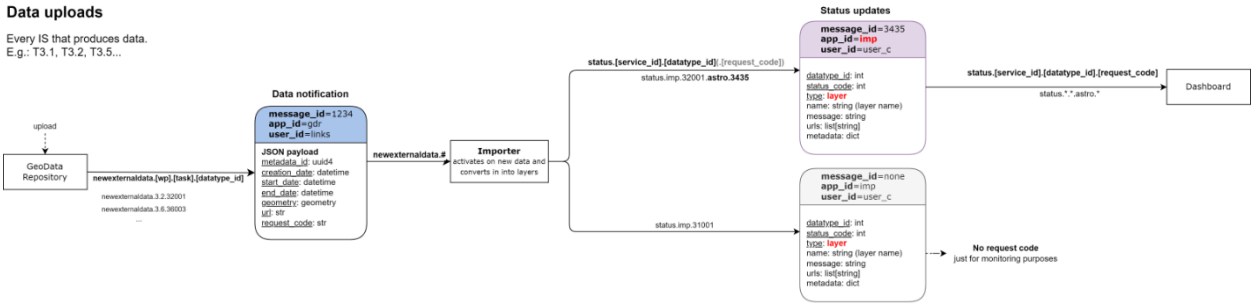


Figure 7 Data Uploads

Once it receives a message that new data is available the Dashboard must determine if that data is appropriate for the end-user of the dashboard; It must correspond to an Area Of Interest (AOI) the user has selected and/or an ongoing Fire Event that they are monitoring. If so, the Dashboard publishes a new

message to the bus in order to request the actual data. The exchange routes this to the appropriate queue and another SAFERS service receives that message which causes them to send yet another message listing the URLs where the data can actually be retrieved.

4. WEB-BASED DASHBOARD FOR OPERATIONAL MANAGEMENT AND DECISION SUPPORT

The user guide for this module is available at: Dashboard User Guide

The SAFERS dashboard is a comprehensive web application designed to provide an intuitive and interactive user interface for operational management. It offers control room users a unified platform to oversee citizens, volunteers, and professionals. Through seamless integration with various data sources, this web-based dashboard, built on the REACT Js framework (version 12.22.12) and enhanced with Typescript (version 3.9.5), interacts with the web-based EO Big Data backend server to access and process relevant information.

4.1 Key Features

1. **Dynamic Map Visualization**: Provides spatial and temporal overviews of key features.
2. **Real-Time Tracking**: Allows tracking of volunteers and professionals via the SAFERS Chatbot.
3. **Multimedia Reports**: Consolidates user-shared multimedia content for control room insights.
4. **Mission Management**: Enables creation and assignment of on-field tasks to volunteers and professionals, generating mission-specific reports.
5. **Field-Cameras**: Displays visual data from various cameras, supporting feedback and tagging for fire or smoke detections.
6. **Communication Updates**: Allows drawing communication areas, specifying user categories and disseminating messages via the SAFERS Chatbot.
7. **Map Requests and Simulations**: Facilitates obtaining fire or flood-related mappings and performing wildfire simulations.
8. **Advanced Filtering**: Provides time and feature type filters through a user-friendly filter bar.
9. **Admin Management**: Enables admin users to manage organizations, teams, and users.
10. **Weather Forecast and Environmental Layers**: Integrates short-term deterministic and medium-range/sub-seasonal probabilistic weather layers for enhanced decision-making. Each layer can be played, downloaded, and further explored through metadata, legend, timeseries, and feature info for individual data points.

4.2 Detailed Features

1. **Mapping and Geospatial Data**: Handles geospatial data visualization and manipulation with `react-map-gl`, `react-map-gl-draw`, `@turf libraries`, and `@types/geojson`.
2. **Data Visualization**: Utilizes libraries like `@nivo` for creating interactive charts (bar, line, pie) and d3 for complex data-driven visualizations.
3. **UI Components**: Employs `@material-ui` and `antd` for a rich set of pre-designed UI components and layouts, ensuring a polished and consistent look and feel.
4. **State Management**: Integrates `@reduxjs/toolkit` and `redux-thunk` for efficient state management, enabling robust and scalable state handling.
5. **Internationalization**: Supports multiple languages using `i18next` and its browser language detector.

4.3 Dependencies Breakdown

- **UI/UX Libraries**:
  - `@material-ui/core`, `@material-ui/icons`, `@material-ui/lab`: Material Design components and icons.
  - `@mui-treasury/layout`, `@mui-treasury/mockup`: Layout management and mockup components.
  - `antd`: Comprehensive set of UI components for React.
- **Data Visualization**:
  - `@nivo/bar`, `@nivo/core`, `@nivo/line`, `@nivo/pie`: Nivo charts for various data visualization needs.
  - `d3`: Powerful library for producing dynamic, interactive data visualizations.
- **State Management**:
  - `@reduxjs/toolkit`, `redux`: Modern, Redux-based state management.
  - `redux-thunk`: Middleware for handling asynchronous actions in Redux.
- **API and Data Handling**:
  - `axios`, `axios-hooks`: HTTP client and React hooks for making API requests.
  - `@fusionauth/typescript-client`: TypeScript client for interacting with FusionAuth.
  - `ermes-backoffice-ts-sdk`, `ermes-ts-sdk`: SDKs for interacting with the Ermes back office and core services.
- **Internationalization**:
  - `i18next`, `i18next-browser-languagedetector`, `i18next-xhr-backend`: Libraries for internationalization and language detection.

- **Geospatial Libraries**:
  - `@turf/difference`, `@turf/helpers`: Geospatial analysis tools.
  - `react-map-gl`, `react-map-gl-draw`: React components for integrating Mapbox GL JS maps.
  - `@types/geojson`: TypeScript definitions for GeoJSON.
- **Utility Libraries**:
  - `lodash.debounce`, fast-deep-equal, seedrandom: Utility functions for various needs.
  - `moment`: Library for parsing, validating, manipulating, and displaying dates.
  - `nanoid`: Secure URL-friendly unique ID generator.
  - `polished`: Utility toolkit for writing styles in JavaScript.
  - `immutable`: Immutable collections for JavaScript.
- **Styling**:
  - `styled-components`: Library for CSS-in-JS.
  - `node-sass`: Node bindings to LibSass for compiling SCSS to CSS.

4.3 Development Scripts

Build and Deployment:

- `build-dev`, `build-test`: Commands tailored for different environments, such as development and testing.
- `deploy-dev`, `deploy-test`: Deployment scripts for different environments, utilizing Azure.

4.4 Conclusion

The SAFERS Frontend Dashboard is a well-architected application that leverages a comprehensive set of modern tools and libraries to deliver a rich, interactive user experience. Its modular design, robust state management, and extensive use of visualization and geospatial tools make it a powerful solution for data management and presentation needs. It provides an intuitive tool for operational management, empowering users with real-time data, communication capabilities, and advanced visualization tools.

5. GEODATA REPOSITORY MANAGEMENT

5.1. Introduction

The Geodata Repository is designed to serve as a comprehensive repository for storing various types of data, including geospatial data. This data can be structured, semi-structured, or unstructured, and is either supplied directly by the system or produced by intelligent services. Data can be uploaded to the platform

through exposed APIs or web-based interfaces, which provide functionalities for uploading, retrieving, and filtering both INSPIRE-compliant metadata and data. The module leverages Big Data storage to maintain data in its raw format and a structured geospatial database (DB) for storing associated metadata compliant with the INSPIRE metadata directive (ISO 19115/19119), as detailed in ANNEX I. The INSPIRE metadata directive ensures interoperability of geospatial data across the European Community. The geospatial DB supports the execution of geospatial queries as well as other queries on the metadata fields. The Geodata Repository communicates with other system modules via a message broker (detailed in Section 3.2) that follows the publisher/subscriber paradigm, notifying subscribed services of new data availability. This component is open-source and available at GitHub repository.

5.2. Architecture

The Geodata Repository is designed to handle large volumes of heterogeneous data, making traditional relational databases impractical for effective management and processing. Instead, a big data storage system is utilized. The concept of a data lake is employed to manage all data generated by the organization, stored in their original forms in an Azure Blob Storage.

The Geodata Repository comprises two main components:

1. **Big Data Storage:** This component stores all data in its raw format. The current version of the component is configured to use Azure Blob Storage.
2. **Data Catalog:** This management component orchestrates data ingestion, handles data requests, and store the metadata. It is implemented levering a custom version of CKAN [12], an open-source data portal designed to allow publishing, sharing, and managing data sets. The basic version of CKAN has been enhanced with the following extensions:
    - Spatial: add a spatial field on the default CKAN dataset schema, it uses PostGIS as the backend and allows to perform spatial queries and to display the dataset extent on the frontend.
    - Datesearch: in-house modified version of [33]. It adds start/end validity date of the metadata, and it allows to search the datasets valid on those time span.
    - OAuth2: It allows integration with FusionAuth or any other OAuth2.0 protocol. Starting from an old version of this plugin [34], we made the necessary conversion to Python3 and Flask to make it compatible with the last version of CKAN.
    - Geoview: It contains view plugins to display geospatial files and services in CKAN
    - Scheming: It allows to create custom metadata fields and validate the entries. It currently implements INSPIRE metadata.
    - Cloudstorage: It allows to use the main cloud storage services for storing data. It is currently set to use Azure Blob Storage.
    - Notify: In-house developed plugin. It allows to send a notification to a messaging bus system everytime a dataset is updated/created/deleted
    - Theme: allows GUI customization
    - Hidegroups: hides 'group' from the Front End (because useless for this purpose)
    - authcheck: In-house developed plugin. Check if the user has the permission to modify / delete a certain dataset (Only the creator can modify/delete its own dataset)

- datatype: In-house developed plugin. It allows to metadata to inherit the datatype ids of the resources associated with it.

The repository interfaces with two key components:

- **Identity Server:** Manages authorization for accessing stored data (Section 3.1).
- **Messaging System:** Notifies other microservices about the availability of new data (Section 3.2).

The architecture of the Geodata Repository is illustrated in Figure 8.

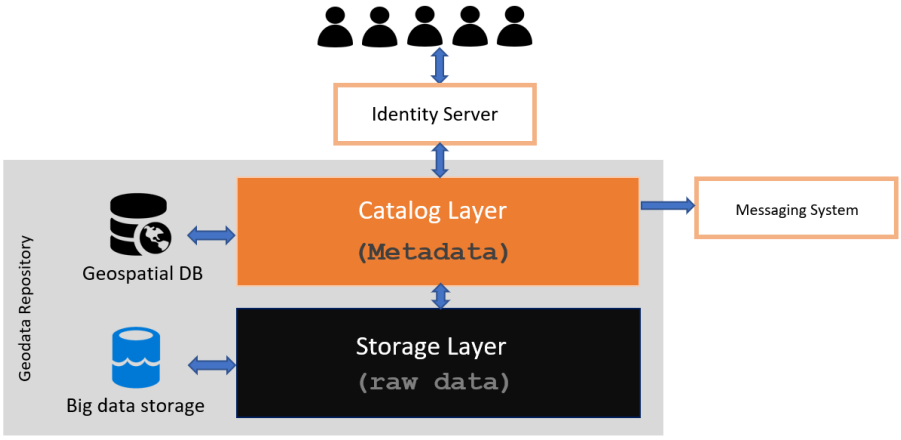


Figure 8 Geodata repository architecture

5.3. Data Flows & Exchanges

Any data provider has the access to the Geodata Repository to upload new resources and to search among the ones uploaded by the other partners. The authenticated user can both use API or web-based graphical interface to add one or more resources accompanied by INSPIRE metadata to describe the files. Following, the Geodata Repository publishes a notification on the messaging bus. Figure 9 shows the data flow and the structure of the notification sent to the messaging system used to allow the communication among the SAFERS platform components.

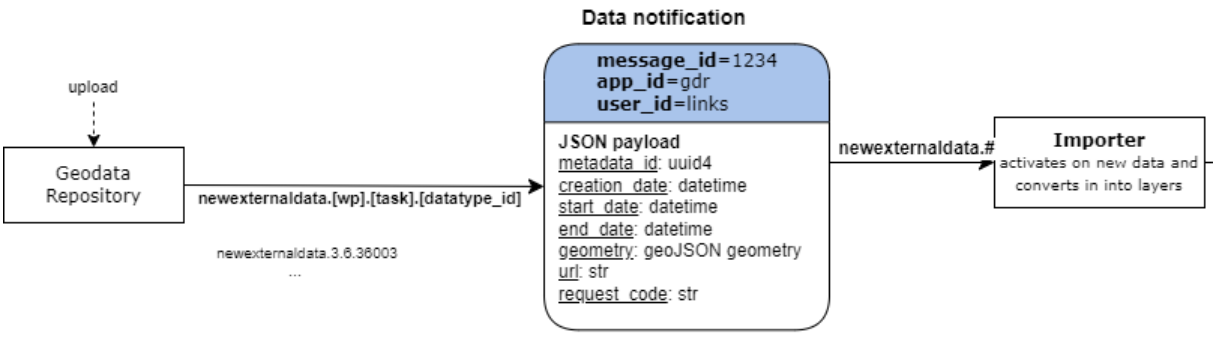


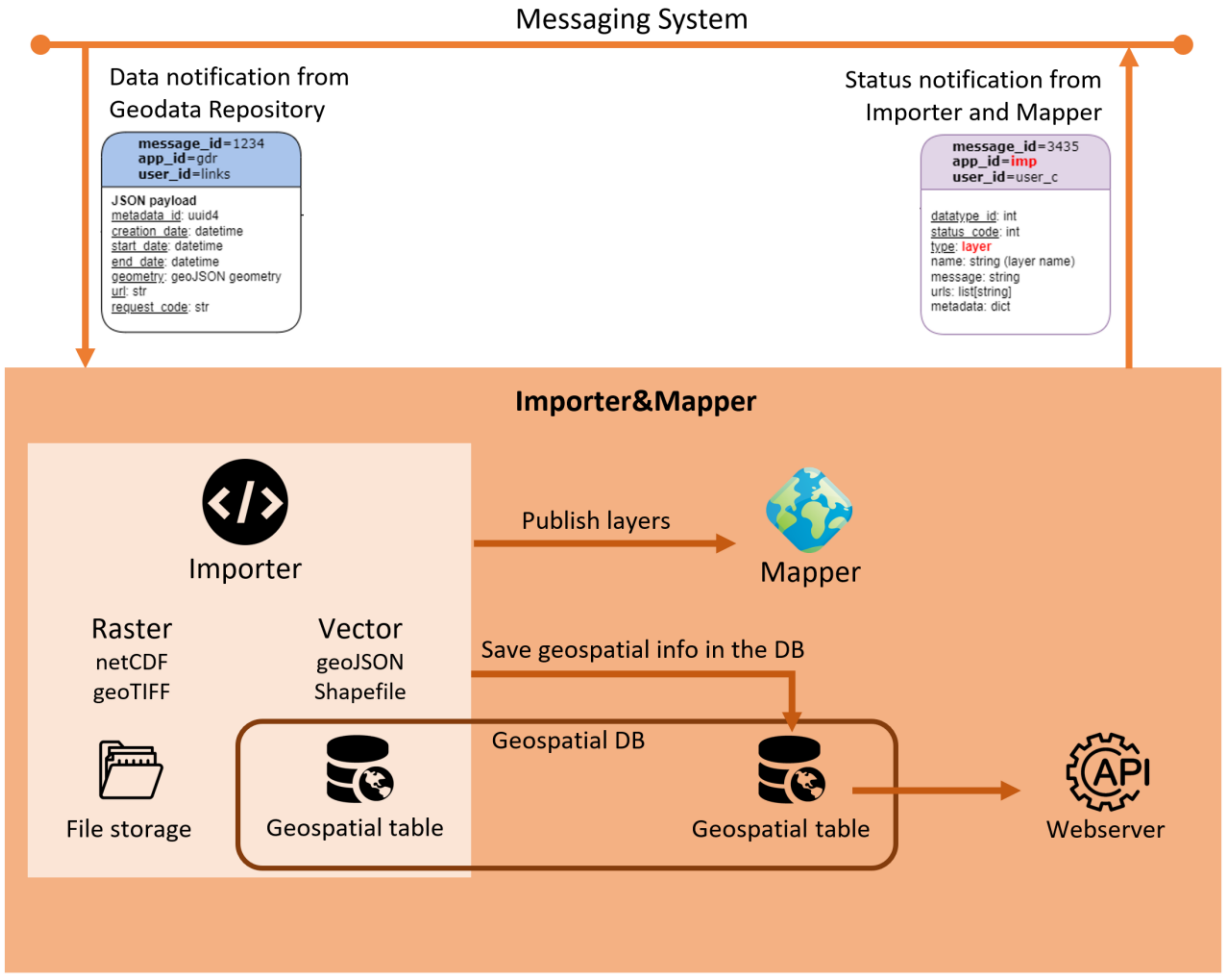
Figure 9 Data flow - Upload to the Geodata Repository to the publishing of the maps as layers.

6. IMPORTER AND MAPPER

6.1. Introduction

This module aims to import the geospatial data uploaded on the Geodata repository into a map server and serve them by means of OGC services such as WMS, WMTS, WFS or WCS. It is available open-source at this link.

6.2. Architecture



1. **Figure 10 Architecture and data flow of the Importer and Mapper**

The module is composed of four main interconnected functional elements (Figure 10):

- Mapper: A map server (GeoServer) exposing and giving access to geospatial data generated via OGC services such as WMS, WMTS, WFS, or WCS.
- Importer: A Python routine that, upon receiving a notification indicating a change in the raw data available in the Geodata Repository, performs the necessary updates to the Mapper. This includes fetching relevant data and creating, updating, or deleting map layers of interest. The Importer also generates a list of available layers, which is provided in a convenient format for subsequent requests.
- A support Geospatial DB (PostgreSQL) to provide persistent storage for the Geoserver and the Importer, holding the resource list, that is the set of available layers and relative metadata, and storing the vector layers themselves.

- An API server (FastAPI) to allows external clients to query and access the list of available layers.

6.3. Data flows

The Importer is listening on the messaging bus for any notification produced by the Geodata Repository. Depending on the notification action type (*create, update, delete*), the Importer performs different actions:

Update/Delete

Retrieves the list of previously created layers associated with the resource ID from the support database, deletes layers from the Mapper, removes local file copies or database tables, and soft deletes DB entries.

Create/Update

Downloads the resource file to local temporary storage, processes the file based on its type (supported formats: GeoJSON, Shapefile, GeoTIFF, NetCDF), and publishes new layers via the Mapper's API. GeoJSON and Shapefile contents, corresponding to vector layers, are transferred to a new table of the supporting database, and the new map layer is created via an API call referencing the table. GeoTIFF and NetCDF files are instead moved from the Importer's temporary storage to the Mapper's local and persistent storage, and those of the first type are published directly as a single layer with one or more variables (or *bands*). NetCDF files originate instead multiple layers, that are loaded one at a time and separately, according to a configuration file attached to the Importer. This configuration lists the native variable names of interest that must be included in the file and maps them to a specific single or multiband datatype: a layer for each of these variables is attempted to be created regardless of its presence in the actual file, or the presence of additional variables that are ignored.

For all filetypes, a style id is then associated to successfully published layers according to their datatype, retrieving from the same configuration file the mapping between layer datatypes and styles. An entry for each of these layers is then added to the resource list stored in the supporting database. Finally, for each of the layers attempted to be imported, a notification indicating the success of the operation or the reason of failure (such as the absence of the relative variable in the file) is emitted on the message broker (Figure 10). Some example of imported layers in Figure 11, Figure 12.

At the end of each import iteration, space-saving constraints are enforced, ensuring that the number of layers or their maximum age does not exceed defined limits.

When queried, the API server retrieves resource entries from the support database based on request parameters, providing formatted lists to clients.

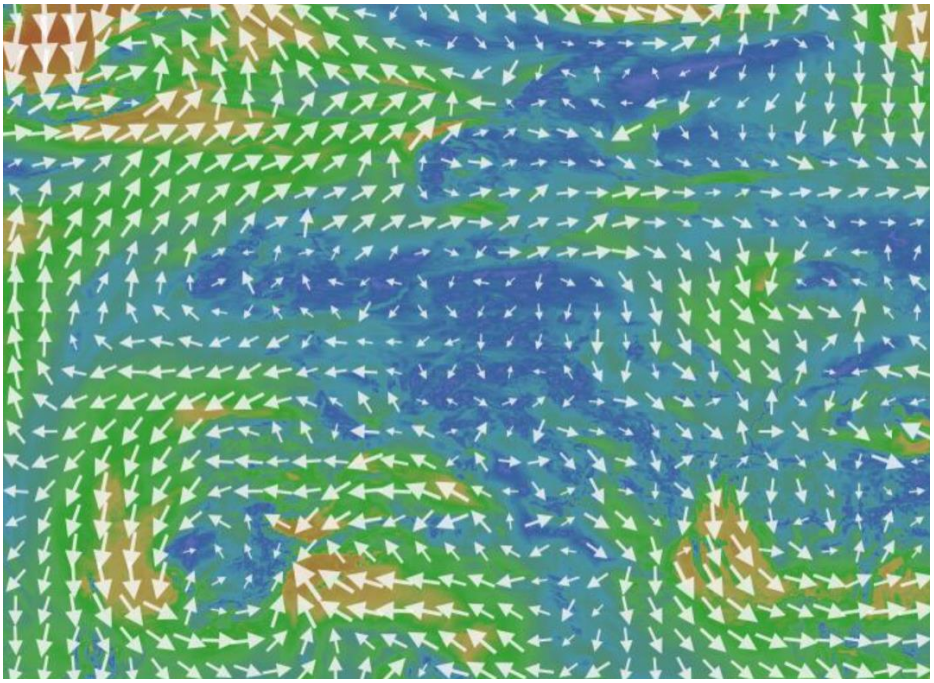


Figure 11 Short term forecast, wind (speed and direction), datatype ID 33110

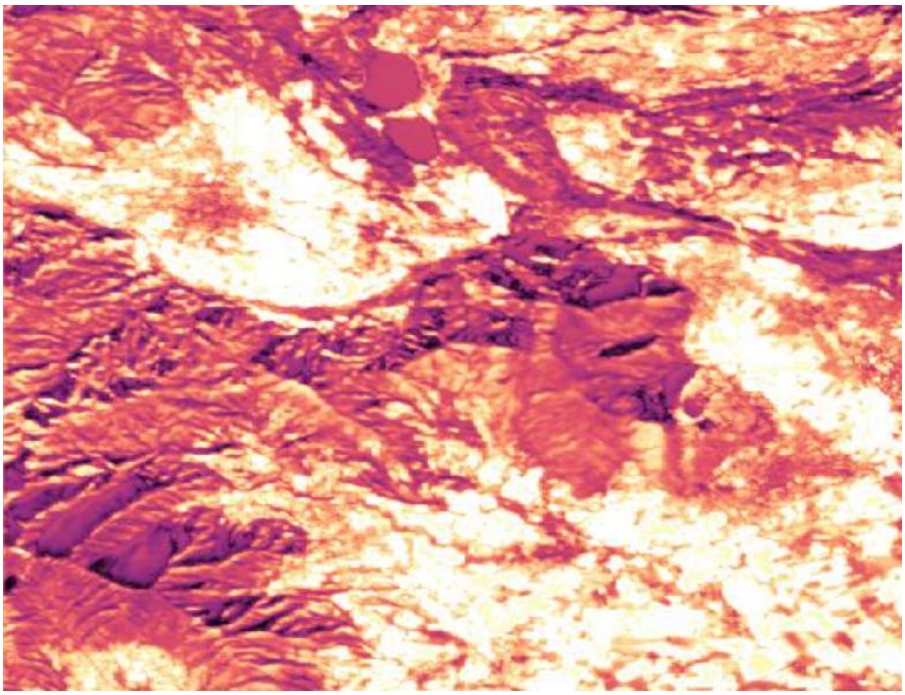


Figure 12 Environment recovery, Burned area severity map, datatype ID 36002

7. CONCLUSIONS

This document has described some of the major components that comprise the SAFERS Architecture:

- The Customer Identity and Authentication Management (CIAM)
- The Message Bus
- The Web-Based Dashboard
- The Backend API
- The Geodata Repository
- The Importer and Mapper

The CIAM provides a common means of authentication for all users of SAFERS. It also stores user profile information in one central place, eliminating the need for the other SAFERS components to manage their own users. *Except for* the Dashboard. As a web-based end-user tool for operational management and decision support, it requires that users are able to save their interest in artefacts associated with particular fire events. This "relationship" information is separate from the user profile information stored in the CIAM. It is stored in the Dashboard and persisted in its Backend API between sessions. The Message Bus is used to route asynchronous communication throughout SAFERS. The Backend API provides a way for the Dashboard to publish messages and subscribe to messages on the Message Bus. Each component has its own dedicated queue in the bus for the RabbitMQ exchange to send the appropriate messages to. Messages are published to the bus when data is added to the Geodata Repository. The Importer and Mapper, since they have subscribed to those types of messages, are triggered to process the data so it can be served by the Map Server and viewed by the Dashboard.

2. REFERENCES

[1] "AspNetBoilerplate," [Online]. Available: https://aspnetboilerplate.com/Pages/Documents/Introduction.

[2] "AspNetBoilerplate layers," [Online]. Available: https://aspnetboilerplate.com/Pages/Documents/NLayer-Architecture.

[3] "ASP.NET Core," [Online]. Available: https://en.wikipedia.org/wiki/ASP.NET_Core.

[4] "EF Core," [Online]. Available: https://docs.microsoft.com/en-gb/ef/core/.

[5] "CIAM Module," [Online]. Available: https://en.wikipedia.org/wiki/Customer_identity_access_management..

[6] "OAuth 2.0," [Online]. Available: https://oauth.net/2/.

[7] "Open ID Connect," [Online]. Available: https://openid.net/connect/.

[8] "GDPR", [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[9] "The OAuth 2.0 Authorization Framework", [Online]. Available: https://www.ietf.org/rfc/rfc6749.txt.

[10] "Apache KAFKA", [Online]. Available: https://kafka.apache.org.

[11] "RabbitMQ", [Online]. Available: https://www.rabbitmq.com/.

[12] "https://ckan.org/".

[13] "Natalia Miloslavskaya, Alexander Tolstoy, Big Data, Fast Data and Data Lake concepts, Procedia Computer Science, Volume 88, 2016, Pages 300-305, ISSN 1877-0509,".".

[14] "Bohlouli, M., Schulz, F., Angelis, L., et al., 2013. Towards an integrated platform for big data analysis. In: Fathi, M. (Ed.), Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives. Springer Berlin Heidelberg, p.47–56".

[15] "Azeem, R., Khan, M.I.A., 2012. Techniques about data replication for mobile ad-hoc network databases. Int. J. Multidiscipl. Sci. Eng.".

[16] "Wang, X., Sun, H.L., Deng, T., et al., 2015. On the tradeoff of availability and consistency for quorum systems in data center networks. Comput. Network".

[17] "Oracle, 2015a. Managing Consistency with Berkeley DBHA (White Paper).".

[18] "Hilker, S., 2012. Survey Distributed Databases—Toad for Cloud".

[19] "Cattell, R., 2010. Scalable SQL and NoSQL data stores. SIGMOD Rec.".

[20] "Siddiqa, A., Karim, A. & Gani, A. Big data storage technologies: a survey. Frontiers Inf Technol Electronic Eng 18, 1040–1070 (2017).".

[21] "Borthakur, D., 2008. HDFS Architecture Guide.".

[22] "Shvachko, K.V., 2010. HDFS scalability: the limits to growth.".

[23] "https://azure.microsoft.com/en-us/services/storage/blobs".

[24] "PostGIs http://postgis.net/".

[25] "OGC Specifications, Simple Feature Access Part 1 - Common Architecture http://www.opengeospatial.org/standards/sfa".

[26] "OGC Specifications, Simple Feature Access Part 2 – SQL Options http://www.opengeospatial.org/standards/sfs".

[27] "ArcGIS http://www.esri.com/software/arcgis".

[28] "GeoServer http://geoserver.org/".

[29] "INSPIRE Metadata technical guidelines," [Online]. Available: https://inspire.ec.europa.eu/documents/inspire-metadata-implementing-rules-technical-guidelines-based-en-iso-19115-and-en-iso-1.

[30] "https://geonetwork-opensource.org/".

[31] "http://geonode.org/".

[32] [Online]. Available: https://github.com/ckan/ckanext-spatial.

[33] [Online]. Available: https://github.com/geosolutions-it/ckanext-datesearch/tree/c007.

[34] [Online]. Available: https://github.com/conwetlab/ckanext-oauth2.

[35] [Online]. Available: https://github.com/FedericOldani/ckanext-oauth2/tree/flask_conversion.

[36] [Online]. Available: https://github.com/ckan/ckanext-geoview.

[37] [Online]. Available: https://github.com/ckan/ckanext-scheming.

[38] [Online]. Available: https://github.com/open-data/ckanext-cloudstorage.

[39] [Online]. Available: https://github.com/okfn/ckanext-hidegroups.

[40] Microsoft Corporation, "Azure Virtual Machines," [Online]. Available: https://azure.microsoft.com/en-us/services/virtual-machines/.

[41] "Geodata Repository," LINKS Foundation, [Online]. Available: https://datalake-test.safers-project.cloud/.

[42] [Online]. Available: https://redis.io/.

[43] "[RD09] The GeoJSON Format https://datatracker.ietf.org/doc/rfc7946/".

[44] "JSON http://www.json.org/".

[45] "ESRI Shapefile Technical Description http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf 01/07/1998".

[46] "GDAL - ogr2ogr," [Online]. Available: https://gdal.org/programs/ogr2ogr.html.

[47] [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/.

[48] [Online]. Available: https://www.ogc.org/standards/netcdf.

[49] LINKS Foundation, "SAFERS Importer & Mapper API Swagger," 2022. [Online]. Available: https://geoapi-test.safers-project.cloud/docs.

[50] I. Inc, "FusionAuth and the GDPR," [Online]. Available: https://fusionauth.io/learn/expert-advice/ciam/developers-guide-to-gdpr#fusionauth-and-the-gdpr.

[51] OSGeo, "GeoServer," [Online]. Available: http://geoserver.org/.

[52] The PostgreSQL Global Development Grou, "PostgreSQL," [Online]. Available: https://www.postgresql.org/.

[53] OSGeo, "PostGIS," [Online]. Available: https://postgis.net/.

[54] OSGeo, "GeoServer API Documentation," [Online]. Available: https://docs.geoserver.org/stable/en/user/rest/index.html.

[55] SQLAlchemy authors and Contributors, "SQLAlchemy," [Online]. Available: https://www.sqlalchemy.org/.

[56] S. Ramírez, "FASTAPI," [Online]. Available: https://fastapi.tiangolo.com/.

[57] Encode OSS, "Uvicorn ASGI Web Server," [Online]. Available: https://www.uvicorn.org/.

[58] Docker, Inc, "Docker," [Online]. Available: https://www.docker.com.

[59] Docker, "Docker Compose," [Online]. Available: https://docs.docker.com/compose/.

[60] Links Foundation, 2022. [Online]. Available: https://geoserver-test.safers-project.cloud/geoserver/web/.

[61] C. Rossi and L. Bruno, "SAFERS Deiverables D2.3 Technical Requirements & System Architecture," 2021.

[62] M. David, "SAFERS Deliverables D2.2 End-User Requirements," 2021.

[63] Rossi, C. and Bruno, L., 2021. Deliverable: D2.3 Technical Requirements & System Architecture. SAFERS Deliverables. pp.17-22..

[64] Martin, D. et al., 2021. Deliverable: D2.2 End-User Requirements. SAFERS Deliverables.

[65] "CIAM Module," [Online]. Available: https://en.wikipedia.org/wiki/Customer_identity_access_management.

3. ANNEX I

All the datasets that will be produced shall be accompanied by metadata. Metadata will be delivered in an INSPIRE Directive (2007/2/EC) conform ISO 19115/ISO 19119 metadata standard.

The overall structure of the ISO 19115/ISO 19119 metadata set supporting the requirements expressed in the INSPIRE Implementing rules for metadata, is included and explained in the "INSPIRE Metadata Implementing Rules: Technical Guidelines based on EN ISO 19115 and EN ISO 19119" document, which is available online (http://inspire.ec.europa.eu/documents/Metadata/MD_IR_and_ISO_20131029.pdf).

The minimum set of metadata elements necessary to comply with INSPIRE Directive have been provided to the partners.

In Table 3: INSPIRE compliant metadata, the INSPIRE[1] compliant metadata are listed. It has been defined selecting the mandatory field of the INSPIRE directive and it is applied to all data that will be inserted into the Geo Importer because through the IDI it implements a data repository capable of sharing data (files) and service map layers through OGC standards.

Table 3 INSPIRE compliant metadata

	Minimum set of metadata elements necessary to comply with Directive 2007/2/EC			
	A) Metadata according to INSPIRE Metadata Regulation (COMMISSION REGULATION (EC) No 1205/2008 of 3 December 2008 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards metadata)			
ref	**group of metadata elements**	**metadata element**	**CKAN API Field name**	**description**
1	IDENTIFICATION			

[1] https://inspire.ec.europa.eu/

1.1		Resource Title	title	This a characteristic, and often unique, name by which the resource is known. The value domain of this metadata element is free text.
1.2		Resource Abstract	notes	This is a brief narrative summary of the content of the resource. The value domain of this metadata element is free text.
1.3		Resource Type	identification_ResourceType	This is the type of resource being described by the metadata. The value domain of this metadata element is defined in Part D.1. 1.1. Spatial data set series (series) 1.2. Spatial data set (dataset) 1.3. Spatial data services (services)
1.4		Resource Locator	url	The resource locator defines the link(s) to the resource and/or the link to additional information about the resource. The value domain of this metadata element is a character string, commonly expressed as uniform resource locator (URL). This field is automatically generated when the file is uploaded in the geodata repository.
1.5		Unique resource identifier	name	A value uniquely identifying the resource. The value domain of this metadata element is a mandatory character string code, generally assigned by the data owner, and a character string namespace uniquely identifying the context of the identifier code (for example, the data owner).
1.6		Coupled Resource	identification_CoupledResource	If the resource is a spatial data service, this metadata element identifies, where relevant, the target spatial data set(s) of the service through

				their unique resource identifiers (URI). The value domain of this metadata element is a mandatory character string code, generally assigned by the data owner, and a character string namespace uniquely identifying the context of the identifier code (for example, the data owner).
1.7		Resource language	identification_ResourceLanguage	The language(s) used within the resource. The value domain of this metadata element is limited to the languages defined in ISO 639-2.
2	CLASSIFICATION OF SPATIAL DATA AND SERVICES			
2.1		Topic category	classification_TopicCategory	The topic category is a high-level classification scheme to assist in the grouping and topic-based search of available spatial data resources. The value domain of this metadata element is defined in Part D.2. (Topic categories in accordance with EN ISO 19115) - see CELEX_32008R1205_EN_TXT.pdf for details
2.2		Spatial Data Service Type	classification_SpatialDataServiceType	This is a classification to assist in the search of available spatial data services. A specific service shall be categorised in only one category. The value domain of this metadata element is defined in Part D.3. (discovery, view, download, transformation, invoke)
3	KEYWORD			If the resource is a spatial data service, at least one keyword from Part D.4 shall be provided. If a

				resource is a spatial data set or spatial data set series, at least one keyword shall be provided from the general environmental multilingual thesaurus (GEMET) describing the relevant spatial data theme as defined in Annex I, II or III to Directive 2007/2/EC. For each keyword, the following metadata elements shall be provided:
3.1		Keyword value	keyword_KeywordValue	The keyword value is a commonly used word, formalised word or phrase used to describe the subject. While the topic category is too coarse for detailed queries, keywords help narrowing a full text search and they allow for structured keyword search. The value domain of this metadata element is free text.
3.2		Originating controlled vocabulary	keyword_OriginatingControlledVocabulary	If the keyword value originates from a controlled vocabulary (thesaurus, ontology), for example GEMET, the citation of the originating controlled vocabulary shall be provided. This citation shall include at least the title and a reference date (date of publication, date of last revision or of creation) of the originating controlled vocabulary. There can be more than one because the keywords can own to more than one vocabulary
4	GEOGRAPHIC LOCATION			
4.1		Geographic Bounding Box	spatial	This is the extent of the resource in the geographic space, given as a bounding box. The bounding box

				shall be expressed with westbound and eastbound longitudes, and southbound and northbound latitudes in decimal degrees, with a precision of at least two decimals.
5	TEMPORAL REFERENCE			This metadata element addresses the requirement to have information on the temporal dimension of the data as referred to in Article 8(2)(d) of Directive 2007/2/EC. At least one of the metadata elements referred to in points 5.1 to 5.4 shall be provided. The value domain of the metadata elements referred to in points 5.1 to 5.4 is a set of dates. Each date shall refer to a temporal reference system and shall be expressed in a form compatible with that system. The default reference system shall be the Gregorian calendar, with dates expressed in accordance with ISO 8601.
5.1		Temporal extent	data_temporal_extent_begin_date data_temporal_extent_end_date	The temporal extent defines the time period covered by the content of the resource. This time period may be expressed as any of the following: — an individual date, - an interval of dates expressed through the starting date and end date of the interval, — a mix of individual dates and intervals of dates.
5.2		Date of publication	temporalReference_dateOfPublication	This is the date of publication of the resource when available, or the date of entry into force. There may be more than one date of publication.

5.3		Date of last revision	temporalReference_dateOfLastRevision	This is the date of last revision of the resource, if the resource has been revised. There shall not be more than one date of last revision.
5.4		Date of creation	temporalReference_dateOfCreation	This is the date of creation of the resource. There shall not be more than one date of creation.
6	QUALITY AND VALIDITY			
6.1		Lineage	quality_and_validity_lineage	This is a statement on process history and/or overall quality of the spatial data set. Where appropriate it may include a statement whether the data set has been validated or quality assured, whether it is the official version (if multiple versions exist), and whether it has legal validity. The value domain of this metadata element is free text.
6.2		Spatial Resolution	quality_and_validity_spatial_resolution_latitude quality_and_validity_spatial_resolution_longitude quality_and_validity_spatial_resolution_scale quality_and_validity_spatial_resolution_measureunit	Spatial resolution refers to the level of detail of the data set. It shall be expressed as a set of zero to many resolution distances (typically for gridded data and imagery-derived products) or equivalent scales (typically for maps or map-derived products). An equivalent scale is generally expressed as an integer value expressing the scale denominator. A resolution distance shall be expressed as a numerical value associated with a unit of length.
7	CONFORMITY			The requirements referred to in Article 5(2)(a) and Article 11(2)(d) of Directive 2007/2/EC relating to the conformity, and the degree of conformity, with

				implementing rules adopted under Article 7(1) of Directive 2007/2/EC shall be addressed by the following metadata elements:
7.1		Specification	conformity_specification_title conformity_specification_dataType conformity_specification_date	This is a citation of the implementing rules adopted under Article 7(1) of Directive 2007/2/EC or other specification to which a particular resource conforms. A resource may conform to more than one implementing rules adopted under Article 7(1) of Directive 2007/2/EC or other specification. This citation shall include at least the title and a reference date (date of publication, date of last revision or of creation) of the implementing rules adopted under Article 7(1) of Directive 2007/2/EC or of the specification.
7.2		Degree	conformity_degree	This is the degree of conformity of the resource to the implementing rules adopted under Article 7(1) of Directive 2007/2/EC or other specification. The value domain of this metadata element is defined in Part D.
8	CONSTRAINT RELATED TO ACCESS AND USE			
8.1		Conditions for Access and Use	constraints_conditions_for_access_and_use	This metadata element defines the conditions for access and use of spatial data sets and services, and where applicable, corresponding fees as required by Article 5(2)(b) and Article 11(2)(f) of Directive 2007/2/EC. The value domain of this metadata

				element is free text. The element must have values. If no conditions apply to the access and use of the resource, 'no conditions apply' shall be used. If conditions are unknown, 'conditions unknown' shall be used. This element shall also provide information on any fees necessary to access and use the resource, if applicable, or refer to a uniform resource locator (URL) where information on fees is available. Additional Requirements on Metadata according to regulation 1312/2014: The technical restrictions applying to the access and use of the spatial data service shall be documented in the metadata element "CONSTRAINT RELATED TO ACCESS AND USE" set out in Regulation (EC) No 1205/2008.
8.2		Limitations on Public Access	constraints_limitation_on_public_access	When Member States limit public access to spatial data sets and spatial data services under Article 13 of Directive 2007/2/EC, this metadata element shall provide information on the limitations and the reasons for them. If there are no limitations on public access, this metadata element shall indicate that fact. The value domain of this metadata element is free text.
9	RESPONSIBLE ORGANISATION			Organisations responsible for the establishment, management, maintenance and distribution of spatial data sets and services

9.1		Responsible party	responsable_organization_name responsable_organization_email	This is the description of the organisation responsible for the establishment, management, maintenance and distribution of the resource. This description shall include: — the name of the organisation as free text, — a contact e-mail address as a character string. Additional Requirements on Metadata according to regulation 1312/2014: The responsible party set out in Regulation (EC) No 1205/2008 shall at least describe the custodian responsible organisation, corresponding to the Custodian responsible party role set out in Regulation (EC) No 1205/2008.
9.2		Responsible party role	responsable_organization_role	This is the role of the responsible organisation. The value domain of this metadata element is defined in Part D.
10	METADATA ON METADATA			
10.1		Metadata Point of Contact	point_of_contact_name point_of_contact_email	This is the description of the organisation responsible for the creation and maintenance of the metadata. This description shall include: - the name of the organisation as free text, - a contact e-mail address as a character string.
10.2		Metadata Date	temporalReference_date	The date which specifies when the metadata record was created or updated. This date shall be expressed in conformity with ISO 8601.

10.3		Metadata Language	metadata_language	This is the language in which the metadata elements are expressed. The value domain of this metadata element is limited to the official languages of the Community expressed in conformity with ISO 639-2.
	B) Additional metadata elements according to INSPIRE Implementing Rules for interoperability of spatial data sets and services (Commission Regulation (EU) No 1089/2010; Art. 13 Metadata required for Interoperability) and relevant amendments			
		Coordinate Reference System	coordinatesystemreference_code coordinatesystemreference_codespace	Description of the coordinate reference system(s) used in the data set.
		Character encoding	character_encoding	The character encoding used in the data set.

Additional metadata have been defined in order to manage the internal operations. The most important additional metadata is the DataTypeID, which is adopted to identify the WP and Task that generates the dataset. The list of additional metadata is reported in Table 4.

Table 4 List of additional SAFERS metadata

11	**SAFERS internal metadata**			Metadata required by SAFERS
11.1		Metadata name	name	Name of the metadata
11.2		Owner Organization	owner_org	Organization which owns the data (written in lower case)

11.3		Visibility	visibility	Visibility of the dataset: “private” if only authenticated Data Lake users can access to the dataset, “public” to make it publicly available.
11.4		External attributes	external_attributes	JSON object to describe any additional attribute to describe the data

Following, an example of the JSON dictionary with the metadata information required to upload a new resource to the Geodata Repository:

```
{
    "title": "EMSR358: Flood in Bosnia and Herzegovina",
    "private": true,
    "notes": "Heavy rainfall that affected central and north-western part of Bosnia and Herzegovina caused
outflows of rivers and their tributaries in Una-Sana canton and along the Bosna river.",
    "identification_ResourceType": "dataset",
    "owner_org": "safers",
    "identification_CoupledResource": "",
    "identification_ResourceLanguage": "eng",
    "classification_TopicCategory": "inlandWaters",
    "classification_SpatialDataServiceType": "",
    "keyword_KeywordValue": "Riverine Flood, Hazard, Delineation Map",
    "keyword_OriginatingControlledVocabulary": "ontology",
    "data_temporal_extent_begin_date": "2019-05-12T00:00:00",
    "data_temporal_extent_end_date": "2019-05-12T00:00:00",
    "temporalReference_dateOfPublication": "2019-05-28T16:48:02",
    "temporalReference_dateOfLastRevision": "2019-05-28T16:48:02",
    "temporalReference_dateOfCreation": "2019-05-28T16:48:02",
    "quality_and_validity_lineage": "Quality approved",
    "quality_and_validity_spatial_resolution_latitude": "0",
    "quality_and_validity_spatial_resolution_longitude": "0",
    "quality_and_validity_spatial_resolution_scale": "25000",
```

```
    "quality_and_validity_spatial_resolution_measureunit": "m",
    "conformity_specification_title": "COMMISSION REGULATION (EU) No 1089/2010 of 23 November 2010 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards interoperability of spatial data sets and services",
    "conformity_specification_dateType": "publication",
    "conformity_specification_date": "2010-12-08T00:00:00",
    "conformity_degree": "true",
    "constraints_conditions_for_access_and_use": "cc-by",
    "constraints_limitation_on_public_access": "",
    "responsable_organization_name": "Copernicus EMS Rapid Mapping Team",
    "responsable_organization_email": "mapping@copernicus.com",
    "responsable_organization_role": "author",
    "point_of_contact_name": "Copernicus EMS Rapid Mapping Team",
    "point_of_contact_email": "mapping@copernicus.com",
    "temporalReference_date": "2020-11-05T00:00:00",
    "metadata_language": "eng",
    "coordinatesystemreference_code": "4326",
    "coordinatesystemreference_codespace":"EPSG",
    "character_encoding":"UTF-8",
    "spatial":{
        "type":"MultiPolygon",
        "coordinates":[
           [
              [
                 [
                    -23.43761444091797,
                    67.78710907838104
                 ],
                 [
                    38.43738555908203,
```

```
                    68.31273876336299
                ],
                [
                    37.03113555908203,
                    54.43598125708997
                ],
                [
                    -0.9376144409179688,
                    54.43598125708997
                ],
                [
                    -23.43761444091797,
                    67.78710907838104
                ]
            ]
        ]
    ]
  }
}
```


Following, for each file uploaded, another small metadata must be filled:

12	**SAFERS resource metadata**			Metadata required by SAFERS
12.1		Datatype ID	datatype_id	SAFERS task in number of five digits, where the first is the SAFERS WP number, the second is the Task number and the following digits are an incremental number that

				specifies the data type. It can be a list if more files are uploaded
12.2		Name	name	Name of the resource
12.3		Format	format	File extension
12.4		Start validity	file_date_start	Start datetime of validity (ISO 8601 format)
12.5		End validity	file_date_end	End datetime of validity (ISO 8601 format)

4. ANNEX II

datatypeID	Group	Subgroup	Responsible	Data Description	Format	Update frequency
32101	environment	forecast	RISC	European Forest Fire Information System (EFFIS) - Fire Weather Index (FWI)	GeoJSON	daily
32005	environment	response	RISC	Get critical points of infrastructure, e.g. airports, motorways, hospitals, etc.	GeoJSON	on demand
37002	environment	response	RISC	Generate burn severity map (dNBR)	GeoTIFF	on demand
37003	environment	recovery	RISC	Generate soil recovery map (Vegetation Index)	GeoTIFF	on demand
37004	environment	response	RISC	Provide landslide susceptibility information	GeoTIFF	on demand
37005	environment	recovery	RISC	Generate historical severity map (dNBR)	GeoTIFF	on demand
37006	environment	recovery	RISC	Generate vegetation recovery map	GeoTIFF	on demand
36001	environment	recovery	LINKS	Burned area delineation map	GeoTIFF	on demand
36002	environment	recovery	LINKS	Burned area severity map	GeoTIFF	on demand
36003	environment	recovery	LINKS	Burned area geospatial image	GeoTIFF	on demand
36004	environment	response	LINKS	Fire front and smoke	GeoTIFF	on demand
36005	environment	response	LINKS	Impact quantification	GeoJSON	on demand
35007	environment	forecast	CIMA	fire perimeter simulation as isochrones maps	GeoJSON	on demand
35008	environment	forecast	CIMA	mean fireline intensity	GeoTIFF	on demand
35009	environment	forecast	CIMA	max fireline intensity	GeoTIFF	on demand

35010	environment	forecast	CIMA	mean rate of spread	GeoTIFF	on demand
35011	environment	forecast	CIMA	max rate of spread	GeoTIFF	on demand
33101	weather	short term forecast	FMI	temperature at 2m	NetCDF	6 hours
33103	weather	short term forecast	FMI	total precipitation	NetCDF	6 hours
33104	weather	short term forecast	FMI	relative humidity	NetCDF	6 hours
33105	weather	short term forecast	FMI	2m dewpoint temperature	NetCDF	6 hours
33110	weather	short term forecast	FMI	wind (speed and direction)	NetCDF	6 hours
33201	weather	subseasonal prediction	FMI	temperature at 2m height	NetCDF	12 hours
33202	weather	subseasonal prediction	FMI	relative humidity	NetCDF	12 hours
33203	weather	subseasonal prediction	FMI	2m dewpoint temperature	NetCDF	12 hours
33204	weather	subseasonal prediction	FMI	total precipitation	NetCDF	12 hours
33205	weather	subseasonal prediction	FMI	max gust	NetCDF	12 hours

33206	weather	subseasonal prediction	FMI	solar net radiation	NetCDF	12 hours
33207	weather	subseasonal prediction	FMI	Minimum temperature at 2 metres in the last 6 hours	NetCDF	12 hours
33208	weather	subseasonal prediction	FMI	Maximum temperature at 2 metres in the last 6 hours	NetCDF	12 hours
33209	weather	subseasonal prediction	FMI	Volumetric soil water layer 1	NetCDF	12 hours
33210	weather	subseasonal prediction	FMI	Volumetric soil water layer 2	NetCDF	12 hours
33230	weather	subseasonal prediction	FMI	wind (speed and direction)	NetCDF	12 hours
33211	weather	subseasonal prediction	FMI	temperature at 2m height - std dev	NetCDF	12 hours
33212	weather	subseasonal prediction	FMI	relative humidity - std dev	NetCDF	12 hours
33213	weather	subseasonal prediction	FMI	2m dewpoint temperature - std dev	NetCDF	12 hours
33214	weather	subseasonal prediction	FMI	total precipitation - std dev	NetCDF	12 hours
33215	weather	subseasonal prediction	FMI	max gust - std dev	NetCDF	12 hours

33216	weather	subseasonal prediction	FMI	solar net radiation - std dev	NetCDF	12 hours
33217	weather	subseasonal prediction	FMI	Minimum temperature at 2 metres in the last 6 hours - std dev	NetCDF	12 hours
33218	weather	subseasonal prediction	FMI	Maximum temperature at 2 metres in the last 6 hours - std dev	NetCDF	12 hours
33219	weather	subseasonal prediction	FMI	Volumetric soil water layer 1 - std dev	NetCDF	12 hours
33220	weather	subseasonal prediction	FMI	Volumetric soil water layer 2 - std dev	NetCDF	12 hours
33240	weather	subseasonal prediction	FMI	wind v10 - std dev	NetCDF	12 hours
33241	weather	subseasonal prediction	FMI	wind u10 - std dev	NetCDF	12 hours
33301	weather	seasonal prediction	FMI	temperature at 2m height	NetCDF	daily
33302	weather	seasonal prediction	FMI	relative humidity	NetCDF	daily
33303	weather	seasonal prediction	FMI	2m dewpoint temperature	NetCDF	daily
33304	weather	seasonal prediction	FMI	total precipitation	NetCDF	daily

33305	weather	seasonal prediction	FMI	max gust	NetCDF	daily
33306	weather	seasonal prediction	FMI	solar net radiation	NetCDF	daily
33307	weather	seasonal prediction	FMI	Volumetric soil water layer 1	NetCDF	daily
33308	weather	seasonal prediction	FMI	Volumetric soil water layer 2	NetCDF	daily
33320	weather	seasonal prediction	FMI	wind (speed and direction)	NetCDF	daily
33311	weather	seasonal prediction	FMI	temperature at 2m height - std dev	NetCDF	daily
33312	weather	seasonal prediction	FMI	relative humidity - std dev	NetCDF	daily
33313	weather	seasonal prediction	FMI	2m dewpoint temperature - std dev	NetCDF	daily
33314	weather	seasonal prediction	FMI	total precipitation - std dev	NetCDF	daily
33315	weather	seasonal prediction	FMI	max gust - std dev	NetCDF	daily
33316	weather	seasonal prediction	FMI	solar net radiation - std dev	NetCDF	daily

33317	weather	seasonal prediction	FMI	Volumetric soil water layer 1 - std dev	NetCDF	daily
33318	weather	seasonal prediction	FMI	Volumetric soil water layer 2 - std dev	NetCDF	daily
33330	weather	seasonal prediction	FMI	wind v10 - std dev	NetCDF	daily
33331	weather	seasonal prediction	FMI	wind u10 - std dev	NetCDF	daily