



Extreme Food Risk Analytics

D4.3: EFRA API Gateway and Data & Analytics Marketplace

(revised based on review recommendations)

Lead Author: Manos Karvounis (Agroknow)



**Funded by
the European Union**

Grant Agreement No.	101093026		
Project Acronym	EFRA		
Project Title	Extreme Food Risk Analytics		
Type of action	HORIZON Research and Innovation Actions		
Call Topic	HORIZON-CL4-2022-DATA-01-05		
Project Start Date	January 1 st , 2023	Project End Date	December 31 st , 2025
Project URL	efraproject.eu		
Work Package	WP4 EFRA Green Data & Analytics Infrastructure		
WP Lead Beneficiary	Agroknow		
Deliverable type ¹ Dissemination level ²	DEM PUB		
Contractual due date	31 December 2023	Actual submission date	December 28 th , 2023 (re-submission on September 20 th , 2024)
Lead Author (s)	Manos Karvounis (Agroknow)		
Contributors	George Marinos (Agroknow), Alessio Bosca (MAIZE), Jakub Janostik (SGS), Morgane Romeu (AGRIVI)		
Internal reviewer(s)	Francesco Lettich (CNR)		

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA) (granting authority). Neither the European Union nor the granting authority can be held responsible for them.

Copyright message

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

¹ R: Document, report; DEM: Demonstrator, pilot, prototype, plan designs; DEC: Websites, patents filing, press & media actions, videos, etc.; DATA: Data sets, microdata, etc.; DMP: Data management plan; ETHICS: Deliverables related to ethics issues; SECURITY: Deliverables related to security issues; OTHER: Software, technical diagram, algorithms, models, etc.

² PU – Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page); SEN – Sensitive, limited under the conditions of the Grant Agreement; Classified R-UE/EU-R – EU RESTRICTED under the Commission Decision No2015/444; Classified C-UE/EU-C – EU CONFIDENTIAL under the Commission Decision No2015/444; Classified S-UE/EU-S – EU SECRET under the Commission Decision No2015/444

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
V0.1	01/10/2023	10%	Initial Deliverable Structure	Manos Karvounis (Agroknow)
V0.2	21/10/2023	30%	First version of content in all sections	Manos Karvounis (Agroknow)
V0.3	14/11/2023	60%	Expanded content in all sections and contribution of APIs	Manos Karvounis (Agroknow), Alessio Bosca (MAIZE), Jakub Janostik (SGS), Morgane Romeu (AGRIVI)
V0.4	20/11/2023	90%	Added section updates	Manos Karvounis (Agroknow), George Marinos (Agroknow), Alessio Bosca (MAIZE), Jakub Janostik (SGS), Morgane Romeu (AGRIVI)
V0.9	23/11/2023	99%	Finalised delivered and prepared for review	Manos Karvounis (Agroknow) Babis Thanopoulos (Agroknow)
V1.0	15/12/2023	100%	Conducted review and finalised document	Manos Karvounis (Agroknow), Francesco Lettich (CNR)
V1.1	16/09/2024	90%	Updated content to address mid-term review revision requests. Specifically, the following were updated: <ul style="list-style-type: none"> • The section on the EFRA Marketplace was entirely replaced by new content. • Section 4.1 introduces the new EFRA Marketplace content. • Section 4.2 presents a current architectural overview. • Section 4.3 presents the key functionalities. • Section 4.4 presents detailed technical specifications, for the entities & relationships 	Manos Karvounis (Agroknow)

(4.4.1), for the front-end layer (4.4.2), and the middle-layer (4.4.3).

- Section 4.5 presents the current **EFRA Marketplace demonstrator with a pointer to it.**

V1.2	18/09/2024	100%	Internal Quality Control	Francesco Lettich (CNR)
V1.3	20/09/2024	100%	Final Revised Version	Manos Karvounis (Agroknow),

EFRA Consortium			
#	Participant Organisation Name	Short name	Country
1	AGROKNOW IKE	AGROKNOW	EL
2	CONSIGLIO NAZIONALE DELLE RICERCHE	CNR	IT
3	STOCKHOLMS UNIVERSITET	SU	SE
4	STICHTING WAGENINGEN RESEARCH	WR	NL
5	MAIZE SRL	MAIZE SRL	IT
6	AGRIVI DOO ZA PROIZVODNJU TRGOVINUI USLUGE	AGRIVI DOO	HR
7	RAINNO IDIOTIKI KEFALAIOUCHIKI ETAIREIA	RAINNO	EL
8	SGS ROMANIA SA	SGS ROMANIA	RO
9	MOY PARK LTD	MOY PARK	UK

Table of Contents

1	Executive Summary	8
2	Introduction	9
3	EFRA API Gateway.....	10
3.1	Integrated API Gateway	10
3.2	API Specifications for Food Risk Intelligence	10
3.3	API Specifications for Food Regulations	14
3.4	API Specifications for Agriculture.....	17
3.5	API Specifications for EFRA Analytics Powerhouse.....	39
	Overview	39
	APIs Specification.....	40
	Docker Registry APIs.....	40
	Powerhouse status API	43
4	EFRA Data & Analytics Marketplace	45
4.1	Introduction	45
4.2	Architecture Overview	46
4.3	Key Functionalities	47
4.4	Technical Specifications.....	48
4.4.1	Core Entities and Relationships	49
4.4.2	Front-End Layer.....	51
4.4.2.1	General screens for asset trading	51
4.4.2.2	Screens tied to the EFRA Platform.....	54
4.4.3	Middle-Layer	57
4.4.3.1	General Principles.....	57
4.4.3.2	User Controller	58
4.4.3.3	Dataset Management Controller	58
4.4.3.4	Dataset Search Controller.....	60
4.4.3.5	AI Model Management Controller.....	60
4.4.3.6	AI Model Search Controller	62
4.5	EFRA Marketplace Demonstrator	62
5	Conclusions	71

List of Tables

Table 1: Adherence to EFRA GA Deliverable & Tasks Descriptions	9
Table 2: User controller functionalities for the middle layer of the EFRA Marketplace	58
Table 3: Data Management controller functionalities for the middle layer of the EFRA Marketplace ..	58
Table 4: Dataset Search controller functionalities for the middle layer of the EFRA Marketplace.....	60
Table 5: AI Model Management controller functionalities for the middle layer of the EFRA Marketplace	60
Table 6: AI Model Search controller functionalities for the middle layer of the EFRA Marketplace	62

List of Figures

Figure 1: The EFRA API Gateway and underlying EFRA tools and data providers	10
Figure 2: The process for uploading and validating a model on the EFRA Powerhouse.....	40
Figure 3: High-level architecture of the EFRA Data & Analytics Marketplace	46
Figure 4: High-fidelity design for the Core Page Layout	63

Figure 5: High-fidelity design for the Asset Search (Datasets) page	64
Figure 6: High-fidelity design for the Asset Preview & Acquire (AI Model) page	65
Figure 7: High-fidelity design for My Assets page	66
Figure 8: High-fidelity design for the Asset Preview & Download (AI Model) page	67
Figure 9: High-fidelity design for the Publish Asset (AI Model) page	68
Figure 10: High-fidelity design for My Details page	69
Figure 11: High-fidelity design for Incident Forecasting Dashboard page	70

1 Executive Summary

This deliverable outlines the initial designs and specifications for two primary components of the EFRA Platform: the API Gateway and the Data & Analytics Marketplace. The API Gateway is engineered to provide third-party applications with access to datasets and AI models within the EFRA ecosystem. In contrast, the Data & Analytics Marketplace is conceptualized as a user-friendly web application, facilitating direct access to data and AI models for interested users.

The EFRA API Gateway functions as a central hub, streamlining the integration of diverse tools and data providers. This centralized system is pivotal for efficiently routing requests to the appropriate tools and interfaces based on their unique requirements. It also enables external applications to integrate seamlessly with EFRA's resources, broadening the scope of data and AI model accessibility and contribution.

The deliverable extensively covers specific API specifications. These include the Food Risk Intelligence APIs, which grant access to detailed food safety incident data and risk assessments, and the Food Regulations APIs, which provides regulatory content for the Food & Feed industries. The Agriculture section, powered by AGRIVI APIs, focuses on data critical for agricultural management, such as pest alarms and pesticide information.

A significant aspect of the EFRA Project is the Analytics Powerhouse. This segment is devoted to managing AI models, encompassing functionalities from uploading to execution and monitoring. AI models are required to be presented as docker images, undergoing a comprehensive validation process on the EFRA platform, thereby ensuring their functionality and effectiveness.

The EFRA Data & Analytics Marketplace, another integral component, is designed to streamline the acquisition, management, and usage of datasets related to food safety. This web application is tailored to be intuitive, enhancing data discoverability and facilitating easy navigation for users of varying expertise levels. It promotes transparent transactions and encourages users to participate in a dynamic marketplace for custom data requests, aiming to elevate the quality and practicality of data.

In summary, this deliverable highlights the EFRA Project's initial phase in developing an integrated system aimed at improving access and management of data and AI models in the food safety domain. It emphasizes the foundational role of the EFRA API Gateway and the Data & Analytics Marketplace in achieving this goal, marking a significant step towards enhancing global standards in food safety data management.

2 Introduction

The goal of this section is to provide a brief outline of the **objectives** of the at-hand EFRA Deliverable (outlined in [Table 1](#)) and how the rest of the document is structured to achieve them.

Table 1: Adherence to EFRA GA Deliverable & Tasks Descriptions

EFRA GA Component Title	EFRA GA Component Outline
Task 4.4 API Gateway and EFRA Data & Analytics Marketplace	This task will design the front-facing components of the EFRA ecosystem: i) an API gateway that will expose data and analytics through well-defined machine readable interfaces, ii) extend the original data marketplace concept of BigDataGrapes into an online marketplace of data and analytics for food risk prevention, iii) integrate data and analytics as discoverable assets within the marketplace, iv) implement functionalities to allow 3rd party apps and users easily discover, purchase, and access relevant resources to help them perform extreme food risk analytics.

This deliverable, a key component of the EFRA project, is structured to align with the objectives outlined in the EFRA GA (Grant Agreement) Deliverable and Task Descriptions, particularly focusing on Task 4.4. The primary aim of this document is to detail the development and structure of the front-facing components of the EFRA ecosystem.

The deliverable begins with an executive summary that provides an overview of the EFRA API Gateway and the Data & Analytics Marketplace, emphasizing their role and functionality within the EFRA ecosystem. Following this, the document delves into detailed descriptions and specifications of the API Gateway, illustrating how it serves as a centralized hub for integrating various tools and data providers. This section also details how the gateway enables third-party applications to access and contribute to EFRA's data and AI models.

Next, the deliverable explores the Data & Analytics Marketplace, describing its design as a user-friendly web application. This section highlights how the marketplace extends the data marketplace concept and integrates data and analytics as discoverable assets, making them readily available for users and third-party applications. The functionalities that facilitate the discovery, purchase, and access of these resources are also detailed.

In summary, this deliverable aligns with the objectives of Task 4.4 of the EFRA GA, detailing the development of the API Gateway and the Data & Analytics Marketplace. It serves as a guide to the initial designs and specifications of these components, outlining their roles in enhancing accessibility, integration, and management of data and AI models within the EFRA ecosystem for effective food risk analytics.

3 EFRA API Gateway

3.1 Integrated API Gateway

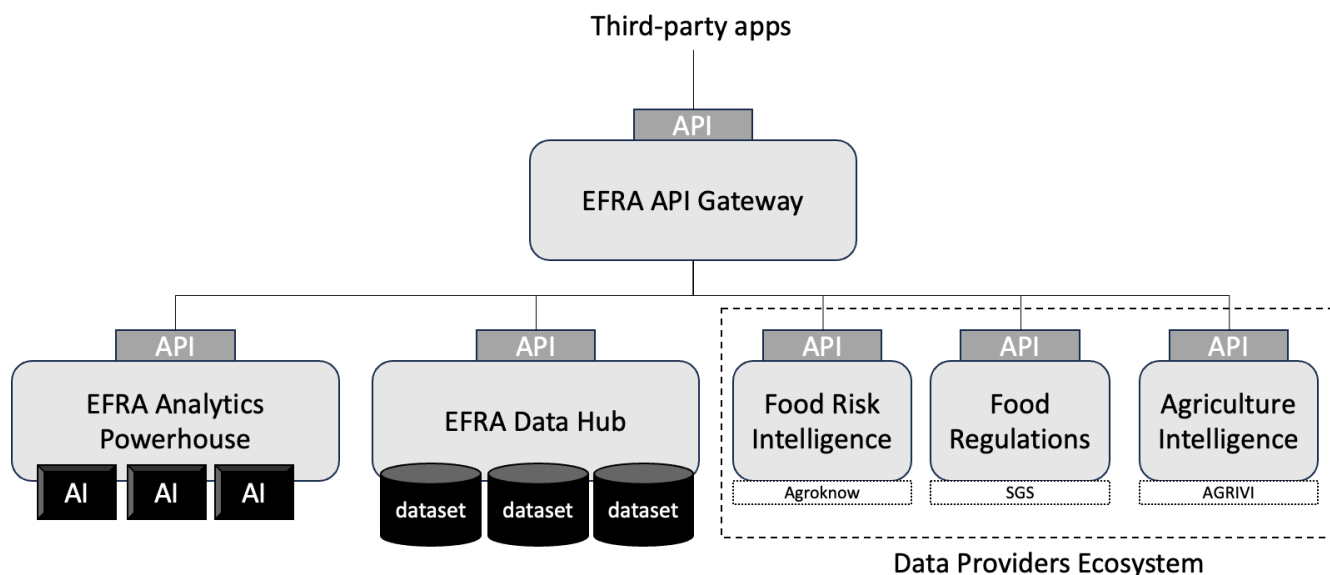


Figure 1: The EFRA API Gateway and underlying EFRA tools and data providers

The EFRA API Gateway serves as a centralized hub, integrating the diverse tools and data providers within the EFRA ecosystem. Each distinct module within this ecosystem features its own dedicated API, allowing for specialized interactions. The role of the API Gateway is crucial as it routes requests to the most suitable underlying tools, determined by the nature of each request (see Figure 1).

A significant advantage of the EFRA API Gateway is its facilitation of external third-party application integration. Through this gateway, these applications gain access to EFRA's extensive data assets and its suite of trained AI models. Moreover, there is provision for these third-party entities to contribute to the EFRA ecosystem by registering their own AI models and data assets, thus enriching the system's capabilities and diversity.

The development of the API Gateway is scheduled for a later phase in the project. This timing ensures it can be effectively tailored to accommodate all the modules and tools that constitute the EFRA ecosystem. The progress and iterative enhancements of the API Gateway will be documented in this deliverable, showcasing its evolution throughout the project lifecycle.

The subsequent sections of this document provide an in-depth examination of the individual modules within the EFRA ecosystem. Each section will detail the specific APIs offered by these modules, elucidating their functionalities and the unique value they add to the EFRA ecosystem. This overview aims to provide a clear understanding of how each module functions within the broader context of the EFRA framework and the unification achieved through the API Gateway.

3.2 API Specifications for Food Risk Intelligence

Overview

This section describes the part of the EFRA APIs powered by the Food Risk Intelligence Data Platform. It allows access to two APIs and related underlying datasets: (a) the Data API, which provides access to all food safety intelligence data provided by EFRA, including food safety incident data and lab tests, and the (b) the Risks API, a

dedicated API that provides access to top risks calculated for various combinations of food companies, countries, and products.

Entity Types

In Riskadata there are various **entityType** one can use to search for the specific data they need. You can find in the table below a list with the most notable entityType present.

You can use the values under the column **EntityType** directly in your search queries.

EntityType	Description
labtest	Results of monitoring programs of Food Safety Authorities worldwide
country_indicator	Country indicators like corruption, risk, trade & production
prediction_trend	Forecasting of incidents for products/hazards/countries of origin
supplier	Supplier/Food Company data
incident	Food recalls & border rejections announced worldwide
prediction_risk	Forecasting of risk value for products/hazards/countries of origin using FOODAKAI's formula
inspection	Inspection data on the premises of food companies & their plants/locations
brand	Product brands involved in recalls around the world
product	FOODAKAI's product vocabulary
hazard	FOODAKAI's hazard vocabulary
fraud_case	Fraud cases for food safety incidents announced by news websites
emerging_risk	New/emerging hazards happening for products for the first time

Authentication

Our Data API uses as authentication mechanism **apikey**. Each request send to any of the endpoints below, should have the **apikey** property set.

Data API

Endpoint: <https://api.foodakai.com/search-api-1.0/search/>

Request Type: POST

Parameter	Data Type	Format Example
aggregations	Object	{"property1": {}, "property2": {}}
apikey	String	"string"
boolQuery	Object	{"property1": true, "property2": true}
context	String	"string"

detail	Boolean	true
entityType	String	"string"
existenceQuery	Array	["string"]
expand	Boolean	true
freetext	String	"string"
from	String (Date)	"yyyy-MM-dd"
fuzzinessScore	Integer	0
fuzzyQuery	Object	{"property1": "string", "property2": "string"}
hazard	String	"string"
highlight	Boolean	true
id	String	"string"
keywords	String	"string"
language	String	"string"
location	String	"string"
numericalQueries	Array	[{}]
page	Integer	0
pageSize	Integer	0
plainQuery	Object	{"property1": "string", "property2": "string"}
product	String	"string"
published	Boolean	true
publishedBy	String	"string"
requestedOn	String	"string"
smart	Boolean	true
sortOn	Object	{"property1": "string", "property2": "string"}
sourceInclude	Array	["string"]
strictQuery	Object	{"property1": "string", "property2": "string"}
to	String (Date)	"yyyy-MM-dd"
updatedOnFrom	String (Date)	"yyyy-MM-dd"
updatedOnTo	String (Date)	"yyyy-MM-dd"
wildcardQuery	Object	{"property1": "string", "property2": "string"}

Risks API

Endpoint: <https://api.foodakai.com/search-api-1.0/risk/search-beta>

Request type: POST

Parameter	Data Type	Format Example
apikey	String	"string"
company	String	"string"
country	String	"string"
data	Boolean	true
fromDate	String	"string"
origin	String	"string"
pastYears	Integer	0
product	String	"string"
productCategory	Boolean	true
severity	Boolean	true
toDate	String	"string"
top	Integer	0

Examples

DATA API

```
# get all border rejections of 2016
request = {
  "apikey": APIKEY,
  "entityType": "incident",
  "strictQuery": {
    "notificationType": "border rejection",
    "detail": "true",
    "from": "2016-01-01",
    "to": "2016-01-31",
    "pageSize": 50,
    "page": 0
  }
}
```

```
# get all incidents for confectionery originating from Mexico
request = {
  "apikey": APIKEY,
  "entityType": "incident",
  "detail": True,
  "strictQuery": {
    "products.value": "confectionery",
    "origin.value": "mexico",
  },
  "from": "2020-01-01",
}
```

```
# distribution of incidents & inspections on supplier
request = {
  "apikey": APIKEY,
  "entityType": "incident|inspection",
  "detail": True,
  "freetext": "Cargill",
}
```

```
# which are the countries that have exported more than 10 tonnes of groundnuts and to which countries
request = {
  "apikey": APIKEY,
  "entityType": "country_indicator",
  "detail": True,
  "strictQuery": {
}
```

```

    "smart": True,
    "page": 0,
    "pageSize": 0,
    "aggregations": {
      "dates": {
        "attribute":
"createdOn",
        "interval": "YEAR",
        "format": "YYYY"
      }
    }
  }
}

```

```

    "tags": "export quantity",
    "information.product": "groundnuts,
shelled",
  },
  "numericalQueries": [{
    "attribute":
"information.indicator_value",
    "operator": ">=",
    "value": 10,
  }],
  "from": "2016-01-01",
  "pageSize": 0,
  "aggregations": {
    "countries": {
      "attribute":
"information.country.keyword",
      "size": 1000,
      "subAggregation": {
        "attribute":
"information.target_country.keyword",
        "size": 100,
      }
    }
  }
}

```

RISKS API

```

# get top 5 risks for nuts or
fish products
request = {
  "apikey": APIKEY,
  "product": "nuts, nut
products and seeds|fish and fish
products",
  "pastYears": 7,
  "top": 5
}

```

```

# get top 5 risks for nuts or fish products
originating from either France or the United
States
request = {
  "apikey": APIKEY,
  "product": "nuts, nut products and
seeds|fish and fish products",
  "country": "france|united states",
  "pastYears": 7,
  "top": 5
}

```

```

# get Cargill's risk assessment
including latest data
request = {
  "apikey": APIKEY,
  "company": "Cargill",
  "country": "",
  "origin": "",
  "pastYears": 7,
  "product": "",
  "productCategory": False,
  "severity": False,
  "top": 1,
  "data": True,
}

```

3.3 API Specifications for Food Regulations

Overview

This section describes the part of the SGS Digicomply APIs. It allows access to underlying dataset of regulatory content collected by SGS Digicomply platform. This dataset covers the global regulatory landscape for Food & Feed industries.

Entity Types

Main entity for the API is Post. This represents a unique article in the dataset. Each post consists of fields described in the table below.

Field	Description
post_id	Unique id of the post.
dc_url	URL to access the post within DC platform.
origin_url	URL of the content found on the web. Can be null for manual uploads.
published_on	Date the content was published on the original website. If not available, it defaults to "created_on".
created_on	Date the article was collected in the system.
source_id	Unique identifier of the website we collected the article from.
source_name	Name of the website we collected the article from.
source_url	Website of the website we collected the article from.
authority	Name of the specific regulatory authority (if applicable).
english_title	Title of the article in english. Same as "original_title" if the original article was in english.
english_content_url	Google storage url to access translated text of the article. Same as "original_content_url" if the original article was in english.
original_title	Title of the article in original language.
original_content_url	Google storage url to access raw text of the article.
types	Type of content. e.g. regulatory article, web article, scientific paper, ...
topics	Topic of the article (based on the text classifier).
original_language	Language key of the original language of the article.
markets	Market of the website the article was collected from. Can contain custom market keys for national organizations, e.g. EU.
product_categories	Mentioned product category (based on text classifier).

substances	Substances mentioned in text (NER).
relations	Related posts (e.g. repeals, amendments, comments, ...).

Post Changes API

Post Changes is a simple API allowing consumers fetch delta of the post content since a specified timestamp. Endpoints are secured and require apikey header to be present.

Common errors

Error	Description
410	Pagination session expired and no longer valid
404	Pagination session token does not exist
403	Resource forbidden. Ensure that headers contain apikey

Standard HTTP codes apply for rate limiting and other common errors.

Request

To fetch data from the API you need to specify following:

- Query: Name of the query which identifies subset of data within our dataset. Must be created before we set the API key.
- Since: Date of the oldest post you're interested in

POST <https://post-changes.prod.c-labs.ch/>

api-key:your-api-key

Content-Type:application/json

```
{
  "query": "ALL_POSTS",
  "since": "2000-01-01T00:00:00"
}
```

As a response you'll receive following:

```
{
  "data": [
    {
      "postId": "...",
      "dcUrl": "https://digicomply.sgs.com/app/p/...",
      "originUrl": "http://...",
      ...
    },
  ],
}
```



```

    ...
  ],
  "hasNext": true,
  "session": "dec02625-26fa-4a3d-a5b5-aa00f6166022",
  "totalSize": 100
}

```

Important part of this request is the session key which can be used to fetch subsequent pages from the response.

POST <https://post-changes.prod.c-labs.ch/>

api-key:your-api-key

Content-Type:application/json

```

{
  "nextForSession": "dec02625-26fa-4a3d-a5b5-aa00f6166022"
}

```

3.4 API Specifications for Agriculture

Overview

This section describes the part of the AGRIVI APIs. It allows access to underlying agriculture datasets collected by the AGRIVI platform. This dataset covers pest alarms, pest and pesticides information, and relevant contextual factors.

Swagger documentation for the API can be found at the following link:

<https://mobile-2-5.agrivi.com/swagger/index.html>

Authentication within the system is achieved through the utilization of OAuth refresh and access tokens. Additional details about the authentication process can be found at: <https://mobile-2-5.agrivi.com/swagger/index.html?urls.primaryName=V1-PESTS#Auth>

PEST ALARM

Endpoint: <https://mobile-2-5.agrivi.com/PestAlarms>

Type: GET

Description: Return all pest reminders for current user (def. ordering: Name ASC)

Parameter	Data Type	Description
Limit	integer(\$int32)	Number of records to return
Offset	integer(\$int32)	Number of records to offset
NeedPaging	boolean	Enable paging
DatabasePaging	boolean	Result paged server
OrderByField	string	Name of the field used for ordering
OrderByDirection	string	Ascending or Descending (asc or dsc)
FacilityID	integer(\$int64)	Crop production ID
Dismissed	Boolean	Does the result include the dismissed alarm

Seen	boolean	Does the result include the seen alarm
CultureID	integer(\$int64)	Crop ID
PestID	integer(\$int64)	Pest ID
AlarmStartDate	string(\$date-time)	Start date
AlarmEndDate	string(\$date-time)	End date
Timestamp	string(\$date-time)	Last synchronisation date
IncludeWeather	boolean	Include weather data
v	string	Fix to value 1

Returns:

Code 200 Success :

```
{
  {
    "apiVersion": "string",
    "context": "string",
    "id": "string",
    "method": "string",
    "params": {},
    "data": {
      "kind": "string",
      "fields": "string",
      "etag": "string",
      "id": "string",
      "lang": "string",
      "updated": "2023-11-10T10:22:08.931Z",
      "deleted": true,
      "currentItemCount": 0,
      "itemsPerPage": 0,
      "startIndex": 0,
      "totalItems": 0,
      "pageIndex": 0,
      "totalPages": 0,
      "pagingLinkTemplate": "string",
      "selfLink": "string",
      "editLink": "string",
      "nextLink": "string",
      "previousLink": "string",
      "items": [
        {
          "pestID": 0,
          "pestName": "string",
          "cropID": 0,
          "cropName": "string",
          "fields": [
            {
              "fieldID": 0,
              "fieldName": "string",
              "fieldLatidute": 0,
```

```

        "fieldLongitude": 0,
        "alarmDateFrom": "2023-11-10T10:22:08.931Z",
        "alarmDateTo": "2023-11-10T10:22:08.931Z"
    }
],
"weatherConditions": [
    {
        "latitude": 0,
        "longitude": 0,
        "dateMeasured": "2023-11-10T10:22:08.931Z",
        "summary": "string",
        "dateStringWithYear": "string",
        "temperatureMin": 0,
        "temperatureMinConverted": 0,
        "tempMin": "string",
        "temperatureMax": 0,
        "temperatureMaxConverted": 0,
        "temperature": 0,
        "temperatureConverted": 0,
        "temperatureString": "string",
        "tempMax": "string",
        "dewPoint": 0,
        "dewPointConverted": 0,
        "dewPointString": "string",
        "devTemp": "string",
        "tempUnit": "string",
        "humidity": 0,
        "humidityString": "string",
        "pressure": 0,
        "pressureConverted": 0,
        "pressureString": "string",
        "precipIntensity": 0,
        "precipIntensityConverted": 0,
        "precipIntensityString": "string",
        "precip": "string",
        "precipType": "string",
        "precipProbability": 0,
        "precipProbabilityString": "string",
        "windSpeed": 0,
        "windSpeedConverted": 0,
        "windSpeedString": "string",
        "windBearing": 0,
        "windBearingString": "string",
        "wind": "string",
        "iconName": "string",
        "icon": "string",
        "frostPossible": "string",
        "weatherIcon": "string",
        "date": "2023-11-10T10:22:08.931Z",
        "dateString": "string",
        "day": "string"
    }
],

```

```

        "alarmSeverityLevel": 0
      }
    ]
  },
  "error": {
    "code": 0,
    "message": "string",
    "errors": [
      {
        "domain": "string",
        "reason": "string",
        "message": "string",
        "location": "string",
        "locationType": "string",
        "extendedHelp": "string",
        "sendReport": "string"
      }
    ]
  }
}

```

Endpoint: <https://mobile-2-5.agrivi.com/PestAlarms/pestAlarms/{id}>

Type: GET

Description: Return single reminder by ID

Parameter	Data Type	Description
id	integer(\$int64)	Pest alarm ID
v	string	Fix to value 1
includeWeather	boolean	Include weather data

Returns:

Code 200 Success

```

{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
    "id": "string",
    "lang": "string",
    "updated": "2023-11-10T10:55:28.900Z",
    "deleted": true,
    "currentItemCount": 0,
    "itemsPerPage": 0,
  }
}

```

```

"startIndex": 0,
"totalItems": 0,
"pageIndex": 0,
"totalPages": 0,
"pagingLinkTemplate": "string",
"selfLink": "string",
"editLink": "string",
"nextLink": "string",
"previousLink": "string",
"items": [
  {
    "pestID": 0,
    "pestName": "string",
    "cropID": 0,
    "cropName": "string",
    "fields": [
      {
        "fieldID": 0,
        "fieldName": "string",
        "fieldLatitude": 0,
        "fieldLongitude": 0,
        "alarmDateFrom": "2023-11-10T10:55:28.900Z",
        "alarmDateTo": "2023-11-10T10:55:28.900Z"
      }
    ],
    "weatherConditions": [
      {
        "latitude": 0,
        "longitude": 0,
        "dateMeasured": "2023-11-10T10:55:28.900Z",
        "summary": "string",
        "dateStringWithYear": "string",
        "temperatureMin": 0,
        "temperatureMinConverted": 0,
        "tempMin": "string",
        "temperatureMax": 0,
        "temperatureMaxConverted": 0,
        "temperature": 0,
        "temperatureConverted": 0,
        "temperatureString": "string",
        "tempMax": "string",
        "dewPoint": 0,
        "dewPointConverted": 0,
        "dewPointString": "string",
        "devTemp": "string",
        "tempUnit": "string",
        "humidity": 0,
        "humidityString": "string",
        "pressure": 0,
        "pressureConverted": 0,
        "pressureString": "string",
        "precipIntensity": 0,
        "precipIntensityConverted": 0,

```

```

        "precipIntensityString": "string",
        "precip": "string",
        "precipType": "string",
        "precipProbability": 0,
        "precipProbabilityString": "string",
        "windSpeed": 0,
        "windSpeedConverted": 0,
        "windSpeedString": "string",
        "windBearing": 0,
        "windBearingString": "string",
        "wind": "string",
        "iconName": "string",
        "icon": "string",
        "frostPossible": "string",
        "weatherIcon": "string",
        "date": "2023-11-10T10:55:28.901Z",
        "dateString": "string",
        "day": "string"
    }
],
    "alarmSeverityLevel": 0
}
]
},
"error": {
    "code": 0,
    "message": "string",
    "errors": [
        {
            "domain": "string",
            "reason": "string",
            "message": "string",
            "location": "string",
            "locationType": "string",
            "extendedHelp": "string",
            "sendReport": "string"
        }
    ]
}
}
}

```

Endpoint: <https://mobile-2-5.agrivi.com/Pests>

Type: GET

Description: Return pests for culture

Parameter	Data Type	Description
Limit	integer(\$int32)	Number of records to return
Offset	integer(\$int32)	Number of records to offset

NeedPaging	boolean	Enable paging
DatabasePaging	boolean	Result paged server
OrderByField	string	Name of the field used for ordering
OrderByDirection	string	Ascending or Descending (asc or dsc)
CultureID	integer(\$int64)	Crop ID
Locale	string	Language (local names)
Timestamp	string(\$date-time)	Last synchronisation date
IncludeCulture	boolean	Include crop data
IncludeActiveSubstance	boolean	Include active substances
v	string	Fix to value 1

Returns:

Code 200 Success

```
{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
    "id": "string",
    "lang": "string",
    "updated": "2023-11-10T11:02:41.869Z",
    "deleted": true,
    "currentItemCount": 0,
    "itemsPerPage": 0,
    "startIndex": 0,
    "totalItems": 0,
    "pageIndex": 0,
    "totalPages": 0,
    "pagingLinkTemplate": "string",
    "selfLink": "string",
    "editLink": "string",
    "nextLink": "string",
    "previousLink": "string",
    "items": [
      {
        "id": 0,
        "name": "string",
        "latinName": "string",
        "typeID": "Disease",
        "type": "string",

```

```

        "description": "string",
        "activeSubstancesString": "string",
        "crops": [
            {
                "id": 0,
                "name": "string"
            }
        ],
        "operations": 0
    }
]
},
"error": {
    "code": 0,
    "message": "string",
    "errors": [
        {
            "domain": "string",
            "reason": "string",
            "message": "string",
            "location": "string",
            "locationType": "string",
            "extendedHelp": "string",
            "sendReport": "string"
        }
    ]
}
}
}

```

Endpoint: <https://mobile-2-5.agrivi.com/Pests/{id}>

Type: GET

Description: Return single pest by ID

Parameters	Data Type	Description
id	integer(\$int64)	ID of pest
IncludeCulture	boolean	Include crop data
IncludeActiveSubstance	boolean	Include Active substances
CultureID	integer(\$int64)	Crop ID
Locale	string	Language of the data
Timestamp	string(\$date-time)	Last synchronisation date
v	string	Fix to value 1

Returns:

Code 200 Success


```

{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
    "id": "string",
    "lang": "string",
    "updated": "2023-11-10T11:06:32.217Z",
    "deleted": true,
    "currentItemCount": 0,
    "itemsPerPage": 0,
    "startIndex": 0,
    "totalItems": 0,
    "pageIndex": 0,
    "totalPages": 0,
    "pagingLinkTemplate": "string",
    "selfLink": "string",
    "editLink": "string",
    "nextLink": "string",
    "previousLink": "string",
    "items": [
      {
        "id": 0,
        "name": "string",
        "latinName": "string",
        "typeID": "Disease",
        "type": "string",
        "description": "string",
        "activeSubstancesString": "string",
        "crops": [
          {
            "id": 0,
            "name": "string"
          }
        ],
        "operations": 0
      }
    ]
  },
  "error": {
    "code": 0,
    "message": "string",
    "errors": [
      {
        "domain": "string",
        "reason": "string",
        "message": "string",
        "location": "string",
        "locationType": "string",
        "extendedHelp": "string",
        "sendReport": "string"
      }
    ]
  }
}

```

Endpoint: <https://mobile-2-5.agrivi.com/WeatherDaily>

Type: GET

Description: Return daily weather data for a specific field (def. ordering: Date DESC)

Parameters	Data Type	Description
Limit	integer(\$int32)	Number of records to return
Offset	integer(\$int32)	Number of records to offset
NeedPaging	boolean	Enable paging
DatabasePaging	boolean	Result paged server
OrderByField	string	Name of the field used for ordering
OrderByDirection	string	Ascending or Descending (asc or dsc)
FieldID	integer(\$int64)	ID of the field
DateFrom	string(\$date-time)	Data from this date
DateTo	string(\$date-time)	Data up to this date
Timestamp	string(\$date-time)	Last synchronisation date
Includes	object	
v	string	Fix to value 1

Returns:

Code 200 Success

```
{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
    "id": "string",
    "lang": "string",
    "updated": "2023-11-10T11:10:13.474Z",
    "deleted": true,
    "currentItemCount": 0,
    "itemsPerPage": 0,
    "startIndex": 0,
    "totalItems": 0,
    "pageIndex": 0,
    "totalPages": 0,
    "pagingLinkTemplate": "string",
    "selfLink": "string",
    "editLink": "string",
    "nextLink": "string",
    "previousLink": "string",
    "items": [
      {
        "latitude": 0,
        "longitude": 0,
        "dateMeasured": "2023-11-10T11:10:13.474Z",
        "summary": "string",
        "dateStringWithYear": "string",
```

```

        "temperatureMin": 0,
        "temperatureMinConverted": 0,
        "tempMin": "string",
        "temperatureMax": 0,
        "temperatureMaxConverted": 0,
        "temperature": 0,
        "temperatureConverted": 0,
        "temperatureString": "string",
        "tempMax": "string",
        "dewPoint": 0,
        "dewPointConverted": 0,
        "dewPointString": "string",
        "devTemp": "string",
        "tempUnit": "string",
        "humidity": 0,
        "humidityString": "string",
        "pressure": 0,
        "pressureConverted": 0,
        "pressureString": "string",
        "precipIntensity": 0,
        "precipIntensityConverted": 0,
        "precipIntensityString": "string",
        "precip": "string",
        "precipType": "string",
        "precipProbability": 0,
        "precipProbabilityString": "string",
        "windSpeed": 0,
        "windSpeedConverted": 0,
        "windSpeedString": "string",
        "windBearing": 0,
        "windBearingString": "string",
        "wind": "string",
        "iconName": "string",
        "icon": "string",
        "frostPossible": "string",
        "weatherIcon": "string",
        "date": "2023-11-10T11:10:13.474Z",
        "dateString": "string",
        "day": "string"
    }
  ],
  },
  "error": {
    "code": 0,
    "message": "string",
    "errors": [
      {
        "domain": "string",
        "reason": "string",
        "message": "string",
        "location": "string",
        "locationType": "string",
        "extendedHelp": "string",
        "sendReport": "string"
      }
    ]
  }
}

```

Endpoint: <https://mobile-2-5.agrivi.com/WeatherHourly>

Type: GET

Description: Return hourly weather data for a specific field (def. ordering: Time DESC)

Parameters	Data Type	Description
Limit	integer(\$int32)	Number of records to return
Offset	integer(\$int32)	Number of records to offset
NeedPaging	boolean	Enable paging
DatabasePaging	boolean	Result paged server
OrderByField	string	Name of the field used for ordering
OrderByDirection	string	Ascending or Descending (asc or dsc)
FieldID	integer(\$int64)	Field ID
DateFrom	string(\$date-time)	Data from this date
DateTo	string(\$date-time)	Data up to this date
Timestamp	string(\$date-time)	Last synchronisation date
Includes	object	
v	string	Fix to value 1

Returns:

Code 200 Success

```

"apiVersion": "string",
"context": "string",
"id": "string",
"method": "string",
"params": {},
"data": {
  "kind": "string",
  "fields": "string",
  "etag": "string",
  "id": "string",
  "lang": "string",
  "updated": "2023-11-10T11:17:37.153Z",
  "deleted": true,
  "currentItemCount": 0,
  "itemsPerPage": 0,
  "startIndex": 0,
  "totalItems": 0,
  "pageIndex": 0,
  "totalPages": 0,
  "pagingLinkTemplate": "string",
  "selfLink": "string",
  "editLink": "string",
  "nextLink": "string",
  "previousLink": "string",
  "items": [
    {
      "id": 0,
      "latitude": 0,
      "longitude": 0,
      "timeMeasured": "2023-11-10T11:17:37.153Z",
      "dateStringWithYear": "string",
      "temperature": 0,
      "temperatureConverted": 0,

```

```

        "temperatureString": "string",
        "dewTemperature": 0,
        "dewTemperatureString": "string",
        "temperatureUnitName": "string",
        "humidity": 0,
        "humidityString": "string",
        "pressure": 0,
        "pressureString": "string",
        "precipType": "string",
        "precipitationIntensity": 0,
        "precipitationIntensityConverted": 0,
        "precipitationIntensityString": "string",
        "precipitationIntensityUnitName": "string",
        "precipitationProbability": 0,
        "precipitationProbabilityString": "string",
        "windSpeed": 0,
        "windSpeedString": "string",
        "windSpeedUnitName": "string",
        "windBearing": 0,
        "windBearingString": "string",
        "windInfo": "string",
        "iconName": "string",
        "weatherIcon": "string",
        "fieldID": 0,
        "frostPossiblity": 0,
        "frostPossible": true,
        "cloudCoverCode": 0,
        "time": "2023-11-10T11:17:37.153Z",
        "dateString": "string",
        "day": "string"
    }
}
],
},
"error": {
    "code": 0,
    "message": "string",
    "errors": [
        {
            "domain": "string",
            "reason": "string",
            "message": "string",
            "location": "string",
            "locationType": "string",
            "extendedHelp": "string",
            "sendReport": "string"
        }
    ]
}
}
}

```

Endpoint: <https://mobile-2-5.agrivi.com/Fields>

Type: GET

Description: Return all fields for current company new (def. ordering: Name ASC)

Parameters	Data Type	Description
Limit	integer(\$int32)	Number of records to return
Offset	integer(\$int32)	Number of records to offset

NeedPaging	boolean	Enable paging
DatabasePaging	boolean	Result paged server
OrderByField	string	Name of the field used for ordering
OrderByDirection	string	Ascending or Descending (asc or dsc)
FacilityID	integer(\$int64)	Crop production ID
OnlyActiveFields	boolean	Only active fields
ResourcesStatus	string	
Timestamp	string(\$date-time)	Last synchronisation date
IncludeWeather	boolean	Include weather data
IncludeFacilities	boolean	Facilities = crop productions
IncludeLinkedFields	boolean	Fields linked to crop production
v	string	Fix to value 1

Returns:

Code 200 Success

```
{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
    "id": "string",
    "lang": "string",
    "updated": "2023-11-10T11:28:06.252Z",
    "deleted": true,
    "currentItemCount": 0,
    "itemsPerPage": 0,
    "startIndex": 0,
    "totalItems": 0,
    "pageIndex": 0,
    "totalPages": 0,
    "pagingLinkTemplate": "string",
    "selfLink": "string",
    "editLink": "string",
    "nextLink": "string",
    "previousLink": "string",
    "items": [
      {
        "surfaceConvertedString": "string",
        "surfaceConvertedBigUnitString": "string",
        "tempMinimum": 0,
        "tempMaximum": 0,
        "company": {
          "id": 0,
          "title": "string",

```

```

"ownerID": 0,
"countryID": "string",
"currencyID": "string",
"currencyName": "string",
"unitSystemID": "string",
"languageID": "string",
"languageName": "string",
"email": "string",
"phone": "string",
"foundationDate": "2023-11-10T11:28:06.252Z",
"dateAdded": "2023-11-10T11:28:06.252Z",
"partnerCode": "string",
"city": "string",
"address": "string",
"postalCode": "string",
"oib": "string",
"mb": "string",
"mbgp": "string",
"sic": "string",
"bankInfo": "string",
"bankAccount": "string",
"taxRate": 0,
"taxID": "string",
"slug": "string",
"logoID": 0,
"logoFull": "string",
"description": "string",
"website": "string",
"showPublicCertificates": true,
"isDemoCompany": true,
"roleOccupied": true,
"headerColorHex": "string",
"isCooperationManager": true,
"ndviAvailable": true,
"hasCooperations": true,
"sprayingAdvisorAvailable": true
},
"id": 0,
"name": "string",
"city": "string",
"agriculturalID": "string",
"cadastralPlotNumber": "string",
"mapUrl": "string",
"latitude": 0,
"longitude": 0,
"surface": 0,
"totalSurface": 0,
"surfaceUnitName": "string",
"surfaceUnitID": 0,
"surfaceString": "string",
"surfaceBase": 0,
"surfaceBaseUnitID": 0,
"surfaceBaseUnitName": "string",
"linkedSurfaceCurrent": 0,
"linkedSurfaceUnitID": 0,
"linkedSurfaceUnitName": "string",
"linkedSurfaceCurrentString": "string",
"linkedSurfaceCurrentBase": 0,
"linkedSurfaceBase": 0,
"linkedSurfaceBaseUnitID": 0,
"linkedSurfaceBaseUnitName": "string",
"utilization": 0,
"facilityIDs": "string",
"facilityNames": "string",
"soilTypeID": 0,

```

```

"soilTypeName": "string",
"ownershipTypeID": 0,
"ownershipTypeName": "string",
"dateDeactivated": "2023-11-10T11:28:06.252Z",
"coordinates": "string",
"coordinateType": "String",
"tempMin": "string",
"coordinatesSource": "string",
"coordinatesArray": [
  {
    "latitude": "string",
    "longitude": "string"
  }
],
"tempMax": "string",
"tempUnit": "string",
"weatherIcon": "string",
"iconName": "string",
"linkedFields": [
  {
    "id": 0,
    "dateLinked": "2023-11-10T11:28:06.252Z",
    "dateUnlinked": "2023-11-10T11:28:06.252Z",
    "facilityID": 0,
    "facilityName": "string",
    "fieldID": 0,
    "fieldName": "string",
    "linkedSurface": 0,
    "linkedSurfaceUnitID": 0,
    "linkedSurfaceUnitName": "string",
    "linkedSurfaceString": "string",
    "linkedSurfaceBase": 0,
    "linkedSurfaceBaseUnitID": 0,
    "linkedSurfaceBaseUnitName": "string",
    "mapUrl": "string",
    "city": "string",
    "utilization": 0,
    "myParcels": [
      {
        "id": 0,
        "coordinates": "string",
        "wktFormat": "string",
        "plantationFieldID": 0,
        "plantationName": "string",
        "plantationDateLinked": "2023-11-10T11:28:06.252Z",
        "plantationDateUnlinked": "2023-11-10T11:28:06.252Z"
      }
    ],
    "operations": 0
  }
],
"operations": 0
}
],
},
"error": {
  "code": 0,
  "message": "string",
  "errors": [
    {
      "domain": "string",
      "reason": "string",
      "message": "string",
      "location": "string",
      "locationType": "string",

```



```

        "extendedHelp": "string",
        "sendReport": "string"
    }
}
}
}

```

Endpoint: <https://mobile-2-5.agrivi.com/Fields/{id}>

Type: GET

Description: Return single field by ID

Parameters	Data Type	Description
id	integer(\$int64)	ID of field
IncludeWeather	boolean	Include weather data
IncludeFacilities	boolean	Include crop production
IncludeLinkedFields	boolean	Fields linked
FacilityID	integer(\$int64)	Crop production ID
OnlyActiveFields	boolean	Active field only
ResourcesStatus	string	
Timestamp	string(\$date-time)	Last synchronisation date
V	string	Fix to value 1

Returns:

Code 200 Success

```

{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
    "id": "string",
    "lang": "string",
    "updated": "2023-11-10T12:05:46.676Z",
    "deleted": true,
    "currentItemCount": 0,
    "itemsPerPage": 0,
    "startIndex": 0,
    "totalItems": 0,
    "pageIndex": 0,
    "totalPages": 0,
    "pagingLinkTemplate": "string",
    "selfLink": "string",
    "editLink": "string",
    "nextLink": "string",
    "previousLink": "string",
    "items": [
      {
        "surfaceConvertedString": "string",

```

```

"surfaceConvertedBigUnitString": "string",
"tempMinimum": 0,
"tempMaximum": 0,
"company": {
  "id": 0,
  "title": "string",
  "ownerID": 0,
  "countryID": "string",
  "currencyID": "string",
  "currencyName": "string",
  "unitSystemID": "string",
  "languageID": "string",
  "languageName": "string",
  "email": "string",
  "phone": "string",
  "foundationDate": "2023-11-10T12:05:46.676Z",
  "dateAdded": "2023-11-10T12:05:46.676Z",
  "partnerCode": "string",
  "city": "string",
  "address": "string",
  "postalCode": "string",
  "oib": "string",
  "mbs": "string",
  "mibgp": "string",
  "sic": "string",
  "bankInfo": "string",
  "bankAccount": "string",
  "taxRate": 0,
  "taxID": "string",
  "slug": "string",
  "logoID": 0,
  "logoFull": "string",
  "description": "string",
  "website": "string",
  "showPublicCertificates": true,
  "isDemoCompany": true,
  "roleOccupied": true,
  "headerColorHex": "string",
  "isCooperationManager": true,
  "ndviAvailable": true,
  "hasCooperations": true,
  "sprayingAdvisorAvailable": true
},
"id": 0,
"name": "string",
"city": "string",
"agriculturalID": "string",
"cadastralPlotNumber": "string",
"mapUrl": "string",
"latitude": 0,
"longitude": 0,
"surface": 0,
"totalSurface": 0,
"surfaceUnitName": "string",
"surfaceUnitID": 0,
"surfaceString": "string",
"surfaceBase": 0,
"surfaceBaseUnitID": 0,
"surfaceBaseUnitName": "string",
"linkedSurfaceCurrent": 0,
"linkedSurfaceUnitID": 0,
"linkedSurfaceUnitName": "string",
"linkedSurfaceCurrentString": "string",
"linkedSurfaceCurrentBase": 0,
"linkedSurfaceBase": 0,

```

```

"linkedSurfaceBaseUnitID": 0,
"linkedSurfaceBaseUnitName": "string",
"utilization": 0,
"facilityIDs": "string",
"facilityNames": "string",
"soilTypeID": 0,
"soilTypeName": "string",
"ownershipTypeID": 0,
"ownershipTypeName": "string",
"dateDeactivated": "2023-11-10T12:05:46.676Z",
"coordinates": "string",
"coordinateType": "String",
"tempMin": "string",
"coordinatesSource": "string",
"coordinatesArray": [
  {
    "latitude": "string",
    "longitude": "string"
  }
],
"tempMax": "string",
"tempUnit": "string",
"weatherIcon": "string",
"iconName": "string",
"linkedFields": [
  {
    "id": 0,
    "dateLinked": "2023-11-10T12:05:46.676Z",
    "dateUnlinked": "2023-11-10T12:05:46.676Z",
    "facilityID": 0,
    "facilityName": "string",
    "fieldID": 0,
    "fieldName": "string",
    "linkedSurface": 0,
    "linkedSurfaceUnitID": 0,
    "linkedSurfaceUnitName": "string",
    "linkedSurfaceString": "string",
    "linkedSurfaceBase": 0,
    "linkedSurfaceBaseUnitID": 0,
    "linkedSurfaceBaseUnitName": "string",
    "mapUrl": "string",
    "city": "string",
    "utilization": 0,
    "myParcels": [
      {
        "id": 0,
        "coordinates": "string",
        "wktFormat": "string",
        "plantationFieldID": 0,
        "plantationName": "string",
        "plantationDateLinked": "2023-11-10T12:05:46.676Z",
        "plantationDateUnlinked": "2023-11-10T12:05:46.676Z"
      }
    ],
    "operations": 0
  }
],
"operations": 0
}
],
},
"error": {
  "code": 0,
  "message": "string",
  "errors": [

```

```

    {
      "domain": "string",
      "reason": "string",
      "message": "string",
      "location": "string",
      "locationType": "string",
      "extendedHelp": "string",
      "sendReport": "string"
    }
  ]
}
}

```

Endpoint: <https://mobile-2-5.agrivi.com/TaskPesticides>

Type: GET

Description: Return all pesticide activities for task (def. ordering: ID ASC)

Parameters	Data Type	Description
Limit	integer(\$int32)	Number of records to return
Offset	integer(\$int32)	Number of records to offset
NeedPaging	boolean	Enable paging
DatabasePaging	boolean	Result paged server
OrderByField	string	Name of the field used for ordering
OrderByDirection	string	Ascending or Descending (asc or dsc)
TaskID	integer(\$int64)	ID of the task
Timestamp	string(\$date-time)	Last synchronisation date
IncludeCompanyDetail	boolean	Include details of the company
IncludeDeleted	boolean	Include deleted pesticides related tasks
Includeltems	boolean	Include the details for items used in the activities
IncludeFields	boolean	Include fields
V	string	Fix to value 1

Returns:

Code 200 Success

```

{
  "apiVersion": "string",
  "context": "string",
  "id": "string",
  "method": "string",
  "params": {},
  "data": {
    "kind": "string",
    "fields": "string",
    "etag": "string",
  }
}

```

```

"id": "string",
"lang": "string",
"updated": "2023-11-10T12:16:43.442Z",
"deleted": true,
"currentItemCount": 0,
"itemsPerPage": 0,
"startIndex": 0,
"totalItems": 0,
"pageIndex": 0,
"totalPages": 0,
"pagingLinkTemplate": "string",
"selfLink": "string",
"editLink": "string",
"nextLink": "string",
"previousLink": "string",
"items": [
  {
    "treatedPestsString": "string",
    "activeSubstancesString": "string",
    "description": "string",
    "isCustom": true,
    "waitingPeriod": 0,
    "protectedSurfaceString": "string",
    "protectedSurface": 0,
    "protectedSurfaceUnit": {
      "id": 0,
      "name": "string",
      "groupID": 0,
      "groupName": "string",
      "isSystemDefault": true,
      "isCurrentDefault": true,
      "isAreaUnit": true,
      "ratio": 0,
      "operations": 0
    },
    "linkedSurface": "string",
    "quantityPerHaString": "string",
    "quantityPerHa": 0,
    "pestsIDs": [
      0
    ],
    "costType": "string",
    "activityCategory": "string",
    "costPerUnit": 0,
    "costItemName": "string",
    "costItemQuantity": 0,
    "costItemUnitName": "string",
    "scoutingActivityID": 0,
    "protectedSurfaceUnitID": 0,
    "waterSolventQuantity": 0,
    "waterSolventUnitID": 0,
    "fieldReEntryPeriod": 0,
    "company": {
      "id": 0,
      "title": "string",
      "ownerID": 0,
      "countryID": "string",
      "currencyID": "string",
      "currencyName": "string",
      "unitSystemID": "string",
      "languageID": "string",
      "languageName": "string",
      "email": "string",
      "phone": "string",
      "foundationDate": "2023-11-10T12:16:43.442Z",
      "dateAdded": "2023-11-10T12:16:43.442Z",

```

```

    "partnerCode": "string",
    "city": "string",
    "address": "string",
    "postalCode": "string",
    "oib": "string",
    "mbs": "string",
    "mibgp": "string",
    "sic": "string",
    "bankInfo": "string",
    "bankAccount": "string",
    "taxRate": 0,
    "taxID": "string",
    "slug": "string",
    "logoID": 0,
    "logoFull": "string",
    "description": "string",
    "website": "string",
    "showPublicCertificates": true,
    "isDemoCompany": true,
    "roleOccupied": true,
    "headerColorHex": "string",
    "isCooperationManager": true,
    "ndviAvailable": true,
    "hasCooperations": true,
    "sprayingAdvisorAvailable": true
  },
  "isPlanned": true,
  "dateRemoved": "2023-11-10T12:16:43.442Z",
  "taskCategories": "string",
  "taskDate": "2023-11-10T12:16:43.442Z",
  "facilityID": 0,
  "facilityName": "string",
  "agriculturalID": "string",
  "cost": 0,
  "costConverted": 0,
  "costCurrencyID": "string",
  "costPerUnitString": "string",
  "quantityUnitName": "string",
  "treatedSurfaceString": "string",
  "entryType": "string",
  "distributionType": "string",
  "id": 0,
  "taskID": 0,
  "date": "2023-11-10T12:16:43.442Z",
  "startTime": "2023-11-10T12:16:43.442Z",
  "finishTime": "2023-11-10T12:16:43.442Z",
  "pesticideID": 0,
  "pesticideName": "string",
  "pesticideAmount": 0,
  "pesticideAmountUnitID": 0,
  "pesticideAmountUnitName": "string",
  "pesticideAmountString": "string",
  "pesticideAmountBase": 0,
  "pesticideAmountBaseUnitID": 0,
  "pesticideAmountBaseUnitName": "string",
  "warehouseID": 0,
  "warehouseName": "string",
  "warehouseBinID": 0,
  "warehouseBinName": "string",
  "storageName": "string",
  "pesticideManufacturerName": "string",
  "taskName": "string",
  "workerID": 0,
  "fieldIDs": [
    0
  ]

```

```

    ],
    "activityFields": [
      {
        "id": 0,
        "amount": 0,
        "fieldID": 0,
        "fieldName": "string",
        "linkedFieldID": 0,
        "type": "string",
        "facilityID": 0,
        "stageID": 0,
        "sortID": 0
      }
    ],
    "operations": 0
  }
],
},
"error": {
  "code": 0,
  "message": "string",
  "errors": [
    {
      "domain": "string",
      "reason": "string",
      "message": "string",
      "location": "string",
      "locationType": "string",
      "extendedHelp": "string",
      "sendReport": "string"
    }
  ]
}
}
}

```

3.5 API Specifications for EFRA Analytics Powerhouse

Overview

The Analytics Powerhouse is the EFRA component responsible for the management of AI models and provides a set of APIs for uploading, validating, executing and monitoring AI models (see **D4.2 Efra Data Hub and Analytics Powerhouse**). It depends on the EFRA Data-Hub component for accessing and storing configuration data and AI models (items 1 and 2 in the following diagram).

In this section we provide an overview of the process for uploading and registering a given AI model in the EFRA Powerhouse. The EFRA platform has been designed according to the micro services approach (see **D4.1 EFRA Architecture & Green Operations**), therefore in order to take advantage of containerization each EFRA AI model must be provided as a docker image.

Once the AI model has been deployed as a docker image, it must be pushed to the EFRA docker registry; however, in order to be available for future executions it must be successfully validated and registered into the EFRA platform. The registration procedure requires:

the URL of the model docker image

- the ID of a validation dataset (already present in the Data Hub).
- a set of (optional) additional metadata for labelling/tagging the model

Each model, to be “trusted”, must be capable of processing an existing EFRA validation dataset without raising errors or exceptions, so whenever a model (by means of its docker image) is registered in the Powerhouse, it is validated by triggering an execution of the model on the specified validation dataset. If the model execution is successful, then it is given an ID and stored in the EFRA AI model registry, otherwise the registration process fails, and the model will not be available for execution. Such validation strategy comes from the assumption that an EFRA AI model is logically a function that processes a dataset and produces in output a new dataset.

The following diagram illustrates the process for uploading and validating a model on the EFRA Powerhouse.

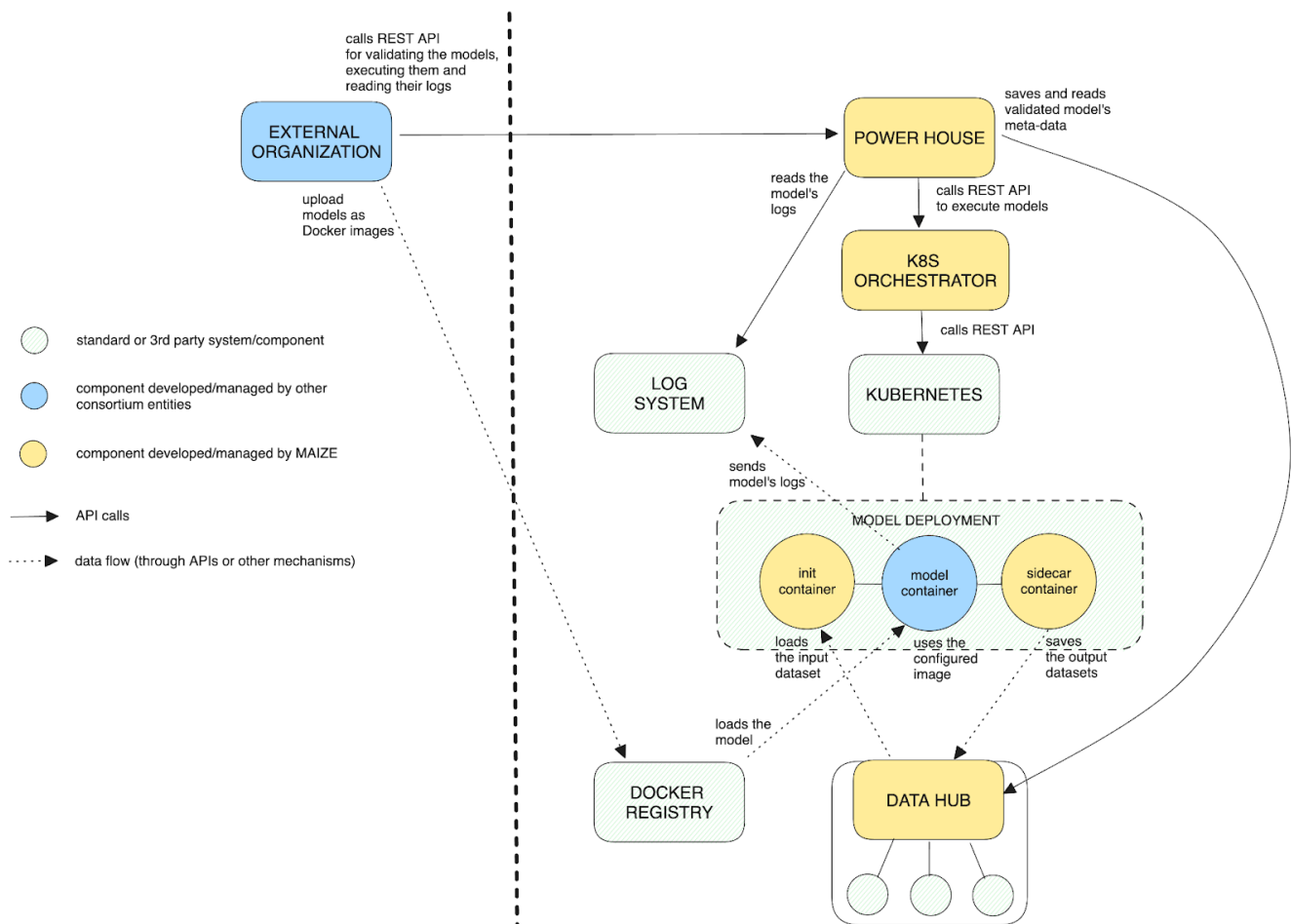


Figure 2: The process for uploading and validating a model on the EFRA Powerhouse

APIs Specification

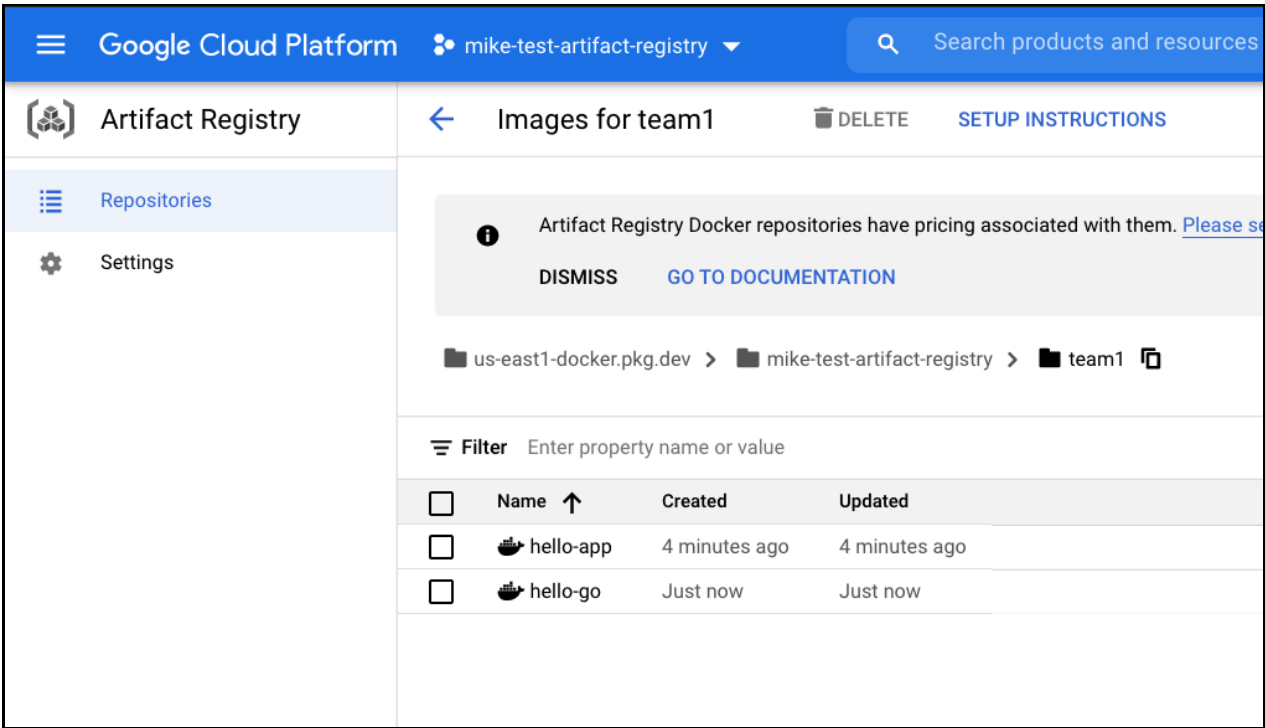
The process for uploading and validating a model on the EFRA Powerhouse potentially requires the interaction with 2 distinct sets of APIs:

- the EFRA Docker Registry APIs: for storing the docker image of the AI model
- The EFRA Powerhouse validation APIs: for registering the AI model in the platform

Docker Registry APIs

The EFRA platform is deployed on a Kubernetes cluster over a cloud infrastructure (*see D4.2 Efra Data Hub and Analytics Powerhouse*) and the selected cloud infrastructure for EFRA is Google Cloud. The docker registry used

in the Data Hub consist of Artifact Registry a cloud service available in Google Cloud infrastructure. The docker registry can be accessed from a web interface as well as through the Docker protocol, allowing the use of common Docker clients, including the official [Command-line Tool](#), for pushing and pulling images.



To access the service, users (e.g. EFRA partners involved in the creation of AI models) must be granted an individual authorization; such authorization must be requested via mail at efra.support@maize.io.

Powerhouse validation API

The validation API is the means by which registering models' metadata into the DataHub, enabling their execution for everyone having access to the Analytics Powerhouse. By now a model given as a Docker image is considered valid if it executes without errors on an input dataset producing the corresponding output dataset. Because the execution of a model may take a lot of time, the validation process is necessarily an asynchronous operation and must be viewed together with the status API.

HTTP status codes

Code	Description
202	Validation process was successfully started
400	Invalid input values were sent with the request
422	Input datasets does not exist into DataHub

500

There was a failure executing the model

RequestPOST <http://{{powerhouse}}/api/v1/validate>

<your-api-key>

Content-Type:application/json

```
{
  "model": {
    "namespace": "string",
    "name": "string",
    "version": "string",
    "image": "string"
    "desc": "string",
    "tags": [ "string", ... ],
  },
  "inputDataset": {
    "namespace": "string",
    "name": "string",
    "version": "string"
  },
  "outputDataset": {
    "namespace": "string",
    "version": "string"
    "desc": "string",
    "tags": [ "string", ... ]
  }
}
```

As a response you'll receive following:

```
{
  "timestamp": "2023-11-10T15:38:37.619+01:00",
  "status": 202,
  "path": "/api/v1/validate",
  "result": {
```

```

    "jobName": "validate",
    "runId": "4d865810-b1a2-45c1-be96-f8417f5a6538",
    "status": "STARTED",
    "terminated": false,
    "validated": false
  }
}

```

Each model validation is accomplished by a job whose id is returned in the call's result. The *jobName* and *runId* parameter must be saved in order to query later the PowerHouse for the job's termination.

Powerhouse status API

In order to know when a model validation or simple execution is terminated, this API must be called specifying the *jobName* and *runId* obtained with the model submission.

HTTP status codes

Code	Description
200	The execution/validation status of the model
400	Invalid input values were sent with the request
404	Job with the name and/or runId specified does not exists
500	There was a failure reading the execution status

Request

GET <http://{{powerhouse}}/api/v1/status/{{jobName}}/{{runId}}>

<your-api-key>

An example of the related response is the following one, showing a model successfully terminated and registered into the DataHub.

```

{
  "timestamp": "2023-11-10T15:49:03.560+01:00",
  "status": 200,
  "path": "/api/v1/status/validate/4d865810-b1a2-45c1-be96-f8417f5a6538",
  "result": {

```

```

    "jobName": "validate",
    "runId": "4d865810-b1a2-45c1-be96-f8417f5a6538",
    "status": "ACQUIRED",
    "startTime": "2023-11-10T15:38:37.600384",
    "endTime": "2023-11-10T15:40:39.406481",
    "elapsed": 121806,
    "acquisitionTime": "2023-11-10T15:41:38.161333",
    "model": {
      "namespace": "datahub-dev",
      "name": "hello-world",
      "version": "0.0.4"
    },
    "terminated": true,
    "validated": true
  }
}

```

The *status* field may be one of the following: UNKNOWN, STARTED, EXECUTING, STOPPED, FAILED, COMPLETED, ACQUIRED. When a job is COMPLETED, it means that the model has successfully terminated but not yet sent to the DataHub. A status of ACQUIRED assures that model's metadata were also correctly registered into the DataHub. In any case, also when errors were raised, the *terminated* flag grants that the job was fully executed.

4 EFRA Data & Analytics Marketplace

4.1 Introduction

The **EFRA Data & Analytics Marketplace** is an advanced web platform that supports data and AI model exchange and provides decision-making tools to stakeholders in the food safety and agricultural sectors. Built for flexibility and scalability, the marketplace offers a range of key functionalities to enhance data use and insights:

- **Data and AI Model Search and Acquisition:** Users can search for and acquire high-quality datasets and AI models. Datasets are curated within the platform, and AI models, packaged as Docker images, are available for integration into user workflows.
- **Interactive Dashboards:** Interactive decision support tools tailored to EFRA use-cases, such as early anomaly detection in poultry farming, pest outbreak predictions, and food safety incident forecasting. These dashboards provide both historical insights and future predictions.
- **Data Mapping Tool:** This tool enables users to map their datasets to the EFRA Semantic Backbone and ontologies, ensuring alignment with standardized vocabularies for better data consistency and usability.
- **Food Safety Data Collection:** Users can suggest new food safety data sources, which are automatically scraped, formatted, and integrated into the platform's dataset repository for broader access.
- **Regulatory Compliance Tools:** The platform includes tools like document summarization, fine-tuned for food regulatory texts, enabling users to quickly understand complex regulations and ensure compliance.
- **Wallet and Transaction Management:** Users can manage their virtual wallets, purchase datasets or AI models using platform credits, and track their transactions for future use.
- **Session and Asset Management:** The platform allows users to securely manage their sessions, view and edit owned or acquired datasets, and control access to their assets, ensuring proper data governance.

In the sections that follow we first introduce its core architectural principles and layering (4.2) and its core functionalities (4.3), detailed technical specifications (4.4), and a demonstrator based on prototypes for all available screens (4.5).

4.2 Architecture Overview

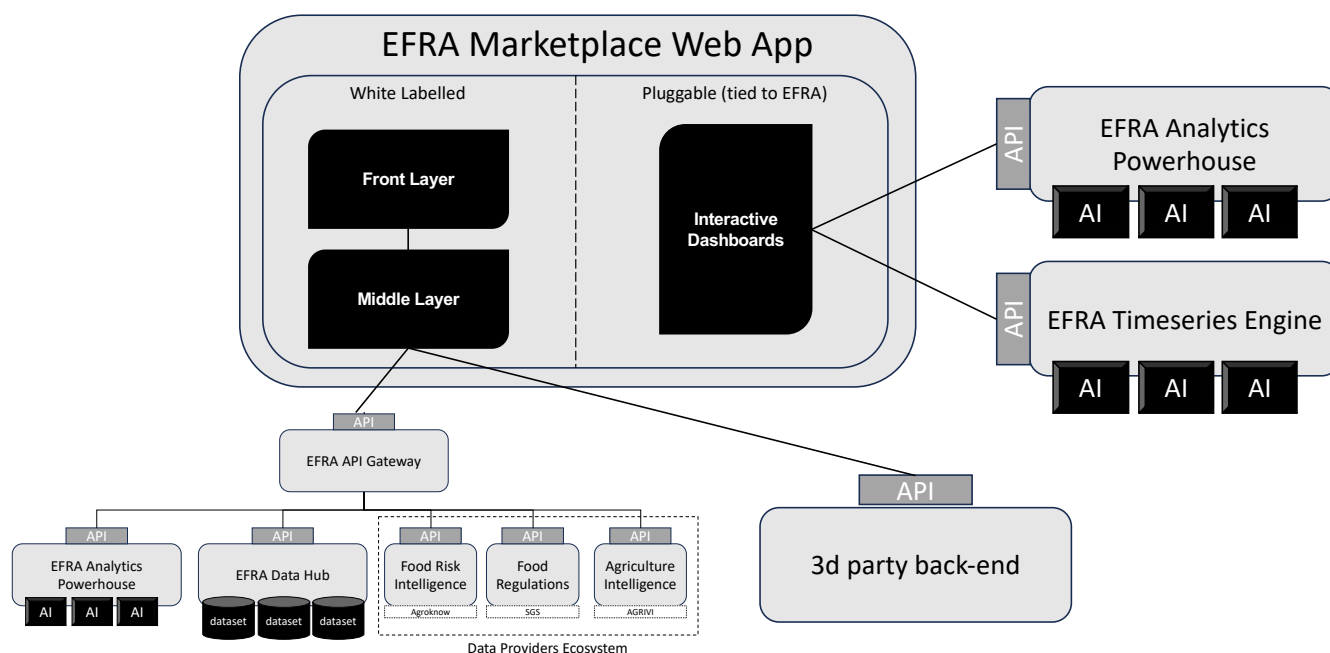


Figure 3: High-level architecture of the EFRA Data & Analytics Marketplace

The EFRA Data & Analytics Marketplace leverages a modular and flexible architecture, built upon a **Front-End Layer** and a **Middle Layer** that abstracts the back-end implementation. This ensures compatibility across different back-end configurations, making the marketplace adaptable and scalable, and allowing it to be white-labelled for potential exploitation by third parties. The marketplace is built on top of and extends the **FSM Data Market**, created during the H2020 Food Safety Market (FSM) EU project. It incorporates many of its existing capabilities while adding new, enhanced features for the EFRA context. The main components of the EFRA Marketplace architecture are the following:

1. **Front-End Layer:** This layer manages all user interactions, rendering the user interface and ensuring seamless access to the marketplace's functionality. Users can search and acquire datasets and AI models, explore analytical tools, and access decision support tools through intuitive interactive interfaces. It connects to the middle-layer to process user actions and dynamically update the interface based on the back-end data and services.
2. **Middle Layer:** The middle-layer abstracts the complexity of back-end services, facilitating easy communication between the front-end and various back-end configurations. Through well-defined APIs (see 4.4.3), it connects the front-end with datasets, analytics, and AI models without exposing the underlying infrastructure. This abstraction enables the marketplace to be configured over different back-ends, providing flexibility and scalability for future expansions.
3. **Back-End Configurations:** The EFRA Data & Analytics Marketplace can be configured over multiple back-ends. Currently, our focus is interconnection with the **EFRA Platform**, which consists of: (a) **EFRA Data Hub:** A repository for curated, high-quality datasets. The EFRA Data Hub provides the core functionalities for searching and acquiring datasets, and for submitting new datasets, (b) **EFRA Analytics Powerhouse:** A collection of machine learning models, dockerized and ready for use by third parties. The Analytics Powerhouse provides the core functionalities for searching and acquiring AI models, and for submitting

new AI models, (c) **EFRA Scraping Platform**: A powerful data collection system that gathers and organizes data from various food safety sources. The Scraping Platform provides access to curated datasets coming from the web.

4. **Pluggable Component for Interactive Dashboards**: The marketplace includes a **pluggable component** for interactive dashboards and decision support tools, tailored to the EFRA use cases. This component connects directly to the **EFRA Timeseries Engine** and the **EFRA Analytics Powerhouse** to interact with the underlying EFRA AI models per use-case and present their results in a user-friendly way through appropriate interactive dashboards.

Its **modular architecture** and **back-end abstraction** enable the EFRA Data & Analytics Marketplace to be **white-labelled** for deployment in different contexts and industries. This allows for additional exploitation pathways beyond the core EFRA project business model, including partnerships with external organizations or commercialization opportunities where the marketplace can be customized and branded according to specific needs.

4.3 Key Functionalities

The core capabilities of the **FSM Data Market** that will be leveraged and appropriately enhanced and integrated in the **EFRA Marketplace** include the following:

1. **User and Session Management**: The EFRA Marketplace will allow users to register using an email and password, log in to receive an authentication token, and manage sessions securely with tokens that expire and require renewal upon login. Users can log out to invalidate their session, ensuring secure access to the platform's assets and functionalities.
2. **Wallet Management**: Users will be able to view their wallet balance, expressed in coins or credits, and use these credits to purchase datasets or AI models within the marketplace. Upon acquiring assets, their wallet balance will be automatically debited, allowing for seamless transactions.
3. **Asset Management**: The platform will allow users to manage their datasets by viewing datasets they own or have acquired. They can download datasets, modify properties such as metadata, publish them for public or private use, and control legal terms like commercial exploitation. Users can also acquire datasets, which will be tracked in their account for future use.

In addition to leveraging the core capabilities of the **FSM Data Market**, the **EFRA Marketplace** introduces several new features to enhance its functionality:

1. **Data Mapping Tool**: The EFRA Marketplace includes a **Data Mapping Tool** that allows users to map their datasets to the **EFRA Semantic Backbone** and its ontologies. This tool simplifies the process of aligning user-uploaded data with standardized vocabularies, ensuring consistency and interoperability across datasets. By leveraging the EFRA Semantic Backbone, users can enhance the discoverability, compatibility, and usability of their data within the EFRA ecosystem.
2. **AI Models as Tradable Assets**: The EFRA Marketplace extends the concept of tradable assets beyond datasets to also include **AI models**, allowing users to search, acquire, and deploy AI models packaged as Docker images. This enables users to access pre-trained models that can be integrated into their workflows, enriching their data analysis capabilities.

3. **Suggest Sources and Receive Food Safety Incident Datasets:** A unique feature of the EFRA Marketplace is the ability to **suggest new food safety incident data sources**. Users can provide the root URL of a potential data source, and the system (using the EFRA Scraping Platform) will automatically retrieve and format the content into a well-structured CSV file that adheres to the **EFRA ontology**. This process provides an incentive to users to provide useful intelligence to the EFRA ecosystem and allows the EFRA Scraping Platform to expand to new data sources.
4. **Food Safety Incidents Forecasts Dashboard:** This dashboard integrates with the EFRA Scraping Platform and EFRA Timeseries Engine to deliver predictive insights into future food safety incidents. It provides forecasts for a set of key commodities and hazards by analysing historical data and trends. Users can visualize anticipated risks for specific commodities, enabling them to take proactive safety measures and manage risks more effectively.
5. **Interactive Decision Support Dashboards:** The EFRA Marketplace also includes a pluggable component that connects directly with the EFRA Timeseries Engine and the EFRA Analytics Powerhouse to interact with and visualise the results of the AI models created for the EFRA use-cases. More specifically, the EFRA Marketplace provides access to the following decision support tools per EFRA use-case scenario:

Scenario #1 Enhancing Safety and Efficiency in Poultry Farming

- **Interactive Dashboard for Early Outlier Detection:** A dashboard offering both historical and predictive visualizations for flock mortality and weight anomalies. It enables users to input flock characteristics to predict outliers using machine learning and explore historical outliers interactively. Visual indicators flag outliers for early intervention.
- **Flock Risk Stratification Dashboard:** This dashboard groups flocks into risk categories (high, medium, low) based on health data. It visualizes mortality and weight gain trends and highlights high-risk flocks, allowing users to drill down into flock details for targeted management.

Scenario #2 Enhanced Predictive Capabilities for Pest Alarms

- **Interactive Decision Tree and Prediction Dashboard:** A tool that visualizes decision trees for specific pests and regions based on AI models. Users can input environmental parameters to predict pest outbreaks. It provides interactive decision tree exploration and predictive outputs on pest risks based on real-time conditions.

Scenario #3 Automated Regulatory Analysis & Summarization Module

- **Regulatory Document Summarization Dashboard:** A dashboard allowing users to upload or select regulatory documents and generate customizable summaries. Summaries can range from high-level overviews to detailed breakdowns, helping compliance teams quickly grasp key points or dive deeper into regulatory sections.

4.4 Technical Specifications

This section presents detailed technical specifications for the EFRA Marketplace, including its core entities and relationships (4.4.1), its front-end layer (4.4.2), and its middle layer (4.4.3).

4.4.1 Core Entities and Relationships

This section provides detailed information for the core entities and relationships relevant to the EFRA Marketplace. In case of a stand-alone back-end deployment, these can be directly used to define the ER schema of the database. In the other deployments, as in the case of the EFRA Platform acting as the back end, the following information can be approached as a concise way to present terms that appear in the rest of the document.

Asset (dataset): This entity represents a data asset stored in the back end. In the EFRA Platform deployment, the asset is stored either in the EFRA Data Hub or retrieved directly from the EFRA Scraping Platform. Its core properties include:

- Id (required)
 - A unique identifier for the asset
- Title (required)
 - A title that characterises the dataset
- Content (required)
 - The content of the dataset (clob, a vector of characters)
- File type (required)
 - The file type of the dataset (e.g., csv, html, pdf)
- Row Delimiter
 - A character that is used to parse the content into separate rows / records
- Column Delimiter
 - A character that is used to parse each row / record into separate columns
- Metadata tags
 - A free list of terms
- Date range
 - A start and end date that define the period that the dataset contains information for (e.g., product recalls where the earliest recalls happened at the start date and the most recent at the end date)
- Countries
 - A list of countries for which there is information in the dataset (e.g., product recalls that involve these countries)
 - The vocabulary must be controlled (e.g., from a standard list of country names)
- Original Source
 - A list of organisation names
 - The vocabulary must be controlled (e.g., from a standard list of public food safety authorities)
- Thematic classes
 - A list of terms that characterize the type of information in the dataset (e.g., food safety incidents, scientific publications, lab tests, etc.)
 - The vocabulary must be controlled

Asset (AI model): This entity represents an AI model asset stored in the back end. In the EFRA Platform deployment, the asset is stored in the EFRA Analytics Powerhouse as a self-contained docker image. Its core properties include:

- Id (required)
 - A unique identifier for the asset
- Title (required)

- A title that characterises the AI model
- Content (required)
 - The Docker image file that contains the executable AI model (tar file)
- Model Description (required)
 - A textual description of the AI model, its purpose, and how it processes data
- Input Description
 - A detailed description of the input data required by the AI model, specifying what types of data the model expects.
- Input Format
 - The format of the input data, such as CSV, JSON, or time-series data, that the AI model is designed to process.
- Output Description
 - A detailed description of the outputs produced by the AI model, such as predictions, classifications, or analysis results.
- Output Format
 - The format in which the AI model outputs its results (e.g., JSON, CSV, or XML).
- Dependencies
 - A list of software dependencies required to run the AI model (e.g., specific libraries, packages, or hardware requirements).
- Metadata Tags
 - A free list of terms that describe the AI model (e.g., "machine learning", "classification", "forecasting") to help categorize and search for models within the platform.

User:

- Email (required)
 - The user's email, must be unique and is used as an identifier
- Password (required)
 - The user's password, should be handled securely (hashed & salted)
- Metadata tags
 - A list of terms
- Available Coins (required, default 0)
 - The coins available in the user wallet

Active Sessions:

- Related_user_email (required)
 - foreign key to User table
- Authkey (required)
 - A secure string that is issued by the back end to uniquely identify an ongoing user session
 - The client is responsible to store this and append it at every subsequent request
- Issued_at (required)
 - A timestamp to check for expiration of the authkey
 - The authkey should expire at a relatively short time frame
- Has_expired (required)
 - To periodically we can remove has_expired = true entries

Asset_Ownership: User <-> Asset relationship (many-to-many)

- Is_owner (required default false)
 - If true, the connected user is the owner of the asset (i.e., they can edit the metadata and key properties of the asset)
- Has_acquired (required, default false)
 - If true, the connected user has acquired the asset (i.e., they can download it at any point)

4.4.2 Front-End Layer

This section presents the screens of the Front-end Layer. They are categorised into two types: (a) general screens for asset trading, presented in section 4.4.2.1, which communicate with the Middle Layer controllers to ensure that they can be easily re-purposed in a white-labelling exploitation pathway, and (b) screens only applicable in a deployment tied to the EFRA Platform, presented in section 4.4.2.2. The latter screens do not require a middle layer component and directly communicate with the EFRA Timeseries Engine and the EFRA Analytics Platform to provide their functionalities.

4.4.2.1 General screens for asset trading

General Principles

Each section below focuses on a particular screen and presents the front-end functionalities that the user can find in the screen. Text in [blue](#) indicates the middle-layer functionality that provide the relevant functionality to the front-end screen. Text in [green](#) indicates that a particular action in a screen redirect to a different screen.

The calls to the middle-layer are always asynchronous. Ideally the front-end leverages this and does not freeze while waiting for a middle-layer call to finish. The front-end should remain responsive while a middle-layer call is being processed.

Register

- User inputs email and password and clicks on Register ([Register](#) is called)
- If register call is successful, the [Login](#) middle-layer function is also called

Login

- User inputs email and password and clicks on login button ([Login](#) is called)
- The token returned by the login call must be securely stored in the client-side

Search Datasets

- Allows searching with the following filters:
 - Key properties ([Dataset Key Properties Search](#))
 - date range, let the user indicate a start and end date
 - thematic classes, list the available thematic classes ([List Thematic Classes](#)) and let the user chose one or more of them
 - countries, list the available countries ([List Countries](#)) and let the user chose one or more of them
 - upper limit for value (coins), let the user indicate the desired number

- Metadata tags search ([Dataset Metadata Tags Search](#))
 - free text, separate tags with comma
- Get list of results with the option to click to preview any dataset
 - Clicking the button redirects to: [Dataset Preview & Acquire Screen](#)
- The filters the user has used up to now and the ids of the datasets in the results are stored so that more filters can be added to further narrow down the current results (the middle-layer facilitates this by allowing the specification of the [dataset_ids] that are the result of previously used filters and returns the result of applying new filters over the specified [dataset_ids], see the [dataset_id] request parameter in the [Dataset Search Controller](#))

Edit Dataset Properties

- Allow the user to specify/edit the properties of the dataset ([Get Dataset Properties](#), [Set Dataset Properties](#))
 - Date range, let the user specify a start and/or end date, also showing the relevant values if previously specified for the dataset
 - List the thematic classes ([List Thematic Classes](#)) and let the user choose zero or more, indicating the ones previously specified for the dataset if they exist
 - List the countries ([List Countries](#)) and let the user choose zero or more, indicating the ones previously specified for the dataset if they exist
 - Allow the user to add/edit metadata tags, indicating the ones previously specified for the dataset if they exist

Dataset Preview & Acquire

- Show price and all other properties of dataset ([Get Dataset Properties](#))
- Show first rows preview ([Get Dataset Preview](#))
- Purchase / Acquire dataset button (clicking on the button triggers [Acquire Dataset](#))

My Datasets

- The user can see the datasets they own ([List Owned Datasets](#)), with the following buttons per dataset
 - Download content, which links to [Dataset Preview & Acquire](#) screen
 - Edit dataset (which links to the [Edit Dataset Properties](#) screen)
- The user can also see in a separate list the datasets they have acquired / purchased ([List Acquired Datasets](#)), with the following buttons per dataset
 - Download content, which links to [Dataset Preview & Acquire](#) screen

Publish Dataset

- Allows the user to choose a file from their computer to upload
- The user is encouraged to do the following:
 - Define the key properties of the dataset; a button redirects to [Edit Dataset Properties](#) and afterwards the user is redirected back to this screen
 - Map the dataset to the EFRA Semantic Backbone and ontologies; a button redirects to the Mapping Tool, which is an external tool of the EFRA Platform. The user should upload the resulting dataset after going through the mapping process.
- The user inputs a monetary value for the dataset (in coins) that can be zero or positive
- A button at the end lets the user publish the dataset ([Publish Dataset](#))
 - Note that the call to [Publish Dataset](#) can take up considerable time, the front-end should not block while the dataset is being published

Search AI models

- Allows searching with the following filters:
 - Key properties ([Model Key Properties Search](#))
 - Model Description: Users can search for AI models based on their detailed descriptions, including the type of model and its purpose.
 - Input Format: Users can specify the format of input data required by the AI model (e.g., CSV, JSON).
 - Output Format: Users can filter models by the format of the output (e.g., JSON, CSV, XML)
 - Monetary value: Users can search AI models based on their price, setting an upper limit for coins or credits.
 - Metadata tags search ([Model Metadata Tags Search](#))
 - free text, separate tags with comma
- Get list of results with the option to click to preview any dataset
 - Clicking the button redirects to: [AI Model Preview & Acquire Screen](#)
- The filters the user has used up to now and the ids of the models in the results are stored so that more filters can be added to further narrow down the current results (the middle-layer facilitates this by allowing the specification of the [model_ids] that are the result of previously used filters and returns the result of applying new filters over the specified [model_ids], see the [model_id] request parameter in the [Model Search Controller](#))

Edit AI Model Properties

- Allow the user to specify/edit the properties of the AI model ([Get Model Properties](#), [Set Model Properties](#))
 - Model Description: Users can edit the textual description of the AI model, detailing its purpose and how it processes data.
 - Input Description: Users can define or modify the description of the input data the AI model requires.
 - Input Format: Users can specify or update the format of input data required by the AI model (e.g., CSV, JSON).
 - Output Description: Users can edit the detailed description of the AI model's outputs, such as predictions or analysis results.
 - Output Format: Users can specify or update the output format (e.g., JSON, CSV, XML).
 - Dependencies: Users can modify the list of required software or hardware dependencies (e.g., specific libraries, packages).
 - Metadata Tags: Users can add or edit metadata tags that help categorize and describe the AI model (e.g., "forecasting", "classification").

AI Model Preview & Acquire

- Show price and all other properties of the model ([Get Model Properties](#))
- Purchase / Acquire dataset button (clicking on the button triggers [Acquire Model](#))

My AI Models

- The user can see the model they own ([List Owned Models](#)), with the following buttons per model
 - Download content, which links to [AI Model Preview & Acquire](#) screen
 - Edit model (which links to the [Edit AI Model Properties](#) screen)

- The user can also see in a separate list the models they have acquired / purchased ([List Acquired Models](#)), with the following buttons per dataset
 - Download content, which links to [AI Model Preview & Acquire](#) screen

Publish AI Model

- Allows the user to choose a file from their computer to upload
- The user is encouraged to do the following:
 - Define the key properties of the dataset; a button redirects to [Edit Dataset Properties](#) and afterwards the user is redirected back to this screen
- The user inputs a monetary value for the model (in coins) that can be zero or positive
- A button at the end lets the user publish the model ([Publish Model](#))
 - Note that the call to [Publish Model](#) can take up considerable time, the front-end should not block while the dataset is being published

4.4.2.2 Screens tied to the EFRA Platform

Source Request

- The Source Request screen allows users to suggest new food safety web sources for scraping. By submitting the root URL and some key information about the source, the user can request the addition of a new dataset to the EFRA Marketplace. Upon approval and successful scraping of the data, a well-formatted dataset will be added to the user's account as a reward.
- **Source Submission:**
 - **Root URL:** Users must input the root URL of the web source they wish to have scraped. The URL should be the primary address of the data source (e.g., a government agency's food recall page).
 - **Source Information:** Users will provide additional key information about the source. This includes:
 - **Source Description:** A brief description of the type of data available at the source (e.g., food safety incidents, product recalls, lab test results).
 - **Source Language:** The primary language of the web source.
 - **Country or Region:** The geographic scope of the data, such as the country or region the source is relevant to.
 - **Data Type:** The type of data expected to be scraped (e.g., product recalls, chemical hazards, food testing results).
- **Review Process:**
 - A human curator will manually review the request, ensuring the source is suitable for scraping and that the information provided by the user is complete and accurate.
 - If the request is approved, the curator will configure the **EFRA Scraping Platform** to begin scraping data from the provided source.
- **Scraping and Dataset Reward:**
 - Once the scraping process is completed and the data is formatted, a well-structured dataset (e.g., in CSV format) will be created following the **EFRA ontology**.
 - This dataset will then be manually added to the user's account in the [My Datasets](#) section, where the user can access, preview and download it.

Incident Forecasting Decision Support Dashboard

- The Incident Forecasting Decision Support Dashboard connects directly with the **EFRA Timeseries Engine** to deliver predictive insights into the occurrence of food safety incidents related to various commodities and hazards. Users can explore forecasts, visualize incident trends, and make data-driven decisions.

- **Core Functionalities:**

- List of Available Forecasts
 - The dashboard presents a comprehensive list of available forecasts for different commodities (e.g., fruits, vegetables, meat products) and hazards (e.g., chemical contamination, microbiological hazards).
 - Users can search through the list of forecasts by entering keywords related to the commodity or hazard they are interested in (e.g., "salmonella", "poultry contamination").
- Forecast Selection
 - After searching or selecting from the list, users can choose a forecast, which will then load the relevant historical and predictive data.
- Interactive Time-Series Graph
 - Once a forecast is selected, an interactive time-series graph is displayed. The graph shows
 - Historical Data: The actual number of incidents reported per month for the selected commodity or hazard over a past period
 - Forecast Data: The expected number of incidents per month, generated by executing forecasting models on the historical incident data. The forecast data is based on advanced AI models running on the **EFRA Timeseries Engine**, providing predictions for future incident trends.
 - Users can interact with the time-series graph by
 - Hovering for Details: By hovering over any point in the graph, users can view precise historical data or forecasted incident numbers for that specific month.
 - Toggle Data: Users can toggle between viewing just the historical data, just the forecasted data, or both.

Interactive Dashboard for Early Outlier Detection in Mortality and Weight Gain

- This dashboard helps farm managers detect early signs of anomalies in flock mortality and weight gain patterns, allowing them to make data-driven decisions to prevent productivity loss. The screen provides both historical and predictive insights, enabling users to input flock characteristics and receive outlier predictions powered by machine learning models. Additionally, the dashboard allows for interactive exploration of historical outliers to identify patterns and risk factors.
- Core Functionalities
 - Flock Characteristics Input
 - The dashboard provides input fields where users can specify the characteristics of their flock, including the evolution of their mortality and weight gain in the early weeks of their lifecycle.
 - After entering the flock characteristics, users click the "Submit" button, which initiates the machine learning model to predict outlier behaviour.
 - Outlier Detection and Probability Prediction
 - After the user inputs flock data, the system runs a machine learning model that predicts the likelihood of the flock exhibiting outlier behaviour (e.g., top 10% in mortality or lowest 10% in weight gain) by the end of its 4-week lifecycle.
 - The results are displayed in the form of
 - Probability Score: Shows the likelihood of the flock becoming an outlier (e.g., 80% likelihood of abnormal mortality).
 - Confidence Interval: A range showing the confidence level in the prediction to help farm managers assess the risk.
 - Visualization of Historical Outliers

- The dashboard presents historical data on previous flocks, highlighting those that were identified as outliers. This helps users understand the context and factors contributing to past anomalies.
- Users can filter the historical data by various filters to better understand the conditions under which outliers most usually happen.

Interactive Dashboard for Flock Risk Stratification

- This dashboard categorizes flocks into different risk groups based on their health and performance data. The screen provides visual insights into how mortality and weight gain rates evolve over time across these different risk groups. The goal is to allow farm managers to focus on high-risk flocks that require immediate attention and drill down into the detailed characteristics of these flocks to identify the underlying causes of poor performance.
- Core Functionalities
 - Risk Grouping Visualization
 - The dashboard assigns each flock to a risk category based on its performance in terms of mortality and weight gain rates.
 - The different groups are presented as Line Charts.
 - Drill-Down on Flock Characteristics
 - Users can select a specific flock group / line chart and drill down into the underlying characteristics of the flocks belonging in the selected group.
 - Once a flock group is selected, users can explore detailed data on the flock group, such as the distribution of different breeds, farms, feed types, etc. in this particular group.

Interactive Decision Tree and Prediction Dashboard for Pest Management

- This dashboard helps users predict pest outbreaks by leveraging historical data and AI models. The dashboard provides decision tree visualizations for specific pests and geographic regions, helping users understand the conditions that lead to pest emergence. By inputting environmental parameters, users can receive predictive insights into the likelihood of pest outbreaks, enabling proactive pest management strategies.
- Core Functionalities
 - Pest and Region Selection
 - Pest Type Selection: The dashboard provides a drop-down menu or searchable list of available pest types (e.g., aphids, mites, caterpillars) based on historical data. Users can select the pest they wish to predict.
 - Region Selection: Users can also select a geographic region (e.g., Europe, North America, specific countries or localities) where they want to analyze pest emergence. This allows the AI model to tailor predictions to location-specific environmental factors.
 - Decision Tree Visualization
 - Tree Structure: Once a pest type and region are selected, the system displays an interactive decision tree. This tree visualizes the environmental conditions under which the selected pest is likely to appear. Decision nodes represent conditions such as temperature thresholds, humidity levels, soil moisture, and crop type.
 - Expandable and Collapsible Branches: Users can expand or collapse branches of the decision tree to explore deeper levels of the model. This allows them to zoom in on specific conditions or explore higher-level summaries of the tree.
 - Parameter Input for Predictive Modelling

- Users can input environmental data to get refined predictions. The parameters include crop type, recent soil moisture, recent humidity, recent temperatures.
- Prediction Output
 - Based on the user's inputs, the AI model generates a probability score indicating the likelihood of a pest outbreak under the current conditions.

Interactive Dashboard for Regulatory Document Summarisation

- This dashboard is designed to help compliance officers and legal teams quickly generate customizable summaries of regulatory documents. Users can upload their own documents or select from pre-existing regulatory documents within the platform. The system offers flexible control over the level of detail in the summaries, ranging from high-level overviews to detailed, section-by-section analyses. This allows teams to efficiently grasp key regulatory points or dive deeper into specific sections as required.
- Core Functionalities
 - Document Upload and Selection
 - The dashboard provides a simple interface where users can upload their own regulatory documents in various formats.
 - In addition to uploading their own documents, users can select from a list of pre-existing regulatory documents available within the platform. These documents cover a wide range of food safety-related topics, including food safety laws, labelling standards, and compliance guidelines.
 - Summary Generation
 - Once a document is uploaded or selected, the system generates a summary based on the user's preference for the level of detail. The system offers two types of summaries: high level (brief, bullet-point style summary that highlights the key points, obligations, and regulatory requirements) or detailed (section-by-section breakdown, providing deeper insights into each regulatory clause).
 - Document Summarization Interface
 - The generated summary is displayed on the dashboard in a clear and concise manner. Users can easily scroll through the summary or navigate between sections of the document.

4.4.3 Middle-Layer

4.4.3.1 General Principles

The Middle Layer is comprised of a set of Controllers that allow the configurable/white-labelled part of the EFRA Marketplace offering asset trading functionalities (see 4.4.2.1) to be independent from the specific back end that supports them. The Controllers are the following:

- **User Controller** Responsible for user and wallet management.
- **Dataset Management Controller** Responsible for managing the dataset assets (publishing, editing, acquiring).
- **Dataset Search Controller** Responsible for searching and retrieving datasets under a variety of filters.
- **AI Model Management Controller** Responsible for managing the AI model assets (publishing, editing, acquiring).
- **AI Model Search Controller** Responsible for searching and retrieving AI model assets under a variety of filters.

Across all controllers, the following principles are applied:

- All responses return an HTTP-like status code (3XX for success, 4XX for client error, 5XX for server error)

- A response with status code 4XX or 5XX also returns a non-null error message in the corresponding response parameter
- Authentication is based on a secure authentication token (auth_token). This must be stored securely in the client-side and provided at subsequent calls.
- Any call that requires the auth_token may fail due to the token having expired. This is indicated with a special response code and error message. The front-end layer is responsible to handle this by redirecting to the login page for the user to get a new auth_token.
- All calls must be appropriately wrapped to become asynchronous (the front-end must not block waiting any middle-layer call to finish)

4.4.3.2 User Controller

Table 2: User controller functionalities for the middle layer of the EFRA Marketplace

Name	Description	Request params	Response params
Login	Attempts to identify user using (email, password)	email password	auth_token
Logout	Logs user out and forces the auth_token to become invalid	auth_token	none
Register	Registers a new user and returns an auth_token as in Login. Checks that the email does not already exist.	email password	auth_token
Get Wallet	Returns the number of coins (value) in the user wallet	auth_token	value

4.4.3.3 Dataset Management Controller

Table 3: Data Management controller functionalities for the middle layer of the EFRA Marketplace

Name	Description	Request params	Response params
Download Dataset	Downloads the content of a dataset that has been acquired by the user	auth_token dataset_id	dataset_content dataset_type
Set Dataset Properties	Sets the specified dataset properties and metadata tags to the ones specified in the request parameters. The user must be the owner of the dataset.	auth_token [metadata_tag] Date_range [original_source] [country] [thematic_class] value	none
List Acquired Datasets	Returns a list of the ids of the datasets acquired by the user	auth_token	[dataset_id]
List Owned Datasets	Returns a list of the ids of the datasets	auth_token	[dataset_id]

	owned by the user (i.e., they have uploaded them)		
Get Dataset Properties	Returns the properties of a dataset and other related information (see properties of dataset entity)	dataset_id	[tag] Date_range [original_source] [country] [thematic_class] value
Get Dataset Preview	Returns the top-n rows (or lines) of a dataset acquired or owned by the user; if dataset is not cannot be split into rows, returns the first n*15 characters	auth_token dataset_id n	[row]
Publish Dataset	Makes a dataset available to be acquired by any user. If a positive value is specified in the relevant request parameter, the user wishing to acquire the dataset must pay that many coins from their wallet to acquire it.	auth_token dataset_id value	none
Acquire Dataset	Indicates that the specified dataset has been acquired by the specified user. If the dataset has a positive value, the user must pay that many coins from their wallet first. The request returns successfully if the necessary coins are available and after they have been subtracted from the user wallet.	auth_token dataset_id	none

4.4.3.4 Dataset Search Controller

Table 4: Dataset Search controller functionalities for the middle layer of the EFRA Marketplace

Name	Description	Request params	Response params
Dataset Metadata Tags Search	Returns the ids of available datasets that contain all the specified tags in their metadata	[tag] [dataset_id]	[dataset_id]
Dataset Key Properties Search	Returns the ids of available datasets that contain all the specified values in the corresponding key properties (i.e., the properties of the dataset entity apart from tags, e.g., date range, thematic classes, legal terms, etc.)	[key_property: [value]] [dataset_id]	[dataset_id]
List Thematic Classes	Returns the set of thematic classes available in the database	none	[thematic_class_name]
List Countries	Returns the set of countries available in the database	none	[country_name]
List Original Sources	Returns the set of original sources available in the database	none	[original_source]

4.4.3.5 AI Model Management Controller

Table 5: AI Model Management controller functionalities for the middle layer of the EFRA Marketplace

Name	Description	Request params	Response params
Download Model	Downloads the content of a model that has been acquired by the user	auth_token model_id	model_content file_type
Set Model Properties	Sets the specified model properties and metadata tags to the ones specified in the request parameters. The user must be the owner of the model.	auth_token [metadata_tag] model_description input_description input_format output_description output_format dependencies value	none

List Acquired Models	Returns a list of the ids of the models acquired by the user	auth_token	[model_id]
List Owned Models	Returns a list of the ids of the models owned by the user (i.e., they have uploaded them)	auth_token	[model_id]
Get Model Properties	Returns the properties of a model and other related information (see properties of AI model entity)	model_id	[metadata_tag] model_description input_description input_format output_description output_format dependencies value
Publish Model	Makes a model available to be acquired by any user. If a positive value is specified in the relevant request parameter, the user wishing to acquire the model must pay that many coins from their wallet to acquire it.	auth_token model_id value	none
Acquire Model	Indicates that the specified model has been acquired by the specified user. If the model has a positive value, the user must pay that many coins from their wallet first. The request returns successfully if the necessary coins are available and after they have been subtracted from the user wallet.	auth_token model_id	none

4.4.3.6 AI Model Search Controller

Table 6: AI Model Search controller functionalities for the middle layer of the EFRA Marketplace

Name	Description	Request params	Response params
Model Metadata Tags Search	Returns the ids of available model that contain all the specified tags in their metadata	[tag] [model_id]	[model_id]
Model Key Properties Search	Returns the ids of available model that contain all the specified values in the corresponding key properties (i.e., the properties of the model entity apart from tags)	[key_property: [value]] [model_id]	[model_id]

4.5 EFRA Marketplace Demonstrator

In this section we present a set of high-fidelity designs for the EFRA Marketplace. The relevant demonstrator can be accessed in this link: <https://bit.ly/efra-marketplace> These high-fidelity designs represent the UI/UX part of the system and will guide the front-end development process. These designs focus on the white-labelled part of the EFRA Marketplace, while the EFRA-specific demonstrators (the interactive dashboards described in section 4.4.2.2) are described in D5.4.

Page Content

Figure 4: High-fidelity design for the Core Page Layout

Figure 4 presents the core page layout that remains the same across all pages. The top part is the Navigation Bar that includes drop-down lists for the Search Assets, My Assets, and Decision Support tools, and the My Account page. The bottom part presents the EFRA logo that links to the project website, the social media icons that connect to the social media pages of the project, and at the bottom the EU funding disclaimer.

The Search Assets list includes pointers to Search Datasets and Search Models pages. The My Assets list includes pointers to My Datasets, My Models, Publish Dataset, Publish Model, and Source Request pages. The Decision Support list includes pointers to all the interactive dashboards described in section 4.4.2.2.

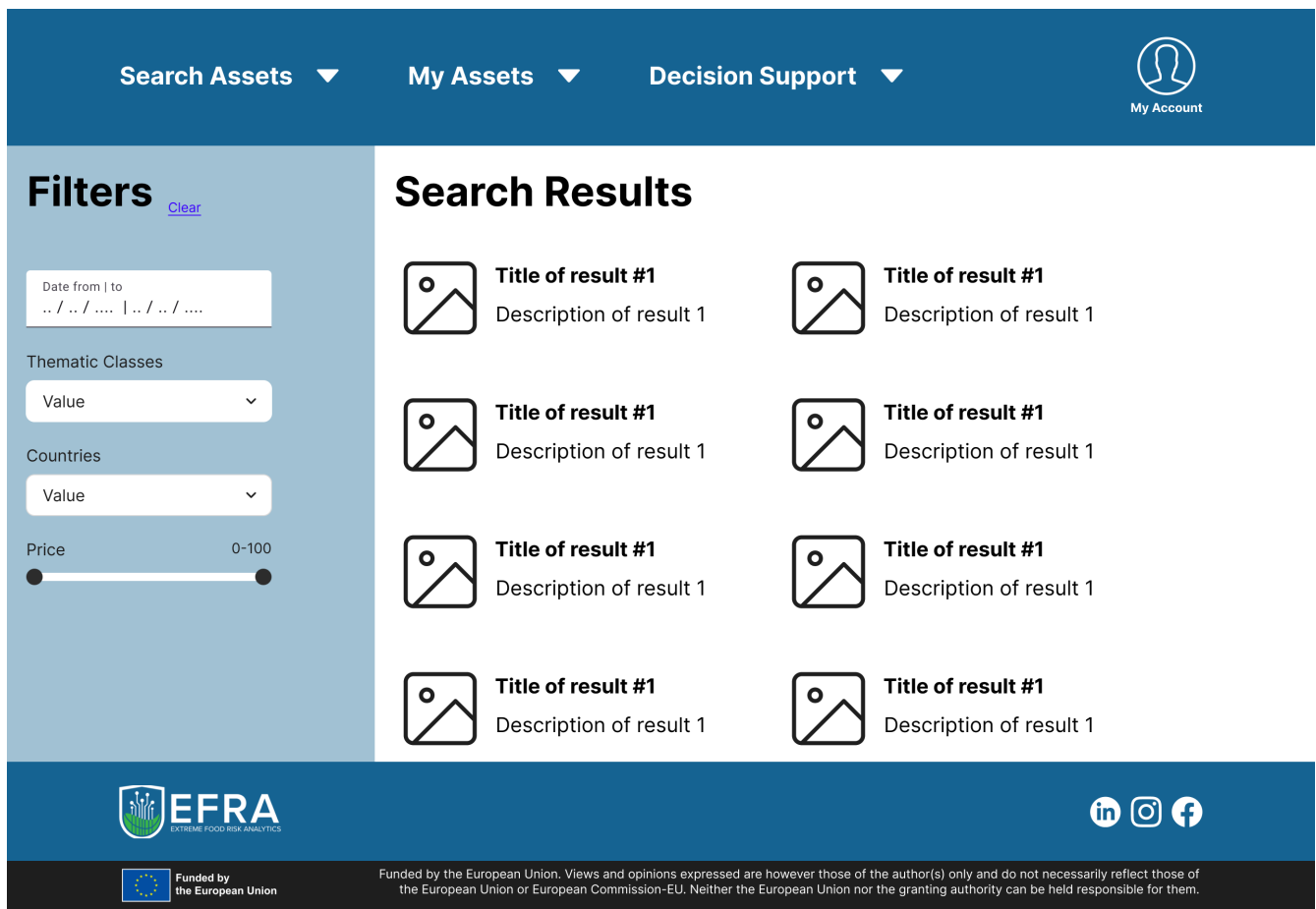


Figure 5: High-fidelity design for the Asset Search (Datasets) page

Figure 5 presents the Asset Search page, focusing on the dataset case (a similar page is available for searching for AI models). On the left-hand side, we see the Filters panel. The user can choose from a set of search filters, including the date covered by the dataset, the thematic classes, the countries, and its minimum and maximum price.

After specifying the filters, the panel on the right-hand side presents the relevant results (datasets in this case). For each result, we can see a representative icon, its title, and part of its description. The results are laid out in a tile format. Choosing a model redirects the user to the Asset Preview & Acquire screen (Figure 6).

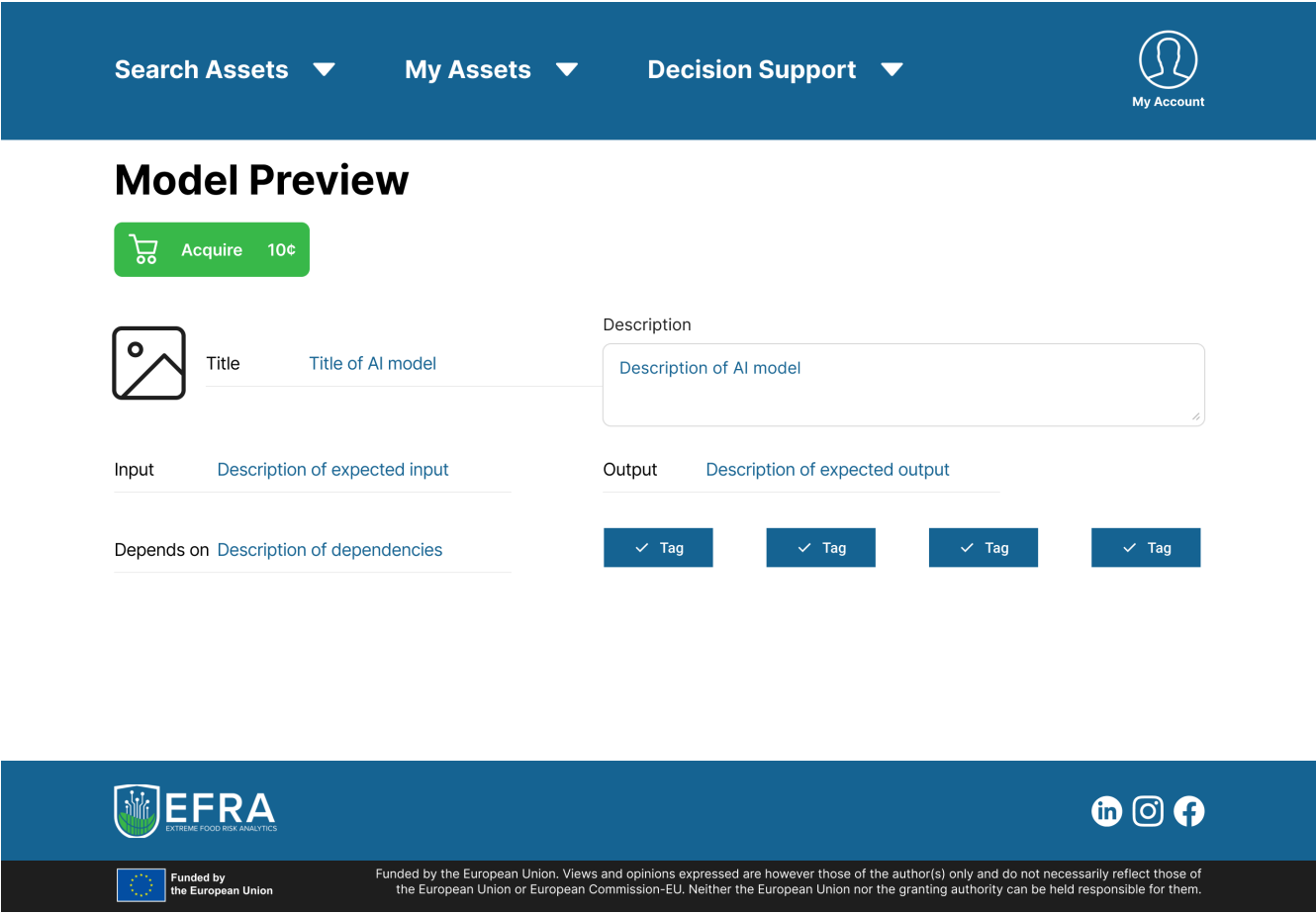


Figure 6: High-fidelity design for the Asset Preview & Acquire (AI Model) page

Figure 6 presents the Asset Preview & Acquire page, focusing on the case of an AI model (a similar page is presented in the case of a dataset asset). In this page, we can see the acquire button that also includes the price of the asset on it. Below, we can see the details of the asset, including a title, a description, the type of input and output expected by the model, any dependencies, and its associated metadata tags.

Clicking on Acquire will subtract the relevant number of coins from the user wallet, add the asset to their My Assets page (Figure 7), and redirect them to the relevant Asset Preview & Download page (Figure 8).

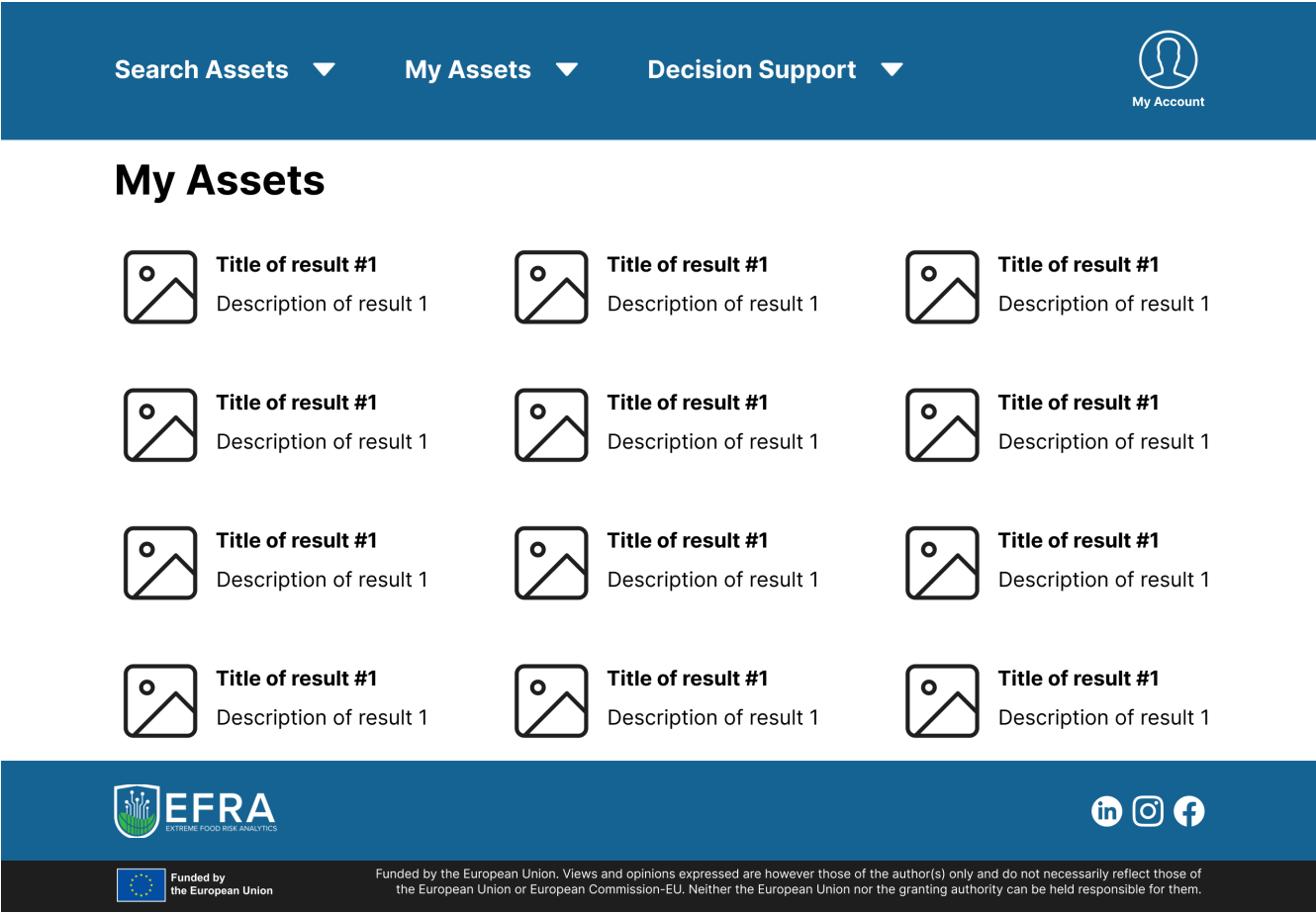


Figure 7: High-fidelity design for My Assets page

Figure 7 presents the My Assets page. Here the user can find all the assets (datasets and AI models) they have acquired or published. For each asset, we can see a representative icon, its title, and part of its description. The assets are laid out in a tiles format.

Clicking on any asset will redirect the user to the relevant Asset Preview & Download page (Figure 8).

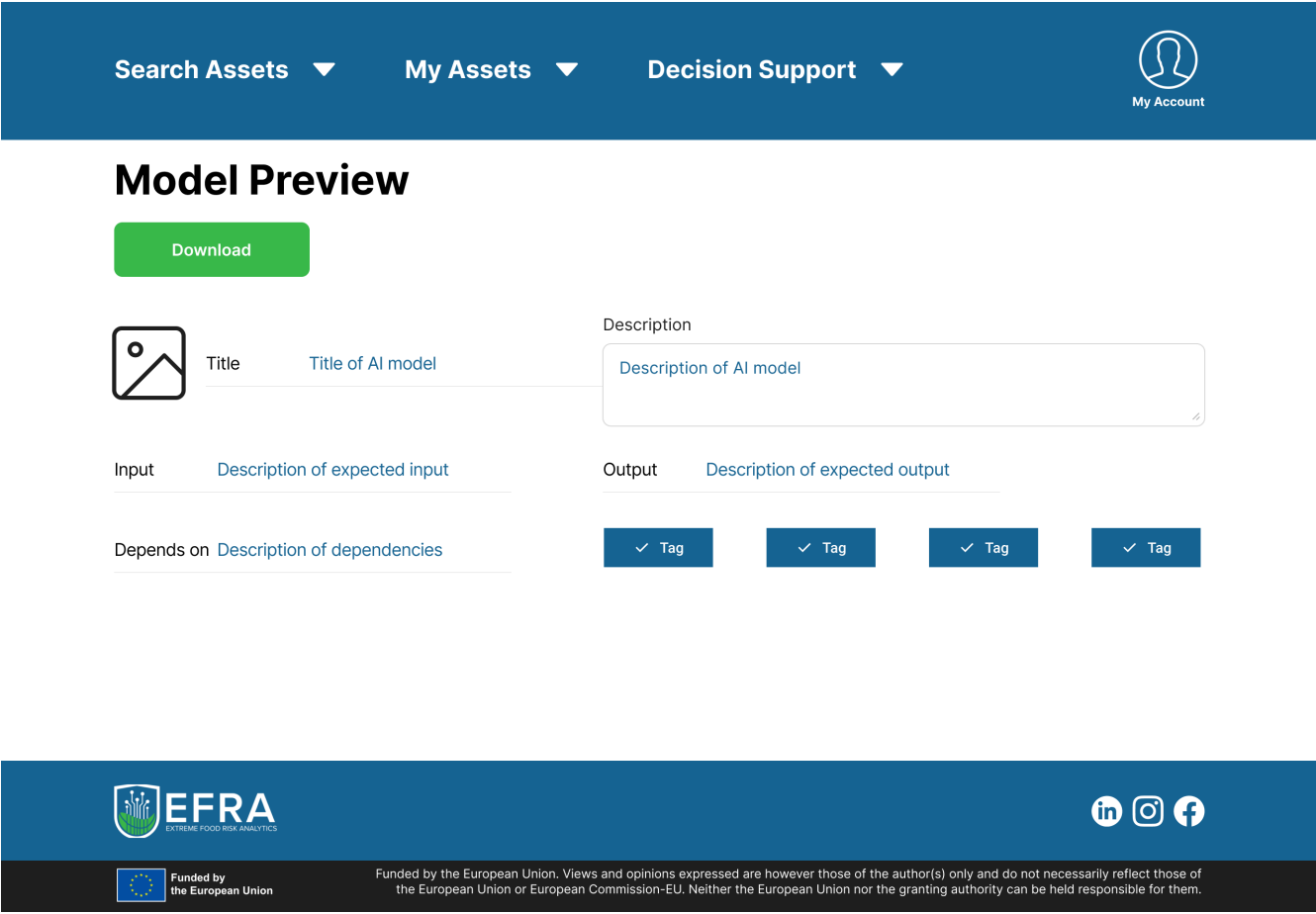


Figure 8: High-fidelity design for the Asset Preview & Download (AI Model) page

Figure 8 presents the Asset Preview & Download page, focusing on the case of an AI model (a similar page exists in the case of a dataset asset). The page is similar to the Asset Preview & Acquire screen but is concerns an asset that the user has already acquired. As a result, the acquire button is substituted for a Download button. If the user clicks on this button, the relevant asset will be downloaded on their device.

Search Assets ▾My Assets ▾Decision Support ▾

My Account

Publish Model

Click to upload asset file

Title

Title of AI model

Description

Description of AI model

Input

Description of expected input

Output

Description of expected output

Depends on

Description of dependencies

Add Tag

Input name of new tag

Price

Price of asset

✓ Tag

✓ Tag

✓ Tag

✓ Tag

Publish

EFRA

EXTREMES FOOD RISK ANALYTICS

in

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission-EU. Neither the European Union nor the granting authority can be held responsible for them.

Figure 9: High-fidelity design for the Publish Asset (AI Model) page

Figure 9 presents the Publish Asset page, focusing on the AI model case (a similar page exists for the dataset asset case). Here the user can upload and share an asset they possess and make it available to be acquired by other users. The page includes a button to upload the asset file, an icon that will represent the dataset (which can be a stock icon, or a custom icon added by the user), a description, the input and output expected by the model, any dependencies, its metadata tags, and its price. After entering all information, the user can click on the Publish button and the asset will be uploaded to the EFRA Platform and made available for acquisition for other users.

© EFRA

Page | 68

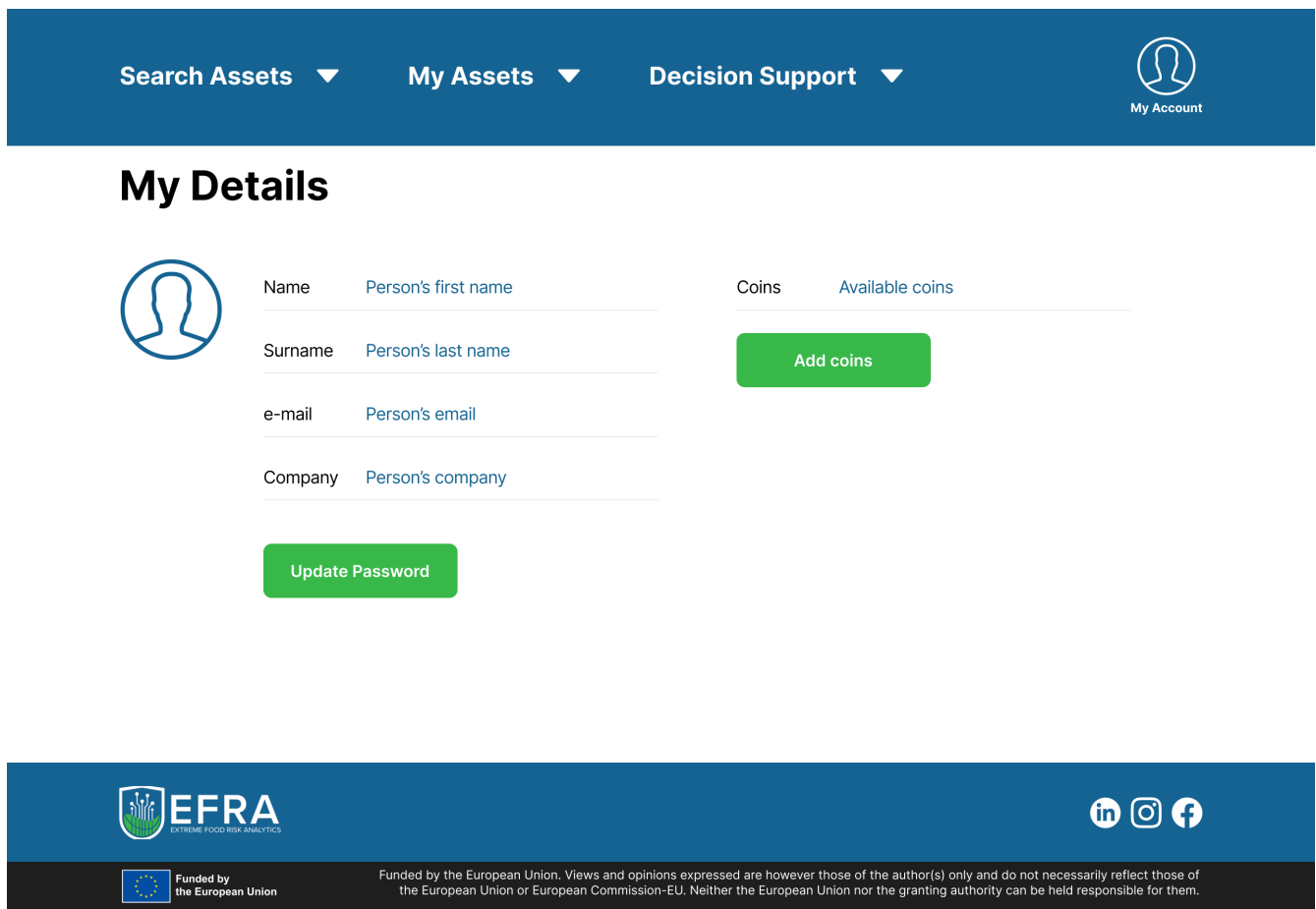


Figure 10: High-fidelity design for My Details page

Figure 10 presents the user details and the coins in their wallet. The user can see their current avatar and click on it to change it. They can also specify their name, surname, email, and company. There is also a button to update their password.

The user can also see their available coins and add coins as necessary. The add coins button will redirect to a third-party service for handling monetary transactions.

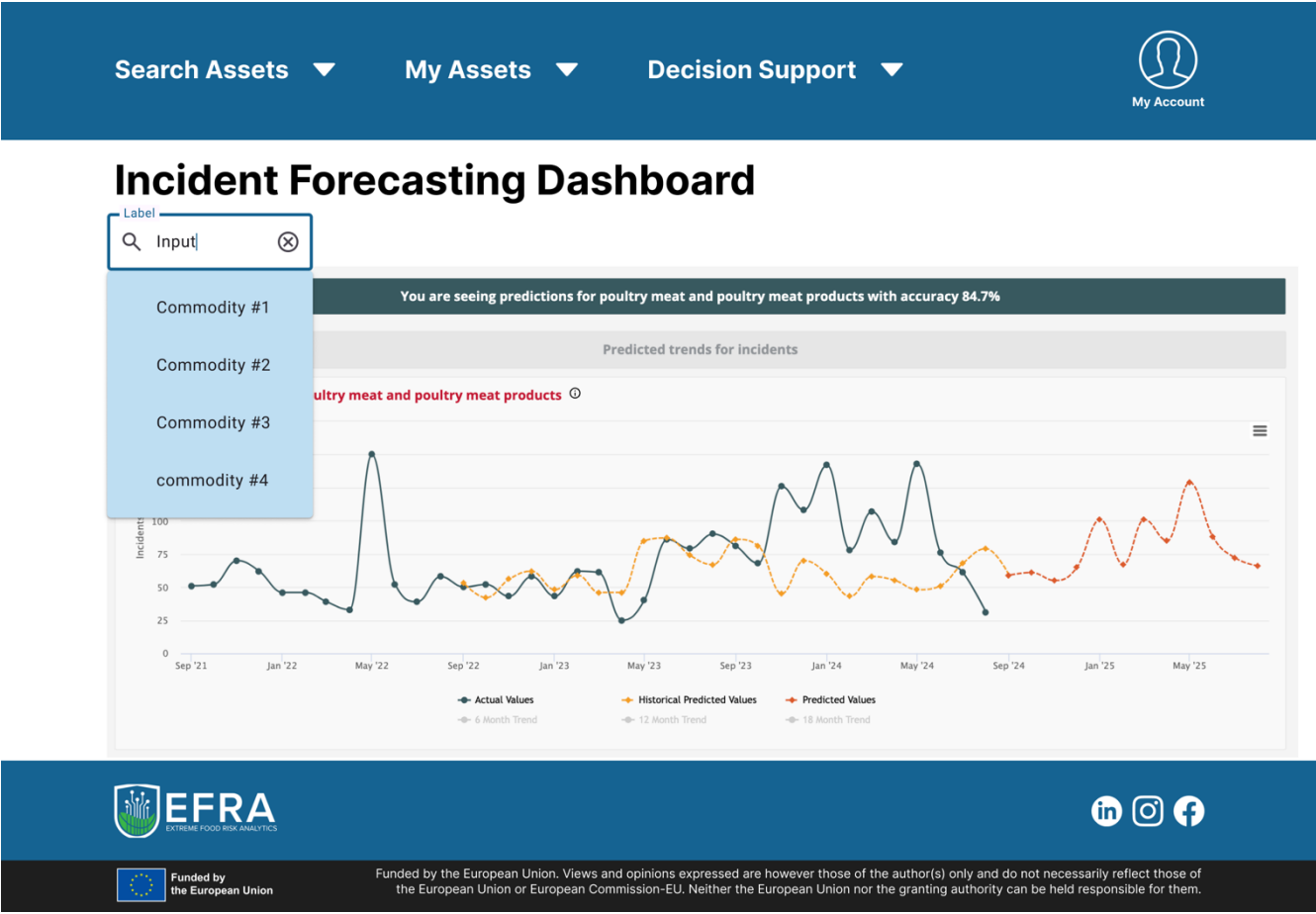


Figure 11: High-fidelity design for Incident Forecasting Dashboard page

Figure 11 presents the Incident Forecasting dashboard page. The user can search for a number of forecasting models available for different food commodities and hazards. Upon their selection, the relevant dashboard appears that presents the historical incidents (black line), forecasted incidents (red line), a forecasted past incidents (orange line) to easily gauge the accuracy of the model.

5 Conclusions

In conclusion, this deliverable outlined the initial designs and specifications of the EFRA API Gateway and the Data & Analytics Marketplace, aligning with the objectives of Task 4.4 of the EFRA project. The API Gateway emerges as an important component, streamlining the integration and accessibility of diverse data and AI models within the EFRA ecosystem for third-party applications. Simultaneously, the Data & Analytics Marketplace, evolving from the BigDataGrapes and TheFSM concepts, establishes itself as a user-friendly platform, enhancing the discoverability and utilization of data and analytics assets. Collectively, these developments mark a significant step towards facilitating advanced food risk analytics, driving the EFRA project's goal of improving global food safety standards through innovative technology and data management practices.