COMPETITIVINESS AND INNOVATION FRAMEWORK PROGRAMME

CIP-ICT-PSP-2013-7 Pilot Type B

WP3 – Service platform integration and deployment in cloud infrastructure

D3.1.1: Heterogeneous data repositories and related services

Deliverable Lead: ATOS

Deliverable due date: 28/02/2015

Actual submission date: 13/04/2015

Version: 1.0

This project is partially funded under the ICT Policy Support Programme (ICT PSP) as part of the Competitiveness and Innovation Framework Programme by the European Commission under grant agreement no. 621074

Document Control Page

Title	D3.1.1: Heterogeneous data repositories and related services
Creator	Miguel Ángel Esbrí (ATOS)
Description	This document introduces the first prototype of the different geospatial data repositories and their associated services
Publisher	FOODIE Consortium
Contributors	
Creation date	28/02/2015
Type	Text
Language	en-GB
Rights	copyright "FOODIE Consortium"
Audience	☐ internal ☒ public ☐ restricted
Review status	☐ Draft ☐ WP leader accepted ☐ Technical Manager accepted ☒ Coordinator accepted
Action requested	☐ to be revised by Partners ☐ for approval by the WP leader ☐ for approval by the Technical Committee ☐ for approval by the Project Coordinator
Requested deadline	

About the project

FOODIE project aims at creating a platform hub on the cloud where spatial and non-spatial data related to agricultural sector is available for agri-food stakeholders groups and interoperable. It will offer: an infrastructure for the building of an interacting and collaborative network; the integration of existing open datasets related to agriculture; data publication and data linking of external agriculture data sources, providing specific and high-value applications and services for the support of planning and decision-making processes.

FOODIE project is addressed to four basic groups of users: a) stakeholders from the agriculture sector as end-users of final applications, b) public sector for communication with farmers about taxation, subsidies and regulation, c) researchers for large scale experimentation on real data and d) ICT companies for the development of new applications for agriculture and food sector, mainly using implemented tools

FOODIE specifically works on three pilots:

- Pilot 1: Precision Viticulture (Spain) will focus on the appropriate management of the inherent variability of crops,
- Pilot 2: Open Data for Strategic and Tactical Planning (Czech Republic) will focus on improving future management of agricultural companies (farms) by introducing new tools and management methods,
- Pilot 3: Technology allows integration of logistics via service providers and farm management including traceability (Germany).

Contact information

Miguel Angel Esbrí

Project Coordinator

Atos Spain, Madrid, Spain

E-mail: miguel.esbri@atos.net

URL: http://www.foodie-project.eu

Twitter: https://twitter.com/FOODIE_Project

Glossary

The glossary of terms used in this deliverable can be found in the public document "FOODIE_Glossary.pdf" available at: http://www.foodie-project.eu

Abbreviations and Acronyms

Abbreviation / Acronym	Description
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
OGC	Open Geospatial Consortium
ORDBMS	Object-Relational Database Management System
RDBMS	Relational Database Management System
RDF	Resource Description Framework
SOS	Sensor Observation Service
WFS	Web Feature Service
WCS	Web Coverage Service
WMS	Web Map Service

Table 1: Abbreviations and Acronyms

Executive Summary

This document introduces the first prototype of different geospatial databases and access services that deployed in FOODIE platform during the first year of the project.

Components and related services overview

As conceptually conceived from the start of the project, FOODIE service platform will comprise a set of diverse repositories necessary for the storage, retrieval and processing of different sources of information such as satellite imagery, cartography (e.g., roads, cadastral information), observations collected from sensors (e.g., meteorological stations provided by farmers) as well as other sources of structured and unstructured VGI data (e.g., individual crop production from farmers, statistics at European level, etc.).

Due to the heterogeneity of the data sources and formats, different storage approaches will be integrated in order to provide an effective, efficient and scalable data storage solution over the time. These will combine the use of files (e.g., for the storage of reports in the form of Excel and Word documents) and relational databases.

In addition, the access to these data repositories will be done through open and standardized interfaces by using OGC services (i.e., WMS, WFS, WCS and SOS) but also through a set of lightweight APIs developed by FOODIE in order to encapsulate the complexity of the OGC ones and facilitate the access to the resources stored in the platform for the non-expert users.

Implementation

In this first stage, efforts where focused on the establishment of a basic set of various database components and access services (that will be increased enhanced in an iterative manner in the next months) from which to start building the rest of capabilities/features offered by FOODIE platform to its users.

As such, in its current status, the data storage and access services comprise the following components (deployed in various VMs for increased performance):

- Databases:
    - Postgres-XL: clusterized version of PostgreSQL database, which enables horizontal scalability, being flexible enough to handle varying database workloads. Used (in combination with PostGIS database extension) for storing vector base geospatial information.
    - Rasdaman database: an array database system, which provides flexible, fast, scalable geo services for multi-dimensional spatio-temporal sensor, image, simulation, and statistics data of unlimited volume. Currently used to store the satellite imagery (in contrast with the typical storage in the file system)
    - Virtuoso: it is a hybrid product that combines the functionality of RDBMS, ORDBMS, XML, RDF, web application server and content management tool. In FOODIE we are mainly interested in the RDF store for the provisioning of a semantic database (triplestore) as a service for the storage of semantic annotations and other RDF data, as well as for their publication as linked data.

For further information about the deployment details of these database components, please refer to deliverable D3.6.1 Deployment and Integration Report.

- Services:
  - OGC Web Map Service (WMS): The service provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc.) that can be displayed.

    The service is provided through the following open-source software implementations: Mapserver, Geoserver and Rasdaman Petascope WMS plugin.
  - OGC Web Feature Service (WFS): The basic Web Feature Service allows querying and retrieval of features. A transactional Web Feature Service (WFS-T) allows creation, deletion, and updating of features.

    The service is provided through the following open-source software implementations: Mapserver, for non-transactional operations and Geoserver for transactional operations – insert, delete, update features in the geospatial database.
  - OGC Web Coverage (WCS): provides a client access to coverages – that is, digital geospatial information representing space/time-varying phenomena (typically represented as raster or gridded data, being common for satellite imagery).

    The service is provided through the following open-source software implementations: Mapserver, Geoserver and Rasdaman Petascope WCS plugin.
  - Orion Context Broker: It is a FI-WARE Generic Enabler that provides RESTful API to managing context information. Orion provides methods to register and discover context producers and provides mechanism to notify changes on context information. Orion implements NGSI9/NGSI10 protocols that are used as standards in FI-WARE.

    Orion is provided as standalone installation and runs as a backend service daemon.
  - OGC Sensor Observation Service (SOS): provides access to sensor data and provides querying and retrieval of observations and sensor system description. The requests and responses are in form of XML documents defined by OGC standard. The OGC SOS service is suitable for any type of sensor systems.

    The service is provided through the following open-source software implementations: MapLog and SensLog.
  - Data Management Service: this service allows uploading geospatial information – through its REST interface – to different types of data repositories (e.g., file system, databases) and publishing its associated metadata in an OGC compliant catalogue as well as publishing the data as a new OGC WMS and WFS layer. Besides, the service provides a web user interface, which simplifies for the non-expert user the data upload and layer management process.

    The service is provided through the following open-source software implementations: LayMan (the Layer Manager) and LayMan Client.

For further information about the deployment details of these service components, please refer to deliverable D3.6.1 Deployment and Integration Report.

In addition to the above databases and services, two complementary back office services/tools have been implemented for supporting the maintenance of the spatial repositories:

- Data Harvester component: used for collecting – in a periodic manner - external data sources which are used to feed FOODIE data repositories. The component is deployed on Apache Tomcat server and in its current version is able to download LANDSAT 8 imagery[1] (used in a later stage for performing other operations with them, such as vegetation, water or moisture index extraction) for the whole country areas of Portugal, Spain, Czech Republic and Germany.
- Satellite imagery pre-processing scripts: used for performing (once the LANDSAT-8 imagery is downloaded) some sanitation operations in the raw raster data previous to be usable by the processing services. The scripts are implemented using Python and relies on GRASS software for performing the specialized operations on the satellite imagery:
    - i.landsat.toar → convert DN to top of atmosphere radiance
    - i.colors.enhance (GRASS 7.x) → auto-enhance colors
    - i.landsat.acca → cloud cover assessment

API details

- OGC Web Map Service (WMS): OpenGIS Web Map Service (WMS) Implementation Specification (http://portal.opengeospatial.org/files/?artifact_id=14416)
- OGC Web Feature Service (WFS): OGC Web Feature Service 2.0 Interface Standard (https://portal.opengeospatial.org/files/09-025r2)
- OGC Web Coverage (WCS): OGC WCS 2.0 Interface Standard - Core, version 2.0.1 (https://portal.opengeospatial.org/files/09-110r4)
- OGC Sensor Observation Service (SOS): OGC Sensor Observation Service 1.0 Interface Standard (http://portal.opengeospatial.org/files/?artifact_id=26667)
- Data Harvester: no plans during the first year for having an API
- Data Management Service: See FOODIE Open API specification [2].
- MapLog: See FOODIE Open API specification [2].
- Orion Context Broker: See FOODIE Open API specification [2]

[1] From EarthExplorer: http://earthexplorer.usgs.gov/

Datasets available in FOODIE repositories:

- LANDSAT-8 imagery for the Spanish, Czech Republic and German pilot areas
- Statistics from Eurostat in RDF format (virtuoso)
- Road Network Model for all Europe (taken from OSM and made INSPIRE compliant)
- Czech Republic pilot
    - LPIS data
    - Machinery Tracking measurements
    - Weather Stations observations
- Spanish pilot:
    - related to FOODIE Data Model: Treatments, Treatment Plans, Fertilizers, Plots, Issues, Annotations, Crop Production, Prunes
    - Weather Stations observations
- German pilot
    - Cadaster data
    - LPIS data

LANDSAT-8 storage details in Rasdaman

The following data type has been defined in Rasdaman database in order to store the LANDSAT-8 band information:

```
## definition of landsat8 type (create a lansat8.dl file)
struct Landsat8Pixel {unsigned short coastal, blue, green, red, nir, swir1, swir2, pan, cirrus, tirs1, tirs2; };
typedef marray <Landsat8Pixel, [*:*,*:*]> Landsat8Image;
typedef set <Landsat8Image> Landsat8ImageSet;
```

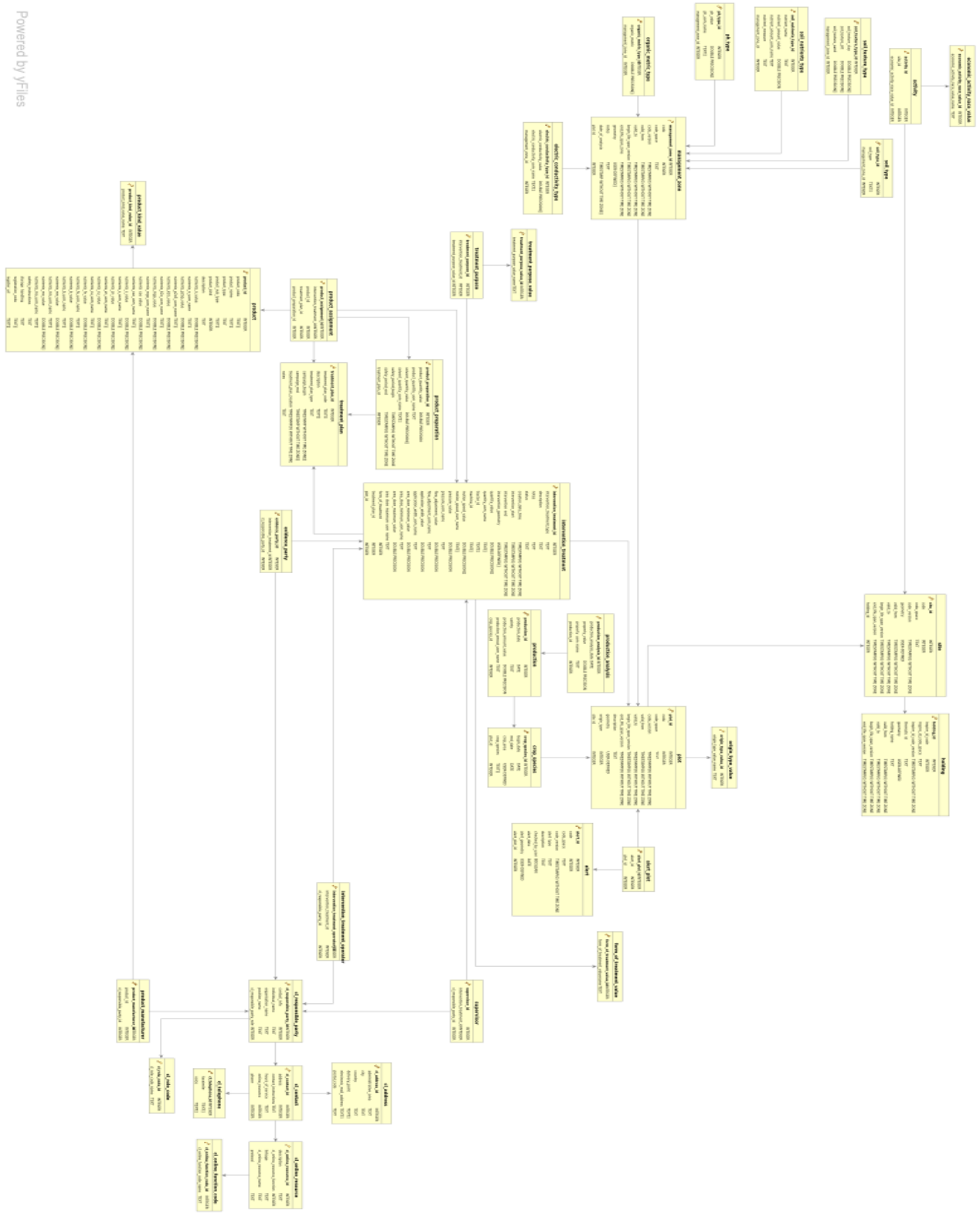
The data type is inserted in the type definition table using the following command from Rasdaman: “rasdl --basename RASBASE --read landsat8.dl –insert”

The following is an example on how to insert an example landsat 8 image using the rasql command provided by Rasdaman (once the former command has been successfully executed in order to create the LANDSAT-8 definition in the database):

“rasql -q 'insert into LANDSAT values inv_tiff($1, "sampletype=ushort")' -f /usr/local/rasdaman/share/rasdaman/examples/images/landsat_untiled_subset_small_nh.tif --user rasadmin --passwd rasadmin”

FOODIE Data Model storage details

The following UML class diagram and associated SQL schema shows how FOODIE UML Data Model (please, refer to UML diagram provided in the Information Viewpoint of FOODIE Architecture [3]) has been adapted to Postgres database.



```
CREATE SCHEMA foodie AUTHORIZATION postgres;

set search_path to foodie, public;

CREATE TABLE product_manofacturer (
        product_manofacturer_id SERIAL,
        product_id integer,
        cl_responsible_party_id integer
);

CREATE TABLE active_ingredients (
        active_ingredients_id SERIAL,
        code integer,
        code_space text,
        code_version timestamp(0) without time zone,
        ingreient_name text,
        ingredient_amount_value double precision,
        ingredient_amount_oum_name text,
        product_id integer
);

create table evidence_party(
        evidence_party_id SERIAL,
        intervention_treatment_id integer,
        cl_responsible_party_id integer
        );

create table intervention_treatment_operator(
        intervention_treatment_operator_id SERIAL,
        intervention_treatment_id integer,
        cl_responsible_party_id integer
        );

create table supervisor(
        supervisor_id SERIAL,
        intervention_treatment_id integer,
        cl_responsible_party_id integer
        );

create table economic_activity_nace_value(
        economic_activity_nace_value_id integer,
        economic_activity_nace_value_name text
        );

create table activity(
        activity_id integer,
        site_id integer,
        economic_activity_nace_value_id integer
        );

create table cl_responsible_party(
        cl_responsible_party_id SERIAL,
        contact_info INTEGER,
        individual_name TEXT,
        organisation_name TEXT,
        position_name TEXT,
        cl_responsible_party_role INTEGER
        );

create table cl_contact(
```

```
        cl_contact_id SERIAL,
        address INTEGER,
        contact_instructions text,
        hours_of_service text,
        online_resource INTEGER,
        phone INTEGER
        );

create table cl_telephone(
        cl_telephone_id SERIAL,
        facsimile TEXT[],
        voice TEXT[]
        );

create table cl_address(
        cl_address_id SERIAL,
        administrative_area text,
        city text,
        country TEXT,
        delivery_point text [],
        electronic_mail_address TEXT[],
        postal_code TEXT
        );

create table cl_online_resource(
        cl_online_resource_id INTEGER,
        description text,
        cl_online_resource_function INTEGER,
        linkage text,
        cl_online_resource_name TEXT,
        protocol TEXT
        );

create table cl_online_function_code(
        cl_online_function_code_id INTEGER,
        cl_online_function_code_name TEXT
        );

create table cl_role_code(
        cl_role_code_id INTEGER,
        cl_role_code_name TEXT
        );

create table product_kind_value(
        product_kind_value_id INTEGER,
        product_kind_value_name TEXT
        );

create table treatment_purpose(
        treatment_purpose_id INTEGER,
        intervention_treatment_id INTEGER,
        treatment_purpose_value_id INTEGER
        );

create table treatment_purpose_value(
        treatment_purpose_value_id INTEGER,
        treatment_purpose_value_name TEXT
        );
create table form_of_treatment_value(
        form_of_treatment_value_id INTEGER,
```

```
        form_of_treatment_valuename TEXT
        );

create table soil_type(
        soil_type_id SERIAL,
        soil_type TEXT [],
        management_zone_id INTEGER
        );

create table soil_texture_type(
        soil_texture_type_id SERIAL,
        soil_texture_clay double precision [],
        soil_texture_silt double precision [],
        soil_texture_sand  double precision [],
        management_zone_id INTEGER
        );

create table electric_conductivity_type(
        electric_conductivity_type_id SERIAL,
        electric_conductivity_value double precision [],
        electric_conductivity_uom_name TEXT [],
        management_zone_id INTEGER
        );

create table soil_nutrients_type(
        soil_nutrients_type_id SERIAL,
        nutrient_name TEXT,
        nutrient_amount_value double precision,
        nutrient_amount_uom_name TEXT,
        nutrient_measure TEXT,
        management_zone_id INTEGER
        );

create table organic_metric_type(
        organic_metric_type_id SERIAL,
        organic_metric double precision[],
        management_zone_id INTEGER
        );

create table ph_type(
        ph_type_id SERIAL,
        ph_value double precision[],
        ph_uom_name TEXT[],
        management_zone_id INTEGER
        );

create table management_zone(
        management_zone_id SERIAL,
        code INTEGER,
        code_space TEXT,
        code_version timestamp(0) without time zone,
        valid_from timestamp(0) without time zone,
        valid_to timestamp(0) without time zone,
        begin_life_span_version timestamp(0) without time zone,
        end_life_span_zone timestamp(0) without time zone,
        geometry geometry[],
        notes text,
        date_of_analysis timestamp(0) without time zone [],
        plot_id INTEGER
        );
```

```
create table product_preparation(
        product_preparation_id SERIAL,
        product_quantity_value double precision,
        product_quantity_uom_name TEXT,
        solvent_quantity_value double precision[],
        solvent_quantity_uom_name TEXT[],
        safety_period_begin timestamp(0) without time zone,
        safety_period_end timestamp(0) without time zone,
        treatment_plan_id  INTEGER
        );

create table treatment_plan(
        treatment_plan_id SERIAL,
        treatment_plan_code TEXT[],
        description text[],
        treatment_plan_type TEXT,
        campaign_begin timestamp(0) without time zone[],
        campaign_end timestamp(0) without time zone[],
        treatment_plan_creation timestamp(0) without time zone,
        notes text
        );


CREATE TABLE product_assignment (
  product_assignment_id SERIAL,
  intervention_treatment_id INTEGER,
  product_id INTEGER,
  treatment_plan_id INTEGER,
  product_preparation_id INTEGER
);

CREATE TABLE alert (
        alert_id INTEGER,
        code INTEGER,
        code_space TEXT,
        code_version timestamp(0),
        alert_type TEXT,
        description text,
        checked_by_user boolean,
        alert_date date,
        alert_geometry geometry,
        alert_plot_id INTEGER
);

create table holding (
        holding_id SERIAL,
        inspire_id_code INTEGER,
        inspire_id_code_space TEXT,
        inspire_id_code_version timestamp(0) without time zone,
        thematic_id text,
        geometry geometry,
        holding_name TEXT,
        valid_from timestamp(0) without time zone,
        valid_to timestamp(0) without time zone,
        begin_life_span_version timestamp(0) without time zone,
        end_life_span_version timestamp(0) without time zone
        );
```

```
CREATE TABLE alert_plot (
        alert_plot_id SERIAL,
        alert_id INTEGER,
        plot_id INTEGER
);

create table site(
        site_id SERIAL,
        code INTEGER,
        code_space TEXT,
        code_version timestamp(0) without time zone,
        geometry geometry,
        valid_from timestamp(0) without time zone,
        valid_to timestamp(0) without time zone,
        begin_life_span_version timestamp(0) without time zone,
        end_life_span_version timestamp(0) without time zone,
        holding_id INTEGER
        );

CREATE TABLE crop_species (
        crop_species_id INTEGER NOT NULL,
        begin_date date,
        end_date date,
        crop_area geometry,
        crop_species TEXT[],
        plot_id INTEGER
);

CREATE TABLE intervention_treatment
( intervention_treatment_id SERIAL,
 intervention_treatment_type TEXT,
 description text,
 notes text,
 status TEXT,
 creation_date_time timestamp(0) without time zone,
 intervention_start timestamp(0) without time zone,
 intervention_end timestamp(0) without time zone,
 intervention_geometry geometry[],
 quantity_value double precision[],
 quantity_uom_name TEXT[],
 tractor_id TEXT[],
 machine_id TEXT[],
 motion_speed_value double precision[],
 motion_speed_oum_name TEXT[],
 pressure_value double precision,
 pressure_uom_name TEXT,
 flow_adjustment_value double precision,
 flow_adjustment_uom_name TEXT,
 application_width_value double precision,
 application_width_uom_name TEXT,
 area_dose_minimum_value double precision,
 area_dose_minimum_uom_name TEXT,
 area_dose_maximum_value double precision,
 area_dose_maximum_uom_name TEXT,
 form_of_treatment INTEGER,
 treatment_plan_id INTEGER,
 plot_id INTEGER
 );

CREATE TABLE origin_type_value (
```

```
        origin_type_value_id INTEGER NOT NULL,
        origin_type_value_name TEXT
);

CREATE TABLE plot (
        plot_id SERIAL,
        code INTEGER,
        code_space TEXT,
        code_version timestamp(0),
        valid_from timestamp(0)without time zone,
        valid_to timestamp(0)without time zone,
        begin_life_span_version timestamp(0)without time zone,
        end_life_span_version timestamp(0)without time zone,
        description text,
        geometry geometry,
        origin_type INTEGER,
        site_id INTEGER
);

CREATE TABLE production (
        production_id SERIAL,
        production_date date,
        variety TEXT,
        production_amount_value double precision,
        production_amout_uom_name TEXT,
        crop_species_id INTEGER
);

CREATE TABLE production_analysis (
        production_analysis_id INTEGER NOT NULL,
        production_analysis_date date,
        property_value double precision,
        property_uom_name TEXT,
        production_id INTEGER
);

create table product(product_id SERIAL,
        product_code TEXT[],
        product_name TEXT[],
        product_type TEXT,
        product_sub_type TEXT[],
        product_kind INTEGER,
        description text,
        nutrients_n_value double precision[],
        nutrients_n_uom_name TEXT[],
        nutrients_p2op_value double precision[],
        nutrients_p2o5_uom_name TEXT[],
        nutrients_k2o_value double precision[],
        nutrients_k2o_uom_name TEXT[],
        nutrients_mgo_value double precision[],
        nutrients_mgo_uom_name TEXT[],
        nutrients_cao_value double precision[],
        nutrients_cao_uom_name TEXT[],
        nutrients_s_value double precision[],
        nutrients_s_uom_name TEXT[],
        nutrients_zn_value double precision[],
        nutrients_zn_uom_name TEXT[],
        nutrients_cu_value double precision[],
        nutrients_cu_uom_name TEXT[],
        nutrients_fe_value double precision[],
```

```
        nutrients_fe_uom_name TEXT[],
        nutrients_b_value double precision[],
        nutrients_b_uom_name TEXT[],
        nutrients_mn_value double precision[],
        nutrients_mn_uom_name TEXT[],
        nutrients_mo_value double precision[],
        nutrients_mo_uom_name TEXT[],
        safety_instructions text,
        storage_handling text,
        registration_code text[],
        register_url text[]
        );

ALTER TABLE active_ingredients add CONSTRAINT PK_active_ingredients
        PRIMARY KEY (active_ingredients_id);

ALTER TABLE product_manofacturer add CONSTRAINT pk_product_manofacturer
        PRIMARY KEY (product_manofacturer_id);

ALTER TABLE intervention_treatment_operator add CONSTRAINT pk_intervention_treatment_operator
        PRIMARY KEY (intervention_treatment_operator_id);

ALTER TABLE supervisor add CONSTRAINT pk_supervisor
        PRIMARY KEY (supervisor_id);

ALTER TABLE evidence_party add CONSTRAINT pk_evidence_party
        PRIMARY KEY (evidence_party_id);

ALTER TABLE economic_activity_nace_value add CONSTRAINT pk_economic_activity_nace_value
        PRIMARY KEY (economic_activity_nace_value_id);

ALTER TABLE activity add CONSTRAINT pk_activity
        PRIMARY KEY (activity_id);

ALTER TABLE cl_responsible_party add CONSTRAINT pk_cl_responsible_party
        PRIMARY KEY (cl_responsible_party_id);

ALTER TABLE cl_contact add CONSTRAINT pk_cl_contact
        PRIMARY KEY (cl_contact_id);

ALTER TABLE cl_telephone add CONSTRAINT pk_cl_telephone
        PRIMARY KEY (cl_telephone_id);

ALTER TABLE cl_address add CONSTRAINT pk_cl_address
        PRIMARY KEY (cl_address_id);

ALTER TABLE cl_online_resource add CONSTRAINT pk_cl_conline_resource
        PRIMARY KEY (cl_online_resource_id);

ALTER TABLE cl_online_function_code add CONSTRAINT pk_cl_online_function_code
        PRIMARY KEY (cl_online_function_code_id);

ALTER TABLE cl_role_code add CONSTRAINT pk_cl_role_code
        PRIMARY KEY (cl_role_code_id);

ALTER TABLE product_kind_value add CONSTRAINT pk_product_kind_value_id
        PRIMARY KEY (product_kind_value_id);

ALTER TABLE treatment_purpose ADD CONSTRAINT pk_treatment_purpose
        PRIMARY KEY (treatment_purpose_id);
```

```
ALTER TABLE treatment_purpose_value ADD CONSTRAINT pk_treatment_purpose_value
        PRIMARY KEY (treatment_purpose_value_id);

ALTER TABLE form_of_treatment_value ADD CONSTRAINT pk_form_of_treatment_value
        PRIMARY KEY (form_of_treatment_value_id);

ALTER TABLE soil_texture_type ADD CONSTRAINT pk_soil_texture_type
        PRIMARY KEY (soil_texture_type_id);

ALTER TABLE soil_type ADD CONSTRAINT pk_soil_type
        PRIMARY KEY (soil_type_id);

ALTER TABLE electric_conductivity_type ADD CONSTRAINT pk_electric_conductivity_type
        PRIMARY KEY (electric_conductivity_type_id);

ALTER TABLE soil_nutrients_type ADD CONSTRAINT pk_soil_nutrients_type
        PRIMARY KEY (soil_nutrients_type_id);

ALTER TABLE organic_metric_type ADD CONSTRAINT pk_organic_metric_type
        PRIMARY KEY (organic_metric_type_id);

ALTER TABLE ph_type ADD CONSTRAINT pk_ph_type
        PRIMARY KEY (ph_type_id);

ALTER TABLE management_zone ADD CONSTRAINT pk_management_zone
        PRIMARY KEY (management_zone_id);

ALTER TABLE product_preparation ADD CONSTRAINT pk_product_preparation
        PRIMARY KEY (product_preparation_id);

ALTER TABLE treatment_plan ADD CONSTRAINT pk_treatment_plan
        PRIMARY KEY (treatment_plan_id);


ALTER TABLE product ADD CONSTRAINT pk_product
        PRIMARY KEY (product_id);

ALTER TABLE product_assignment ADD CONSTRAINT pk_product_assignment
        PRIMARY KEY (product_assignment_id);

ALTER TABLE alert_plot ADD CONSTRAINT pk_alert_plot
        PRIMARY KEY (alert_plot_id);

ALTER TABLE holding ADD CONSTRAINT pk_holding
        PRIMARY KEY (holding_id);

ALTER TABLE alert ADD CONSTRAINT pk_Alert
        PRIMARY KEY (alert_id);

ALTER TABLE crop_species ADD CONSTRAINT pk_crop_species
        PRIMARY KEY (crop_species_id);
ALTER TABLE intervention_treatment ADD CONSTRAINT pk_Intervention_treatment
        PRIMARY KEY (intervention_treatment_id);

ALTER TABLE origin_type_value ADD CONSTRAINT pk_origin_type_value
        PRIMARY KEY (origin_type_value_id);

ALTER TABLE plot ADD CONSTRAINT pk_plot
        PRIMARY KEY (plot_id);
```

```
ALTER TABLE site ADD CONSTRAINT pk_site
        PRIMARY KEY (site_id);

ALTER TABLE production ADD CONSTRAINT pk_production
        PRIMARY KEY (production_id);

ALTER TABLE active_ingredients  ADD CONSTRAINT active_ingredients_product FOREIGN KEY (product_id)
    REFERENCES product(product_id);

ALTER TABLE product_manofacturer  ADD CONSTRAINT product_manofacturer_product FOREIGN KEY (product_id)
    REFERENCES product(product_id);

ALTER TABLE product_manofacturer ADD CONSTRAINT product_manofacturer_cl_responsible_party FOREIGN KEY
(cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE evidence_party  ADD CONSTRAINT fk_evidence_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE supervisor  ADD CONSTRAINT fk_supervisor_cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE intervention_treatment_operator  ADD CONSTRAINT fk_intervention_treatment_operator_cl_responsible_party
FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE intervention_treatment_operator  ADD CONSTRAINT fk_intervention_treatment_operator_intervention_treatment
FOREIGN KEY (intervention_treatment_id)
    REFERENCES intervention_treatment(intervention_treatment_id);

ALTER TABLE evidence_party  ADD CONSTRAINT fk_evidence_party_intervention_treatment FOREIGN KEY (interven-
tion_treatment_id)
    REFERENCES intervention_treatment(intervention_treatment_id);

ALTER TABLE supervisor  ADD CONSTRAINT fk_supervisor_intervention_treatment FOREIGN KEY (intervention_treatment_id)
    REFERENCES intervention_treatment(intervention_treatment_id);

ALTER TABLE activity ADD CONSTRAINT fk_activity_economic_activity_nace_value FOREIGN KEY (econom-
ic_activity_nace_value_id)
    REFERENCES economic_activity_nace_value(economic_activity_nace_value_id);

ALTER TABLE activity  ADD CONSTRAINT fk_activity_site FOREIGN KEY (site_id)
    REFERENCES site(site_id);

ALTER TABLE production_analysis ADD CONSTRAINT pk_production_analysis
        PRIMARY KEY (production_analysis_id);

ALTER TABLE cl_responsible_party ADD CONSTRAINT fk_cl_responsible_party_cl_role_code FOREIGN KEY
(cl_responsible_party_role)
    REFERENCES cl_role_code(cl_role_code_id);
ALTER TABLE cl_responsible_party  ADD CONSTRAINT fk_cl_responsible_party_cl_contact FOREIGN KEY (contact_info)
    REFERENCES cl_contact(cl_contact_id);

ALTER TABLE cl_contact  ADD CONSTRAINT fk_cl_contact_cl_online_resource FOREIGN KEY (online_resource)
    REFERENCES cl_online_resource(cl_online_resource_id);

ALTER TABLE cl_contact  ADD CONSTRAINT fk_cl_contact_cl_address FOREIGN KEY (address)
    REFERENCES cl_address(cl_address_id);
```

```
ALTER TABLE cl_contact  ADD CONSTRAINT fk_cl_contact_cl_phone FOREIGN KEY (phone)
    REFERENCES cl_telephone(cl_telephone_id);

ALTER TABLE product  ADD CONSTRAINT fk_product_product_kind_value FOREIGN KEY (product_kind)
    REFERENCES product_kind_value(product_kind_value_id);

ALTER TABLE cl_online_resource  ADD CONSTRAINT fk_cl_online_resource_cl_online_function_code FOREIGN KEY
(cl_online_resource_function)
    REFERENCES cl_online_function_code(cl_online_function_code_id);

ALTER TABLE treatment_purpose  ADD CONSTRAINT fk_treatment_purpose2 FOREIGN KEY (treatment_purpose_value_id)
    REFERENCES treatment_purpose_value(treatment_purpose_value_id);

ALTER TABLE treatment_purpose  ADD CONSTRAINT fk_treatment_purpose1 FOREIGN KEY (intervention_treatment_id)
    REFERENCES intervention_treatment(intervention_treatment_id);

ALTER TABLE intervention_treatment  ADD CONSTRAINT fk_intervention_treatment_form_of_treatment_value FOREIGN KEY
(form_of_treatment)
    REFERENCES form_of_treatment_value(form_of_treatment_value_id);

ALTER TABLE organic_metric_type
 ADD CONSTRAINT fk_organic_metric_type_management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone (management_zone_id);

ALTER TABLE soil_type  ADD CONSTRAINT fk_soil_type_management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE management_zone  ADD CONSTRAINT fk_management_zone_plot FOREIGN KEY (plot_id)
    REFERENCES plot(plot_id);

ALTER TABLE soil_texture_type  ADD CONSTRAINT fk_soil_texture_type_management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone(management_zone_id);


ALTER TABLE electric_conductivity_type  ADD CONSTRAINT fk_electric_conductivity_type_management_zone FOREIGN KEY (man-
agement_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE soil_nutrients_type  ADD CONSTRAINT fk_soil_nutrients_type_management_zone FOREIGN KEY (manage-
ment_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE ph_type  ADD CONSTRAINT fk_ph_type_management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE product_assignment
 ADD CONSTRAINT fk_product_assignment_product_preparation FOREIGN KEY (product_preparation_id)
    REFERENCES product_preparation (product_preparation_id);

ALTER TABLE production
 ADD CONSTRAINT fk_production_crop_species FOREIGN KEY (crop_species_id)
    REFERENCES crop_species (crop_species_id);

ALTER TABLE product_preparation
 ADD CONSTRAINT fk_product_preparation_treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);

ALTER TABLE intervention_treatment
 ADD CONSTRAINT fk_intervention_treatment_treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);
```

```
ALTER TABLE product_assignment
 ADD CONSTRAINT fk_product_assignment_treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);

ALTER TABLE product_assignment
 ADD CONSTRAINT fk_product_assignment_product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

ALTER TABLE product_assignment
 ADD CONSTRAINT fk_product_assignment_intervention_treatment FOREIGN KEY (intervention_treatment_id)
    REFERENCES intervention_treatment (intervention_treatment_id);

ALTER TABLE intervention_treatment
 ADD CONSTRAINT fk_intervention_treatement_plot FOREIGN KEY (plot_id)
    REFERENCES plot (plot_id);

ALTER TABLE site
 ADD CONSTRAINT fk_site_holding FOREIGN KEY (holding_id)
    REFERENCES holding (holding_id);

ALTER TABLE alert_plot ADD CONSTRAINT fk_alert_plot_alert
            FOREIGN KEY (alert_id) REFERENCES alert (alert_id);

ALTER TABLE alert_plot ADD CONSTRAINT fk_alert_plot_plot
            FOREIGN KEY (plot_id) REFERENCES plot (plot_id);

ALTER TABLE crop_species ADD CONSTRAINT fk_crop_species_plot
            FOREIGN KEY (plot_id) REFERENCES plot (plot_id);

ALTER TABLE plot ADD CONSTRAINT fk_plot_origin_type_value
            FOREIGN KEY (origin_type) REFERENCES origin_type_value (origin_type_value_id);

ALTER TABLE production_analysis ADD CONSTRAINT fk_production_analysis_production
            FOREIGN KEY (production_id) REFERENCES production (production_id);

ALTER TABLE plot
 ADD CONSTRAINT fk_plot_site FOREIGN KEY (site_id)
    REFERENCES site (site_id);
```

References

	References
01	Palma R., et al "Deployment and Integration Report". D3.6.1 FOODIE Deliverable. March 2015
02	Suarez I., et al "D3.2.1. Open API specification". D3.2.1. Deliverable. April 2015
03	Esbrí M., et al "D2.2.1 Platform Specification Report". D2.2.1 Deliverable. April 2015

Table 2: References