PUBLICAMUNDI

SCALABLE
REUSABLE
OPEN
GEOSPATIAL
DATA

REPORT FOR

DELIVERABLE DI.3

1 INTRODUCTION

This report provides an overview of the technical characteristics and functionality of deliverable D1.3 “Production System”. Its purpose is to provide a short introduction to the processes designed and implemented enabling the final production installation (geodata.gov.gr) of the various software developed and reused by PublicaMundi.

The reader is encouraged to visit the software’s repository (https://github.com/PublicaMundi) to receive:

- Up-to-date versions of the software, along with documentation targeted to developers
- Detailed information regarding all development effort (commits, activity, issues)
- Instructions regarding the installation of the software and its dependencies

2 DESCRIPTION OF TASK

PublicaMundi aims to make open geospatial data easier to discover, reuse, and share by fully supporting their complete publishing lifecycle in open data catalogues. To achieve this, we are extending and integrating leading open source software for open data publishing and geospatial data management.

The goal of Task 1.3 is to apply agile development principles with periodic integration in order to maintain relevant contributions. Integration is performed on labs.geodata.gov.gr, a beta-testing facility, where new functionalities are tested on real data and users. New versions are being deployed automatically and user feedback is collected with online tools bringing them in contact with developers.

For the purposes of our production deployment, we built upon our first deployment of labs.geodata.gov.gr, extending our integration environment (D1.2) and through automation procedures we established the production environment. On this environment, a stable version of the software has been rolled out, after extensive testing of the components of PublicaMundi. We focused on stabilizing our software, integrate more geospatial functionality on CKAN, and deploy the stable part of the system into production, providing open access to data publishers and developers for Greek open geospatial data.An overview of the software components is presented in the architecture diagram (Figure 1 from Deliverable D1.1).

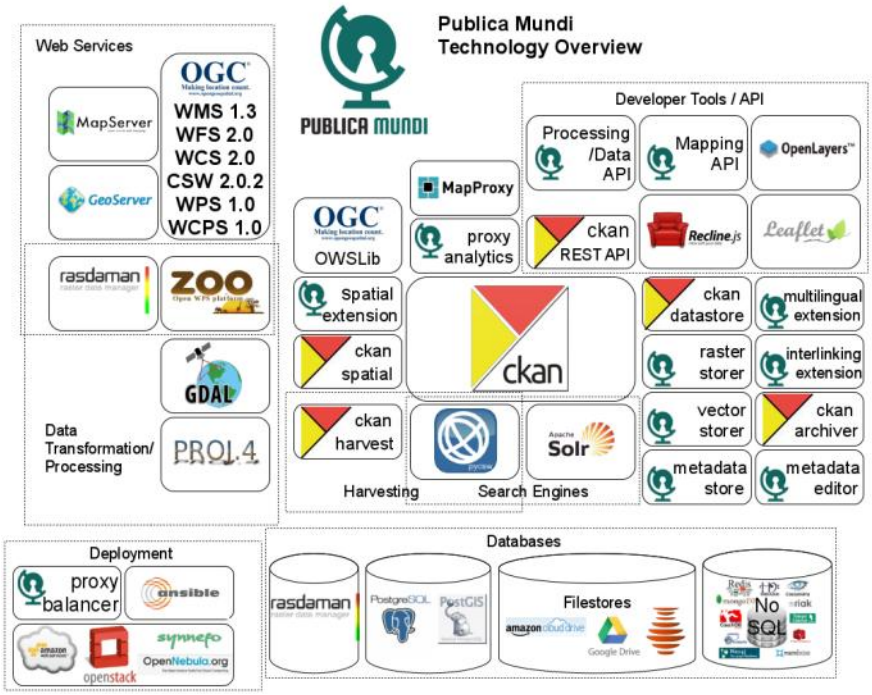

Figure 1: Overview of PublicaMundi components as defined in D1.1

The production system has been deployed in a private cloud (Debian GNU/Linux Virtual Machine cluster deployed on the Synnefo cloud stack) which hosts all software components of PublicaMundi under geodata.gov.gr. The cloud environment is located on a separate private network on the hardware facilities used in D1.2 (Integration environment).

In the following chapter, the implementation details of the production system are presented.

3 PRODUCTION SYSTEM

3.1 PRODUCTION INFRASTRUCTURE (CLOUD)

The production system of PublicaMundi was deployed on top of the Synnefo cloud stack, within a number of virtual machines. Synnefo is a complete open source cloud stack written in Python that provides Compute, Network, Image, Volume and Storage services, similar to those offered by AWS.

Synnefo manages multiple Ganeti clusters at the backend for handling low-level VM operations and uses Archipelago to unify cloud storage. To boost 3rd-party compatibility, Synnefo exposes the OpenStack APIs to users. In Figure 2, an overview of the Synnefo services is presented. Synnefo keeps a clear separation between the traditional cluster management layer and the cloud layer. This unique design approach leads to a completely layered architecture.

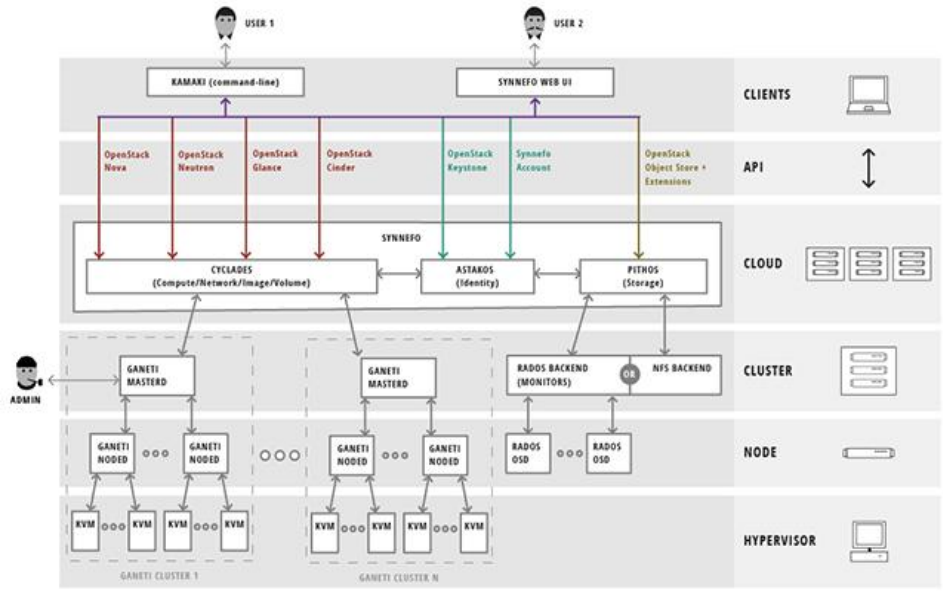

Figure 2: A detailed description of the Synnefo Architecture

For the deployment of production system, we used the Synnefo stack that was already installed on the available server infrastructure, through the Web User Interface (UI) that was available to the administrators in order to maintain and manage the cloud resources of the system.

In Figure 3, the administration page of the Cyclades system (Synnefo UI) is shown. Through that interface, we configured a virtual private network, installing several Debian 7 GNU/Linux virtual machines to host the production software of PublicaMundi.

Figure 3: The web interface for virtual machine administration

For our production system, we have deployed the following VMs:

- 2 CKAN VMs
- 2 Database VMs (*with replication and hot standby*)
- 1 SOLR VM
- 1 Proxy/Analytics VM
- 1 Tiles/Caching VM
- 2 GeoServer VMs (clustered GeoServer with 6 instances each)
- 1 Rasdaman VM
- 1 ZOO WPS VM

- 1 Storage VM
- 1 System Monitoring VM

In Figure 4 the private network setup is presented through the Synnefo UI.

Figure 4: The virtual private network setup and IPv6 configuration on Synnefo

For hosting the geospatial data of geodata.gov.gr, as well as the database instances, map servers data, tiles etc, a virtual network storage was implemented, on 2 NFS nodes (Figure 5). Regular backups are scheduled, and monitoring services are available (Figure 6).

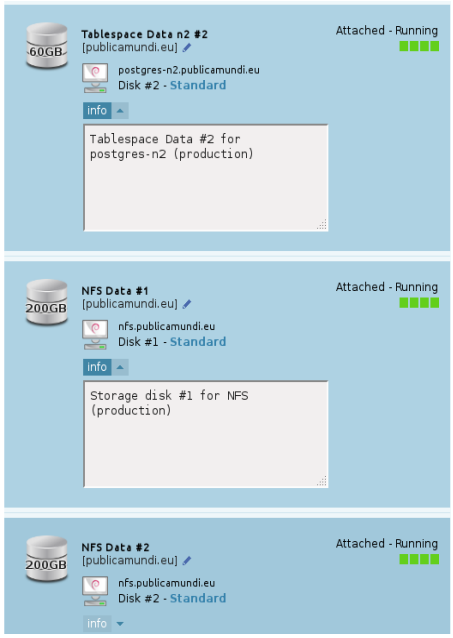

Figure 5: The virtual data store infrastructure for geodata.gov.gr

Figure 6: Monitoring services for each virtual machine and the network is available from Synnefo deployment

3.2 PRODUCTION ARCHITECTURE

The software components of PublicaMundi (Figure 1) were deployed for production into 10 virtual clusters, with the provision of spinning up more virtual machines into each cluster if necessary, through Synnefo and Ansible (see section 3.3). The production system is deployed on several subsystems that are loosely coupled.

The virtual clusters used to deploy geodata.gov.gr are:

- proxy.publicamundi.eu: This is the virtual cluster where the proxy and analytics module is deployed. This is *the only virtual machine visible* from the external network. The rest are behind a multi-layer firewall, having only NATed access to the Internet. All the web requests that are arriving from the internet to geodata.gov.gr are logged and redirected to the proper system service. This cluster is based on the very high availability HAProxy server. The server holds separate logs for each system endpoint; the analytics module exploites these logs to compute usage and spatial statistics from every provided API or sub-system.
- solr.publicamundi.eu: This is the virtual cluster for the catalog indexing service, offering high availability of search results. Several

Apache SOLR instances are deployed on top of several Apache Tomcat servers, which power the full text search capabilities of the production system. This cluster is scheduled to also support the interlinking capabilities, once they have been rolled-out to the production system.

- ckan.publicamundi.eu: This is the virtual cluster of the virtual machines that host the CKAN catalogue, along with its extensions. The Apache web server is used as a WSGI server. On the same WSGI server, the CSW interface, implemented by pycsw is deployed and tightly integrated to CKAN. Other CKAN extensions installed include ckanext-archiver, ckanext-datastorer, ckanext-harvest, ckanext-publicamundi (*with raster storer and vector storer plugins included*), ckanext-spatial, as well as PublicaMundi's MapClient, Mapping API and Data API.
- nfs.publicamundi.eu: This is a single VM that provides access to the Network File System, so that virtual servers can communicate on file system level. This file system holds the analytics store, the data storer files, the generated map tiles and the setting folders of several OGC services.
- postgres.publicamundi.eu: This is the database cluster that hosts the PostgreSQL 9.3 service extended with PostGIS 2.1. The database cluster is used for the core CKAN application, for CKAN tabular resources (*provided by datastorer*), for CKAN vector resources (*provided by vectorstorer*), for raster resources, and for aggregated monitor-related data. Also, as a part of the CKAN application, this database stores the original metadata XML files. The cluster consists of 2 actual database servers in master/hot-standby formation and is based on PostgreSQL's streaming replication capabilities. Furthermore, the master node is also hosting a pgpool-2 service in order to provide a single database frontend (to applications) and load-balancing for read-only workloads.
- geoserver.publicamundi.eu: This is the virtual cluster for deploying GeoServer 2.6 instances. Each virtual machine hosts 6 GeoServer instances. Each instance is deployed within a separate Apache Tomcat 6 container and is available through a different network port. The configuration files for GeoServer are stored in the NFS and are shared between the instances. The master instance is the only one providing access to the GeoServer UI and the GeoServer REST API, thus controlling the overall settings of the cluster. The GeoServer cluster is load-balanced behind the HAProxy service, thus providing a

single end-point for OGC Web Services (WMS, WFS etc.). This way, we achieve high availability and performance of our services. This cluster is mostly used for providing the vector storer OGC interfaces (i.e. WMS and WFS for the vector data) and is directly connected to the PostgreSQL/PostGIS cluster.

- rasdaman.publicamundi.eu: This is the virtual cluster where rasdaman 9.1 is deployed. This cluster is responsible to store all uploaded raster files within the raster storer database and provide WCS, WCS-T, WCPS and WMS services with end-points advertised through the CKAN catalogue and the data resources. Rasdaman is being deployed on a file-system setup to enhance performance of accessing raster data.
- zoo.publicamundi.eu: In this the virtual cluster the ZOO WPS engine is deployed. ZOO is responsible for providing OGC WPS services and the geoprocessing algorithms behind them. It is deployed as a FastCGI application and is coupled with the ckanext-wps CKAN extension.
- tiles.publicamundi.eu: This is the virtual cluster for serving high performance map tiles. MapProxy 1.7 is deployed under a WSGI server (gunicorn) and can serve tiles generated from a variety of sources (*WMS, tile cache, MBTile files, TMS etc*). MapProxy is used to speed-up WMS services, with automatic generation of tiles according to map usage from users and analytics.
- monitor.publicamundi.eu: This is a single VM responsible for monitoring all other VMs of our production deployment. All machines collect various statistics about themselves with the collectd daemon (https://collectd.org/) and periodically push them to the monitor node (acting as the collectd master node). Thus, the monitor node is able to present a centralized view of resource utilization on the entire deployment, detect abnormal behaviours (e.g. excessive use of available resources) and emit the appropriate notifications on an alarming situation.

3.3 DEPLOYMENT ORCHESTRATION

For the deployment of the final production system, we used the previously created GitHub repository[1] to store the configuration files of each virtual machine used (*excluding the password information and other security settings*

which are not public). For the production deployment of our software stack into the PublicaMundi infrastructure, the Ansible engine is used.

Ansible is a simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs. Being designed for multi-tier deployments, Ansible models IT infrastructure by describing how all of the systems inter-relate, rather than just managing one system at a time. It uses no agents and no additional custom security infrastructure, so it's easy to deploy — and most importantly, it uses a very simple language (YAML, in the form of Ansible Playbooks) that allows administrators to describe their automation jobs in a way that approaches plain English. Ansible works by connecting to the system nodes and pushing out small programs, called "Ansible Modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (*over SSH by default*), and removes them when finished.

In order to deploy the production system on the Synnefo infrastructure, we developed several ansible scripts, available in our GitHub. Furthermore we developed additional plugins[2] and ansible roles[3]. In essence, the deployment procedure consists of 3 phases:

- Setup the administrative node (*the Ansible control machine*)
- From the administrative node, setup the internal network of nodes
- From the administrative node, apply roles to all nodes

The final result of the application of Ansible Playbooks is the production system (presented on next section). Software updates are handled using git with simple code updates in the needed system folders, thus making maintenance an easy task.

3.4 PRODUCTION SYSTEM

After the deployment of virtual clusters on the production system through Ansible scripts and modules, the production catalog is available, along with the OGC services and APIs. Figure 7 presents the new CKAN theme that was developed during the last period of the project. The production system was

[1] https://github.com/PublicaMundi/labs.geodata.gov.gr

[2] https://github.com/PublicaMundi/ansible-plugins

[3] https://github.com/PublicaMundi/ansible-roles

translated on Greek and the translations were contributed to the CKAN project. The translated user interface is available on geodata.gov.gr.

On Figure 8, the dataset page of the catalog is presented, with OGC resource links.

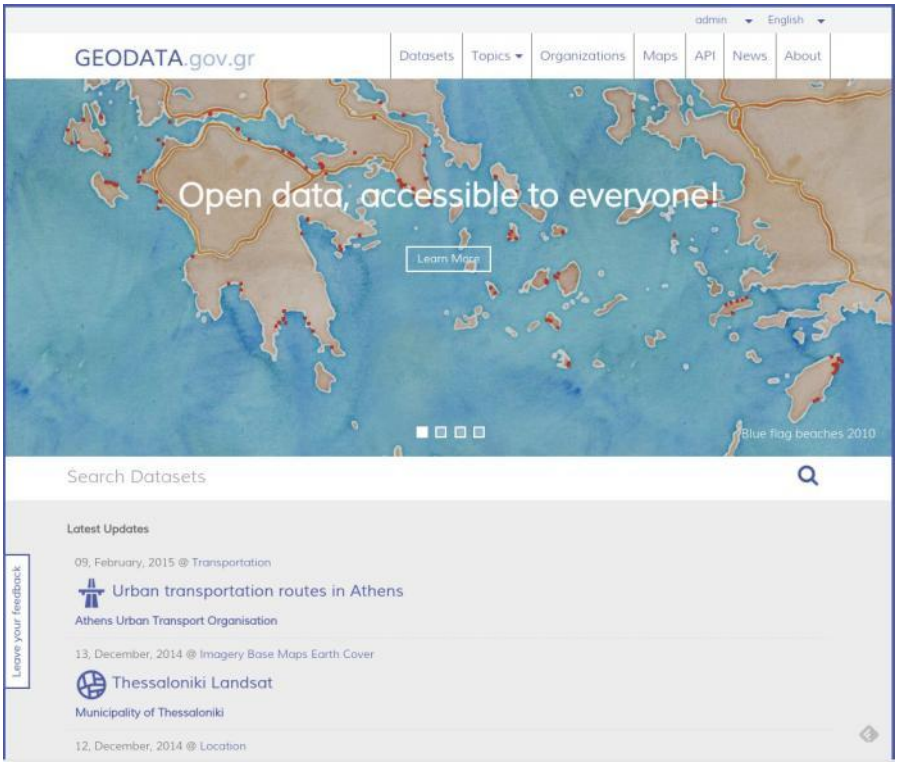

Figure 7: PublicaMundi CKAN theme

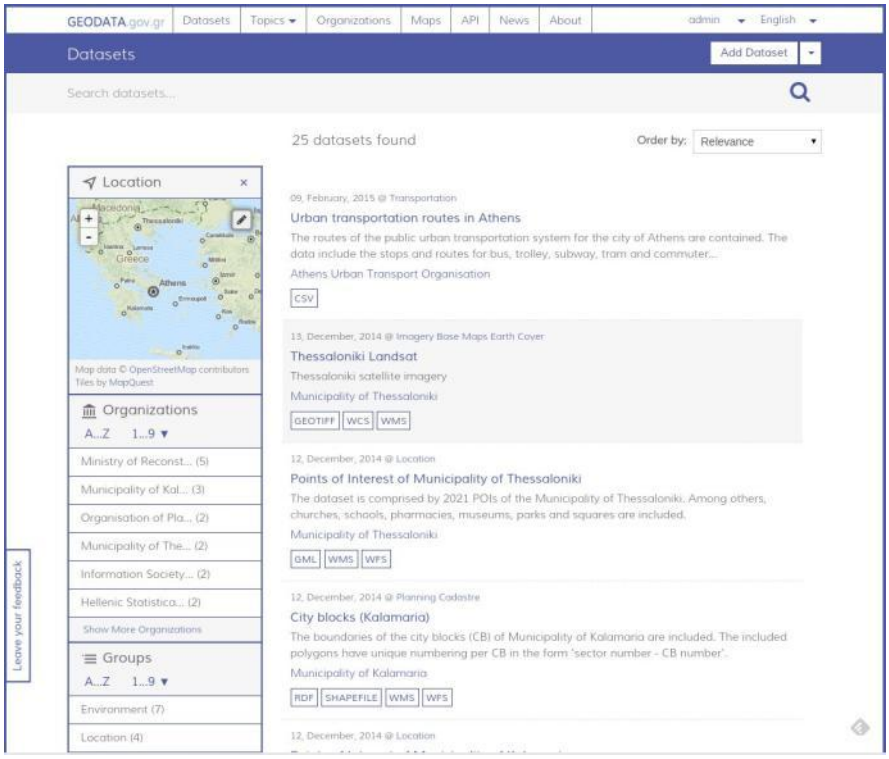

Figure 8: Dataset page of the catalogue

The final version of the Mapping API is now deployed, supporting several formats of vector and raster data, along with all the major OGC service APIs, like WMS, WFS, and WCS (Figure 9). The OGC CSW catalog service is now integrated directly on CKAN and available on the metadata page (Figure 10).

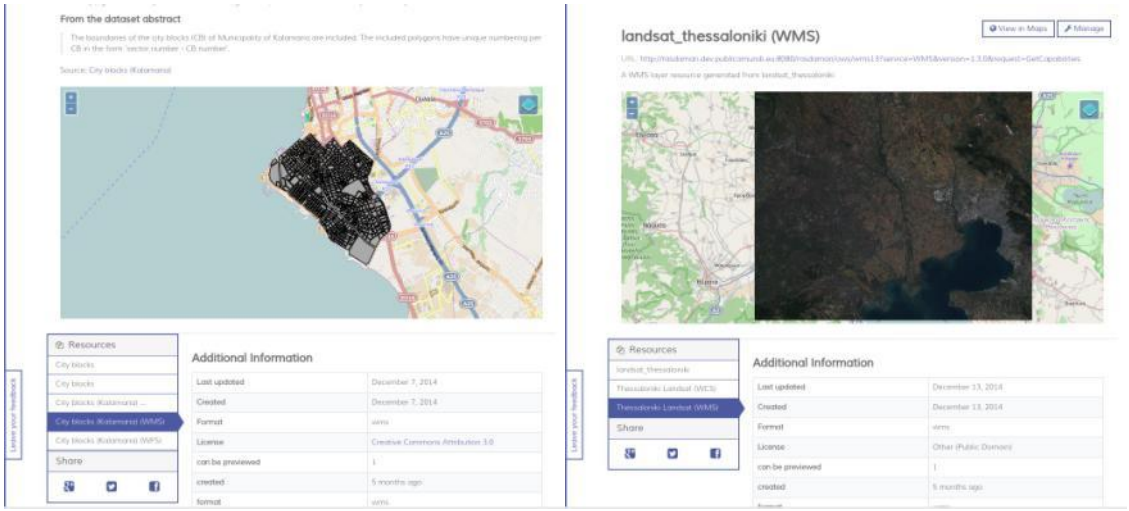

Figure 9: Final geospatial data previewers, supporting both vector and raster data

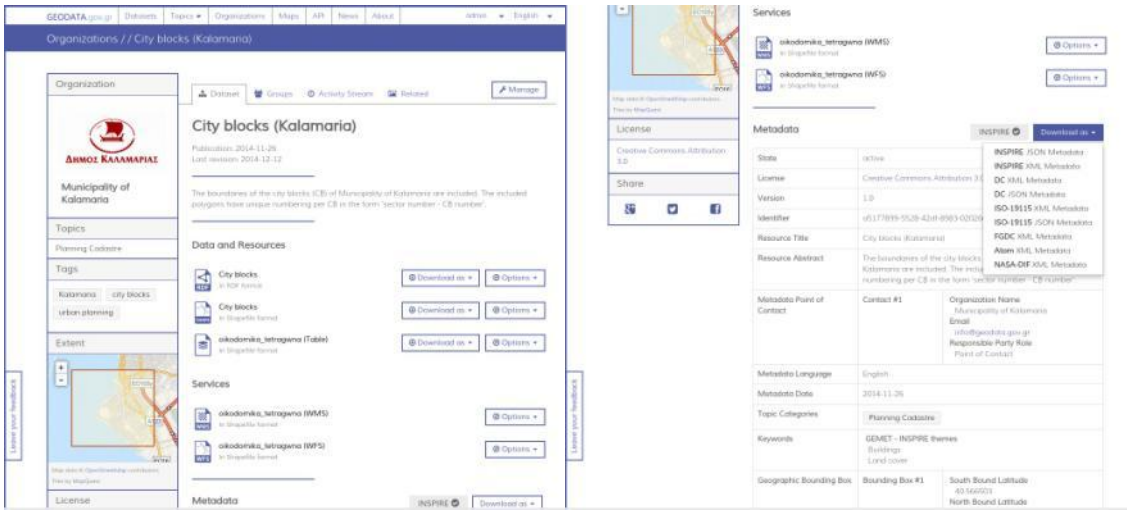

Figure 10: Geospatial dataset page example, supporting automatically generated OGC services and INSPIRE metadata

The production system also includes a full map application, supporting searching, filtering and downloading data from the catalog (Figure 11)

Moreover, the data publishing workflow has been finalized and deployed on the production system. The new dashboard for the ingestion of spatial datasets into PostGIS and automatic creation of OGC services has been deployed in production (Figure 12). The final deployment of geodata.gov.gr also integrates a WordPress in the News section, with a new custom theme developed and used in production (Figure 12).

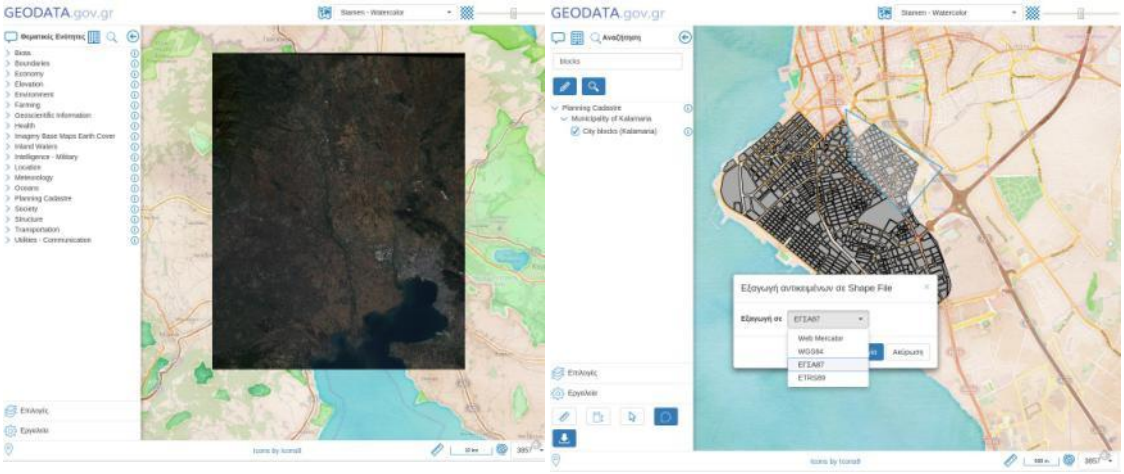

Figure 11: Production system previewers for raster and vector datasets, through the Mapping API

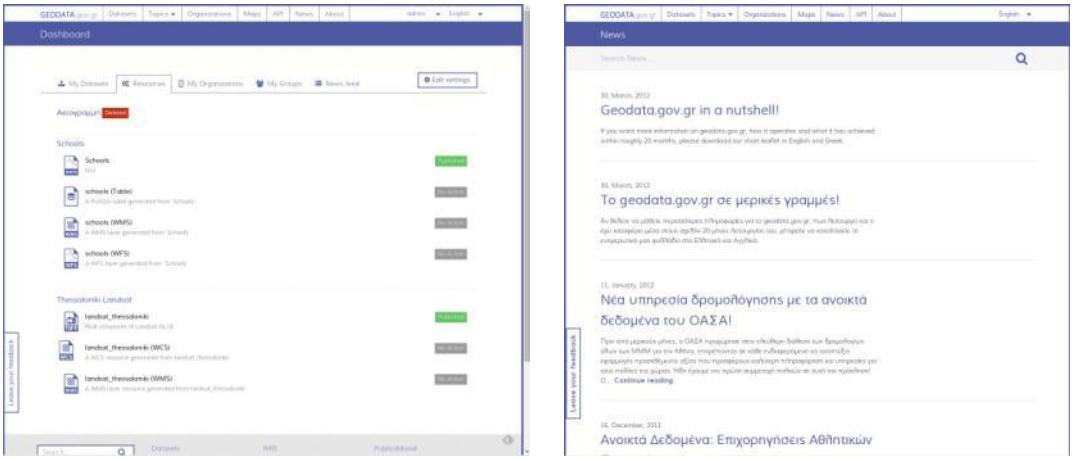

Figure 12: Final version of the PublicaMundi dashboard for ingestion of spatial resources (left). The WordPress theme for geodata.gov.gr (right)