Interreg Co-funded by the European Union

Italy – Croatia

Digital Plan

Project DIGITAL PLAN

Technical Manual

Deliverable no. D.1.5.2.

Done by LP GAL Molise

Interreg Co-funded by the European Union

Italy – Croatia

Digital Plan

Project Acronym	Digital Plan
Project ID Number	ITHR0200430
Project Title	Civil Protection Plan Digitalization through Internet of Things Decision Support System based Platform
Priority Axis	2 - Green and resilient shared environment
Specific objective	SO 2.1 - Climate change adaptation
Work Package	WP1 Digitalization model of civil protection plans and IoT EDSS Platform Development
Activity	A.1.5 Development of Internet of Things Decision Support System Platform
Deliverable	D.1.5.2. Technical Manual
Partner in Charge	LP GAL Molise
Status	Final
Distribution	External

Index

1. Introduction and Purpose 4

2. Platform Overview 4

3. System Architecture 5

3.1 Overview of the platform's components 7

4. Main Application and Public Application 8

5. Main Backend Service 10

6. Identity Provider 12

7. OGC Services 13

8. API Gateway 14

9. Decision Support System 15

10. Environments 16

11. Release and Publishing Flow 17

12. Security Overview 18

1. Introduction and Purpose

This manual is intended for technical and decision making staff who require an understanding of what the platform is built on, for example when evaluating its architecture from a governance, procurement or integration perspective. A separate user manual describes how the platform is used, and a separate API appendix accompanies this manual for readers who require the detailed surface of the platform's own backend API. This manual identifies each component involved and explains the function it performs within the platform, in depth for what this platform builds itself and more briefly for the open source services integrated into its infrastructure and for the separate systems it integrates with.

The scope of this manual is to describe the technologies used to build the platform, which work together to provide the service. Readers unfamiliar with the platform should proceed to the following chapter, Platform Overview, which introduces the two applications that constitute the platform at a product level, before continuing to System Architecture for the complete picture of every component and how they fit together. The chapters that follow take each component in turn, followed by the environments the platform runs in, its release process, and a security overview.

2. Platform Overview

DigitalPlan is delivered as two separate web applications, each built on the same underlying technology stack and communicating with the same family of backend components, but serving different audiences.

The main application is used by authenticated staff for the full range of management activity: building digital territorial plans, managing documents, geographic layers, users, partner organisations and risk monitoring configuration. Access to the main application requires authentication for every function it exposes.

The public application is aimed at members of the public and does not require an account. It exposes a defined subset of the platform's functionality, namely a public map, public plan documents and a public risks and alarms page, in each case as a read only, simplified view of information maintained through the main application.

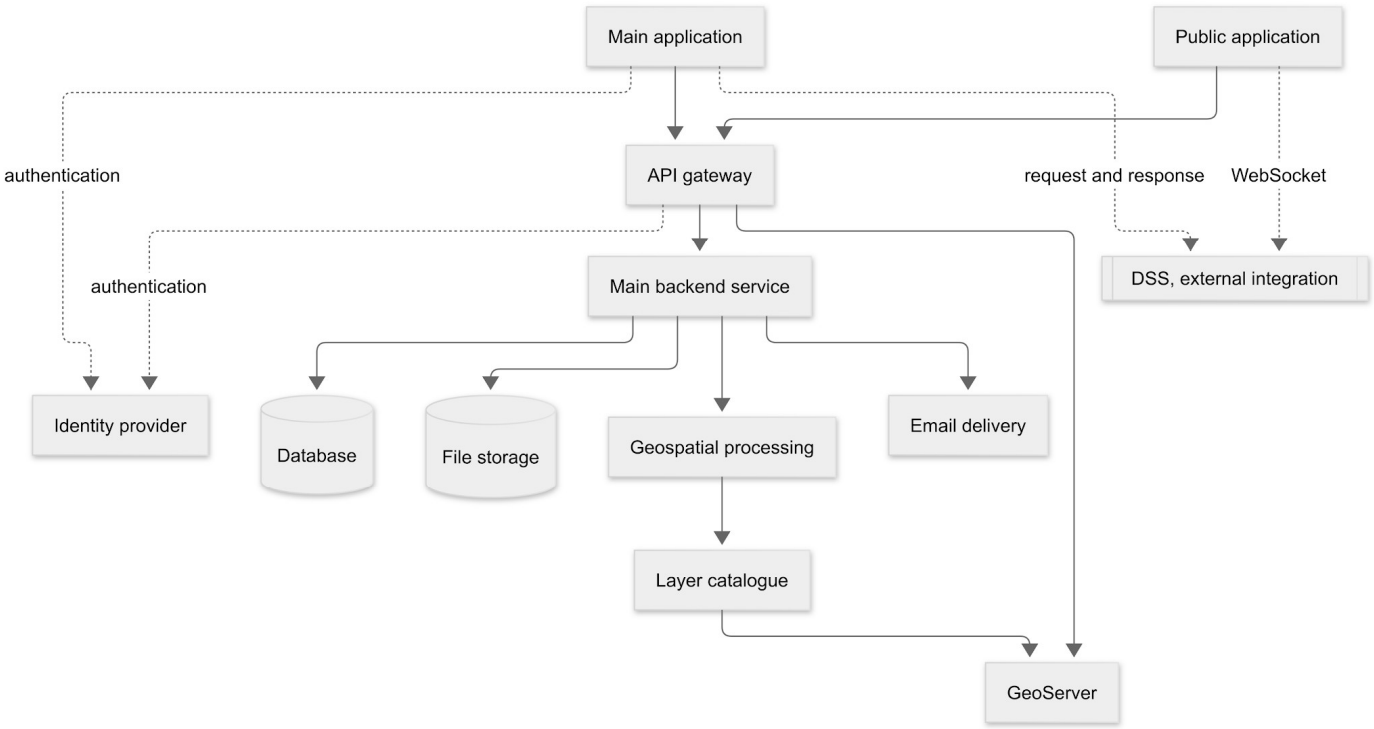
Both applications are single page web applications, meaning that navigation between screens takes place within the browser without a full page reload, and both depend on the backend components described in the System Architecture chapter and the chapters that follow it for their data. The technology choices described in the following chapters apply equally to both applications unless stated otherwise.

3. System Architecture

This chapter brings together every component that makes up the platform into a single, overall picture, before the chapters that follow describe each one individually. It states, for each component, whether this platform develops it or adopts it as an established product or external integration; the depth of the chapters that follow reflects that same distinction, substantial for what this platform builds, brief for the open source services integrated into its infrastructure and for the separate systems it integrates with.

The platform's two frontend applications are the entry point for every user, whether authenticated staff or a member of the public, and every request they make for data passes through the API gateway before reaching the component responsible for handling it. Authentication requests are directed to the identity provider rather than to any backend component directly. The gateway directs the remaining requests either to the main backend service or to GeoServer, depending on what is being requested. The main backend service in turn depends on its database for persistence, on a file storage component for uploaded files, on a geospatial processing step and layer catalogue component when a geographic layer is being published, and on an email delivery component when a notification must be sent. Separately from all of this, both frontend applications also communicate with the DSS, a system integrated with this platform over the connections described in the Decision Support System chapter.

The diagram below summarises these relationships.



3.1 Overview of the platform's components

The main application and the public application, described in the next chapter, present the platform's functionalities to their respective users via web browsers; they serve as the interface and point of access to all services. The main backend service, described in a separate chapter, implements the platform's business logic, together with the geospatial processing components and the associated layer catalogue, as well as the database, file storage and email dispatch components on which it depends. The identity provider, described in the chapter 'Identity Provider', is Keycloak, a well-established open-source product that this platform uses for secure user login and identity management. GeoServer,

described in the chapter 'OGC Services', is also a well-established open-source product used as an OGC server. The API gateway, described in a separate chapter, is Apache APISIX, also a well-established open-source product used as an ingress and API gateway. The DSS, described in the chapter 'Decision Support System', is an external backend service that handles threshold calculations and alerts, with which this platform integrates via an API.

4. Main Application and Public Application

The main application and the public application are the two frontend services this platform builds and maintains, and this chapter describes, in depth, the technology each is built on.

Both applications are built using Angular, a mature, widely adopted framework for building single page web applications, maintained by Google and used extensively in enterprise software. Angular was designed for building large, structured applications composed of many independent screens and components, which suits a platform of DigitalPlan's scope, spanning plan digitisation, document management, geographic layer management, user administration and risk monitoring within a single, consistent interface, and both applications are single page applications, meaning that navigation between screens takes place within the browser without a full page reload.

Alongside the framework itself, the platform uses a centralised state management approach. In an application of this size, the same underlying data, for example the currently selected plan, or the list of available themes and classes, is often needed by several screens or components at once, and centralised state management keeps a single, authoritative copy

of this data, ensuring that every part of the interface referencing it remains consistent, rather than each screen maintaining its own, potentially divergent, copy.

The platform's visual components, tables, forms, dialogs, menus and similar interface elements, are provided by PrimeNG, an open source component library built specifically for Angular, offering a wide range of ready made, accessible components together with a theming system. Using an established component library rather than building every interface element from first principles allows the platform to offer a consistent, accessible and well tested user experience, and benefits from ongoing maintenance and security updates provided independently of the platform itself.

Interactive maps across both applications are built using Leaflet, a widely adopted, open source JavaScript library for mobile friendly interactive maps, presented through a shared internal component so that maps behave and appear consistently across every module that uses one. Leaflet was chosen for its maturity, its lightweight footprint and its broad support for the open standards used elsewhere in the platform's geographic data handling, and it renders, on both applications' maps, the geographic data served by GeoServer, described in the OGC Services chapter.

Interface text throughout both applications is managed through ngx-translate, a widely used internationalisation library for Angular applications, rather than being embedded directly within individual screens. Under this approach, each piece of interface text is stored as a translation key, and a separate translation file, one per supported language, maps each key to the corresponding text in that language, allowing a new language to be added, or an

existing translation corrected, without any change to the application's underlying logic. Italian and English are fully translated and available throughout both applications.

The public application receives the environmental risk and alarm updates described in the Decision Support System chapter over a WebSocket connection, a persistent, bidirectional communication channel that remains open between the browser and the server, so that new sensor readings and alarms reach the map as soon as they occur, rather than the browser needing to repeatedly request updates on a fixed schedule. Plan documents and other file attachments are handled through standard web file upload technology, supporting both drag and drop and manual file selection, with upload progress reported to the user for the duration of the transfer, a conventional approach for browser based file handling that does not rely on any specialised or proprietary upload mechanism. Uploaded PDF documents are rendered directly within the browser through a built in viewer, rather than requiring the file to be downloaded first or rendered through an external document viewing service, and a file type restriction, the PDF only rule applied to plan documents described in the user manual, is enforced as part of this upload handling, and the uploaded files themselves are retained by the main backend service, described in its own chapter, rather than by either frontend application.

5. Main Backend Service

The main backend service is the other component this platform builds and maintains, alongside the two frontend applications described in the preceding chapter. It holds the platform's core operational data, plans, plan documents, geographic layer metadata, themes

and classes, users and partner organisations, and is the primary counterpart to the majority of the modules described in the user manual; the detailed operations it exposes are listed in the API appendix that accompanies this manual.

The service is built in Python using FastAPI, a modern framework for building web APIs, and exposes its operations as a REST API described by an OpenAPI 3.1 specification, the same specification the API appendix is derived from; it stores its operational data in PostgreSQL, a mature, widely used open source relational database.

Files uploaded through either frontend application, plan documents as much as the shapefiles used to define a plan's area of application in Plan Management, are retained in a dedicated file storage component, a storage technology suited to holding a large number of discrete files reliably and separately from the service's own database, rather than being written to the service's local file system.

Before a newly uploaded geographic layer becomes available on any map, it passes through a dedicated geospatial data processing step, which validates and converts the uploaded file, followed by a layer catalogue component that publishes the result to GeoServer, described in the OGC Services chapter. This is the processing a layer moves through while its status, described in the user manual's Layer Management chapter, shows as loaded or in progress, before it reaches the published status at which point it becomes available wherever the layer catalogue is consulted.

Outgoing email, for example the alarm notifications configured in DSS Management, is sent through a dedicated email delivery component rather than being handled directly by the application logic that triggers it.

6. Identity Provider

Authentication to the main application is based on OAuth2 and OpenID Connect, commonly abbreviated as OIDC, the industry standard protocols for delegated authentication used extensively across enterprise and public sector software. Under this model, the DigitalPlan application itself never receives or stores a user's password; instead, it redirects the user to a dedicated identity provider, which performs the actual verification of credentials and then issues the application a token attesting to the user's identity and permissions.

Keycloak serves as this identity provider. This platform uses Keycloak for authentication via the OAuth2 protocol and as its identity manager, relying on it as an open source identity and access management solution that adds authentication and single sign on to applications with minimal effort, providing user federation, strong authentication, user management and fine grained authorization, based on standard protocols including OpenID Connect, OAuth2 and SAML. Centralising authentication in this way allows a single sign on experience across the organisation's software, and ensures that a change in authentication policy, for example a password rotation requirement, need only be enforced in one place.

Once authenticated, a user's session carries the single role granted to their account by an administrator, which the main application uses to determine which sections of the interface are available to that user, a pattern generally referred to as role based access control; this

separates the question of who a user is, established once through the identity provider, from the question of what a user is allowed to do, which can be adjusted at any time by changing the role associated with their account. Sessions are refreshed automatically in the background while a user remains active, and an expired or otherwise invalid session results in the user being returned to the identity provider's login page. The public application does not enforce this authentication step for its public facing functionality, while the same identity provider remains available to any visitor who chooses to authenticate from the public application's entry point.

7. OGC Services

This platform uses GeoServer to expose its geographic layers on the map through OGC services. GeoServer is an open source server, written in Java, that publishes geospatial data from a wide range of sources using open standards, and is the reference implementation of the Open Geospatial Consortium's Web Feature Service, or WFS, standard.

WFS is an open, vendor neutral standard, maintained by the Open Geospatial Consortium, that defines web interface operations for querying vector geographic features, points, lines and polygons, together with their descriptive attributes, allowing a client to discover which data is available, describe its structure, and query it, all over a network connection. Building on an open standard in this way means the platform's geographic data is not tied to a single proprietary format, and can in principle be consumed by other GIS compliant systems if required.

This is the platform's general approach to any functionality involving a map, including the working map used to digitise plan content, the layer preview shown while managing the geographic layer catalogue, and the two public facing map pages, each rendering GeoServer's data through the mapping technology described in the Main Application and Public Application chapter.

8. API Gateway

Every request either frontend application makes for data is received by an API gateway before reaching the component responsible for handling it. This platform uses Apache APISIX as its ingress and API gateway, a dynamic, high performance, open source API gateway built for cloud native architectures and maintained as a top level project of the Apache Software Foundation.

An API gateway is a software component that sits in front of a group of backend services to act as their single entry point, routing each incoming request to the service responsible for it, rather than requiring every client to know the address of every individual service or requiring every service to implement common concerns, such as access policy, independently. Within this platform, the gateway routes requests to the main backend service or to GeoServer depending on what is being requested, as described in the System Architecture chapter.

9. Decision Support System

Environmental risk monitoring, encompassing weather and hydrological sensors, monitoring stations, alert thresholds and alarms, is provided by a backend service referred to within the platform as the Decision Support System, or DSS. The DSS is a separate backend service, maintained independently of this platform, that this platform integrates with over an API rather than builds or operates; its own technology and internal structure are outside the scope of this manual, and are described instead in a separate technical manual dedicated to it.

The public risks and alarms page receives updates from the DSS over a WebSocket connection, a persistent, bidirectional communication channel that remains open between the browser and the server, in contrast to the conventional request and response pattern used elsewhere in the platform. This allows new sensor readings and alarms to reach the public map as soon as they occur, without the browser needing to repeatedly request updates on a fixed schedule, a technique commonly referred to as polling, which is less timely and places unnecessary load on the server. The DSS Management screens used by staff to configure sensors, thresholds and alarm notifications, described in the user manual, currently load and refresh their data on demand, for example on page load or when a filter changes, rather than over the same live channel; the real time channel is used specifically for the public facing risk display.

10. Environments

The platform is operated across multiple separate environments, each an independent, self contained deployment of the applications and their supporting configuration.

A development environment is used while new functionality is being built and verified. A staging environment is used for pre release verification, allowing changes to be validated under conditions similar to production before they are made available to real users. A production environment serves the platform's actual users, both authenticated staff and members of the public.

Each environment is configured with its own addresses for the API gateway, the main backend service, the DSS, GeoServer and the identity provider, ensuring that activity in the development or staging environments cannot affect production data, production users or citizen facing services. A local development mode is also available, in which the frontend applications can be configured to communicate with a backend service running on a developer's own machine rather than a remote environment, which is useful during active development of backend functionality alongside the frontend.

This separation of environments is a standard practice for software of this kind, allowing new functionality to be verified in progressively more production like conditions before it is exposed to real users.

11. Release and Publishing Flow

Changes to the platform reach the development environment through an automated process. Once a change has been integrated into a project's main development branch, a build pipeline is triggered automatically, without manual intervention. The two frontend applications and the main backend service, each maintained in its own project, follow the same pipeline pattern: a build packages the changed project into a container image, a self contained, portable unit of software that includes everything it needs to run, and publishes that image to a private image registry, a repository dedicated to storing and distributing these container images.

A record of the change is added automatically to the platform's changelog at the same time, providing a running history of what has been deployed and when. The reference to the newly published image is then propagated to a separate deployment repository, which determines what is actually running in the development environment; this separation between the source code repository and the deployment repository is a deliberate control point, ensuring that publishing a new image does not, by itself, automatically change what is running anywhere.

Promotion of a given release to the staging and production environments follows the same automated path through the deployment repository, allowing releases to be verified in staging before being made available in production.

The overall effect of this process is that every change ultimately deployed to any environment can be traced back to a specific, recorded build, packaged in a consistent format, rather than being deployed through an ad hoc or manual procedure.

12. Security Overview

Access to the main application is governed by the OAuth2 and OpenID Connect authentication standards described in the Identity Provider chapter, with Keycloak acting as the identity provider and the role associated with a signed in user's account determining what they are able to see and do. This model ensures that authentication decisions are made in a single, centrally managed location rather than independently within each application.

User credentials are never stored or verified by the DigitalPlan applications themselves; that responsibility belongs entirely to the identity provider, which is a deliberate design choice that limits the platform's exposure to credential related risk and allows organisation wide security policy, for example password complexity or multi factor authentication requirements, to be enforced consistently and independently of the platform's own release cycle.

All environments described in the Environments chapter communicate over encrypted connections, using HTTPS for conventional web traffic and its encrypted WebSocket equivalent for the real time channels described in the Decision Support System and Main Application and Public Application chapters, ensuring that data in transit between the browser and the platform's components is protected from interception.

The public application's design reflects a security relevant boundary: functionality capable of creating, modifying or deleting data is deliberately excluded from the public facing pages, regardless of role, so that a public visitor is structurally limited to read only access, rather than relying solely on the absence of a login to enforce this restriction.