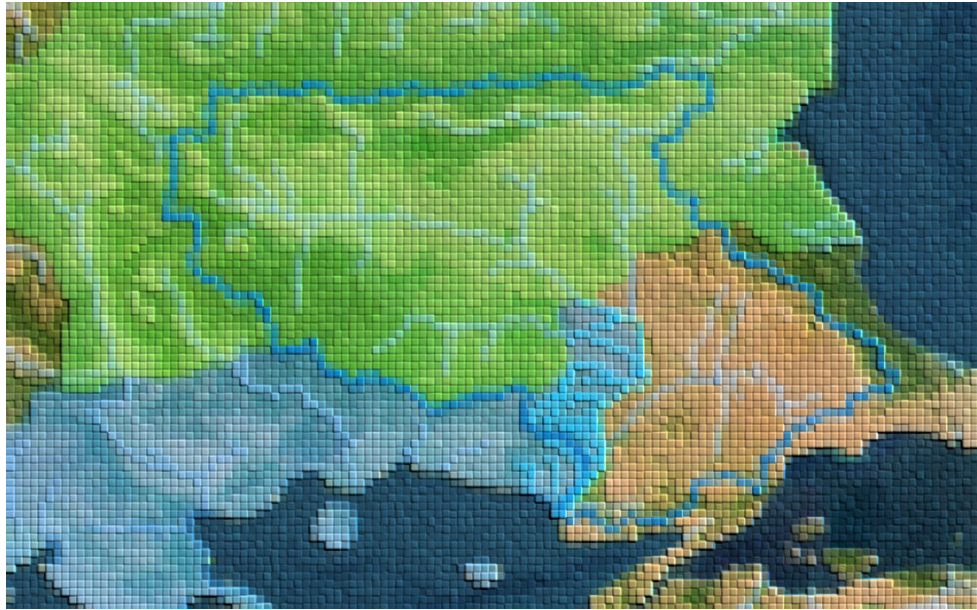




HELLENIC REPUBLIC



MINISTRY OF ENVIRONMENT AND ENERGY
GENERAL SECRETARIAT FOR NATURAL
ENVIRONMENT AND WATER
DIRECTORATE GENERAL FOR WATER



Interreg

Greece-Bulgaria

European Regional Development Fund



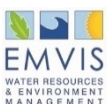
EUROPEAN UNION

FLOODGUARD

Integrated actions for joint coordination and responsiveness to
flood risks in the Cross Border area

DELIVERABLE 4.6.5.Γ

AUTOMATED OPERATIONAL WEB PLATFORM FOR FLOODGUARD EARLY WARNING SYSTEM



EMVIS Consultants SA

July 2023

The Project is co-funded by the European Regional Development Fund (ERDF) and by national funds of the countries participating in the Interreg V-A "Greece-Bulgaria 2014-2020" Cooperation Programme



Version	Date	Comments
1.0	July 24 th , 2023	1 st version

This report was compiled by EMVIS Consultants SA on behalf of General Secretariat for Natural Environment and Water, Directorate General for Water, under the frame of the INTERREG Project “Integrated actions for joint coordination and responsiveness to flood risks in the Cross Border area – FLOODGUARD”

Disclaimer: The contents of this report are sole responsibility of *General Secretariat for Natural Environment and Water, Directorate General for Water* and can in no way be taken to reflect the views of the European Union, the participating countries the Managing Authority and the Joint Secretariat

TABLE OF CONTENTS

1	Introduction	6
1.1	The Floodguard Project	6
1.2	Scope of present report	6
2	Functional requirements	9
2.1	Meteorological app	9
2.2	Hydrological app.....	9
2.3	Hydraulic app	10
2.4	Early Warning System app	11
2.5	General requirements	11
3	Technical requirements.....	12
3.1	General requirements	12
3.2	Connectivity and Communication Requirements	12
3.3	Data curation requirements	13
3.4	Data storage requirements	13
3.5	User interface requirements	14
3.6	Security requirements.....	14
4	Platform architecture	16
4.1	Open source architecture.....	16
4.2	Hardware layer	16
4.3	Integration with existing workflows.....	18
4.4	High level architecture	18
4.5	Web framework	20
4.5.1	State-of-the art analysis	20
4.5.2	Django web framework	21
4.6	Web mapping.....	22
4.6.1	Geoserver	23

4.7	Database and File System	24
4.7.1	State of the art analysis	24
4.7.2	PostgreSQL	26
4.7.3	File system	31
4.8	Application Programming Interface	34
4.9	Web server and Application server	34
4.10	Graphical User Interface	34
4.10.1	State of the art analysis of web mapping libraries	35
4.10.2	Leaflet	36
4.10.3	State of the art analysis for interactive charts.	36
4.10.4	Chart.js.....	36
4.11	Early Warning System	38
4.11.1	Early Warning system for hydrological alerts.....	38
4.11.2	Early Warning system for hydraulic alerts.....	39
4.12	Data curation.....	40
4.13	Security.....	40
4.13.1	Access control module and user authentication	41
4.13.2	Security of stored information	42
4.13.3	Firewalls.....	43
4.13.4	User authentication	44
4.13.5	Security in data exchanged through APIs	44
5	Technology stack	45



LIST OF ACRONYMS

Acronym	Definition
API	Application Programming Interface
EWS	Early Warning System
GIS	Geographic Information System
GUI	Graphical User Interface
HDF	Hierarchical Data Format
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
OGC	Open Geospatial Consortium
ORM	Object-Relational Mapping
OS	Operating System
RPC	Remote Procedure Calls
RDBMS	Relational Database Management System
RDP	Remote Desktop Protocol
REST	Representational State Transfer
SFTP	Secure File Transfer Protocol
SSH	Secure Shell
SQL	Structured Query Language
SSL	Secure Sockets Layer
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Mapping Service

1 INTRODUCTION

1.1 THE FLOODGUARD PROJECT

The project focuses on the challenges of flood risk management via integrated actions for joint coordination and responsiveness to flood risks in the Cross-Border (CB) area of Greece and Bulgaria. This area is vulnerable to climate change and its negative impacts, which needs adequate coordination for cross-border management. The territories covered by the project need to increase significantly their adaptive capacity to climate change, i.e. the international river basins of the cross-border rivers - Strymon-Struma, Nestos-Mesta, Ardas-Arda and Evros-Maritsa. The effective management of flood risks is of high importance and affects high number of population and surface area. The project actions are directly oriented to joint coordination of five flood risks management plans (FRMP) for international rivers - two plans for Bulgarian territory (FRMPs of East Aegean and West Aegean) and three plans for Greek territory (FRMP of Eastern Macedonia, Thrace Water District and Evros river).

The project, which started on Apr 01, 2019, has as a main objective to strengthen the capacity of authorities to ensure effective, integrated joint coordination and responsiveness to flood risks in the CB area. The project addresses different aspects of flood risk management in order to provide integrated results and the following sub-objectives have been defined:

- (1) introducing a joint institutional approach for assessment, planning, prevention and fighting against floods of authorities;
- (2) increasing the technical and administrative capacity of civil protection services for joint actions in the field of preparedness and response in cases of floods;
- (3) introduction of effective information exchange structures and systems for flood risk assessment, mitigation and management;

The project introduces the latest achievements in information systems and will add value by combining the capacity, planning and future development of national policies from the two countries in the field of flood risk management.

The objectives for improved administrative capacity will be achieved through provision of training facilities and joint trainings of first responders, supply of equipment and training software, as well as elaboration of bilateral standards and operational procedures for actions in cases of floods. The introduction of a joint institutional approach for flood risk management will be achieved through the establishment of a Joint Working Group for prevention and reaction in cases of floods. Introduction of effective information exchange structures and systems for flood risk assessment, mitigation and management will be achieved through development of data collection, analyses and assessment tools, as well as development of information tools and structures for flood risk mitigation via GIS functionalities, early warning systems, transfer lines, flood forecasting systems, development of a common regional database management system.

1.2 SCOPE OF PRESENT REPORT

The FLOODGUARD project develops a set of modelling components that are used for alarming and mitigating the risk of floods in the Evros river area. This will be provided through an operational web-based platform

that generates short-term forecasts by combining meteorological, hydrological and hydraulic models, to support proactive decision making in the wider Evros area. Th

This report aims to provide a comprehensive documentation of the high-level FLOODGUARD platform architecture as well as the required IT components that will allow us to accomplish the desired features and functionalities of the platform.

Chapter 2 describes the functional requirements of the platform, i.e what a user would like to be able to accomplish through the FLOODGUARD platform in each one of the provided applications which are the 1) Meteorologic app, 2) the Hydrological app, 3) the Hydraulic app and 4) the Early Warning app. Separate functional requirements have been identified for each one of the 4 applications, while a fifth category with general platform requirements has been also compiled.

Chapter 3 focuses on the technical requirements of the platform which describe how the system should work and behave so as to deliver the desired functional requirements to satisfy user's needs and expectations. The technical requirements are categorized in 6 groups namely general requirements, connectivity and communication requirements, data curation requirements, data storage requirements, user interface requirements and security requirements.

Chapter 4 describes the overall architecture of the system. Where possible open-source IT tools and programming stacks were preferred while all components have been selected carefully to ensure the robustness, flexibility, and extensibility of the final solution. The DJANGO web framework will serve as the back-end core of the web-based FLOODGUARD platform. PostgreSQL database with PostGIS extension will be used to store all types of numeric and geo-spatial information that needs to be managed by the platform, while GeoServer will implement the web mapping service that will provide high quality maps of environmental information to the end users through their web-browsers. The development of the Graphical User Interface will be based on React JavaScript for the interactive web elements and user controls, and on Leaflet for the interactive mapping environment of FLOODGUARD platform. At the same time the platform facilitates data exchange and communication of the different technological components and workflows that need to be coupled for operationalizing the complete service line of FLOODGUARD. A rich set of API services have been created to allow the individual workflows (e.g., Meteorological, Hydrological and Hydraulic models) to communicate with each other as well as to push the generated data into the Graphical User Interface.

Finally, Chapter 5 presents an overview of the IT components that have been adopted for the implementation of the operational FLOODGUARD platform which is available at le at <https://20.23.54.98/>.

Table 1-1. Working group.

NAME	QUALIFICATIONS
Apostolos Tzimas, Coordinator	Civil Engineer, MSc in Science and Technology Policy
Evangelos Romas	Civil Engineer, MSc in Water Resources and Environment
Alexandros Ziogas	Civil Engineer, PhD, MSc Water Resources and Environment
Kiriakos Kandris	Civil Engineer, PhD, MSc
Hundecha Yeshewatesfa	Civil Engineer, PhD, MSc Water Resources Engineering and Management
Hans Bjorn	Civil Engineer, MSc
Aristides Bartzokas	Professor, Physics, PhD, MSc, Meteorology and Climatology
Aikaterini Lioni	Geologist, MSc

Table 1-2. Review and Receipt Committee of the Contracting Authority.

Permanent Members	
Dr Eleni Athanasiou	Head of Department of Flood Risk - Water Scarcity Management & Management of Water Demand, General Secretariat for Natural Environment & Water, Ministry of Environment & Energy
Konstantinos Papaspyropoulos	Department of Costing and Pricing of Water Services, General Secretariat for Natural Environment & Water, Ministry of Environment & Energy
Stylianios Koutrakis	Department of Flood Risk - Water Scarcity Management & Management of Water Demand, General Secretariat for Natural Environment & Water, Ministry of Environment & Energy
Reserve Members	
Athanasia Pardali	Department of Flood Risk - Water Scarcity Management & Management of Water Demand, General Secretariat for Natural Environment & Water, Ministry of Environment & Energy
Anna Fokaefs	Department of Flood Risk - Water Scarcity Management & Management of Water Demand, General Secretariat for Natural Environment & Water, Ministry of Environment & Energy
Dr Dionysios Marinos	Department of Flood Risk - Water Scarcity Management & Management of Water Demand, General Secretariat for Natural Environment & Water, Ministry of Environment & Energy

2 FUNCTIONAL REQUIREMENTS

The FLOODGUARD platform is a web-based interactive platform that allows user to visualize the forecasted information from the 3 coupled models, i.e., the meteorological model, the hydrological model, and the hydraulic model. The identified user requirements are actually formulating the core features and functionalities of the operational FLOODGUARD platform and will be implemented as individual components. The following table (Table 2-1) provides a list of the desired functional requirements for the Meteorological application of the FLOODGUARD platform. Table 2-2 to Table 2-4 provide a list of the desired functional requirements for the early warning system of the FLOODGUARD platform. Finally, Table 2-5 presents general functional requirements of the overall system.

2.1 METEOROLOGICAL APP

Table 2-1 provides a list of the desired functional requirements for the Meteorological application of the FLOODGUARD platform.

Table 2-1. Functional requirements for Meteorological app.

Specification	Description
Select meteorological model setup	Select the meteorological model setup to display through a dropdown menu.
Select meteorological parameter	Select the meteorological parameter to visualize through a dropdown menu.
Select exact day and time of the forecast	Use a slider to select the desired timestep from the forecasting period.
Display web-maps with meteorological parameter	View raster maps for selected parameter for any timestep within the next 5 days.
Display timeseries for the forecasting period	Display on a graph the 5-day forecast of the selected meteorological parameter for any point on the map.
Display skill	Display the points where the meteorological model is evaluated and provide performance indicators on a monthly basis.
Display basemap layers	Use different layers as a basemap for the web-map window (e.g., satellite, topographical, streets)
Display additional information layers in the web-map	Display layers with additional geographical information (e.g., cities, country borders, water bodies) in the map to facilitate user navigation and identification of areas of interest.

2.2 HYDROLOGICAL APP

Table 2-2 provides a list of the desired functional requirements for the Hydrological application of the FLOODGUARD platform.

Table 2-2. Functional requirements for Hydrological app.

Specification	Description
Select hydrological model setup	Select the hydrological model setup to display through a dropdown menu.
Select hydrological parameter	Select the meteorological parameter to visualize through a dropdown menu.
Display hydrological catchments	The user can view on the map the upstream hydrological basins contributing to Evros river.
Display flow lines and outlet points of hydrological catchments	The user can view on the map the outlet (discharge) points of the hydrological catchments and a schematical view of the river flow lines.
Display timeseries for the forecasting period	Display on a graph the 5-day forecast of the selected hydrological parameter for any point on the map.
Display Early Warning Points	Display in the map the points where hydrological thresholds (river discharges) are defined, colorized with different colors depending on the warning level.

2.3 HYDRAULIC APP

Table 2-3 provides a list of the desired functional requirements for the Hydraulic application of the FLOODGUARD platform.

Table 2-3. Functional requirements for Hydraulic app.

Specification	Description
Select hydraulic model setup	Select the hydraulic model setup to display through a dropdown menu.
Select hydraulic parameter	Select the hydraulic parameter to visualize through a dropdown menu.
Select timestamp	Use a slider to select the desired timestep from the forecasting period.
Display web-maps with results of hydraulic forecasting	View raster maps for selected parameter for any timestep within the next 5 days.
Display timeseries for the forecasting period	Display on a graph the 5-day forecast of the selected hydraulic parameter for any point on the map (within the computational area of HECRAS).
Display basemap layers	Use different layers as a basemap for the web-map window (e.g., satellite, topographical, streets).
Display additional information layers in the web-map	Display layers with additional geographical information (e.g., cities, country borders, water bodies) in the map to facilitate user navigation and identification of areas of interest.

Display Early Warning Points	Display in the map the points where water depth thresholds are defined, colored with different colors depending on the warning level.
------------------------------	---

2.4 EARLY WARNING SYSTEM APP

Table 2-4 provides a list of the desired functional requirements for the Early Warning System of the FLOODGUARD platform.

Table 2-4. Functional requirements for the Early Warning System.

Specification	Description
Define alarm levels for river discharges.	The user may define different discharge thresholds for any point of interest.
Send email notifications from hydrological early warning system.	Notify through email a specific number of recipients in case hydrological thresholds are likely to be exceeded according to the hydrological forecasting.
Define alarm levels for water depths	The user may define different water depth thresholds for any point of interest.
Send email notifications from hydraulic early warning system.	Notify through email a specific number of recipients in case water depth thresholds are likely to be exceeded according to the hydraulic forecasting.

2.5 GENERAL REQUIREMENTS

Table 2-5 provides a list of some general functional requirements of the FLOODGUARD platform.

Table 2-5. General requirements of the FLOODGUARD platform.

Specification	Description
Web-based access	FLOODGUARD platform should be accessible through a web browser without need for installation of additional software.
Authorized platform access	Access to the FLOODGUARD operational platform will be given only to authorized users through appropriate credentials (username/password).
Multilingualism	The platform should be available in Greek and English languages, and provision to support additional languages at a later stage.
Info boxes	The platform should contain info boxes with short descriptive material and quick summaries of information presented in each application.

3 TECHNICAL REQUIREMENTS

This section provides a list of the technological requirements of FLOODGUARD operational platform. Technical requirements are grouped into the following categories:

- GA** General Architectural Requirements
- CC** Connectivity and Communication Requirements
- DC** Data Curation Requirements
- DS** Data Storage Requirements
- UI** User Interface Requirements
- SE** Security Requirements

Sections 3.1 to 3.6 provide the requirements identified in the FLOODGUARD project for the above-mentioned categories.

3.1 GENERAL REQUIREMENTS

Table 3-1 provides a listing of the general architectural requirements that will be considered during the implementation of the FLOODGUARD platform.

Table 3-1. General requirements identification.

GA-01	Remote access The FLOODGUARD platform infrastructure shall be accessed through the Internet. Working environment should include internet connectivity (cable or wireless).
GA-02	Simplified access A mechanism for accessing the platform through standard devices is needed. Desktop PC or laptop or personal mobile devices shall be used as a user terminal.
GA-03	Easy access Browser-based remote administration. Platform administration will be performed through a standard web browser without the need to install additional plugins or software.
GA-04	Ensure scalability The system must be extensible and flexible to handle increasing numbers of users, case studies, models etc.

3.2 CONNECTIVITY AND COMMUNICATION REQUIREMENTS

Table 3-2 provides a listing of the connectivity and communication requirements that will be considered during the implementation of the FLOODGUARD platform.

Table 3-2. Connectivity and communications requirements identification.

CC-01	Web-based access The services should use encryption for secure communications between the FLOODGUARD servers. All communications will be implemented according to the Hypertext Transfer Protocol Secure (HTTPS) standards.
--------------	---

CC-02	Ensure exploitation of existing workflows The system should be able to connect, acquire and deliver data from and to external databases and services through REST APIs.
CC-03	Integrate data from existing operational systems The system should be able to connect and acquire data from ground sensors and IoT devices of different manufactures in real-time or scheduled intervals.
CC-04	Integrate data flows from existing operational systems The system should be able to connect and acquire data from existing operational workflows (e.g., global meteorological service providers) and data management systems in real-time or scheduled intervals.
CC-05	The system must be easily expandable to handle new functionalities and services The system should be able to link different service components.

3.3 DATA CURATION REQUIREMENTS

Table 3-3 provides a listing of the data curation requirements that will be considered during the implementation of the FLOODGUARD platform.

Table 3-3. Data curation requirements identification.

DC-01	Ensure system integrity The system should be able to cleanse the datasets (e.g., remove duplicates, identify inconsistent values, outliers, or values with wrong types).
DC-02	Ensure data accuracy and reliability Constraints should be used to limit the type of data that can be stored in each column of database tables. Data transactions that violate constraints are automatically aborted by the database engine.
DC-03	Ensure system reliability The system should be able to curate large datasets in a timely and efficient manner.
DC-04	Ensure system interoperability The system should allow for efficient transformation of data between various formats (e.g., netcdf, geotiff, csv, ascii)

3.4 DATA STORAGE REQUIREMENTS

Table 3-4 provides a listing of the data storage requirements that will be considered during the implementation of the FLOODGUARD platform.

Table 3-4. Data storage requirements identification.

DS-01	Ensure system flexibility The system should be able to store and manage various types of datasets including numerical and spatial datasets in raster and vector format.
--------------	---

DS-02	Ensure system performance Setup of different data storage environments should be allowed in parallel to accommodate different storage needs (Relational Databases, File systems, FTP servers).
DS-03	Ensure system integrity and scalability Maintain and backup large amounts of data.

3.5 USER INTERFACE REQUIREMENTS

Table 3-5 provides a listing of the Graphical User Interface requirements that will be considered during the implementation of the FLOODGUARD platform.

Table 3-5. User interface requirements identification.

UI-01	Graphical User interfaces will be jointly co-developed with end-users FLOODGUARD operational platform shall be provided through a web-based Graphical User Interface.
UI-02	Ensure compatibility FLOODGUARD will be supported by all modern web-browsers e.g., Google Chrome, Microsoft Edge, Mozilla Firefox, Opera, Safari.
UI-03	Ensure service transferability Multilingual Interfaces. System design should allow for easy installation of additional languages.
UI-04	Ensure platform usability The system should be able to perform queries over the stored data.
UI-05	Enhance user experience The system should be able to visualize real-time and forecasted modeled data using chart and graphs.
UI-06	Enhance user experience The system should be able to visualize stored and forecasted spatial data using web mapping.

3.6 SECURITY REQUIREMENTS

Table 3-6 provides a listing of the security requirements that will be considered during the implementation of the FLOODGUARD platform.

Table 3-6. User interface requirements identification.

SE-01	Ensure confidentiality Security protocols to protect user data over the Internet, should be based on HTTPS, SSL
SE-02	Enforce strong password policies <ul style="list-style-type: none"> •User authentication should be based on the combination of username and password credentials •Allow only complex passwords •Account lockout or temporal suspension for

	<p>unsuccessful login attempts •Apply password rotation •Store only hashed passwords and use password salting •Do not require repetitive password input as long as the user session stays active</p>
SE-03	<p>Set tight server permissions</p> <ul style="list-style-type: none"> •Use firewalls and allow only specific ports for SSL (80, 443) and Remote Administration (SSH 22 or RDP 3389) •Disable register_globals on the server to prevent possible XSS problems in third-party scripts •Disable users from embedding multimedia objects (like Flash) within texts using explicit EMBED and OBJECT tags in their HTML
SE-04	<p>Ensure data integrity</p> <ul style="list-style-type: none"> •Set a root user password in servers and system database •Turn off network access •Use separate data schemas for each application •Keep database server on separate machine from web server •Automate regular backups and store them at different physical machines.
SE-05	<p>Ensure integrity and availability, minimize system downtime</p> <ul style="list-style-type: none"> •Remove unnecessary/obsolete services, modules, add-ons, and plugins •Automate procedures for backups at regular intervals and test system restoration functionalities •Apply security updates and general software updates, using configuration and patch management and deployment tools
SE-06	<p>Ensure integrity</p> <p>Suitable Network Security Architecture</p> <ul style="list-style-type: none"> •The system should offer native security and encryption mechanisms •Stick to the principle of separated environments for different modelling components •If possible, apply segmentation to applications and functions to further improve security and make security measures scalable.

4 PLATFORM ARCHITECTURE

4.1 OPEN SOURCE ARCHITECTURE

The FLOODGUARD operational platform needs to store, process and manipulate various dataset that are being generated by the modelling forecasted components (i.e., meteorological, hydrological and hydraulic). For the development of the FLOODGUARD platform free and open-source software (FOSS) will be used whenever possible. Some of the advantages of employing FOSS are summarized below:

Increased Security. Security is a concern for every application today. While acquiring software based on source code that is open for anyone to edit, might sound like a bad idea, the truth is that the availability of the source code is actually one of the greatest advantages of this technology. Due to the source code being accessible, developers can locate, and fix bugs faster than in commercial products. With the code being open to the public, the so-called Linus's law applies: "Given enough eyeballs, all bugs are shallow".

Flexibility. Another great benefit of open-source software is increased flexibility. By acquiring an open-source solution, businesses are also avoiding the vendor lock-in that normally applies with licensed software. The lock-in means that businesses are forced to follow the software provider's requirements, priorities, updates and prices.

Higher Quality. With open-source software, you are likely to receive a higher quality than with a licensed software. This is simply because it has been created by thousands of developers worldwide ensuring increased security, enhanced toolsets, and innovative features. Open-source developers add what they think the software needs in order to be better instead of traditional software developers that provide what they think the customer want.

A World of Support. The open-source communities are incredibly active online, meaning that there are often excellent support options available for any IT team free of charge, such as forums, live support chat, and documentation.

Lower costs. Open-source software can be obtained free of charge and does not entail fees for updates. Even using paid support options by open-source developers, this is normally far more affordable than support offered by the software companies.

Interoperability. Open-source software developers usually abide by open specifications and standards for data and web services that allows seamless interaction with other products.

Collaboration. Enables collaboration with others, as the software is accessible and can be used by anyone.

4.2 HARDWARE LAYER

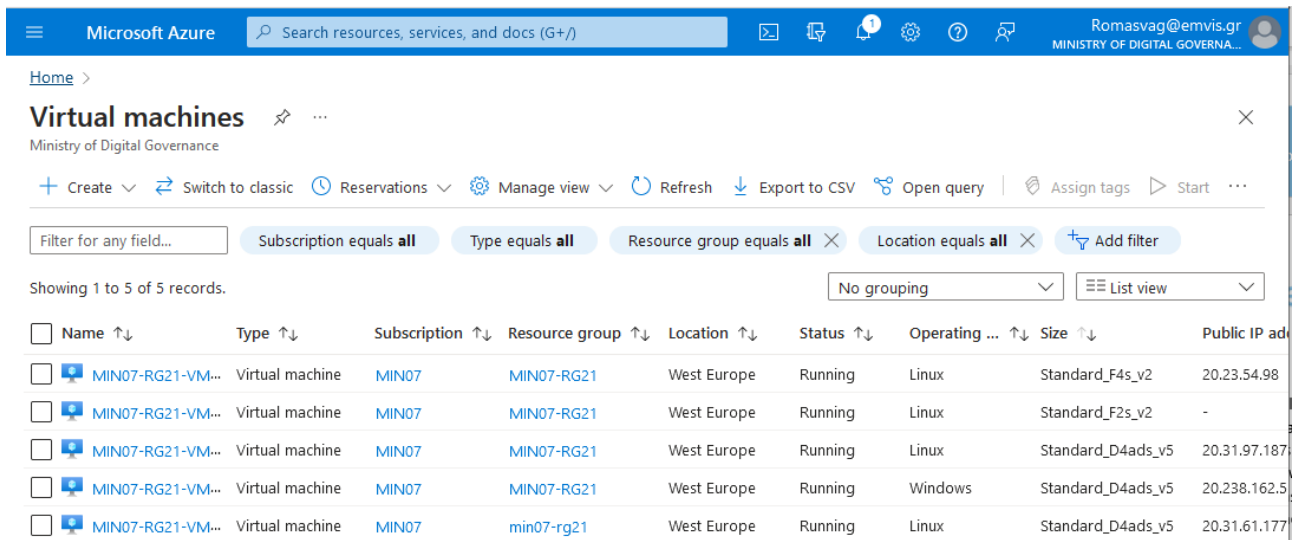
FLOODGUARD platform will be deployed using cloud computing services. This means that hardware layer will not have a physical instance on our side, instead it will be held on a GCloud service using Microsoft Azure platform. This approach offers great advantages especially during the development stages of a project, since developers can focus on code implementation without worrying about hardware maintenance and parameterization, installation of OS updates, etc. that are usually required in case of a local (on premises)

datacenter. Furthermore, cloud service providers are employing state of the art solutions for automated data backup, and physical redundancy over multiple data centers, offering unmatched reliability and dependability. The following table (Table 4-1) summarizes some advantages of cloud computing services.

Table 4-1. Advantages of cloud-based services

Advantages of cloud-based services
Low costs: Most services are offered under the “pay as you operate” rule so they are free from high capital expenditure.
24 X 7 Availability: Most of the cloud providers are truly reliable in offering their services, with most of them maintaining an uptime of 99.9%.
Scalability and flexibility: Hardware resources like CPU, storage and RAM can be upscaled or downscaled on demand, depending on the development stage of the project.
Accessibility: Cloud computing offers the advantage of working from anywhere across the globe, as long as there is an internet connection.
Automated Updates on Software: In cloud computing, the server suppliers regularly update your software including the updates on security, so that you do not need to waste time on maintaining the system.
Integrity and Security: Cloud computing services implement state-of-the-art security protocols, encryption techniques, and backup processes.

The FLOODGUARD project is deployed in 5 individual virtual servers provided by the GCloud platform through Microsoft Azure (Figure 4-1).



The screenshot shows the Microsoft Azure portal interface for the 'Virtual machines' section. It displays a table with 5 records, all of which are 'Virtual machine' instances in the 'West Europe' region, running in a 'Running' state. The instances are named 'MIN07-RG21-VM...' and are associated with the 'MIN07' subscription and 'MIN07-RG21' resource group. The operating systems are Linux (3 instances) and Windows (1 instance). The sizes are 'Standard_F4s_v2', 'Standard_F2s_v2', and 'Standard_D4ads_v5'. The public IP addresses are 20.23.54.98, -, 20.31.97.187, 20.238.162.5, and 20.31.61.177.






Name	Type	Subscription	Resource group	Location	Status	Operating ...	Size	Public IP ad
MIN07-RG21-VM...	Virtual machine	MIN07	MIN07-RG21	West Europe	Running	Linux	Standard_F4s_v2	20.23.54.98
MIN07-RG21-VM...	Virtual machine	MIN07	MIN07-RG21	West Europe	Running	Linux	Standard_F2s_v2	-
MIN07-RG21-VM...	Virtual machine	MIN07	MIN07-RG21	West Europe	Running	Linux	Standard_D4ads_v5	20.31.97.187
MIN07-RG21-VM...	Virtual machine	MIN07	MIN07-RG21	West Europe	Running	Windows	Standard_D4ads_v5	20.238.162.5
MIN07-RG21-VM...	Virtual machine	MIN07	min07-rg21	West Europe	Running	Linux	Standard_D4ads_v5	20.31.61.177

Figure 4-1. The 5 VM servers of FLOODGUARD in Microsoft Azure.

The VM01 server runs on Ubuntu and is used for the meteorological simulation. The VM02 server runs on Centos and is used for the hydrological simulation. VM03 is a Windows server machine which runs the hydraulic simulation with HECRAS software. VM04 and VM05 are two Centos machines which run the web

and database server. The following table (Table 4-2) provides details on the operating system and the hardware specifications of the virtual machines used in the FLOODGUARD project.

Table 4-2. Technical specifications of FLOODGUARD's virtual machines.

Server	Description	Operating System	vCPUs	RAM
 MIN07-RG21-VM01	Meteorological App	Linux (ubuntu 22.04)	4	16 GiB
 MIN07-RG21-VM02	Hydrological App	Linux (centos 7.9.2009)	2	4
 MIN07-RG21-VM03	Hydraulic App	Windows (Windows Server 2022 Datacenter Azure Edition)	4	16
 MIN07-RG21-VM04	Web server	Linux (centos 7.9.2009)	4	8
 MIN07-RG21-VM05	Database server	Linux (centos 7.9.2009)	4	16

4.3 INTEGRATION WITH EXISTING WORKFLOWS

FLOODGUARD platform aims to integrate different operational workflows that are supported by different modelling components. To allow them to communicate and exchange information in an efficient way the FLOODGUARD platform focuses on facilitating the exchange, usage, and repurposing of information between different systems and platforms and ensure a high level of interoperability through a rich set of Application Programming Interfaces (APIs).

Furthermore, the FLOODGUARD platform has established real-time communication with existing monitoring stations located in rivers and streams in the wider Evros area. These data are retrieved every 1 hour through API and are stored in the central database, so they are available to the hydrological model for data assimilation (see also del 4.6.5.B, section 3.4). The following table (Table 4-3) provides a list of the measuring stations where operational communication has been established in the FLOODGUARD platform.

Table 4-3. Monitoring stations with real time data retrieval.

Station	Param id	Parameter	Measuring frequency
Erythropotamos	9824	Discharge	15 mins
Ardas	10014	Discharge	15 mins
Pythio	9831	Discharge	15 mins

4.4 HIGH LEVEL ARCHITECTURE

The FLOODGUARD platform is a web-based mapping application that provides forecasted information generated by the meteorological, hydrological and hydraulic modelling components. The FLOODGUARD platform is able to:

- Facilitate the interoperability between the meteorological and the hydrological modelling components by managing and integrating all data streams required for the operationalization of the forecasting service line.
- Facilitate the interoperability between the hydrological and the hydraulic modelling components by managing and integrating all data streams required for the operationalization of the forecasting service line.
- Provide river discharge measurements in-situ monitoring stations installed in tributaries of Evros river.
- Visualize forecasts produced from the different meteorological model setups that have been deployed for the wider Evros area.
- Visualize forecasts produced from the different meteorological model setups that have been deployed for the wider Evros area.
- Visualize forecasts parameters generated from the hydrological model (i.e., river discharges at all upstream catchments contributing to Evros river including both the Greek and Bulgarian areas.
- Visualize water depth forecasts generated from the 2-dimensional hydraulic model, within the Greek part of Evros river.
- Generate and display Early Warnings based on forecasted river discharges and water depths, and send email notifications.

After carefully analyzing the technical and functional requirements that have been identified, we are able to design the high-level architecture of the platform that will allow for the realization of the envisaged workflows and functionalities of the platform. A modular way that ensures maintenance and extensibility has been adopted for selecting the different platform components and defining their logical hierarchy.

Web Framework. A Web Framework provides a standardized way for managing the different components of a web application and offers significant advantages over pure code solutions in terms of efficiency, security and code reusability.

Web Mapping Service. A Web Mapping Service will allow FLOODGUARD platform to handle spatial data and offer dynamic web maps of monitored or modelled water quantity and quality characteristics over the Evros area.

Storage layer. A combination of a relational database and a file system will manage all data streams that need to be retrieved, stored and served to the end users through the web platform.

Application Programming Interfaces (APIs). The API is a software intermediary that allows two different applications to interact and talk to each other through a set of predefined rules and endpoints. APIs are supported and designed in the Web Framework.

Web Server. The web server processes and interprets all the requests made by the end-users through the interactive web-application and delivers the web pages through the web-browser.

Graphical User Interface (GUI). The graphical user interface is the front-end of the platform that offers users a single endpoint of visualizing and accessing the available information and interacting with the application.

Data Curation. An offline module to ensure the integrity of the datasets manipulated by the FLOODGUARD database through their lifecycle.




Security mechanism. A robust security mechanism is considered essential for ensuring the integrity of the data and the offered services. An access control module will be designed to assign specific permissions to the content of the FLOODGUARD platform.





4.5 WEB FRAMEWORK

4.5.1 State-of-the art analysis

Web Frameworks are software frameworks that facilitate the development of web applications, by offering common libraries and templates that promote code reusability instead of the “reinventing the wheel” approach that a pure-code approach would entail. As Web Frameworks automate the overhead associated with common activities performed in web development, they allow for fast and agile development minimizing the required resources. As this technology has become the definitive solution in creating modern interactive web-applications there are numerous popular Web Frameworks. A non-exhaustive list of the most popular Web Frameworks is presented in Table 4-4.

Table 4-4. State of-the-art analysis for Web Frameworks

Web Framework	Characteristics
<p>Django</p> 	<p>Language: Python Type: Server-Side Web application Framework Architecture: Model-View-Controller Pros: ORM, human-readable, fast development, large community, built-in Admin Panel</p>
<p>Laravel</p> 	<p>Language: PHP Type: Server-Side Web application Framework Architecture: Model-View-Controller Pros: Fast Development, Security, Email service, built-in authentication</p>
<p>Ruby on Rails</p> 	<p>Language: Ruby Type: Server-Side Web application Framework Architecture: Model-View-Controller Pros: ORM, human-readable, fast development, large community, authorization, security</p>

Web Framework	Characteristics
ASP.NET 	Language: C# Type: Server-Side Web application Framework Pros: Stable, Large support, Advanced UI Controls
Angular 	Language: Typescript Type: Front-end Web application Framework Architecture: Component Based Pros: Improved Speed and Performance, High quality
Express 	Language: JavaScript Type: Server-Side Web application Framework Pros: Easy to Learn, Same language frontend and backend
Spring 	Language: Java Type: Server-Side Web application Framework Architecture: Model-View-Controller Pros: ORM, large community, security, authorization

4.5.2 Django web framework

The development team has selected the Django Web Framework which is based on Python programming language. Python is a widely used programming language for scientific applications with a large number of open-source libraries. It is a high-level object-oriented language which offers high performance, and it is easy to use at the same time.

Django implements a collection of packages or modules which allow the developers to focus on their own code without having to handle the low-level components of a web application such as protocols, sockets, or process/thread management.

4.5.2.1 MODEL-VIEW-CONTROLLER LAYOUT

Django, like other modern web frameworks, supports the model-view-controller (MVC) design pattern. The MVC design pattern specifies that an application consist of a data **model**, a **view** for presentation of information, and a **controller** that lies between the view and the model (Figure 4-2).

The **model** (for example, the data information) contains only the pure application data; it contains no logic describing how to present the data to a user. A model is the single, definitive source of truth about data. It contains the essential fields and behaviors of the data that need to be stored.

The **view** (for example, the presentation information) presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.

Finally, the **controller** (for example, the control information) exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to those events. In most cases, the reaction is simply a call on a method of the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view.

The MVC programming paradigm allows us to speed up FLOODGUARD platform development by separating user interface and business logic layers.

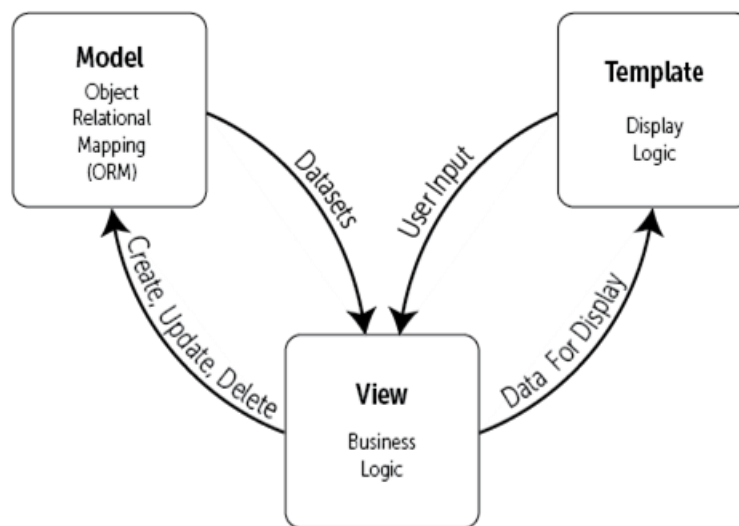


Figure 4-2. The Model-View-Controller layout used in for the development of FLOODGUARD Applications

4.5.2.2 APPLICATIONS

Each concrete group of functionalities of the FLOODGUARD platform will be built as a separate application within Django implementing their own MVC design. FLOODGUARD platform consists of 3 different applications:

- the Hydrological Forecasting application
- the Hydrological Forecasting application
- the Hydraulic Forecasting application
- the Early Warning system

4.6 WEB MAPPING

FLOODGUARD will serve through its web-based platform maps for various meteorological, hydrological and hydraulic parameters generated by the modelling components. Several machines are required to create,

serve and use a web map as depicted in the figure below (Figure 4-3). However, these separate tiers can be hosted in one physical machine.

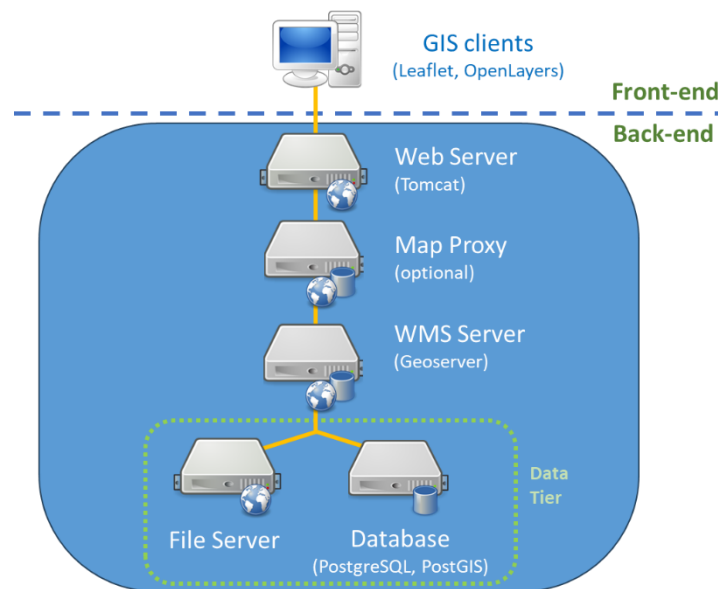


Figure 4-3. System architecture for interactive web mapping

The **database and file servers** will store all project data and it will include regular backup scripts to ensure data protection from corruption and/or loss of data.

The **WMS server** accomplishes tasks related to preparing, transforming, reprojecting, tiling, and serving raster data. This server also includes a web-based administration console, where we can create data stores and expose them as layers.

An optional **MapProxy** could be used to speed up mapping applications by pre-rendering maps and storing them in a local cache.

The **Web Server**, also called a proxy server, acts as a web entry point into our platform and hosts our web map application code (HTML, JavaScript, CSS files).

4.6.1 Geoserver

The Web Mapping Service (WMS) will be implemented by Geoserver. GeoServer is an open source server written in Java for sharing geospatial data. It is designed for interoperability and excels at publishing any major spatial data source using open standards. Geoserver is suitable for handling very large raster or vector datasets, produces high quality rendering of maps and can manage hundreds of map layers easily. Apart from WMS, GeoServer implements other industry standard OGC protocols such as Web Feature Service (WFS), and Web Coverage Service (WCS). GeoServer acts as a middleware between the backend database and the Graphical User Interface.

Geoserver will be configured to serve the following layers of information:

- **Basemap layers** that are responsible for providing geographic context for our map and allow our map to be easily interpreted. Geoserver supports multiple providers including free ones generated by OpenStreetMap as well as proprietary datasets owned by HERE, Google, Tencent, TomTom, and others. FLOODGUARD platform will use only free datasets mainly from OpenStreetMap.
- **Thematic layers** (the layers that are the focus of the map), are brought in as one or more separate web services, and placed on top of the basemap. Also known as business or operational layers, they work together with basemap layers to form an effective web map. The thematic layers of FLOODGUARD include prefectures, countries boundaries, river water bodies, etc.
- **Interactive elements** assist in learning more about the layers in the map. They include popups that appear when we click on a feature, legends that are useful for interpreting the data ranges, charts and graphs that are drawn in a separate part of the page, slider bars that adjust the time slice of data displayed in the map, analysis tools, ability to toggle layers, adjust layer transparency, search for a location, and so forth. These elements make the map come alive and hence we should include the most useful to the end-users, without however overwhelming them with options or making the tasks too complicated.

4.7 DATABASE AND FILE SYSTEM

When designing our data tier, we had to decide whether to store data in a series of files, or in a database that incorporates spatial data support. A file-based data approach is simpler and easier to set up than a database if datasets are not changing on a frequent basis and are of manageable size. However, databases are more appropriate when we have a lot of data to store or when data is being edited frequently by different parties. In our design we propose a hybrid storage solution consisting of both a Database Server and a File System.

In database we store:

- Usernames and passwords
- The forecasts generated by the Meteorological model
- The forecasts generated by the Hydrological models
- Data retrieved from in-situ sensors
- Other information required by the front end (model setup names, parameters names, units, etc)

In the file system we store:




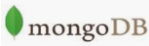
- The forecasts generated by the Meteorological model
- The forecasts generated by the Hydrological models
- The forecasts generated by the Hydraulic models


Note that some datasets are stored both in the database and in the filesystem to increase flexibility and allow easier communication between modelling components.

4.7.1 State of the art analysis

FLOODGUARD platform needs to store and handle large amounts of data that are being generated on a daily basis by the forecasting models. The following table (Table 4-5) presents an analysis of some the most widely databases used for web GIS applications.

Table 4-5. State-of-the-art analysis for Database Servers

Database	Characteristics
 MySQL	<p>Description: MySQL enables data to be stored and accessed across multiple storage engines, including InnoDB, CSV, and NDB. MySQL is also capable of replicating data and partitioning tables for better performance and durability</p> <p>Type: RDBMS database</p> <p>License: GNU General Public License</p> <p>Pros: The world's most popular open-source database, Full text search, ACID compliant</p>
 PostgreSQL	<p>Description: mature open-source Relational Database Management System that is based on the object relational DBMS Postgres, with great support for geospatial data.</p> <p>Type: RDBMS database</p> <p>License: MIT</p> <p>Pros: Extremely popular open-source DB, Fast, Support variety of data types including geospatial data.</p>
 Cassandra	<p>Description: an open-source distributed database management system designed to operate across multiple commodity servers, originally developed for Facebook. Cassandra is a column-based NoSQL data storage</p> <p>Type: NoSQL database</p> <p>License: Apache License (AL) 2.0</p> <p>Pros: Offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients. Highly scalable and universally available with no single point of failure. SQL-like query language and support search through secondary indexes.</p> <p>Cons: No Support for ACID Properties, No support for Aggregates, Data Duplication, Slow Reads</p>
 MongoDB	<p>Description: an open-source, cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.</p> <p>Type: Object oriented NoSQL database</p> <p>License: Server-Side Public License (SSPL) v1.0</p> <p>Pros: By storing the majority of the data in RAM, query performance in MongoDB is much quicker, Expressive query language, MongoDB query syntax is simple and much easier to understand than SQL, MongoDB has no predefined schema and thus has very dynamic schematics architecture for unstructured data and storage options, Reliable scalability, Uses shards for horizontal scalability which makes it easier to increase storage capacity.</p> <p>Cons: MongoDB's quick speeds and high performance is only possible with the right indexes, With shoddily implemented indexes and out of order composite indexes MongoDB operates at a shockingly slow speed, Relationships in MongoDB are not typically well-defined and the resulting duplicate data sets can be hard to handle</p>

Database	Characteristics
 Rasdaman	<p>Type: Client/server DBMS</p> <p>License: GNU General Public License</p> <p>Pros: Has no limitation in the number of dimensions - it can serve, for example, 1-D measurement data, 2-D satellite imagery, 3-D x/y/t image time series and x/y/z exploration data, 4-D ocean and climate data, and even beyond spatio-temporal dimensions, Large arrays are decomposed into smaller units which are maintained in a conventional DBMS</p> <p>Cons: Not practical for a web app database, mainly used for multidimensional big data, Big learning curve</p>

4.7.2 PostgreSQL

The database of FLOODGUARD platform will be implemented in PostgreSQL. PostgreSQL is an open-source Object-Relational Database Management System (ORDMS), that is being supported by a wide community of developers and volunteers around the world. It is extremely powerful in handling structured data, i.e., data incorporating relations among entities and variables. Additionally, PostgreSQL is fully compatible with Django's MVC design pattern.

PostgreSQL is widely used in the domain of GIS mainly due to PostGIS, a software extension that adds support for geographic objects to the PostgreSQL database. PostGIS supports a variety of geographic features both in raster and vector formats as well as hundreds of topological functions for querying of spatial properties and relationships, typed in SQL. As PostGIS minimizes the usage of hardware resources (i.e., RAM and processing power), it proves to be very fast and stable, while it is also compliant with the Open Geospatial Consortium (OGC) Standards and Simple Features Access (SFA) for SQL.

Every application of the back-end (e.g., Hydrology, Meteorology) will have its own schema which defines a set of attributes of the database, such as tables, columns, and properties. This design technique allows to define different user permissions per schema while also offers an extra layer of security, as in case of a malicious software or user only one database will be affected. Finally, common database tasks like Cleanup, Backup, Restore and Indexes are performed much faster.

4.7.2.1 METEOROLOGICAL FORECASTING SCHEMA.

This schema stores information required by the Meteorological Forecasting application to present the short-term forecasts of meteorological parameters in each computational domain.

- Table **setup**: contains information about the properties of meteorological model setup.
- Table **param**: contains the parameter name and slug along with a foreign key on table *Unit* of default database.
- Table **forecast**: contains three foreign keys. One for *catchment*, one for table *setup* and one for table *param*. Those keys along with the *issued_date* of the forecast are combined to define a unique forecast.
- Table **point**: contains the coordinates of the centers of the computational cells used in meteorological simulation for each different model. Contains foreign key to the *setup* table.

- Table **forecast_Value**: contains one foreign key to the *Forecast* table record. It contains the attribute values of each forecast i.e., target_date (date of prediction) value and member. With this design we can change the model_setup, or parameter without having to change *forecast* table. Deterministic forecasts have a single member value, while probabilistic forecasts may have multiple members.
- Table **setup_users**: Contains foreign keys to the users table in order to define the accessibility of meteorological model setups by different users.

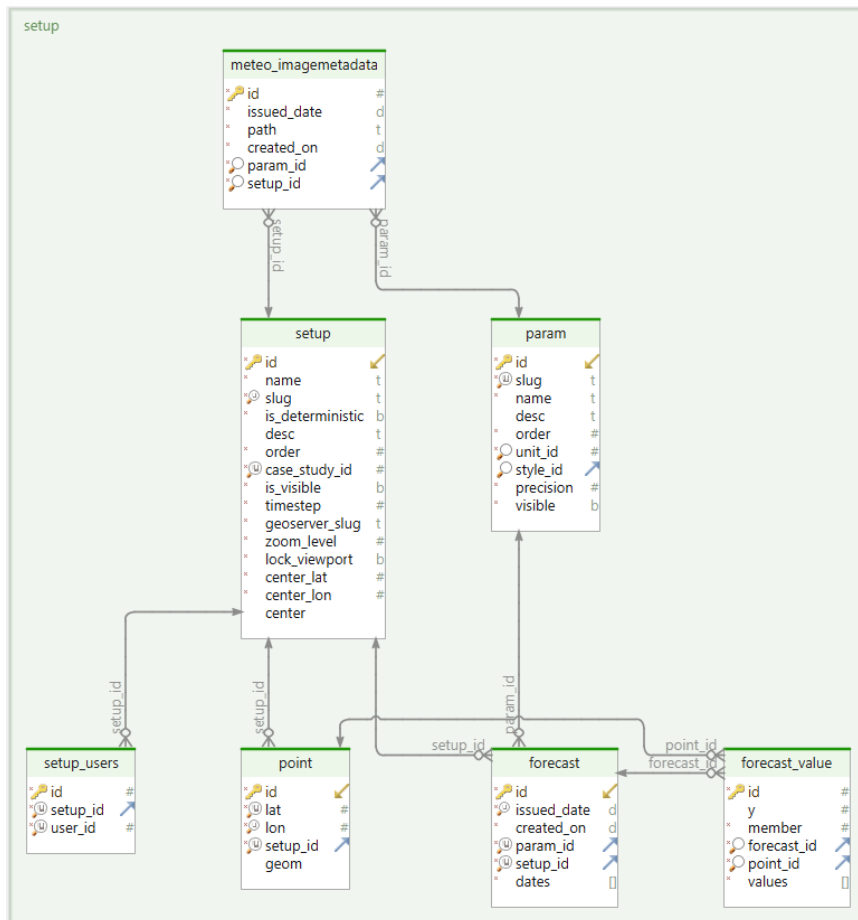


Figure 4-4. Schematic overview of the database schema for the Meteorological Forecasting application.

4.7.2.2 HYDROLOGICAL SCHEMA

This schema stores information required by the Hydrological Forecasting application to present the short-term forecasts of hydrological parameters in the upstream catchments of Evros river. The major tables of this schema are:

- Table **Catchment**: contains the same schema with the shapefile used to store the geometry of the upstream catchments. Its primary key is the *subid* of the case study. Keeps a foreign key to the table *casestudy* of Default schema. When new shapefiles are uploaded to the server, this table is populated automatically.

- Table **Param**: contains the parameter name and slug along with a foreign key on table *unit* of Default database schema.
- Table **Setup**: contains model setups names, slugs and a boolean characterizing if the model setup is deterministic or probabilistic.
- Table **Casestudy_Setup**: contains two foreign keys one for the lake and one of the setup. This table creates a relationship between table *casestudy* and table *setup*. This is necessary since each case-study might have multiple forecasting models.
- Table **Forecast**: Contains three foreign keys. One for catchment, one for model setup and one for param. The combination of those three foreign keys along with the *issued_date* of the forecast define a unique forecast record.
- Table **Forecast_Value**: contains one foreign key to the table *forecast*. It contains the attribute values of each forecast including *target_date* (the date that the forecast refers to), *value* and *member_id* for distinguishing between individual models of probabilistic ensembles. This design allows to change the *model_setup*, *param* or *catchment* attributes without modifying table *forecast*.
- Table **Statistic_Type**: stores ids of different statistic types (e.g., average, min, max, percentiles, standard deviation).
- Table **Forecast_Statistic**: contains one foreign key to table *forecast* and one key to the relevant record of table *statistic_Type*. It stores the values of each statistic that is computed for each forecast. This design allows support to additional statistic types at a later stage.
- Table **Historical_Statistic**: This table stores statistical properties of historical simulations (usually longterm) of hydrological parameters in each hydrological catchment that are used for comparing short-term forecasts with historical datasets. This table has 3 foreign keys, one for table *catchment*, one for table *statistic_type* and one for table *param* which are combined with the day and month of each forecasted parameter.

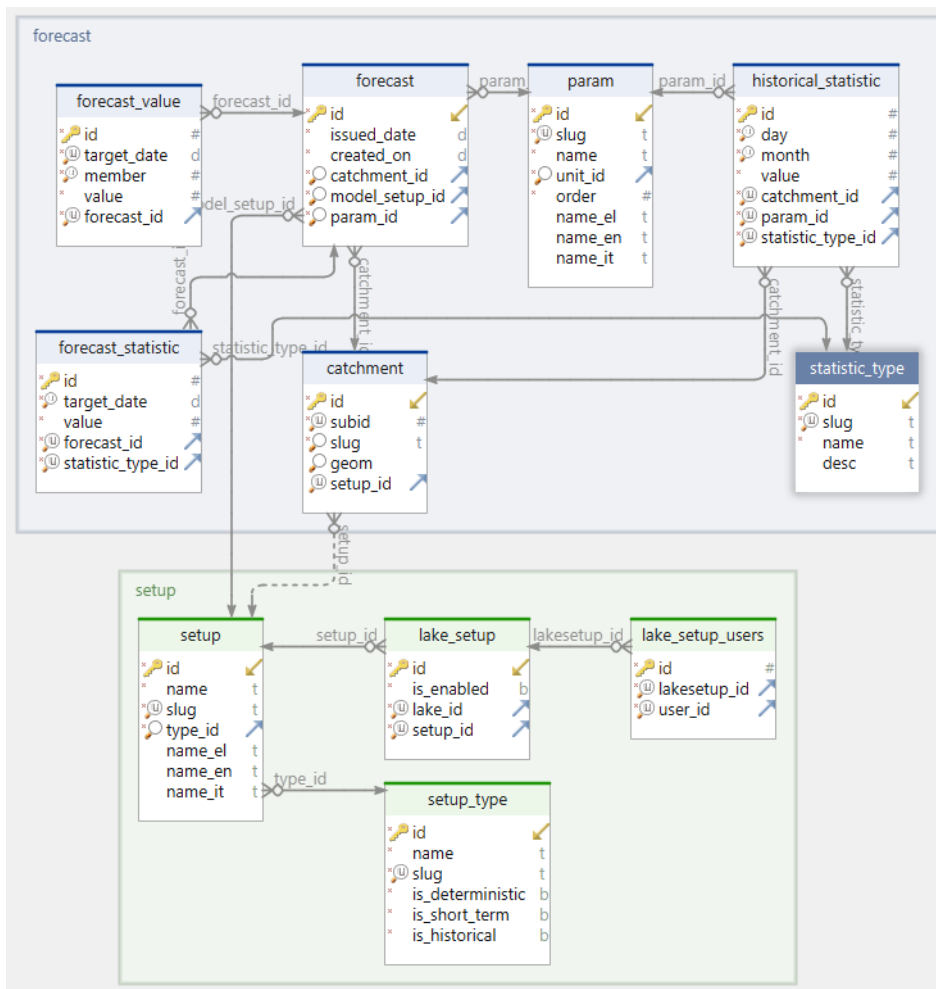


Figure 4-5. Schematic overview of the database schema for the Hydrological Forecasting application.

4.7.2.3 HYDRAULIC SCHEMA

This schema stores information required by the Hydraulic Forecasting application to present the short-term forecasts of hydraulic parameters in the computational grid of the HEC-RAS model. The major tables of this schema are:

- Table **setup**: contains information about the properties of the hydraulic model setup.
- Table **params**: contains the parameter name and slug along with a foreign key on table *unit* of default database.
- Table **setup_users**: Contains foreign keys to the *users* table in order to define the accessibility of hydraulic model setups by different users.

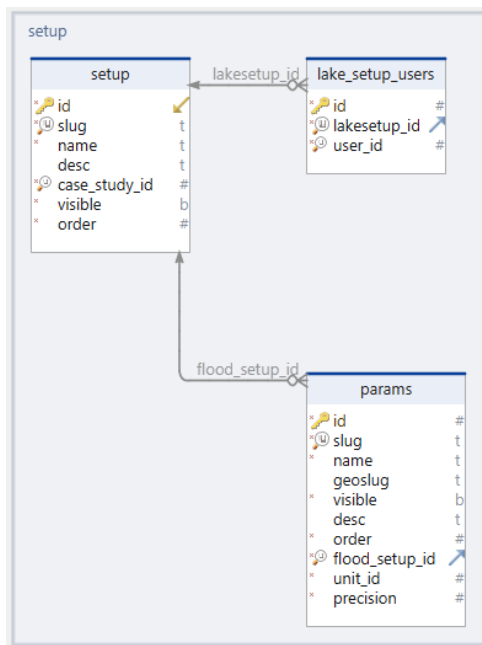


Figure 4-6. Schematic overview of the database schema for the Hydraulic Forecasting application.

4.7.2.4 DEFAULT SCHEMA

This schema contains tables that are used by all applications (e.g., table *units* for storing unit metrics and values) as well as some default tables that are created by the Django framework (e.g., tables *django_migrations* and *django_admin_log*) and tables required for managing user access permissions for each application (e.g., tables *auth_user_groups* and *auth_group_permission*).

- Table **project**: contains information about the properties of all the projects stored in the database (e.g. FLOODGUARD is a project in the database).
- Table **case_study**: contains information about the properties of all the case studies that are stored in the database. Contains a foreign key to the project *table* since all case studies belong to a single project. (e.g., Evros is a case study of the FLOODGUARD project). One project can contain more than one case studies.
- Table **service**: Contains the services (applications) that are available in the platform can support (e.g., Hydrological app, Meteorological app, Hydraulic app).
- Table **case_study_service**: Contains foreign keys to the tables *case_study* and *service*. This table defines which services are available in each case study.
- Table **case_study_users**: Contains foreign keys to the users *table* and the *case_study* table in order to define the accessibility of the case studies by different users.
- Table **base_layers**: contains information and metadata about the layers that are used as base layers in the web-mapping application.
- Table **base_layers_mapping**: contains three foreign keys to tables *base_layer*, *case_study* and *service*. The purpose of this table is to define in which services (apps) each base-map layer will be shown. A single base-map layer can be visible at multiple services and multiple case studies.

- Table **language_project**: contains the languages that are available for each project. Contains two foreign keys on tables *project* and *language*.

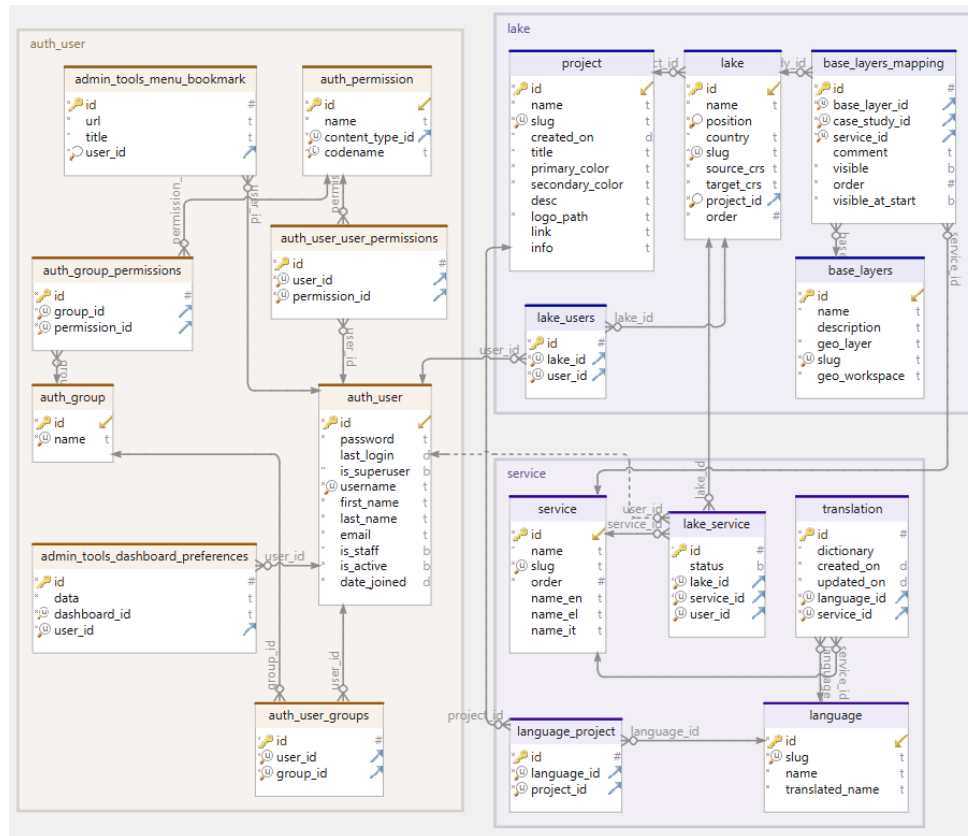


Figure 4-7. Schematic overview of the default schema of the FLOODGUARD platform (used by all applications).

4.7.3 File system

As described previously the results of the modelling components apart from the database are also stored on a file server to increase flexibility and productivity. The forecasted modelling results which are stored to the database are mostly used by the front end (data are fetched to the GUI through an API whenever a user wants to display a map or graph), while the results which are stored in the file server are mostly used from the other modelling components (e.g., the hydrological model retrieves the meteorological forecasts from a predefined folder). The uploading and downloading of these files are performed through SFTP (Secure File Transfer Protocol) and requires user authentication.

All data are stored in the folder **/home/ftpupload** in the DB server Virtual Machine (20.31.61.177). Different folders are created for the 3 different modelling components and each folder contains in turn subfolders named after year, month, day and hour, where the results of the simulations are stored following specific naming conventions. The following figures (Figure 4-8, Figure 4-9) present schematically the structure of the folders used for storing modelling results in the FLOODGUARD project.




/home/ftpupload/				
Name	Size	Changed	Rights	Owner
 HecRas		07/07/2023 13:44:37	rwXrwXrwX	root
 Hydrology		11/07/2023 17:54:54	rwXrwXrwX	root
 Meteorology		04/07/2023 20:08:20	rwXrwXrwX	root
		04/07/2023 14:07:09	rwXrwXrwX	root

Figure 4-8. Structure of the ftpupload folder in the DB server (20.31.61.177) where the folders with the results of the meteorological, hydrological and hydraulic models are stored.

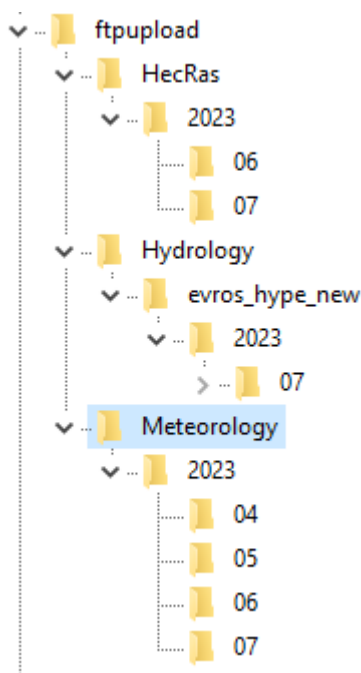


Figure 4-9. Tree view of the results folders in the DB server (20.31.61.177)


4.7.3.1 METEOROLOGICAL MODEL

The meteorological model stores the forecasts produced each month in a separate folder. Within a “month” folder different netcdf files for the 3 meteorological model setups are stored with the following names:

Naming convention	Description
gefs_2023-07-07_12:00.nc	The results of NOAA’s Global Ensemble Forecast System (GEFS) are produced once a day at H12
wrfout_d01_2023_07_07_00:00:00.nc wrfout_d01_2023_07_07_12:00:00.nc	The results of WRF model for the large domain (D01) are produced twice a day at H00 and H12.
wrfout_d02_2023_07_07_00:00:00.nc wrfout_d02_2023_07_07_12:00:00.nc	The results of WRF model for the small domain (D02) are produced twice a day at H00 and H12.

4.7.3.2 HYDROLOGICAL MODEL

The hydrological model runs twice a day and the results are stored in separate folders for each run. For example, the folder `/home/ftpupload/Hydrology/2023/07/07/12` contains the results of the run made at 7th July 2023 at 12 noon. Each folder has different txt files for each simulated parameter (e.g., `timeCOUT.txt` for catchment outflow, `timeWTMP.txt` for water temperature), which contains the forecasted timeseries for all catchments (Figure 4-11).

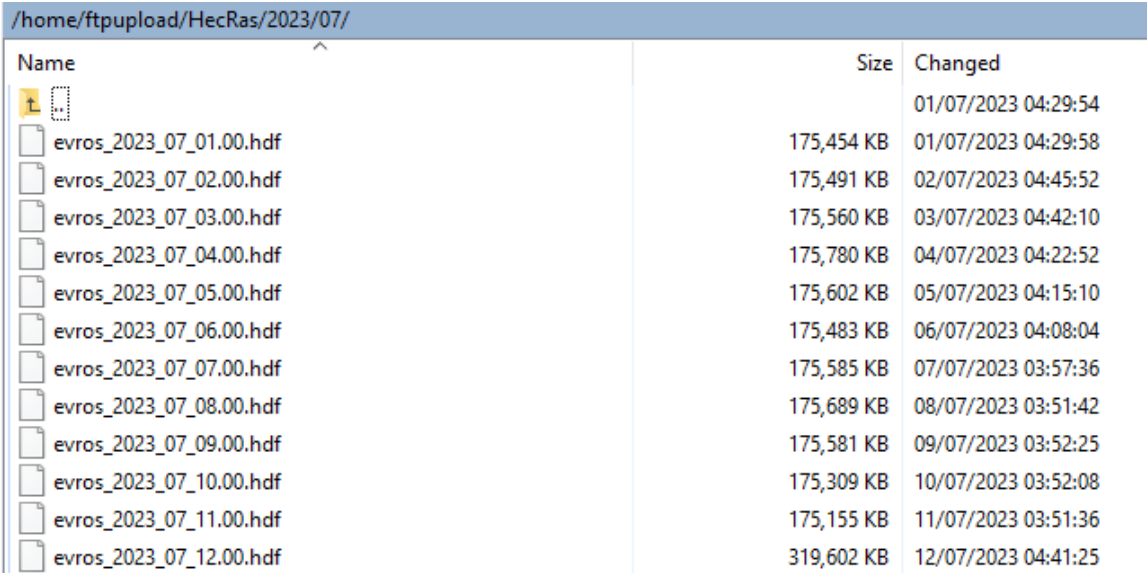


Name	Size	Changed
timeWTMP.txt	970 KB	07/07/2023 13:10:07
timeEVAP.txt	970 KB	07/07/2023 13:10:08
timeCTMP.txt	970 KB	07/07/2023 13:10:08
timeCPRC.txt	970 KB	07/07/2023 13:10:08
timeCOUT.txt	970 KB	07/07/2023 13:10:07
timeCINF.txt	1,056 KB	07/07/2023 13:10:07

Figure 4-10. Folder containing the forecasting results of the hydrological model.

4.7.3.3 HYDRAULIC MODEL

The hydraulic model runs stores its results in HDF (Hierarchical Data Format) format in folders named after the year and the month. Within each folder, files are named following the `YYYY_MM_DD.HH` pattern (see Figure 4-13).



Name	Size	Changed
evros_2023_07_01.00.hdf	175,454 KB	01/07/2023 04:29:54
evros_2023_07_02.00.hdf	175,491 KB	02/07/2023 04:45:52
evros_2023_07_03.00.hdf	175,560 KB	03/07/2023 04:42:10
evros_2023_07_04.00.hdf	175,780 KB	04/07/2023 04:22:52
evros_2023_07_05.00.hdf	175,602 KB	05/07/2023 04:15:10
evros_2023_07_06.00.hdf	175,483 KB	06/07/2023 04:08:04
evros_2023_07_07.00.hdf	175,585 KB	07/07/2023 03:57:36
evros_2023_07_08.00.hdf	175,689 KB	08/07/2023 03:51:42
evros_2023_07_09.00.hdf	175,581 KB	09/07/2023 03:52:25
evros_2023_07_10.00.hdf	175,309 KB	10/07/2023 03:52:08
evros_2023_07_11.00.hdf	175,155 KB	11/07/2023 03:51:36
evros_2023_07_12.00.hdf	319,602 KB	12/07/2023 04:41:25

Figure 4-11. Folder containing the HDF files produced by the HECRAS hydraulic model.

4.8 APPLICATION PROGRAMMING INTERFACE

An Application Programming Interface (API) is a computing interface which defines interactions between multiple software intermediaries. It defines all kinds of calls or requests that can be made, the data formats that should be used, the conventions to follow, etc.

An API can be entirely custom, specific to a component, or it can be designed based on an industry-standard to ensure interoperability. Through information hiding, APIs enable modular programming, which allows multiple developers to use the interface independently of the implementation, without even requiring to understand what is happening behind the scenes. This logic is ideal for FLOODGUARD platform which integrates different computational and modelling workflows.

All APIs are implemented in the Django Web Framework and will be compatible with the Representational State Transfer (REST) software architecture allowing their usage through HTTPs request operations. This approach provides a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (e.g., SOAP, WSDL). Its advantages are that it is platform-independent (runs on any platform regardless of the server and client platform), it is language-independent, standards-based (runs on top of HTTPs) and can easily be used in the presence of firewalls.

In FLOODGUARD, APIs are used in a two-fold way: first as a communication channel between back-end and front-end layers and second for the communication of different services and workflows with the database. All API's require authentication to execute which prevents abuse by unauthorized or malicious users.

4.9 WEB SERVER AND APPLICATION SERVER

A Web Server is necessary to allow our platform to reach the outside world through a web-browser interaction. While a Web Server can serve files (HTML, images, CSS, etc.) directly from the file system, it is not able to communicate directly with Django applications. We therefore need a dedicated Application Server that will feed the requests from web clients (i.e., web browsers) to the applications and return their responses.

When the Web Server accepts a request, it takes care of the general domain logic and handles the HTTPS connections. For those requests that are meant to reach the application layer, the Web Server will invoke the Application Server to pass the necessary arguments to the application itself. At that point, the application code does not care about anything except being able to process the incoming request.

In FLOODGUARD we use and configure Tomcat and Nginx (web servers) and Gunicorn (application server).

4.10 GRAPHICAL USER INTERFACE





The Graphical User Interface (GUI) acts as the presentation layer of the platform that receives the stored information from the database through the GeoServer and renders the interactive maps in the web browser. For the development of the GUI we are using JavaScript Libraries for interactive mapping. These libraries allow us to implement basic web mapping functionalities like pan, zoom, home, info window, measure, legend button, geo-location, search bar, layer selection, mini map, attribution, mouse position etc.


Apart from the interactive maps and geospatial data FLOODGUARD platform needs to visualize data in the form of interactive graphs (e.g., timeseries of forecasted hydrological or hydraulic parameters). A dedicated JavaScript library will be selected for the implementation of the charting functionality.

4.10.1 State of the art analysis of web mapping libraries

Mapping visualizations are designed to provide users with insights into their data, their inherent structure and relations, thus augmenting human situational awareness and decision making. Since there is not a single visualization recipe for all possible datasets, we will use web mapping libraries that provide great flexibility through a diverse collection of visualization templates. There are more than 50 JavaScript web GIS libraries however the most popular of them could be summarized in the following table (Table 4-6).

Table 4-6. State-of-the-art analysis for Web mapping libraries

Web mapping library	Characteristics
Leaflet http://leafletjs.com/ 	Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 39 KB of JS, it has all the mapping features most developers will ever need. Leaflet is designed with simplicity, performance, and usability in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code.
OpenLayers https://openlayers.org 	OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles, vector data and markers loaded from any source. OpenLayers has been developed to allow the use of geographic information of all kinds. It is completely free, open-source JavaScript, released under the 2-clause BSD License (also known as the FreeBSD).
MapBox https://www.mapbox.com 	A great cloud platform for creating and sharing beautiful interactive maps. It has some amazing map tiles, great features, a ton of examples, documented API and many other commodities. MapBox comes with a pricing.
Google Maps Platform https://cloud.google.com/maps-platform/  Google Maps Platform	Probably the most popular Mapping library, offering great documentation and a ton of examples. State-of-the-art technologies, advanced functionalities (e.g., street view) and many features for every use case. It comes with a pricing.
ModestMaps http://modestmaps.com 	A genuinely nice and fast JavaScript mapping library from the makers of MapBox. It might not be a complete solution to all mapping needs, but it is free and lightweight, with a simple API which can easily deliver wonderful interactive maps.

Web mapping library	Characteristics
PolyMaps http://polymaps.org 	A free JS library for making dynamic, interactive maps in modern web browsers. It is a small library created to be fast and support multi-zoom datasets over various tiled maps. It uses SVG which means that it will not work in IE7 and IE8.



4.10.2 Leaflet

Leaflet seems an easy choice for implementing the interactive mapping GUI of FLOODGUARD platform. It is an open-source software (free to use and distribute under the BSD-2 Clause License) with many community members to maintain and release new plugins and functionalities. It holds the leading position among open JavaScript libraries, being easy-to-use, efficient and lightweight.

4.10.3 State of the art analysis for interactive charts.

The most popular open-source libraries for interactive web graphs are Chart.js and D3. Both libraries enable the creation of responsive charts such as bar charts, line charts and scatter plots but their approaches differ significantly. Chart.js provides a selection of ready to go charts which can be styled and configured while D3 offers building blocks which are combined to create virtually any data visualization.

Table 4-7. State-of-the-art analysis for interactive charts libraries

Library	Characteristics
D3 https://d3js.org 	<p>Description: JavaScript library for visualizing data with HTML, SVG, and CSS</p> <p>License: BSD-3/Free for use AS IS</p> <p>Pros: More dynamically created graph, more options</p> <p>Cons: Larger Learning Curve, coding for legends, tooltips, interactivity, heavy</p>
Chart.js https://www.chartjs.org  Chart.js	<p>Description: Simple and engaging HTML5 based JavaScript library for charts</p> <p>License: open source, MIT license</p> <p>Pros: Ready to go charts, minimum configurations, built-in legend and tooltips, interactivity</p> <p>Cons: Limited set of charts to choose from, difficulty to construct non-standard graphs.</p>

4.10.4 Chart.js

While both D3 and Chart.js are excellent libraries for visualizing data, Chart.js requires far less effort to create common chart types. Given the decent visualization with minimum effort Chart.js seems a safe choice for the needs of FLOODGUARD platform. Indicatively Chart.js will be used to implement the graphs of forecasted

parameters from the meteorological (Figure 4-12), hydrological (Figure 4-13) and hydraulic models (Figure 4-14).

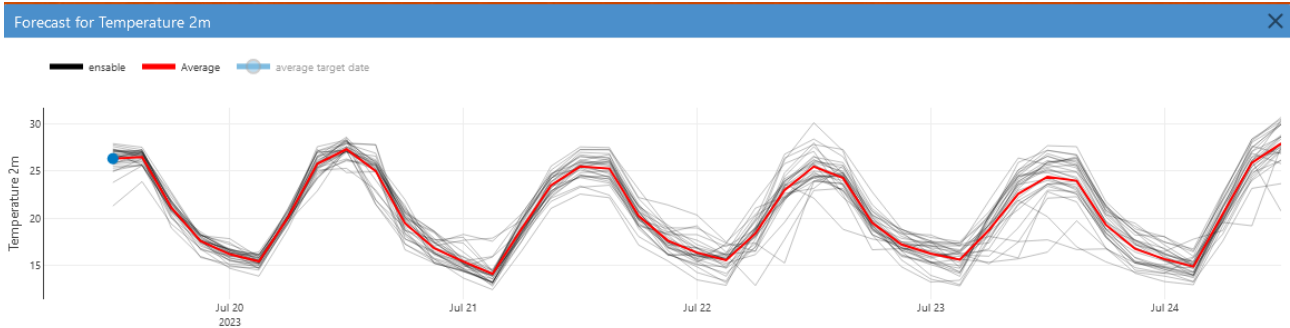


Figure 4-12. Timeseries graph of forecasted air temperature in the meteorological model

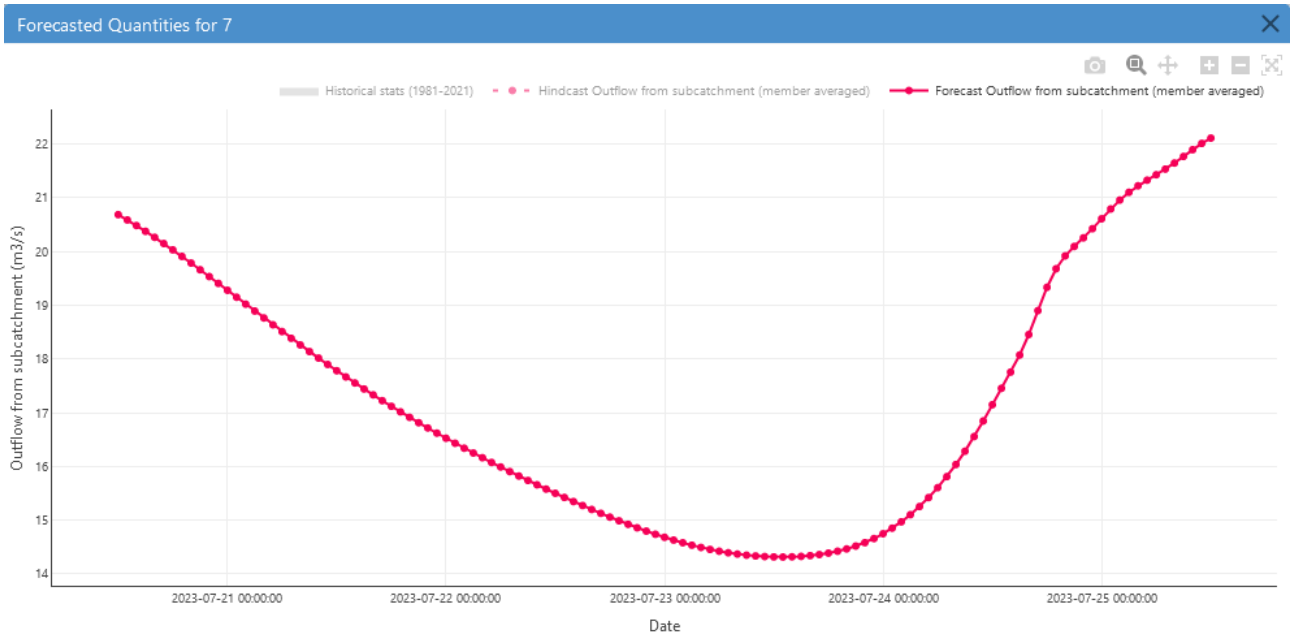


Figure 4-13. Timeseries graph of forecasted river discharge in the hydrological model

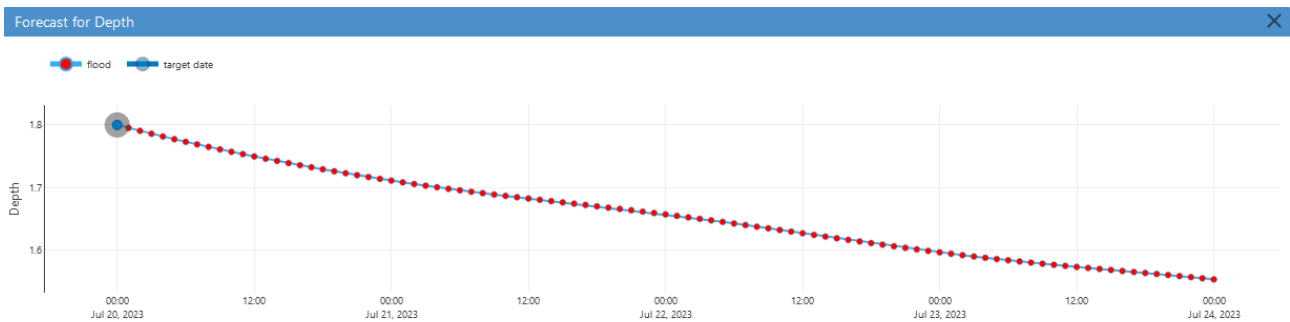


Figure 4-14. Timeseries graph of water depth in the hydraulic model

4.11 EARLY WARNING SYSTEM

FLOODGUARD platform implements a comprehensive Early Warning System (EWS) that is able to send alerts in system administrators whenever the forecasting service line detects signs of flooding, so we can protect people and assets, and minimize business disruptions. Upon discussions with the Contracting Authority, the EWS uses a two-fold approach for issuing alerts i.e. (i) alerts based on river flows and (ii) alerts based on water depths, as described in the following sections 4.11.1 and 4.11.2.

4.11.1 Early Warning system for hydrological alerts.

The hydrological Early Warning System will be able to send email to the administrators whenever specific thresholds are exceeded. The rules are defined through a text editor in JSON (JavaScript Object Notation) format. JSON is a lightweight and very easy way to store, read and understand data (Figure 4-15). This format gives great flexibility, since the user can define alarms for an unlimited number of points (hydrological catchments), using unlimited number of flow thresholds, and notify several recipients through email in case these are going to be exceeded, according to the results of the hydrological forecasting service. The service runs twice a day, immediately after the hydrological simulation, and sends an email to the predefined recipients with the following information:

- Name and position where flow exceedance is expected.
- Value of flow threshold to be exceeded.
- Forecasted value of flow.
- Type of threshold exceeded (e.g., critical, warning).
- Days when the exceedance is expected (e.g., Monday 15/10/2022)

The rules file for the Hydrological Early Warning System is located in the DB server (VM05) in C:\Users\localadmin\Documents\HydrologicalEWS.

```

1 - "DischargeAlarms": [
2 -   {
3 -     "PointName": "Ardas",
4 -     "HydrologicalCatchmentID": 989,
5 -     "AlarmLevels": [
6 -       {
7 -         "FlowThreshold": 100,
8 -         "AlarmDescription": "Info"
9 -       },
10 -      {
11 -        "FlowThreshold": 200,
12 -        "AlarmDescription": "Warning"
13 -      },
14 -      {
15 -        "FlowThreshold": 300,
16 -        "AlarmDescription": "Critical "
17 -      }
18 -     ],
19 -     "MailingList": [
20 -       "mail1@mail.com",
21 -       "mail2@mail.com"
22 -     ]
23 -   },
24 -   {
25 -     "PointName": "Evros (Bulgarian borders)",
26 -     "HydrologicalCatchmentID": 1033,
27 -     "AlarmLevels": [
28 -       {
29 -         "FlowThreshold": 400,
30 -         "AlarmDescription": "Alert level 1"
31 -       },
32 -       {
33 -         "FlowThreshold": 600,
34 -         "AlarmDescription": "Warning"
35 -       },
36 -       {
37 -         "FlowThreshold": 800,
38 -

```

```

DischargeAlarms: [Array]
  [0]: [Object]
    PointName: "Ardas"
    HydrologicalCatchmentID: 989
    AlarmLevels: [Array]
      [0]: [Object]
        FlowThreshold: 100
        AlarmDescription: "Info"
      [1]: [Object]
        FlowThreshold: 200
        AlarmDescription: "Warning"
      [2]: [Object]
        FlowThreshold: 300
    MailingList: [Array]
      [0]: "mail1@mail.com"
      [1]: "mail2@mail.com"
  [1]: [Object]
    PointName: "Evros (Bulgarian borders)"
    HydrologicalCatchmentID: 1033
    AlarmLevels: [Array]
    MailingList: [Array]
  [2]: [Object]
    PointName: "Erginis (Entrance from Turkey)"
    HydrologicalCatchmentID: 99
    AlarmLevels: [Array]
    MailingList: [Array]

```

Figure 4-15. Structure of a sample rules file used for the Hydrological Early Warning System with dummy values.

4.11.2 Early Warning system for hydraulic alerts.

Apart from the hydrological early warning system, FLOODGUARD platform implements a similar approach using the results of the hydraulic simulation. The rules are defined in a JSON text file, similar to the one used for the hydrological Early Warning system, with the major difference being that thresholds are based on water depths this time. The user is able to define an unlimited number of points to be checked using coordinates (lat, lon), and an unlimited number of depth thresholds. The service runs immediately after the hydraulic simulation and in case a depth threshold is likely to be exceeded according to the results of the hydraulic simulation, an email will be sent to the appropriate recipients containing the following information:

- Name and position (lat, lon) where water depth exceedance is expected.
- Value of water depth threshold to be exceeded.
- Forecasted value of water depth.
- Type of threshold exceeded (e.g., critical, warning).
- Days when the exceedance is expected (e.g., Monday 15/10/2022)

The rules file for the Hydrological Early Warning System is located in the DB server (VM05) in C:\Users\localadmin\Documents\HydraulicEWS.

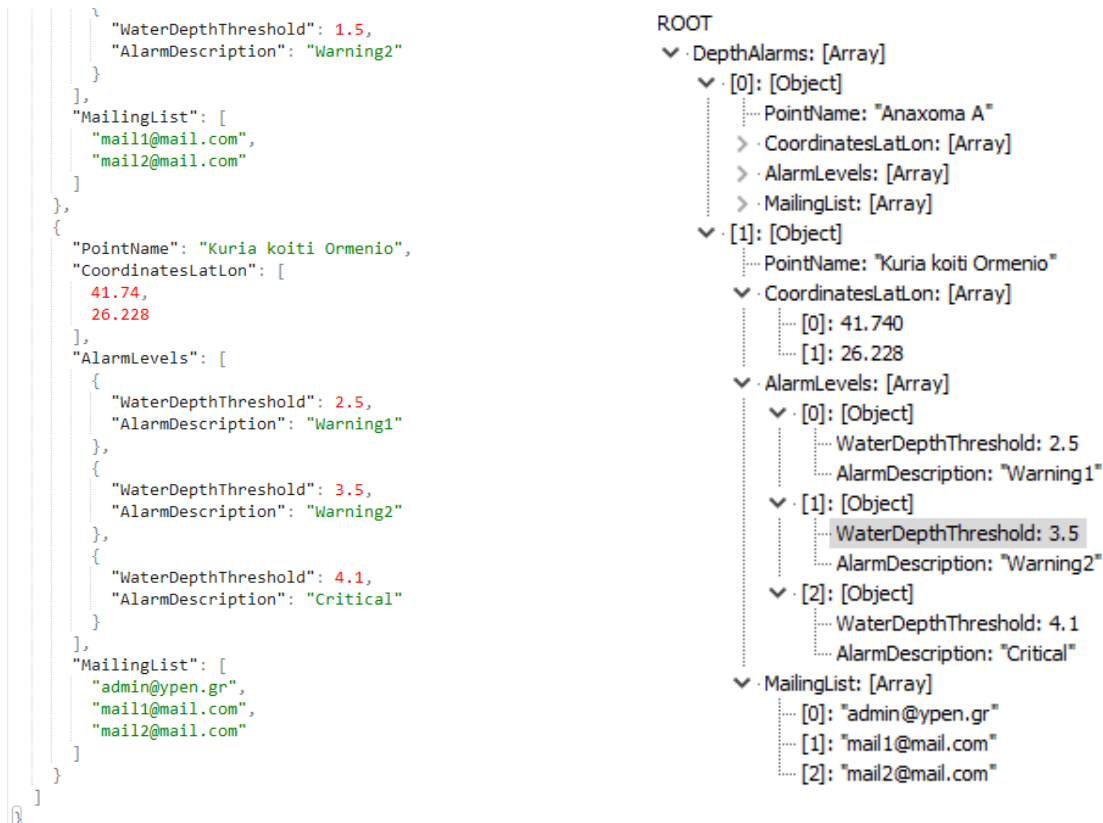


Figure 4-16. Structure of a sample rules file used for the Hydraulic Early Warning System with dummy values.

4.12 DATA CURATION

An offline data curation module will be developed in Python to ensure the integrity, the completeness, and the consistency of all datasets stored in the FLOODGUARD database. The core functionalities of this module are listed below:

- Identification of incomplete, inaccurate, incorrect, and irrelevant records from datasets and initiation of predefined procedures for replacement, modification, or removal of erroneous records.
- Handling of duplicate and missing entries to safeguard the appropriateness and reliability of the information provided in the dataset.
- Performing automated data cleaning process for detection and correction of corrupted dataset records, null values, unknown characters, extreme values etc.

The development of the data curation module will build upon the Pandas and NumPy Python libraries.

4.13 SECURITY

A robust security mechanism is considered essential for ensuring the integrity of the developed products and services as well as for providing to end users a secure virtual working environment through the web based

FLOODGUARD platform. FLOODGUARD security mechanisms are not a single standalone component, but instead a set of technologies and tools that are utilized within the individual platform components as well as a set of practices and principles that applied to each layer of the system's architecture. This design ensures that each component implements a different set of security actions which can be extended or replaced to achieve the desired level of security of the complete system. Furthermore, by implementing the security actions in a modular and "pluggable" approach, allows to further develop and customize the system after the end of the project's life, addressing the subsequent needs of custom, stricter and varying requirements.

It should be noted that many of the security requirements identified in paragraph 3.6 are already implemented by the software components that have been selected for the development. For example, modern Web Frameworks like Django come with great built-in security features that allow developers to use out-of-the-box predetermined security settings without coding everything from scratch.

The security mechanisms of FLOODWATER target the following project-specific layers of security:

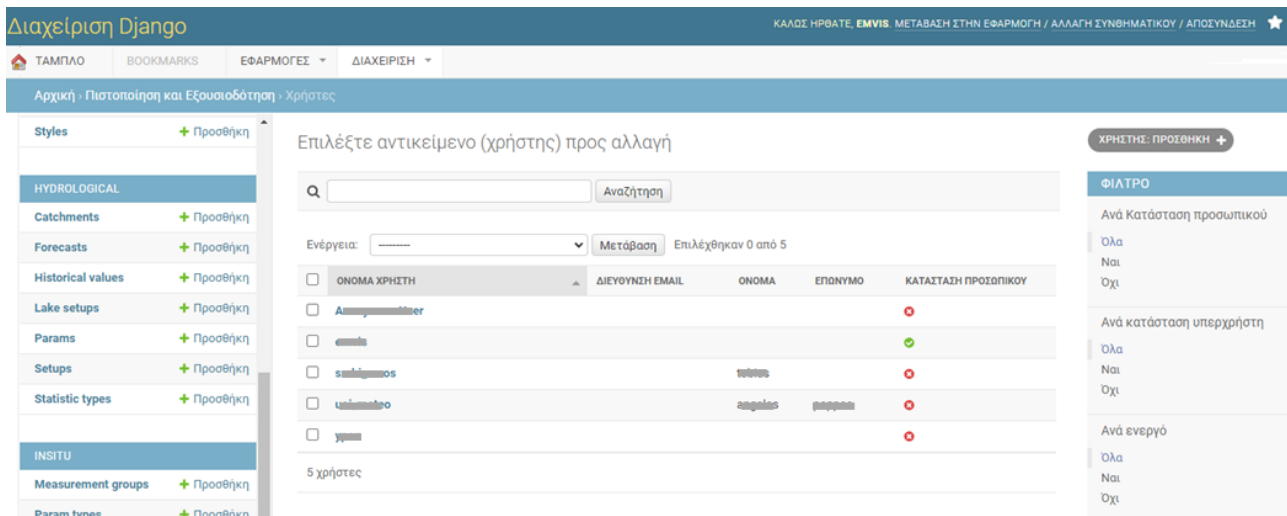
- Access control module and user authentication
- Security of stored information
- Firewall
- User authentication
- Security in data exchange through APIs

4.13.1 Access control module and user authentication

FLOODGUARD platform provides variable levels of access into its product and services. An access control module has been used to implement the desired set of access policy rules in every resource of the system. This allows the creation of users with different access rights to the platform components. For example, a user can have access only to the hydrological model results or only to a specific model setup of the meteorological model. Whenever a user request access through the platform to a selected dataset or functionality the access control module will enquire its properties to decide whether to grant or deny the access.

The user authentication process in the platform starts with a web-based login form to obtain information on credentials (username/password) and to exchange those for a temporary session token. Once the credentials have been submitted, the access control module produces a token which formulates the basis of all subsequent communications. The system is then able to check that the user is authenticated and authorized to perform the requested action in the system (e.g., view certain information). In cases where the security credentials are wrongly supplied, an exception will be produced, and each service will handle it accordingly.

The user authentication and access control module are built in Django Web Framework which provides a complete and widely used suite for user authentication, password management and encryption, pluggable sign-up forms etc. Users are managed through Django's web panel (Figure 4-17).



Διαχείριση Django

ΚΑΛΩΣ ΗΡΘΑΤΕ, ΕΜΒΙΣ. ΜΕΤΑΒΑΣΗ ΣΤΗΝ ΕΦΑΡΜΟΓΗ / ΑΛΛΑΓΗ ΣΥΝΘΗΜΑΤΙΚΟΥ / ΑΠΟΣΥΝΔΕΣΗ

ΤΑΜΠΛΟ BOOKMARKS ΕΦΑΡΜΟΓΕΣ ΔΙΑΧΕΙΡΙΣΗ

Αρχική · Πιστοποίηση και Εξουσιοδότηση · Χρήστες

Επιλέξτε αντικείμενο (χρήστης) προς αλλαγή

Αναζήτηση

Ενέργεια: ----- Μετάβαση Επιλέχθηκαν 0 από 5

<input type="checkbox"/>	ΟΝΟΜΑ ΧΡΗΣΤΗ	ΔΙΕΥΘΥΝΣΗ EMAIL	ΟΝΟΜΑ	ΕΠΙΘΥΜΟ	ΚΑΤΑΣΤΑΣΗ ΠΡΟΣΩΠΙΚΟΥ
<input type="checkbox"/>	Αναγνωστήρας				✖
<input type="checkbox"/>	admin				✔
<input type="checkbox"/>	superadmin		super		✖
<input type="checkbox"/>	user	user@user.com	user	password	✖
<input type="checkbox"/>	user				✖

5 χρήστες

ΧΡΗΣΤΗΣ: ΠΡΟΣΘΗΚΗ +

ΦΙΛΤΡΟ

Ανά κατάσταση προσωπικού

Όλα
Ναι
Όχι

Ανά κατάσταση υπερχρήστη

Όλα
Ναι
Όχι

Ανά ενεργό

Όλα
Ναι
Όχι

Figure 4-17. User management through Django's web panel.

4.13.2 Security of stored information

FLOODGUARD platform will ensure the security and integrity of all datasets or metadata that are preserved in the storage layer of the system. The storage layer of FLOODGUARD platform consists of two different solutions, namely a relational database and a separate file system.

Database security will be handled at different levels:

- the data-level, i.e., protect the data itself from theft or tampering.
- the system level, i.e., protect the hardware and the server from any malevolent activity through inbound and outbound traffic.
- and the user level, i.e., prevent users from performing unintended operations that might compromise data.

An automated backup mechanism ensures that database snapshots and backups are taken at regular intervals and are stored in a different physical machine. A mechanism for reverting the database to previous states has been considered through the Microsoft Azure platform.

For data stored in the **file system** a similar backup mechanism will be implemented and scheduled at user defined intervals to ensure data integrity. Both full and incremental backups will be planned. High security file systems like NTFS and ext3 which allow to set specific permission to local files and folders, will be preferred over older file systems like FAT32 which allow only shared permissions. Data access on the file system will be controlled through an access-control list (ACL). Permissions for file access will be preferably set at a user group level instead of individual users.

4.13.3 Firewalls

Firewalls will be used to isolate the different VM servers from the internet by blocking all incoming traffic except for specific web or application servers that are considered part of the FLOODGUARD system. Firewall rules are managed through GCloud (Microsoft Azure platform) which is the provider of the virtual machines of the project (Figure 4-18).

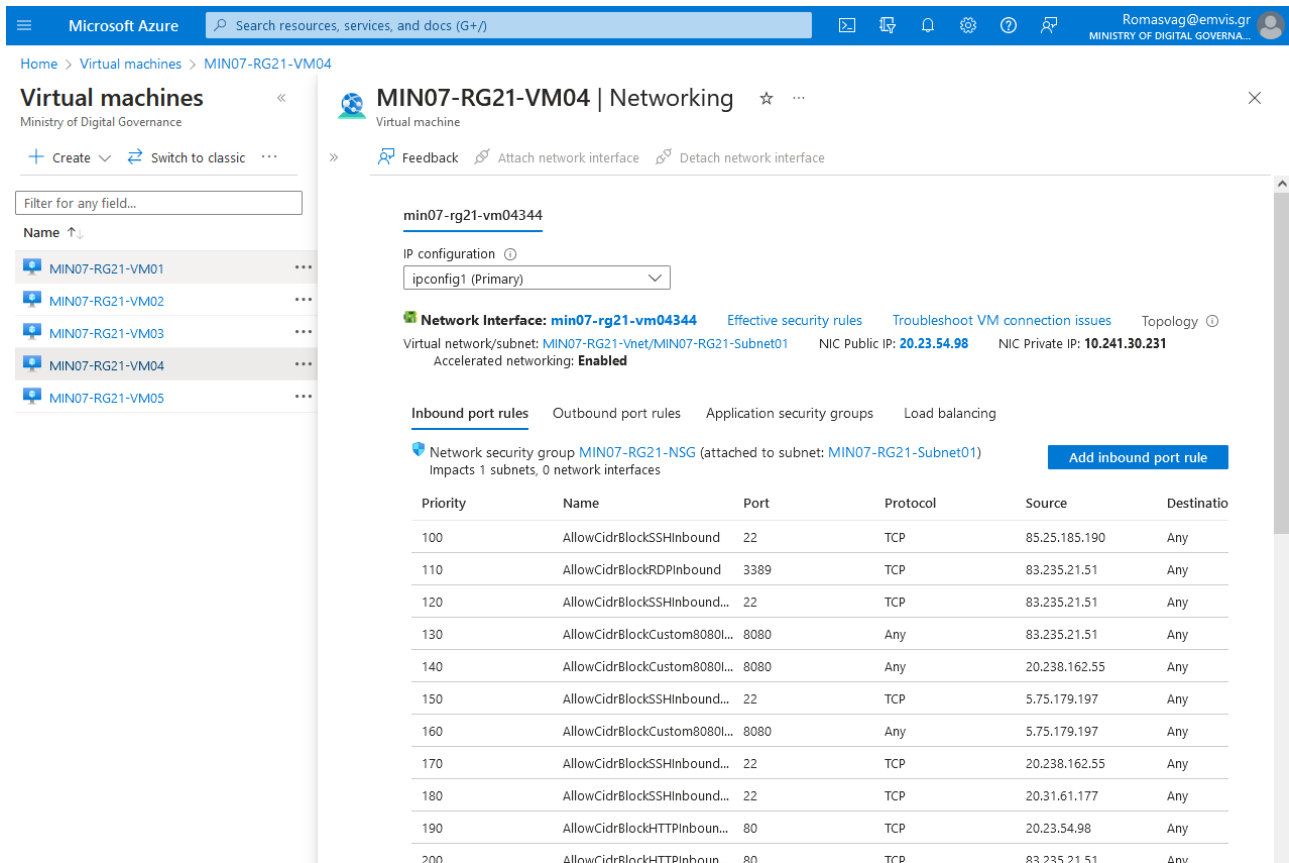







Figure 4-18. Management of firewall port rules in Microsoft Azure GCloud platform.

The user can set individually for each Virtual Machine the inbound and outbound rules. In general, all outbound traffic is allowed (e.g., a FLOODGUARD server is able to reach any external server), and all inbound traffic is blocked (access to a FLOODGUARD server by an external machine is prohibited). Specific exceptions have been set to the inbound rules to allow the FLOODGUARD servers to communicate with each other, and also to allow the developing team to gain access to the FLOODGUARD servers from their physical working environment (Table 4-8).

Table 4-8. Firewall exclusions for inbound traffic in FLOODGUARD servers.

Server	Allowed inbound rules from
 MIN07-RG21-VM01 Meteorological App	IPs of developing team in port 22 (SSH)

Server	Allowed inbound rules from
 MIN07-RG21-VM02 Hydrological App	IPs of developing team in port 22 (SSH)
 MIN07-RG21-VM03 Hydraylic App	IPs of developing team in port 3389 (RDP)
 MIN07-RG21-VM04 Web server	IPs of developing team in port 22 (SSH) Any IP in ports 80 (HTTP) and 443 (HTTPS)
 MIN07-RG21-VM05 Database server	IPs of developing team in ports 22 (SSH) and 8443 (Geoserver) IPs of VM01, VM02, VM03 in port 22 (SSH) IP of VM04 in ports 8443 (Geoserver) and 22 (SSH)

4.13.4 User authentication

All users will be authenticated prior to accessing any database resources, while queries and user actions will be performed only through programmatic methods such as stored procedures. The rule of “least privilege” will be adopted, which implies that users should be granted the minimum number of privileges required to perform their tasks. The number of root accounts and super-users should be kept as low as possible. Strict passwords policies will be applied, and user passwords will be stored in the database only after salting and hashing.

4.13.5 Security in data exchanged through APIs

Data transfer between different components and sub-systems of the FLOODGUARD platform will be performed through APIs. All API endpoints will communicate always over HTTPS protocol. HTTP and other non-secure protocols will be disabled. APIs should be stateless which means that authentication and authorization will depend on credentials validated on server-side for every request and not on cookies or sessions. OAuth 2.0 authentication form will be preferred over basic authentication, to allow establishing trusted identities by using tokens. Tokens will expire at predetermined intervals (e.g., 24 hours) so even in case a token gets leaked the impact of the breach is minimized.

Exposure of sensitive information through query strings in URL can be exploited by attackers. Usernames passwords or tokens should never appear in URLs. User passwords will be always hashed using salting techniques to protect the system (or minimize impact) in case it is compromised for any number of reasons. Another good practice to secure APIs is to apply rate limits and throttling to avoid resources starvation or Denial-of-Service attacks. Setting a maximum limit on how often a method can be called will discourage bot attacks or prevent a bad programmed routine (e.g., calling an API in an endless loop) from an authorized user, to stress the system.

Finally, IP filtering, i.e., enabling API calls only from specific IP addresses, adds another layer of security. This can be applicable for specific FLOODGUARD APIs that are designed for the communication of existing workflows operated by consortium members.

5 TECHNOLOGY STACK

This document has presented the interpretation of the project requirements into a detailed listing of technical requirements and functionalities that the FLOODGUARD platform should provide to realize the objectives of the project. This in turn has led to the definition of the high-level architecture and the selection of the different IT components which were combined to deliver the operational FLOODGUARD platform. The proposed IT architecture is composed of mostly open-source and well proven technologies that will allow us to create a flexible and scalable solution. Figure 5-1 presents the selected technology stack that has been used for the implementation of FLOODGUARD platform.

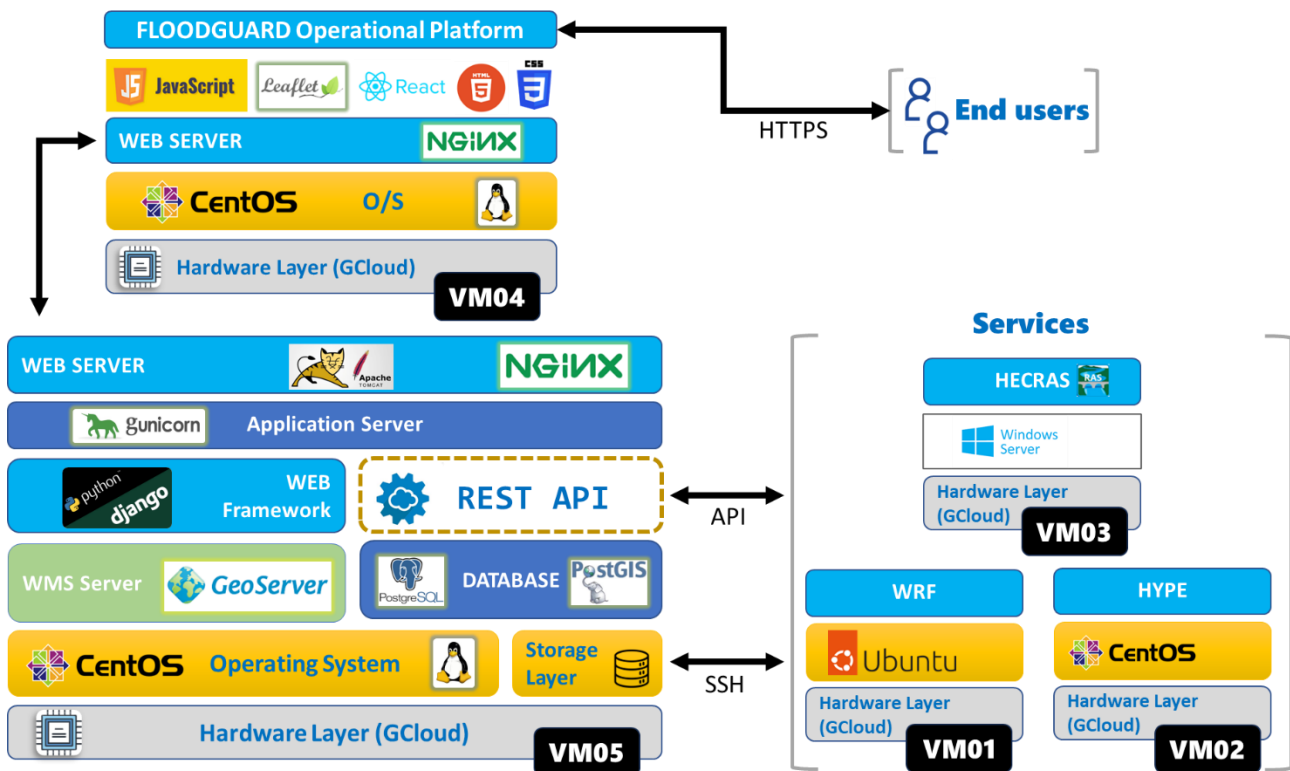


Figure 5-1. Technology stack of the 5 VM servers of the FLOODGUARD platform.

The operational FLOODGUARD platform is available at <https://20.23.54.98/>.