D8.8 – GIS-Based Early Warning and Digital Twin Platform integration report

DATE OF DELIVERY - 22/12/2023

AUTHORS – Giovanni Serafino (MBI), Ivan Boscaino (MBI), Andrea Rucci (MBI), José Gómez-Barrón (UCD)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003534

DOCUMENT TRACKS DETAILS

Project acronym	SCORE
Project title	Smart Control of the Climate Resilience in European Coastal Cities
Starting date	01.07.2021
Duration	48 months
Call identifier	H2020-LC-CLA-2020-2
Grant Agreement No	101003534

Deliverable Information	
Deliverable number	D8.8
Work package number	WP8
Deliverable title	GIS-Based Early Warning and Digital Twin Platform integration report
Lead beneficiary	MBI srl
Author(s)	Giovanni Serafino (MBI), Ivan Boscaino (MBI), José Gómez-Barrón (UCD), Andrea Rucci (MBI)
Due date	31/12/2023
Actual submission date	22/12/2023
Type of deliverable	Report
Dissemination level	Public

VERSION MANAGEMENT

Revision table			
Version	Name	Date	Description
V 0.1	Giovanni Serafino (MBI), Ivan Boscaino (MBI), Andrea Rucci (MBI), José Gómez-Barrón (UCD)	12/12/2023	First draft
V 0.2	Giovanni Serafino (MBI), José Gómez-Barrón (UCD), Ivan Boscaino (MBI)	11/12/2023	Updated draft internally reviewed (Cécil J.W. Meulenberg, ZRS, Göksu Soydan Oksal, SAMU)
V 0.3	Giovanni Serafino (MBI)	21/12/2023	Updated draft after contribution from partners
V1.0	Giovanni Serafino (MBI) Iulia Anto (ATU), Salem Gharbia (ATU)	22/12/2023	Final version

LIST OF ACRONYMS AND ABBREVIATIONS

Acronym / Abbreviation	Meaning / Full text
2D	Two-dimensional
3D	Three-dimensional
AI	Artificial Intelligence
API	Application Programming Interface
CCLL	Coastal City Living Lab
DB	Database
DEM	Digital Elevation Model
DMP	Data Management Plan
DSM	Digital Surface Model
DT	Digital Twin
DTM	Digital Terrain Model
EBA	Ecosystem-Based Approach
EWS	Early-Warning Support
GeoJSON	Geographic Javascript Object Notation
GIS	Geographic Information System
GUI	Graphical User Interface
IoT	Internet of Things
IPCC	Intergovernmental Panel on Climate Change
JSON	Javascript Object Notation
ML	Machine Learning
RCP	Representative Concentration Pathway
REST	Representational State Transfer
RPO	Regional Planning Organization
SIP	SCORE Information and communication technologies (ICT) platform
SME	Small-Medium Enterprise
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
USE	User Scenario Evaluation
WFS	Web Feature Service
WP	Work Package
XML	Extensible Markup Language

BACKGROUND: ABOUT THE SCORE PROJECT

SCORE is a four-year EU-funded project aiming to increase climate resilience in European coastal cities.

The intensification of extreme weather events, coastal erosion and sea-level rise are major challenges to be urgently addressed by European coastal cities. The science behind these disruptive phenomena is complex, and advancing climate resilience requires progress in data acquisition, forecasting, and understanding of the potential risks and impacts for real-scenario interventions. The Ecosystem-Based Approach (EBA) supported by smart technologies has potential to increase climate resilience of European coastal cities; however, it is not yet adequately understood and coordinated at European level.

SCORE outlines a co-creation strategy, developed via a network of 10 coastal city 'living labs' (CCLLs), to rapidly, equitably and sustainably enhance coastal city climate resilience through EBAs and sophisticated digital technologies.

The 10 coastal city living labs involved in the project are: Sligo and Dublin, Ireland; Barcelona/Vilanova i la Geltrú, Benidorm and Basque Country, Spain; Oeiras, Portugal; Massa, Italy; Piran, Slovenia; Gdansk, Poland; Samsun, Turkey.

SCORE will establish an integrated coastal zone management framework for strengthening EBA and smart coastal city policies, creating European leadership in coastal city climate change adaptation in line with The Paris Agreement. It will provide innovative platforms to empower stakeholders' deployment of EBAs to increase climate resilience, business opportunities and financial sustainability of coastal cities.

The SCORE interdisciplinary team consists of 28 world-leading organisations from academia, local authorities, Regional Planning Organizations (RPOs), and small-medium enterprises (SMEs) encompassing a wide range of skills including environmental science and policy, climate modelling, citizen and social science, data management, coastal management and engineering, security and technological aspects of smart sensing research.

EXECUTIVE SUMMARY

This document is a deliverable of the SCORE project, funded under the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003534.

The core of the activities carried out in the work package (WP) 8 consist in the design, development, and deployment of a digital twin-early warning support (DT-EWS) system. The process related to its implementation followed two concurrent paths that have been articulated in tasks T8.2 "Development of the Early Warning Support and Digital Twin Platform" and T8.3 "Development of tools to access and visualise SCORE data and outcomes". The outputs of the activities of these tasks are reported in the already released deliverables D8.4 [1], D8.5 [2], D8.6 [3], and D8.7 [4].

This deliverable reports on the approach adopted in Task T8.4 "Integration and Deployment of GIS based Early Warning and Digital Twin Platform" and on the related activities that have been carried out focusing on the integration of the back-end and the front-end parts of the DT-EWS system, that have been developed in the above-mentioned tasks T8.2 and T8.3. The front-end of the system is a graphic user interface (GUI) allowing users to interact with the simulation engine, simplifying their experience. As a matter of fact, the integration of the two system sections is crucial for the system deployment. However, it is important noting that the DT platform is made by two main subsystems, namely the User Scenario Evaluation (USE) and the Early-Warning Support (EWS) module, with which the user interacts in different ways. For this reason, the GUI is implemented to cover their two different scopes.

To enable a smooth communication and data exchange between the two system sections, specific representational state transfer (REST) application programming interfaces (APIs) have been developed on purpose. Indeed, the users will be allowed, for instance, to set parameters related to the simulations execution on the pages of the GUI, to virtually implement EBAs, and to visualize the simulations results. Each action on the GUI is associated to the call of an API method for a back-end function corresponding to that particular action. In this document, also the complete list of the employed APIs is reported, with a description.

LINKS WITH OTHER PROJECT ACTIVITIES

The activities and the related results reported in this document can be seen as a follow-up of the tasks T8.2 and T8.3, related to the design and development of the DT-EWS system back-end and front-end, respectively. Therefore, the inputs of this deliverable directly come from the content of D8.4 [1], D8.5 [2], D8.6 [3], and D8.7 [4] and, indirectly, from D8.1 [5], D8.2 [6], and D8.3 [7]. Indeed, the latter contain the system requirements and design guidelines, as they have been formulated thanks to the co-design activities carried out with the CCLLs and the relevant stakeholders.

The outputs reported in this document represent the SCORE WP8 system development completion, and are crucial for the subsequent activities of system deployment, validation, and assessment, whose plan is reported in detail in D8.10 [4] (project task T8.5). These activities will involve a larger number of interactions with other project WPs, in particular with WP2, since they will imply a strong collaboration with CCLLs core teams and stakeholders, as well as with WP9, since the DT deployment will represent a good opportunity for the dissemination of the SCORE project activities and results.

TABLE OF CONTENT

1. Introduction 10

1.1 Scope of the Document ________ 11

2. System architecture 12

2.1 Components of the Back-end ________ 12

2.1.1 The User Scenario Evaluation Subsystem 12

2.1.2 The Early-Warning Support Subsystem 13

2.1.3 The Database 14

2.1.4 Communication Channels 14

2.2 Components of the Front-end ________ 14

3. System back-end and front-end Integration 16

3.1 Approach in the Integration Process ________ 16

3.2 Middleware and Communication Protocol ________ 17

3.3 Data Flow between Back-end and Front-end ________ 18

3.4 Integration of User Interface Components ________ 18

3.5 Interaction between Back-end and Front-End ________ 19

3.5.1 API Requests and Endpoints 19

3.5.1.1 Authentication 20

3.5.1.2 Users 20

3.5.1.3 Scenarios 20

3.5.1.4 EBA Type 20

3.5.1.5 EBA Management 20

3.5.1.6 Sea State 21

3.5.1.7 Rain Rate 21

3.5.1.8 River Discharge 21

3.5.1.9 USE Simulation 21

3.5.1.10 EWS Simulation 21

3.5.1.11 USE Results Publication....21
3.5.1.12 EWS Results Publication....21
3.5.1.13 Sensors Management....22
3.5.1.14 Long-Term Scenarios....22
3.5.2 Visualization of the Simulation Results....23
3.5.2.1 USE Section....23
3.5.2.2 EWS Section....23
3.5.2.3 The Tile Map Server....23
4. Conclusions....24
5. References....25

INDEX OF FIGURES

Figure 1. Overall architecture of the DT-EWS and the connections among its subsystems and other external systems. GUI: Graphical User Interface; DB: Database; REST-API: Representational State Transfer – Application Programming Interface; USE: User Scenario Evaluation; EWS: Early-Warning Support; SIP: SCORE ICT Platform. ..12
Figure 2. Example of a schema of the data returned by the GET method of the Users API......................................20
Figure 3. Tree structure of the selection procedure for the long-term scenarios. This structure is the same for river discharge and sea state events. Each tree leaf is associated to a specific event data set. 22

1. INTRODUCTION

In modern software development, the seamless collaboration between the back-end and the GUI is pivotal for delivering a robust and user-friendly application. This integration involves a series of coordinated activities aimed at ensuring efficient communication, data flow, and synchronization between the server-side logic and the user-facing components.

The integration process begins considering the system architecture. The back-end, responsible for data processing, storage, and business logic, must seamlessly interact with the GUI, which facilitates user interaction and presentation of information. This collaboration typically occurs through well-defined APIs and communication protocols. Integration activities prioritize the establishment of an efficient data flow between the back-end and the GUI. Middleware solutions, such as message queues or RESTful APIs, play a crucial role in facilitating this communication. They ensure that data are transmitted securely and in formats understandable by both ends, maintaining consistency and reliability.

The back-end integration encompasses multiple facets, including data storage and retrieval, business logic implementation, and user authentication. Database integration ensures that data is stored and retrieved seamlessly, while business logic integration involves translating complex processes into functionalities accessible by the GUI. On the front-end, integration involves the incorporation of user interface components, such as widgets and GUI frameworks, and the establishment of communication channels with the back-end through API calls and endpoints. This integration ensures that user interactions trigger appropriate actions on the back-end, allowing for real-time updates and responsive user experiences.

An API, or Application Programming Interface, is a set of rules and tools that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information. APIs are crucial for enabling the integration of different software systems and services, allowing them to work together seamlessly. APIs can be used for various purposes, including retrieving data from a server, sending data to a server, or performing specific actions within an application. For example, in the context of web development, APIs are often used to enable communication between a web application's front-end (i.e., what users interact with) and its back-end (i.e., where data are processed and stored).

An endpoint is a specific Uniform Resource Locator (URL) or Uniform Resource Identifier (URI) that an API exposes to enable communication. In simpler terms, an endpoint is a point of interaction with the API. Each endpoint is associated with a specific function or resource in the API and is accessed through a specific hypertext mark-up transfer protocol (HTTP) method (such as GET, POST, PUT, DELETE). As an example, a weather API might have different endpoints for retrieving the current weather, the forecast, or historical weather data. Each of these endpoints would be accessed using a specific URL, and the type of data returned would depend on the HTTP method used and the parameters provided.

For the integration of the DT-EWS front-end and back-end, as it will be better explained in the following, REST APIs have been implemented. This choice is motivated by the fact that they offer system flexibility and scalability. Moreover, since they are generally and widely adopted for software system development, they also ensure interoperability with other external systems.

1.1 Scope of the Document

This report is part of the technical deliverables of WP8 "Development of integrated Early Warning Support System and spatial Digital Twin solution prototypes" of the SCORE project. In particular, this deliverable is aimed at describing the integration between the back-end and front-end sections of the DT-EWS system, operated thanks to on-purpose developed APIs, which are employed to share and exchange information and commands between internal interfaces and external systems.

This document is structured as follows: after the Introduction giving a brief overview on back-end and front-end integration and on RESTful APIs as the key element for integration, in Sect. 2, a description of the system architecture and functionalities is outlined, since it is relevant to exposing the integration among the system sections. Then, in Sect. 3, the integration approach and methods are illustrated, giving details about the data flows between GUI and back-end. Moreover, the APIs involved in the connection of the system front-end and back-end are briefly described, and the link to the complete and updated documentation is reported. Finally, in the Conclusions, a summary of the document is sketched, and the next activities are outlined.

2. SYSTEM ARCHITECTURE

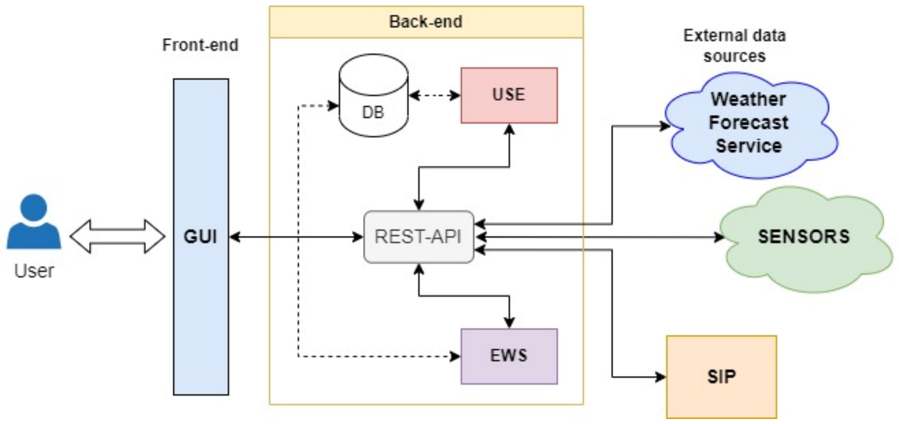


Figure 1. Overall architecture of the DT-EWS and the connections among its subsystems and other external systems. GUI: Graphical User Interface; DB: Database; REST-API: Representational State Transfer – Application Programming Interface; USE: User Scenario Evaluation; EWS: Early-Warning Support; SIP: SCORE ICT Platform.

2.1 Components of the Back-end

To better expose in the following the integration among the system components, here we briefly report the architecture of the implemented DT-EWS, considering all the main subsystems of the back-end and the front-end. Figure 1 depicts the high-level system architecture, which has been already thoroughly described in [1] and [2].

2.1.1 The User Scenario Evaluation Subsystem

The back-end of the DT-EWS system is composed of two main blocks, the USE subsystem and the EWS subsystem. The former is employed for running simulations enabling what-if and long-term analyses on scenarios that can be created by users as a mixture of real data related to the coastal city, and other hypothetical data. These data can be related, e.g., to weather or river discharge events, sea state conditions, or to EBAs solutions to be implemented on the considered study area. In this way, the user is allowed to simulate and contextually evaluate the impact of an EBA in case of extreme flooding events. The user-defined scenarios are composed by users assembling a heterogeneous set of data from different sources:

- *The coastal cities baseline data*: these data are not selectable by the users, since they represent the minimum required, ground-truth data for running the simulation, and are associated to the user's coastal city. They are represented by the city digital elevation model (DEM), digital terrain model (DTM), or digital surface model (DSM), a map with the land use and cover of the study area, the geometry and hydraulic properties of the rivers flowing through the CCLL area. In addition, baseline data include the baseline financial exposure and hazard maps calculated on the study area, which are reported in [8].
- *The climate and weather data*: the users can decide what kind of weather events to simulate, e.g., by inserting values for parameters like rain rate, river flow, and sea state. Alternatively, they can employ pre-set events related to long-term models, elaborated agreeing to the Representative Concentration Pathways (RCPs) models adopted by the Intergovernmental Panel on Climate Change (IPCC). These models are related to the projections on greenhouse gas concentration in the coming decades [9].
- *EBAs*: the user can choose from an EBA palette prepared for the SCORE project [10], including 33 possible EBAs. Once the type of EBA solution is selected, the user can trace polygons on the study area

map, selecting the zones for realizing EBAs, to simulate its impact in the what-if analysis associated to the particular scenario and the set parameters.

It is important to underline which data are employed by the system since they represent one of the pivotal elements concerning the interactions with the users.

The USE subsystem output is composed of several maps corresponding to different times within the simulation set time frame. The maps report the eventual flooding and related water level in the study area, produced by the conditions set by the user for the simulation. In addition, the recalculated financial hazard maps are also produced showing the estimated amounts of damages that the simulated events can virtually cause. As reported more in detail in [2], when a simulation finishes, the system generates a human-readable report that can help the user understanding whether the output is consistent. This way, a user can decide if the newly produced maps are worth to be stored and further analysed, or if they can be discarded.

2.1.2 The Early-Warning Support Subsystem

The EWS subsystem runs continuously without needing inputs from the users, since it has to monitor the current weather and hydraulic situation. This system block can make projections of the city flooding risk in the short or medium term. This is possible thanks to processing of real-time data from a weather forecasts service and from sensors distributed over the study area, making use of artificial intelligence (AI) and machine learning (ML)-based algorithms. In this case, the users can only check the status of the simulation, look at the system outputs, and make few other operations, as it is explained in the following. To ensure its correct operation, the system automatically takes the following data:

- *The coastal city baseline data*, in the same way as the USE, including financial exposure and hazard maps.
- *Data from sensors*: the system needs data streams from stations distributed on the study area, sensing relevant parameters like rain rate, rivers levels, sewage network discharge, and so on. They are crucial for monitoring the actual hydraulic situation of the city.
- *Weather forecasts*: beyond data from measurements, to be able to make projections on the flooding risk, the system must know the expected weather evolution, that can have an impact on the current hydraulic situation.

As in the case of the USE, the outputs of this subsystem consist of:

- Flooding maps and the associated human and financial damages; it is important noticing that these maps represent projections made by the system, based on the real-time information it gets from weather forecasts and sensors.
- Alerts of imminent flooding in case the water levels calculated by the system projections can put at risk inhabitants or cause major economic damages. These alerts are showed on the screen, if the user has opened a GUI, and can also be sent to designated users by mail or through other message services.
- Warnings, that are not related to simulation outcomes and expected risks, but to the data sources. Indeed, the EWS subsystem also regularly performs a check on the consistency of the received data, as described in [2]. It warns the user in case, e.g., there is a data feed that does not fit with other data streams, or in case of any mismatch with what is expected. In such a case, the EWS simply raises a warning to let the users be aware of the data inconsistency so that, if needed, they can exclude that particular sensor from the system inputs and, eventually, proceed to an *in-situ* check of the sensor conditions.

2.1.3 The Database

The system internal database (DB) is built using the PostgreSQL object-relational database system. The DB contains all the mentioned baseline data related to the user's coastal city and all the data related to the scenarios created by the user, such as weather, sea state, river discharge events, and EBAs. The DB also stores the simulation output maps, for the USE as well as for the EWS subsystem. More details on this system block can be found in [2].

2.1.4 Communication Channels

As depicted in Figure 1, the system employs RESTful APIs to ensure connection and interactions between its front-end and back-end, but also to exchange data with external systems. The APIs are a fundamental block in the system, allowing the call of functions from the GUI and the exchange of data among all the system blocks.

2.2 Components of the Front-end

The front-end of the DT-EWS is a web user interface that consists of interconnected components that serve specific roles within the user's journey, centralizing the functionality necessary for stakeholders to interact effectively with the system. The components of the front-end are:

- *The Web Application:* This foundational component delivers static content and orchestrates the interactions between the Web GIS, the USE Interface and the EWS subsystem web components, to present its messages and outputs. It is responsible for presenting views and functionalities that allow users to navigate and utilize the integrated components and features seamlessly. Also, it integrates the **Authentication and Session Management** security feature, with a robust user authentication system that manages credentials, roles, and session tokens interacting with the APIs. This ensures secure access to the system and protects sensitive data from unauthorized use.
- *Web GIS:* Serving as the interactive geospatial platform within the DT-EWS, the Web GIS component is equipped with the following sub-components:
  - *Geospatial Catalogue Explorer:* It integrates a multitude of data sources with DT-EWS outcomes, offering users an extensive platform for data exploration and export, ensuring a comprehensive understanding of the geographic context of the USE and EWS simulations outputs.
  - *Map Viewer:* A dual 2D/3D visualization tool that allows geospatial layer overlays and attribute data querying. Users can share their map views and print maps directly from the platform.
  - *Sensor Explorer:* Provides users with access to sensor observations, enabling visualization of data time series from multiple sensors in near real-time. As above mentioned, authorized users can disable or enable the official sensors that are used in the EWS simulations.
- *USE Interface:* The USE Interface is dedicated to the evaluation and management of user-defined scenarios and integrates with the Web GIS for a cohesive experience:
  - *EBAs Editor:* Allows users to digitally map and geolocate EBAs within their scenarios, supporting the strategic placement and assessment of environmental interventions.
  - *Parameters Editor:* Facilitates users in the selection or manual input and modification of the scenario parameters, including sea state, rainfall data, river flows, and other long-term scenarios with predefined

hydrographical and weather factors and conditions, to inform simulations run by the USE subsystem.

- *EWS Components:* These web sections request and transform the data from the EWS. Here, the different messages from the EWS subsystem are presented to inform users at different levels of information, warnings, or alerts, including when a sensor used in the EWS simulations is not working correctly. It also integrates with the Web GIS for a consistent experience where users can visualise the different geospatial outputs (i.e., flooding maps).
- *Subsystem API Integration:* The USE and EWS subsystems are connected to the front-end via their respective APIs. As better explained in the next section, these APIs enable the execution of scenarios and the retrieval of data and simulation results, thereby translating user interactions into actionable insights and visualizations within the DT-EWS system.

3. SYSTEM BACK-END AND FRONT-END INTEGRATION

The design and development of the DT-EWS system back- and front-end have been carried out from the start on parallel paths in two separated WP8 tasks. Nonetheless, the constant, common focus on users' operations and needs has ensured that all the system parts remained consistent and suitable for integration.

3.1 Approach in the Integration Process

The activities for the integration of the back-end and front-end sections of the SCORE DT-EWS system have constantly followed a user-oriented approach. Indeed if, on one hand, the system back-end modules have been designed to implement all the system functionalities to meet the defined functional requirements, on the other hand, the system front-end has been developed considering how users will operate these system functions. This, of course, also influenced to some extent the back-end section design.

The partners directly involved in the WP8 design and development activities are MBI srl and University College of Dublin. In particular, the former developed the back-end section of the system in Task T8.2, and the latter the front-end in Task T8.3. However, frequent interactions and co-working sessions took place among the involved developers, already in the phase of system design, with the aim of preparing the work to be done in task T8.4 and to avoid losing the coherence of the DT-EWS parts. Therefore, the integration activities have been conducted smoothly by creating an API and the corresponding end-points and methods for each system functionality exposed to the user. In general, the GUI has been developed considering all the possible users' operations, which are thoroughly described and depicted in [2] and [4], based on the required system functions. Then, the following steps have been followed to connect GUI and system back-end:

1. Associate a REST API and an end-point to each user operation (e.g., EBA implementation, rain or river discharge events setting, sea-state setting)
2. Associate HTTP methods to the user actions (e.g., setting parameters, retrieving data, editing or deleting already set parameters)
3. Design the API end-points structured dictionary of data defining the schema or format of the data that are exchanged
4. Client-server interaction: the GUI client uses those data dictionaries by handling requests to and responses from the API (e.g. by parsing and using the data received).

This way, every part of the front-end and the corresponding parts of the back-end have been tailored to allow the users to have an as much as possible simple and intuitive interaction with the USE and the EWS. The two subsystems interact with the users in different ways, due to their different operation modes.

However, since one of the pillars of the SCORE project is represented by a strong interaction between the CCLLs components and stakeholders and the DT-EWS system developers, this implementation and integration activities must not be considered complete and finalised at the end of Task T8.4. Indeed, other iterations with the final users will be required, exploiting their feedback to fine-tune the system.

3.2 Middleware and Communication Protocol

As depicted in Figure 1, the system employs REST APIs to ensure connection and interactions between its front-end and back-end. They represent a key component of the system that allows data flowing in and out of the system, also ensuring the communication between the various subsystems.

REST APIs have emerged as a fundamental architectural style for designing interconnected applications on the web. REST emphasizes a stateless client-server interaction model, scalability, and a uniform interface to enable a wide range of systems to communicate effectively. Understanding the principles and characteristics of REST APIs is essential for modern web development. The REST paradigm is widely employed in various domains, including web development, mobile application development, Internet of Things (IoT), and microservices architecture. Its simplicity and scalability make it a popular choice for exposing and consuming web services.

At the core of REST is a set of architectural principles that guide the design of APIs:

- *Statelessness*: Each request from a client to a server must contain all the information needed to understand and fulfil the request. The server should not store any client state between requests. This enhances scalability and simplifies the server's architecture.
- *Client-Server Architecture*: REST separates the client and server concerns, allowing them to evolve independently. The client is responsible for the user interface and user experience, while the server manages resources and business logic. This division enhances modifiability and scalability. In the SCORE DT-EWS system, the back-end section plays the role of the server, whereas the front-end section represents the client.
- *Uniform Interface*: REST promotes a uniform and consistent interface between components. This includes resource identification through URIs, a resource representation that can be manipulated by the client, and a set of standard methods (GET, POST, PUT, DELETE) for interacting with resources. A uniform interface simplifies the architecture and promotes interoperability.
- *Resource-Based*: REST views data and services as resources, each identified by a unique URI. Resources can represent entities such as objects, services, or concepts. Clients interact with these resources using standard HTTP methods, and resource representations are exchanged between client and server.

As above mentioned, RESTful APIs leverage standard HTTP methods for operations on resources. The most used methods include GET (retrieve a resource), POST (create a new resource), PUT (update an existing resource), and DELETE (remove a resource). Resources are represented in a format that can be easily consumed by clients, often in JavaScript Object Notation (JSON) or Extensible Markup Language (XML). This decoupling of resource representation from resource state allows clients to understand and manipulate resources without knowing the internal details of the server.

RESTful APIs exhibit many advantages: firstly, they are characterized by an inherent scalability, due to REST' stateless nature and separation of concerns between client and server. This makes RESTful APIs suitable for applications ranging from small-scale services to large-scale distributed systems. Secondly, this kind of APIs is characterized by enhanced interoperability: indeed, REST's reliance on standard HTTP methods and status codes fosters interoperability across different systems and platforms. This enables diverse clients, including web browsers, mobile applications, and other servers, to interact seamlessly with RESTful services. Finally, another important feature of the REST paradigm is related to its simplicity and ease of use, thanks to its clear principles and standard practices, that make it accessible to developers. Creating, consuming, and maintaining RESTful APIs is relatively straightforward compared to more complex architectures.

The REST API interaction processes with a client ensure that the system design is both comprehensive and practical for the end-user, facilitating the bridge between the system capabilities and the user experience.

3.3 Data Flow between Back-end and Front-end

The data flow between the back-end and the front-end is designed to be seamless, accurate and efficient, ensuring effective data exchange and management across the various DT-EWS components. The data flow aims to establish a bidirectional stream of data between the server (back-end) and the client (front-end), supporting dynamic and near real-time interactions essential for USE and EWS subsystems. It facilitates the dissemination of warnings and alerts, integration of sensor data streams, and enables user interactions within a GIS-based Digital Twin environment, which includes baseline geographic contextual information and the processing of subsystem outputs. The key aspects of the data flow are:

- *Data request and retrieval*: in this process, the front-end sends a request to the API to fetch data, and the back-end retrieves and returns the requested information for display and further user interaction.
- *Data submission and update*: this process is related to the submission of new data or inputs, as well as the updating of existing data, to reflect the changes made by the final users or background system processes.
- *Near real-time data streaming*: a process to maintain an open channel for the back-end continuously streams of data to the front-end, allowing near real-time data and notifications to be pushed to the user's interface.
- *Data synchronization*: ensures that the data presented in the front-end is current and consistent with the back-end, minimizing discrepancies and maintaining system reliability.
- *Data transformation and visualisation*: processes for transforming raw data into a user-friendly format, such as tables, graphs, or maps, enabling users to easily interpret and interact with complex datasets.

3.4 Integration of User Interface Components

The integration of the DT-EWS system's front-end with its back-end was designed to provide an intuitive and interactive user experience. This integration depends on the interaction between the GUI and the various subsystems that compose the back-end, facilitated by a set of REST API endpoints. The integration followed a user-centric design approach, ensuring that each component not only functions in isolation but also as part of a greater whole, having the following aspects as guidelines:

- The GUI is the point of contact for the user, presenting a responsive interface that interacts with both the USE and the EWS subsystems. The integration ensures that the user's actions, such as scenario creation and configuration, or messages and alerts examination and updates, are exchanged effectively with the back-end through RESTful API calls.
- Data flow between the back-end and front-end is bi-directional, enabling both the retrieval of baseline data and user-generated content from the DB and the submission of user inputs, for example new or updated

scenario parameters or EWS sensor configurations, to the back-end. This seamless data exchange is critical for maintaining up-to-date and accurate representations within the user interface.

- Near real-time data streaming and data synchronization is implemented to provide users with immediate updates from USE and EWS simulations, as well as sensor inputs and weather forecast services, which are crucial for the EWS subsystem's continuous monitoring and alerting functionalities. This streaming capability ensures that the front-end reflects the most current situation of the monitored coastal city.
- The Web Application component within the front-end includes robust authentication and session management features, ensuring secure access and protecting sensitive information. This integration of security features is critical for maintaining the integrity and confidentiality of the data and the user sessions.
- The integration process ensures that the functionalities provided by the back-end are accessible and usable through the front-end. The REST APIs serve as the axis for this coherence, providing well-defined endpoints for each user operation, such as Scenario EBAs implementation or setting of various environmental parameters.
- The client-server interaction is designed to utilize the structured data dictionaries defined by the API endpoints, handling requests and responses to facilitate all user operations. The GUI client leverages these dictionaries to parse and utilize data received from the back-end, ensuring a dynamic and responsive user experience.

As the SCORE project and the DT-EWS evolves, the integration of the system's user interface components will continue to be refined, taking into account user feedback and iterative testing. This iterative process is essential for ensuring that the system remains user-friendly and effective for the diverse needs of the CCLL stakeholders.

3.5 Interaction between Back-end and Front-End

In this section, the employed RESTful APIs enabling back-end and front-end connection are listed and briefly described.

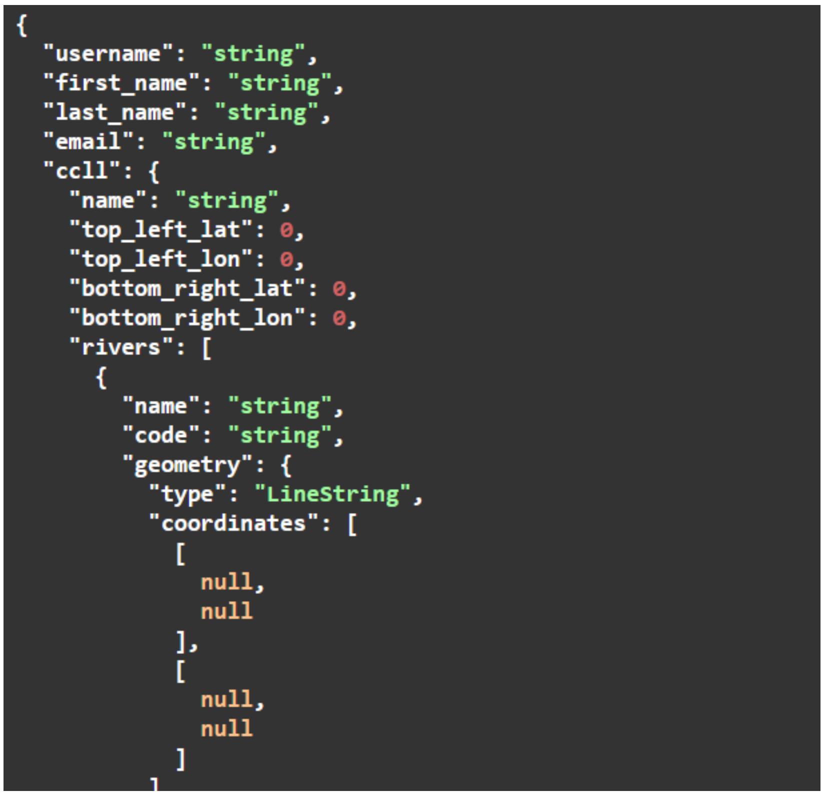
3.5.1 API Requests and Endpoints

The implemented RESTful APIs are reported in the following subsections. Rather than giving a complete list with APIs detailed description, here we explain the functions each interface allows to implement. The complete APIs list and related documentation can be found at the following link:

https://score-dt-api.mbigroup.it/docs#/

where the reader can find the most up-to-date APIs and related schemas, that can evolve in time agreeing to different system releases or to optimization activities, which also can be driven by feedback provided by the users.

At this page, the methods for each API are enumerated and the required input parameters and the response schemas are reported. As an example, the schema depicted in Figure 2 represents the data returned by the GET method of the *Users* API, which retrieves the information on the current user: username, first name, family name, but also data related to the CCLL and its area, like the list of rivers and other eventual city features.



```
{
  "username": "string",
  "first_name": "string",
  "last_name": "string",
  "email": "string",
  "ccll": {
    "name": "string",
    "top_left_lat": 0,
    "top_left_lon": 0,
    "bottom_right_lat": 0,
    "bottom_right_lon": 0,
    "rivers": [
      {
        "name": "string",
        "code": "string",
        "geometry": {
          "type": "LineString",
          "coordinates": [
            [
              null,
              null
            ],
            [
              null,
              null
            ]
```

Figure 2. Example of a schema of the data returned by the GET method of the Users API.

3.5.1.1 Authentication

This API is related to the user authentication at the time of login into the system through the GUI. It enables the generation (login) and deletion (logout) of an authentication token. This API can be employed for the USE as well as for the EWS subsystem.

3.5.1.2 Users

This API is employed to retrieve information related to the current logged-in user, like its username, first and last name, CCLL, and similar. It can be employed for the USE as well as for the EWS subsystem.

3.5.1.3 Scenarios

This API is devoted to scenarios management, allowing for their creation, updating, and deletion. Moreover, it also gives the possibility to list all the user-created scenarios and retrieve the related data. It is employed for the USE, since no scenario is created by users in the EWS subsystem.

3.5.1.4 EBA Type

Each user can simulate the implementation of one or more EBAs, choosing them from a palette of 33 possible solutions indicated by the SCORE project, reported in [10]. This API allows listing all the available EBA types. It is employed for the USE, since no scenario EBA is tested by users in the EWS subsystem.

3.5.1.5 EBA Management

The EBAs can be created and included in an user-created scenario thanks to this API, which is employed in general for managing the simulation of EBA solutions: it can retrieve and list data, or update the features associated to an EBA already existent in the scenario. Like the previous one, this API is exclusively employed in the USE subsystem, since no EBA is tested by users in the EWS.

3.5.1.6 Sea State

In the current issue of the DT-EWS system, for what-if analyses employing the USE simulations, the user has two options: arbitrarily setting the sea level height and time duration, or choosing from a list of pre-configured sea-state data sets derived from projections based on RCP scenarios [9]. This API allows the users to arbitrarily set the sea height and the timeline; it contains methods that allow users to read and update the sea state in the scenario. Since the users cannot arbitrarily set the sea state in the EWS, this API is exclusively employed in the USE subsystem.

3.5.1.7 Rain Rate

Users can set rain events defining their duration and the time distribution of the rain rate. This API and its methods enable the users to read and update the rain rates of an already created event in a scenario. Since the users cannot arbitrarily define weather events in the EWS, this API is exclusively employed in the USE subsystem.

3.5.1.8 River Discharge

In the creation of their scenarios, users can set the water flow for each river defined in the study area of the coastal city, and each river will be assigned an identification number. Like in the case of sea state conditions, users can either arbitrarily set the river discharge events in terms of timeline and intensity, or select from a list of 30 long-term events derived from RCP projections [9]. This API allows users to set or retrieve values for parameters related to the river discharge for each stream considered in the scenario, and to eventually update it. Since the users cannot arbitrarily set river flows in the EWS, this API is exclusively employed in the USE subsystem.

3.5.1.9 USE Simulation

This API is intended to give a user the methods to create and run the simulation of a scenario. In addition, through it, it is also possible to get the information on the simulation status or to delete it. This API is employed for the USE subsystem, since the EWS tasks run in loop, without needing inputs from the users after the start.

3.5.1.10 EWS Simulation

This API allows users to get the status of the elaboration performed by EWS subsystem. Indeed, the EWS block runs its calculations on the actual situation of the coastal city each 15 minutes. As explained in Subsect. 2.1.2, warnings and alerts can be generated; as long as there is no inconsistency in the data and there is no foreseen risk of flooding, there is no warning or alert to be raised, and the system just compiles a log file [2]. This API is periodically and automatically called to get the status of the system processing and, if there is an alert or warning, the system notifies the users with a message on the GUI. It is worth noticing that only in case of an alert the EWS subsystem sends direct messages to one or more designated users.

3.5.1.11 USE Results Publication

When a simulation ends, it produces different outputs, as described in detail in [2], and the owner of the simulation is asked whether to publish or not the full package of output files, i.e., to store them on the SIP. Thanks to this API, the results can be uploaded by the user on the SIP server, and data on the status of the publishing process can be retrieved. This API can be employed exclusively for the USE subsystem, once the outcomes of a simulation are ready.

3.5.1.12 EWS Results Publication

Since the EWS subsystem does not produce results as outputs of user-launched simulations, but runs calculations yielding output maps with flood risk every 15 minutes, the last *N* outputs are stored in the internal system DB. *N* is an integer number chosen by the user at the moment of system setting and deployment. However, after being notified of an alert, the user can choose to save the output maps related to the warning on the SIP, so to have the possibility to analyse them once the eventual flood emergency is over.

3.5.1.13 Sensors Management

This API endpoint is crucial for the functionalities of the EWS, since it enables listing the sensor stations located in the users' CCLL area and the generated data stream from each sensor. These official sensors and their data observations, apart from the standard visualisation in the front-end, are used for the EWS subsystem simulations. If the EWS detects a problem with the operation of a sensor, or with the accuracy of the data that can affect the continuous simulations, the sensor is flagged and this is notified on the front-end. Thus, the user can update the availability of a sensor, i.e., it is possible to check the sensor data and decide to exclude (disable) or include (enable) a given sensor from the sources of data used by EWS simulations. This endpoint is not employed by the USE directly, whose operation is focused on what-if analysis, however the visualisation of sensors and observations data streams are available as layers on the Web GIS front-end interface.

3.5.1.14 Long-Term Scenarios

As mentioned in Sec. 2.1.1, and in subsect. 3.5.1.6 and 3.5.1.8, in the creation of scenarios in the USE subsystem, the users can decide to employ the data sets related to river discharge and/or sea state events from the long-term scenarios, as they have been defined following the IPCC projections on RCPs [9]. These pre-compiled events can be selected by the user thanks to a step-by-step guided procedure from the GUI, following the tree structure sketched in Figure 3, where each leaf is associated to a data set for the relevant parameters of the considered events.

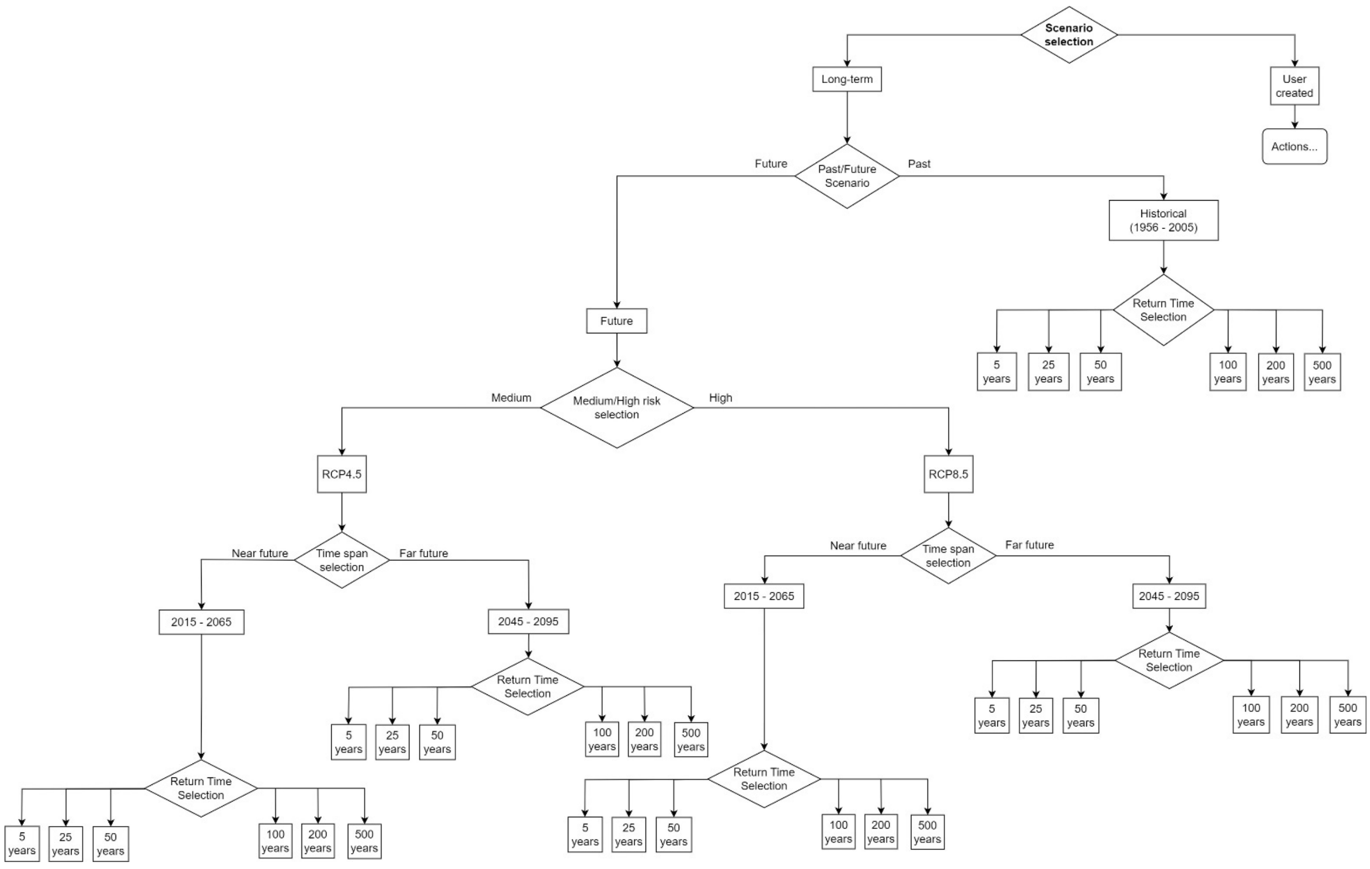


Figure 3. Tree structure of the selection procedure for the long-term scenarios. This structure is the same for river discharge and sea state events. Each tree leaf is associated to a specific event data set.

When setting the parameters for events related to sea state conditions or to river discharge, users are first asked to choose among user-created and long-term pre-compiled scenarios. In the first case, they will insert the relevant parameter and timeline, and the GUI will call the methods of the APIs described in Subsects. 3.5.1.6 and 3.5.1.8. In the second case, the users keep following the guided procedures thanks to combo boxes appearing on the GUI that will ask to select between past or future scenarios, risk levels of scenarios (medium, associated to RCP4.5, or high, associated to RCP8.5 [9]). Then, another binary choice is asked regarding the time span over which the analysis is considered and, finally, the user must select an event based on the associated return time. The same structure applies to sea state and to river discharge events.

The selection steps are managed through calls to the API, thanks to the GET method. As shown in the related schemas *LongTermScenarioCode*, *LongTermScenarioChoice*, and *LongTermScenarioQuestion*, each selection option has an associated code that the GUI and the back-end exchange at each step, guiding the users in their choice. These schemas are included in the documentation page whose link is reported at the beginning of this section.

3.5.2 Visualization of the Simulation Results

The GUI is the fundamental visualization tool that will enable the users to view and evaluate the outcomes of the USE simulations or of the EWS operations.

3.5.2.1 USE Section

At the end of each simulation, the system produces different output map layers with associated files and a human readable report [2]. The latter includes pictures of the most relevant of those layers, to let a user evaluate the consistency of the outputs or its interest in further analysing them. If the user deems the results worthy to be thoroughly analysed, the user has the possibility to save all the produced outputs on the SIP, thanks to the publication API. However, each produced map can always be visualised on the GUI individually.

3.5.2.2 EWS Section

As already illustrated in Sect. 2.1.2, the EWS has a different operational mode than the USE, since users do not launch simulations on it, but rather they are alerted in case of flood risk or warned in case of anomalies in the data received from sensors. However, when an alert is raised, the user is notified and can immediately visualise on the GUI the associated flood risk map and the human and financial risk map. The risk is represented by color-coding the map cells, depending on the risk level, along with the expected damage expressed in Euros.

3.5.2.3 The Tile Map Server

As already mentioned, the users are enabled to visualize the output map layers on the GUI itself, without employing external tools, thanks to an on-purpose developed tile map server. The employed server supports the Web Feature Service (WFS) interface standard [12], which allows for requests for geographical features across the web using platform-independent calls.

At the output, both the USE and EWS subsystems generate the map layers as raster files with associated features contained in GeoJSON files, that are uploaded on the tile server. An URL that the GUI can call is generated for these contents and, through WFS methods, users can browse the maps and visualize the map features, with the possibility of zooming in and out.

4. CONCLUSIONS

One of the main objectives of the SCORE project is the realization of a DT-EWS system to assist the governance of the involved coastal cities in collaborative climate resilience management strategies, allowing the evaluation of EBAs and supporting in disaster prevention initiatives.

In the first quarter of 2024, the deployment and in-field testing of the implemented DT-EWS will begin, starting from the coastal city of Massa (Italy), representing the pilot case for the WP8 activities. Later, the activities will continue with the other WP8 front-runners CCLLs. Deployment operations will not only imply the acquisition of all the city baseline data and the data streams from sensors and weather forecasts: it will also be crucial to introduce the DT-EWS to a group of test users, along with the training to explain them how to operate with the system. Therefore, to ensure efficient and easy interactions between the users and the DT-EWS, a proper design of the front-end is of paramount importance, together with its integration with the back-end section of the system.

In this document, we reported about the approach followed for the integration of the DT-EWS back-end with the *ad-hoc* developed GUI, also illustrating the internal interactions between the system sections, which are governed by the described REST APIs.

REST APIs embody a set of principles and characteristics that prioritize simplicity, scalability, and interoperability. By adhering to a stateless client-server model, utilizing standard HTTP methods, and representing resources through URIs, RESTful APIs provide a versatile and widely adopted approach to building modern, distributed web systems. Their simplicity and compatibility make them a cornerstone of the development of systems like the SCORE DT-EWS, since they foster efficient communication between diverse applications and services. Thanks to these APIs, users can easily call system functions from the GUI, for selecting data to be inserted in their scenarios, setting events parameters, retrieve information, and visualising the outcomes of the DT-EWS system processing. Since REST APIs are widely adopted by many software systems, they guarantee a large interoperability of the DT-EWS with external systems.

The forthcoming deployment activities will be based on the interactions between users and system developers, and the latter will exploit the feedback received from the former to finely tune the system design and tailor it on the users' needs. This will pave the way to a complete testing and validation of the system functionalities and performance, following the procedures extensively described in the assessment plan D8.10 [11].

5. REFERENCES

[1] SCORE D8.4 – Early Warning and Digital Twin Platform prototype

[2] SCORE D8.5 – Early Warning and Digital Twin Platform release notes and user manual

[3] SCORE D8.6 – Tools to access and visualise SCORE data and outcomes

[4] SCORE D8.7 – Tools to access and visualize SCORE data and outcomes release notes

[5] SCORE D8.1 – GIS Based Early Warning and Digital Twin Platform functional requirements

[6] SCORE D8.2 – GIS Based Early Warning and Digital Twin Platform system architecture and design

[7] SCORE D8.3 – GIS Based Early Warning and Digital Twin Platform Interface Control Document

[8] SCORE D6.3 – Exposure database and vulnerability curves for the frontrunners CCLLs

[9] IPCC, *Climate Change 2014: Synthesis Report*. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 2014, 151 pp.

[10] https://storymaps.arcgis.com/stories/6cdbb2f6ab0744b89dffda2664dd877e (last accessed 11/12/2023)

[11] SCORE D8.10 – Early-Warning and Spatial Digital Twin Assessment Plan

[12] https://www.ogc.org/standard/wfs/ (last accessed 11/12/2023)