



RESILOC			
<i>Resilient Europe and Societies by Innovating Local Communities</i>			
Grant Agreement No		833671	
Start date	01.06.2019	End date	31.05.2022

D4.1 – RESILOC Inventory Developed

Submission date: 20.12.2021

Western Norway Research Institute, Norway

Revision	Organization & Person	Date
Written by	WNRI, Hoang Long Nguyen	06.12.2021
Checked and approved by	FhG, Shayan Davari Fard	10.12.2021
Validated and released by	NKUA, Vassilis Papataxiarhis	17.12.2021



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833671

I. Deliverable Information

Deliverable Number	D4.1
Work Package	WP4
Date of Issue	20/12/2021
Version Number	3.0
Nature of Deliverable	Report
Dissemination Level (PU / RE / CO)	PU

Author(s)	Hoang Long Nguyen (WNRI) Rajendra Akerkar (WNRI) Salvatore Marchese (IES) Valentino Gandolfo (IES) Leonardo Luca Trombetta (IES) Vassilis Papataxiarhis (NKUA) Mohammadrahim Masoumi (FhG)
Keywords	RESILOC Inventory, Agile development, Web-based application, Resilience-related information

Abstract

This document provides a comprehensive description of the RESILOC Inventory developed as a web-based set of user interfaces, backend services, and databases. To achieve this, RESILOC follows an Agile approach by breaking the development task into four sprints, which involves constant collaboration with community users, offering continuous improvement, and getting approval from stakeholders at every stage. Moreover, technical features used to develop inventory frontend, inventory backend, and open data middleware are presented in detail to illustrate how the system is developed. Finally, the deliverable gives an overview of the verification plan of the inventory.

II. Document History

Date	Version	Modified by (first name, name, organization)	Remarks
04.10.2021	0.1	Hoang Long Nguyen, Rajendra Akerkar (WNRI)	Deliverable outline and table of contents
07.10.2021	0.2	Hoang Long Nguyen (WNRI), Salvatore Marchese, Valentino Gandolfo, Leonardo Luca Trombetta (IES)	Consolidation of the first round of contributions
11.10.2021	1.0	Hoang Long Nguyen (WNRI), Valentino Gandolfo, Leonardo Luca Trombetta (IES), Vassilis Papataxiarhis (NKUA), Mohammadrahim Masoumi (FhG)	Contributions to the live document
22.10.2021	1.1	Hoang Long Nguyen, Rajendra Akerkar (WNRI)	Interim review
25.10.2021	1.2	Hoang Long Nguyen (WNRI), Salvatore Marchese, Valentino Gandolfo, Leonardo Luca Trombetta (IES), Mohammadrahim Masoumi (FhG)	Additional contributions to the live document and updates after interim review
02.12.2021	1.3	Hoang Long Nguyen (WNRI), Salvatore Marchese, Valentino Gandolfo, Leonardo Luca Trombetta (IES), Vassilis Papataxiarhis (NKUA), Mohammadrahim Masoumi (FhG)	Internal check
06.12.2021	2.0	Hoang Long Nguyen (WNRI)	Updates after internal check
08.12.2021	2.1	Shayan Davari Fard (FhG)	Internal peer review
13.12.2021	2.2	Hoang Long Nguyen (WNRI), Valentino Gandolfo (IES)	Updates after internal peer review
14.12.2021	2.3	Vassilis Papataxiarhis (NKUA)	Quality assurance
17.12.2021	2.4	Hoang Long Nguyen (WNRI)	Minor improvements after quality assurance
20.12.2021	3.0	Hoang Long Nguyen, Rajendra Akerkar (WNRI)	Final document for submission

Disclosure Statement:

The text, figures and tables in this report can be reused under a provision of the Creative Commons Attribution 4.0 International License. Logos and other trademarks are not covered by this license. The content of the documents marked as restricted or confidential are not to be disclosed externally without prior written consent from the RESILOC Consortium, that can be requested via resiloc-dpo@fraunhofer.de. The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services. While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the RESILOC consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose. Neither the RESILOC Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein. Without derogating from the generality of the foregoing neither the RESILOC Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

III. Table of Contents

I.	Deliverable Information.....	i
II.	Document History.....	ii
III.	Table of Contents.....	iv
IV.	List of Figures.....	vi
V.	List of Tables.....	viii
VI.	List of Acronyms.....	ix
1	Executive Summary	1
2	Introduction	2
2.1	Purpose and Scope.....	2
2.2	Target Audience	3
3	Inventory Specification	4
3.1	Overview	4
3.2	Technical Environment.....	5
3.2.1	Server and Hardware	5
3.2.2	Development Techniques.....	7
3.2.3	Privacy and Security.....	9
3.3	System Requirements	12
3.3.1	Usability	13
3.3.2	Reliability.....	13
3.3.3	Performance	13
3.3.4	Interfacing	14
3.3.5	Packaging	14
3.3.6	Supportability	14
3.4	Ethics Considerations and Data Protection	14
4	AGILE Implementation of RESILOC Inventory	16
4.1	Development Process	16
4.2	User Feedback Analysis.....	19
4.3	Backend Service and Functions	21
4.3.1	Backend Endpoints	21
4.3.2	Response Status Code	28
4.3.3	Utilisation Functions	29
4.4	User Interface.....	32
5	Open Data.....	41
5.1	Open Data Introduction	41
5.2	Open Data Platform.....	41
5.3	Architecture.....	42

5.4	Middleware Technical Overview	43
6	System Verification.....	44
7	Conclusion	46
VII.	List of References	i
VIII.	Annex A: Ethics Self-Assessment Sheet	ii
IX.	Annex B: RESILOC Inventory User Manual.....	v

IV. List of Figures

Figure 1. The RESILOC Inventory development process.	17
Figure 2. The RESILOC Inventory API documentation.	18
Figure 3. The RESILOC Inventory backend observability.	19
Figure 4. Authentication APIs.	22
Figure 5. User APIs.	22
Figure 6. User role APIs.	22
Figure 7. Community APIs.	23
Figure 8. RESILOC scenario APIs.	23
Figure 9. RESILOC indicator APIs.	24
Figure 10. RESILOC proxy APIs.	25
Figure 11. Community scenario APIs.	25
Figure 12. Community indicator APIs.	26
Figure 13. Community proxy APIs.	26
Figure 14. Snapshot APIs.	27
Figure 15. Timeline APIs.	27
Figure 16. Open data APIs.	28
Figure 17. HTTP status codes of a user GET endpoint.	29
Figure 18. Multiple languages supported for translation purpose.	29
Figure 19. Pagination parameters of GET endpoints.	30
Figure 20. Ordering and sorting parameters of user GET endpoints.	30
Figure 21. Filtering parameters of user GET endpoints.	31
Figure 22. Sample GET request to get community indicators according to different criteria.	32
Figure 23. Sign in and sign-up module.	32
Figure 24. RESILOC Inventory dashboard.	33
Figure 25. User menu.	34
Figure 26. Sidebar layout and structure of community side.	34
Figure 27. Sidebar layout and structure of RESILOC admin side.	35
Figure 28. The main body of the page showing the panel containing the functionalities associated with the selected item (the 'Indicators' menu).	36
Figure 29. Examples of the RESILOC Inventory interface tooltips, providing clarification where labels do not apply.	37
Figure 30. Explanatory instructions and warning systems to simplify the filling in of specific form.	38
Figure 31. Language selection function by drop-down list.	39
Figure 32. An example of the filtering function applied to the list of RESILOC Indicators.	39
Figure 33. An example of the sorting function applied to the list of Indicators from the RESILOC Admin interface.	40

Figure 34. RESILOC Open Data architecture.	42
---	----

V. List of Tables

Table 1. Servers and hardware specifications used in the RESILOC project.	6
Table 2. Create, Read, Update, and Delete (CRUD) operations by user roles.....	12
Table 3. RESILOC Inventory backend response status codes.....	28

VI. List of Acronyms

Acronym	Meaning
API	Application Programming Interface
CA	Certification Authority
CRUD	Create Read Update Delete
CPU	Central Processing Unit
D	Deliverable
DB	Database
EU	EU
H2020	Horizon 2020 Programme
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ISO	International Organization for Standardisation
IQ	Installation Qualifications
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JWT	JSON Web Token
LRTs	Local Resilience Teams
LTS	Long Term Support
MVC	Model View Controller
NoSQL	Not Only Structured Query Language
ODM	Object Data Modelling
OpenAIRE	Open Access Infrastructure for Research in Europe
OQ	Operational Qualifications
ORM	Object Relational Mapper
OS	Operating System
PQ	Performance Qualifications
RAM	Random Access Memory
RBAC	Role-Based Access Control
RDF	Resource Description Framework
REST	Representational State Transfer
RTM	Requirements Traceability Matrix
SaaS	Software as a Service
SSH	Secure Shell

SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
T	Task
TLS	Transport Layer Security
UAT	User Acceptance Test
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
WP	Work Package

1 Executive Summary

This D4.1 entitled “AGILE Implementation of RESILOC Inventory” presents the AGILE development details of the inventory, which is the first released tool in the Software phase of the RESILOC project. This deliverable is heavily interlinked and built on top of the outcomes and knowledge extracted from the design performed in D2.7 (Architecture of the RESILOC Inventory) and made use of the feedback derived from communities in 4 different sprints from December 2020 to November 2021 to identify the essential features and required refinements on the components of the inventory. Generally, this is a public document and can serve as a comprehensive view of the proposed solution, used for either the Software or the Trial phase, to communicate the project’s accomplishments.

More specifically, the objectives of the deliverable D4.1 are as follows:

- Provide information at high-level technical features, hardware and software components, and critical quality requirements of the inventory.
- Present details about the Agile implementation approach of the inventory, which considered all the users' feedback derived from 4 sprints, including the methodologies, technical documentation management, source code management and deployment of both frontend and backend.
- Present technical details about the fully integrated Inventory in both frontend and backend development.
- Provide specifics on how RESILOC develops and publishes community data as open data.
- Provide overview of planned verification and validation of the inventory system.

While the deliverable D4.1 is the first document released in the Software phase, it accompanies in month 30 with the D4.6 (RESILOC Platform Developed) – an internal release that summarises the technical activities that were engaged so far in WP4, sketches the current plans, and provides an outlook of how we plan to construct the RESILOC Platform. Moreover, for the cooperation with WP5 for the trials, a thorough and comprehensive understanding of what the inventory is going to be will also be very useful.

2 Introduction

Deliverable 4.1 is the first of six deliverables in Work Package 4 (WP4). In this section, a general introduction of this document, the RESILOC Inventory, as well as target audience are given.

2.1 Purpose and Scope

As a part of technical activities in the Software phase, Deliverable 4.1 reports how RESILOC uses an Agile approach (Cockburn, 2006) to incrementally and iteratively develop the inventory – a scalable database solution to facilitate resilience-related information awareness, comprehensive data management, and open data exchange for all community partners. Ultimately, this leads to the development of an integrated RESILOC cloud platform afterwards, which will be used to assess resilience at the community level.

The RESILOC Inventory is developed as a web-based set of interfaces, services, and datastores, providing end-users and researchers the possibility to populate and make use of the RESILOC Inventory SaaS. The tool will be developed on top of Open Source based technologies (e.g., JavaScript, PostgreSQL, and MongoDB) and will run on a Cloud infrastructure (either commercial or research provided) to be easily scaled and maintained. The approach to follow will be distributed, and the inventory itself will be highly customisable for future evolution and future-proof capabilities.

In general, the inventory provides an effective UI to help users manage their data. The inventory UI also offers the user varied features for collecting and organising data through numerous functions at hand such as template inheritance, data duplication, event notification, and searching, sorting & filtering options. Running in parallel and handling requests from the UI is the inventory backend that aims at processing all heavy tasks in the background. It ensures performing all the required actions in the most efficient and logical manner, spending as little time and resources as possible.

In the inventory, each entity is hierarchically linked to its higher-level semantics. The basic idea involved here is that a proxy is an atomic entity, which can be associated with multiple indicators. Similarly, each indicator can be connected to numerous scenarios. The level of consistency of entities' relationships along with defined parameters tied to those relationships ensures that resilience assessment is achievable and dependable.

The data entry level, i.e., snapshot – a logical copy of a scenario at a particular point in time, is continuously monitored by the inventory. Once any of them are submitted by users, the Semantic Layer service will be alerted by the inventory so that it can start assessing the resilience value of a snapshot. The resilience calculation is based on previously accumulated information/knowledge entered by community users. Hence, the users certainly have not to worry about manually calculating resilience as the inventory and the Platform will inevitably carry out it for the user.

We also propose to include in this deliverable a special feature known as “Open Data”. RESILOC open data keeps track of and publishes all community data, which is configured by community users as “public” visibility, for the sake of impactful re-use, especially through inter-sectoral collaborations and partnerships among communities.

The inventory will be used by the four local community partners of the RESILOC project. Target groups are the local authorities, first responders and their emergency managers, citizens, and civil society. A potential extension to further target groups should be expected.

A functional mock-up version of the inventory had been introduced with end-users in December 2020 in the 1st sprint, followed by a hands-on user story and real-time use in the 2nd sprint in April 2021. Next, it had been made available for actual access and use by all communities since the 3rd sprint in September 2021. RESILOC developers leveraged those sprints as an excellent opportunity to derive detailed feedback and suggestions from communities for further and essential improvement. Concerning each sprint, end-users were requested to “give an approval to continue the implementation” or “request to stop and rethink”.

The RESILOC Inventory was up and will be running from September 2021 until the end of the RESILOC project. Since the inventory is a Web-based service and always online, all changes that have been quality controlled can be immediately reflected in the live mode for actual uses by users. Possible more extended usage should also be considered. Maintenance after the project will need to be discussed and agreed upon during the course of the project.

2.2 Target Audience

This document will be made publicly available. The results of the inventory developed are especially interesting for people working for organisations in the local authorities (e.g., policy makers, policy consultants, and technical services operating in the community), first responders, emergency services, and the LRTs.

More specifically, the target audience for this deliverable includes:

- Partners and Advisory & Reference Group in the RESILOC project.
- European Commission.
- European Commission's delegates.
- Horizon 2020 projects and related resilience projects in the H2020-EU.3.7.5. program.
- Communities involved in the RESILOC trials.
- Authorities (regional & local level) aiming at increasing resilience to crises and disasters.
- Public and private organisations that develop and reinforce their capacity to react to disasters.

This deliverable is applicable towards non-technical audiences since a high-level overview of the inventory is also described at the beginning of the deliverable, i.e., Section 2 (Introduction) and Section 3 (Inventory Specification).

3 Inventory Specification

This section first provides a high-level overview of the RESILOC Inventory features along with user roles defined. Furthermore, the technical specifications are described in detail, showcasing how the inventory components are developed, connected, and interact efficiently with each other. Finally, quality requirements are mentioned to ensure the effectiveness of the entire system. While D2.7 focuses on the architecture of the RESILOC Inventory, this section concentrates more on the implementation process.

3.1 Overview

The RESILOC Inventory aims at providing an efficient interface for different types of community users to manage community-based resilience data. It shall provide the users with varied options to manage their data effectively and efficiently through various functions at hand.

Overall, the inventory supports the primary following features:

- RESILOC admin side:
 - Admin Dashboard.
 - RESILOC user management – create, read, update, and delete (CRUD) users.
 - RESILOC community management – CRUD communities.
 - RESILOC proxy management – CRUD RESILOC proxies.
 - RESILOC indicator management – CRUD RESILOC indicators.
 - RESILOC scenario management – CRUD RESILOC scenarios.
 - Open data creation and sharing
- RESILOC community side:
 - User registration and authentication.
 - Community subscription – let users follow communities.
 - Community profile configuration.
 - Community user management – update the role of community users.
 - Community proxy management – import, create, read, update, and delete (ICRUD) community proxies.
 - Community indicator management – ICRUD community indicators.
 - Community scenario management – ICRUD community scenarios.
 - Community data entry management – CRUD snapshots.
 - Assessment timeline overview.

With regards to RESILOC-elements, i.e., RESILOC proxies, indicators, or scenarios, they are formal templates generated by the RESILOC admin according to what is researched, defined, and verified in the T3.1 (Definition of resilience indicator and matrix) and T3.2 (Definition on new strategies for improving resilience). If community users find any of those templates to be helpful and relevant to their communities, they can inherit them to form community proxies, indicators, and scenarios without the need to put everything in by hand or from scratch. Our inventory allows users to highly customise the data to suit the needs of each community individually. Once inheriting from RESILOC templates, they have the freedom to modify inherited entities from proxy level, or users can create their own community proxies, indicators, or scenarios if there are no appropriate templates provided.

The users of the RESILOC Inventory will have different permissions and abilities in accordance with the role they are given for a particular community. The roles currently defined in the platform are listed below:

- RESILOC admin: users who come from the RESILOC consortium responsible for the management and configuration of the RESILOC entities mentioned above and guarantee the inventory's performance, integrity, and security.
- Community side:
 - Community admin: the users who are responsible for managing and assigning roles for other community users. Community admins are set by RESILOC admins to generate a working environment for a community.
 - Local manager: the users who have control over community settings (such as community description, metadata, or privacy settings) and can create, view, edit, and remove community entities, i.e., community proxies, indicators, scenarios, and snapshots.
 - Resilience expert: the users who own in-depth knowledge about resilience. They can assist local managers in organising community entities for resilience assessment.
 - Local Resilience Team: the members of LRTs that have full access to view all the community configurations that resilience assessment results.
 - Citizen: this is the lowest-level role that is automatically given to a user when he/she starts following a community. This user will have the ability to view public community configuration and resilience assessment information only.

For a detailed description of the RESILOC Inventory, we can refer to the deliverable D2.7. After 21 months, the implementation of the RESILOC Inventory came to an end. The system undergoes several quality checks by development teams and uses by community users before being declared finalised. Further minor changes will be possible throughout the duration of WP4.

3.2 Technical Environment

The RESILOC Inventory consists of different IT components which need to be developed, assembled, and executed by technology partners of the consortium:

- The inventory hardware components are provided and supported by the National and Kapodistrian University of Athens.
- The inventory frontend and backend are designed and implemented by IES Solutions and Western Norway Research Institute, respectively. Each team is given the freedom to set up its own development environment. However, specific interfaces were discussed, agreed upon, and set up to integrate these two components so that they can interoperate, communicate, and work together in an incorporated manner.

3.2.1 Server and Hardware

In RESILOC, a cloud-based infrastructure was set up to be used as an integration environment in the form of interworking Virtual Machines (VMs) in order to host the foreseen RESILOC services and provide the required computational, memory and storage resources. For the sake

of generality, all the resources of the RESILOC Platform are given in this section but the ones related to the inventory will be highlighted in bold in Table 1.

The following resources were allocated in total:

- 22 cores in total,
- 46 GB of RAM,
- 790GB of disk space.

The above cores/memory/disk were distributed to several RESILOC VMs according to the project needs. In particular, the technical characteristics of the RESILOC VMs are the following.

Table 1. Servers and hardware specifications used in the RESILOC project.

#VM	Description	# CPU cores	#RAM	#Disk	OS
VM1 resiloc1.di.uoa.gr	FRONT-END	2	4 GB	40 GB	Ubuntu 20.04.02 LTS Server
VM2	API GATEWAY	2	4 GB	40 GB	Ubuntu 20.04.02 LTS Server
VM3	INVENTORY BACKEND & DB - PRODUCTION ENVIRONMENT	2	4 GB	100 GB	Ubuntu 20.04.02 LTS Server
VM4	CLOUD PLATFORM BACKEND & DB	2	4 GB	100 GB	Ubuntu 20.04.02 LTS Server
VM5	SEMANTIC LAYER	2	4 GB	100 GB	Ubuntu 20.04.02 LTS Server
VM6	MOBILE APP BACKEND + DB	2	4 GB	40 GB	Ubuntu 20.04.02 LTS Server
VM7	DYNAMIC DATA GATHERING SERVICE	2	4 GB	100 GB	Ubuntu 20.04.02 LTS Server
VM8	GITLAB	2	8 GB	50 GB	Ubuntu 20.04.02 LTS Server
VM9	REPOSITORY BACKUP SERVICE	2	4 GB	100 GB	Ubuntu 20.04.02 LTS Server
VM10	SURVEY TOOL	2	2 GB	20 GB	Ubuntu 20.04.02 LTS Server
VM11	INVENTORY BACKEND & DB – DEVELOPMENT ENVIRONMENT	2	4 GB	100 GB	Ubuntu 20.04.02 LTS Server
	TOTAL	22	46 GB	790 GB	

Each VM is based on a command-line Ubuntu server environment (Ubuntu 20.04.02 LTS Server). All the VMs were physically hosted in the cloud environment of NKUA located in the campus of the NKUA Department of Informatics and Telecommunications (DIT) in Athens, Greece. The VMs were equipped with an Ubuntu Server OS environment and separate IP addresses were allocated for each VM providing access through a high-speed internet connection. In particular, a dedicated network was provided with a bandwidth up to 1000Mbps (through ethernet) both for uplink and downlink. However, since all the RESILOC VMs are hosted in the same physical server, the communication between VMs is in the range of several Gbps (depending on the RAM allocated for each VM). Secure remote access was provided to the responsible partners (using SSH) to customize and configure each one of the RESILOC services. More CPU, RAM, and disk space resources can be considered depending on the actual usage and throughput.

3.2.2 Development Techniques

Selecting precise technology stack is one of the most vital steps of the entire development life cycle. The technology stack refers to the set of technologies, programming languages, frameworks, and tools used to develop and deploy the inventory. Additionally, since the microservices designed in RESILOC interact with each other, the selection of tools and techniques must follow several standards, which are required to ensure full functionality as below:

- The inventory frontend and backend primarily communicate over the REST APIs (i.e., RESTful HTTP (W3C, 2011)). It is also the method for the inventory to communicate with other microservices in RESILOC, e.g., with the Semantic Layer or Open Data services.
- The communication between the client (frontend) and the server (backend) is JSON - a widely used standard for web applications.
- To be given access, all requests must be authenticated using a token-based approach, meaning that logged-in users will obtain an access token - OAuth 2.0 Bearer Token (Jones and Hardt, 2012), valid for a limited period. It will allow users to retrieve and fetch specific information from the backend.
- Docker (Raj, Chelladurai and Singh, 2015) is used to deploy the inventory components quickly and reliably on developers' computers or servers (both development and production environments).

3.2.2.1 Frontend

The design of the RESILOC Inventory UI was done following specific criteria to simplify the work done by the end-user. Based on this conclusion, the approach we have followed is to create a UI that is simple (including elements, such as buttons, that are necessary within it, and removing the useless elements), intuitive (using interface components to help the user), and fast (using a framework to give quick responses to the user).

After an accurate analysis, we chose to build the user interface using a framework that meets all the required points and allows us to be very responsive to satisfy further end-user requests. The chosen framework is Angular¹ together with several add-ons that enrich the user

¹ <https://angular.io/docs>

experience with simple and intuitive features. It provides the tools as well as a design model to develop a project in a sustainable manner.

Angular is a structural MVC (Model-View-Controller) framework, based on the TypeScript programming language (Fenton, 2014), designed for developing web applications. An MVC framework, like Angular, is based on the split of tasks between the software components, it consists of:

- Model: is responsible for providing methods to get the application data.
- View: is dedicated to displaying data and iterating with users.
- Controller: receives the user's commands (through the view) by interacting with the model.

Angular provides a fast and straightforward implementation method; moreover, applications made in Angular are executed entirely by the browser after being downloaded from the server. Furthermore, it is supported by the most used browsers at the moment, like Chrome, Firefox, Edge, and Safari. The version used is 11.22.8.

The executions of Angular's code are done by NodeJS², based on JavaScript, C, and C++. Nodejs is a runtime system (a software that runs programming code) optimised for web applications. This system is object-oriented, meaning that it does not execute a list of predefined commands in order, but commands are executed based on external event interactions, e.g., when a user clicks a button in the interface. The current version used is 12.22.3.

The user interface is realised by the Nebular graphics library³, which adds additional components to facilitate the user experience. As well as providing support for responsive graphics, it offers several modules to support authentication and platform security. The version used is 7.0.0.

3.2.2.2 Backend

RESILOC makes use of NestJS⁴ for developing the inventory backend. The rationale behind the decision is that NestJS is an open-source, scalable, and progressive Node.js framework for creating compelling and demanding server-side applications. Moreover, it also supports diverse types of databases (e.g., PostgreSQL, MongoDB, and MySQL) and lots of nest-specific modules that help easily integrate with standard technologies and concepts like TypeORM, Mongoose, Passport, or Caching, and much more. Finally, with its ever-growing community (more than 40,000 stars on GitHub currently⁵), it turns out to be easier and simpler to derive a response to the burning queries rapidly.

The inventory uses PostgreSQL (Momjian, 2001) to store all the static, structured, and cross related data, including RESILOC templates and community proxies, indicators, scenarios, snapshots, and timelines because this SQL relational database offers full atomicity, consistency, isolation, and durability (ACID) properties at a high level. Moreover, PostgreSQL also supports JSON data types and operations, it facilitates the capacity to interact with a NoSQL database if required. We use TypeORM⁶, which is one of the most mature Object

² <https://nodejs.org/en/>

³ <https://akveo.github.io/nebular/7.0.x/>

⁴ <https://docs.nestjs.com/>

⁵ <https://github.com/nestjs/nest>

⁶ <https://typeorm.io/>

Relational Mapper (ORM) available in the Node.js world, to connect and model entities in the PostgreSQL database.

On the other hand, we use MongoDB (Bradshaw, Brazil and Chodorow, 2019) to store less relational data like communities, users, and system notifications. Furthermore, it acts as a backup plan to further store unstructured data such as social media and sensor data if required. MongoDB is considered one of the most common NoSQL stores in general and the most popular in the document store category guaranteeing low latency, distribution, and scalability and being able to deal with diverse types and structures of data. The data is stored in MongoDB as documents nested at arbitrary levels. Different documents may have different structures or fields, whilst document structure can be changed or updated over time. Mongoose⁷ – an Object Data Modelling (ODM) library – acts as an intermediate between MongoDB and NestJS server.

Lastly, we implement Redis⁸ as a caching solution to temporarily store frequently accessed data such as generated authentication tokens. Since data is stored in RAM instead of a hard disk, it enables faster responses.

3.2.3 Privacy and Security

The objective of this section is to describe the security strategies and regulations that are applied to guarantee the security of the inventory and the confidentiality, privacy, and visibility of the data collected, stored, and used in RESILOC. The security practices and principles implemented for both RESILOC Inventory frontend and backend are mentioned in the following sections.

3.2.3.1 Frontend

The frontend is developed following several Cybers Security guidelines aiming to ensure compliance with the security properties of *Integrity, Authentication, Confidentiality, Privacy of Personal Data, and Freshness* of information kept in the browser.

Integrity, Authentication, Confidentiality: in the field of communications over the Internet, the Authentication property is a noticeably significant question as there is no authentication at Network Layer in the ISO/OSI stack currently implemented with the IPv4 protocol. To mitigate the lack of network level authentication, what is secured is the *HTTP* protocol via the *Transport Layer Security (TLS)* protocol (Rescorla and Dierks, 2018), which provides authenticated *HTTP* connections with integrity and confidentiality properties, resulting in *HTTPS*.

Authentication is also carried out on another level, which is that of the authenticity of the public key of the web application: the clients that connect must have a guarantee that the key they use to authenticate the server is authentic. Through the *X.509* Certificate that is issued by the *Certification Authority (CA)*, it is possible to carry out the trust chain that guarantees that the public key used is really connected to the web application. The protocol that the RESILOC Web Application server supports is version *TLS1.3*. The certificate is of type RSA 2048 bits (SHA256withRSA) and the issuing Certificate Authority is "*Fraunhofer Service*".

⁷ <https://mongoosejs.com/>

⁸ <https://redis.io/>

With the authentication process described above, it would also be possible to authenticate the client through a certificate, but this is an authentication method that is not used nowadays, as it is preferable to authenticate through user registration (Sign Up process) with the web application. When a user authenticates to the system, a JSON Web Token (*JWT*)⁹ - a JSON encoded representation of a claim(s) that can be transferred between two parties - is issued and stored in the Local Storage of the client's browser.

The JWT is produced by an encoding algorithm known as the HS512 that encodes 2 JSON objects by applying a 256-bit signature to make the token unique. In the RESILOC Web Application the token encodes objects like the following:

```
{
  "alg": "HS512",
  "typ": "JWT"
}

{
  "username": "communityTester",
  "iat": 1635348774,
  "exp": 1635435174
}
```

The digest which is a string like the following

```
"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImNvbW11bml0eXRlc3RlcilzImIhdCI6MTYzNTM0ODc3NCwiZXhwIjoxNjM1NDM1MTc0fQ.
gAl27NmQSmARGDrN0AwCO0FpzFqYHBKpFGpr7p2VhCWsy1LSPaWrO1SuCqxi_27DBC
6iKStSicHHcOAmf6zdVQ"
```

is injected into the request header of the HTTP datagram whenever a request is submitted through the API to the backend. The HTTP datagram is modified Just-In-Time by an Angular interceptor specifically implemented to alter outgoing requests (Arias, 2020).

Freshness: Cookies are small pieces of information that a web application stores in the client for multiple uses. In common jargon, cookies are associated with information that is saved to identify a user during their navigation in order to provide them with advertisements that are tailored to their tastes, so-called profiling. In addition to these purposes, cookies are used to store personal preferences as well as parameters that the web application stores to work properly (Barth, 2011). Among the latter is the JWT parameter that is the result of the current client authentication implementation at the application level of the RESILOC web application. It is important to keep this information fresh, since the theft of this token by malicious users would make it possible to authenticate with an identity that is not one's own. To do this, the web application implements mechanisms to clean the Local Storage, effectively invalidating the token in the event of user disconnection. This is a precaution that safeguards the user from the possibility of disclosure of information such as the username.

⁹ <https://jwt.io/>

3.2.3.2 Backend

Authentication: this refers to proving the correct identity of a user. Except for only user registration, all other endpoints require authentication. The inventory backend applied Passport, which is the most popular node.js library, well-known by the community and successfully employed in various production applications, for the sake of secure authentication. From a high-level perspective, Passport help the backend executes a series of steps to:

- Authenticate a user by verifying their credentials including his/her username/email and password and return a JWT. From the backend side, all the passwords are encrypted and not possible to reverse the hashing process to reveal the original ones.
- Manage authenticated state based on through the issued JWT.
- Embed information regarding the authenticated user to the Request object for additional use in route handlers.

Authorisation: this refers to the method that defines what a user is able to carry out. For example, a local manager is granted full administrative permissions with community proxies while a user with a citizen role is only authorised to read them. We applied Role-based access control (RBAC), which is a policy-neutral access-control mechanism defined around roles and privileges, using Nest guards¹⁰ for authorisation. When applying RBAC (Sandhu, 1998), there are three guiding rules:

- Role assignment: A user can access an endpoint only if he/she has been assigned a role (by RESILOC admin or community admin).
- Role authorisation: A user's active role must be authorised. Together with the rule above, we ensure that users can take on only roles for which they are granted.
- Endpoint authorisation: A user can exercise an endpoint only if the endpoint is authorised for the user's active role. Along with the two rules above, this rule ensures that users can exercise only endpoints for which they are authorised.

Visibility: All the data in the inventory has privacy and sharing implications owing to a visible value that defines and targets its audiences. When saving any data, the users must specify whether the data is one of those visibilities:

- Draft: data marked as the draft visibility is accessible only with people who configurate or is responsible for organising this data. For example, a RESILOC proxy with draft visibility is available by RESILOC admins only, not yet by community users. From the community side, local managers and resilience experts can access community draft data.
- Restricted: data indicated as restricted is visible by only people in charge of the resilience assessment process, i.e., local managers, resilience experts, and LRTs.
- Community: data stated as the community visibility is accessible by all users that followed the community to which the data belongs.
- Public: this is the most open privacy level. Data declared as the public visibility is available from any users in any community. Public data will be shared as open data.

¹⁰ <https://docs.nestjs.com/guards>

Going along with visibilities mentioned are granular CRUD operations performed on all data as depicted in Table 2 below assuring that the whole database is protected. It is worth mentioning that we use hard delete (i.e., an operation in which data is completely erased from the database) when users request.

Table 2. Create, Read, Update, and Delete (CRUD) operations by user roles.

Data	RESILOC admin	Community admin	Local manager	Resilience expert	LRTs	Citizen
1. User						
a. Users of RESILOC	CRUD					
b. Users of a community	CRUD	RUD	R			
2. Community	CRUD	R	RU	R	R	R
3. RESILOC proxy, indicator, scenario						
a. Having draft and restricted visibility	CRUD		R	R		
b. Having public visibility	CRUD	R	R	R	R	R
4. Community proxy, indicator, scenario						
a. Having draft visibility	CRUD		CRUD	CRUD		
b. Having restricted visibility	CRUD		CRUD	CRUD	R	
c. Having community and public visibility	CRUD	R	CRUD	CRUD	R	R
5. Snapshot						
a. Having draft visibility	CRUD		CRUD	CRUD		
b. Having restricted visibility	CRUD		CRUD	CRUD	R	
c. Having community and public visibility	CRUD	R	CRUD	CRUD	R	R
6. Timeline	CRUD	R	CRUD	CRUD	R	R

3.3 System Requirements

The purpose of this section is to provide an overview of the essential requirements that the RESILOC Inventory should satisfy based on discussions with end users during the sprints, grant agreement, and developers' experience. For the sake of transparency and consistency, the defined conditions are categorised into six groups according to the quality attributes to which they are most relevant. Functional specification considering end-users' demands, collected during various meetings and gatherings, have been fulfilled and adopted as key concepts during the inventory design process as mentioned in the D2.7.

3.3.1 Usability

- The inventory must be easy to use and promptly accessible by local managers, resilience experts, LRTs, and citizens without the need to read and follow an extensive number of manuals.
- The inventory must be intuitive and straightforward in presenting all essential data and related relationships.
- All the menus, buttons, tooltips, etc. of the inventory must be easily discoverable, navigable, and comprehensible by all the users.
- The inventory should offer all the filtering, sorting, and searching functions to let users get the data in the desired way.
- The inventory must provide multi-language features using a one-to-one translation mechanism that can be used to translate any kind of content for the sake of convenience towards local users.

3.3.2 Reliability

- The inventory must provide both authentication and authorization mechanisms to users to confirm the identity and prevent breaches and any foreign entity retrieving, updating, or deleting the data.
- The inventory must react gracefully to improper usage and give precise and regular notification to the users to help them recognise any inaccuracies and errors that can cause mistakes or missing data.
- The inventory should provide the user information on the completion of requests and if any of them fail, it should give the user apparent reasons for the failure.
- The inventory should not make changes to any data in the database for any unsuccessful requests.

3.3.3 Performance

- The inventory must return data without noticeable delay. This mainly concerns pages with personalised dynamic contents e.g., community configuration and resilience assessment pages.
- Where applicable, asynchronous technologies such as AJAX¹¹ should be utilised to partially update views or continuously save information.
- All the essential functions of the inventory must be introduced to users and be ready every time the system is started. In case the inventory is down, it should be able to resume operation with the last stable state before the crash.
- Resource usage e.g., CPU, memory, and disk, must be monitored at the server side to predict peak loads and compare performance measurements at different points.

¹¹ https://www.w3schools.com/js/js_ajax_intro.asp

3.3.4 Interfacing

- The inventory's user interface should be designed to offer an efficient, ergonomic, and intuitive way of viewing and organising data. In addition, the interface provides roll-back designs for users to return previous steps if they have wrong operations.
- Pagination techniques must be applied to ensure not so much information is shown on the same page.
- To maintain consistency, the inventory should apply one template for different pages. Further, it must keep the interface behaviour the same for all actions to avoid causing users' surprise.
- Simple and pleasing will guide the inventory design.

3.3.5 Packaging

- The inventory must come along with a short manual that instructs users to precisely work on functions. Where essential, descriptions or tooltips should be placed onto the user interface directly.
- EU regulations and national laws regarding public web services need to be respected and followed. This particularly concerns accessibility, privacy, and security.

3.3.6 Supportability

- The inventory is built as a web service; hence, it is extremely compatible, and users can access the inventory proficiently even on computers not having modern specifications.
- The inventory must be adaptable even if additional communities, user roles, or languages are added later.
- The community data should be published as open data according to users' configuration to make the inventory more shareable, usable, and extensible.

3.4 Ethics Considerations and Data Protection

All activities that have been included in the report comply with Regulation (EU) 2016/679 known as General Data Protection Regulation (GDPR) and 2002/58/EC Directive on privacy and electronic communications as well as with relevant national data protection and privacy laws, codes of practice and guidelines. Two activities were conducted as part of Task 4.1, which involved data collection from individuals and, therefore, subject to ethical considerations. These included:

- Participation in 4 sprints, which are chances for the consortium to meet, evaluate, and give feedback on all software components. They are used to keep the requirements as close as possible to the needs of the end-users.
- User registration process allows users to access and interact with RESILOC Inventory functionalities during and after sprints. Personal data, including first name, surname, email address, and phone are stored. However, we implemented different security

methodologies to protect personal information as aforementioned in Section 3.2.3 (Security and Privacy).

All the sprints were done by providing participants with an information sheet to ensure that they were fully aware of the aims and purpose of the meetings and how the data collected would be used and asking them to also complete and sign an informed consent form before attending. The consent form provided additional information on how their data would be protected, how they could withdraw the participation from sprints, what their rights as research participants were, and who to contact if they had any questions or other concerns.

The work with research participants for this deliverable did not involve recruitment via social media. The process and activities were approved by the Author's Data Protection Officer as registered through the RESILOC Management. Only the data used in compiling this report has been retained. All other data collected incidental to this report has been permanently deleted on submission of this report.

4 AGILE Implementation of RESILOC Inventory

This chapter describes the usage of Agile software development (Schwaber and Beedle, 2002) to develop the RESILOC Inventory. It is open to changing requirements over time and encourages regular feedback from the end-users. The Agile implementation guarantees that the inventory is iterative, incremental, evolutionary, and adaptive through continuous delivery, integration, and collaboration. The agile methodology aims to shorten the time between the decision-making process and the feedback gained from the community users. This is a software development model used to build complex software and product development with iterations that are of definite duration, known as “sprints”.

The purpose of the sprint is to verify if users’ requirements and expectations have been touched or there are any recommendations to adopt some modifications, afterward gathering and translating these feedbacks as new tasks for developers. This is particularly valuable towards RESILOC for the reason that it allows news features to be reviewed by community users, thereby leading to changes to existing requirements or the discovery of new ones. Each sprint has been leveraged as a chance to remind the users about RESILOC scientific timeline, the sprint series, the RESILOC vision, and what RESILOC will make available relevant to the subsequent sprints.

Moreover, the sprint aims to keep informed users on the implementation progress, present the solutions, and discuss its usability through the developed UI. The community users have the opportunity for real-time interaction and provide feedback and suggestions for improvement. A sprint is an occasion to get approval from users to carry on and continue the implementation of the proposed solutions. The analysis of users’ feedback derived from sprints is also provided in this chapter.

Finally, technical descriptions given in this chapter will offer a better understanding of both the inventory frontend and backend developed.

4.1 Development Process

The inventory development task is resource intensive, requiring sufficient time for a completed final product to be provided in a timely manner and adapting to users’ requirements. A way of effective and efficient management is by implementing a development methodology that would promote disciplined project management to structure, plan, and control the process of developing a system within a reasonable timeframe and without extra effort and costs.

The development is started based on a set of designs and requirements described in the deliverable D2.7; however, further updates are necessary since we might realise that some features and functionalities in D2.7 are not obligatory or feasible for implementation because we come to new concepts in the process of development based on users’ feedback and developers’ emerging ideas.

During the development process, developers always categorise the tasks into different groups based on their complexity for the purpose of better management and a more organised workflow. We prioritise to develop baseline features or the ones having strong associations with other services (i.e., open data service, semantic layer, or survey tool) in the initial versions, while more complicated ones are implemented in subsequent releases.

To make the inventory development effective, some essential stages need to be taken into account as in Figure 1 including development, integration, and production stages. As sketched already, the general idea in the development of the inventory is to follow a circle of very small via the communication among developers from WNRI and IES and verification steps through different sprints with all community users. This enables the desired feedback loop, in which inputs from the RESILOC community users lead to new functionality and changes of existing features or the implementation of new functionalities according to users' needs.

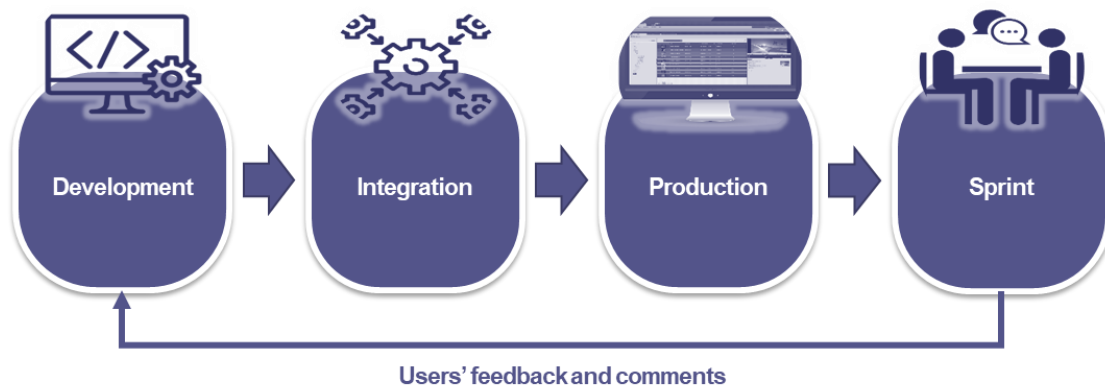


Figure 1. The RESILOC Inventory development process.

- **Development:** the development teams from WNRI and IES organised meetings at the beginning to identify, discuss, and analyse overall requirements of users to make adjustments and to ensure that system functions appropriately at the end. At this stage, the development teams also set standards and stick to it to avoid possible flaws later on. After guaranteeing that all components and functionalities were considered, developers begin the actual implementation process thoroughly. The backend in this stage uses WM3 as the development environment.
- **Integration:** the objective of the stage is to perform integration of the RESILOC Inventory frontend and backend. Before starting the integration, the development teams must ensure that both the frontend and backend are error-free and fit the designs. Each function on the UI is mapped precisely to developed endpoints given by the inventory backend. Communications between frontend and backend will be done using Hypertext Transfer Protocol (HTTP) using a REST architectural style. A number of assessments including security, accessibility, performance, stress, and compatibility tests are conducted to make sure that systems meet all the technical requirements with the components integrated.
- **Production:** this is where the community end-users access and actually use the latest version of the inventory after all of the updates and testing from the integration stage. In RESILOC, the time to release the inventory to production is several weeks before each sprint. This is to help developers have adequate time for final checks and verification to certify that the system is up and running and all functionalities operate fine with this latest and updated version before giving it to users during the sprints.
- **Sprint:** this stage aims at removing any gaps between community users and the development team. To plan and execute sprints productively and correctly, we articulate the sprint goals, inform these objectives to end-users, and provide them with well-prepared documentations. Each sprint is divided into two sessions separated by a week to allow users to interact and experience new components. The first session is

dedicated to the presentation and real-time usage of the latest developed features while the second one includes an in-depth exchange of ideas on the functionalities shown and tried by the users during previous days, aimed at investigating their impression, collecting comments and judgements to revise and/or validate what has been developed. Between the two sessions, the development team is always available to assist users if requested.

During all phases of the development lifecycle, API documentation and observability are developed to turn the lights on, see, and understand details and states of components of the inventory.

Documentation: To support developers of frontend and backend teams communicate and integrate as quickly as possible to move forward in the inventory development, we implement and release a live API documentation using Swagger Documentation¹² as depicted in Figure 2. This is a technical content deliverable containing detailed instructions about how to use and integrate with the inventory backend endpoints effectively. It is a concise reference manual comprising all the essential information required to work with the inventory backend APIs, with details about the functions, parameters, request schema, response types, queries, and more, supported by examples. Since this is a live documentation, it ensures up to date, accurate information, helps developers have a direct correlation on API adoption, and decreases the amount of time spent to understand how to work with backend endpoints.

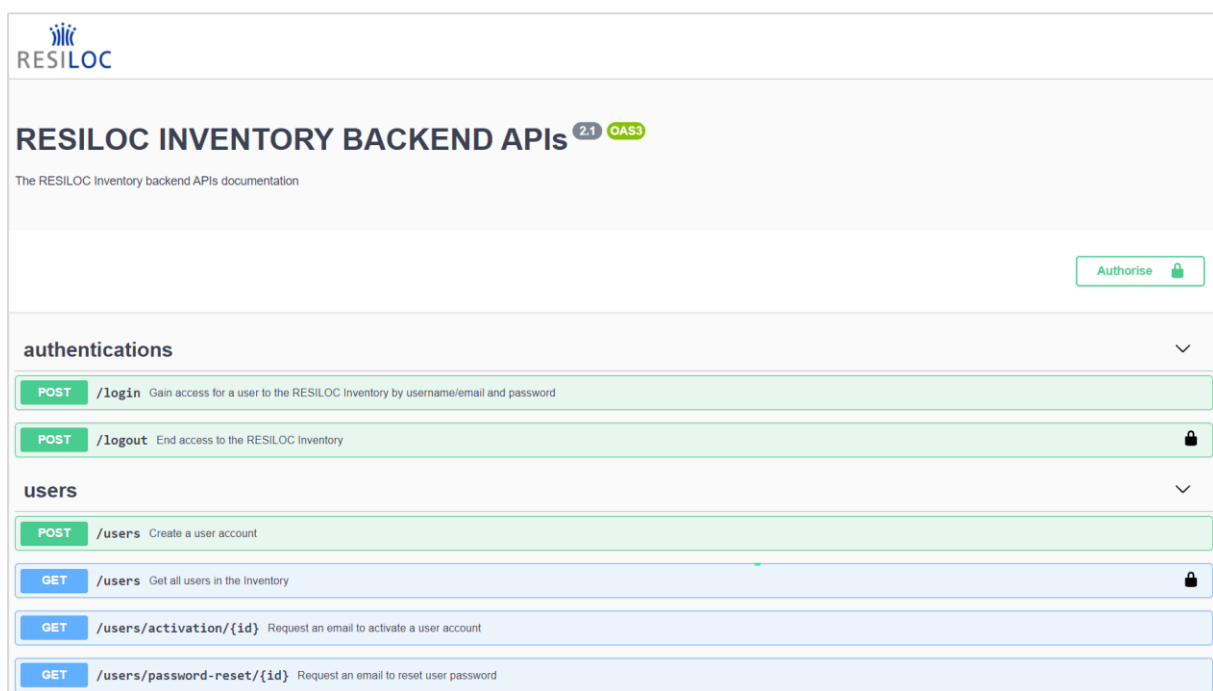


Figure 2. The RESILOC Inventory API documentation.

Observability: It is essential to have metrics and logs to observe, understand and act on the state of our system. We use swagger-stats¹³ as an API Observability to understand the state of workloads of the inventory backend as shown in Figure 3. This tool help to know the details about:

¹² <https://swagger.io/docs/>

¹³ <https://swaggerstats.io/>

- CPU and memory usage,
- number of requests and responses (total and by response class),
- processing time, content length and details of requests and responses,
- rates for requests and errors,
- statistics by request (baseline stats collected for each request method),
- timeline (baseline stats collected for each 1-minute interval to analyse trends),
- errors (number of responses for each error code and details for the last 100 errors),
- top 100 requests that took the longest time to handle.

Developers from the inventory backend side frequently check the tool dashboard to monitor the API performance, and in case there are any failures, they can promptly recognise any identify their root causes. Additionally, the observability also helps us derive insights into how community users make use of the inventory or making data-driven decisions about the system roadmap.

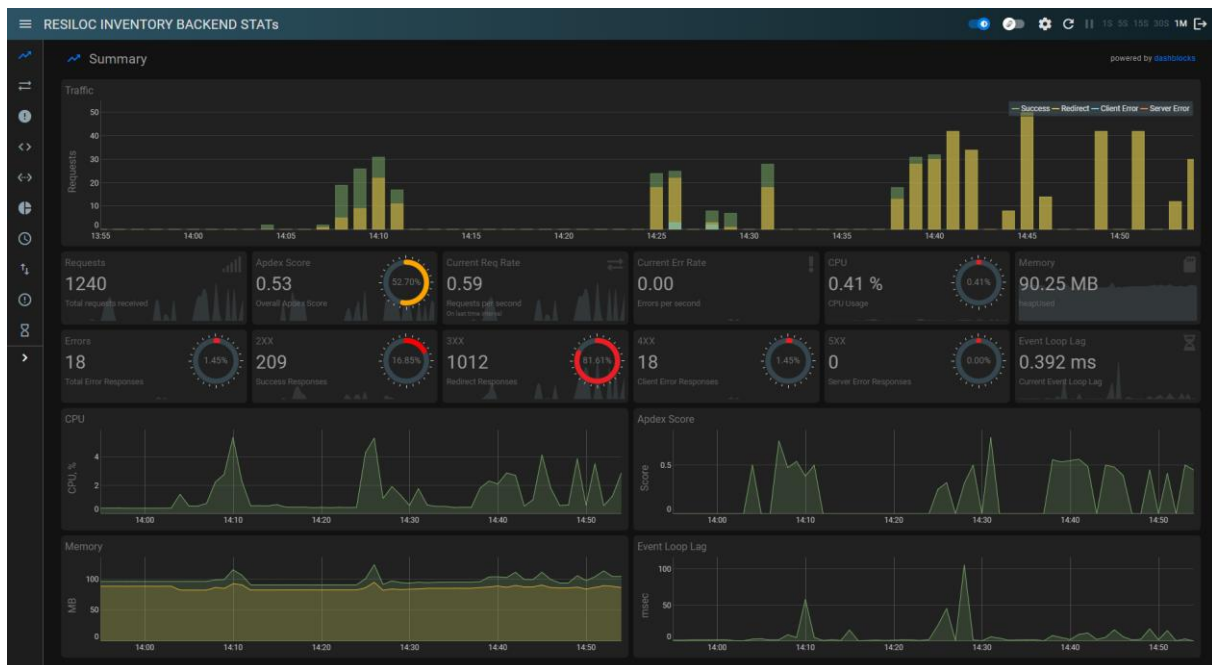


Figure 3. The RESILOC Inventory backend observability.

4.2 User Feedback Analysis

The development of the RESILOC Inventory was characterised by an AGILE implementation process including interaction phases with the other consortium partners as well as with the end-user representative communities. These phases, called Sprints, were conducted with the aim of verifying whether the users' needs, and expectations were met, or they could be implemented by the developers in a more successful way. Specifically, the RESILOC Inventory was presented during the 1st Sprint, introducing a first mock-up and describing its earliest functionalities. From it the RESILOC Inventory was further developed and represented the main focus of the 2nd Sprint, and partially reintroduced in the 3rd Sprint. During these Sprints, the inventory was presented, users were updated on the progress of the implementation and its usability was discussed in detail via the developed user interface (UI). In addition, the users were allowed to interact with the inventory in real time (interactive session between developers,

researchers, and users), in order to give feedback and suggestions on the functionalities offered and exchange thoughts on their use and effectiveness; such information was useful for improvement but above all to obtain the users' approval to continue the implementation.

The floor was given to community representatives to raise questions or comments and to share their experience on the inventory usage; the highlights and key points for the improvement of the RESILOC Inventory, grouped by topics, are presented below.

On usability aspects, researchers and community representatives gave a positive response, expressing appreciation for the current version, reporting a simple and user-friendly environment to enter the data required. However, they reserve the opportunity to raise some remarks once intensive use of the system begins, so that some features could be identified to be optimised in order to improve usability.

Accessibility, role management and role assignment were discussed extensively; in particular, the conversations involved the way in which roles are managed and assigned to inventory users, focusing on the possibility of having multiple roles assignable to each user and the minimum number of roles required to have a community capable of operating the inventory comprehensively. The accepted solution envisages that each user has a unique account, whereby he/she can login to the inventory and have access to different communities; in each community, a user may have different roles properly assigned according to his/her expertise and responsibilities, wherein each role provides different capabilities to interact with the system. In addition, every community needs to have users holding the role of local manager, resilience expert and community admin, even if assigned to the same person and thus to a single user. It was also strongly suggested to make visible on the screen the roles played by the user for the community selected; the development team proposed a possible solution, consisting in displaying the roles played by the user next to the name of the community he/she is operating into.

The usability of the application on mobile devices was another topic raised by the representatives of the communities; however, the layouts used are responsive (i.e., the contents are automatically organised and modelled) so all pages can be used on mobile devices as they are automatically adapted (some specific pages may be more difficult to use due to their higher complexity).

On the possible corrections/changes to be done, users pointed out a very short autosave time delay (introduced to avoid data loss in case of connection drop) when entering certain values, e.g., min and max target, causing trouble in filling in the form. In reply, the delay time of the autosave function has already been modified, extending it by an appropriate amount of time to make the required operation easier and more fluent. Regarding the roles management and assignment, it was suggested to make the users' roles management layout as compact as possible, aiming to facilitate the users' identification in case there are several registered.

The execution of the Sprints allowed not only the collection of productive feedback and suggestions for the inventory improvement, but also stimulated the users to identify beneficial/useful features. During the interactive phase, the users came up with a requirement to include a filter feature in the proxy, indicator, and scenario lists layout; so that they could be easily retrieved by typing their names into a search box. Likewise, introducing the import and management functionalities for proxies, indicators and scenarios, some users expressed the interest in having a feature that allows the duplication of these elements. In addition to what has already been stated, users manifested the necessity of a widespread use of tooltips, transversal to all the sections composing the inventory; their implementation is aimed at favouring an exhaustive understanding of the items and terminologies presented on screen.

In addition to what has already been stated, users manifested the necessity of a widespread use of tooltips, transversal to all the sections composing the inventory; their implementation is aimed at favouring an exhaustive understanding of the items and terminologies presented on screen. The implementation of a notification system within the inventory was also suggested, in order to notify designated users (e.g., Community Admins) on the execution of certain operations involving their community (e.g., if a user follows his community).

The requests and suggestions collected have been evaluated and prioritised; some of the previously mentioned functionalities have already been developed and are currently available in the inventory, such as some tooltips and the filtering and duplication features, while the implementation of the notification system has not been released yet.

During the presentation of the latest implementations (Sprint #4), which included the language settings component, the new sorting and searching functionalities, and the duplication functionality; the users showed appreciation for the accomplishment of their previous requests, stating also that the tooltips integrated and deployed are easy to understand. The users provided comments on the utility to have a what-if tool, suggesting some functionality in this respect, like a sort of versioning tool for the proxies.

According to the discussions on the inventory emerged during the sprints, users appreciated and approved what has been developed in the inventory so far, states that the inventory does not have components that needs to be redone or replaced and says that the system is flexible for further small adaptations.

4.3 Backend Service and Functions

The RESILOC Inventory backend provides functionalities via RESTful APIs with a JSON payload. Designing and implementing accurate, fast-response, long-lasting APIs is considered a critical task of the backend service since it is strongly connected with other services in the RESILOC Platform.

With this in mind, we adopt API-First design (Dudjak and Martinović, 2020) as our fundamental engineering principle to ensure that all the information inside the inventory is accessible and manageable through the APIs. To accomplish this, we design the API contract before implementing the web service and develop the frontend and backend along with one another based on the API contract. The live and usable documentation mentioned in section 4.1 plays a major role since it helps developers, who have not been involved in APIs creation, understand and explore APIs easily. Currently, the backend service is up and running in both development and production mode in VM3 and VM11, respectively.

4.3.1 Backend Endpoints

Based on resources initially designed in D2.7, we need to determine what actions apply to them and how those would map to our APIs. The below subsections list all available endpoints offered by the RESILOC Inventory backend, along with the description for each one. These APIs are used to interact with the inventory UI and other services alike (e.g., semantic layer, open data middleware, survey tool). Details of CRUD operations according to each user role were mentioned in

Table 2.

For the sake of clarity and coherence, the defined endpoints are classified into 13 groups (i.e., categories). The endpoints with the padlock icons imply access to these methods is secured; therefore, we must provide a valid JWT to unlock. On the other hand, no padlock indicates free/public access.

4.3.1.1 Authentication

The authentication APIs allow managing all aspects of user identity. It offers endpoints so that users can log in and log out as shown in Figure 4.


POST	/login	Gain access for a user to the RESILOC Inventory by username/email and password	
POST	/logout	End access to the RESILOC Inventory	

Figure 4. Authentication APIs.

4.3.1.2 User

The user APIs provides operations to manage users in the inventory as shown in Figure 5.








POST	/users	Create a user account	
GET	/users	Get all users in the Inventory	
GET	/users/activation/{id}	Request an email to activate a user account	
GET	/users/password-reset/{id}	Request an email to reset user password	
GET	/users/{username}	Get information of a user	
PUT	/users/{username}	Update user information	
DELETE	/users/{username}	Delete a user account	
PUT	/users/activation/{userActivationToken}	Activate a user account	
PUT	/users/password-reset/{passwordResetToken}	Reset user password	
PUT	/users/password/{username}	Change user password	
PUT	/users/roles/{username}	Assign roles for a user	
PUT	/users/delete/roles/{username}	Remove roles of a user	

Figure 5. User APIs.

4.3.1.3 User Role

The user role APIs provides operations to get roles of registered users as shown in Figure 6.





GET	/user-roles	Get defined user roles in the Inventory	
GET	/user-roles/myself	Get roles of a user by token and flid	
GET	/user-roles/{username}	Get roles of a user by communities	
GET	/user-roles/{username}/community/{communityId}	Get roles of a user by a community	

Figure 6. User role APIs.

4.3.1.4 Community

The community APIs provide operations to generate communities, manage communities' profiles, and organise users of these communities, as shown in Figure 7.

POST	/communities	Create a community	🔒
GET	/communities	Get all communities in the Inventory	🔒
GET	/communities/follow/{username}	Get available communities for user to follow	🔒
GET	/communities/followed/{username}	Get followed communities of a user	🔒
GET	/communities/{id}	Get information of a community	🔒
PUT	/communities/{id}	Update community information	🔒
DELETE	/communities/{id}	Delete a community	🔒
GET	/communities/languages/{id}	Get defined languages of a community	🔒
GET	/communities/users/{id}	Get users of a community	🔒
PUT	/communities/follow/{id}	Let a user follow a community	🔒
PUT	/communities/unfollow/{id}	Let a user unfollow a community	🔒

Figure 7. Community APIs.

4.3.1.5 RESILOC Scenario

The RESILOC scenario APIs enable RESILOC admins to perform creating, getting, and updating on RESILOC scenarios, as shown in Figure 8. Authorised community users can also retrieve RESILOC scenarios.

POST	/resilloc-scenarios	Create a RESILOC scenario	🔒
GET	/resilloc-scenarios	Get all RESILOC scenarios in the Inventory	🔒
POST	/resilloc-scenarios/duplicate	Create a RESILOC scenario by duplicating from another RESILOC scenario	🔒
GET	/resilloc-scenarios/resilloc-indicator/{resillocIndicatorId}	Get linked RESILOC scenarios of a RESILOC indicator	🔒
GET	/resilloc-scenarios/visible	Get all RESILOC scenarios that are visible towards communities (i.e., not draft) in the Inventory	🔒
GET	/resilloc-scenarios/visible/tag/{tagName}	Get all RESILOC scenarios that are visible towards communities (i.e., not draft) in the Inventory by tag name	🔒
GET	/resilloc-scenarios/draft	Get all draft RESILOC scenarios in the Inventory	🔒
GET	/resilloc-scenarios/draft/tag/{tagName}	Get all draft RESILOC scenarios in the Inventory by tag name	🔒
GET	/resilloc-scenarios/tag/{tagName}	Get all RESILOC scenarios in the Inventory by tag name	🔒
GET	/resilloc-scenarios/tags	Get all tags of RESILOC scenarios in the Inventory	🔒
GET	/resilloc-scenarios/{id}	Get information of a RESILOC scenario	🔒
PUT	/resilloc-scenarios/{id}	Update a RESILOC scenario	🔒
DELETE	/resilloc-scenarios/{id}	Delete a RESILOC scenario	🔒
PUT	/resilloc-scenarios/resilloc-indicators/{id}	Associate RESILOC indicators to a RESILOC scenario	🔒
PUT	/resilloc-scenarios/scenario-proxy/{id}	Update RESILOC scenario associated values	🔒
PUT	/resilloc-scenarios/{id}/translation/{language}	Translate a RESILOC scenario	🔒

Figure 8. RESILOC scenario APIs.

4.3.1.6 RESILOC Indicator

The RESILOC indicator APIs enable RESILOC admins to perform creating, getting, and updating on RESILOC indicators as well as associate them to RESILOC scenarios, as shown in Figure 9. Authorised community users can also retrieve RESILOC indicators.

POST	/resilloc-indicators	Create a RESILOC indicator	🔒
GET	/resilloc-indicators	Get all RESILOC indicators in the Inventory	🔒
POST	/resilloc-indicators/duplicate	Create a RESILOC indicator by duplicating from another RESILOC indicator	🔒
GET	/resilloc-indicators/resilloc-proxy/{resillocProxyId}	Get linked RESILOC indicators of a RESILOC proxy	🔒
GET	/resilloc-indicators/visible	Get all RESILOC indicators that are visible towards communities (i.e., not draft) in the Inventory	🔒
GET	/resilloc-indicators/visible/tag/{tagName}	Get all RESILOC indicators that are visible towards communities (i.e., not draft) in the Inventory by tag name	🔒
GET	/resilloc-indicators/draft	Get all draft RESILOC indicators in the Inventory	🔒
GET	/resilloc-indicators/draft/tag/{tagName}	Get all draft RESILOC indicators in the Inventory by tag name	🔒
GET	/resilloc-indicators/tag/{tagName}	Get all RESILOC indicators in the Inventory by tag name	🔒
GET	/resilloc-indicators/tags	Get all tags of RESILOC indicators in the Inventory	🔒
GET	/resilloc-indicators/{id}	Get information of a RESILOC indicator	🔒
PUT	/resilloc-indicators/{id}	Update a RESILOC indicator	🔒
DELETE	/resilloc-indicators/{id}	Delete a RESILOC indicator	🔒
PUT	/resilloc-indicators/resilloc-proxies/{id}	Associate RESILOC proxies to a RESILOC indicator	🔒
PUT	/resilloc-indicators/unassign/{id}	Unassign a RESILOC indicator from all RESILOC scenarios	🔒
PUT	/resilloc-indicators/{id}/translation/{language}	Translate a RESILOC indicator	🔒

Figure 9. RESILOC indicator APIs.

4.3.1.7 RESILOC Proxy

The RESILOC proxy APIs enable RESILOC admins to perform creating, getting, and updating on RESILOC proxies as well as associate them to RESILOC indicators, as shown in Figure 10. Authorised community users can also retrieve RESILOC proxies.

POST	/resilloc-proxies	Create a RESILOC proxy	🔒
GET	/resilloc-proxies	Get all RESILOC proxies in the Inventory	🔒
POST	/resilloc-proxies/duplicate	Create a RESILOC proxy by duplicating from another RESILOC proxy	🔒
GET	/resilloc-proxies/visible	Get all RESILOC proxies that are visible towards communities (i.e., not draft) in the Inventory	🔒
GET	/resilloc-proxies/visible/tag/{tagName}	Get all RESILOC proxies that are visible towards communities (i.e., not draft) in the Inventory by tag name	🔒
GET	/resilloc-proxies/draft	Get all draft RESILOC proxies in the Inventory	🔒
GET	/resilloc-proxies/draft/tag/{tagName}	Get all draft RESILOC proxies in the Inventory by tag name	🔒
GET	/resilloc-proxies/tag/{tagName}	Get all RESILOC proxies in the Inventory by tag name	🔒
GET	/resilloc-proxies/tags	Get all tags of RESILOC proxies in the Inventory	🔒
GET	/resilloc-proxies/{id}	Get information of a RESILOC proxy	🔒
PUT	/resilloc-proxies/{id}	Update a RESILOC proxy	🔒
DELETE	/resilloc-proxies/{id}	Delete a RESILOC proxy	🔒
PUT	/resilloc-proxies/unassign/{id}	Unassigned a RESILOC proxy from all RESILOC indicators	🔒
PUT	/resilloc-proxies/{id}/translation/{language}	Tranlate a RESILOC proxy	🔒

Figure 10. RESILOC proxy APIs.

4.3.1.8 Community Scenario

The community scenario APIs allow RESILOC admins and community users to create, read, update, and delete their own community scenarios as shown in Figure 11.

POST	/scenarios/create/community/{communityId}	Create a new scenario for a community	🔒
POST	/scenarios/inherit/community/{communityId}	Create a scenario for a community by inheriting from a resilloc scenario	🔒
POST	/scenarios/duplicate/community/{communityId}	Create a scenario for a community by duplicating from another scenario	🔒
GET	/scenarios	Get all community scenarios in the Inventory	🔒
GET	/scenarios/indicator/{indicatorId}	Get linked scenarios of a community indicator	🔒
GET	/scenarios/user/{username}	Get scenarios that are visible towards a user by his/her communities	🔒
GET	/scenarios/community/{communityId}	Get scenarios of a community	🔒
GET	/scenarios/{id}	Get information of a community scenario	🔒
PUT	/scenarios/{id}	Update a community scenario	🔒
DELETE	/scenarios/{id}	Delete a community scenario	🔒
PUT	/scenarios/indicators/{id}	Associate community indicators to a community scenario	🔒
PUT	/scenarios/{id}/translation/{language}	Tranlate a community scenario	🔒

Figure 11. Community scenario APIs.

4.3.1.9 Community Indicator

The community indicator APIs allow RESILOC admins and community users to create, read, update, and delete their own community indicators as well as associate them to community scenarios, as shown in Figure 12.

POST	/indicators/create/community/{communityId}	Create a new indicator for a community	🔒
POST	/indicators/inherit/community/{communityId}	Create an indicator for a community by inheriting from a resiloc indicator	🔒
POST	/indicators/duplicate/community/{communityId}	Create an indicator for a community by duplicating from another indicator	🔒
GET	/indicators	Get all community indicators in the Inventory	🔒
GET	/indicators/static-proxy/{staticProxyId}	Get linked indicators of a community proxy	🔒
GET	/indicators/user/{username}	Get indicators that are visible towards a user by his/her communities	🔒
GET	/indicators/community/{communityId}	Get indicators of a community	🔒
GET	/indicators/{id}	Get information of a community indicator	🔒
PUT	/indicators/{id}	Update a community indicator	🔒
DELETE	/indicators/{id}	Delete a community indicator	🔒
PUT	/indicators/static-proxies/{id}	Associate community proxies to a community indicator	🔒
PUT	/indicators/unassign/{id}	Unassign a community indicator from all community scenarios	🔒
PUT	/indicators/{id}/translation/{language}	Tranlate a community indicator	🔒

Figure 12. Community indicator APIs.

4.3.1.10 Community Proxy

The community indicator APIs allow RESILOC admins and community users to create, read, update, and delete their own community proxies, associate them to community indicators as well as input snapshot-related values, as shown in Figure 13.

POST	/static-proxies/create/community/{communityId}	Create a new proxy for a community	🔒
POST	/static-proxies/inherit/community/{communityId}	Create a proxy for a community by inheriting from a resiloc proxy	🔒
POST	/static-proxies/duplicate/community/{communityId}	Create a proxy for a community by duplicating from another proxy	🔒
GET	/static-proxies	Get all community proxies in the Inventory	🔒
GET	/static-proxies/user/{username}	Get proxies that are visible towards a user by his/her communities	🔒
GET	/static-proxies/community/{communityId}	Get proxies of a community	🔒
GET	/static-proxies/scenario/{scenarioId}	Get community proxies of a scenario	🔒
GET	/static-proxies/snapshot/{snapshotId}/indicator/{indicatorId}	Get community proxies by an indicator of a snapshot	🔒
GET	/static-proxies/snapshot/{snapshotId}	Get community proxies of a snapshot	🔒
GET	/static-proxies/{id}	Get information of a community proxy	🔒
PUT	/static-proxies/{id}	Update a community proxy	🔒
DELETE	/static-proxies/{id}	Delete a community proxy	🔒
PUT	/static-proxies/min-max-target/scenario/{scenarioId}	Update min-max targets of community proxies of a scenario	🔒
PUT	/static-proxies/relevance-direction/snapshot/{snapshotId}/indicator/{indicatorId}	Update relevance, direction of community proxies of an indicator of a snapshot	🔒
PUT	/static-proxies/value-metadata/snapshot/{snapshotId}	Update value and metadata of community proxies of a snapshot	🔒
PUT	/static-proxies/{id}/translation/{language}	Tranlate a community proxy	🔒

Figure 13. Community proxy APIs.

4.3.1.11 Snapshot

The community indicator APIs allow RESILOC admins and community users to create, read, update, submit, and delete their own snapshots, as shown in Figure 14.

POST	/snapshots/community/{communityId}	Create a snapshot for a community	🔒
GET	/snapshots/community/{communityId}	Get snapshots of a community grouped by community scenarios	🔒
GET	/snapshots	Get all snapshots in the Inventory	🔒
GET	/snapshots/indicator/{indicatorId}	Get linked snapshots of a community indicator	🔒
GET	/snapshots/{id}/calculation	Get information of a snapshot to calculate its resilience value	🔒
PUT	/snapshots/{id}/calculation	Update calculated values for community indicators of a snapshot	🔒
GET	/snapshots/user/{username}	Get snapshots that are visible towards a user by his/her communities	🔒
GET	/snapshots/scenario/{scenarioId}	Get snapshots of a community scenario	🔒
GET	/snapshots/community/{communityId}/onhold	Get on-hold snapshots of a community grouped by community scenarios	🔒
GET	/snapshots/scenario/{scenarioId}/onhold	Get on-hold snapshots of a community scenario	🔒
GET	/snapshots/community/{communityId}/submitted	Get submitted snapshots of a community grouped by community scenarios	🔒
GET	/snapshots/scenario/{scenarioId}/submitted	Get submitted snapshots of a community scenario	🔒
GET	/snapshots/{id}	Get information of a snapshot	🔒
PUT	/snapshots/{id}	Update a snapshot	🔒
DELETE	/snapshots/{id}	Delete a snapshot	🔒
PUT	/snapshots/submit/{id}	Submit a snapshot	🔒
PUT	/snapshots/{id}/translation/{language}	Translate a snapshot	🔒

Figure 14. Snapshot APIs.

4.3.1.12 Timeline

The timeline APIs provide the ability to get timeline of community as depicted in Figure 15.

GET	/timelines/default/community/{communityId}/scenario/{scenarioId}	Get timeline of a community by community scenarios	🔒
-----	--	--	---

Figure 15. Timeline APIs.

4.3.1.13 Open Data

The open data APIs provide GET operations to return all the data in the inventory that are set as *public* visibility, as shown in Figure 16.

GET	/open-data/static-proxies	Get all public community proxies in the Inventory
GET	/open-data/indicators	Get all public community indicators in the Inventory
GET	/open-data/scenarios	Get all public community scenarios in the Inventory
GET	/open-data/snapshots	Get all public community snapshots in the Inventory
GET	/open-data/resiloc-proxies	Get all public RESILOC proxies in the Inventory
GET	/open-data/resiloc-indicators	Get all public RESILOC indicators in the Inventory
GET	/open-data/resiloc-scenarios	Get all public RESILOC scenarios in the Inventory

Figure 16. Open data APIs.

4.3.2 Response Status Code

All the inventory endpoints' responses come along with a numeric status code (W3C, 1992) to HTTP requests as shown in Table 3. It is sent back by the backend to specify how the server handled the transaction, whether a particular request has been successfully completed or failed with errors.

Table 3. RESILOC Inventory backend response status codes.

Type	Code Syntax	Code	Description
Success	2XX	200	OK
		201	Created
		202	Accepted
Redirection	3XX	304	Not Modified
Error	4XX	400	Bad request
		401	Unauthorised
		403	Forbidden
		404	Not found
	5XX	500	Internal Error
		501	Not implemented

The inventory backend is designed to return different status codes according to an endpoint, context, and action. This way, if a request results in an error, the frontend can promptly get insight into what went wrong. Figure 17 shows the list of possible HTTP status codes for the /user/{username}.

Code	Description
200	OK
400	Value validation failed
401	Unauthorised
403	Forbidden resource
404	{ "message": "Username {username} does not exist" }

Figure 17. HTTP status codes of a user GET endpoint.

4.3.3 Utilisation Functions

One essential requirement of the inventory backend is to ensure that endpoints would allow for very flexible querying. This would mean including multilingual, paging, sorting, and filtering functionalities.

4.3.3.1 Multilingual Function

Supporting multilingual function is a special request that we obtained during the sprints. This is the process of designing endpoints so that users can (1) translate all RESILOC and community elements, and afterward, (2) the backend can automatically build results in various languages according to users' preferences. For the time being, the inventory backend supports four different languages that are English (en), Italian (it), Bulgarian (bg), and Greek (el), as depicted in Figure 18. However, it is feasible to extend the system for all other languages.

Our approach uses an explicit request parameter based upon a language code in the URL to determine a language, e.g., &language=en. Translation features are available on both admin and community sides. It is worth noting that items are not translated automatically. Each item created in the community default language can have a corresponding field for a selected fallback language so that community users can manually translate via PUT endpoints. After translating, the items will automatically be displayed in the users' preferred language.

language * required
string
(query)

The language code of resiloc proxies to be retrieved
Available values : bg, en, el, it
Example : en

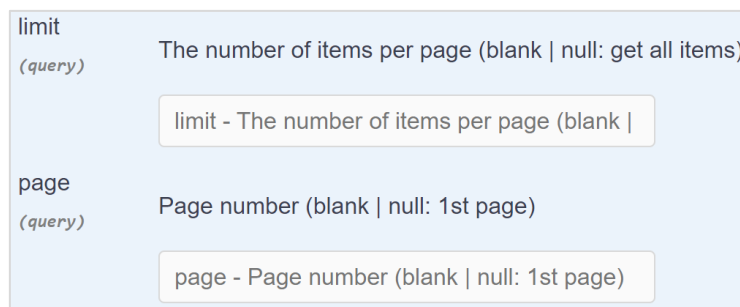
en

Figure 18. Multiple languages supported for translation purpose.

4.3.3.2 Paging Function

To avoid displaying hundreds or thousands of data (e.g., a bunch of community proxies, indicators, scenarios, etc.) in the frontend application, all the GET endpoints paginate the results into discrete pages to make sure responses are easier to navigate and manage as well as give a better user experience.

Limit and page fields are used for pagination as shown in Figure 19. We have defined default values for limit and page on results, but it is always recommended to set these parameters explicitly to ensure how many results are returned. Do not rely on the defaults since the response might not be what we expect.

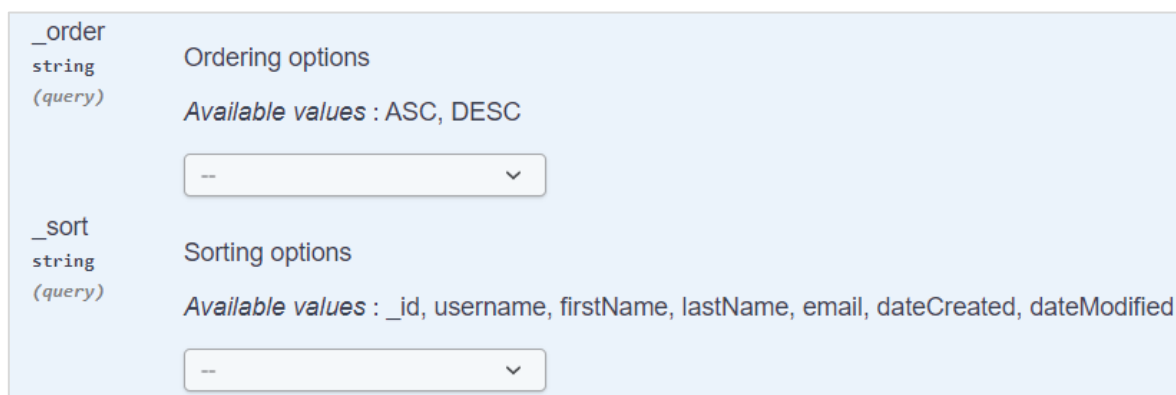


limit (query)	The number of items per page (blank null: get all items)
limit - The number of items per page (blank null: get all items)	
page (query)	Page number (blank null: 1st page)
page - Page number (blank null: 1st page)	

Figure 19. Pagination parameters of GET endpoints.

4.3.3.3 Sorting Function

Sorting is an important mechanism for any GET endpoints that return a lot of data by placing the items in order. If we want to get a list of users, the APIs can provide options to sort by `_id`, username, first name, last name, email, date created, or last date modified by ascending or descending order as shown in Figure 20. By default, we sort items by date modified in descending order.



_order string (query)	Ordering options Available values : ASC, DESC
--	
_sort string (query)	Sorting options Available values : _id, username, firstName, lastName, email, dateCreated, dateModified
--	

Figure 20. Ordering and sorting parameters of user GET endpoints.

4.3.3.4 Filtering Function

In addition to pagination and sorting, we also apply filters to all the GET endpoints to limit results by specific criteria as shown in Figure 21. Values given to a filtering query should be a valid format following LHS brackets, which is an effective way to encode operators is the use of square brackets on the key name. With LHS brackets we can have as many operators as we need such as [lte], [gte], etc. It is also possible to combine as many filters as we want.

For example, the request GET /users?filter[firstName]=lionel&filter[dateCreated][gte]=2020-01-01 will return all the users where first name includes “lionel” and account creation date is after “01.01.2020”. We choose LHS brackets since they are easy to use and offer better flexibility in the filter value for clients, even though it is a little harder and complicated to parse on the server-side.

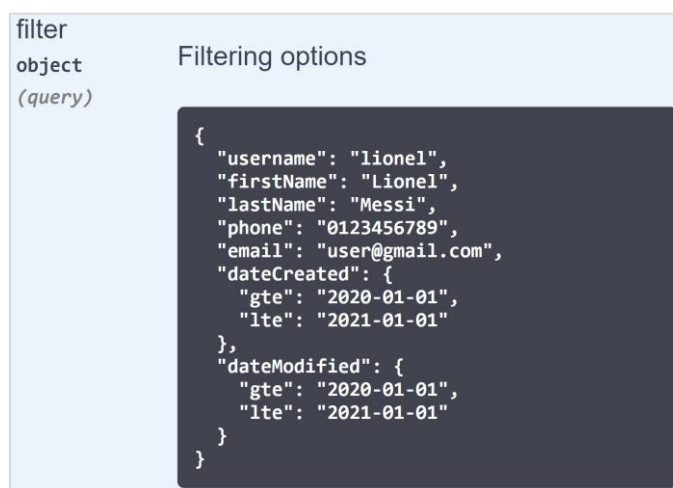


Figure 21. Filtering parameters of user GET endpoints.

Combining utilisation functions all together:

According to functionalities mentioned in this section, assuming that we want to get all the community indicators in the inventory that satisfy the below criteria:

- They have a “Community Structure and Agency” dimension.
- They have the text “building” in their names.
- They were created between 01.01.2020 and 01.01.2021.
- They were modified between 01.11.2021 and 15.11.2021.
- They have a “settlement” tag.
- The results are sorted by name in ascending order.
- The first 10 results will be returned.
- The community indicators’ information is displayed in Italian.

In this circumstance, the request should be precisely constructed as in Figure 22 to obtain expected results:

```
/indicators?filter[dimension]=community_structure_and_agency&filter[name]=
building&filter[dateCreated][gte]=2020-01-01&filter[dateCreated][lte]=2021-01-
01&filter[dateModified][gte]=2021-11-01&filter[dateModified][lte]=2021-11-15 &
filter[tag]= settlement&_order=ASC&_sort=name&limit=10&page=1&language=it
```

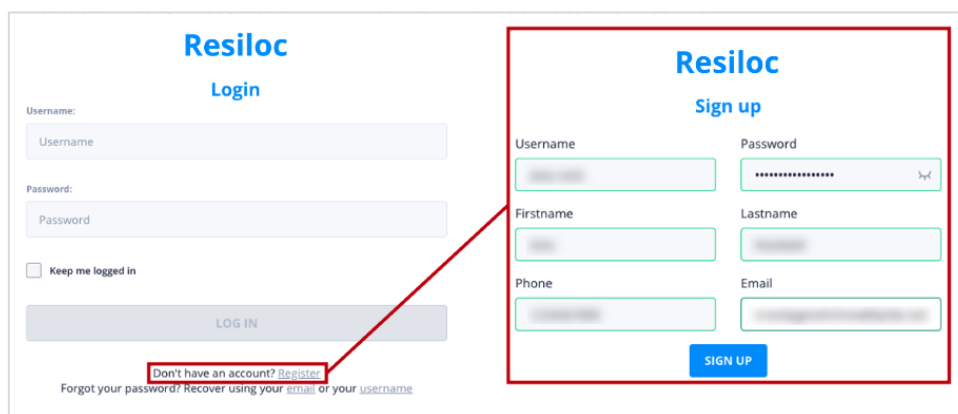
Figure 22. Sample GET request to get community indicators according to different criteria.

4.4 User Interface

The RESILOC Inventory Web Application is the primary vehicle of interaction between the RESILOC main actors and all the functionality it exposes. Each component of the interface connects "the user" with a specific subsystem, interacting mainly with a data processing and analysis module. The interface components use the backend service for data retrieval (e.g., representation of evaluation results) and for information input (e.g., creation of proxies, indicators or scenarios). In order to meet the functional requirements gathered from end users, all interface components are designed to maintain a high level of usability, security and efficiency.

The RESILOC Inventory Web Application can be accessed via web browser, either from a desktop or from a mobile device, allowing end-users to access the offered functionalities and services. Using the web interface, users are able to easily perform edits on generic community information, utilise the functionalities needed to define and manage community proxies, indicators and scenarios, as well as access all the functionalities necessary to manage the information and enter data in a structured way into the inventory DB. Such approach is convenient as it adequately prepares the data for the resilience assessment process performed by the Cloud Platform Subsystem; this system mainly interacts with the inventory backend through HTTP requests, calling the exposed REST API.

The interface offers a secure access form for users, obscuring fields containing sensible data (e.g., passwords). To log in, the user must have an account and a pair of credentials: username and password; if the user is not registered, the interface offers a dedicated environment to fill in the registration form as in Figure 23.



The screenshot displays two side-by-side forms for user authentication. The left form, titled 'Resiloc Login', includes fields for 'Username' and 'Password', a 'Keep me logged in' checkbox, and a 'LOG IN' button. Below the login form is a link that reads 'Don't have an account? Register'. The right form, titled 'Resiloc Sign up', is enclosed in a red border and includes fields for 'Username', 'Password', 'Firstname', 'Lastname', 'Phone', and 'Email', along with a 'SIGN UP' button. A red arrow originates from the 'Register' link in the login form and points to the 'Sign up' form.

Figure 23. Sign in and sign-up module.

Once logged in, the user interface of the RESILOC Inventory Web Application appears divided into 3 main areas as shown in Figure 24.

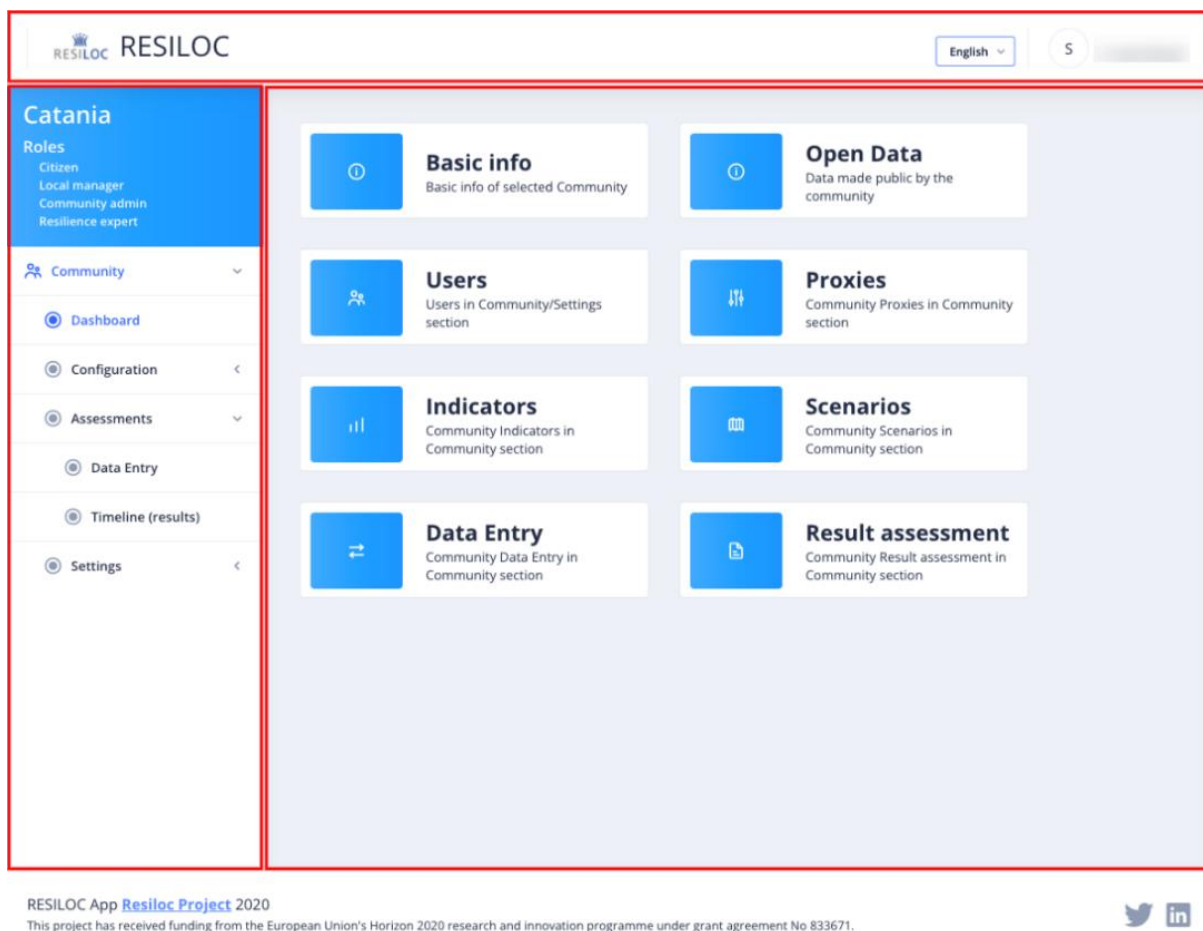


Figure 24. RESILOC Inventory dashboard.

On top of the page the RESILOC project logo and the User menu are displayed; this menu is identified by the username chosen by the user and can be opened by clicking on it. From the User menu it is possible to access buttons (Figure 25) that allow the user to:

- Update the user's current information ("Profile" button),
- Select the community the user wants to work on ("Select community" button),
- Leave the inventory ("Logout" button).

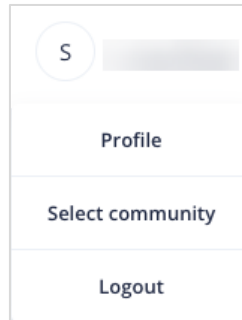


Figure 25. User menu.

On the left side of the page, the interface shows the name of the community in which the user is currently operating; the roles played by the user are displayed below. In addition, a dropdown sidebar menu is provided, containing different sections based on the roles played; each section is represented by a dropdown menu that reveals different sub-items, enabling access to each of them (the sidebar menu is responsive and automatically hidden to suit small windows or screens). The visibility of the different sections housed in the sidebar menu (and their usage) is based on the role assigned to the user for that specific community as depicted in Figure 26 and Figure 27. A user who plays in distinct communities' different roles, will see diverse sections in the sidebar menu, according to the role assigned.

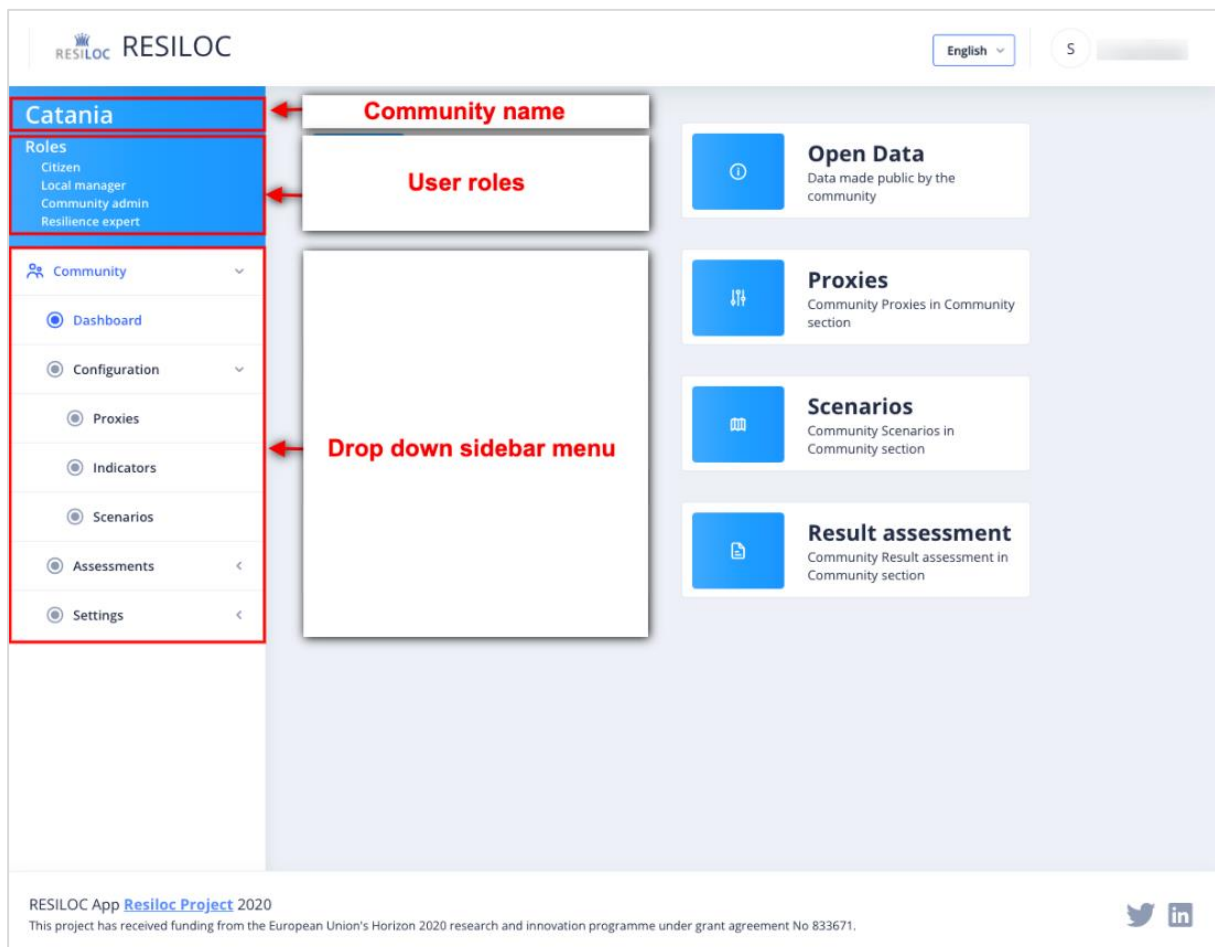


Figure 26. Sidebar layout and structure of community side.

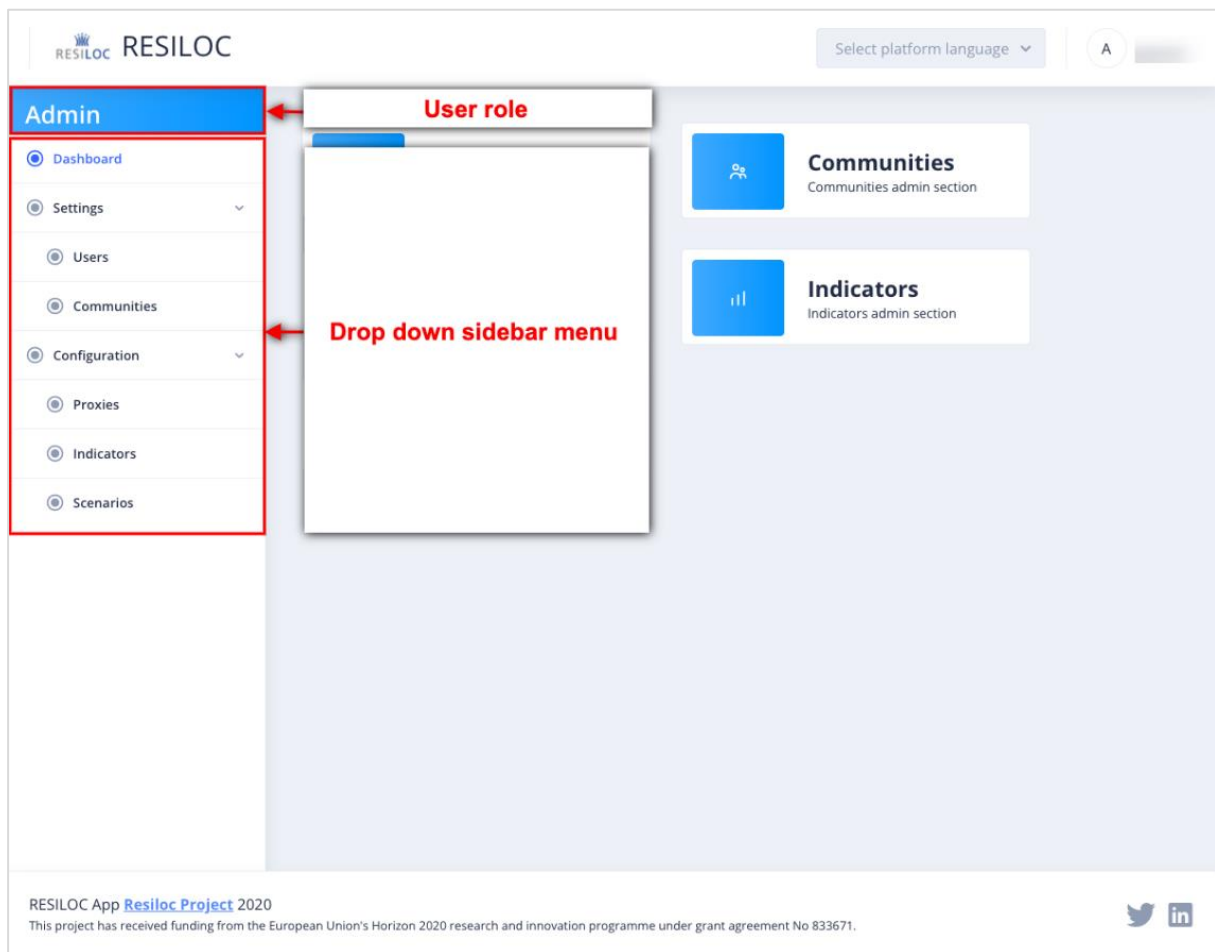


Figure 27. Sidebar layout and structure of RESILOC admin side.

The main body of the page is designed to display the panels linked to the menu items accessible from the sidebar. After the selection of the community to work on, the system displays in the main body of the page a dashboard, containing all the sections available for the user (in line with what is visible in the sidebar).

By selecting one of the available menu items, the system returns in the main body of the page the panel containing all the functionalities associated with the selected item. Figure 28 shows, as an example, the view returned by selecting the "Indicators" menu item in the Configuration section.

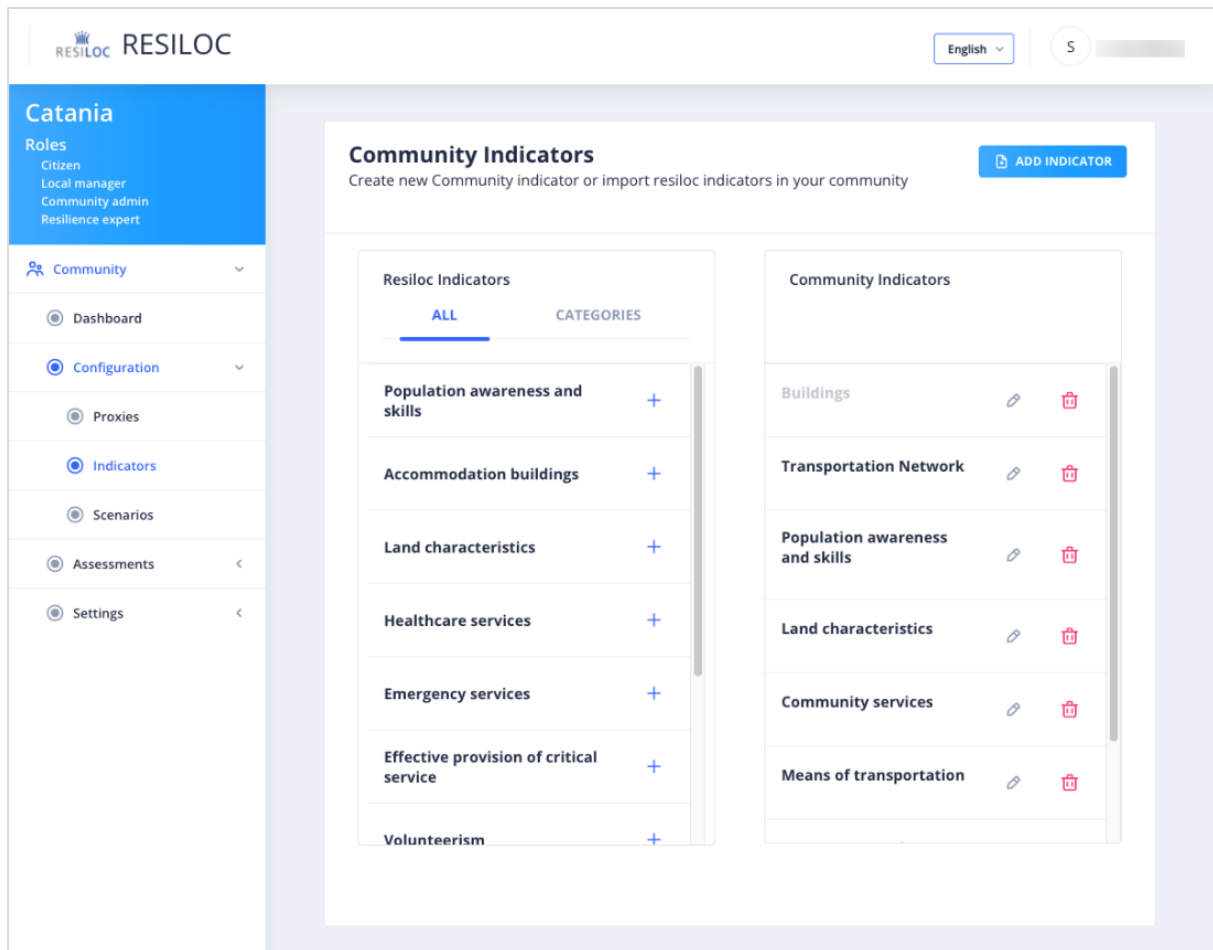


Figure 28. The main body of the page showing the panel containing the functionalities associated with the selected item (the 'Indicators' menu).

The user interface of the RESILOC Inventory is the result of a continuous user-oriented development process aimed at improving its usability; promptness and easy interaction with the system are the most appreciated features by the end-user and the ones that most affect the design of the interface. Each functionality of the interface component has to be presented in a well-defined and understandable way for the user; accordingly, the RESILOC Inventory interface is characterised by buttons with clear labels indicating the assigned function and by tooltips providing clarification where these labels do not apply as shown in Figure 29.

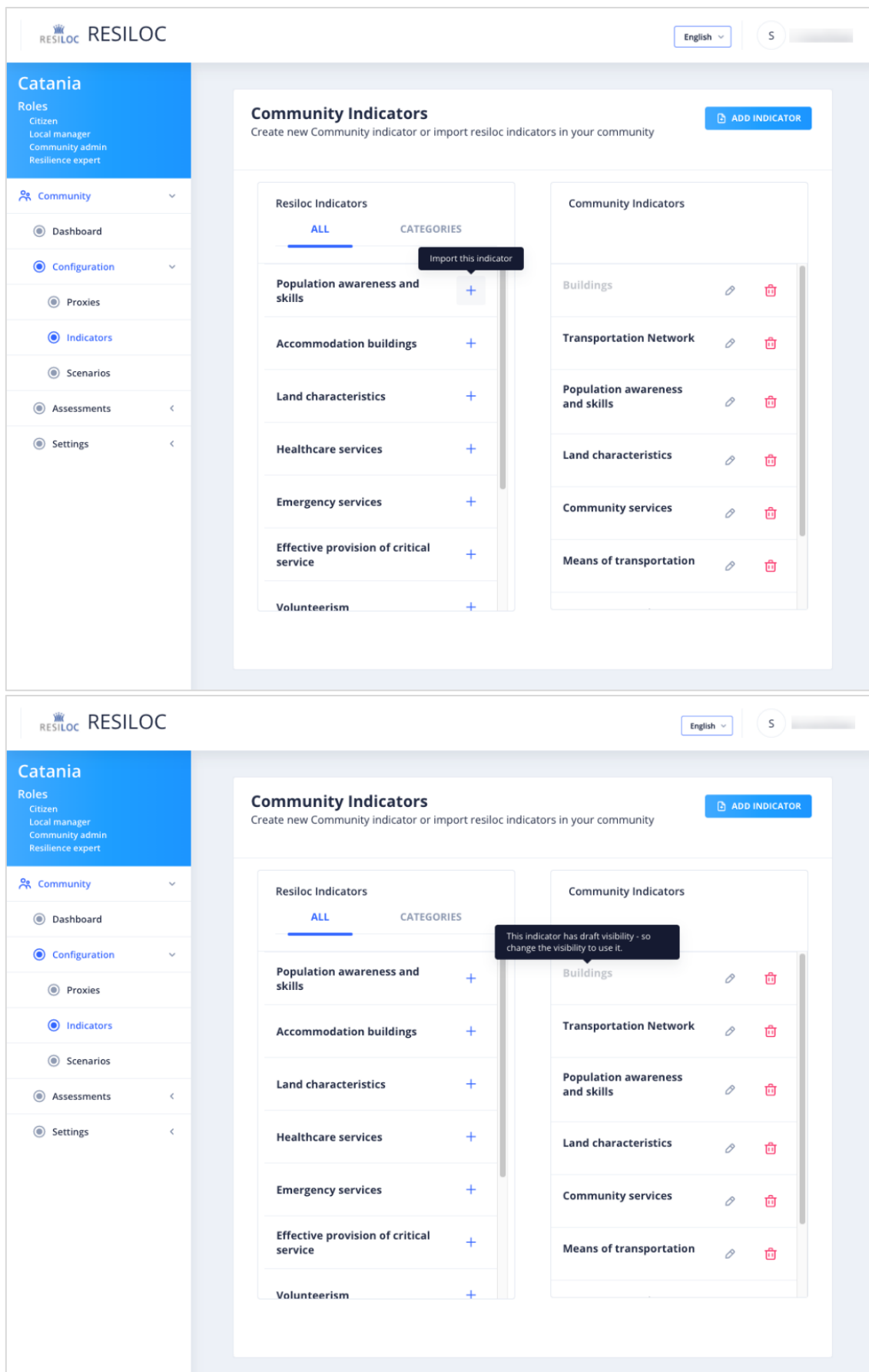


Figure 29. Examples of the RESILOC Inventory interface tooltips, providing clarification where labels do not apply.

The availability of functionalities with a medium-high level of complexity, such as those involving structured data entry, means that each interface component must be very responsive during user interaction; supporting the user in a clear way during the required operations, providing explanatory instructions and using appropriate warning systems when a component involves an operation which requires specific information as presented in Figure 30.

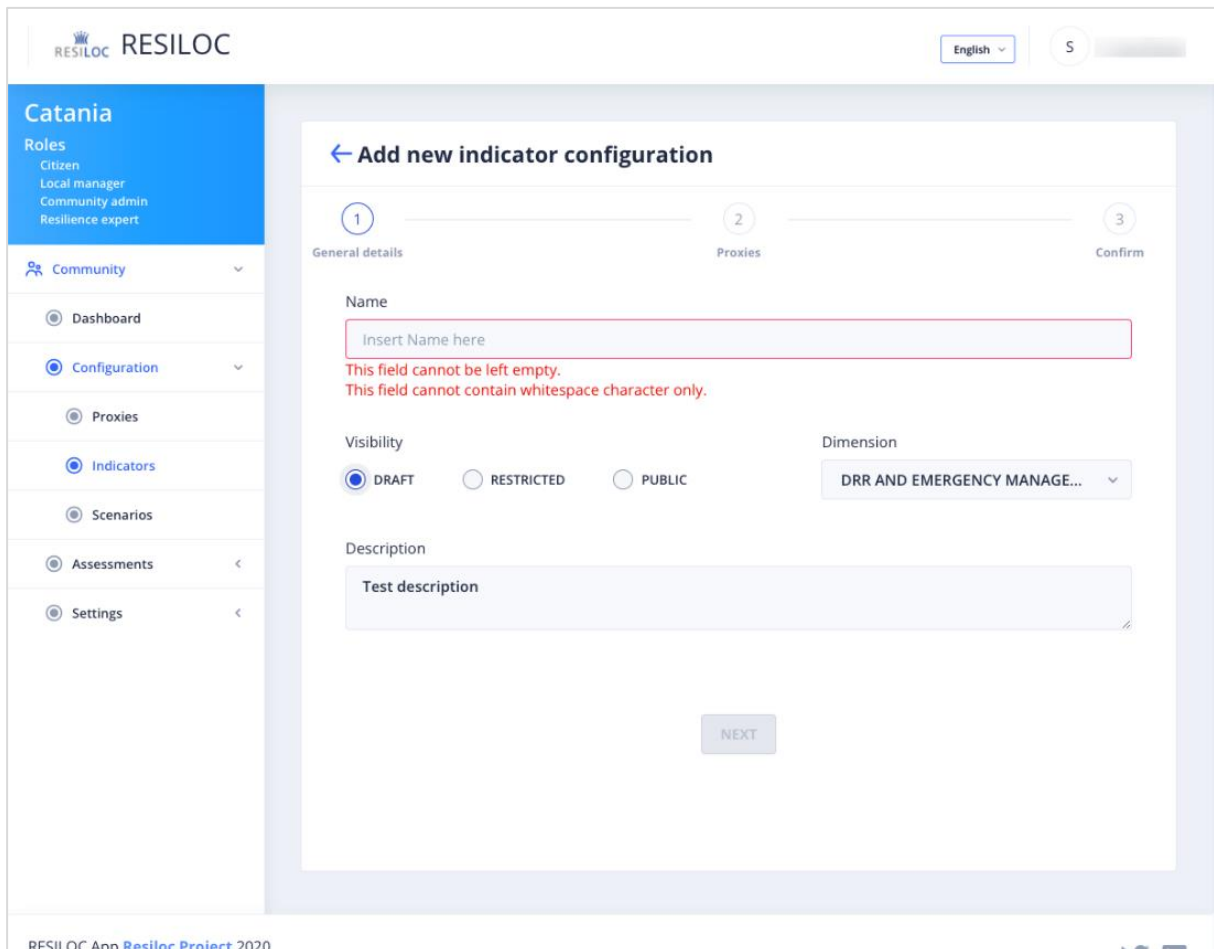


Figure 30. Explanatory instructions and warning systems to simplify the filling in of specific form.

The RESILOC Inventory offers the option to select the language used for the interface labels, allowing the user to switch between English (the default language) and the local language (e.g., Italian, Greek or Bulgarian). Figure 31 shows the language selection function, which is displayed in the interface by a drop-down list allowing the selection of the desired language.

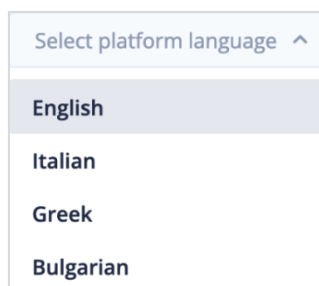


Figure 31. Language selection function by drop-down list.

The RESILOC Inventory interface offers the option to select the elements housed into list or tables adopting filtering and sorting features, allowing the user to find and retrieve the desired ones. The filtering function is displayed on top of the lists and adopts a non-case sensitive character research (Figure 32), while the sorting function allow to display each column by name (alphabetical order) or by date (Figure 33). The functioning of these features is straightforward, for the filter the platform searches for the character or the word, chosen by us, within the name, for example if we search the character "d" inside proxies list, the system will return all proxies that contain that character. Currently in the platform, we use the filtering function only by names, but we have ready to use it with other fields, e.g., for proxies we can apply a filter by type, creation date, last update date and visibility. Regarding sorting, which we display in the tables, it is available also for data not displayed actually in the lists.

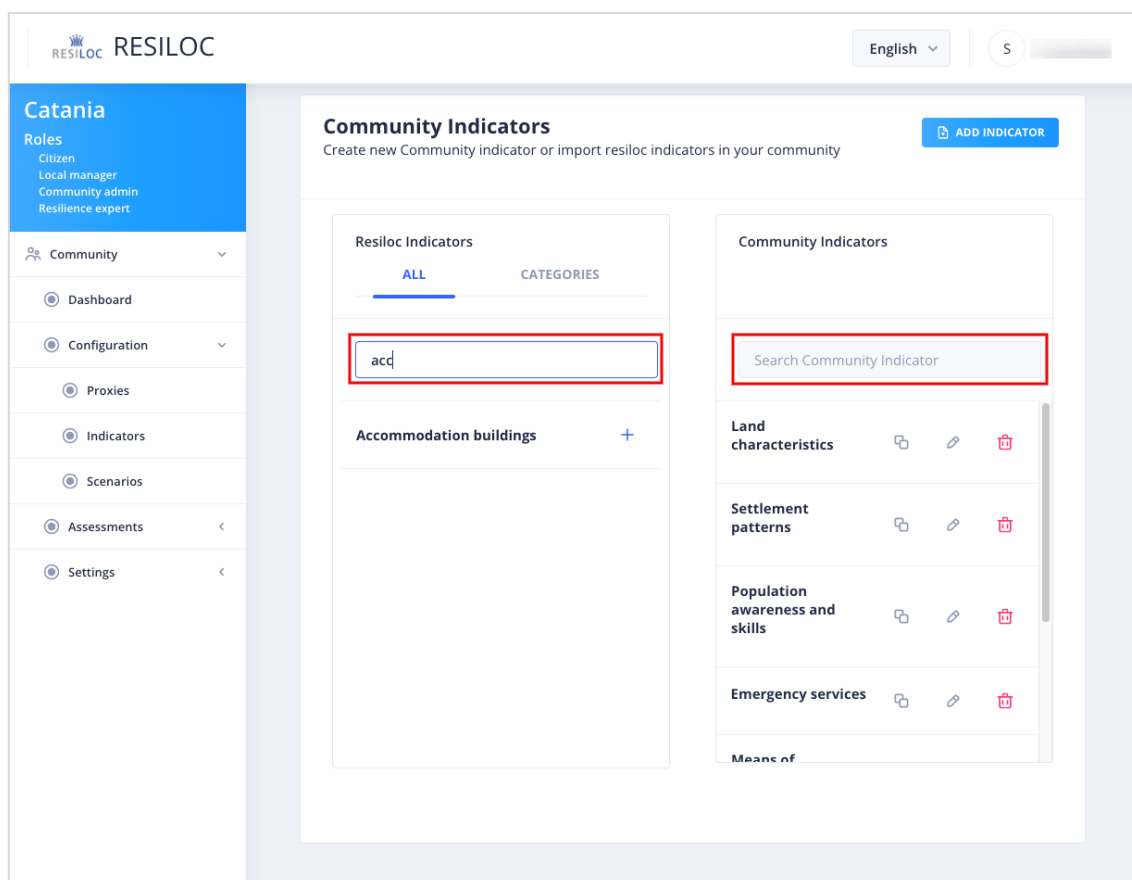


Figure 32. An example of the filtering function applied to the list of RESILOC Indicators.

RESILOC

English

A

Admin

Dashboard

Settings

Configuration

Proxies

Indicators

Scenarios

Indicators

+ ADD INDICATORS

ALL (11)

VISIBLE (11)

DRAFT (0)

Name	Created	Actions
<input type="text" value="Name"/>		
Community services	30/08/2021	
Buildings	30/08/2021	
Means of transportation	30/08/2021	
Settlement patterns	30/08/2021	
Volunteerism	30/08/2021	
Effective provision of critical service	30/08/2021	
Emergency services	31/08/2021	
Healthcare services	31/08/2021	
Land characteristics	31/08/2021	
Accommodation buildings	05/09/2021	

<<

<

1

2

>

>>

Figure 33. An example of the sorting function applied to the list of Indicators from the RESILOC Admin interface.

5 Open Data

This chapter provides vital information about how RESILOC develops, builds, stores, and publishes open data. All the RESILOC-elements (i.e., RESILOC proxies, indicators, and scenarios) and community elements (i.e., community proxies, indicators, scenarios, and snapshots) with “public” visibility, which are configured by authenticated and authorised users of the inventory, can be shared as open data. Even though the Open Data function is not directly a component of the inventory, we mention it in this document since it is considered as public storage of the data. Community users were informed about the Open Data functionality during the 3rd Sprint.

5.1 Open Data Introduction

The full Open Data definition gives precise details as to what this means. To summarise the most important conceptions:

- **Availability and Access:** the data must be available as a whole and at no more than a reasonable reproduction cost, preferably by downloading over the internet. The data must also be available in a convenient and modifiable form.
- **Re-use and Redistribution:** the data must be provided under terms that permit re-use and redistribution including the intermixing with other datasets.
- **Universal Participation:** everyone must be able to use, re-use and redistribute - there should be no discrimination against fields of endeavour or against persons or groups. For example, ‘non-commercial’ restrictions that would prevent ‘commercial’ use, or restrictions of use for certain purposes (e.g., only in education), are not allowed.

The term open in open data is another definition for interoperability. Interoperability denotes the ability of diverse systems and organizations to work together (inter-operate). In this case, it is the ability to interoperate - or intermix - different datasets.

5.2 Open Data Platform

OpenAIRE¹⁴ is chosen as the suitable open data platform for providing access to the RESILOC public data. OpenAIRE is a network of Open Access repositories, archives, and journals that support Open Access policies. OpenAIRE being a project funded from the European Union's Horizon 2020 research and innovation program makes it very promising to use it as a platform to store RESILOC public data there.

OpenAIRE also provides a SPARQL endpoint which makes it very convenient for power users. SPARQL¹⁵ is a query language that enables users to retrieve and manipulate data stored in an open data platform. By using the SPARQL endpoint, users can query and retrieve the exact part of the data they want, and the undesired part of data can be ignored.

¹⁴ <https://www.openaire.eu/>

¹⁵ <https://www.w3.org/TR/rdf-sparql-query/>

5.3 Architecture

Design Architecture of storing RESILOC Open Data is described in the Figure 34 below.

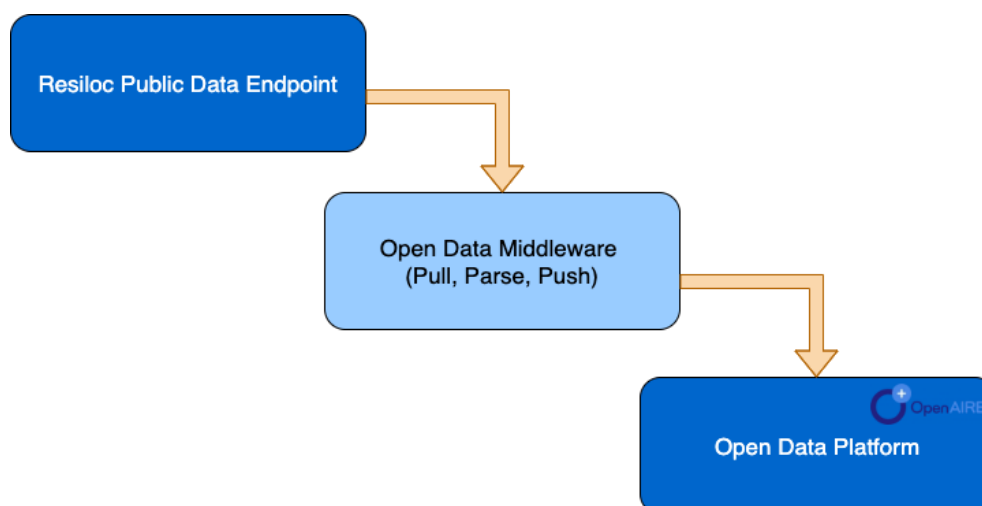


Figure 34. RESILOC Open Data architecture.

RESILOC Open Data Middleware is the bridge between RESILOC and the open data Platform. The middleware consumes data from the open-data endpoints in RESILOC Project, parses the JSON response to compatible RDF format, and finally pushes the data to the Open Data platform. For using the middleware endpoints, one must be logged in and have the necessary permissions. After a successful login, the user calls export API in the middleware and specifies what is going to be exported. Middleware calls the fitting RESILOC API endpoint (based on the type of specified data like proxy, indicator, or scenario) and gets the JSON response (Pull). The JSON format is not a fitting format for storing data in the Open Data platform and it should be converted to a suitable format (Parse). JSON-LD¹⁶ is our choice for making open data publicly available. JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. PyLD library¹⁷ in python is used to convert JSON files to JSON-LD. PyLD is extended to our needs so that it adjusts the JSON files so they can be queried by SPARQL Endpoints. The middleware takes care of this conversion and makes everything ready for export. Finally, an appropriate API endpoint of the Open Data platform is called to send the data into the repository (Push).

JSON-LD conversion:

The JSON-LD syntax does not change the structure of the JSON file, and it does not require many applications to change their JSON. The syntax is designed to not disturb already deployed systems running on JSON but provide a smooth migration path from JSON to JSON with added semantics.

¹⁶ <https://json-ld.org/>

¹⁷ <https://github.com/digitalbazaar/pyld>

A JSON file consists of key-value pairs. In JSON files you cannot typically define types for the values but with JSON-LD we can achieve that by using context. Let's say in our JSON file we have such a key-value pair:

```
"dateCreated": "2021-07-19T09:40:58.746Z"
```

Our middleware converts it to

```
"https://schema.org/dateCreated": [{"@value": "2021-07-19T09:40:58.746Z"}].
```

DateCreated is a valid defined format from the schema.org website and a power user can combine different types to query or merge multiple datasets.

With this, we can include different types in our file and enrich it.

Worth mentioning, the format is intended to be fast to parse, fast to generate, stream-based and document-based processing compatible, and requires a very small memory footprint in order to operate.

5.4 Middleware Technical Overview

In this section, we will discuss the technical part of RESILOC Open Data Middleware. The goal of middleware is to continuously provide RESILOC public data in the Open Data Platform (Zenodo OpenAIRE) so everyone can use these data. Manually converting and putting RESILOC Open Data was not an option because it's tedious work. Therefore, Middleware has a job Scheduler that takes public data from RESILOC Inventory and converts it to JSON-LD and pushes it to Open Data Platform. Job Scheduler is modified to be run monthly. By the first of each month job scheduler calls RESILOC inventory APIs that consist of these seven endpoints namely static-proxies, indicators, scenarios, snapshots, RESILOC-proxies, RESILOC- indicators, and RESILOC-scenarios. Each job calls all of the endpoints, receives the necessary data, converts them to JSON-LD, and finally puts them in the Open Data Platform. The Middleware is written in Python and all HTTP requests are handled by Request Library in Python. Middleware is hosted on one of Fraunhofer Server respecting all of the security measurements like firewalls, IP backlisting in case of bots, and so on. Due to the fact that middleware does not provide any open port to outside, the security measurements are easier met.

6 System Verification

This section lists and describes the actions for the verification of the validity of the system and the conformity of its elements.

As stated in ISO 9000:2005¹⁸: “*Validation is the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment (i.e., meet stakeholder requirements) in the intended operational environment (ISO 2005)*”.

These actions have to be performed in a methodological way, planned and executed during the whole life cycle of the system. If intended as a process, validation is a transversal activity at each stage of the system life cycle. The validation process is a flow that has to be executed during the whole system life cycle, in combination with the definition and implementation processes, the results of which are continuously verified.

In the System Validation design of the RESILOC Inventory backend, we planned to carry out the validation tests during the final phase of the development process. The reason for this decision is that in the design of the RESILOC Inventory it was not possible to identify in the beginning all the requirements that would have to be developed during the project. The requirements were progressively collected during the Sprints in order to create a product as suitable as possible for the needs of the end-users.

The tool that will be used to perform the requirements validity test will be the Requirements Traceability Matrix (RTM) (Cleland-Huang, Gotel and Zisman, 2012). This tool is proposed as a document that allows mapping use cases to individual validation processes (described below) that characterise each functionality to be validated. In general, there are 3 different types of validation that differ on the nature of the requirement type to be validated:

- **Installation Qualifications (IQ):** Verify that the machine on which they are deployed has all the necessary environments to be running properly.
- **Operational Qualifications (OQ):** Verify that the machine on which the services are deployed is sufficiently performing.
- **Performance Qualifications (PQ):** verify that systems perform tasks in real-world conditions.

As a product that has been designed with continuous interaction with end-users, we have included another metric:

- **User Acceptance Test (UAT):** Users produce feedback confirming the effectiveness of a developed specification,

to perform a validation process, the actions that should be performed are as follows and in the given order:

1. Identification of the requirements and use cases that have been collected during the previous sprints and interactions with end-users.
2. Translation of requirements into system specifications.
3. Identification of the results expected by the end-users based on the expectations that led to the identification of the use cases and requirements.
4. Writing of the Requirements Traceability Matrix.

¹⁸ <https://www.iso.org/standard/42180.html>

Once all of the above verifications have been carried out, it will be possible to go on to compile the traceability matrix and this will be included as a result of the process in D4.3. Deliverable D4.3 is a report that contains the results of all the components developed in Work Package 4. The results are the output of the validation tests carried out on the basis of the requirements collected from the end-users and allow to give a general but complete view on the compliance of the developed functionalities and the collected requirements.

7 Conclusion

The present deliverable was prepared in the context of the WP4 Implementation of the RESILOC Platform, with the primary goal to document the efforts undertaken within the context of the T4.1 - AGILE Implementation of RESILOC Inventory. The development specifications, the inventory implementation's workflows, and the components that compose the inventory presented in this document provide deeper insights into how the inventory is created.

The RESILOC project has adopted an AGILE approach for the inventory development, which has started based on architecture and specifications described in the D2.7, then continuing by essential updates, enhancements, and adjustments that are the results of a comprehensive analysis performed on the feedback received from the community users through 4 different sprints. By facilitating an agile implementation process, we ensure that the inventory development is highly dynamic, flexible, and collaborative.

The user manual included in this document describes several use cases with detailed steps that users must take in order to effectively interact with the inventory according to different user roles.


Verification activities, which are used to ensure the fulfilment of the developed RESILOC Inventory towards the design and user requirements, will be performed by T4.3 through dedicated verification workshops and will be mentioned in detail in D4.3.

In the following steps, the outcomes of this deliverable will support the implementation activities of the RESILOC Platform. The Platform will include, once completed, the RESILOC Inventory that will be, nonetheless, available also as a stand-alone product.

VII. List of References

- Arias, D. (2020) *The Complete Guide to Angular User Authentication with Auth0*. Available at: <https://auth0.com/blog/complete-guide-to-angular-user-authentication/> (Accessed: 15 November 2021).
- Barth, A. (2011) *HTTP State Management Mechanism*, Internet Engineering Task Force. Available at: <https://datatracker.ietf.org/doc/html/rfc6265> (Accessed: 18 November 2021).
- Bradshaw, S., Brazil, E. and Chodorow, K. (2019) *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media.
- Cleland-Huang, J., Gotel, O. and Zisman, A. (eds) (2012) *Software and Systems Traceability*. Springer, London. doi: <https://doi.org/10.1007/978-1-4471-2239-5>.
- Cockburn, A. (2006) *Agile software development: the cooperative game*. Pearson Education.
- Dudjak, M. and Martinović, G. (2020) 'An API-first methodology for designing a microservice-based backend as a service platform', *Information Technology and Control*, 49(2), pp. 206–223. doi: 10.5755/j01.itc.49.2.23757.
- Fenton, S. (2014) *Pro Typescript: application-scale JavaScript development*. Apress. Available at: <https://books.google.co.uk/books?id=ZEtADwAAQBAJ&pg=PA108>.
- Jones, M. and Hardt, D. (2012) *The oauth 2.0 authorization framework: Bearer token usage, IETF Rfc*.
- Momjian, B. (2001) *PostgreSQL: introduction and concepts*. Addison-Wesley New York.
- Raj, P., Chelladurai, J. S. and Singh, V. (2015) *Learning Docker*. Packt Publishing Ltd.
- Rescorla, E. and Dierks, T. (2018) *The Transport Layer Security (TLS) Protocol Version 1.3*, Internet Engineering Task Force. Available at: <https://datatracker.ietf.org/doc/html/rfc8446> (Accessed: 18 November 2021).
- Sandhu, R. S. (1998) 'Role-based access control', in *Advances in computers*. Elsevier, pp. 237–286.
- Schwaber, K. and Beedle, M. A. (2002) *Agile software development with Scrum*. Prentice Hall Upper Saddle River.
- W3C (1992) *Status codes*. Available at: <https://www.w3.org/Protocols/HTTP/HTRESP.html> (Accessed: 29 October 2021).
- W3C (2011) *REST*. Available at: <https://www.w3.org/2001/sw/wiki/REST> (Accessed: 1 November 2021).

VIII. Annex A: Ethics Self-Assessment Sheet

RESILOC		RESILOC ethics self-assessment sheet			
This document is a self-assessment sheet that must be filled out by owners of RESILOC deliverables. This is to ensure that research and/or development activities related to each project deliverable comply with requirements of RESILOC Guidelines on Ethics and Data Protection (GDPR).					
This RESILOC ethics self-assessment sheet must be used as part of each project deliverable that involves humans either in an active (e.g., data subjects) or passive (e.g., affected by tools) manner. Project reports (e.g., management or financial reports) are not required to undergo this ethics assessment.					
This document is an important exercise part of the RESILOC Ethics Framework as it allows the owner of each RESILOC deliverable to reflect on ethical consideration and data protection requirements in a structured and approved manner before submitting the document to the Commission for review.					
The document shall be used in line with the RESILOC Ethics Framework including the guidelines and procedures under deliverables D9.1 to D9.12 (all documents are made available on the RESILOC Own Cloud). The ethics self-assessment sheet must be included as the 1st Appendix A of the each RESILOC deliverable. In addition to filling out the sheet, authors must provide explanations of the answers given on the main table. Such explanations must be provided in the methodology section of the deliverable using the headline "Ethics Considerations and Data Protection". The ethics self-assessment sheets of private deliverables must be assessed through the responsible position within the issuing organisation. However, for public deliverables, the ethics self-assessment sheet must be approved by the RESILOC Internal Ethics Board. For that, please send this document to the Internal Ethics Board.					
For Information or assistance contact:				helena.marruecos@iml.fraunhofer.de	
The self-assessment was conducted by:				The self-assessment was approved by:	
Name	Hoang Long	Name	Rajendra		
Surname	Nguyen	Surname	Akerkar		
Institution	WNRI	Institution	WNRI		
Date	15.02.2021	Date	26.02.2021		
				yes	no
				n/a	
G	GENERAL				
a	Did the research for this deliverable involve the collection of personal data?			x	
b	Does this deliverable, and the activities that have fed into it, comply with Regulation (EU) 2016/679 known as GDPR and 2002/58/EC Directive on privacy and electronic communications?			x	
c	Does this deliverable, and the activities that have fed into it, comply with the relevant national data protection and privacy laws, codes of practice and guidelines?			x	
d	Are there any ethics risk identified related to your work under this deliverable?				x
1	Human Participation/ Informed Consent				
1.1	Procedures and criteria that will be used to identify/recruit research participants (D9.1)				x

a	Did the research for this deliverable involve the recruitment of research participants? <i>(this includes surveys and interviews)</i>			
b	Did you identify selection, inclusion, & exclusion criteria?			
1.2	Recruitment of respondents via social media (D9.4)			x
b	Were special measures taken to ensure that the participants are adults?			
c	Did the research for this deliverable involve data collection using social media?			
d	Were measures taken to use only public profiles for the collection of data?			
		yes	no	
1.3	Use of the informed consent forms and Info sheets to recruit research participants (D9.2)			x
a	Consent Form was issued			
b	Information sheet was issued			
c	Combined sheet was issued			
			Issued in local language	
1.4	Use of the informed consent forms and information sheets on data processing (D9.9)			x
a	Consent Form was issued			
b	Information sheet was issued			
c	Combined sheet was issued			
			Issued in local language	
2	Organizational measures			
2.1	Data Protection Officer or contact person (D9.5)			
a	Do you have a Data Protection Officer or contact person for participants?	x		
b	Was this contact mentioned on the Informed Consent Forms?		x	
3	Technical measures			
3.1	Technical safeguard mechanisms for handling of personal data (PD) and special categories of personal data (SCOPD) (D9.6 / D9.8) (SCOPD include information such as ethnic origin, political opinions, data concerning health, etc. For more details see Article 9(1) GDPR).			
a	Did the research for this deliverable involve the collection of SCOPD? (D9.6)		x	
b	Which mechanisms were used to safeguard the personal data collected?			
	pseudonymisation		x	
	encryption	x		
	access restriction	x		
	anonymization	x		
	other (specify in the line below)			
3.2	Data minimisation (D9.7)			
a	Has as little as possible data been collected throughout the research process?	x		
b	If more data was collected than initially needed, did you ensure the data was deleted?	x		
3.3	Data profiling (D9.10)			
a	Was or will the data collected in the deliverable be used for data profiling?		x	
b	Were all data subjects informed of the profiling and its possible consequences? (as part of the Inform Consent Form and the Information Sheet)			x

c	Were sufficient measures in place to safeguard their fundamental rights?			x		
3.4	Processing of previously collected personal data (D9.11)			x		
a	Did you obtain consent to use personal data from previously executed research?					
b	Are technical/organisational measures required to safeguard the rights and freedoms of the data subject according to EU and national legislation in place in your organisation?					
4	Other Issues of ethical concern					
a	Were there any other ethical considerations detected during the work of this deliverable that are not covered by the list above?		x			
b	If yes, please list the concerns below and elaborate on the related counter measures in the methodology section of this document					
5	Opinions/approvals provided by ethics committees and other experts					
5.1	Following documents received opinions/approvals provided by ethics committees and other experts for the research conducted for this deliverable.					
		yes	no			
a	Informed Consent Forms and Information sheet	IEB	x	EEA	x	
		DPO	x	LEB	x	
b	Questionnaires / Surveys	IEB		EEA		x
		DPO		LEB		
c	Design / Methodology of research activity	IEB		EEA		x
		DPO		LEB		

IX. Annex B: RESILOC Inventory User Manual

The users of the RESILOC Inventory may have different capabilities according to the role they hold for a specific community. The roles currently defined in the inventory are listed below:

- RESILOC admin
- Community admin
- Local manager
- Resilience expert
- Local Resilience Team
- Citizen

Each role involves the usage of different developed functionalities; these may be presented by playing the shoes of different users, simulating their interaction with the platform and performing operations based on duties linked to assigned role. Some users may have one or more roles assigned, according to expertise and the availability of community resources.

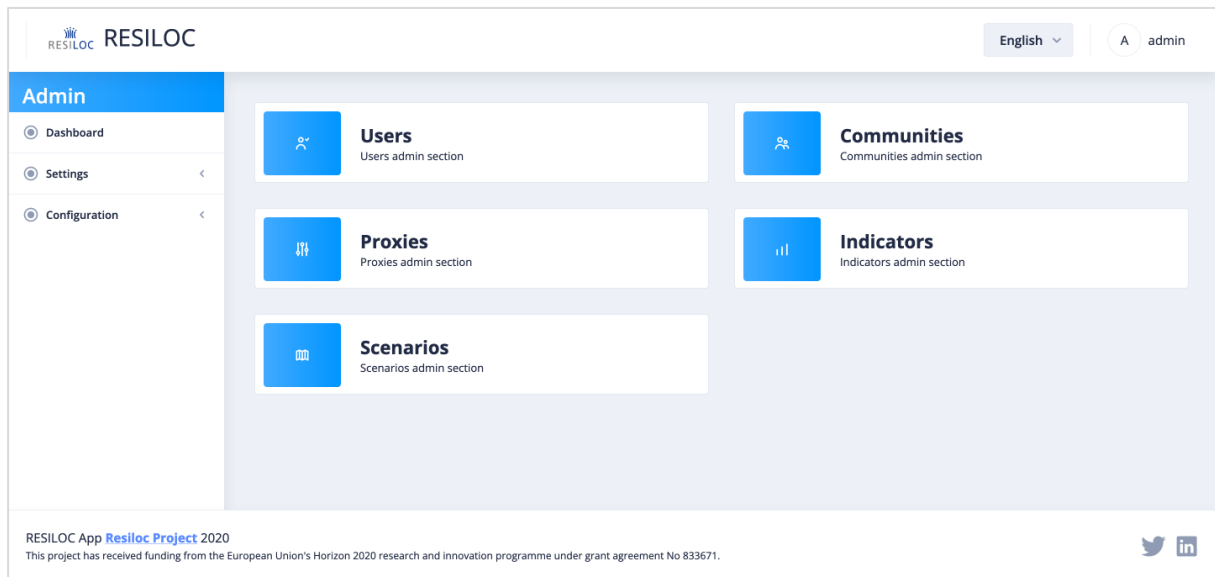
In order to provide a better understanding on how the users could interact with the RESILOC Inventory, a user manual supported by several use cases (told as a story) are presented below, playing the shoes of different fictional characters.



A. RESILOC admin

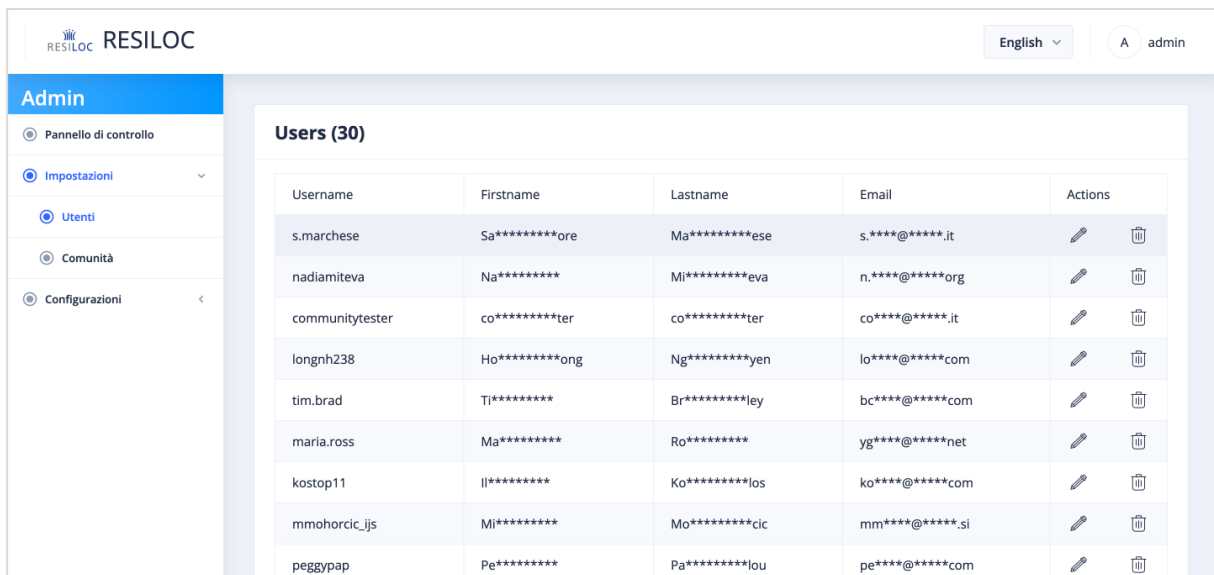
The **RESILOC admin** role is assigned to one or more users who has a solid background in resilience-related topics and a comprehensive knowledge of the project; specifically, on the methodology behind the process of the resilience assessment in RESILOC. After the end of the project, it should be given to a public body for the management of the system (e.g., ERCC).

Once the **RESILOC admin** logged in, the inventory shows an overview of the different components it has to offer. The **admin** can then select the section he/she wants to work on.



In the "Users" section, the **admin** is able to consult the list of all users registered in the system. Each user is featured by several information (username, first name, last name, email address).

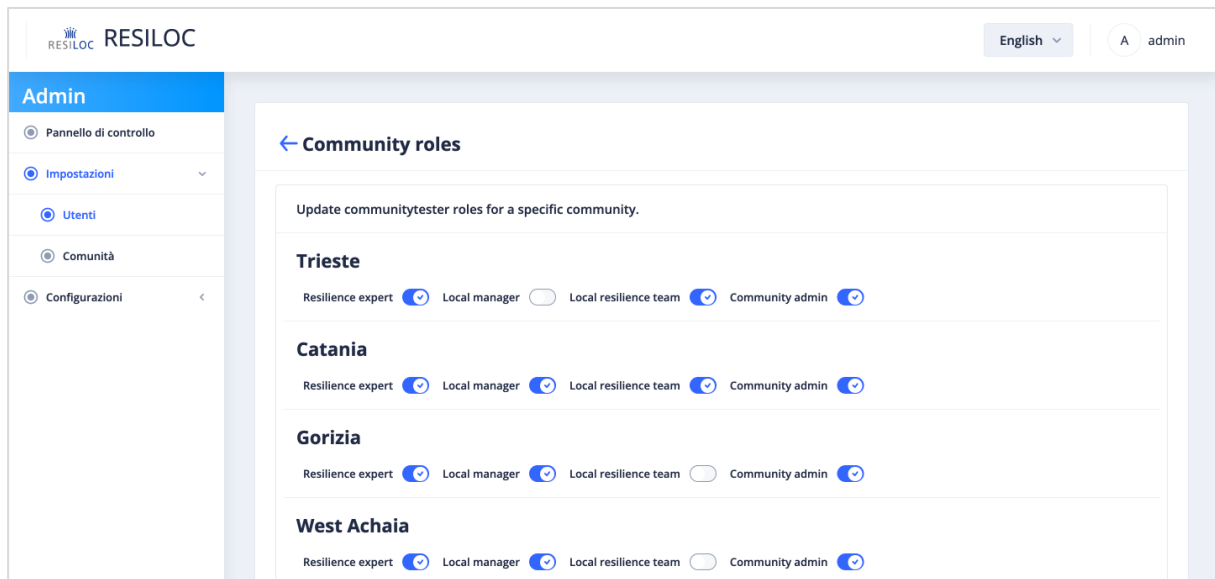
The **admin** here can execute some actions, he/she has the capabilities to change the user role by clicking on pencil icon or delete the user itself (removing it from the system) by clicking on the basket icon.



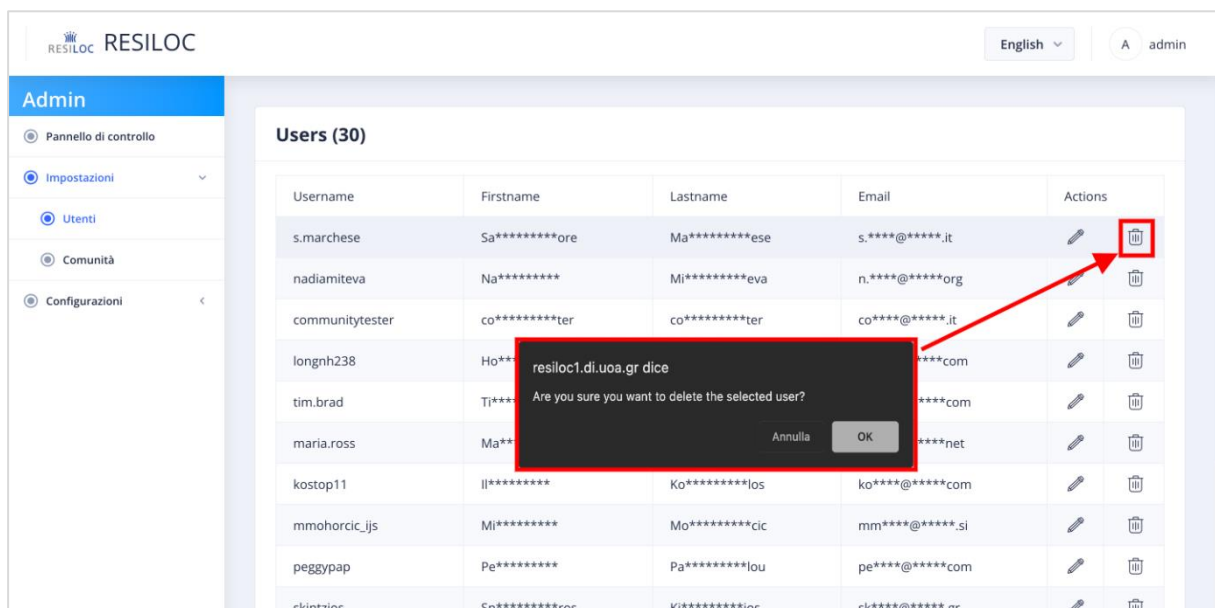
The screenshot shows the 'Users (30)' section of the RESILOC Admin interface. It displays a table with columns: Username, Firstname, Lastname, Email, and Actions. The table lists 10 users, each with a pencil icon for editing and a trash icon for deletion.

Username	Firstname	Lastname	Email	Actions
s.marchese	Sa*****ore	Ma*****ese	s.****@****.it	
nadiamiteva	Na*****	Mi*****eva	n.****@****.org	
communitytester	co*****ter	co*****ter	co****@****.it	
longnh238	Ho*****ong	Ng*****yen	lo****@****.com	
tim.brad	Ti*****	Br*****ley	bc****@****.com	
maria.ross	Ma*****	Ro*****	yg****@****.net	
kostop11	Ij*****	Ko*****ios	ko****@****.com	
mmohorcic_ijs	Mi*****	Mo*****cic	mm****@****.si	
peggyap	Pe*****	Pa*****lou	pe****@****.com	

By selecting the pencil icon, the **admin** accesses the user role edit page. For each community followed by the user, the **admin** can manage the user role switching on/off the corresponding button.



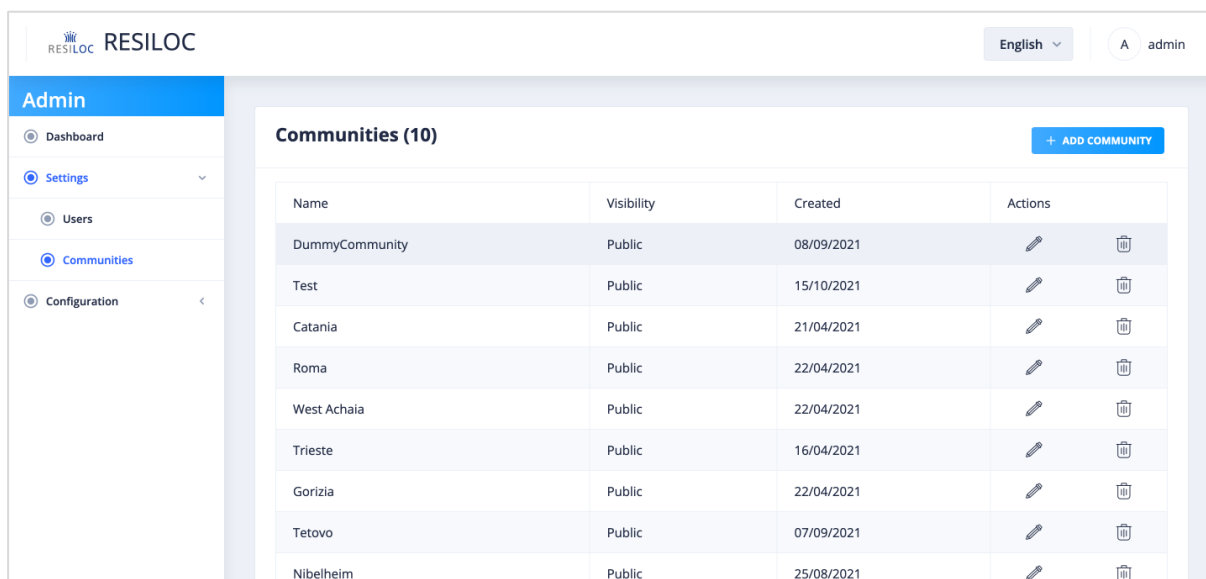
Registered users can be removed from the system by searching them in the list and clicking on the basket icon. The system then displays a confirmation message to proceed with the removal of the user.



In the "Communities" section, the **admin** is able to consult the list of communities registered in the system. For each community are showed several information: community name, visibility, creation date within the system. Using the functionalities available under "Actions" column, it is possible to:

- edit the community information,
- delete the community itself.

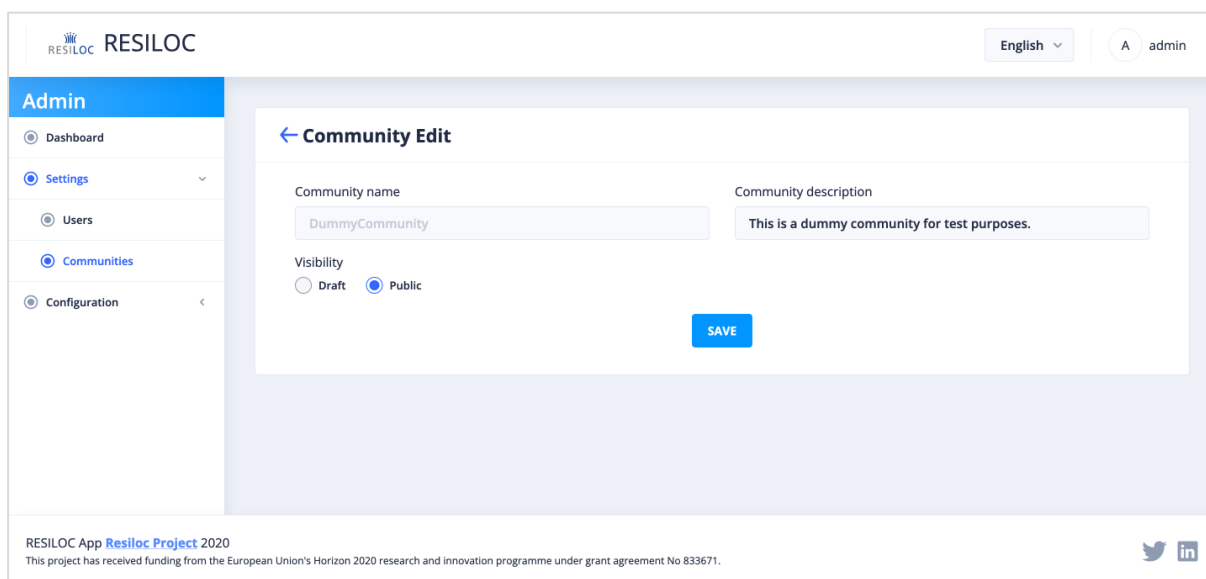
The **admin** can also add a new community by clicking on "+ ADD COMMUNITY" button.



The screenshot shows the RESILOC Admin interface. On the left is a sidebar with navigation links: Admin, Dashboard, Settings, Users, Communities (selected), and Configuration. The main content area is titled 'Communities (10)' and includes a '+ ADD COMMUNITY' button. Below this is a table listing communities with columns for Name, Visibility, Created, and Actions.

Name	Visibility	Created	Actions
DummyCommunity	Public	08/09/2021	
Test	Public	15/10/2021	
Catania	Public	21/04/2021	
Roma	Public	22/04/2021	
West Achaia	Public	22/04/2021	
Trieste	Public	16/04/2021	
Gorizia	Public	22/04/2021	
Tetovo	Public	07/09/2021	
Nibelheim	Public	25/08/2021	

By selecting the pencil icon, the **admin** accesses the community edit page. The system shows the community's information already entered, allowing to modify them and save the changes.



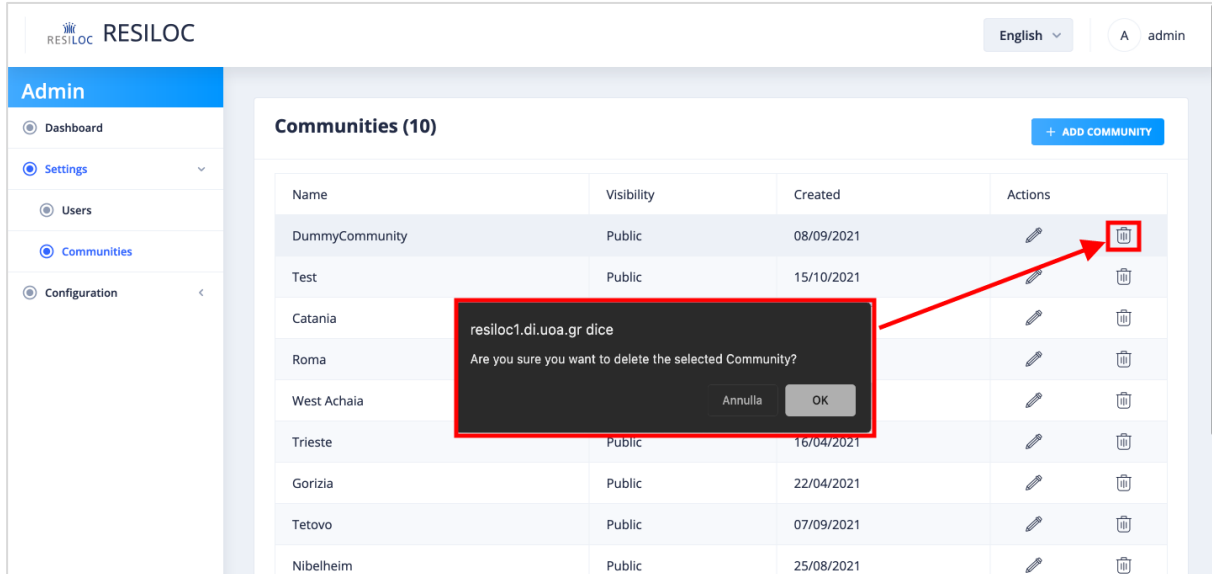
The screenshot shows the 'Community Edit' page in the RESILOC Admin interface. The sidebar is the same as in the previous screenshot. The main content area is titled 'Community Edit' and contains a form with the following fields:

- Community name:
- Community description:
- Visibility: ☐ Draft ☒ Public

A 'SAVE' button is located at the bottom right of the form.

At the bottom of the page, there is a footer with the text: 'RESILOC App Resiloc Project 2020. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833671.' and social media icons for Twitter and LinkedIn.

A registered community can be deleted from the system by searching it in the list and clicking on the basket icon. The system then displays a confirmation message to proceed with the removal of the community.

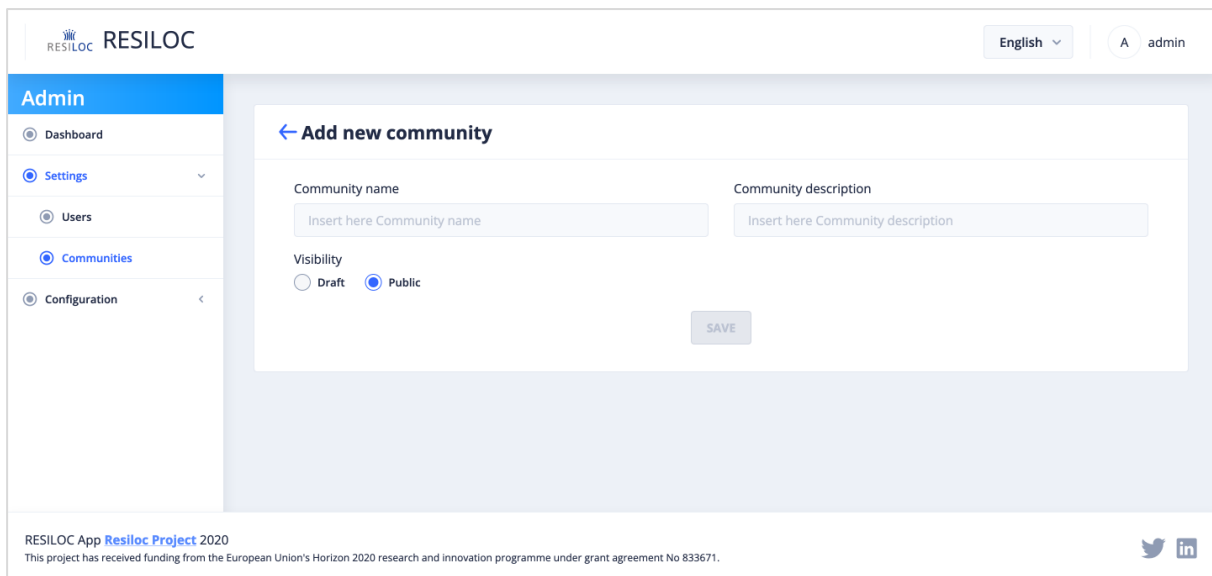


The screenshot shows the RESILOC Admin interface. On the left is a sidebar with navigation links: Dashboard, Settings, Users, Communities (selected), and Configuration. The main content area is titled 'Communities (10)' and features a '+ ADD COMMUNITY' button. Below this is a table with columns: Name, Visibility, Created, and Actions. The table lists 10 communities, including 'DummyCommunity', 'Test', 'Catania', 'Roma', 'West Achaia', 'Trieste', 'Gorizia', 'Tetovo', and 'Nibelheim'. A modal dialog is overlaid on the table, asking 'Are you sure you want to delete the selected Community?' with 'Annulla' and 'OK' buttons. A red arrow points from the delete icon in the 'Actions' column of 'DummyCommunity' to the modal.

Name	Visibility	Created	Actions
DummyCommunity	Public	08/09/2021	[Edit] [Delete]
Test	Public	15/10/2021	[Edit] [Delete]
Catania			[Edit] [Delete]
Roma			[Edit] [Delete]
West Achaia			[Edit] [Delete]
Trieste	Public	16/04/2021	[Edit] [Delete]
Gorizia	Public	22/04/2021	[Edit] [Delete]
Tetovo	Public	07/09/2021	[Edit] [Delete]
Nibelheim	Public	25/08/2021	[Edit] [Delete]


By clicking on the "+ ADD COMMUNITY" button the **admin** accesses the page for the creation of a new community. The system asks the **admin** to enter a name and a brief description of the community along with the visibility selection:

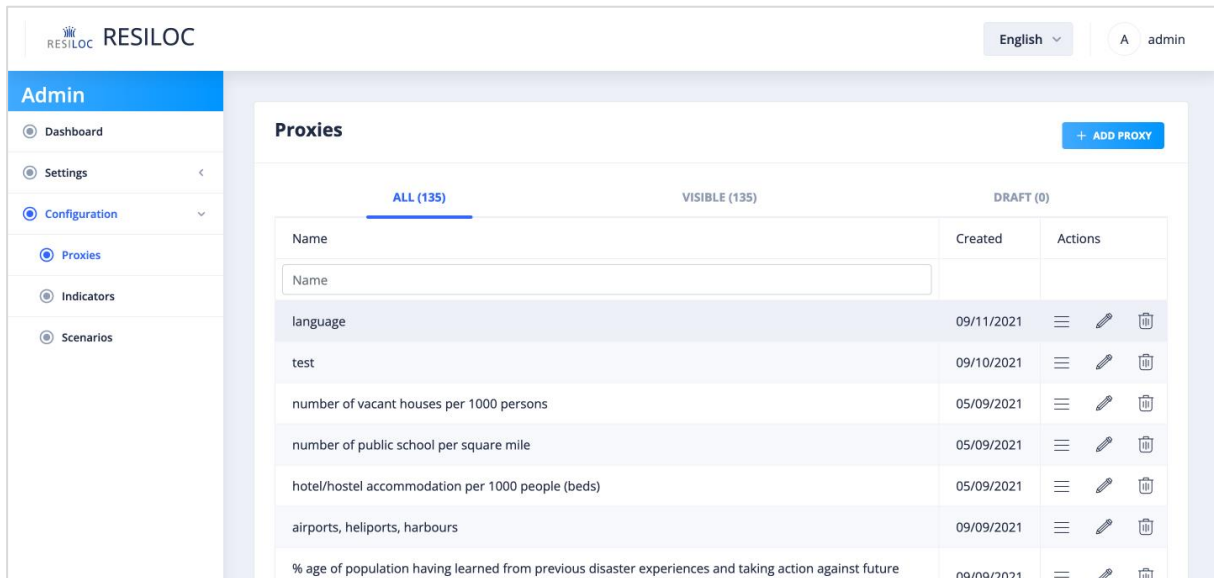
- Draft – not visible in the list of followable communities (admin only).
- Public – visible to all users (open data).



The screenshot shows the 'Add new community' form in the RESILOC Admin interface. The form has two input fields: 'Community name' and 'Community description'. Below these is a 'Visibility' section with two radio buttons: 'Draft' and 'Public' (selected). A 'SAVE' button is located at the bottom right of the form. The footer of the page contains text about the RESILOC App and funding from the European Union's Horizon 2020 research and innovation programme.

In the "Proxies" section, the **admin** is able to consult the list of all proxies added into the system and made available to RESILOC communities. The list is also organised into 3 tabs: ALL (all proxies), VISIBLE (submitted only), DRAFT (still in draft and not available for communities).

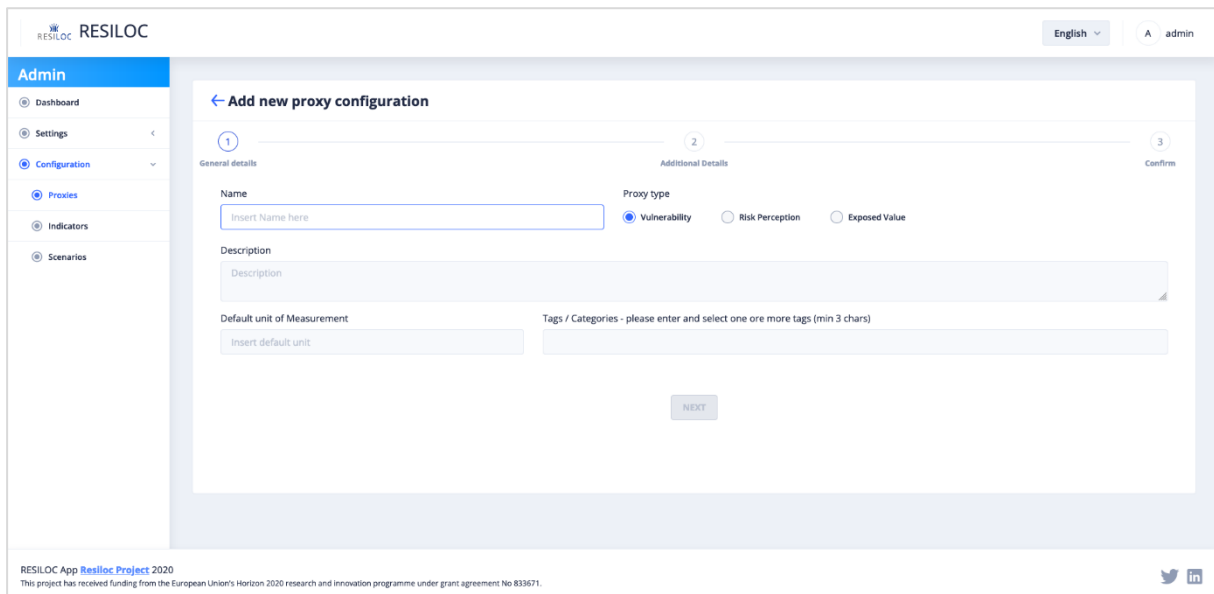
Each proxy is identified by a name and a typology, it can be modified in any part by clicking on the pencil icon or deleted by clicking on the basket button (if not linked to an indicator). The **admin** has also the capabilities to create a new proxy by clicking on "+ ADD PROXY" button or to duplicate any proxy by clicking on  button.



The screenshot shows the RESILOC Admin interface. On the left is a sidebar with the 'Admin' menu and sub-items: Dashboard, Settings, Configuration (selected), Proxies, Indicators, and Scenarios. The main content area is titled 'Proxies' and has a '+ ADD PROXY' button. Below the title, there are three tabs: 'ALL (135)', 'VISIBLE (135)', and 'DRAFT (0)'. The 'ALL (135)' tab is active, displaying a table of proxies. The table has columns for Name, Created, and Actions. The 'Name' column contains a search input field and a list of proxy names: 'language', 'test', 'number of vacant houses per 1000 persons', 'number of public school per square mile', 'hotel/hostel accommodation per 1000 people (beds)', 'airports, heliports, harbours', and '% age of population having learned from previous disaster experiences and taking action against future'. The 'Created' column shows dates: '09/11/2021', '09/10/2021', '05/09/2021', '05/09/2021', '05/09/2021', '09/09/2021', and '09/09/2021'. The 'Actions' column contains icons for edit, delete, and a menu.

By clicking on the "+ ADD PROXY" button the **admin** accesses the page for the creation of a new proxy. The system asks the **admin** to enter a name and a brief description of the proxy along with the proxy's type selection, unit of measurement definition and tags association.

The association of tags may be useful to the user when he/she is searching for a proxy or even for clustering them.



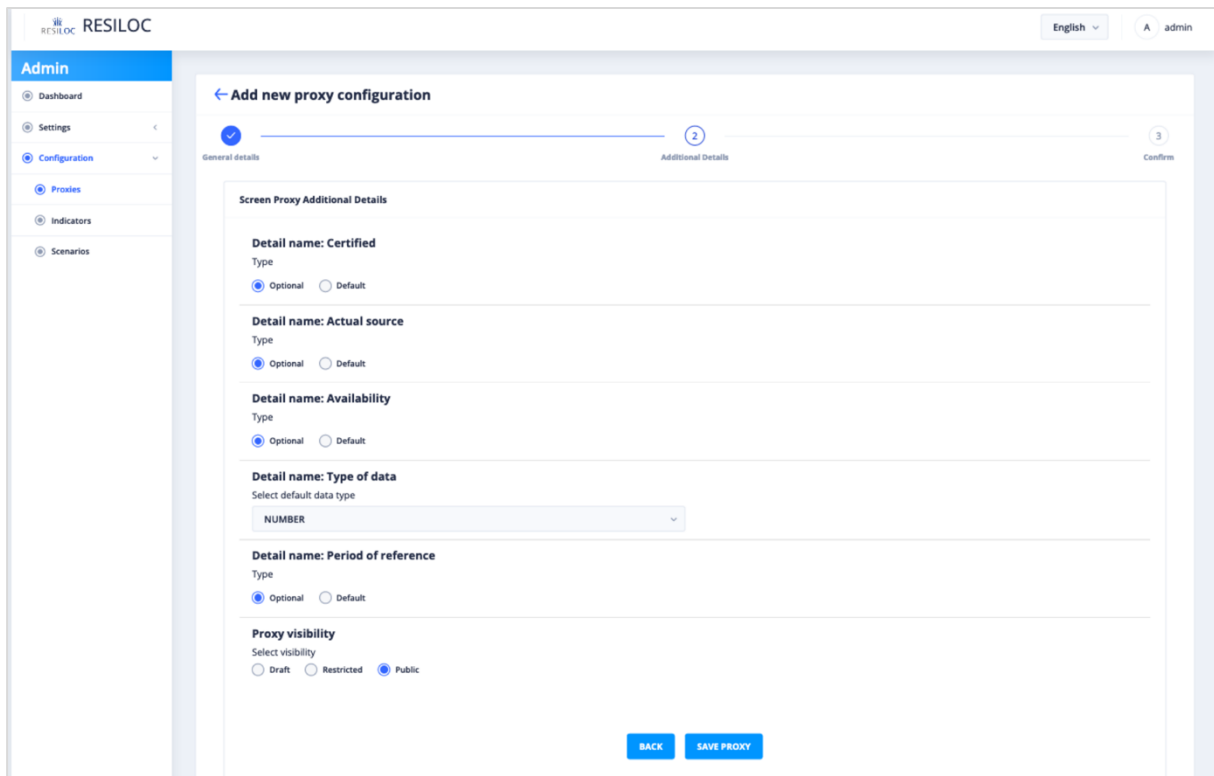
The screenshot shows the 'Add new proxy configuration' form in the RESILOC Admin interface. The form is divided into three sections: 'General details', 'Additional Details', and 'Confirm'. The 'General details' section contains a 'Name' field with a placeholder 'Insert Name here', a 'Description' field with a placeholder 'Description', and a 'Default unit of Measurement' field with a placeholder 'Insert default unit'. The 'Additional Details' section contains a 'Proxy type' section with three radio buttons: 'Vulnerability' (selected), 'Risk Perception', and 'Exposed Value'. Below this is a 'Tags / Categories - please enter and select one ore more tags (min 3 chars)' field. A 'NEXT' button is located at the bottom of the form. The footer of the page contains the text 'RESILOC App Resilloc Project 2020' and 'This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833671', along with social media icons for Twitter and LinkedIn.

Once the general info section has been completed, the system asks **admin** to set up some proxy related additional details:

- certified – specifies whether the data is issued by a certified entity, i.e., obtained from an official source.

- actual source – specifies the actual source of data entity/body/organization that provides the data.
- availability – specifies the data availability (measured or derived).
- type of data – specifies the data typology, e.g., number or string.
- period of reference – period of data validity, e.g., the data is surveyed annually so it is valid for 1 year.

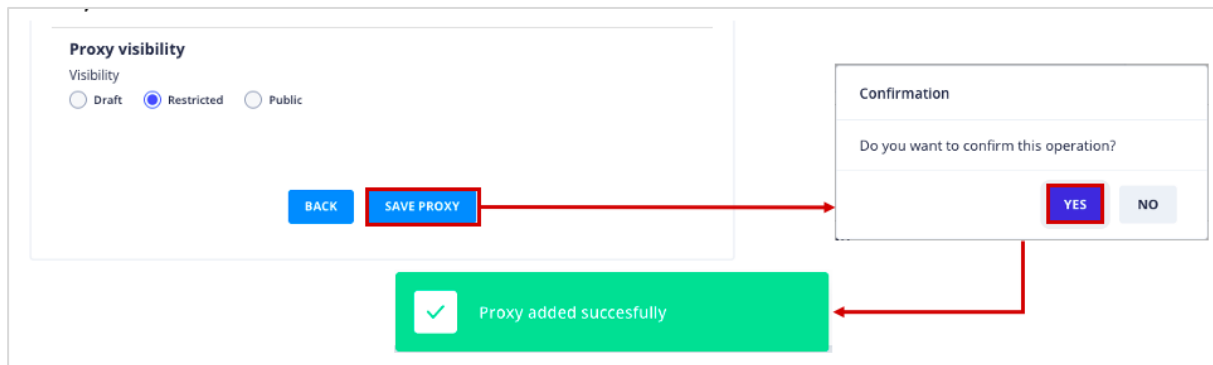
By selecting "optional", the filling-in of the specific additional detail is left to the end user. Selecting "default" gives the admin the faculty to set a default value for the additional detail considered; this will be however modifiable by the communities that decide to use this proxy.




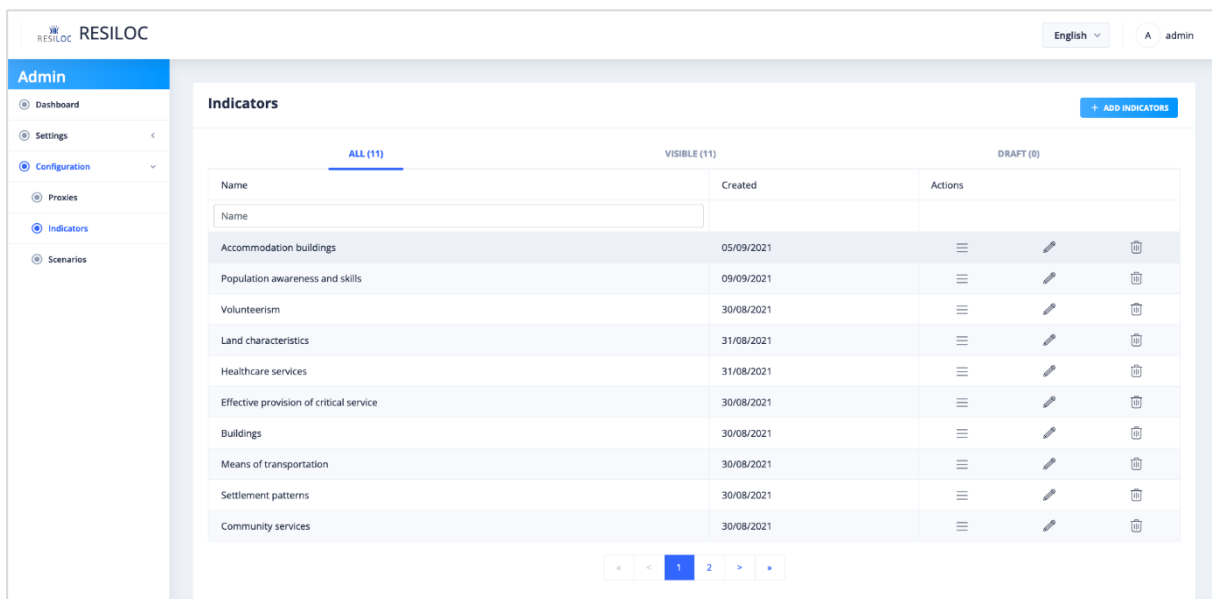
The **admin** completes the procedure selecting the proxy's visibility and clicking on "SAVE PROXY" button.































- Draft – not visible (admin only).
- Restricted – visible only to resilience experts, local manager and LRT.
- Public – visible to all, including citizens (open data).

Once confirmed, the system will save the proxy and add it to the list of available proxies (so called RESILOC Proxies).



In the "Indicators" section, the **admin** is able to consult the list of all indicators added into the system and made available to RESILOC communities. The list also here is organised into 3 tabs, ALL (all indicators), VISIBLE (submitted only), DRAFT (still in draft and not available for communities). Each indicator is identified by a name and can be modified in any part by clicking on the pencil icon or deleted by clicking on the basket button (if not linked to a scenario). The **admin** has also the capabilities to create a new indicator by clicking on "+ ADD INDICATOR" button or to duplicate any indicator by clicking on  button.

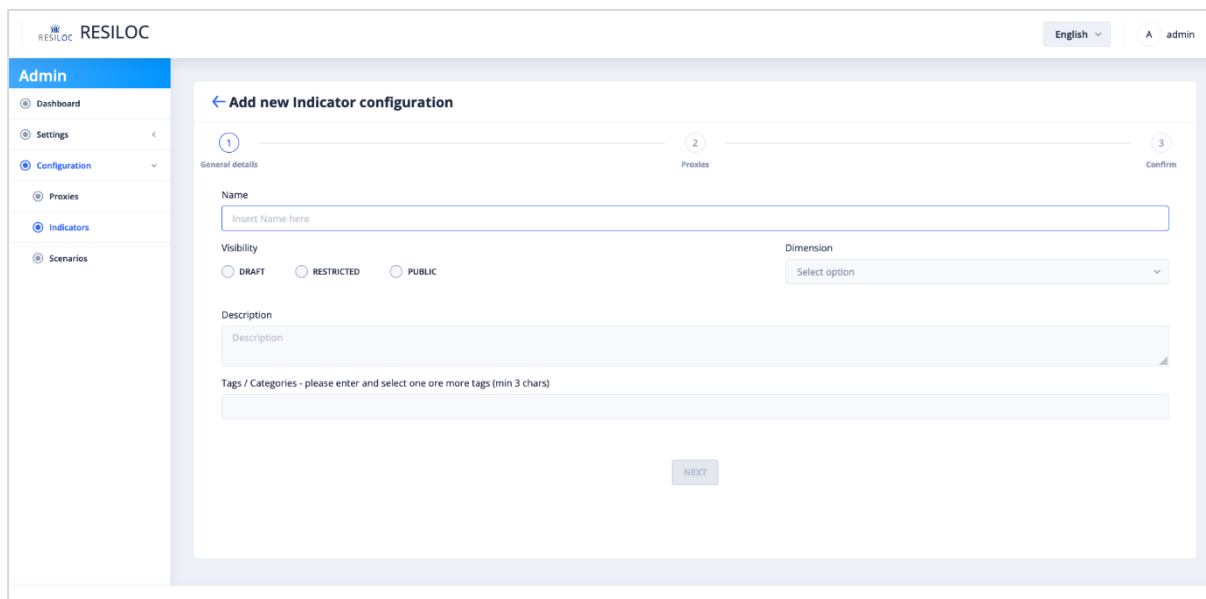


Name	Created	Actions
Accommodation buildings	05/09/2021	  
Population awareness and skills	09/09/2021	  
Volunteerism	30/08/2021	  
Land characteristics	31/08/2021	  
Healthcare services	31/08/2021	  
Effective provision of critical service	30/08/2021	  
Buildings	30/08/2021	  
Means of transportation	30/08/2021	  
Settlement patterns	30/08/2021	  
Community services	30/08/2021	  

By clicking on the "+ ADD INDICATOR" button the **admin** accesses the page for the creation of a new indicator. The system asks the **admin** to enter a name and a brief description of the indicator along with its visibility and dimension selection, ending then with tags association.

Visibility (target roles)

- *Draft: Admin only.*
- *Restricted: Local Manager, Resilience expert, LRT.*
- *Public: All users.*

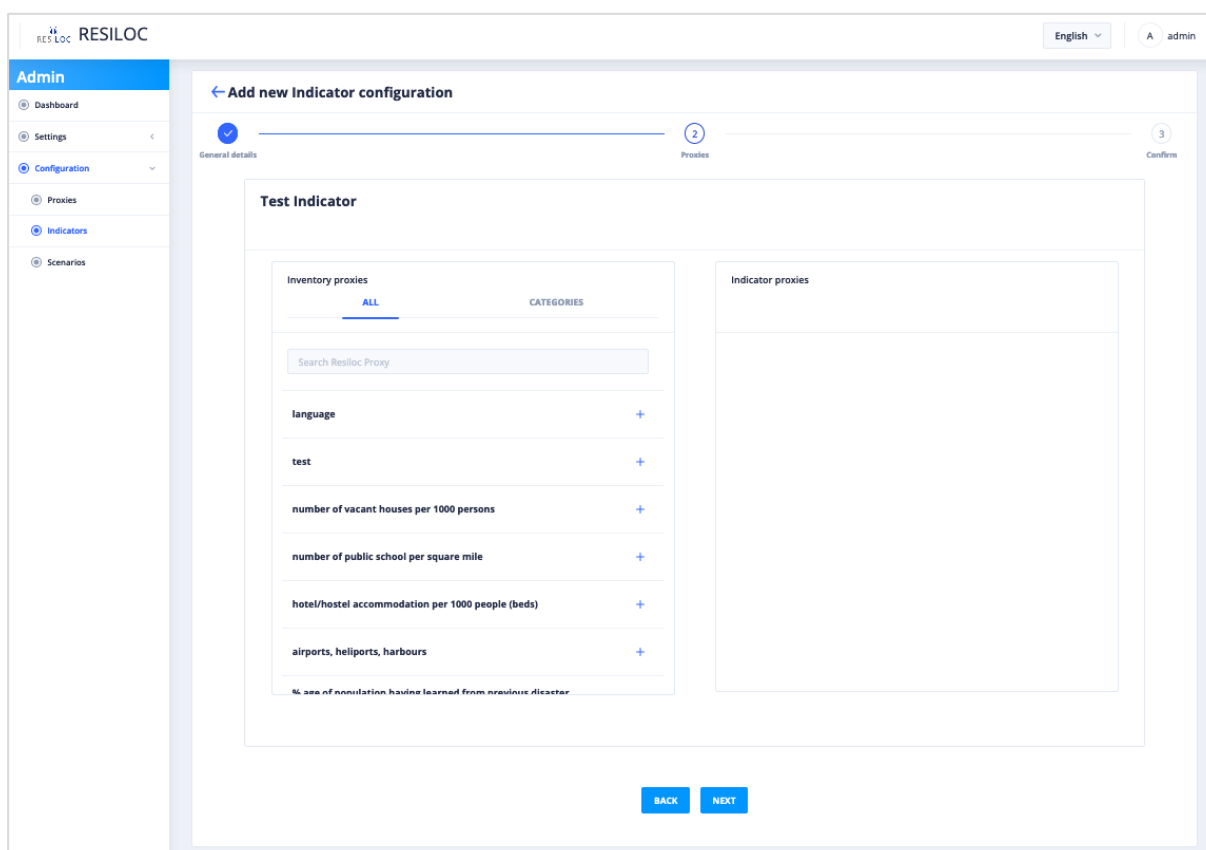


The screenshot shows the 'Add new Indicator configuration' form in the RESILOC Admin interface. The form is divided into three steps: 1. General details, 2. Proxies, and 3. Confirm. Step 1 is currently active. The form includes fields for Name, Visibility (DRAFT, RESTRICTED, PUBLIC), Dimension (Select option), Description, and Tags / Categories (please enter and select one or more tags (min 3 chars)). A 'NEXT' button is visible at the bottom right.

Once the general details section has been compiled, the system asks to select the proxies involved into indicator calculation.

By clicking on "+" the **admin** can then select from a list of available proxies those he/she want to use and add them to the indicator's proxies list.

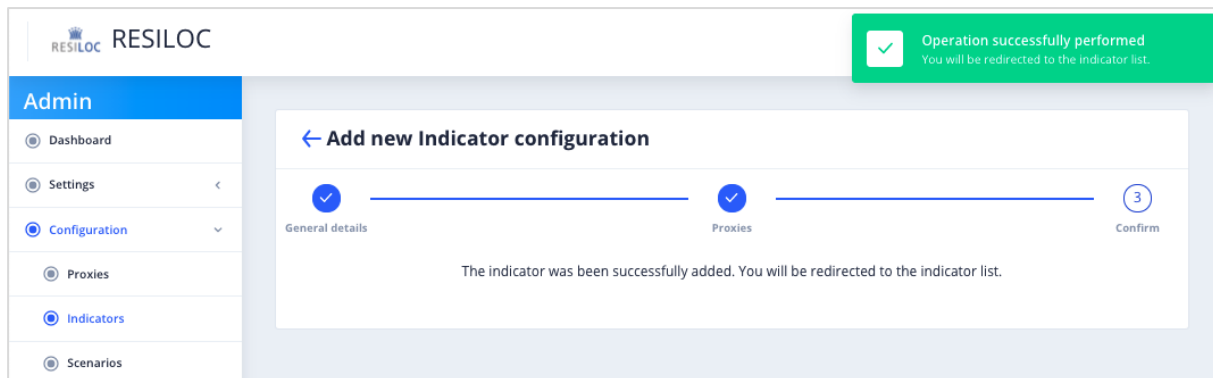
The available proxies can be displayed by categories or showed without any specific grouping.




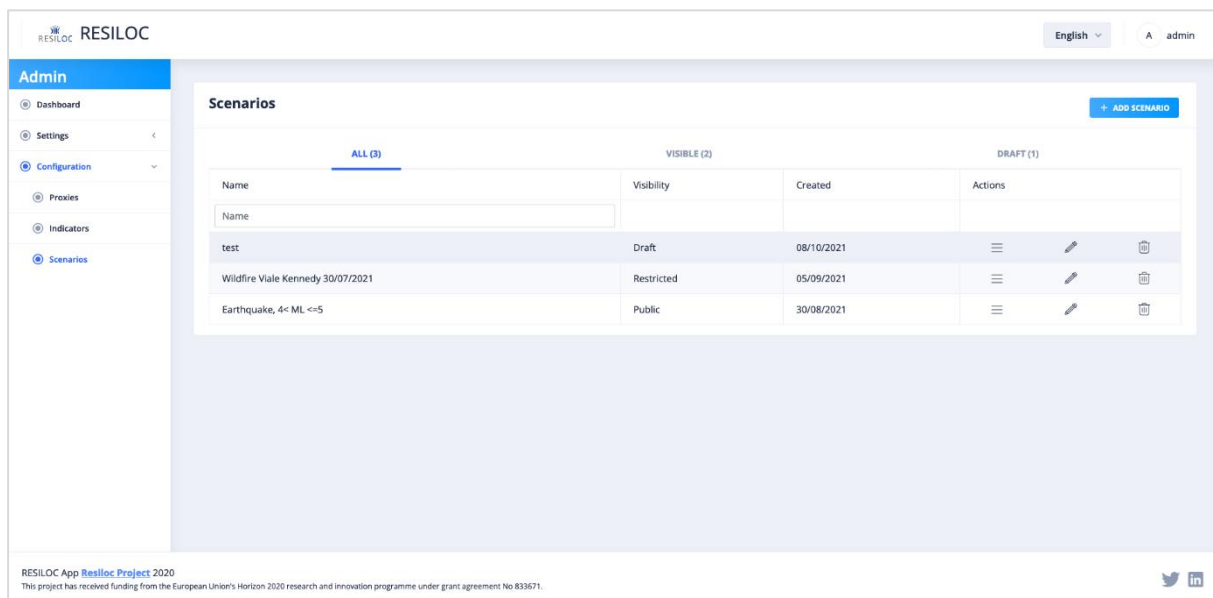
The screenshot shows the 'Add new Indicator configuration' form in the RESILOC Admin interface, Step 2: Proxies. The form is divided into three steps: 1. General details, 2. Proxies, and 3. Confirm. Step 2 is currently active. The form includes a 'Test Indicator' section with a table of available proxies. The table has columns for 'Inventory proxies' and 'Indicator proxies'. The 'Inventory proxies' column lists various categories with a '+' button next to each. The 'Indicator proxies' column is empty. A 'BACK' button is visible at the bottom left, and a 'NEXT' button is visible at the bottom right.

Inventory proxies	Indicator proxies
language	
test	
number of vacant houses per 1000 persons	
number of public school per square mile	
hotel/hostel accommodation per 1000 people (beds)	
airports, heliports, harbours	
% pop. of population having learned from previous disaster	

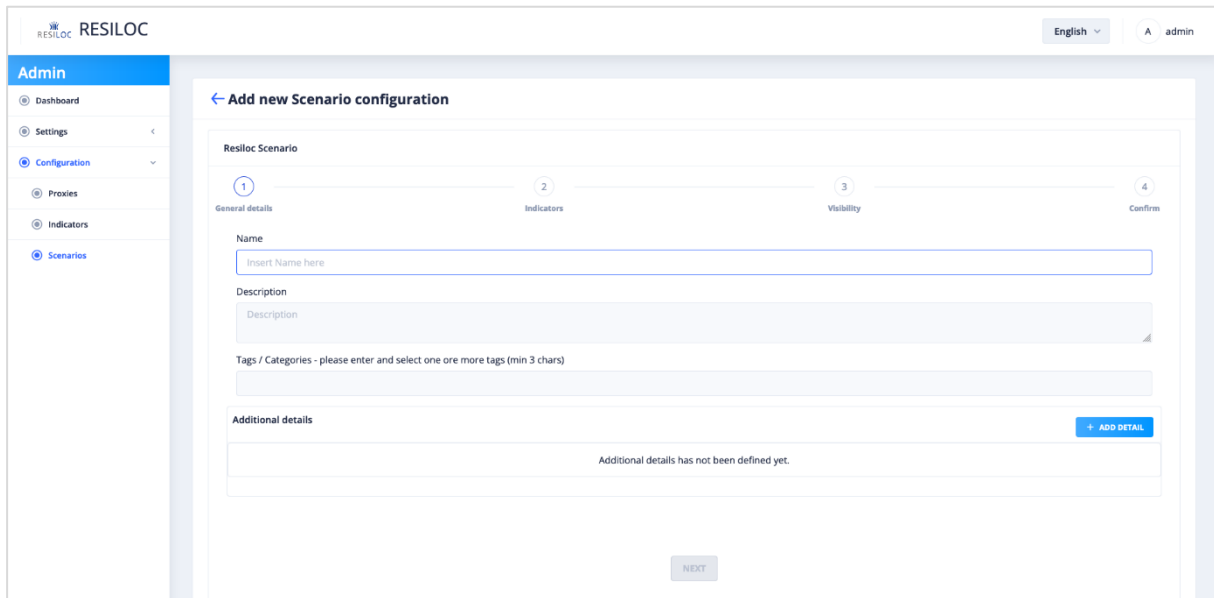
After having complete the proxies' selection, the system will allow the **admin** to save the newly defined indicator. Once confirmed, the system will save the indicator and add it to the list of available indicators (so called RESILOC Indicators).



In the "Scenarios" section, the **admin** is able to consult the list of all scenarios created into the system and made available to RESILOC communities. The list also here is organised into 3 tabs, ALL (all scenarios), VISIBLE (submitted only), DRAFT (still in draft and not available for communities). Each scenario is identified by a name and can be modified in any part or deleted by clicking on the corresponding button. The **admin** has also the capabilities to add a new scenario by clicking on "+ ADD SCENARIO" button or to duplicate any scenario by clicking on the  button.

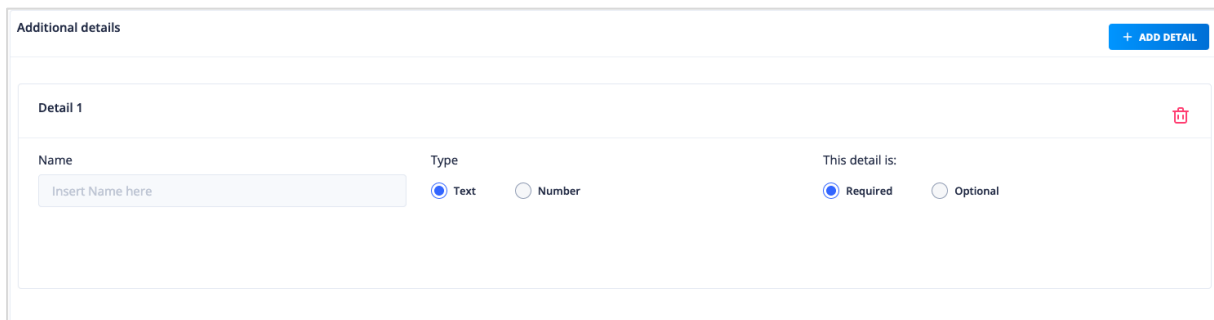


By clicking on the "+ ADD SCENARIO" button the **admin** accesses the page for the creation of a new scenario. The system asks the **admin** to enter a name and a brief description of the scenario along with tags association.



Once the general details section has been compiled, the system asks to set up some scenario related additional details. By clicking on the "+ ADD DETAIL" button the system displays a window for the creation of a new one. This will be required or optional during the scenario configuration phase, the aim is to collect useful information for a richer description of the event.

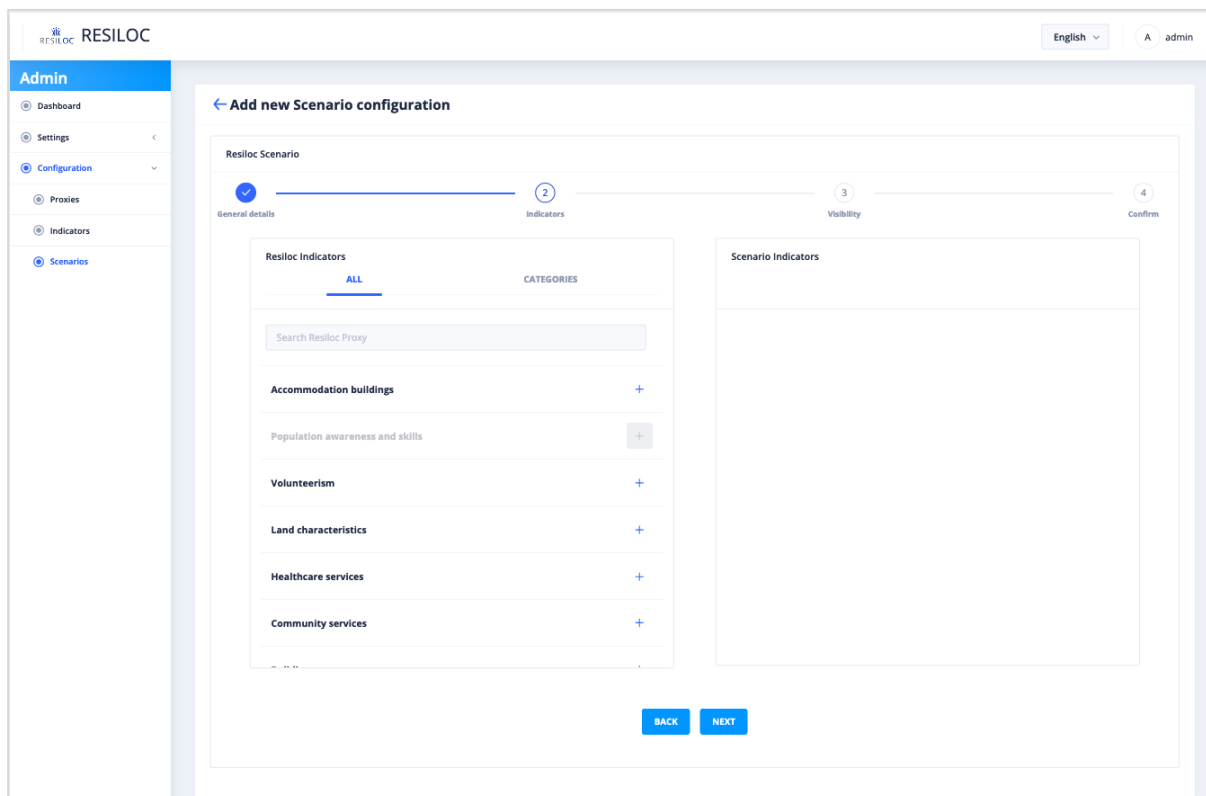
The system asks the **admin** to define a name, select the typology and specifies if it will be required or not to the end-user during the scenario configuration for a given community.



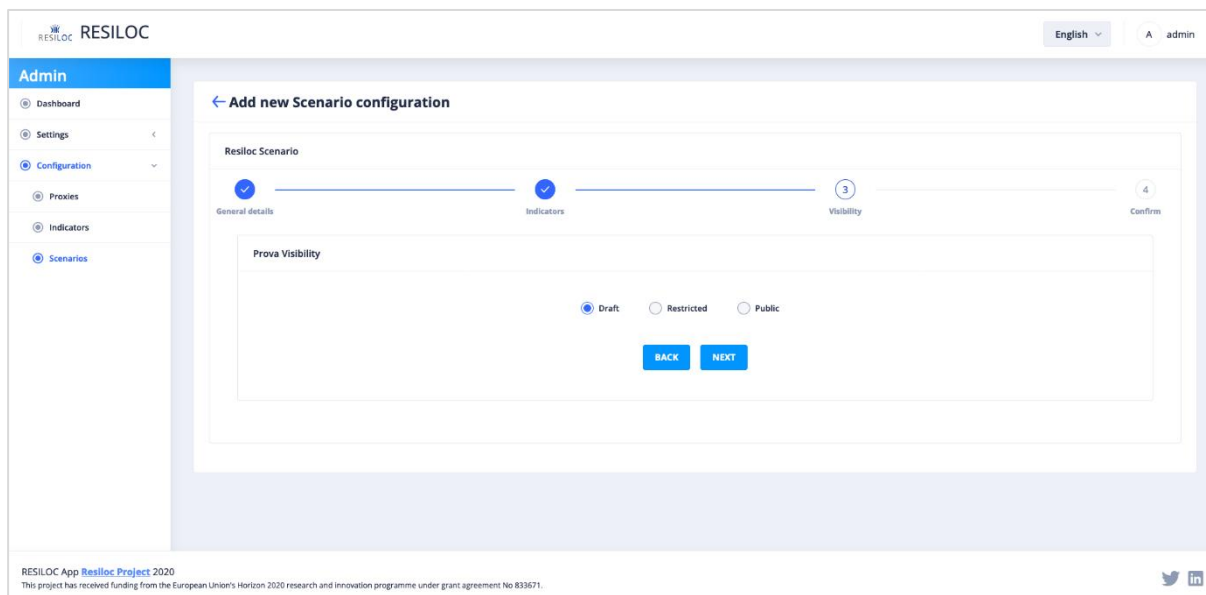
Once the general details section has been compiled, the system asks to select the indicators involved into scenario considered.

By clicking on "+" the **admin** can then select from a list of available indicators those he/she want to use and add them to the scenario's indicators list.

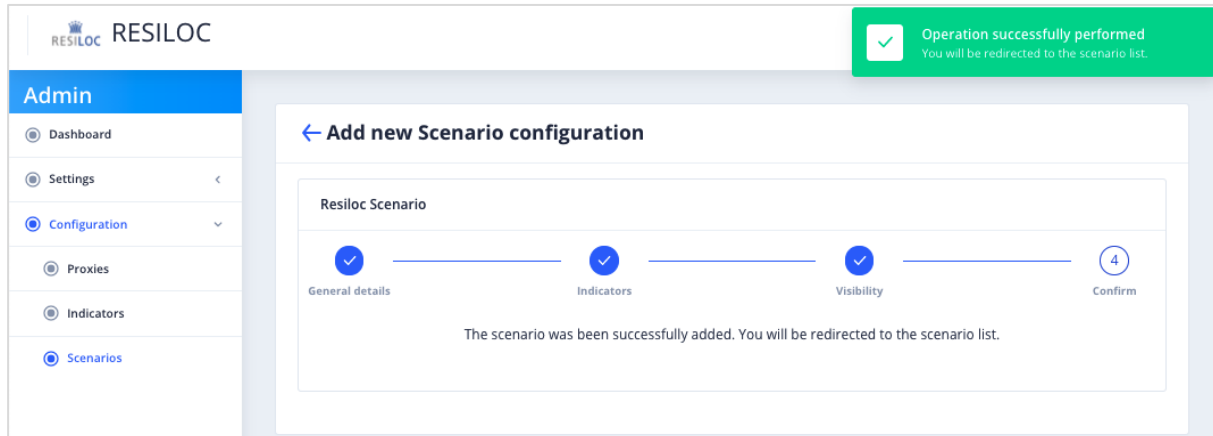
The available indicators can be displayed by categories or showed without any specific grouping.



After having complete the indicators' selection, the system will allow the **admin** to select scenario's visibility and proceed to save the newly defined scenario.



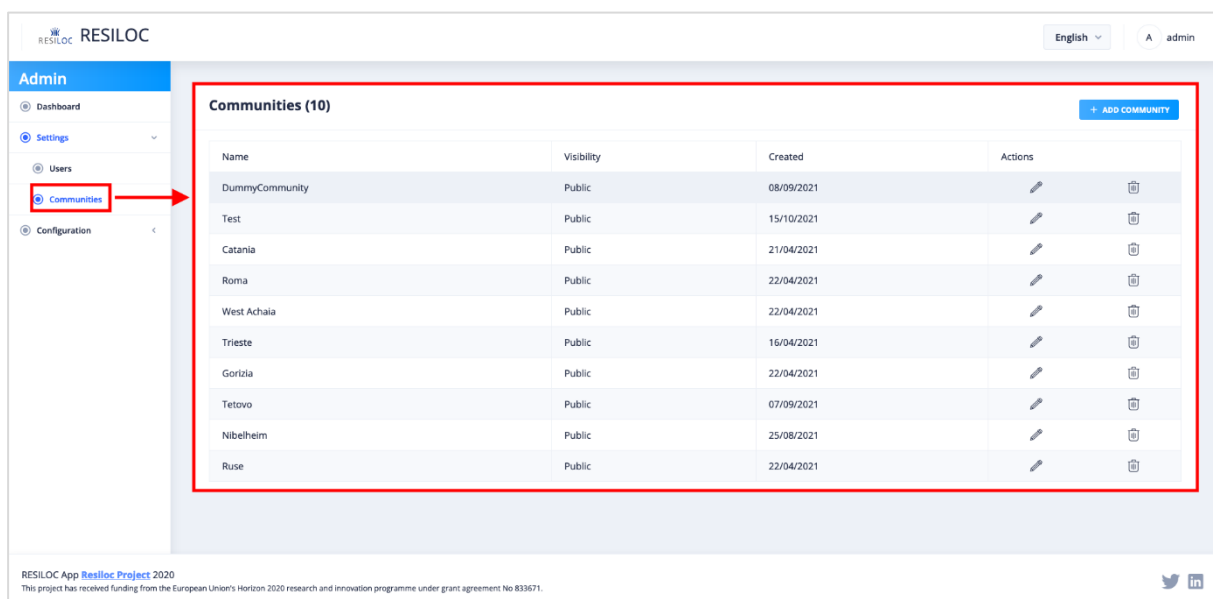
Once confirmed, the system will save the scenario and add it to the list of available scenarios (so called RESILOC Scenarios).



RESILOC Admin use case – new community creation

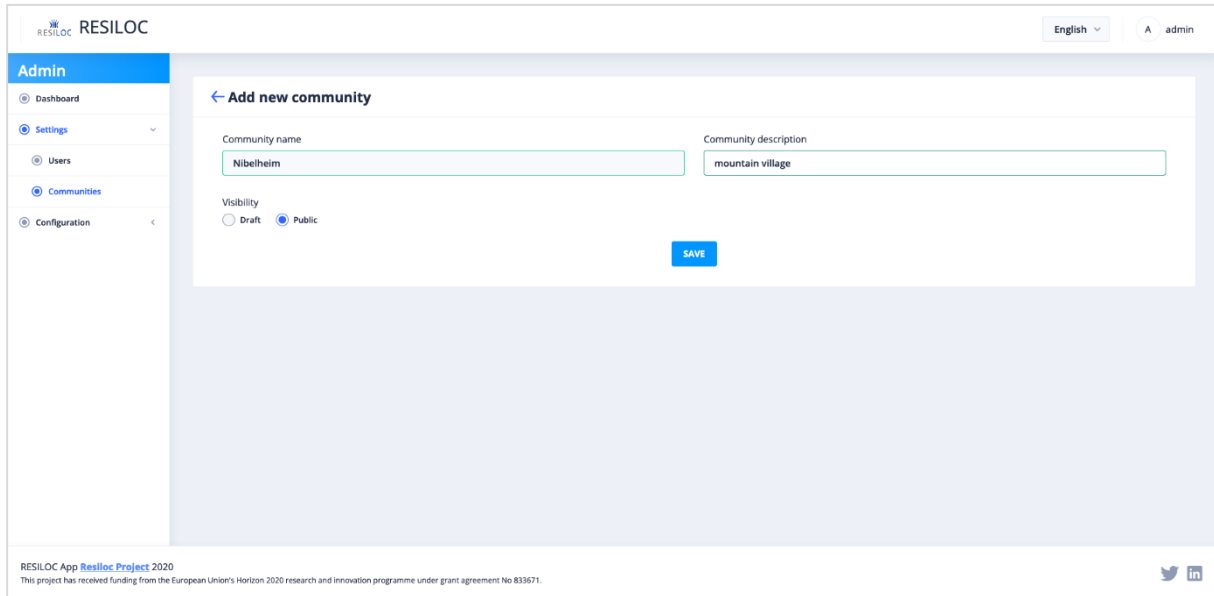
Edward, as RESILOC admin receives a request to add a new community that will participate in the RESILOC project. Once logged in the RESILOC Platform, Edward selects the "Communities" section.

In the "Communities" section, Edward is able to consult the list of the communities registered in the system and several related information (community name, visibility, date of creation into the system).



By clicking on the "+ ADD COMMUNITY" button, Edward accesses the page for the creation of a new community. The system now asks to Edward to enter a name and a brief description of the community along with the visibility selection.

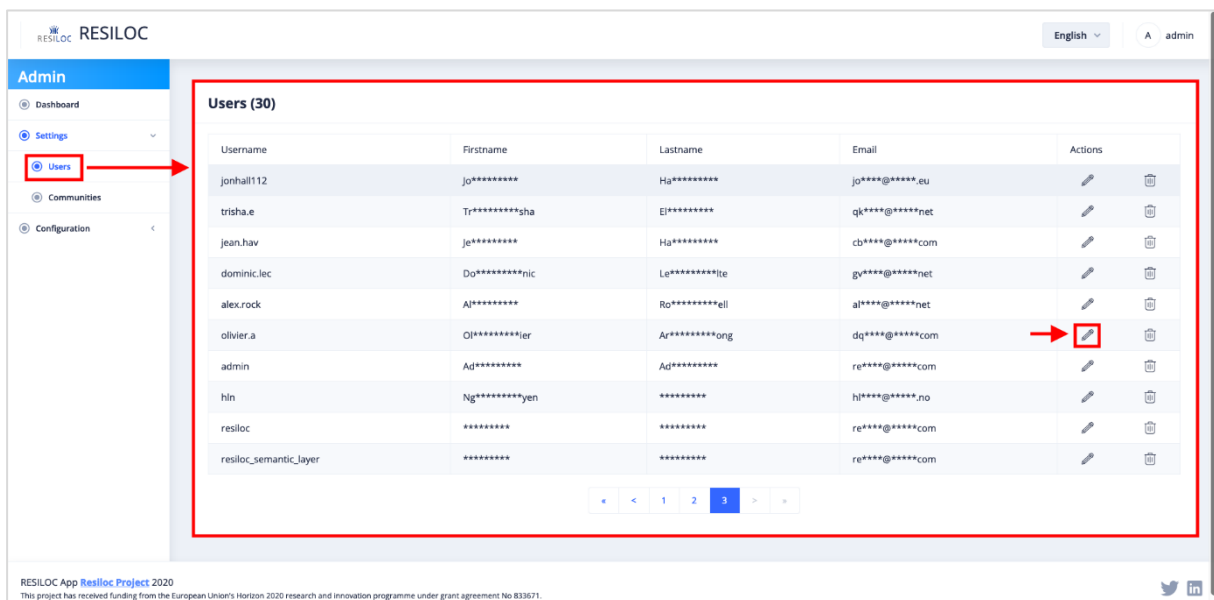
Edward now enters the name of the new community that will participate in the RESILOC project (e.g., Nibelheim), adds a short description and sets the visibility to 'Public' (making it visible to all users).























RESILOC App [Resiloc Project](#) 2020
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833671.

Once the community has been created, Edward can proceed to assign the role of “Community admin” to a user registered in the system and follower of the newly created community (e.g., Olivia).

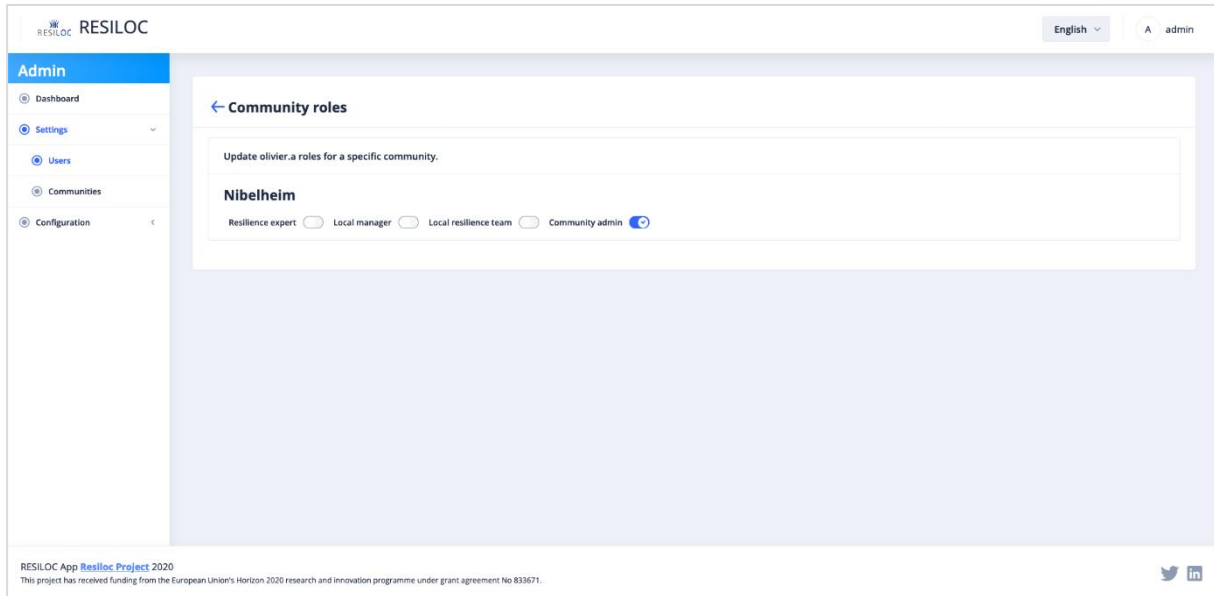
Once Olivia is found, Edward has the capabilities to access the user role edit page by selecting the pencil icon in the “actions” column.



Username	Firstname	Lastname	Email	Actions
jonhall112	Jo*****	Hg*****	jo****@****.eu	 
trisha.e	Tr*****sha	El*****	ql****@****.net	 
jean.hav	Je*****	Hg*****	cb****@****.com	 
dominic.lec	Do*****njc	La*****je	gv****@****.net	 
alex.rock	Al*****	Ro*****ell	al****@****.net	 
olivier.a	Ol*****ier	Ar*****ong	dq****@****.com	 
admin	Ad*****	Ad*****	re****@****.com	 
hin	Ng*****yen	*****	hl****@****.no	 
resiloc	*****	*****	re****@****.com	 
resiloc_semantic_layer	*****	*****	re****@****.com	 

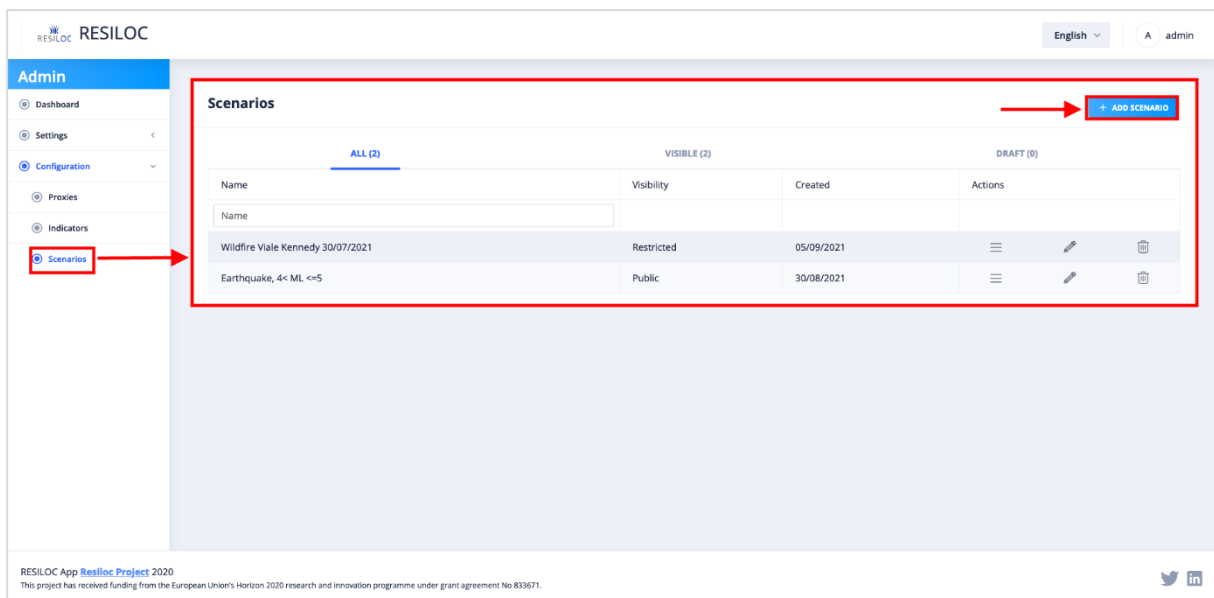
RESILOC App [Resiloc Project](#) 2020
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833671.

The role of "Community admin", for the Nibelheim community, can then be attributed to Olivia by switching on the corresponding button. Now Olivia can manage the roles of each Nibelheim community follower.



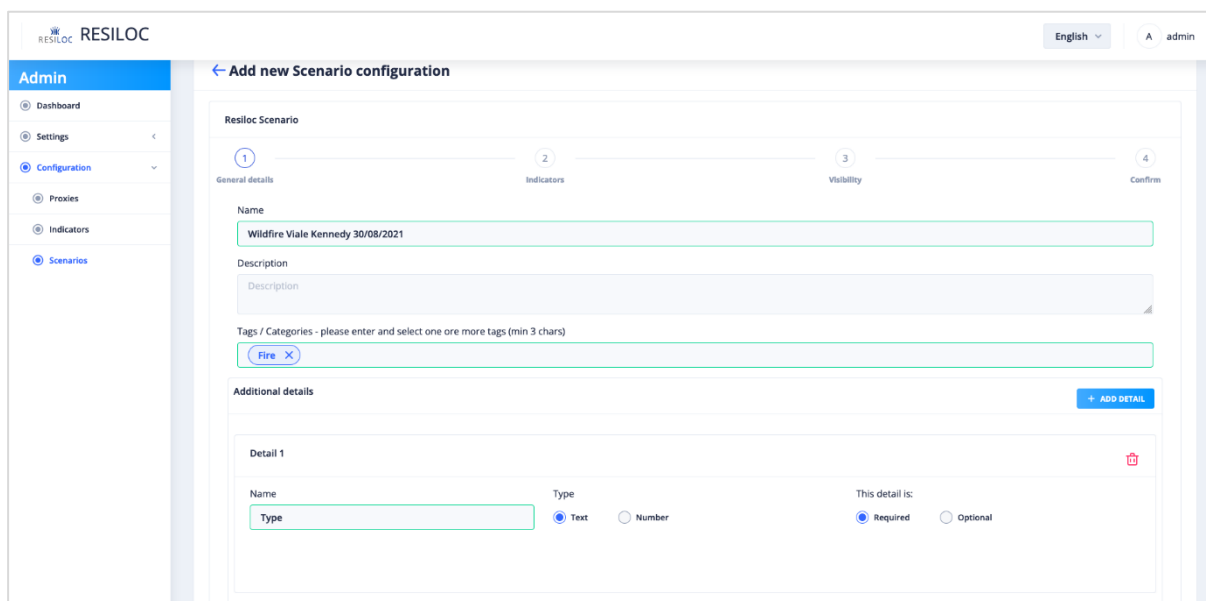
RESILOC Admin use case – new scenario creation

Edward has recently been notified to create a new wildfire scenario for RESILOC communities. So, he logs-in and navigates to Scenario section; here by clicking on the "+ ADD SCENARIO" button, he opens the page for the creation of a new scenario.

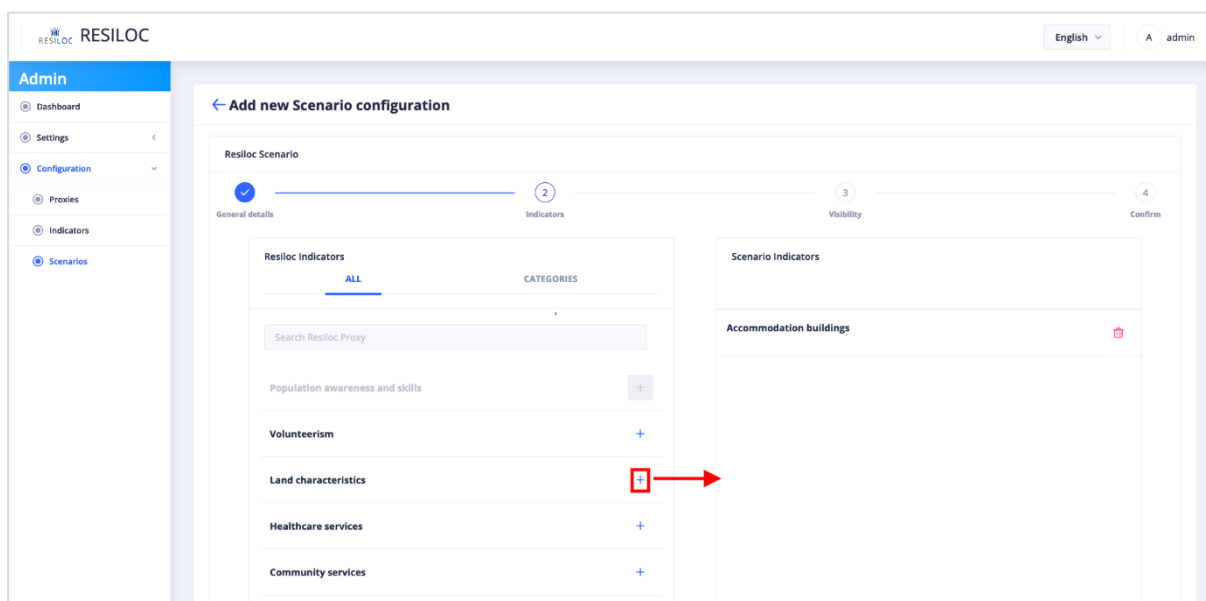


In the new scenario creation page, Edward enters a name and a brief description of the scenario. Below he proceeds to setting up some scenario related additional details.

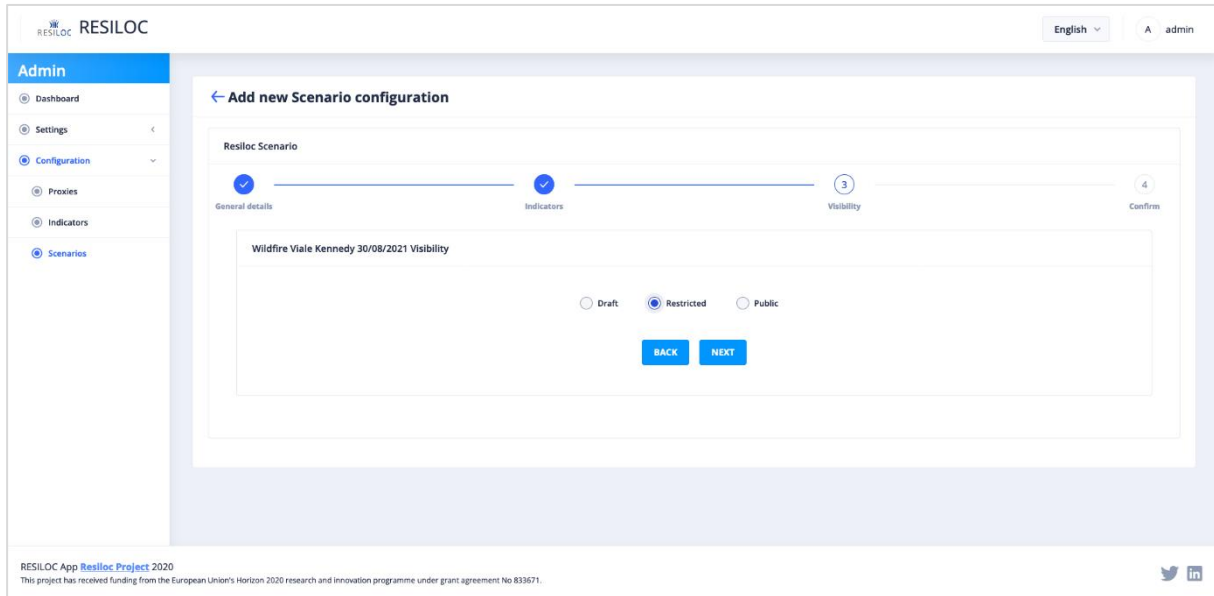
By clicking on the "+ ADD DETAIL" button he creates a new one for the scenario, entering the name, selecting the datatype and specifying if it will be required or not to the end-user. These additional details will be requested (optionally or mandatorily) during the scenario filling-in phase, the aim is to collect useful information for a richer description of the event.



In the following step, Edward selects one or more indicators deemed relevant for the given scenario from the RESILOC Indicators list, adding them to Scenario Indicators list.



After having completed the indicators' selection, Edward sets the scenario visibility and proceeds to confirm the newly defined scenario.

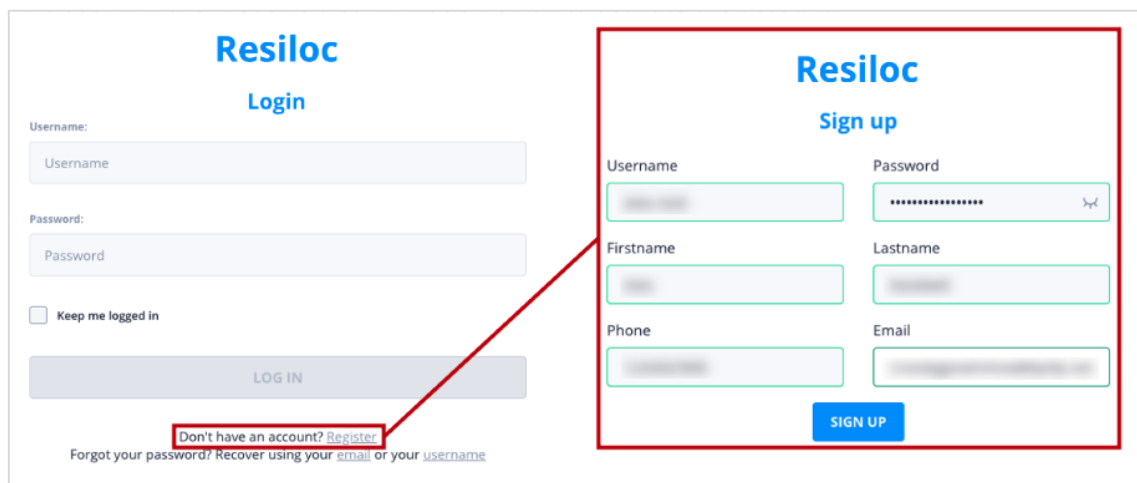


The wildfire scenario will then be added to the list of RESILOC scenarios and will be available for RESILOC communities.

B. RESILOC sign-up and sign-in

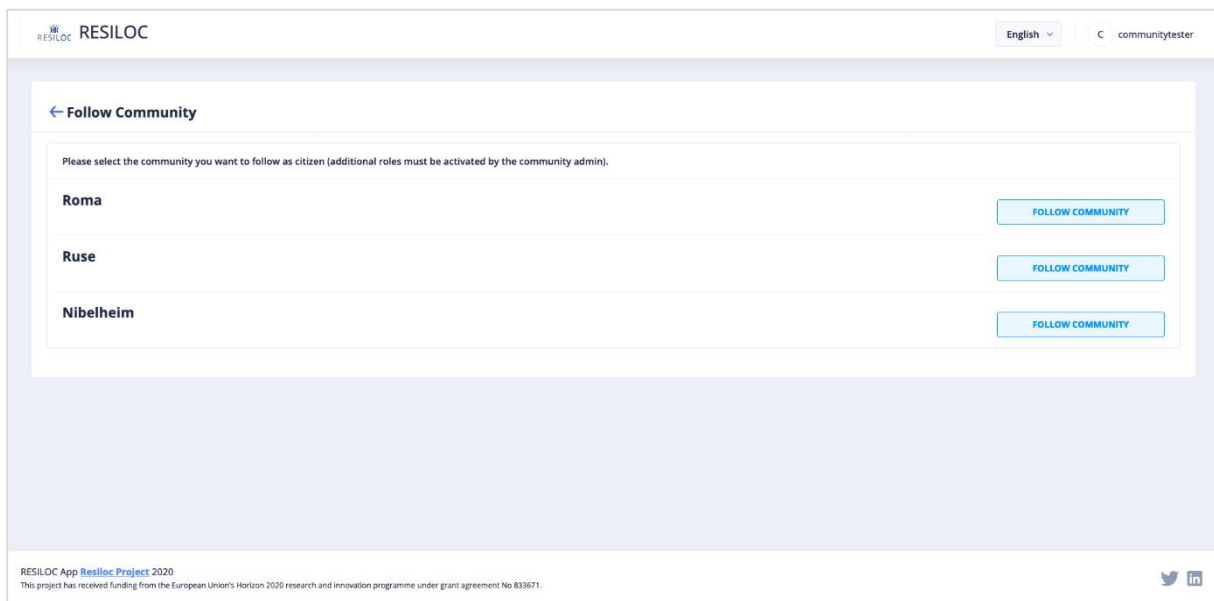
A resident of any community, having heard and read about the RESILOC project via several promotional campaigns, may decide to register on the RESILOC platform.

Accessing on the web application page, he/she can proceed with the registration using the dedicated form. The user will then receive a confirmation email to activate the account.

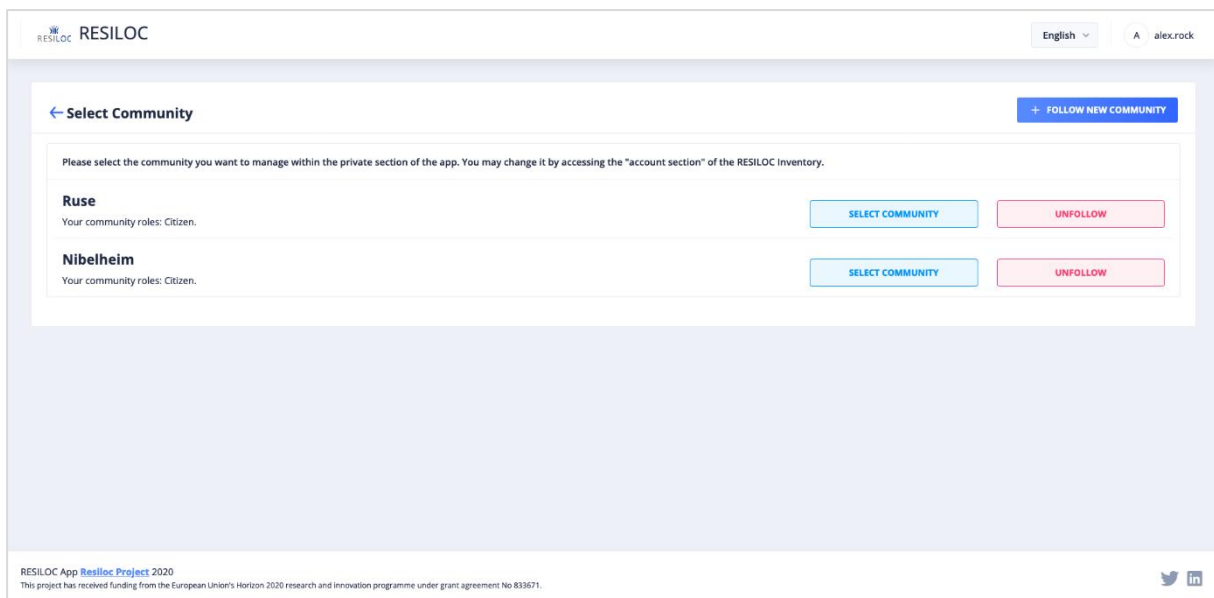


After the first logging in, the early step a non-admin user can take is to follow one of the RESILOC registered communities.

He/she can choose from a list one or more communities to follow; once the community is selected, the user is marked by the system as a "citizen" for that community.

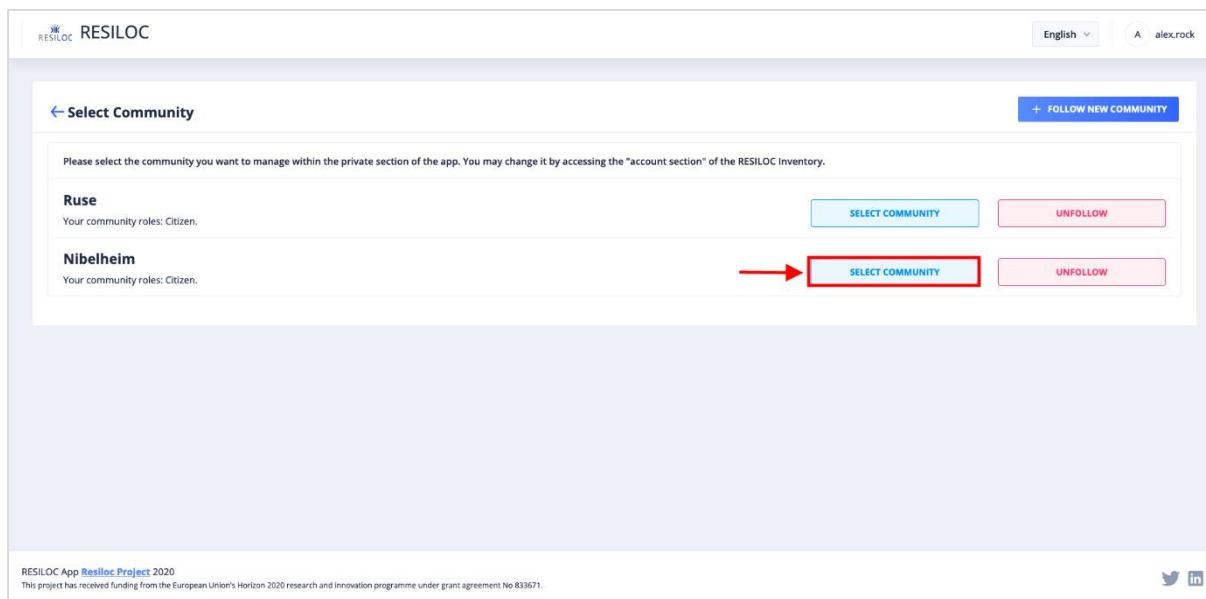


The users may also decide to follow more than one community; to do so, he/she uses the "follow new community" functionality and accesses the catalogue of all communities registered on RESILOC adding them to his list.

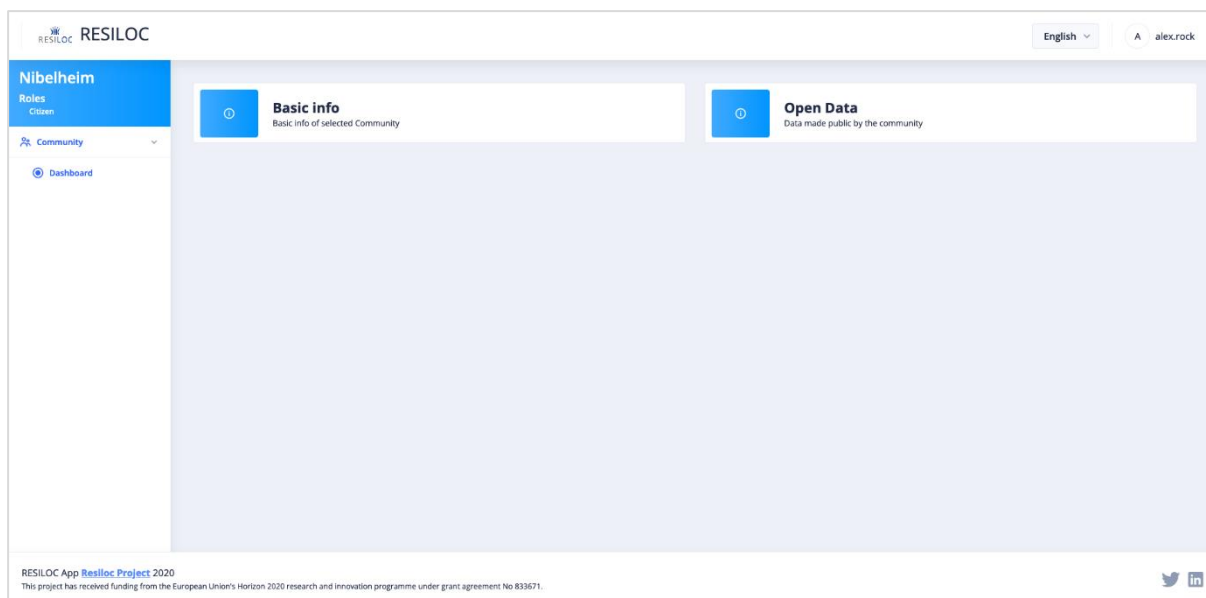


The user will access the catalogue of all communities registered on RESILOC; he/she can then select a new community and add it to his/her list of followed communities.

Now the user can decide which community he/she wants to access by selecting one of the communities he follows.



Once the user has selected the community, he/she wants to view, the system redirects the user to a management dashboard. The platform displays an overview of the available sections, depending on the user role. From the community dashboard a user will be able to access as a "citizen" to general information and open data made available by the community (marked as "Public").



RESILOC Citizen use case – user registration and becoming a community citizen

Alex is a resident of the Nibelheim community. Having heard and read about the RESILOC project via several promotional campaigns run by his community, he decided to register on the RESILOC platform.

Once he accesses the web application page, Alex can proceed with the registration using the dedicated form. The user will then receive a confirmation email to activate the account.

Resiloc

Login

Username:

Password:

☐ Keep me logged in

LOG IN

Don't have an account? [Register](#)

Forgot your password? Recover using your [email](#) or your [username](#)

Resiloc

Sign up

Username

 alex.rock

Password

Firstname

 Alex

Lastname

 Rockbell

Phone


 1234567890

Email

 rrxosbjgmahvhoe@bptfp.net

SIGN UP

After the first logging in, the early step Alex can take is to follow one of the RESILOC registered communities, he can choose from a list one or more communities to follow. Once the community is selected Alex is marked by the system as a "citizen" for that community.


RESILOC

English

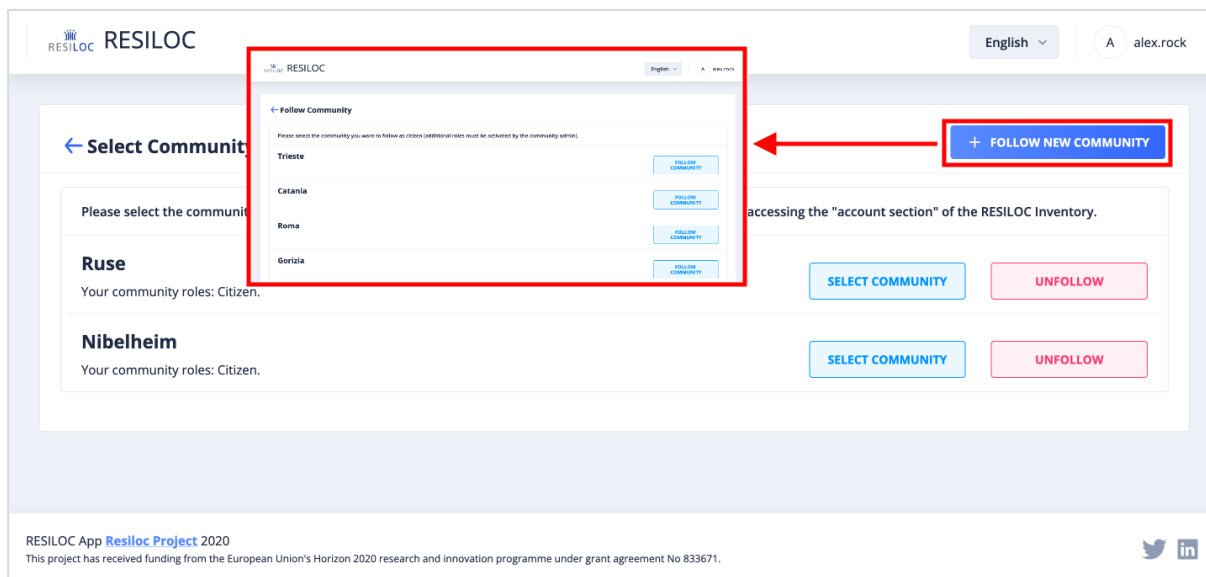
A alex.rock

Follow Community

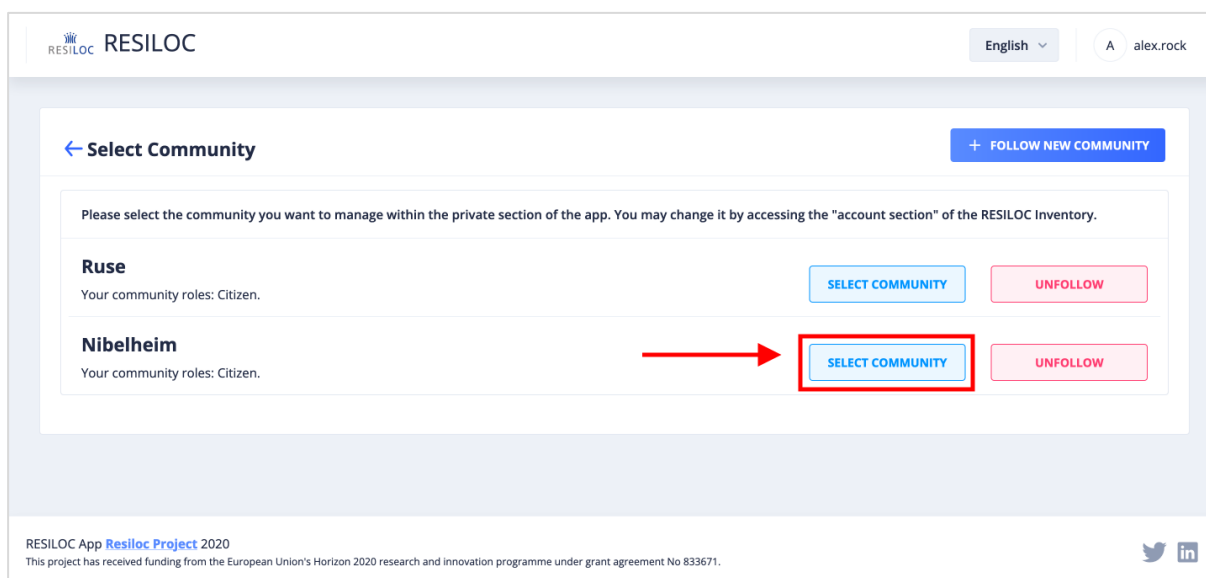
Please select the community you want to follow as citizen (additional roles must be activated by the community admin).

Trieste	FOLLOW COMMUNITY
Catania	FOLLOW COMMUNITY
Roma	FOLLOW COMMUNITY
Gorizia	FOLLOW COMMUNITY

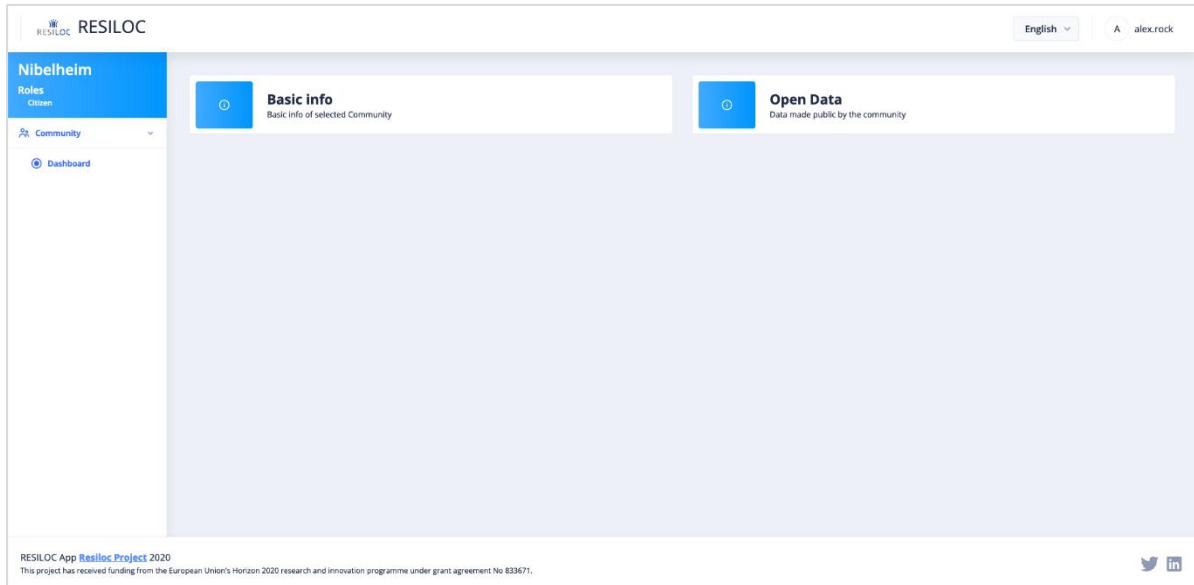
Alex may also decide to follow more than one community; to do so, he uses the "Follow new community" functionality and accesses the catalogue of all communities registered on RESILOC adding them to his list.



Now Alex can decide which community he wants to access by selecting one of the communities he follows (e.g., Nibelheim).



From the dashboard of Nibelheim community Alex will be able to access as a "citizen" to general information and open data made available by the community (marked as "Public").

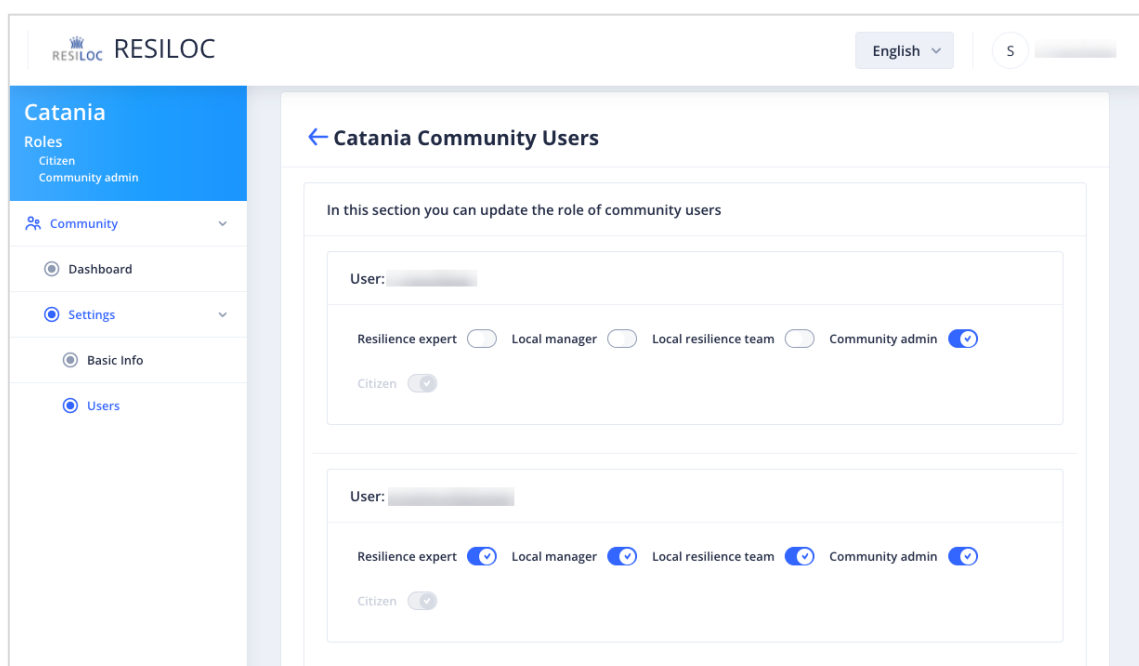


C. RESILOC Community's roles assignment

The management of community's user roles in the RESILOC Inventory is performed via a dedicated module that is accessible only to users with the 'Community admin' role.

The "Community admin" is the role assigned by the RESILOC admin to the user who has the responsibility to administer the users pertaining to their community; the "Community admin" assigns roles and therefore competences and responsibilities to the users of his community, taking into account their skills and knowledge.

To manage the roles of the community users, the "Community admin" uses the functionalities provided by "Users" item in Settings menu. The system displays the list of community users (i.e., "citizen" users); once the designated user has been identified, switching on one or more buttons, the corresponding role will be assigned.

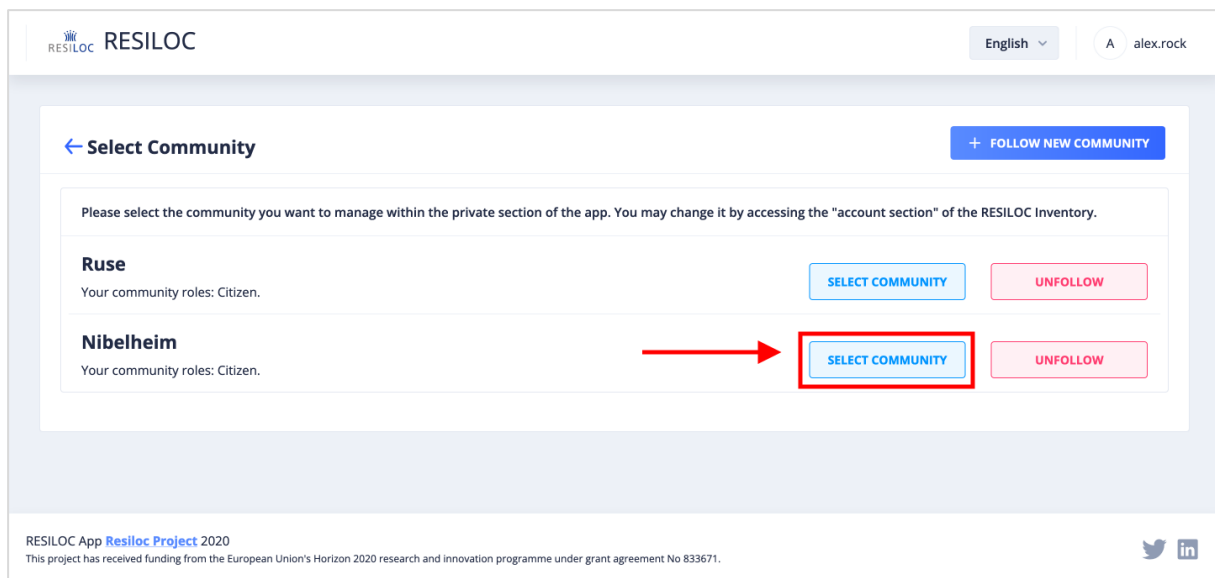


Each user can have one or more roles assigned, according to his skills and the availability of community resources.

RESILOC Community admin use case – roles assignment

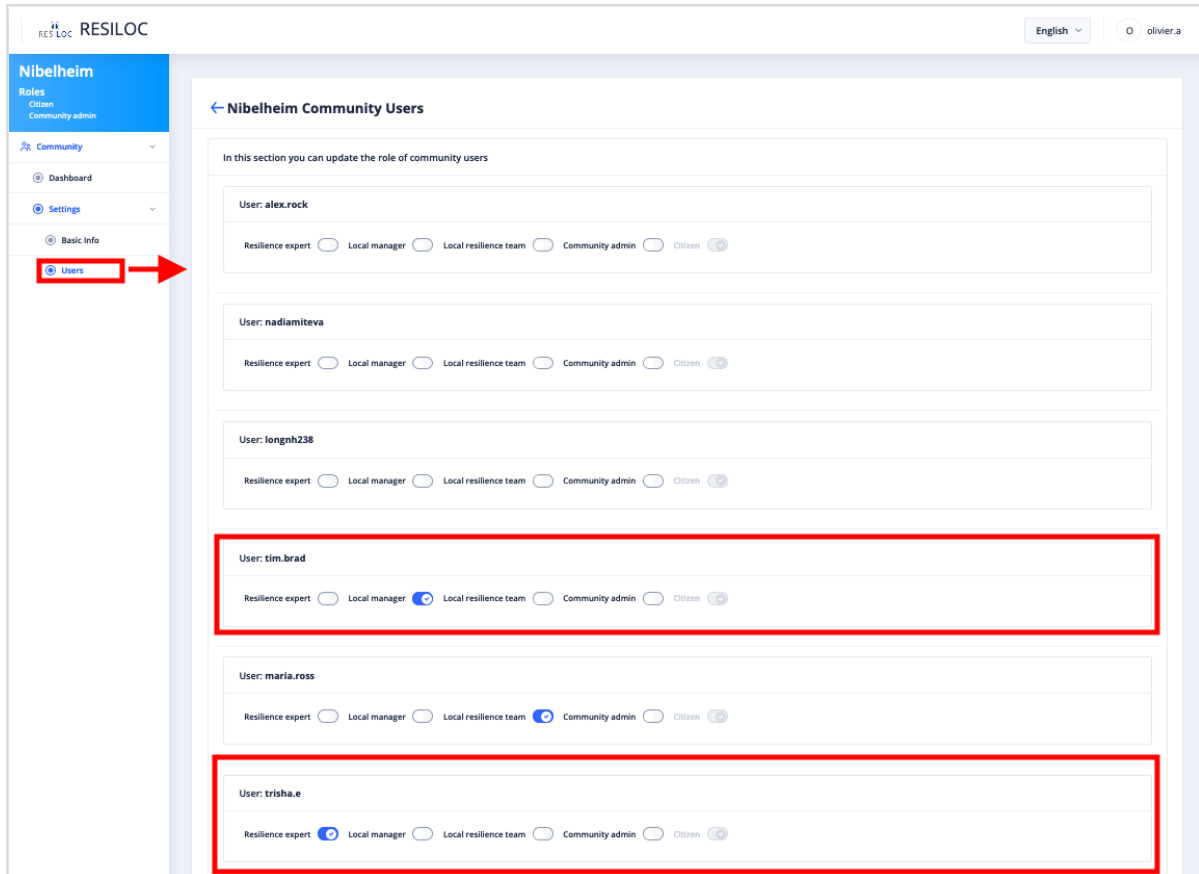
Olivia is the "Community admin" of Nibelheim's community, due to workforce adaptation and the hiring of new personnel, two new employees of the municipality (Tim and Trisha) are appointed as "local manager" and "resilience expert".

After logging in with her credentials, Olivia selects the Nibelheim community among those she follows.



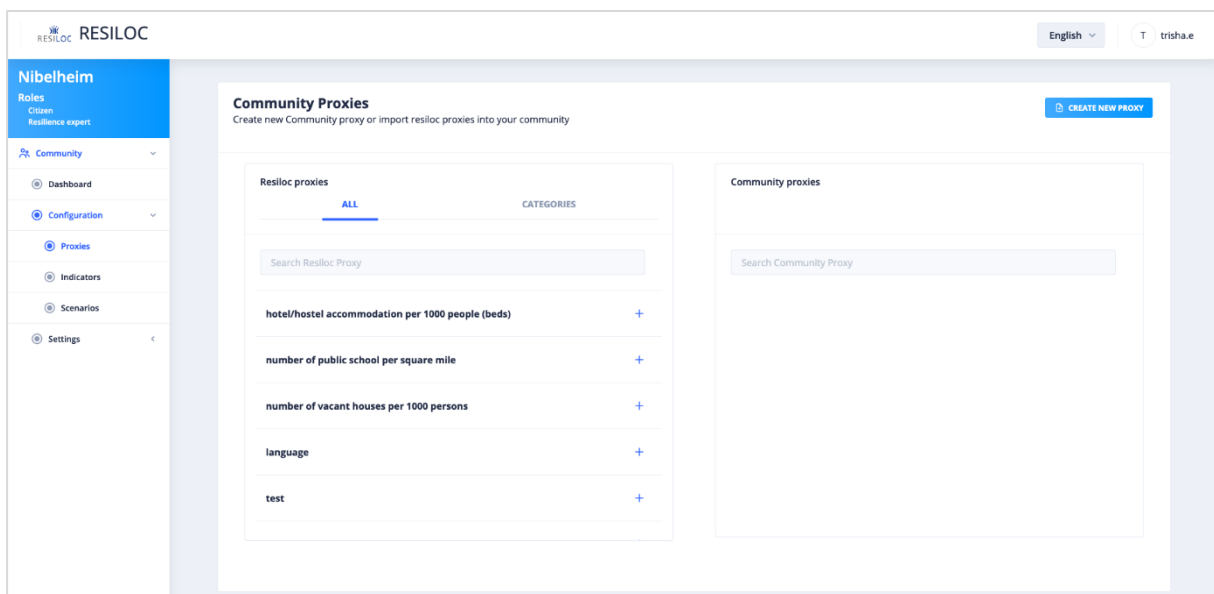
Olivia browses to the user section and accesses the role edit page in order to update the role of users Tim and Trisha. Switching on the corresponding button, she assigns the role of "local manager" and "resilience expert" respectively to Tim and Trisha, already followers of the Nibelheim community.

Afterwards, Tim and Trisha will have the capabilities to carry out their new role related tasks.



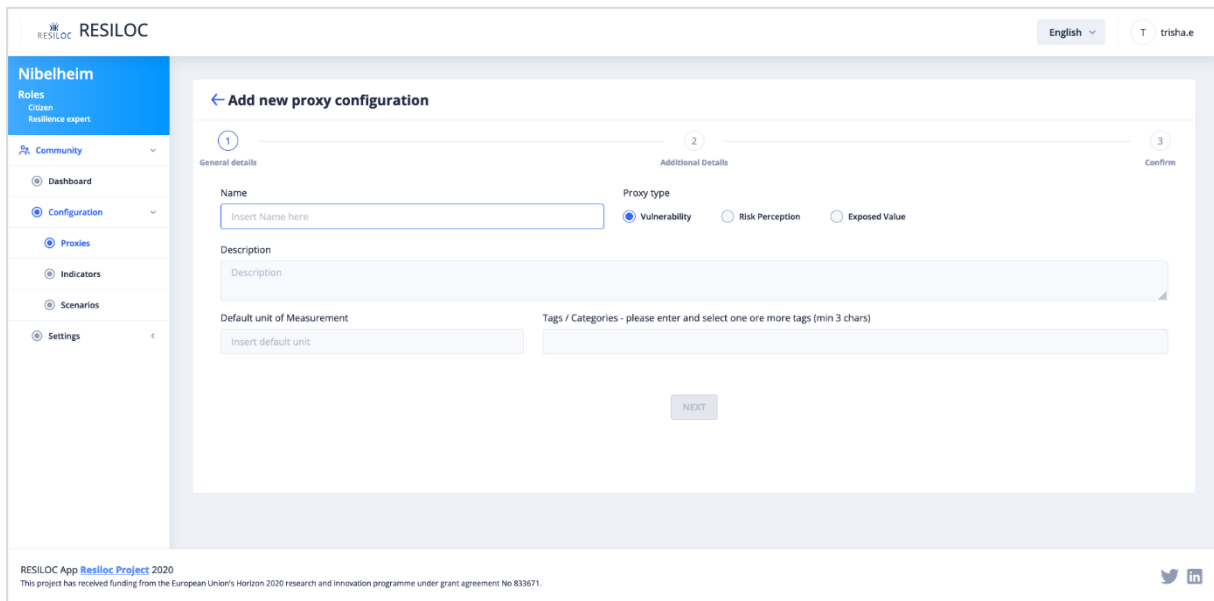
D. RESILOC community's "Configuration" menu

In the configuration menu, "Proxies" section, the Resilience expert, Local manager and LRT members of a community are able to consult the list of proxies available into the system (the so called "RESILOC proxies"); so, Resilience expert and Local manager can add some proxies to community's proxies list. Once imported, the community proxies could be edited by clicking on the pencil icon, or eventually removed if necessary.



By clicking on the "create new proxy" button, the Resilience expert and Local manager may access the page for the creation of a new community proxy. The system asks to enter a name and a brief description of the proxy along with the proxy's type selection, unit of measurement definition and tags association.

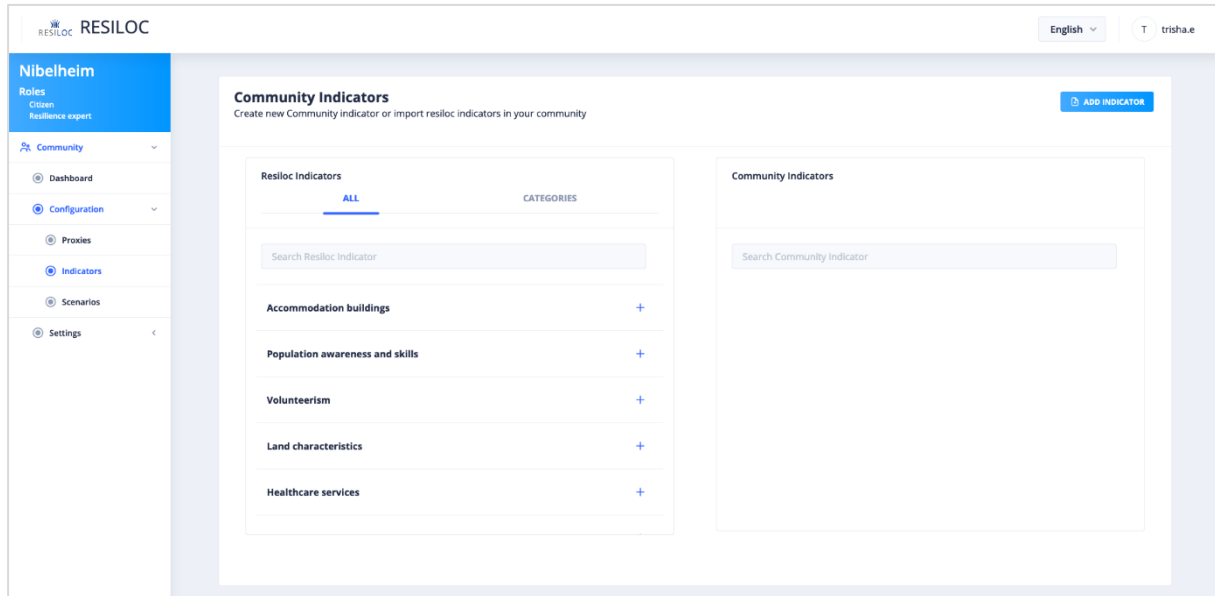
The creation of a new community proxy follows the same path already described for the creation of a new proxy from **RESILOC admin** side/point of view (page vi).



In the "Indicators" section, the Resilience expert and the Local manager are able to consult the list of indicators available into the system (the so called "RESILOC Indicators"), so they can add some indicators to community's indicator list. Once imported, the community indicator could be edited clicking on the pencil icon, or eventually removed if necessary.

By clicking on the "add indicator" button, the Resilience expert and Local manager may access to new community indicator creation page.

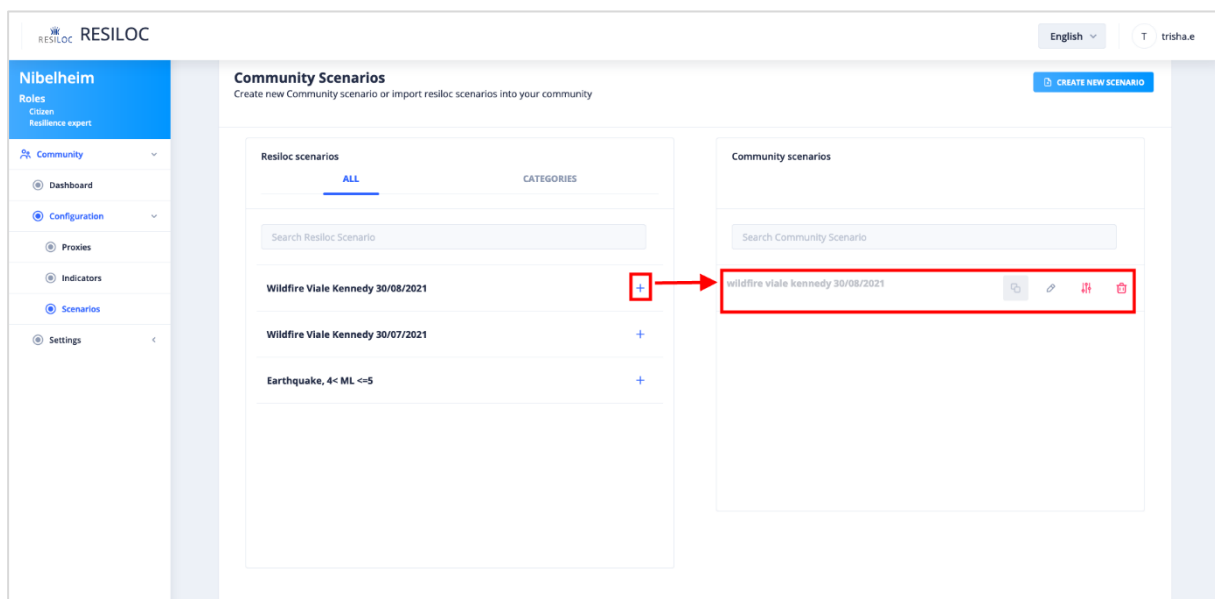
The creation of a new community indicator will follow the same path already described for the creation of a new indicator from **RESILOC admin** side (page ix).



In the "Scenarios" section, the Resilience expert and the Local manager are able to consult the list of scenarios added into the system (the so called "RESILOC scenarios"), so they can add some scenarios to community's scenario list. Once imported, the community scenario could be edited clicking on the pencil icon, whilst metadata and proxies' targets could be set up by clicking on equaliser icon.

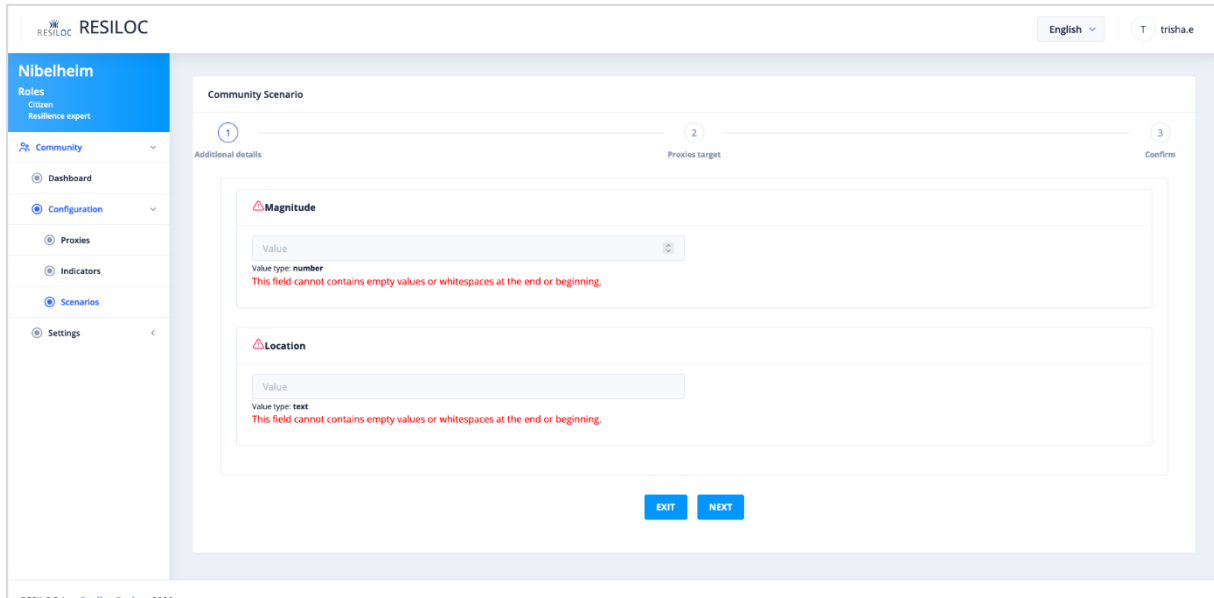
Using the "create new scenario" button, the Resilience expert may access to new community scenario creation page.

The creation of a new community scenario will follow the same path already described for the creation of a new scenario from **RESILOC admin** side (page xi).



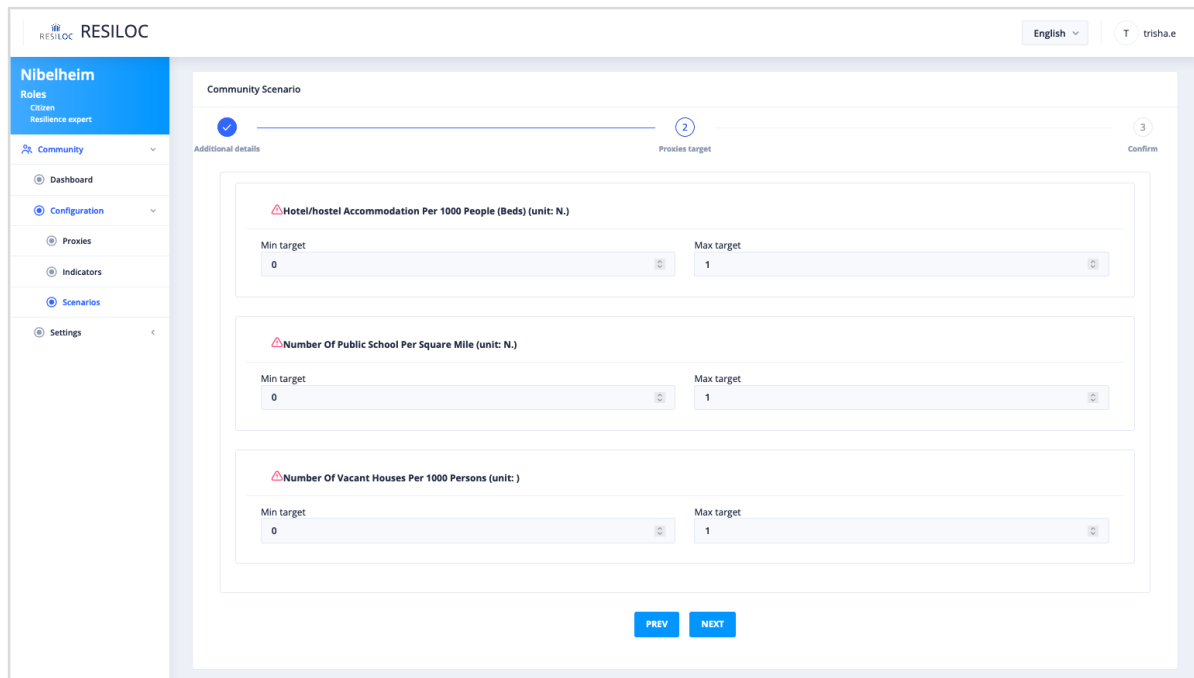
By clicking on the equalizer button, the system give access to scenario setting up page; here users can enter the required additional details and appropriate proxies' targets for their own community.

In the metadata section the user has to enter the required metadata information for the specific scenario; mandatory metadata are marked with an alert symbol, once filled in they are flagged with a green check. Optional metadata are always marked with the green tick.



The User can then set the Min and Max target values for each proxy involved into scenario selected; these are mandatory to carry out the scenario assessment.

After setting the Min and Max target values, the proxy will be marked with a green tick.



RESILOC

English T trisha.e

Nibelheim
Roles
Citizen
Resilience expert

Community

Dashboard

Configuration

Proxies

Indicators

Scenarios

Settings

Community Scenario

Additional details

Proxies target

Confirm

Hotel/hostel Accommodation Per 1000 People (Beds) (unit: N.)

Min target 0 Max target 1

Number Of Public School Per Square Mile (unit: N.)

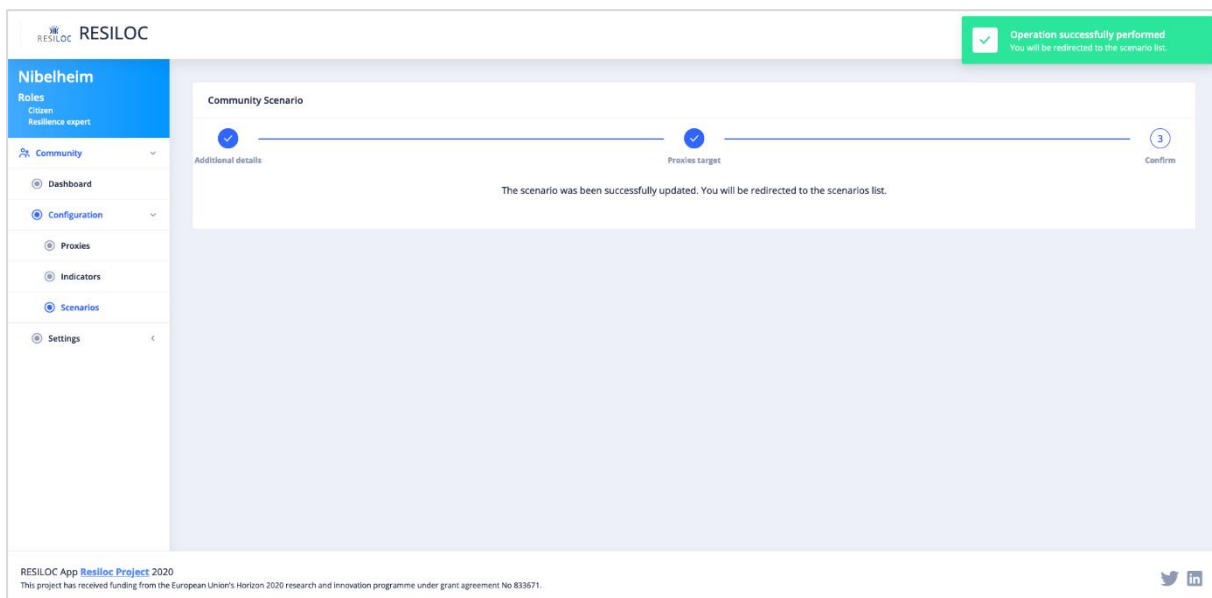
Min target 0 Max target 1

Number Of Vacant Houses Per 1000 Persons (unit:)

Min target 0 Max target 1

PREV NEXT

Once confirmed, the system will save the scenario and redirect the user to the list of community scenarios.



RESILOC

Operation successfully performed
You will be redirected to the scenarios list.

Nibelheim
Roles
Citizen
Resilience expert

Community

Dashboard

Configuration

Proxies

Indicators

Scenarios

Settings

Community Scenario

Additional details

Proxies target

Confirm

The scenario was been successfully updated. You will be redirected to the scenarios list.

RESILOC App Resilic Project 2020
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833671.

Twitter LinkedIn

RESILOC LRT members use case

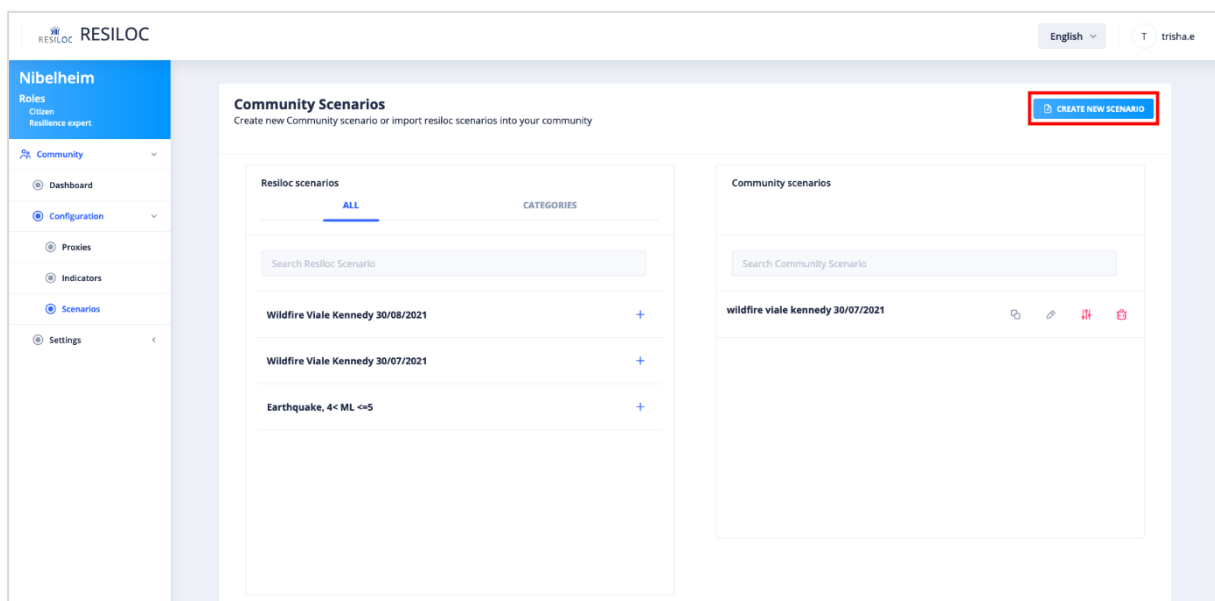
Jean, Maria and Dominic, as Local Resilience Team members, have full access to the Configuration menu; the LRT role allow them to consult all the available community information. They can access to all the details of proxies, indicators, and scenarios, but they do not possess the capabilities that permit them to manipulate these elements/items.

Doing so, allows them to tap into the information they need to provide support to local manager and resilience expert.

RESILOC resilience expert use case

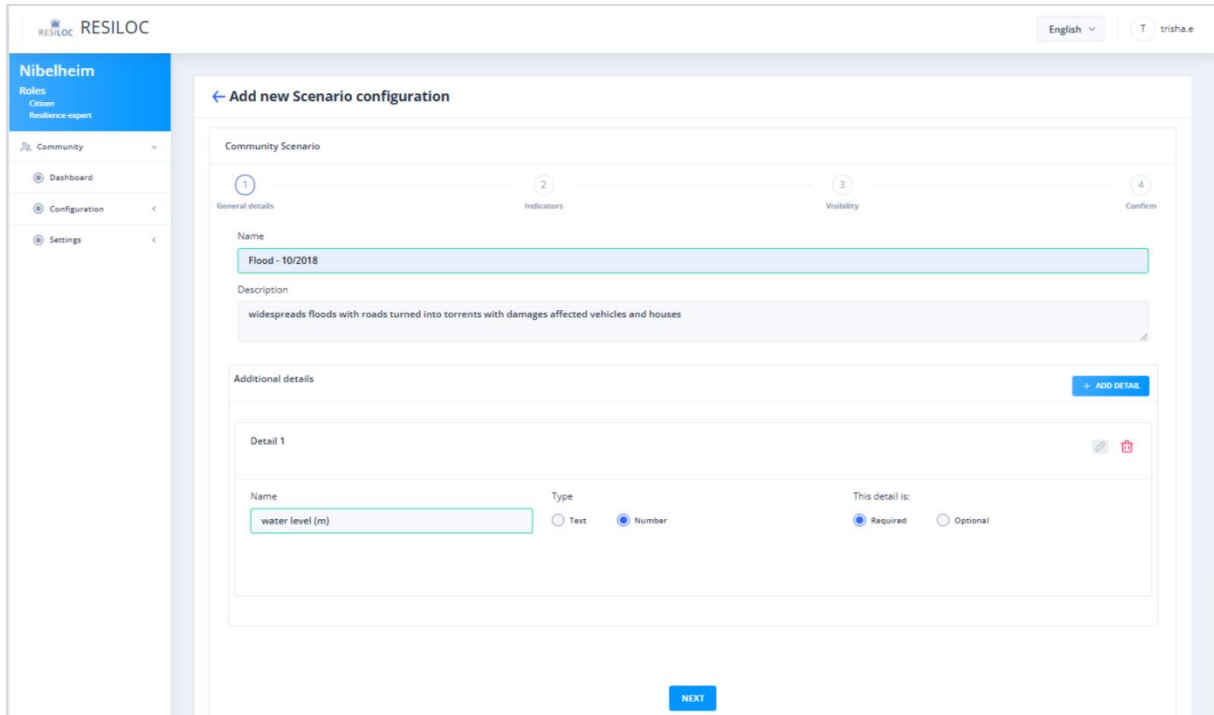
Trisha has recently been appointed as "resilience expert" for the Nibelheim community; following the occurrence of a flood, she plans to create a new flood scenario in the platform, in order to have an additional way to assess the resilience of the Nibelheim community against such events.

Trisha accesses to the RESILOC platform and navigates to Scenario section; here by clicking on the "CREATE NEW SCENARIO" button, she opens the page for the creation of a new scenario.

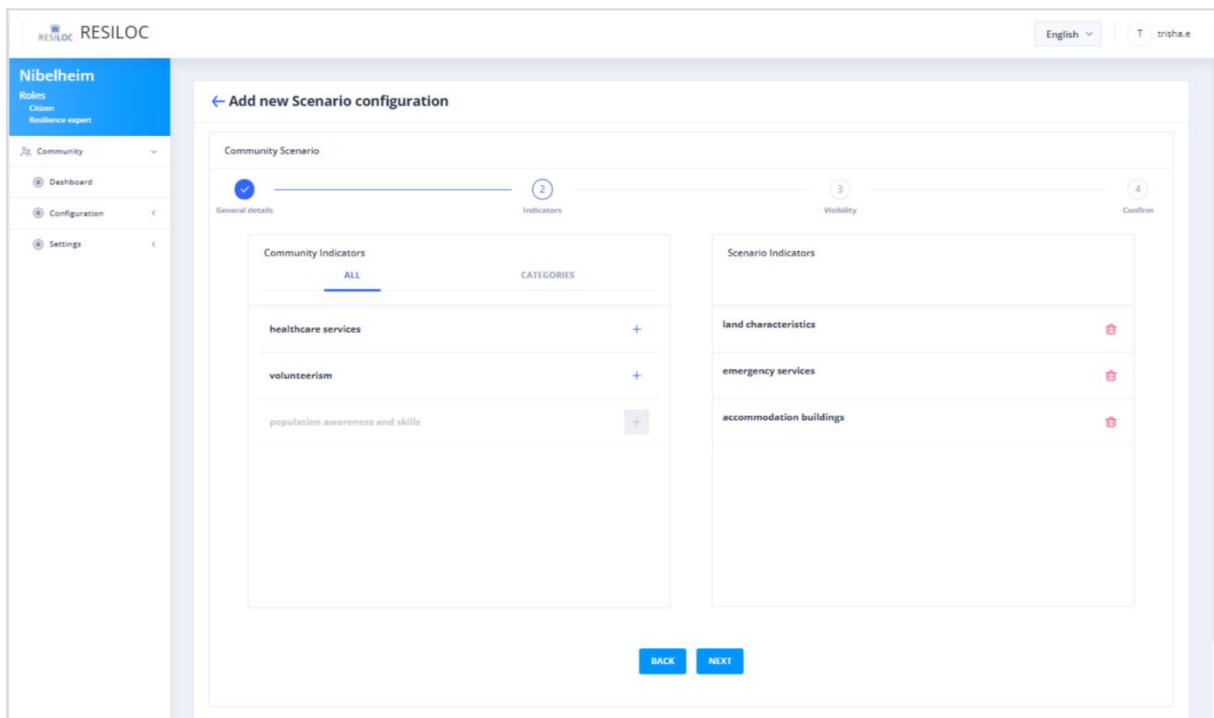


In the new scenario creation page, Trisha enters a name and a brief description of the scenario. Then she proceeds to setting up some scenario related additional details; by clicking on the "+ ADD DETAIL" button creates a new one for the scenario, entering the name, selecting the datatype and specifying if it will be required or not to the end-user.

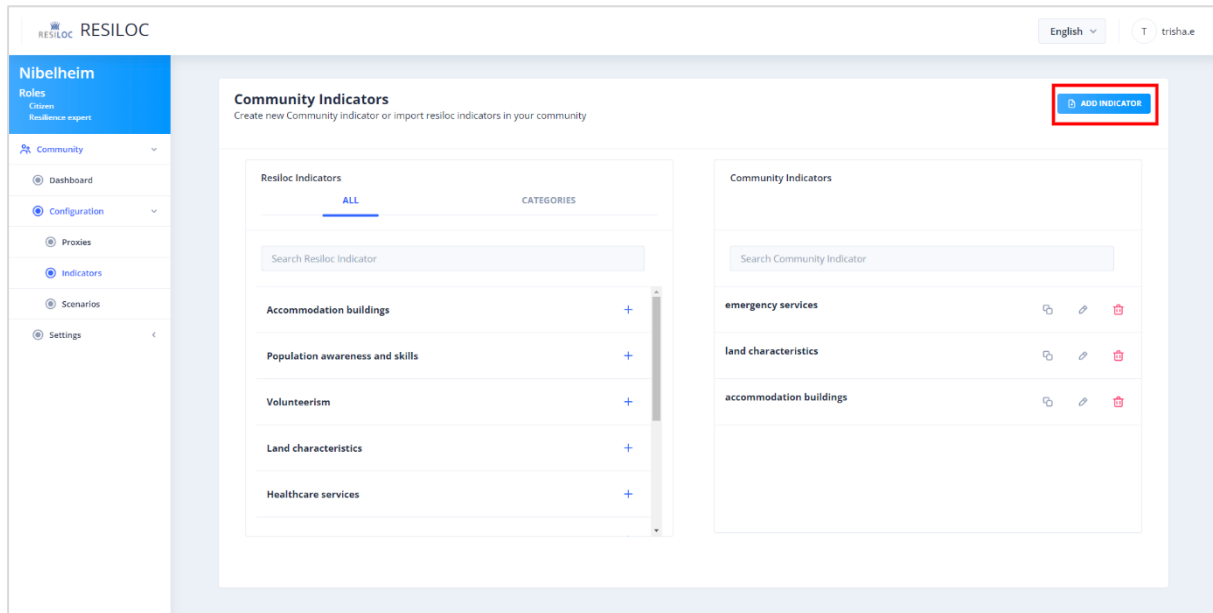
This information will be requested (optionally or mandatorily) during the scenario filling-in phase, the aim is to collect useful information for a richer description of the event.



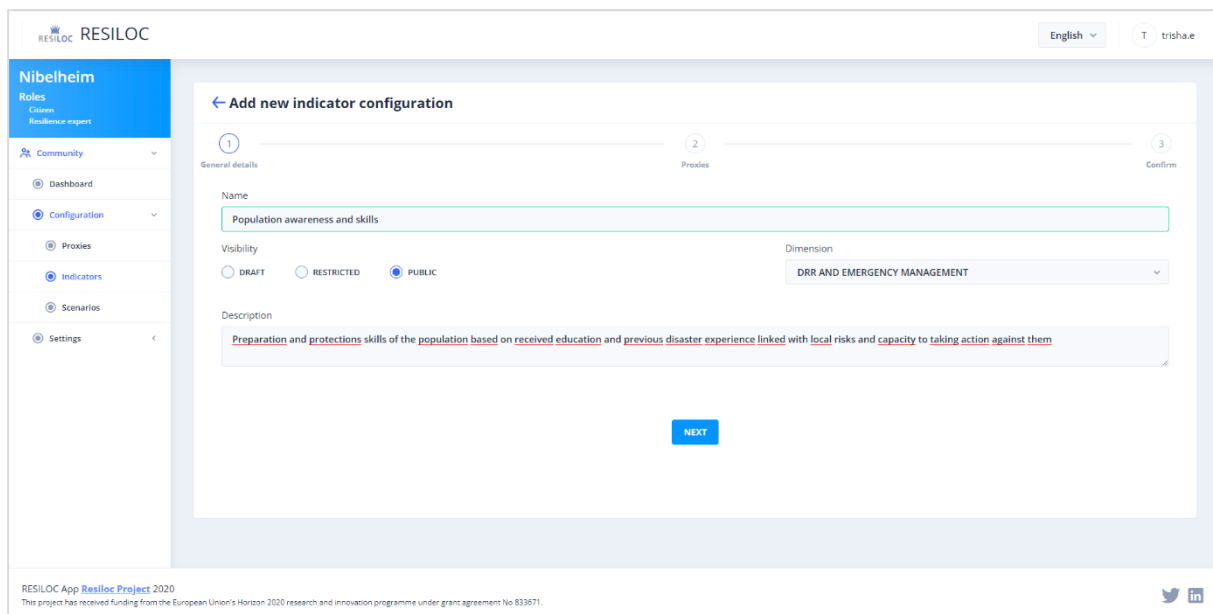
During the scenario indicators selection, Trisha wants to include an indicator that is not currently available in the list of Community indicators, so she proceeds to create it. Once ready, it can be selected and included by Trisha in the list of scenario indicators.



Since Trisha wants to include in the scenario an indicator that is not currently available in Community Indicators list, she moves to Indicators section to create it. By clicking on the "+ ADD INDICATOR" button, Trisha opens the page for the creation of a new indicator.

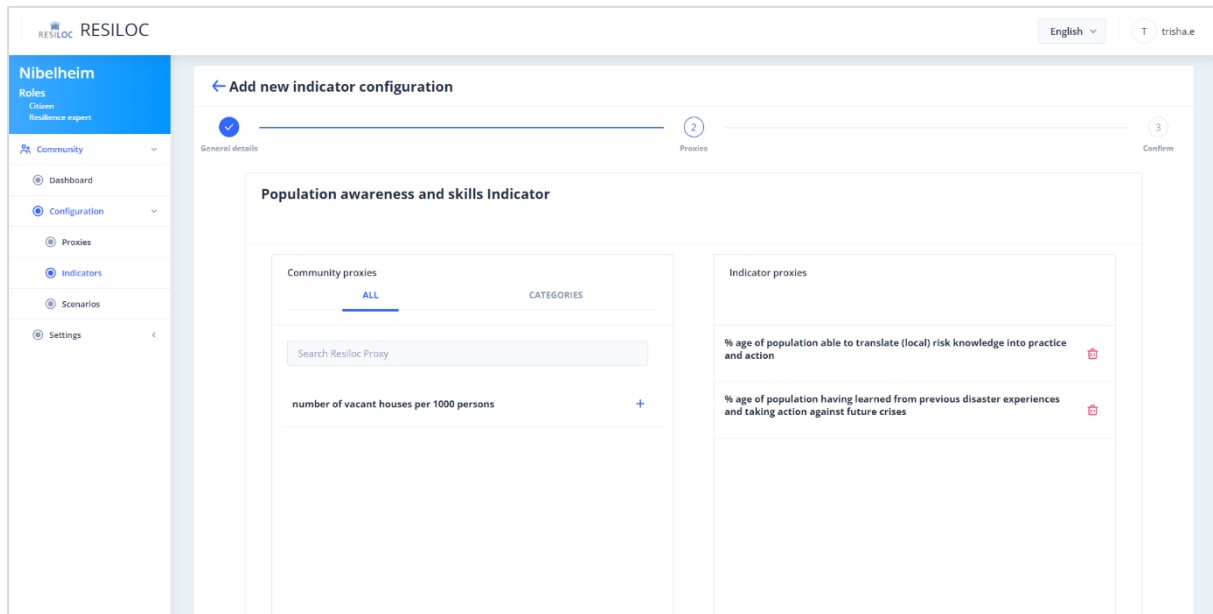


In the new indicator creation page, Trisha enters the name and a brief description of the indicator needed; then she chooses the indicator's visibility and selects the right dimension.



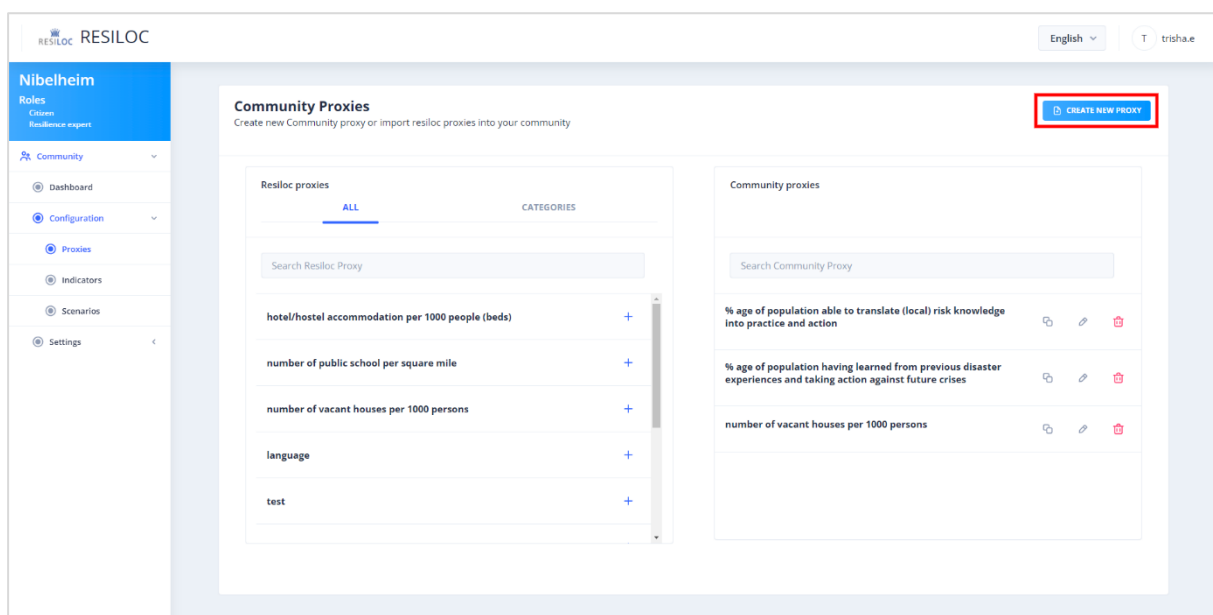
Once the general details section has been compiled, Trisha proceeds to select the proxies involved into indicator calculation. By clicking on "+" she selects from a list of available proxies those she wants to use and adds them to the Indicator's proxies list.

During this process Trisha interacts several times with Tim to choose the appropriate proxies for the indicator.

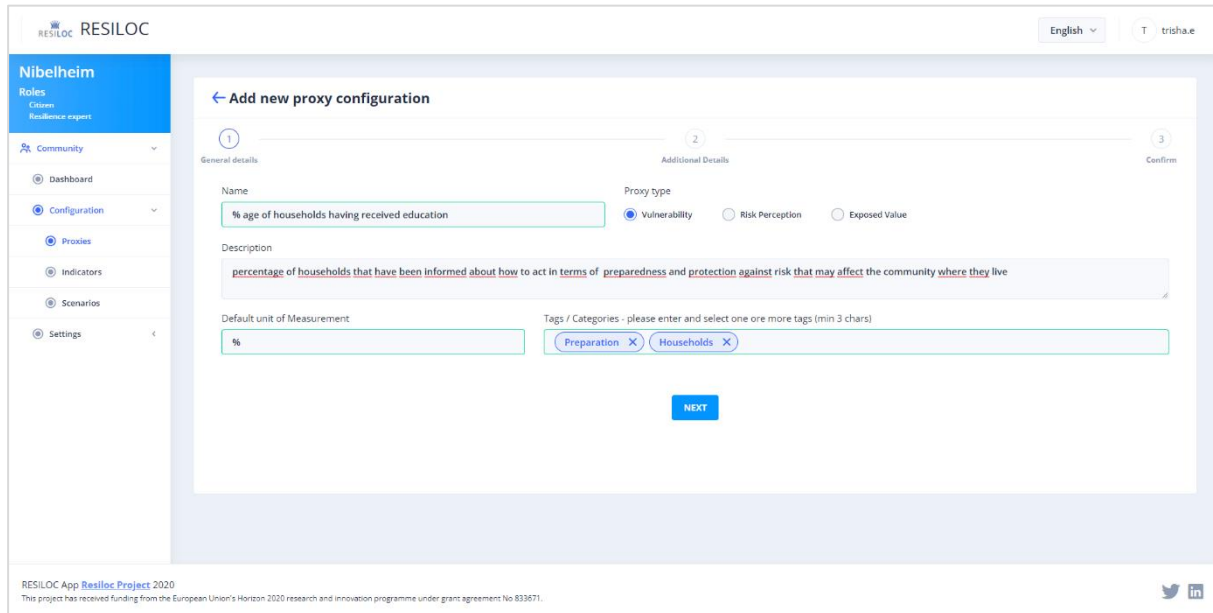


Consulting the list of available proxies (the so called "community proxies") Trisha and Tim notice that is valuable to add some new proxy to community's proxies list. So, Trisha, moves to proxies' section and search from the RESILOC proxies list those that are comparable/equivalent to desired ones. Couldn't find one of the desired/suitable proxy, Trisha decides to create a new one.

By clicking on the "create new proxy" button, she accesses the page for the creation of a new community proxy.



On the first page, Trisha enters a name and a brief description of the proxy, selects the proxy type, defines the unit of measurement, and associates some tags.

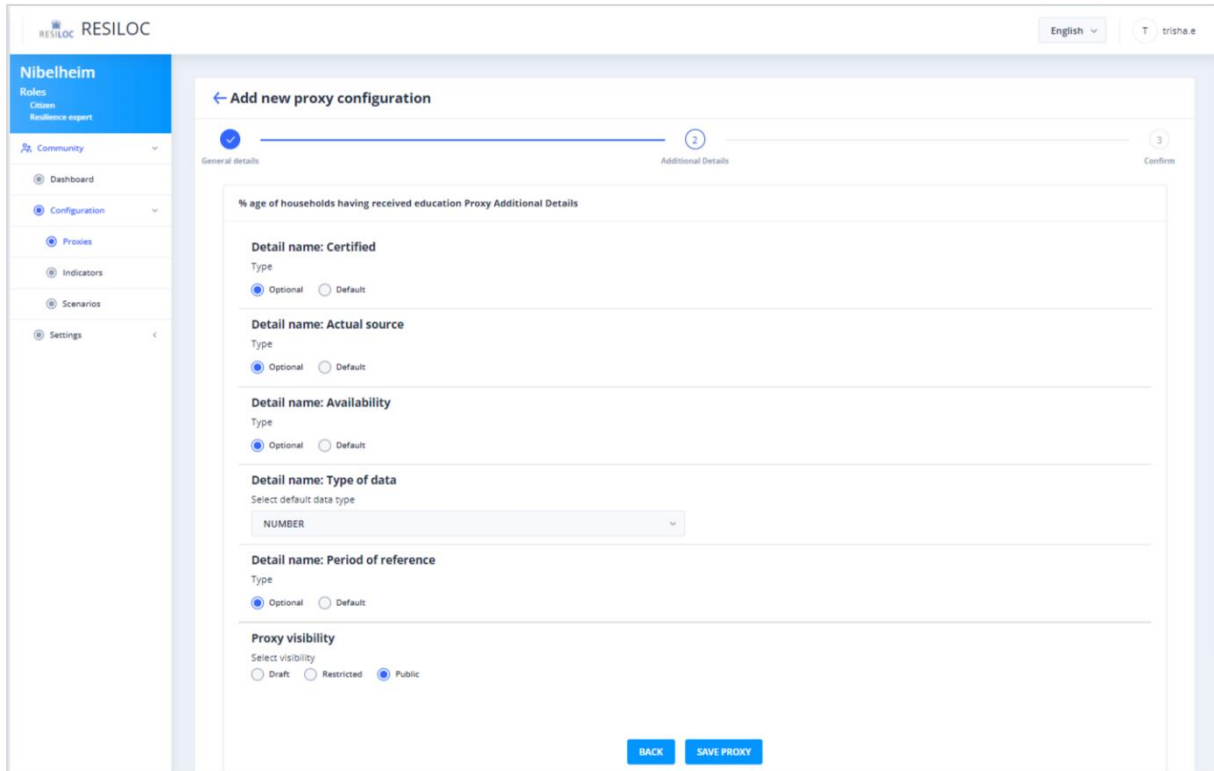


Once the general info section has been completed, Trisha can set up some proxy related additional details:

- Certified – specifies whether the data is issued by a certified entity, i.e., obtained from an official source.
- Actual source – specifies the actual source of data entity/body/organization that provides the data.
- Availability – specifies the data availability (measured or derived).
- Type of data – specifies the data typology, e.g., number or string.
- Period of reference – period of data validity, e.g., the data is surveyed annually so it is valid for 1 year.

These additional details could be:

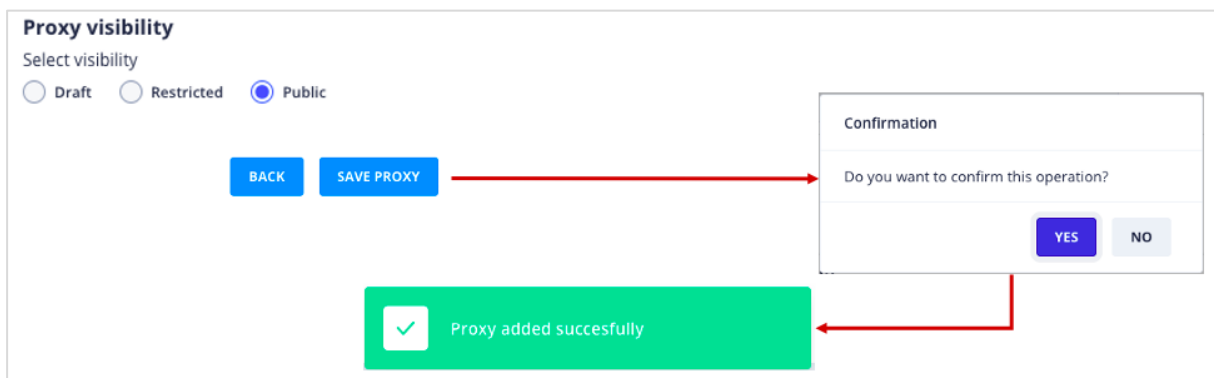
- Optional: the filling-in of the specific additional detail is left to the end user (will not be mandatory).
- Default: gives to Edward the faculty to set a default value for the additional detail considered (this will be however modifiable).



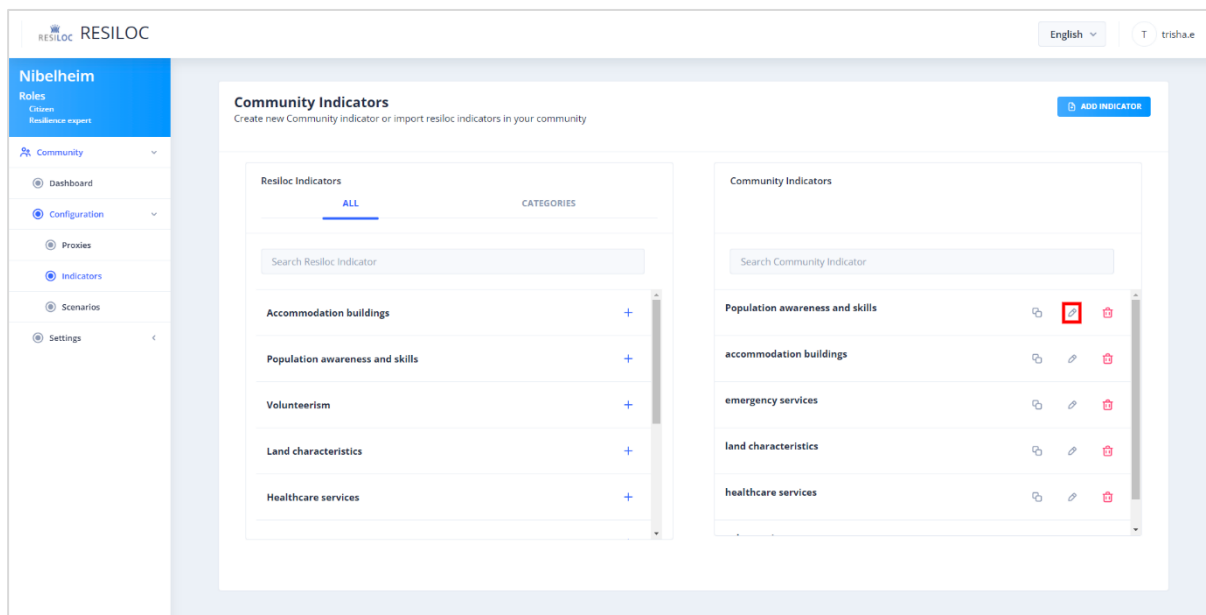
Trisha finalizes the procedure selecting the proxy's visibility and clicking on "SAVE PROXY" button.

- Draft – not visible (admin only).
- Restricted – visible only to resilience experts, local manager, and LRT.
- Public – visible to all, including citizens (open data).

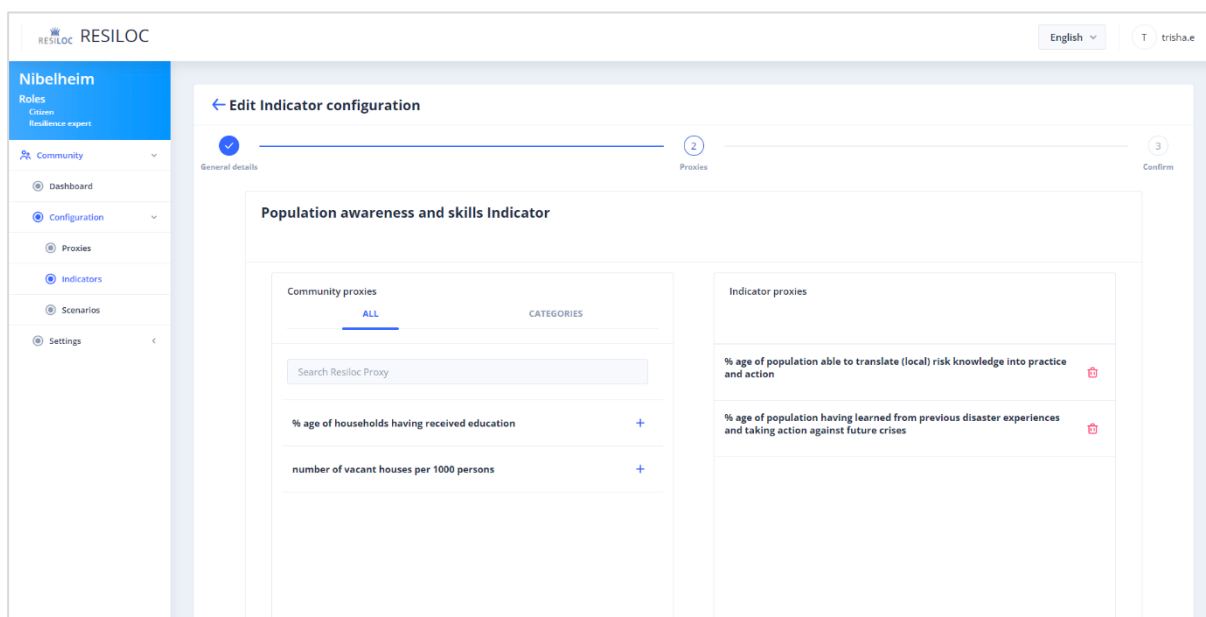
Once confirmed, the system saves the proxy and adds it to the list of available community proxies.



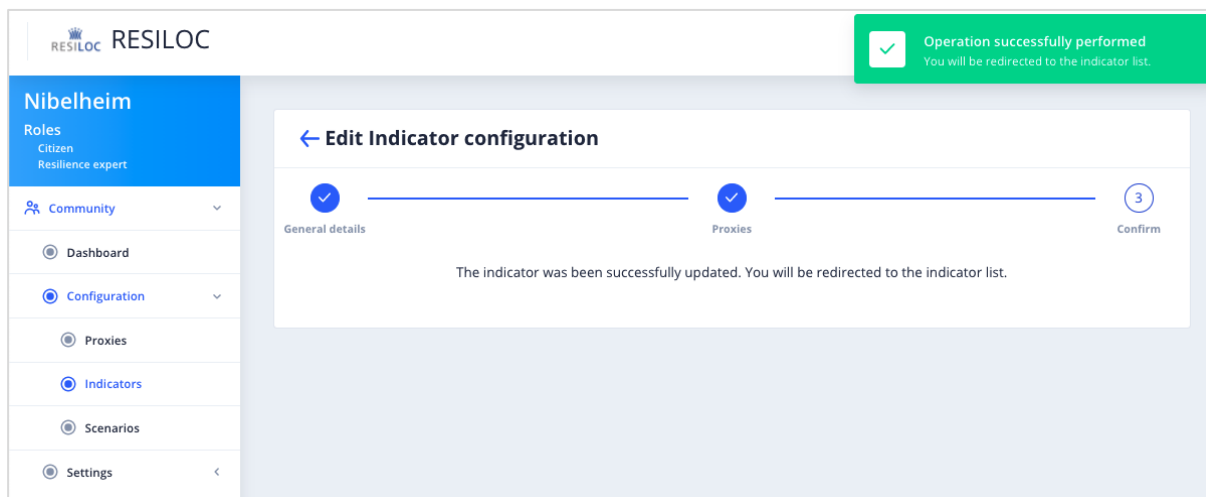
Now Trisha returns to indicator configuration section and by clicking on the pencil icon (edit indicator) continues the indicator creation procedure.



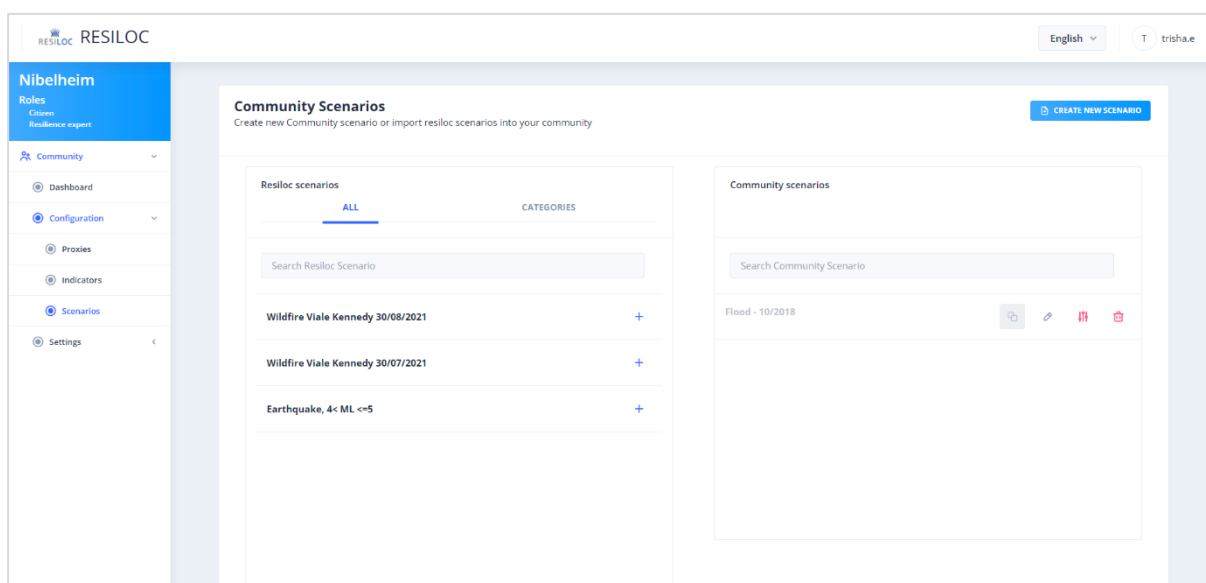
Then she adds the newly created proxy, completing the indicator creation procedure.



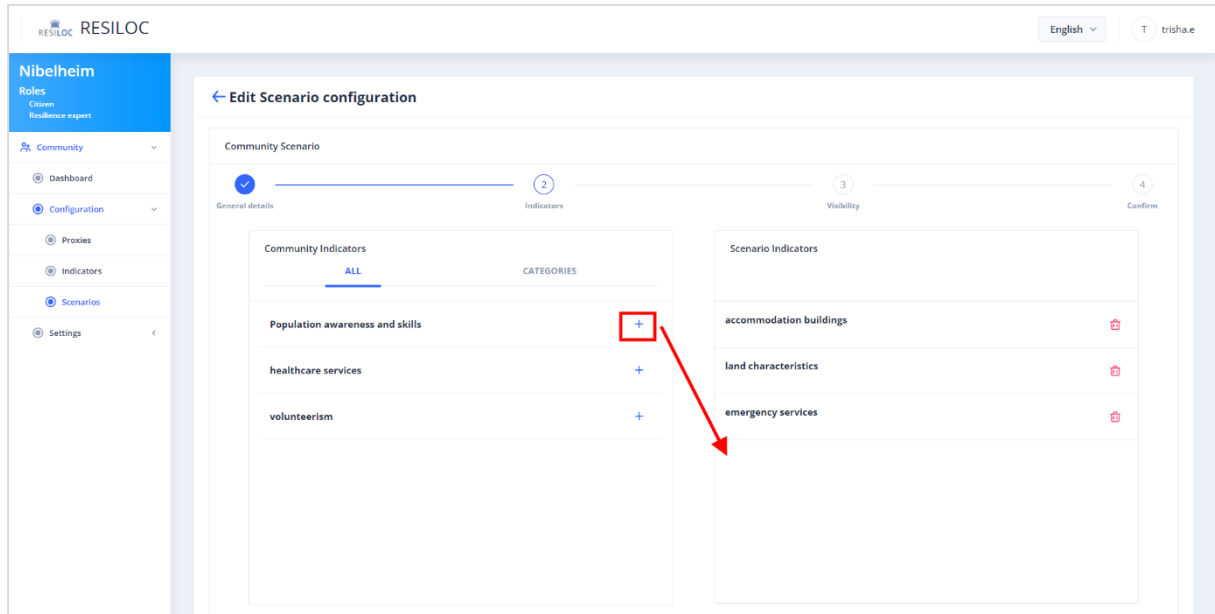
The system will save the newly defined indicator and add it to the list of available community indicators. Now, it will be ready to be used by Trisha in the new scenario designed for the Nibelheim community.



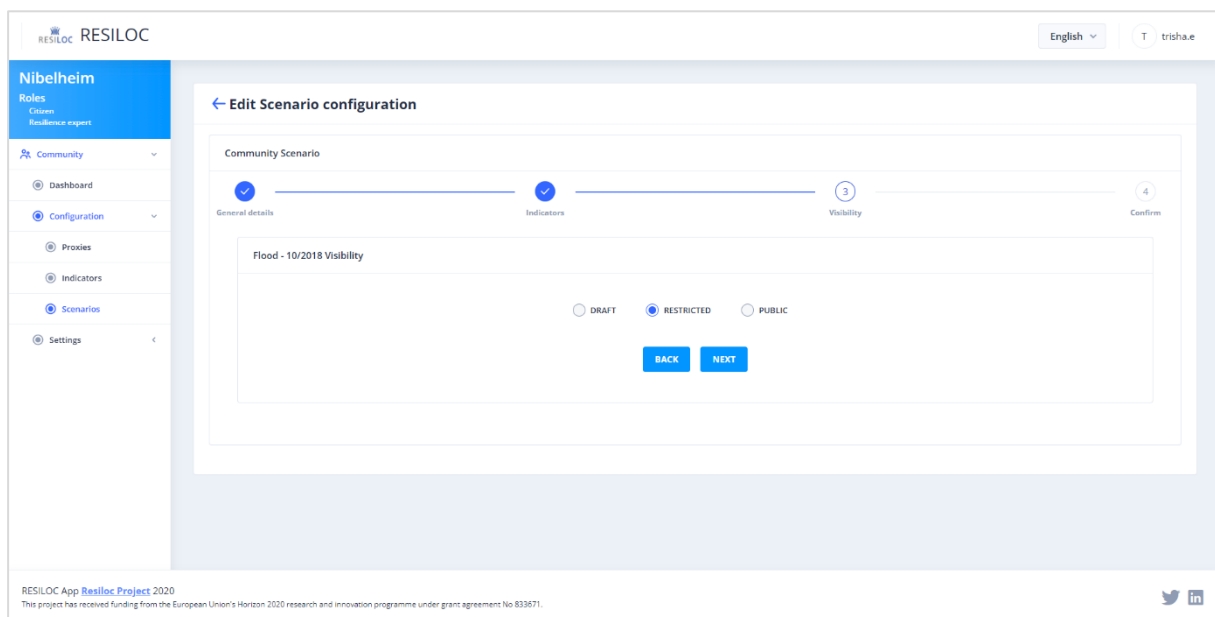
Trisha returns to scenario community list. So, she can edit the scenario by clicking on the pencil icon and continues with the creation of the desired scenario left pending.



Trisha can now include the new indicator in the Scenario indicators list.

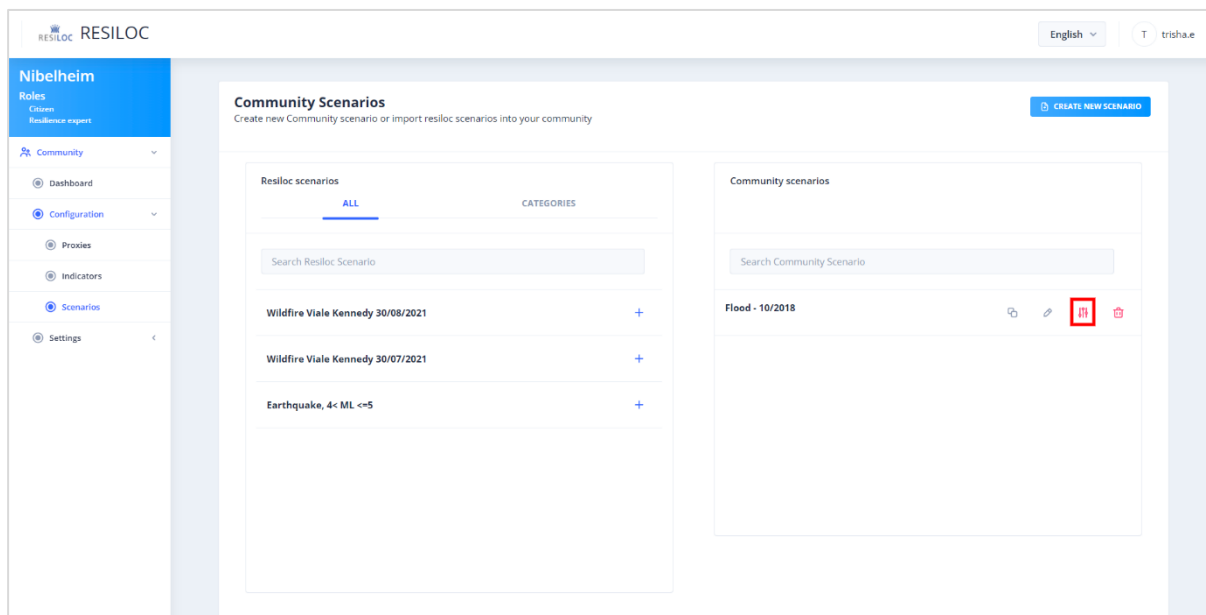


After having complete the indicators' selection, Trisha sets the scenario visibility and proceeds to confirm the newly defined scenario; it will be added to Nibelheim community scenario list and will be available for future resilience assessment.

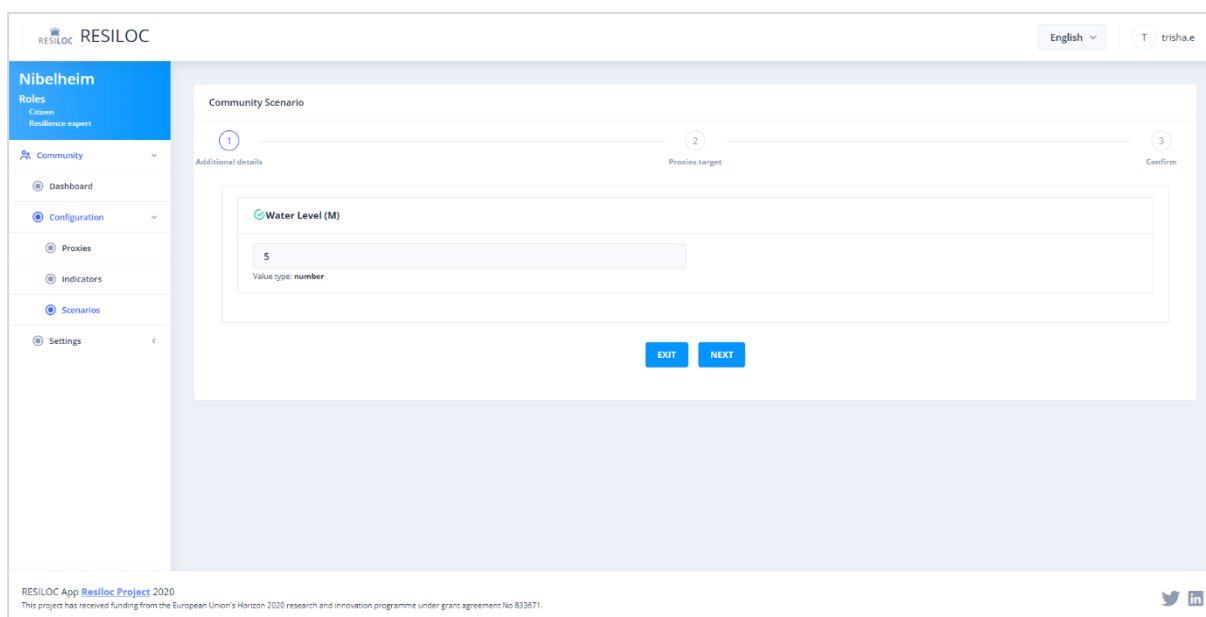


RESILOC local manager use case

Tim as “local manager” of Nibelheim can complete the configuration of the newly defined scenario, setting up respective additional details (if required) and proxies targets by clicking on equaliser button.

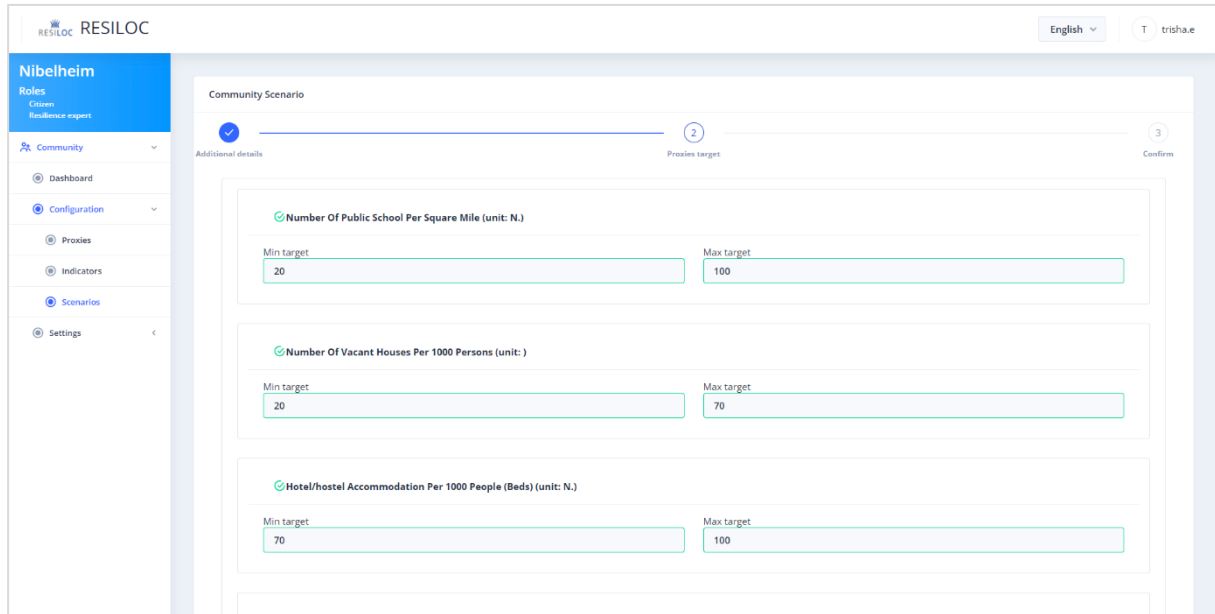


First of all, Tim has to fill-in all additional details specified as required during the scenario creation phase.



Afterward he has to entry the Min and Max target values for each proxy involved into calculation of indicators selected for the scenario at hand.

Tim, taking into account its knowledge of the community and seeking advice and support from LRT members, assigns targets values considering proxy's unit of measurement specified right after the name.



Once all required additional details and targets has been filled-in, the equaliser button turns green. Tim can now produce an assessment for the newly defined scenario.

