ISABEL

MINIMIZE COSTS AND MAXIMIZE BENEFITS IN THE TRANSITIONARY EUROPEAN REGIONAL LABOUR MARKETS

mInimize coSts and mAximize Benefits in the transitionary European regional Labour markets

Deliverable No 5.3

Title: First ISABEL Platform

WP5: Data Analysis Tools and Visualisations on JCD and its uneven effects

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

DOCUMENT ID	D5.3
DUE DATE	30/06/2026
DELIVERY DATE	30/06/2026
WORK PACKAGE	WP5
TASK(S)	T5.3
DISSEMINATION LEVEL	PU
VERSION	V2
AUTHOR/S	CERTH, PROG
FILENAME	ISABEL_D5.3_First_ISABEL_Platform_V2

Abstract

Deliverable 5.3 presents the first operational version of the ISABEL Platform, the central digital infrastructure developed to support the data management, analysis, and policy-support activities of the ISABEL project. The platform provides researchers, policymakers, and industry stakeholders with a secure, scalable, and interoperable environment for exploring information related to Job Creation and Destruction (JCD), labour market transitions, and skills shortages across European regions. The initial release establishes the core technical foundations of the platform through a modular, service-oriented architecture developed and maintained by CERTH. At its centre is an AI middleware layer, built on Open WebUI and powered by the Mistral-Small-24B-Instruct-2501 large language model served via vLLM, which provides conversational AI capabilities, retrieval-augmented generation, and a programmable API surface consumed by both the PROGRESSUS-developed frontend and by project partner tools. Document ingestion and knowledge extraction are handled by the Docling document processing framework, which transforms heterogeneous research documents into normalised, machine-readable representations through an automated Medallion Architecture pipeline. This layered data infrastructure constitutes the technical foundation of the Smart oNe-stop digitAl data shoP (SNAP) and ensures full data lineage, transparency, and reproducibility across all platform datasets. The platform integrates dedicated components for identity and access management, structured and analytical data storage, real-time session management, and secure API gateway routing. A custom Data Lake API provides structured programmatic access to the Medallion data lake for downstream tools and the user-facing frontend. Integration points are established with the project's principal analytical modules (SNAP, FAROROMO, and DPWA) each of which consumes the platform's published API interfaces to deliver its domain-specific capabilities.

Views and opinions expressed in this document are those of the author(s) and do not necessarily reflect those of the European Union or REA. Neither the European Commission nor the granting authority can be held responsible for them.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Contributors

NAME	ORGANISATION
Konstantinos Marthoglou	CERTH
Michalis Angelou	CERTH
Nicholas Vretos	CERTH
Symeon Samaras	CERTH
Vasileios Rekkas	CERTH
Christos Semertzidis	PROGRESSUS

Peer Reviewers

NAME	ORGANISATION
Bojan Ristić	UB
Łukasz Trembaczowski	US

Versions

Version	Date	Author/Organisation	Description
V0.1	20/05/2026	CERTH	ToC
V0.9	16/06/2026	CERTH	Sections 1,2,3,4,5,6,7,9 Added
V1.0	16/06/2026	PROGRESSUS	Front End Section Added
V2.0	29/06/2026	CERTH	Comments integrated, update

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Executive summary

The Deliverable 5.3 “Initial ISABEL Platform” presents the first version of the ISABEL Platform, developed as the central digital environment supporting the data collection, management, analysis, and visualization activities of the ISABEL project. The platform has been designed to provide researchers, policymakers, and industry stakeholders with a secure, scalable, and user-friendly ecosystem for exploring information related to Job Creation and Destruction (JCD), labour market transitions, and skills shortages across European regions.

The initial release establishes the technological foundations required for the project's analytical and policy-support activities. It integrates modern artificial intelligence technologies, structured data management capabilities, document processing services, and data access mechanisms within a unified platform architecture. Through the implementation of a layered data infrastructure and a modular service-oriented design, the platform enables the ingestion, processing, storage, retrieval, and analysis of heterogeneous datasets and research documents originating from multiple sources.

Particular emphasis has been placed on ensuring interoperability, security, and extensibility. The platform incorporates identity and access management mechanisms, AI-powered knowledge extraction capabilities, scalable storage services, and APIs that facilitate integration with project tools and future services. The adoption of a Medallion Architecture further supports data governance, transparency, and reproducibility by providing clear data lineage from raw source material to curated analytical assets.

Funded by the European Union

The ISABEL project has received funding from the European Union’s Horizon research and innovation actions program under grant agreement No 101177687

Contents

Executive summary 4
1. Introduction 8
1.1 Purpose and Scope of the Deliverable 8
1.2 Structure of the Document 8
2. Integration 9
2.1 ISABEL Platform Hardware Configuration 9
2.2 Dockerized Services 9
2.3 CI/CD 10
3. ISABEL Base Platform 11
3.1 OpenWebUI 11
3.1.1 Functional Capabilities 11
3.1.2 External APIs 12
3.2 Security, Identity, and Access Management (IAM) / Authentication 13
3.2.1 SSL Authentication and Encryption 13
3.2.2 KeyCloak 14
3.2.3 Database 14
3.3 Language Model Infrastructure 14
3.3.1 Frameworks 14
3.3.2 Models 15
3.4 Document Processing and Knowledge Extraction 15
3.4.1 Docling Framework 15
4. Data Access and Storage Components 17
4.1 Database Architecture 17
4.1.1 Database Type and Technology 17
4.1.2 Schema Design and Data Models 18
4.2 Data Lake Architecture (Medallion Approach) 19
4.2.1 Bronze Layer 20
4.2.2 Silver Layer 20
4.2.3 Gold Layer 21
4.3 Data Access 21
4.4 Data Backup and Consistency 21
5. Testing and Quality Assurance 22
5.1 Test Frameworks and Tools 22
5.2 Test Data and Mocking 22
5.3 Code Quality Analysis 22
6. Supporting Components and Utilities 23
6.1 Scheduled Jobs and Background Tasks 23
6.2 Batch Processing 24
6.3 File Storage and Management 24
6.4 Search and Indexing 24
7. Integration and API Components 26
7.1 RESTful API Endpoints 26
7.1.1 Data Lake API 26
7.1.2 Internal Service APIs 27
7.2 External System Integrations 28
7.3 Message Queue and Event Handling 29

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

7.4 API Gateway and Rate Limiting 30
7.4.1 TLS and Transport Security 30
7.4.2 Path-Based Routing 30
7.4.3 Payload Limits and Security Headers 31
7.4.4 Rate Limiting 32
8. Front End 33
8.1 Technology Stack 33
8.2 Implementation & Verification 34
8.3 Tools UI 35
8.4 Roadmap 36
9. Conclusion & Future Work 38
Bibliography 39

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

List of Figures

Figure 1 CI/CD Pipeline 10
Figure 2 Docling UI 16
Figure 3 SNAP Data Ingestion Tool UI 33
Figure 4 Persona Generator Tool UI 34
Figure 5 SNAP Data Explorer 36

List of Tables

Table 1.Functional Capabilities of OpenWebUI 11
Table 2 External APIs 12
Table 3 Storage Management 14
Table 4 Core Database Components of ISABEL 17
Table 5 PostgreSQL instances 18
Table 6 Data Lake Architecture Layers 19
Table 7 Background Tasks in ISABEL platform 23
Table 8 DuckDB Tools 25
Table 9 Data Lake API Methods 26
Table 10 Internal Service APIs 28
Table 11 External System Integrations 29
Table 12 Gateway Routing Paths 31

List Of Abbreviations

API	Application Programming Interface
CERTH	Centre for Research and Technology Hellas
CI/CD	Continuous Integration / Continuous Deployment
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DPWA	Digital Policy Writer Assistant
FAROROMO	FAir Green transitiOn of Regional labOur Markets Observatory
GPU	Graphics Processing Unit
IAM	Identity and Access Management
JCD	Job Creation and Destruction
MCP	Model Context Protocol
NLP	Natural Language Processing
OCR	Optical Character Recognition
OIDC	OpenID Connect
POST	HTTP POST Method
RAG	Retrieval-Augmented Generation
RBAC	Role-Based Access Control
REST	Representational State Transfer
SNAP	Smart oNe-stop digitAl data shoP
SSL	Secure Sockets Layer
SSO	Single Sign-On
TLS	Transport Layer Security
UI	User Interface
UOWM	University of Western Macedonia
VRAM	Video Random Access Memory

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

1. Introduction

The ISABEL platform serves as the central digital ecosystem designed to transform how researchers, policymakers, and industry stakeholders interact with regional labour market data across Europe. As regional economies undergo rapid transformations, understanding the dynamics of employment shifts and skills alignment is critical for strategic, evidence-based planning. This initial version of the platform integrates various tools and methodologies to handle these complex requirements efficiently.

1.1 Purpose and Scope of the Deliverable

Deliverable 5.3 marks the initial operational release of this platform under Task 5.3. The primary purpose of this document is to introduce the architectural design, core backend technologies, and data governance frameworks that constitute the platform's foundation.
Developed and maintained primarily by CERTH, the backend architecture establishes a modular, secure, and highly extensible service-oriented environment. This infrastructure integrates advanced artificial intelligence engines, automated document ingestion, and a robust data lake to support the project's analytical requirements. The frontend user experience, which directly consumes these backend services and APIs, is designed and developed by the PROGRESSUS team and is seamlessly linked to this infrastructure. Ultimately, this initial version lays the technical groundwork for a user-friendly system that empowers users to build personalized datasets, execute complex analytical queries, and generate policy-support materials.

1.2 Structure of the Document

To provide a comprehensive overview of the system's technical design and deployment, this deliverable is organized into the following sections:

- **Section 2: Integration** – Focuses on the hardware deployment configuration, microservice containerization using Docker, and the local GitLab-driven Continuous Integration and Continuous Deployment (CI/CD) pipeline.
- **Section 3: ISABEL Base Platform** – Highlights the core AI and service abstractions, centering on the OpenWebUI middleware, KeyCloak Identity and Access Management (IAM), the vLLM language model infrastructure running Mistral-Small-24B, and document parsing via the Docling framework.
- **Section 4: Data Access and Storage Components** – Explains the heterogeneous database architecture (PostgreSQL, MinIO, DuckDB, and Redis) and the multi-layered Medallion Architecture (Bronze, Silver, and Gold layers) powering the Smart oNe-stop digitAl data shoP (SNAP).
- **Section 5: Testing and Quality Assurance** – Outlines the current testing baseline, deployment validations handled through CI/CD pipelines, and the collaborative defect-reporting process.
- **Section 6: Supporting Components and Utilities** – Discusses autonomous background tasks (such as the Medallion Processor), file upload lifecycles, and on-demand search and schema-indexing functions provided by DuckDB.
- **Section 7: Integration and API Components** – Details the custom Data Lake API, internal system interfaces, real-time WebSocket event streaming via Redis/Valkey, and Nginx API Gateway routing and transport security policies.
- **Section 8: Front End** – Outlines the frontend technology stack, interface integration tools, and development roadmaps.
- **Section 9: Future Work** – Summarizes planned optimizations, including GPU-accelerated document parsing, persistent analytical indices, and advanced data backup strategies for future releases.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

2. Integration

2.1 ISABEL Platform Hardware Configuration

The ISABEL Platform is deployed on a dedicated, self-hosted high-performance workstation running a Linux-based operating system. The hardware configuration has been selected to meet the demanding computational requirements of large-scale language model inference and high-throughput data processing.

The centrepiece of the ISABEL server's compute capability is its dual NVIDIA RTX 5090 GPU configuration, which provides a combined 64 GB of video memory (VRAM). This capacity is a direct requirement of the platform's primary language model: Mistral-Small-24B-Instruct-2501, with its 24 billion parameters, exceeds the memory capacity of a single GPU and must be distributed across both devices to run. The vLLM inference engine handles this distribution automatically through tensor parallelism, splitting the model's weight tensors across the two GPUs and coordinating computation between them transparently.

Local storage is provided by an NVMe array, chosen for its high sequential read throughput and low access latency. These characteristics are particularly important for two categories of data access: loading large language model weight files into GPU memory at service startup, and serving local Parquet datasets and vector index files that are read directly by the platform's analytical components.

2.2 Dockerized Services

Docker is an open-source platform that enables developers to build, deploy, run, update and manage containers. Containers are standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

The benefits of Dockerized services, especially in large platforms are i) **environment consistency**, which ensures code behaves exactly the same on a developer's laptop as it does on massive staging and production servers, drastically reducing configuration errors, ii) **streamlined CI/CD pipelines**, which means that a team can pull, build, and push exact image versions across every step of a development pipeline, and iii) **isolation**, which means that a malfunction or a security vulnerability in one microservice will not bring down the entire server or negatively impact other services running alongside it.

The ISABEL Platform is deployed as a single, coordinated Docker Compose application that orchestrates all backend services within a shared private network. This arrangement ensures that every service can communicate with its dependencies by name, that no internal service is reachable from outside the platform unless explicitly exposed through the Nginx API gateway, and that the entire platform can be started, stopped, and updated as a single unit. The platform currently comprises the following containerised services:

All services are configured with restart policies that ensure automatic recovery in the event of an unexpected failure, with the exception of the one-time bucket initialisation job, which exits after completion and does not restart. Persistent data across the platform is managed through named Docker volumes, which decouple the lifecycle of stored data from the lifecycle of the containers that produce it. The platform volumes and their contents are summarised below:

The three most critical volumes, which are open-webui-data, openwebui-db-data, and KeyCloak -db-data, are defined as external named volumes, meaning they are provisioned independently of the Compose deployment and persist even if the entire platform stack is torn down and

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

redeployed. This protects user data, conversation histories, and identity records from accidental deletion during platform updates.

The Data Lake API is deployed as a dedicated Docker container, built from a custom Dockerfile defined within the datalake-api service directory. The service connects to MinIO via the S3 API to serve file catalogue, search, preview, and extraction requests over the Medallion data lake. The Data Lake API container is configured to start only after the MinIO service has passed its health check, ensuring that the object storage layer is fully operational before the API begins serving requests.

2.3 CI/CD

Continuous integration (CI) refers to the practice of automatically and frequently integrating code changes into a shared source code repository. Continuous delivery and/or deployment (CD) is a 2-part process that refers to the integration, testing, and delivery of code changes. Continuous delivery stops short of automatic production deployment, while continuous deployment automatically releases the updates into the production environment



Figure 1 CI/CD Pipeline

For the purposes of the ISABEL project a **locally** hosted GitLab is used. GitLab is a comprehensive, web-based software development and DevSecOps platform. A local or self-hosted GitLab instance gives teams complete autonomy over their code, strict data privacy, and zero reliance on external cloud availability. Given that the ISABEl project requires role-based access to repositories, as well as complete autonomy, control, and granular permissions, a self-hosted GitLab provides the best configuration for the needs of the project.

Through the CI/CD pipelines in GitLab, the developers of the project can push code that can pass automated tests and finally be automatically built and deployed in the live server. This can also alleviate the problem of pushing problematic code to the live server, given that for the updated code to be deployed it should first pass the tests designed by the developing team. If the code doesn't pass the tests, then there is no update to the deployed project, as well as **no downtime** for the users, ensuring reliability and safety.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

3. ISABEL Base Platform

The ISABEL Platform is designed to serve researchers, policymakers, and industry stakeholders as a comprehensive, AI-powered environment for data analysis, knowledge extraction, and evidence-based policy support. Its architecture is structured around a clear separation between a backend layer — responsible for data management, model inference, authentication, and API services — and a frontend layer that delivers the user-facing experience. The frontend is developed by the PROGRESSUS team and is discussed in a subsequent chapter of this deliverable. The backend, described in this chapter, is developed and maintained by CERTH and is built on a curated stack of open-source components.

3.1 OpenWebUI

The backbone of the ISABEL backend is Open WebUI, an open-source, self-hosted AI platform originally designed to provide a feature-rich, ChatGPT-like interface for interacting with large language models across a variety of serving backends. Open WebUI is architected to operate entirely on the operator's own infrastructure, with no dependency on external cloud services, making it well suited to deployments where data sovereignty, security, and configurability are priorities.

Within the ISABEL Platform, Open WebUI is deployed exclusively as middleware. Its graphical user interface is not exposed to ISABEL users; instead, the platform leverages Open WebUI's comprehensive REST API to integrate its capabilities into the ISABEL-specific user experience developed by the PROGRESSUS team. This approach allows ISABEL to benefit from Open WebUI's extensive feature set — model management, retrieval-augmented generation, tool calling, knowledge base management, and multi-user administration — while maintaining full control over how those capabilities are presented and accessed by end users.

Open WebUI provides a stable, well-documented API surface that abstracts the complexity of managing multiple language model backends, handling streaming inference, and coordinating file storage, authentication, and session management — all of which would otherwise require substantial custom development effort.

3.1.1 Functional Capabilities

The principal functional capabilities provided by Open WebUI are summarised in the table below:

Table 1.Functional Capabilities of OpenWebUI

Capability	Description
Multi-backend Model Management	Connects simultaneously to multiple LLM serving backends via OpenAI-compatible APIs, allowing different models to be orchestrated from a single platform.
REST API	Exposes a comprehensive REST API that enables external applications to access all platform capabilities programmatically — including chat completions, file management, knowledge base queries, and user administration.
Real-Time Streaming	Supports token-by-token response streaming over WebSocket connections, ensuring a responsive experience for users interacting with large language models whose generation may span several seconds.
Retrieval-Augmented Generation (RAG)	Allows documents uploaded to the platform to be indexed and retrieved as contextual evidence at query time, augmenting

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Capability	Description
	language model responses with grounded, source-traceable information.
Knowledge Bases	Supports the creation and management of curated knowledge collections that can be attached to model configurations, enabling domain-specific retrieval contexts tailored to the needs of the ISABEL project.
Tool and Function Calling	Enables language models to invoke external tools and structured functions during a conversation, allowing the platform to connect model reasoning to live data retrieval, computation, and API interactions.
Pipelines (Plugin System)	Provides an extensible plugin architecture for defining custom processing pipelines in Python, which can intercept, transform, or augment model interactions before and after inference.
Multi-User Management and RBAC	Supports multiple concurrent users with role-based access control (RBAC), allowing different permission levels to be assigned across user groups.
OAuth 2.0 / OIDC Integration	Natively supports federated authentication via OpenID Connect, enabling seamless single sign-on (SSO) integration with external identity providers such as KeyCloak .
S3-Compatible File Storage	Integrates directly with S3-compatible object storage systems, enabling uploaded files to be stored in and retrieved from the platform's MinIO data lake rather than a local file system.

3.1.2 External APIs

The ISABEL Platform backend serves as a shared infrastructure layer consumed by several analytical modules and user-facing tools developed across the ISABEL project consortium. Each of these components integrates with the platform through its published API interfaces (primarily the Open WebUI chat completions API and the Data Lake API) and constitutes what the platform treats as an external API consumer. The table below summarises the four principal external components and their respective integration points:

Table 2 External APIs

Component	Developed by	Backend APIs consumed	Primary function
SNAP	CERTH	Open WebUI chat completions API; Data Lake API; vLLM inference endpoint	Synthetic persona generation, academic literature retrieval, and unstructured data extraction
FARAROMO	UOWM and partners	Data Lake API; Gold-layer datasets via MinIO S3	Interactive geospatial observatory for JCD analysis and policy visualisation

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

DPWA	CERTH and partners	Open WebUI chat completions API; Data Lake API (Gold layer)	AI-assisted policy drafting tool for policymakers grounded in ISABEL evidence
Frontend	PROGRESSUS	Open WebUI REST API; Data Lake API	User-facing interface for all platform capabilities

SNAP is the ISABEL project's integrated research infrastructure for synthetic demographic simulation, academic literature discovery, and unstructured data extraction. In terms of platform integration, SNAP consumes the Open WebUI chat completions API for all LLM-driven tasks (persona roleplaying, document summarisation, table extraction, and analytical synthesis) routing requests through the platform's Mistral inference backend. Extracted and processed outputs are stored within the Medallion Architecture: raw source files in the Bronze layer, structured JSON and CSV artefacts in Silver, and dynamically created knowledge bases registered in Open WebUI from Gold-layer content. The Data Lake API is available to SNAP components for catalogue browsing and content retrieval across the platform's managed datasets.

FARO ROMO is an interactive geospatial observatory designed to support researchers, policymakers, and the general public in navigating and interpreting the complex, spatially uneven dynamics of JCD processes across European regions. FARO ROMO integrates with the ISABEL Platform backend by consuming curated datasets from the Gold layer of the Medallion data lake — including JCD indicators, skills forecasting outputs, and regional labour market metrics — accessed through the Data Lake API. These curated datasets serve as the evidence base for the observatory's dashboards and spatial analyses.

The Digital Policy Writer Assistant (DPWA) is an AI-powered tool designed to support policymakers in drafting evidence-based policy recommendations grounded in the verified datasets and analytical outputs of the ISABEL project. In terms of platform integration, the DPWA draws directly on the Gold-layer knowledge assets of the ISABEL data lake — including JCD datasets, skills forecasting indicators, and curated research outputs — accessed via the Data Lake API. For generative tasks, the DPWA calls the Open WebUI chat completions API to access the platform's Mistral-Small-24B-Instruct-2501 inference backend, using it to synthesise policy language, generate structured recommendations, and incorporate expert annotations provided by users into the suggestion process.

3.2 Security, Identity, and Access Management (IAM) / Authentication

Access to the ISABEL Platform is governed by a layered security architecture that combines transport-layer encryption, federated identity management, and role-based access control. These mechanisms work in concert to ensure that all communication with the platform is encrypted, that user identities are verified by a dedicated and trusted authority, and that each user's access rights are enforced consistently across all platform services.

3.2.1 SSL Authentication and Encryption

All communication between clients and the ISABEL Platform is encrypted using Transport Layer Security (TLS). The Nginx reverse proxy, which serves as the platform's single entry point, is responsible for TLS termination: it accepts encrypted connections from external clients, decrypts them at the gateway, and forwards requests to backend services over the platform's private internal network. Any unencrypted HTTP connection arriving at the gateway is automatically redirected to HTTPS, ensuring that no data is transmitted in plaintext under any circumstances.

The platform enforces TLS versions 1.2 and 1.3, with a cipher suite configured for forward

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

secrecy. TLS certificates are managed at the gateway level and mounted into the Nginx service at deployment time. This centralised approach to TLS termination means that individual backend services do not require their own certificate infrastructure.

3.2.2 KeyCloak

User authentication and authorisation across the ISABEL Platform are managed by KeyCloak, an open-source Identity and Access Management (IAM) solution that implements the OAuth 2.0 and OpenID Connect (OIDC) protocols. KeyCloak serves as the platform's centralised identity provider, handling all aspects of user login, token issuance, session management, and role assignment.

When a user accesses the platform, they are directed to KeyCloak for authentication. Upon successful login, KeyCloak issues a signed identity token that is passed to Open WebUI, which validates it before granting access to platform resources. Open WebUI is configured to perform automatic account provisioning on first login, matching users by email address to ensure consistent identity across authentication sessions. Role-based access policies, which determine which users can access which platform features and datasets, are defined and managed centrally within KeyCloak, maintaining a clean separation between the identity layer and the application layer.

3.2.3 Database

All data pertaining to security, identity, and access management is stored across three dedicated components, each scoped to a specific category of information. This separation ensures that sensitive identity data is isolated from application data and that ephemeral session state is never conflated with persistent records. The storage arrangement is summarised in the table below; a more detailed treatment of these components and their schemas is provided in Section 4 of the document.

Table 3 Storage Management

Data Category	Storage Component	Content
Identity and Access Data	KeyCloak PostgreSQL	User identities, role assignments, OAuth2/OIDC client configurations, realm settings, and authentication session records.
Application User Data	OpenWebUI PostgreSQL	Platform user profiles, linked identity records, role mappings, and access preferences as maintained by the application layer.
Active Session State	Redis (Valkey)	Ephemeral WebSocket session tokens and in-flight event data for active user connections. Not a store of record; data is discarded when sessions expire.

3.3 Language Model Infrastructure

The language model infrastructure of the ISABEL Platform is responsible for serving, managing, and routing inference requests to the large language models that power the platform's AI capabilities. It is composed of two serving frameworks — each suited to different model sizes and use cases — operating behind a unified API surface provided by Open WebUI.

3.3.1 Frameworks

The main model serving framework deployed in the ISABEL Platform is vLLM. vLLM is a high-throughput, memory-efficient inference engine designed for serving large language models in

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

production environments. It implements PagedAttention and continuous batching techniques that maximise GPU utilisation and enable the concurrent handling of multiple inference requests without sacrificing latency.

vLLM exposes an OpenAI-compatible REST API, making it straightforward to integrate with Open WebUI and with any external tool that targets the standard chat completions interface. In the ISABEL deployment, vLLM operates across two dedicated GPUs using tensor parallelism, allowing it to serve a large-scale model that would exceed the memory capacity of a single device.

3.3.2 Models

In the current deployment of the ISABEL Platform, the primary language model in active use is Mistral-Small-24B-Instruct-2501, served by the vLLM inference engine. This model is a 24-billion-parameter instruction-following model developed by Mistral AI, capable of complex reasoning, structured output generation, and tool-assisted interactions. It is deployed with a maximum context length of 32,768 tokens and with tool-calling support enabled, making it suitable for the range of tasks the ISABEL Platform requires.

Mistral-Small-24B-Instruct-2501 serves a dual role within the platform. It handles direct user-initiated conversations — responding to research queries, summarising documents, and assisting with analytical tasks — and it also serves as the inference backend for externally developed ISABEL tools. These tools communicate with the platform programmatically via Open WebUI's chat completions API, using the model to generate structured outputs, perform knowledge synthesis, and support automated workflows such as knowledge table generation for the Gold layer of the Medallion Architecture.

The model weights are stored locally on the deployment host and are mounted into the vLLM container in read-only mode. No outbound calls to the HuggingFace Hub are made at runtime, ensuring that model serving is entirely self-contained and not dependent on external network connectivity.

3.4 Document Processing and Knowledge Extraction

A central capability of the ISABEL Platform is its ability to transform heterogeneous research documents — spanning statistical reports, policy papers, tabular datasets, and structured data files — into machine-readable knowledge that can be leveraged by the platform's language models. This transformation is handled by a dedicated document processing pipeline that converts uploaded files into normalised representations suitable for ingestion by LLM-based analysis and knowledge generation workflows.

3.4.1 Docling Framework

Document parsing and content extraction in the ISABEL Platform are handled by Docling, an open-source document processing framework developed to simplify the preparation of complex documents for AI ecosystems. Docling is deployed as a dedicated service within the platform and is invoked automatically by the Medallion processing pipeline whenever a new file is ingested into the Bronze storage layer.

Docling supports a wide range of file formats, including PDF, DOCX, PPTX, XLSX, HTML, EPUB, audio formats (WAV, MP3), WebVTT, email formats (EML, MSG), images (PNG, JPEG, and others), LaTeX, and plain text. Its PDF processing capabilities are particularly relevant to the ISABEL project, as many of the statistical reports and policy documents in the platform's datasets are distributed in PDF form. Docling performs advanced PDF analysis (page layout, reading order, and table structure) to ensure that the extracted content faithfully represents the information as organised in the original document. Where documents contain text embedded in images or

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

scanned pages, Docling applies Optical Character Recognition (OCR) to convert that content into editable, searchable digital text.

In the ISABEL deployment, Docling produces Markdown (.md) and Parquet (.parquet) output for the processed files. Markdown is a normalised intermediate format, which is well suited to LLM consumption: it preserves document structure — headings, tables, lists, and prose — in a compact, text-based representation that models can parse reliably, without the ambiguities that arise when a language model attempts to interpret a raw binary format directly. The resulting Markdown and Parquet files are written to the Silver layer of the Medallion Architecture, where they become available for retrieval, preview, and knowledge generation.

In the current platform deployment, Docling operates on the CPU. While this configuration is sufficient for the expected document volumes in the initial release, Docling natively supports GPU acceleration, which can substantially reduce processing times for large files and high-concurrency scenarios. GPU-accelerated document processing is identified as a planned enhancement for the subsequent release of the ISABEL Platform, to be introduced as user volumes and document throughput requirements grow.

Docling also provides a dedicated web-based user interface that gives platform administrators direct access to its processing capabilities, including the ability to configure OCR engines, select export formats, and test document parsing interactively. This interface is accessible to administrators through the platform's API gateway.

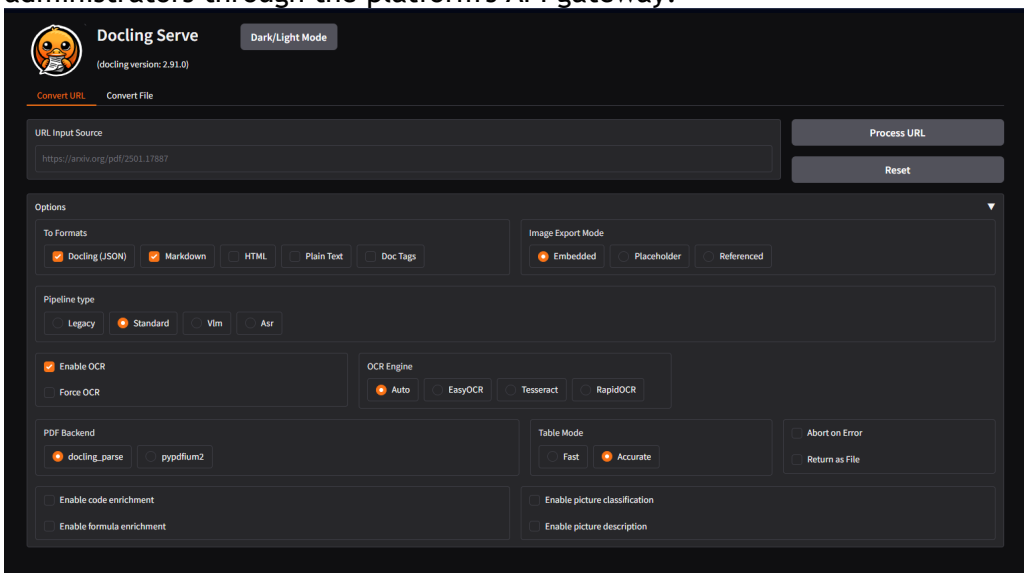


Figure 2 Docling UI

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

4. Data Access and Storage Components

4.1 Database Architecture

The ISABEL Platform is underpinned by a layered, heterogeneous database architecture designed to satisfy the distinct requirements of data ingestion, secure user management, analytical querying, and real-time platform operation. Rather than relying on a single, monolithic data store, the platform deliberately separates concerns across four complementary storage technologies, each selected for its suitability to a specific functional domain. Together, these components form a coherent and scalable foundation that supports the full data lifecycle — from raw data ingestion through to curated, analysis-ready datasets made available to researchers and policy stakeholders.

The table below summarises the core database components of the ISABEL Platform and their respective roles:

Table 4 Core Database Components of ISABEL

Component	Technology	Role	Primary Use
Relational Databases	PostgreSQL 16	Structured Data Store	User data, platform metadata, authentication records
Object Storage	MinIO (S3-compatible)	Scalable File Repository	Documents, datasets, and processed files across Medallion layers
Analytical Engine	DuckDB	In-process Analytics	On-demand querying of structured datasets and metadata
Session Store	Redis (Valkey)	In-memory Cache	Real-time user sessions, WebSocket management

4.1.1 Database Type and Technology

The ISABEL Platform employs four distinct storage technologies, each fulfilling a specific role within the overall data infrastructure. Their combined use ensures that structured, unstructured, analytical, and operational data are all handled in the most appropriate and efficient manner.

Relational Databases

Two dedicated PostgreSQL 16 instances form the relational backbone of the platform. PostgreSQL was selected for its proven reliability, strong support for data integrity and transactions, and wide adoption in production environments. One instance serves the platform application, managing user accounts, conversations, knowledge base records, and file metadata. A second, entirely separate instance is dedicated to the identity and access management subsystem, ensuring that authentication data remains isolated from application data. Both databases are configured with built-in health monitoring to guarantee availability across platform services.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Object Storage

All files, documents, and datasets uploaded to or generated by the platform are stored in MinIO, a high-performance, S3-compatible object storage system. MinIO was chosen because it provides the same programmatic interface as Amazon S3 while operating entirely within the platform's own infrastructure, avoiding dependency on external cloud providers. Storage is organised into three logical buckets — Bronze, Silver, and Gold — following the Medallion Architecture pattern described in Section 4.2. This arrangement allows data to progress through successive stages of quality and enrichment in a structured and traceable manner.

Analytical Engine

DuckDB serves as the platform's embedded analytical query engine. Unlike traditional analytical databases that require dedicated server infrastructure, DuckDB operates in-process alongside the services that call it, enabling fast, on-demand querying of structured datasets without additional operational overhead. It is used primarily to query dataset metadata, run analytical workloads over files stored in the Silver and Gold layers of the Medallion Architecture, and to support the platform's data catalogue and exploration features. Its ability to query files directly — including formats stored in S3-compatible object storage — makes it particularly well suited to the ISABEL Platform's data landscape.

Session and Cache Store

Redis (deployed via the Valkey distribution) provides an in-memory key-value store used for ephemeral, high-speed data. In the ISABEL Platform, Redis manages real-time user sessions and WebSocket connections, enabling responsive, multi-user interactions within the platform's chat and analysis interfaces. Data stored in Redis is intentionally short-lived and is never used as a source of record; its sole purpose is to enhance platform responsiveness and support concurrent usage by multiple users simultaneously.

4.1.2 Schema Design and Data Models

The ISABEL Platform's data model is structured around two complementary design patterns: a conventional relational schema for structured application and identity data, and a Medallion Architecture (Section 4.2) for managing the lifecycle of research datasets and documents. Together, these approaches provide a clear, maintainable framework for data governance that scales with the growing volume and diversity of the project's data assets.

Two PostgreSQL instances maintain a well-defined schema boundary. The application database, managed entirely by the platform's web interface service, stores all data related to platform usage: registered users, model configurations, conversation logs, knowledge base entries, and references to files stored in object storage. The identity database, managed by the authentication service, stores user identities, role assignments, and the configuration of OAuth 2.0 and OpenID Connect clients used for single sign-on. Maintaining these as separate instances enforces a clean architectural boundary between operational platform data and security-sensitive identity information.

Table 5 PostgreSQL instances

Database Instance	Managed By	Scope and Content
openwebui (PostgreSQL)	Open WebUI service	Stores user profiles, conversation histories, knowledge base metadata, model configurations, and file references. Serves as the central application database for all platform interactions.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Database Instance	Managed By	Scope and Content
KeyCloak (PostgreSQL)	KeyCloak service	Manages identity and access control data: user identities, roles, OAuth2/OIDC client configurations, and authentication session records.

Both instances are provisioned with automatic restart policies and health checks, ensuring that dependent services only become available once the databases are fully ready to accept connections.

4.2 Data Lake Architecture (Medallion Approach)

The ISABEL Platform organises its data holdings according to the Medallion Architecture, an established data engineering pattern that structures a data lake into three progressively refined layers: Bronze, Silver, and Gold. The central principle of this approach is incremental data enrichment — at each layer transition, raw evidence is subjected to a defined set of transformations that improve its quality, structure, and fitness for use, without ever discarding or overwriting the source material from which it was derived.

This architecture is the technical foundation of the Smart oNe-stop digitAl data shoP (SNAP), which serves as the unified data access and storage infrastructure of the ISABEL project. SNAP is responsible for storing, sorting, and homogenising the datasets produced through the project's data collection and ingestion activities, and for making them available to both researchers and platform tools through a structured, catalogued interface. The Medallion Architecture enables SNAP to fulfil this role by providing a clear, auditable progression from raw data ingestion through to curated, policy-ready outputs.

Each layer serves a distinct purpose within the data lifecycle, and each marks a critical transformation stage — moving from raw evidence at the point of ingestion, to normalised, machine-readable content, to fully curated knowledge assets ready to support evidence-based policy analysis and decision-making. The table below summarises the three layers and their respective roles:

Table 6 Data Lake Architecture Layers

Layer	Data State	Input	Output / Consumer
Bronze	Raw, unprocessed	User-uploaded files in original format (PDF, CSV, XLSX, DOCX, JSON, Parquet, etc.)	Persistent original-source archive; input to the Silver processing pipeline
Silver	Structured, normalised	Bronze-layer files processed by Docling document parser	Markdown (.md) representations of all ingested files; input to Gold curation and LLM-based analysis
Gold	Curated, analysis-ready	Validated Silver-layer content, enriched datasets, and expert-annotated knowledge artefacts	ISABEL intelligence engine; Digital Policy Writer Assistant (DPWA); policy stakeholders

A key property of this architecture is that each layer retains its data independently. No

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

transformation step modifies or removes data from a prior layer; instead, enriched outputs are written to the next layer as new artefacts. This ensures complete data lineage across the entire knowledge stack: it is always possible to trace any Gold-layer output back through its Silver intermediate representation to its original Bronze source. This property is fundamental to the transparency and reproducibility requirements of a research platform serving policymakers, academic institutions, and industry stakeholders.

4.2.1 Bronze Layer

The Bronze layer constitutes the primary ingestion point of the ISABEL data lake. When a user uploads a file to the platform, it is immediately written to the Bronze storage bucket in its original, unmodified form. No transformations, parsing operations, or format conversions are applied at this stage. The file is preserved exactly as received from the source system, ensuring that the platform always retains an authoritative copy of every dataset in its original state.

This design decision serves two important purposes. First, it guarantees that the original source material remains available regardless of how downstream processing stages evolve — should the parsing logic applied in the Silver layer be revised or improved in a future platform release, files can be reprocessed from their Bronze originals without any loss of fidelity. Second, it provides the foundation for complete data lineage: every subsequent representation of a dataset can be traced back unambiguously to its Bronze-layer source, supporting audit and reproducibility requirements.

The Bronze layer is format-agnostic and is designed to accommodate the full diversity of datasets relevant to the ISABEL project. This includes structured statistical exports such as Eurostat regional data dumps in CSV and XLSX format, NUTS2 regional statistical reports distributed as PDF documents, as well as structured data files in JSON and Parquet formats.

4.2.2 Silver Layer

The Silver layer is the platform's data normalisation stage. Its purpose is to transform the heterogeneous, format-specific files held in the Bronze layer into a consistent, structured representation that is both machine-readable and suitable for ingestion by large language models and analytical tools. This transformation is the mechanism through which raw evidence becomes computable knowledge.

The transformation process is handled by Docling, the platform's integrated document processing service. Upon ingestion of a new file into the Bronze layer, the Medallion processing pipeline automatically submits the file to Docling for conversion. Docling extracts the document's content — including body text, tables, and structural elements — and produces a normalised Markdown (.md) representation, which is then written to the Silver storage bucket. The original file in the Bronze layer is left unchanged.

This processing pipeline is applied uniformly across all supported file types. Text-rich documents such as PDFs and Word files undergo full content extraction and structural normalisation; structured data formats such as JSON and Parquet are parsed and rendered into a tabular Markdown representation that preserves their schema and values in a human- and machine-readable form. Markdown is compact, widely supported, and well-suited to consumption by large language models, ensuring that the Silver layer serves as a reliable, format-independent interface between raw data and the platform's analytical capabilities.

By standardising all ingested content into a unified intermediate format, the Silver layer ensures that the platform's language models and downstream processing tools can operate consistently across the full range of ISABEL datasets — regardless of the original source format or document complexity.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

4.2.3 Gold Layer

The Gold layer represents the highest level of data maturity within the ISABEL Medallion Architecture. It contains curated, validated, and analytically prepared knowledge assets derived from the normalised content of the Silver layer. Where the Silver layer is concerned with consistency and machine-readability, the Gold layer is concerned with relevance, quality, and fitness for direct use in research and policy analysis.

The files stored in the Gold layer are structured knowledge packages; parsed, validated, and enriched artefacts that encapsulate the datasets, metrics, research papers, and domain-specific information that are central to the ISABEL project's analytical mission. These assets are drawn from the project's core evidence base, including data related to Job Creation and Destruction (JCD) and skills forecasting.

The Gold layer is the primary data source for the ISABEL platform's intelligence engine and serves as the evidence base for the Digital Policy Writer Assistant (DPWA). The DPWA draws on Gold-layer content to support policymakers in developing evidence-based policy recommendations targeting Fair Transition strategies grounded in the verified, curated datasets that the Gold layer provides.

Content reaches this layer only after having been preserved in its original form in Bronze and normalised into a consistent intermediate representation in Silver. This lineage ensures that every Gold-layer asset is fully traceable, auditable, and reproducible — qualities that are essential when the outputs of the platform are used to inform policy decisions.

4.3 Data Access

Access to the ISABEL Platform's data repository is provided through three complementary interfaces, each suited to a different access pattern and user type.

The S3 API is the primary and most frequently used access method. Every platform component that reads from or writes to the data lake (including Open WebUI when handling file uploads, the Medallion Processor when routing files between layers, Docling when writing processed output to Silver, and the DuckDB-MCP service when querying file contents) does so exclusively through the S3 API.

The MinIO web console provides a visual overview of all bucket contents, storage utilisation, and access policies, and is accessible through the platform's Nginx gateway at the /minio/ path.

4.4 Data Backup and Consistency

Formal data backup and consistency mechanisms have not yet been implemented in the initial release of the ISABEL Platform, however establishing a structured backup strategy is identified as an item for the final version.

The three MinIO Medallion buckets collectively represent the most critical backup target: the Bronze layer in particular holds the only copies of original source documents, and its loss would be unrecoverable. The two PostgreSQL databases (one serving the platform application and one serving the KeyCloak identity provider) are equally important, as they hold the structured records that underpin all user access and platform activity. Redis session data, by contrast, is ephemeral by design and does not require the same level of backup protection.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

5. Testing and Quality Assurance

Testing and quality assurance in the initial version of the ISABEL Platform are approached pragmatically, reflecting the realities of an active, early-stage development cycle in which the primary focus is on delivering a functional, integrated platform infrastructure. The practices described in this section represent the current baseline, established across all platform components.

5.1 Test Frameworks and Tools

Development and deployment validation are currently supported through the platform's GitLab CI/CD pipeline infrastructure, which allows developers to define per-service jobs covering tasks such as linting, deployment etc. The GitLab CI/CD infrastructure already in place provides a natural execution environment for automated test jobs, meaning that the transition to a test-driven pipeline can be achieved incrementally without requiring significant infrastructure changes.

5.2 Test Data and Mocking

In the initial platform release, test data and the identification of platform defects are managed through a collaborative, feedback-driven process. Team members across the ISABEL project — including developers, integration partners, and members of the PROGRESSUS frontend team — interact with the platform during development and report observed issues by raising tickets in the project's GitLab instance.

5.3 Code Quality Analysis

Code quality in the ISABEL Platform is currently maintained through a peer review process enforced via GitLab merge requests. Automated static analysis and code quality tooling — such as linters, formatters, type checkers, or dedicated code quality platforms — are not yet enforced as part of the CI/CD pipeline in the current release.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

6. Supporting Components and Utilities

Beyond the primary services that handle user interactions, data storage, and model inference, the ISABEL Platform relies on a set of supporting components that manage background processing, file lifecycle operations, and data discovery. These components operate largely outside the direct request-response cycle, ensuring that the platform's data assets remain current, consistently organised, and queryable without requiring manual intervention from developers or administrators. This section describes the platform's background task infrastructure, its approach to batch processing, the mechanisms governing file storage and management, and the search and indexing capabilities provided by the DuckDB analytical engine.

6.1 Scheduled Jobs and Background Tasks

The ISABEL Platform currently operates three distinct background processes, each fulfilling a specific operational role. These range from continuous data pipeline tasks to one-time initialisation routines and low-level infrastructure persistence mechanisms. The table below summarises all active background tasks in the current platform release:

Table 7 Background Tasks in ISABEL platform

Task	Service	Trigger	Output
Bronze-to-Silver document processing	Medallion Processor	Continuous polling of Bronze bucket	Normalised Markdown files written to Silver bucket
Storage bucket initialisation	create-bucket (MinIO CLI)	One-time execution at platform startup	Bronze, Silver, and Gold buckets created and configured in MinIO
Redis on-disk persistence	Redis (Valkey)	Automatic: every 30 seconds if data has changed	In-memory session state persisted to disk, enabling recovery after restart

The most significant of these is the Medallion Processor, which constitutes the platform's primary automated data pipeline. It runs as a continuously active service, monitoring the Bronze storage bucket for newly uploaded files and automatically initiating the processing workflow that transforms them into Silver-layer Markdown representations via the Docling document parsing service, which keeps the Silver layer consistently up to date with the contents of the Bronze layer.

The bucket initialisation task, which is executed once at platform startup by the MinIO client, ensures that the three Medallion storage buckets (Bronze, Silver, and Gold) exist and are correctly configured before any other service begins operation.

Redis performs its own internal persistence routine independently, writing a snapshot of its in-memory state to disk every thirty seconds when changes have occurred. This ensures that active session data and WebSocket event state can be recovered in the event of a service restart, without requiring any application-level intervention.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

6.2 Batch Processing

In the current release of the ISABEL Platform, dedicated batch processing infrastructure has not yet been implemented. The platform's processing workloads are handled in two ways: document processing is carried out asynchronously and continuously by the Medallion Processor, which handles one file at a time as it polls the Bronze bucket; language model inference requests are managed by the vLLM serving engine, which implements continuous batching internally.

6.3 File Storage and Management

All file storage in the ISABEL Platform is handled by MinIO, a self-hosted, S3-compatible object storage system that serves as the unified repository for every file type managed by the platform. File organisation within MinIO follows the Medallion Architecture, described in detail in Section 4.2.

File uploads from platform users are handled by Open WebUI, which is configured to write all received files directly to the Bronze bucket via the S3 API. This integration means that the act of uploading a file through the platform interface is simultaneously the act of ingesting it into the data lake. Once a file lands in Bronze, it is automatically detected by the Medallion Processor and submitted for processing, without requiring any further action from the user.

Access to files stored across the Medallion buckets is provided through two interfaces. The Data Lake API offers a structured, HTTP-based access layer (see Section 7. 1.1). The DuckDB-MCP service provides a complementary, query-oriented interface that allows language models and tools to read and analyse file contents directly from MinIO using SQL and natural-language queries. Together, these two interfaces cover the full range of file access patterns required by the platform's users and components.

MinIO also provides a web-based administrative console, accessible to platform administrators through the Nginx gateway, which supports direct inspection and management of bucket contents, access policies, and storage metrics.

6.4 Search and Indexing

Search and data discovery in the ISABEL Platform are provided through two complementary mechanisms, each suited to a different category of query. The Data Lake API offers keyword-based catalogue search over file names and folder-derived tags in the Bronze layer, as described in Section 7.1.1. The DuckDB-MCP service provides a more powerful, query-oriented discovery and analysis layer that operates directly against file contents across all three Medallion buckets.

DuckDB's role in the ISABEL Platform is that of an on-demand analytical query engine. Rather than maintaining a separate index structure that must be kept synchronised with the underlying storage, DuckDB reads files directly from MinIO at query time using its native S3-compatible file readers. Any file in any bucket can be queried immediately without requiring a separate indexing step, and the query language (standard SQL) is expressive enough to support a wide range of analytical operations that a keyword index could not accommodate.

The DuckDB-MCP service exposes this capability to the platform's language models and tools through a set of registered tool functions, which the Mistral model can invoke autonomously as part of a tool-calling interaction. The currently available tools and their respective functions are summarised in the table below:

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Table 8 DuckDB Tools

Tool	Target Layer	Function
`list_files`	Bronze / Silver / Gold	Browses the contents of any Medallion bucket. Returns a list of all matching file paths.
`execute_query`	Gold / Bronze	Executes an arbitrary SQL query directly against files stored in MinIO (including Parquet and CSV files). Returns results as a formatted Markdown table.
`describe_file`	Gold / Bronze	Retrieves the column names and data types of a Parquet or CSV file without loading its full contents.
`sample_file`	Gold / Bronze	Returns a configurable preview of the first N rows of a structured file, allowing the model or user to inspect data content.

In the current implementation, DuckDB performs no persistent pre-indexing of file contents. Each query initiates a fresh read from MinIO. As the volume of data in the platform's Silver and Gold layers grows, the introduction of persistent DuckDB views, materialised query results, or a dedicated vector index for semantic search over Silver-layer content are identified as natural future enhancements that would improve query performance at scale.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

7. Integration and API Components

The ISABEL Platform is built around a service-oriented architecture in which discrete components interact through well-defined interfaces. Integration across services is achieved primarily through RESTful HTTP APIs, a WebSocket layer for real-time communication, and a shared object storage interface accessible to all platform components. A centralised API gateway — implemented in Nginx — sits in front of all services, providing a single, secured entry point for both external clients and the PROGRESSUS-developed frontend application.

This section describes the platform's API endpoints, its integrations with external and partner systems, its approach to real-time event handling, and the gateway configuration that governs how traffic is routed and secured across the platform.

7.1 RESTful API Endpoints

The ISABEL Platform exposes two categories of REST API: a custom-built Data Lake API developed specifically for the project, and a set of internally accessible service APIs offered by the platform's infrastructure components. All endpoints are served exclusively over HTTPS and, where required, are protected by token-based authentication issued by the KeyCloak identity provider.

7.1.1 Data Lake API

The Data Lake API is a purpose-built RESTful service, developed with FastAPI, that provides structured programmatic access to the platform's Medallion data lake. It is the primary integration point for the PROGRESSUS frontend and for any downstream analytical tool that needs to discover, inspect, or retrieve content from the platform's managed datasets. All endpoints are available under the public path prefix /api/v1/lake/ and are routed through the Nginx API gateway.

The API is designed around a clear separation between the Bronze layer — which holds raw, unprocessed files — and the Silver layer — which holds their extracted, structured text counterparts. Catalogue and search operations always target Bronze, reflecting the full inventory of ingested data; preview and extraction operations target Silver, ensuring that only quality-assured, processed content is surfaced to consumers.

Table 9 Data Lake API Methods

Method	Endpoint	Function	Input	Output
GET	/api/v1/lake/catalog	Returns a paginated catalogue of all raw files held in the Bronze storage layer, with metadata including type, size, and tags	Optional query parameters: file_type, page, limit	Paginated list of file metadata records
POST	/api/v1/lake/search	Performs keyword search	JSON body: search query string;	Ranked list of matching

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Method	Endpoint	Function	Input	Output
		across file names and folder-derived tags in the Bronze layer	optional file type filter	file records with relevance scores
GET	/api/v1/lake/files/{id}	Retrieves full metadata for a specific file identified by its storage path	File identifier in the URL path (supports nested paths)	Single file metadata record
GET	/api/v1/lake/files/{id}/preview	Returns a text preview (up to 1,500 characters) of a document's processed content from the Silver layer	File identifier in the URL path	Markdown-formatted content preview and total word count
POST	/api/v1/lake/files/extract	Performs bulk content extraction of one or more files from the Silver layer, for use in AI-assisted analysis and context hydration	JSON body: list of file identifiers	Full extracted text content per file

7.1.2 Internal Service APIs

Beyond the Data Lake API, several infrastructure services expose their own APIs, which are consumed internally by other platform components. These services are not intended for direct access by end users, but are integral to how the platform's components collaborate. The following table summarises all active internal API services:

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Table 10 Internal Service APIs

Service	Internal Port	API Standard	Purpose
Open WebUI	8080	RESTful HTTP + WebSocket	User management, conversation handling, knowledge base, model orchestration.
vLLM Inference Server	8000	OpenAI-compatible REST API	Serves Mistral-Small-24B for high-throughput text generation and tool-calling. Exposed externally at /vllm/ through the API gateway.
Docling	5001	REST API	Document conversion and text extraction service. Accepts uploaded files and returns structured markdown content. Reachable internally and at /docling/ through the gateway.
DuckDB-MCP	8765	REST / MCP Protocol	Exposes analytical query capabilities over the DuckDB warehouse and the OpenWebUI database. Used by the platform's AI tools for on-demand structured data queries.
Data Lake API	8050	RESTful HTTP (FastAPI)	Custom-built API providing structured access to the Medallion data lake. Publicly accessible at /api/v1/lake/ through the API gateway.

All user-facing interfaces — including the chat interface, data exploration views, and visualisation dashboards — are designed and developed by the PROGRESSUS team, which consumes Open WebUI's API and the Data Lake API to power its frontend.

7.2 External System Integrations

The ISABEL Platform integrates with a number of external and partner systems to deliver its authentication, storage, inference, and presentation capabilities. These integrations are designed to be loosely coupled, meaning that each external system communicates with the platform through a standard interface and can, in principle, be replaced or upgraded independently.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Table 11 External System Integrations

System	Integration Type	Role	Notes
KeyCloak (Identity Provider)	OAuth 2.0 / OpenID Connect	Centralised single sign-on (SSO) and access control for all platform users	Users authenticate via KeyCloak ; tokens are validated by Open WebUI before API access is granted
MinIO (Object Store)	S3-compatible REST API	Primary file and dataset repository across all Medallion layers	All platform services interact with MinIO using the standard S3 API, ensuring portability and interoperability
PROGRESSUS Frontend	REST / WebSocket (HTTP)	User-facing interface consuming Open WebUI's API and the Data Lake API	Open WebUI operates exclusively as middleware in this deployment
HuggingFace Model Hub	Offline file system mount	Source of pre-downloaded language model weights served by vLLM	The hub is mounted in read-only mode; no outbound calls are made at runtime (HF_HUB_OFFLINE=1)

The integration between Open WebUI and the KeyCloak identity provider is implemented via the OpenID Connect (OIDC) protocol, with the platform configured to perform automatic account provisioning upon a user's first successful authentication. Accounts are matched by email address, ensuring that users who authenticate through different OIDC flows are correctly associated with a single platform identity. Role assignment and access control policies are managed centrally within KeyCloak, keeping authorization logic separate from the application layer.

7.3 Message Queue and Event Handling

Real-time communication and event-driven data flow within the ISABEL Platform are managed through two complementary mechanisms: an in-memory message broker for live session events, and an automated data pipeline for asynchronous document processing.

Live, bidirectional communication between the PROGRESSUS frontend and the Open WebUI backend is handled via WebSocket connections through the dedicated /ws/ endpoint. A dedicated Redis instance (deployed via the Valkey distribution) acts as the central message broker for all WebSocket events, enabling the platform to support multiple concurrent users without losing event state.

When a user engages with a conversation, submits a query to a language model, or triggers a document analysis task, the response is streamed back in real time through this WebSocket channel rather than held until completion. This approach is essential for maintaining a responsive user experience when working with large language models, where generation can take several seconds. WebSocket connections are maintained with a 24-hour timeout, accommodating long-running analytical sessions without requiring reconnection.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Redis is used exclusively for ephemeral, in-flight event data — it is not a store of record and holds no persistent platform data. All durable state is written to the appropriate PostgreSQL database or MinIO storage layer.

Document ingestion and Medallion layer progression are handled by an asynchronous processing pipeline that operates independently of user-facing requests. When a file is uploaded to the platform and written to the Bronze storage layer, the Medallion Processor service detects the new object and initiates a processing workflow. This workflow submits the file to the Docling document conversion service, which extracts structured text content and returns it in a normalised markdown format. The resulting output is then written to the Silver layer, making it available for preview and extraction through the Data Lake API.

This pipeline operates on a polling basis — continuously monitoring the Bronze bucket for new or updated files — rather than relying on a traditional message queue with a dedicated broker. While this approach is well suited to the current scale of the platform, it provides a clear foundation on which a more event-driven architecture (for example, using MinIO's native bucket notification system) could be introduced as ingestion volumes grow.

The DuckDB-MCP service complements this pipeline by making the processed contents of the Silver and Gold layers queryable through a structured interface. It is consumed directly by the platform's AI tool-calling infrastructure, allowing language models to run targeted analytical queries over platform datasets as part of a user interaction — effectively bridging the gap between the data lake and the conversational AI layer.

7.4 API Gateway and Rate Limiting

All external traffic entering the ISABEL Platform passes through a centralised Nginx reverse proxy, which serves as the API gateway for the entire system. The gateway is responsible for TLS termination, request routing, protocol handling, and the application of baseline security controls. No service is directly reachable from outside the platform's internal network without passing through this gateway, providing a single, consistent enforcement point for all inbound communication.

7.4.1 TLS and Transport Security

All connections to the platform are encrypted using TLS. The gateway enforces HTTPS exclusively: any HTTP request received on port 80 is immediately redirected to its HTTPS equivalent on port 443, with no unencrypted responses served.

TLS certificates are managed at the gateway level; backend services communicate over the platform's private internal network without requiring their own certificate infrastructure.

7.4.2 Path-Based Routing

The gateway uses path-based routing to direct incoming requests to the appropriate upstream service. Each service is assigned a distinct URL prefix, ensuring clean separation between the application interface, authentication flows, data lake access, model inference, and administrative tools. The routing table is summarised below:

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Table 12 Gateway Routing Paths

Public Path	Upstream Target	Protocol	Notes
/	Open WebUI	HTTPS + WebSocket	Primary application entry point; WebSocket upgrade enabled for real-time features
/ws/	Open WebUI	WebSocket	Dedicated WebSocket path; 24-hour timeout to support long-running sessions
/api/v1/lake/	Data Lake API	HTTPS	All Data Lake API endpoints publicly accessible under this prefix
/authKeyCloak /	KeyCloak	HTTPS	Authentication and SSO flows; WebSocket support enabled for KeyCloak admin console
/vllm/	vLLM (/v1/)	HTTPS	Exposes the OpenAI-compatible inference API; path is rewritten to /v1/ internally
/docling/	Docling	HTTPS	Document processing API; available for authorised internal service calls
/minio/	MinIO Console	HTTPS	Administrative object-storage console; intended for platform administrators only
/health	Nginx	HTTP/HTTPS	Lightweight liveness check

Sensitive internal services (including the PostgreSQL databases, and Redis) are deliberately not exposed through the gateway and remain accessible only within the platform's private Docker network.

7.4.3 Payload Limits and Security Headers

The gateway enforces a maximum request body size of 50 MB across all endpoints. This limit is designed to accommodate the largest document files that researchers are expected to upload while preventing excessively large payloads from affecting platform stability. All responses include standard HTTP security headers: X-Content-Type-Options is set to prevent MIME-type sniffing, and X-XSS-Protection is enabled to instruct supporting browsers to block reflected cross-site scripting attempts. Additionally, access to any .env configuration files is denied at the

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

gateway level, preventing accidental exposure of environment configuration.

7.4.4 Rate Limiting

In the current release of the ISABEL Platform, application-level rate limiting is not yet configured at the gateway. Access control is instead enforced through KeyCloak -issued authentication tokens, which ensure that only authenticated and authorised users can interact with protected endpoints. The introduction of formal rate limiting policies, to guard against abuse and ensure fair resource allocation across user groups, is identified as an enhancement for a subsequent platform release, to be implemented in alignment with observed usage patterns and stakeholder requirements.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

8. Front End

8.1 Technology Stack

The ISABEL Platform frontend is built on a modern, high-performance stack designed for scalability, accessibility, and consistency. The application uses Nuxt 4 operating in compatibility mode, utilizing Vue 3's Composition API for reactive and modular logic.

Nuxt UI version 4.5 and above serves as the primary component library, ensuring a consistent design system. Styling is powered by TailwindCSS 4, which provides a utility-first approach with advanced typography support. Nuxt Content version 3 is used to manage the platform's extensive help system and documentation as a collection of version-controlled Markdown files.

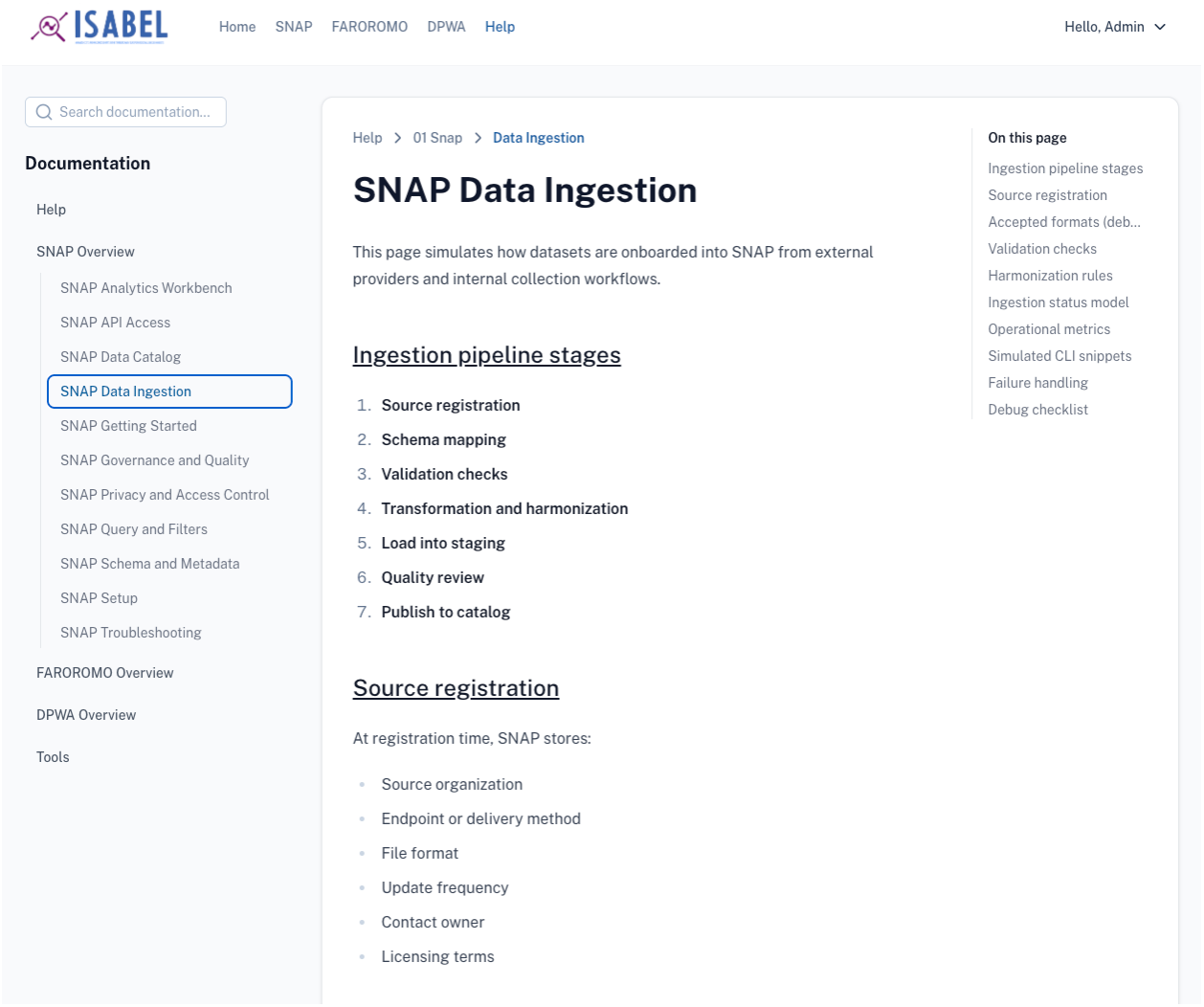


Figure 3 SNAP Data Ingestion Tool UI

The architecture follows a service-oriented model where the frontend consumes a distributed set of APIs. A purpose-built RESTful Data Lake API handles Medallion architecture interactions. A dedicated Flask-based Persona API supports synthetic population generation. Open WebUI serves as a central AI middleware for chat completions, retrieval-augmented generation, and tool-calling orchestration.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

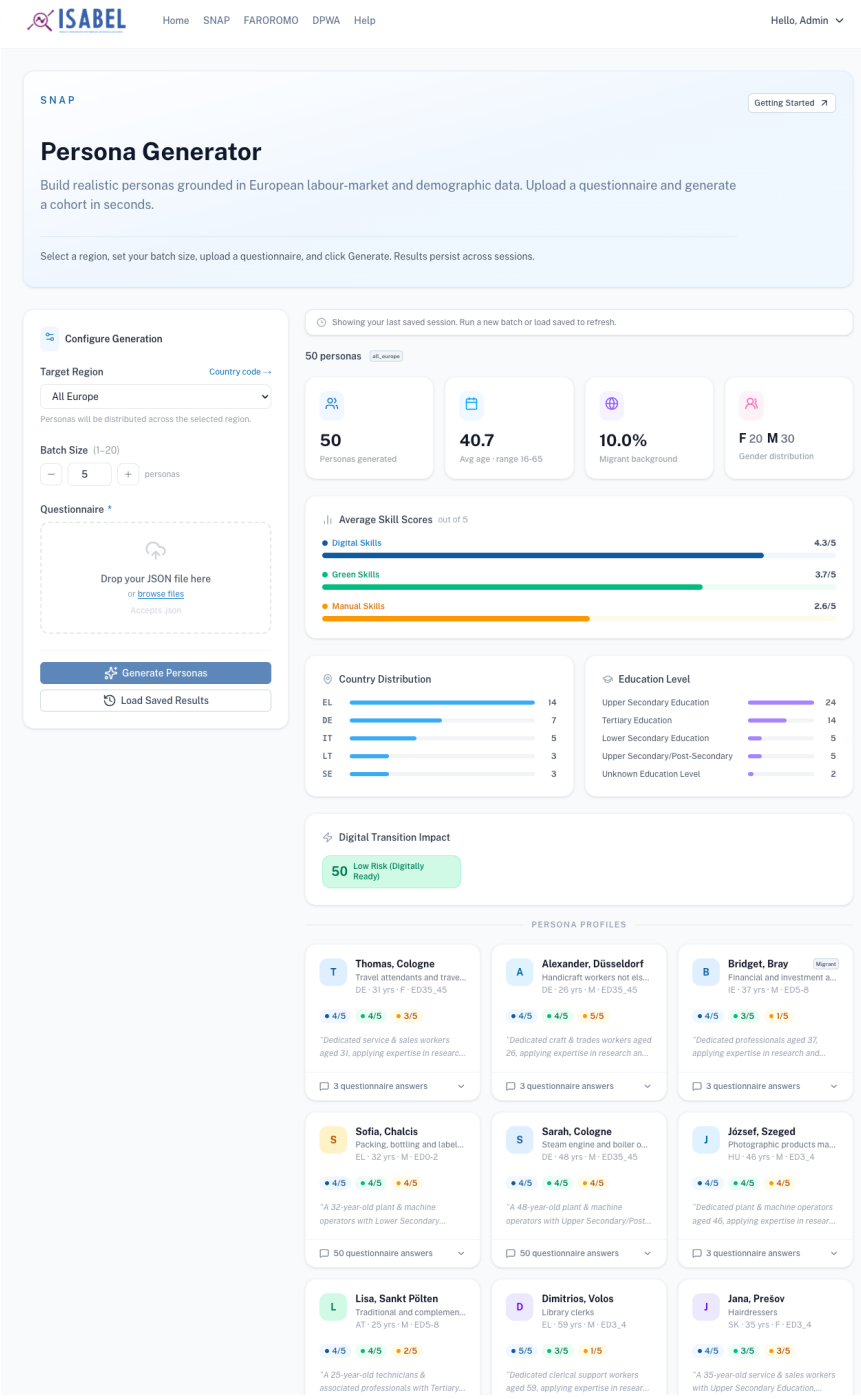


Figure 4 Persona Generator Tool UI

Iconify integration provides consistent iconography through the Lucide and Simple Icons sets. The platform uses Public Sans as its primary variable font to optimize performance and readability. Nitro is employed as the server engine, utilizing pre-compression techniques like Gzip and Brotli along with immutable cache headers for optimized delivery.

8.2 Implementation & Verification

The platform's development follows rigorous engineering standards to ensure reliability and maintainability. Implementation follows a component-driven pattern. Reusable primitives like

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

the PlatformHeader and centralized state management through custom composables allow for rapid development of complex tool interfaces.

Secure session management is handled via dedicated logic with server-side rendering cookie forwarding, ensuring a seamless and protected user experience across the SNAP, FARO ROMO, and DPWA tools. Code quality is maintained through systematic enforcement of standards using ESLint and strict TypeScript type-checking. Performance bottlenecks are resolved through deep architectural analysis. For example, the 3-column workspace scrolling in the DPWA tool was improved by optimizing component lifecycles and CSS containment patterns.

The verification process is collaborative and iterative. The platform undergoes continuous testing including automated linting, type validation, peer reviews via GitLab merge requests, and manual functional verification by integration partners and the development team.

8.3 Tools UI

The ISABEL Platform provides a unified interface for several high-impact research and policy tools.

- The SNAP tool is a comprehensive data explorer that allows users to filter, search, and analyze regional labour market datasets with a paginated, mobile-responsive layout.
- The FARO ROMO observatory features interactive dashboards and maps that visualize European regional trends, job creation and destruction patterns, and skills dynamics.
- The Digital Policy Writer Assistant is a sophisticated 3-column workspace designed for policymakers. It integrates AI-assisted chat with source management and an interactive results area.
- Specialized analytical tools like PaperQA and the Persona Generator support academic literature discovery and the generation of synthetic demographic personas through streaming protocols.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

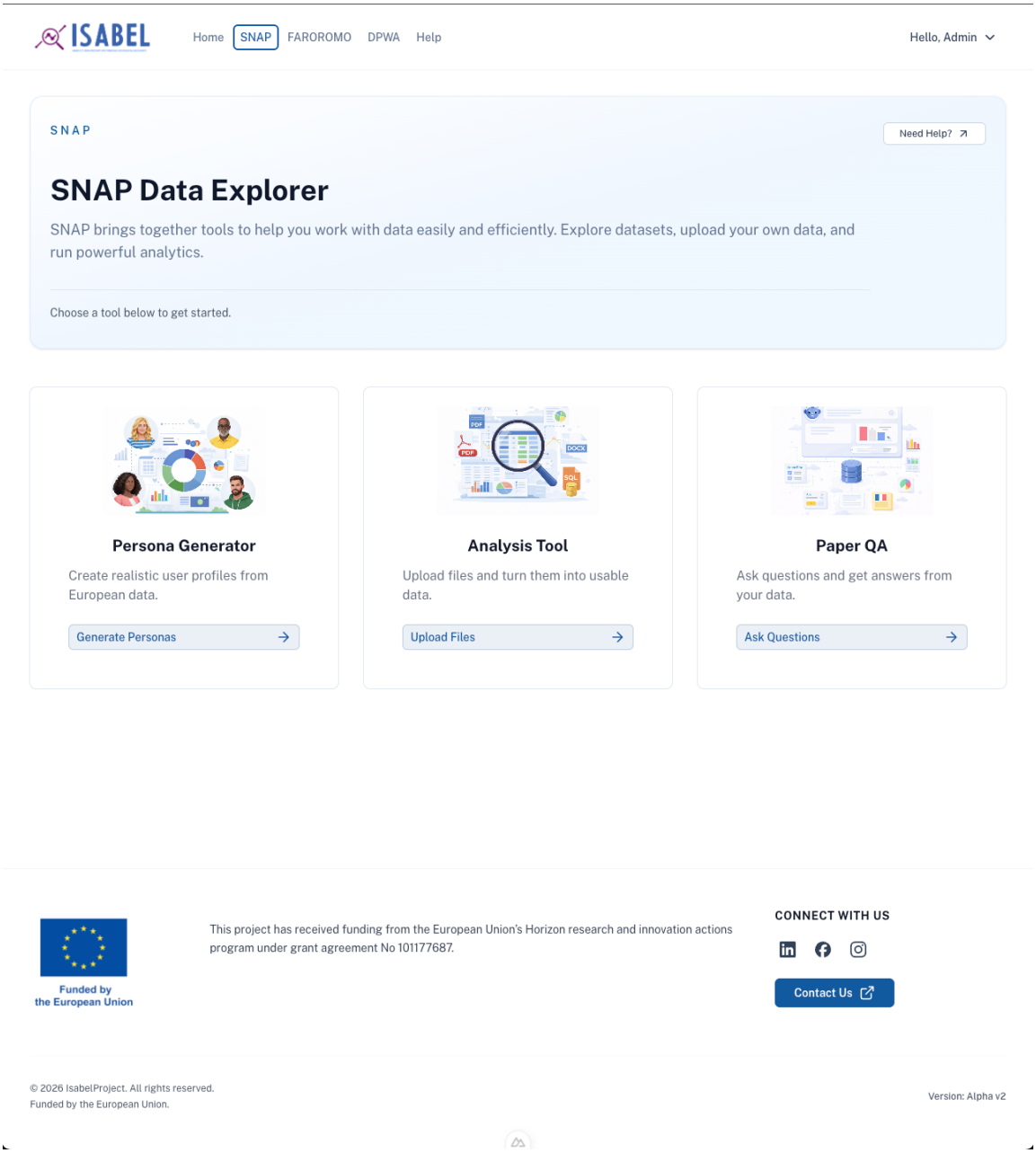


Figure 5 SNAP Data Explorer

The unified user experience is centered on a consistent visual identity using the ISABEL Blue theme color. It features a responsive navigation system, persistent headers, and interactive UI enhancements.

8.4 Roadmap

Future development will focus on transitioning from the current alpha state to a fully production-ready ecosystem. The transition of tool endpoints from mock demonstrations to live production services will connect the interface directly to the Medallion data lake layers.

The implementation of GPU-accelerated document processing will allow the platform to handle high volumes of research papers and statistical reports more efficiently. Deployment of persistent analytical indices and vector search over processed content will improve the accuracy and speed

of AI-driven retrieval.

The platform will undergo infrastructure hardening including the formal implementation of rate-limiting policies at the API gateway and the introduction of advanced data backup and disaster recovery strategies.

Continued refinement of the Digital Policy Writer Assistant collaborative workspace and expansion of the help system will include comprehensive video tutorials and technical walkthroughs.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

9. Conclusion & Future Work

The first version of the ISABEL Platform successfully establishes the foundational infrastructure required to support the project's data-driven research and policy-support objectives. Through the integration of secure data management services, AI-enabled processing capabilities, scalable storage mechanisms, and analytical access tools, the platform provides a unified environment for managing and exploring information related to labour market transitions, Job Creation and Destruction, and skills shortages.

The current release demonstrates the feasibility of combining advanced language model technologies, automated document processing workflows, and structured data lake architectures within a single platform ecosystem.

Future development activities will focus on extending both the functional and analytical capabilities of the platform. Planned enhancements include the introduction of expanded analytical services, improved performance and scalability features, enhanced testing and quality assurance processes, and more comprehensive data protection and backup strategies.

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687

Bibliography

Open WebUI. (2025). Open WebUI Documentation. Retrieved from https://docs.openwebui.com/

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP '23). https://doi.org/10.1145/3600006.3613165

vLLM Project. (2025). vLLM Documentation. Retrieved from https://docs.vllm.ai/

Mistral AI. (2025). Mistral-Small-24B-Instruct-2501 Model Card. Retrieved from https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501

Docling Project / IBM Research. (2024). Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion. arXiv preprint. Retrieved from https://arxiv.org/abs/2408.09869

Docling Project. (2025). Docling Documentation. Retrieved from https://docling-project.github.io/docling/

Keycloak. (2025). Keycloak Documentation. Retrieved from https://www.keycloak.org/documentation

PostgreSQL Global Development Group. (2024). PostgreSQL 16 Documentation. Retrieved from https://www.postgresql.org/docs/16/

MinIO, Inc. (2025). MinIO Object Storage Documentation. Retrieved from https://min.io/docs/minio/linux/index.html

Raasveldt, M., & Mühleisen, H. (2019). DuckDB: An Embeddable Analytical Database. Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19). https://doi.org/10.1145/3299869.3320212

DuckDB Foundation. (2025). DuckDB Documentation. Retrieved from https://duckdb.org/docs/

Redis Ltd. (2025). Redis Documentation. Retrieved from https://redis.io/docs/

Valkey Project / Linux Foundation. (2025). Valkey Documentation. Retrieved from https://valkey.io/topics/introduction/

F5, Inc. / NGINX. (2025). NGINX Documentation. Retrieved from https://nginx.org/en/docs/

Docker, Inc. (2025). Docker Documentation. Retrieved from https://docs.docker.com/

Funded by the European Union

The ISABEL project has received funding from the European Union's Horizon research and innovation actions program under grant agreement No 101177687