Integrated Development Platform for Customer and Aggregator Energy Management Systems (Grant Agreement No 101172675)

D5.1 Initial DevSecOps & Investments Support Package

Date: 2026-06-12

Version 1.1

Published by the INDEPENDENT Consortium

Dissemination Level: PU - Public

Co-funded by the European Union

Document control page

Document file: D5.1 Initial DevSecOps & Investments Support Package_v1.1.docx
Document version: 1.1
Document owner: FORTISS

Work package: WP5 - Plug-and-play deployment and sustainable market adoption support
Deliverable type: DEM - Demonstrator, pilot, prototype

Document status: ☒ Approved by the document owner for internal review
☒ Approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Jessy Matar (FOR)	2025-09-11	Initial table of content
0.2	Rita Beigaite (VTT)	2025-11-19	Contribution to Section 4
0.3	Tilen Barbirič (SCOM)	2025-11-19	Contribution to Section 3
0.4	Mahsa Faraji Shoyari (FOR)	2025-11-30	Finalisation Section 3
0.5	Gašper Ravnak (ECE)	2025-12-08	Contribution to Section 5
0.6	Jessy Matar (FOR)	2025-12-09	Introduction and conclusion added, 1st Draft complete
0.7	Tilen Barbirič (SCOM)	2025-12-18	Addressed reviewer comments
0.8	Gašper Ravnak (ECE)	2025-12-22	Addressed reviewer comments
1.0	Markus Duchon (FOR), Jan Mayer (FOR)	2025-12-30	Final version submitted to the European Commission
1.1	Markus Duchon (FOR), Jan Mayer (FOR)	2026-06-12	Revised according review comments

Internal review history:

Reviewed by	Date	Summary of comments
Dusan Gabrijelcic (JSI)	2025-12-16	Accepted with minor comments
Michael Kluge (CON)	2025-12-30	Accepted without further comments

Legal Notice

The information in this document is subject to change without notice.

The Members of the INDEPENDENT Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the INDEPENDENT Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

INDEPENDENT is funded by the European Union's Horizon Europe Framework Programme for Research and Innovation under Grant Agreement No 101172675. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor CINEA can be held responsible for them.

Index:

1. Executive summary 4
2. Introduction 5
2.1. Purpose, context and scope 5
2.2. Content and structure 5
3. DevSecOps Toolbox 6
3.1. DevSecOps pipeline 6
3.2. Tools 7
3.3. Kubernetes deployment tool 9
3.4. Integration and Future Work 10
3.4.1. Introduction 10
3.4.2. Tools and Technologies Used 10
GitLab Server 10
Harbor Registry 11
3.4.3. System Architecture 11
3.4.4. Challenges Faced 11
3.4.5. Conclusion 12
3.5. Outlook 12
4. Data Spaces for Buildings and Industrial Sites 13
4.1. Minimum Viable Data Space Implementation 13
4.2. Data Space User Interfaces 14
4.3. Outlook: Data Sharing through the Data Space 17
5. Data-driven Tool for Investment Analysis 18
5.1. Introduction 18
5.2. Tools and Methodological Framework 18
5.3. System Architecture 20
5.3.1. Input Parameters 21
5.3.2. External Data Sources 22
5.3.3. Simulation Procedures 22
5.4. Economic Evaluation Framework 23
5.4.1. Technical and Economic Outputs 23
5.4.2. Post-Investment Validation and Deviation Analysis 24
5.5. Challenges faced 25
5.6. Conclusion 25
5.7. Outlook 25
6. Conclusion 27
7. List of Figures 28

1. Executive summary

This deliverable is an initial prototype of the DevSecOps & Investments Support Package accompanied with a public documentation that demonstrates how the modules are used in practice within pilot environments.

The document brings together three core developments: the DevSecOps toolbox, data-space components for buildings and industrial sites, and a data-driven tool for energy-related investment analysis. Together, these core tasks form an integrated framework which supporting secure development, scalable deployment, interoperable data exchange, and technically grounded decision-making.

The first part of the deliverable outlines the DevSecOps toolbox, describing how automated pipelines, containerization practices, and Kubernetes-based deployment workflows have been set up to support consistent and secure software delivery. It also covers the integration work carried out with partners in the consortium, the technologies selected, operational challenges encountered, and the direction for further development.

The second part focuses on data spaces for buildings and industrial facilities. It explains the minimum viable implementation developed in this phase, including data models, interfaces, and mechanisms for sharing information between stakeholders. The task highlights how the data-space approach enables structured, standardized data exchange in environments where interoperability is essential.

The third major component is the data-driven tool for investment analysis. This task showcases the modelling approach, simulation architecture, and economic framework used to assess different technologies such as PV systems, heat pumps, and battery storage. It explains how the tool incorporates input parameters, external datasets, and high-resolution energy flow simulations to generate technical outputs and economic indicators. The document also discusses post-investment validation, where simulated results are compared with real operational data to refine assumptions and improve reliability.

Across these components, the deliverable emphasizes readiness for continued expansion. By combining the developments for the three tasks, this deliverable provides a foundation for secure development, transparent data handling, and informed decision-support in energy-related applications.

2. Introduction

2.1. Purpose, context and scope

The purpose of this deliverable is to report on the work done in WP5 - Plug-and-play deployment and sustainable market adoption support package. The overall objective of this WP is to support scalable deployment and sustainable market adoption of demand-side flexibility management systems in different market settings and business cases. To fulfil this task, the work package is divided into five tasks: T5.1 - DevSecOps Toolbox, T5.2 - Data spaces for buildings and industrial sites, T5.3 - Data-driven Tool for Investment Analysis, T5.4 - Business models and value sharing mechanisms and T5.5 - Market adaptation and TSO and DSO cooperation. The first three tasks contribute to this deliverable, the latter ones will contribute to D5.3 (M34)

2.2. Content and structure

The deliverable starts with an Introduction section.

The T5.1 work is reported in Section 3. This part introduces the pipeline, tools, Kubernetes deployment tool, and the initial integration with the RM/CEM packages.

The T5.2 work is reported in Section 4. In this section, the minimum viable data space implementation is described. Section 4 also includes a report on the data space user interfaces current state.

The T5.3 work is reported in Section 5. This section outlines a data-driven tool designed to evaluate investments in photovoltaic systems, battery storage, and heat pumps. It highlights the need for high-resolution modelling. By combining Python-based computation with Excel-based visualization, the tool offers a structured, transparent framework for analysing technical performance and economic outcomes.

The conclusions are given in Section 6.

3. DevSecOps Toolbox

Task 5.1 (DevSecOps toolbox) focuses on the creation and implementation of a DevSecOps pipeline, along with the development of a Kubernetes deployment tool and the integration of all components into a unified environment. The main goal of this task is to automate and secure the entire software lifecycle—from code commit to deployment—by applying the so called “shift-left” philosophy[1], where testing, validation, and security checks occur as early as possible. In addition to building a secure and automated CI/CD pipeline, the task also includes the development of an infrastructure provisioning tool for Kubernetes clusters running on Proxmox virtual environment[2].

3.1. DevSecOps pipeline

The main focus of the task so far was to create a DevSecOps pipeline that implements the shift-left philosophy.

The DevSecOps pipeline is shown in Figure 1 and currently consists of the following stages:

1. Pre-commit stage, which is executed before the code is even committed.
2. Testing stage, which runs automated tests.
3. Pre-build stage, which generates a version number and opens an engagement in DefectDojo[3].
4. Build-release stage, which builds the image, creates a GitLab[4] release, and pushes the image to the artifact repository.
5. Security stage, which runs various image and filesystem scans and stores the results in DefectDojo.
6. Deployment stage, which deploys the image to the Kubernetes cluster using ArgoCD[5].

The idea of the pre-commit stage is to prevent leaks and misconfigurations before the code reaches the repository. In this stage we use the tool called *pre-commit* [6]with two hooks: *gitleaks*[7], which scans the code for leaked credentials and other sensitive information, and *commitlint*[8], which enforces a consistent commit message format needed for automated versioning. The pre-commit tool is installed directly on the developer machine and runs every time the *git commit* command runs.

The testing stage runs the tests provided by the codebase, including unit tests, integration tests, and end-to-end tests. If the tests fail, the pipeline also fails, preventing broken code from progressing further.

The pre-build stage serves as a helper stage that generates the version number of the release and opens an engagement in DefectDojo. An engagement is a structure within DefectDojo that groups test results and vulnerabilities for a specific version.

The build-release stage then builds and tags the image, pushes it to the artifact repository, and creates a GitLab release corresponding to the new version.

The security stage performs a static application security test (SAST) analysis using the SemGrep[9] scanner and an image scan using Trivy[10]. The scan results are uploaded to DefectDojo and linked to the engagement created in the pre-build stage. This process ensures that vulnerabilities are tracked and associated with specific releases.

The deployment stage runs only when the current branch is designated as a deployment branch. For production deployments, the stage first checks the number of vulnerabilities found in the image. If this number exceeds a defined threshold, the deployment is blocked. When the image passes the check, the version number in the target manifest is updated to the new one generated in the pre-build stage. ArgoCD then automatically detects the manifest change and deploys the new version to the target Kubernetes environment.

[1] https://www.sonarsource.com/resources/library/shift-left/
[2] https://pve.proxmox.com/wiki/Main_Page
[3] https://owasp.org/www-project-defectdojo/
[4] https://gitlab.com/rluna-gitlab/gitlab-ce
[5] https://argo-cd.readthedocs.io/en/stable/
[6] https://pre-commit.com/
[7] https://github.com/gitleaks/gitleaks
[8] https://github.com/conventional-changelog/commitlint
[9] https://github.com/semgrep/semgrep
[10] https://trivy.dev/

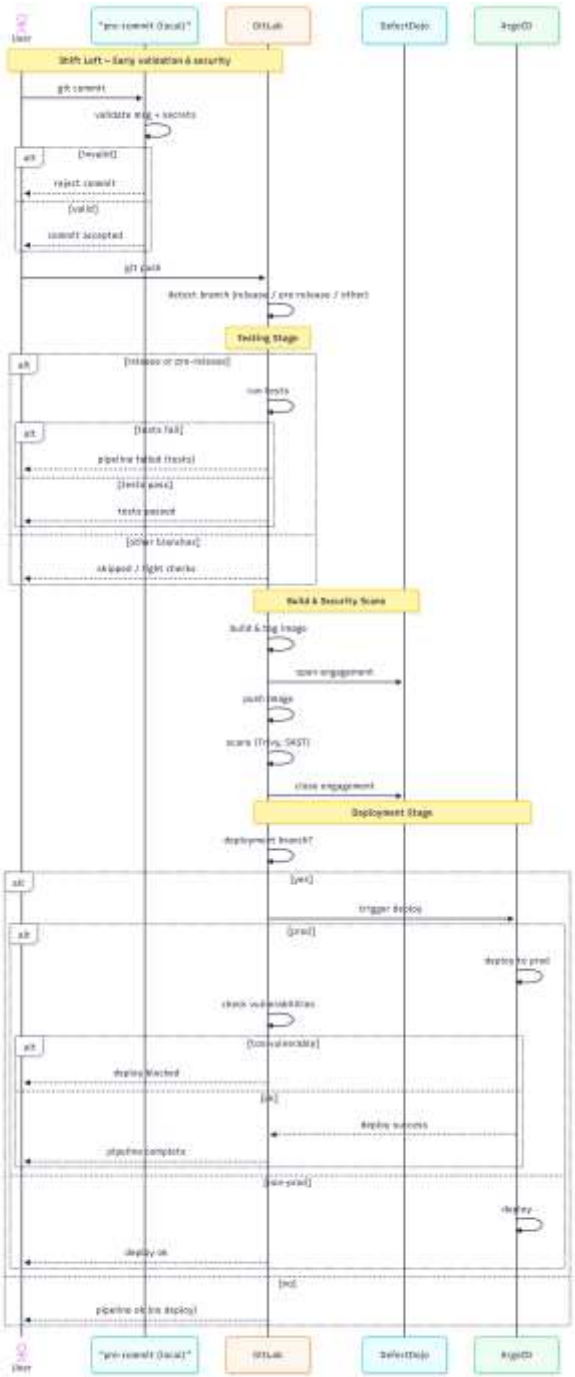


Figure 1 DevSecOps pipeline workflow

3.2. Tools

The tools used so far are GitLab CE for version control, Proxmox VE for virtualization, DefectDojo for vulnerability and threat management, and Harbor[11] for the container image registry. On the Kubernetes side, we use Cilium[12] as the CNI (Container Network Interface), OpenBao[13] for secret management, and ArgoCD for deployment. All the tools used are open-source and can be self-hosted.

[11] https://goharbor.io/
[12] https://cilium.io/
[13] https://openbao.org/docs/what-is-openbao/

GitLab CE is an open-source Git-based version control platform that combines source code management with powerful CI/CD capabilities. Its integration of version control, issue tracking, and pipeline automation allows teams to build, test, and deploy applications securely and efficiently within a single environment.

Proxmox VE is an open-source virtualization platform that supports both KVM virtual machines and LXC containers. It offers high flexibility for managing compute resources, supports clustering and live migration, and provides a user-friendly web interface that simplifies infrastructure administration and monitoring.

DefectDojo is an open-source vulnerability management and orchestration tool that centralizes and tracks security findings from various scanners. It provides automation for deduplication, risk assessment, and reporting, while integrating directly into CI/CD pipelines to support continuous and scalable security monitoring.

Harbor is an open-source container image registry designed for secure and efficient management of container images. It extends Docker Registry functionality with features such as role-based access control, image signing, replication between registries, and built-in vulnerability scanning through the integrated Trivy scanner, ensuring continuous image security.

Cilium is an eBPF-based networking and security solution for Kubernetes that provides advanced network observability, policy enforcement, and load balancing. It enhances both performance and security while integrating deeply with Kubernetes to deliver fine-grained control over network traffic between workloads.

OpenBao is an open-source secrets management solution used to securely store and manage sensitive data such as passwords, tokens, and certificates. It supports dynamic secret generation, detailed access control policies, and comprehensive auditing, enabling consistent and compliant secret handling across all environments.

ArgoCD is a declarative GitOps-based continuous delivery tool for Kubernetes that ensures deployed applications remain synchronized with their definitions in Git. It enables automated rollbacks, supports multi-cluster deployments, and provides both a web interface and CLI for managing and visualizing the deployment process, improving consistency and transparency in delivery workflows.

3.3. Kubernetes deployment tool

Because one of the goals of this task is also to provide a composition service that will be integrated into the CI/CD pipeline, we aimed to automate and simplify the process of deploying, updating, and managing Kubernetes clusters. The workflow of the tool can be seen in Figure 2 below.

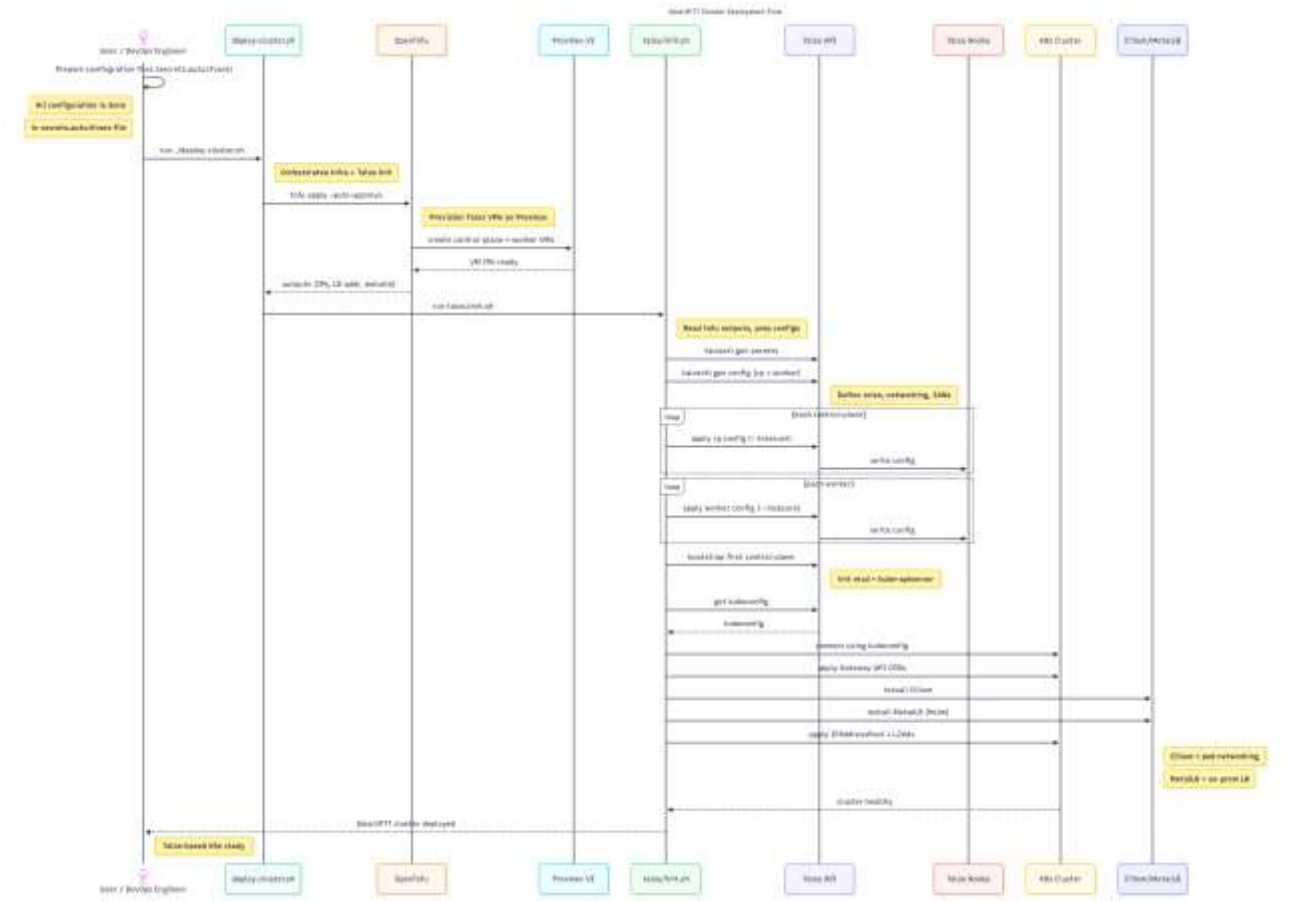


Figure 2 Kubernetes deployment tool workflow

The tool consists of two main components: OpenTofu[14] and a set of Bash scripts. OpenTofu is responsible for provisioning the cluster nodes through the Proxmox VE API, automatically creating the control plane and worker virtual machines. Once the nodes are provisioned and their network information is available, the Bash scripts handle the next phase of the process.

The Bash portion of the tool uses the *"talosctl"* command-line utility to configure and initialize the Talos[15] operating system on the created nodes. It generates and applies the configuration files, defines control plane and worker roles, bootstraps the Kubernetes control plane, and retrieves the *kubeconfig* for cluster access. After the cluster is initialized, the script deploys the networking and load-balancing components. Cilium[16] is installed using the Cilium CLI, which ensures correct setup of the networking layer and enforces Kubernetes network policies. MetalLB[17] is then deployed to provide load balancing for on-premises environments.

The main value of this provisioning tool lies in its simplicity and ability to rapidly deploy Talos-based Kubernetes clusters in a Proxmox environment. It transforms what is typically a lengthy and complex manual setup into a fast, reliable, and repeatable process.

The operating system used on the cluster nodes is Talos, a security-hardened Linux distribution built specifically for Kubernetes. Talos includes only the binaries required to run Kubernetes, omitting unnecessary components such as SSH for improved security. Talos images are very customizable, we can for example add

[14] https://opentofu.org/
[15] https://docs.siderolabs.com/talos/v1.11/overview/what-is-talos
[16] https://cilium.io/
[17] https://metallb.io/

GPU driver support or even kernel NFS support. All configuration and management are performed via the Talos API using the *talosctl* command-line utility. This design minimizes the attack surface, simplifies maintenance, and enables consistent and automated updates across the entire cluster.

3.4. Integration and Future Work

3.4.1. Introduction

Automation and security are essential aspects of modern software delivery processes. As applications increasingly rely on containerized environments, it becomes important to ensure that container images are built, validated, and stored in a secure and automated manner. Continuous Integration and Continuous Deployment (CI/CD) pipelines play a key role in achieving these goals by reducing manual intervention, improving reliability, and enabling consistent security checks throughout the development lifecycle.

In this project, an **internal CI/CD implementation** (GitLab Server, GitLab Runner, and Harbor Private Registry) was developed to evaluate and demonstrate the interaction between a source code management platform, a build execution service, and a private container registry. The primary goal of this implementation was to verify that container images can be automatically built, scanned for vulnerabilities, and stored securely within a private environment.

This implementation serves as a **proof of concept**, illustrating how DevSecOps principles can be applied in a controlled setup. While the solution was implemented on a limited scale, it reflects commonly used practices in larger DevOps and DevSecOps environments and can be extended to integrate with broader infrastructures in future stages.

3.4.2. Tools and Technologies Used

GitLab Server

GitLab Server was used as the central platform for source code management and CI/CD pipeline configuration. All project files, including application code and CI configuration files, were stored in GitLab repositories. GitLab allowed us to define automated jobs using a .gitlab-ci.yml file, which describes how and when pipelines should run.

Key functions of GitLab Server in this project:

- Repository hosting and version control
- Pipeline definition and execution management
- Integration with GitLab Runner
- Pipeline logging and result visualization

GitLab Runner

GitLab Runner is responsible for executing the jobs defined in the GitLab pipeline. In this project, GitLab Runner was installed and configured to run in Docker mode, which allowed it to build container images directly.

Main responsibilities of GitLab Runner:

- Fetching project code from GitLab repositories
- Executing build scripts

- Building Docker images
- Logging pipeline outputs
- Pushing images to Harbor Registry

The runner listens for jobs from GitLab Server and performs each task automatically when a new commit is detected.

Harbor Registry

Harbor is a private container image registry that provides secure storage and built-in vulnerability scanning. Instead of pushing images to a public registry such as Docker Hub, we configured our CI/CD pipeline to push images to Harbor, ensuring better security, access control, and reliability.

Key features of Harbor used:

- Private image storage
- Role-based access control
- Vulnerability scanning
- Image signing (optional)
- Secure authentication

Harbor automatically scanned pushed images and displayed vulnerability results through its web interface. This allowed the team to verify image security before deployment.

3.4.3. System Architecture

The CI/CD workflow followed a structured architecture:

1. Developer pushes code to GitLab repository
2. GitLab triggers a pipeline
3. GitLab Runner executes pipeline instructions
4. Docker image is built
5. Image is tagged and pushed to Harbor Registry
6. Harbor scans the image for vulnerabilities
7. Results are displayed in Harbor dashboard

This automated process ensured continuous integration and security checking without requiring manual intervention.

3.4.4. Challenges Faced

Several challenges were encountered during the project:

- Runner configuration issues
- Harbor SSL certificate setup
- Authentication between GitLab and Harbor
- Docker permission errors
- Image tagging strategy

These were resolved through configuration updates, documentation review, and team collaboration.

3.4.5. Conclusion

This section presented an internal CI/CD implementation designed to validate the integration between automated build pipelines and a private container registry. The solution demonstrates how container images can be securely built, scanned, and stored using standard DevOps and DevSecOps practices and contributes to secure deployment and operations (DR1, D.2.1).

As future work, this implementation can be extended to support **integration with external or centralized container registries as mentioned in 3.2 and 3.3**, allowing images built by internal CI/CD pipelines to be shared across multiple environments or teams. Additional enhancements may include automated testing stages, policy-based security enforcement, deployment automation, and monitoring integrations. Hence, by applying the final implementation of DevSecOps tools and principles (KER10), security and privacy can be integrated as a shared responsibility throughout the whole DSFM system life cycle from architecture design to deployment and operations (KPI2).

3.5. Outlook

In the upcoming months, we would also like to integrate other tools for improved automation, observability, and security within the DevSecOps pipeline.

In recent provisioning tool developments, testing has begun with Cluster API using the CAPMOX[18] provider. The goal is to redesign the current provisioning process so that OpenBao deploys a single-node control cluster. This control cluster will host all necessary Custom Resource Definitions (CRDs) and components required by the *clusterctl* tool. From there, it will provision and manage the actual workload clusters directly through the Proxmox API. For shared Kubernetes storage, several options were evaluated, including Rook Ceph and the Proxmox CSI driver. The latter is more flexible as it delegates storage provisioning to Proxmox itself, rather than handling it within Kubernetes, while the former is more suitable for cases where we do not have an existing highly available storage backend (Ceph RBD (Raw Block Device)[19] or NFS (Network File System)). Adopting this method will significantly simplify cluster upgrades and lifecycle management, as well as improve scalability compared to the current setup. It represents a major step toward a more modular, maintainable, and production-ready cluster management architecture.

We also plan to provide deployment examples and Helm charts for all tools used within the scope of this task, allowing them to be mixed and matched as needed. For example, if an organization already has its own GitLab instance or container registry, they could choose to deploy only the security-related components and integrate them with their existing systems.

A big focus will also be placed on monitoring, observability, alerting and validation, all of which will complete the DevSecOps toolbox.

[18] https://github.com/ionos-cloud/cluster-api-provider-proxmox
[19] https://ceph.io/en/discover/

4. Data Spaces for Buildings and Industrial Sites

This section outlines the current progress in developing a data space for buildings and industrial sites. A Minimum Viable Data Space (MVDS) architecture has been implemented, and user interfaces have been developed in parallel to support key functionalities: data provision, discovery and consumption. Section 4.1 details the MVDS architecture, Section 4.2 introduces the user interfaces and Section 4.3 discusses the next steps, practical use and data sharing through the data space within the project.

4.1. Minimum Viable Data Space Implementation

MVDS[20], as defined by the International Data Spaces Association (IDSA), is a set of essential components that enables the creation of a functional data space for secure and sovereign data exchange. It can be quickly implemented, tested and adapted to meet user requirements.

Our implemented MVDS consists of the following components that behave according to IDS specifications (reference architecture model IDS-RAM[21]):

- Two certified VTT DSIL[22] data space connectors acting as data provider and data consumer.
- An identity provider component, which consists of the Dynamic Attribute Provisioning Service (DAPS) and Certificate Authority (CA).
- A metadata broker as an additional component of MVDS to allow searching for data sets.

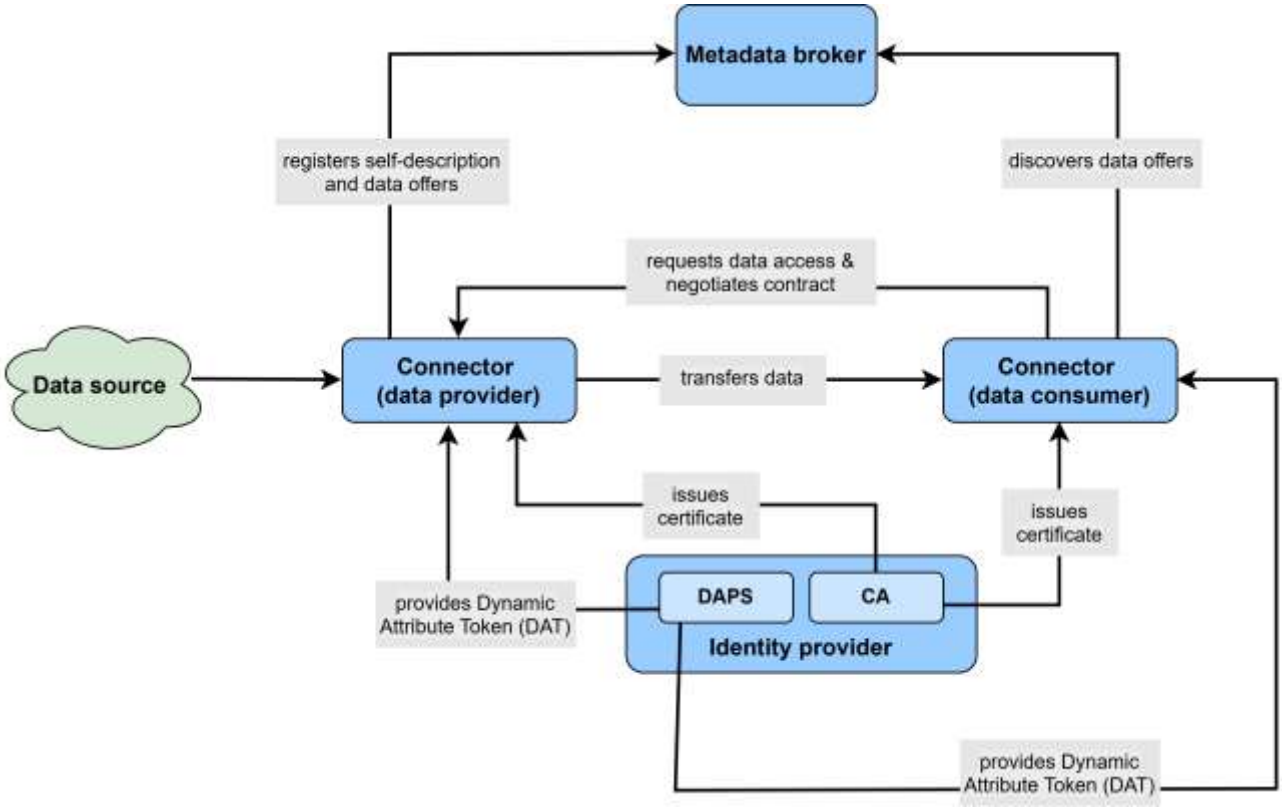


Figure 3 Logical architecture diagram of the implemented MVDS. To maintain visual clarity, the registration of the metadata broker with the identity provider is not depicted. The DAPS provides a DAT and the CA issues a certificate for the metadata broker through an analogous interaction to that between the identity provider and the connectors.

[20] https://docs.internationaldataspaces.org/ids-knowledgebase/ids-reference-testbed/minimum-viable-data-space/mvds
[21] https://internationaldataspaces.org/offers/reference-architecture/
[22] https://internationaldataspaces.org/data-space-connector-report/

Figure 3 The data provider's connector registers its data offers and self-description with the metadata broker. The self-description includes defined terms of data access and usage. The data consumer's connector uses the metadata broker to discover data offers. Both connectors obtain certificates from the CA and DATs from the DAPS to establish trust and secure communication. The data consumer's connector initiates the formal contract negotiation (accepting the terms of the data offer) and then uses its credentials to request data access from the data provider's connector. If authorised, the data provider's connector securely transfers data to the data consumer's connector.

4.2. Data Space User Interfaces

User interfaces have been developed on top of the core components of the data space (connector, metadata broker and identity provider) to allow users to interact with and use their functionality. They enable users to provide data to the data space (Figure 3 and Figure 4), search for data in the catalogue of available data offers (Figure 5) and consume selected data offer(s) (Figure 6). The server-side architecture and template rendering for these interfaces were developed using the Python Django[23] framework, which allows for rapid development and includes built-in security features.

When providing data to the data space, users must complete required metadata fields, including the title of the offered data, description, keywords, publisher name, and the language of the offer. After uploading a data file or providing a file URL, users must also create a data access policy (data access rules) and select the validity period for the data offer (Figure 4). Currently, the supported data access policy is *access between the dates*, where the dates correspond to the offer's validity period. This access policy is enforced through the data space connector, and downloading or viewing data through the access URL is not possible after the offer's validity period ends.

[23] https://www.djangoproject.com/

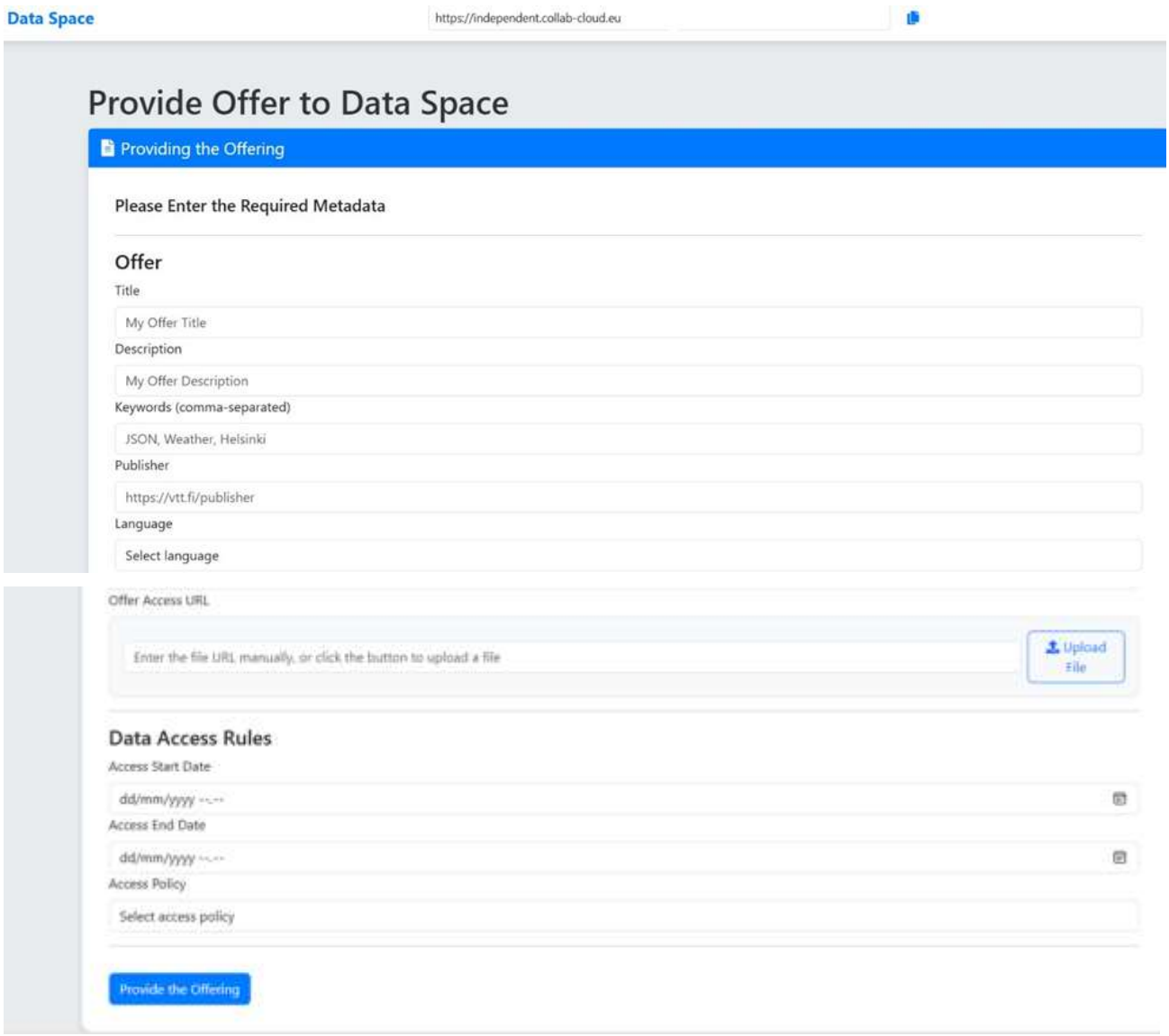


Figure 4 Data space user interface. Creating a data offer and filling metadata fields.

After users fill in the required offer metadata fields and press the *provide the offering button*, the data offer and its metadata become visible in the *available offers* user interface. This interface allows users to browse the catalogue of data offers in the form of cards, each displaying the entered metadata. Users can search for data offers by keywords, which filters the offers and highlights the search terms within the cards. Once a user finds an offer of interest, they can then select the corresponding card. After reviewing the details, the user can either return to the catalogue view or proceed to consume the offer.

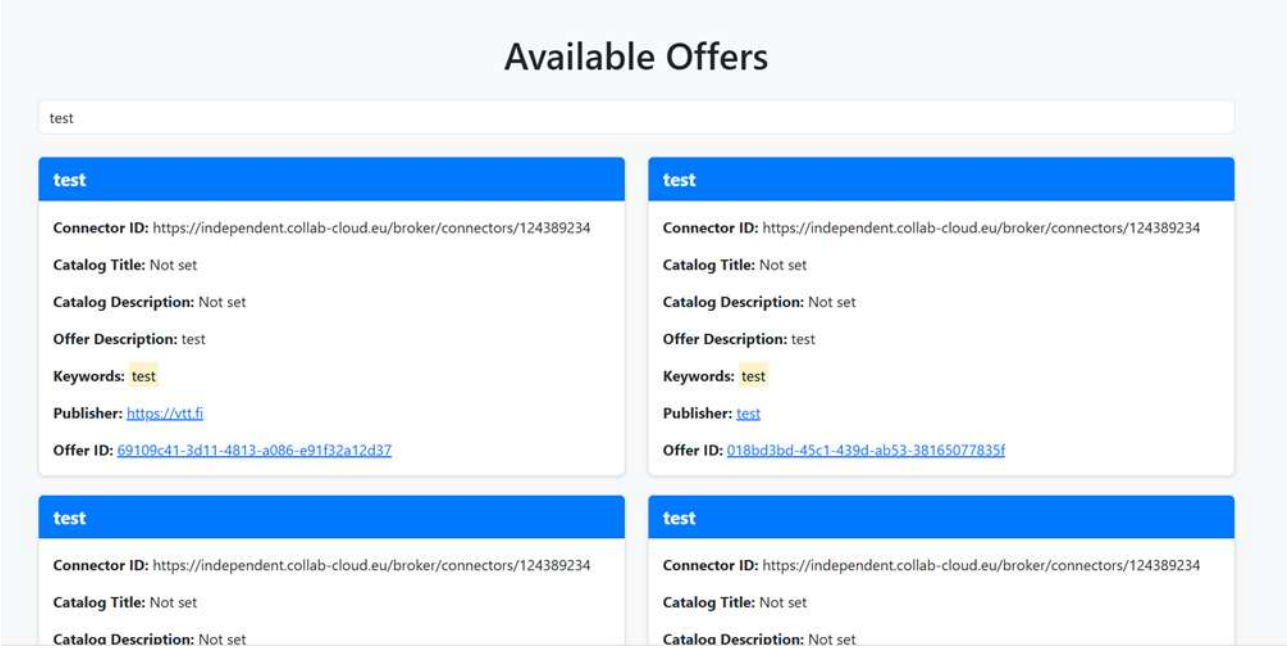


Figure 5 Data space user interface. Data search.

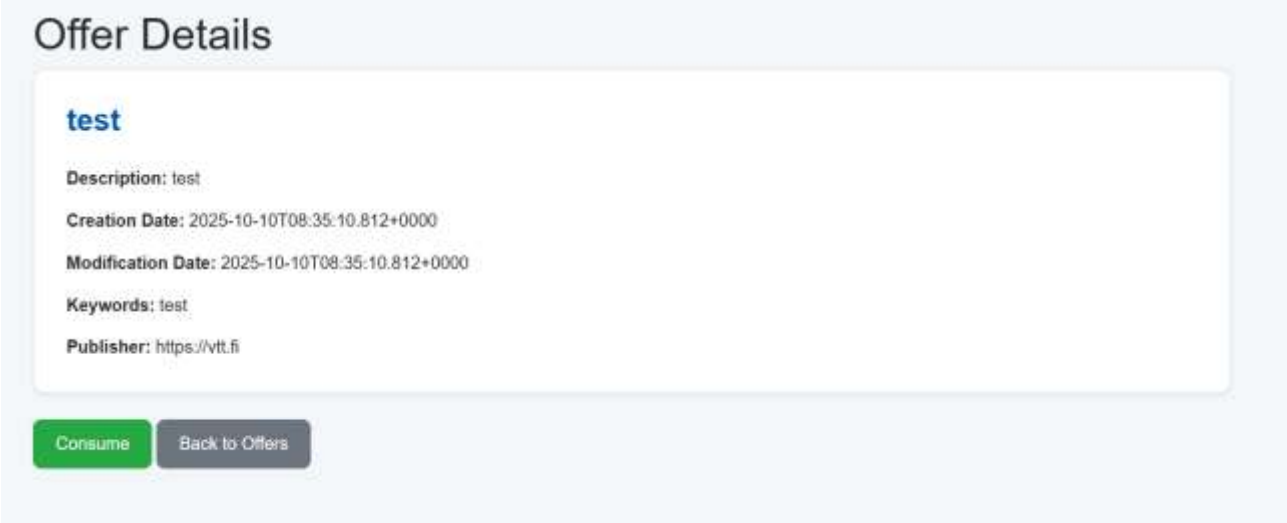


Figure 6 Data space user interface. Consuming the selected offer.

The Provider user interface facilitates data sharing by triggering a structured backend workflow, which is managed by the Django framework. When a new data offer is submitted, the Django backend orchestrates a multi-step registration process through sequential HTTP POST requests to the data space connector. This process starts by publishing all necessary metadata to the connector: creating a catalogue entry, defining the resource offer, and specifying the data representation. Data sharing is governed by policies with access rules linked to the offer. The last step is generating an artefact, which contains the access URL and the authentication credentials needed for secure data access. Rather than directly sharing the data, the Provider user interface utilises the backend to safely register where the data is and how it can be accessed.

The Search/Consumer user interface relies on the backend to follow the data space protocol for safe data fetching. When a user selects a data offer, the workflow first enters the discovery phase, in which the backend dynamically queries the broker service to display a catalogue of available offers. Then, upon the confirmation of consumption, the negotiation phase starts by issuing sequential HTTP requests to the provider's connector. Once it is completed, the backend makes a final GET request using embedded authentication for secure access to the data.

4.3. Outlook: Data Sharing through the Data Space

The next phase of data space development will focus on user onboarding and facilitating data sharing among project partners. Before initiating user onboarding, a rulebook will be developed as part of the organisational building block and governance framework for the data space. This rulebook will define the operational principles and regulatory requirements governing the INDEPENDENT project data space. Subsequently, role-based identity management will be implemented to ensure that only authorised users can access the data space and view available data offers. Once these preparations are complete, we will begin data sharing within the project. The first shared dataset will come from the Finnish pilot, serving as an initial test case for the project's data space. The user interface and rulebook will then be refined and iteratively adapted in response to user feedback.

The Data Sharing through the Data Spaces is close to work covered in the Data Management Plan as was initiated in the deliverable D1.3[24]. The data shared through the data spaces is part of the project data life-cycle and contributes to the data management by increasing the level of the FAIR attributes of the dataset shared through the data spaces. Information on the datasets shared through the data spaces will be incorporated in the Data Management Plan revised and final report.

[24] D1.3 Data Management Plan, INDEPENDENT project deliverable. 2025.

5. Data-driven Tool for Investment Analysis

This chapter provides a structured overview of a data-driven tool developed for technological and economical assessment of photovoltaic (PV) systems, battery energy storage systems (BESS), and heat pumps (HP). The tool combines Python-based numerical simulation with Excel-based economic reporting, ensuring computational rigor and practical usability.
This tool was developed primarily for the Slovenian electricity market, and its assumptions, tariff structures, and regulatory logic reflect the current national framework. In particular, the model accounts for the termination of annual net metering, the introduction of net billing mechanisms, and recent reforms in network tariff design. Nevertheless, the methodological structure of the tool is intentionally generic: market-specific parameters such as tariff schemes, settlement periods, and price structures are treated as configurable inputs. This approach allows the tool to be adapted to other markets with similar regulatory characteristics, provided that local pricing rules and grid charges are properly defined.
Its development reflects current energy trends, including regulatory changes, dynamic pricing schemes, and prosumer-oriented energy markets.

5.1. Introduction

The shift toward decentralized energy generation and the discontinuation of annual net-metering mechanisms require investors to evaluate energy solutions through high-resolution consumption and production models. The introduction of a five-tier time-of-use network tariff structure, real-time pricing elements, and increasing electricity costs further reinforce the need for detailed simulation.
The five-tier time-of-use network tariff structure and real-time pricing elements implemented in the model correspond to the current Slovenian regulatory framework, which entered into force in October 2024. These tariffs were introduced to achieve temporal differentiation of network charges, explicitly incentivizing load shifting, flexibility, and optimized use of distributed energy resources. Within the tool, this structure is implemented as a parameterized tariff model rather than a hard-coded assumption, enabling sensitivity analyses and potential application to other markets that employ multi-period or dynamic network pricing schemes.
The tool is designed to model 15-minute or hourly energy flows; evaluate PV generation, HP consumption, and BESS operation; and compute economic indicators that support investment decisions.
Its hybrid architecture follows established academic principles of model separation, transparency, and reproducibility.

- Python for large data calculations and computation
- Excel for visualization

5.2. Tools and Methodological Framework

The simulation and analysis environment consists of two primary technological components:

- Python: handles numerical modeling, time-series processing, BESS control logic, tariff application, costs calculation and aggregation.
- Excel: provides a flexible interface for generating investment reports, KPI summaries, financial projections, and scenario visualizations.

The Python–Excel environment can be formalized as a two-step transformation from input data to investor-oriented outputs. Python performs the numerical transformation, while Excel structures the simulated results into decision-support indicators: [25]

- **Python simulation engine:**
  $Y_{sim} = f(X_{input}, D_{time\text{-}series}, R_{tariff}, L_{BESS})$.
  This expression summarizes the numerical role of Python without repeating the detailed modelling tasks described above.
- **Time-series transformation:** $Y_t = h(D_t, X_{tech}, X_{econ})$. At each simulation interval, measured or generated time-series data are combined with technical and economic assumptions to produce interval-level model outputs.
- **Simulation aggregation:** $Y_{annual} = \Sigma_{t=1..n} Y_t$. The interval-level outputs are subsequently aggregated to annual values, which represent the analytical basis for economic and technical interpretation.
- **Cost calculation:** $C_{sim} = \Sigma_{t=1..n}(E_{grid\text{-}t} \cdot p_{energy\text{-}t} + NC_t + Tax_t - R_{export\text{-}t})$. The simulated cost is obtained by summing, over all time intervals, the energy cost, network charge components, taxes, and export-related compensation.
- **Excel reporting layer:**
  $R_{Excel} = g(Y_{sim}, A_{fin}, S_{scenario})$.
  This expression summarizes the reporting role of Excel as a transformation of simulation outputs into financial and scenario-based decision indicators.
- **KPI generation:** $KPI = q(Y_{annual}, C_{sim}, I_{net})$. The reported indicators are derived from aggregated simulation outputs, total simulated costs, and the net investment value.
- **Scenario comparison:** $\Delta KPI_s = KPI_s - KPI_{base}$. Scenario-specific indicators are compared with a baseline case in order to quantify the incremental effect of alternative investment or operating strategies.

[25] **Legend of Abbreviations and Symbols:**
Y_{sim} = complete set of simulation outputs generated by the Python engine.
X_{input} = general input assumptions used in the simulation.
f(...) = numerical transformation function implemented in Python.
$D_{time\text{-}series}$ = input time-series data, such as consumption, production, or price profiles.
R_{tariff} = applicable tariff rules used in the simulation.
L_{BESS} = BESS control logic applied by the simulation engine.
Y_t = interval-level simulation output in time interval t.
D_t = time-series input value in time interval t.
X_{tech} = technical assumptions, such as PV, HP, or BESS parameters.
h(…) = interval-level transformation function for time-series simulation.
X_{econ} = economic assumptions, such as prices, taxes, subsidies, or financing terms.
Y_{annual} = aggregated annual simulation output.
t = simulation time interval; n = total number of intervals in the simulation horizon.
C_{sim} = total simulated cost.
$E_{grid\text{-}t}$ = electricity imported from the grid in time interval t.
$p_{energy\text{-}t}$ = electricity energy price in time interval t.
NC_t = network charge cost in time interval t.
Tax_t = taxes and levies in time interval t.
$R_{export\text{-}t}$ = export revenue or export compensation in time interval t.
R_{Excel} = Excel-based reporting output.
g(...) = reporting transformation function implemented in Excel.
A_{fin} = financial assumptions used in Excel reporting.
$S_{scenario}$ = scenario definition used for comparison.
KPI = key performance indicator.
q(...) = KPI calculation function used to derive investor-relevant indicators.
I_{net} = net investment used in KPI calculation.
KPI_s = KPI value for scenario s.
KPI_{base} = KPI value for the baseline scenario.
ΔKPI_s = difference between scenario-specific KPI and baseline KPI.

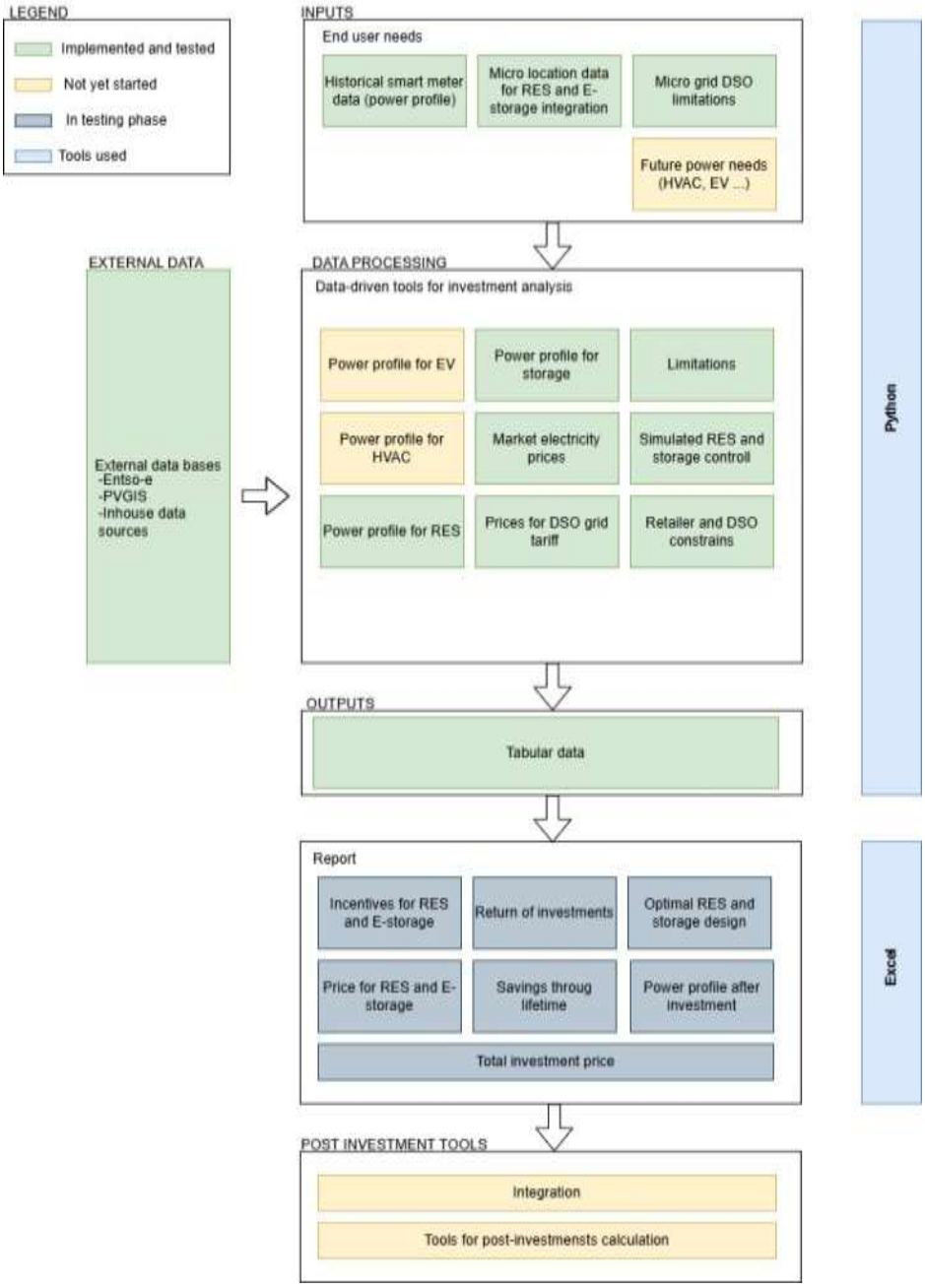


Figure 7 Schematic architecture of tool for investment analysis.

This architecture shown in Figure 7 ensures that computational complexity does not restrict user experience while maintaining methodological transparency.

5.3. System Architecture

The overall architecture of the tool includes the following elements:

1. Input parameters
2. External data sources
3. Data-processing unit (Python simulation engine)
4. Output economic and statistical reports
5. Post-investment validation tools

This modular design facilitates updates, version tracking, and integration of new tariff models, BESS control strategies, and consumption/production profiles.

5.3.1. Input Parameters

Input parameters are divided into technical characteristics, consumption/production profiles, and economic variables.

- Technical information for the measurement point
- PV system details (nominal power, orientation, slope, or empirical PV production profile)
- HP consumption or nominal power
- BESS capacity, power, and efficiency parameters
- Grid cost parameters (energy price, power tariffs, taxes, the new five-tier network tariff model)

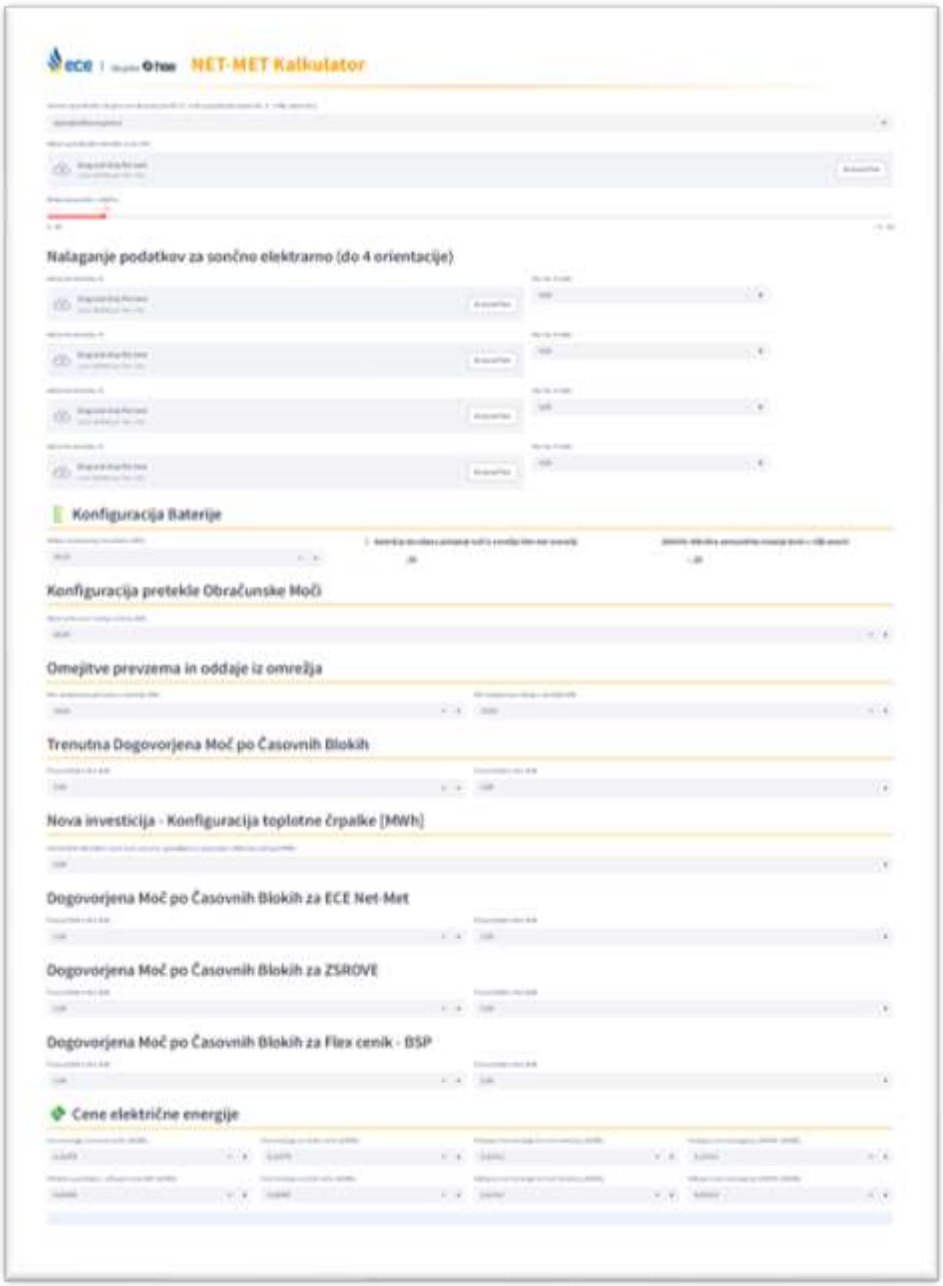


Figure 8 User interface for simulation tool developed in Python.

The simulation input interface shown in Figure 8 provides a structured environment for defining the key technical and economic parameters required to model the operation of a distributed energy system. Users select the consumer category, upload consumption and photovoltaic production profiles, configure battery

storage parameters, and define grid-related constraints and electricity prices relevant to the simulation scenarios.

The interface is implemented in the Slovenian language, as it is primarily developed for use on the Slovenian electricity market.

These parameters define the boundary conditions for all subsequent simulations.

5.3.2. External Data Sources

The tool integrates external datasets to ensure robust modelling:

- Historical consumption data
- Standardized PV production datasets based on geographical location, slope and orientation of the PV production site
- Real-time BSP flexibility prices (where applicable)

Such datasets enable the simulation engine to reflect variability in environmental and market conditions.

5.3.3. Simulation Procedures

The simulation engine evaluates:

- 15-minute or hourly production and consumption
- Grid costs, taxes, and dynamic tariff periods
- BESS energy flows and state-of-charge behaviour
- Import/export profiles under different BESS scheduling strategies

Three primary BESS operation models based on available types of business models will be simulated:

1. **NET–MET Model** (developed in first phase)
   The model represents a net billing self-consumption framework with annual settlement, in which electricity imported from and exported to the grid is valued at the same energy price. Within the scope of the project, this model is being developed and evaluated as a pilot concept, drawing on principles of former net-metering schemes while adapting them to the current regulatory environment in the Slovenian electricity market.
   Within this framework, the battery energy storage system operates dynamically to balance short-term differences between on-site production and consumption; however, the overall economic outcome is determined primarily by the annual net energy balance, which limits the relevance of short-term price arbitrage. The NET–MET[26] model therefore serves as a transitional reference scenario, enabling comparison between legacy net-metering concepts and newer market-oriented self-consumption models.

2. **BSP Model**
   The BESS follows a predetermined charge–discharge schedule, while financial performance depends on fluctuating BSP flexibility prices. This model is suitable when the battery provides system services rather than maximizing onsite self-consumption.

3. **ZSROVE Model**
   The BESS prioritizes maximizing PV self-consumption because import prices exceed export compensation. This model fits prosumers aiming to reduce grid dependence and minimize costs in systems with differentiated tariffs represent a market framework based on basic electricity supply offers defined in accordance with the Slovenian Act on the Promotion of the Use of Renewable Energy Sources (Zakon o spodbujanju rabe obnovljivih virov energije – ZSROVE)

These scenarios capture the regulatory and operational diversity present in the Slovenian energy market described in 5.1. In the first phase the simulation of NET-MET model was developed.

[26] https://www.ece.si/produkt/elektricna-energija-paket-samooskrba-net-met/

5.4. Economic Evaluation Framework

Economic evaluation combines classical financial indicators with energy-specific KPIs. The Excel tool includes transparent financial tables, visual outputs, and comparative analysis.
The following concepts are embedded:

- Simple Payback Period: years required to recover initial investment
- Break-Even Point: threshold where savings equal costs
- Lifetime[27] Savings : long-term accumulated economic benefits
- CO_2 Emission Savings: quantification of environmental benefits

Since the tool is intended for the average customer, it is logical that it should provide information that is significant and easily understandable to investors. Both concepts—simple payback period and break-even point—are straightforward, intuitive, and require minimal financial expertise, which makes them ideal for average investors. They answer key questions such as: “How soon will I recover my investment?” and “What is the minimum performance required to avoid losses?” This simplicity supports quick decision-making and reduces the need for complex financial modelling. The simple payback period indicates how long it takes for an investment to recover its initial cost through net cash inflows, making it a practical measure of liquidity and risk. The break-even point shows the level of sales or revenue at which total costs equal total revenues, resulting in neither profit nor loss.
The tool will be expanded to support multiple business models, enabling the definition of custom endpoints and tailored analysis for various investment options.
These measures collectively describe liquidity, profitability, cost-competitiveness, and sustainability.

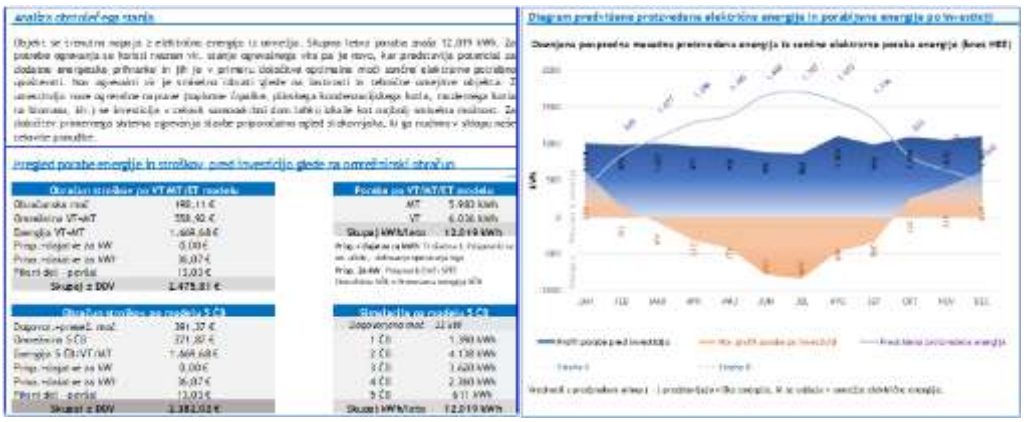
5.4.1. Technical and Economic Outputs

The Excel report includes:
Technical outputs:

- PV self-consumption and self-sufficiency
- BESS cycling, state-of-charge patterns
- Import and export volumes
- Peak load estimations
- Time-based energy balances

Economic outputs:

- Annual cost comparison (before/after investment)
- Scenario comparison that will be added in the future (BSP, NET–MET, ZSROVE)
- Subsidy calculation
- Loan repayment structure
- Annual net savings



[27] Lifetime refers to the operational lifetime of the solar power plant (30 years) and includes potential costs associated with BESS refurbishment

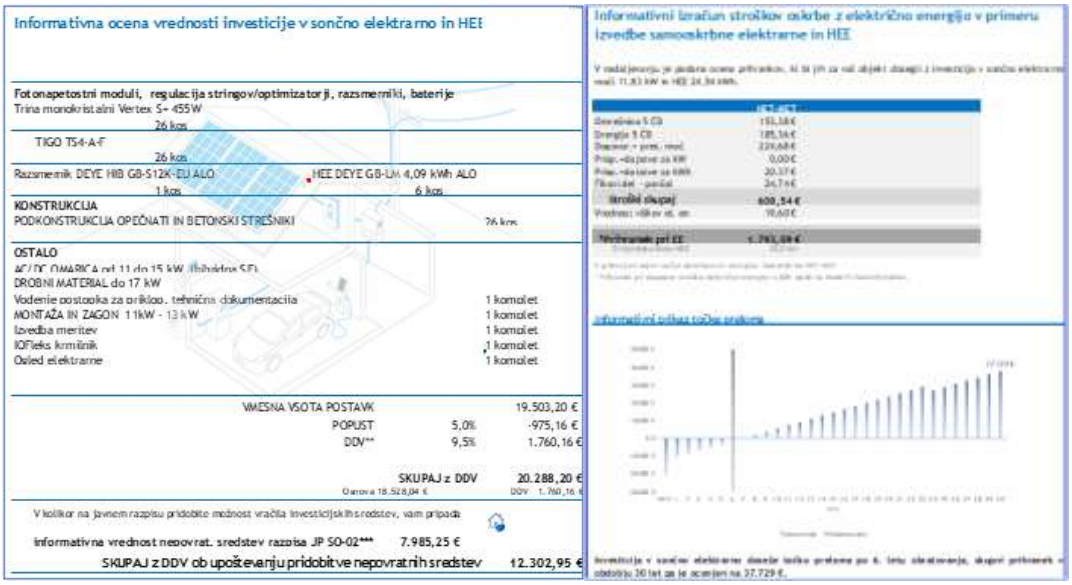


Figure 9 Sample of some technical and economic outputs visible to customers.

Figure 9 shows a collage of some selected outputs generated by the Excel reporting tool. The top-left section presents an analysis of the existing state, including current electricity consumption, energy costs, and grid-related charges. The top-right section illustrates the estimated monthly balance between photovoltaic electricity production and electricity consumption after the investment, visualized through a comparative energy diagram. The bottom-left section provides an informative breakdown of the proposed investment, including main technical components of the PV system and battery energy storage system, as well as indicative investment costs and available subsidy. The bottom-right section displays the calculated electricity supply costs and savings after the implementation of a self-consumption system with battery storage, including an indicative payback period visualization.

These outputs support evidence-based investment decision-making.

5.4.2. Post-Investment Validation and Deviation Analysis

A planned extension of the tool will compare simulated results with real measured data after system installation. A key component is validation of simulated results against real operating data.

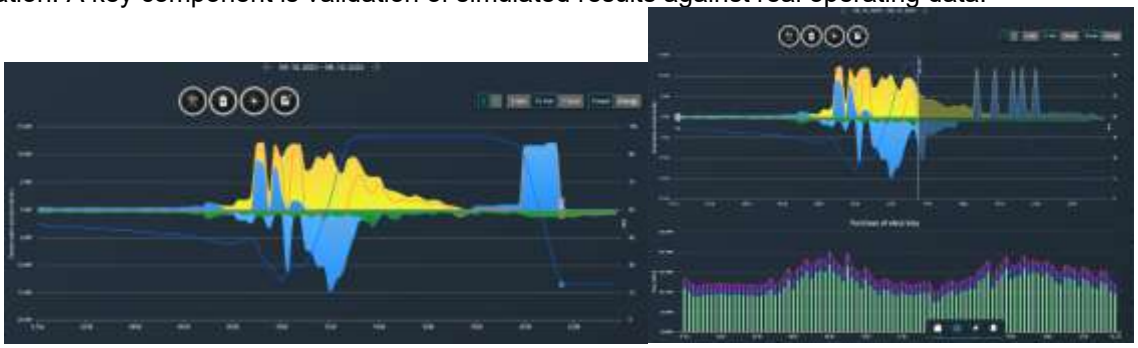


Figure 10 Graphs that indicates the deviations between simulation output and real data.

The Figure 10 indicate deviation between the simulated outputs and the observed empirical data (after approx. 13:30 hour), suggesting that the model does not capture the underlying system dynamics with sufficient accuracy.

Deviation analysis will focus on:

- Weather deviations vs. assumed PV profiles
- Behavioural deviations from expected consumption
- BESS performance differences (efficiency, limits, cycling)
- Tariff model mismatches
- Sensor or measurement inaccuracies

Such a feedback loop significantly increases the value of the analytical tool, transforming it from a pre-investment evaluation system into a lifecycle-support instrument.In this context, the improvement process can be structured using a Plan–Do–Check–Act (PDCA) logic, where simulation assumptions are defined, operational behavior is evaluated against measured data, deviations are assessed, and model parameters are adjusted accordingly. This conceptual framework provides a clear and systematic way to organize iterative model refinement without prescribing a specific external methodology.

A PDCA cycle (Plan–Do–Check–Act) is proposed to continuously update model parameters and improve prediction accuracy.

5.5. Challenges faced

Several challenges were identified during the development of the tool, largely driven by regulatory changes and data availability constraints specific to the Slovenian electricity market. A major challenge was the introduction of the new network tariff methodology in October 2024, which replaced the previous calculation approach with a time-block–based system and differentiated network charges. This change required substantial adaptation of the tariff logic, input parameters, and simulation structure to correctly reflect contracted power levels, time blocks, and their impact on total electricity costs.

Another significant challenge concerned the availability of high-resolution consumption data for heat pumps. While heat pumps represent an increasingly important share of residential electricity demand, measured 15-minute profiles are still not widely available, making it necessary to rely on interpolated or representative profiles in many cases. This introduces additional uncertainty and requires careful handling within the simulation framework.

A challenge in BSP price modelling is that until recently, BSP prices were not available in a consistent 15-minute resolution. Therefore, the simulation currently relies on hourly BSP price data, which may not fully reflect current market conditions and introduces a potential source of inaccuracy, especially for short-term flexibility analyses.

The workflow is based on exporting results from the Python simulation engine and importing them into Excel for further analysis. To ensure reliability, output data formats are standardized, including time resolution, structure, and units, enabling consistent interpretation and comparison of results across different simulation scenarios.

5.6. Conclusion

The tool provides a robust and practical framework for evaluating PV, HP, and BESS investments. Its modular architecture, integration of economic theory, scenario-based modelling, and post-investment validation capability make it suitable for research, industry analysis, and customer decision support.
Further development will include improved BESS intelligence, extended databases for consumption and production profiles, additional regulatory support, and enhanced financial modelling features.

5.7. Outlook

Further development of the investment analysis tool will primarily focus on improving the accuracy of simulation results by incorporating real post-installation measurement data. Comparing simulated outputs with actual system behavior will enable continuous calibration of consumption profiles, PV production models, and BESS performance assumptions, thereby increasing the reliability of future analyses.

In addition, the tool will benefit from results generated in other work packages and tasks, particularly in the areas of heat pump consumption modelling and electricity price forecasting. The simulation framework will be extended with higher-resolution databases for heat pump consumption and PV generation across different system configurations, as well as with forecast-based and 15-minute BSP price data to support more advanced flexibility modelling. Furthermore, additional BESS operation models will be introduced to reflect a broader range of operational and business scenarios. These upgrades will allow the tool to better capture ongoing

regulatory changes, dynamic tariff structures, and evolving prosumer-oriented market conditions, ensuring its long-term relevance for both academic research and practical investment.

6. Conclusion

This document provides information on the work done in WP5 tasks T5.1, T5.2, and T5.3, in addition to the outlooks for the next period in each task.

The task 5.1 sets up the shared development environment used across the project. It brings together version-control tools, automated build and deployment pipelines, and the systems needed to document and operate the IDOP. It also introduces monitoring and Kubernetes orchestration so that all services can be deployed and updated in a coordinated way. The security and privacy choices defined in the project are carried through here and built directly into the workflow. The goal is a development setup where creating, securing, and running the system all flow together.

The task 5.2 creates a common data space that different partners can trust and use. The work sets up the components needed for controlled data exchange with a minimum viable architecture set up and implemented. The data space supports both service development and investment-related analysis making data sources easy to find, reuse, and extend while keeping the whole environment accountable.

The task 5.3 presents a data-driven tool already offering a solid foundation for assessing PV, heat pump, and battery storage investments, combining modular modelling, economic rigor, and the ability to validate assumptions after installation. The next development steps include richer datasets with expanded regulatory coverage and strengthened financial modelling. This aims to push accuracy further. The tool will evolve into an even more reliable guide for research with real measurement data integrated.

The work progress presented here is a prototype of the package to be extended and evaluated. The Final DevSecOps & Investments Support Package (D5.2) will present a complete implementation of the package accompanied with a public documentation that demonstrates how the modules are used in practice.

7. List of Figures

Figure 1 DevSecOps pipeline workflow 7
Figure 2 Kubernetes deployment tool workflow 9
Figure 3 Logical architecture diagram of the implemented MVDS. To maintain visual clarity, the registration of the metadata broker with the identity provider is not depicted. The DAPS provides a DAT and the CA issues a certificate for the metadata broker through an analogous interaction to that between the identity provider and the connectors. 13
Figure 4 Data space user interface. Creating a data offer and filling metadata fields. 15
Figure 5 Data space user interface. Data search. 16
Figure 6 Data space user interface. Consuming the selected offer. 16
Figure 7 Schematic architecture of tool for investment analysis. 20
Figure 8 User interface for simulation tool developed in Python. 21
Figure 9 Sample of some technical and economic outputs visible to customers. 24
Figure 10 Graphs that indicates the deviations between simulation output and real data. 24