

EFPP: European Connected Factory Platform for Agile Manufacturing



European Factory
Platform

WP5: EFPP Add-ons

D5.16: EFPP Interfacing, Evolution and Extension – Final Report Vs: 1.0

Deliverable Lead and Editor: Anouar Mabrouk, ASC

Contributing Partners: ASC, CERTH, C2K, FIT, ICE, LINKS, SRDC, SRFG, VLC

Date: 2022-12

Dissemination: Public

Status: EU Approved

Short Abstract

This deliverable is an update to **D5.13: EFPP Interfacing, Evolution and Extension** that was published in M18 of the project duration, in June 2020. The deliverable summarizes the final outcomes of tasks: T3.3 “Integrated Marketplace Framework and Realisation”, T5.2 “EFPP Portal” and T2.5 “Ecosystem, Evolution and Extensions Requirements”.

Grant Agreement:
825075



Document Status

Deliverable Lead	Anouar Mabrouk, ASC
Internal Reviewer 1	Simon Osborne, C2K
Internal Reviewer 2	Martin Lorenz, ASI
Type	Deliverable
Work Package	WP2: Requirements Elicitation and Pilot Scenarios WP5: EFPP Add-ons
ID	D5.16: EFPP Interfacing, Evolution and Extension
Due Date	2022-12
Delivery Date	2022-12
Status	EU Approved

History

See Annex B.

Status

This deliverable is subject to final acceptance by the European Commission.

Further Information

www.efpf.org

Disclaimer

The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.

Project Partners:



Executive Summary

This deliverable summarizes the activities that took place in three interrelated tasks during the EFPP project. These tasks relate to Ecosystem, Evolution and Extension Requirements; Integrated Marketplace and the EFPP Portal. The deliverable presents key rationale of these tasks, along with the overview of the technical progress and the latest status.

The EFPP Marketplace has been extended reshaped and redesigned to serve a much-improved user experience. The marketplace has been also extended with connections to additional external marketplaces and with an automated agent-based marketplace, which supports online bidding. Additionally, to further support the business case of the EFPP Marketplace, the Accountancy Service has been integrated to provide event logging and invoice data retrieval and generation from connected marketplaces."

The EFPP Portal has been set as the unified entry point of the whole EFPP platform allowing users to register, login, browse and access all the tools and services provided by the ecosystem. Since then, the EFPP Portal has grown in terms of tools, platforms and documentations

Table of Contents

0	Introduction	6
1	Ecosystem, Evolution and Extensions Requirements	8
1.1	Requirements for a Federated Ecosystem	8
1.2	Requirements for the Evolution of a Federated Ecosystem.....	12
1.3	Requirements for the Extension of a Federated Ecosystem.....	14
1.4	Positive Governance for the Growth of the EFPF Platform Ecosystem	16
2	EFPF Marketplace	19
2.1	Integrated Marketplace Framework.....	19
2.1.1	Scope and Relationship with Other EFPF Components.....	20
2.1.2	Requirements and Realisation	25
2.1.3	Deployment.....	26
2.1.4	Execution and Usage	27
2.1.5	Limitations and Further Development.....	27
2.2	Automated Agent-based Marketplace and Online Bidding	28
2.2.1	Scope and Relationship with Other EFPF Components.....	28
2.2.2	Requirements and Realisation	30
2.2.3	Deployment.....	33
2.2.4	Execution and Usage	39
2.2.5	Major Updates from M18.....	43
2.3	Accountancy Service	43
2.3.1	Scope and Relationship with Other EFPF Components.....	43
2.3.2	Requirements and Realisation	44
2.3.3	Deployment.....	44
2.3.4	Execution and Usage	44
2.3.5	Limitations and Further Development.....	47
3	EFPF Portal	48
3.1	Current Status	48
3.1.1	User Resgistration.....	49
3.1.2	Login	49
3.1.3	Company Registration.....	50
3.1.4	Dashboard.....	52
3.1.5	Value Propositions	53
3.1.6	Federated Search	54
3.2	Scope and Relationship with Other Components	56
3.3	Requirements and Realisation.....	57
3.4	Deployment in the EFPF Platform (Installation).....	57
3.5	Execution and Usage	58
3.6	Limitations and Further Developments	58
4	Conclusion and Outlook.....	59
	Annex A: History	60
	Annex B: References	60
	Annex C: Data Models and Interfaces	Error! Bookmark not defined.

0 Introduction

0.1 EFPF Project Overview

EFPF - European Connected Factory Platform for Agile Manufacturing - is a project funded by the H2020 Framework Programme of the European Commission under Grant Agreement 825075 and conducted from January 2019 until December 2022. It engages 30 partners (Users, Technology Providers, Consultants and Research Institutes) from 11 countries with a total budget of circa 16M€. Further information: www.efpf.org

In order to foster the growth of a pan-European platform ecosystem that enables the transition from “analogue-first” mass production, to “digital twins” and lot-size-one manufacturing, the EFPF project will design, build and operate a federated digital manufacturing platform. The Platform will be bootstrapped by interlinking the four base platforms from FoF-11-2016 cluster funded by the European Commission, early on. This will set the foundation for the development of EFPF Data Spine and the associated toolsets to fully connect the existing platforms, toolsets and user communities of the 4 base platforms. The federated EFPF platform will also be offered to new users through a unified Portal with value-added features such as single sign-on (SSO), user access management functionalities to hide the complexity of dealing with different platform and solution providers.

0.2 Deliverable Purpose and Scope

This is a final report presented at M48 milestone of the EFPF project. The purpose of this document “D5.16 EFPF Interfacing Evolution and Extension – Final Report”, is to present a technical overview of the EFPF Portal services as well as EFPF Integrated Marketplace and a final update about the ecosystem creation, evolution, and extensions of the EFPF platform. This deliverable articulates the vision behind introduction and continuous development of these services as well as their benefits to prospective users. The report presents an account of the development made since project start, with more focus on elaborating new developments after M18.

0.3 Target Audience

This deliverable aims primarily at external platform and marketplace provider, which want to offer their services and tools through the EFPF platform.

0.4 Deliverable Context

This document provides information to multiple software components of the EFPF platform. Its relationship to other documents is as follows:

- **D2.4: EFPF Platform Requirements:** Provides detailed information about the EFPF platform and its requirements.
- **D5.2: EFPF Security and Governance:** Provides detailed information about security, privacy, and governance in the context of EFPF.

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1: Ecosystem, Evolution and Extensions Requirements:** Provides information about the implementation of requirements needed for ecosystem creation, evolution and extension of the EFPF platform.
- **Section 2: EFPF Marketplace:** Provides information about the software component produced by Task 3.3: Integrated Marketplace Framework and Realisation.
- **Section 2.3.1: Scope and Relationship with Other EFPF Components**

The Accountancy Service has been developed within the project as an integral part of the EFPF Marketplace Framework and provides insight into users' interactions with the EFPF Platform as well as its connected marketplaces. This includes transactions that EFPF users make on different marketplaces, which are linked with the EFPF Marketplace Framework.

A taxonomy has been setup to identify the trackable user actions in which action items are listed in 'subject, verb, object' manner and these actions include users' basic interactions with various parts of the EFPF Platform such as login, register, inviting other users as well as payments realized on external marketplaces if the user has initiated his/her journey from EFPF Marketplace. In this way, when a user performs a certain action on either EFPF Portal or a connected marketplace, corresponding information is sent to Accountancy Service to be persisted so that it can later be visualized to extract valuable information.

The Accountancy Service has been developed based on Elastic Stack which comprises the following components:

- **Elasticsearch:** Stores, indexes, provides, and manages user logs to be later analysed. Since relational databases are not well-suited for managing log data, a NoSQL database like Elasticsearch is preferred due to their flexible and schema-free document structures, enabling analytics of the log data.
- **Logstash:** Gathers user behaviour data from various components of the EFPF platform, executes different transformations and filters the content, before sending the data to the Elasticsearch component
- **Kibana:** Enables interactive dashboards, filters and advanced data analysis and exploration of user logs.

In addition, the following custom modules listed below were developed to provide additional functionality:

- **Reporting Component:** Creates periodic (i.e. monthly) reports for each dashboard at the end of each month in PDF format and sends it as an email
- **Invoicing Component:** Processes all the payment data accumulated within each month, sums all the amounts from successful transactions realized on each marketplace, calculates a corresponding cashback amount, and creates a detailed invoice with the information including purchased products, dates of transactions as well as the calculated commission for each product. The invoice will then be used to charge marketplaces.

0.5.1 Requirements and Realisation

Tracking the user behaviour enables businesses to make productive decisions and develop effective business strategies. This is a valuable feature of a federated digital platforms, and the Accountancy Service provides this to support the long-term sustainability of the EFPF

platform, beyond the span of the project. The Accountancy Service also aims to track and trace a user’s journey across the EFPF ecosystem and collect data about the transactions they make on different marketplaces. The collected data logs are then used to carry out a cashback mechanism enabling a commission charge or a referral fee to be made to the marketplace where a EFPF user carries out a business transaction (Figure 23). In addition to the log collection, customizable dashboards are also needed for better and easier tracking of user interactions. Therefore, in order to process the accumulated log data and address all the above requirements, the Accountancy Service uses Elastic Stack (Elasticsearch, Logstash, Kibana) as an advanced log persistence, monitoring, processing, and visualization framework.

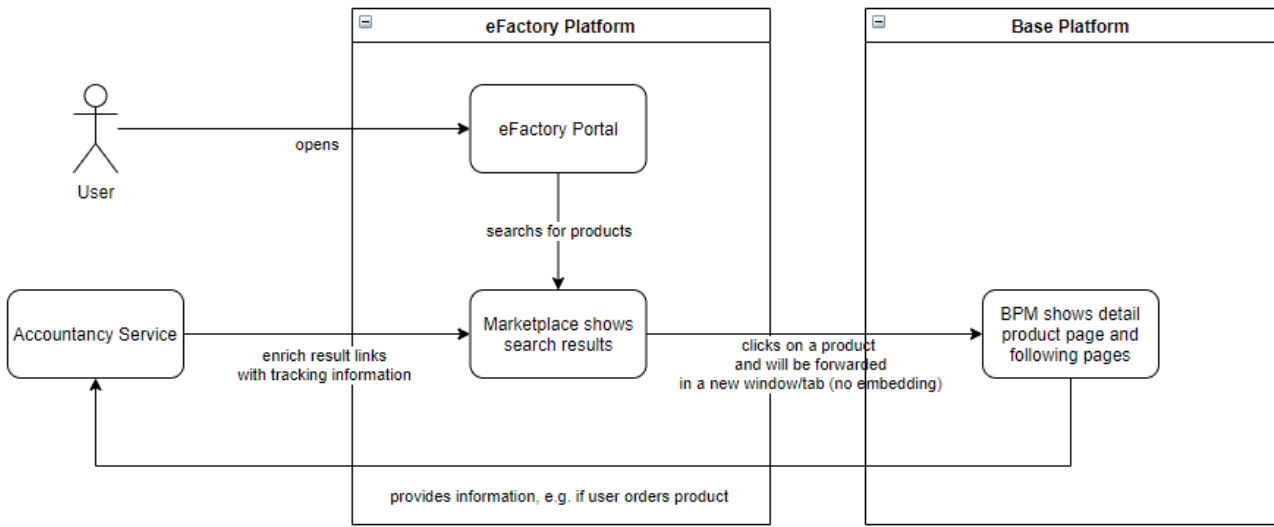


Figure 23: Accountancy Service - Cashback Process

0.5.2 Deployment

Accountancy Service is a standalone component that runs independently of existing EFPF tools and services and can be integrated with unlimited number of external marketplaces. Each of the 5 components of the Accountancy Service has its own Docker image and runs on a corresponding Docker container. Moreover, a production-ready configuration allowing Elasticsearch running on a multi-node cluster is also available. Currently, all the components of the Accountancy Service run on a production server hosted by C2K and a test server hosted by SRDC.

0.5.3 Execution and Usage

Accountancy Service uses Logstash as a data ingestion and server-side data processing pipeline. The logs sent to Logstash are forwarded to Elasticsearch for persistence after executing certain ingestion pipelines; and then Kibana dashboards are automatically updated based on the certain fields stored on Elasticsearch. In other words, Accountancy Service uses a basic data model for visualization and logs must conform to a basic data model so that Kibana dashboards can be updated automatically.

Currently, there is a running instance of Logstash component that is registered to the EFPF Service Registry, which is publicly available through a public endpoint so that events from EFPF Portal and external marketplaces can be sent using HTTP POST method. Events are modelled as a JSON message related with the action conforming to the data model. This will be enough for the Accountancy Service to capture the data and update the dashboards.

Accountancy Service uses Kibana to visualize the data ingested through Logstash and persisted on Elasticsearch. Kibana provides mechanisms to create interactive dashboards with filtering as well as advanced data analysis capabilities and Accountancy Service provides 4 different dashboards for log visualization. Furthermore, all these dashboards can either be accessed on the Kibana instance hosted on EFPF production environment and EFPF Portal Admin pages. Details of the 4 dashboards can be seen below:

- **Payments Dashboard:** Displays all payments realized on marketplaces as well as the corresponding cashback (commission) amounts calculated for each transaction (Figure 24).
- **Marketplace Usage Dashboard:** Visualizes marketplaces usages in terms of most frequently used search keywords, queried platforms, and their distribution (Figure 25 *Error! Reference source not found.*).
- **Platform Engagement Dashboard:** Displays base platform visits and tool/service usages and tracks the frequency of these usages (Figure 26).
- **User Activities:** Visualizes user actions such as login and register (Figure 27).

In addition to the central functionalities offered by the Elastic Stack, Accountancy Service also provides extra features such as preparing monthly reports based on the accumulated data and generate invoices in accordance with the commissions calculated for successful transactions that users perform on the external marketplaces connected to the EFPF Platform. Since these functionalities require custom implementations, these add-on modules are written in JavaScript and provided as a Docker image. At the end of each month, both modules process relevant data of the target month and generates 2 PDF documents: One for the monthly report including all charts updated for the target month and one for the invoice document containing all the transaction details (e.g. purchased products, their prices, transaction dates, calculated commission, etc.) to charge each external marketplace.

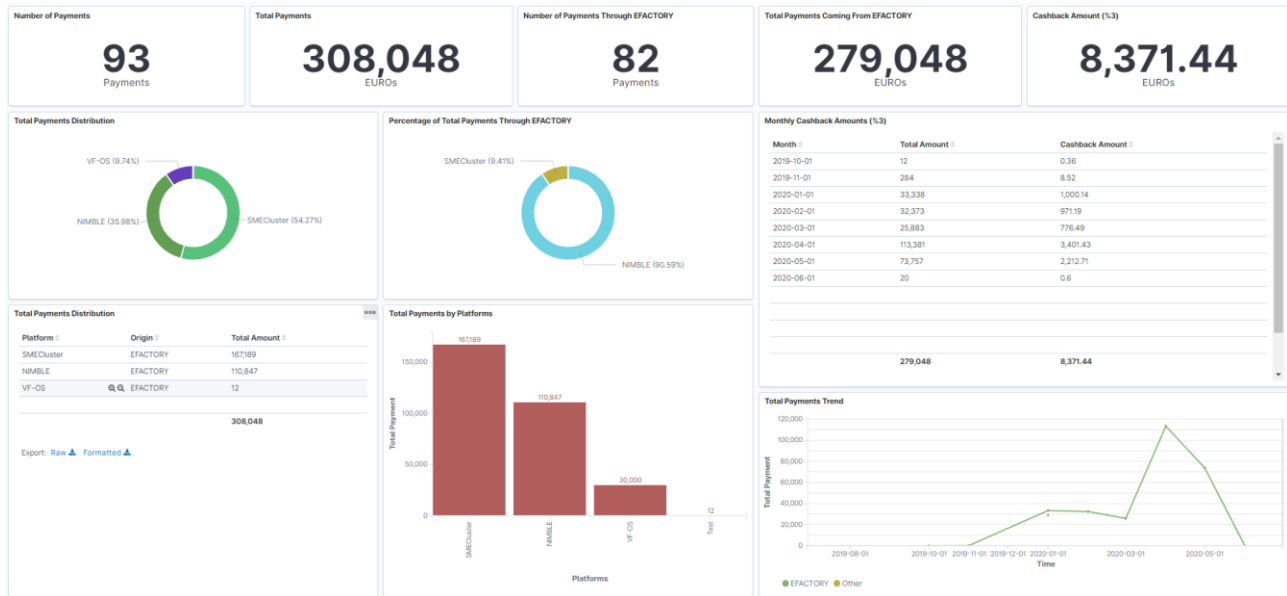


Figure 24: Accountancy Service - Payments Dashboard

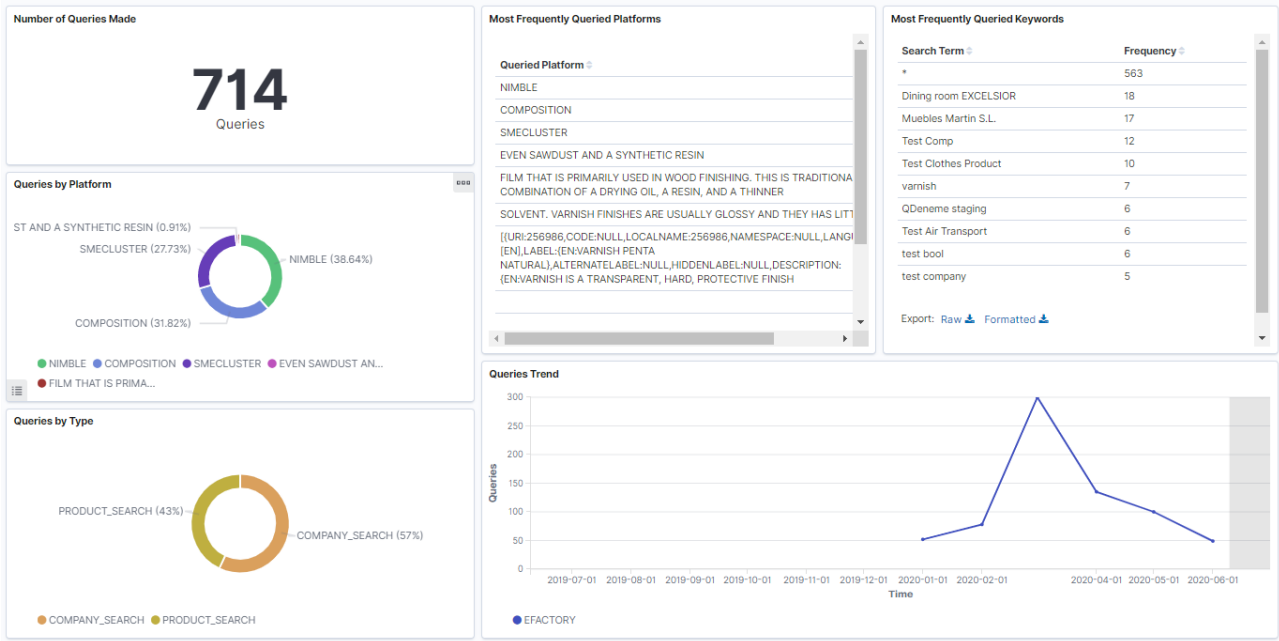


Figure 25. Accountancy Service Marketplace Usage Dashboard

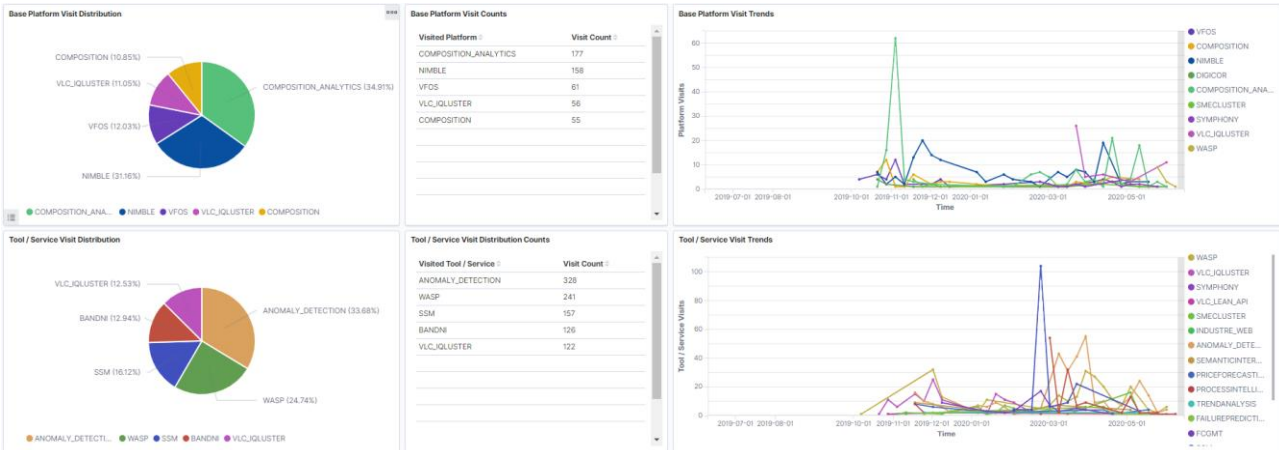


Figure 26: Accountancy Service - Platform Engagement Dashboard



Figure 27: Accountancy Service - User Activities Dashboard

0.5.4 Limitations and Further Development

Currently, there are no limitations regarding the Accountancy service development and previously identified limitations were resolved. New marketplaces can be integrated, and new dashboards will be created upon request to track different types of user interactions.

- EFPP Portal: Provides information about the software component produced by Task 5.2: EFPP Portal.
- **Section 4: Conclusion and Outlook:** Concludes this deliverable and provide a brief outlook for each component.
- **Annexes:**
 - **Annex A:** Document History
 - **Annex B:** References

0.6 Document Status

This document is listed in the Description of Action as public. Therefore, details which may temper the security of the platform, or its components are not content of this deliverable.

0.7 Document Dependencies

This document is final part of the deliverables that describes the EFPP portal together with the EFPP Integrated Marketplace and its subcomponents. The last iteration has been submitted in M18 in D5.3 - EFPP Interfacing, Evolution and Extension. This document summarize the update that took place since then.

0.8 Glossary and Abbreviations

A definition of common terms related to EFPP.

0.9 External Annexes and Supporting Documents

Annexes and Supporting Documents:

- None

0.10 Reading Notes

- None

1 Ecosystem, Evolution and Extensions Requirements

1.1 Requirements for a Federated Ecosystem

A core goal of the EFPF project is to develop an ecosystem that demonstrates a federated approach to functionality, processes, and users in a bid to ensure the initiative is sustainable. This

A core goal of the EFPF project has been to develop an ecosystem that demonstrates a federated approach to functionality, processes, and users in a bid to ensure the initiative is sustainable. This is motivated by the shift in architecture design in recent times in favour of microservices enabling enhanced flexibility, rather than the proprietary digital platform approach that has been attributed with vendor lock in, proprietary standards, inhibiting evolution.

A federated approach was needed to bring together different stakeholders into a level playing field where they could perform collaborative activities and develop relationships such as those between providers and suppliers of products and services. The federation model also supported the creation of competitiveness networks, which can be composed of multi-industry partners; and facilitates knowledge exchange between different stakeholders through collaborative processes. Requirements to guide these processes needed to be understood e.g. what is the source of the requirements, their type (whether user-oriented, non-functional, technical), who are they applicable to etc. In this respect, a dedicated task in the EFPF Project (T2.5: EFPF Interfacing, Evolution and Extension) gathered requirements from a range of ecosystem stakeholders starting with the partners involved in the task, which come from a variety of domains including Industrial, Research and Technical. Further requirements were collected from partner interactions, interactions and collaborations with different projects, from the open calls experimenters and other activities and stakeholders engaged through the project duration. Task 2.5 (T2.5) ran throughout the duration of the project with the aim to continuously feed new/emerging requirements into the project (development, business modelling, ecosystem creation etc) activities.

The requirements for a federated ecosystem gathered in the project are relevant to the following roles:

- **Developers:** Technical users who may develop and wish to deploy Tools and Services in the EFPF ecosystem
- **Systems Integrators:** Technical users who want to leverage the latest smart factory and digital manufacturing toolsets, in order to be able to offer them to customers
- **Platform managers:** Interested in extending the scope of an existing platform or to start a new platform
- **Manufacturing Managers:** Business focussed users who would like to access, reliable mature and productivity generating tools and services

The guiding requirements for the federated ecosystem, established through the EFPF project are described in D5.3 and an update on these is provided in this deliverable. These requirements are defined in the context of a federated ecosystem of the digital manufacturing platforms. Naturally, these requirements have evolved overtime as the project developed links with other stakeholders and explored new relationships (e.g. through open-call mechanism).

The guiding requirements for the federated ecosystem established through the EFPF project are described in D5.3 and copied below. Alongside these, justification of how each guiding requirement has been realised through the duration of project is also provided.

Guiding Requirement	Realisation Details
<p>A federated ecosystem should have multiple systems and platform virtually connected through software interfaces for seamless authorised access. Ideally, the connectivity should extend beyond the software accessibility and interoperability aspects to also support interactions between tools/service providers and user communities</p>	<p>The EFPF platform successfully integrated and federated the 4 key base platforms. In addition, the project also federates multiple external platforms such as SMECluster, ValueChain’s Network Portal, B2Bmarket, and ZDMP. To enable interactions between user communities, EFPF provides Matchmaking and Team Formation mechanisms that can be used to find suitable partners and collaborators. Details of the realisation are documented in D5.4 final report.</p>
<p>The federation should provide necessary trust, security, and privacy mechanisms to ensure that the partners and their interactions in the ecosystem are safe from potential threats like cyber-attacks, session riding, hijacking etc</p>	<p>The EFPF Platform developed and implemented the EFPF Security Portal to provide the necessary trust, security and privacy mechanism through the use of Industry Standards such as OAuth2.0 and Json Web Tokens (JWT’s). In addition, a comprehensive governance framework is developed (in WP5 and documented in D5.15) to provide necessary guidelines for preserving trust and privacy aspects. The prescribed guidelines are translated into technical requirements and implemented at the Platform and Portal level to address security and privacy concerns.</p>
<p>The federation should support the heterogeneity of the ecosystem and provide necessary governance mechanisms that allow the relevant control (data, access, business models etc) to remain within the partners</p>	<p>Based on the definition of a comprehensive platform governance framework, necessary governance mechanisms are implemented as terms and conditions and also security policies that have been put in place to support the control of data by respective data owners. The implementation of the governance mechanism not only covers the access and utilisation of the platform but also the exchange and use of data across different systems and services. In this respect, specific security policies are implemented to secure the Service Registry, pub-sub message bus and the API Security Gateway</p>
<p>The federation should implement governance procedures and technology</p>	<p>The governance framework (developed in WP5 and documented in D5.2 and D5.15) establish the essence of the federation</p>

<p>mechanism that act as a deterrent for any potential monopoly</p>	<p>model adopted by the EFPF platform. Based on the federation model, the EFPF platform provides level playing field for all platforms and systems. There are no preferences and privileges assigned to any specific platform or system in the federation and open nature of the EFPF platform allows for new entrants to be easily integrated in the federation through interoperability of the APIs supported by the Data Spine. Therefore, the user needs and ease of use are the key criteria of success expected to drive the popularity and growth of the federated platforms in the EFPF ecosystem.</p>
<p>The platforms in the federation should be conducive to integration and interoperability. The federation should allow users to connect and operate through APIs, applications, and third-party service libraries. This would further eliminate the possibilities of monopoly and vendor lock-in</p>	<p>While the interoperability of connected platforms was first realised with the integration of the 4 base platforms, the realisation of this guiding requirement has also been enhanced through the integration of additional platforms through respective API's. This has included the integration of platforms such as SMECluster, ValueChain's Network Portal, B2BMarket and ZDMP. The API connectivity is tracked and updated through a dedicated interface contract management mechanism developed in the EFPF platform</p>
<p>The federated ecosystem should provide an easy to use service integration and interoperability platform/mechanism to make the provision and consumption of services as easy as possible, enhance developer productivity and enable collaboration amongst them.</p>	<p>The platform integration process has been designed to allow new platforms to easily integrate their products, service and tools, through the Service Registry Tool. Through the registration of external marketplace API's in the Service Registry, federated platforms can easily integration current offerings in the EFPF marketplace. Detailed documentation is made available on an open documentation portal accessible through the EFPF Portal. The validation of service integration and interoperability has have been performed during the piloting and open-call experimentation phases of the project.</p>
<p>The service integration and interoperability platform/mechanism offered by the federated ecosystem should make use of standards and its design should be flexible enough to have the choice of multiple open source technologies to realise its</p>	<p>The integrations and interoperability in the EFPF federation are supported the Data Spine, which implements standardised data management techniques, and itself is standardised through a CWA. The Data Spine is designed as a flexible and open</p>

<p>conceptual components. This would also help in avoiding vendor lock-in.</p>	<p>mechanism that can be used to integrate platforms and services at different levels.</p>
<p>The service integration and interoperability platform/mechanism offered by the federated ecosystem should have inbuilt support for standard communication protocols and for data transformation tools/languages, such as XSLT (Extensible Stylesheet Language Transformations), that are widely used in the industry.</p>	<p>Through the development and implementation of the Data Spine's Integration Flow Engine, the EFPF platform is able to offer inbuilt support for data transformation to a wide array of formats / protocols/ standards. This is also evidenced by the implementation of indexing flows in federated search to federate data from individual platforms which all have different data formats.</p>
<p>The service integration and interoperability platform/mechanism offered by the federated ecosystem should be scalable and should support high availability and high throughput.</p>	<p>In the development of the Data Spine's Integration Flow Engine, designed as an interoperability mechanism, Apache NiFi was selected as the base technology due to the considerations for scalability and extensibility built in to the tool. This included extension points such as Processes, Controller Services, Reporting Tasks, Prioritizers, and Customer User Interfaces.</p>
<p>The federation should promote the use of standards for the design and implementation of components of the federated ecosystem to make the integration of new services, tools, and platforms effortless.</p>	<p>The EFPF Project has provided a focus on promoting standards throughout the project with OAuth2.0 standards implemented as a means for both securing applications and enabling SSO between federated platforms. Alongside this, the provision of the Data Spine Service registry which implements standards such as JSON, for the registration of services operating on a range of communication protocols such as HTTP, MQTT, AMQPS.</p>
<p>The federation should provide adequate documentation in the form of HowTos, Tutorials and API documentation of provided services in order to ease the development of applications using distributed services from the various platforms and service providers within the EFPF ecosystem. The federation should promote the use of standard deployment practices such as containerization to make the deployments of components of the federated ecosystem easy to manage and maintain.</p>	<p>The EFPF Project has provided extensive documentation to all components of the platform, including connected tools and services. This has been unified in central interface, The EFPF Dev Portal. In this documentation portal, where appropriate, each tools, service, or platforms has provided an array of guides including; Overview, Quickstart Guide, Admin Guide, Developer Guide & User Guide.</p>
<p>The federated ecosystem should inherently provide services, tools, utilities that are widely used to realise the common use cases in the domain.</p>	<p>Alongside the core Data Spine components, the EFP project has also provided a set of tools, or "ecosystem enablers", to support the realisation of common use cases. An example of this has</p>

	included the Access Consent Delegation Framework, or Pub Sub Security Service, to allow secure and user managed interaction with the Data Spine Message Bus
The legislative regulation across different regions may be different so it is important to have a fair geographical distribution of vendors or datacentres to sustain the federated ecosystem	The EFPF Platform and in particular, the EFPF Data Spine has been designed to be deployable in a distributed manner through clustering mechanisms. With this approach the EFPF Platform can be deployed across geographically distant servers.

1.2 Requirements for the Evolution of a Federated Ecosystem

The process of evolution of a federated ecosystem should be self-governing in nature and encompass change across all aspects of the characteristics that make up a federated system. Naturally, evolution happens over time based on many different factors that are often not controllable or even predictable (such as behaviours of different actors, environmental changes, interactions between different entities etc). As an ecosystem grows (e.g. with the number of actors, the interactions, different types of activities etc) the need for supporting new actors, new roles, new interactions as well as adaptation of existing systems and processes also grows. Therefore, the requirements for the evolution of a federated ecosystem need to highlight the need for certain level of flexibility, openness, and adaptability in the underlying federation (mechanisms) to ensure its evolution overtime.

While the guiding principles defined for the evolution of a federated ecosystem were defined in D5.13, the realisation of the guiding principles can be seen documented below.

Guiding Principle	Realisation Details
Diverse user base: This will ensure that the market vulnerability isn't a risk by naturally evolving to target users in new domains.	While the EFPF platform was initial setup and tested for the 3 pilot scenarios, the platforms user base was then extended through the projects' open call for experimentation, in which 20 sub-projects were selected from a diverse range of domains including Agricultural, B2B documentation exchange and project planning use cases.
Heterogeneity: Use of microservice architecture should support a natural evolution of functionality and process to support in an adaptive robust way.	The core components of the Data Spine have been setup under a microservice architectural approach to allow for the extensibility of the platform in a standardised manner.
Geographical distribution: This is also an essential characteristic, where evolution can be naturally supported through	The EFPF Platform has been designed to be deployed in a geographically distributed manner through clustering techniques.

distributed deployment of microservices and interoperability, but ultimately driven by human processes.	Through the development of integration and deployment scripts, microservices can be added and removed from the deployment stack as needed.
Interoperability: Interoperability mechanisms can support these evolutionary step changes	A core aspect of the EFPF Platform is the EFPF Data Spine, which provides platform users with mechanisms for interoperability through Data Spine components such as Service Registry and Integration Flow Engine, in which data transformation processes can be designed to enable or enhance interoperability.

Based on the above guiding principles, the guiding requirements for the evolution of a federated ecosystem were also defined in D5.13, and can be seen below, alongside a description of how each requirement has been realised within the Platform.

Guiding Requirement	Realisation Details
A federated ecosystem in essence should constantly evolve to include new partners, tools and services; in order cater to the users need in present and in the future.	Extensibility of the platform to new partners, tools and services has been demonstrated through the completion of the open call phase of the project, with the integration of 20 sub-projects represented by 22 additional partners, tools, and service within the EFPF federated ecosystem.
There should be feedback loop to capture user experiences and proper user interfaces should be defined to allow the multi-part interactions to take place and keep the ecosystem moving.	As part of the EFPF Open Call activity, the setup and deployment of the Tikki Ticketing platform allowed for a live feedback loop in which users could report any issues being faced or request support from EFPF partners and administrators. Tikki is available through the EFPF Portal and with the Integration to the EFPF Security Portal, this also allowed owners of deployed tools and service to directly respond to relevant issues in Tikki.
The interoperability feature of the platform should attract new software integration to develop opportunities for different types of interactions and business.	Through the completion of the open call for experimentation, the EFPF platform successfully integrated a range of new software components, thus increasing its overall offering and attractiveness to potential customers, including those in new domains such as Agriculture.
Overtime the ecosystem should adapt and accommodate new stakeholders and usage scenarios.	The adaptability of the platform and relevant ecosystem was also tested through the open-call where sub-projects from different domains and business scenarios tested existing solutions and developed new

	solutions to demonstrate the usage potential of EFPF services in diverse domains
In a federated ecosystem, the involved parties rely on the agreed API Contracts or Interface Contracts for communicating with each other. However, as the participant services evolve, the upgradation of APIs becomes necessary and inevitable. Therefore, the federated ecosystem should define Interface Contract policies that allow the Service Providers to convey plans to deprecate/upgrade their APIs to the Service Consumers in advance allowing a smoother transition/collaboration.	Through the design and development of the Data Spine Service Registry, it was ensured that an adequate versioning system was in place to enable API integrations to be managed more easily. Through the addition of versioning and expiry fields in the service registry schema, this enables developers to effectively plan around API upgrade and deprecations. In addition, integrations with the EFPF Message Bus has allowed for the announcement of service registrations, and de-registrations through subscription to a defined topic.

1.3 Requirements for the Extension of a Federated Ecosystem

To support the goals of evolution step changes, a framework of processes will be needed to support the concrete development steps and allow the extension to happen in a managed manner that is disseminated to all stakeholders. The microservices architecture and REST communications, rather than close coupled systems, naturally supports extension of the ecosystem. However, topics including deployment, version management, service lifecycles and protocol support will all need to be planned for and delegated to ensure the process of extending the ecosystem is handled particularly as the it grows.

The high-level guiding requirements defined in D5.13 for the extension of the platforms federated ecosystem have been listed below alongside a justification of how this guiding requirement has been realised throughout the course of the EFPF Project.

Guiding Requirement	Realisation Details
The federated platform should be extensible and there should be interfaces for the inclusion of new services and gateways available for data exchange.	The EFPF Platform has now developed and made publicly available, a range of comprehensive guides that document the process for the inclusion of new Tools and Services. Several guides have been developed depending on the particular scenario faced by the user, and are available in the EFPF Dev Portal.
There should be openly accessible APIs for the integration of external tools.	A range of API's have been made available in the EFPF Platform for the integration of new tools, mainly the Service Registry API, in which all registered users are able to view and discover currently registered services. Through obtaining the correct roles or permissions, users can then also add new or update existing services, due to the

	security provided by the API Security Gateway.
The federation should provide necessary support, not only in terms of interfaces and connectors, but also in terms of human resources (experts) who can facilitate the onboarding of new partners and technologies.	Through the establishment of Tikki, the dedicated ticketing support platform, a direct channel of communication was enabled between all EFPF users and the relevant experts from each component, tool, or service. In areas of support where issues may be expected to be reported more frequently, due to high usage (e.g. EFS Integration), dedicated teams were setup to increase the availability of expert resources and ensure all tickets can be handled in a timely fashion.
The federated ecosystem should promote cross domain and cross border interactions and therefore the extension mechanisms should be able to capture the differences in languages, approaches and techniques.	The cross-domain interactions were supported through a number of pilot and open-call applications. For example, a blockchain-based track and trace application in the CE pilot of the EFPF project facilitates the interactions between different types of users. Moreover, several open call sub-projects developed solutions that involved cross domain interactions and interconnectivity. However, the cross-border aspects (i.e. differences in languages, approaches and techniques) has not be tested so far. The reason for that is non-availability of relevant requirements and user needs both from the pilot partners and also from the open-call sub-projects.
Documentation should be available to describe the scenarios in which different extensions should be done.	Within the EFPF Platform, an extensive documentation hub has been provided, the EFPF Dev Portal, to support users in the scenarios they face. In this regard, 4 key guides were introduced to enable new users to get started with sufficient information. This included a “User Guide 101” for; “Tool/Service Consumers”, “Composite Application Developers”, “Tool, Service, Data Providers”, and “Platform Providers”.
The service integration and interoperability platform/mechanism offered by the federated ecosystem should follow a modular and extensible architecture, thereby making it possible to add support for new communication protocols, data transformation tools, etc.	Within the EFPF Date Spine, the Service Registry has been designed to support the registration of services independent of the communication protocol used and where the communication protocol can be defined. Although the service is predominately designed for the registration of REST API based services, Message Bus Topics can

	<p>also be registered with either MQTTS or AMQPS protocols. While it is expected the Data Spine Integration Flow Engine will be able to handle the data transformation needs of EFPP users, should a use case arise where NiFi is not considered fit for purpose, additional transformation tools, could also be added through the offered integration and deployment mechanism.</p>
--	--

1.4 Positive Governance for the Growth of the EFPP Platform Ecosystem

The complexity of digital platform ecosystems comes with the decision-making twist between different sectors and different actors operating at multiple levels, including organisational, sectoral, local, regional, national, international, etc. The EFPP is an emerging ecosystem of multi-sided digital platforms and requires governance mechanisms to be in place, to effectively reach its goals and create sustainable outcomes. The governance mechanisms for digital platform ecosystems need to reflect on the lawful interactions of key stakeholders, be they owners of the platforms, companies using the platform, or developers, users, advertisers, economists, computer scientists, governments, or regulators. To stimulate positive interaction payoffs within the platform ecosystem, both platform stakeholders and platform technology enablers must be regulated and governed.

The D5.2 "EFPP Security and Governance" describes the EFPP Governance Framework (eFGF) that incorporates platform organizational standards, strategic planning, business rules and norms of behaviour within the ecosystem, software standards, regulatory requirements, and other aspects which need to be continuously monitored and assessed from the perspective of various EFPP users (stakeholders).

The eFGF covers five functional areas:

- AREA 1: Terms and conditions ("Terms of Use", "User Agreement", or "Terms of Service Agreement"): it outlines the terms and conditions the user must agree to in order to interact with the EFPP platform. Well-established terms and condition-related rules prevent misunderstanding between the EFPP platform owner and the users, by defining e.g. Intellectual Property Rights (IPRs) protection, limiting responsibilities towards third parties, setting platform rules and the consequences for violating these rules, etc.
- AREA 2: Architecture and IT governance: it includes a policy-based control of information to meet all legal, regulatory, risk, and business demands. The focus is on the actual software development and maintenance activities of the IT that are aligned with the business objectives of the platform, e.g. with the European Factory Foundation.
- AREA 3: Data governance, service policies, APIs policies and SDKs include those rules that define i.e., processes and controls to ensure that information at the data level is true, accurate, and unique (not redundant). Data policies address policies related to personal data (data belonging to the user of the EFPP platform), corporate data (data belonging to the company that user of the EFPP platform represents), community data (data belonging to the group of users that get together over the EFPP platform for

sharing specific interests, e.g. forum, working teams, etc.), sectoral data (data that constitute a specific sector, e.g. Smart Manufacturing sector, Automotive Driving sector, etc.) and potential infrastructural data lakes (data that constitute a larger body of data, e.g. Common European Data Space [EC-DATA20]).

- AREA 4: Marketplace rules, trust and reputation that define the expected behaviour in the platform ecosystem, in order to create expected positive effects.
- AREA 5: The existing laws and regulations at the international and national levels, including the EU Network and Information Security (NIS) Directive with the goal to enhance cybersecurity across the EU; Incident notification for DSPs in the context of the NIS Directive; GDPR [GDPR18] that adopts the core principles required for personal data processing; Ethics Guidelines for Trustworthy AI, by Independent High-Level Expert Group (HLEG) on AI set by the EC [HLEG19]; Ethically Aligned Design (EAD) [EAD19] which is an initiative created by the IEEE Standards Association and covers many topics of interest to EFPP development, including e.g. general (ethical) principles; how to embed values into autonomous intelligent systems; methods to guide ethical design; safety and beneficence of artificial general intelligence and artificial superintelligence; personal data and individual access control; reframing autonomous weapons systems; economics and humanitarian issues; law; affective computing; classical ethics in AI; policy; mixed-reality, and well-being.

One of the greatest business challenges is how to get competitors to cooperate with each other in the platform ecosystem. Here, efficiency gains, Return of Investments (ROIs) and other incentives need to be supported through platform ecosystems by selecting the adequate governance mechanisms. To support the further growth of the EFPP ecosystem, we focus on these attributes that promote the common good and limit negative impact and conflicts either via the platform or in the platform ecosystem. Some examples of such attributes are: Inclusion, participation, accountability, etc.

The authors in [GRIN07][GISS12] emphasize good governance as an essential driver of sustainable development. Implementing good governance practices requires knowledge about existing actors (stakeholders) and their business- and government-driven rules, as well as knowledge about possible benefits and losses, and other external effects that can cause both positive and negative effects (conflicts).

In EFPP, we adopt the concept of "positive platform governance" as discussed in [MATR17], that emphasizes the importance of shared decision making with the platforms' contributors, through the following principles (see Figure 1 below):

- Inclusion - which is about defining the roles and power for actors with diverse levels of participations.
- Participation - which is based on principles of fairness, simplicity, transparency, and trust integrated in all decision-making processes of the platform.
- Autonomy - which helps assuring that all contributors affected by those decisions will be able to participate.
- Recognition of the generated value - ensuring reward system for all contributors in the ecosystem.

Principle	Mechanism
Inclusion of the max number of stakeholders in decision-making, ensuring that contributors have decision-making power that's proportionate to their level of activity and engagement	<ol style="list-style-type: none"> 1) Binary power system (power defined by the owner) 2) Smooth power system (power defined by the level of contribution and commitment)
Fostering participation by embedding the principles of fairness, simplicity, transparency, and trust into decision-making process of the platform	<ol style="list-style-type: none"> 1) Online participation in the decision-making (comm. channels/tools) 2) Participatory mechanism (consensus, majority, holacracy) 3) Collab. decision-making
Embedding autonomy in the decision-making to keep agility while scaling, but assuring that all contributors affected by those decisions will be able to participate	
Recognition of the generated value: Ensure a fair reward systems for all value contributors on the platform	Adding value accounting system (algorithms, peer feedback, etc.) that measures and accounts for contributors, with well defined rules.
Welfare: Inclusion of protection and security for contributors at the core of the governance model	Policies; Insurance; etc.

Figure 1: Adopting Positive Governance Mechanisms in EFPF

2 EFPF Marketplace

This section provides a general view on digital marketplaces in the EFPF context and information about the EFPF Marketplace and components created within this task, which are the following:

- **Integrated Marketplace Framework:** This component provides access to items listed on external marketplaces at different platforms provided by EFPF partners.
- **Automated Agent-based Marketplace and Online Bidding:** This component provides an agent-based marketplace with online bidding functionality.
- **Accountancy Service:** This subcomponent provides features to track & trace and credit users of connected marketplaces.

Marketplaces are common components in digital manufacturing platforms. They provide collaboration and interaction between a platform and its service and tool provider and its customers via a catalogue framework. This allows products to be listed and therefore browsed based on categorisation and searched according to search terms and filter conditions. An outcome of this is the natural knowledge transfer and technology exchange between users. Marketplaces may list only products provided by its platform or provide products coming from third party platforms.

In this context marketplaces can be divided into two types:

- **Hosting marketplace:** Products listed are provided by the hosting platform of the marketplace, which can offer standard ecommerce functionalities such as listing new products, which are uploaded directly to the marketplace, and checkout functionalities, which will handle customer transactions within the marketplace.
- **Integrated marketplace:** Products listed are provided by marketplaces from external platforms. This type does not offer the above-mentioned ecommerce functionalities. In this scenario products are listed on the external marketplace, which also provides the transaction and checkout functionalities.

2.1 Integrated Marketplace Framework

The Integrated Marketplace Framework (IMF) within EFPF is implemented as two components: The EFPF Marketplace Backend and the EFPF Marketplace UI. External marketplaces provide their products through API interfaces, so the EFPF Marketplace Backend can retrieve the products and the EFPF Marketplace UI can list these products, providing typical filter and sorting mechanisms, with the ecommerce featured provided by the external marketplace.

To expand the product offering, users will be enabled to publish products. As connected external marketplaces may provide different feature sets, the IMF provides a selection of marketplaces for the best fit of new products. With this approach it will be possible to support different types of products and their needs. For example, a physical product has other marketplace requirements than a software product. Therefore, it may be possible that future external marketplace may be added and offer itself as an additional target for new products.

The above-mentioned processes allow a fast integration of external marketplaces and their products into the IMF and therefore is more suited for a federated environment, where multiple marketplaces co-exist. As each marketplace may have a different data model and also a different design of the marketplace, the EFPF Marketplace Backend as a part of the IMF can list products in a unified interface.

2.1.1 Scope and Relationship with Other EFPF Components

The Integrated Marketplace Framework provides access to multiple external marketplaces. Currently the following external marketplaces are connected and listed in the EFPF Marketplace UI:

- vf-OS¹
- Nimble²
- SMECluster³
- WASP
- ZDMP⁴

External marketplaces can be adapted to enable Single Sign-On (SSO), so users registered on the EFPF platform can access these marketplaces without having to register again. Nevertheless, an external marketplace may ask for additional information.

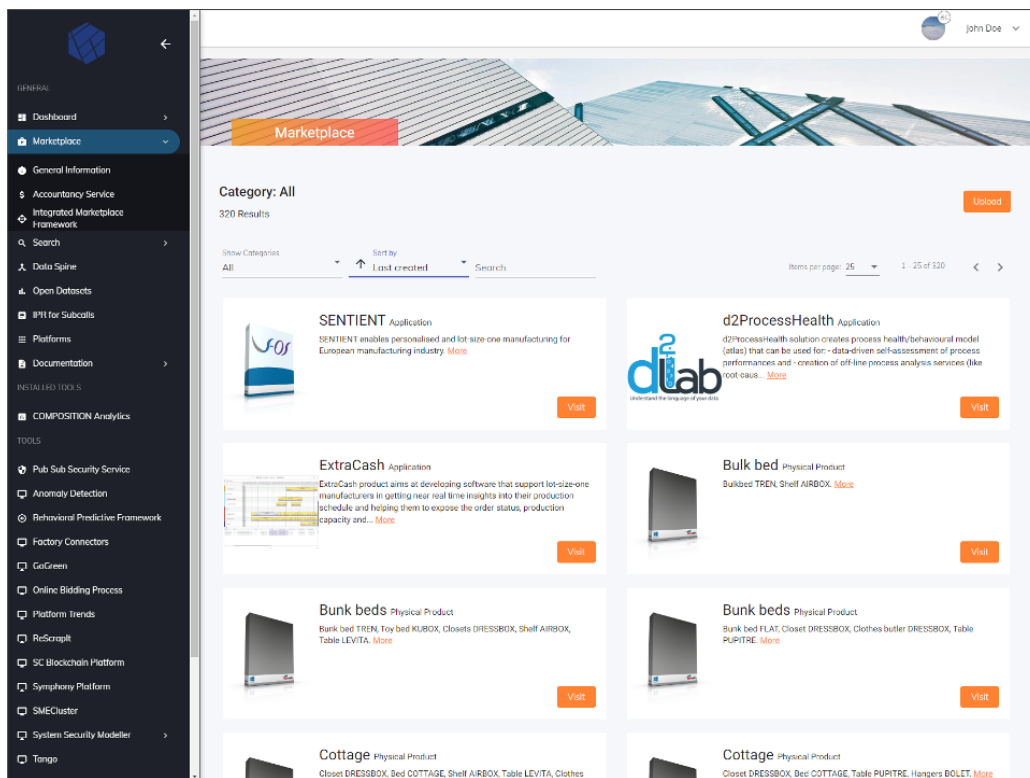


Figure 2: Marketplace Framework - User Interface Screenshot from EFPF Portal

The following list provides an overview of the relationships to connected EFPF components and a description of the functionality they enable:

- 1 <https://www.vf-os.eu/>
- 2 <https://www.nimble-project.org/>
- 3 <https://www.smecluster.com/>
- 4 <https://www.zdmp.eu/>

- **EFPF Portal:** This component acts as a container for the EFPF Marketplace UI as can be seen in Figure 2. The EFPF portal provides the EFPF Marketplace UI for non-registered users on its landing page and for registered users via the menu.
- **EFPF Service Registry:** This component provides the API endpoints and other information to the EFPF Marketplace Backend to be able to retrieve product data from external marketplaces.

2.1.1.1 External Marketplaces

This section provides an overview and description of each connected marketplace. Each marketplace exposes its product catalogue via REST interfaces. The internal marketplace framework can retrieve the products and list them including a link to the details page of a product inside each of the external marketplaces.

vf-OS Store

The vf-OS Store has been developed in the scope of the European research project vf-OS. It provides a one-stop-shop for the extension of the vf-OS platform, which are called vApps. Users can browse listed applications, filtering by categories, providing feedback and view dependencies. It provides an easy checkout through various payment methods.

Key features:

- Browser applications as a customer
- Easy checkout
- Manage uploaded applications
- Product Upload

The vf-OS Store is also being used for uploading products for any downloadable software applications by the Integrated Marketplace Framework inside the EFPF Portal.

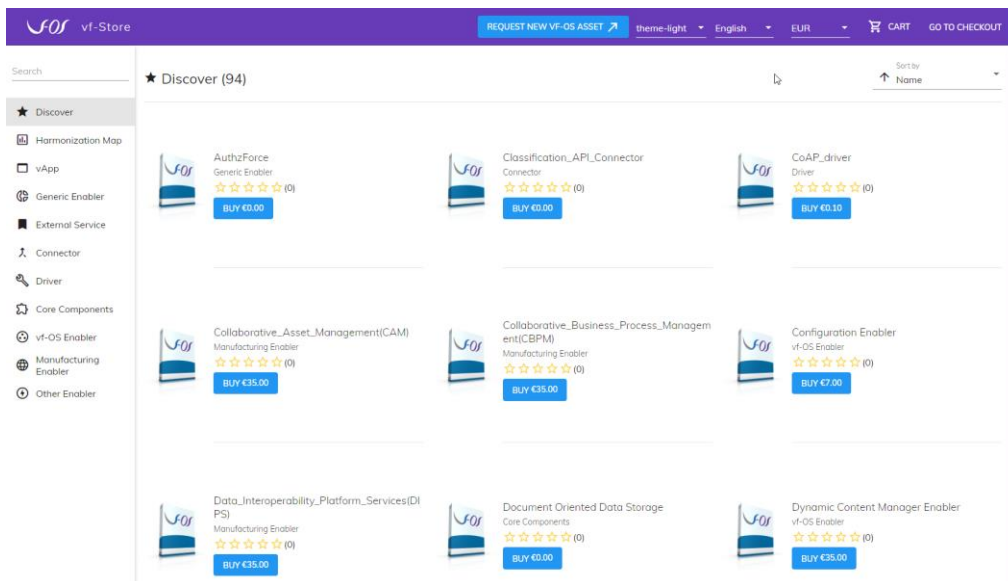


Figure 3: Marketplace Framework - vf-OS Marketplace Screenshot

SMECluster Marketplace

The SMECluster Marketplace is an integrated part of the SMECluster website providing an overview of available tools and services. These tools and services may consist both of software and hardware and additional set-up work. Therefore, many listings cannot be bought right away but the customer provides its contact data for more information and a personal contact.

Key features:

- Browse products by categories
- Provide rich descriptions
- Support products including hardware setup
- Checkout and payment feature
- Product Upload

The SMECluster Marketplace is being used to provide tools and services which require a manual configuration or adaptations to be used at the customers premises. Products can be listed on the SMECluster Marketplace through a manual process by contacting the SMECluster Marketplace owner.

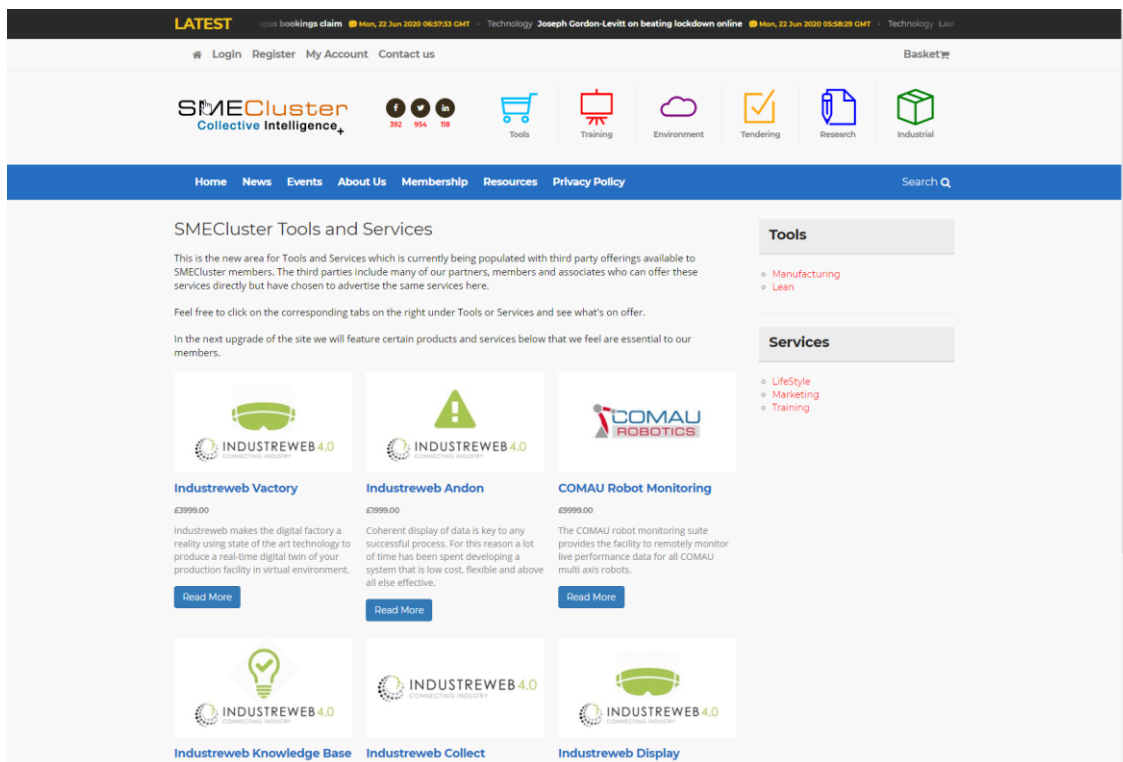


Figure 4: Marketplace Framework - SMECluster Marketplace Screenshot

Nimble Marketplace

The Nimble Marketplace has been developed in the scope of the European research project Nimble and provided a federated, multi-sided and cloud services-based business ecosystem. The Nimble Marketplace is the front-end of this ecosystem and enables users to buy, offer and negotiate.

Key features:

- Company and product search

- Product catalogues
- Advanced checkout incl. negotiations
- Product Upload

The Nimble Marketplace is used to provide physical products. The Nimble Marketplace will be used to provide physical products. Product upload can be performed by the registered EFPF users.

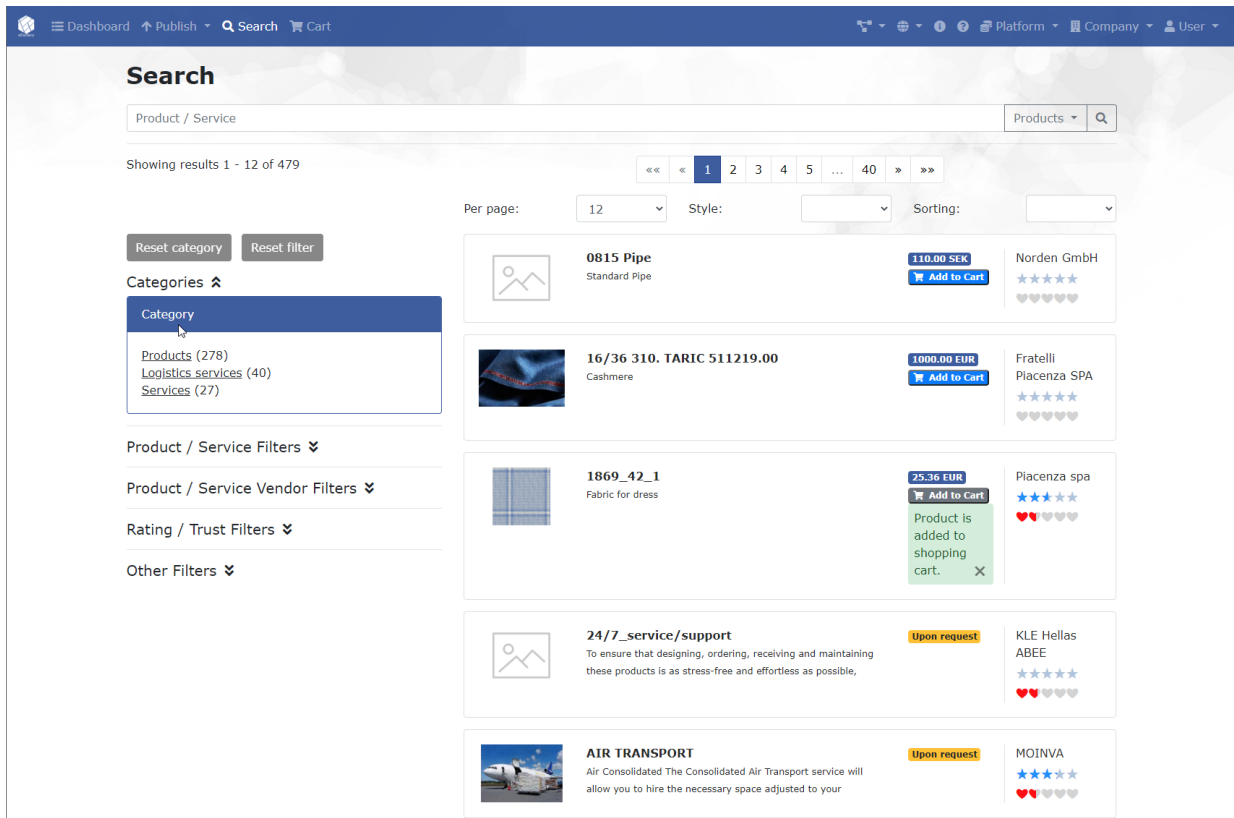


Figure 5: Marketplace Framework - Nimble Marketplace Screenshot

WASP Marketplace

The WASP Marketplace is part of the Workflow and Service Automation Platform. This marketplace provides software services ready to use in websites and other components. Services are sorted by categories and detailed information are provided.

Key features:

- Web services
- Bubble navigation

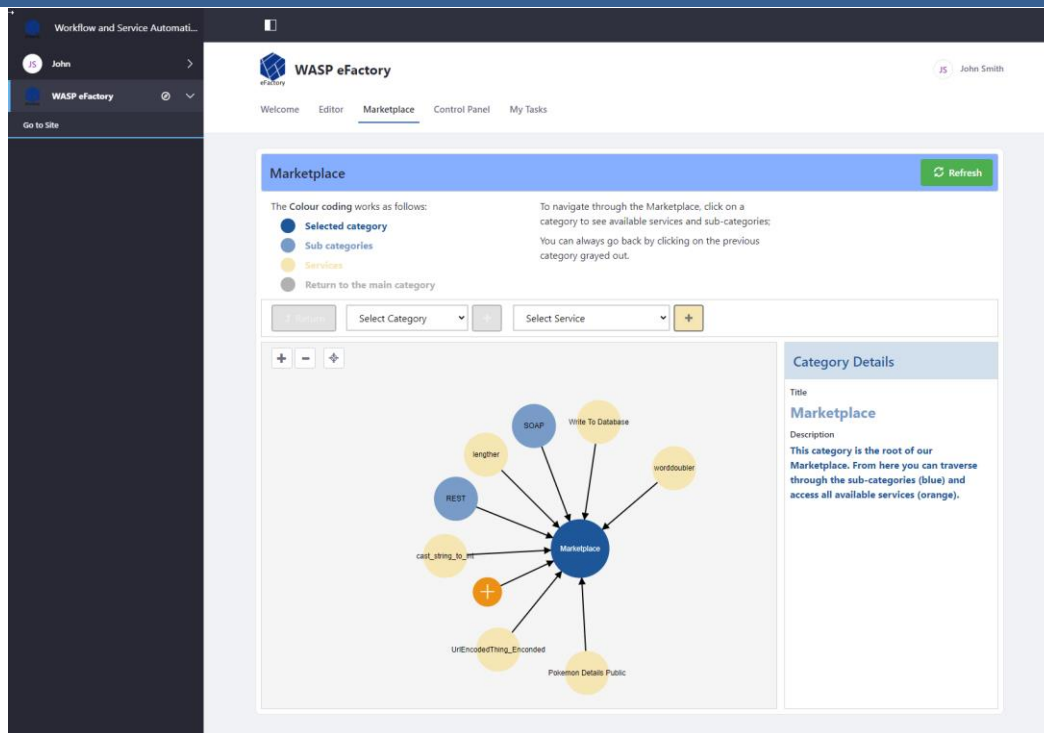


Figure 6: Marketplace Framework - WASP Marketplace Screenshot

ZDMP Marketplace

The ZDMP Marketplace is being developed in the scope of the European H2020 research project ZDMP⁵ and provides “...a digital platform for connected smart factories for achieving excellence in manufacturing through zero-defect processes and products thanks to the use of zero-defect core services for developing APPs”. The ZDMP Marketplace is the front-end of this ecosystem and enables users to buy, offer and upload applications.

Key features:

- Browse products by categories
- Checkout and payment feature

⁵ <https://www.zdmp.eu/>

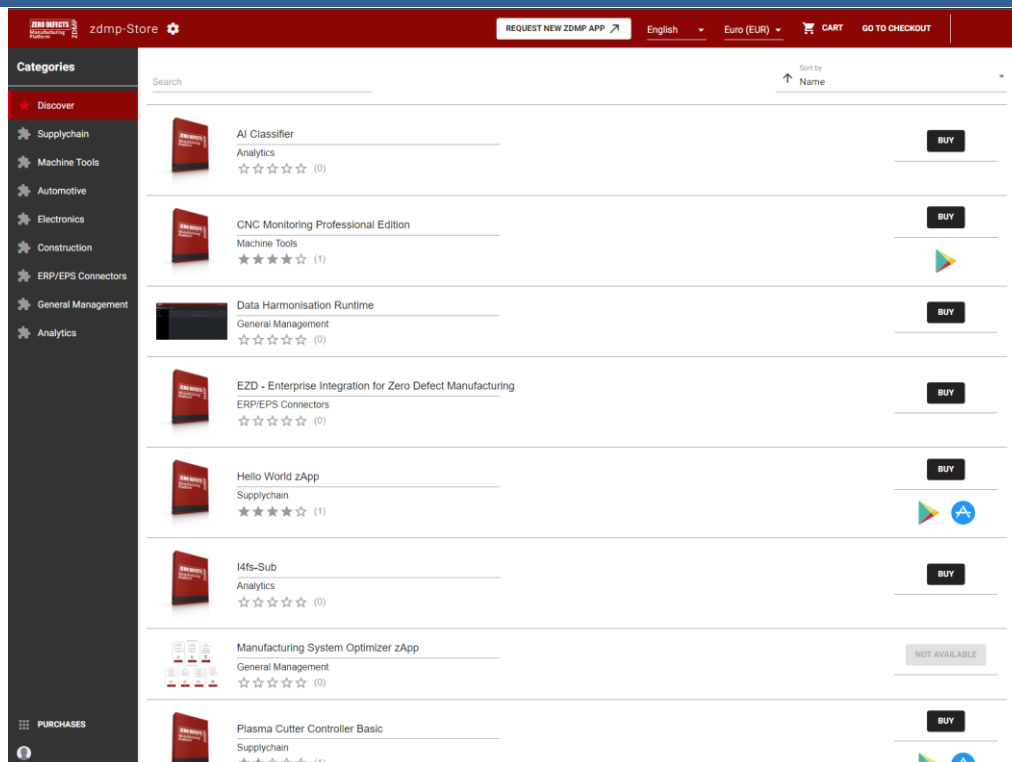


Figure 7 Marketplace Framework – ZDMP Marketplace Screenshot

2.1.2 Requirements and Realisation

The Integrated Marketplace Framework has the following requirements:

- List products provided by external marketplaces
- Track, trace and credit users
- Upload new items

The first requirement is already implemented by listing products from external marketplaces mentioned in Section 2.1.1. The second requirement (which is a set of requirements) are being provided by the Accountancy Service. The third requirement has been fulfilled by enabling the product upload through external marketplaces (vf-OS, SMECluster and Nimble).

The EFPF Marketplace UI is implemented as a web component based on the Angular web application framework. As a web component, it has to be embedded in a website in order to use it (realized via the EFPF Portal). To retrieve products from external marketplaces this component connects to the EFPF Marketplace Backend, which retrieves the API endpoints of external marketplaces via the EFPF Service Registry.

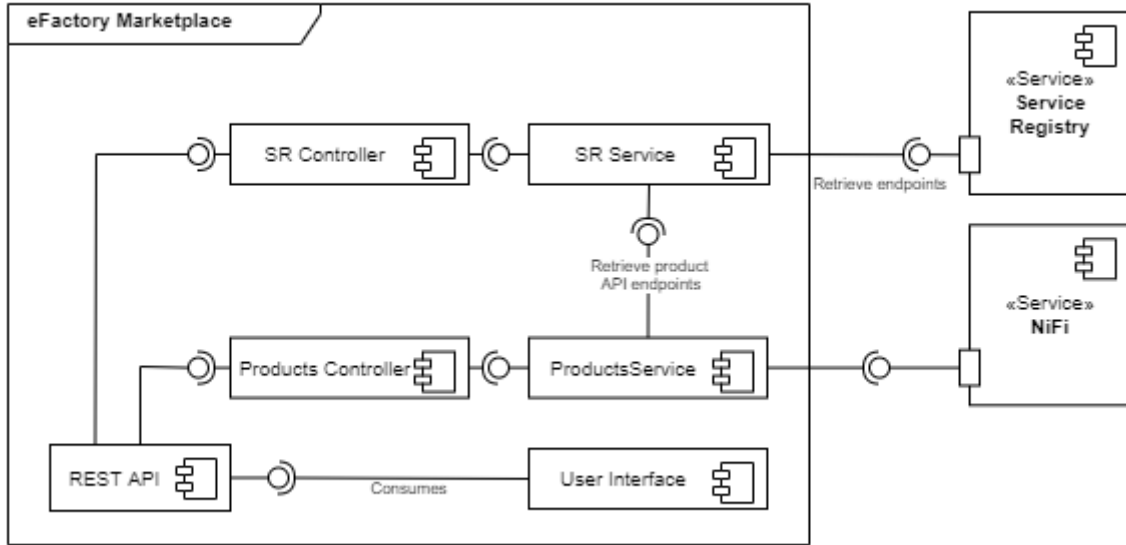


Figure 8 EFPF Marketplace - Architecture Snapshot

2.1.3 Deployment

The following sections will provide basic information about the deployment of both components of the Integrated Marketplace Framework.

2.1.3.1 EFPF Marketplace UI

The EFPF Marketplace UI in its current form will be integrated as a JavaScript file in a website and requires little integration effort. After adding the file to the website, which may differ based on the used framework, the developer can use the “EFPF-marketplace” HTML element in their source code as shown in **Error! Reference source not found..** At this point the element requires a valid Java Web Token (JWT) to retrieve products from the EFPF platform.

```

1 |
2 |
3 | <div class="container">
4 |   <efpf-marketplace [IdToken]="idToken" [Configuration]="configuration"></efpf-marketplace>
5 | </div>
6 |
7 |

```

Figure 9 Marketplace Framework – Source Code Deployment

As to be seen in **Error! Reference source not found.** the EFPF Marketplace UI requires two attributes to be provided:

- **IdToken:** This optional attribute contains the access token of the user, if registered. It will be used for user tracking.
- **Configuration:** This mandatory attribute expects a class containing the following information:

- URI to the EFPF Marketplace Backend
- URI to the EFPF Portal backend
- UUID for this EFPF Marketplace instance

2.1.3.2 EFPF Marketplace Backend

For the deployment of the EFPF Marketplace Backend a Docker image is being created via GitLab CI/CD and is being used by the CI script provided by the <https://gitlab.fit.fraunhofer.de/efpf-pilots/efpf-integration-and-deployment> repository. When running the deployment CI script, configuration data will be injected by replacing the placeholders mentioned in the developer section.

The injected configuration data consists of the following data:

- URI and credentials (Client ID and secret) of the EFPF Security Portal
- URI of the EFPF Service Registry
- Credentials to the ZDMP Marketplace (Client ID, username and password)

2.1.4 Execution and Usage

The following sections will provide basic information about the execution and usage of both components of the Integrated Marketplace Framework.

2.1.4.1 EFPF Marketplace UI

By default, the EFPF Marketplace UI will list all available items sorted by the name attribute. Additionally, the user can sort the products based on the item category, which is provided by the source marketplace. The user has also the option to negate the sorting.

The user can also use the provided filters, which allows to filter the list based on the source marketplaces and the categories. Both filters can be reset to their default values.

Each item provides basic information, which are the name, categories, item image and an external link, which leads the user to the detail page of the source marketplace.

2.1.4.2 EFPF Marketplace Backend

The EFPF Marketplace Backend is being used only by the EFPF Marketplace UI component, which can be deployed independent of the EFPF Marketplace Backend. The EFPF Marketplace Backend provides one API endpoint for retrieving products. Detailed information are provided via Swagger⁶.

2.1.5 Limitations and Further Development

The integration of a new external marketplace requires manual adaptations of both components (EFPF Marketplace UI and Backend). Future developments could adapt the EFPF Marketplace UI in a way, that it could handle products from a new external marketplace without adaptations.

⁶ <https://efpf.smecluster.com/marketplace-backend/api/>

2.2 Automated Agent-based Marketplace and Online Bidding

2.2.1 Scope and Relationship with Other EFPF Components

Besides the overall marketplace framework of the project, an agent-based marketplace will be also available to users. This type of marketplace aims to provide mechanisms that enable the automated negotiations within the participants. In this marketplace, each company sets up an agent which represents the company in an Online Bidding Process (OBP) in which the agents negotiate for specific goods and services. This provides automation and time costs reduction of traditional manual procedures in negotiation processes carried out by companies mutually connected by a production and service supply chain. Moreover, OBP facilitates and promotes a more efficient circular economy by enabling marketplaces where provider companies may offer production waste materials as by-product for possible buyer companies, following an industrial symbiosis paradigm.

The Online Bidding Process component is composed of three main components:

- Agent ecosystem
- Matchmaker
- User Interface

The platform is derived from the COMPOSITION project and has been refactored in EFPF and integrated within the its ecosystem to provide automated bidding process functionalities to EFPF users. This is done to propose a faster way to manage supply/demand matching to speed up the process of reaching agreement between two parties.

The Agent ecosystem is composed by a set of agents built to communicate with each other and with other platform's components to perform an automated negotiation. To achieve this the Agents implement a standard communication protocol based on FIPA ACL⁷ and AMQP⁸ to implement an asynchronous publish and subscribe communication approach. The OBP Agents are categorized by the role which they take within the OBP ecosystem:

- **Requester:** Represents a company who requests for a service or/and a related good.
- **Supplier:** Represents a company who provides a service.

The Matchmaker component provides a full semantic framework for the agents with CRUD operations for agents/companies and two type of matchmaking functionalities. It matches requesters with suppliers for a service/good and in a second level matches the request with the best available offer (coming from supplier agents' bids) based on different evaluation criteria such as price, payment and delivery methods, reliability etc.

The UI of the bidding process is web-based and enables a user to carry out the following functions:

- Register/set up and agent
- Initialize a bidding process in order to request for a service/good (requester)
- Add information and priorities for the request (requester)
- Online monitoring of the process for both suppliers and requester
- Opportunity for supplier to bid for a request

⁷ <http://www.fipa.org/repository/aclspecs.html>

⁸ <https://www.amqp.org/>

- Suggestion for best offer and details of all submitted offers in the case the user want to ignore system suggestion and evaluate by himself (requester)
- Visual notification for win/lose in bidding process (supplier)
- Bids history available to user (requester)

The system is made available to the user through the EFPF portal where an authenticated user is able to log in and access the OBP, like represented in **Error! Reference source not found.**, as well as configure and set up an agent and to control the bidding processes by the means of UI, according to the schema in **Error! Reference source not found.**

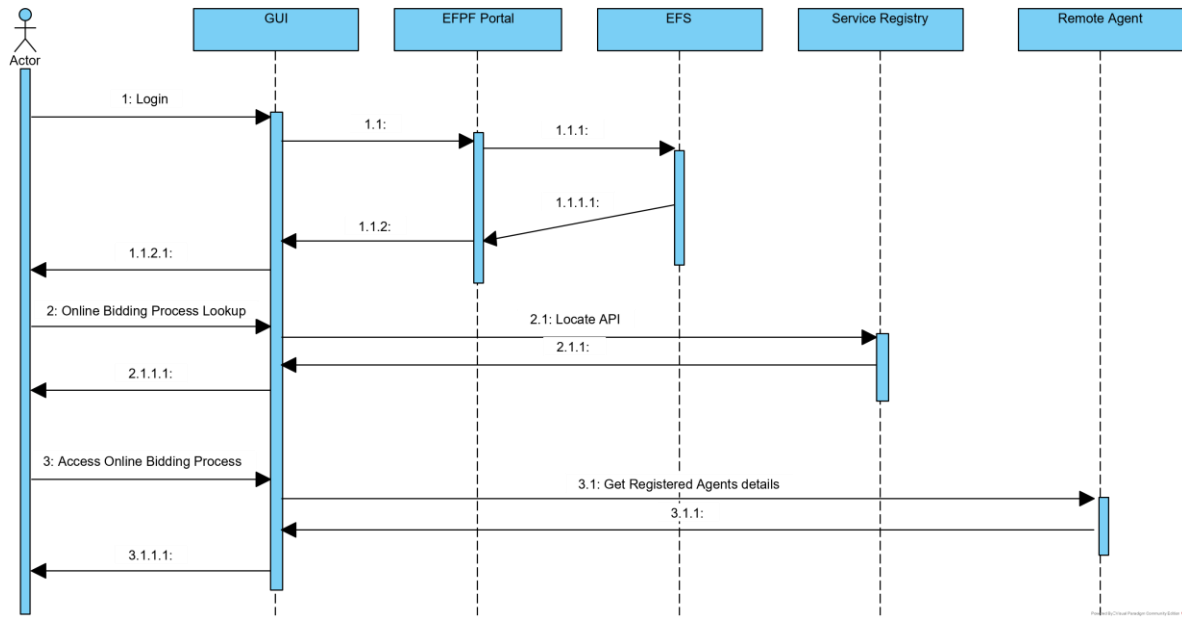


Figure 10 Online Bidding Process Platform Login

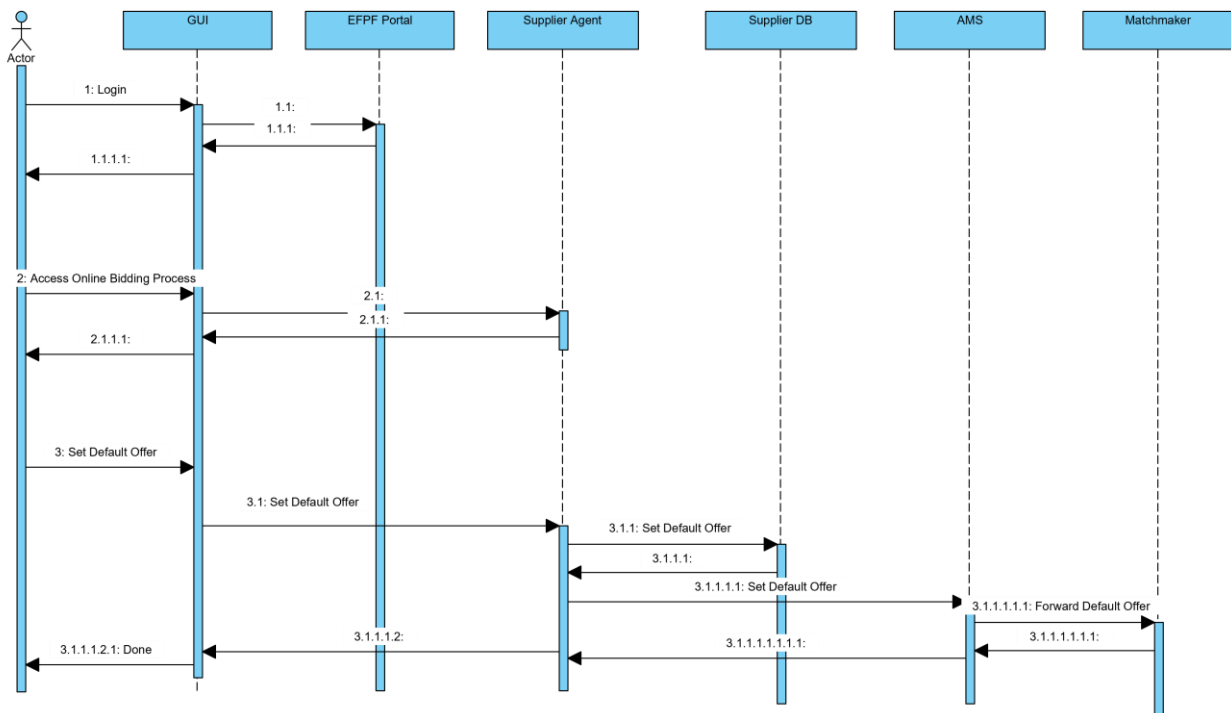


Figure 11 Online Bidding Process – Set Default Offer

2.2.2 Requirements and Realisation

Design Requirements

The Online Bidding Process is a stand-alone component that is made by a network of automated and semi-automated software agents representing companies enabled to perform automated negotiation following predefined protocols and logics. The platform coming from the COMPOSITION project expects that each agent should be manually built and deployed together with an ad-hoc designed UI for each company taking into account the needs and the use cases. The EFPF platform has a wider scope with respect to the COMPOSITION project from which this tool derives from.

To overcome these issues and to integrate the solution with EFPF platform, new components have been developed (see **Error! Reference source not found.**):

- the Bidding Proxy (a.k.a. Agent Deployer) and its API;
- the Agent Management System (AMS);
- a new User Interface (UI);
- extensions of matchmaking functionalities.

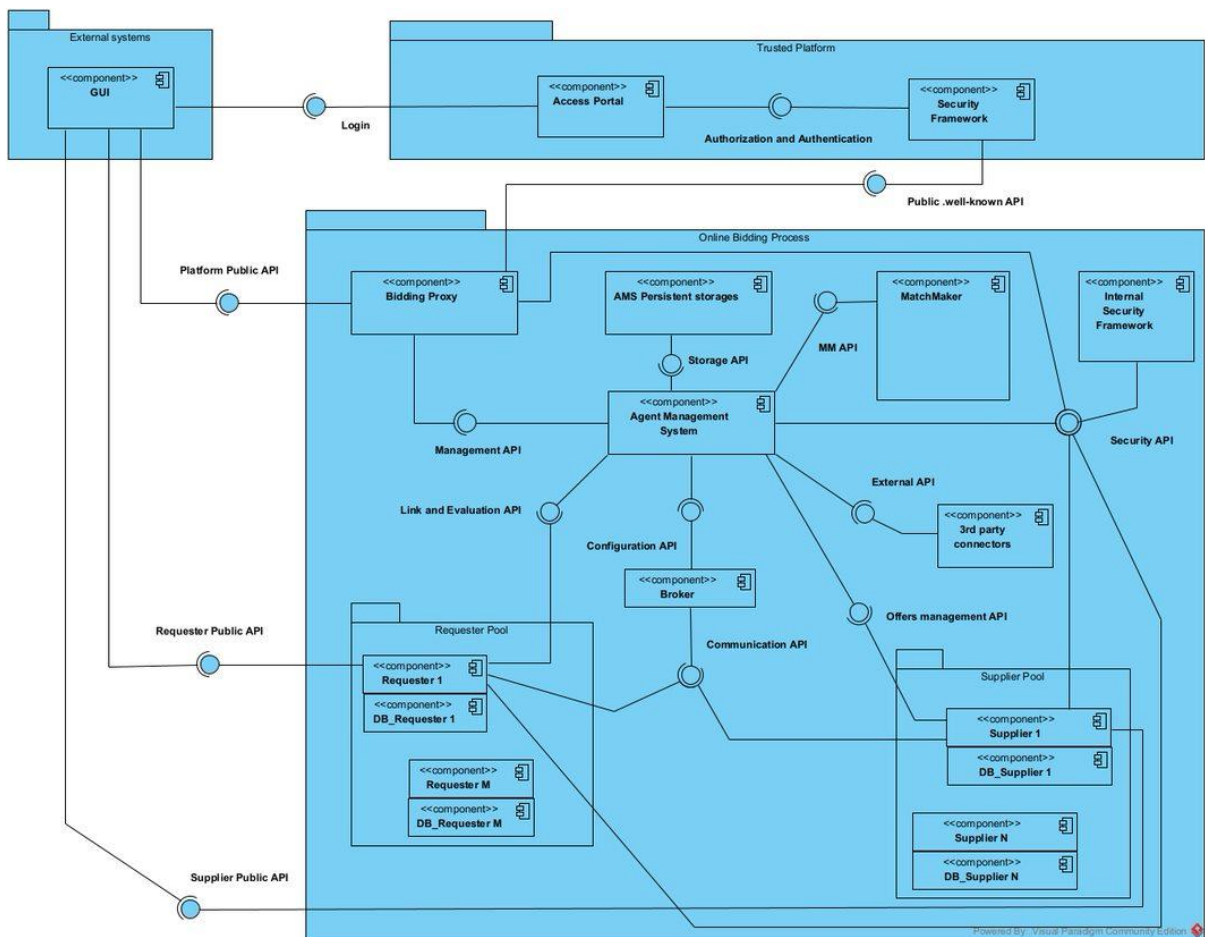


Figure 12 Online Bidding Process architecture

The Bidding Proxy is the main access point of the whole OBP. It is responsible for the creation, deployment and management of the Agent, as shown in **Error! Reference source not found.** Whenever a new request for an Agent creation is issued by the user, the Bidding Proxy instantiate and configure a new Agent by generating a new docker container which

embeds the software implementing generic Requester or Supplier Agents. The Bidding Proxy contacts, through a secure SSH connection, the machine where the Agent needs to be deployed (it might be the same machine where the Bidding Proxy is running or a remote one), providing the relevant configuration parameters generated by the AMS.

The AMS handles the registration of the Agents ID and relevant properties on a Postgre database, and gives also the possibility to register the Agents and their transactions (including smart contracts) on a blockchain. Even if an Agent is deleted, its historical transactions are kept on the DB/blockchain and the relevant information can be gathered by the users by querying the provided REST API of the Bidding Proxy.

The OBP provides an interface of more than 120 REST API to enable the user to create, destroy and configure new Agents, define the ranking of the offers, get the list of Agents, the list of Agents per user, gather the historical transactions, etc.

ReDoc pages describing API are available after the components deploy on the path: <https://ip:port/{nginx-link}/api/v1/docs>

Postman API are available at:

<https://documenter.getpostman.com/view/8641456/UyrDEG6L>

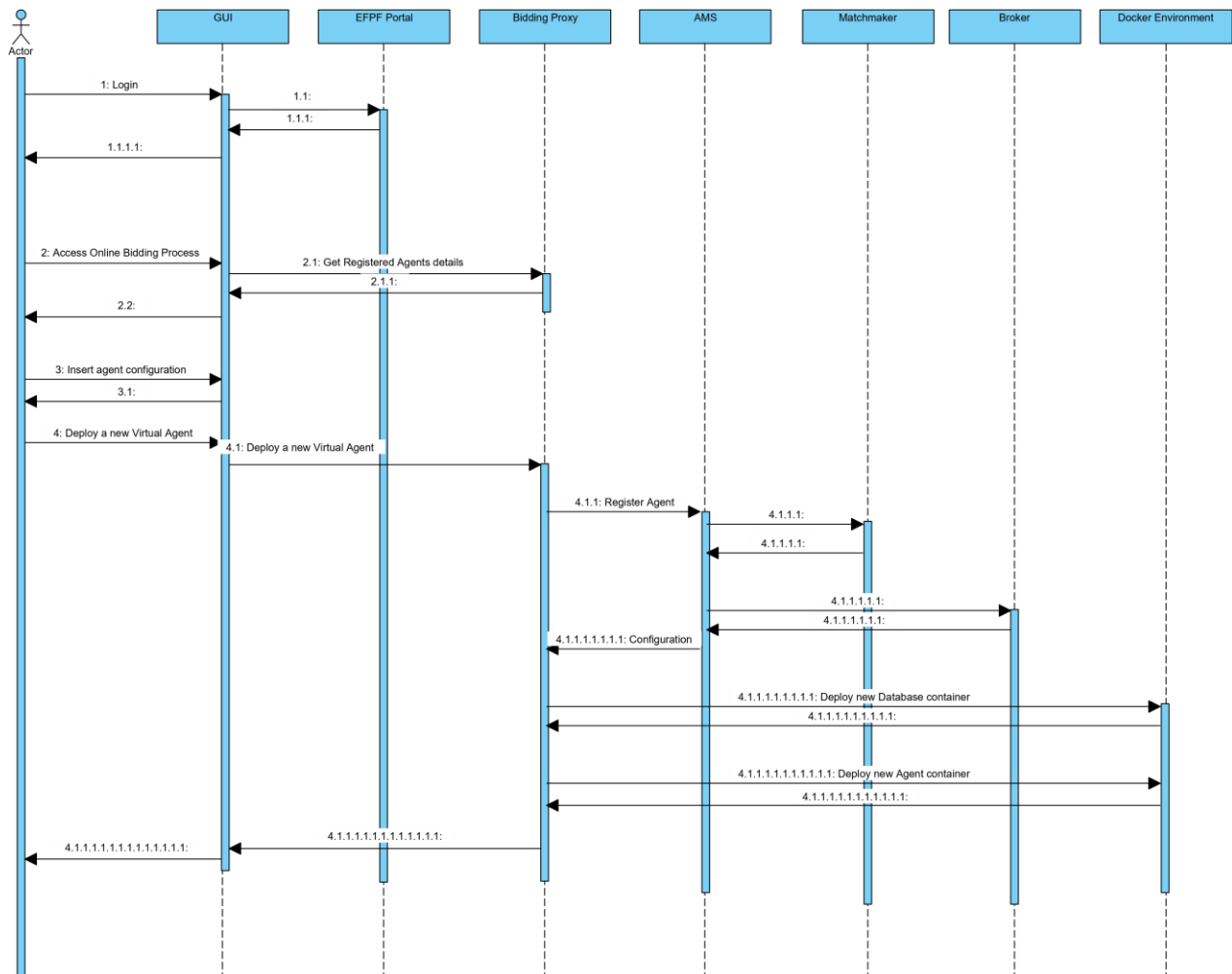


Figure 13 Agent deployment workflow

The API is used by the new OBP UI, which has been developed within EFPF as a web form that enables users to create their own agents, enabling a more generic but flexible approach to Agent creation. The new UI allows the dynamic rendering of the views from the semantic

framework rather than being designed explicitly for two or three partners only. The general design approach for EFPF was to move from a custom solution to a more generic one in order to satisfy user requirements.

For the Matchmaker side that provides CRUD operations to agents, new web services have been developed to meet the Bidding Proxy and UI needs.

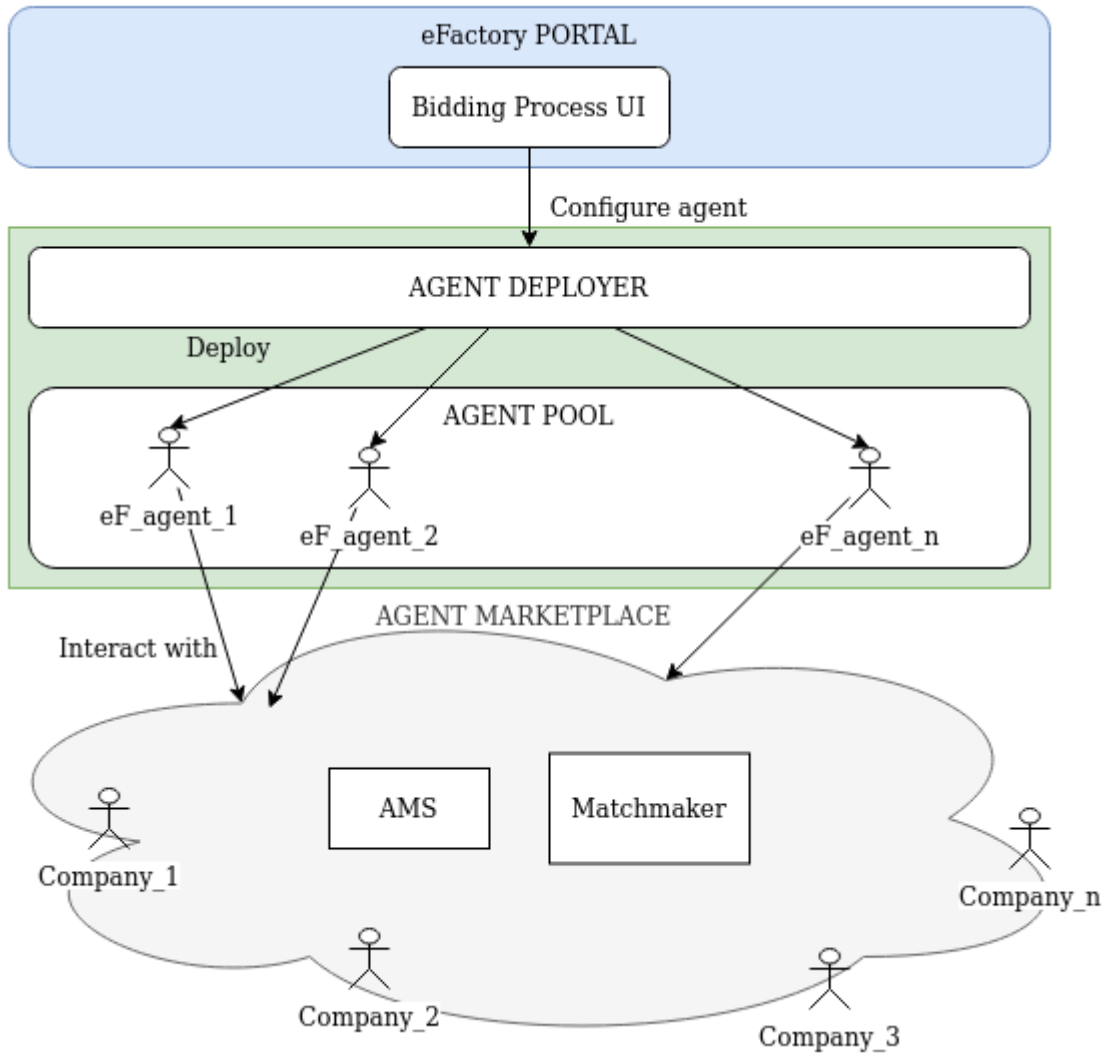


Figure 14: Agent Marketplace - Deploying Process

For the Matchmaker side that provides CRUD operations to agents, new web services have been developed to meet the Agent Deployer and UI needs.

In addition to the components that have been designed for EFPF needs, developments to existing modules have been applied or scheduled for future developments. All the updates to both front end UI and back end services are driven by the need for generic services throughout the platform.

User Requirements

Besides the requirements that have been extracted by the project design approach, the agent-based bidding process is a core component for Circular Economy pilot of the project.

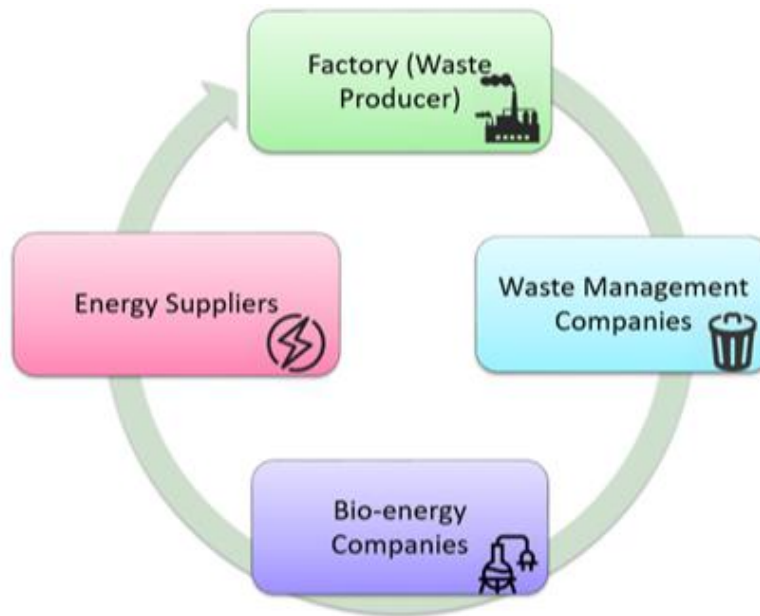


Figure 15: Agent Marketplace - Process

In this pilot there is the major requirement for the users to be able to negotiate automatically for some goods/services in this closed loop. The pilot partner that manufactures lifts wants to negotiate its scraps automatically through by using its agent and the bidding process. The waste management company that wins on bidding process wants to sell the processed waste to a bio-energy company by using the same system as well. Finally, the produced energy by the scraps returns to lift manufacturer in this circular scenario.

Therefore, a Purchasing manager or purchasing specialist wants to negotiate prices and contracts so to obtain high-quality services/products at reasonable prices in an automatic way. This need led to design this automated online bidding mechanism that provides suggestions of best available supplier for a service and enables the automation of negotiation procedures in a close loop. More detailed pilot requirements are available on the corresponding deliverable.

2.2.3 Deployment

The Online Bidding Process infrastructure can be deployed from scratch following the instructions below.

On a clean & updated VM:

Install docker & base libraries & tools

```
sudo -S apt-get update
```

```
sudo -S apt-get install nginx, openssh-server
```

```
sudo -S apt-get install apt-transport-https ca-certificates curl software-properties-common
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | echo ${sudo_password} | sudo -S apt-key add -
```

```
sudo -S add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
sudo -S apt-get update
sudo -S apt-get install docker-ce
Enable docker usage without root credentials
sudo -S usermod -aG docker ${user_name}
su - ${USER}
```

Define and deploy a docker network and its static IP (and ports) for the main components:

Create the bridged docker network:

```
docker network create \
--driver=bridge \
--subnet=172.23.0.0/24 \
--ip-range=172.23.0.0/24 \
--gateway=172.23.0.1 \
mas_docker_network \
```

Network plan example:

```
network Gateway=172.23.0.1
rabbitMQ=172.23.0.2:5672 (default)
Keycloak=efpf.polito.it:42003 (nginx)
AMS PostgreSQL storage=172.23.0.3:5432 (default)
AMS=172.23.0.4:5587 (synched among .env files of bidding-proxy & AMS)
bidding-proxy=172.23.0.5:45000 (synched only in .env files of bidding-proxy)
```

The nginx rules will forward external requests to the bidding proxy at the configured public_ip+port (e.g. efpf.polito.it:45002)

Then, define and activate the nginx-rules and configuration folders enabled for the MAS (+ user-rules)

Build ssl certificates for the host machine

```
PASSPHRASE=$(head -c 500 /dev/urandom | tr -dc a-z0-9A-Z | head -c 128; echo)
echo ${PASSPHRASE} > /home/test/EFPP/global.pass
openssl req -x509 -newkey rsa:4096 -keyout /home/test/EFPP/key.pem -out
/home/test/EFPP/cert.pem -passin env:${PASSPHRASE} -sha256 -days 365
```

Write the nginx rule to:

enable communication with the Agent Deployer
create and link a folder for the future agents configuration (include sys-path)
add the sudoers role to let the base user to reload nginx rules
reload rules

Example of nginx rule (/etc/nginx/sites-available/efpf-docker.conf)

-> soft linked in (/etc/nginx/sites-enabled/efpf-docker.conf):

```
server {  
listen 45002 ssl;  
listen [::]:45002 ssl;  
server_name 127.0.0.1;  
ssl_password_file /home/test/EFPF/global.pass; \  
ssl_certificate /home/test/EFPF/cert.pem;  
ssl_certificate_key /home/test/EFPF/key.pem;  
location /efpf_docker/ {  
include proxy_params;  
proxy_pass http://172.23.0.5:45000/;  
proxy_set_header Host $host;  
proxy_set_header X-Real-IP $remote_addr;  
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
proxy_set_header X-Forwarded-Host $host;  
proxy_set_header X-Forwarded-Proto $scheme;  
}  
include /home/test/EFPF/configs/*.conf;  
}
```

Example of sudoers permission file, if user_name=test -> (/etc/sudoers.d/test):

```
test ALL=(ALL) NOPASSWD: /usr/sbin/nginx -s reload
```

```
sudo -S nginx -s reload
```

Deploy a postgresSQL database for AMS

Inject settings in docker run & set values in AMS .env file

```
docker run -d --restart always --net=mas_docker_network --ip=172.23.0.3 --name
db_agent_ams -e POSTGRES_USER=agent_ams -e
POSTGRES_PASSWORD=ams_password -e PGDATA=/var/lib/postgresql/data/pgdata -v
db_agent_ams_data:/var/lib/postgresql/data postgres
```

Deploy rabbitMQ

Inject settings in docker run & set values in AMS .env file

[port mapping is not required since communications are internal -> only for debug]

```
docker run -d --restart always --net=mas_docker_network --ip=172.23.0.2 --name
rabbit_broker -p 5672:5672 -p 15672:15672 rabbitmq:3-management
```

AMQP Communication models: <https://www.rabbitmq.com/tutorials/amqp-concepts.html>

RabbitMQ settings:

Base user & password are: guest guest

Default Virtual host is /

Default Exchanges are:

fanoutexchange

directexchange

If you want to increase the security of the solution

(OPTIONAL, otherwise rely on guest user but avoid port mapping for platform-external requests):

Create a dedicated user to interact with rabbitMQ

```
rabbitmqctl add_user username password
```

(Only for administrators):

```
rabbitmqctl set_user_tags username administrator
```

Set permissions

```
rabbitmqctl set_permissions -p / username "." "." "*" "
```

Delete Default user?

```
rabbitmqctl delete_user guest
```

If you want to further increase the security of the solution add the user-agent-dedicated credentials within the registration phase performed by the AMS during the deploy.

IF the exchanges defined in .envs are not the default ones -> create them via UI interface (172.23.0.2:15672)

If you need to delete lot of queues / exchanges you need to define (and then remove) an expiration policy:

Get in the UI and login -> Admin (Up-Menu) -> Policies (Right-Menu):

Name: *Expire_all_policy*

Pattern: *. **

Definition: *expires = 1 [Number in drop-down Menu]*

Wait for a couple of seconds and remove the policy -> all the data has been erased

Deploy Keycloak or exploit a remote running instance

Readme at: <https://www.keycloak.org/getting-started/getting-started-docker>

Define the nginx rules to have access on it or integrate certs for direct access

Via UI (e.g.: <https://130.192.85.226:42003/auth/admin/master/console>):

Define at least 1 client (confidential) & 2 users in Keycloak (user-agent-base & proxy-user) + debug user agents

If access is granted by the same instance of Keycloak used for the platform-internal communications:

then create a public client and the users of interests + their password credentials

(then collect & set these values in AMS and bidding-proxy .env file)

Download, Build & Deploy AMS

git pull <https://git.pertforge.ismb.it/efpf/agent-ams.git>

Add fully configured .env file & run (it will create the db & the tables)

docker build . -t agent_ams

Port mapping is required only if you want to interact with it for debug

*if you want to deploy bidding_proxy & AMS in different machines you will need dedicated nginx rules for it *

docker run -d --restart always --net=mas_docker_network --ip=172.23.0.4 --name agent_ams agent_ams

Get inside the AMS DB docker:

docker exec -it db_agent_ams bash

Get inside the DB:

psql postgres -U agent_ams

and connect:

*\c *ams_storage**

Insert in AMS postgresSQL database static network-settings in net_info

insert into net_info values('gateway','Docker','172.23.0.1','');

insert into net_info values('rabbitMQ','MAS','172.23.0.2','');

insert into net_info values('db_agent_ams','MAS','172.23.0.3','');

insert into net_info values('ams','MAS','172.23.0.4','');

```
insert into net_info values('bidding_proxy','MAS','172.23.0.5',");
```

These are useful for debug & for the quick and dirt ip selection function performed during the registration phase

required to assign static free IP and consequently to dynamically build the nginx rules to access directly the agent!

Download & Build the agent-requester (docker images)

(Environment file will be produced by the bidding-proxy at run-time and automatically injected)

```
git pull https://git.pertforge.ismb.it/efpf/agent-requester.git
```

```
docker build . -t agent_req
```

Download & Build the agent-supplier (docker images)

*(Environment file will be produced by the bidding-proxy at run-time and automatically injected) *

```
git pull https://git.pertforge.ismb.it/efpf/agent-supplier.git
```

```
docker build . -t agent_sup
```

Download, Build & Deploy bidding-Proxy (with fully configured .env file)

```
git pull https://git.pertforge.ismb.it/efpf/bidding-proxy.git
```

```
docker build . -t bidding_proxy
```

```
docker run -d --restart always --net=mas_docker_network --ip=172.23.0.5 --name bidding_proxy bidding_proxy
```

if ufw active, enable the bidding-proxy port, & the keycloak port (if hosted and configured here)

```
sudo ufw status
```

```
sudo ufw allow 45002
```

Get a Keycloak Token via Postman and start interacting with the bidding-proxy and the components behind.

The build process deploys aside each agent a postgres_db. To manually access it for debug:

```
docker exec -it db_{agent_id} bash
```

Get inside the DB:

```
psql postgres -U agent_requester
```

or

```
psql postgres -U agent_supplier
```

and connect (exploit autofill via tab):

```
\c "{agent_id}"
```

Finally, if you prefer a GUI to monitor the docker containers, install portainer:

```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data \
portainer/portainer-ce:2.9.3
```

2.2.4 Execution and Usage

The features provided by this tool are available through the User Interface integrated into the EFPF portal. The basic UI that has been implemented in Angular 9 by keeping the design direction adopted by COMPOSITION and CNET. However, all the static attributes replaced with dynamic ones that enable the interfaces usage by different partners. The UIs are available to the EFPF user are:

- A registration form

The screenshot shows the 'Agent Registration Form' within the 'eFactory' portal. The form is titled 'Agent Registration Form' and contains the following fields:

- Agent Owner *
- Address/Street, Number, Postal Code
- City/Town, State/Province, Country
- Company Legal Name *
- Company Short Description
- Agent Role *, Business Type *
- Service Name *, Service Category *, Good *

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

Figure 16: Agent Marketplace - Agent Registration Form

By using this form, a company register its information to the semantic framework that consists the knowledge base of the agent ecosystem. After that, the agent for this company is automatically created. This agent can be used to represent the company in online bidding processes for specific services and goods.

- Bidding Process Management Dashboard

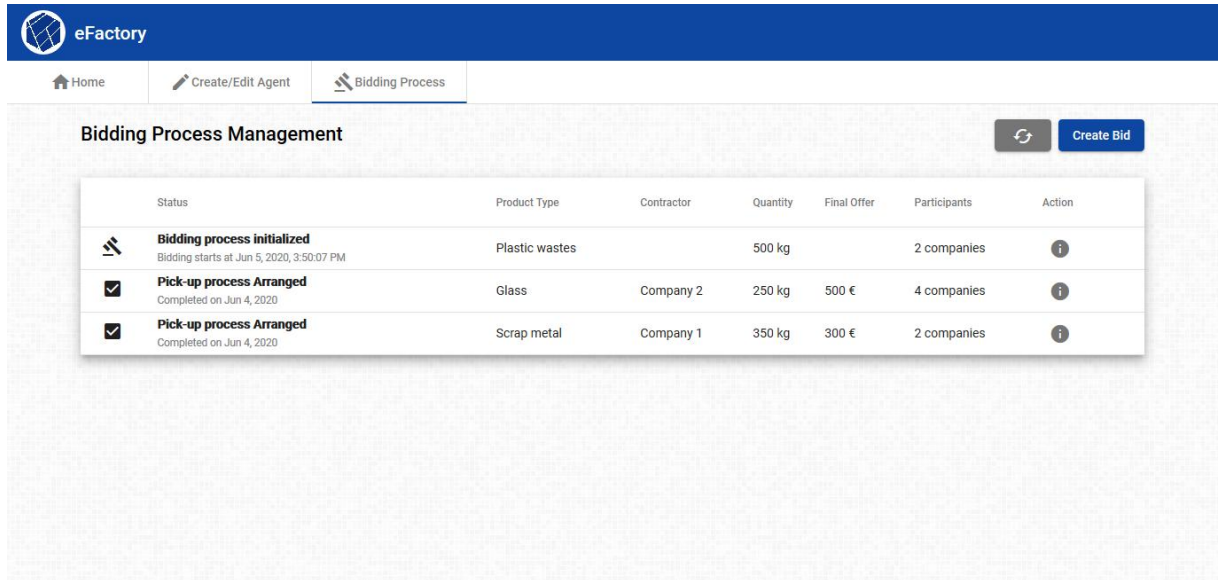


Figure 17: Agent Marketplace - Bidding Process Management Dashboard

In this interface, the user can explore details about previous bidding process that are closed (bidding history) or watch the bidding processes that are active right now. In this interface, the user can select the button 'Create Bid' in order to initialize a new online bidding process. Furthermore, the status bar for each bidding process is automatically updated during the process. The states are: Initialize -> Supplier selection -> Pick-up arrangement

- Initialize/create new bidding process

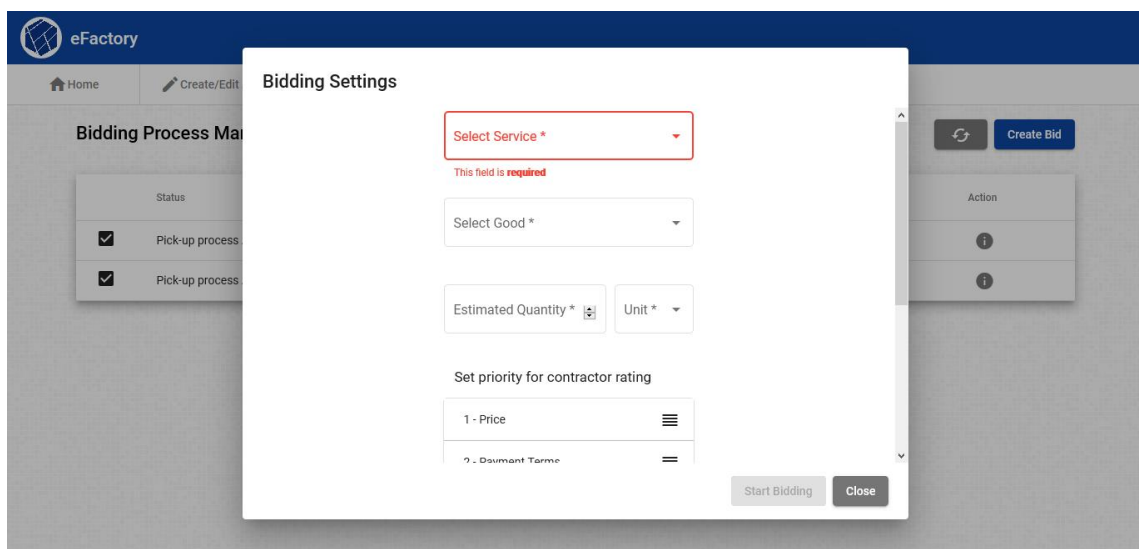


Figure 18: Agent Marketplace -Bidding Settings Popup I

In this menu the user can select a requested service, the corresponding good, to add the quantity and the unit of measurement, to set the priorities for his request and finally to start the bidding process. Next figures depict the available options for services, goods and supported priorities by the matchmaking/evaluation engine:

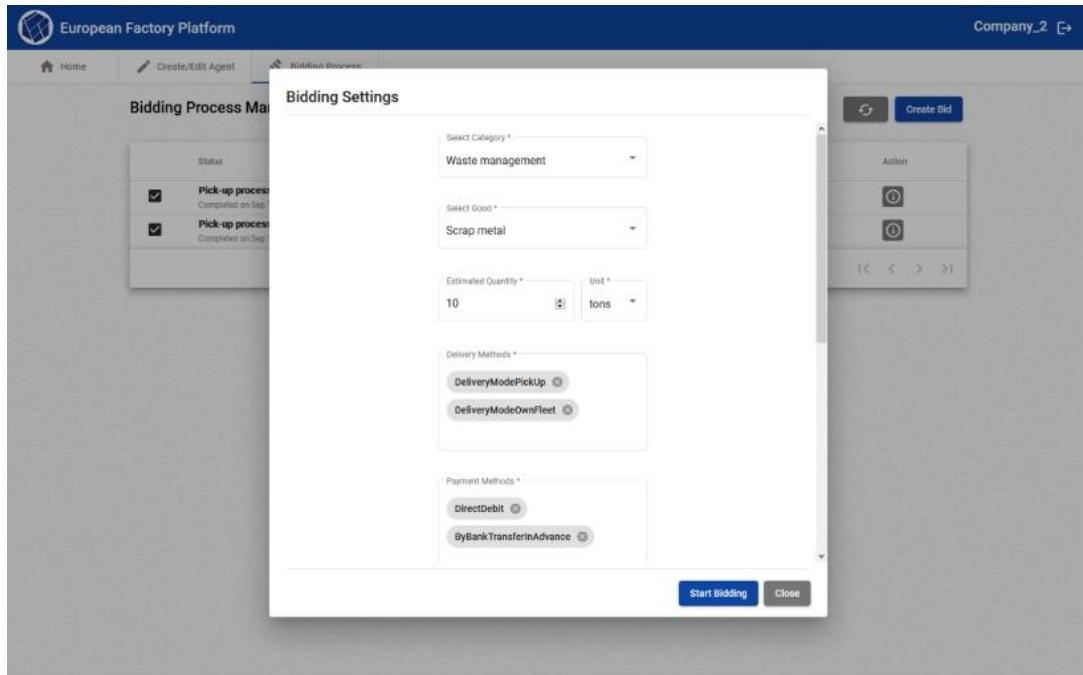


Figure 19: Agent Marketplace - Bidding Settings Popup II

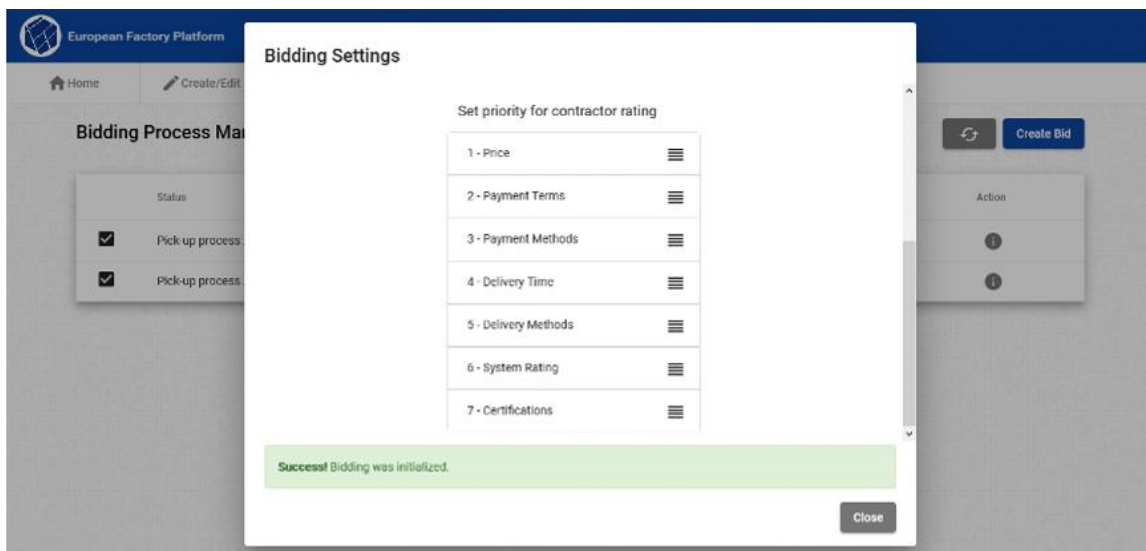


Figure 20: Agent Marketplace - Bidding Settings Popup III

Furthermore, a user is able to explore offer details or create pre-defined bids that enable the fully automation of the process from suppliers' side.

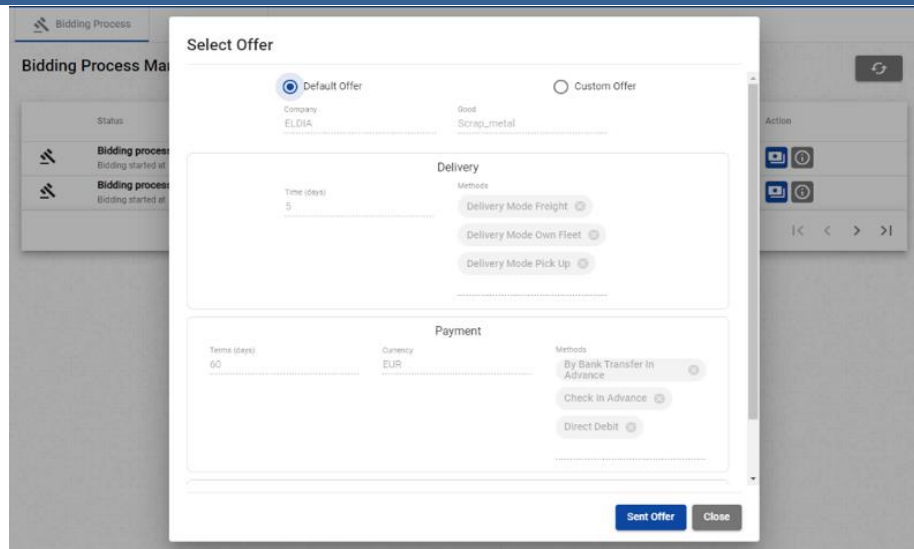


Figure 21: Agent Marketplace – Setup Default Bid

The Matchmaker component is available as an API. The component is an application for automated online bidding through agent-level and offer-level matchmaking. It is an Ontology based framework which applies semantic rules and SPARQL queries to the dedicated Ontology for requesters and suppliers straightforward matching and implements weighted criteria assessment for offer evaluation and best offer suggestion. The Matchmaker is connected with the Marketplace agents and stakeholders through RESTful web services and HTTP protocol.

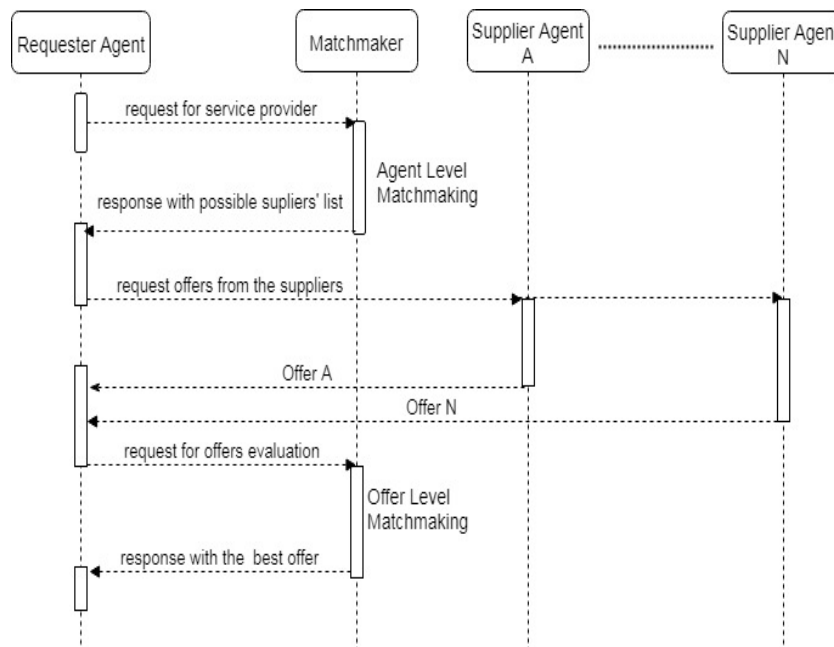


Figure 22: Agent Marketplace - Information Flow

More details about Matchmaker component are available on D5.3: Matchmaking and Intelligence Gathering.

The Agent Deployer provides a REST endpoint to be used by the UI in order to provide the customisation agent and deploying service. The actual endpoint implementation is described in Annex C: Data Models and Interfaces. The data model used by the API has

been designed according to the needs of the integration process. It is composed by two main sections: one dedicated to new agent configuration and one to the matchmaker input data.

2.2.5 Major Updates from M18

Further research and development were conducted related to agent framework, matchmaking, and overall bidding process. Especially the agents' back-end part was fully updated in order to support the automated creation of agents by end-users.

New interfaces for exploring offers history and bidding delays were designed and developed alongside with interfaces (and back-end services) for setting up default offers/bids.

The agent communication mechanism was aligned with Data Spine integration needs. Same for UIs and matchmaking component. A lot of integration rounds among agents, matchmaking back-end and UIs have taken place until the final delivery.

2.3 Accountancy Service

2.3.1 Scope and Relationship with Other EFPF Components

The Accountancy Service has been developed within the project as an integral part of the EFPF Marketplace Framework and provides insight into users' interactions with the EFPF Platform as well as its connected marketplaces. This includes transactions that EFPF users make on different marketplaces, which are linked with the EFPF Marketplace Framework.

A taxonomy has been setup to identify the trackable user actions in which action items are listed in 'subject, verb, object' manner and these actions include users' basic interactions with various parts of the EFPF Platform such as login, register, inviting other users as well as payments realized on external marketplaces if the user has initiated his/her journey from EFPF Marketplace. In this way, when a user performs a certain action on either EFPF Portal or a connected marketplace, corresponding information is sent to Accountancy Service to be persisted so that it can later be visualized to extract valuable information.

The Accountancy Service has been developed based on Elastic Stack which comprises the following components:

- **Elasticsearch:** Stores, indexes, provides, and manages user logs to be later analysed. Since relational databases are not well-suited for managing log data, a NoSQL database like Elasticsearch is preferred due to their flexible and schema-free document structures, enabling analytics of the log data.
- **Logstash:** Gathers user behaviour data from various components of the EFPF platform, executes different transformations and filters the content, before sending the data to the Elasticsearch component
- **Kibana:** Enables interactive dashboards, filters and advanced data analysis and exploration of user logs.

In addition, the following custom modules listed below were developed to provide additional functionality:

- **Reporting Component:** Creates periodic (i.e. monthly) reports for each dashboard at the end of each month in PDF format and sends it as an email
- **Invoicing Component:** Processes all the payment data accumulated within each month, sums all the amounts from successful transactions realized on each

marketplace, calculates a corresponding cashback amount, and creates a detailed invoice with the information including purchased products, dates of transactions as well as the calculated commission for each product. The invoice will then be used to charge marketplaces.

2.3.2 Requirements and Realisation

Tracking the user behaviour enables businesses to make productive decisions and develop effective business strategies. This is a valuable feature of a federated digital platforms, and the Accountancy Service provides this to support the long-term sustainability of the EFPF platform, beyond the span of the project. The Accountancy Service also aims to track and trace a user’s journey across the EFPF ecosystem and collect data about the transactions they make on different marketplaces. The collected data logs are then used to carry out a cashback mechanism enabling a commission charge or a referral fee to be made to the marketplace where a EFPF user carries out a business transaction (Figure 23). In addition to the log collection, customizable dashboards are also needed for better and easier tracking of user interactions. Therefore, in order to process the accumulated log data and address all the above requirements, the Accountancy Service uses Elastic Stack (Elasticsearch, Logstash, Kibana) as an advanced log persistence, monitoring, processing, and visualization framework.

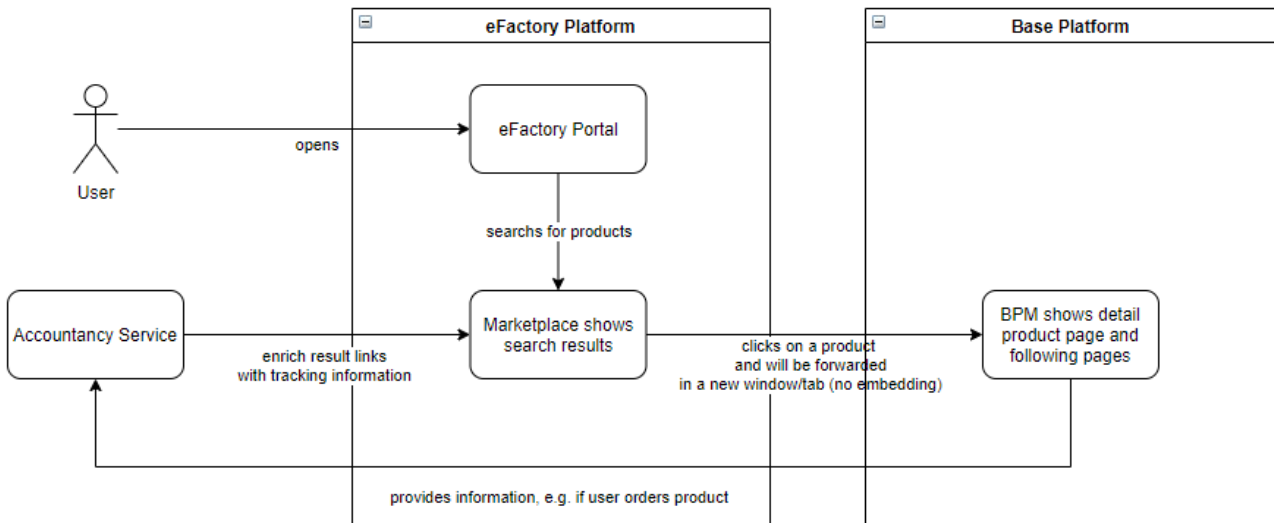


Figure 23: Accountancy Service - Cashback Process

2.3.3 Deployment

Accountancy Service is a standalone component that runs independently of existing EFPF tools and services and can be integrated with unlimited number of external marketplaces. Each of the 5 components of the Accountancy Service has its own Docker image and runs on a corresponding Docker container. Moreover, a production-ready configuration allowing Elasticsearch running on a multi-node cluster is also available. Currently, all the components of the Accountancy Service run on a production server hosted by C2K and a test server hosted by SRDC.

2.3.4 Execution and Usage

Accountancy Service uses Logstash as a data ingestion and server-side data processing pipeline. The logs sent to Logstash are forwarded to Elasticsearch for persistence after

executing certain ingestion pipelines; and then Kibana dashboards are automatically updated based on the certain fields stored on Elasticsearch. In other words, Accountancy Service uses a basic data model for visualization and logs must conform to a basic data model so that Kibana dashboards can be updated automatically

Currently, there is a running instance of Logstash component that is registered to the EFPF Service Registry, which is publicly available through a public endpoint so that events from EFPF Portal and external marketplaces can be sent using HTTP POST method. Events are modelled as a JSON message related with the action conforming to the data model. This will be enough for the Accountancy Service to capture the data and update the dashboards.

Accountancy Service uses Kibana to visualize the data ingested through Logstash and persisted on Elasticsearch. Kibana provides mechanisms to create interactive dashboards with filtering as well as advanced data analysis capabilities and Accountancy Service provides 4 different dashboards for log visualization. Furthermore, all these dashboards can either be accessed on the Kibana instance hosted on EFPF production environment and EFPF Portal Admin pages. Details of the 4 dashboards can be seen below:

- **Payments Dashboard:** Displays all payments realized on marketplaces as well as the corresponding cashback (commission) amounts calculated for each transaction (Figure 24).
- **Marketplace Usage Dashboard:** Visualizes marketplaces usages in terms of most frequently used search keywords, queried platforms, and their distribution (Figure 25 **Error! Reference source not found.**).
- **Platform Engagement Dashboard:** Displays base platform visits and tool/service usages and tracks the frequency of these usages (Figure 26).
- **User Activities:** Visualizes user actions such as login and register (Figure 27).

In addition to the central functionalities offered by the Elastic Stack, Accountancy Service also provides extra features such as preparing monthly reports based on the accumulated data and generate invoices in accordance with the commissions calculated for successful transactions that users perform on the external marketplaces connected to the EFPF Platform. Since these functionalities require custom implementations, these add-on modules are written in JavaScript and provided as a Docker image. At the end of each month, both modules process relevant data of the target month and generates 2 PDF documents: One for the monthly report including all charts updated for the target month and one for the invoice document containing all the transaction details (e.g. purchased products, their prices, transaction dates, calculated commission, etc.) to charge each external marketplace.

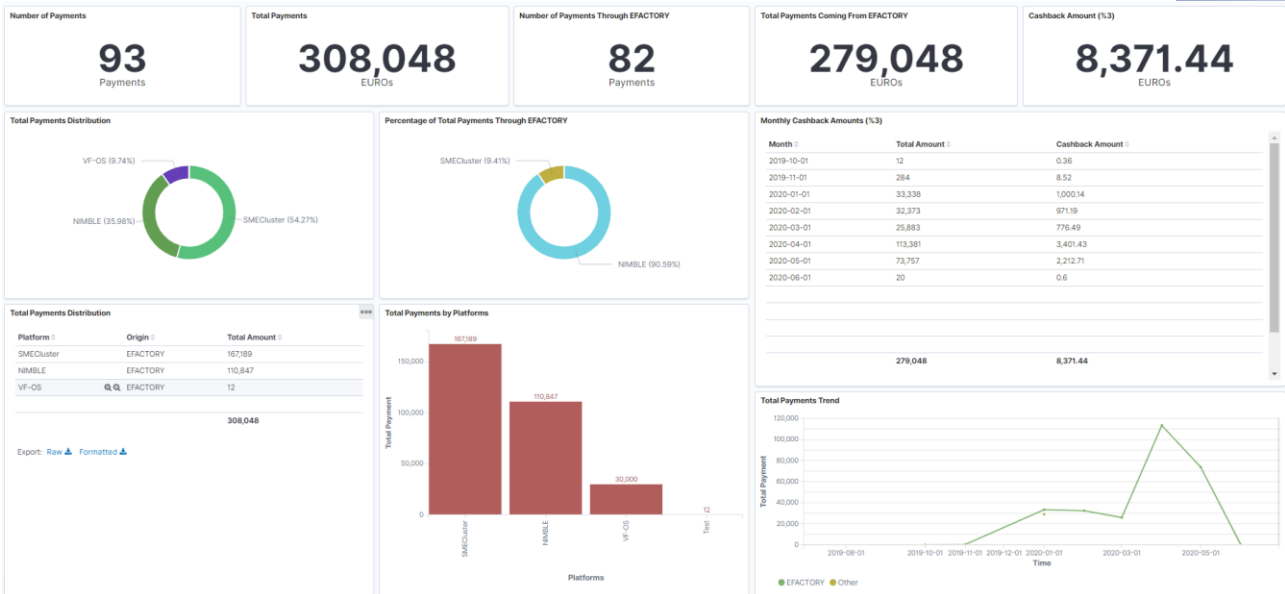


Figure 24: Accountancy Service - Payments Dashboard

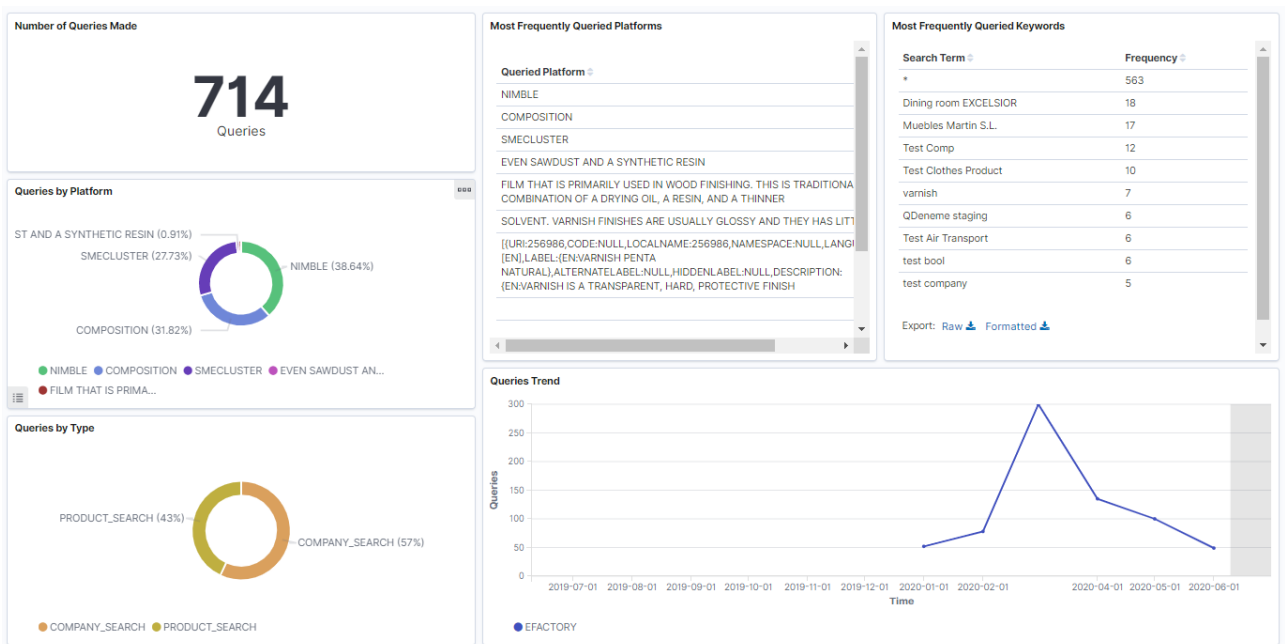


Figure 25. Accountancy Service Marketplace Usage Dashboard

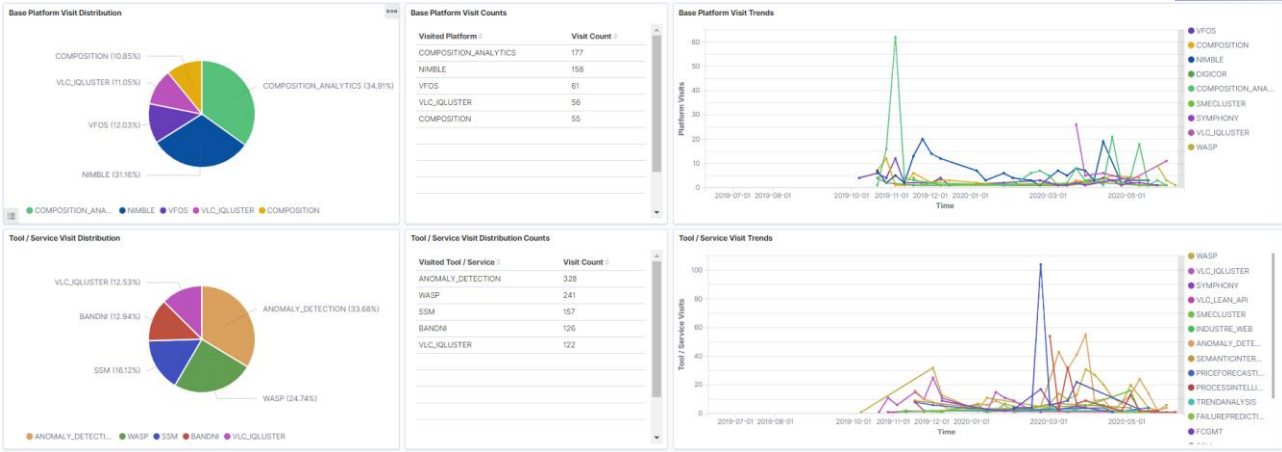


Figure 26: Accountancy Service - Platform Engagement Dashboard



Figure 27: Accountancy Service - User Activities Dashboard

2.3.5 Limitations and Further Development

Currently, there are no limitations regarding the Accountancy service development and previously identified limitations were resolved. New marketplaces can be integrated, and new dashboards will be created upon request to track different types of user interactions.

3 EFPF Portal

The EFPF Portal component is the unification point of distributed tools and platforms in the EFPF ecosystem. It allows the user to access connected tools, base platforms, marketplaces, experiments, and pilots through a unified interface.

3.1 Current Status

This section provides an account of the current status of EFPF Portal and will describe a typical EFPF user journey.

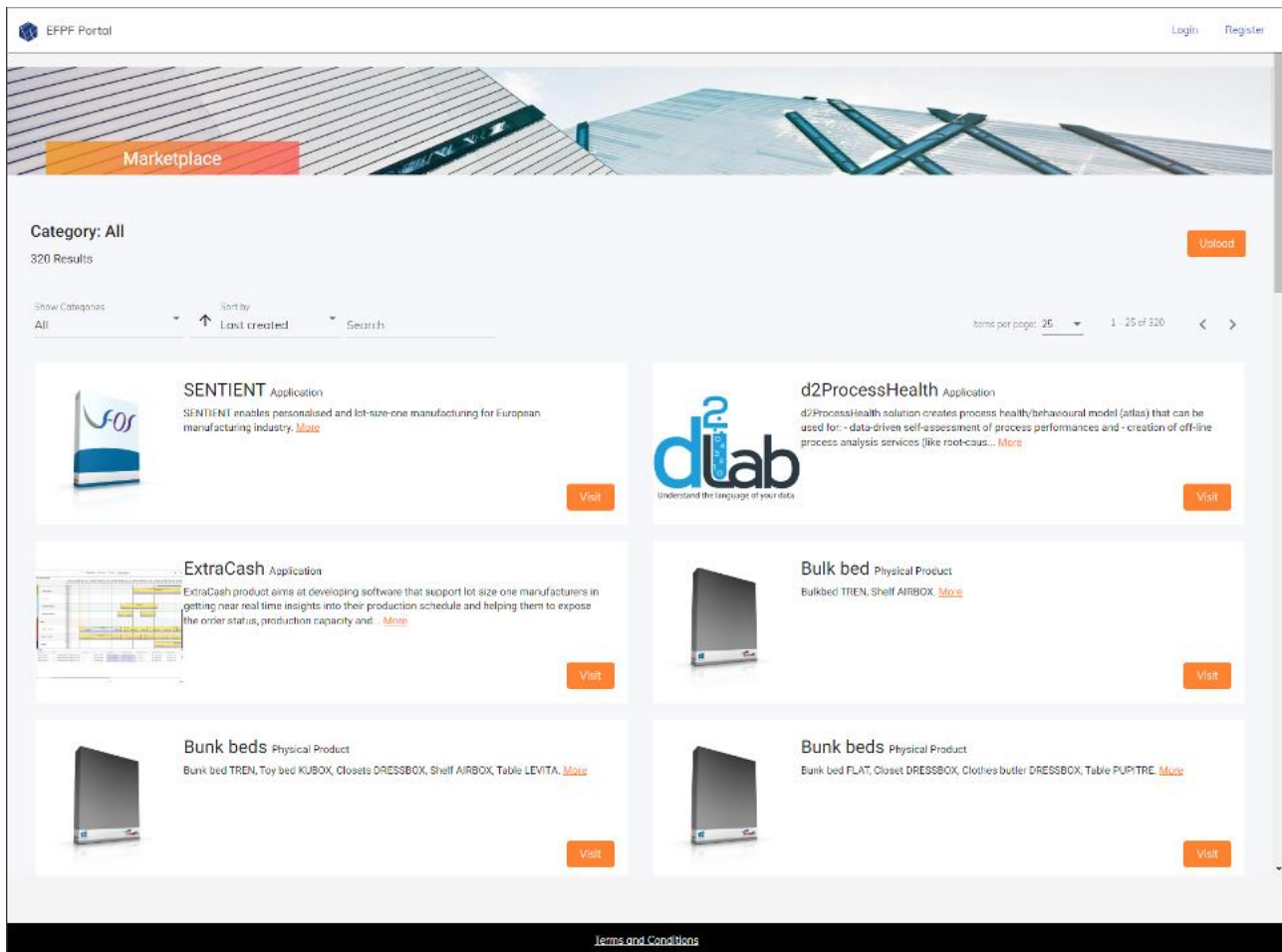


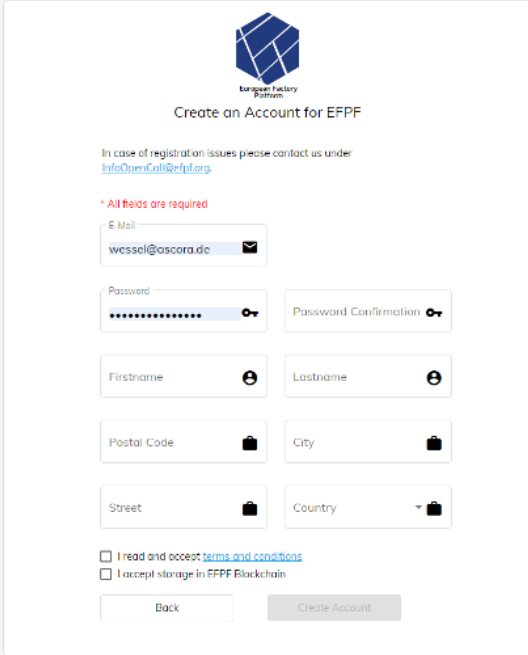
Figure 28: EFPF Portal - Landing Page

The first page a user will open is the landing page of the EFPF Portal, which can be accessed by this link⁹. The landing page provides the EFPF Marketplace UI, which allows the user to browse through available products provided by connected marketplaces. Additionally, users can login with an existing EFPF account or create a new EFPF account by registering themselves to the EFPF Portal. The landing page also provides the user with a link to the Terms and Conditions, so that they can read through them and ensure that it aligns with their expectations before joining. The rest of the user journey is explained through separate sections below.

⁹ <https://portal.efpf.org>

3.1.1 User Registration

First time visitors can click on ‘Register’ button to sign up to the platform. Upon clicking on Register, a simple and quick registration form appears on the screen as shown in Figure 29.



The screenshot shows a registration form titled 'Create an Account for EFPF'. At the top is the EFPF logo and the text 'European Factory Platform'. Below the title, it says 'Create an Account for EFPF'. A note states: 'In case of registration issues please contact us under InfoOpenCall@efpf.org'. A red asterisk indicates '* All fields are required'. The form contains the following fields: 'E Mail' (with the example 'wessel@oscora.de'), 'Password' (masked with dots), 'Password Confirmation', 'Firstname', 'Lastname', 'Postal Code', 'City', 'Street', and 'Country'. At the bottom, there are two checkboxes: 'I read and accept [terms and conditions](#)' and 'I accept storage in EFPF Blockchain'. There are 'Back' and 'Create Account' buttons.

Figure 29: EFPF Portal - Registration Form

The registration form asks for typical personal information. Once the user provided the required information and clicked on “Create Account”, a user account has been set up and a welcome email will be sent to the new user providing general information about the EFPF Portal and the next steps. A new account requires a manual review by default, where an EFPF admin checks the details of the account first. If the account is valid, the admin enables it and the user can continue to set up the account by verifying the email address.

3.1.2 Login

A registered user can click on ‘Login’ button on the landing page. The user is then redirected to the login screen (Figure 30) where they can enter their credentials and will be forwarded to the dashboard of the EFPF Portal.



Figure 30: EFPF Portal - Login Screen

3.1.3 Company Registration

The EFPF Portal provides a company management feature, which can be used in the future for managing access to different tools and services of the EFPF platform. Currently this feature allows the following use cases:

- Create, edit and delete companies
- Join, remove and change roles of company members
- Manage companies for EFPF platform admins

The user can create or join an existing company via the company registration dialog (see **Error! Reference source not found.**) which will be automatically shown upon first login of the EFPF Portal.

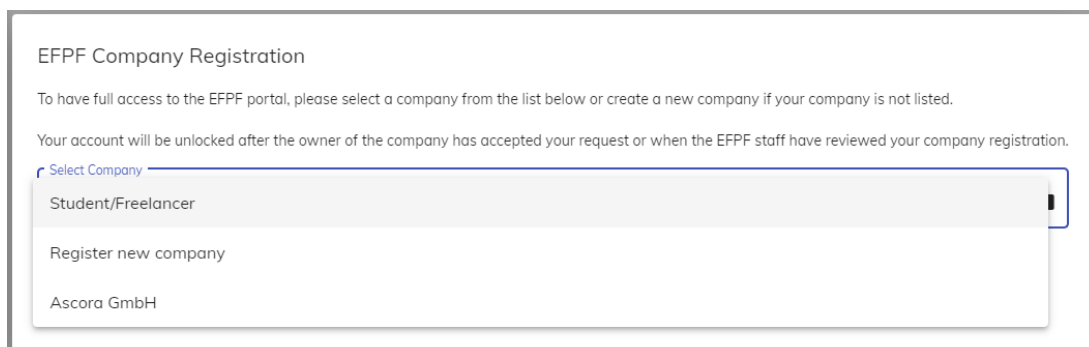


Figure 31 EFPF Portal - Company Registration - Company Selection

If the user’s company is not available, it can be registered by selecting “Register new company” and providing the required information shown in Figure 32.

The screenshot shows the 'EFPF Company Registration' page. At the top, there is a heading and two lines of instructional text. Below this is a 'Select Company' dropdown menu with 'Register new company' selected. Underneath are several input fields: 'Company Name', 'VAT identification number', 'Street', 'Postal code', 'City', 'Country', and 'Sector'. Each field has a small lock icon on the right side. At the bottom of the form, there is a grey 'Confirm' button and a blue 'Log out' button.

Figure 32: EFPF Portal - Company Registration - Register New Company

After the registration has been completed, the user can start using the EFPF Portal. In the meantime, an EFPF admin will review the provided company details. Once accepted, the user can enter the company management view via the “Manage company” menu entry as seen in Figure 33.

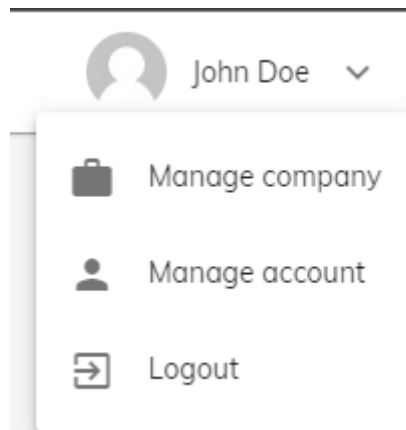


Figure 33: EFPF Portal - Company Registration - Manage Company Menu Entry

The company owner can edit the company details, enable the listing of the company in the company registration dialog (see **Error! Reference source not found.**) and manage members including roles.

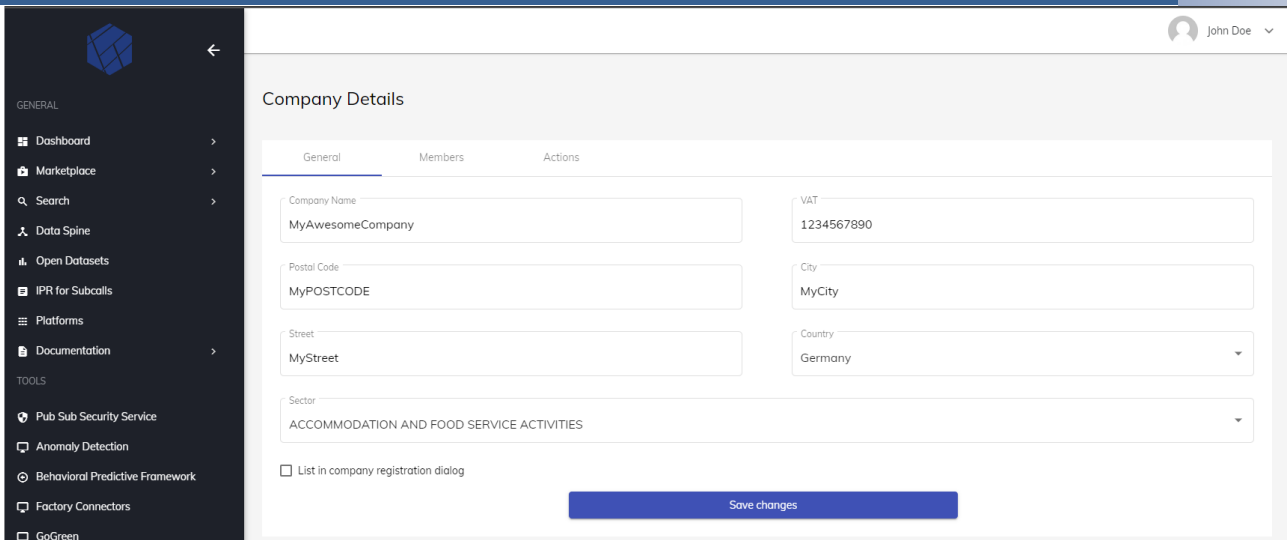


Figure 34: EFPF Portal - Company Registration - Manage Company

As a EFPF admin it is possible to manage all companies including activation, deactivation and deleting (see Figure 35).

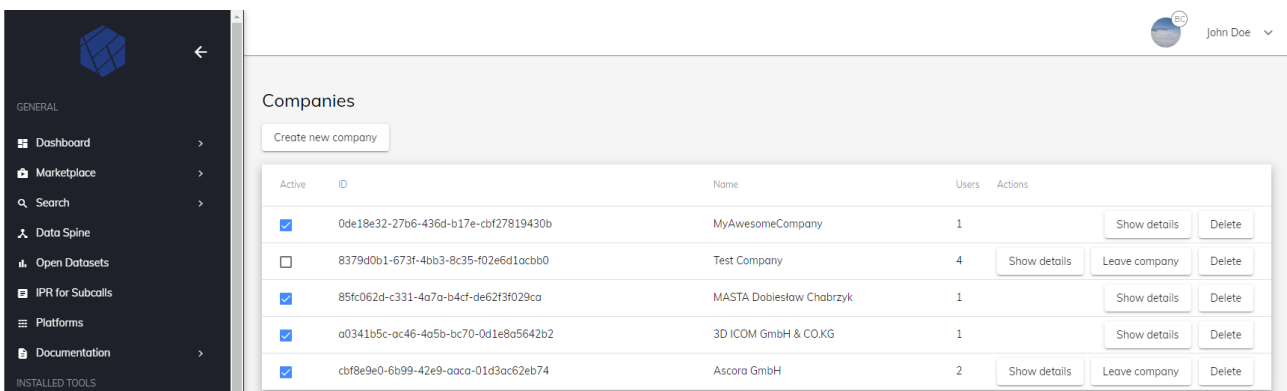


Figure 35: EFPF Portal - Company Registration - Manage Companies

3.1.4 Dashboard

After login, the dashboard is the default website being shown. The dashboard has been designed with a view to showcase the breadth of capabilities available on the platform. The messaging has been adapted to bring out customer benefits, so that they can understand the application and usefulness instantly.

A direct link to access the marketplace is provided for those specifically looking to explore the digital marketplace.

Further an array of tiles each pointing to a different value propositions (see Section 0) available on the platform can be seen in the centre of the dashboard. The aim is to provide the user ease of navigation by bundling supplementary tools and services based on their application. Figure 36 shows the overall layout of the dashboard.

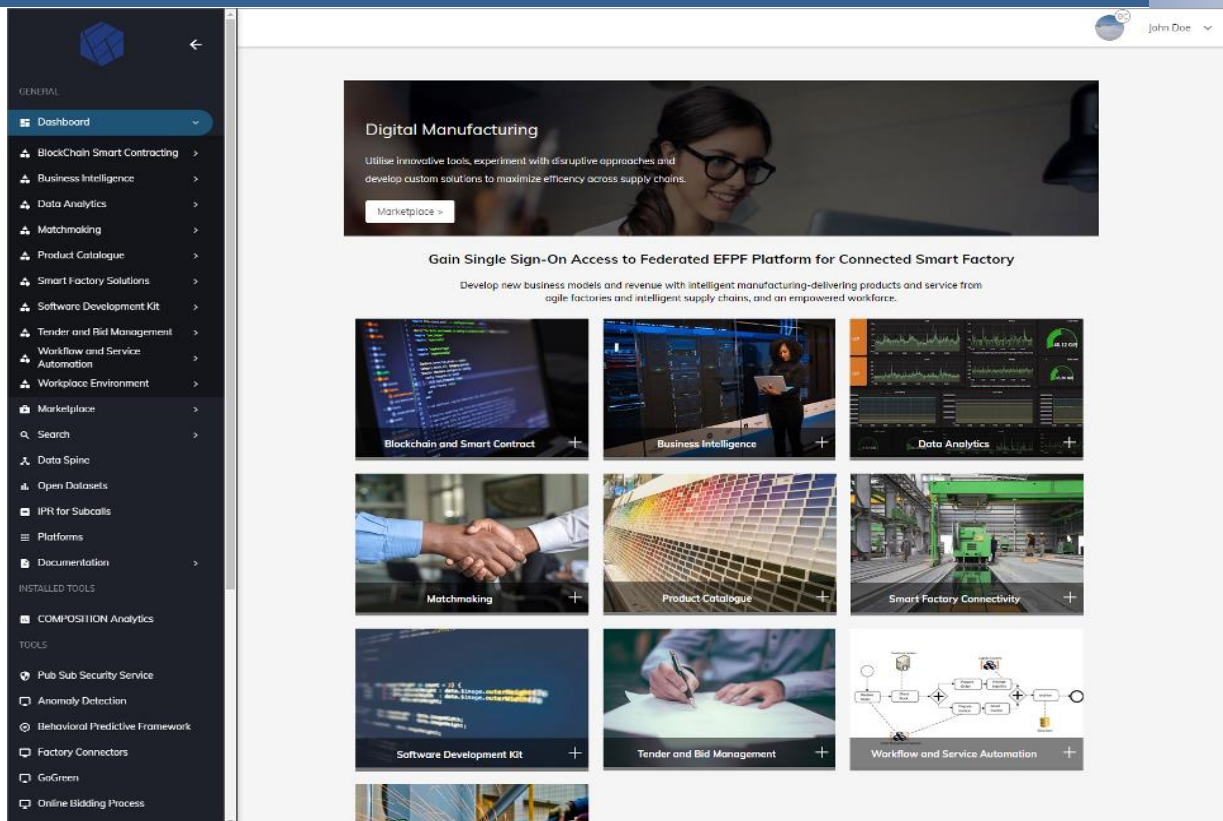


Figure 36: EFPF Portal - Dashboard

The side menu provides the general navigation for the EFPF Portal, which can be used at any moment while on the platform. Finally, at the bottom of the dashboard page 5 key links are provided, the use and application of which are described below:

- **Contact Us:** Intended to provide users with contact details of the governing body (in this case, European Factory Foundation) including contactable email, phone number and other relevant details.
- **Find a Partner:** This links the user to the company matchmaking search, where users filter and search for companies based on multiple attributes like sectors and business types.
- **Support:** This link will be forwarding the user to a support page, which enables users and developers to get in contact with the platform support team.
- **EFPF Ecosystem:** This links the user to European Factory Foundation website. This website can also be reached by following this link - <http://ef-foundation.com/> .
- **Public Portal:** This links the user to the public website for the EFPF project. This website can also be reached by following this link - <https://www.efpf.org/> . The project website aims to disseminate project's efforts through regular updates on the technical progress, shows attended and constant blog posts.

3.1.5 Value Propositions

In order to describe the unique value propositions (VP) that the platform can deliver for the user, the dashboard will provide hyperlinks to VP pages to provide detail. The VP page also presents actual solutions to bring out the range of possibilities. This VP has been designed and refined to provide detailed information about the possibilities of the platform to attract users and create engagement. Following this, individual VP's were created and organised

in a simple to understand fashion in the portal. Initially the VP explains overall ambition of the solution in a few lines. It is then followed by a concise description of the key benefits as well as real life implementation by pilot partners and their experience.

Blockchain and Smart Contracting
Seamless connectivity and secure data exchange for a new level of digital transformation in dynamic industrial supply chains

Reduce cost and establish trust in the delivery process
A flexible and open ecosystem that uses the blockchain based distributed ledger technology to provide a transparent, secure and reliable log of the distributed activities. The blockchain technology allows new actors to easily join the existing supply chains.

Provides transparency support for the Circular Economy
Blockchain provides a secure platform for transparent information exchange among different stakeholders in the circular economy. The blockchain platform also provides a trusted and cost-effective way to exchange business documents where the origin and integrity of data is preserved.

KLEEMANN ELDIA miloil

Manufacturing companies like KLEEMANN, ELDIA and MIL Oil are using the blockchain technology to track and trace the flow of materials in closed-loop supply chains. The use of blockchain technology allows the companies to determine the status of their shipments in real-time e.g. where is a specific shipment at any moment and who is handling it in the supply chain. The eFactory blockchain platform provide mobile app and web-based user interfaces that allow users to make use of the rather complex blockchain technology with much ease. Using the mobile app, the shipment handlers can record the handover processes (with relevant information, including multimedia-based evidence) in the distributed ledger for future reference. The use of blockchain technology in the circular economy offers new possibilities for data-driven business models in the industrial sectors.

Blockchain-based Solutions for Industrial Supply Chains

Delivery process support Circular economy support

Delivery process support

The blockchain-based delivery tracking solution uses a distributed ledger for recording transactions and events. Smart contract process logic and a DApp framework is developed to provide a configurable system for monitoring delivery processes and verifying events in the supply chain. Users can define a delivery process based on the pre-defined handover procedures and verify identity of actors and products using mobile units, QR codes and NFC. The web interface and DApps can be used to manage and monitor the status of active processes. This solution provides a flexible open way to ensure trust, lower handling costs and reduce errors in the delivery chain.

[Get Started >](#)

Figure 37: EFPF Portal - Value Proposition Page Example

3.1.6 Federated Search

The federated search component, which has been developed and provided by T4.5 - Matchmaking and Agile Networks Creation, offers the user t three different types of searches:

- Products (See Figure 38)
- Companies (See Figure 39)
- Business Opportunities (See **Error! Reference source not found.**)

More information regarding this component can be found in deliverable D5.1: EFPF Matchmaking and Intelligence Gathering.

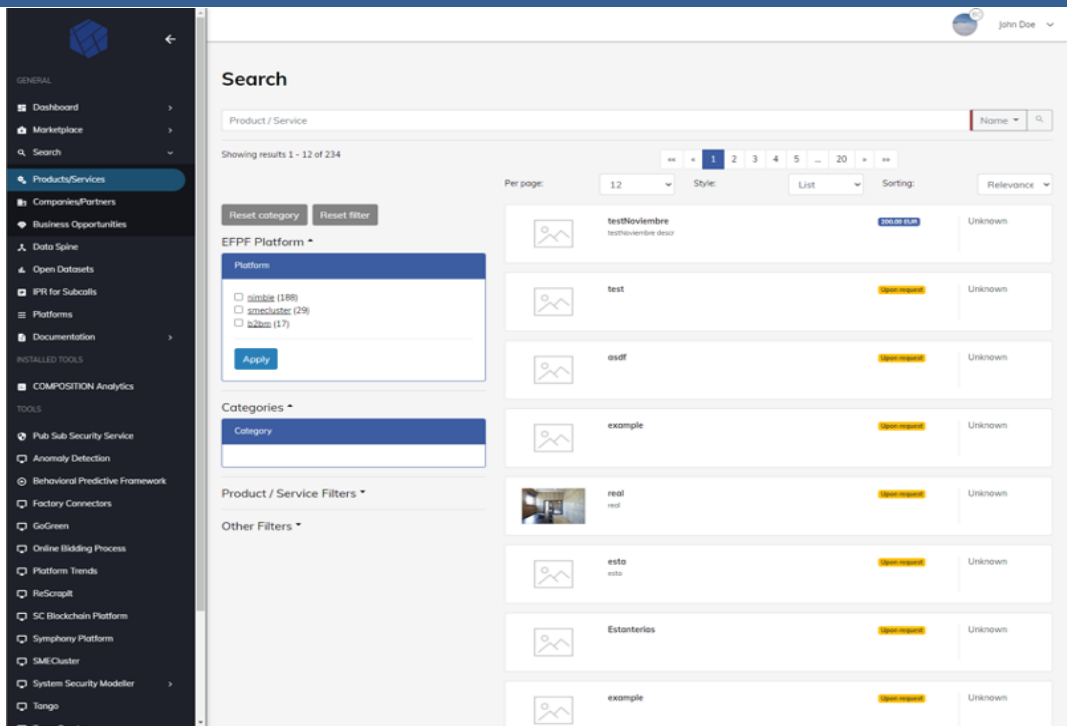


Figure 38: EFPF Portal - Federated Search – Products/Services

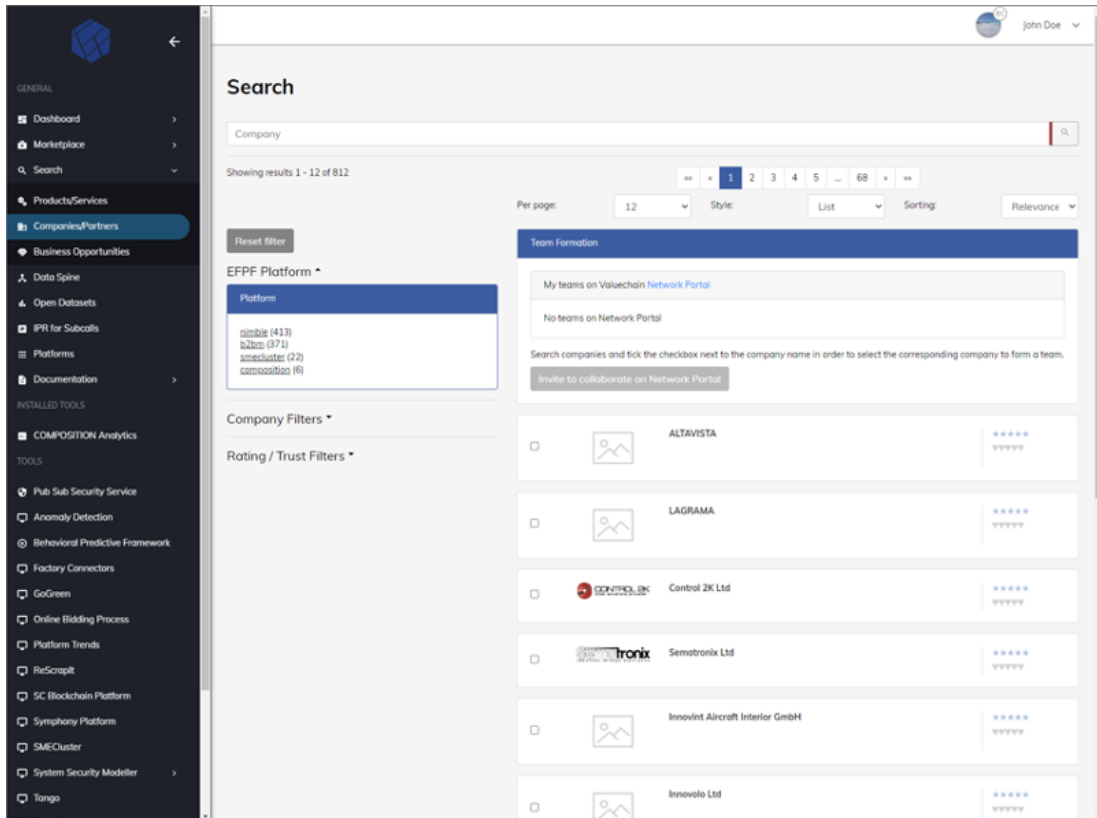


Figure 39: EFPF Portal - Federated Search – Company/Partners

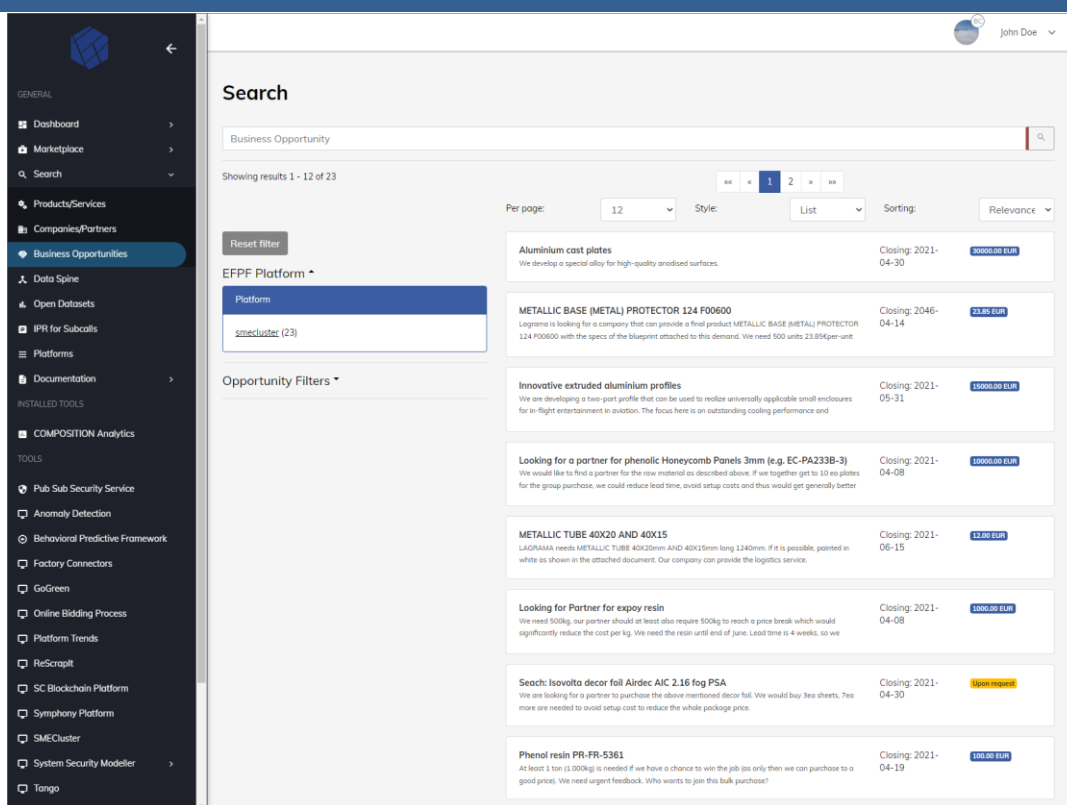


Figure 40 EFPF Portal – Federated Search – Business Opportunities

3.2 Scope and Relationship with Other Components

The EFPF Portal component provides the main user interface of the EFPF platform. It is being connected to other components in the EFPF platform. The following list provides an overview of the relationship regarding connected components and a description of the functionality it enables:

- **EFPF Security Portal:** This component secures access and communication of the EFPF portal.
- **Smart contracting component:** This component is being used at user registration time and enables the storage of registration information, i.e. does the user agreed to the EFPF Terms and Condition.
- **Accountancy Service:** This component is being used to manage track user events like visiting an external platform or conducting a product search.
- **External mail service:** The EFPF Portal Backend requires an external mail service to send out notification or confirmation emails, i.e. when a user registered successfully a welcome email will be sent out with the next steps.

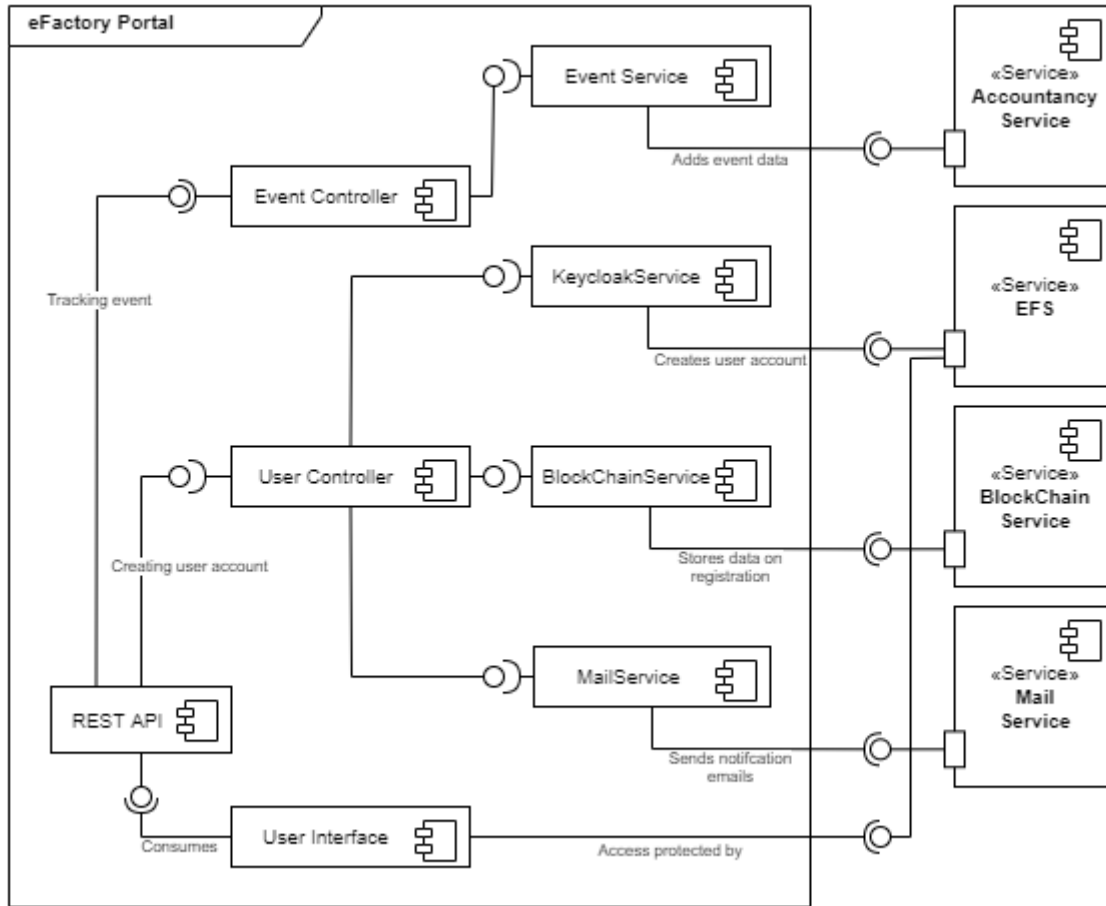


Figure 41: EFPF Portal - Architecture Diagram

3.3 Requirements and Realisation

The EFPF Portal component is divided in two components. The first component is the EFPF Portal, which is a single-page application (SPA) based on the Angular web application framework¹⁰. The second component is the EFPF Portal Backend, which is responsible for providing the required data for the frontend. It handles also different tasks like registration and event logging.

Both components are provided as Docker images, which enables a fast and uncomplicated way to deploy them. It is recommended to use the Linux distribution Ubuntu 22.04 LTS as the operating system with Docker Engine v20.10 installed. Any other operating system can be used too, but it cannot be guaranteed that it will work successfully with the Docker version.

3.4 Deployment in the EFPF Platform (Installation)

This section will provide information on how the EFPF Portal components can be deployed. Both components (EFPF Portal and EFPF Portal Backend) are provided as Docker images build by the CI/CD feature of GitLab¹¹. GitLab stores Docker images in an integrated Docker

¹⁰ <https://angular.io/>

¹¹ <https://about.gitlab.com/>

container registry, to which a Docker Engine can connect and retrieve the required Docker images. As the Docker registry is not public, the Docker Engine installation first must be connected to the Docker registry. This can be done by the Docker login¹² command.

\$ docker login --username foo --password-stdin registry:8080

As the server environment may use Portainer¹³ for Docker container management, a connection to the GitLab registry does not require a manual login but will be able through UI configuration.

There are currently two environments in use: The “Test” and the “Prod” environment. In order to deploy the components in any of the environments, a repository with scripts have been set up, which are being maintained by their respective component owners. Therefore, the deployment has no need for command line interaction but is being done via the GitLab CI/CD UI and pipelines.

3.5 Execution and Usage

Both components will be available after the deployment. The documentation of the EFPF Portal Backend REST API is provided by Swagger¹⁴ and provides information to developer how the EFPF Portal is communicating with its backend component.

The EFPF Portal can be accessed via browser (see Section 3.1 for more information). After a successful login, the user will be forwarded to the Dashboard and can use the provided tools and services of the portal.

3.6 Limitations and Further Developments

The following limitations for the EFPF Portal exist:

- User registration process requires manual actions

¹² <https://docs.docker.com/engine/reference/commandline/login/>

¹³ <https://www.portainer.io/>

¹⁴ <https://swagger.io/>

4 Conclusion and Outlook

In this final report, the following items have been presented: the implementation of the requirements established in M18 to establish a federated ecosystem and its evolution and extension; the required governance leading for a growth of such an ecosystem; and the current state of the design and realisation of the EFPF Marketplace and the EFPF Portal.

The EFPF Portal provides a unified interface for the user to access all the tools and services provided by the EFPF ecosystem. Within the portal integrates the EFPF Marketplace providing access to multiple third-party marketplaces in a onestop-shop approach. EFPF Marketplace is free to access and provides advanced federated search functionality to facilitate the lookup for apps, tools, products, software as well as consultancy services, hosted or offered by multiple Marketplaces and platforms.

The Accountancy Service provides interactive dashboards to visualize collected information and allows you to pull insights out of user behaviour and transaction data. The interactive dashboards provide advanced filtering and data analysis mechanisms to help you to reach specific information in a faster way. In addition, the Accountancy Service also offers automated tools for preparing monthly reports based on the accumulated data and generating monthly invoices in accordance with the commissions calculated for successful transactions that users perform on the marketplaces connected to the EFPF Platform.

Annex A: History

Annex B: References