



Grant Agreement No.: 732638
 Call: H2020-ICT-2016-2017
 Topic: ICT-13-2016
 Type of action: RIA



D4.02: First TaaS Prototype

Work package	WP 4
Task	Task 4.2
Due date	30/06/2018
Submission date	10/11/2018
Deliverable lead	Fraunhofer
Version	2.3
Authors	Jonas Rook (TUB), Alexander Willner (Fraunhofer), Francesco Verde (Martel), Loic Baron (UPMC), Radomir Klacza (UPMC)
Reviewers	Francesco Verde (Martel)

Abstract	This deliverable describes the first prototype with the one-stop-shop with matchmaking capabilities, including the interactions between several components. A manual describes how customers and testbed providers can interact with the prototype.
Keywords	One-Stop-Shop, matchmaking, integration, testbed provider, spread-activation.



D4.02: First TaaS Prototype

Document Revision History

Version	Date	Description of change	List of contributors
V0.1	01/04/2018	TOC	Jonas Rook (TUB)
V0.2	14/04/2018	Clean-up	Jonas Rook (TUB)
V0.3	06/06/2018	Added more content and responsibilities.	Jonas Rook (TUB)
V0.4	26/06/2018	Added content for the sections Design and Manual. Added abstract and keywords.	Jonas Rook (TUB)
V0.5	27/06/2018	Added text for Odoo. Added introduction.	Francesco Verde (Martel) Alexander Willner (Fraunhofer)
V0.6	29/06/2018	Added content for the matchmaking capabilities. Added content for the conclusion.	Jonas Rook (TUB)
V0.7	02/07/2018	Added executive summary. Added D4.1 summary. Added Architecture description. Added citations.	Jonas Rook (TUB)
V0.8	23/09/2018	Added content for the section Design for MySlice.	Loic Baron (UPMC)
V0.9	27/09/2018	Added content for the manual. Added content for the section Future Work	Alexander Willner (Fraunhofer) Radomir Klacza (UPMC)
V1.0	10/10/2018	Changed the lead creation section	Francesco Verde (Martel)
V2.0	25/10/2018	Odoo: added more details	Francesco Verde (Martel)
V2.1	01/11/2018	Updated Executive Summary, Introduction, Design and Manual sections.	Alexander Willner (Fraunhofer)
V2.2	07/11/2018	Fixed: colours/fonts, styles, match figure-number page. Comments.	Francesco Verde (Martel)
V2.3	10/11/2018	Final review, rephrasing, clean-up, fixed formatting & styles, fixed references, fixed spelling, fixed tables, fixed aspect ratio of figures, fixed references, fixed abbreviations, fixed capitalization, fixed fonts, ...	Alexander Willner (Fraunhofer)

D4.02: First TaaS Prototype

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the **Federation for FIRE Plus (Fed4FIRE+)**; project's consortium under EC grant agreement **732638** and do not necessarily reflect the views of the European Commission.

The European Commission is not liable for any use that may be made of the information contained herein.

COPYRIGHT NOTICE

© 2017-2021 Fed4FIRE+ Consortium

ACKNOWLEDGEMENT



Co-funded by the
European Union



Co-funded by the
Swiss Confederation

This deliverable has been written in the context of a Horizon 2020 European research project, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation. The opinions expressed, and arguments employed do not engage the supporting parties.



D4.02: First TaaS Prototype

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		R
Dissemination Level		
PU	Public, fully open, e.g. web	X
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to FED4FIRE+ project and Commission Services	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

EXECUTIVE SUMMARY

The most valuable assets testbeds have are actual knowledge and equipment. Therefore, the main objective of this deliverable is to describe a “Testbed as a Service (TaaS)” broker that allows potential customers to book added-value services that might be offered in a testbed in a federation. Such a one-stop-shop can be used to search, find and book a best-matched testbed according to the customer requirements. This might allow new revenue streams to be created.

This deliverable analyzes and describes the needed capabilities that such a system should have in this project. This analysis includes the tools and technologies used to implement the system and provides an overview of a proposed matchmaking algorithm. The implementation is based on the gap analysis conducted in deliverable D4.01 [1]. Therefore, an overview of the deliverable D4.01 will be discussed in brief in the next section of this deliverable.

To sum up, an introduction to the individual components used for the building a TaaS broker is given, followed by a description of their interactions and a manual on how to use the TaaS broker is given. We conclude with an outlook.

TABLE OF CONTENTS

DISCLAMER	3
COPYRIGHT NOTICE	3
ACKNOWLEDGEMENT	3
1 INTRODUCTION	10
1.1 OBJECTIVES	10
1.2 SUMMARY OF THE FIRST DELIVERABLE D4.01.....	10
1.3 CONCLUSION	11
2 DESIGN	12
2.1 SELECTION OF TOOLS	12
2.1.1 MySlice	12
2.1.2 Odoo.....	13
2.1.3 Odoo Implementation Details	15
2.2 MARKETPLACE.....	17
2.3 ARCHITECTURE DESCRIPTION	20
2.4 DESCRIPTION OF MATCHMAKING CAPABILITIES	20
3 MANUAL	22
3.1 SERVICE PUBLICATION	22
3.1.1 Register the Testbed Provider	23
3.1.2 Register the Services.....	24
3.2 SERVICE SELECTION.....	26
3.2.1 Query Demand.....	27
3.2.2 Selection Summary.....	29
3.3 LEAD CREATION.....	30
3.4 EXECUTION	36

D4.02: First TaaS Prototype

3.4.1 Feedback between Customer and Vendor	36
3.4.2 Customer Accepts the Quotation.....	37
3.5 CONCLUSION	39
3.5.1 The Sales Order Becomes an Invoice	39
3.5.2 Invoice and a Confirmation of Customer’s Payment.....	39
3.5.3 Evaluation	41
4 CONCLUSION/FUTURE WORK	42
4.1 MARKETPLACE.....	42
4.2 MATCHMAKING.....	42
4.3 MYSLICE	42
4.4 ODOO	42
5 WORKS CITED	44

LIST OF FIGURES

FIGURE 1: THE INITIAL INTEGRATION PLAN FOR THE PROTOTYPE.....	11
FIGURE 2: MYSLICE-ARCHITECTURE.	12
FIGURE 3: CONFIGURATION OF AN INCOMING MAIL SERVER ON ODOO.	16
FIGURE 4: CONFIGURATION OF SALES CHANNEL.	17
FIGURE 5: MYSLICE INTEGRATION WITH THE MARKETPLACE AS A PLUG-IN.	17
FIGURE 6: CURRENT START PAGE OF THE MARKETPLACE COMPONENT.	18
FIGURE 7: STRUCTURE AND DATA FLOW IN FLUX [5].....	19
FIGURE 8: STRUCTURE AND DATA FLOW OF THE MARKETPLACE PLUG-IN.	19
FIGURE 9: ARCHITECTURE WITH ALL COMPONENTS.	20
FIGURE 10: HYBRID APPROACH FOR SEARCHING IN THE SEMANTIC WEB [4]	21
FIGURE 11: OVERVIEW OF ACTIVITIES FOR SERVICE PUBLICATION.....	23
FIGURE 12: REGISTRATION OF THE FUSECO PLAYGROUND TESTBEDS	24
FIGURE 13: REGISTERING THE SERVICES OF FUSECO PLAYGROUND	25
FIGURE 14: CREATING/ADDING A NEW SERVICE.....	26
FIGURE 15: OVERVIEW OF ACTIVITIES FOR SERVICE PUBLICATION.....	27
FIGURE 16: EXAMPLE OF THE SERVICE SELECTION.....	28
FIGURE 17: SEARCHING SERVICES FOR TESTING BY USING MATCHMAKING.	29
FIGURE 18: REQUEST FURTHER INFORMATION AFTER THE SELECTION PHASE. ...	30
FIGURE 19: OVERVIEW OF ACTIVITIES FOR LEAD CREATION.	31
FIGURE 20: VENDOR’S ODOO NOTIFICATION EMAIL.....	31
FIGURE 21: ODOO LOG-IN PAGE.....	32
FIGURE 22: ODOO BACKEND – VENDOR’S RFQ VIEW.....	32
FIGURE 23: ODOO BACKEND – VENDOR’S NEW QUOTATION VIEW.....	33
FIGURE 24: ODOO BACKEND – VENDOR’S NEW CUSTOMER CONTACT FORM.	34
FIGURE 25: ODOO BACKEND – VENDOR’S NEW QUOTATION VIEW.....	34
FIGURE 26: ODOO BACKEND – VENDOR’S EMAIL VIEW.	35
FIGURE 27: ODOO FRONTEND – CUSTOMER’S WELCOME PAGE.....	35
FIGURE 28: OVERVIEW OF ACTIVITIES FOR EXECUTION.....	36
FIGURE 29: ODOO FRONTEND – EMAIL & MESSAGE SECTION.....	37
FIGURE 30: ODOO FRONTEND - PAYMENT CONFIRMATION VIEW.....	38
FIGURE 31: OVERVIEW OF ACTIVITIES FOR CONCLUSION.....	39
FIGURE 32: ODOO BACKEND – VENDOR’S SALES ORDER VIEW.....	39
FIGURE 33: ODOO BACKEND – VENDOR’S INVOICE VIEW.....	40
FIGURE 34: ODOO FRONTEND – CUSTOMER’S PAYMENT LIST PAGE.	41



ABBREVIATIONS

BSS	Business Support System
CRM	Customer Relationship Management
eTOM	Enhanced Telecom Operations Map
FanTaaS	Testbeds-as-a-Service for the EIT ICT Labs
FedSM	Federated IT Service Management
FIRE	Future Internet Research and Experimentation
IIoT	Industrial Internet of Things
KPI	Key Performance Indicator
MVC	Model View Controller
NLP	Natural Language Processing
OSS	Operations Support System
QoS	Quality of Service
RFQ	Request For Quote
SFA	Slice Federation Architecture
SLA	Service Level Agreements
OPC UA	Open Platform Communication Unified Architecture
TaaS	Testbed as a Service
TSN	Time Sensitive Networking

1 INTRODUCTION

1.1 OBJECTIVES

In a number of conducted studies, the concept “Testbed as a Service” has been covered before. Some work suggests to provide brokering capabilities, in order to allow resource providers to offer different kinds of services in federated infrastructures. For instance, FanTaaS was a project with the EIT ICT Labs initiative, that aimed at brokering services towards large-scale European experimental facilities. Based on similar concepts, within WP4 aims at building a sustainable marketplace, that is, both allowing testbed providers to offer their own services, and allowing the customers to find and book them easily. As a result, testbed providers might be able to create an additional source of revenue. The objectives can be summarised as follows: allowing new revenue streams, creating an open, self-sustainable marketplace for services, and focus on intelligent matchmaking and allow on-demand booking.

This deliverable introduces a prototype that implements these points and thus makes it possible to offer the testbeds as a service. The starting point of this report is the gap analysis that emerged from deliverable D4.1 [1]. The gap analysis showed the lessons learned from similar projects and it’s also presented various tools with which the implementation can be done.

The remainder of this document is structured as follows. Since the results of the analysis have a great impact on this report and on TaaS, it will be listed here again. The design section describes the tools that make up the prototype and how they interact with one other. The manual section describes how the customer, the vendor and the federation can use the TaaS broker. Finally, there is a brief summary and outlook for each individual component and for the TaaS Prototype.

1.2 SUMMARY OF THE FIRST DELIVERABLE D4.01

The gap analysis from deliverable D4.1 showed, on the basis of previous or similar projects, what needs to be considered and what experiences can be taken from them. This included identifying some user groups, functions that must be made available by the system, and what the initial plan for implementation might look like. Figure 1 shows this initial plan.

As can be seen in the figure, the idea was to extend the existing systems with new components so that they meet the new requirements. The Figure 1 also shows the connection to the tasks in Work Package 3, where each testbed has an application with which it can load its services into the Semantic Directory. This in turn enables users to search for them via the TaaS portal. In addition, it should be possible to import data from the existing databases of the testbeds. The Semantic Directory is responsible for processing the uploaded data. It provides the appropriate results on the user requests and should be able to integrate all existing data, such as registered users or testbeds, like the applications for the testbeds.

D4.02: First TaaS Prototype

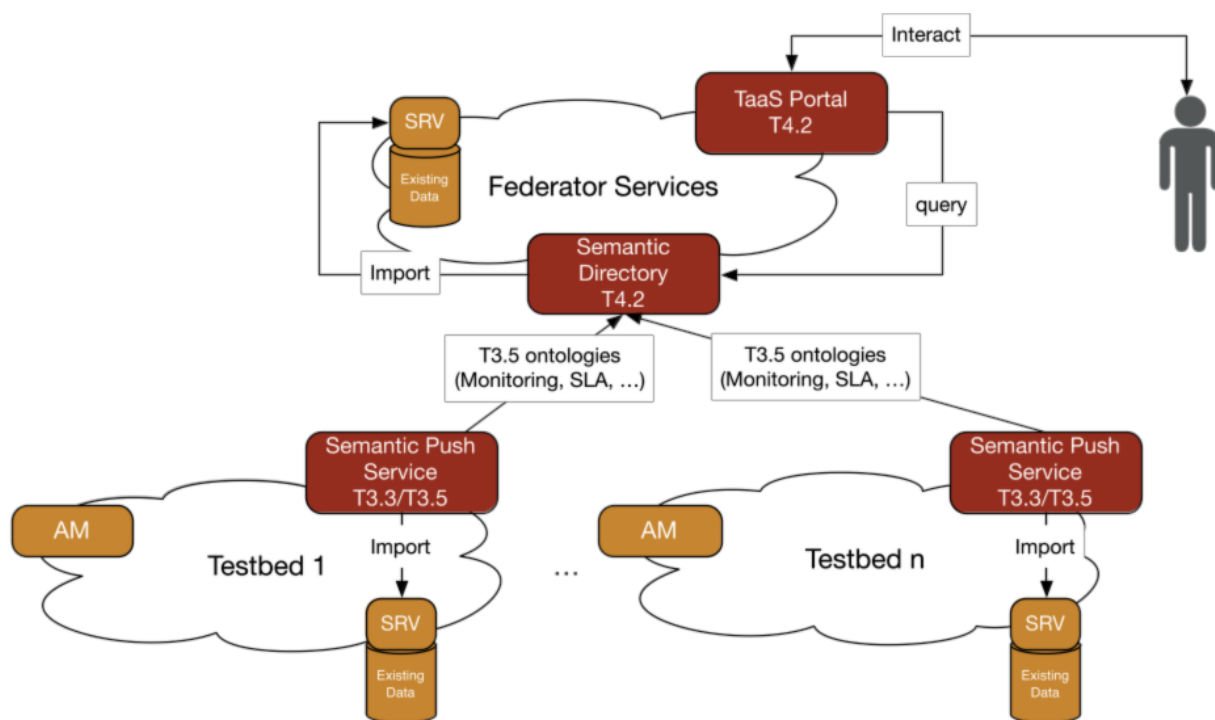


Figure 1: The initial Integration Plan for the Prototype.

1.3 CONCLUSION

Hence, this deliverable presents a prototype and implementation of the Task T4.2. Using D4.01 as starting point, the first prototype has been developed. The extent to which the initial plan was implemented is examined and explained in more detail in the Design section. In the following sections we are going to present the **Design** concept in detail, the **Manual** with a detail description on the usage of the One-Stop-Shop system, the **Conclusion** with the entire deliverable in the nutshell and the **Future Works** with some possible developments and chances in order to make the TaaS broker more efficient.

2 DESIGN

This chapter deals in detail with the selected tools which will be discussed in the next sections. It includes a detailed description and its purpose. It also describes how the tools are connected in order to build the architecture of the first prototype. At the end, the matchmaking system will be presented.

2.1 SELECTION OF TOOLS

Before considering the prototype, all components are introduced. Each of them has its own task, which takes care of the relationship between customers and testbed providers (vendors). In addition, the architectures of the components are examined in more detail and certain design decisions are explained clearly.

2.1.1 MySlice

The goal of the GUI component is to provide a simple tool to use, low barrier entry point to allow customers to discover available offerings (technical and non-technical). While the initial discovery phase should not require any prerequisite from the user (e.g., only a modern web browser or client for RESTful services), a handover to an authenticated service request is envisioned. One existing tool that can be reused and extended is MySlice, which has already been developed and used within a number of FIRE projects. It is envisioned to extend it towards the specific requirements for WP4. Its architecture is depicted in Figure 2 and services are responsible for gathering data from the distributed sources of the architecture. The Web

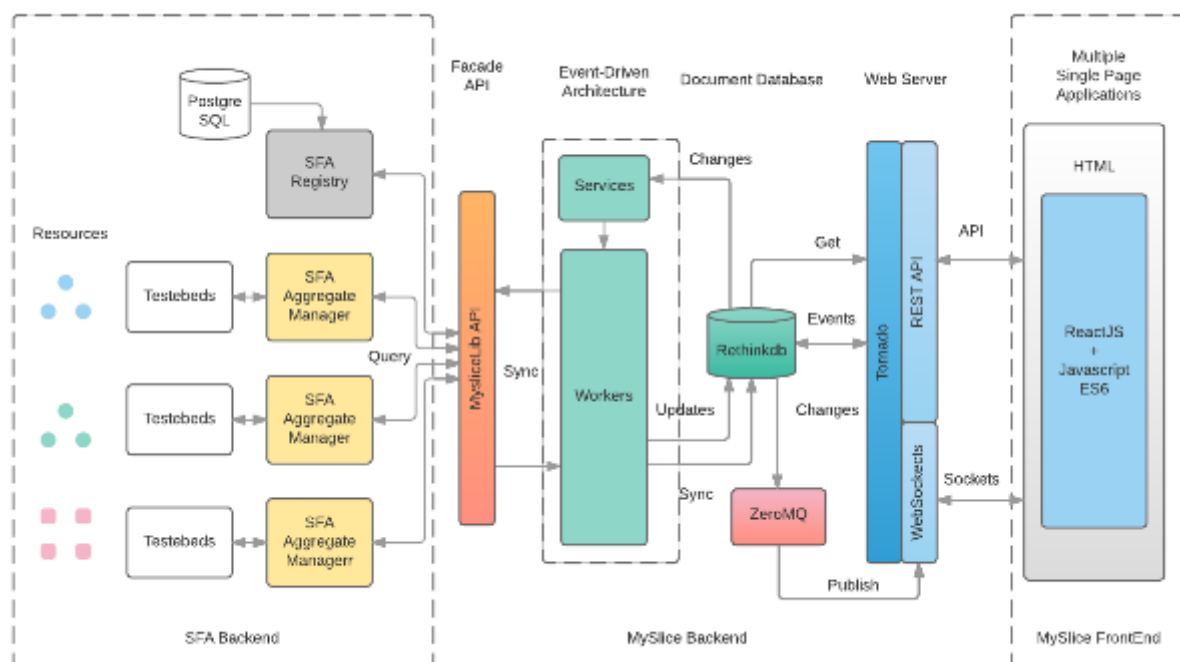


Figure 2: MySlice-Architecture.

frontend interacts with a document database (RethinkDB), which is used as a caching system. It is also used to store data specific to the frontend. This real-time database offers the possibility

D4.02: First TaaS Prototype

to the services of the MySlice API to subscribe to events. As soon as the web frontend updates a record, the services are notified and can trigger events. This architecture allows decoupling the frontend from the complexity of processing results from distributed data sources (AMs, Registry).

This modular architecture allows to be extended on both sides either to support new services or to enhance the web frontend. In the Fed4FIRE+ context it will be extended with matchmaking functionality and allow users to reserve not only Fed4FIRE+ specific physical/virtual resources but also to match specific user requests related to non-technical resources.

MySlice also provides the functionality to register and manage users. The communication with the testbeds is ensured using the SFA AM API. The communication with the Registry relies on the SFA Registry API. This will be extended to support Odoo AuthN module.

In the following subsections we are providing information about the most important MySlice components.

2.1.3.1 SFA Registry

This module provides a registry service as specified by the SFA. It can be queried by the MySlice core as well as by some remote registries and provides information about the SFA objects which this domain is responsible for.

2.1.3.2 MySliceLib

This module supports the interaction of MySlice with multiple Aggregate Managers that are part of a federation of testbeds.

2.1.3.3 MySlice Backend

This module supports all the advanced functionality provided by MySlice, such as, the distributed management of queries, asynchronous handling of the queries, caching of the results.

2.1.3.4 MySlice Frontend

This module is in charge of the frontend that exposes the underlying MySlice functionality to various users/experimenters.

2.1.2 Odoo

The Fed4Fire+ Federation (called “Federation”) requests are:

- Customer Quotation Management.
- Customer Billing Management.
- Customer Helpdesk Management.

A common method to manage a company's interaction with current and potential customers is to use a Customer Relationship Management (CRM) system. According to the Wikipedia: “One

D4.02: First TaaS Prototype

important aspect of the CRM approach is the system of CRM that compile data from a range of different communication channels, including a company's website, telephone, email, live chat, marketing materials, and more recently, social media. Through the CRM approach and the system used to facilitate it, businesses learn more about their target audiences and how to best cater to their needs.”

There are several Opensource CRMs such as

- vTiger (<https://www.vtiger.com>)
- SugarCRM (<https://www.sugarcrm.com>)
- Odoo (<https://www.odoo.com>)

Odoo was the choice within WP4 for three interesting features:

1. **Customer Billing Management and Customer Helpdesk Management APIs.** These APIs make faster and simpler the integration among Odoo, MySlice and Marketplace. For example, a customer (user) can create a request for a quotation from the Fed4FIRE catalogue quickly and easily as an online purchase. Under the hoods, Odoo creates a lead, sends the request to a vendor/Federation (testbed provider) and comes back to the customer with a vendor's message or email. Customers and vendors do not need to install any software or app on their computers.
2. **Multi-company functionality.** Both Federation and vendors can manage billing, accounting and so forth directly. They only need an Odoo account; the software will be the same for all of them.
3. **Online payment services.** PayPal, Stripes and other electronic payment methods are available in order to simplify and speed up the business between the customers and the vendors

Furthermore, Odoo is an all-in-one management software that offers a range of business applications that form a complete suite of enterprise management applications targeting companies of all sizes. Odoo is an all-in-one business software which includes accounting, billing, CRM, website/e-commerce, manufacturing, warehouse and inventory.

2.1.3.1 System Description

Odoo runs on a server hosted by UPMC. The server uses a GNU/Linux Ubuntu distribution as an operating system.

There are two different editions of Odoo: Community and Enterprise versions. The Community is a free and open source version, the Enterprise extends the Community version with commercial features and services. Currently, we use the Community version.

Odoo is a combination of two parts: a frontend and a backend part.

The frontend part is the public one that is, what the other people can see or manage. For example, the homepage, the log-on or the sign-up pages belong to the frontend part. The customers use this part to communicate with the vendors (e.g. testbed providers/Federation) or check their bookings, their bills, their payments etc.

The backend part is reserved only for the vendors and Federation. The backend is a dashboard

D4.02: First TaaS Prototype

where they can manage their own business such as customer messages, requests for quotation (RFQ), invoices etc.

Odoo is basically built and developed in Python language, but it also uses CSS and jQuery languages for a better user interface (UI) and user experience (UX).

The advantage of this software is the module, formerly “add-on”. In this way, it is possible to add and remove features without editing the core of Odoo. Everyone can develop some add-ons or get them from a dedicated place where all add-ons are collected [7]. The add-ons are free or paid.

2.1.3.2 Dataset Description

For our simulation we need three things:

- A customer email address is required for three reasons:
 - for the customer’s Odoo account (frontend).
 - for receiving all Odoo notifications.
 - for a direct communication between the customer and the vendor
- A vendor email address is required for three reasons:
 - for the vendor’s Odoo account (backend).
 - for receiving all Odoo notifications.
 - for a direct communication between the customer and the vendor.
- A list of vendors’ services and products:
 - Marketplace/MySlice (market.fed4fire.eu) and Odoo (crm.fed4fire.eu) are synchronized through Odoo’s native API (see External API section for further details). So, when a service or a product is created, updated or deleted on Odoo, it will be created, updated or deleted on Marketplace/MySlice automatically.

2.1.3 Odoo Implementation Details

2.1.3.1 Details of the Simulation Environment

Odoo is not a standalone software like Office suite, but it needs to be used as a website. Indeed, there is a log-on page at the URL: crm.fed4fire.eu. After logging in, both the customers and the vendors can access their own accounts and start their own business. Therefore, no specific software, tools or devices are not required, just a computer with an OS (Windows, GNU/Linux, Apple etc.), a browser (Firefox, Chrome, Internet Explorer etc.) and a network connection

2.1.3.2 Integration

The integration among Odoo, Marketplace and MySlice is provided by:

External API

Odoo provides its native Application Programming Interface (API). Odoo documentation gives a good definition of that:

D4.02: First TaaS Prototype

“Odoo is usually extended internally via modules, but many of its features and all of its data are also available from the outside for external analysis or integration with various tools. Part of the Model Reference [8] API is easily available over XML-RPC [9] and accessible in a variety of languages (Python, Ruby, PHP, Java – author’s note).”

It is not the goal of this document to explain the API in detail, but it is worth saying that:

- It is possible to use it in a block of code or through a command line interface (CLI)
- It is accessible in a variety of languages and they are independent of the OS or the browser, so the compatibility with other systems is granted.
- The response of the API is compatible with almost all languages/software because the API uses a JSON data format that grants universal compatibility.

Opportunistic Odoo Settings

Odoo can be set to send an email notification to an associated email address when Odoo receives an email for a given account. It is also possible to associate a different email address with different email accounts. In our case, we use the first scenario. So, now we briefly describe how to do that.

The first step is to set the Incoming Email Server with all required parameters (Figure 3).

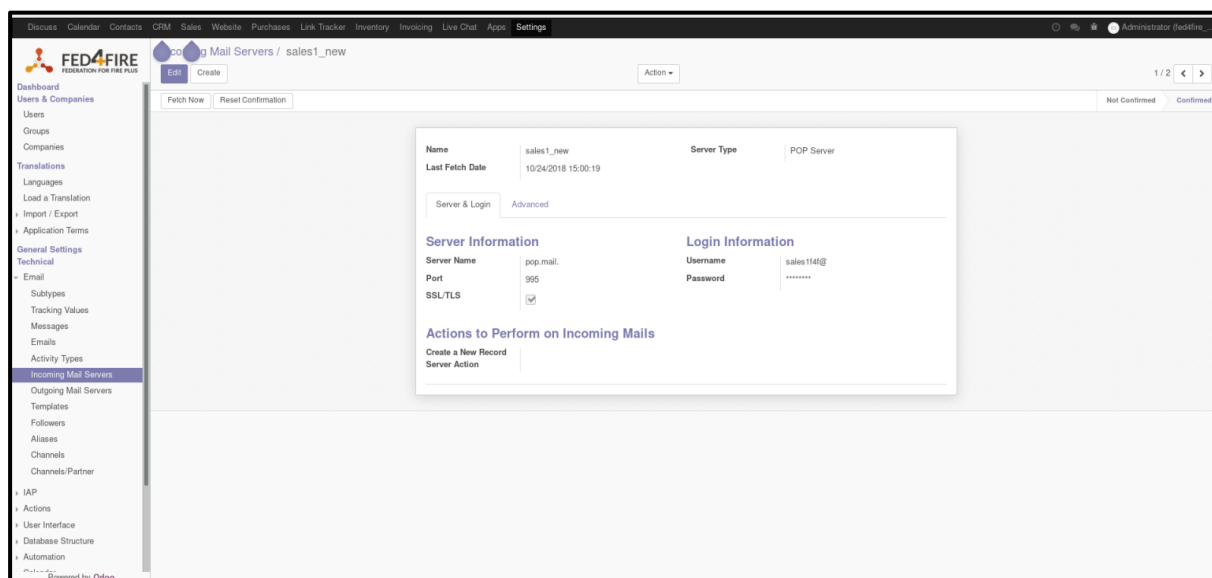


Figure 3: Configuration of an Incoming Mail Server on Odoo.

The second step is to associate the vendor’s email address with the email address of the incoming email server (Figure 4).

D4.02: First TaaS Prototype

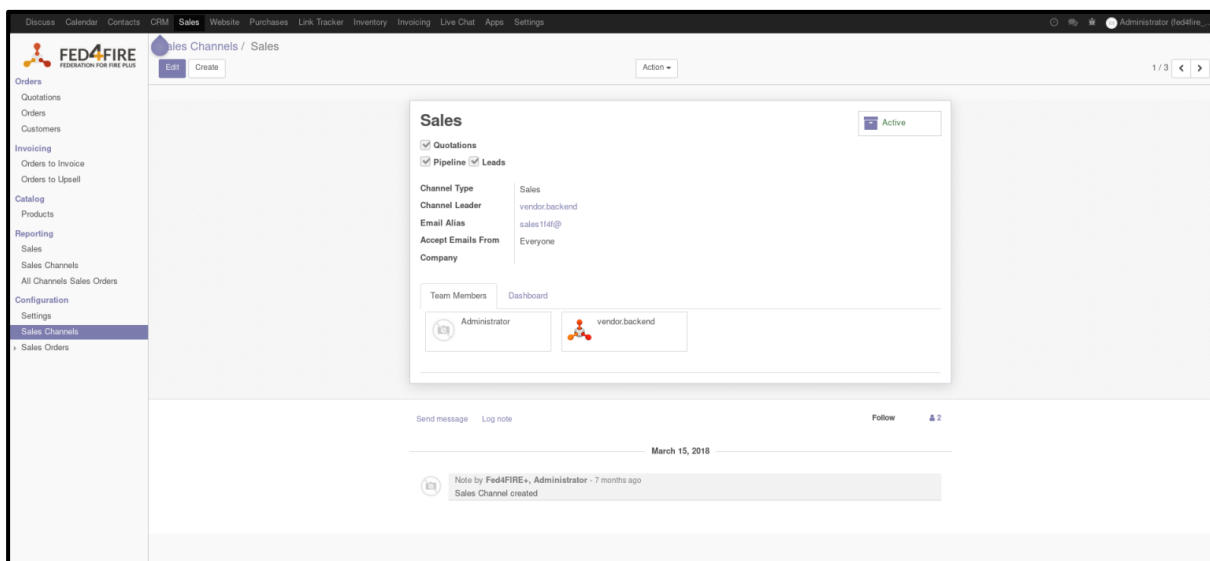


Figure 4: Configuration of Sales Channel.

The third step is optional, we can add other followers, that is other people (contacts) stored in Odoo, regardless whether they are customers, vendors etc.

2.2 MARKETPLACE

In comparison to the other components presented, which are reused, the Marketplace component was newly developed for this project. The Marketplace component offers on-demand booking for the customers. Currently it offers two features. The first one is to enable testbed providers to register themselves and their services and the second one is to enable customers to search and book services.

The Marketplace component is developed as a plug-in for MySlice, because it also uses the framework React for the frontend part and therefore it is integrated with MySlice. Figure 5 shows the current version of the Marketplace component integrated in MySlice.



Figure 5: MySlice integration with the Marketplace as a plug-in.

Currently, the Marketplace component only offers the features to search for services on the

D4.02: First TaaS Prototype

start page. Further components will be added to the next prototype (see Next Steps section). FireRabbit is the current internal project name but will be renamed to Marketplace for the next prototype.

For the design of the pages, we use Google's Material Design [3]. Material-UI [2] implements this design for React components and is available as open source software. That is the reason why it is also used in the development of the Marketplace plug-in.

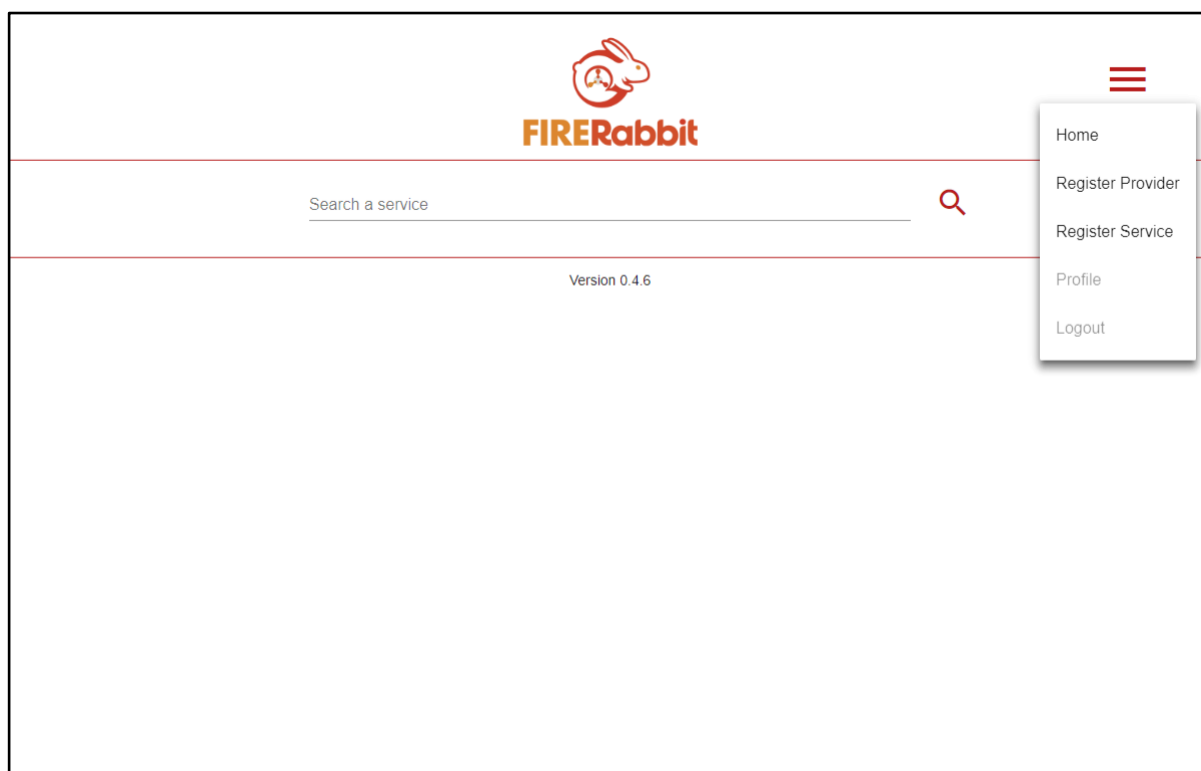


Figure 6: Current Start Page of the Marketplace Component.

As already mentioned, the Marketplace plug-in was developed with the React framework. React is a JavaScript framework developed by the Facebook's developers to build user interfaces. It provides a template language that allows you to build simple views that React efficiently updates as changes occur.

The framework allows the developers to define the remaining technology stack themselves, as it does not prescribe any dependencies in this respect. Looking at the MVC model, this means that React provides the view part but not the model or controller. For the missing components, a pattern is used that also comes from the Facebook's developers. It is called Flux [5] and does not follow the MVC model to allow unidirectional data flow.

Figure 7 shows the structure and the data flow that run through the components.

D4.02: First TaaS Prototype

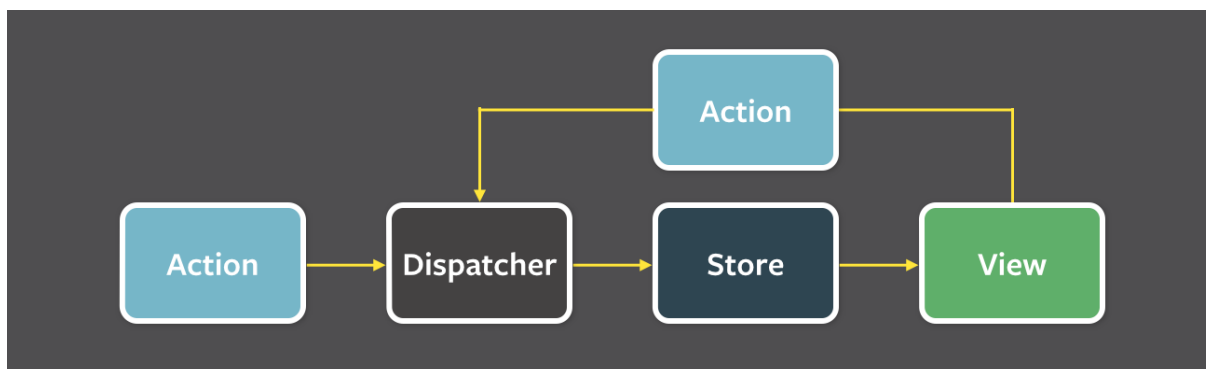


Figure 7: Structure and Data Flow in Flux [5].

Unidirectional data flow means that when a user interacts with a view, it transmits an action to the store through a central dispatcher. Stores hold data of the application and update the corresponding views in the event of a change.

The structure for the Marketplace plug-in is as follows:

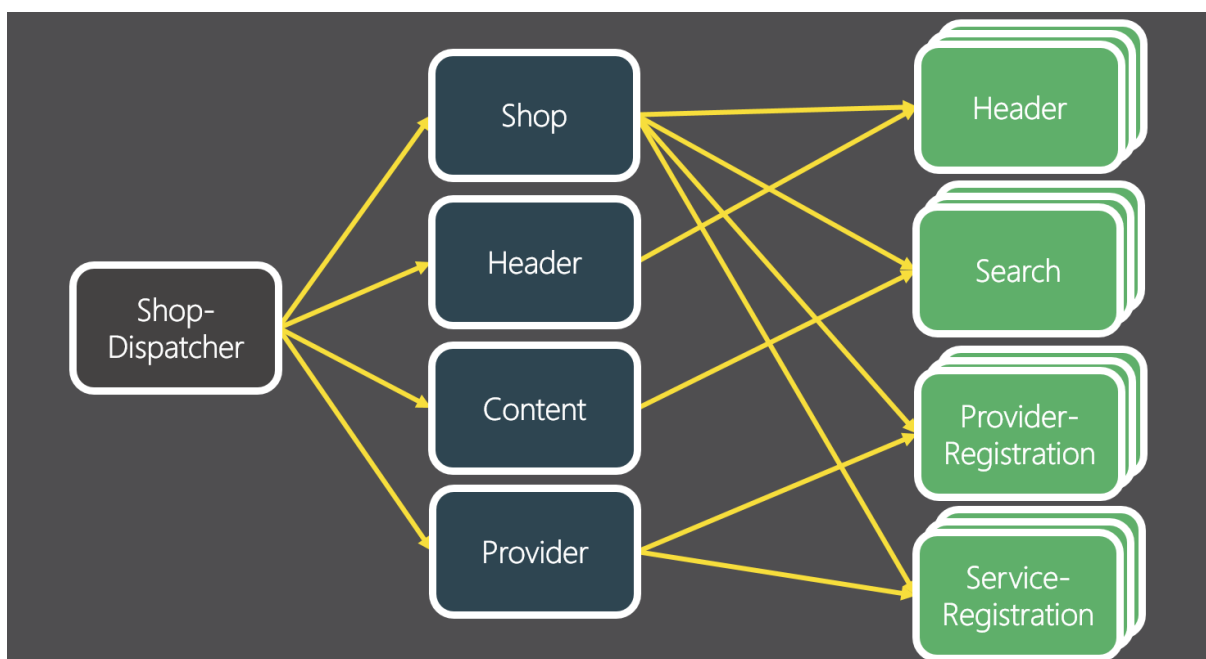


Figure 8: Structure and Data Flow of the Marketplace Plug-in.

The colour coding corresponds to the previous figure. The actions have been removed from the image to keep good readability. Since each flux-based application should only have one dispatcher, it also has only the shop dispatcher that forwards all actions to the stores. There are currently four stores that manage the states of different parts of the application. The shop store is an exception, as it manages data that affects the entire application. For example, the current theme or loaded configuration file.

If the theme changes, all views are notified and re-rendered. The other stores are limited to

D4.02: First TaaS Prototype

individual parts, such as registering the testbed providers and their services. As with the stores, there are several views for one area. The header view contains the logo and the menu from which the other pages can be accessed. Search view contains the text field for the search and all subsequent components, such as the list of results and the summary of the booking.

2.3 ARCHITECTURE DESCRIPTION

After the subcomponents of the prototype have been presented, the architecture of the three components is shown in this section.

Figure 9 shows how the different user groups interact with the components and how the components themselves interact with one other. The Marketplace plug-in is part of the MySlice component and Fed4Fire+ Offerings represent the Knowledge Graph.

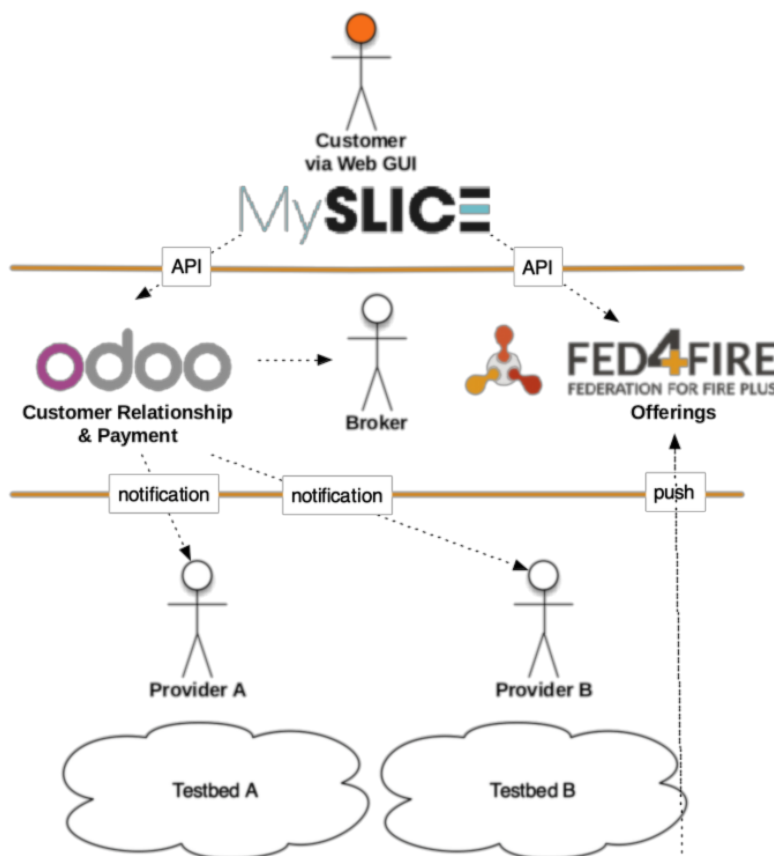


Figure 9: Architecture with all Components.

2.4 DESCRIPTION OF MATCHMAKING CAPABILITIES

For the first prototype, a hybrid version of a Semantic Web search was chosen. This approach is presented in the paper "A Hybrid Approach for Searching in the Semantic Web" [4].

The idea is that the user starts a search with keywords, whereupon the matchmaking system searches for these terms in the knowledge graph, which is a knowledge base, a store, first using a traditional search engine, and then expands the list of results with a technique called

D4.02: First TaaS Prototype

"Spread Activation". Figure 10 shows the architecture of this approach.

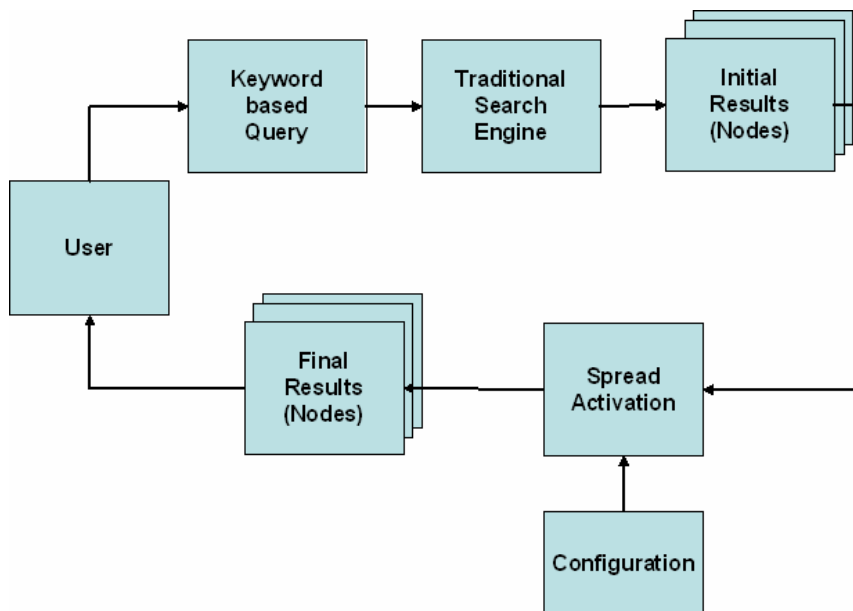


Figure 10: Hybrid Approach for Searching in the Semantic Web [4]

The flow starts on the left side of the user. As you can see, the user first enters his keywords as a query. Using a traditional search engine, all entries related to these terms are listed in the knowledge graph. Before the results are forwarded to the user, Spread Activation is used to search the Knowledge again for results that are related to the first results list and exceed a certain coverage level for the search terms.

Spread activation works by linking individual entries in the descriptions of the testbed services and adding weights to these links. An algorithm for this technology works through the knowledge graph with the results already found and selects all that activates the search function in the knowledge graph.

Finally, the original and newly found hits are delivered or shown to the user. This approach is thus implemented in the prototypes.

3 MANUAL

This chapter describes the operation of the prototype. The operation is separated into five steps, based on the activities a customer goes through to book services.

1. **Service Publication** – It allows testbed providers to register themselves and their services.
2. **Service Selection** – It provides customers the opportunity to query services and to select them out of a list of best matches.
3. **Lead Creation (Feedback)** – it notifies testbed providers about requests and allows both sides (customer and vendor) to communicate with one other.
4. **Execution (Bilateral incl. Legal and Financial Aspects)** – It includes the negotiation between the customer and the testbed provider and the service execution.
5. **Conclusion** – It covers the invoice creation and the payment.

The following sections describe these steps in detail and with examples. This includes how the different components are communicating with one other and how the customers and testbed providers can interact with the current version of the prototype.

3.1 SERVICE PUBLICATION

If the testbed providers decide to offer their services to the customers via Fed4Fire, they must first register themselves and then register their services. Currently, some information are requested such as the service name, the type of company and the contact person of the company. For the registration of services, a description, an image and the type of service must also be specified.

The information is stored in the Knowledge Graph after the registration, so that it can be considered in the matchmaking system. In addition, they are saved in Odoon in order to manage easily the administrative and financial aspects.

Figure 11 shows the process for registering and updating all information and which components are involved in this process.

D4.02: First TaaS Prototype

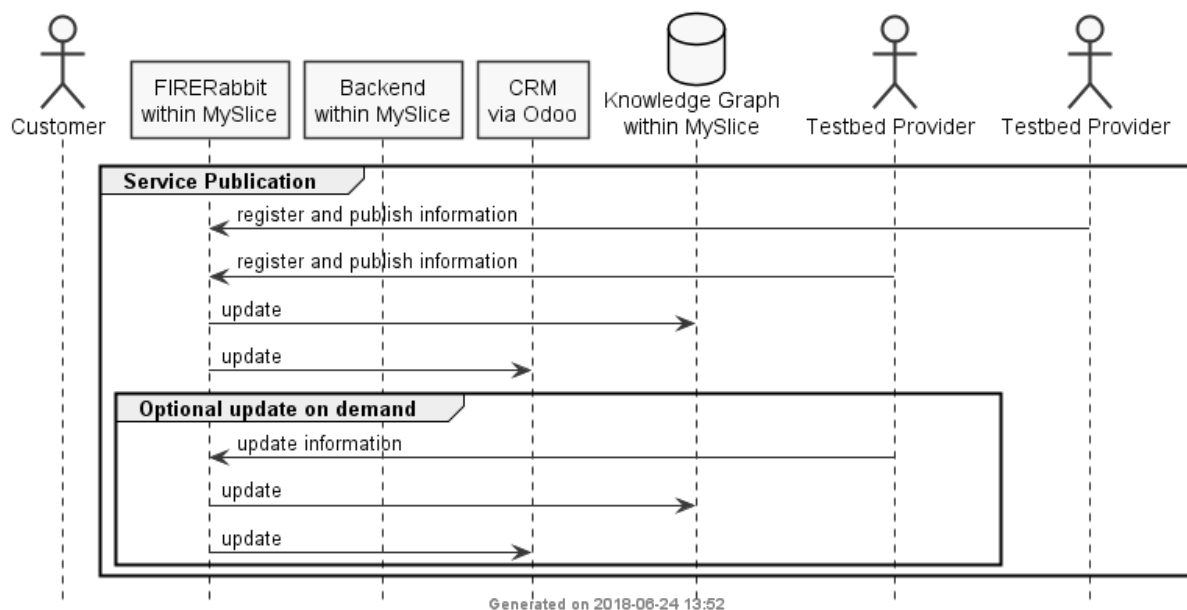


Figure 11: Overview of Activities for Service Publication.

As can be seen in the figure, the testbed providers use the Marketplace component to register. The reason for that is related to two problems that arise due to the architecture and the technologies used. About the first problem, all information about testbed providers and services must be stored in two locations. All of that is difficult to manage for the testbed providers.

The Marketplace solves that by offering a central point for registration and then forwarding the information to the required points itself. The same applies to updates of the information.

The second problem comes from the use of ontologies for the presentation of information. Testbed providers would need to know how information is presented using ontologies and how our ontologies are structured. However, this would be an additional barrier to offering the services of the testbed providers. That is also solved by the Marketplace component by using a registration form. The information is then described by ontologies from the Marketplace and forwarded to the system.

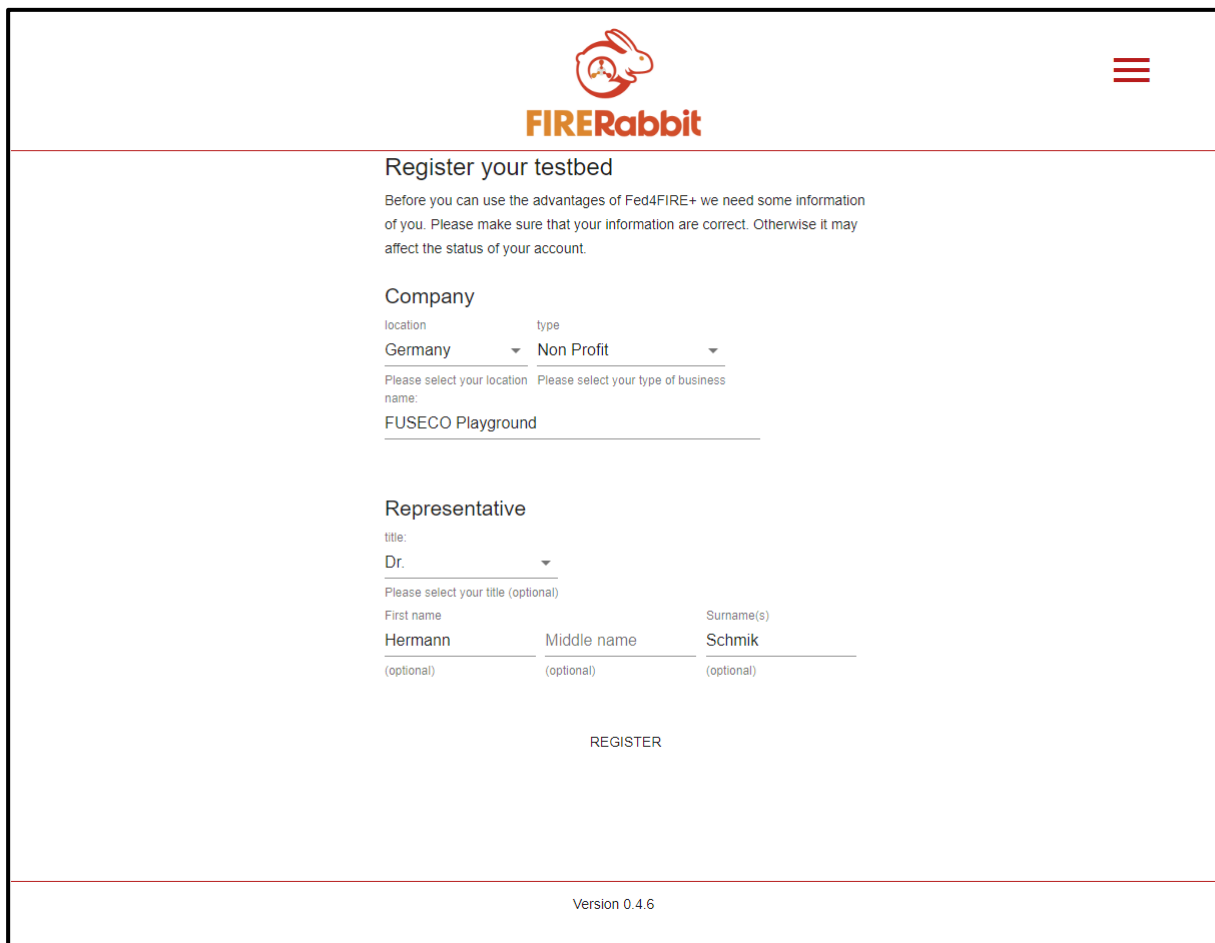
3.1.1 Register the Testbed Provider

The following example shows how to register the testbed provider and its services in the current prototype. The scenario is as follows:

"The Fraunhofer Institute FOKUS would like to register the FUSECO Playground Testbed. Dr. Hermann Schmik will serve as contact person. To begin, they register the services *OPC UA Compliance Test* and *IIoT Consultant Alexander Willner*."

In the first step, the testbed provider will be registered. To do this, go to the Register Provider page from the menu previously presented in Figure 6.

D4.02: First TaaS Prototype



FIRERabbit

Register your testbed

Before you can use the advantages of Fed4FIRE+ we need some information of you. Please make sure that your information are correct. Otherwise it may affect the status of your account.

Company

location type
Germany Non Profit

Please select your location Please select your type of business

name:
FUSECO Playground

Representative

title:
Dr.

Please select your title (optional)

First name Middle name Surname(s)
Hermann Schmik

(optional) (optional) (optional)

REGISTER

Version 0.4.6

Figure 12: Registration of the FUSECO Playground Testbeds

Figure 12 shows the page with the information from the example above. After entering information and clicking on the register button, information is distributed in the prototype.

3.1.2 Register the Services

In the second step, the services are registered. To do this, go to the *Register Service* page.

D4.02: First TaaS Prototype



FIRERabbit

Register your services

+ OPC UA Compliance Test ▼

+ IIoT consultant Alexander Willner ▼

+ ADD SERVICE

REGISTER

Version 0.4.6

Figure 13: Registering the Services of FUSECO Playground

As can be seen in Figure 13, the service *OPC UA Compliance Test* and *IIoT Consultant Alexander Willner* were entered for this testbed. information is integrated into the system after clicking on Register, as with registering the testbed itself.

With these two steps, both the FUSECO Playground Testbed and its two services are added to the system and can be found and booked via the search engine using proper matchmaking, which leads to the next step in the manual.

As can be seen in Figure 14, we can add new services. For example, from the above Figure 13, OPC UA Compliance test is an application/service whereas, **IIoT Consultant Alexander Willner** is both a consultant/training provider and a service. So, If the testbed provider wants to introduce a new service, they can add it by filling the form and simply clicking **ADD SERVICE**. A new service added to the One-Stop-Shop can be registered later. Therefore, the service will be also available in the search summary.

D4.02: First TaaS Prototype

Register your services

⊕ OPC UA Compliance Test
▼

⊕ IIoT Consultant Alexander Willner
▼

⊕ Add a Service
▲

Service Title

Service Title
(optional)

Consultant Title ▼
(optional)

First name <small>(optional)</small>	Middle name <small>(optional)</small>	Surname(s)/Last name <small>(optional)</small>
<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Upload Image"/>	

Service Description

Service Description
(optional)

Else upload your description here:

Service Pricing

Price <small>(optional)</small>	Currency ▼ <small>Please select currency</small>
------------------------------------	--

ADD SERVICE

Figure 14: Creating/Adding a new service.

3.2 SERVICE SELECTION

In the second step, the Service Selection, the customer can use the search field to find services that meet their needs. For the customer to receive results that are related to his query, it must be processed correctly in the backend. In this case, processing is handled by the matchmaking system. The plug-in searches for all entries that match the query into the Knowledge Graph

D4.02: First TaaS Prototype

store and returns them to Marketplace. Figure 15 shows these steps:

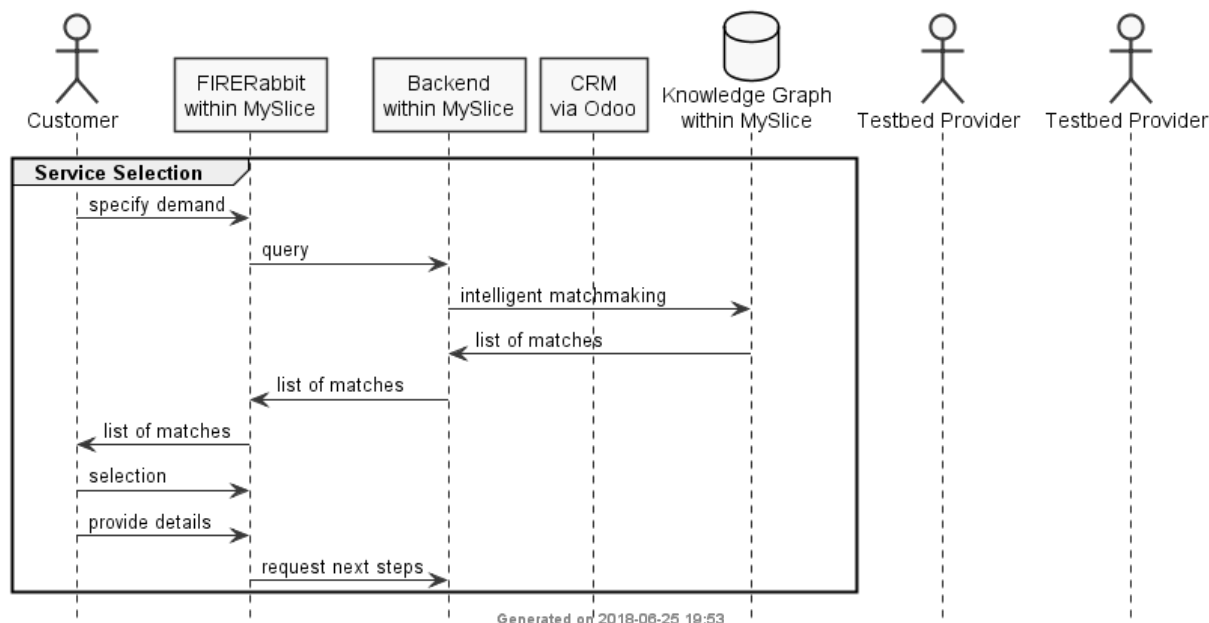


Figure 15: Overview of Activities for Service Publication.

After the matchmaking system has sent the list of matches to Marketplace, the latter processes the list into a human-readable format and displays it in a list. The customer can select the desired services from this list. After the selection, the customer is asked to provide additional information for his booking. This includes his idea or project and his details for any further questions. The reason why the idea should be included is that it allows the selected testbed provider to see what the customer is planning and, if necessary, to give an advice, such as selecting another service. Once information has been entered and the selection posted, information is forwarded to the backend.

3.2.1 Query Demand

As with the Service Registration, an example shows how the steps for the Service Selection in the current prototype look like. Since the focus here is on the customer, the scenario already described is extended as follows:

"Smart and Medium Enterprise Solutions Inc. wants to launch its new TSN network switch on the IIoT market. But before they do that, they want to perform tests with the switch and consult help to find the most suitable customer market."

D4.02: First TaaS Prototype

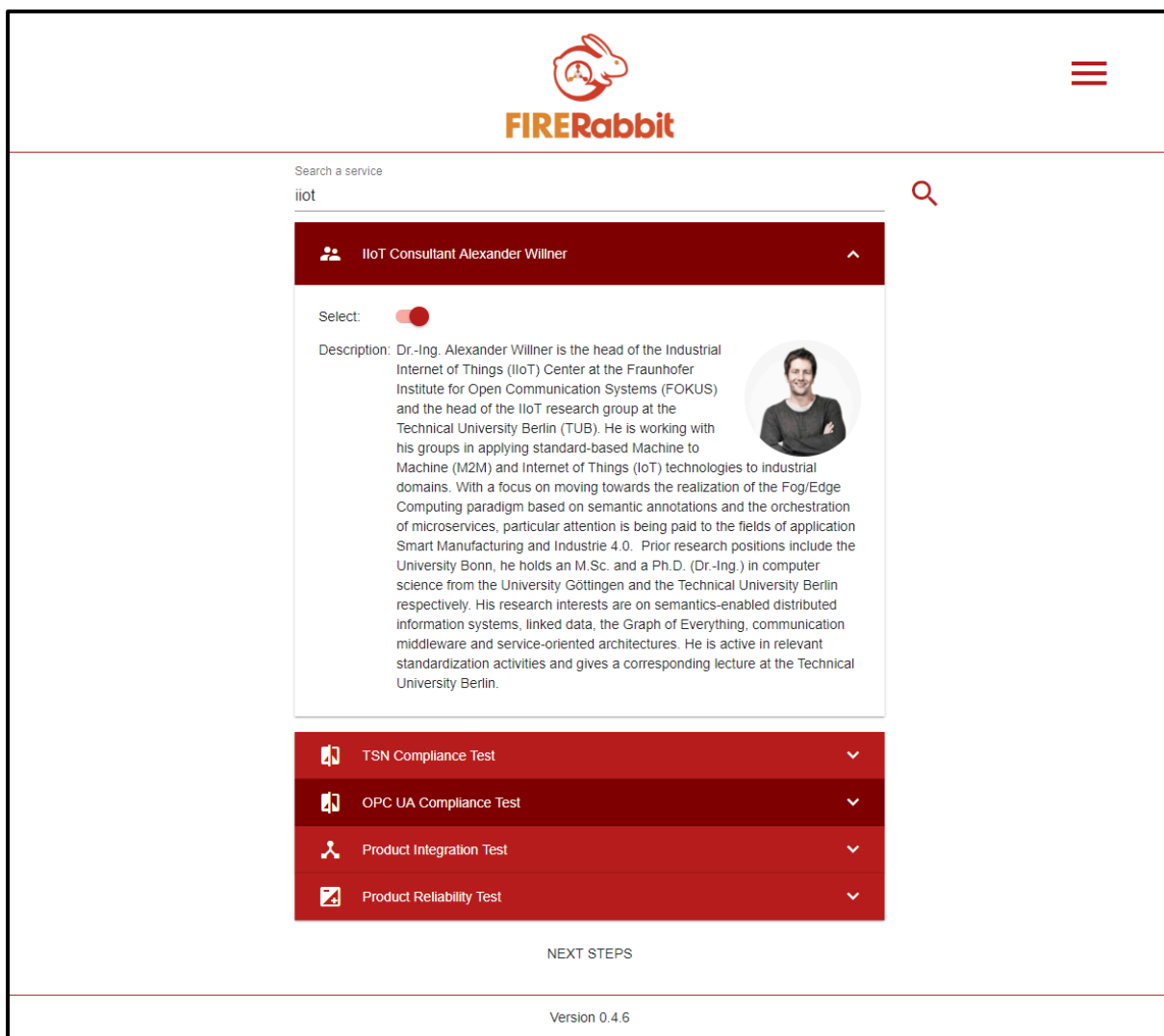


Figure 16: Example of the Service Selection.

First, the company calls up the Marketplace and enters "IIoT" as the search term. The matchmaking system finds five services for the search term that match the term. Figure 16 shows the list with the appropriate services and the company selection.

The figure summarizes the search and selection, as they differ slightly. As can be seen in the figure, three other services have been listed in addition to the FUSECO Playground services that match the search term. As can be seen in the first service, each service has a description and picture in order to describe what its features are.

Another matchmaking capability of the system is the possibility for the user to search only for just a specific service, for example, he/she can write 'test' in the search field and can find all the services that provides only testing facilities. Figure 17 shows a list of testing services offered by the One-Stop-Shop.

D4.02: First TaaS Prototype

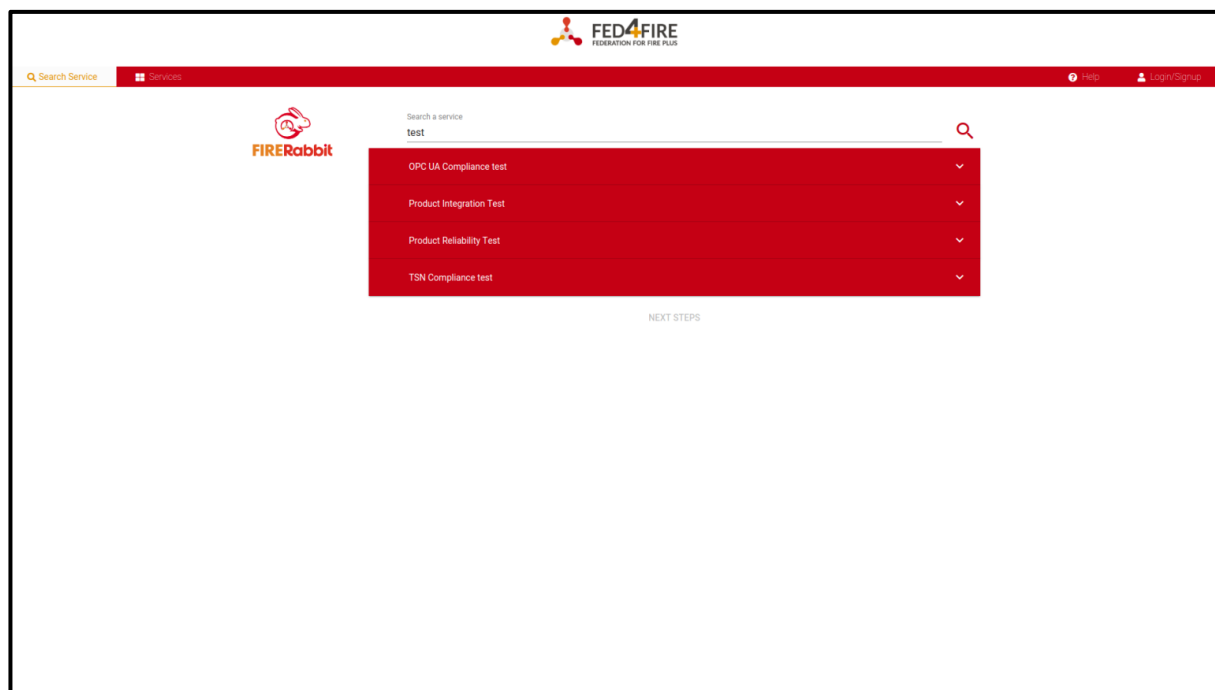


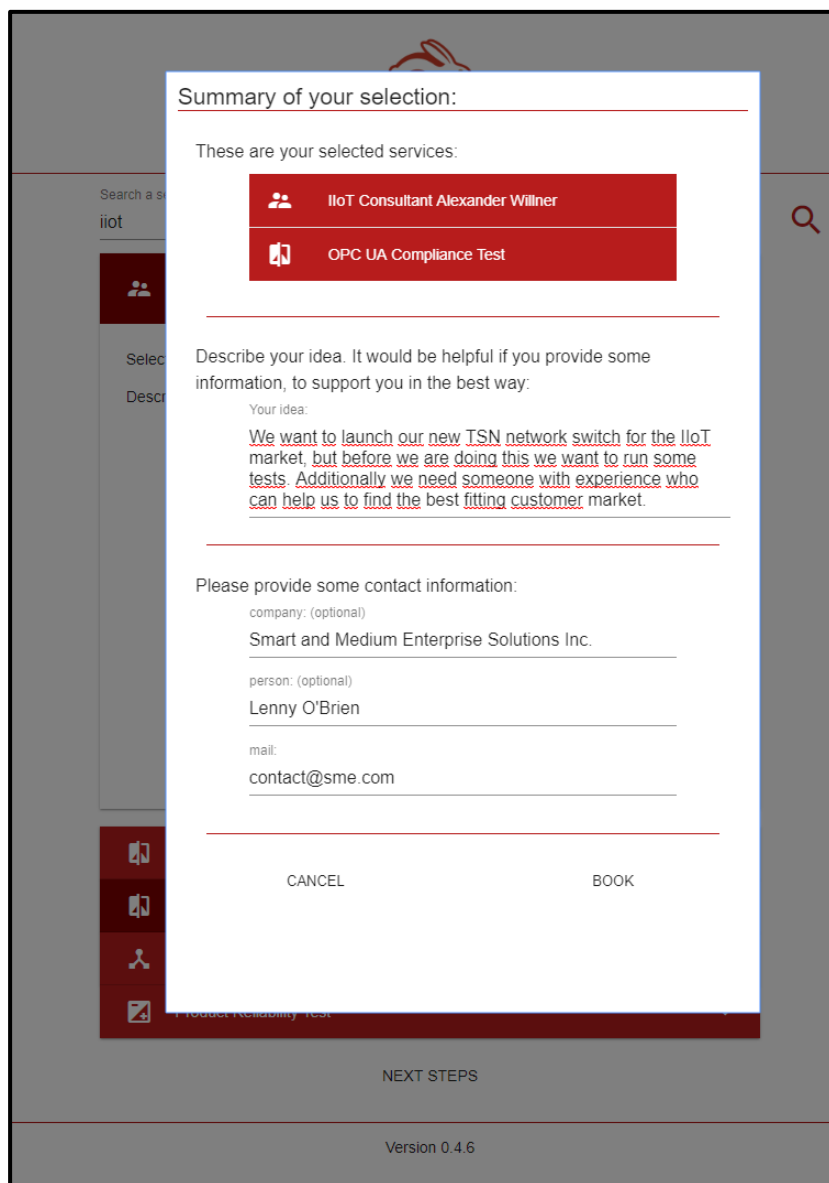
Figure 17: Searching services for testing by using matchmaking.

3.2.2 Selection Summary

After selecting the desired services and clicking on Next Steps button, a summary of the selected services is shown. The customer has the possibility to add some additional information to the specific fields under the review section.

Figure 18 shows this summary with some additional information, ready to be sent to the testbed providers.

D4.02: First TaaS Prototype



Summary of your selection:

These are your selected services:

- IIoT Consultant Alexander Willner
- OPC UA Compliance Test

Describe your idea. It would be helpful if you provide some information, to support you in the best way:

Your idea:

We want to launch our new TSN network switch for the IIoT market, but before we are doing this we want to run some tests. Additionally we need someone with experience who can help us to find the best fitting customer market.

Please provide some contact information:

company: (optional)
Smart and Medium Enterprise Solutions Inc.

person: (optional)
Lenny O'Brien

mail:
contact@sme.com

CANCEL BOOK

NEXT STEPS

Version 0.4.6

Figure 18: Request further Information after the Selection Phase.

Moreover, Lenny O'Brien was given as the person to call, so that the company can be contacted as necessary. After clicking on the Book button, the selection with the information is sent to the system, which will send a notification to the testbed providers.

3.3 LEAD CREATION

This last step completes the Service selection. The following activities are described in Lead Creation section.

D4.02: First TaaS Prototype

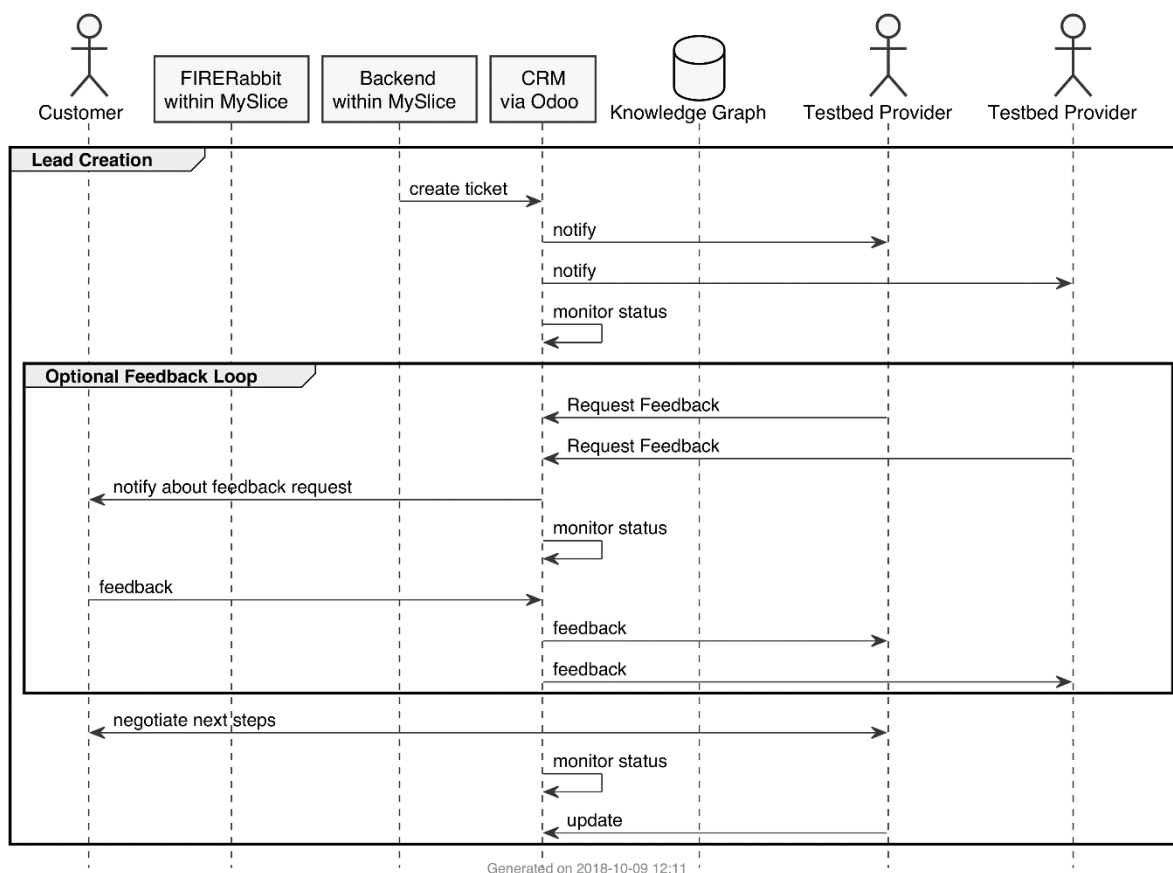


Figure 19: Overview of Activities for Lead Creation.

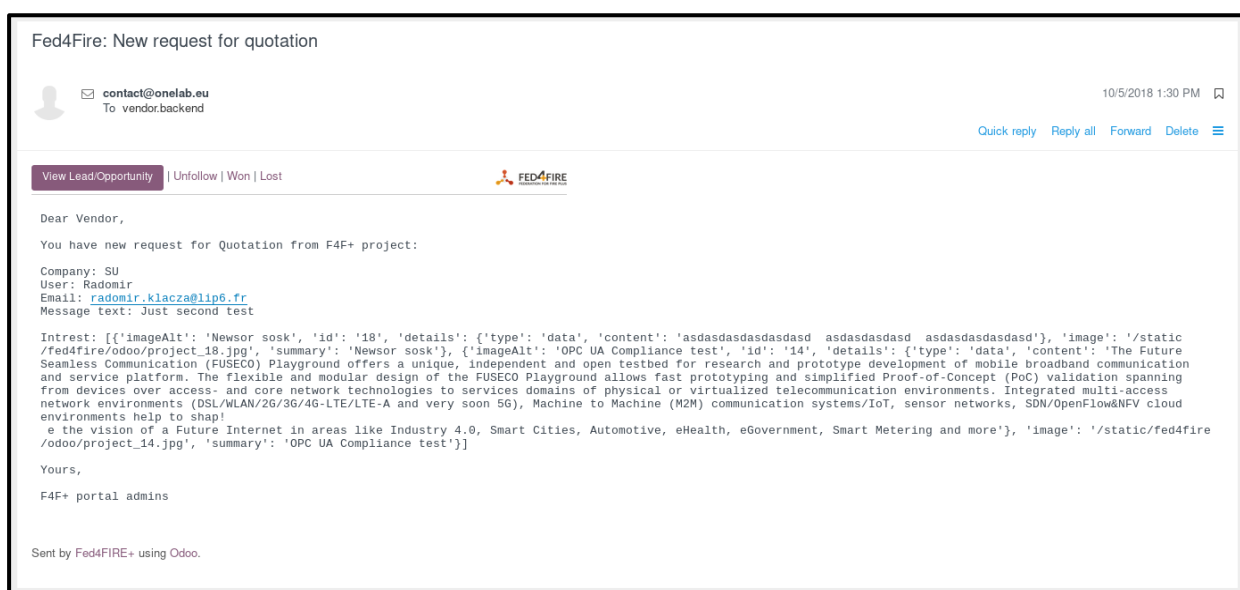


Figure 20: Vendor's Odoo notification email.

A customer books a service on market.fed4fire.eu. The vendor receives an Odoo notification via email. He clicks on the “View Lead/Opportunity” button (Figure 20) and sign-in to Odoo system (Figure 21).

D4.02: First TaaS Prototype

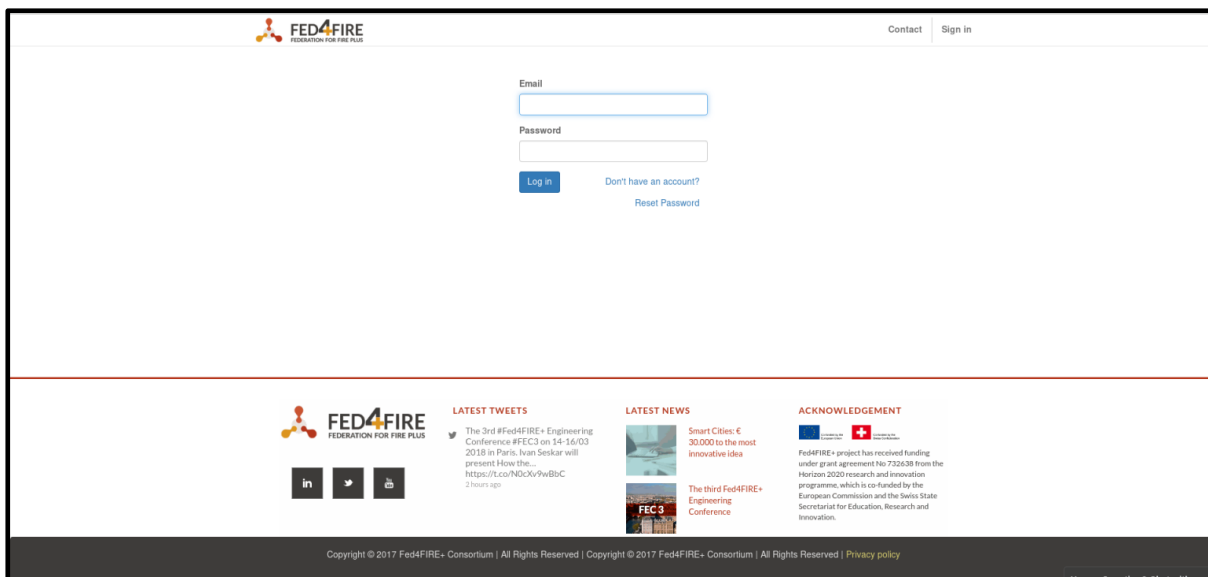


Figure 21: Odoo Log-in Page.

Figure 22 shows a page where the vendor can send a message to the customer (Send message button) or create an offer (New Quotation button).

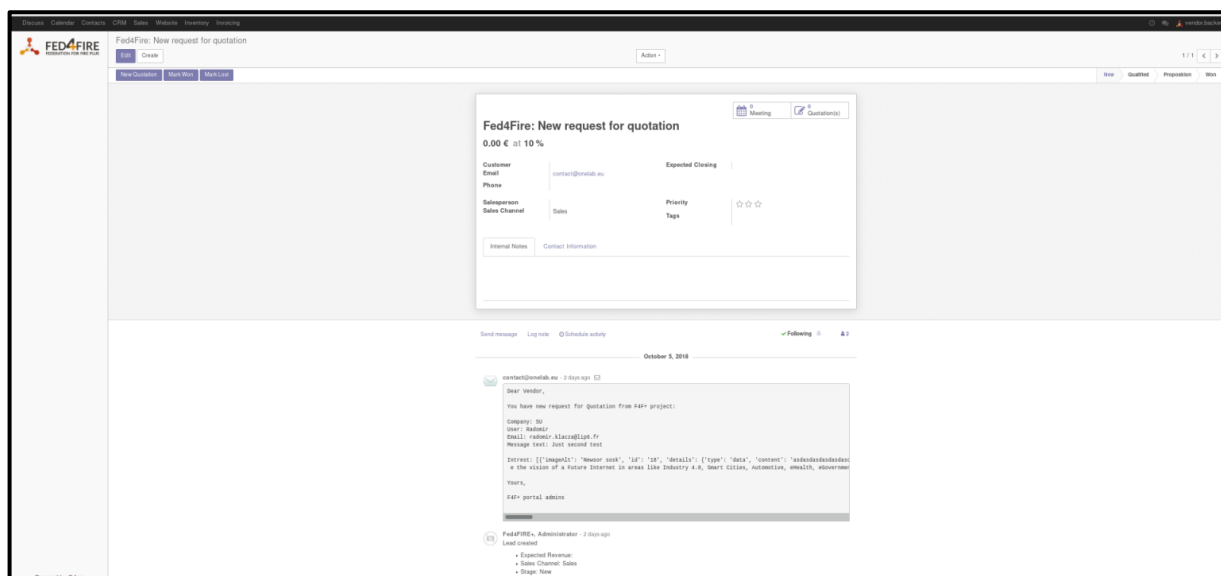


Figure 22: Odoo Backend – Vendor's RFQ view.

D4.02: First TaaS Prototype

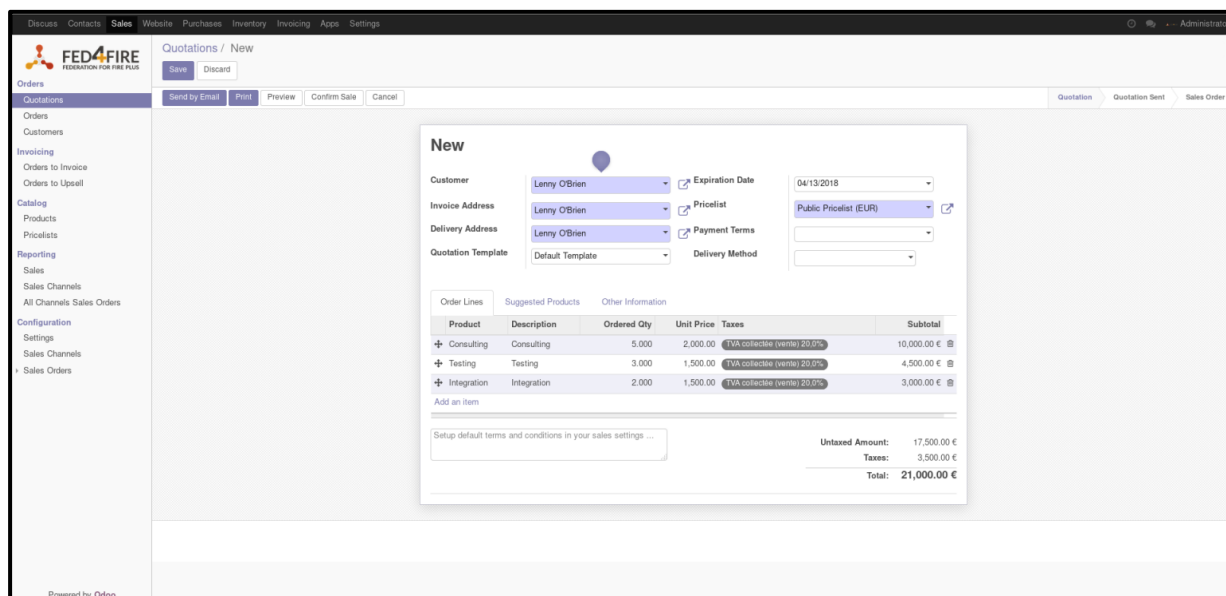


Figure 23: Odoo Backend – Vendor’s New Quotation View.

Create a new quotation and fill in the form (Figure 23). In the case of a new customer, click on the drop-down list and select the “create and edit” option (Figure 24) and add a new contact to the Odoo system (Figure 25).

D4.02: First TaaS Prototype

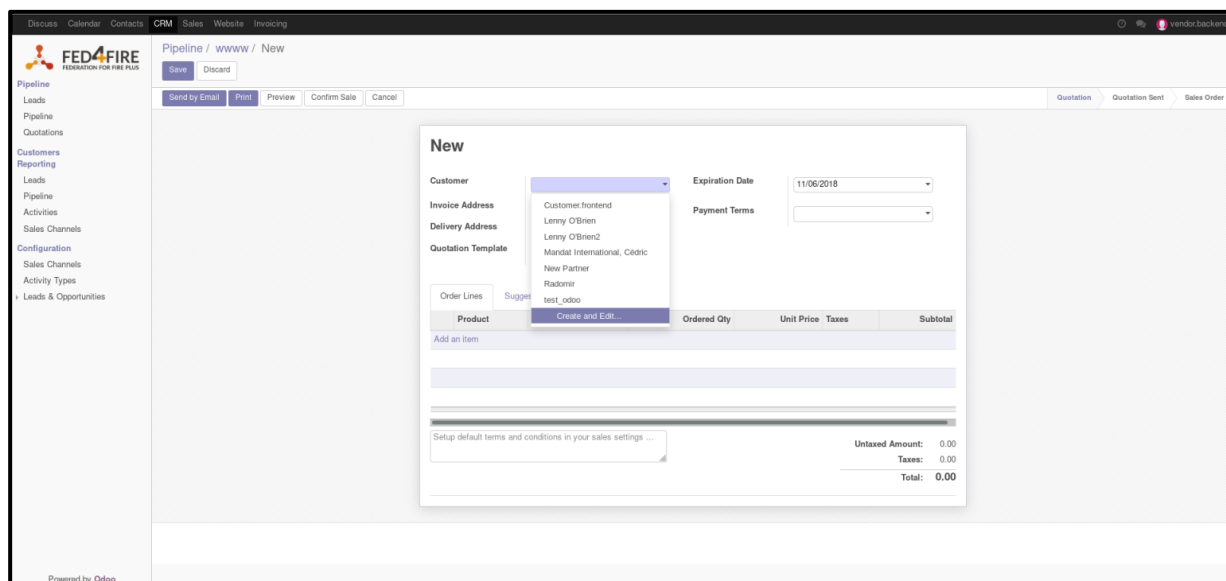


Figure 25: Odoo Backend – Vendor’s new Quotation View.

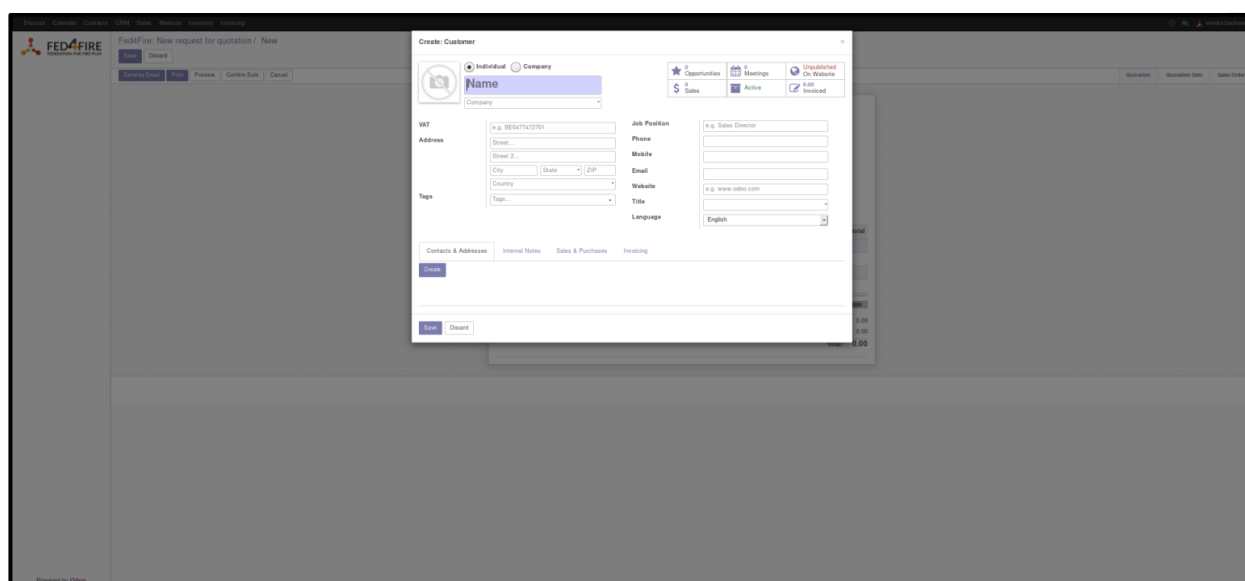


Figure 24: Odoo Backend – Vendor’s new Customer Contact Form.

After a click on the “Send by email” button, the quotation will be saved automatically in the Odoo system (Figure 26).

D4.02: First TaaS Prototype

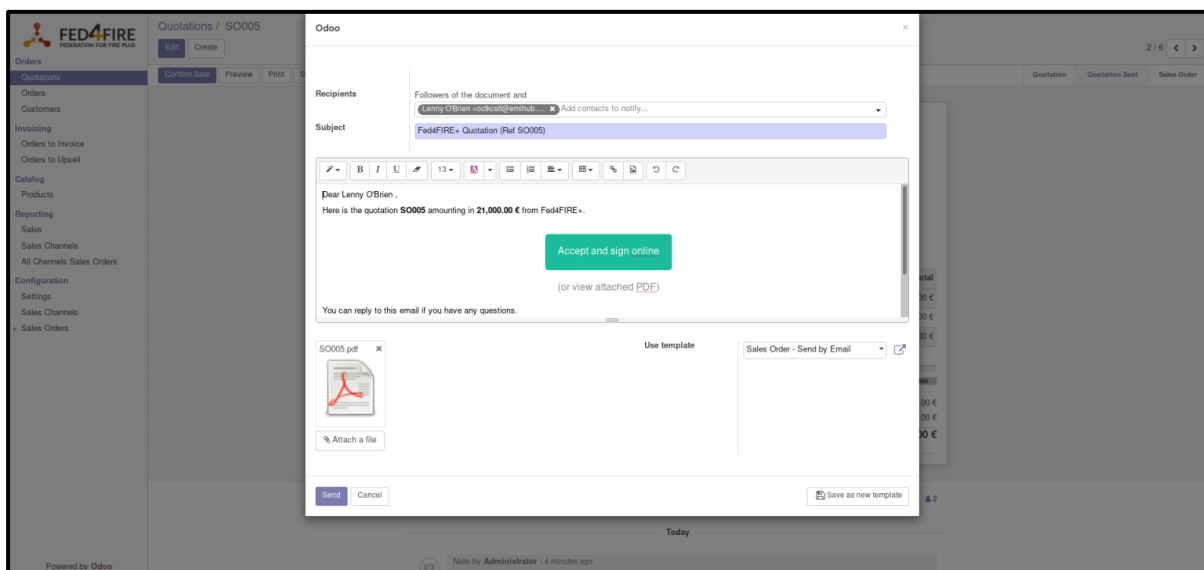


Figure 26: Odoo Backend – Vendor's Email View.

All quotations are stored in the Odoo system, so both the customer and the vendor can view them at any time (Figure 27).

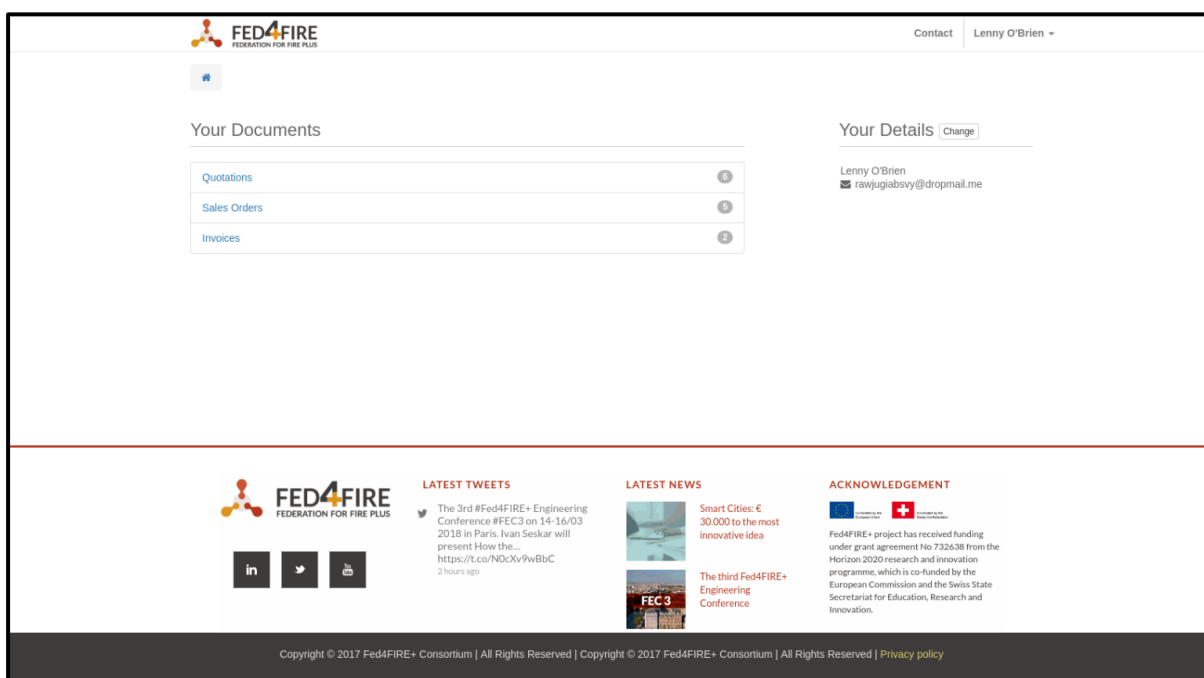


Figure 27: Odoo Frontend – Customer's Welcome Page.

3.4 EXECUTION

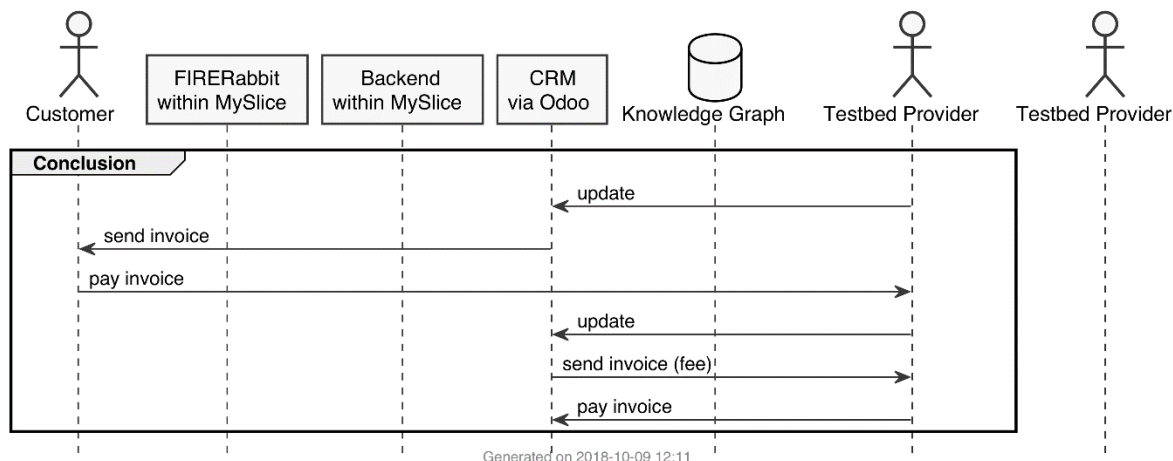


Figure 28: Overview of Activities for Execution.

After the request for quotation phase, the next steps are:

- Feedback between the customer and the vendor.
- The customer accepts the quotation.
- The quotation becomes a sale order.

3.4.1 Feedback between Customer and Vendor

In the phase, customer and vendor/Federation can exchange messages, emails and can modify the quotation about the type of services, quantity and costs. All is stored in Odoo system, so both documents and messages can be read, collected and shared with other people and printed anytime (Figure 29).

D4.02: First TaaS Prototype

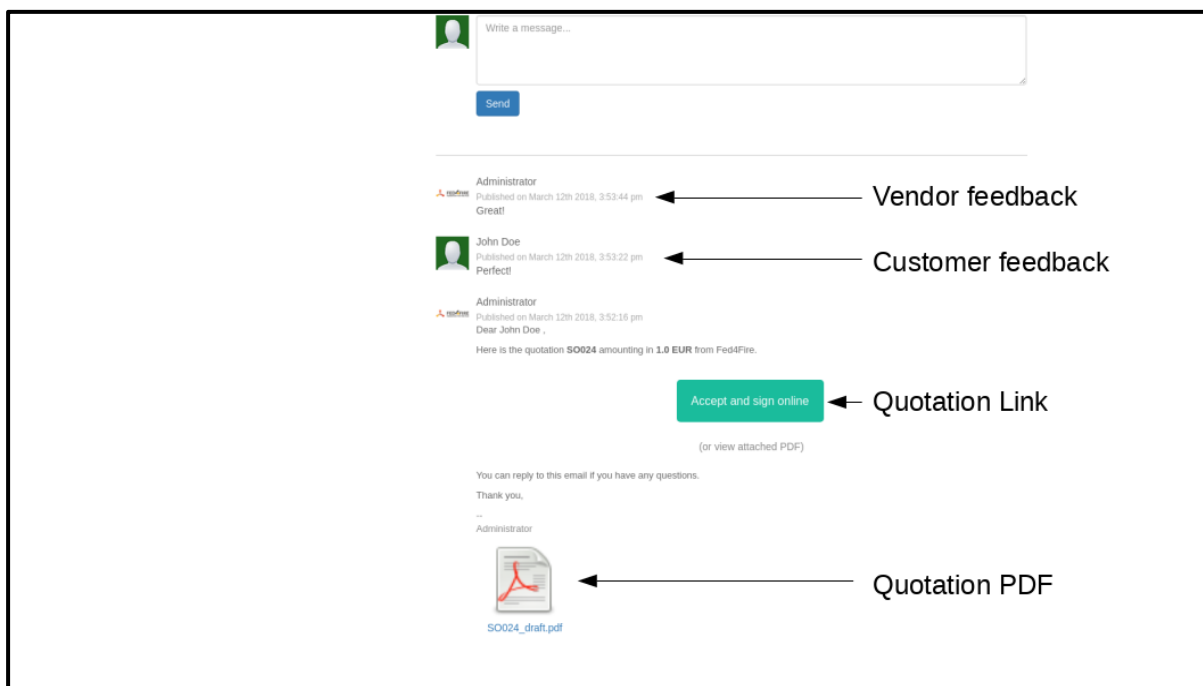


Figure 29: Odoo Frontend – Email & Message Section.

3.4.2 Customer Accepts the Quotation

When a customer accepts a quotation, it becomes a sales order. The customer can confirm his purchase in many ways such as wired payment, digital signature and so on so forth. The most common method is through an online payment. The latter is the quickest and the most secure method for both customer and vendor/Federation (Figure 30).

D4.02: First TaaS Prototype

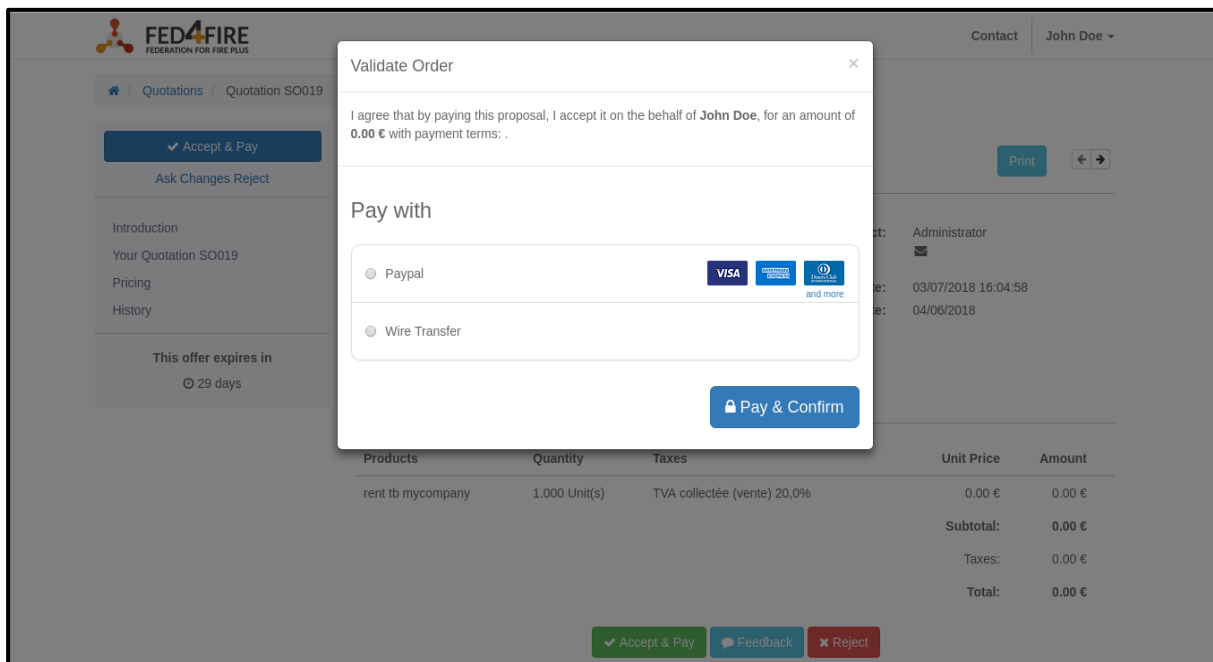


Figure 30: Odoo Frontend - Payment Confirmation View.

3.5 CONCLUSION

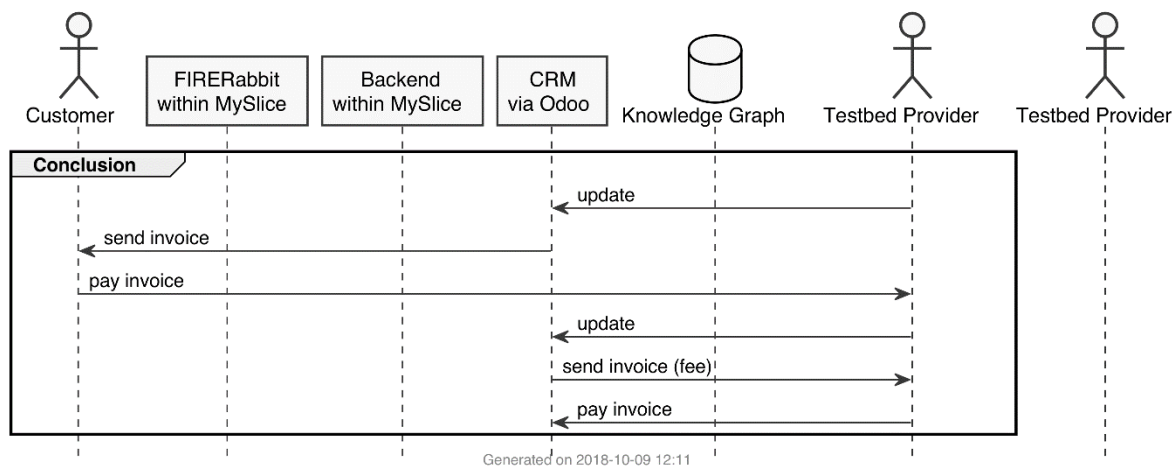


Figure 31: Overview of Activities for Conclusion.

3.5.1 The Sales Order Becomes an Invoice

A sales order in Figure 32 is ready to become an invoice.

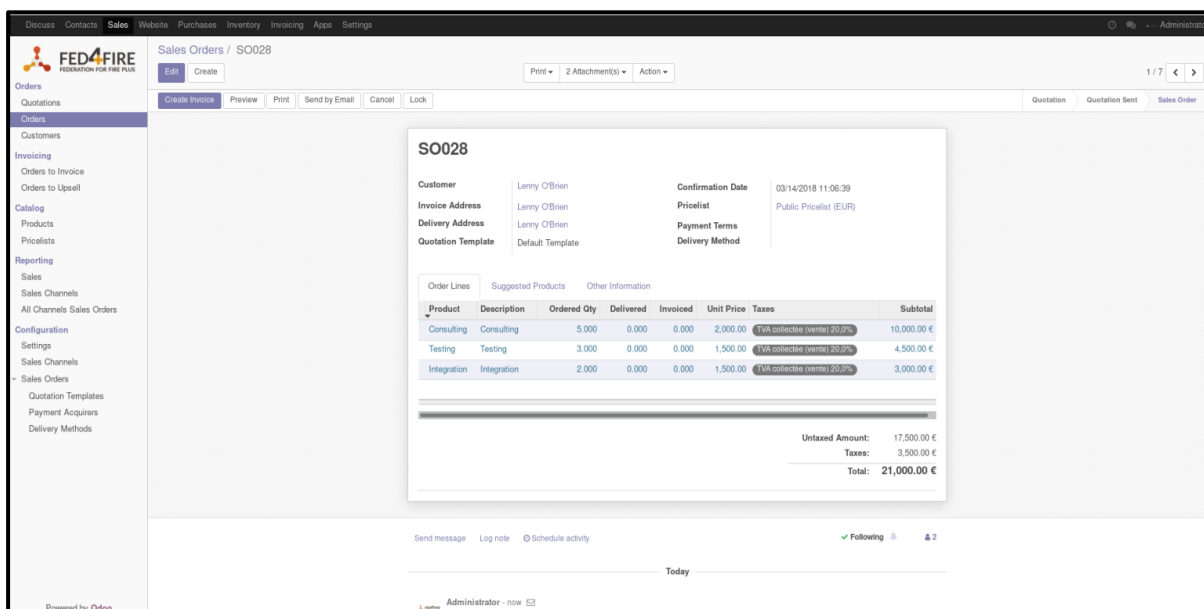


Figure 32: Odoo Backend – Vendor's Sales Order View.

3.5.2 Invoice and a Confirmation of Customer's Payment

As noted above, the customer can confirm his order in two manners: either through a digital signature or through an online payment. In the first case, the vendor has to check that he has received the customer's payment in his bank account (out of Odoo), confirm the payment manually on Odoo, then send a payment confirmation to the customer (Figure 33). In the



D4.02: First TaaS Prototype

second case, the payment is confirmed automatically, an online payment provider sends a notification to the vendor and the electronic financial transaction (EFT) is done immediately.

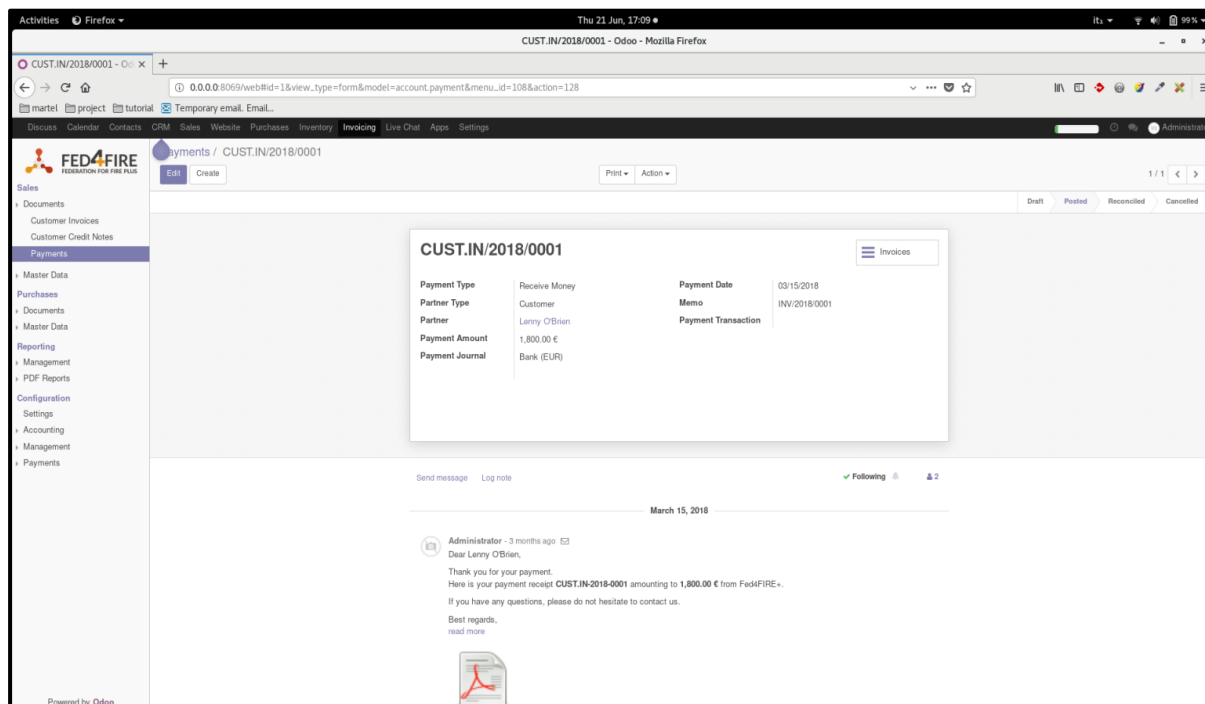


Figure 33: Odoo Backend – Vendor's Invoice View.

Now the customer can get his service requested (Figure 34).

D4.02: First TaaS Prototype

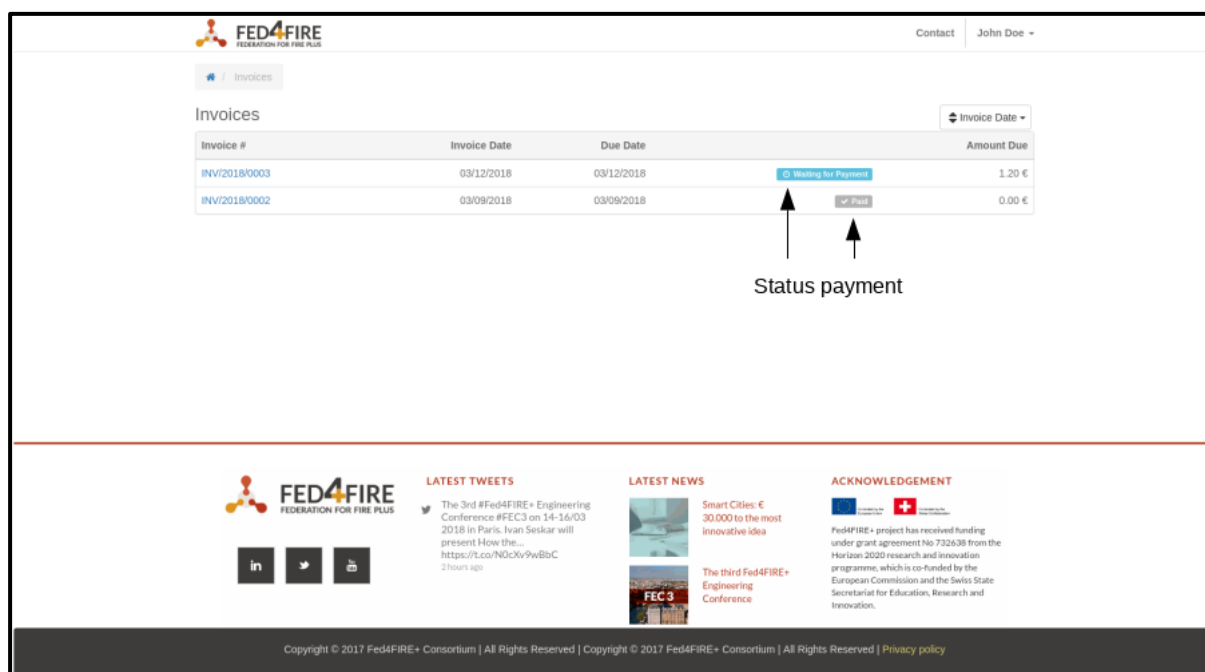


Figure 34: Odoo Frontend – Customer's Payment List Page.

3.5.3 Evaluation

Through Odoo, External API and opportunistic settings, we can create an integration among Marketplace, MySlice and Odoo itself. In detail, the customer books a service/product on market.fed4fire.eu, filling in a digital form. Marketplace/MySlice system sends all data contained in the form to Odoo, which automatically stores them in its database. Odoo sends a notification to the vendor. Depending on what service/product the customer chooses, the Marketplace/MySlice system sends an email to the email address associated with the service/product selected.

For example, IIoT service is associated with the email address: sales1f4f@fed4fire.eu. The vendor's email address is associated with this email: vendorf4f@fed4fire.eu.

Now the customer selects IIoT service at market.fed4fire.eu, Marketplace/MySlice system sends an email to sales1f4f@fed4fire.eu.

Odoo receives the email, creates a lead/opportunity (the step before being an RFQ) in the associated vendor account and sends an email notification to vendor@fed4fire.eu.

The vendor gets the notification, logs in to his account (crm.fed4fire.eu) and converts the lead/opportunity to an RFQ. The RFQ is sent to the customer.

The customer gets an email with all RFQ data. He logs in to his account (crm.fed4fire.eu) to manage it or to simply communicate with the vendor and the business starts.

4 CONCLUSION/FUTURE WORK

By combining existing tools, it was possible to create a prototype that has already met the first prototype requirements, i.e., only some components such as the marketplace or the matchmaking system have been developed. The next steps are outlined below.

4.1 MARKETPLACE

For the Marketplace component there is still a wide range of features that can be implemented. This includes the customer and testbed provider interaction as well as communication with some other components. The list of additional features:

- Using the Odoo's SSO.
- Extension of the start page for users.
- Stronger integration of Odoo into the frontend.
- Further tests for the registration of testbed providers and their services.

4.2 MATCHMAKING

As with the Marketplace, the Matchmaking System also has features that can be added to it. Starting with the refinement of the presented architecture up to the replacement of individual steps by more suitable methods, as for example neural networks for the determination of the weights.

The list of additional features:

- Refinement of the current architecture in order to create an interoperable and intelligent system.
- Replacement of individual components by more suitable methods.
- Integration of The Natural Language Processing (NLP) concept.
- Integration of neural networks for improved weight calculation of a term or words.

4.3 MYSLICE

After initial integration of MySlice and One-Stop-Shop frontend there is a need for:

- Further integration of the frontend (One-Stop-Shop) visible to potential customers and Odoo.
- Single accounts for the users and providers on both systems.

4.4 ODOO

After the integration among Odoo, MySlice and the marketplace, the next step is a single sign-on valid for all systems involved. Further future work includes:

- Creating a common graphic design for all systems involved.
- Setting the online payment methods.

D4.02: First TaaS Prototype

- Integrating Odoo with the social network.
- Improving customer account.

5 WORKS CITED

1. Klacza, R., Vaissade, F., Willner, A., Ahmed, Z., & Rook, J. (2017). D4.01: TaaS Gap Analysis Report.
2. Nguyen, H., Tassinari, O., Brookes, M., Ross, K., Marks, N., Sebald, S., Crockett, T. (n.d.). Material-UI. Retrieved from <https://material-ui.com/>
3. Google. (n.d.). Design - Material Design. Retrieved from <https://material.io/design/>
4. Rocha, C., Schwabe, D., & Poggi de Aragao, M. (2004, May). A hybrid approach for searching in the semantic web. Proceedings of the 13th international conference on World Wide Web.
5. Facebook Inc. (2015). Flux - In Depth Overview. Retrieved from <https://facebook.github.io/flux/docs/in-depth-overview.html#content>
6. Fernandes, Joao, et al. "IoT Lab: Towards co-design and IoT solution testing using the crowd." 2015 International Conference on Recent Advances in Internet of Things (RIoT). IEEE, 2015.
7. Odoo Apps. Official Odoo apps store. Retrieved 10:24, November 02, 2018, from <https://apps.odoo.com/apps/browse>
8. Odoo, ORM API. Official Odoo Documentation. Retrieved 10:24, November 02, 2018, from www.odoo.com/documentation/11.0/reference/orm.html#reference-orm-model
9. "XML-RPC." Wikipedia, The Free Encyclopedia. 16 September 2018, at 15:33(UTC). <<https://en.wikipedia.org/wiki/XML-RPC>>.