pharaon

PILOTS FOR HEALTHY AND ACTIVE AGEING

Grant Agreement: 857188

D4.9

Service Composition Support Tools – Final

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 857188.

Document Information

Deliverable number:	D4.9
Deliverable title:	Service composition support tools – Final
Deliverable version:	V1
Work Package number:	4
Work Package title:	Technology support tools
Date of delivery:	M54 (May 2024)
Dissemination level:	Public (PU)
Type	Other (O)
Editor(s):	Alex McDonald (ICE)
Contributor(s):	Ann Campbell (ICE)
Reviewer(s):	Sergio Abbenante (CETEM)
Project name:	Pilots for Healthy and Active Ageing
Project Acronym	PHArA-ON
Project starting date:	01/12/2019
Project duration:	60 months
Rights:	PHArA-ON Consortium

Document history

Version	Date	Beneficiary	Description
0.1	22.02.2024	Alex McDonald (ICE)	Initial version of the document
0.2	04.04.2024	Alex McDonald (ICE)	General updates
0.3	08.05.2024	Alex McDonald (ICE)	Updates based on review feedback
0.4	22.05.2024	Alex McDonald (ICE)	Adding clarifying comments
0.5	31.05.2024	Alex McDonald (ICE)	Updating the Concluding Remarks
0.6	18.06.2024	Sergio Abbenante (CETEM)	Review of the document
1.0	18.06.2024	Alex McDonald (ICE)	Final approval

PM Efforts per Beneficiary having contributed to the Deliverable

#	Partner	PM effort in D4.8
32	ICE	2.17
7	CETEM	0.05
TOTAL	**Pharaon Consortium**	**2.22**

Acknowledgement: This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 857188.

Executive Summary

This is a software deliverable. Therefore, the present report is intended to be brief. The approach followed by Phara-On for service composition consists of using the ICE Orchestration tool to design processes that represent the actual composition of services.

As this is the third, and final iteration in Task 4.3 Service Composition and Support Tools, this will go over the development throughout previous iterations and show the progress.

Progress Beyond the State of the Art and Play

The ICE Orchestration Tools are a BPMN suite that contains different modules which are detailed both in previous documents, and later in this document. The modules can be divided into 'Design' and 'Runtime' categories.

The Design modules are intended to be used by technical people, or people with a stronger knowledge of scripting languages and data structures.

The Runtime modules can be used by any person, as their content is created in the Design stage; the only limitation depends on whatever screens and workflows have been created to use.

The initial version of this deliverable in D4.7 discussed the development plans for the service composition tools and covered the basics of how to use the tools, from Design time, where users are technical users creating workflows, to utilizing the workflow in action, where users could be anyone, from highly technical people to care workers.

The previous deliverable, D4.8, went into more depth around the new features introduced to the service tools in much more technical depth. We introduced the Service Registry and API Gateway in order to work with APIs, completed the general orchestration process to a higher level, utilizing a custom API to manage the workflows created by the technical users.

During this period, we worked on upgrading the composition tools to a modern architecture after running into some important issues with the old system. We refactored the composition tools from its previously Java based backend, and upgraded functionality with a series of microsystems using NodeJS. We've also finished integration with the greater PHArA-On Ecosystem.

In addition, work done during this period compared to the previous one:

- We refactored and upgraded the control panel, or the Processes View, to give users more information on active, completed and deleted processes. It shows users a wealth of information on processes, specific instances of processes and we can show the workflow in a BPMN model.
- We created the Dash Button microservice that allows developers to easily interact with Keycloak.
- We integrated the Service Directory so that users using Designtime modules can now utilize all services available through the repository.
- We integrated the API Gateway so that users can input services into the Service Registry that use OAuth2 or different types of security. Services added this way can then be utilized in the Designtime modules in the same way as the simple OpenAPI services.
- The ICE Orchestration Tools are integrated with the PHArA-On Ecosystem Hub, allowing users with Ecosystem Hub credentials to access a PHArA-On specific build of the tools.
- In conjunction with the PHArA-On-wide automated testing, we implemented unit testing on all of the ICE Orchestration Tools.

Contents

Figures

Tables

1. Introduction

1.1. Overview

This Deliverable is of type Other, which means **it is a software deliverable**. Therefore, the present document is meant to be compact, briefly explaining the approach and the technology used in task T4.3 Service composition support tools; it also provides a glimpse at how the technology works and details to access the live version of the primary tool used in this task. Finally, there are some brief concluding remarks which contain some reflection on the changes throughout the lifecycle of the Service Composition Tools within the PHArA-On and what may lie in the future beyond the project.

The main objective of the Orchestration software is to visually manage the interaction between services, focusing on the business requirements by each company. Since it's intended to be used by domain/business experts, the tool tries to hide those technical complexities that are intrinsic to any dynamic software design.

1.2. Relation to other tasks and deliverables

This Deliverable is related to the following other PHArA-ON tasks and deliverables:

Receives inputs from:

***Table 1**. Inputs from other deliverables and tasks.*

Input description	Deliverable or task	Work package
User requirements and use cases	D2.1User and pilot requirements	WP2 Pilots and user requirements, initial ecosystem architecture
Reference architecture	D2.2 Pharaon initial ecosystem architecture	WP2 Pilots and user requirements, initial ecosystem architecture

Provides outputs to:

***Table 2**. Outputs to other deliverables and tasks.*

Output description	Deliverable or task	Work package
Approach to service composition; ICE Orchestration tool	D3.9 Service composition support tools – intermediate	WP3 Secure Interoperability Solution
ICE Orchestration and service composition UI for non-technical users	D5.2 Pharaon ecosystem – intermediate	WP5 Technology ecosystem integration

1.3. Structure of the Deliverable

The initial description of the software was described in D4.7. In this deliverable, an overview of the features and modules, both new and those mentioned in D4.8, are shown here. The sections of this document can be summarized as:

- Section 2 shows the production ready view of the ICE Orchestration Tools, from the Design components, to the Runtime components.
- Section 3 describes the functionality of the Service Registry and API Gateway and highlights the integration with the different ICE Orchestration Tools.
- Section 4 describes the functionality of the Dash Button and how it connects to Keycloak.

1. ICE Orchestration Tools

The ICE Orchestration Tools are a suite of multiple modules that make up a collection of tools that facilitate the orchestration of workflows, and it supports the use of services within this process.

The live PHArA-On service can be found at https://pharaon.icelab.cloud/. Whilst due to security reasons and the public nature of deliverable D4.9, ICE cannot disclose any account (username and password) in this document. However, the credentials needed at the as credentials needed for access to any part of the Ecosystem Hub.

As previously noted, the tools can be broken down in two sections, 'Design' modules and 'Runtime' modules.

The Design Modules consist of the following:

- The Process Designer
- The Form Designer
- The Service Registry

Whilst the Runtime Components consist of:

- The Processes View
- The Task Manager

1.1. Process Designer

The functionality of the Process Designer is explained in the deliverable D4.7 and no major changes have been made to impact general use.

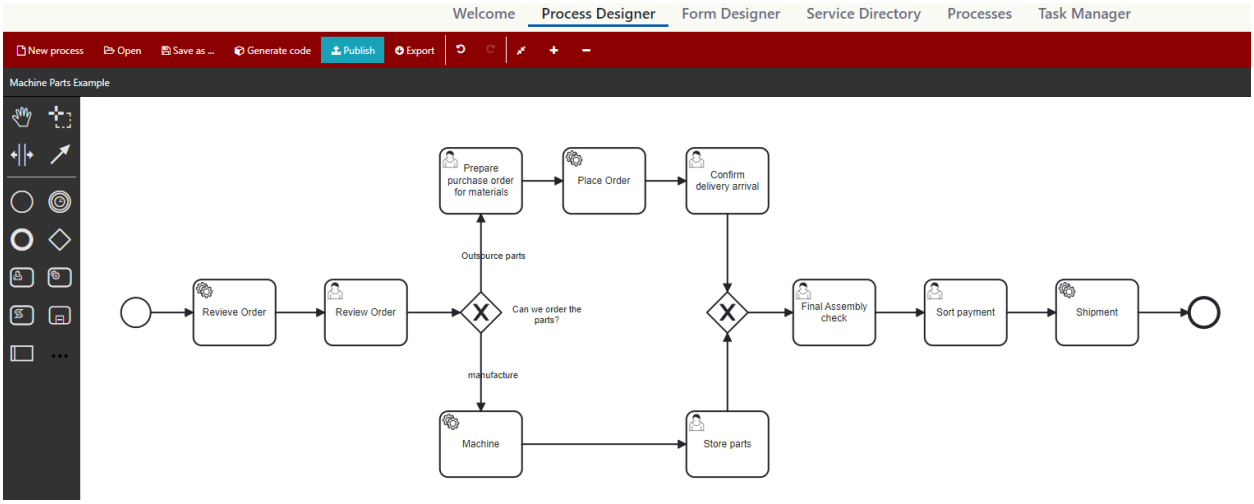

***Figure 1**: Example of process composing services and human activities.*

Prior to this deliverable, the Process Designer consisted of an iframe that hosted a view within a collection of Java tools. Now that we're using NodeJS as a base, we can integrate the Camunda supported library, BPMN.io. This allows us to better integrate the features of the Process Designer developed by ICE as previously discussed in D4.7.

Code generation, as discussed in D4.7 now has its own independent task, called the Script Task.

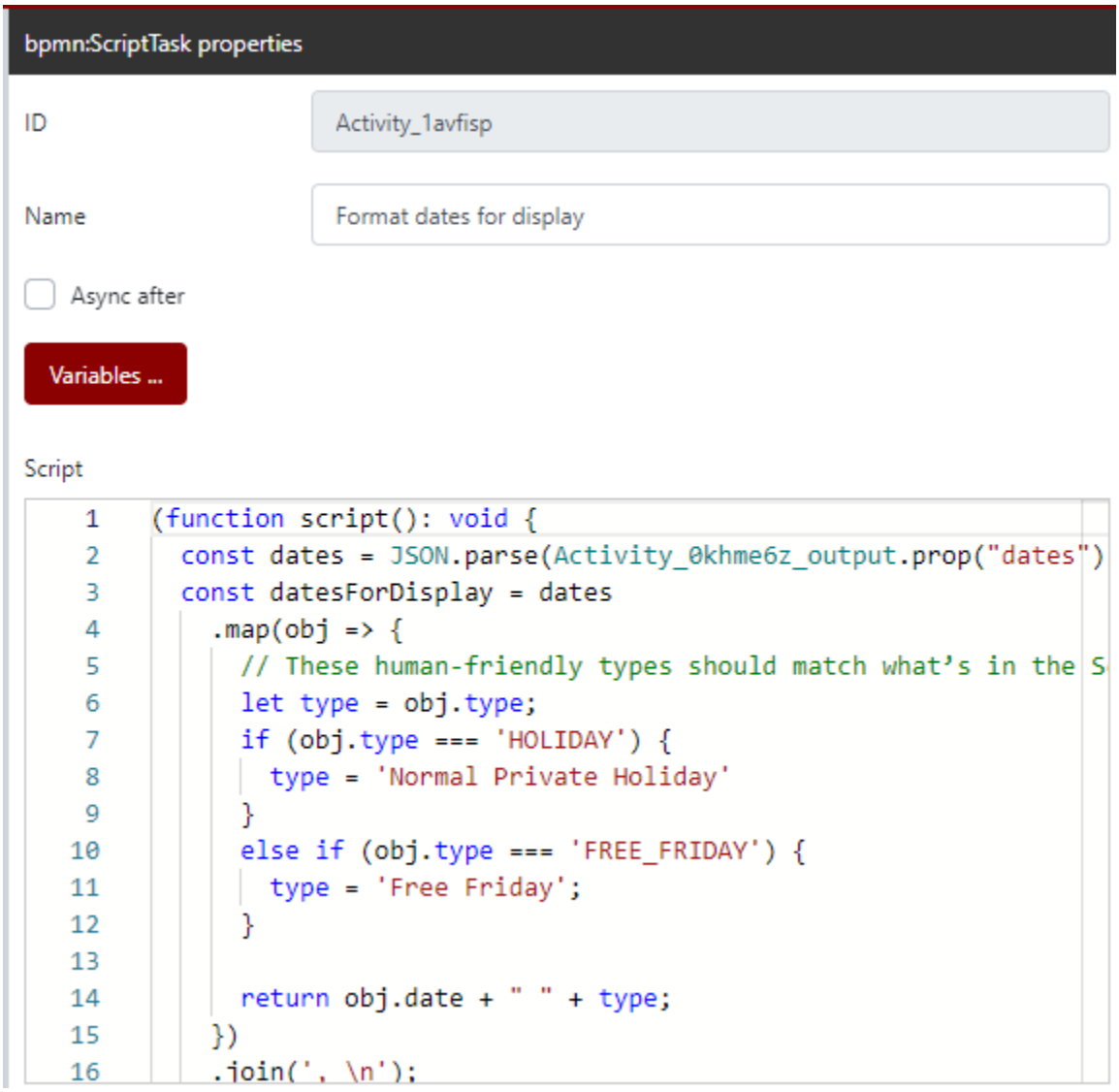

***Figure 2**: Example of the properties of a Script Task showing a task written in Javascript.*

Script tasks allow users to fully create their own tasks themselves using Javascript. It will help validate Javascript code, and it will throw error warnings if there is something wrong.

1.2. Form Designer

The smallest individual module of the ICE Orchestration Tools, its very existence is tied to the Process Designer; user tasks. User tasks are tasks that require user interaction, most of the time this involves a form. A form allows a user to enter multiple different types of data and this data can then be passed around the workflow, being used by services or being manipulated by script tasks.

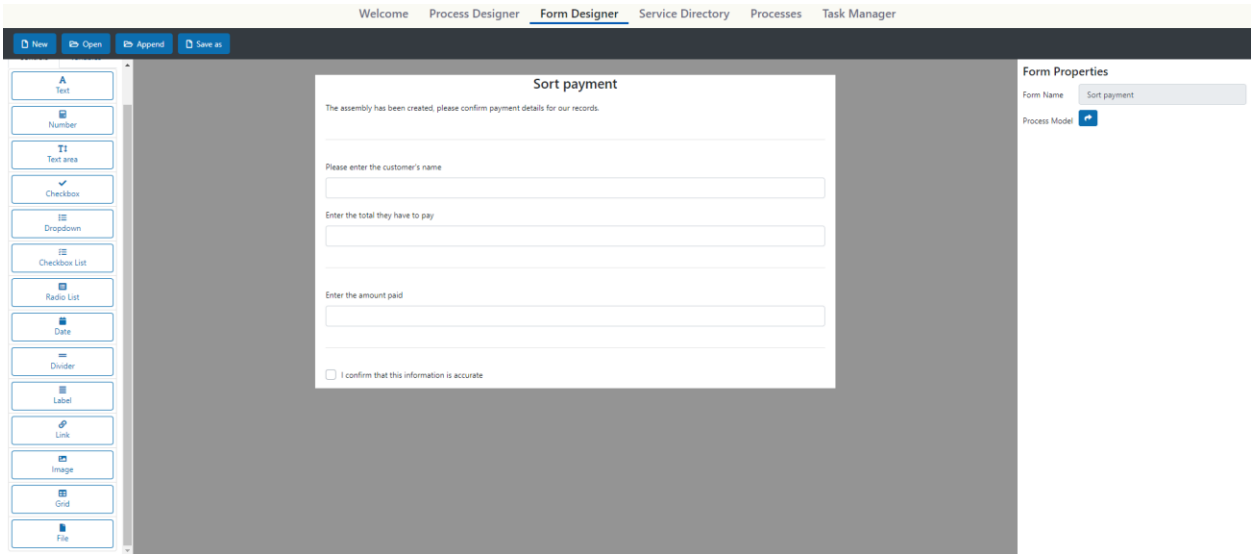

***Figure 3**: Example of a Sort Payment form.*

Users can drag elements from the palette on the left side of the screen onto the center canvas, and customize the form as they see fit. These forms are then linked to user tasks within the Process Designer, and they create tasks for users within the Task Manager.

1.3. Processes View

In deliverable D4.7, this was referred to as the Control Panel, and its functionality is broadly similar. It had to be completely remade in AngularJS from the previous build. The functionality is the same, and will not be repeated here, however the following screenshots show the UI changes over time.

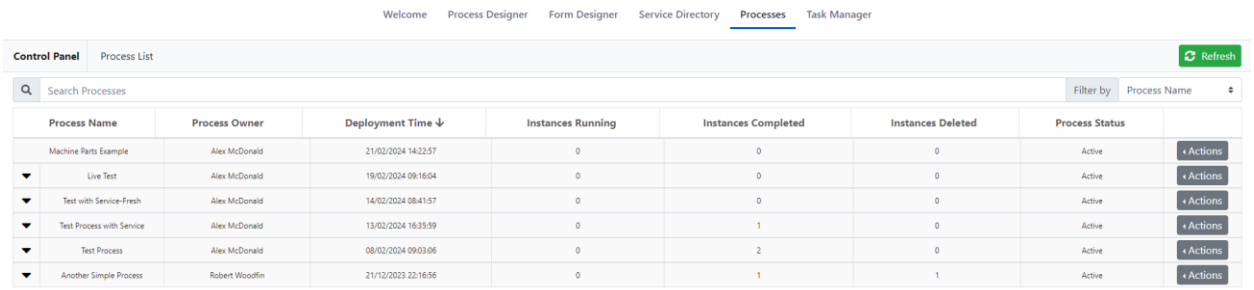

***Figure 4**: A screenshot of the Processes View.*

The Processes View starts by showing you the list of processes. You can select a particular process, or use the action button to view all the instances of that process.

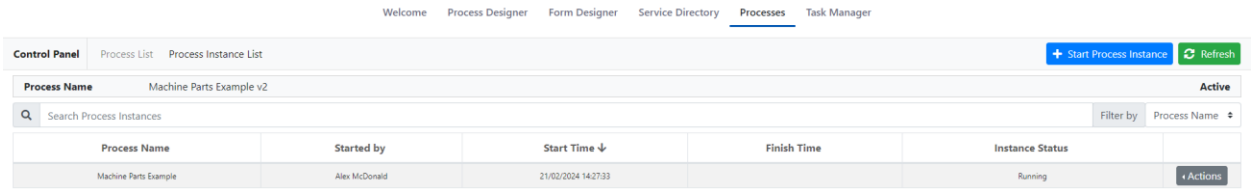

***Figure 5**: A screenshot of the Process Instances View of the Machine Parts Example.*

There is a similar view on the Process Instance View. Here you can interact with the instances; starting new instances, deleting instances or viewing their variables. You can also view a specific instance.

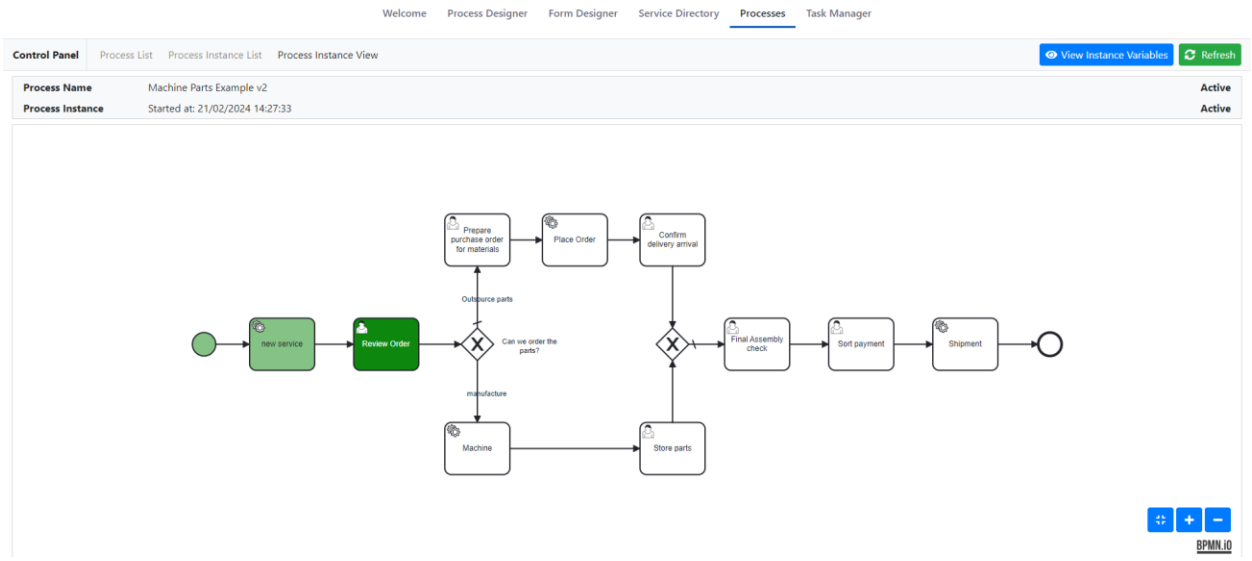

***Figure 6**: A screenshot of the Process Instance View of an instance of the Machine Parts Example process.*

This final view utilizes the BPMN.io library to show the workflow in the same manner that would happen during the Process Designer.

1.4. Task Manager

In deliverable D4.7, the Task Manager was called My Tasks, however, its functionality has expanded since the initial build.

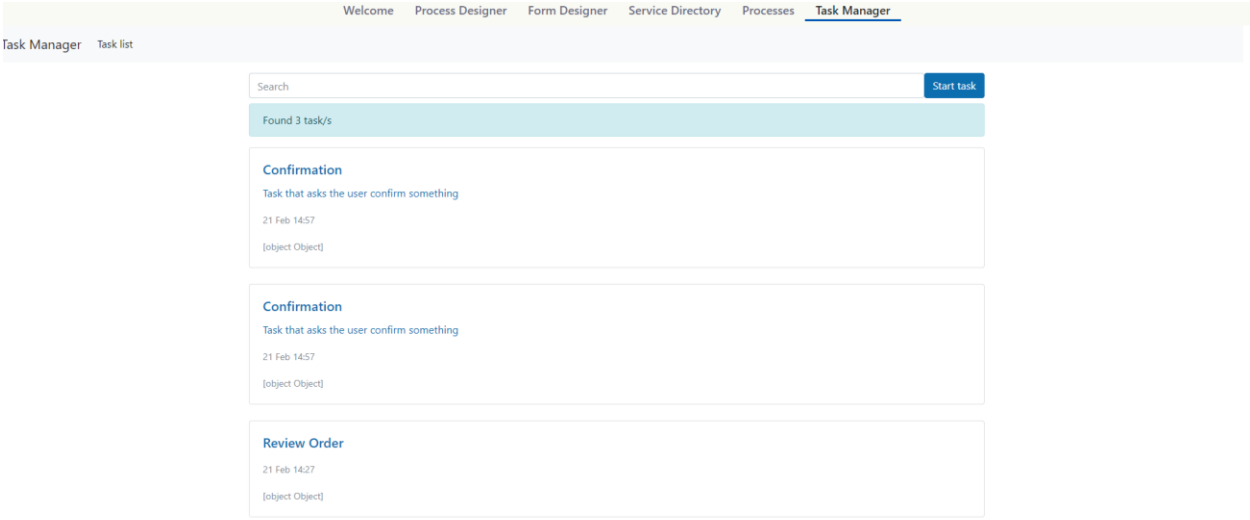

***Figure 7**: A screenshot of the Task Manager showing the active tasks for the signed in user.*

The Task Manager details a list of active tasks that are all assigned to the currently signed in user. However, you see both tasks assigned to you, as well as all active tasks. You can search for specific tasks using the search bar.

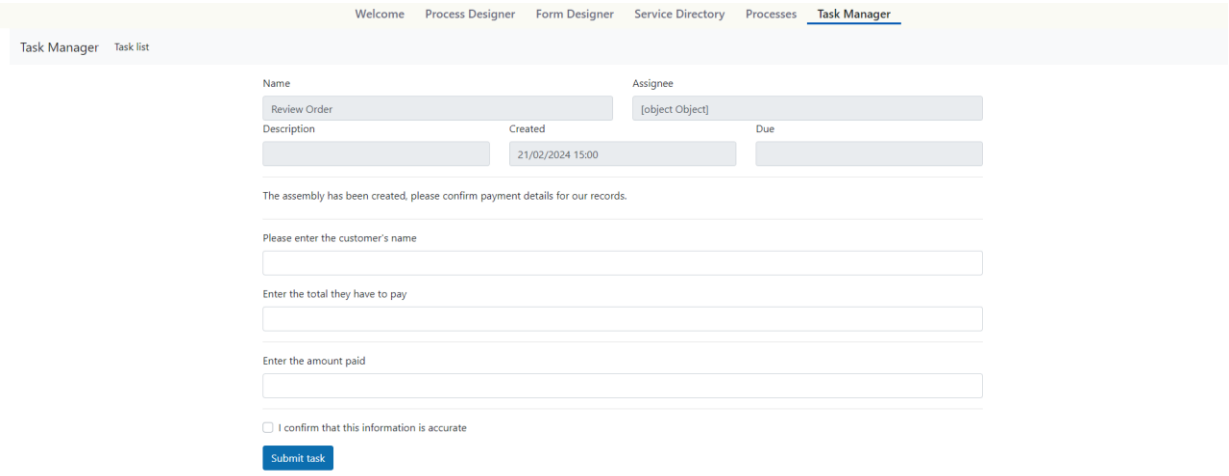

***Figure 8**: A screenshot of an active task, specifically the task view of the form created in Figure 3.*

When a user opens up their task, they're taken to the task view, where they see the task in the shape of the form created using the form designer. From here, they can fill out details of the form and when they click the Submit Task button, the task closes, and the workflow it's attached to continues.

2. Service Registry and API Gateway

The Service Registry was introduced in the D4.8 deliverable, where the functionality is explained in more detail. It is a module that is a central REST API that returns a list of services and their operations; it contains an OpenAPI endpoint that provides an automatic description. It was developed to meet the Data Interoperability needs of the PHArA-ON project.

It should be noted that the Service Registry is sometimes referred to as the Service Directory or the API Repository; these names are interchangeable and refer to the same component. The Ecosystem Hub refers to it as an API Repository because it holds the API services, whereas the Service Composition Tools refer to it because it registers services in an API format. This document will refer to it as the Service Registry, but documents outside of the scope of this one may refer to it differently.

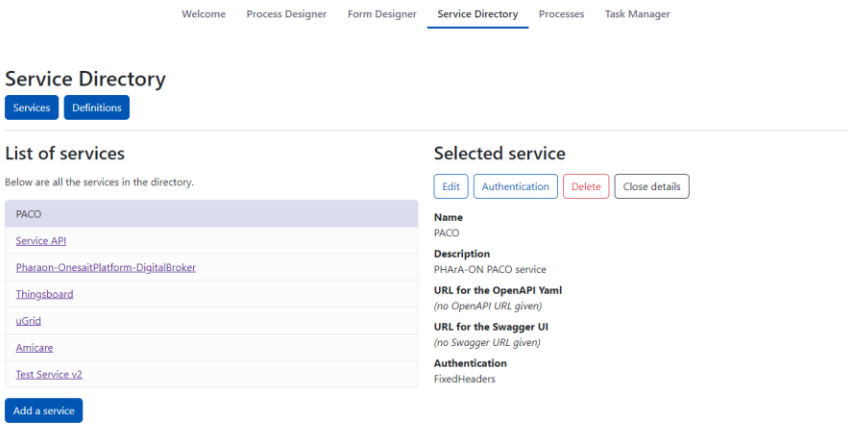

***Figure 9**: A screenshot of the Service Registry showing the PHArA-On Pilots that have services registered.*

Should the service have a security layer, a user can also add in authentication, allowing all services to be interacted with through the API Gateway. The API Gateway, which is also explained in more detail in the previous D4.8 deliverable, acts as a security authentication, using defined security credentials that the user sets through the Service Registry.

When a service is registered, it can then be interacted by different components. With the ICE Orchestration Tools, the Process Designer can view them directly.

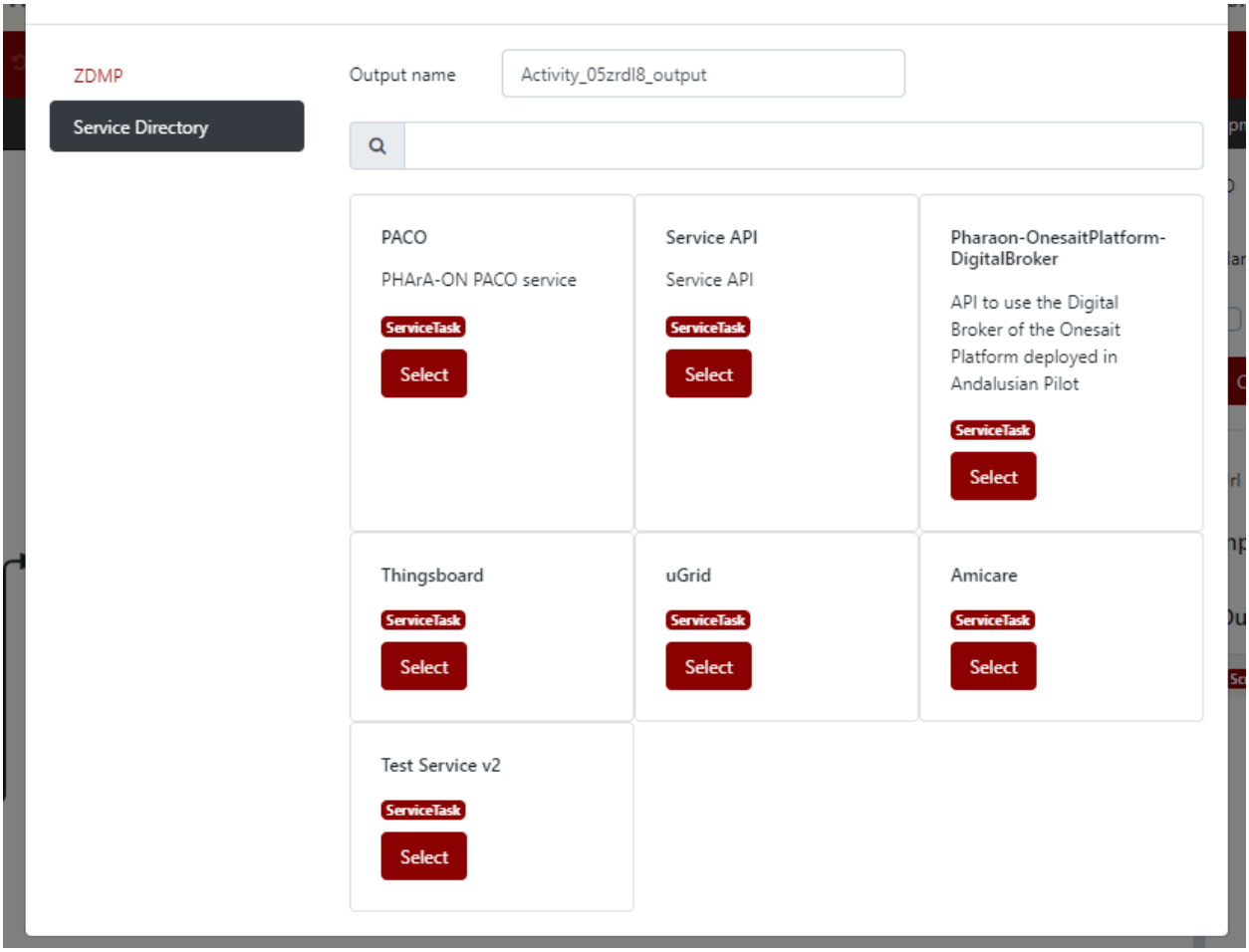

***Figure 10**: A screenshot of the model view that appears when a user creates a service task on the Process Designer.*

The Process Designer utilizes Service Tasks to interact with the services that are held on the Service Registry via the API Gateway.

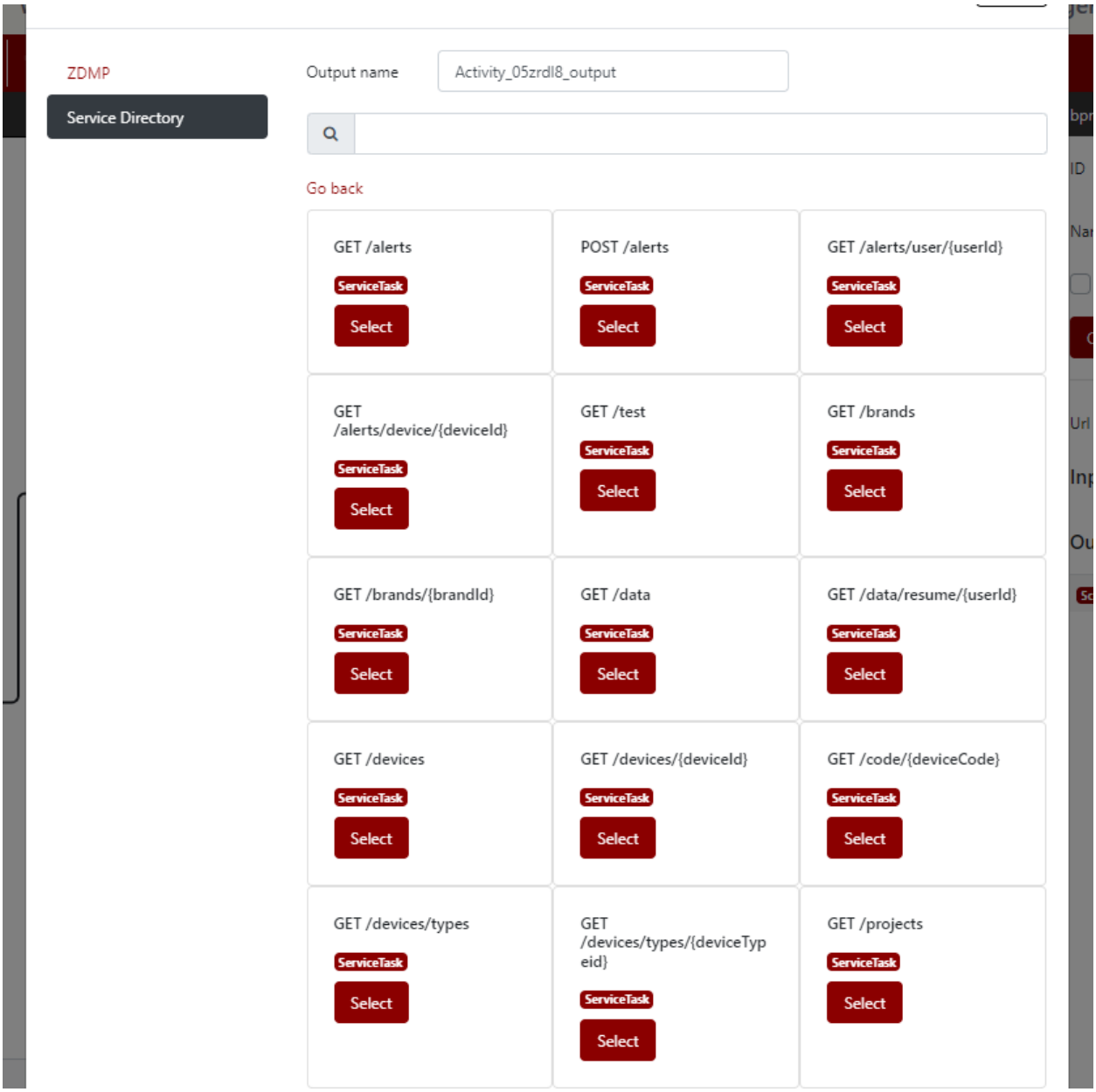

***Figure 11**: A screenshot of the uGrid API calls available on the Process Designer.*

When you click on the services, you can see the various API calls as defined in their own OpenAPI definitions. These calls can then be utilized in the BPMN workflow, from retrieving information from the calls, to passing data garnered by forms to them.

This functionality was highlighted in the D4.8 deliverable, but the UI has been remade to be more modern and user friendly.

3. Dash Button and Keycloak Integration

As a part of the future steps section of the D4.8 deliverable, it includes the task of meeting the authentication requirements of the central PHArA-On Ecosystem Keycloak. The Dash Button is the solution to this task. The Dash Button, a shortened form of "Dashboard Button" is an AngularJS component that can be integrated into a webpage, and with the accompanying library, it allows developers to easily connect straight to a Keycloak instance.

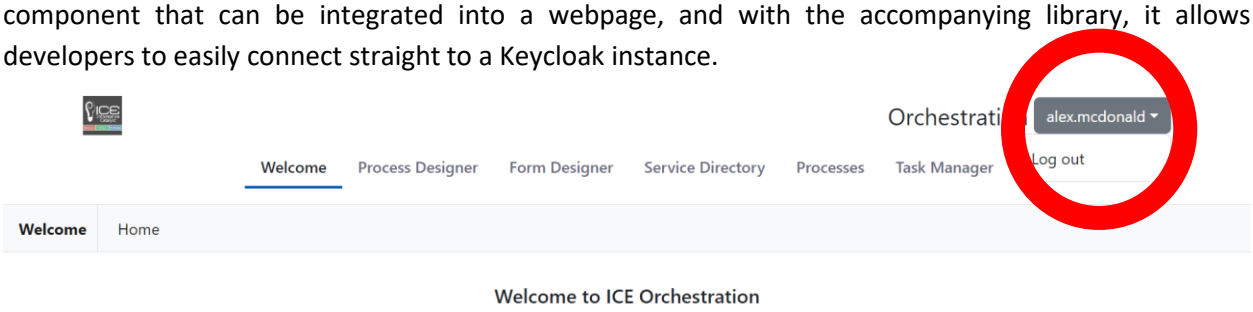

***Figure 12**: A screenshot of the top of the Orchestration Tools when someone is signed in; circled in red is the Dash Button after being clicked.*

The user can click on the Dash Button to to sign into the Orchestration Tools, and when signed in, they can click it to get the option to log out.

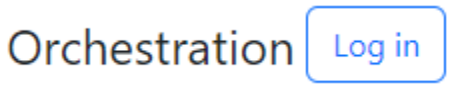

***Figure 13**: A screenshot of the Dash Button when a user is not signed in.*

When you attempt to log into the Orchestration Tools by clicking on the Log in view of the Dash Button, you'll be taken to the Ecosystem Keycloak log in page, where you can then log in via your Ecosystem credentials.

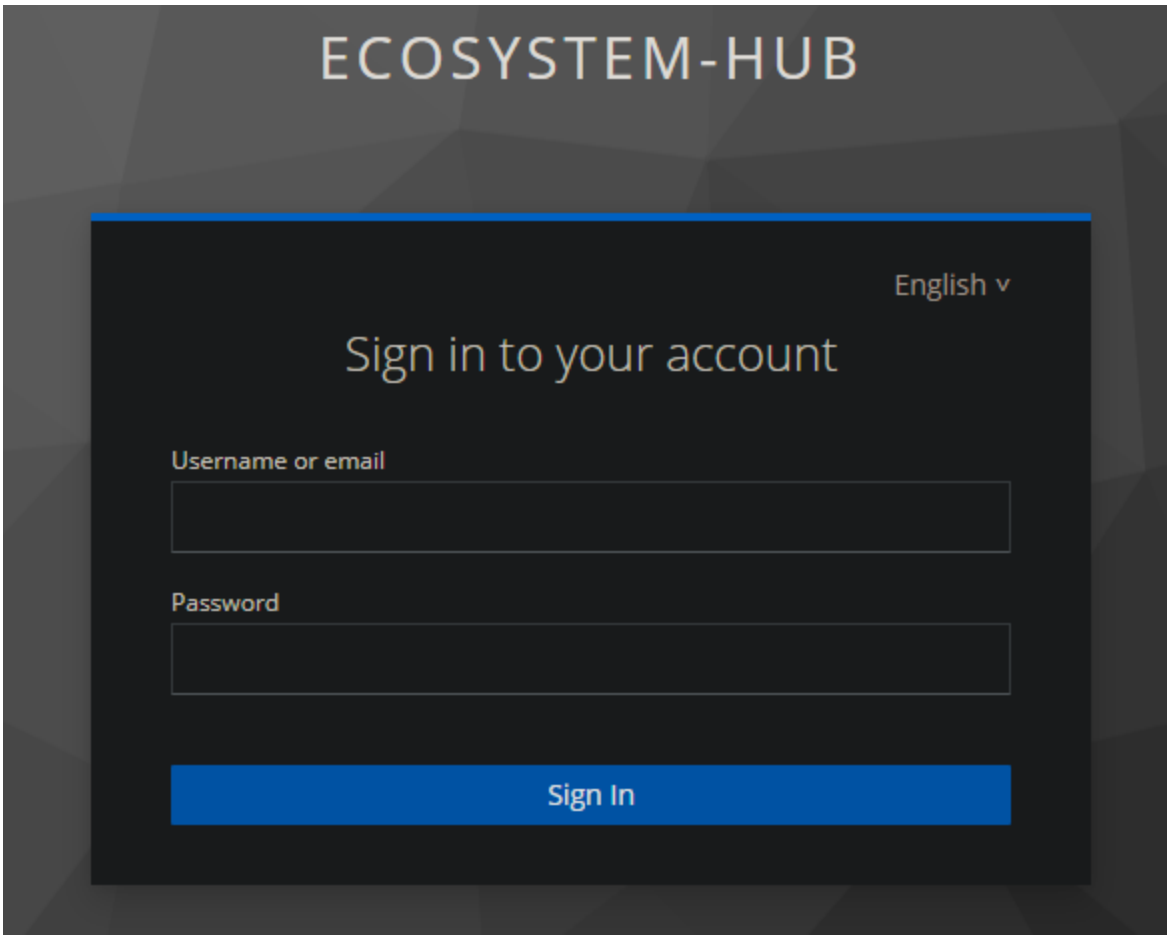

***Figure 14**: A screenshot of the Ecosystem-Hub Keycloak login page.*

The Dash Button works by saving the user's ID in cookies, and it passes it through the various modules, allowing users with different levels of access to use different tools. In particular, Keycloak uses the concept of "realms" to control access. The Orchestration Tools have access to a specific realm, which the Dash Button will check for when the user opens a particular module. Should they lack access, they will be denied.

4. Concluding remarks and the future

This deliverable marks the final steps for development of the service composition tools across the PHArA-On ecosystem. The work may be done for PHArA-On but the project has given ICE plenty of ideas on how to develop the service composition tools in the future beyond the project.

Overall, the tools that were brought in by ICE prior to PHArA-On, including the Process Designer and Form Designer, both of which heavily interact with the third-party tool Camunda, were expanded on to meet the Service Composition needs of PHArA-On. The Control Panel and Task Manager were built around such expansions. In addition, the Data Interoperability requirements of PHArA-On resulted in the creation of the API Repository and API Gateway which further guided development of the Service Composition Tools.

Below, listed in no particular order, is a list of features that can be developed using all the information PHArA-On has brought to the tools:

- In the both the previous initial and intermediate versions of this document mentioned the concept of a less technical user interface, removing the limitation that the Orchestration Tools has of requiring some technical knowledge in order to utilize it.
- The Service Composition Tools were much better suited towards developers, and this shows by how they were utilised by the Ecosystem Hub rather than the pilots. The pilots were offered access to the tools, especially In OC2, but only part of the Service Composition Tools were useful for them (the Dash Button and the API Repository in particular)
- During the project, the service tools moved from using an integrated Liferay build with a Java backend, to a modern architecture utilizing NodeJs and microservices. There are some less vital features that were cut during the refactoring process that would help increase the amount of information a user could receive, for example, in the Process Instance View and showing data on script tasks
- Expansion of the Ecosystem-Hub tools, including the API Repository and the API Gateway. These components were developed for the project, and they will be expanded upon with further development
- Development of the Dash Button has resulted it being in other projects, as pilots utilise it as a Keycloak single sign on tool
- General bug fixing; every component is going to have unforeseeable bugs appear when developing, the service composition tools are no different