OACore: Software de gestión de información

Proyecto de Fin de Grado en Ingeniería del Software

Grado en Ingeniería del Software

Facultad de Informática

Universidad Complutense de Madrid

Autor: Adrian Estiben Sanchez Hurtado

Dirigido por: Belén Díaz Agudo

Codirigido por: José Luis Jorro Aragoneses

Curso 2018/2019

Dedicatoria

Al ingeniero e increíble maestro, a aquel que un día prefirió dejarlo todo por brindarnos libertad. Al Creador del mundo y fuente de toda creatividad y diseño. A Dios sea la gloria y el honor por inspirar cada buena obra.

Resumen

Durante las últimas décadas la transformación digital empresarial ha sido una realidad que ha traído como consecuencia el uso cada vez más abundante de distintas tecnologías en todos los niveles de organización con el fin de mantener la competitividad en el mercado ya sea en términos de producto, producción, costes, calidad y otros más. Esto trae como consecuencia la necesidad de un equipo técnico (departamento de tecnología de la información o IT) experto en todas y cada una de las tecnologías usadas con el fin de integrarlas de acorde a los procesos de negocio. Poseer un equipo tal suele ser inexistente en pequeñas y medianas empresas, generando costes muy altos en la contratación de servicios y limitando la toma de decisiones en la empresa debido a la complejidad o al desconocimiento de la parte que aborda cada tecnología dentro de los procesos de negocio.

Este proyecto tiene como objetivo iniciar el desarrollo de un software capaz de ejecutar, modelar y simplificar la integración de distintos programas informáticos a través del diseño y la configuración de flujos de trabajo. Con este, las empresas podrían integrar distintas tecnologías sin la necesidad de contar con expertos de cada uno de los programas informáticos y reducir así los costes de contratar servicios externos.

Se ha logrado desarrollar un servicio web en el que se usa el elemento Canvas de HTML5 que permite la representación gráfica para el diseño de flujos de trabajo, dentro de un software desarrollado en el lenguaje de programación java con el que automatizar la ejecución de programas informáticos y la comunicación entre estos. Esta comunicación se ha realizado a través de la entrada estándar de proceso y el uso de sockets de internet.

PALABRAS CLAVE

Flujo de trabajo, Automatización de procesos, Proceso, IPC, Comunicación entre procesos, Diseño de flujos, Programación Reactiva

Abstract

During the last decades, the digital business transformation has been a reality that has resulted in the increasingly abundant use of different technologies at all levels of organization in order to maintain competitiveness in the market whether in terms of product, production, costs, quality and others. This results in the need for a technical team (department of information technology or IT) expert in each and every one of the technologies used in order to integrate them according to business processes. Owning such a team is usually non-existent in small and medium-sized companies, generating very high costs in contracting services and limiting decision-making in the company due to the complexity or ignorance of the part that addresses each technology within business processes.

This project aims to initiate the development of software capable of executing, modeling and simplifying the integration of different computer programs through the design and configuration of workflows. With this, companies could integrate different technologies without the need of contracting experts from each of the computer programs and thus reduce the costs of hiring external services.

It has been possible to develop a web service in which the HTML5 Canvas element is used to allows the graphic representation for the design of workflows, within a software developed in the java programming language with which to automate the execution of computer programs and communication between these. This communication has been made through the standard process input and the use of internet sockets.

KEYWORDS

Workflow, Process Automation, Process, IPC, Inter-Process-Communication, Flow Design, Reactive Programming

Índice

1. Introducción 12
1.1. Antecedentes 12
1.2. Estado del arte 13
1.3. Objetivos 16
1.4. Estructura del Documento 16
2. Sistema OACore 17
2.1. Diseño 18
2.2. Detalles adicionales de ejecución de procesos 24
2.3. Binders 25
2.3.1. DirectStdInBinder 27
2.3.2. SplitBinder 27
2.3.3. ExitCodeBinder 27
2.3.4. RunnableBinder 27
2.3.5. StdInBinder 28
2.3.6. StdOutBinder 28
2.4. Workflow 28
2.5. Servicio web 30
2.6. Diseño de flujos de trabajo 33
3. Conclusiones 38
3.1. Conclusiones 38
3.2. Trabajo Futuro 39
3. Conclusions 40
3.1. Conclusions 40
3.2. Future Work 41
4. Bibliografía 42

Índice de figuras

Ilustración 1: Componentes del proyecto OACore 18

Ilustración 2: Flujo de ejecución de procesos y binders 19

Ilustración 3: Diagrama de clases que implementan la interfaz Binder 20

Ilustración 4: Diagrama de clases que son publicadoras 21

Ilustración 5: Diagrama de clases que son suscriptoras 21

Ilustración 6: Diagrama de clases entre workflow, procesos, binders y otros. 22

Ilustración 7: Diagrama de clases de proceso en la capa de modelo del módulo web 23

Ilustración 8: Flujo de navegación de usuario en la web y la interacción con el controlador web REST 24

Ilustración 9: Formulario de registro e inicio de sesión web. 31

Ilustración 10: Formulario web de ajustes de usuario. 32

Ilustración 11: Formulario web de perfil de usuario. 32

Ilustración 12: Interfaz de diseño de flujos de trabajo. 34

Ilustración 13: Formulario para la asignación de propiedades de proceso. 35

Ilustración 14: Interfaz de diseño de flujos de trabajo con tres procesos enlazados. 36

Ilustración 15: Formulario de botones para procesos en la interfaz. 36

Ilustración 16: Formulario de asignación de propiedades de binders. 37

1. Introducción

1.1. ANTECEDENTES

Debido a la evolución tecnológica y a la inclusión de esta en el mundo empresarial, el amplio desarrollo de diversos programas informáticos presenta una problemática a la hora de integrarlos en un mismo sistema informático. Esta integración también conocida como integración de aplicaciones para empresas o EAI[1] hace uso de principios de arquitectura que pueden usarse como bases o estándares tecnológicos con los que definir métodos con los que integrarlos, "buscando simplificar y automatizar los procesos de negocio sin requerir cambios integrales en las aplicaciones y estructuras de datos existentes" (Lui, Gray, Chan, & Long, 2011).

Por lo tanto, dada la existencia dentro de una empresa, de que se encuentren distintos programas informáticos y que estos mismos necesiten intercambiar información operativa o financiera, es necesario que el envío de la información sea efectivo. En otro caso, la intervención humana sería necesaria creando cuellos de botella que causan que la empresa se centre en utilizar recursos tanto coordinando las operaciones necesarias o bien ingresando la información manualmente. Es por esto, que una arquitectura EAI correcta permitiría que la organización pueda centrarse más en generar valor a la empresa que en coordinar estas labores operativas.

Así mismo, cuando se quiere integrar un software con otro (ya sea dentro de una arquitectura EAI) normalmente se requiere implementar dentro del sistema, el mecanismo especifico que utiliza el otro software y viceversa. Es decir, agregar toda una nueva funcionalidad o adaptar la funcionalidad actual. De igual manera, cada vez que se quiere integrar un nuevo software o bien este cambia, se necesita aplicar métodos de reingeniería. Un ejemplo de esto es la integración de un programa que lanza eventos a partir de las interacciones de usuarios en una web con un servidor de correo electrónico. Al ser estas dos últimas diferentes, es necesario que una de las dos utilice la API o interfaz de programación de la otra (en caso de estar disponible) e implementarla para poder integrarla. Este problema puede ejemplificarse fácilmente en la actualidad con el uso de la API Graph[2] de Facebook que, al mantener un constante cambio, muchas empresas dejan de lado si quiera considerar hacer uso de esta API porque en pocos meses o días recibirán una nueva actualización con cambios considerables.

Otra de las problemáticas que trae el uso de programas informáticos en los distintos niveles de la organización es la falta de comprensión del papel que realmente abordan estos a lo largo de los procesos de negocio. Por ejemplo, en el uso de una plataforma de mantenimiento del inmobiliario en donde el proceso de negocio define que cada mueble dañado de la empresa debe registrarse en la plataforma y en tres días ser reemplazado o arreglado. Desde el punto de vista del jefe del departamento de sistemas (IT), debe conseguir una forma de que cada vez que un mueble se dañe, sea agregado a través de la web de la empresa y este a su vez volcado a la plataforma de mantenimiento. Sin embargo, al considerar el punto de vista de otro jefe de

[1] MSquare Systems "Types of EAI". Archivado el 21 de mayo de 2014 en la Wayback Machine. MSquare Systems Retrieved on 28 May 2014

[2] API Graph, Facebook 2019, https://developers.facebook.com/docs/graph-api?locale=es_ES

departamento que no tenga conocimientos técnicos, no es exacto el uso de recursos de la empresa al considerar el coste de integrar la plataforma de mantenimiento con la web, o bien, a la hora de cambiar el proceso de negocio con un añadido como resolver cada incidencia en un plazo menor de 2 horas. Es decir, no se conoce ni de forma breve el uso real que se le da a los programas informáticos dentro de estos procesos ni los efectos de cambios en los procesos de negocio que tendrían sobre los sistemas informáticos.

Por otro lado, en el ámbito de educación superior, se logran sentar increíbles bases de conocimiento de la ingeniería, y más en concreto de la ingeniería informática y relacionadas, para que el alumno que se termine graduando pueda comenzar un proceso de ahondar más en cada tecnología fácilmente y poder salir al mundo laboral con conocimiento suficiente como para comenzar a formarse en las tecnologías que correspondan. Pero, el alumno no tiene los conocimientos suficientes o la experiencia necesaria en el uso de aplicaciones empresariales como para poder comenzar sin ninguna otra formación a ejercer su labor profesional. Esto queda claro, por ejemplo, en un estudiante de ingeniería del software que quiera aplicar a un puesto de desarrollador o consultoría de software de SAP[3] de SAP SE. (un software de planificación de recursos empresariales), si bien puede entender algunas de las tecnologías que pueda utilizar, no tiene la experiencia necesaria para comprender como todas estas tecnologías interactúan entre sí al formar este software.

1.2. ESTADO DEL ARTE

Durante varios años, los sistemas empresariales habían sido utilizados para un solo usuario o un grupo de usuarios limitando la comunicación de estos con otros sistemas y por esto que en el 2003 se reportó un fallo involucrando el 70%[4] de todos los proyectos EAI (Integración de aplicaciones para empresas) y más aun con el avance de la transformación digital, y la necesidad de compartir datos en tiempo real y a través de distintos sistemas al mismo tiempo, que la EAI ha tenido que avanzar dejando tecnologías antiguas a un lado y abordando cada vez más tecnologías mas modernas, es por eso que se han desarrollado distintas herramientas software para EAI como en el caso de WSO2[5] que basados en Apache Synapse[6], ofrece todos sus productos de código abierto enfocados en una arquitectura orientada a servicios (SOA) y destaca gracias a la aplicación de estándares de comunicación abierta como por ejemplo en su servidor de identidad que "admite capacidades de integración sencillas y fáciles con aplicaciones, almacenes de usuarios, directorios y otros sistemas de gestión de identidad." (Wimalasiri & Tharindu, 2019) y cuyo diseño "facilita varios tipos de escenarios de integración. El middleware de integración, como los ESB (Bus de Servicio Empresarial), facilita las características clave como comunicación, mediación, orquestación, transformación, QoS, seguridad, monitoreo, administración y manejo para

[3] SAP ERP, https://www.sap.com

[4] "Dancing Around EAI 'Bear Traps'", Gian Trotta, ebizQ , 12/15/2003
http://www.ebizq.net/topics/int_sbp/features/3463.html?page=1

[5] https://wso2.com/integration/

[6] https://synapse.apache.org/

satisfacer estas necesidades de integración dispares." (Indrasiri, 2016). Al estar basado en Apache Synapse, soporta distintos formatos para intercambio de contenido como texto plano, binarios, Hessian[7] y JSON[8] a través de distintos adaptadores para el transporte como "HTTP/S, Mail (POP3, IMAP, SMTP), JMS, TCP, UDP, VFS, SMS, XMPP and FIX." (Apache Software Foundation, 2019). WSO2 ofrece además un entorno de desarrollo gráfico para la integración con distintas herramientas y conectores; pero este, es una herramienta de escritorio y cada vez más orientado a servicios en la nube.

Otra de las herramientas más usada es IBM App Connect Enterprise "plataforma de integración para satisfacer las necesidades de las empresas que necesitan aprovechar las oportunidades digitales mediante el uso de tecnologías en la nube para crear aplicaciones híbridas y arquitecturas de datos" (IBM United States, 2018). Esta incluye además un entorno de desarrollo para escritorio (App Connect studio) en el que diseñar de forma gráfica los flujos dirigidos por eventos y una conectividad extensa a través de servicios en la nube, software como servicios (SaaS), plataformas en la nube y aplicaciones on-premise. Además, se integra en servicios HTTP con serialización en formato JSON a través de transferencias de estado representacional (RESTful) con múltiples opciones de conexión[9]. También, ofrece una interfaz web orientada a integrar servicios en la nube logrando así "velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio" (Gauchat, 2012), pero con la limitación de adquirir licencias costosas que pequeñas y medianas empresas que aún se encuentran entre las "muchas marcas [que] siguen sin tener presencia digital" (Rattinger, 2013), y dejan de lado también aquellos sistemas que se integran dentro de una red intranet.

Existen también otras alternativas como Software AG webMethods[10] que usa servicios web para conectar aplicaciones a través de internet, TIBCO Cloud Integration[11] con integración en la nube sin necesidad de software o hardware adicional, Azure Service Bus[12] como servicio de mensajería entre aplicaciones que permite además crear flujos de trabajo con topologías complejas en la nube, y el ya mencionado Apache Synapse, entre otros.

Estas herramientas hacen uso del diseño de flujos que es muy semejante a la notación y modelos de procesos de negocio (BPMN) permitiendo que al ser estos presentados (en por ejemplo una junta administrativa en la que los jefes de departamentos no son técnicos), todos entiendan una visión mucho más detallada del uso de los programas informáticos a lo largo de cada proceso de negocio y del funcionamiento de estos con relación a estos procesos de negocio y en general poder mejorar la toma de decisiones a través de una visión más completa. Pero que

---

[7] http://hessian.caucho.com/

[8] https://json.org/

[9] Nigel Williams, Richard Gamblin, Rob Jones, IBM Redbooks (Julio de 2018). IBM Z Integration Guide for the Hybrid Cloud and API Economy. REDP-531902.

[10] https://www.softwareag.com/corporate/products/webmethods_integration/default

[11] https://cloud.tibco.com/

[12] https://azure.microsoft.com/es-es/services/service-bus/

sin embargo, dejan de lado el aprovechamiento de tecnologías web en sistemas que no están expuestos a internet.

Propuesta

Este proyecto intenta iniciar el desarrollo de una aplicación web simple a comparación de los entornos de desarrollo de escritorio existentes, pero que cuente con las características ya ofrecidas por aplicaciones como IBM App Connect y el integrador WSO2 en referencia a la gran variedad de adaptadores para el transporte de datos y en el uso de representaciones graficas para los flujos sin la necesidad de tener todos los sistemas expuestos en internet. De esta manera, aportaría el uso de tecnologías web y graficas para la representación y gestión en la integración de aplicaciones a sistemas conectados en intranets o que funcionan dentro de una misma máquina. Este proyecto podría combinarse también con soluciones de código abierto como las ofrecidas por WSO2 con el fin de seguir expandiendo la disciplina EAI o bien tornándose en un conector de los servicios expuestos en internet a los no expuestos, en intranets.

En referencia a las tecnologías implicadas en este proyecto, se hará uso de la programación reactiva[13] en Java (versión 12), enfocada en el flujo de datos de manera asíncrona aprovechando las ventajas que brinda el marco de trabajo para java: Spring Boot[14]. Este último permite construir aplicaciones web siguiendo el patrón Modelo-Vista-Controlador (MVC) de manera "escalable, fácil de desarrollar y de mantener" (Yates, Ladd, Deinum, Serneels, & Vanfleteren, 2012) y cuenta además con otros marcos adicionales para la seguridad como Spring Security el cuál es un "poderoso y flexible marco de trabajo para la seguridad en aplicaciones web basadas en java" (Reddy, 2017) y facilidades para construir servicios REST como el Spring Data REST con el que exponer repositorios de datos como REST. Para el almacenamiento de datos, se usará repositorios compatibles con Spring Boot como es el caso de MongoDB[15] (versión 4.2), un sistema de bases de datos NoSQL orientado a documentos que se almacenan en formato BSON (similar a JSON), ideal para servicios REST.

Además, se hará uso del marco de trabajo para construcción de webs con diseño adaptable Bootstrap[16] (versión 4.3), cuya "combinación de HTML, CSS y JavaScript hace fácil el desarrollo de sitios web robustos sin necesidad de gran cantidad de código" (Spurlock, 2013) y PixiJS[17] (versión 5.3.1), un "extremadamente rápido renderizador de Sprites en 2D que ayuda a mostrar, animar y manejar gráficos interactivos" (Spuy, 2015) que es muy útil para el desarrollo de la herramienta de diseño de flujos de trabajo. Además, con el fin de empaquetar los distintos módulos en que pueda dividirse el desarrollo web, se usará Webpack[18] (versión 4.39.3) que

[13] https://www.reactivemanifesto.org/

[14] https://spring.io/projects/spring-boot

[15] https://www.mongodb.com/

[16] https://getbootstrap.com/

[17] https://www.pixijs.com/

[18] https://webpack.js.org/

requiere para su uso NodeJS[19] (versión 10.16.3), un entorno de ejecución para el lenguaje de programación interpretado JavaScript creado a partir del motor de JavaScript V8 de Chrome que "compila y ejecuta código JavaScript dentro de una VM (Máquina Virtual)" (Pasquali, 2013).

1.3. OBJETIVOS

A continuación, se listan los objetivos de este proyecto:

- Iniciar el desarrollo de un software capaz de ejecutar, modelar y simplificar la integración de distintos programas informáticos a través de un servicio web en el cual diseñar y configurar flujos de trabajo, y de un módulo que ejecute estos flujos de trabajo con el fin de alcanzar pequeñas y medianas empresas que no poseen presupuestos grandes para la contratación de licencias.
- Desarrollo de la interfaz con la que diseñar los flujos de trabajo debido a su uso como herramienta para solucionar este tipo de problemas en la organización. Así, se propone también hacer uso de tecnologías web de desarrollo de videojuegos con el fin de aprovechar la interactividad entre los usuarios y el sistema.
- Acercar al alumno del grado en Ingeniería de Software en la aplicación de los conocimientos en ingeniería y seguridad web con el desarrollo de un sistema web seguro que pueda estar al nivel de sistemas web usados en organizaciones y tomando en cuenta el diseño web adaptable. Así, este podrá también acercarse al desarrollo de aplicaciones web empresariales para enriquecer su formación práctica.
- Poner en práctica conceptos básicos de sistemas operativos como sincronización entre procesos, control de flujos, señales y construcción de procesos debido a que la base Arnion en el proyecto tomará en cuenta la interacción entre programas que se ejecutan en una misma máquina y que no toma en cuenta arquitecturas de sistemas más complejos. Se limita también a comunicar estos procesos usando la entrada/salida estándar y sockets orientados a la conexión por la complejidad del proyecto.

1.4. ESTRUCTURA DEL DOCUMENTO

El contenido de esta memoria está organizado principalmente en el capítulo 2. Sistema OACore que contiene la sección 2.1. del diseño de la solución propuesta pero que se continua a lo largo de las siguientes secciones. La sección 2.2. contiene además detalles de la implementación para la ejecución de flujos de trabajo. Las secciones 2.3. y 2.4. reúnen consideraciones adicionales en la implementación para la definición de enlaces y flujos. Finalmente, las secciones 2.5. y 2.6. detallan además la implementación del servicio web y el diseño de flujos de trabajo.

[19] https://nodejs.org/es/

2. Sistema OACore

El sistema OACore debe entonces lograr integrar distintas aplicaciones que se encuentren dentro de una misma red que no necesariamente sea internet. Así al analizar las diferentes funcionalidades o características que soluciones actuales ofrecen, se han identificado distintos requisitos mínimos que debe satisfacer:

- El sistema permitirá iniciar la ejecución de programas informáticos con independencia del sistema operativo en el que se ejecute.

  Justificación: Las aplicaciones empresariales varían en el uso de sistemas operativos "pues en el mercado existen algunos que funcionan solo con sistemas operativos Microsoft Windows o con algún distribuidor específico de Linux, como Red Hat" (Reynaldo Guevara, 2018)

- El sistema brindará la posibilidad de gestionar los distintos programas informáticos a través de un servicio web.

  Justificación: Se aprovechará las ventajas del uso de tecnologías web por su accesibilidad y adaptabilidad.

- El sistema permitirá la autenticación de usuarios para brindar seguridad al servicio web.

  Justificación: Es un hecho común los ataques para burlar la seguridad web tanto en internet como en intranets y mucho más con relación a entornos empresariales.

- El sistema posibilitará el diseño gráfico por medio de tecnologías web que modelen la integración entre los distintos aplicativos.

  Justificación: Con el fin de soportar la toma de decisiones empresariales a través de la comprensión del uso de programas informáticos en la empresa, y simplificar la integración entre estos, herramientas como IBM App Connect y otras, usan diseños de flujos para modelar la relación entre las aplicaciones.

- El sistema soportará distintos formatos de transferencia de datos con los cuales integrar los distintos programas informáticos.

  Justificación: Las aplicaciones pueden implementar uno o varios de los protocolos y estándares para la transferencia de datos existentes en la actualidad.

- El sistema permitirá la conversión entre distintos formatos de transferencia de datos con el fin de soportar la compatibilidad entre las aplicaciones.

  Justificación: No todos los métodos de transferencia de datos funcionan de la misma manera, por ejemplo, la transferencia de datos en pequeñas partes no es igual a la una transferencia datos de gran tamaño y su protocolo o configuración puede variar bastante.

Hay otras características que pueden identificarse como en el caso de la monitorización de ejecución de programas informáticos, el cálculo de estadísticas de uso y en el flujo de datos y

muchas otras más que por ahora quedan fuera de este proyecto debido a la limitación de los recursos involucrados con relación al tiempo y número de personas.

Para cumplir con los requisitos ya mencionados es posible utilizar los productos de código abierto de WSO2 y agregar el diseño de flujos de trabajo en web logrando ampliar la funcionalidad que estos ya ofrecen y las cualidades por las que destaca en su variedad de adaptadores para el transporte de datos, pero con el fin de acercar al alumno al desarrollo de aplicaciones empresariales y ampliar su visión de la ingeniería de software, se ha decidido desarrollar este sistema desde cero y formar así una alternativa que no requiera el uso de licencias costosas más pueda ser continuada dentro de las aplicaciones de EAI. Por esto, se presenta a continuación el diseño y la implementación de la solución ofrecida en el sistema OACore.

2.1. DISEÑO

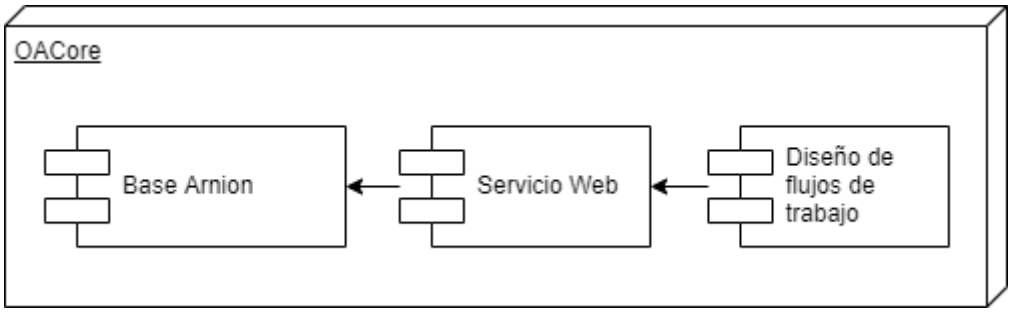

Ilustración 1: Componentes del proyecto OACore

Este proyecto OACore, se ha dividido en 3 módulos fundamentales (Ilustración 1: Componentes del proyecto OACore): el desarrollo de la base Arnion encargada de la ejecución de procesos, la definición de procesos y la integración entre estos; el servicio web que ofrece la autenticación de usuarios y expone el diseño de flujos de trabajo; y finalmente, el módulo de diseño de flujos de trabajo.

En primer lugar, la base Arnion consiste en el desarrollo de un módulo de software usando el lenguaje de programación java, encargado de planificar la ejecución de programas informáticos, enlazar los recursos necesarios para la integración y finalmente realizar el enlazado entre los procesos o programas en ejecución.

En segundo lugar, el diseño de flujos de trabajo en la web es el módulo de software que sirve el servicio web del tercer módulo en el cual diseñar y gestionar los flujos de trabajo mediante los cuales modelar la ejecución de los distintos programas informáticos y la integración entre estos.

Así, en la base Arnion, la ejecución de programas informáticos crea instancias llamadas procesos que en la base Arnion se han usado como modelo y definición para que una misma aplicación pueda ejecutarse en varias etapas de un flujo de secuencia y de diferentes formas o bien asignando diferentes parámetros para su ejecución; estos parámetros de ejecución pueden ser asignados en la misma llamada al sistema (mediante la cual se crea el proceso a partir de la

aplicación) o usando módulos que preparen los recursos necesarios para su ejecución, estos últimos se han llamado binders en este proyecto.

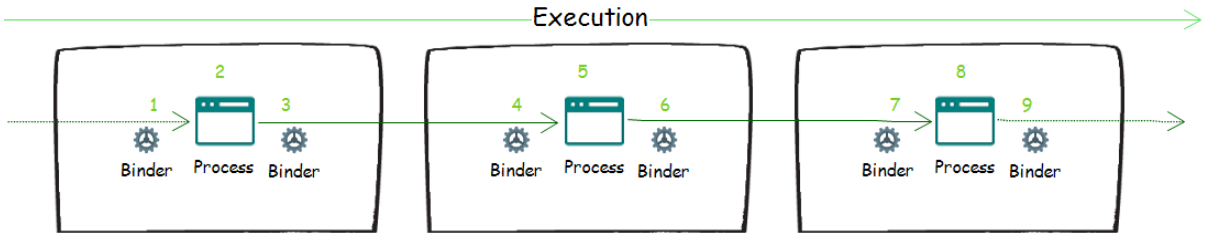

Ilustración 2: Flujo de ejecución de procesos y binders

También, estos módulos: binders, se encargarían de la integración o de la realización de la comunicación entre los procesos brindando así mayor cohesión en la preparación de recursos para la ejecución de un programa como en la transferencia o enlazado de datos en tiempo de ejecución. La relación que guardan estos puede verse en la Ilustración 2: Flujo de ejecución de procesos y binders, en donde se puede ver la secuencia de ejecución que sigue el flujo, así como el uso de dos binders para la modelización de la integración entre dos procesos. En primer lugar, el binder inicial es asociado al proceso y este prepara todos los recursos como copia de ficheros, recibir datos de otros binders y demás, necesarios para que el proceso pueda ejecutarse. Luego, el proceso en sí es ejecutado (sin necesariamente finalizar su ejecución), es decir, se realiza la llamada al sistema siguiendo la definición de proceso para ejecutar la aplicación. Finalmente, el segundo binder es ejecutado y este se encarga de extraer o recopilar todos los datos que se necesiten enviar al siguiente proceso en el flujo de ejecución.

Por otra parte, siguiendo un paradigma orientado a objetos, la representación de los datos que se transfieren entre procesos ha sido encapsulado en una clase llamada “Transfer” y estos transferes deben contener todo tipo de datos (validos en Java) necesarios para la ejecución de procesos, desde simples cadenas de caracteres, números, hasta instancias de clases y ficheros abiertos.

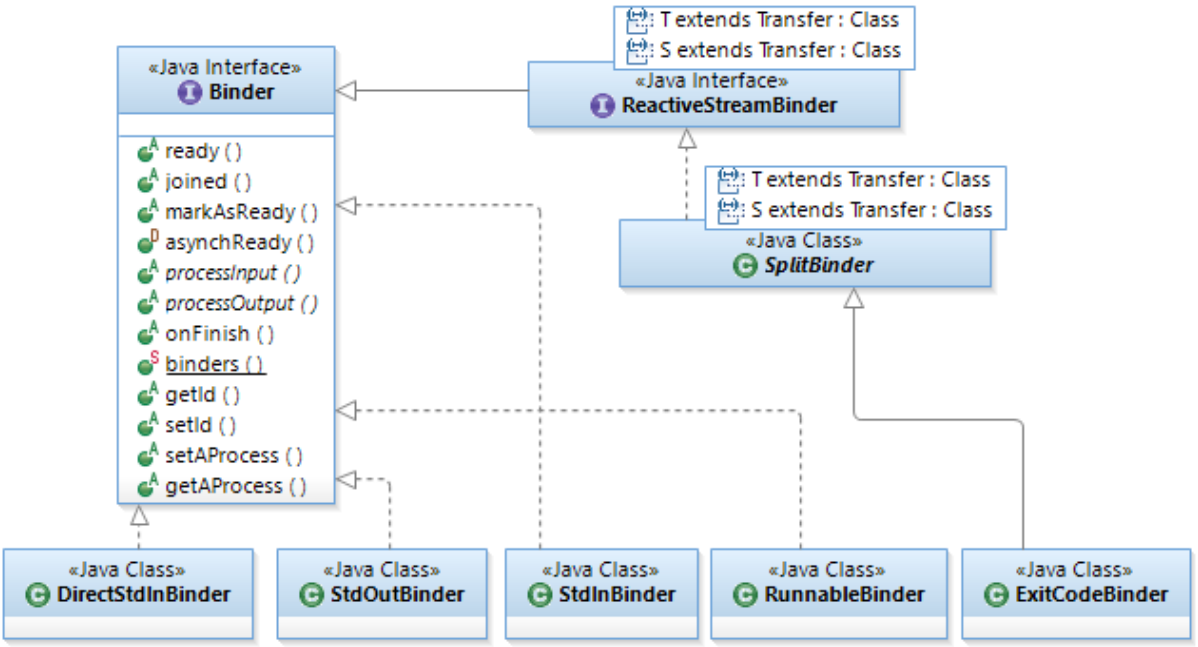

Ilustración 3: Diagrama de clases que implementan la interfaz Binder

Además, al aplicar los principios de la programación reactiva en el uso de los binders, el flujo de mensajes o transferes necesitan de publicadores (o publishers por su nombre en inglés) que "produzcan" o publiquen estos transferes y de suscriptores (o subscribers) que recibirían finalmente como destinatarios de estos transferes. De esta manera, el flujo de transfer a lo largo del flujo de ejecución y por medio de los binders hace uso necesario de publishers y subscribers más allá de simplemente copiar o pasar referencias a parámetros propios en la definición de procesos para así poder hacer uso de una arquitectura en la que varios procesos puedan recibir un mismo mensaje (según el paradigma de la programación reactiva). Es por lo que se ha definido las interfaces de "APublisher" (Ilustración 4: Diagrama de clases que son publicadoras) y "ASubscriber" (Ilustración 5: Diagrama de clases que son suscriptoras) que serán implementados por clases que cumplan el rol de publicadores y/o suscriptores de transferes que bien podrían ser los mismos binders (Ilustración 3: Diagrama de clases que implementan la interfaz Binder); también, se ha definido clases publicadoras y suscriptoras orientadas a la conexión con sockets (como en el caso de "SocketListenerPublisher", "SocketServerPublisher", "SocketSubscriber" y "SocketServerSubscriber") y una clase suscriptora que permita almacenar en la base de datos llamada "TransferStoreSubscriber". A la hora de enlazar procesos, es posible enviar un publicador disponible tras ejecutar el segundo binder o binder de origen del enlace al siguiente proceso, es decir, a su primer binder de preparación o binder de destino de enlace para que este o su siguiente binder puedan suscribirse y recibir transferes útiles para su ejecución o necesarios para la ejecución de la aplicación. Así, la suscripción se hace de proceso a proceso a través de los binders y a lo largo de la ejecución en caso de que esta suscripción a un publicador esté disponible.

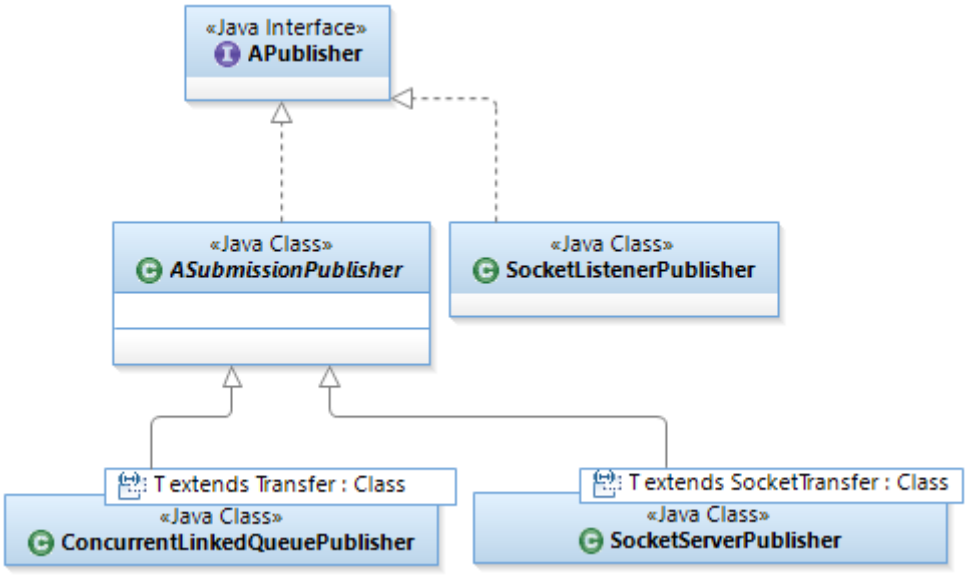

Ilustración 4: Diagrama de clases que son publicadoras

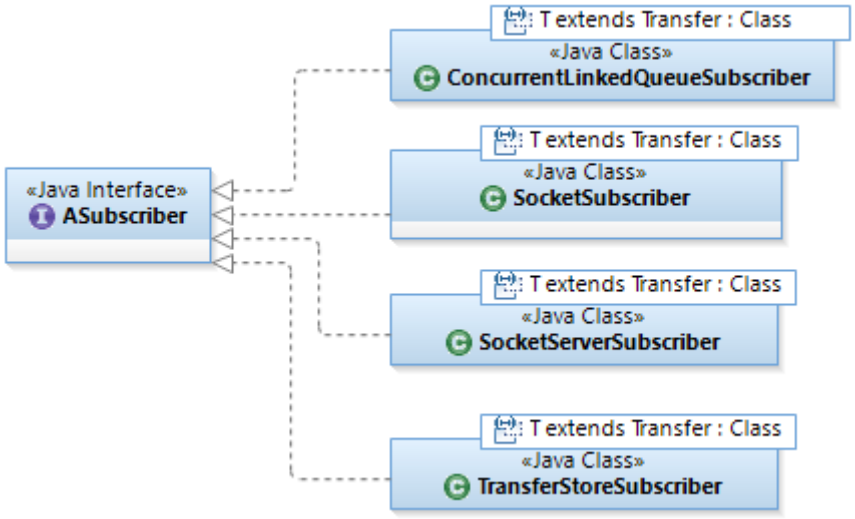

Ilustración 5: Diagrama de clases que son suscriptoras

Así mismo, en la Ilustración 3: Diagrama de clases que implementan la interfaz Binder, se puede ver la interfaz "Binder" y las distintas clases que se han definido como binders mediante los cuales implementar la transferencia de datos entre procesos usando la entrada y salida estándar y el uso de sockets orientados a conexión. Entre los métodos que se proponen para estos, se encuentra el método "ready" que permite consultar en el caso de que sea un binder principal que prepare recursos, si ya ha preparado todo. La clase "DirectStdInBinder" enlaza directamente las entradas y salidas de dos procesos entre sí, asignando la salida estándar de un proceso a la entrada del otro, mientras que las clases "StdOutBinder" y "StdInBinder" separan estas dos, permitiendo que puedan ser transferidas a otros binders. Además, el binder con clase "RunnableBinder" ofrece la posibilidad de asociar un publicador, suscriptor o binder que contenga en si una lógica que ejecutar para la realización del enlace, por ejemplo, abrir una conexión. Por último, la interfaz "ReactiveStreamBinder" junto a su implementación de clase

abstracta “SplitBinder”, permite definir aquellos binders que en si mismos son suscriptores y publicadores al mismo tiempo, como es el caso del “ExitCodeBinder” que recibe el código de salida del proceso anterior y envía el de su proceso asociado al siguiente proceso.

También, con el fin de asociar la definición de un proceso con el proceso de sistema generado por la llamada a este, se ha creado la clase “ProcessExecutor” que sirve para asociar el proceso con el binder principal de preparación. A toda esta definición del flujo de ejecución o flujo de trabajo se ha denominado “Workflow” y se ha definido de igual manera una clase que los agrupe (Ilustración 6: Diagrama de clases entre workflow, procesos, binders y otros.).

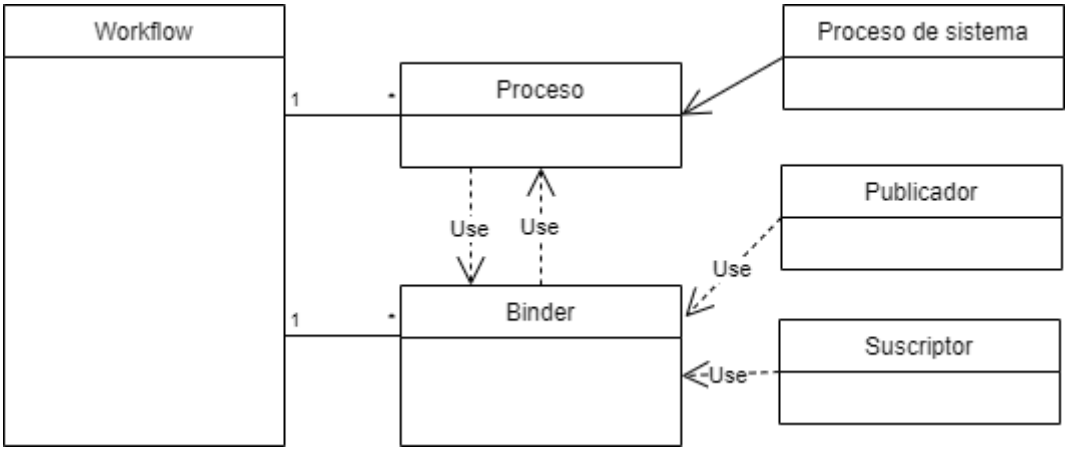

Ilustración 6: Diagrama de clases entre workflow, procesos, binders y otros.

El segundo modulo, el servicio web, sacando provecho del marco de trabajo Spring Boot, aplicará el patrón MVC en el que se definen tres capas fundamentales: Modelo-Vista-Controlador. En la capa de modelo se podrá hacer uso de repositorios con los que almacenar las definiciones de proceso, binders y transferes en la base de datos servida por MongoDB, para los cuales debe existir conversores entre objetos y documentos de MongoDB (como por ejemplo para los procesos de la clase nombrada “AProcess”,

Ilustración 7: Diagrama de clases de proceso en la capa de modelo del módulo web). En la capa de vista se hará uso de Bootstrap con diseños simples pero que brinden adaptabilidad a los distintos dispositivos (simples desde el punto de vista que el diseño de flujos de trabajo es el contenido principal y no otras secciones que podría contener la web) haciendo uso del modulo de Spring Security para autenticar usuarios y garantizar así la seguridad en el uso de la aplicación. La ultima capa, la del controlador, consistirá en los controladores para el flujo de navegación web (registro de usuarios y zona protegida para usuarios con privilegios-cualidad ofrecida por el módulo de Spring Security) y de un controlador REST que permita crear, consultar, modificar y eliminar los distintos objetos de un workflow (procesos, binders, publicadores, suscriptores) para el diseño de flujos de trabajo.

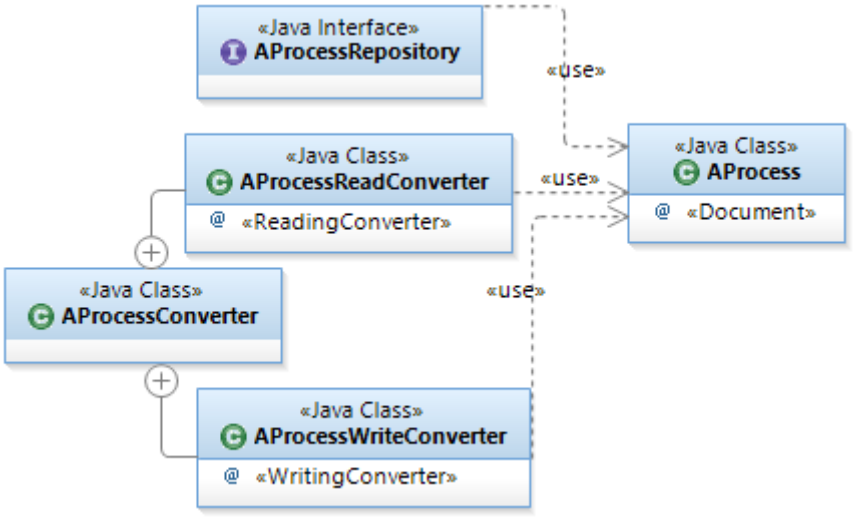

Ilustración 7: Diagrama de clases de proceso en la capa de modelo del módulo web

El tercer y último módulo consiste en la interfaz gráfica para diseñar flujos de trabajo, esta, hará uso del marco PixiJS con el cual diseñar cada workflow. Esta debe permitir agregar nuevas definiciones de proceso, enlazar estos procesos y definir los tipos de binder a usar (incluyendo el uso de suscriptores y publicadores). Todo esto a través de la interacción del controlador REST mencionado antes y mediante el cual guardar el diseño realizado por el usuario privilegiado (Ilustración 8: Flujo de navegación de usuario en la web y la interacción con el controlador web REST).

Detalles adicionales del diseño se han dejado para el momento de la implementación debido a la investigación de cada tecnología involucrada y que puede ser esencial para mantener una visión más global de lo que puede llegar a ser el proyecto.

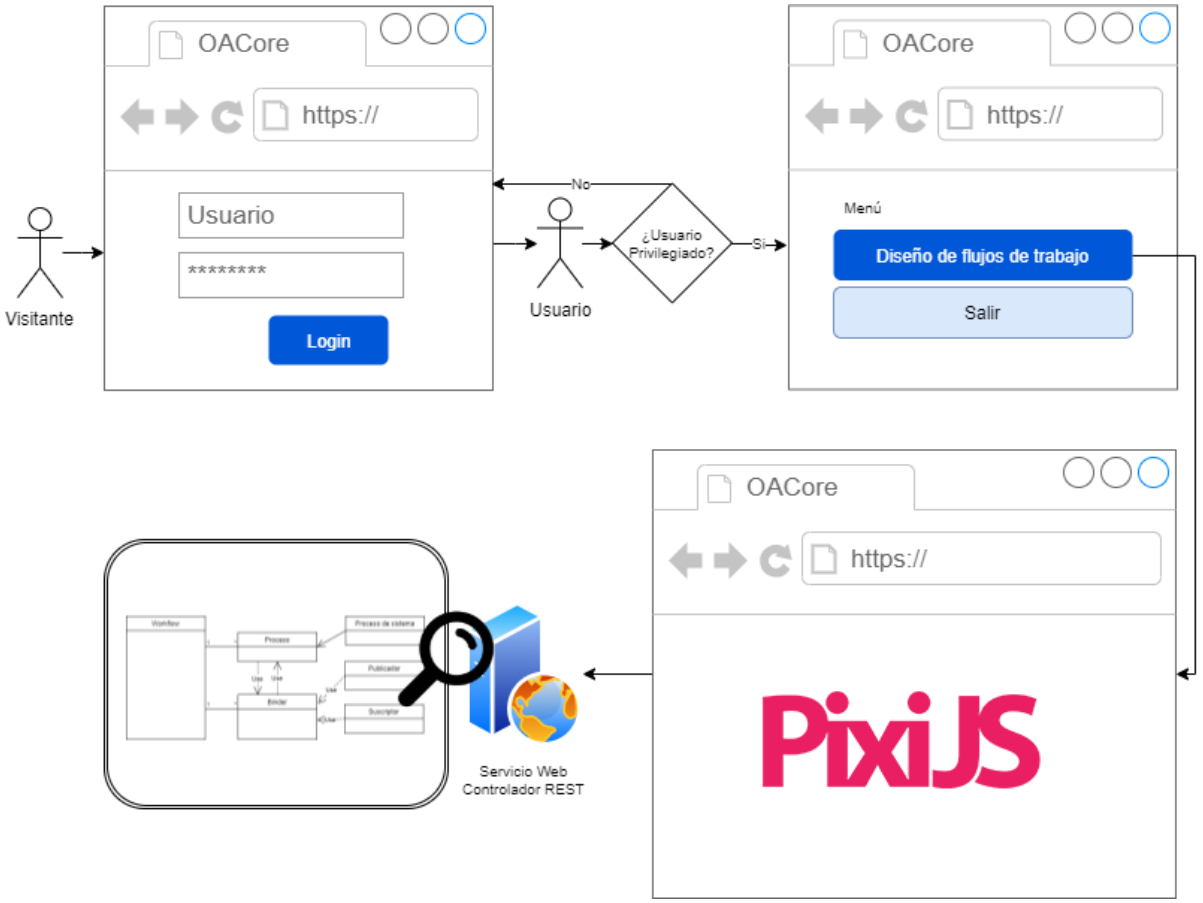

Ilustración 8: Flujo de navegación de usuario en la web y la interacción con el controlador web REST

2.2. DETALLES ADICIONALES DE EJECUCIÓN DE PROCESOS

A partir del diseño para la ejecución de las definiciones de proceso, se ha realizado la implementación haciendo uso del "ProcessBuilder" de Java mediante el cual realizar la llamada al sistema para la construcción de procesos. A este, se le pueden establecer distintas propiedades que se han incluido como propiedades de la definición de procesos:

- Comando que ejecutar, consiste en una cadena de caracteres que contiene la ruta al programa que se ejecutara y los argumentos que se pasaran a este.
- Variables de entorno a modificar, siendo pares clave y valor de variables del sistema a modificar que utiliza el proceso en su ejecución. Por ejemplo, la variable "PATH" en el sistema operativo Windows contiene la ruta por defecto en donde localizar los ejecutables del sistema.
- Directorio de trabajo, en el que el proceso encontrará los archivos que necesite para su ejecución.
- Destino de la entrada y salida estándar de proceso.

Asimismo, gracias a que la construcción de procesos se realizará en el contexto de cada flujo de ejecución que invoca un usuario en específico, es necesario que esta sea realizada de forma asíncrona, para ello se ha hecho uso de hilos de ejecución a través de las clases "ExecutorService"

(servicios de ejecución) ofrecidas para la programación concurrente en java. Estos, proveen métodos para manejar la terminación y la gestión del progreso de tareas asíncronas a través de objetos "Future" (referidos en este documento como futuros) que representan resultados de ejecuciones asíncronas. El uso de estos objetos "Future" se ha aprovechado también para obtener las propiedades de los procesos que se ejecutan (propiedades como la asociación del proceso de sistema con la definición del proceso en la clase "ProcessExecutionDetails"). Las propiedades que se obtienen en tiempo de ejecución o tras haberse ejecutado un proceso son:

- ¿Se ha ejecutado?: Obteniéndose como un futuro booleano, representa si se ha ejecutado el proceso (y ha terminado su ejecución).
- Proceso del sistema: Haciendo uso de futuros, se obtiene el proceso del sistema nativo para poder usar su entrada/salida estándar, consultar el estado, o realizar acciones directamente sobre este, por ejemplo, en el envío de señales.
- Código de retorno (o exit-code): Devuelto tras finalizar la ejecución de una aplicación, se asocia a un "CompletableFuture"; un tipo de futuro (clase) que puede completarse a través de llamadas a métodos de esta.

Además, otras propiedades son asociadas al proceso, como el binder principal que preparara la entrada y los recursos necesarios para la ejecución del proceso.

Por otro lado, se ha definido una clase adicional llamada "ProcessExecutor" con una instancia única en la que es posible enviar nuevas tareas (solicitudes de crear procesos o bien binders en sí mismos) a un "ExecutorService" especifico, de esta manera, dentro de un "Workflow" puede asociarse distintos servicios de ejecución de distintos tipos, ya que java ofrece a través de la clase "Executors" la posibilidad de crear desde hilos únicos de ejecución hasta grupos de hilos en los que ejecutar tareas tras un tiempo determinado. También, se ha hecho uso de la clase "Function" de java para usar como llamada de retorno tras iniciar la ejecución de un proceso. Así, esta llamada de retorno permite saber cuando asociar todas aquellas propiedades que se obtienen en tiempo de ejecución o tras la ejecución de un proceso como en el caso del código de retorno.

2.3. BINDERS

En el flujo de datos a través de binders entre procesos, los llamados transferes definen el tipo de datos del flujo o transfieren objetos que son necesarios para la ejecución de procesos o binders posteriores. Estos se ha decidido que contengan además una marca de tiempo de creación con la cual mantener un registro del origen del dato o recurso. Por eso, se han implementado en las siguientes clases.

- "ServerSocketTransfer": Transfer cuyo dato principal consiste en un "ServerSocket" de java que sirve un socket por el que enviar datos, es decir, ofrece un recurso.
- "SocketTransfer": Ofrece un recurso cuyo tipo de dato principal es un socket en el que la conexión ya ha sido establecida.
- "StreamTransfer": Usado para el intercambio de flujos de entrada "InputStream" de java, contiene en concreto dos flujos de entrada, la entrada estándar de proceso y la entrada de error estándar de proceso.
- "StringTransfer": Almacena una cadena de caracteres.

- “StringCollectionTransfer”: Almacena conjuntos de cadenas de caracteres.
- “IntegerTransfer”: Almacena un numero “Integer” como dato.

Así, a estos transferes se les ha añadido un método llamado “parse” desde el cual des serializar objetos de estos a partir de texto en formato JSON. Además, se ha implementado la interfaz “Transfer” como extensión de la interfaz de MongoDB para Spring Boot llamada “Document” con el fin de que estos transferes puedan ser almacenados como documentos de MongoDB en los repositorios de la base de datos. Sin embargo, con el fin de brindar la posibilidad a que un binder pueda almacenar aquellos transferes que no haya podido enviar (o publicar) en la base de datos, se ha creado dos clases llamadas “TransferService” y “TransferStore” usando una lista enlazada “LinkedList” de java. Estas dos formas de persistencia de transferes tienen sentido cuando se trata de gestionar y mantener la ejecución de un workflow, pero en este trabajo se ha limitado a la definición y ejecución simple sin gestionar en profundidad el estado de ejecución de cada proceso o el envío de transferes entre binders y procesos.

Por otro lado, gracias a que los binders tienen suficiente lógica en si mismos, se ha decidido que extiendan de la clase java “Runnable” con el fin de que estos (así como también en algunos publicadores y suscriptores) puedan ser ejecutados usando binders del tipo “RunnableBinder” o bien a través del “ProcessExecutor”. En relación con estos publicadores se detallan a continuación algunas de las implementaciones (Ilustración 4: Diagrama de clases que son publicadoras). Estas han hecho uso de una cola enlazada para el almacenamiento de los suscriptores.

- “ASubmissionPublisher”: Clase abstracta que extiende a la clase java “SubmissionPublisher” para poder definir el comportamiento de enviar transferes que puedan ser distribuidos entre sus suscriptores.
- “ConcurrentlinkedQueuePublisher”: Hace uso de una cola enlazada para el registro de los suscriptores.
- “SocketListenerPublisher”: Tras la apertura de una conexión en un socket que es pasado a este, esta clase hace uso de un buffer para leer cada mensaje recibido y lo almacena como un “StringTransfer” que es enviado a cada suscriptor.
- “SocketServerPublisher”: Funcionando como un “SubmissionPublisher”, éste envía los transferes a cada suscriptor y también a través del socket de servidor.

Además, con relación a la suscripción de transferes, se han implementado también suscriptores que almacenan en colas concurrentes aquellos transferes recibidos. En compatibilidad con los publicadores, estos suscriptores (Ilustración 5: Diagrama de clases que son suscriptoras) reciben transferes del mismo tipo que son enviados por los publicadores, es decir, transferes con datos o con recursos como “SocketServerSubscriber” y “SocketSubscriber”. Cabe mencionar, que el “TransferStoreSubscriber” es usado para almacenar en la base de datos aquellos transferes que le sean enviados.

A continuación, se detallan las implementaciones realizadas de los binders (Ilustración 3: Diagrama de clases que implementan la interfaz Binder).

2.3.1. DirectStdInBinder

Este binder ofrece la posibilidad de enviar (publicador) dos flujos de entrada: la entrada estándar de proceso y la entrada de error estándar de proceso a partir de un "ProcessExecutionDetails" asociado y de escribir en la salida estándar de proceso. Estos dos primeros flujos se envían usando un "StreamTransfer". Pero el tercero se usa solamente al recibir el "StreamTransfer". Es decir, recibe una entrada estándar de proceso que se enlaza a una salida estándar de proceso, logrando así que dos procesos puedan intercambiar distintos datos sin usar más que un solo "StreamTransfer" mediante el cual establecer la comunicación.

Los primeros dos flujos de entrada se abren cuando se ejecuta el binder, luego este envía el "StreamTransfer" y en caso de que este (u otra instancia de "DirectStdInBinder") lo reciba, lo lee y envía los bytes directamente al flujo de salida. En la implementación se ha limitado solo a la lectura de la entrada estándar de proceso mientras que la entrada de error estándar de proceso no se toma en cuenta, pero bien podría enlazarse a este u otro proceso. Este binder es marcado como listo (ready) al recibir (suscriptor) el "StreamTransfer".

2.3.2. SplitBinder

Tras realizar la creación del "DirectStdInBinder", se ha definido la interfaz "ReactiveStreamBinder" para recoger la funcionalidad de flujos reactivos en binders por sí mismos, y es esta última la que se implementa en la clase abstracta "SplitBinder". Esta clase, permite entonces vincular una clase de suscriptor y una de publicador que no necesariamente van vinculados entre sí y que tampoco necesitan la misma implementación de transfer. De esta manera, la funcionalidad de suscripción y publicación de mensajes se realiza en este binder al mismo tiempo. Una implementación final de esta clase es el "ExitCodeBinder".

2.3.3. ExitCodeBinder

Con el fin de recibir el código de retorno con el cual un proceso de sistema termina, se ha creado la clase "ExitCodeBinder". Este binder pide un "IntegerTransfer" al publicador asociado en nombre del suscriptor asociado y recibe el código de retorno de la ejecución del programa asociado y lo publica a través del publicador. Así, hace uso del futuro devuelto por el "ProcessExecutionDetails" asociado. Finalmente, tras haberse publicado el código de retorno, el publicador es cerrado debido a que la ejecución del proceso ocurre una sola vez en el enlace. Este binder además está listo justo tras recibir el "IntegerTransfer" del publicador al que se suscribió el suscriptor.

2.3.4. RunnableBinder

Debido a que algunas implementaciones de publicadores o suscriptores tienen funcionalidad en sí mismos e implementan la interfaz java "Runnable", estos pueden ser ejecutados a través de los "Executors" (de java) siendo encapsulados en un binder. Así, se puede asociar y ejecutar un "Runnable" en esta misma clase. Consultar si está listo este binder no tienen sentido y por eso devuelve siempre "true".

2.3.5. StdInBinder

Este binder extrae el flujo de entrada estándar de proceso y el flujo de entrada de error estándar de proceso, de un proceso que es asociado. Tras realizar la extracción, cada flujo es leído a través de buffers (usando un "BufferedReader" de java) y enviado a través del publicador correspondiente que se ha asociado, es decir, un publicador para cada flujo. La lectura de los buffers se almacena en transferes "StringTransfer" que son enviados al correspondiente publicador. Al usar futuros, la ejecución de este binder queda bloqueada en espera del proceso del sistema del "ProcessExecutionDetails" asociado. Con el fin de evitar un bloqueo doble al consultar si esta listo, se requiere que este binder sea forzado a estar listo a través del método "markAsReady", de otra manera, el consultar este (mientras espera que el futuro del proceso de sistema se complete), llevaría a que todo el hilo que consulta se bloquee también.

2.3.6. StdOutBinder

El "StdOutBinder" es en sí un suscriptor para que este pueda ser suscrito a cualquier publicador y enviar transferes (en este caso se permiten solo "StringTransfer"). Estos transferes se escriben en el flujo de salida estándar del proceso que se ha asociado y en su ejecución el flujo de salida estándar es obtenido, resultando así en un bloqueo del hilo de ejecución hasta que el futuro del proceso de sistema sea resuelto. Es de esperarse que errores de sincronización puedan ocurrir al intentar escribir en el flujo cuando aún este no se ha abierto, debido a que el futuro "CompletableFuture" del cual se obtiene el proceso de sistema funciona como cerrojo, pero no cuando el transfer es enviado al método a este binder desde otro hilo de ejecución. Por esto, para resolver este problema se ha utilizado un "TransferStore" en el que almacenar los transfers hasta que el flujo sea abierto y estos sean enviados.

2.4. WORKFLOW

Un workflow o flujo de trabajo (también referido como de ejecución) son un estudio para la tecnología de procesos en la secuencia de tareas o trabajos (que en este proyecto son los programas de computadora), el flujo que se realiza a través de estos y el control en cada una de las partes involucradas. Así, la clase "Workflow" (Ilustración 6: Diagrama de clases entre workflow, procesos, binders y otros.), contiene además las coordenadas 2D (dos ejes, dos dimensiones) en las que se posicionan las definiciones de proceso y los binders enlazados, necesarios para su representación grafica en el marco de trabajo PixiJS.

Además, es necesario almacenar el orden de los enlaces entre los procesos, para eso, se ha almacenado el identificador del objeto (ObjectID de documentos de MongoDB) a través de estructuras java "HashMap" que almacenan pares clave, valor. Así, a partir del enlace de cada proceso con los binders, se forma una estructura de la secuencia que sigue cada proceso. Es decir, de la forma: proceso, binder de origen, binder de destino, proceso, … (Ilustración 2: Flujo de ejecución de procesos y binders).

Con respecto a la implementación del almacenamiento de estos flujos en la base de datos usando los repositorios del marco Spring Boot, ha sido necesario registrar los distintos conversores entre documentos y objetos en un "Bean" (contenedor que administra objetos) que retorna una instancia de la clase "MongoCustomConversions" de Spring Boot. Así, cada uno de los

conversores extrae las propiedades de cada binder o proceso en cadenas de caracteres y las agrega al documento para que estos después puedan ser leídos y convertidos de nuevo en binders o procesos (igual ocurre con los suscriptores y publicadores). En el caso de convertir binders a documentos se han usado los siguientes nombres para sus propiedades:

- "_id": Identificador del binder.
- "binderType": Nombre de la clase del binder.
- "associatedProcess": Identificador del proceso asociado al enlace del binder.
- "processId": Identificador del proceso del cuál extraer datos, usado en los binders: "DirectStdInBinder", "ExitCodeBinder", "StdInBinder" y "StdOutBinder".
- "subscriberId": Identificador del suscriptor asociado al binder, usado en el binder "ExitCodeBinder".
- "publisherId": Identificador del publicador asociado al binder, usado en el binder "ExitCodeBinder".
- "runnableType": Tipo de objeto asociado al "RunnableBinder", puede ser: "Binder" para referirse a un binder, "APublisher" a un publicador, "ASubscriber" a un suscriptor o "Void" como ninguno o "null".
- "runnableId": Identificador del objeto asociado al "RunnableBinder".

Luego, para la conversión de suscriptores y publicadores a documentos se han usado los siguientes nombres para sus propiedades:

- "_id": Identificador del suscriptor.
- "subscriberType": Nombre de la clase del suscriptor.
- "data": Transferes sin enviar.
- "socket_port": Número del puerto al que comunicarse en el caso de usar sockets.
- "socket_host": Nombre del host al que comunicarse usando sockets.

También, para la definición de proceso se han utilizado nombres semejantes a sus propiedades. Así, se logra almacenar cada uno de estos contenidos en un flujo, pero además se han de establecer otros nombres adicionales que son asignados como propiedades especificas de los flujos como:

- "process": Colección de identificadores de procesos asociados al flujo.
- "executorServices": Nombres de los "ExecutorService" en los que se ejecutan procesos, binders, suscriptores y publicadores.
- "binder": Colección de identificadores de binders.
- "objCoords": Coordenadas 2D de los objetos del flujo.
- "executorAssigned": Estructura que relaciona los binders y procesos con "ExecutorService" en los que ser ejecutados.
- "binderProcessesOrig": Estructura con la relación entre binders de origen y procesos.
- "binderProcessesDest": Estructura con la relación entre binders de destino y procesos.

2.5. SERVICIO WEB

Usando el marco de trabajo de Spring Boot a partir de una plantilla básica[20] creada por Manuel Freire (profesor de ingeniería web de la Universidad Complutense de Madrid) y del proyecto TNoticias[21] creado por Adrian Sanchez (alumno que desarrolla este trabajo), se ha implementado el servicio web para la autenticación de usuarios y la exposición del diseño de flujos de trabajo con una configuración adicional para el modulo de Spring Security en la que se hace uso del filtro "spring-security-csrf-filter"[22] creado por Allan Ditzel (Director de ingenieria de software de DC Metro Area, US). Este filtro ha sido necesario al encontrarse con la protección contra ataques CSRF (Cross-site request forgery o falsificación de petición en sitios cruzados) del módulo de Spring Security que bloquea toda petición de creación o modificación como consultas HTTP (protocolo de transferencia de hipertexto en la web) del tipo POST, PUT y REMOVE, usados en servicios REST. Así, este filtro agrega en la cabecera de la consulta el token de autenticación del usuario, el cual es enviado en cada consulta para validar la petición. Esto es realmente útil para las consultas que realizara el diseño de flujos de trabajo al servidor.

Por otro lado, los detalles de usuario se han definido en la clase "User". Entre estos se encuentra la contraseña que utiliza la función hash criptográfica BCrypt [23] para su almacenamiento encriptado, nombre, apellido, dirección del avatar y una pregunta junto a una respuesta de seguridad para el restablecimiento de contraseña. Estos, han sido usados en la implementación de la vista (dentro del modelo-vista-controlador) a través del marco Bootstrap manteniendo un diseño adaptable.

El diseño ha sido implementado usando un diseño lineal en el que se muestra la barra o menú de navegación seguido del contenedor principal de la página y terminando con el contenedor de pie de página (Ilustración 9: Formulario de registro e inicio de sesión web.). Estos contenedores principales cambian en cada página como en el caso del formulario de registro, ajustes de usuario (Ilustración 10: Formulario web de ajustes de usuario.) y perfil de usuario (Ilustración 11: Formulario web de perfil de usuario.), con excepción de la página única del diseño de flujos de trabajo que no tiene contenido ni diseño sino que en esta se carga el marco de PixiJS.

[20] https://github.com/manuel-freire/iw-base

[21] https://github.com/AdrianSh/TNoticias

[22] https://github.com/aditzel/spring-security-csrf-filter

[23] http://www.mindrot.org/projects/jBCrypt/

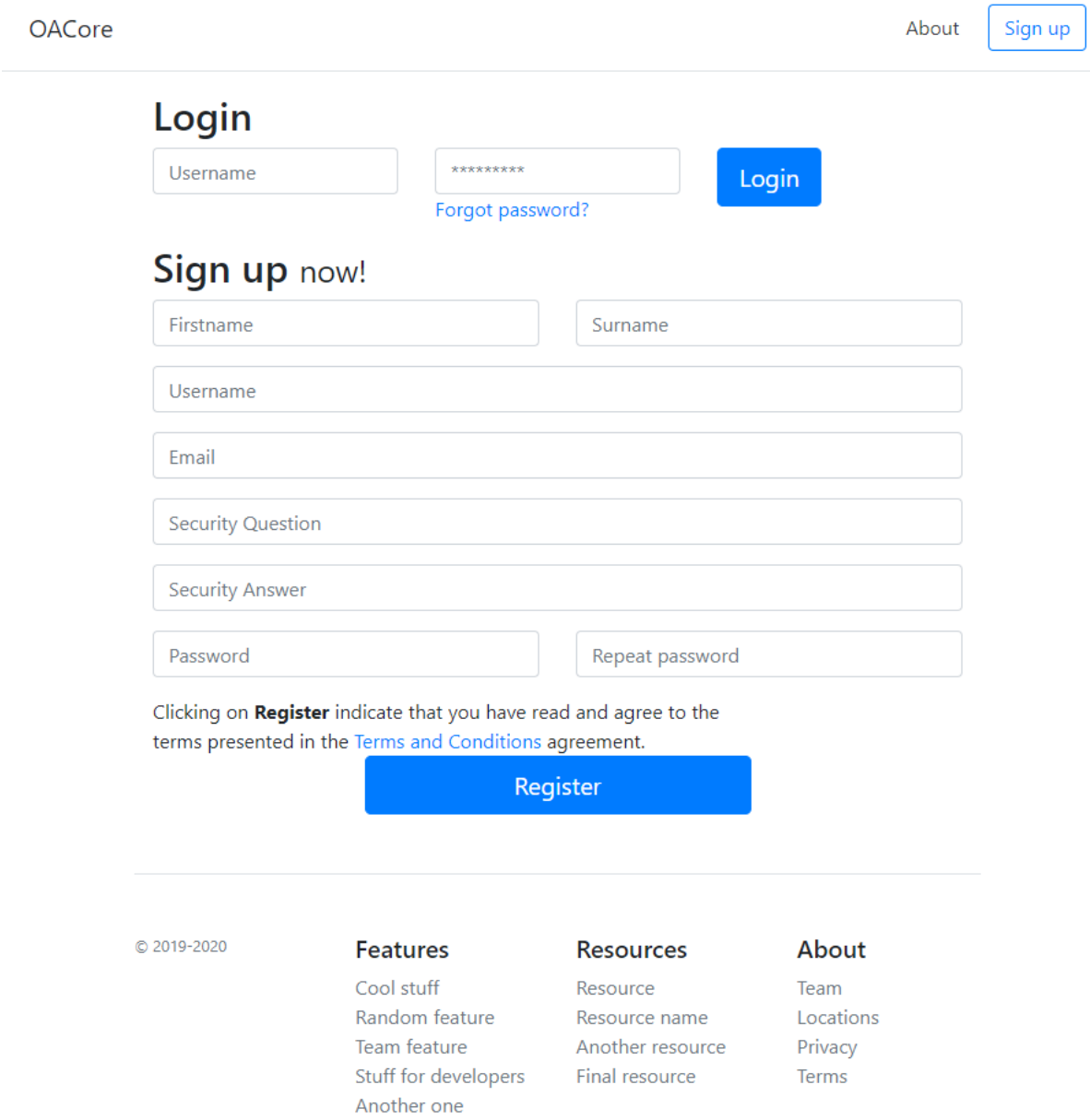

Ilustración 9: Formulario de registro e inicio de sesión web.

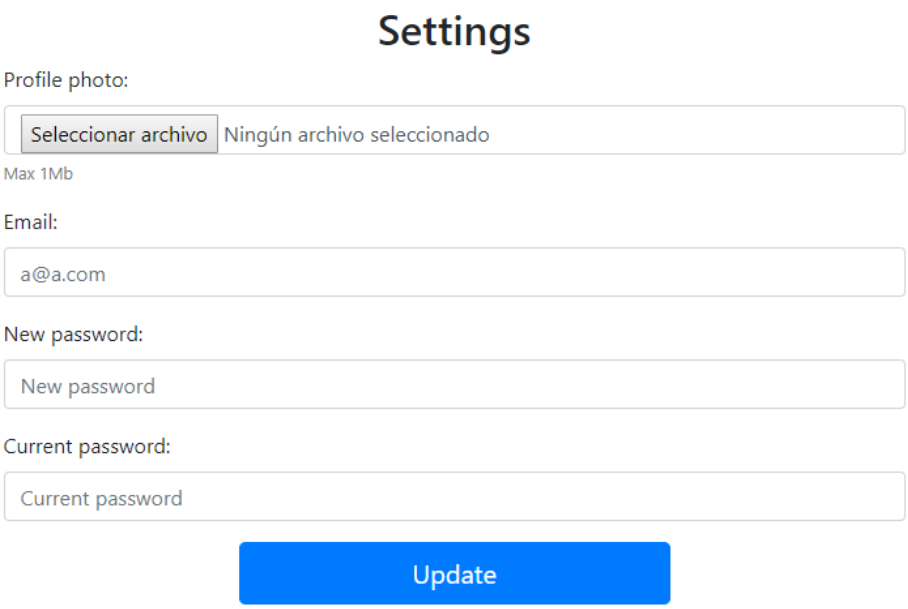

Ilustración 10: Formulario web de ajustes de usuario.

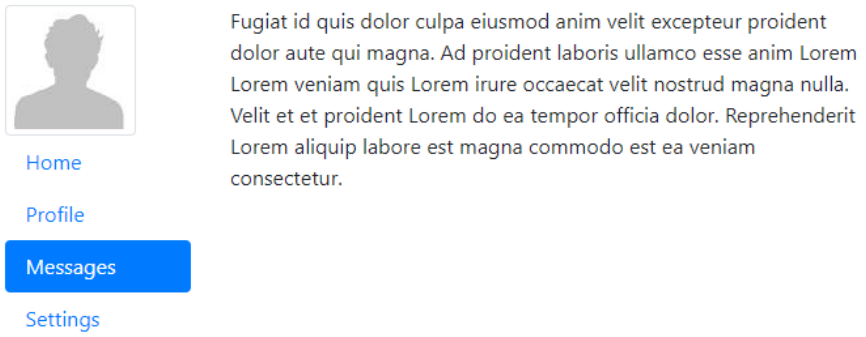

Ilustración 11: Formulario web de perfil de usuario.

Además, en la capa de controlador (del modelo-vista-controlador) se han definido tres controladores para los servicios web

- "HomeController": Controlador de la página principal en la que se muestran los formularios de inicio se sesión y otras páginas estáticas de información.
- "UserController": Reuniendo la funcionalidad de usuario, este controlador gestiona los ajustes de usuario, registro, perfil, y el restablecimiento de la contraseña de usuario.
- "AdminController": Controlador del servicio REST al que se comunicará el diseño de flujos de trabajo.

Por lo tanto, el controlador "AdminController" mapea las entradas HTTP que ofrecen el servicio REST para la creación, consulta, modificación y eliminación de recursos. Se han definido entonces las siguientes entradas:

- "start": Entrada del tipo POST (contiene en su cuerpo los datos de la consulta), encargada de la ejecución de flujos de usuario. Recibe una estructura JSON, con los identificadores de cada uno de los procesos iniciales de la secuencia. Luego, a partir de cada uno de estos se inicia la ejecución siguiendo el flujo.

    - En primer lugar, es asignado el binder de destino de la anterior iteración (un "RunnableBinder" sin ninguna tarea para los procesos iniciales).
    - Luego, en caso de que hayan publishers disponibles a los que subscribirse, se suscribe el binder y se ejecuta.
    - Luego, el proceso es ejecutado si su binder principal está listo.
    - Después, siguiendo el flujo (detallado en el Diseño) se itera cada proceso enlazado (a través de cada binder de origen y destino) comprobando si el binder de origen puede suscribirse al publicador que esté disponible. Y entonces, cuando este binder es ejecutado, sigue la ejecución.
- "generic/Workflow" o "generic/AProcess": Entrada de tipo GET (los datos están contenidos en la misma URL) que permite consultar el primer workflow o proceso que se encuentre en la base de datos a partir del identificador de usuario. Así, a través de este se obtiene los datos del primer workflow asociado al usuario, y en caso de que no exista ninguno, un nuevo workflow es creado y asociado al usuario.
- "generic/" seguido del nombre de la clase del objeto, seguido del identificador del objeto: Entrada de tipo GET para consultar procesos, publicadores, suscriptores, binders o flujos. Además, si la consulta a esta entrada es de tipo DELETE, elimina el objeto de dicha clase.
- "generic/" seguido del nombre de la clase del objeto: Entrada de tipo POST o PUT para la creación y modificación de procesos, publicadores, suscriptores, binders o flujos.

Estas entradas, utilizan los mensajes de estado (o códigos de estado) de la respuesta a la consulta para definir si un error ha ocurrido o si la consulta ha sido satisfactoria. Así mismo, los datos consultados y devueltos se pasan en formato de texto plano o en formato JSON.

2.6. DISEÑO DE FLUJOS DE TRABAJO

Haciendo uso del servicio estático de archivos de Spring Boot, se ha implementado el diseño de flujos de trabajo en JavaScript usando el marco PixiJS de forma modular y empaquetando estos módulos a través de WebPack. Para eso, se ha usado el módulo de viewport (polígono grafico que se asemeja a la funcionalidad que da una ventana a través de la cual ver distintas partes de un lugar) llamado "pixi-viewport"[24] desarrollado por David Figatner (Desarrollador de videojuegos indie). Dicho módulo, es muy configurable y permite entre otras características, realizar acercamientos, arrastrar y soltar, que son útiles para el diseño de flujos.

En primer lugar, se han implementado distintos módulos para la representación gráfica de botones, procesos, binders y enlaces que extienden de la clase de PixiJS "PIXI.Sprite". Estos además definen su reacción ante los distintos eventos de puntero (que se disparan tanto con ratones, como con eventos de toque en pantallas táctiles) y hacen uso de modales (contenedores HTML que enfocan un formulario a modo de alerta dentro de la misma web) para la asignación de propiedades a objetos como binders, procesos, productores o suscriptores. Así mismo, tras ejecutar cambios en el diseño, se hacen las llamadas correspondientes al servidor web en las entradas ofrecidas por el controlador REST "AdminController".

[24] https://github.com/davidfig/pixi-viewport

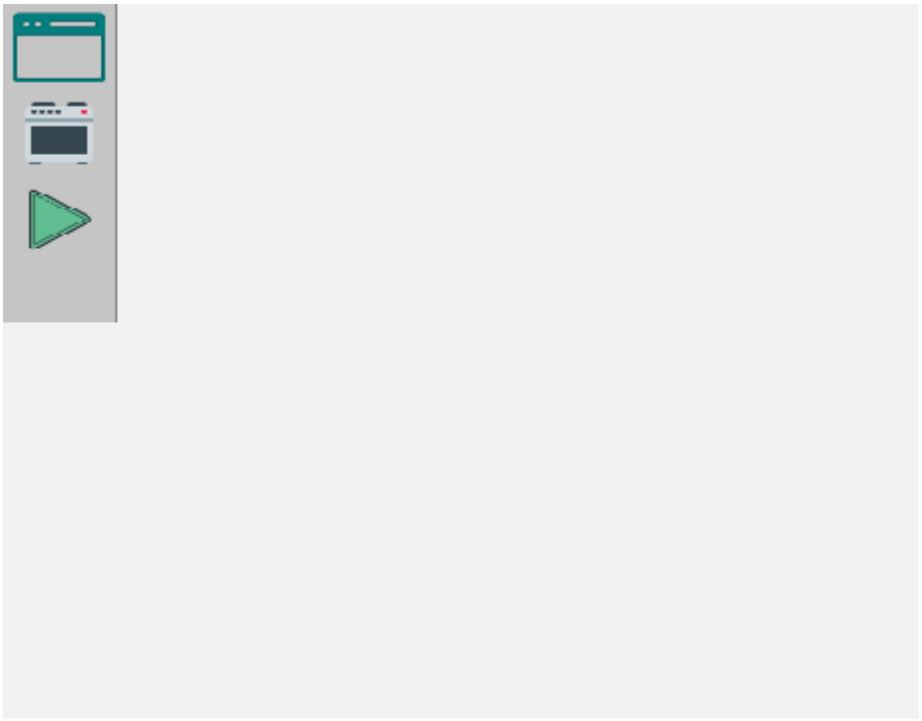

Ilustración 12: Interfaz de diseño de flujos de trabajo.

El diseño de la interfaz se ha mantenido simple y horizontal, agregado una barra de botones a la izquierda (Ilustración 12: Interfaz de diseño de flujos de trabajo.) con los cuales ejecutar distintas acciones sobre el workflow.

El primero de los botones (), permite agregar una nueva definición de proceso al workflow actual. Así, haciendo uso de Bootstrap, se muestra un modal (Ilustración 13: Formulario para la asignación de propiedades de proceso.) cómo formulario de aquellas propiedades del proceso para luego ser almacenadas en el servidor.

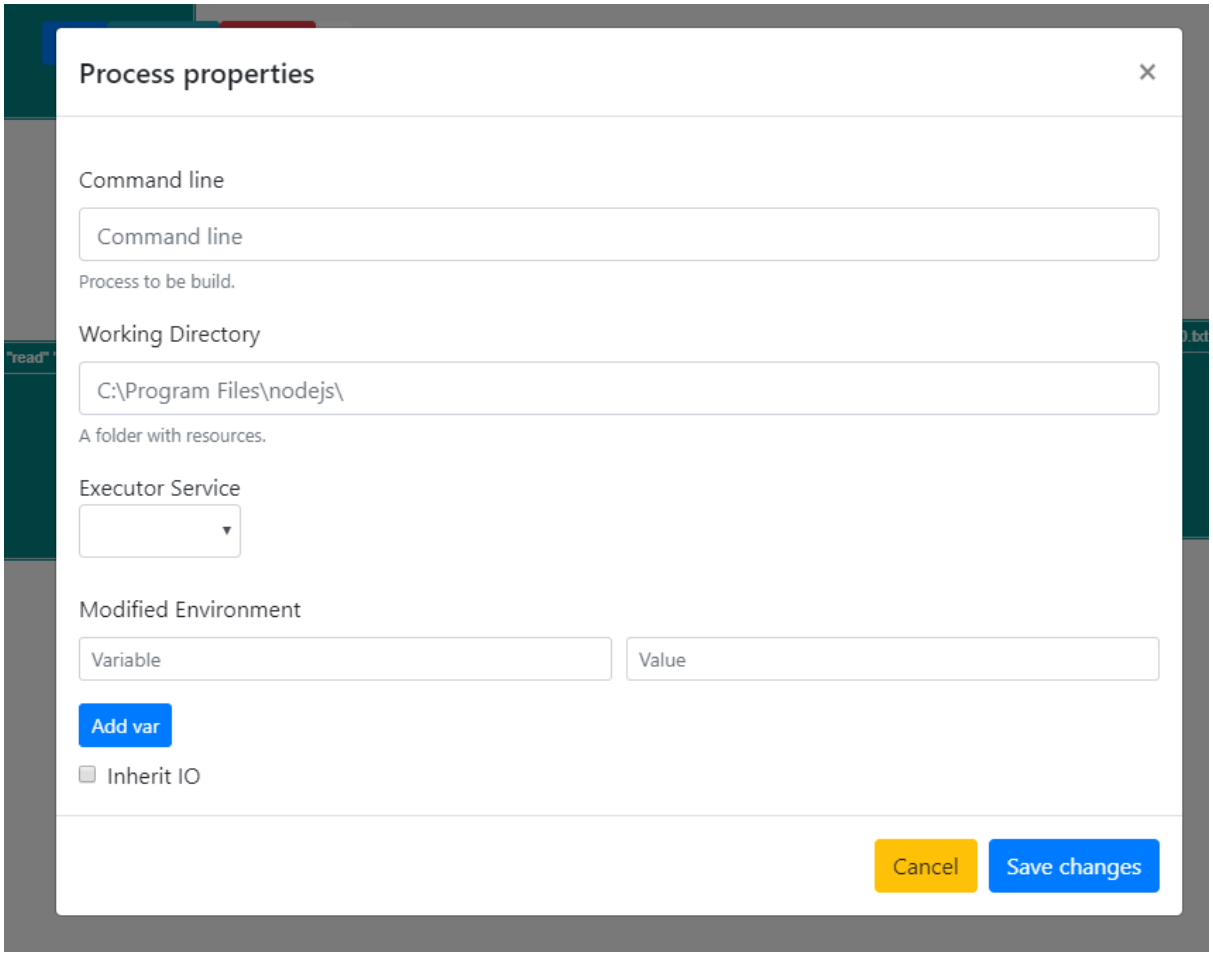

Ilustración 13: Formulario para la asignación de propiedades de proceso.

El segundo botón ( ), permite añadir un nuevo "ExecutorService" al workflow, que debido a la complejidad de este trabajo se ha limitado a hilos simples.

El tercer botón ( ), analiza los procesos agregados en el workflow obteniendo los nodos iniciales (fácilmente reconocibles al no tener binders de entrada) y los envía al servidor web en la entrada "start" del servicio REST para iniciar la ejecución del flujo actual (por ejemplo, Ilustración 14: Interfaz de diseño de flujos de trabajo con tres procesos enlazados.).

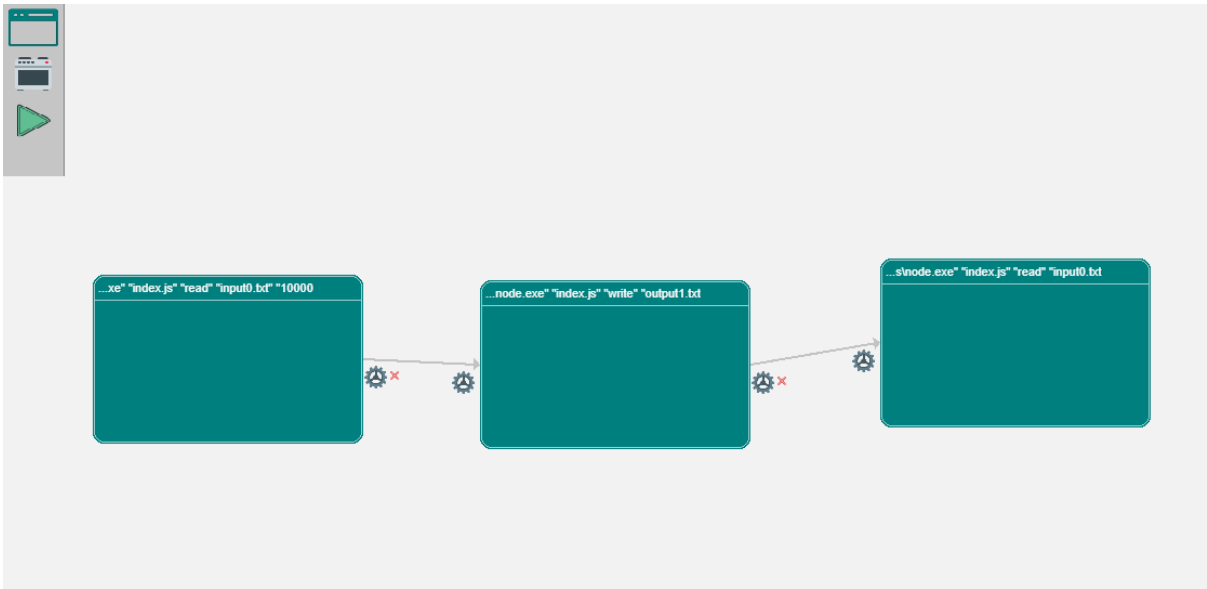

Ilustración 14: Interfaz de diseño de flujos de trabajo con tres procesos enlazados.

Además, se ha implementado que al mover un proceso (arrastrando y soltando) se muestre un menú de opciones (Ilustración 15: Formulario de botones para procesos en la interfaz.) sobre el proceso con tres botones. El primero, "Bind" permite identificar el proceso que formara un enlace (cuando ya se ha identificado un proceso antes, se inicia la creación de un nuevo enlace). El segundo, "Properties" abre el modal de propiedades de proceso (Ilustración 13: Formulario para la asignación de propiedades de proceso.) mediante el cual modificar la definición del proceso. El tercer botón, "Remove" permite la eliminación del proceso actual y finalmente, el botón "X" cierra el menú.

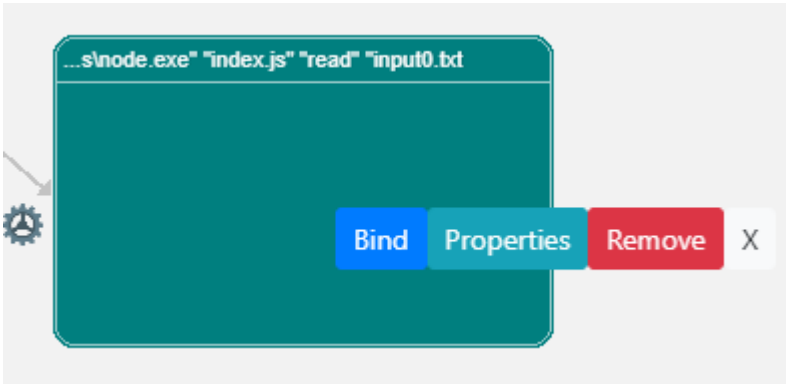

Ilustración 15: Formulario de botones para procesos en la interfaz.

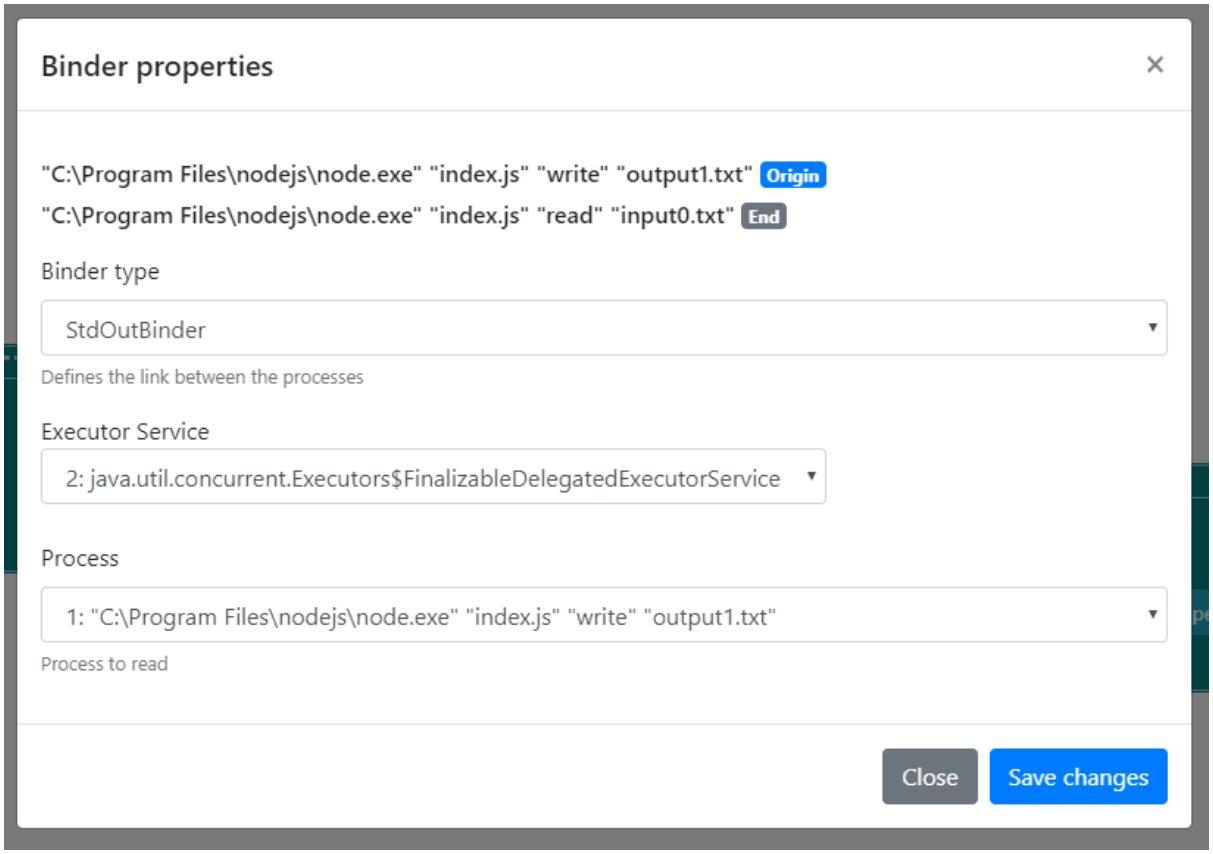

Ilustración 16: Formulario de asignación de propiedades de binders.

Finalmente, al formar un enlace () entre dos procesos (mediante las dos acciones del botón "Bind" del menú de proceso, Ilustración 15: Formulario de botones para procesos en la interfaz.), se muestra el modal de propiedades de binder (Ilustración 16: Formulario de asignación de propiedades de binders.) que asigna las propiedades al primer binder del enlace o binder de origen. El segundo binder o binder de destino, puede modificarse a través del botón del enlace (), mostrando así el modal de asignación de propiedades de binders para este. Por defecto este segundo binder es un RunnableBinder de tipo "Void" (null) debido a que este siempre estará listo al comprobarse la condición para ejecutar el siguiente proceso, es decir, no necesita preparar nada para la ejecución del siguiente proceso.

3. Conclusiones

3.1. CONCLUSIONES

El gran avance de las distintas herramientas para la EAI tanto con el soporte de distintos formatos de comunicación como con el uso de herramientas gráficas para el diseño de integraciones han demostrado ser todo un éxito como con el uso de la API Connect de IBM que ha sido usada por Telefónica, "El crecimiento fue exponencial y gracias a la solución de IBM empezamos a subir diferentes servicios de forma muy fácil, algo que antes nos costaba mucho" explica Santiago Gatti, empleado de Telefónica (IBM apuesta a la transformación de los negocios con las nuevas tecnologías, 2019). Por eso, este proyecto, aunque con un gran camino que seguir, ha demostrado que puede ofrecer funcionalidades muy similares a las ofertadas por IBM App Connect en la simplificación de integración de aplicaciones, pero con la ventaja de alcanzar pequeñas y medianas empresas que muchas veces evitan la transformación digital por los costes altos; de esta manera, este proyecto ha logrado iniciar el desarrollo de un software que pueda ejecutar, modelizar y simplificar la integración de distintos programas informáticos aunque en un nivel muy básico, pero que puede continuarse en el uso de software libre como WSO2.

Por otro lado, la simplicidad del diseño de flujos de trabajo en web ha permitido entender fácilmente la relación existente entre las diferentes aplicaciones que puede usar una empresa y de esta forma acercar a los encargados de la organización hacia una mejor toma de decisiones en relación con el involucramiento de aplicaciones en la organización. Esto, ya ha sido demostrado en el éxito de la API Connect de IBM ya mencionada. Además, al hacerse uso de tecnologías web normalmente orientadas a videojuegos, como en nuestro caso PixiJS, se saca muchísimo más provecho a la interacción entre el cliente y el sistema debido a que estas se desarrollan como uno de sus propósitos para la interacción con el cliente. Así, interactuar con el teclado con movimientos o efectos específicos, con el teclado y aun con pantallas táctiles, se hace muy ameno el desarrollo y mejora la eficiencia en el uso de gráficos para representar modelos como en nuestro caso, flujos entre programas. Un ejemplo claro de esto ha sido el desarrollar toda la interfaz web para el diseño de flujos de trabajo usando imágenes para cada tipo de objeto. A estos se les ha podido fácilmente agregar funcionalidades de interacción y más específicas para cada caso. Estos combinados con el uso de tecnologías web como Bootstrap, permitieron además desarrollar formularios dentro de la interfaz web tan sencillos e interactivos y, sin embargo, sin perder calidad y adaptabilidad. De esta manera, demostraron que la tendencia de hacer uso de tecnologías web en el desarrollo de interfaces de software puede ofrecer resultados eficientes como los ofrecidos por interfaces de escritorio.

Además, el desarrollo de este proyecto ha sido un gran complemento en la formación del grado en Ingeniería de Software. La variedad de tecnologías que han sido involucradas y la profundidad de cada una de estas al interactuar entre sí, permiten tener una visión más amplia de la ingeniería aplicada en las aplicaciones empresariales. Así, la construcción de procesos, el uso de señales, el control de flujos y la comunicación entre procesos ha sido uno de los énfasis principales en el desarrollo de todo el proyecto que ha permitido profundizar mucho más en el funcionamiento de integraciones entre distintas aplicaciones, pero comprendiendo aún más las implicaciones de decisiones que puedan tomarse para afectar el desarrollo de estas. También, ha permitido entender la importancia de la seguridad en aplicaciones web, desde el desarrollo de

funcionalidades básicas del sistema, hasta el uso que se hace de estas en todo el sistema en conjunto. Así, pequeñas decisiones de diseño pueden afectar todo el funcionamiento de un sistema y la forma en la que el usuario usa este; y es por esto, que la fase de diseño de estos componentes esenciales debe ser con una visión tan completa del sistema, que permita luego ofrecer las funcionalidades que el usuario necesita, pero logrando mantener la integridad y eficiencia del sistema.

Uno de los mayores beneficios que se han adquirido en el desarrollo de este proyecto, es poder investigar el uso de las nuevas tecnologías (o tendencias actuales pero desconocidas por el alumno) y profundizar mucho más en el uso de las ya conocidas para poder así, intentar desarrollar una aplicación tan robusta como herramientas actuales de tan grande envergadura como IBM App Connect y las otras ya mencionadas. Pero es de reconocer que ha sido muy provechoso todo el conocimiento adquirido por la formación en el grado de Ingeniería de Software. Este ha permitido avanzar tan rápido al aprender que ha permitido diseñar y rediseñar varias veces cada una de las componentes de este proyecto con el fin de mantener una visión a futuro de integrar más y más de estas tecnologías.

3.2. TRABAJO FUTURO

Debido al desarrollo desde cero de este trabajo y las limitaciones de tiempo, el trabajo futuro es muy amplio. Una de las principales ventajas que podrían haberse aprovechado era el uso de los proyectos de código abierto de WSO2 con el fin de aprovechar su éxito en el uso de estándares de comunicación tan variados.

Además, el diseño de flujos de trabajo aun contiene errores como en el caso de bucles anidados que deben ser corregidos. Así como también, la gestión de los diferentes procesos podría aprovecharse para no solo realizar ejecuciones de aplicaciones sino también para extraer estadísticas de uso de cada una de ellas.

También, agregar otros protocolos de comunicación de procesos, así como también, permitir mayores posibilidades de configuración en el uso de los ya implementados como en el caso de los sockets, abriría un abanico más amplio de configuraciones para comunicar procesos tanto en sistemas distribuidos como en no distribuidos.

Otra característica que aprovechar es en la ejecución de procesos, en donde se podría permitir la ejecución en distintos servidores para alcanzar una escala distribuida de computación o hacer uso de contenedores para aislar la ejecución de procesos y/o permitir combinar distintos sistemas.

Además, el uso de los modales de Bootstrap podría mejorarse en el diseño de flujos para simplificar la construcción de formularios de forma más dinámica y simple usando marcos basados en componentes.

Finalmente, sería interesante realizar investigaciones sobre nuevas formas de comunicación entre procesos usando la interfaz gráfica o bien automatizando las interacciones normales de un usuario a través de esta.

3. Conclusions

3.1. CONCLUSIONS

The great progress of the different tools for the EAI, both with the support of different communication formats and with the use of graphic tools for the design of integrations have proven to be a success as with the use of the IBM Connect API that has been used by Telefónica, "The growth was exponential and thanks to the IBM solution we started to upload different services very easily, something that used to cost us a lot" explains Santiago Gatti, employee of Telefónica (IBM bets on the transformation of business with the new technologies, 2019). Therefore, this project, although with a great path to follow, has shown that it can offer features very similar to those offered by IBM App Connect in the simplification of application integration, but with the advantage of reaching small and medium enterprises that many times avoid digital transformation due to high costs; In this way, this project has managed to start the development of software that can execute, model and simplify the integration of different computer programs, although at a very basic level, but which can continue in the use of free software such as WSO2.

On the other hand, the simplicity of the design of web workflows has allowed us to easily understand the relationship between the different applications that a company can use and, in this way, bring the people in charge of the organization towards a better decision making in relation to the involvement of applications in the organization. This has already been demonstrated in the success of the IBM Connect API already mentioned. In addition, when using web technologies normally oriented to video games, such as in our case PixiJS, much more benefit is taken from the interaction between the client and the system because they are developed as one of their purposes for interaction with the client. Thus, interacting with the keyboard with specific movements or effects, with the keyboard and even with touch screens, the development becomes very enjoyable and improves the efficiency in the use of graphics to represent models as in our case, flows between programs. A clear example of this has been the development of the entire web interface for the design of workflows using images for each type of object. These have been easily able to add interaction and more specific functionalities for each case. These combined with the use of web technologies such as Bootstrap, also allowed to develop forms within the web interface so simple and interactive and yet without losing quality and adaptability. In this way, they demonstrated that the tendency to make use of web technologies in the development of software interfaces can offer efficient results such as those offered by desktop interfaces.

In addition, the development of this project has been a great complement in the formation of the degree in Software Engineering. The variety of technologies that have been involved and the depth of each of these when interacting with each other, allow to have a broader view of the engineering applied in business applications. Thus, the construction of processes, the use of signals, the control of flows and the communication between processes has been one of the main emphasis in the development of the whole project that has allowed to deepen much more in the operation of integrations between different applications, but understanding even more the implications of decisions that can be taken to affect the development. Also, it has allowed us to understand the importance of security in web applications, from the development of basic system functionalities, to the use made of these in the whole system as a whole. Thus, small design

decisions can affect the entire operation of a system and the way in which the user uses it; and that is why, the design phase of these essential components must be with such a complete vision of the system, that it can then offer the functionalities that the user needs, but managing to maintain the integrity and efficiency of the whole system.

One of the greatest benefits that have been acquired in the development of this project, is to investigate the use of new technologies (or current trends but unknown by the student) and deepen much more in the use of those already known to be able to try to develop an application as robust as current tools as large as IBM App Connect and the others already mentioned. But it is to be recognized that all the knowledge acquired by training in the degree of Software Engineering has been very helpful. This has allowed us to move forward so quickly by learning that it has allowed us to design and redesign each of the components of this project several times in order to maintain a vision for the future of integrating more and more of these technologies.

3.2. FUTURE WORK

Due to the development from scratch of this work and time constraints, future work is very broad. One of the main advantages that could have been taken advantage of was the use of WSO2 open source projects in order to take advantage of their success in using such varied communication standards.

Also, the workflow design still contains errors as in the case of nested loops that must be corrected. As well as, the management of the different processes could be used not only to execute applications but also to extract usage statistics from each of them.

Also, adding other process communication protocols, as well as allowing greater configuration possibilities in the use of those already implemented as in the case of sockets, would open a wider range of configurations to communicate processes both in distributed systems and in not distributed

Another feature to take advantage of is the execution of processes, where it could allow the execution on different servers to reach a distributed scale of computing or make use of containers to isolate the execution of processes and / or allow to combine different systems.

In addition, the use of Bootstrap modals could be improved in the design of flows to simplify the construction of forms in a more dynamic and simple way using component-based frameworks.

Finally, it would be interesting to conduct research on new forms of communication between processes using the graphical interface or by automating the normal interactions of a user through it.

4. Bibliografía

- Apache Software Foundation. (11 de Septiembre de 2019). *Synapse*. Obtenido de Apache Synapse Enterprise Service Bus (ESB): https://synapse.apache.org/
- Gauchat, J. D. (2012). *El gran libro de HTML5, CSS3 y JavaScript.* Marcombo.
- IBM apuesta a la transformación de los negocios con las nuevas tecnologías. (3 de Junio de 2019). *El Cronista*. Obtenido de https://www.cronista.com/brandstrategy/IBM-apuesta-a-la-transformacion-de-los-negocios-con-las-nuevas-tecnologias-20190530-0048.html
- IBM United States. (2018). *Software Announcement 218-037.* Obtenido de https://www-01.ibm.com/common/ssi/rep_ca/7/897/ENUS218-037/ENUS218-037.PDF
- Indrasiri, K. (2016). *Beginning WSO2 ESB.* Apress. doi:10.1007/978-1-4842-2343-7
- Lui, M., Gray, M., Chan, A., & Long, J. (2011). *Enterprise Application Integration Fundamentals.* doi:10.1007/978-1-4302-3346-6_1
- Pasquali, S. (2013). *Mastering Node.js.* Packt Publishing.
- Rattinger, Á. (2013). *Nuevo juego, nuevas reglas: marketing de vanguardia.* México: Ediciones Felou.
- Reddy, K. S. (2017). *Beginning Spring Boot 2: Applications and Microservices with the Spring Framework* (Vol. 1st ed. Edition). Apress.
- Reynaldo Guevara, S. (29 de Mayo de 2018). Cómo elegir Windows o Linux como SO para su empresa. *TechTarget*. Obtenido de https://searchdatacenter.techtarget.com/es/cronica/Como-elegir-Windows-o-Linux-como-SO-para-su-empresa
- Sommerville, I. (2005). *«Ingeniería del software» (séptima edición).* Madrid, España: Pearson Educación.
- Spurlock, J. (2013). *Bootstrap: Responsive Web Development.* New York: O'REILLY.
- Spuy, R. v. (2015). *Learn Pixi.js.* Apress.
- Wimalasiri, P., & Tharindu, B. (4 de Julio de 2019). Integrating Office 365 with WSO2 Identity Server. *WSO2 Library*. Obtenido de https://wso2.com/library/articles/2019/07/integrating-office-365-with-wso2-identity-server/
- Yates, C., Ladd, S., Deinum, M., Serneels, K., & Vanfleteren, C. (2012). *Pro Spring MVC: With Web Flow.* Apress.