Resum

El desenvolupament de la ferramenta Orienta-T aborda la problemàtica de la incertesa vocacional entre els estudiants. Esta ferramenta integra múltiples plataformes i servicis en el núvol utilitzant una arquitectura API-Led, facilitant recomanacions personalitzades de formacions i la gestió de cursos d'especialització.

Per a això, s'han identificat i analitzat solucions tecnològiques adequades per a la gestió d'usuaris, recomanació de formacions, notificacions i transaccions de pagament.

El projecte inclou el disseny de APIs en diferents capes (sistema, procés i experiència) per a assegurar una integració eficient dels servicis. Durant la implementació, s'han efectuat proves per a garantir la qualitat del middleware.

Els resultats obtinguts mostren una ferramenta funcional i eficient que complix amb uns objectius plantejats. Es conclou reflectint l'èxit del projecte i es destaca la importància de la integració d'aplicacions i l'ús de tecnologies avançades per a resoldre problemes reals en l'àmbit educatiu.

Paraules clau: incertesa, API Led, integració, prototip, arquitectura, endpoint

Resumen

El desarrollo de la herramienta Orienta-T aborda la problemática de la incertidumbre vocacional entre los estudiantes. Esta herramienta integra múltiples plataformas y servicios en la nube utilizando una arquitectura API-Led, facilitando recomendaciones personalizadas de formaciones y la gestión de cursos de especialización.

Para ello, se han identificado y analizado soluciones tecnológicas adecuadas para la gestión de usuarios, recomendación de formaciones, notificaciones y transacciones de pago.

El proyecto incluye el diseño de APIs en diferentes capas (sistema, proceso y experiencia) para asegurar una integración eficiente de los servicios. Durante la implementación, se han efectuado pruebas para garantizar la calidad del middleware.

Los resultados obtenidos muestran una herramienta funcional y eficiente que cumple con unos objetivos planteados. Se concluye reflejando el éxito del proyecto y se destaca la importancia de la integración de aplicaciones y el uso de tecnologías avanzadas para resolver problemas reales en el ámbito educativo.

Palabras clave: incertidumbre, API Led, integración, prototipo, arquitectura, endpoint

Abstract

The development of the Orienta-T tool addresses the issue of vocational uncertainty among students. This tool integrates multiple platforms and services in the cloud using an API-Led architecture, facilitating personalised training recommendations and the management of specialisation courses.

To this end, suitable technological solutions for user management, training recommendations, notifications and payment transactions have been identified and analysed.

The project includes the design of APIs at different layers (system, process and experience) to ensure an efficient integration of the services. During implementation, tests have been carried out to ensure the quality of the middleware.

The results obtained show a functional and efficient tool that meets the objectives set. The conclusion reflects the success of the project and highlights the importance of application integration and the use of advanced technologies to solve real problems in the educational field.

Key words: uncertainty, API Led, integration, prototype, architecture, endpoint

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

La incertidumbre vocacional entre los jóvenes es una realidad palpable en nuestra sociedad actual. Frente a la difícil decisión de qué continuar estudiando, muchos estudiantes se encuentran paralizados por la inseguridad, sin tener claridad sobre qué elección tomar. Esta falta de información y orientación no solo genera estrés en los estudiantes, sino que también puede conllevar decisiones académicas y profesionales poco acertadas.

Este proyecto se propone una alternativa, la herramienta Orienta-T, que ayudará a definir los intereses formativos y laborales a fin de elegir formaciones que puedan proporcionar a los estudiantes una mayor satisfacción.

Con esta herramienta se pretende limitar la dificultad que muchos jóvenes experimentan cuando tienen que decidir sobre su futuro profesional. En ella se combinan recomendaciones personalizadas de formaciones tanto universitarias como no universitarias con una herramienta de comercio electrónico que permita un acceso directo a recursos educativos.

Esta solución integrada ofrecerá ayuda al estudiante, permitiendo así que aclare sus dudas, que explore posibilidades que quizás no había considerado, y que tenga la oportunidad de poner el foco y aplicarse en los temas que más le interesen y se adecúen a su talento.

Considerando el problema identificado y la necesidad de encararlo, la solución propuesta consiste en la integración de múltiples plataformas mediante APIs, con el fin de establecer una arquitectura API Led. Este tipo de conectividad nos aportará diversos beneficios, destacando entre ellos la sencilla tarea de incorporación de fuentes de información heterogéneas y distribuidas como InfoJobs o PoliformaT en algunas de las plataformas de las que haré uso. Por otro lado, aportará la capacidad de reutilizar Apis ya existentes, como las de pasarelas de pago, adaptándolas según las necesidades específicas.

Algunas de las plataformas que se podrán ver en la integración son: Amazon RDS, Salesforce, PayPal y SNS. Respecto a las herramientas que se utilizarán para el desarrollo del prototipo API Led, encontramos Anypoint de MuleSoft y AWS. Estas tecnologías proporcionarán la capacidad necesaria para gestionar las APIs y los servicios en la nube de manera eficiente y escalable, garantizando la viabilidad y el éxito del proyecto.

1.1 Motivación

A día de hoy, el campo que abarca la informática es muy extenso y a pesar de su grandeza, es un campo en constante expansión, abarcando cada rincón del mundo que habitamos y transformando nuestra manera de vivir.

Durante mi formación académica he tenido la oportunidad de explorar la inmensidad de este dominio, descubriendo que en la informática siempre hay algo nuevo y emocionante que aprender.

Una de las áreas que más ha captado mi interés recientemente, siendo previamente desconocida para mí, es la integración de aplicaciones. Comenzó a llamarme la atención este curso al tener una asignatura en cuarto que se basaba en ello. Posteriormente, al apuntarme a la Cátedra (organizada por DISID, Mulesoft y la UPV) relacionada con este tema también y realizar una Hackathon, terminé de consolidar mi entusiasmo por el área. La posibilidad de conectar diferentes sistemas y plataformas para que trabajen de manera conjunta no solo es fascinante, sino que también es crucial en el mundo tecnológico actual.

Teniendo en cuenta todo lo aprendido, tuve claro, dadas las enormes posibilidades que ofrece, que quería tener como base de mi Trabajo Fin de Grado la integración de aplicaciones. Ahora tenía que definir la finalidad, para qué iba a servir esta integración, y fue tras una conversación con mi hermana menor, la cual me pidió que le ayudase a decidir sobre qué estudiar una vez finalizase su etapa de enseñanza postsecundaria, cuando empecé a pensar en una herramienta que sirva de apoyo y orientación a las personas jóvenes que quieren continuar con sus estudios. Es frecuente que cuando llega el momento en el que tienes que elegir, plantearte a qué te quieres dedicar en tu etapa laboral surjan dudas, indecisiones, inseguridades y ansiedades por no tener claro si tu decisión es o no la acertada.

Tras esto, me propuse desarrollar, mediante la integración, una herramienta que ofreciese información, orientación, cursos y propuestas a fin de que los alumnos más indecisos puedan hacer su elección sobre qué estudios continuar cursando con mayor seguridad, teniendo una mayor garantía de satisfacción personal y éxito posterior. Pudiendo así poner en práctica lo aprendido durante el curso y la Cátedra.

1.2 Objetivos

El objetivo de este trabajo es desarrollar una herramienta que haga función de enlace entre la incertidumbre vocacional del estudiante y la toma de decisiones fundamentadas, proporcionando así apoyo a los estudiantes para que logren definir y construir una formación académica que se adapte y acomode a sus aspiraciones personales. En la solución se incluirá el análisis de datos, el uso de servicios en la nube, la gestión de clientes y transacciones financieras.

Esto se logrará gracias a la arquitectura API-Led de tres capas, la cual nos permitirá que, esta herramienta, además de ser funcional, sea flexible, escalable y segura, entre otras características.

1.3 Metodología

Para lograr los objetivos mencionados en la sección 1.2, se seguirá una planificación que se dividirá en varias fases, cada una con sus respectivas tareas y objetivos.

Dicha planificación dictará la metodología a seguir, adoptando un enfoque ágil para permitir flexibilidad y adaptación a los cambios. La metodología ágil nos permitirá una gestión incremental más sencilla y la entrega continua de versiones funcionales de la herramienta, lo cual es importante, ya que los requisitos pueden evolucionar con el tiempo.

A continuación se detallarán, las fases de planificación que se seguirán:

- Fase de análisis y definición de requisitos: en esta etapa, se identificarán los requisitos de Orienta-T.
- Fase de diseño: en esta fase, se definirá la estructura de la herramienta y se esbozarán diagramas en los que se podrá ver cómo queremos que los componentes de Orienta-T interaccionen entre ellos.
- Fase de desarrollo: una vez tengamos pensado como queremos establecer y construir nuestra herramienta, procederemos a desarrollar la solución mediante el uso de frameworks. Esta fase estará por otras etapas, las cuales incluyen: la configuración del entorno desarrollado, la implementación de las APIs, la integración de los sistemas y la transformación de datos.
- Fase de pruebas: tras acabar con la implementación, deberemos de probar el funcionamiento de la solución desarrollada y comprobar que los resultados corresponden a lo esperado (a lo establecido durante la fase de análisis y definición de requisitos).
- Fase de despliegue: tras verificar el correcto funcionamiento de la herramienta mediante las pruebas, se podrá desplegar en CloudHub.

1.4 Estructura de la memoria

La memoria de este Trabajo de Fin de Grado está organizada en varios capítulos, cada uno de los cuales abordan diferentes aspectos (muchos de ellos tienen relación con las fases mencionadas en la sección 1.3). A continuación, se presenta una visión general de lo que el lector encontrará en cada capítulo.

El primer capítulo (sin contar el introductorio),“Contexto tecnológico”, proporcionará la base teórica necesaria para comprender las soluciones implementadas en el proyecto. Entre sus contenidos, encontraremos conceptos fundamentales de la integración de aplicaciones, tipos de middleware, tecnologías de comunicación, frameworks de integración y la arquitectura API-Led.

En el capítulo “Requisitos y análisis de la solución de integración”, se detallan los requisitos funcionales y no funcionales del sistema. Además realizaremos un análisis de las posibles soluciones para cada funcionalidad que se pretende implementar.

Las decisiones de diseño tomadas para la implementación del proyecto se presentan en el capítulo denominado “Diseño”, explicando también las APIs pertenecientes a cada capa.

La descripción de las tecnologías y herramientas empleadas en el desarrollo del se realiza en el capítulo “Tecnología utilizada”.

En el capítulo con nombre “Desarrollo de la solución”, se describe como se han implementado las funcionalidades clave del proyecto, explicando para ello los flujos de trabajo comunes y específicos de cada API.

El siguiente paso será desplegar la plataforma recién implementada, dichos pasos se especificarán en el capítulo nombrado “Implantación y pruebas”, en el que, además de desplegar, se comentará cual ha sido el proceso para verificar el correcto funcionamiento de la solución desarrollada.

Finalmente, en el capítulo “Conclusiones”, se evalúan los resultados obtenidos, los problemas encontrados y las habilidades adquiridas durante el desarrollo del trabajo. También se proponen otras ideas para mejorar y ampliar la funcionalidad del middleware.

CAPÍTULO 2

Contexto Tecnológico

La integración de aplicaciones nació para solventar el problema de la comunicación entre programas, los cuales con la acelerada evolución del mundo informático, no son compatibles entre ellos; convirtiéndose así en una solución decisiva a la hora de simplificar operaciones cruzadas y obtener mayor eficiencia funcional. [1] [2]

Este método es la base de la infraestructura tecnológica de muchas empresas en la actualidad. Ésta técnica trata de la unión y cohesión, como la primera palabra de su nombre (integración) lo indica, de sistemas y software distintos logrando así una comunicación escalable, flexible y automática. [3]

Se propone como objetivo el conseguir una mejora en el intercambio de datos, siendo este un punto crucial ya que hay casos en los que los datos necesitan ser transformados para poder ser examinados por otras aplicaciones; en la eficiencia, al realizar de forma automática las rutinas de operaciones y suprimir actividades manuales, consiguiendo a su vez una disminución en los costos.

Antes de adentrarnos más en este tema, debemos tener en cuenta los siguientes conceptos:

- API: son las siglas de Interfaz de Programación de Aplicaciones en inglés. Se trata de un conjunto de funciones y protocolos que permiten la comunicación entre componentes de software. Dentro de ellas, albergan distintos tipos, sin embargo, las que más se nombrarán durante este trabajo son las API REST, estas son las más populares y las que más se adecuan a este proyecto, permitiendo realizar un intercambio de datos mediante el protocolo HTTP, e implicando de esta manera que los datos se guarden en caché, que no se envíe el estado en las peticiones y definiendo permisos de acceso a datos entre aplicaciones. [4] [5]

- Middleware: software que actúa como intermediario entre las aplicaciones y sistemas operativos para permitir la gestión de datos y comunicación entre ellos. Está relacionado con las APIs, ya que emplea estas interfaces para ofrecer y acceder a sus servicios. [13]

- Eventos: representa un incidente que sucede en las aplicaciones conectadas y que de manera automática dará lugar a acciones, en las cuales se hayan operaciones estándar o funciones específicas.

- Mapeo de datos: es un proceso que consiste en transformar datos y transmitirlos de manera que coincidan con los campos de distintas fuentes, pudiéndose transferir, de esta forma, sin ser alterados conservando su consistencia e integridad. [12]

  Para realizar esta operación disponemos de herramientas de transformación de datos.

- Conectores: componentes que permite la comunicación y el intercambio de datos. Ejercen de interfaces que conectan aplicaciones y están diseñados para manejar caracteres específicos de los sistemas con los que interactúan.

2.1 Tipos de integración

Dentro de la integración encontraremos diversos enfoques los cuales se adecuarán y nos permitirán adaptarnos a las distintas necesidades y contextos. La selección del tipo de integración a utilizar será de gran importancia y dependerá de factores como la coexistencia entre sistemas, localización de las dispositivos tecnológicos y de las exigencias específicas que necesitemos.

En este apartado, se nombrarán tres tipos teniendo en cuenta los ámbitos en los que se realizan la integración.

2.1.1. Integración de datos

La integración de datos es un proceso que implica la combinación y ajuste de datos provenientes de diversas fuentes para crear una vista unificada y coherente. Este proceso facilita el acceso y análisis de la información, permitiendo usar los datos consolidados para diversos fines.

Además, la integración de datos conforma la base para el correcto funcionamiento de la de aplicaciones, asegurando una conexión precisa de datos entre las aplicaciones.

Pasos a seguir

Cuando realizamos este tipo de integración, probablemente tengamos que efectuar alguno de los siguientes procedimientos que se nombran a continuación (figura 2.1):

- Tener en cuenta las bases de datos,documentos (XML, EDI), ficheros (de texto plano, de formatos CSV, JOSN) servicios, APIs y sistemas de los que tomaremos datos.

- Tomaremos los datos de las fuentes que hemos nombrado en el punto anterior mediante el uso de herramientas de extracción.

- Crearemos una correspondencia entre los distintos datos extraídos haciendo uso de un esquema de mapeo de datos, ya que, los sistemas de los que hemos obtenido los datos probablemente hagan una representación distinta de ellos.

- Como paso previo a la transformación de los datos, deberemos verificar su integridad, consistencia y comprobar si hay errores en ellos. Una vez hecho esto, convertiremos todos los datos a un mismo formato, formato canónico (significa que es común a todos), para poder

manipularlos con más precisión y garantizar compatibilidad entre ellos. Para ello, disponemos de soluciones o librerías predefinidas específicas como lo son los lenguajes declarativos (destinados a los documentos XML) y lenguajes de programación acompañados de parsers.

- Antes de analizar los datos, éstos deberán de estar almacenados en el mismo sitio para que la sincronización de ellos se realice de manera eficaz y se mantengan siempre actualizados.
- En caso de que estemos tratando con datos privados, se deberán de implementar prácticas robustas de gobernanza de datos (conjunto de políticas y procedimientos que garantizan la adecuada gestión de los datos y el cumplimiento de los requisitos de privacidad). Otro procedimiento opcional que facilitaría la comprensión del contexto, significado y fuente de los datos, sería la gestión de los metadatos.
- Finalmente, podremos realizar el análisis de los datos y acceder a ellos.

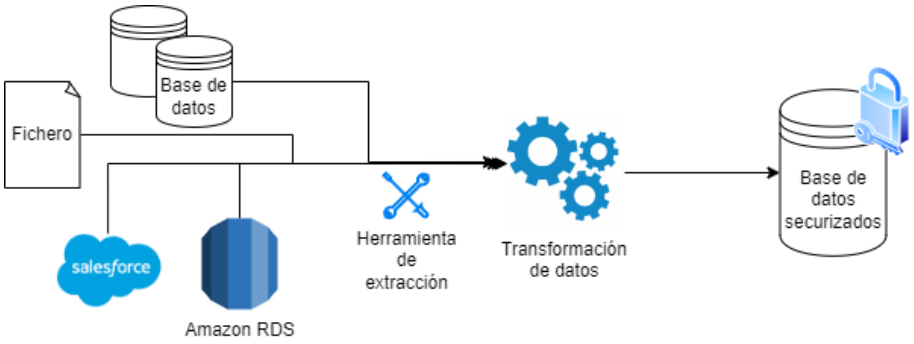

Figura 2.1: Pasos a realizar para la integración de datos

Técnicas

Para alcanzar una integración de datos eficiente y efectiva, se emplean distintas técnicas especializadas que facilitan el manejo de información de múltiples fuentes. A continuación, se presentan las más comunes:

- ETL (Extract Transform Load): hace referencia a los tres pasos (extraer, transformar y cargar) que permiten utilizar y mezclar datos de múltiples fuentes para construir un único repositorio o almacén de datos (figura 2.2). También existe otra técnica, ELT, en la cual, se carga en el repositorio antes de transformar los datos, des esta manera, se hace un mejor uso de los sistemas de almacenamiento, sin embargo, no se le da tanta importancia a la transformación, al no validar y limpiara los datos.

Figura 2.2: ETL (Extract Transform Load)

- Virtualización de datos: consiste en establecer una capa virtual que ofrece una vista unificada de los datos provenientes de diferentes fuentes. Al ser virtual, se puede acceder a ella de manera remota sin importar su ubicación física (figura 2.3).

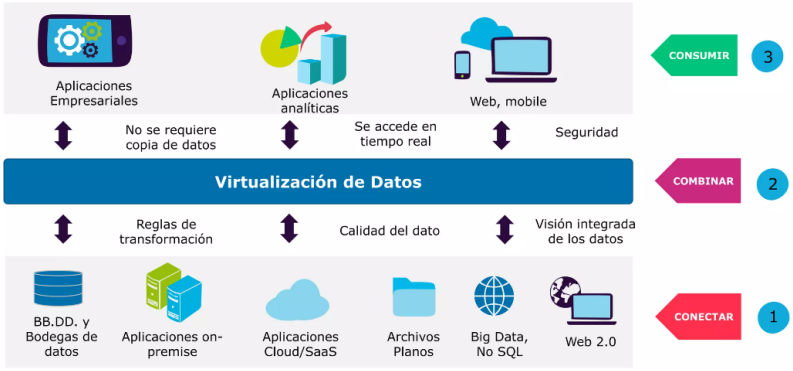

Figura 2.3: Virtualización de datos

- Replicación de datos: es la técnica más básica, trata de copiar datos e integrarlos entre sistemas, de esta manera se accederá en todo momento a información actualizada.

2.1.2. Integración de procesos

También conocida como integración de procesos de negocio (BPM), permite conectar distintas aplicaciones y sistemas para automatizar y optimizar flujos de trabajo.

Mientras que la integración de datos es más útil en un contexto en el que se requiere un análisis de ellos, la de integración de procesos obtiene mejores resultados en circunstancias en las que los procesos operativos requieren de coordinación entre múltiples sistemas y aplicaciones.

Técnicas

Las técnicas para llevar a cabo este tipo de integración son las siguientes:

- Interfaces de programación de aplicaciones (API): en una integración que se basa en conectar aplicaciones, es totalmente necesario un elemento que sirva para enlazarlas. Aquí es donde las APIs se hacen necesarias, permitiendo la comunicación entre las aplicaciones.

- Arquitectura orientada a servicios (SOA): es un marco de trabajo para el desarrollo de software que organiza la lógica de negocio en servicios (componentes que encapsulan una funcionalidad de negocio específica) reutilizables, éstos se comunican entre sí a través de una red. Los servicios son consumidos por aplicaciones web, de escritorio, móviles... Y son ofrecidos por sistemas que gestionan su ejecución. [10]

  Se puede asociar con otras tecnologías, con servicios web SOAP (utilizan dicho protocolo, el cual permite interoperabilidad entre aplicaciones, y es ideal para entornos distribuidos además de permitir comunicación a través de mensajes XML). SOAP utiliza WSDL (Web Services Description Language) para describir los servicios, especificando qué funciones

están disponibles y cómo pueden ser invocadas, facilitando así la interacción entre distintos sistemas. También se puede asociar con servicios REST (basados en HTTP, exponen funcionalidades como recursos accesibles mediante operaciones estándar). [9] [11]

La figura 2.4 muestra cómo se realiza la interacción entre un proveedor de servicios, un directorio de servicios y un solicitante de servicios en una arquitectura SOA utilizando WSDL para describir los servicios y SOAP para la comunicación.

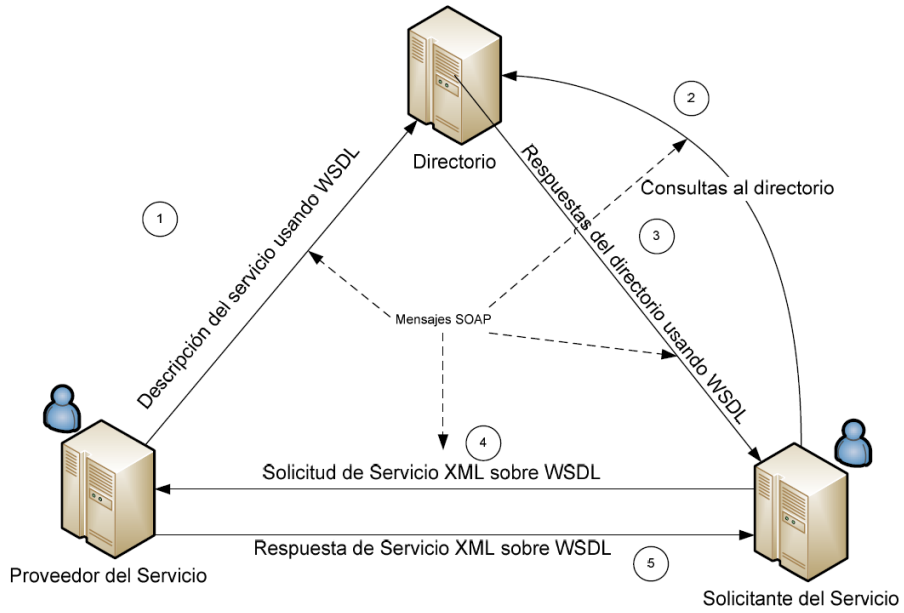

Figura 2.4: Arquitectura Orientada a Servicios

- Gestión de procesos de negocio (BPM): es una metodología usada para automatizar y modelar procesos de negocio logrando mayor efectividad en las operaciones empresariales. Para llevar a cabo esta técnica se emplean diagramas de flujo para el modelado de procesos o herramientas BPMN además de una correcta gestión del flujo de ejecución de tareas.

2.2 Modelos de integración

Mientras que en los tipos de integración se hace hincapié en el qué se está integrando, los modelos de integración responden la pregunta de cómo se lleva a cabo dicho proceso. En este apartado se exploran los distintos modelos y cómo cada uno sirve para satisfacer diferentes necesidades específicas, entre otros puntos; así como las estrategias y arquitecturas subyacentes que dichos modelos proporcionan.

2.2.1. Modelo punto a punto

En este modelo se establece una conexión directa entre sistemas individuales, para ello, se usan APIs o programas personalizados. Ambos sistemas, receptor y emisor, están conectados entre sí sin intermediarios de por medio.

Algunos procesos que se deben realizar para crear de manera correcta esta arquitectura (figura 2.5) es transformar y mapear los datos ya que probablemente haya diferencia de formato entre los datos de las aplicaciones, enrutar los mensajes, hacer uso de colas para gestionar el flujo de datos y crear medidas de seguridad como el cifrado y la autenticación. [14]

Esta arquitectura puede llegar a ser muy fácil de implementar, siendo este uno de sus beneficios, sin embargo, deja de ser útil cuando los sistemas a integrar no son pocos, es decir, la escalabilidad le supone un gran problema, al igual que también lo el maniobrar con grandes magnitudes de datos, siendo esto un impedimento para realizar un proceso de sincronización de datos urgentes.

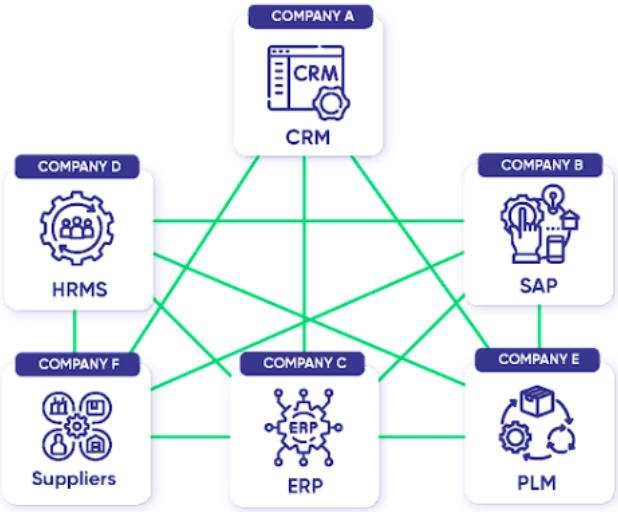

Figura 2.5: Modelo punto a punto

2.2.2. Hub-and-spoke

Este modelo es útil para solucionar los problemas de escalabilidad que surgen al tener un modelo "Punto a Punto". En un mundo tecnológico en constante crecimiento, donde la cantidad de aplicaciones y sistemas a integrar aumenta ininterrumpidamente, el modelo "Hub-and-Spoke" ofrece una estructura centralizada que facilita la integración, reduciendo la complejidad y mejorando la capacidad de gestionar y escalar conexiones. [15]

Esta arquitectura (figura 2.6) está dividida en dos partes principales: un hub (dispositivo que permite establecer conexión entre un número indefinido de sistemas, permitiendo un intercambio de datos) localizado en el centro ejerciendo un rol de mediador y la otra parte, la conformarán los sistemas, que se conectarán al hub. Éste, se encargará de enrutar, transformar los datos y será el único punto de monitoreo. Sin embargo, al otorgar tanta responsabilidad al hub (el cual dispone de una compleja configuración y mantenimiento), en caso de que falle, todas las conexiones se verán afectadas.

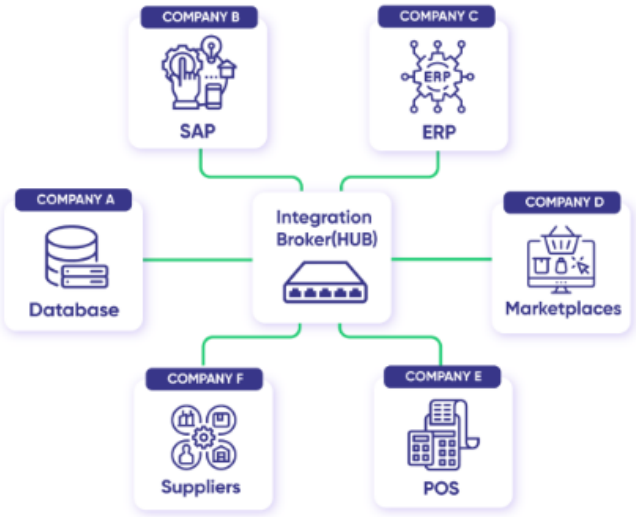

Figura 2.6: Modelo hub-and-spoke

2.2.3. ESB (Bus de servicio empresarial)

El modelo "ESB" trata del uso de un bus de comunicación para integrar diferentes aplicaciones, habilitándolas para que realicen la conexión con el bus (figura 2.7). De esta manera, los sistemas se encontrarán desacoplados entre ellos y se podrán comunicar sin dependencia o sin saber que otros sistemas se encuentran en el bus.

El tipo de bus que se usa (el "ESB") es un componente básico de SOA (Arquitectura Orientada a Servicios), que facilita la creación de nuevos servicios al coordinar servicios ya existentes, permitiendo la integración de sistemas antiguos, que usualmente emplean protocolos y formatos de datos obsoletos, con los protocolos de red modernos utilizados en SOA.

Este modelo se diferencia del "Hub-and-Spoke" en: el grado de flexibilidad que ofrece, siendo mayor la del ´´ESB", y en tener mayores capacidades de enrutamiento y transformación de datos.

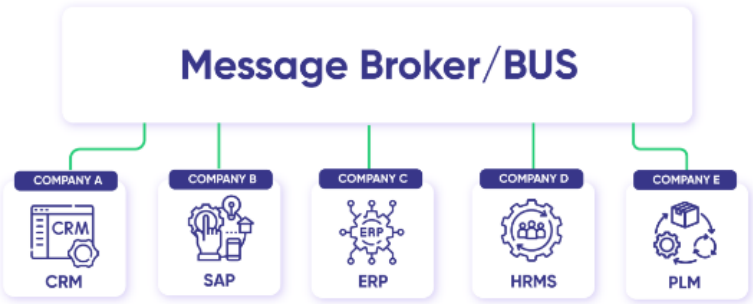

Figura 2.7: Modelo ESB

2.2.4. iPaaS (Plataformas de Integración en la Nube)

Estas plataformas ofrecen un enfoque distinto al de los modelos anteriores y es que, esta vez, la solución se basará en la nube, es decir, que no se usa ningún middleware externo o hardware para llevarla a cabo. [16]

Este modelo suele estar formado por componentes como: interfaces de usuario basadas en web, mecanismos robustos de seguridad y un motor de integración el cual ejercerá de núcleo, encargado de manejar la lógica de integración. Además incluye conectores preconfigurados o aplicaciones SaaS (modelo de distribución de software en la nube), los cuales permiten una buena comunicación entre sistemas, un ejemplo de ello es Salesforce, la cual se usará durante el desarrollo de este proyecto.

2.3 Mecanismos de transferencia y conectividad

El transporte y conectividad son aspectos fundamentales ya que determinan cómo se transmiten los datos entre los diferentes sistemas y aplicaciones, en esta sección se muestran cuales son los métodos y tecnologías que se usan para ello.

2.3.1. Sistemas Gestores de Bases de Datos (SGBD)

Son un modo común de transportar datos. Proporcionan procesos para replicar datos entre sistemas de bases de datos del mismo tipo y gateways (puentes) para interactuar con otros SGBD distintos, como controladores JDBC-ODBC (permiten la comunicación entre aplicaciones Java y bases de datos que usan el estándar ODBC traduciendo llamadas). La arquitectura en la que se basan es cliente - servidor, realizando distinción entre clientes específicos y genéricos, éstos últimos necesitan un driver para realizar la conexión al servidor de datos, en el driver se deberá indicar el host, número de puerto que escucha para conexiones entrantes, el nombre de la base de datos, usuario y contraseña.

Estos mecanismos facilitan la obtención de un intercambio de datos rápido, sin embargo, es poco flexible y, en caso de querer realizar una ampliación del SGBD, aunque es posible realizarla, en la práctica, puede presentar complicaciones significativas como, por ejemplo, la implementación de cambios necesarios para escalar el sistema.

Algunos ejemplos de SGBDs más populares son Oracle, MS SQL SERVER y MySQL.

2.3.2. Transferencia de ficheros

Implica el uso de un directorio local para distribuir información entre aplicaciones. Es considerado un método fácil y permite tanto un uso local (en la misma máquina) como en distintas máquinas que comparten la misma red. No obstante, un método tan simple presenta algunas desventajas, como por ejemplo, la escasa seguridad que ofrece, lo que requiere la implementación de mecanismos adicionales para mejorarla, así como la baja escalabilidad.

2.3.3. Colas de mensajería

Las colas son el componente encargado de almacenar los mensajes para su posterior retransmisión, ya que permiten un intercambio de información mediantes asíncronos. Se tratan de un elemento clave en la implementación de middleware orientado a mensajes (MOM). Según el or-

den de gestión que empleen las colas, serán de un tipo o de otro: si la gestión de mensajes se basa en prioridades o en el orden de llegada (FIFO, LIFO...).

Modos de empleo

A la hora de usar colas de mensajería, distinguimos entre dos modos para ello:

- Aplicación a Aplicación: las aplicaciones necesitan saber con quien se conectarán para enviarles mensajes. Para llevar a cabo este modo solo se podrá utilizar una comunicación punto a punto.
- Publicador/Suscriptor: las aplicaciones se suscriben a temas específicos, y los mensajes sobre esos temas son enviados por otras aplicaciones al servidor para ser consumidos por ellas. A diferencia del otro modo, éste permite una comunicación uno a muchos, en otras palabras, los topics (elementos a los que se suscriben las aplicaciones) pueden tener más de un suscriptor.

2.3.4. Internet y protocolos de red

El hecho de que Internet posea accesibilidad global y ofrezca una gran diversidad protocolos, hace que sea es uno de las mecanismos más usados y más convenientes para la integración de aplicaciones. Nos focalizaremos en los siguientes protocolos brindados por Internet:

- FTP (Protocolo de Transferencia de Archivos): indica el comportamiento que deben tener dos sistemas para entenderse y transferir un archivo. FTP no encripta los datos durante la transferencia, lo que representa un riesgo para la seguridad. Para establecer una conexión FTP, se requiere la dirección donde reside el SGBD, el puerto de conexión (generalmente el 21), la ubicación específica en el servidor donde se almacenarán o desde donde se transferirán los archivos y las credenciales para autenticarse en el servidor.
- HTTP (Protocolo de Transferencia de Hipertexto): se utiliza para intercambiar datos entre un cliente y un servidor web. Se usa comúnmente para enviar páginas HTML, aunque también puede ser utilizado para cualquier tipo de documento siempre que se pueda acceder a él mediante URLs que apunten a dichos documentos. Este protocolo, al igual que el anterior, no cifra los datos, por lo que se recomienda usar una versión más seguras como HTTPS.
- SMTP (Protocolo Simple de Transferencia de Correo): se usa para el intercambio de mensajes entre servidores por medio de Internet. Para poder realizar el envío se necesita la dirección del servidor de correo, las credenciales para autenticarse y el puerto (por defecto es el 25). Los servidores SMTP suelen manejar las colas de mensajes para garantizar la entrega en caso de que el destinatario no esté disponible.
- POP3 (Protocolo de Oficina de Correos 3): similar al protocolo anterior, POP3, también ofrece una solución basada en correos. Sin embargo, su método de funcionamiento distinto: recupera mensajes descargándolos del servidor y, generalmente, eliminándolos después. Se utiliza para leer correos electrónicos a través de Internet.

2.4 Arquitectura y Conectividad API-Led

En esta sección nos adentraremos más en las APIs, explorando una de sus arquitecturas y modos de empleo. Aunque ya han sido mencionadas previamente, en este apartado profundizaremos en sus detalles y aplicaciones específicas.

2.4.1. Conectividad API-Led

Una conectividad basada en API se enfoca en establecer conexiones entre datos y aplicaciones haciendo uso de APIs reutilizables, creadas para realizar funciones específicas. El tipo de conectividad que más interesa para la realización de este trabajo es la conectividad API-Led. Esta metodología, además de conectar datos, ofrece una arquitectura dividida en tres capas, donde cada API dentro de una capa tiene roles, responsabilidades y funcionalidades distintas (figura 2.8). [6] [7]

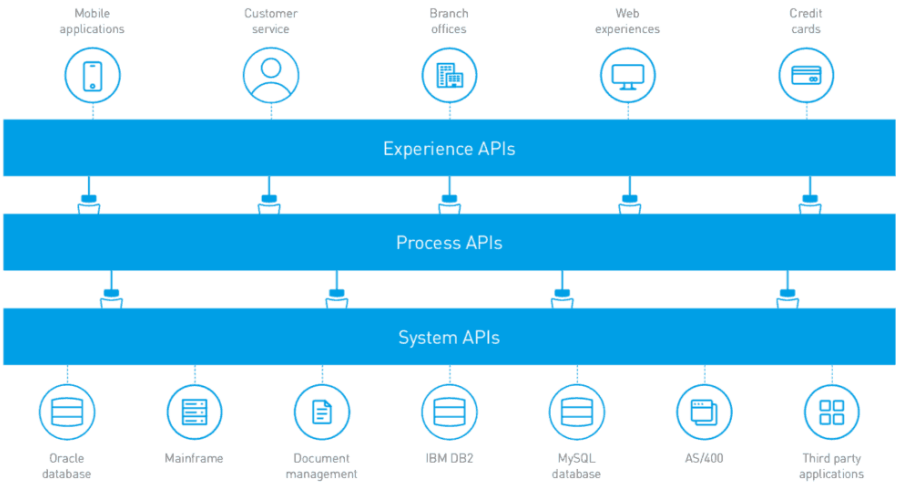

Figura 2.8: Ejemplo arquitectura API Led

API de sistema

Las APIs de sistema se encuentran en la capa más baja de la arquitectura (también conocida como capa de sistemas). Son las encargadas de conectar con los sistemas y recuperar los datos de ellos, pudiendo llegar a transformarlos en caso de que el formato no sea adecuado. Además de con los sistemas, están conectadas con las APIs de proceso, haciéndoles llegar los datos, mediante su URL interna, de manera fiable gracias a la gestión de errores.

A la hora de crearlas, es importante que no se expongan públicamente, debido a la función que ejercen, por lo que se requiere que se implementen en un puerto o red privado. Además, deberemos de crear una API por cada sistema con el que vayamos a interactuar, a menos que se siga un diseño basado en dominios, lo cual no ocurrirá en esta herramienta.

API de proceso

Forman parte de la capa intermedia por lo que tienen conexión tanto con la capa de sistemas como con la de experiencia. Al ejercer el rol de intermediaria entre ambas capas, deben de realizar las siguientes funciones: implementar la lógica de negocio, realizar el enrutamiento, capacidad de organizar llamadas múltiples a la capa de sistema, recopilar, combinar y, en caso de que no se haya hecho en la capa inferior, transformar los datos.

Tampoco deberán de ser expuestas públicamente, deberán de disponer de una URL interna para que la capa superior, capa de experiencia, se pueda comunicar con ella, esta interacción se realizará a través de REST, concepto que se explicará con más detalle en breves.

API de experiencia

Se encuentran en la capa superior de la arquitectura, por lo que serán las APIs con las que interactúen los clientes, es decir, están orientadas al usuario. No son responsables ni de la organización, ni del enrutamiento, y tampoco de la lógica empresarial, solo deben de ofrecer una experiencia agradable al consumidor y mostrar los datos de manera correcta.

2.4.2. REST

REST es un patrón de diseño que tiene como objetivo construir servicios en la web que apliquen los principios de funcionamiento de la misma y se basa en recursos vinculados a una URL, a los cuales se puede acceder mediante métodos HTTP.

REST juega un papel fundamental en el contexto de la conectividad API-Led a la hora de establecer una comunicación entre las capas. Esto se debe a que, en primer lugar, REST utiliza métodos HTTP, ofreciendo una manera consistente de interactuar con los recursos, además, es un protocolo sin estado, lo cual significa que no requerirá de servidores que gestionen el estado de las sesiones de los usuarios. Por estas razones, la flexibilidad que aporta es fundamental para las APIs de experiencia, las cuales precisan de adaptar los datos para distintas interfaces de usuario y dispositivos. [8]

Por lo que, resumiendo lo anterior, en REST, un recurso es cualquier objeto que pueda ser referenciado con una identidad propia (páginas web, documentos, dispositivos...). Los recursos, deben ser explicativos, a fin de que tengan una fácil accesibilidad para los usuarios y tienen una URI, lo que implica que tendrán tanto nombre como dirección.

Otra de las características de este patrón es que los servidores no almacenan información de las peticiones, en consecuencia, las solicitudes HTTP actúan de manera aislada, sin necesidad de indicar el estado. Las peticiones HTTP disponibles en REST son las predefinidas: GET, POST, PUT, DELETE, HEAD y OPTIONS.

2.5 Ciclo de vida de las APIs

En el contexto de la arquitectura API-Led, es fundamental entender el ciclo de vida completo de las APIs. [17] [19] [18] Este ciclo comprende de varias fases, las cuales se detallarán a continuación:

- Diseño: en esta fase, se define el propósito de la API, los objetos que manejará y las operaciones que permitirá. En este proyecto, se utilizará RAML para especificarlas detalladamente.
- Desarrollo: una vez diseñado, se procederá a implementar la API según las especificaciones. Esto incluye la codificación, configuración de seguridad y conexión a los servicios backend.
- Prueba: antes de la implementación, se realizarán pruebas sobre la API para garantizar que funcione correctamente y cumpla con los requisitos. Entre las pruebas que se utilizan, destacan las unitarias, de integración y de carga.
- Implementación: la API se despliega en un entorno de producción. Aquí, se configuran las herramientas de monitoreo y se establece la infraestructura necesaria para soportar el tráfico esperado.
- Monitoreo: para esta fase se utilizan herramientas de monitoreo para rastrear el uso, identificar cuellos de botella y detectar posibles problemas de seguridad.
- Retiro: cuando una API ya no es necesaria o ha sido reemplazada, se procede a retirarla.

CAPÍTULO 3

Requisitos y análisis de la solución de integración

Conforme se aproxima el fin de las enseñanzas de bachiller o similar (de acuerdo a la Clasificación Internacional Normalizada de Educación, CINE 4, educación postsecundaria no terciaria), a pesar de haber estudiado como mínimo 14 años y habiendo tenido que hacer diversas elecciones a lo largo de estos años, aparecen en el estudiante inseguridades y ansiedades, ya que entre otros posibles motivos, desconocen las opciones de estudios universitarios y no universitarios disponibles. Resulta muy frustrante estudiar algo que no te gusta, que no se ajusta a tus intereses y aptitudes. Una elección no alineada a los intereses y aptitudes personales puede conllevar un posterior abandono de los estudios o acabar en una vida laboral insatisfactoria.

Ante esta situación, los alumnos pueden disponer de diversas opciones, entre ellas: asistir a jornadas de puertas abiertas en universidades, ferias educativas, talleres y seminarios de orientación vocacional. También pueden realizar entrevistas con orientadores y/o contactar con diversos profesionales, observando y descubriendo en qué consiste el día a día de su jornada laboral.

Estas opciones, tienen entre todas ellas una cosa en común: son actividades presenciales. Esto supone que, aunque pueden llegar a ser de gran utilidad, requieren esfuerzo, tiempo y desplazamiento, lo que puede resultar tedioso y complicado para muchos estudiantes.

Por eso me propuse crear una herramienta on-line, Orienta-T, que centralice la información y facilite el acceso a recursos de consejos vocacionales. Esta herramienta ofrece al usuario comodidad y accesibilidad, ya que le podrá facilitar la toma de decisiones, le ayudará a explorar posibilidades de formaciones que quizás no había considerado y le propondrá las formaciones más acordes a sus preferencias y aptitudes sin necesidad de salir de casa, a través de internet mediante Internet, desde cualquier lugar y en cualquier momento.

3.1 Componentes

Esta herramienta tendrá los siguientes componentes:

- Sistemas:

  - Amazon RDS: se establecerá una base de datos gracias a este servicio en la nube, en la cual, almacenaremos:
    - Formaciones tanto universitarias (grados universitarios) como programas de cualificación profesional no universitaria (formación profesional de grado superior) según intereses y aptitudes. A continuación, en esta memoria, se le pasarán a nombrar como carreras, sin embargo, este concepto englobará ambas formaciones.
    - Cursos de especialización, en adelante cursos, que se ofrecerán en la herramienta según la formación que seleccione el usuario.
    - Un registro de las compras realizadas.
  - Salesforce: permitirá gestionar a los usuarios, almacenando sus datos y credenciales en una base de datos que se creará dentro de este sistema.
  - Amazon SNS: realizará la función de enviar notificaciones y confirmaciones vía correo electrónico.
  - Pasarela de pago: se utilizará para procesar pagos de servicios adicionales o cursos.

- Middleware: ejercerá el rol de intermediario entre una página web y los sistemas, esto significa que deberá de ser capaz de manipular y combinar datos de distintos tipos de manera correcta. Estará basado en las reglas de una API Rest.

3.2 Identificación y análisis de soluciones posibles

A la hora de implementar algunas funcionalidades de la herramienta, encontraremos más de una opción viable, cada una de ellas tendrá sus propias ventajas y desventajas, por lo que la elección adecuada dependerá de varios factores, como la complejidad de implementación, el coste, la escalabilidad y la robustez. En esta sección se procede a analizar las diferentes soluciones posibles para implementar funcionalidades clave como la recomendación de carreras, gestión de usuarios, de cursos y pasarela de pago, dando para ello un análisis de cada alternativa para determinar cual es la más adecuada.

El objetivo es garantizar que las decisiones tomadas no solo cumplan con los requisitos funcionales y no funcionales, sino que, además, faciliten el mantenimiento y evolución de la plataforma. A continuación se expondrán las opciones opciones a analizar de cada funcionalidad a implementar.

3.2.1. Gestión de usuarios

Para implementar la gestión de usuarios en Orienta-T se barajan tres sistemas, los cuales son:

- Amazon RDS: para ello, se creará una base de datos MySQL relacional en dicho sistema y se diseñarán tablas para usuarios, roles, permisos... Tras ello, se desarrollarán APIs que interactúen con ella para la gestión de usuarios. Esto nos permite obtener una gran flexibilidad debido a que podremos personalizarla según necesidades específicas.

- Firebase Authentication: se trata de un servicio de autenticación que facilita la validación de usuarios almacenando su información en Firestore (base de datos que ofrece Firebase alojada en la nube). Es un servicio que ofrece una rápida implementación y configuración, además de fácil integración que maneja automáticamente la escalabilidad y seguridad. Sin embargo, tendremos menor control sobre la estructura de los datos (no la podremos personalizar a nuestro gusto) y dependeremos de la infraestructura de Google, suponiendo un riesgo a largo plazo en caso de que quisiéramos migrar a otro proveedor o sistema.
- Salesforce: proporciona herramientas para la gestión de usuarios, medidas robustas de seguridad robustas con normativas de protección de datos y una infraestructura escalable. Por estas razones, podríamos almacenar y gestionar en dicho servicio los datos de los usuarios.

Por lo que, teniendo estas tres opciones presentes, llegamos a las siguientes conclusiones: tanto Firebase Authentication como Salesforce son servicios más enfocados a la gestión de usuarios que implementan medidas de seguridad y facilitan la funcionalidad de gestión. Por otra parte, si queremos simplificar la integración e implementación en Orienta-T, las opciones más adecuadas serían Salesforce y Amazon RDS, ya que el entorno con el que vamos a implementar las APIs es Anypoint Studio, el cual ofrece conectores nativos para ambos sistemas.

En consecuencia, considerando los dos factores mencionados, el sistema que nos permite la integración de ambos es Salesforce, por lo que será éste el que se use para la gestión de usuarios.

3.2.2. Recomendación de carreras

Respecto a la funcionalidad de "Recomendar" en una plataforma, hay varias opciones para realizar dicha implementación. Tras realizar una investigación, se considera que las opciones más viables y que más pueden rentabilizar el modelo de arquitectura API-Led añadiendo sistemas, son:

- Creación de un algoritmo de recomendación: esta opción se implementaría utilizando AWS Lambda, un servicio de computación en la nube que sirve para ejecutar código en respuesta a eventos. Este sistema lo tenemos disponible al poder acceder a AWS Academy. Este algoritmo puede llegar a ser complejo ya que supone aplicar técnicas de machine learning para identificar patrones y sugerir diversas carreras, utilizando para ello modelos entrenados que relacionen intereses y aptitudes con carreras. Sin embargo, para simplificar y reducir dicha complicación, se podría mockear (simular una funcionalidad para comprobar su funcionamiento) el comportamiento del algoritmo, es decir, en vez de realizar un análisis complicado, una función Lambda podría ejecutar un código que aleatoriamente devuelva las carreras recomendadas basadas en los intereses del usuario.
- Consulta a una base de datos: esta opción se implementaría usando Amazon RDS, servicio que facilita la configuración, operación y escalado de una base de datos en la nube. Dicho servicio también podrá ser utilizado al tener acceso a AWS Academy. Para ello, se tendrá que crear una base de datos en el sistema mencionado, seleccionando el motor de base de datos que más se adecúe. Tras ello, se diseñará la base de datos, creando una relación entre carreras e intereses. De esta manera se podría realizar un matching de intereses, usando para ello consultas SQL que busquen carreras según el interés del usuario.

Teniendo en cuenta estas dos opciones, tras analizar como se realizarían ambas implementaciones, la dificultad que supondrían y las ventajas e inconvenientes que supondrían cada alternativa en la herramienta; se considera que la opción más completa es hacer una mezcla de ambas. Es decir, tener en la base de datos una tabla para almacenar carreras y que estas carreras se le pasen, juntos con los intereses que seleccione el cliente, a Lambda.

En este trabajo no se desarrollará el algoritmo que devuelva la carrera recomendada en base a los intereses, pero se ofrecerán conectores para que se haga correctamente la llamada al sistema, adjuntando en ella los datos que necesite.

3.2.3. Gestión de cursos

Como se ha comentado en el apartado anterior, en Amazon RDS se van a almacenar las carreras e intereses ya que debe de haber una relación entre ellas para que se realice correctamente la funcionalidad de recomendación. Como queremos que en la herramienta se pueda realizar una filtración de cursos por carreras relacionadas, deberá de existir una conexión entre ambos conjuntos de datos; en conclusión, para no complicarnos estableciendo dicho vínculo, utilizaremos la base de datos creada en Amazon RDS.

Sin embargo, en caso de no tener que relacionar carreras con cursos o en un contexto en el que éstos no estuvieran almacenados en Amazon RDS, la gestión de cursos se podría efectuar mediante otras bases de datos como MongoDB, Google Firebase... Aunque solo se hayan mencionado soluciones basadas en bases de datos (debido a que es la opción con la que más familiarizados estamos) para componer esta funcionalidad, existen varias alternativas:

- Sistema de Gestión de Contenidos: para ello, habría que instalar plugins específicos para la funcionalidad que se quiere implementar y crear cursos como si fueran páginas en el sistema, incluyendo descripción y más datos. Ejemplos de este sistema son WordPress, Drupal y Joomla.

- Plataformas de Aprendizaje en Línea: se podrán crear desde ellas cursos y gestionarlos, ya que debido a que son plataformas diseñadas específicamente para la educación en línea, disponen de herramientas específicas para diseñar módulos o temas. Además, esta opción nos permitiría implementar calificaciones, evaluaciones automáticas e incluso recursos multimedia y foros. Esta solución sería muy adecuada para cumplir la función de gestión de usuarios si no fuera por la complejidad de uso y administración que supone. Algunos ejemplos son Moodle, Blackboard o Canvas.

- APIs de terceros: también facilita la creación de cursos como la alternativa anterior y también proporciona herramientas para integrar los pagos y los precios. El uso de estas APIs puede suponer un costo basado en suscripciones. Algunas de estas APIs pueden ser de Thinkific (plataforma de creación y venta de cursos en línea que permite a los usuarios diseñar y comercializar sus propios cursos) o Idemy for Business (versión de Udemy pero orientado a empresas, permite gestionar cursos en línea).

3.2.4. Notificar al usuario

Para notificar al usuario tras haber creado su cuenta de manera correcta encontramos varios sistemas, todos están basados en la nube y en el envío de correos electrónicos.

Los sistemas a elegir son Amazon SNS, SendGrid y Mailgun, de entre todos ellos, el que más nos conviene utilizar es Amazon SNS ya que es el único que se encuentra integrado con la infraestructura AWS (la cual tenemos disponible), permitiendo esto una implementación fluida dentro del conjunto de servicios que estamos utilizando, como lo son Amazon RDS y Anypoint. Nos permitirá manejar grandes volúmenes de correos electrónicos en caso de que lo necesitamos o el proyecto se expanda en un futuro, asegurando así un alto grado de escalabilidad y fiabilidad.

3.2.5. Pasarela de pago

Existen diversas opciones para implementar la pasarela de pago. Cada opción tiene sus ventajas y desventajas en términos de integración, seguridad y costes. Algo que comparten todas las plataformas que se mencionarán es la robusta seguridad que ofrecen, la cual es esencial para proteger la información sensible de los usuarios y garantizar transacciones seguras. A continuación se presentan las alternativas más viables y sus características clave:

- PayPal: una de las plataformas de pago on-line más populares y utilizada en la actualidad. Para ello, configuraremos una cuenta de negocios en dicha plataforma y haciendo uso de las APIs que ofrece integraremos la pasarela de pago en Orienta-T. Es una implementación que no supondría mucha dificultad ya que las APIs están bien documentadas. Sin embargo, Paypal puede llegar a retener fondos en determinadas ocasiones y por cada transacción se carga un coste que puede ser elevado.

- Stripe: otra plataforma de pago más enfocada a los desarrolladores. También tendríamos que crear y configurar una cuenta para obtener las claves API, las cuales podremos utilizar para procesar pagos, gestionar suscripciones y manejar reembolsos, de manera opcional, también se podrán implementar mecanismos (webhooks) para recibir notificaciones de eventos. A diferencia de PayPal, Stripe nos permite personalizar totalmente el flujo de pago, ofreciendo así una mejor experiencia al usuario.

- Square: su función e implementación es fácil, muy parecida a la de PayPal, además ofrece su propia API, como el resto de plataformas mencionadas. Sin embargo, no ofrece funcionalidades avanzadas como lo hace Stripe y no está disponible en todos los países, limitando así el acceso para algunos de nuestros usuarios.

Por lo que, la pasarela de pago que se decidirá utilizar en el desarrollo de este trabajo será Stripe debido a la API que ofrece y la facilidad con la que se puede integrar con Anypoint Studio, la cual veremos en el apartado de la implementación de las APIs.

3.3 Procesos del usuario

Un usuario accederá a la plataforma Orienta-T y se encontrará con que puede obtener recomendaciones de carreras, consultar cursos y comprarlos (figura 3.1).

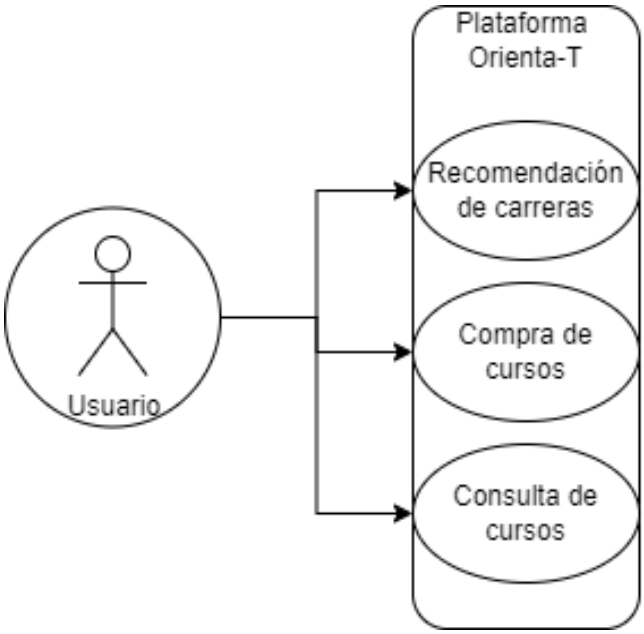

Figura 3.1: Caso de uso Orienta-T

Teniendo los componentes de la herramienta en cuenta, se podrán indicar los pasos que seguirá el usuario, más detalladamente, cuando acceda o quiera hacer uso de alguna funcionalidad de Orienta-T.

El usuario podrá hacer las siguientes solicitudes a la herramienta,las cuales se realizarían mediante una página web: crear una nueva cuenta en caso que sea la primera vez que hace uso de Orienta-T. Si ya tiene una cuenta, podrá iniciar sesión introduciendo sus credenciales establecidas. Ambas operaciones se harían gracias a la base de datos alojada en Salesforce, ya que en ella se guardarán los datos de los usuarios. Hay que incidir en que una vez el usuario haya creado su cuenta, recibirá una propuesta de confirmación y notificación de dicha acción por correo, gracias a la implementación del sistema Amazon SNS.

Una vez el usuario haya accedido con su cuenta a Orienta-T, podrá hacer uso de las funcionalidades de recomendación de carreras y de la búsqueda y compra de cursos.

Si quiere que se le recomienden carreras, tendrá que rellenar un cuestionario diseñado para conocer mejor sus preferencias y aptitudes que deberá de completar y enviar. Tras ello, se procesará el algoritmo de recomendación de Lambda y se le mostrará al usuario cual es la formación que más se adecua a las respuestas que ha dado en el formulario.

En cambio, si opta por seleccionar la búsqueda de cursos, tendrá la opción de filtrarlos los cursos por carreras relacionadas, utilizándose para ello los datos que se hayan en Amazon RDS. Si selecciona uno de ellos, le aparecerá una breve descripción además de más detalles y si finalmente decide comprarlo, se realizará el pago mediante la pasarela de pago implementada (Stripe).

Los procesos mencionados se pueden ver en el esquema ofrecido en la figura 3.2

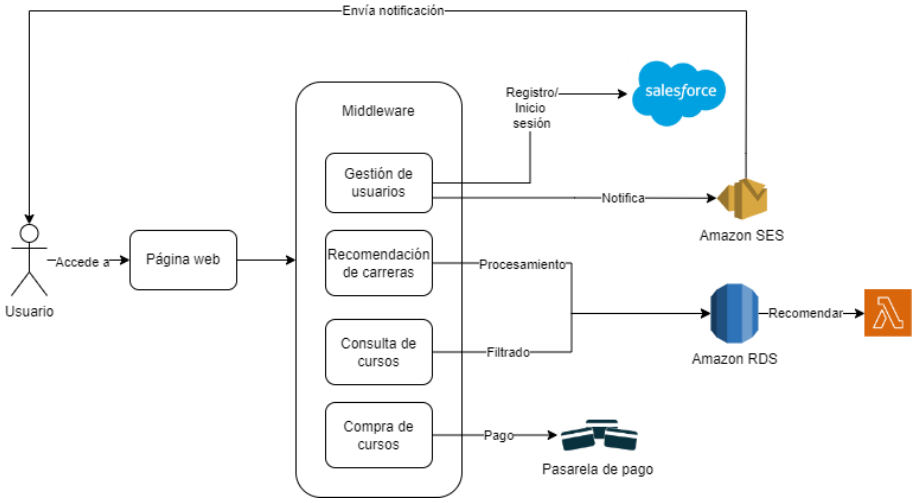

Figura 3.2: Esquema Orienta-T

CAPÍTULO 4

Diseño

Para la implementación de la herramienta, Orienta-T, se ha elegido, como ya se ha mencionado una arquitectura API-Led basada en servicios REST, permitiendo que los diferentes componentes del sistema se comuniquen entre sí a través de APIs bien definidas, facilitando la integración y la reutilización de servicios. En la figura 4.1 se puede ver cuál será el diseño de la arquitectura y como se conectarán entre si las APIs y sistemas.

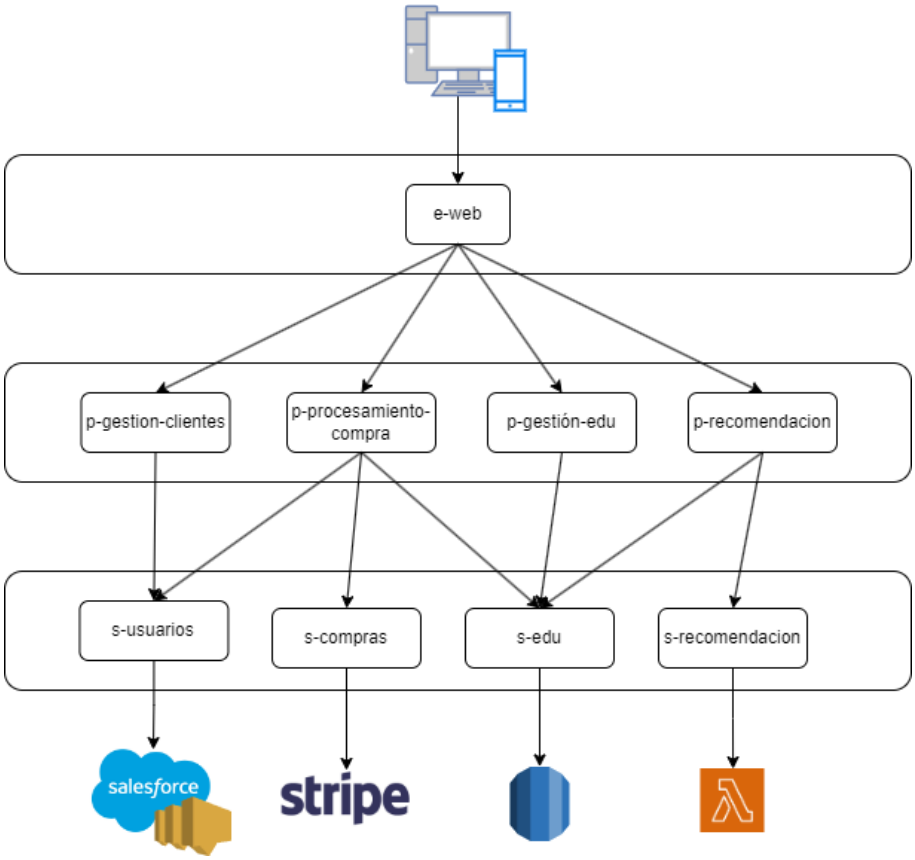

Figura 4.1: Arquitectura API Led

Una vez vista la arquitectura, procederemos a describir detalladamente cada API y su función según la capa en la que se halle.

Podremos ver en qué capa se sitúa la API, ya que para darles nombre, se ha seguido la convención de nombres con la que trabaja Mulesoft. Esta convención dicta que, según la capa a la que pertenezcan, tendrá que empezar por una letra u otra. Es decir, si queremos poner nombre a una API que se encuentra en la capa de procesos, su nombre deberá de empezar con la letra 'p'. Ese será nuestro primer elemento, el segundo será una palabra descriptiva de los objetos que manipulará, como 'compras' para una API que maneja procesos de compra o 'usuarios' para una que maneje el registro de usuarios. El último elemento será la palabra 'api'. Finalmente, cabe indicar que estos tres elementos estarán separados por guiones.

La descripción de las APIs comenzará con la capa de sistema y finalizará con la capa de experiencia. Este enfoque se debe a que las APIs se implementan en ese orden, empezando desde la base. Además, la capa de sistemas es la más importante, ya que muchas otras APIs dependen de ella.

4.1 APIs en la capa de sistema

En esta capa se encuentran las APIs encargadas de interactuar directamente con los sistemas y sus datos. Es la capa en la que más APIs hay, una por sistema, a excepción de Amazon SNS, sistema que se controlará con la misma API que la de Salesforce.

4.1.1. s-edu-api

Es la encargada de comunicarse con la base de datos almacenada en Amazon RDS, ya hemos comentado que, en ella, se almacenarán las carreras, los cursos y un registro de las compras realizadas. Sin embargo, esta API sólo utilizará los datos relacionados con los cursos y las carreras, almacenándolos, recuperándolos y actualizándolos.

Opera con los siguientes objetos: "Cursos", los cuales tienen un atributo id, nombre, precio y formación relacionada asociados, "CursosRespuesta", con los mismos atributos que "Cursos" a diferencia de la formación relacionada (la cual usaremos como URI param para implementar un endpoint), "Carreras" (aunque tenga el nombre de carreras, englobará tanto formaciones universitarias como no universitarias), el cual consta de un nombre y una descripción, y "Compras", con atributos nombre de usuario, de curso y fecha de compra.

En esta API se implementan los siguientes endpoints, que son URLs específicas donde se pueden realizar solicitudes para interactuar con los datos de la API:

- "GET/cursos": se devolverán todos los cursos que se encuentren en la base de datos, sin realizar ningún filtro sobre ellos. Se devolverá en formato JSON el objeto "CursosRespuesta".

- "POST/cursos": servirá para crear un curso, para ello, le tendremos que hacer llegar a nuestra API un id, nombre, precio y carrera relacionada del nuevo curso que queramos. Permitirá crear cursos más fácil y rápidamente, sin necesidad de realizar entradas manuales en la base de datos mediante el uso de la herramienta MySQL Workbench.

- "GET/cursos/carreraRelacionada": este endpoint nos permitirá buscar cursos en base a una carrera relacionada, la cual se pasará como URI param. Será de gran utilidad una vez el usuario sepa cuál es la carrera que más se adecúa a sus intereses y quiera buscar cursos que correspondan con ella. También devolverá un array de "CursosRespuesta" como en el GET normal.

- "GET/cursos/nombre": es igual que el endpoint anterior, a diferencia de que, esta vez, buscaremos en base a un nombre. Esto nos servirá para implementar la funcionalidad de comprar cursos, ya que se le pasa el nombre de un curso, el cual buscaremos mediante este endpoint.

- "GET/carreras": devolverá todas las carreras que se encuentren en la base de datos, las cuales se le pasarán a la API encargada de la recomendación.

- "POST/carreras": añadimos un endpoint para crear nuevas instancias de carreras en la base de datos RDS en caso de que se consideren necesarias.

- "POST/compra": añadirá un nuevo registro de compra, requiriendo para ello los atributos que forman parte del objeto "Compra". Se llamará a este endpoint cuando se haya realizado una compra de manera exitosa.

4.1.2. s-usuarios-api

La API s-usuarios-api interactúa con Salesforce y Amazon SNS es decir, se encarga de la gestión de usuarios.

Utiliza los siguientes objetos: "Usuarios" el cual tiene como propiedades nombre, contraseña, número de teléfono y correo electrónico al cual se le mandará la notificación tras crear su cuenta, y "UsuarioRespuesta" el cual, a diferencia del objeto anterior, no tiene contraseña asignada.

Los métodos de esta API son los siguientes:

- "GET/usuarios": devolverá todos los usuarios que ya tengan una cuenta creada, es decir, que estén registrados en Orienta-T y se encuentren en la base de datos de Salesforce (lugar en el que se almacenan los datos de ellos).

- "POST/usuarios": servirá para crear un usuario nuevo en la base de datos de Salesforce; es el endpoint encargado de realizar la funcionalidad de registrar a los nuevos usuarios. Este método requiere que se le pasen las propiedades del objeto 'Usuarios'.

  Más adelante, en el capítulo del desarrollo de la solución (capítulo 6), se podrá ver como, tras crear el nuevo usuario, se llamará a Amazon SNS para que se encargue de su notificación.

- "GET/usuarios/nombre/contraseña": se le pasará como URI param el nombre y la contraseña, tras ello se comprobará si existe un usuario en la base de datos de Salesforce cuyos datos coincidan con las credenciales introducidas. Devolverá el objeto "UsuarioRespuesta", si lo devuelve vacío (sin datos) significará que no existe dicho usuario, en caso contrario, si tiene contenido, será que si existe.

4.1.3. s-recomendacion-api

Es la API encargada de realizar la llamada a Lambda en el cual se ejecutará el algoritmo para realizar la recomendación de carreras.

Utiliza los objeto “Carreras” (tiene propiedades como id, nombre, descripción e intereses), “Intereses” (se le añadirán las propiedades que se consideren convenientes, por ejemplo: preferencias, motivacionales, expectativas, inclinaciones o interés por algún tema o actividad ocupacional, capacidad o habilidades potenciales para realizar eficazmente determinados estudios y/o actividades profesionales u ocupacionales, asignatura favorita), “SolicitudRecomendacion” (solo tiene como propiedades los intereses) y “RespuestaRecomendacion” (consta de un array del objeto “Carreras”).

Solo dispone de un endpoint: “POST/recomendar”, al cual se le entregan los intereses seleccionados (el objeto ’“SolicitudRecomendacion”), y nos propondrá las carreras que devuelva el algoritmo ejecutado en Lambda que más se adecúen a ellos (el objeto “RespuestaRecomendacion”).

4.1.4. s-compras-api

Esta API se encarga de procesar las transacciones financieras a través de la pasarela de pago.

El objeto principal que maneja esta API es “Compras”, el cual incluye varias propiedades esenciales para la transacción de compra: el nombre y precio del curso que se va a comprar, el nombre del usuario que va a realizar la compra y el token de Stripe, el cual se crea a partir de la información de pago del usuario (número de tarjeta, CVV, titular, fecha de vencimiento), aportando una seguridad extra al intercambiar información sensible de pago.

La API define un solo endpoint, /compra, que acepta solicitudes POST. Este endpoint sirve para la creación de nuevas compras y el procesamiento del pago correspondiente. El cuerpo de la solicitud debe ser un objeto JSON que siga la estructura definida por el tipo Compras. La documentación del endpoint incluye una descripción clara de su propósito y las posibles respuestas que puede generar. El cuerpo de la solicitud debe ser un objeto JSON que siga la estructura del tipo “Compras”.

En cuanto a las respuestas, la API puede devolver diferentes mensajes:uno que indica que la compra se ha almacenado habiéndose procesado el pago correctamente y otro que indique que ha ocurrido un error durante el procesamiento del pago.

4.2 APIs en la capa de proceso

4.2.1. p-gestion-edu-api

Esta API se ocupa de la lógica de negocio requerida para la creación y actualización de la oferta de cursos y carreras. Por lo que simplemente extenderá los endpoints de la API de sistema encargada de interactuar con Amazon RDS que interactúen con dichos objetos, es decir, todos menos la solicitud POST del objeto compras.

Además de especificar los endpoints, deberemos especificar los objetos con los que opera, es un proceso común a todas las APIs, no solo a las de sistemas.

4.2.2. p-gestion-clientes-api

Esta API, también propagará los endpoints de solo una API (figura 4.1), como la anterior, es decir, los encargados de obtener todos los usuarios, de recuperarlos en base a un nombre y contraseña dados, y de crearlos.

4.2.3. p-procesamiento-compra-api

Para esta API, necesitaremos objetos provenientes de más de una API de sistemas, a diferencia de las dos anteriores.

Primero, necesitaremos tener acceso a la base de datos de RDS para poder obtener el precio del curso que se pretende comprar, por lo que deberemos de propagar el endpoint de "GET/cursos/nombre" proveniente de la API s-edu. Además, como queremos registrar en una base de datos algunos detalles de las compras realizadas, especificaremos el "POST/compras" de dicha API.

También requeriremos de datos del usuario, por lo que añadiremos uno de los endpoints de la API de sistemas que interactúa con Salesforce: s-usuarios.

Finalmente, extenderemos el endpoints de s-compras, "POST/compra" de s-compras, el cual requerirá para su implementación los objetos obtenidos mediante las otras APIs de sistema mencionadas.

4.2.4. p-recomendacion-api

El objeto 'Carreras' que se necesita para realizar la recomendación (según se ha comentado en la sección 4.1.3), se obtiene de la API s-edu, por lo que tendremos que crear una instancia al endpoint "GET/carreras" de dicha API en esta capa de procesos.

Como los intereses provienen de la página web, y no de ningún sistema, no tendremos que añadir otro endpoint para obtenerlos, por lo tanto, solo faltará añadir el endpoint "POST/recomendacion" de la API s-recomendacion.

4.3 API en la capa de experiencia

En esta capa solo encontraremos una API y esta tomará el rol de intermediaria entre una página web y las APIs de proceso. Por lo tanto, deberá ser capaz de manejar las solicitudes entre una interfaz de usuario y encaminarlas a los endpoints adecuados en la capa de proceso. Esta capa es fundamental para proporcionar al usuario una experiencia fluida y eficiente cuando haga uso de Orienta-T.

Esta API recibirá el nombre de e-web-api y actuará como punto de entrada para todas las interacciones del usuario con la plataforma, es decir, gestionará las solicitudes que le lleguen desde la página web.

Cuando el usuario introduzca sus datos de registro o inicie de sesión en Orienta-T, esta API recibirá dicha información y la enviará mediante el endpoint “POST /usuarios” a la p-gestion-clientes-api para su procesamiento.

Si el usuario introduce sus respectivos intereses mediante una solicitud POST, se dirigirá dicha solicitud al endpoint “POST/recomendar”.

Sin embargo, si el usuario decide consultar todos los cursos disponibles o alguno en específico, la API dirigirá la solicitud a través de los endpoints “GET /cursos” o ”GET /cursos/nombre” o “GET /cursos/carreraRelacionada” a la p-gestion-cursos-api para poder acceder a la base de datos de Amazon RDS y devolver la información solicitada.

Una vez el usuario haya seleccionado un curso que le interese adquirir, se obtendrán todos los datos para realizar la transacción y se llamará al endpoint “POST/compras” de la API p-procesamiento-compra.

4.4 Diagramas de flujo

Ahora que ya conocemos la estructura de nuestro proyecto, las APIs que lo conforman, sus responsabilidades y los endpoints que ofrece, podemos hacer un seguimiento más detallado de los flujos que se producen al ejecutarse las funcionalidades.

Los diagramas de flujo mostrados nos permitirán visualizar cómo interactúan los diferentes componentes y cómo se manejan las solicitudes y respuestas a través del sistema.

Los analizaremos de uno en uno y en base a la funcionalidad a la que correspondan.

Cabe destacar que se supondrá la existencia de una página web para entender mejor el envío de solicitudes.

4.4.1. Gestión y notificación de usuarios

En la figura 4.2 se muestra un diagrama de flujo del registro de un usuario nuevo en Orienta-T y su respectiva notificación.

En el primer paso, el usuario introducirá sus datos (nombre, número de teléfono y correo electrónico) en el formulario de registro en una página web. La página enviará al endpoint “POST/usuarios” una solicitud a e-web-api con los datos del usuario.

La API de experiencia reenviará la solicitud a p-gestion-clientes y ésta le hará llegar los datos a s-usuarios, interactuando esta con Salesforce, pero, antes de ello, en esta misma API, se deberá de realizar un mapeado de datos.

En el capítulo de desarrollo de la solución (6) veremos cómo desde Anypoint Studio, se gestiona la creación de una cuenta en Salesforce con los datos proporcionados. Una vez se haya creado la cuenta, se realizará una llamada a Amazon SNS con el correo electrónico del usuario recién

registrado. Este proceso permitirá enviar un correo al usuario solicitando la verificación de su registro. Si el usuario confirma que, efectivamente, ha sido él quien se ha registrado, se le enviará otro correo indicando que su cuenta ha sido creada correctamente.

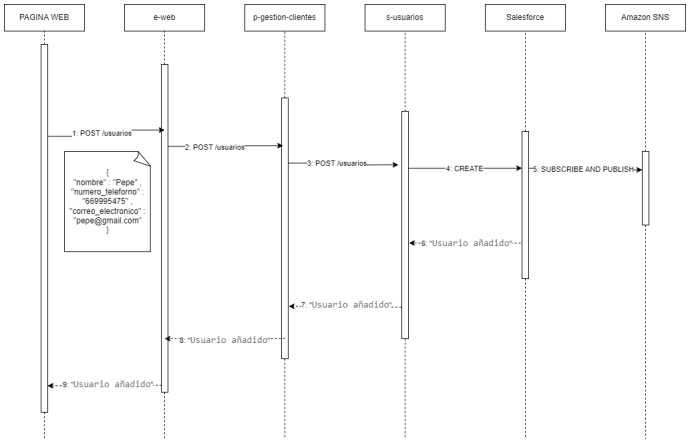

Figura 4.2: Diagrama de flujo del registro y notificación de usuario

Por otra parte, el flujo de inicio de sesión (figura 4.3) dará comienzo una vez el usuario introduzca su nombre y contraseña en una página web, enviando tras esto una solicitud. Dicha solicitud se realiza mediante un HTTP GET a la API de experiencia con el nombre y contraseña introducidos como parámetros.

La API e-web reenvía a p-gestion-clientes los parámetros y ésta, vuelve a enviarlos a s-usuarios. En esta última API es donde se accederá a la base de datos de usuarios alojada en Salesforce y se comprobará si existe o no un usuario con dicho nombre y contraseña. Si existe, se devolverá y mostrará en la página un mensaje en el que se indica que el usuario ha iniciado sesión correctamente, y en caso de que no esté registrado, se indicará que las credenciales proporcionadas no son válidas.

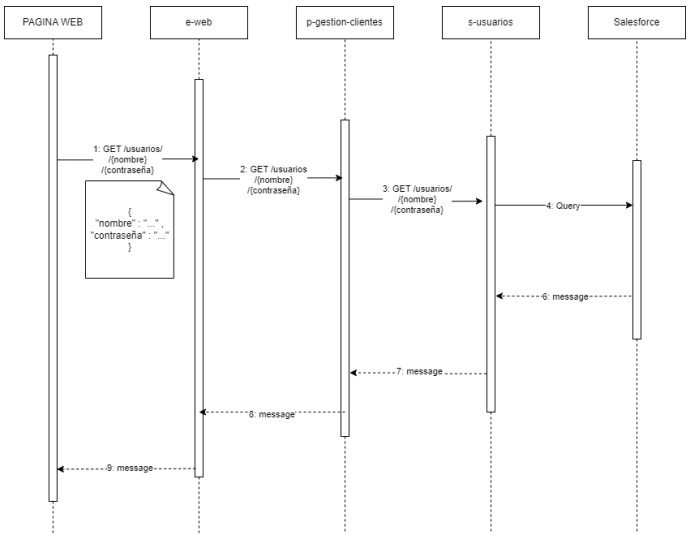

Figura 4.3: Diagrama de flujo del inicio de sesión

4.4.2. Búsqueda y compra de cursos

En esta sección, veremos los flujos de las funcionalidades relacionadas con el acceso a la base de datos alojada en Amazon RDS, es decir, la búsqueda de los cursos y su posterior compra.

El siguiente diagrama (figura 4.4) muestra el proceso de búsqueda de cursos que se ofertan en Orienta-T.

Para comenzar, el usuario, indicará la formación de la cual le interesan conocer cursos relacionados, este atributo se usará como URI param y se enviará en una solicitud a e-web-api al endpoint "GET /cursos/carreraRelacionada".

Las APIs seguirán el flujo que se ha indicado en la figura 4.1 hasta llegar a s-cursos. En dicha API, se realizará una consulta SQL en la base de datos Amazon RDS, devolviendo así una lista de cursos que cumplan con lo establecido en la consulta.

Antes de mostrarlos en la página web, los datos devueltos se transformarán a formato JSON.

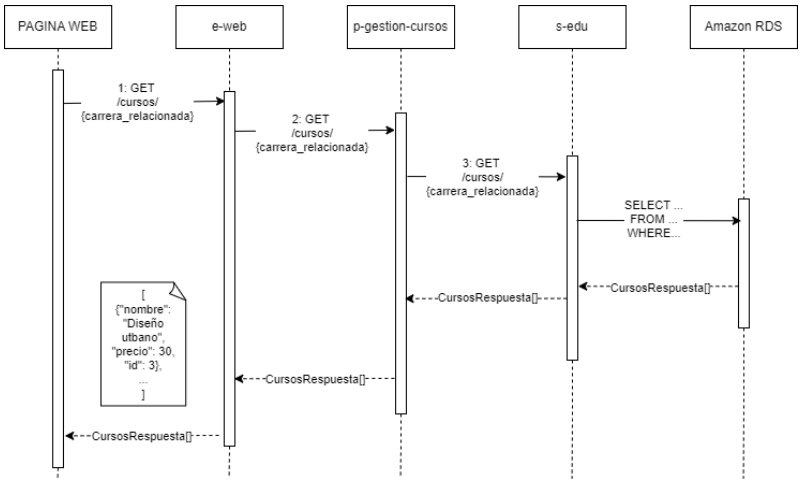

Figura 4.4: Diagrama de flujo de la búsqueda de cursos por carrera

El flujo que se seguirá para realizar una compra será el mostrado en la figura 4.5: el usuario realizará una solicitud POST al endpoint /compra desde la página web, recibiendo así a la API de experiencia, el nombre del usuario, el nombre del curso que ha decidido comprar y un token.

Se le redirigirán los datos a p-gestion-compras, la cual obtendrá mediante una solicitud GET a s-edu en el endpoint /cursos/nombre, utilizando para ello el nombre del curso que se pretende comprar, el precio del mismo, almacenado en la base de datos de RDS.

Una vez se ha obtenido el curso, se podrá llamar a s-compras-api, la cual será la encargada de transferir los datos a Stripe para efectuar la compra.

Si la compra se realiza de manera satisfactoria, se le pasarán los datos de el usuario que realizó la compra, el curso que se adquirió y la fecha en la que se realizó a p-gestión-compras, y éste, a su vez, a s-edu-api, para almacenar en el registro de las compras en Amazon RDS la nueva transacción.

Y, finalmente, se devolverá un mensaje de confirmación o de error a la página web, indicándole al usuario si la compra se ha procesado exitosamente o no.

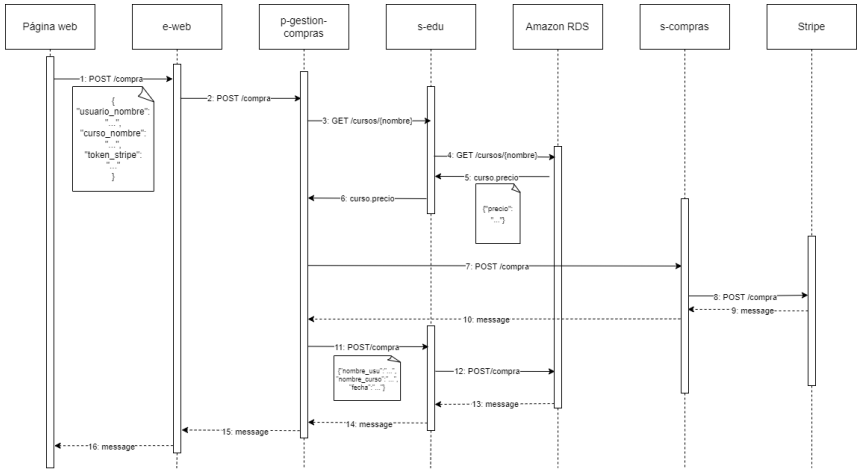

Figura 4.5: Diagrama de flujo de la compra de cursos

4.4.3. Recomendación de carreras

Para realizar la funcionalidad de recomendación de carreras, se seguirá el flujo que se muestra en la figura 4.6. En ella, podemos ver como el primer paso consiste en enviar los intereses ingresados en el formulario mediante una solicitud HTTP POST a la API de experiencia.

Dichos intereses se reenviarán a la API de proceso encargada de gestionar la funcionalidad de la recomendación. Para esta funcionalidad se requerirán, además de los intereses, una lista de carreras, las cuales, se obtendrán, realizando una solicitud GET a s-edu-api.

Una vez, se le hayan enviado a p-gestion-recomendaciones todas las que se ofertan en la herramienta almacenadas en la base de datos de Amazon RDS, podrá enviar a s-recomendacion, tanto las carreras como los intereses.

Con estos datos, se llamará a Lambda, el cual procesará la información para encontrar la formación más adecuada basada en los intereses del usuario. Después de este proceso, se le hará llegar la respuesta al usuario mostrándosela en la página.

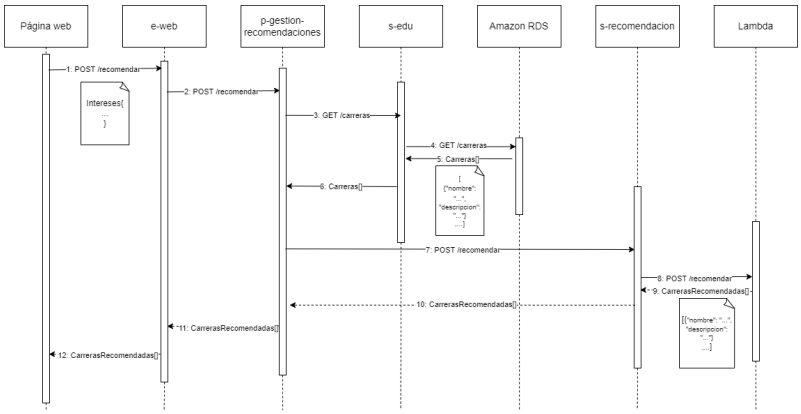

Figura 4.6: Diagrama de flujo de la recomendación de carreras

CAPÍTULO 5
Tecnología utilizada

En este capítulo se describirán las tecnologías, herramientas y frameworks (estructura de soporte que permite un desarrollo más sencillo) que se utilizarán para desarrollar la plataforma Orienta-T. La elección de las herramientas que se van a mencionar se basa en la facilidad de implementación, compatibilidad con otros sistemas y el conocimiento previo de ellas.

Cabe mencionar que ya tengo experiencia con la mayoría de estas tecnologías. No solo he utilizado Anypoint Studio en las prácticas de la asignatura de "Integración de Aplicaciones", sino que durante la cátedra también aprendí y usé muchas de estas herramientas. Esta experiencia previa ha facilitado el desarrollo de este trabajo.

A continuación, se describirán detalladamente las tecnologías y herramientas que se emplearán, explicando el papel que se pretenda que tengan en el proyecto y los beneficios que aportarán.

5.1 Anypoint Platform

Es una solución, desarrollada por MuleSoft, para integrar aplicaciones y datos. Esta plataforma nos será de gran utilidad ya que ofrece un conjunto de herramientas y servicios diseñados para simplificar la creación, gestión y monitoreo de APIs y flujos de integración. [20]

En Anypoint Platform nos encontraremos diversos módulos que nos podrán resultar útiles, entre ellos, cabe destacar [21]:

- Design Center: desde este entorno se nos permitirá crear nuevas especificaciones de APIs, definiendo como queremos que se comporte y como se comunique con el resto de APIs y de los sistemas. Podremos elegir el lenguaje en el que queremos especificar la API, pudiendo escoger entre las distintas versiones de RAML y OAS. Para este proyecto se optará por RAML ya que su sintaxis e más clara, facilitando así tanto su lectura como escritura. Además, podremos generar documentación de la API fácilmente y reutilizar fragmentos y patrones comunes.

  Desde esta herramienta y gracias a RAML, podremos establecer los objetos con los que operará la API, los endpoints, el protocolo y el tipo de formato de los datos.

  Otra función disponible será añadir dependencias, permitiendo así, si fuese preciso, hacer uso de bibliotecas, módulos y conectores adicionales.

Una vez hayamos finiquitado la especificación de la API, podremos publicarla al módulo Exchange, del cual se hablará a continuación.

- Exchange: actúa como repositorio y en él podremos encontrar las APIs que hayamos especificado y publicado. También, hay un filtro, el cual nos permite buscar otros elementos que se hayan publicado en el Exchange por otras personas, no solo dsitintos tipos de APIs (REST, SOAP y HTTP), también veremos que hay conectores, plantillas (patrones de integración que abordan casos de uso comunes, por lo que sería preciso añadir la información específica de nuestro escenario), políticas... Es esta una de las razones por las que se considera la integración de aplicaciones un proceso basado en la reutilización.

  Gracias a esta herramienta, podremos importar las APIs a Anypoint Studio de una manera sencilla para realizar su implementación.

- API Manager: se trata de una herramienta fundamental para gestionar, desde una única consola, el ciclo de vida de las APIs (consta de las siguientes 6 etapas: diseño, publicación, implementación, monitoreo, mantenimiento y retiro). Nos permitirá aplicar políticas de seguridad (por ejemplo de autenticación) y controlar el tráfico, evitando así el uso excesivo de las APIs, así como visualizar métricas en gráficas relacionados con los tiempos de respuesta (estos datos son útiles para optimizar el rendimiento) entre otras funciones.

- Runtime Manager: una vez las APIs ya estén implementadas, deberemos de subirlas a la nube. Ahí es donde aparece esta herramienta, permitiendo un despliegue, gestión y monitorización de aplicaciones y APIs.

  Respecto al despliegue, las opciones que encontraremos disponibles serán: CloudHub, Runtime Fabric, Hybrid y Private Cloud Edition.

  De las opciones mencionadas , la que más nos interesará es la de CloudHub al tratarse de la alternativa más rápida y rentable, ya que MuleSoft cubre los costos de infraestructura. Además, CloudHub permite seleccionar la cantidad de vCores necesarios según los recursos requeridos. Para el desarrollo de Orienta-T, se utilizará la opción de 0.1 vCore, al satisfacer ésta adecuadamente las necesidades del proyecto.

5.2 Anypoint Studio

Una vez especificadas y publicadas en Exchange las APIs procederemos a implementarlas y para ello, se utilizará un entorno de desarrollo, proporcionado por MuleSoft, como Anypoint Platform.

Ambos entornos están altamente integrados, pudiendo crear fácilmente un nuevo proyecto Mule en Studio importando una API publicada en Anypoint Exchange. Dicho proceso será el que seguiremos para tener la API en Studio y desde este entorno se desarrollará su implementación.

Podremos ver los flujos de los métodos HTTPs definidos en la API (GETS y POSTS), configurarlos y añadir lógica adicional según sea necesario, para ello, Anypoint Studio ofrece una amplia gama de conectores ya construidos para integrar con diferentes sistemas y servicios, entre ellos,

encontraremos algunos específicos para sistemas que se usarán para desarrollar Orienta-T como lo son Salesforce y Amazon RDS.

Según la función que queramos lograr o con que elemento queramos interactuar utilizaremos un tipo de conector u otro, alguno de estos tipos son de bases de datos (para conectar con base de datos), de servicios web (para interactuar con APIs REST), Saas (para integrarse con aplicaciones como Salesforce) y de mensajería (para integrar con sistemas de mensajería como Amazon SNS).

Los conectores se podrán configurar de manera detallada para adaptarse a las necesidades específicas de cada integración. Esta configuración abarca desde propiedades básicas, como nombre o descripción, hasta la introducción de credenciales para autenticar el servicio externo, ajustar tiempos de espera,límites de solicitudes...

Durante la implementación, lo más probables es que tengamos que realizar una transformación de datos. Sin embargo, esto no supone un problema para Anypoint Studio, ya que ofrece un lenguaje, DataWeave, que soluciona estas necesidades de manera eficiente.

DataWeave es un lenguaje de transformación de datos ofrecido por MuleSoft, diseñado específicamente para realizar dicha función entre diversos formatos, como XML, JSON, CSV, entre otros. Su sintaxis intuitiva permite definir transformaciones complejas con facilidad, asegurando que los datos se adapten a los requisitos específicos de cada sistema y flujo de integración.

Una vez configurados los flujos y conectores, los proyectos Mule pueden ejecutarse directamente desde Anypoint Studio. Esto nos permitirá probar y depurar las integraciones en un entorno controlado antes de su despliegue en CloudHub, para ello tendremos herramientas de depuración para identificar y solucionar problemas.

5.3 Amazon Web Service

AWS es una plataforma de servicios en la nube, ofreciendo, entre otros, servicios de computación, almacenamiento, bases de datos, análisis, redes, móviles, herramientas de desarrollo y más.

AWS ofrece una iniciativa educativa llamada AWS Academy, la cual proporciona recursos y programas de formación para adquirir habilidades en la nube. También ofrece un entorno de laboratorio: AWS Academy Learner Lab, desde el cual podremos hacer uso de los sistemas de AWS como Amazon RDS y Amazon SNS.

Para poder hacer uso de los sistemas, será preciso acceder al entorno de sandbox de AWS y procederemos a encender la máquina virtual, desde la cual podremos buscar y utilizar dichos sistemas. Resultará de gran importancia, tener guardados o anotados algunos detalles de la máquina, el “awsAccesKey”, “awsSecretAccessKey” y “awsSessionToken” para configurar las conexiones con los sistemas en nuestro proyecto Mule.

5.4 Postman

Postman es una herramienta que nos será de gran utilidad para realizar las pruebas de las APIs y verificar su funcionamiento ya que permite enviar solicitudes HTTP,tanto GETs como POSTs

(dándonos la opción de que formato queremos utilizar para mandar la información), y ver las respuestas de manera clara y organizada.

Para probar los endpoints de las implementaciones de Orienta-T con Postman, introduciremos el enlace donde se encuentra la API desplegada en Cloudhub, por ejemplo: "http://p-gestioncompras-api.us-e2.cloudhub.io/api/usuarios". Tras ello seleccionaremos que método querremos utilizar, configurando antes los encabezados, parámetros y cuerpo de la solicitud. Mandaremos la petición y verificaremos las respuestas.

5.5 MySQL Workbench

MySQL Workbench es una herramienta de administración que proporciona una interfaz gráfica para trabajar con bases de datos MySQL.

En este proyecto, se utilizará para conectarnos a la instancia de MySQL creada con AWS y alojada en Amazon RDS. Mientras que Amazon RDS facilita la configuración, operación y escalado de la base de datos en la nube, MySQL Workbench nos permitirá administrarla de manera gráfica y ver los cambios que realicemos al ejecutar endpoints específicos.

La conexión se configurará proporcionando las credenciales y el endpoint de la instancia RDS.

Con esta herramienta también diseñaremos el esquema de la base de datos, incluyendo la creación de tablas, definición de relaciones entre ellas y claves ajenas necesarias para asegurar la integridad.

Otra de las funciones que podremos realizar será la inserción de datos en las tablas de manera manual. Como alternativa a esto, y para simplificar dicho proceso, se implementan algunos endpoints en Orienta-T, como "POST/cursos" o "POST/carreras".

CAPÍTULO 6

Desarrollo de la solución

En este capítulo, se describirán las etapas clave del proceso de implementación y desarrollo de la solución propuesta en el capítulo anterior, comenzando con la configuración de los sistemas y la definición de las APIs necesarias.

También se abordarán aspectos como la autenticación de usuarios, la gestión de objetos en la base de datos Amazon RDS y la recomendación de carreras mediante AWS Lambda. Además, se explicará cómo se han integrado los diferentes servicios e implementado las funcionalidades de compra de cursos.

6.1 Flujos comunes

Para implementar una API, deberemos de importarla desde Exchange como proyecto Mule, al realizar dicha acción, se crearán de manera automática, en un archivo xml del nuevo proyecto, dos flujos: el flujo principal (main) y el de consola (console).

6.1.1. Flujo principal

Podemos ver un ejemplo en la figura 6.1, el cual consta de un componente (Listener) que escuchará las solicitudes entrantes en el puerto 8081 y otro (APIkit Router) que enrutará dichas solicitudes a los flujos correspondientes según hayamos especificado en la API.

Como podemos ver, hay una sección en el flujo en la que se manejan los errores, dicha sección funciona de manera que según el tipo de error que devuelva, se activará el flujo que esté asociado a él ejecutando así lo que está implementado en él. Todos los flujos de manejo de error tienen un solo componente (Tansform Message), el cual mostrará por pantalla un mensaje relacionado con el error para que el usuario sepa que es lo que falla. De esta manera aseguraremos que, por lo menos, los errores comunes se manejen de manera uniforme y que las respuestas de error sean claras y consistentes para el cliente.

Dichos tipos de errores y los mensajes que muestran son los siguientes:

- BAD REQUEST VALIDATION: se activará cuando haya errores de validación en las solicitudes.

- NOT FOUND: se dará cuando el recurso solicitado no se encuentre.
- METHOD NOT ALLOWED: en caso de que se intente utilizar un método HTTP que no esté permitido.
- NOT ACCEPTABLE: ocurrirá cuando la solicitud no es aceptable.
- UNSUPPORTED MEDIA TYPE: se accionará si se envía un tipo de medio que no concuerde con lo establecido en la especificación de la API.
- NOT IMPLEMENTED: se activará cuando se intente acceder a una funcionalidad no implementada.

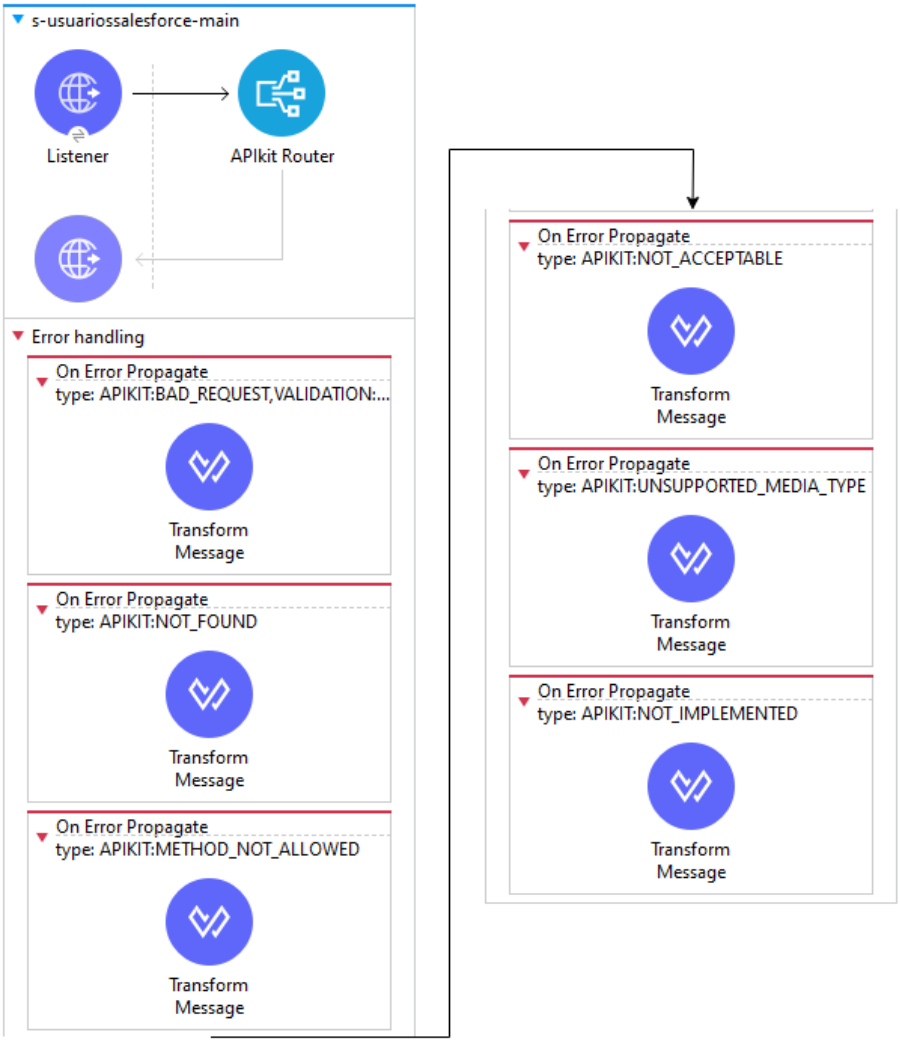

Figura 6.1: Flujo principal

6.1.2. Flujo de consola

Otro de los flujos creados automáticamente y común a todas las APIs importadas es el de consola (figura 6.2), el cual es parecido al flujo main a excepción de que el segundo componente por el que está compuesto es un APIKit Console en vez de Router, permitiéndonos tener una consola

de depuración interactiva para realizar pruebas con la API. Respecto al manejo de errores, solo encontramos un flujo que manejará errores cuando el recurso solicitado no se encuentre (NOT FOUND).

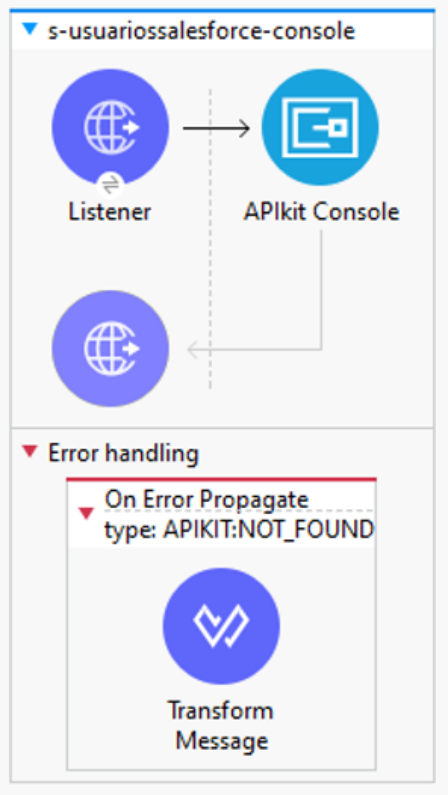

Figura 6.2: Flujo de consola

6.2 INICIO DE SESIÓN Y REGISTRO DE USUARIOS

La API de sistema encargada del inicio de sesión y registro de los usuarios es s-usuarios-api, la cual especificaremos en el Design Center de Anypoint Platform con el lenguaje RAML (proceso común a todas las APIs).

En la especificación deberemos indicar, como se ha mencionado en la sección 4.1.2, que utilizará el protocolo HTTP, maneja datos en formato JSON, el tipo de datos que usará son objetos "Usuarios" con sus respectivas propiedades.

Después, estableceremos los endpoints de los métodos que deseamos implementar, especificando las posibles respuestas para cada solicitud, como códigos de estado y mensajes asociados.

Tras esto, tendremos finalizada nuestra API de sistema encargada de la gestión de usuarios (figura 6.3), por lo que se procederá a publicarla en Exchange y a importarla en Anypoint Studio.

```
#%RAML 1.0
title: s-usuarios-api
description: Nueva API de sistema para interactuar con la base de datos de usuarios en Salesforce
mediaType:
  - application/json
version: "v1"
protocols:
  - HTTP

types:
  Usuarios:
    type: object
    properties:
      nombre: string
      contrasena: string
      numero_telefono: string
      correo_electronico: string
  UsuarioRespuesta :
    type: object
    properties:
      nombre: string
      numero_telefono: string
      correo_electronico: string

/usuarios:

  get:
    displayName: get usuarios
    description: get all usuarios
    responses:
      200:
        body:
          application/json:
            type: UsuarioRespuesta[]
      400:
        body:
          application/json:
            type: string
            example: |
              {
                "message": "Bad request"
              }
      404:
        body:
          application/json:
            type: string
            example: |
              {
                "message": "No user found with that name"
              }
  post:
    displayName: Crear nuevo usuario
    description: Crear  un nuevo usuario
```

Figura 6.3: Especificación RAML de la API s-usuarios

Una vez hayamos creado el nuevo proyecto Mule importando la API, en el archivo xml en el que se muestran los flujos, se crearán algunos por defecto (mencionados en la sección 6.1).

Como se ha mencionado en la sección 5.2, también se crearán por defecto los flujos correspondientes a los endpoints especificados en la API. Sin embargo, son flujos muy básicos creados simplemente para proporcionar una estructura inicial, sirven como punto de partida, por lo que se hace preciso agregar los componentes y conectores necesarios para que funcione acorde a los requisitos que hemos establecido.

6.2.1. Configuración previa de los sistemas

Antes de comenzar a explicar los flujos correspondientes a los endpoints es necesario crear y configurar nuevas funcionalidades en los sistemas que vamos a utilizar para esta API:

- Salesforce: este sistema dispone de un objeto, “Cuentas”, diseñado para almacenar información de clientes, el cual usaremos para registrar los datos de los usuarios que utilicen la herramienta Orienta-T.

  Este objeto está formado por varios campos que están añadidos por defecto y que sirven para capturar la información fundamental sobre las cuentas, entre los diferentes ejemplos que nos aportan, los que más nos interesan para nuestra herramienta son: nombre de la cuenta, su número de teléfono y un identificador asociado a ella.

Además de estos campos, tendremos la posibilidad de crear campos personalizados, pudiendo así almacenar información que no esté cubierta por los campos que por defecto nos ofrece el objeto.

Teniendo esto en cuenta, solo necesitaremos crear un nuevo campo para almacenar la contraseña, el cual se tratará de texto y cifrado, indicando también que el máximo de caracteres deberá de ser 20.

Para ofrecer mayor seguridad, se seleccionará la opción de aplicar máscara a todos los caracteres, consiguiendo de esta manera que la contraseña no sea visible y protegiendo la confidencialidad del usuario.

- Amazon SNS: crearemos un nuevo tema (topic) desde AWS Learner Lab (figura 6.4), a ese tema se le asignará un ARN (nombre que identifica de manera exclusiva los recursos de AWS). Los temas nos permitirán el envío de mensajes a múltiples suscriptores, para ello, habrá que suscribir los endpoints del destinatario al tema. También podremos mandar mensajes de manera asíncrona a todos lo suscriptores del tema, pudiendo así lograr, que todos los usuarios reciban un mensaje cuando se haya implementado una nueva funcionalidad o cuando, por ejemplo, se actualice Orienta-T.

Figura 6.4: Temas en SNS

6.2.2. Flujo “POST /usuarios”

Para detallar la implementación del flujo, se seguirá el orden de la figura 4.2, es decir, primero se verá la API de experiencia, luego la de procesos y finalmente la de sistema, sin embargo, este no ha sido el orden en el que se han implementado las APIs, sino al revés (empezando por la API de sistema).

Cuando la página envíe una petición “POST/usuarios”, se llamará al endpoint de la e-web-api cuyo flujo se muestra en la figura 6.5. Es un flujo básico que consta de un conector que, buscaremos en la sección de Exchange en Anypoint Studio, y se encargará de conectar directamente al flujo de la API de procesos que habremos desplegado en CloudHub.

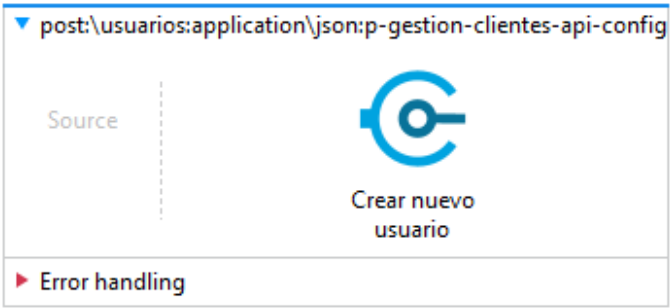

Figura 6.5: Flujo POST /usuarios de p-gestion-clientes-api

También tendremos la opción de usar un conector Request (figura 6.7), logrando lo mismo que el anterior conector, pero su método de funcionamiento es distinto. Es decir, de esta manera, se hará una solicitud HTTP a la API de sistema desplegada en la URL externa.

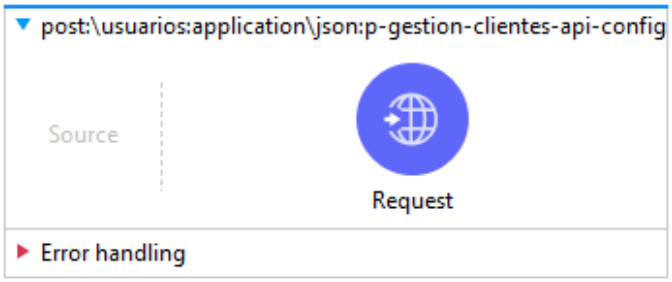

Figura 6.6: Alternativa flujo POST /usuarios de p-gestion-clientes-api

Para la api de procesos p-gestion-clientes-api, se implementará este flujo como lo hemos hecho anteriormente, a diferencia que, ahora usaremos la url de la API de sistemas.

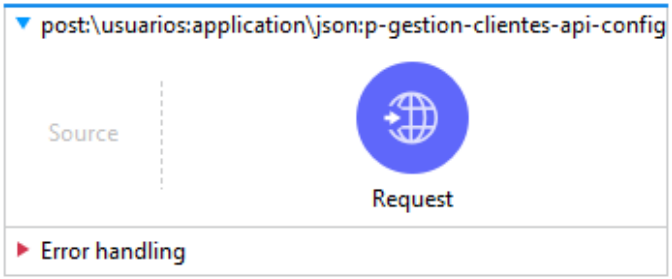

Figura 6.7: Alternativa flujo POST /usuarios de p-gestion-clientes-api

Tras estas implementaciones, llegará la solicitud a la API de sistemas, cuyo flujo (figura 6.8) será el más importante y veremos con detalle a continuación.

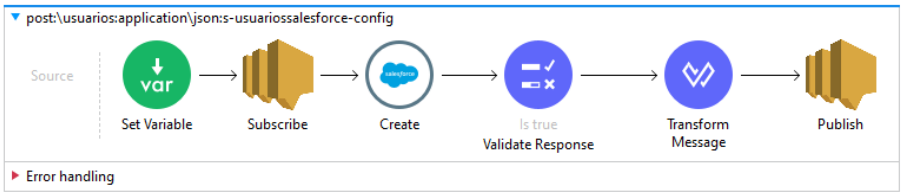

Figura 6.8: Flujo POST /usuarios de s-usuarios-api

El primer conector (Set Variable) sirve para guardar el valor del correo electrónico que se le pasará al programa en una nueva variable.

Antes de crear a los usuarios en Salesforce nos aseguraremos de que los datos que ha introducido son los suyos de registro gracias al componente Subscribe (específico de Amazon SNS). Dicho conector funcionará de la siguiente manera, suscribirá el correo que le pasemos en la variable guardada al ARN del tema creado. De esta manera, el usuario recibirá un correo en el que deberá de confirmar la suscripción al topic (figura 6.9) para que siga el proceso de creación de la cuenta. Si el correo no se confirma, será porque el usuario ha introducido un correo que no es el suyo, denegándole así tener una cuenta en Orienta-T.

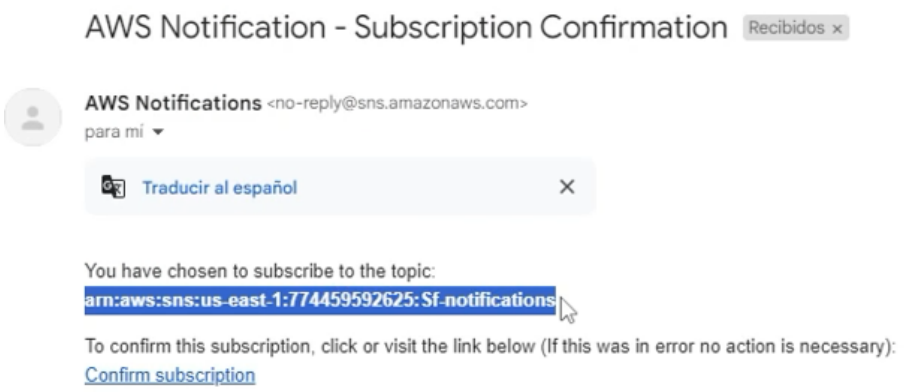

Figura 6.9: Correo de confirmación de suscripción

Desde AWS Learner Lab podremos ver un registro de los correos de verificación que se han enviado y su estado, es decir, si el usuario ha confirmado la suscripción o aún está pendiente (figura 6.10).

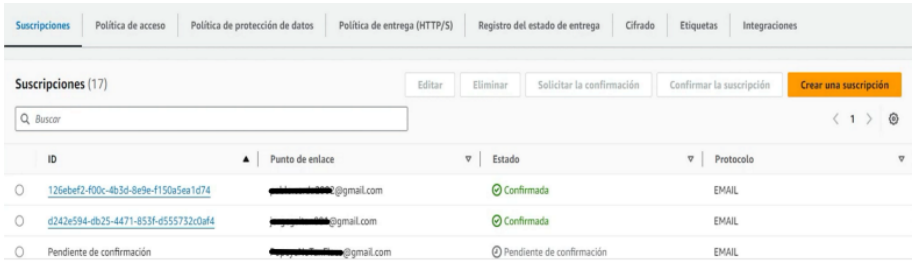

Figura 6.10: Registro de suscripciones

El componente Create es específico de Salesforce y permitirá crear una nuevo registro de usuario dentro del objeto “Cuentas”. Para ello, habrá que mapear los datos especificando los campos del objeto Salesforce y asignándoles los valores que se le han pasado al realizar la solicitud.

Para la configuración de dicho conector, se ha creado un archivo yaml (figura 6.11) en el cual se anotarán las credenciales de acceso a sistemas.

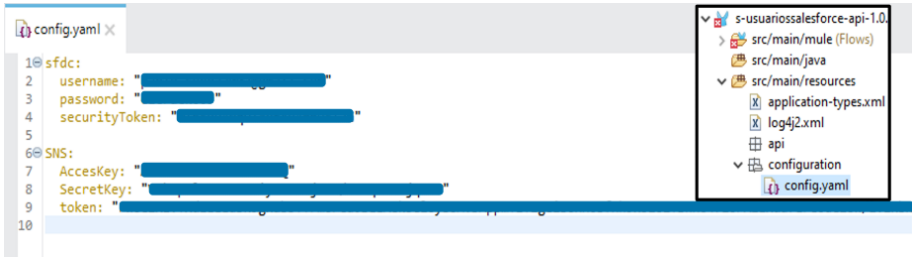

Figura 6.11: Contenido y ruta del archivo config.yaml

Este archivo, con nombre config.yaml se referencia en la creación del elemento de configuración de Salesforce (figura 6.12) y de Amazon SNS, para ello se rellena cada campo con su propiedad respectiva del archivo. De esta manera, en caso de que haya un cambio en la API y tuviéramos varios conectores que utilizasen esa conexión, solo tendríamos que cambiar el contenido del archivo y de las credenciales para que el cambio se aplicará a todos los conectores.

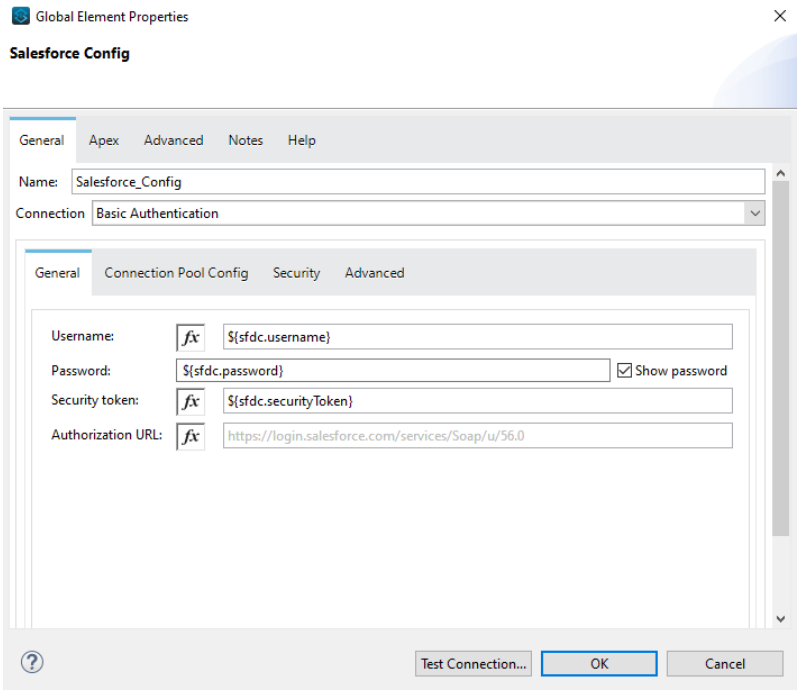

Figura 6.12: Configuración del elemento Salesforce Config a través de las propiedades de config.yaml

Tras configurar el conector específico de Salesforce y antes de realizar el proceso de notificar al usuario (indicándole que su cuenta se ha creado) deberemos de comprobar si la creación del usuario en Salesforce se ha realizado de manera correcta. Para ello comprobaremos mediante la función "payload.successful" si la respuesta obtenida tras intentar crear el usuario ha sido exitosa, es decir, si devuelve 'true'. En ese caso se continuará con el flujo, imprimiendo por consola un mensaje indicando que el usuario se ha creado en Salesforce. Sin embargo, esto no lo podrá ver el usuario, por lo que, con el conector Publish, le volveremos a mandar un correo a su dirección, mediante el tema al que está suscrito, en el que se le muestre que se ha dado de alta correctamente.

Por el contrario, si la respuesta obtenida no es exitosa, se indicará que no se ha podido crear la cuenta.

6.2.3. Flujo "GET /usuarios/nombre/contrasena"

Este flujo será el responsable del inicio de sesión de los usuarios, y, como se ha detallado en la figura 4.3 (de la sección 4.4), se reenviará la solicitud de GET hasta llegar a la API de sistema.

Como los flujos implementados en las APIs de sistema y de proceso son idénticas a las detalladas en el apartado anterior (sección 6.2.2), a excepción de que, esta vez, emplean un conector adicional de buscar un usuario por nombre y contraseña; no se comentarán en este apartado.

El que si se hace necesario explicar con más detalle es el flujo (figura 6.13) implementado en la API de sistemas.

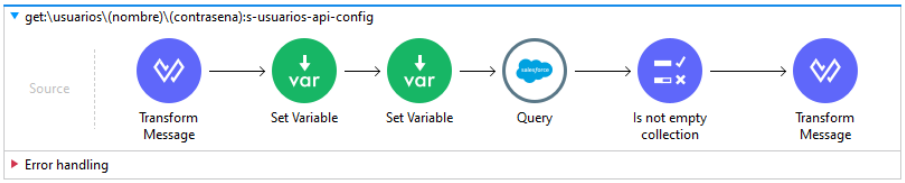

Figura 6.13: Flujo GET /usuarios/nombre/contraseña de s-usuarios-api

El primer conector (Transform message) estará configurado para extraer los parámetros nombre y contraseña de la URL de la solicitud y convertirlos en formato JSON.

Tras esto, almacenaremos ambos valores en dos variables distintas (mediante los conectores Set Variable) para hacer su uso posterior en el flujo.

Una vez tengamos los datos mapeados y almacenados en variables, haremos una búsqueda en el objeto "Cuentas" de Salesforce gracias al conector Query, en el cual podremos ingresar una consulta SQL que nos devuelva la cuenta con dicho nombre y contraseña.

Para comprobar si se nos ha devuelto un array del objeto "UsuarioRespuesta" vacío o no, haremos uso del conector "Is not empty collection", el cual, en caso de que el array no tenga ningún contenido, paralizará el flujo y devolverá un mensaje indicando que no existe el usuario que se busca, además de un status 404.

La otra opción es que se encuentren los datos del objeto, siguiendo así el flujo e imprimiendo un código de respuesta de estado satisfactorio (200 OK).

6.3 BÚSQUEDA Y CREACIÓN DE OBJETOS EN AMAZON RDS

Las implementaciones de la búsqueda y creación de cursos y carreras las encontramos en la api s-edu-api, ya que es la que interactúa directamente con la base de datos de Amazon RDS.

En esta sección analizaremos la implementación de los endpoints establecidos y nombrados en la sección 4.1.1.

6.3.1. Configuración previa de Amazon RDS

Antes de realizar la implementación de los flujos, deberemos de configurar el sistema para tener nuestra base de datos en la que almacenar los cursos, carreras y compras.

Para ello, desde AWS, crearemos una instancia de base de datos en Amazon RDS, utilizando MySQL como el motor ya que, además del buen rendimiento y escalabilidad que ofrece, es el más fácil de usar. También le asignaremos un identificador, nombre de usuario y contraseña del administrador a la base de datos.

Estos últimos elementos, acompañados del puerto y del endpoint, nos servirán para conectarnos a ella mediante MySQL Workbench.

Una vez nos hayamos conectado correctamente a la instancia de la base de datos, podremos crear las tablas de cursos, carreras (se le ha puesto el nombre de carreras, para que no haya malentendidos con los nombres de los endpoints) y compras, indicando en ellas sus respectivos atributos.

Otra funcionalidad que podremos realizar desde MySQL Workbench será añadir manualmente instancias de cursos o de cualquier objeto que veamos necesario, aunque para facilitar y agilizar esta tarea, se han incluido endpoints.

En la figura 6.14 se puede ver la simple estructura de la base de datos creada .

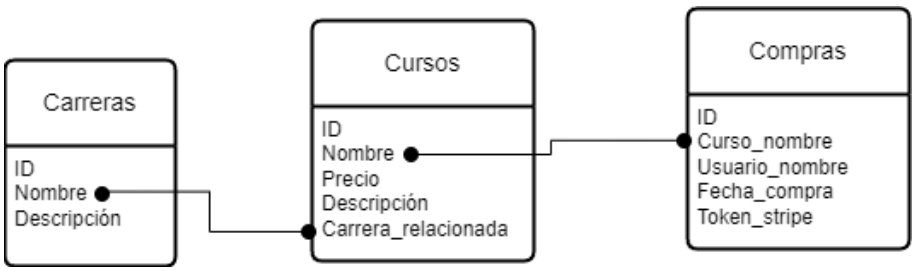

Figura 6.14: Base de datos RDS

6.3.2. Flujos de GETs y POSTs de cursos

Primero veremos la implementación del endpoint “GET/cursos”, mediante el cual podremos visualizar todos los cursos que se encuentren almacenados en la base de datos.

Esta implementación será muy simple, bastándonos sólo con redirigir la solicitud de la API de sistema a la de sistema y realizar una consulta a la base de datos para recuperar todos los cursos (en la tabla de dicho objeto), mapeando tras esto los datos obtenidos (figura 6.15).

Figura 6.15: Mapeo de los cursos obtenidos

En la figura 6.16 podremos ver como se ha implementado el flujo de este endpoint, simplemente con dos conectores, los cuales realizarán las funciones que se han descrito en el párrafo anterior.

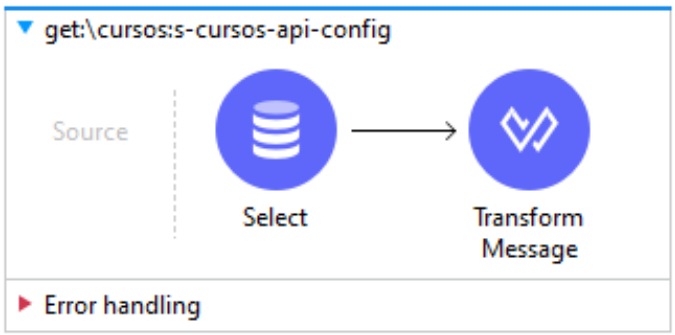

Figura 6.16: Flujo GET /cursos de s-edu-api

En segundo lugar, los endpoints "GET/cursos/carreraRelacionada" (figura 6.17) y "GET/cursos/nombre" los cuales son muy similares entre ellos y al anterior ("GET/cursos"), a diferencia que, tenemos un parámetro el cual utilizaremos para buscar los cursos en base a él.

Debido a su similitud, la implementación es igual, entre ellos, solo cambia el parámetro por el cual hacemos la consulta, para evitar ser reiterativo, solo se explicará el proceso de uno de estos dos endpoints ("GET/cursos/carreraRelacionada").

Lo primero que se hará para desarrollar dicho flujo es mapear el valor enviado como uriParam, para poder realizar la consulta SQL referenciándolo, ya que se deberá comprobar si en la base de datos existe algún curso que tenga en la columna de "CarreraRelacionada" la formación introducida.

Posteriormente, comprobaremos si el componente nos ha devuelto un objeto vacío o no, y en caso de que no lo esté, se mostrará la información de los cursos obtenidos.

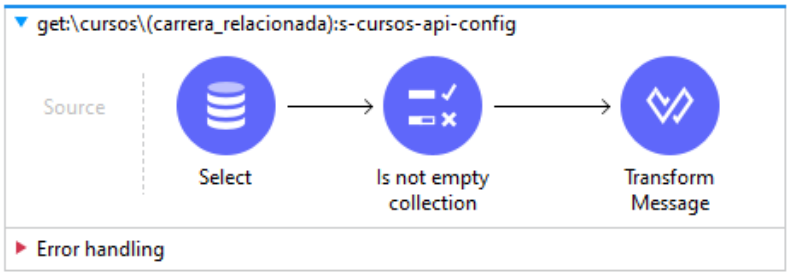

Figura 6.17: Flujo GET /cursos/carrera de s-edu-api

Para acabar con este apartado, se detallará la implementación del endpoint de "POST/cursos" (figura 6.18) en s-edu-api, para ello, haremos uso del componente Insert, el cual nos permite añadir los datos que se le pasan en la solicitud POST en la tabla que seleccionemos, mapeando antes los datos para insertarlos correctamente en sus respectivas columnas. En el siguiente conector, se añade el ID generado a los datos introducidos al principio. De esta manera se formará la respuesta que se devolverá mediante el nuevo evento, creado mediante el último componente para notificar la creación del curso.

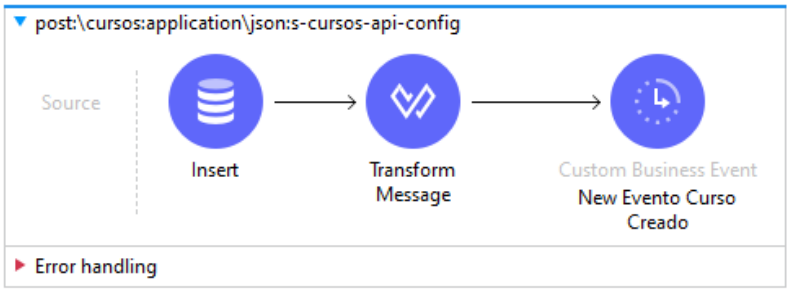

Figura 6.18: Flujo POST /cursos de s-edu-api

6.3.3. Flujos de GETs y POSTs de carreras o compras

La implementación de los endpoints que se detallarán en esta sección son las de "GET/carreras", "POST/carreras" y "POST/compra", sin embargo, éstas, no presentan ninguna novedad respecto a las explicadas en la sección anterior (6.3.2), ya que guardan mucha similitud a las de "GET/cursos" y "POST/cursos".

La única diferencia es que, en vez de realizar las consultas de seleccionar e insertar, sobre la tabla de cursos, las haremos sobre la tabla de carreras o de compras.

Por esta razón, simplemente se mencionarán y no se harán mucho más hincapié sobre ellas.

6.4 RECOMENDACIÓN DE CARRERAS

6.4.1. Configuración previa de AWS Lambda

Para implementar la recomendación de carreras, habrá que configurar primero Lambda. Para ello, accederemos a AWS hasta llegar a Lambda, una vez seleccionado dicho sistema, configuraremos una nueva función desde cero. Le pondremos un nombre a la función que vayamos a crear; en nuestro caso se le pondrá el nombre de 'recomendarCarrera' y se seleccionará Python 3.8 para ejecutarla.

Se creará un nuevo rol con permisos básicos de ejecución para permitir que la función escriba logs en CloudWatch (servicio de monitoreo de AWS).

Una vez creada la función, se accederá a la sección de edición de código, aquí es donde iría el algoritmo de recomendación.

Es importante mencionar que el desarrollo del algoritmo de recomendación en sí no es el objetivo de este trabajo. Sin embargo, Orienta-T ofrece una fácil conexión al algoritmo, proporcionando los objetos de intereses y carreras necesarios para su desarrollo.

6.4.2. Flujo POST/recomendar

La implementación del endpoint "POST/recomendar" en la API p-recomendacion-api (figura 6.19) constará de componentes para guardar en variables los intereses (introducidos al realizar el POST) y las carreras disponibles (obtenidas mediante el conector con nombre 'get carreras', el

cual llama al endpoint de “GET/carreras” de la API s-edu-api). Para finalizar esta implementación, se ejecutará el conector que redirige la solicitud, incluyendo las variables, a la API de sistema s-recomendacion-api.

Figura 6.19: Flujo POST /recomendar de p-recomendacion-api

Una vez la API de sistema reciba la solicitud, el conector Invoke ejecutará el algoritmo que se haya definido en la función creada en Lambda.

Para la configuración de dicho conector, necesitaremos las credenciales de AWS (como el resto de sistemas ofrecidos por AWS), el nombre de la función que hemos creado y las variables declaradas (carreras e intereses), las cuales formarán parte del cuerpo que se enviará a la función.

Finalmente se mostrarán los resultados de las carreras relacionadas gracias al mapeo de datos establecido en el último conector.

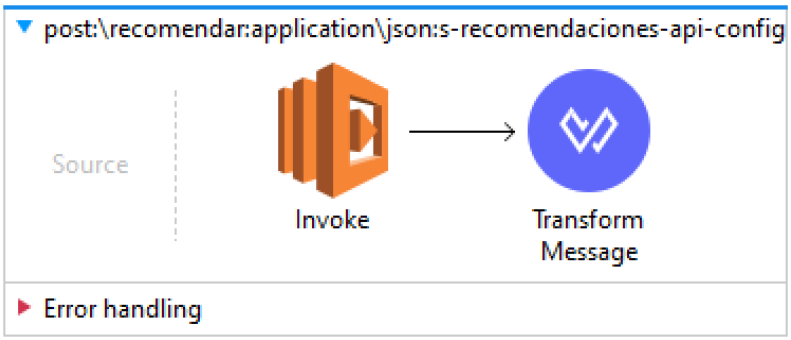

Figura 6.20: Flujo POST /recomendar de s-recomendacion-api

6.5 COMPRA DE CURSOS

6.5.1. Configuración previa de los sistemas

Antes de implementar la funcionalidad de la compra de cursos, necesitaremos una cuenta en Stripe, por lo que nos registraremos, obteniendo así las claves de la API que ofrece.

Una de las claves será pública (será la que nos permitirá crear los tokens de pago a partir de los datos del cliente) y la otra secreta para realizar las transacciones.

Una vez hayamos realizado estos pasos, tendremos disponible Stripe para hacer la solicitud de pago con los datos necesarios y se pueda realizar la transacción.

Cabe destacar que utilizaremos Stripe en modo desarrollador, por lo que los pagos no serán reales y solo se realizarán a modo de prueba pudiendo así verificar la funcionalidad sin procesar transacciones monetarias reales.

6.5.2. Flujo POST/compras

La implementación de la funcionalidad de la compra de cursos se iniciará en p-gestion-compras-api (figura 6.21), para ello, como paso previo estableceremos un conector para mapear los datos introducidos y poder realizar el POST (nombre del curso, nombre del usuario y token).

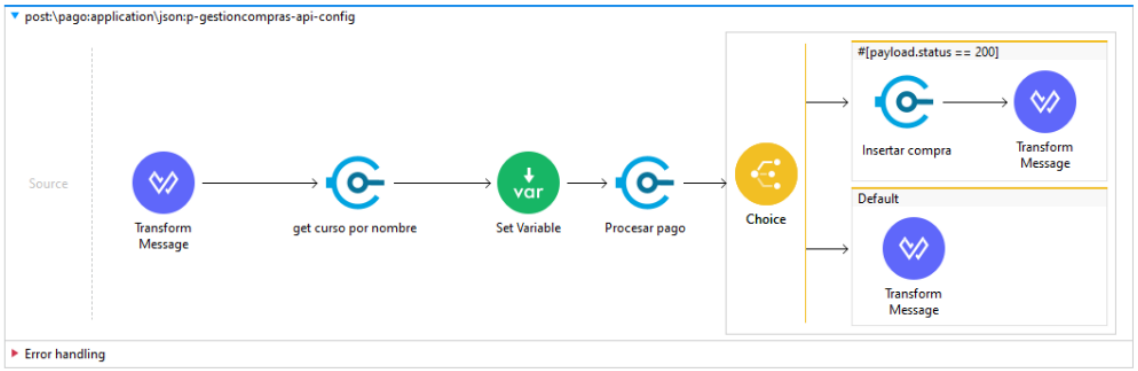

Figura 6.21: Flujo POST /compras de p-gestion-compras-api

Se realizará una consulta a la base de datos de RDS para buscar el curso mediante el nombre ingresado en la solicitud mediante el segundo conector (6.21, endpoint importado de la API s-edu-api), tras esto, guardaremos el precio del curso obtenido en una variable y se redirigirá la solicitud a s-compras-api.

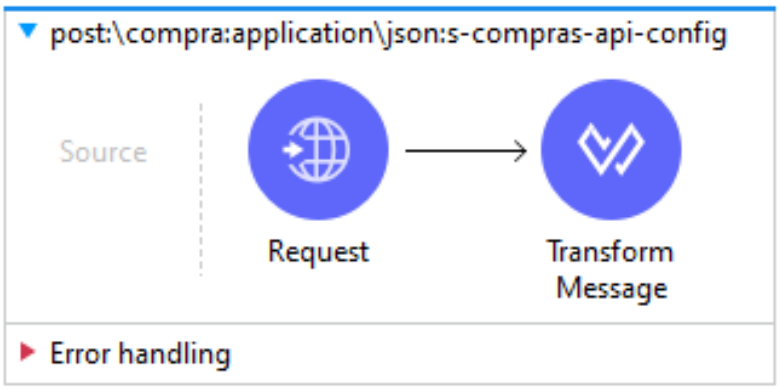

Figura 6.22: Flujo POST /compras de s-compras-api

Con el conector Request del flujo de la API de sistemas (figura 6.22) enviaremos una solicitud POST a la API de Stripe, añadiendo como cuerpo de dicha solicitud los datos que requerirá la pasarela de pago para realizar la transacción, es decir, el precio a pagar, la moneda y el token (contendrá los datos del método de pago que use el cliente).

Como cabecera, añadiremos la clave secreta obtenida de Stripe y el tipo de contenido igual a 'application/x-www-form-urlencoded' para especificar el tipo de contenido de la solicitud. Estas cabeceras son esenciales para autenticar la solicitud y asegurar que Stripe la procese correctamente.

El endpoint de la API de Stripe a donde enviaremos la solicitud será "/v1/charges", éste se utiliza para crear un cargo en una tarjeta de crédito, para procesar pagos y registrar la transacción en el sistema de Stripe.

Para acabar con el flujo de la API de sistemas, el último conector, devolverá a la API de procesos el estado de la compra.

Si a p-gestion-compras le llega un código de estado 200, la transacción se almacenará en la tabla 'compras' de la base de datos de Amazon RDS (también mediante un endpoint de s-edu-api)insertando en ella el usuario que ha realizado la compra, el curso que ha comprado y la fecha en la que se realizó la compra. También se le indicará al usuario que ya tiene el curso disponible.

Sin embargo, si ocurre algún error durante el proceso de pago (el código de estado es distinto a 200), el conector Choice (de la figura 6.21) seguirá el flujo alternativo, haciéndole saber al usuario que la operación de pago no se ha realizado de manera exitosa.

CAPÍTULO 7
Implantación y pruebas

7.1 Implantación

En este capítulo se detalla la fase de implantación de la solución desarrollada. Esta fase es crucial ya que al poner en marcha la solución, se podrá probar y evaluar los resultados.

7.1.1. Pasos para la implantación

Inicialmente, la solución ha sido desarrollada y desplegada en CloudHub. A continuación, se explican los pasos realizados para implantar Orienta-T:

- Primero se realizará el proceso de iniciar AWS Learner Lab para poder hacer uso de los sistemas alojados en este entorno, como Amazon RDS, Lambda y SNS.

- Tras finalizar este proceso se comprobará que las variables de entorno, incluyendo las credenciales de acceso a AWS, Salesforce y Stripe, estén bien introducidas en los conectores que hagan referencia a ellas en Anypoint Studio.

- Si todas las comprobaciones son correctas, se podrán desplegar las APIs en CloudHub desde Anypoint Studio.

Para probar las APIs, tendremos dos opciones:

- Consola de Anypoint Studio: una vez se haga la operación “run” de la API y se despliegue correctamente, nos aparecerá una consola en la que podremos probar de manera local los endpoints implementados.

- Postman: cuando hayamos desplegado una API en CloudHub, podremos visualizar, desde la sección de Runtime Manager en Anypoint Platform, la url de su endpoint público. Esta url se podrá introducir en Postman (como ya se ha mencionado en la sección 5.4) y se le podrán hacer solicitudes para ver si la respuesta es la esperada.

7.1.2. Integración en una página web

Como ya se ha comentado, Orienta-T es un middleware que ofrece conectores y APIs para la integración de diversos sistemas. Para completar su funcionalidad e implantarla adecuadamente en un entorno práctico, una de las opciones que han sido mencionadas sería integrarla a una página web, ya que al ser REST, efectúa las solicitudes por HTTP y la página web es la opción que mejor se adecua a esto.

Para realizar la integración de Orienta-T en una página web, se podrían seguir estos pasos:

- Desarrollo del frontend: crear una interfaz de usuario utilizando tecnologías web como HTML, CSS y JavaScript. Esta interfaz permitirá a los usuarios interactuar con la plataforma Orienta-T. [22] [24]
- Conexión con las APIs: utilizar AJAX para hacer solicitudes HTTP a los endpoints de Orienta-T desde la página web para recuperar y enviar datos entre el frontend y las APIs.
- Visualización de datos: finalmente, quedaría mostrar los datos obtenidos de las APIs en la interfaz de usuario de manera dinámica. [23]

7.2 Pruebas

En este capítulo se mostrarán las pruebas realizadas para comprobar que los endpoints implementados en las APIs funcionan de manera correcta y cumplen con las expectativas que se han ido desarrollando a lo largo de esta memoria.

Como se ha comentado en capítulo 7, existen dos procedimientos de prueba funcional (permiten verificar que los componentes del middleware cumplen los requisitos especificados) para las APIs pruebas funcionales . Sin embargo, debido a la expiración del período de prueba de CloudHub que se me proporcionó durante el desarrollo de la Cátedra y Hackaton, las pruebas se han realizado en un entorno local utilizando la consola ofrecida por Anypoint Studio.

Esto no afectará a la validez de los resultados, ya que las pruebas locales también permiten verificar el correcto funcionamiento de las APIs y su integración con los demás componentes del sistema. Sin embargo, no podremos realizar pruebas sobre conectividad, tiempos de respuesta, etc. en un entorno de ejecución real (en CloudHub) .

También se utilizará Postman para enviar solicitudes a los enlaces locales generados por Anypoint Studio tras ejecutar la API.

Las pruebas funcionales que se realizarán sse detallarán en los siguientes apartados.

7.2.1. Creación de un usuario y la búsqueda en base a su nombre y contraseña

Para realizar esta prueba enviaremos una solicitud “POST/usuarios” a la API de experiencia (la url será ’http://localhost:8081/api/usuarios’) añadiendo en formato JSON los datos que queremos que tenga nuestro nuevo usuario (figura 7.1).

Figura 7.1: Datos nuevo usuario

Tras confirmar el mensaje que nos llega al correo introducido (figura 6.9), llegará otro mensaje indicando que nos hemos dado de alta correctamente (figura 7.2).

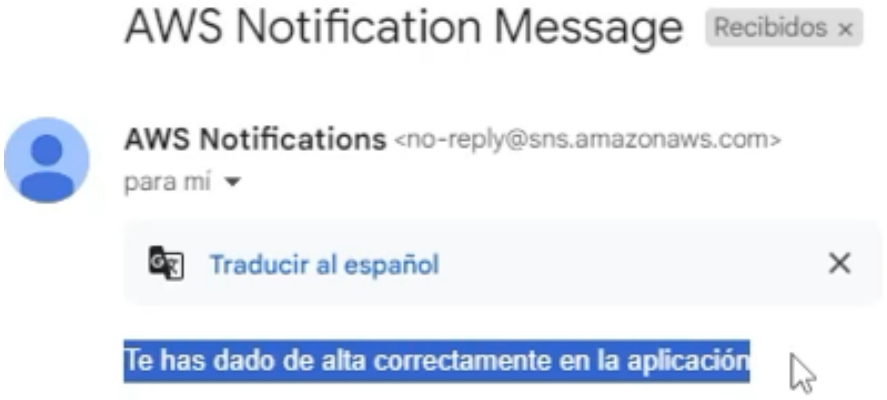

Figura 7.2: Notificación por correo de alta en Salesforce

Para comprobar que, efectivamente, el usuario recién creado esté almacenado en Salesforce haremos una solicitud con el nombre y contraseña como parámetros URI al siguiente enlace: 'http://localhost:8081/api/usuarios/JuanLuis/JuanLuperez08' . El cual nos devuelve el resultado que se muestra en la figura 7.3: un código de estado 200 y los datos del usuario solicitado.

Figura 7.3: Respuesta del endpoint GET /usuarios/nombre/contrasena

7.2.2. Creación de un curso

En esta prueba, se hará, de nuevo, una solicitud a e-web-api, pero, esta vez al endpoint "POST/cursos" y será una solicitud POST, en el cuerpo añadiremos los datos del nuevo curso (figura 7.4).

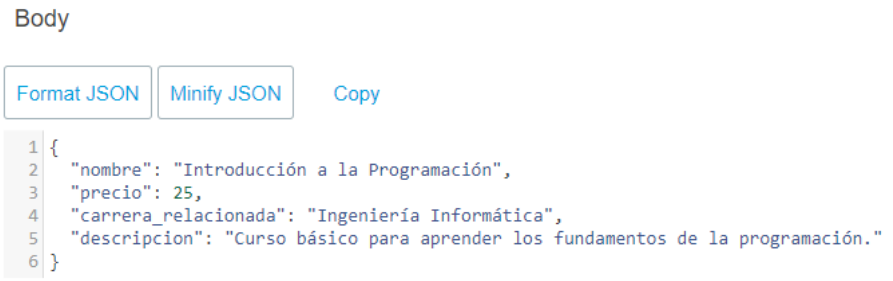

Figura 7.4: Datos nuevo cursos

Tras enviar la solicitud, se devuelve un código de estado 200 y si accedemos a nuestra base de datos desde MySQL Workbench, podremos ver como se ha introducido el nuevo objeto en la tabla de cursos (figura 7.5).

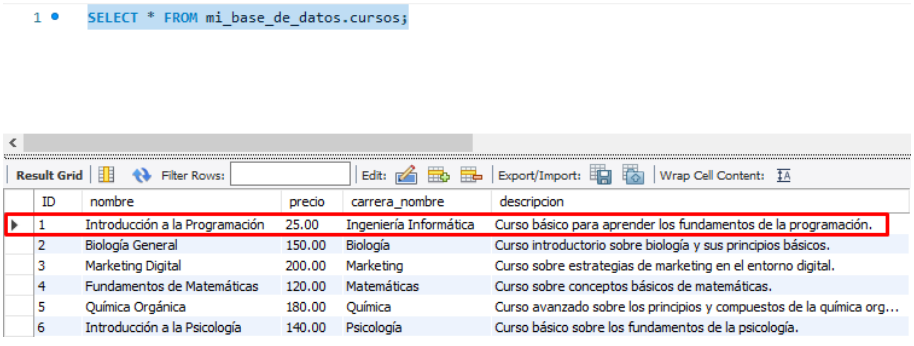

Figura 7.5: Nuevo curso añadido en la base de datos

7.2.3. Realización de recomendaciones

Para probar esta funcionalidad se modificará el flujo debido a que el algoritmo de recomendación no está desarrollado. Quitaremos el conector de Lambda y veremos si se devuelven correctamente los intereses introducidos y las carreras de la base de datos, éstos serán los elementos que necesitará Lambda para ejecutar el algoritmo. En la figura 7.6, se puede ver un ejemplo de intereses que se introducirán como cuerpo de la solicitud.

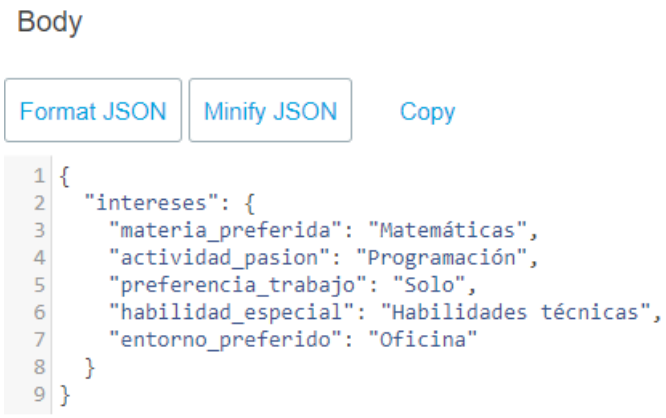

Figura 7.6: Ejemplo de objeto interés

Tras enviar la solicitud, obtendremos, tanto los intereses como las carreras (figura 7.7), confirmando así que los datos que se necesitan para el algoritmo, lleguen al conector de Lambda correctamente.

Figura 7.7: Resultados obtenidos POST/recomendacion

7.2.4. Realización de compra

Para comprobar si la función de comprar cursos funciona correctamente, mandaremos al endpoint “POST/compra” una solicitud que contenga como cuerpo los siguientes atributos: nombre de usuario (JuanLuis), curso (Marketing Digital) y el token de Stripe (tokVisa, es un identificador de prueba que se utilizado para simular transacciones con una tarjeta de crédito).

Si la transacción se ha realizado correctamente, se mostrará por pantalla el mensaje de la figura 7.8 .

Figura 7.8: Resultado POST/compra

Si buscamos en el registro de compras en la tabla ’carreras’ de la base de datos de RDS, veremos la incorporación de la nueva compra (figura 7.9).

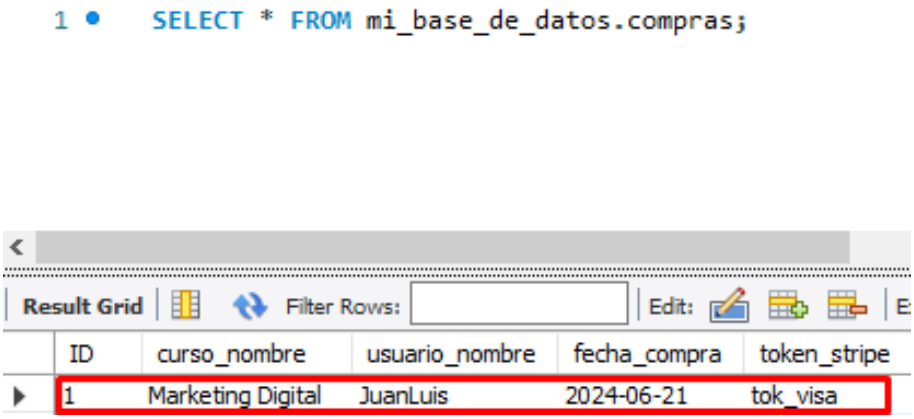

Figura 7.9: Tabla compras

Finalmente, si accedemos a nuestra cuenta de Stripe en la cual se hacen los cargos, veremos los datos del pago (figura 7.10): el precio del curso (obtenido consultando la tabla de cursos), el usuario que ha realizado la compra y el método de pago.

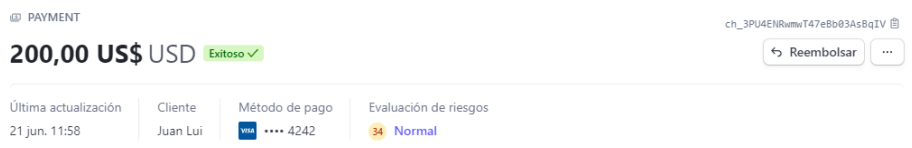

Figura 7.10: Pago en Stripe

CAPÍTULO 8
Conclusiones

Durante la elaboración de este proyecto, me he encontrado con varios desafíos que han requerido soluciones creativas, así como la integración de diversas tecnologías. Uno de los problemas más significativos ha sido gestión y sincronización de datos entre los diferentes servicios en la nube. Sin embargo, este problema se pudo solucionar mediante el uso de la arquitectura API-Led, permitiendo una gestión eficiente de los datos y una fácil integración de nuevas funcionalidades.

El principal objetivo del proyecto era desarrollar una herramienta de orientación vocacional que facilitara la toma de decisiones informadas por parte de los estudiantes y proporcionase conectores para integrarse a una futura página web, creando así una plataforma. Este objetivo no se ha alcanzado por completo, ya que durante el desarrollo, hemos descubierto que la creación de un algoritmo de recomendación de carreras basado en machine learning no es viable dentro del alcance de este proyecto. A pesar de no haber alcanzado este objetivo específico, hemos logrado desarrollar una herramienta funcional que ofrece conectores que servirán como base para una futura implementación mejorada de las recomendaciones personalizadas de carreras, y la gestión de cursos de especialización, cumpliendo así parcialmente con los objetivos planteados inicialmente.

El desarrollo de este proyecto ha sido una muy buena oportunidad para profundizar en mis conocimientos sobre la integración de aplicaciones. Tras haber tenido una primera toma de contacto en la asignatura correspondiente y al participar en la Hackathon y Cátedra organizada por DISID, MuleSoft y la UPV, el TFG me ha permitido ampliar y consolidar mis conocimientos. He resuelto dudas que tenía sobre conceptos clave y he alcanzado un dominio significativo en tecnologías avanzadas como Anypoint Platform de MuleSoft, sistemas de AWS, así como en la integración de servicios como Salesforce. Esto no solo ha permitido resolver el problema planteado, sino también adquirir habilidades muy valiosas para mi futuro profesional.

8.1 Trabajos futuros

El desarrollo de Orienta-T ha logrado alcanzar los objetivos propuestos, sin embargo, existen diversas oportunidades para mejorar y expandir este proyecto. Estas propuestas, además de mejorar la herramienta, se proponen explorar nuevas áreas que podrán beneficiar al middleware.

- Creación de una interfaz web: esta era uno de los objetivos que se plantearon durante el desarrollo del proyecto, pero no se ha podido alcanzar. Al disponer de una interfaz web, ofreceríamos una plataforma completa (en lugar de solo un middleware) donde los usuarios puedan interactuar de manera más visual y dinámica con las funcionalidades de Orienta-T.

- Funcionalidades adicionales: al disponer de una interfaz web, podríamos incluir en ella funcionalidades como foros de discusión o chats para que los usuarios alumnos puedan pedir ayuda a los docentes de los cursos, realizar tutorías en línea y se podrían diseñar herramientas de seguimiento del progreso en los cursos.

- Algoritmo de recomendación avanzado: es una propuesta que se ha comentado durante el desarrollo de esta memoria, implementar un algoritmo basado en machine learning para recomendar de manera precisa y personalizada las formaciones. Esto haría ganar más reconocimiento a nuestra plataforma, ya que a día de hoy no existe en el mercado tecnológico una herramienta que incluya dicha funcionalidad.

- Integración con otros sistemas: también se podrían llegar a integrar otros sistemas como por ejemplo Linkedin, para añadir en el perfil de los estudiantes los cursos finalizados, o Trello para ayudar a los usuarios a organizar sus tareas y proyectos.

- Implementación de colas: finalmente, otra idea sería añadir un sistema de colas utilizando RabbitMQ (sistema en la nube ofrecido por AWS) que gestionase el acceso a la pasarela de pago. Esto mejoraría la escalabilidad y robustez de la herramienta, permitiendo manejar un mayor volumen de transacciones simultáneas y asegurando que las solicitudes de pago se procesen de manera ordenada.

8.2 Relación del trabajo desarrollado con los estudios cursados

Durante la realización de mis estudios del grado de Ingeniería Informática, he cursado asignaturas que me han aportado distintos conocimientos y habilidades esenciales para el desarrollo de este Trabajo de Fin de Grado.

A continuación se detallarán las asignaturas que más me han servido de ayuda y cuales de sus competencias transversales he empleado más para el desarrollo de este proyecto.

- “Integración de aplicaciones”: los conocimientos aprendidos en esta materia han supuesto la base de este trabajo. En ella he aprendido conceptos fundamentales sobre la integración de aplicaciones que se han empleado en gran medida durante este proyecto. Un ejemplo de ellos son los tipos de middleware, los frameworks que facilitan la comunicación entre sistemas, las técnicas y lenguajes que permiten el mapeo de datos y el patrón arquitectónico SOA entre otros.

- “Bases de datos y sistemas de información”: en esta asignatura tuve un primer acercamiento a las bases de datos. Los conocimientos aprendidos los he aplicado en el diseño de bases de datos y el poder interactuar con ellas realizando consultas SQL. [26]

- "Tecnología de bases de datos": al ser una continuación de la anterior asignatura, me ha ayudado a desarrollar con claridad los pasos seguir para gestionar y administrar sistemas de bases de datos relacionales.

- "Desarrollo web": en esta asignatura adquirí conocimientos para saber cómo poder integrar Orienta-T con una página web, el funcionamiento de ellas y como se pueden enviar solicitudes HTTP a través de eventos como, por ejemplo, el envío de formularios. En la asignatura "Desarrollo centrado en el usuario", también aprendimos cómo se envían y funcionan de los formularios. [25] Todos estos conocimientos adquiridos en ambas asignaturas me han servido de ayuda para entender cómo se llamarían a los endpoints implementados en las APIs en caso de que las hubiéramos integrado con una interfaz web.

Bibliografía

[1] SAP, *What is Enterprise Integration/Application Integration?*, SAP, 2023, https://www.sap.com/products/technology-platform/what-is-enterprise-integration/application-integration.html.

[2] Ian Blair, *What is Application Integration?*, BuildFire blog, 2023, https://buildfire.com/what-is-application-integration/.

[3] IBM, *Application Integration*, 2023, https://www.ibm.com/topics/application-integration, 2024.

[4] AWS, *What is an API?*, AWS informational content, 2003, https://aws.amazon.com/es/what-is/api/.

[5] Red Hat, *What Are Application Programming Interfaces?*, Red Hat, 2023, https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces.

[6] Gregor Hohpe, Bobby Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley, 1st edition, ISBN 978-0321200686, 2003.

[7] Martin Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 1st edition, ISBN 978-0321127426, 2003.

[8] Herbjorn Wilhelmsen, Thomas Erl, *SOA with REST: Principles, Patterns and Constraints for Building Enterprise Solutions with REST*, Prentice Hall, 1st edition, ISBN 978-0137012510, 2013.

[9] Leonard Richardson, Sam Ruby, *RESTful Web Services*, O'Reilly, ISBN 978-0596529260, 2007.

[10] David Chappell, Tyler Jewell, *Java Web Services*, O'Reilly, ISBN 978-0596002695, 2002.

[11] Gustavo Alonso, *Web Services: Concepts, Architectures and Applications*, Springer, ISBN 978-3540440080, 2004.

[12] Astera Blog, *Understanding Data Mapping and Its Techniques*, 2023, https://www.astera.com/es/type/blog/understanding-data-mapping-and-its-techniques/.

[13] AWS, *What is Middleware?*, AWS informational content, 2003, https://aws.amazon.com/es/what-is/middleware/.

[14] Merge Blog, *Point-to-Point Integration*, https://www.merge.dev/blog/point-to-point-integration, 2024.

[15] Adeptia Blog, *Eliminate Point-to-Point Integration Pain with Hub-and-Spoke Model for Enterprises*, https://www.adeptia.com/blog/eliminate-point-point-integration-pain-hub-spoke-model-enterprises.

[16] Qlik, *Data Integration: iPaaS*, https://www.qlik.com/us/data-integration/ipaas, 2024.

[17] Brenda Jin, Saurabh Sahni, Amir Shevat, *Designing Web APIs: Building APIs That Developers Love*, O'Reilly Media, 2020.

[18] Moesif Blog, *Mastering the API Lifecycle: Essential Stages and Proven Strategies for Success*, https://www.moesif.com/blog/api-analytics/What-is-API-Lifecycle-Management/.

[19] Postman Blog, *The API Lifecycle*, https://blog.postman.com/the-api-lifecycle/.

[20] MuleSoft, *Anypoint Studio | Integrated Development Environment (IDE)*, https://www.mulesoft.com/platform/studio, 2024.

[21] NTT Data, *Qué es MuleSoft Anypoint y Principales Características*, https://ifgeekthen.nttdata.com/es/que-es-mulesoft-anypoint-y-principales-caracteristicas, 2024.

[22] Mario Rubiales Gómez, *Curso de desarrollo web: HTML, CSS, y JavaScript*, Anaya Multimedia, Primera edición, Segunda edición, junio de 2021, ISBN 978-8441534209.

[23] Rebecca Murphey, *jQuery Fundamentals: A Guide to the Basics of jQuery*, ISBN 978-0989511400.

[24] Elisabeth Robson, Eric Freeman, *Head First HTML and CSS*, O'Reilly, Second edition (updated for HTML5), ISBN 978-0596159900, 2012.

[25] Jesse James Garrett, *The Elements of User Experience: User-Centered Design for the Web and Beyond [electronic resource]*, New Riders, 2nd edition, ISBN 978-0321683687, 2011.

[26] Ramez Elmasri, Shamkant B. Navathe, *Fundamentos de sistemas de bases de datos*, Pearson Educación, D.L. 2007, ISBN 978-8483224230.

APÉNDICE A

Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**			X	
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**		X		
ODS 4. **Educación de calidad.**	X			
ODS 5. **Igualdad de género.**	X			
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**	X			
ODS 11. **Ciudades y comunidades sostenibles.**			X	
ODS 12. **Producción y consumo responsables.**			X	
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**			X	
ODS 17. **Alianzas para lograr objetivos.**	X			

Este proyecto ha sido una iniciativa que, aunque en un principio estaba centrada en el ámbito educativo y tecnológico, tiene una relación significativa con varios objetivos ODS.

Los ODS más relevantes en nuestro proyecto son:

- Educación de calidad (ODS 4): este objetivo es uno de los que tiene más relación con nuestro proyecto, ya que Orienta-T pretende garantizar una buena educación promoviendo oportunidades de aprendizaje mediante la oferta de cursos de especialización que mejoren las habilidades y conocimientos de los estudiantes.

- Trabajo decente y crecimiento económico (ODS 8): la recomendación de formaciones alineadas con los intereses y habilidades de los estudiantes puede aumentar su satisfacción laboral y mejorar su desempeño profesional. Contribuyendo al crecimiento económico mejorando la preparación de los estudiantes con mayor motivación y aumentando su productividad. Evitando elecciones erróneas de formaciones y posterior vida laboral, lo que redundará una mejora tanto de la salud y como del bienestar (ODS 3) de las personas y, que a su vez, conllevará comportamientos más responsables (ODS 12, Producción y consumo responsables).

- Industria, innovación e infraestructura (ODS 9): el proyecto utiliza tecnologías avanzadas y enfoques innovadores, como la arquitectura API-Led y servicios en la nube de AWS. Cabe destacar que la integración de diversas plataformas y la implementación de APIs demuestran un compromiso con la innovación y la creación de infraestructuras modernas.

- Alianzas para lograr los objetivos (ODS 17): en este punto volveremos a mencionar la integración, concepto también nombrado en el punto anterior. Y es que, Orienta-T se beneficia de la integración con diversas plataformas y servicios, demostrando así el valor de las alianzas para alcanzar los objetivos. La colaboración con servicios como Amazon RDS, Salesforce y Stripe no solo mejora la funcionalidad de la plataforma, sino que también ejemplifica cómo las alianzas pueden ayudar a obtener mejores resultados.

- Reducción de las desigualdades (ODS 10): la herramienta Orienta-T propone recomendaciones de formaciones basándose únicamente en las motivaciones, intereses y aptitudes de las personas que la usen, sin que interfieran sesgos por razón de sexo, clase social, estatus económico localización geográfica...

- Igualdad de género (ODS 5): considero que Orienta-T puede ayudar a minimizar la brecha de género en la elección de estudios y profesión. Los estereotipos de género condicionan la elección, favoreciendo que las mujeres suelan escoger formaciones relacionadas con la educación la salud y el bienestar, y los hombres se decanten por formaciones científico-tecnológicas e industriales (profesiones STEM). Nuestra herramienta Orienta-T, al basar las recomendaciones en intereses, hará que sus usuarios, tanto hombres como mujeres, lleguen a plantearse realizar formaciones que inicialmente no se planteaban. Podrán hacer elecciones que ayuden a revertir esta brecha de género y que podrán elegir su proyecto de vida laboral libre de estos sesgos.

También deberemos de tener en cuenta los objetivos que no tienen mucha relevancia en nuestro proyecto como por ejemplo el número 1 (Fin de la pobreza), ya que tiene un impacto indirecto leve

al mejorar la orientación vocacional y educativa, permitiendo a los estudiantes a acceder a mejores oportunidades laborales en el futuro.

En conclusión, aunque algunos ODS tienen una relación indirecta con el proyecto, la plataforma Orienta-T se alinea fuertemente con varios objetivos clave, demostrando que la integración de tecnologías innovadoras y la promoción de la igualdad pueden tener un impacto significativo en la educación y en el desarrollo sostenible.