UTPL TEC

Interoperabilidad Empresarial

Guía didáctica

UTPL

Unidad Académica Técnica y Tecnológica

Tecnología Superior en Transformación Digital de Empresas

Interoperabilidad Empresarial

Guía didáctica

Carrera	*PAO Nivel*
▪ *Tecnología Superior en Transformación Digital de Empresas*	III

Autor:

Quiñonez Cuenca Felipe David

DSOF_2046

Universidad Técnica Particular de Loja

Interoperabilidad Empresarial
Guía didáctica
Quiñonez Cuenca Felipe David

Diagramación y diseño digital:

Ediloja Cía. Ltda.
Telefax: 593-7-2611418.
San Cayetano Alto s/n.
www.ediloja.com.ec
edilojacialtda@ediloja.com.ec
Loja-Ecuador

ISBN digital - 978-9942-39-817-8

25 de mayo, 2023

Índice

1. Datos de información

1.1. Presentación de la asignatura

1.2. Competencias genéricas de la UTPL

- Pensamiento crítico y reflexivo.

1.3. Competencias específicas de la carrera

- Gestionar y desarrollar aplicaciones empresariales aplicando enfoques centrados en la nube.

1.4. Problemática que aborda la asignatura

Gestionar procesos de transformación organizacional mediados por tecnologías digitales. Por esta razón, se ha considerado a la Formación Empresarial como un pilar fundamental en esta carrera y se han definido un conjunto de asignaturas eminentemente prácticas enfocadas a que

el estudiante pueda gestionar el cambio organizacional y se transforme en un líder responsable de formular estrategias y métodos capaces de transformar la mentalidad y la cultura para que los proyectos de transformación digital tengan éxito.

2. Metodología de aprendizaje

Para el aprendizaje de la asignatura se trabajará con la Metodología Basada en Problemas (ABPR), la cual se enfocará en la identificación de problemas empresariales desafiantes, la investigación y análisis, la identificación de soluciones, el desarrollo de soluciones utilizando enfoques centrados en la nube, así como la evaluación y mejora de la solución desarrollada. Todo esto con el objetivo de fomentar el pensamiento crítico y reflexivo de los estudiantes, así como el desarrollo de competencias específicas relacionadas con la gestión y desarrollo de aplicaciones empresariales.

3. Orientaciones didácticas por resultados de aprendizaje

Primer bimestre

Resultado de aprendizaje 1

- Comprende principios, técnicas y mecanismos para la integración de ecosistemas digitales de negocio empresariales.

Al igual que en un ecosistema natural, estamos todos conectados, también existen ecosistemas digitales. Estos hacen referencia a los ajustes internos de una compañía, así como a sus alianzas con otras organizaciones para ofrecer un mejor servicio y calidad en los productos. Para lograr este resultado de aprendizaje realizaremos un análisis conceptual para entender los principios básicos de la interoperabilidad.

Contenidos, recursos y actividades de aprendizaje

Semana 1

Estimado estudiante, iniciamos el estudio de la asignatura de Interoperabilidad Empresarial, donde los contenidos, recursos y actividades están orientados a lograr que usted conozca los conceptos de la interoperabilidad dentro de un ecosistema digital y como estos pueden ser abordados con diferentes técnicas.

A lo largo del curso iremos revisando algunos conceptos, herramientas y técnicas que le mostrarán como se puede aplicar los principios en escenarios reales.

Es fundamental tener en cuenta que la información que se transmite en esta asignatura se actualiza constantemente, ya que la son un área en constante progreso.

En esta asignatura no contaremos con un **texto básico**, en lugar de ello tendrá acceso a diferentes recursos que le ayudarán en su aprendizaje.

Unidad 1. Fundamentos de la interoperabilidad empresarial

Esta primera unidad está orientada a la revisión de algunos conceptos, herramientas y técnicas que le mostrarán cómo se pueden aplicar los principios de la interoperabilidad empresarial en escenarios reales. Esto le permitirá tener una comprensión de los fundamentos, preparándolo para abordar de mejor forma los diversos desafíos del mundo profesional.

1.1. Definición de la interoperabilidad

La interoperabilidad es una característica clave en un mundo digitalizado, y su importancia aumenta a medida que más personas y empresas dependen de los dispositivos conectados para obtener información, comunicarse y realizar transacciones.

Según (Kotzé & Neaga, n.d.), todavía no existe una definición única para el término interoperabilidad. El término interoperabilidad a menudo se usa en un sentido de ingeniería de sistemas técnicos, o alternativamente en un sentido amplio, teniendo en cuenta los factores sociales, políticos y organizacionales que afectan el rendimiento del sistema a sistema. A continuación, se presenta algunas definiciones para interoperabilidad:

- El Open Group(The TOGAF Standard, Version 9.2 - Interoperability Requirements, n.d.), define la interoperabilidad, en el contexto de la versión 9 de TOGAF, como la capacidad de “compartir información y servicios” , de “ dos o más sistemas para intercambiar y usar información” y de “ sistemas para proporcionar y recibir servicios de otros sistemas” para permitirles que funcionen juntos de manera eficaz.

- La (“IEEE Standard Glossary of *Software* Engineering Terminology,” 1990), por ejemplo, define la interoperabilidad como “ la habilidad

para que dos o más sistemas o componentes puedan compartir información y, a su vez, aprovechar la información compartida” .

Los conceptos anteriores, como se menciona en ¿Qué Es La Interoperabilidad y Cómo Puede Lograrla Mi Empresa?, 2019, nos hacen recordar la antigua visión del ser humano de poder compartir información de forma mundial, sin importar la tecnología que se usa para almacenar, procesar o distribuir la información. Con el objetivo de conseguir esto, se inventó la imprenta en 1440 y en 1969 se desarrolló ARPANET, que luego se convirtió en la *Internet* que conocemos.

Ahora que ya tenemos claro el concepto de Interoperabilidad, pasamos a revisar qué es la interoperabilidad empresarial que, acorde Kotzé & Neaga (Kotzé & Neaga, 2010), las empresas modernas deben ser interoperables para enfrentar los desafíos actuales del negocio, tanto en términos de sus sistemas de TI, como de sus procesos de negocio, sus aplicaciones e incluso sus recursos humanos, ya sea desde un punto de vista intra o interorganizacional.

La interoperabilidad empresarial se refiere a la interoperabilidad entre unidades organizacionales o procesos de negocio, ya sea dentro de una gran empresa (distribuida) o dentro de una red empresarial. El desafío radica en facilitar la comunicación, la cooperación y la coordinación entre estas unidades y procesos. Desde una perspectiva de arquitectura empresarial y de ingeniería de sistemas, operar en un entorno en red coloca los requisitos de interoperabilidad junto a los requisitos de mantenibilidad, fiabilidad, seguridad y soportabilidad de un sistema.

En conclusión, la interoperabilidad empresarial fomenta la colaboración entre las empresas, posibilitando el intercambio de información y recursos, lo que se traduce en un incremento de la productividad.

1.2. Niveles de la interoperabilidad

De acuerdo con Kotzé & Neaga (2010) y Solano (2020), se identifican tres niveles de interoperabilidad, los mismos que se explican a detalle en la siguiente infografía:

Niveles de la interoperabilidad.

Esta infografía muestra estos principales niveles de interoperabilidad. Las flechas muestran que para lograr la interoperabilidad entre el sistema A y el sistema B que están utilizando diferentes marcos arquitectónicos, se consideran diferentes niveles de interoperabilidad.

1.3. Principales beneficios de la interoperabilidad empresarial

Con base en lo presentado por Naser (2021), se evidencia los siguientes beneficios de la interoperabilidad que aportan a las empresas y a los usuarios:

- **Agilidad y calidad del servicio mediante la automatización**: los usuarios tienen la posibilidad de tener acceso a la información a cualquier hora y desde cualquier ubicación, aumentando así la calidad del servicio que reciben. De esta manera, la interoperabilidad se convierte en la base para la automatización.
- **Reducción de costos para las organizaciones y el ciudadano:** las instituciones que utilizan archivos y papel para transmitir información pueden ahorrar costos y tiempo al usar mecanismos electrónicos. La interoperabilidad y la disponibilidad de servicios en línea permiten reutilizar los recursos humanos, tecnológicos y logísticos para generar mayor valor.
- **Mayor transparencia:** los sistemas interoperables proporcionan datos veraces y fiables, permitiendo una mayor transparencia en su manipulación, gestión y publicación. Esto facilita la generación de información agregada, cruces de datos, validación de transacciones, aplicación de reglas y políticas, generación de informes, detección de situaciones anómalas y publicación de información veraz.

- **Posibilidad de mantenimiento:** la utilización de estándares facilita el mantenimiento de aplicaciones, ya que garantiza la existencia de personas con los conocimientos necesarios sobre tecnologías universalmente aceptadas, además de la disponibilidad de servicios en la nube como geolocalización, catálogos estandarizados y códigos reutilizables, garantizando la continuidad en el tiempo.

- **Información cohesionada:** se identifican los sistemas de información que operan de manera independiente para localizar la información redundante con el fin de mejorar la comunicación entre ellos.

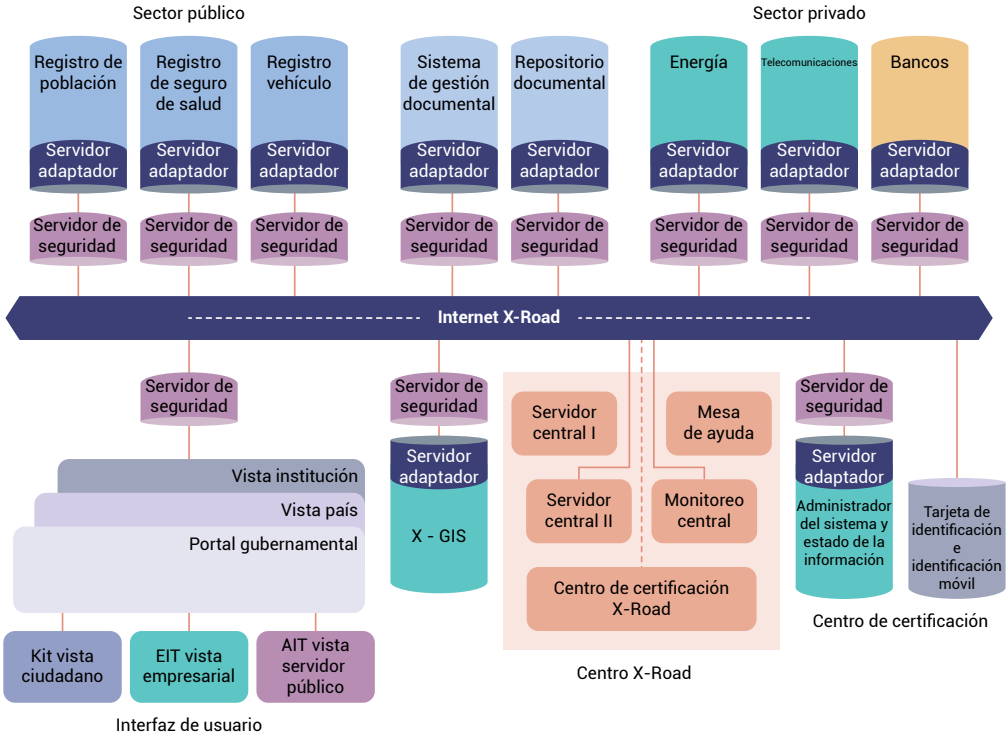
1.4. Experiencia de interoperabilidad: Estonia

Para ilustrar de manera práctica cómo se aplica la interoperabilidad en otros países, nos centraremos en la experiencia de Estonia. Este país ha implementado con éxito un sistema de interoperabilidad empresarial que se considera uno de los más avanzados del mundo. Gracias a esta iniciativa, las empresas estonias pueden comunicarse e intercambiar información con facilidad y rapidez, lo que ha impulsado su competitividad y su capacidad para adaptarse a los cambios del mercado. A continuación, veremos cómo funciona la interoperabilidad en Estonia y cómo ha beneficiado a su economía.

Como se menciona por parte de Naser (2021), Estonia es el séptimo mejor país de Europa según el índice de economía digital de la Unión Europea (Comisión Europea, 2020) y continúa siendo el líder en ofrecer sus servicios de manera digital. Prácticamente, el 100 % de los médicos recetan en línea gracias al sistema X-Road, el cual conecta a más de 1.000 bases de datos públicas y privadas, permitiendo el acceso a 1.700 trámites y servicios digitales, lo cual genera alrededor de 50 millones de transacciones por mes. Estas comunicaciones e interoperabilidades se aplican tanto a nivel nacional como transaccional, especialmente en el ámbito de la salud.

El estado está abierto las 24 horas de los siete días de la semana, donde los trámites se realizan totalmente digitalmente, salvo algunas excepciones como la compra de bienes inmuebles, matrimonio o divorcio. Para garantizar la seguridad de estos sistemas, se han implementado sellos de tiempo y se han registrado los ingresos a estos servicios digitales.

Figura 1.
Arquitectura X-Road.



Nota. Adaptado de niveles de interoperabilidad [Fotografía], por European Virtual Laboratory for Enterprise Interoperability, 2023, Interop-vlab (Enlace). CC BY 2.0

Como se evidencia en la f igura 1, X-Road está diseñada para permitir que los organismos de todo el mundo compartan datos de forma segura, confiable y eficiente. Está estructurada como una red de administradores de nodos que se conectan entre sí para compartir datos entre entidades. Estos nodos se conectan a través de una capa de seguridad y autenticación que se encarga de garantizar que todos los datos que pasan entre los nodos sean seguros y autenticados. Los nodos de esta red pueden ser aplicaciones, servicios *web*, bases de datos, sistemas de información geográfica o cualquier otro servicio digital que deseen intercambiar información.

En el siguiente video le invito a revisar más detalles de cómo Estonia se convierte en país digitalizado.

Bien, como habrá podido darse cuenta, en el video se menciona la importancia de la interoperabilidad en la era digital y cómo eso puede

ayudar a mejorar la calidad de los servicios públicos y la atención al ciudadano. Adicionalmente, se presenta una descripción del funcionamiento de la plataforma de X-Road.

Estimado estudiante, continuemos con el aprendizaje mediante su participación en la actividad que se describe a continuación:

Actividad de aprendizaje recomendada

Elabore un cuadro comparativo sobre los diferentes niveles de interoperabilidad empresarial.

Procedimiento: revise los conceptos descritos en el literal 1.3 de esta guía, así como otros conceptos relacionados que pueda encontrar en la *web*. Una vez que haya revisado estos conceptos, identifique posibles elementos comparativos que puedan utilizar para realizar una comparativa entre ellos.

Nota. conteste la actividad en un cuaderno de apuntes o documento Word.

Semana 2

Estimado estudiante, esta semana continuamos con el estudio de la unidad 1 de la asignatura de Interoperabilidad Empresarial, correspondiente a fundamentos de la interoperabilidad. Esta semana, vamos a examinar cómo la arquitectura de *software* puede contribuir a la interoperabilidad entre los sistemas de una empresa.

1.5. Arquitecturas de *software* y su papel en la interoperabilidad

La arquitectura de *software* es fundamental para la interoperabilidad, ya que proporciona una estructura y una guía para el desarrollo de un *software*. Esta arquitectura ayuda a determinar cómo se almacenarán los datos, cómo se comunicarán los diferentes componentes y cómo se conectarán a otros sistemas. La arquitectura de *software* también ofrece una visión de cómo se debe construir el *software*, de modo que sea escalable y fácil de mantener. Esto es especialmente importante para los

desarrolladores, ya que, sin una arquitectura adecuada, no hay forma de saber a dónde se dirige el proyecto y cómo llegar allí.

La interoperabilidad en la arquitectura de *software* se puede comparar con el canal de Panamá, ya que ambas permiten a los usuarios conectar dos sistemas diferentes, permitiendo que los datos o información fluya entre los dos. De la misma manera, el canal de Panamá conecta dos océanos, permitiendo que los buques de carga y los barcos de pasajeros puedan pasar de un lado a otro.

Adicional OpenGroup (2018), señala que la arquitectura de *software* tiene un enfoque fuerte no solo en la estructura del código, sino también en todos los requerimientos no funcionales tales como:

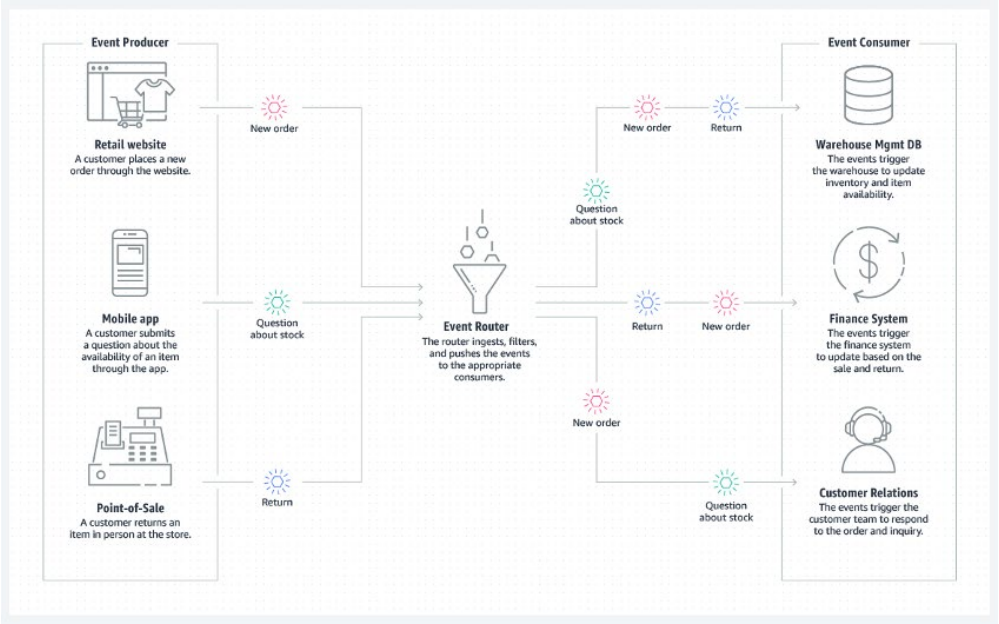
- Seguridad (autenticación, autorización, integridad y confidencialidad, auditabilidad, etc.).
- Fiabilidad.
- Rendimiento.
- Operabilidad.
- Mantenibilidad.
- Interoperabilidad.

1.5.1. Arquitectura B asada en E ventos (EDA)

La arquitectura basada en eventos se refiere al diseño de sistemas mediante la transferencia de mensajes asíncronos, denominados eventos, entre los componentes. Estos eventos son generados cuando suceden cosas y proporcionan una forma de conectar los componentes del sistema. Esta arquitectura permite construir sistemas distribuidos y altamente concurrentes de manera escalable.

Esta arquitectura se basa en cuatro elementos clave: generadores, consumidores, distribuidores y almacenes de eventos. Los generadores se encargan de crear los eventos que serán enviados a los consumidores, quienes tienen la tarea de procesarlos. Por otro lado, los distribuidores tienen la función de dirigir los eventos hacia los consumidores adecuados. Finalmente, los almacenes de eventos facilitan el almacenamiento y la recuperación de estos para un procesamiento futuro.

Figura 2.
Ejemplo de arquitectura basada en eventos



Nota. Tomado de Amazon (2023).

En la figura 2, se presenta un ejemplo de una arquitectura basada en eventos para un sitio de comercio electrónico. El beneficio que aporta este tipo de arquitectura es que el sitio reaccione a los cambios procedentes de diversas fuentes durante los momentos de mayor demanda, sin que se bloquee la aplicación ni se sobreaprovisionen.

1.5.1.1. Empresas que utilizan la arquitectura basada en eventos

Solace (2022), señala que, hacia mediados del año 2021, el 13 % de las organizaciones había alcanzado la madurez total de la arquitectura basada en eventos. A pesar de que las empresas de todos los tamaños en todo el mundo utilizan la arquitectura basada en eventos en diferentes grados, existen algunas que destacan en su implementación, como Heineken, Schwarz y Unilever. Además, otras organizaciones utilizan la arquitectura basada en eventos para casos de uso específicos, como Walmart, que la utiliza para el seguimiento de la cadena de suministro, y Uber, que la utiliza para la gestión de flotas y la asignación de viajes.

Heineken, como parte de su estrategia a largo plazo para convertirse en la mejor cervecería conectada y conectar más de 4.500 aplicaciones críticas

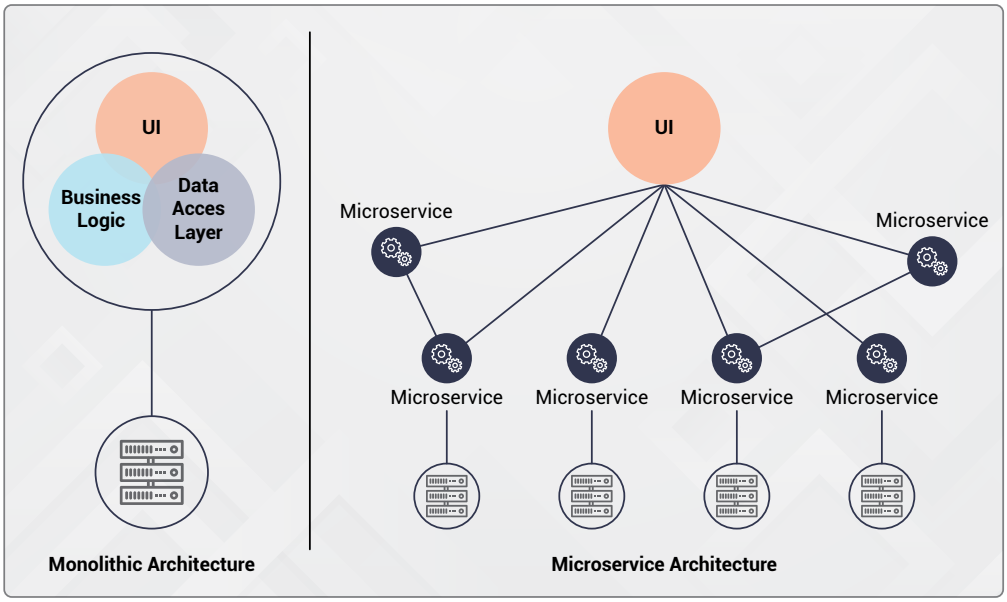
para el negocio en áreas como pagos, logística, gestión de inventario y más, decidieron pasar a una arquitectura basada en eventos. Debido a las muchas adquisiciones que tuvo, Heineken se enfrentaba a operaciones con sistemas empresariales heredados y fuentes de datos dispersas geográficamente que confiaban en la comunicación punto a punto con API de REST. El cambio a un enfoque basado en eventos ha tenido como resultado una mejor gestión de los picos y menos interrupciones en la producción, mejorando su eficiencia general.

1.5.2. Arquitectura basada en microservicios

La arquitectura basada en microservicios busca abordar los problemas asociados con las aplicaciones monolíticas que son muy comunes en el entorno empresarial. Esta arquitectura se basa en dividir la aplicación en módulos autocontenidos que se exponen a través de múltiples API, con el objetivo de maximizar la cohesión y minimizar el acoplamiento.

A diferencia de las arquitecturas monolíticas, los microservicios, pueden operar una o varias instancias en un solo servidor o en varios servidores, los recursos se escalan dinámicamente para satisfacer las necesidades de la carga de trabajo. Estos microservicios individuales se alojan a menudo en contenedores para incrementar su portabilidad y escalabilidad.

Figura 3.
Ejemplo de arquitectura monolítica y basada en microservicios



Nota. Ejemplo de una aplicación monolítica en la parte izquierda. En la parte derecha una arquitectura basada en microservicios: Tomado de Hiberus (2023).

1.5.2.1. Características

Según Intel (2021), los microservicios tienen ciertas características comunes, aunque no universales. Estas características incluyen:

- **Componentes:** los microservicios se componen generalmente de partes de *software* aisladas que pueden actualizarse y reemplazarse individualmente. Esta estructura tiene repercusiones en la gestión de la tecnología de la nube, pues cada uno de los microservicios debe ser configurado, supervisado y renovado de forma independiente.
- **Servicios:** los componentes incluyen servicios que están disponibles para ser solicitados, pero no necesariamente se mantienen activos entre los pedidos y las invocaciones.
- **Implementación independiente:** la mayoría de los componentes de los servicios individuales funcionan de forma separada entre sí dentro de la estructura de microservicios. Si se modifica o actualiza un componente, el efecto en otros servicios y componentes es bajo,

especialmente en comparación con una arquitectura monolítica más antigua.

- **Seguridad:** la transmisión de información entre los microservicios generalmente se lleva a cabo con cifrado de Seguridad de la Capa de Transporte Mutuo (mTLS) para proteger los datos de *software* malicioso y ataques externos mientras se encuentran en tránsito.
- **Contenerización:** estos elementos contribuyen a lograr una mayor escalabilidad y portabilidad.

1.5.2.2. Contenerización

La arquitectura de microservicios ofrece grandes ventajas, pero también supone enormes desafíos. Dockerizar la arquitectura de microservicios puede ayudar a gestionar estos desafíos, ya que permite a los desarrolladores empaquetar y desplegar fácilmente sus aplicaciones como contenedores individuales, lo que les permite optimizar el uso de la memoria, aumentar la velocidad de desarrollo y reducir el tiempo de inactividad. Esto, a su vez, facilita la escalabilidad y la flexibilidad de la infraestructura, permitiendo a los equipos de desarrollo implementar rápidamente nuevas características y solucionar los problemas de forma más eficiente.

De acuerdo a Jera(Jeremy H, 2022), Docker y Kubernetes son herramientas bien conocidas para la creación y administración de microservicios contenerizados. Estas herramientas automatizan el proceso de usar cgroups y namespaces de Linux para crear y administrar contenedores. Mientras que Docker se enfoca en crear contenedores, Kubernetes se concentra en la orquestación de contenedores. La Orquestación de Contenedores (CO, por sus siglas en inglés) es el proceso automático de administrar el trabajo de contenedores individuales para aplicaciones basadas en microservicios dentro de múltiples *clústeres*.

Estimado estudiante, continuemos con el aprendizaje mediante su participación en las actividades que se describen a continuación:

Actividades de aprendizaje recomendadas

1. Ingresar a Google y buscar información de la arquitectura basada en eventos. Pueda utilizar la frase “qué es la arquitectura basada en eventos”, una página que puede ser de gran ayuda es la página de TIBCO.

2. Siga los pasos detallados en la Guía a paso a paso para trabajar con una arquitectura basada en eventos, y aplique el caso propuesto. De esta manera, obtendrá una experiencia práctica en el manejo de **arquitecturas basadas en eventos.**

3. Siga los pasos detallados en la Guía a paso a paso para trabajar con una arquitectura basada en microservicios, y aplique el caso propuesto. De esta manera, obtendrá una experiencia práctica en el manejo de **arquitecturas basadas en microservicios.**

Nota. conteste las actividades en un cuaderno de apuntes o documento Word.

Semana 3

Estimado estudiante, esta semana continuamos con el estudio de la unidad 1 de la asignatura de Interoperabilidad Empresarial, correspondiente a fundamentos de la interoperabilidad. Esta semana, vamos a examinar qué beneficios aporta el modelado C4 a la interoperabilidad.

1.6. Beneficios que aporta el modelado C4 a la interoperabilidad empresarial

El modelo C4 (*Context-Container-Component-Code*) es una herramienta útil para la planificación y diseño de sistemas empresariales interoperables. Este modelo ayuda a los equipos de desarrollo a visualizar de manera clara y concisa las distintas capas de la arquitectura empresarial, desde el contexto del negocio hasta la implementación de código.

Tanto los arquitectos de *software* como los de edificios utilizan herramientas visuales para representar sus respectivos diseños.

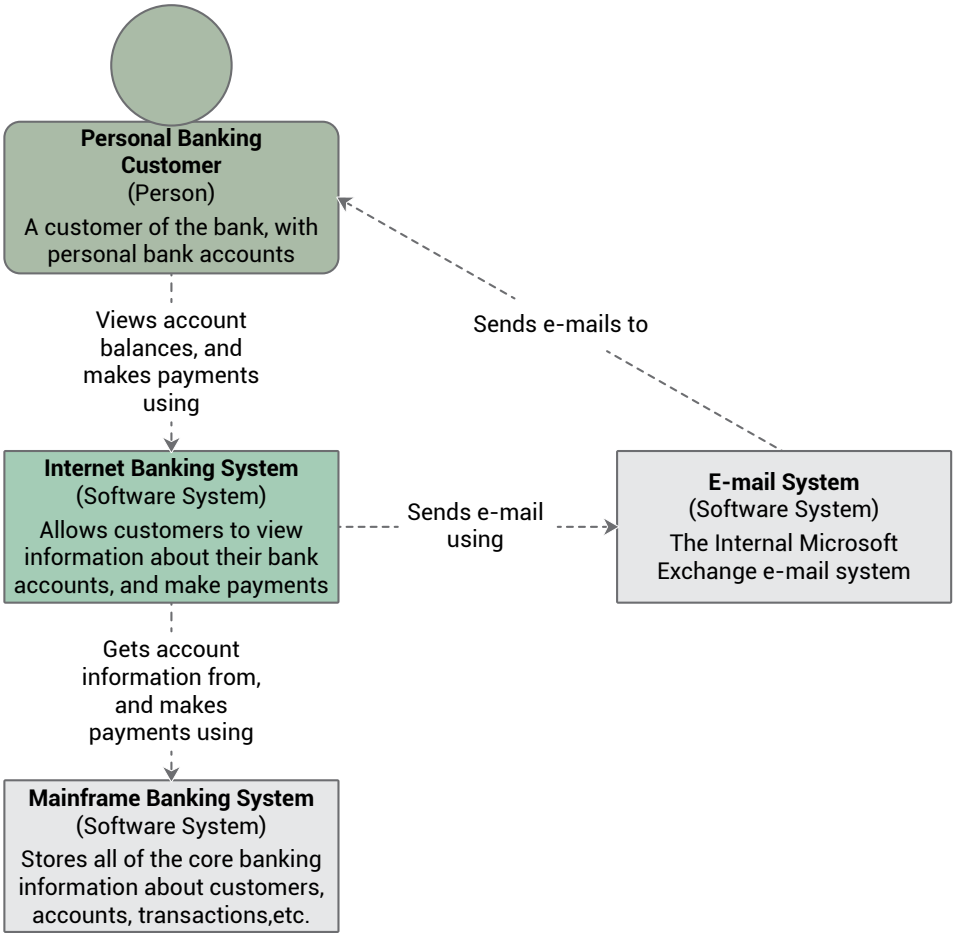
Mientras que los arquitectos de edificios utilizan planos y documentos de construcción, los arquitectos de *software* utilizan diagramas para dar significado visual a los conceptos abstractos. Sin embargo, el uso de formas, colores o símbolos no estandarizados puede generar confusión en el público. El modelo C4 ofrece una solución a este problema, ya que permite a los desarrolladores crear sistemas coherentes y escalables que optimizan la comunicación e intercambio de información entre diferentes sistemas empresariales.

1.6.1. Niveles principales del modelado C4

- **Contexto**: (Brown, 2018), menciona que el diagrama de contexto del sistema es un buen punto de partida para diagramar y documentar un sistema de *software*, permitiéndole dar un paso atrás y ver el panorama general. Se debe dibujar un diagrama que muestre el sistema como una caja en el centro, rodeado de sus usuarios y los demás sistemas con los que interactúa.

Figura 4.

Nivel de contexto en la se representa las relaciones del sistema con otros sistemas y con los diferentes actores.



Nota. Tomado Brown (2022).

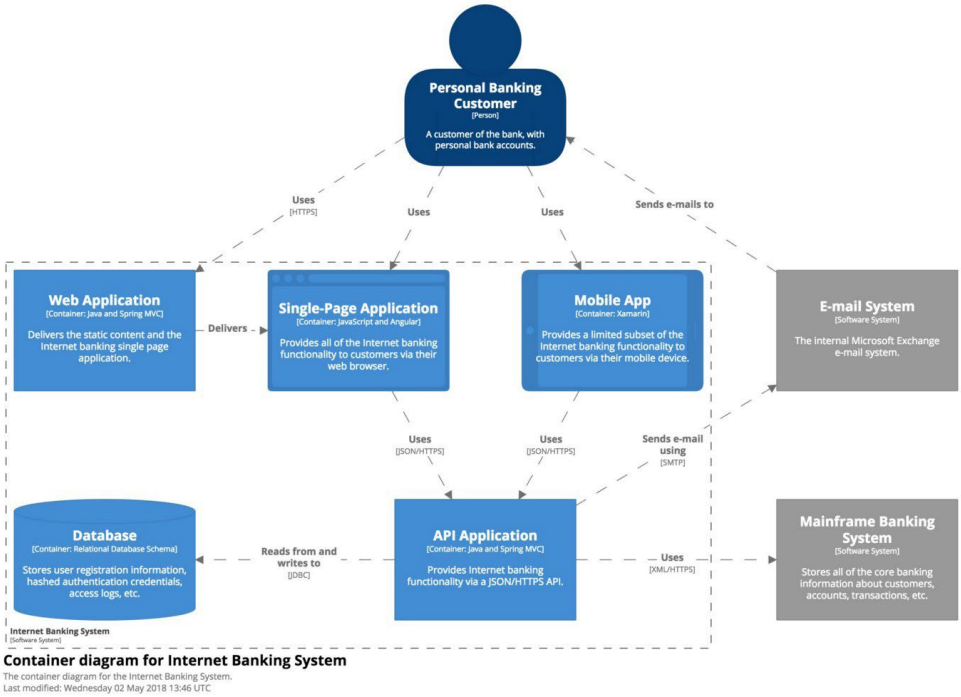
El nivel de contexto es importante para entender la interoperabilidad de una empresa, ya que permite visualizar la relación entre diferentes sistemas y su interacción con otros sistemas y usuarios. Al utilizar un diagrama de contexto del sistema, se puede identificar de manera clara y sencilla las dependencias entre sistemas y la información que fluye entre ellos. Esto es fundamental para garantizar una interoperabilidad efectiva y eficiente en la empresa, lo que a su vez permite una mejor integración de los procesos y una mayor eficiencia en la toma de decisiones.

- **Contenedor**: *u* na vez que entienda cómo su sistema se ajusta al entorno de TI en general, un paso realmente útil es acercarse al límite

del sistema con un diagrama de contenedor. Un “contenedor” es algo como una aplicación *web* del lado del servidor, una aplicación de página única, una aplicación de escritorio, una aplicación móvil, un esquema de base de datos, un sistema de archivos, etc. En esencia, un contenedor es una unidad ejecutable/desplegable por separado (por ejemplo, un espacio de proceso separado) que ejecuta código o almacena datos. El diagrama de contenedor muestra la forma de alto nivel de la arquitectura de *software* y cómo se distribuyen las responsabilidades en ella. También muestra las principales opciones de tecnología y cómo los contenedores se comunican entre sí. Es un diagrama simple y de alto nivel centrado en la tecnología que es útil tanto para desarrolladores de *software* como para el personal de soporte/operaciones.

Figura 5.

Nivel de contenedor, en la que se presenta tecnologías utilizadas en cada uno de los elementos

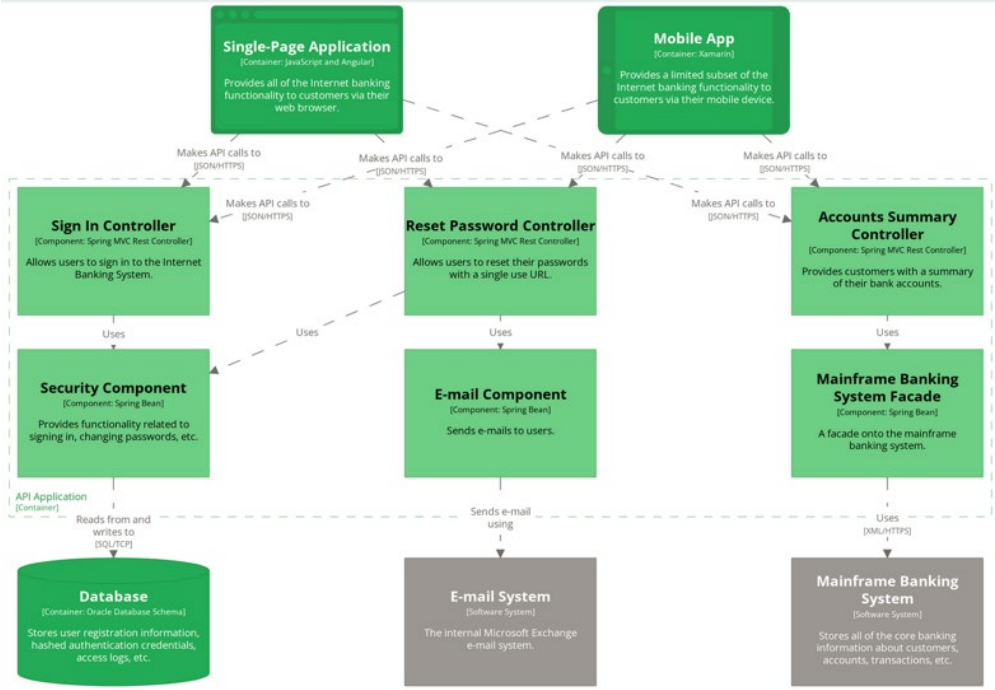


Nota. Tomado Brown (2022).

- **Componente**: *u*n paso más profundo que el diagrama de contenedor, el diagrama de componentes detalla grupos de código dentro de un

solo contenedor. Estos componentes representan abstracciones de su base de código.

Figura 6.
Nivel de componente, en la que se presenta abstracciones del código base.



Nota. Tomado Brown (2022).

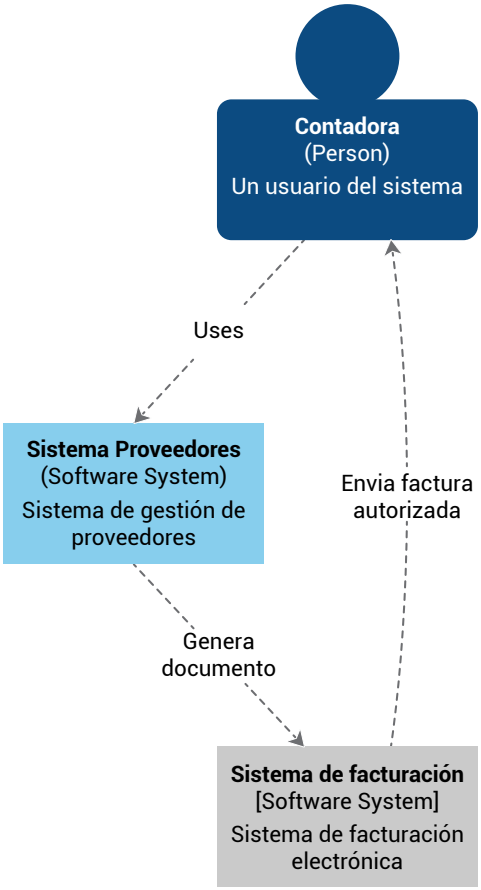
1.6.2. Caso práctico

Para comprender mejor el tema de esta semana, vamos a plantearnos un escenario que nos permitirá visualizar la importancia del modelado C4 en la integración de sistemas.

Imaginemos que una empresa necesita integrar dos sistemas diferentes, uno para la gestión de proveedores y otro para la facturación electrónica, con el objetivo de cumplir la normativa fiscal. Cabe destacar que ambos sistemas han sido desarrollados por empresas diferentes y utilizan tecnologías distintas.

Partiremos con el diagrama de contexto, que nos dará una visión del sistema y su entorno. En la figura 7, se evidencia la relación entre los sistemas de facturación y proveedores.

Figura 7.
Diagrama de contexto para el caso de integración con facturación electrónica.

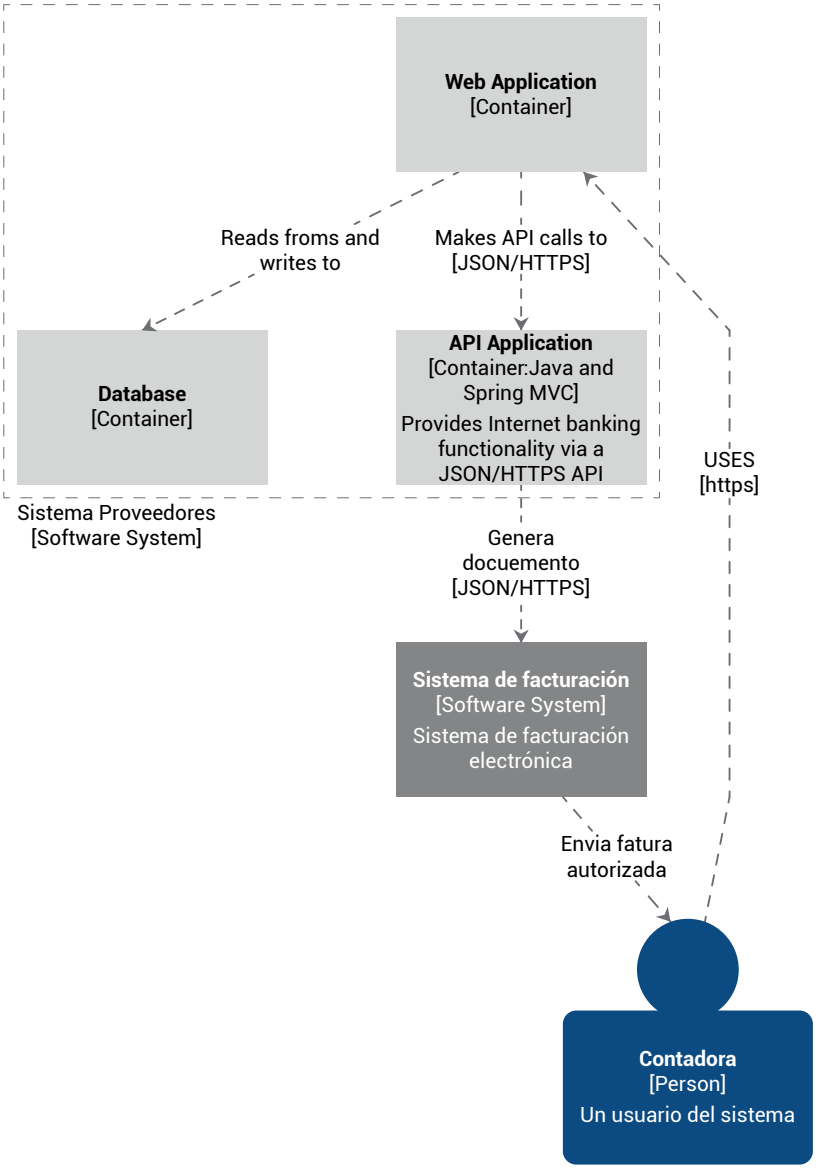


Nota. Quiñonez, F. 2023.

Ahora realizaremos el diagrama de contenedor, para tener una visión de cómo se interconectan los sistemas. En la figura 8, se evidencia la integración a nivel de componentes, entre la API del sistema de proveedores y la API del sistema de facturación.

Figura 8.

Diagrama de contenedor para el caso de integración con facturación electrónica



Nota. Quiñonez, F. 2023.

Como ha podido evidenciar el modelado C4 nos ha permitido representar con claridad la arquitectura de ambos sistemas, incluyendo los componentes que los conforman, las relaciones entre ellos y los protocolos de comunicación que se utilizan. Como se puede apreciar, este enfoque ha sido fundamental para obtener una visión completa y detallada de

la estructura de los sistemas en cuestión. Para visualizar a detalle el caso realizado lo invito a revisar el documento sobre el Caso práctico del Sistema de facturación.

Estimados estudiantes, continuemos con el aprendizaje mediante su participación en las actividades que se describen a continuación:

Actividades de aprendizaje recomendadas

1. Elabore un diagrama C4 que represente las vistas de contexto y contenedor de un sistema de préstamo de libros en una biblioteca. El sistema incluirá un sitio *web* de solicitud y un sitio *web* de administración. Además, se integrará con la plataforma de envío de mensajes WhatsApp (API) para confirmar el préstamo de un libro a un estudiante.

   Procedimiento: analice el contenido descrito en el literal 1.6 de esta guía, así como otros recursos relacionados que pueda encontrar en la *web*. A continuación, utilice una herramienta como Structurizr para realizar su diagrama en línea. Para apoyar esta actividad revise la Guía paso a paso C4 model con Structurizr.

2. Una vez completadas las primeras tres semanas de estudio, le recomiendo llevar a cabo la siguiente autoevaluación con el objetivo de verificar el aprendizaje obtenido. Al concluir la guía, se proporcionan las respuestas a dichas autoevaluaciones; no obstante, se insta a revisarlas únicamente después de haber registrado sus propias soluciones, permitiéndole de esta manera evaluar adecuadamente el nivel de conocimientos adquiridos.

Autoevaluación 1

Lea detenidamente cada una de las preguntas y seleccione la alternativa según corresponda.

1. ¿Cuál es una de las definiciones de interoperabilidad según El Open Group en el contexto de la versión 9 de TOGAF?

   a. La capacidad de un sistema para funcionar de manera independiente sin interacción con otros sistemas.

   b. La habilidad de un sistema para protegerse de amenazas externas y mantener la integridad de los datos.

   c. La capacidad de compartir información y servicios, de intercambiar y usar información entre dos o más sistemas, y de proporcionar y recibir servicios de otros sistemas para funcionar juntos de manera eficaz.

2. Según Kotzé & Neaga, ¿a qué se refiere la interoperabilidad empresarial?

   a. A la interoperabilidad entre dispositivos electrónicos y sistemas operativos.

   b. A la interoperabilidad entre unidades organizacionales o procesos de negocio, ya sea dentro de una gran empresa (distribuida) o dentro de una red empresarial.

   c. A la capacidad de un sistema para adaptarse a diferentes entornos y condiciones.

3. ¿Cuál de las siguientes opciones no es un nivel de interoperabilidad?

   a. Organizacional.
   b. Semántica.
   c. Funcional.

4. ¿Cuál de los siguientes beneficios de la interoperabilidad empresarial es falso?

   a. Mayor agilidad y calidad del servicio mediante la automatización.
   b. Aumento de costos para las organizaciones y los ciudadanos.
   c. Mayor transparencia en la manipulación, gestión y publicación de datos.

5. ¿Cuál de los siguientes componentes de una arquitectura basada en eventos es responsable de enrutar los eventos a los consumidores apropiados?

   a. Generadores de eventos.
   b. Consumidores de eventos.
   c. Sistemas de distribución de eventos.

6. ¿Cuál de las siguientes características es común en la arquitectura basada en microservicios?

   a. Componentes monolíticos.
   b. Implementación independiente.
   c. Servicios siempre activos.

7. ¿Cuál es una diferencia clave entre la arquitectura de microservicios y la arquitectura basada en eventos?

   a. Los microservicios se centran en módulos autocontenidos, mientras que la arquitectura basada en eventos se enfoca en la transferencia de mensajes asíncronos.

   b. Ambas arquitecturas requieren una implementación monolítica.

   c. La arquitectura de microservicios se basa en la transferencia de mensajes asíncronos, mientras que la arquitectura basada en eventos se centra en módulos autocontenidos.

8. ¿Cuál es el propósito principal del nivel de contexto en el modelo C4?

   a. Identificar las tecnologías utilizadas en cada elemento del sistema.
   b. Mostrar la arquitectura de *software* de alto nivel y cómo se distribuyen las responsabilidades en ella.
   c. Visualizar la relación entre diferentes sistemas y su interacción con otros sistemas y usuarios.

9. En el modelo C4, ¿qué se detalla en el diagrama de componentes?

   a. La relación entre diferentes sistemas y su interacción con otros sistemas y usuarios.
   b. Las tecnologías utilizadas en cada elemento del sistema y cómo se comunican los contenedores entre sí.
   c. Grupos de código dentro de un solo contenedor, representando abstracciones de la base de código.

10. ¿Cuál es uno de los principales beneficios del modelo C4 en comparación con otras herramientas visuales utilizadas por los arquitectos de *software*?

    a. Permite a los desarrolladores utilizar formas, colores y símbolos no estandarizados.

    b. Facilita la creación de sistemas coherentes y escalables que optimizan la comunicación e intercambio de información entre sistemas empresariales.

    c. Proporciona una herramienta exclusiva para arquitectos de edificios.

Ir a solucionario

Semana 4

Estimado estudiante, esta semana continuamos con el estudio de la unidad 1 de la asignatura de Interoperabilidad Empresarial, correspondiente a fundamentos de la interoperabilidad. Esta semana, vamos a examinar los estándares y protocolos.

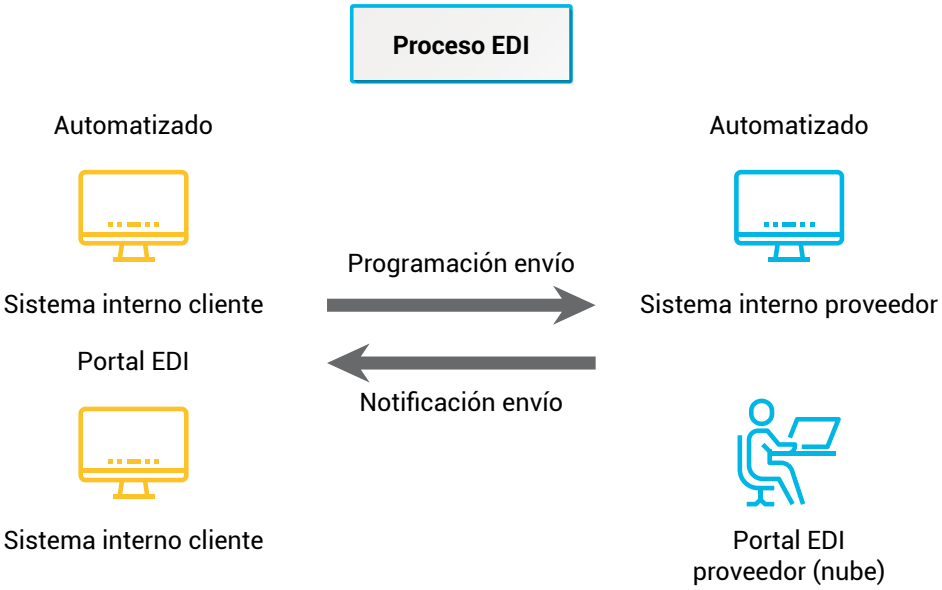
1.7. Estándares y protocolos de interoperabilidad

Es esencial comprender la importancia de los estándares y protocolos de interoperabilidad empresarial en el contexto actual de la economía global y digital. Estos mecanismos estandarizados facilitan la colaboración y la comunicación entre diferentes sistemas, aplicaciones y dispositivos dentro de una organización o entre múltiples organizaciones, garantizando la eficiencia y la productividad en los procesos empresariales. Al adoptar y seguir estos estándares, las empresas pueden enfrentarse a retos tecnológicos y de mercado con mayor confianza, agilidad y escalabilidad.

1.7.1. EDI

EDI (*Electronic Data Interchange*) es un conjunto de normas que permite el intercambio electrónico de documentos y datos comerciales estructurados entre organizaciones, agilizando y automatizando procesos como la emisión de órdenes de compra, facturación y seguimiento de envíos.

Figura 9.
Proceso EDI del flujo de información entre los sistemas



Nota. Quiñonez, F. 2023.

La figura 9 ilustra el flujo de información en el proceso EDI, donde los datos recolectados se interpretan e incorporan directamente en los sistemas de información para su procesamiento instantáneo y sin la necesidad de intervención humana.

Los mensajes estandarizados EDI se expresan en términos de sintaxis y vocabulario específicos. Para que las computadoras puedan leerlas e interpretarlas utilizando un lenguaje compartido. La información se recopila y se convierte en elementos de datos tales como:

- Identificación del transmisor (identidad).
- Identificación del destinatario.
- Dirección.
- Número del documento.
- Fecha.
- Número de producto.
- Cantidad.
- Precio unitario.
- País (un código dedicado).
- Tipo de interlocutor (proveedor o cliente).
- Tecla de control (códigos de barras).

Un ejemplo destacado de una empresa que utiliza EDI es Walmart, el gigante minorista. Como menciona (*EDI Para Walmart | Meade Willis Inc.*, n.d.), Walmart ha implementado EDI para optimizar la comunicación con sus numerosos proveedores y mejorar la gestión de su cadena de suministro, al facilitar el intercambio de documentos comerciales como órdenes de compra, avisos de envío y facturas electrónicas. Esta implementación ha permitido a Walmart reducir costos, disminuir errores y aumentar la velocidad y precisión en sus operaciones logísticas.

1.7.2. XML

El Lenguaje de Marcado Extensible (XML, por sus siglas en inglés) es un estándar ampliamente utilizado para estructurar y almacenar datos de manera jerárquica y legible tanto para humanos como para computadoras (Allen, 2006). XML es altamente flexible, lo que permite a los desarrolladores crear etiquetas personalizadas y definir esquemas para describir la estructura y el contenido de los datos. Esta flexibilidad ha llevado a la adopción de XML en diversas aplicaciones, como la transferencia de datos entre sistemas, la configuración de aplicaciones y el almacenamiento de documentos.

La adopción de XML en los servicios SOAP ha permitido una mayor integración y comunicación entre sistemas distribuidos, facilitando el intercambio de datos y la implementación de servicios *web* en una amplia variedad de aplicaciones empresariales y tecnológicas.

En Ecuador, el formato XML se utiliza ampliamente en la facturación electrónica, facilitando la emisión, recepción y procesamiento de comprobantes fiscales digitales, como facturas, notas de crédito y débito, y guías de remisión. La adopción de XML en la facturación electrónica permite una representación estructurada y estandarizada de la información contenida en los documentos fiscales, lo que garantiza una mayor eficiencia en la comunicación entre contribuyentes y el Servicio de Rentas Internas (SRI). Además, el uso de XML en este contexto promueve la transparencia fiscal, la trazabilidad de las transacciones comerciales y la disminución de errores en la información, lo que contribuye a la mejora de la gestión tributaria y la prevención del fraude fiscal en el país.

1.7.3. SOAP

SOAP (*Simple Object Access Protocol)* es un protocolo de comunicación basado en XML diseñado para facilitar la interoperabilidad entre aplicaciones a través de la *web* (Box, 2000). Desarrollado por la World Wide *Web* Consortium (W3C), SOAP permite el intercambio de información estructurada entre sistemas heterogéneos, independientemente del lenguaje de programación o la plataforma utilizada.

SOAP desempeña un papel crucial en la interacción controlada y restringida entre sistemas. En lugar de otorgar acceso completo al servidor al cliente solicitante, un protocolo como SOAP permite limitar el acceso a las funciones necesarias. La arquitectura del protocolo brinda un marco en el que la aplicación puede integrarse, lo que permite la colaboración entre sistemas muy diferentes (Ionos, 2020).

Numerosos servicios *web* se basan en SOAP. Por ejemplo, Amazon y eBay utilizan parcialmente este protocolo de red para sus operaciones (Ionos, 2020). Al comprender y utilizar SOAP, los desarrolladores pueden aprovechar sus ventajas para integrar y coordinar sistemas heterogéneos.

- **Estructura SOAP**

Usualmente, SOAP se incorpora al protocolo HTTP. El transporte se efectúa a través del protocolo y se fusiona con su estructura. Un mensaje HTTP que incluye una solicitud SOAP posee la estructura ilustrada en la figura 10.

Figura 10.
Estructura SOAP de un mensaje enviado por HTTP.

```
POST /example HTTP/1.1
Host: example.org
Content-Type: text/xml; charset=utf-8
…
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
...
   <SOAP-ENV:Header>
      ...
   </SOAP-ENV:Header>

   <SOAP-ENV:Body>
      ...
   </SOAP-ENV:Body>

</SOAP_ENV:Envelope>
```

Nota. Quiñonez, F. 2023.

En la figura 10, se puede visualizar que la solicitud se inicia con un encabezado HTTP. Luego, se encuentra el *Envelope* (sobre) de SOAP, que encapsula el contenido real del mensaje, similar a un sobre. Los componentes principales de SOAP son el Header (encabezado) y el Body (cuerpo) (Ionos, 2020).

- ***Header*:** el encabezado de la solicitud SOAP incluye metadatos, como el tipo de cifrado utilizado. Su uso es opcional.
- ***Body*:** El cuerpo del mensaje contiene los datos en sí mismos.

Los conceptos empleados en el *Body* no están relacionados con SOAP, sino que dependen totalmente de la aplicación (Ionos, 2020).

Frecuentemente, el protocolo se combina con el lenguaje WSDL (Web Services Description Language), el cual es lenguaje de descripción específico para servicios *web* independiente de la plataforma. Con su ayuda, un cliente puede identificar los servicios que ofrece un servicio *web*. A partir del archivo WSDL, el cliente determina las opciones disponibles para realizar una solicitud SOAP. La combinación de WSDL y SOAP permite que dos sistemas diferentes se comuniquen sin necesidad de adaptaciones previas.

En ejemplo de WSDL se puede encontrar en la especificación del servicio *web* de facturas electrónicas del SRI.

1.7.4. REST

La arquitectura Representational State Transfer (**REST**), introducida por Roy Fielding en su tesis doctoral en el año 2000, ha emergido como un enfoque predominante para diseñar aplicaciones *web* escalables y eficientes (Fielding, 2000). Las aplicaciones basadas en REST aprovechan los protocolos y principios de la *web*, como el P rotocolo de T ransferencia de H ipertexto (HTTP) y el uso de U niform R esource I dentifiers (URI), para facilitar la comunicación entre clientes y servidores. Estos servicios *web*, conocidos como RESTful, proporcionan una interfaz simple y estandarizada para el intercambio de información entre diferentes sistemas.

En la tabla 1, se presentan los principales métodos soportados por HTTP, los mismos que son utilizados para la arquitectura REST.

Tabla 1.
Métodos HTTP

Método HTTP	Descripción
POST	Utilizado para la creación de un nuevo recurso
PUT	Modificar un recurso existente
GET	Consultar información de un recurso
DELETE	Eliminar un recurso especifico
PATCH	Actualizar únicamente un atributo de un recurso

Nota. Quiñonez, F. 2023.

Una URI (*endpoint*) actúa como un identificador exclusivo para cada recurso en el sistema REST, por lo que no se permite que más de un recurso comparta la misma dirección. La estructura básica de una dirección URL se presenta de la siguiente manera:

```
{protocolo}://{hostname}:{puerto}/{ruta_del_recurso}?{parámetros_de_filtrado(opcional)}
```

Un ejemplo de un *endpoint*, es la API que ofrece Facebook para el envío de mensajes por WhatsApp :

```
https://graph.facebook.com/2/234234234/messages
```

Otro punto importante a tener con REST, es que debe utilizar códigos de respuesta HTTP para indicar el resultado de una solicitud. Estos códigos de estado son números de tres dígitos que proporcionan información sobre el éxito, la falla o la necesidad de más acciones en una solicitud. Por ejemplo, el código de respuesta 200 indica que la solicitud se ha completado con éxito y se ha devuelto el contenido solicitado, mientras que el código 404 señala que el recurso solicitado no se encontró en el servidor. Los códigos de respuesta HTTP son esenciales en las aplicaciones RESTful, ya que ayudan a los desarrolladores a diagnosticar y solucionar problemas en la comunicación entre el cliente y el servidor.

En tabla 2 se presentan los códigos de respuesta HTTP más utilizados en REST.

Tabla 2.
Códigos de respuesta HTTP

Código	Descripción
200	OK: La solicitud se completó con éxito
201	Created: Se creó un nuevo recurso
204	No Content: Solicitud exitosa sin contenido para devolver
400	Bad Request: La solicitud es incorrecta o mal formada
401	Unauthorized: Se requiere autenticación
403	Forbidden: El cliente no tiene permiso para acceder al recurso
404	Not Found: El recurso no se encontró en el servidor
500	Internal Server Error: Error en el servidor al procesar la solicitud
503	Service Unavailable: El servidor no puede manejar la solicitud en este momento debido a mantenimiento o sobrecarga

Nota. Quiñonez, F. 2023.

Actividades de aprendizaje recomendadas

Estimado estudiante, continuemos con el aprendizaje mediante su participación en las actividades que se describen a continuación:

1. Revise el documento Ficha técnica comprobantes electrónicos del SRI, en las páginas a partir de la 43, en dichas páginas encontrarán la estructura de los comprobantes electrónicos en formato XML.

2. Siga los pasos detallados en la Guía a paso a paso consumo de api rest utilizando postman, y aplique el caso propuesto. De esta manera, obtendrá una experiencia práctica en el consumo de un api utilizando herramientas como Postman.

Nota. conteste la actividad en un cuaderno de apuntes o documento Word.

Semana 5

Estimado estudiante, esta semana continuamos con el estudio de la unidad 1 de la asignatura de Interoperabilidad Empresarial, correspondiente a fundamentos de la interoperabilidad. Esta semana, vamos a examinar diferentes estilos de integración empresarial tales como de aplicación, datos, procesos y dispositivos.

1.8. Estilos de integración empresarial: aplicaciones, datos, procesos y dispositivos

Los estilos de integración empresarial se pueden dividir en cuatro categorías: aplicaciones, datos, procesos y dispositivos.

1.8.1. Aplicaciones

La integración de aplicaciones se enfoca en conectar aplicaciones y sistemas dispares dentro de una empresa. La integración de aplicaciones puede realizarse mediante el uso de *middleware*, adaptadores de *software* o servicios *web*.

Un ejemplo de este tipo de integración se puede encontrar en una institución educativa donde se integran el sistema de matrículas y el sistema de biblioteca. La integración entre ambos sistemas permite una comunicación fluida y una gestión más eficiente de la información del estudiante.

En el proceso de matrícula, los estudiantes proporcionan sus datos personales y se inscriben en cursos específicos. Esta información es almacenada y gestionada por el sistema de matrículas. Por otro lado, el sistema de biblioteca gestiona la disponibilidad de libros y recursos, así como el préstamo y devolución de estos materiales por parte de los estudiantes.

Al integrar ambos sistemas, se pueden compartir datos relevantes, como la información del estudiante, su estado académico y los cursos en los que están matriculados. Por ejemplo, cuando un estudiante se matricula en un nuevo curso, su información se actualiza automáticamente en el sistema de biblioteca. Esto permite al sistema de biblioteca otorgar acceso a los recursos relacionados con el curso y mantener un registro de los préstamos de libros y materiales de los estudiantes.

Figura 11.
Integración de aplicaciones mediante un middleware.

Nota. Quiñonez, F. 2023

Además, la integración de estos sistemas puede facilitar el acceso a los servicios bibliotecarios en línea. Los estudiantes pueden iniciar sesión en el sistema de biblioteca utilizando las mismas credenciales que usan para el sistema de matrículas, lo que simplifica el proceso de autenticación y reduce la necesidad de mantener múltiples contraseñas. A través del sistema integrado, los estudiantes pueden buscar y reservar recursos, ver sus préstamos actuales y recibir notificaciones sobre fechas de vencimiento o multas.

1.8.2. Datos

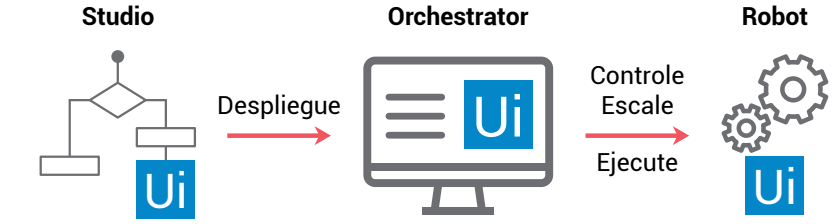
La integración de datos implica la consolidación y sincronización de información proveniente de diferentes fuentes de datos en una organización. Esto puede incluir la combinación de bases de datos, la transformación de datos en diferentes formatos y la creación de repositorios de datos unificados. La integración de datos permite a las organizaciones acceder a información coherente y actualizada en tiempo real, lo cual es fundamental para la toma de decisiones efectiva.

1.8.3. Procesos

La integración de procesos se centra en coordinar y automatizar los procesos comerciales que involucran múltiples sistemas y aplicaciones. La integración de procesos puede incluir la implementación de flujos de trabajo, la definición de reglas de negocio y la orquestación de servicios para garantizar que los procesos se ejecuten de manera eficiente y se adapten a las cambiantes necesidades del negocio.

Herramientas como UiPath, una plataforma líder en A utomatización de P rocesos R obóticos (RPA), juegan un papel fundamental en este tipo de integraciones. UiPath permite a las organizaciones diseñar, implementar y gestionar *robots* de *software* que pueden interactuar con sistemas y aplicaciones existentes de manera similar a cómo lo haría un humano, automatizando tareas repetitivas y reduciendo errores humanos. Al integrar diferentes sistemas empresariales mediante la automatización de procesos, UiPath contribuye a mejorar la eficiencia operativa, optimizar los recursos y aumentar la agilidad en la toma de decisiones dentro de la organización.

Figura 12.
Principales componentes de UiPath. Tomado de FirstBit (2023)



Nota. Tomado de Principales componentes de UiPath [Ilustración], por FirstBit, 2023, (Enlace), CC BY 2.0

En la figura 12 podemos revisar los componentes principales de UiPath. En los que se destaca:

- **UiPath Studio:** es el Entorno de Desarrollo Integrado (IDE) de UiPath que permite a los desarrolladores diseñar y crear flujos de trabajo de automatización de manera visual e intuitiva. Studio ofrece una amplia gama de actividades predefinidas y personalizables que se pueden arrastrar y soltar en el lienzo para construir secuencias y diagramas de flujo. Además, UiPath Studio es compatible con varios lenguajes de programación, lo que facilita la creación de soluciones de automatización más avanzadas y personalizadas.

- **UiPath Orchestrator:** es el componente central de administración y supervisión de la plataforma UiPath. Orchestrator permite a las organizaciones gestionar y controlar múltiples *robots* y procesos de automatización desde un único panel de control.

- **UiPath Robot:** es el componente encargado de ejecutar las tareas de automatización diseñadas en UiPath Studio. Los *robots* de UiPath pueden interactuar con sistemas y aplicaciones existentes de la misma forma que lo haría un usuario humano, lo que permite automatizar procesos repetitivos y reducir errores. Estos *robots* pueden funcionar en dos modos: asistido y desasistido.

1.8.4. Dispositivos

Por último, la integración de dispositivos se refiere a la conexión de dispositivos físicos y sistemas de *hardware* en la infraestructura

empresarial. Esto puede incluir la incorporación de dispositivos móviles, sensores, actuadores y otros dispositivos de *Internet* de las cosas (IoT) en los sistemas de información de la empresa. La integración de dispositivos en una organización facilita la recopilación y análisis de datos en tiempo real, mejorando así la eficiencia, la toma de decisiones y el monitoreo de operaciones.

Actividad de aprendizaje recomendada

Estimado estudiante, continuemos con el aprendizaje mediante su participación en la actividad que se describe a continuación:

Siga los pasos detallados en la Guía paso a paso de automatización de un proceso utilizando la herramienta Uipath, y aplique el caso propuesto. De esta manera, obtendrá una experiencia práctica de utilizar herramientas como UIpath para integrar diferentes aplicativos.

Nota. conteste la actividad en un cuaderno de apuntes o documento Word.

Semana 6

Estimado estudiante, esta semana continuamos con el estudio de la unidad 1 de la asignatura de Interoperabilidad Empresarial, correspondiente a fundamentos de la interoperabilidad. Esta semana, vamos a examinar la integración de sistemas en la nube, especialmente abordaremos la temática de despliegues en la nube. En esta semana es muy importante que revise las actividades recomendadas, ya que se incluirá algunas prácticas que es necesario que aplique para fortalecer el conocimiento teórico.

1.9. Integración de sistemas en la nube

La integración de sistemas en la nube es una de las tendencias más importantes en el desarrollo de aplicaciones en la actualidad. Esta tecnología permite conectar aplicaciones, servicios y datos de forma segura en un entorno virtual. Esto permite a los usuarios obtener una

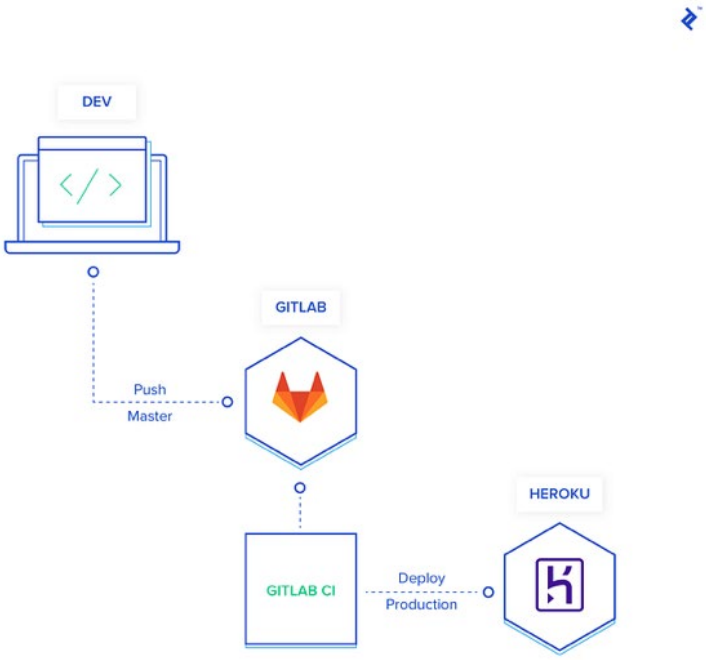
mayor eficiencia y una mejor experiencia de usuario, al tiempo que reducen los costos de desarrollo y mantenimiento.

1.9.1. Heroku (plataforma como servicio)

Heroku es una plataforma en la nube que nos permite construir, desplegar, gestionar y escalar aplicaciones *web* de manera rápida y eficiente. Ofrecida como un Servicio de Plataforma (PaaS), Heroku es compatible con una amplia variedad de lenguajes de programación, como Python, Ruby, Java, Node.js, entre otros. Su diseño modular y flexible permite la implementación de aplicaciones utilizando una amplia gama de tecnologías y servicios adicionales, como bases de datos, servidores *web* y herramientas de análisis. Adicional, esta herramienta proporciona un entorno de desarrollo y producción completamente administrado, permitiendo a los desarrolladores centrarse en escribir y mejorar su código sin preocuparse por la infraestructura subyacente.

Uno de los aspectos clave en el que Heroku ayuda a los equipos de desarrollo es en la implementación de prácticas de Integración Continua y Despliegue Continuo (CI/CD). La plataforma ofrece un flujo de trabajo simplificado para la automatización de pruebas, integración y despliegue de aplicaciones, facilitando así la detección temprana de errores y reduciendo el tiempo de lanzamiento al mercado. Además, Heroku se integra fácilmente con herramientas de control de versiones como Git, permitiendo una colaboración eficiente entre los miembros del equipo. Entre los componentes esenciales de Heroku se encuentran los “ Dynamos” , que son contenedores ligeros y eficientes para ejecutar aplicaciones.

Figura 13.
How to Build an Effective Initial Deployment Pipeline



Nota. Tomado de Young (2018)

En la figura 13, podemos observar como es el flujo desde origen de desarrollo hasta el despliegue en la plataforma de Heroku.

Actividad de aprendizaje recomendada

Estimado estudiante, continuemos con el aprendizaje mediante su participación en la actividad que se describe a continuación:

Siga los pasos detallados en la Guía a paso a paso para desplegar una aplicación dentro de Heroku desde GitHUB, y aplique el caso propuesto. De esta manera, obtendrá una experiencia práctica de cómo se puede trabajar con la integración continua en entornos *cloud*.

Estimado estudiante, esta semana continuamos con el estudio de la unidad 1 de la asignatura de Interoperabilidad Empresarial, correspondiente a fundamentos de la interoperabilidad. Esta semana, vamos a examinar un producto de Microsoft Azure que permite realizar una gestión de API. En esta semana es muy importante que revise las actividades recomendadas, ya que se incluirá algunas prácticas que es necesario que aplique para fortalecer el conocimiento teórico.

1.10. Revisión de componentes de la nube que permite la interoperabilidad empresarial

La nube ha jugado un papel fundamental en el logro de esta interoperabilidad, ofreciendo componentes y servicios que facilitan la comunicación y el intercambio de datos entre sistemas heterogéneos. Algunos de estos componentes incluyen API , *gateways*, servicios de mensajería y almacenamiento de datos. Estos elementos permiten a las organizaciones aprovechar al máximo las capacidades de la nube, integrando aplicaciones y sistemas en múltiples plataformas y tecnologías.

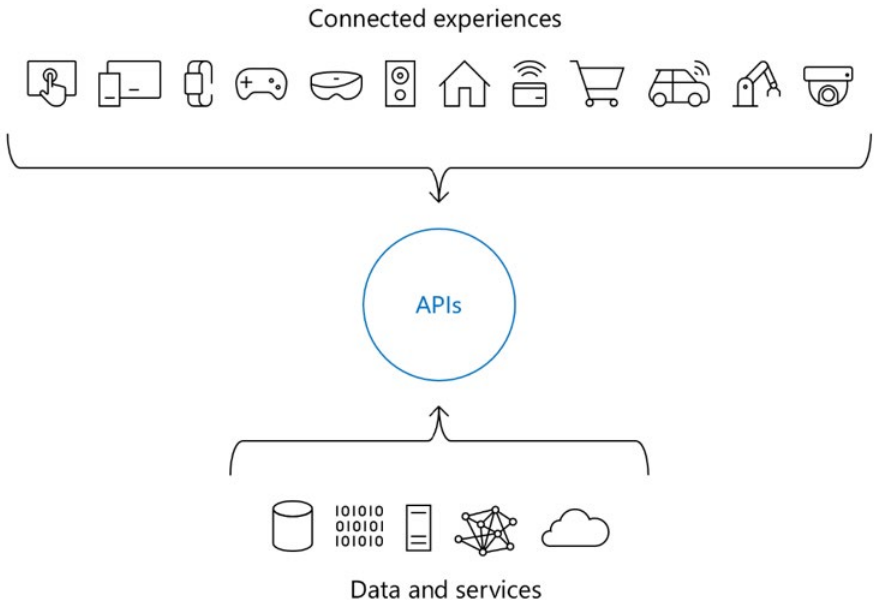
Un punto muy importante es que Microsoft Azure API Management es una solución integral que permite a las empresas publicar, proteger y supervisar sus API en un entorno escalable y seguro. Esta herramienta facilita la creación y gestión de API , permitiendo a las organizaciones exponer sus servicios internos y externos de forma unificada y controlada. Con Azure API Management, las empresas pueden definir políticas de acceso y autenticación, limitar las tasas de solicitudes y analizar el uso de las API para mejorar el rendimiento y la eficiencia. Además, esta solución ayuda a garantizar que las API sean compatibles con múltiples protocolos y formatos de datos, promoviendo así la interoperabilidad empresarial.

1.10.1. Introducción a Microsoft Azure API Management

La plataforma Azure API Management, que abarca múltiples nubes e híbrida, permite gestionar las API en todos los contextos. Al ser un servicio de plataforma, API Management respalda el ciclo de vida integral de la API (Microsoft, 2023).

A medida que las API se vuelven más extendidas y las organizaciones dependen cada vez más de ellas, es esencial que se gestionen como recursos de primer nivel a lo largo de todo su ciclo de vida. En la figura 14, se puede ver las API como un componente central al momento de realizar integraciones.

Figura 14.
Exponer de forma segura un API.



Nota. Tomado de Microsoft (2023).

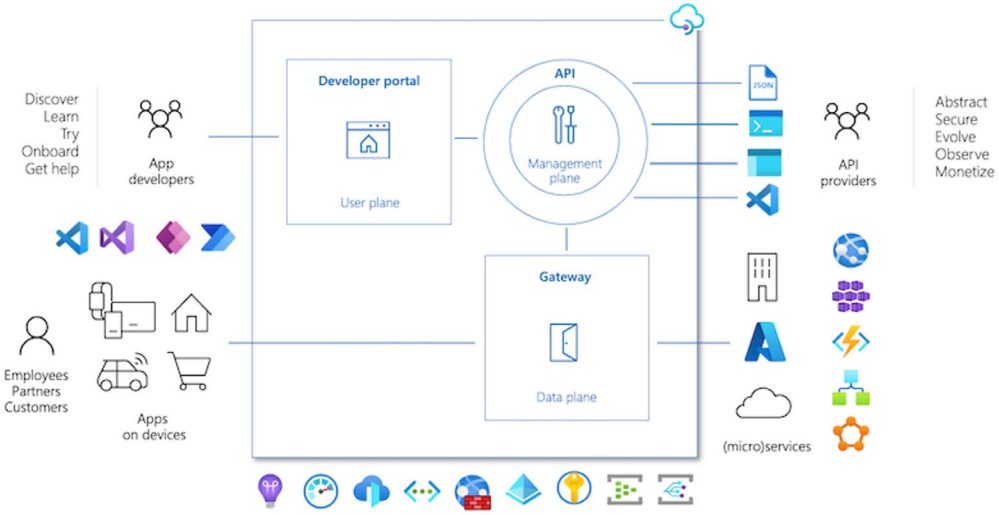
Adicional esta herramienta permite superar de forma más transparente los siguientes problemas:

- Exponer de forma segura los servicios.
- Proteger y supervisar las API.
- Facilitar la identificación y el uso de las API por parte de usuarios internos y externos.

1.10.2. Componentes

Azure API Management incluye dentro de sus componentes: una puerta de enlace de API, un plano de gestión y un portal para desarrolladores, todos los elementos mencionados se encuentran alojados en la nube Azure y administrados de manera predeterminada (Microsoft, 2023). En la figura 15 se puede encontrar la disposición de cada elemento.

Figura 15.
Elementos del ecosistema de Azure Microsoft Api



Nota. Tomado de Microsoft (2023).

Actividades de aprendizaje recomendadas

Continuemos con el aprendizaje mediante su participación en las actividades que se describen a continuación:

1. Siga los pasos detallados en la Guía a paso a paso para desplegar un api utilizando las herramientas de microsoft, y aplique el caso propuesto. De esta manera, obtendrá una experiencia práctica de cómo desplegar una API en un entorno *cloud*.

2. Una vez completado el estudio de la unidad 1, se aconseja llevar a cabo la siguiente autoevaluación con el objetivo de verificar el aprendizaje obtenido. Al concluir la guía, se proporcionan las respuestas a dichas autoevaluaciones; no obstante, se insta a revisarlas únicamente después de haber registrado sus propias soluciones, permitiéndole de esta manera evaluar adecuadamente el nivel de conocimientos adquiridos.

Autoevaluación 2

Lea detenidamente cada una de las preguntas y seleccione la alternativa según corresponda.

1. ¿Cuál es el principal objetivo de los estándares y protocolos de interoperabilidad empresarial?

    a. Facilitar la colaboración y comunicación entre diferentes sistemas, aplicaciones y dispositivos dentro de una organización o entre múltiples organizaciones.

    b. Asegurar la compatibilidad exclusivamente entre dispositivos de una misma organización.

    c. Evitar la adopción de tecnologías de vanguardia en las empresas.

2. ¿Cuál es el principal propósito del EDI (Electronic Data Interchange)?

    a. Facilitar la comunicación entre empleados dentro de una organización.

    b. Permitir el intercambio electrónico de documentos y datos comerciales estructurados entre organizaciones, agilizando y automatizando procesos como la emisión de órdenes de compra, facturación y seguimiento de envíos.

    c. Establecer protocolos de seguridad para proteger la información de las organizaciones.

3. ¿Cuál de las siguientes afirmaciones describe correctamente la estructura de un mensaje SOAP?

   a. SOAP utiliza una estructura basada en JSON para transmitir mensajes entre aplicaciones.

   b. SOAP es un protocolo de transporte que solo funciona con HTTP.

   c. SOAP es un protocolo basado en XML que consta de una envoltura (*envelope*), un encabezado (*header*) opcional y un cuerpo (*body*) que contiene la información del mensaje.

4. ¿Cuál es el principal objetivo de la arquitectura REST en el diseño de aplicaciones *web*?

   a. Crear aplicaciones *web* basadas en animaciones y gráficos interactivos.
   b. Diseñar aplicaciones *web* escalables y eficientes utilizando protocolos y principios de la *web*.
   c. Desarrollar aplicaciones *web* que solo funcionen en dispositivos móviles.

5. ¿Cuál de los siguientes códigos de respuesta HTTP es típicamente utilizado en servicios REST para indicar que una solicitud ha sido procesada con éxito?

   a. 200 OK.
   b. 404 Not Found.
   c. 500 Internal Server Error.

6. ¿Cuál es el principal objetivo de la integración de aplicaciones en un entorno empresarial?

   a. Consolidar y sincronizar información proveniente de diferentes fuentes de datos.

   b. Conectar aplicaciones y sistemas dispares dentro de una empresa para facilitar la comunicación y la gestión eficiente de la información.

   c. Coordinar y automatizar los procesos comerciales que involucran múltiples sistemas y aplicaciones.

7. ¿Qué tipo de integración empresarial permite a las organizaciones acceder a información coherente y actualizada en tiempo real a partir de diferentes fuentes de datos?

   a. Integración de aplicaciones.
   b. Integración de datos.
   c. Integración de procesos.

8. ¿Qué tipo de servicio ofrece Heroku para la construcción y despliegue de aplicaciones *web*?

   a. *Software* como S ervicio (SaaS).
   b. Infraestructura como S ervicio (IaaS).
   c. Plataforma como S ervicio (PaaS).

9. ¿En qué aspecto clave Heroku ayuda a los equipos de desarrollo al utilizar su plataforma?

   a. Facilitar la creación de documentación de proyectos.
   b. Implementar prácticas de Integración Continua y Despliegue Continuo (CI/CD).
   c. Ofrecer herramientas de diseño gráfico para aplicaciones *web*.

10. ¿Cuál es el propósito principal de Azure API Management en el entorno empresarial?

    a. Crear y administrar bases de datos en la nube.
    b. Proporcionar soluciones de almacenamiento en la nube.
    c. Gestionar el ciclo de vida integral de las API.

Ir a solucionario

Semana 8

Actividades finales del bimestre

¡Felicitaciones! Hemos concluido con éxito el primer bimestre.

Este tiempo debe utilizarlo en prepararse para las evaluaciones presenciales, para ello le recomiendo que realice una revisión de las actividades de aprendizaje de la unidad 1, en la cual se aborda la temática de fundamentos de la interoperabilidad empresarial. Asegúrese de comprender las temáticas tratadas, las actividades al final de cada unidad le ayudarán a consolidar lo aprendido.

Segundo bimestre

Resultado de aprendizaje 2

- Integra diferentes productos o sistemas computarizados para conectarse e intercambiar fácilmente información entre sí, ya sea en la implementación o el acceso, sin restricciones.

Durante las próximas 7 semanas se profundizará en temas clave como la gestión de una API, el ciclo de vida de una interfaz y el uso de la herramienta WSO2 para lograr un flujo constante y efectivo de información entre sistemas. Comprender el ciclo de vida de una API es esencial para asegurar la calidad y mantenimiento adecuado de las interfaces, lo que permitirá diseñar y gestionar interfaces que faciliten una comunicación eficiente entre sistemas y productos. La gestión nos permite establecer cómo los diferentes sistemas se comunican entre sí, mientras que la herramienta WSO2 proporciona una plataforma integral para administrar, diseñar y controlar la interoperabilidad entre sistemas computarizados, permitiendo el intercambio de información sin barreras.

Contenidos, recursos y actividades de aprendizaje

Semana 9

Estimado estudiante, esta semana damos inicio al estudio del segundo bimestre. Adicional en esta semana iniciamos el estudio de la unidad 2 de la asignatura de Interoperabilidad Empresarial, correspondiente a la interoperabilidad en los negocios digitales. Puntualmente, estaremos analizando los diferentes tipos de negocio digitales B2B y B2C.

Unidad 2. Interoperabilidad, datos y negocios digitales

Con esta segunda unidad usted estará preparándose para abordar de mejor forma los diversos desafíos del mundo profesional. Una comprensión sólida de los fundamentos de la gestión de una API, el ciclo de vida de

una interfaz y el uso de herramientas como WSO2 le permitirá desarrollar soluciones eficaces para los problemas técnicos y de comunicación que surgen en el ámbito laboral. Además, contar con un conocimiento profundo de estos temas puede ser una ventaja competitiva en el mercado laboral actual, ya que las empresas buscan constantemente profesionales que posean habilidades en tecnología y comunicación para optimizar sus procesos y mejorar su eficiencia.

2.1. Interoperabilidad en negocios digitales

En el ámbito del ecosistema digital, la capacidad de intercambio y procesamiento de datos e información útil entre distintos sistemas, aplicaciones o componentes es conocida como interoperabilidad. A pesar de ser un concepto esencial en una sociedad moderna altamente interconectada, muchas veces pasa inadvertido. Ejemplos de esto son la facilidad para realizar llamadas telefónicas internacionales sin preocuparse por estándares de señalización o cables transoceánicos, y el envío y recepción de correos electrónicos en diferentes dispositivos y navegadores. Esta habilidad de interacción es clave en el desarrollo del *Internet* de las cosas. Por ello, resulta fundamental establecer un entendimiento compartido sobre su funcionamiento, costos y beneficios potenciales, así como las distintas estrategias para impulsarla.

2.1.1. Tipos de negocios B2B

B2B, que significa "B*usiness-to-*B*usiness*" (o negocio a negocio en español), hace referencia a los modelos de negocio en los que dos empresas llevan a cabo transacciones de bienes o servicios. Aunque el término B2B se asocia comúnmente con el comercio al por mayor, también puede abarcar la provisión de servicios y el consumo de contenidos entre empresas. En el ámbito B2B, las organizaciones trabajan en conjunto, estableciendo relaciones comerciales que les permiten aprovechar las habilidades y recursos de cada una para lograr objetivos comunes. Esto implica un enfoque más colaborativo y estratégico en comparación con las transacciones tradicionales entre Empresas y Consumidores (B2C), donde las interacciones tienden a ser más directas y enfocadas en la venta al público en general (Sánchez, n.d.).

En el ámbito de negocios B2B, generalmente podemos encontrar que los proveedores ofrecen sus servicios de tres maneras distintas:

- ***Ventas exclusivas a empresas***: esto implica la oferta de productos o servicios únicamente a otras empresas, lo cual puede deberse a factores como la naturaleza del producto (por ejemplo, productos industriales) o a las cantidades involucradas en la venta (como en el caso de las ventas al por mayor). Estos procesos de venta se adaptan a las necesidades de los clientes empresariales, incluyendo métodos y plazos de pago específicos.

- ***Ventas indiferenciadas a empresas y clientes finales:*** este tipo de transacciones es más común en pequeños comercios que no diferencian entre productos, precios o condiciones, ya sea que los compradores sean empresas o consumidores individuales.

- **Ventas diferenciadas entre empresas y clientes**: algunos comercios ofrecen sus productos tanto a empresas como a particulares, pero con tiendas o áreas diferenciadas para cada segmento de clientes. En el caso de las ventas a empresas, es común mostrar los productos sin IVA, ofrecer mayores cantidades de producto e incluso aplicar descuentos basados en el volumen de compra. Un ejemplo de esto se encuentra en las tiendas de telefonía, donde los productos, ofertas y precios varían para clientes particulares y empresariales.

2.1.2. Tipos de negocios B2C

B2C, que significa *Business to Consumer* (o "de empresas a consumidores" en español), es un modelo de negocio en el que las empresas ofrecen productos y servicios al público en general, buscando obtener rentabilidad en cada transacción. Su objetivo principal es el consumidor final.

Las empresas B2C también deben ocuparse de aspectos logísticos, contables y de almacenamiento, los cuales suelen ser invisibles para los consumidores. Además, ofrecen servicios adicionales como atención al cliente y soporte posventa. Este último aspecto es crucial, ya que gran parte del éxito depende de la satisfacción del cliente tanto con el producto como con el servicio recibido.

Es evidente que este modelo de negocio es bastante antiguo y, probablemente, el más familiar para la mayoría de las personas, ya que

cualquier comercio, desde tiendas físicas hasta *e-commerce*, en el que se realicen compras de manera regular, puede considerarse un negocio B2C.

Autoevaluación 3

Una vez completadas la primera semana de estudio, se aconseja llevar a cabo una autoevaluación con el objetivo de verificar el aprendizaje obtenido. Al concluir la guía, se proporcionan las respuestas a dichas autoevaluaciones; no obstante, se insta a revisarlas únicamente después de haber registrado sus propias soluciones, permitiéndole de esta manera evaluar adecuadamente el nivel de conocimientos adquiridos.

Lea detenidamente cada una de las preguntas y seleccione la alternativa según corresponda

1. ¿Cuál es la diferencia entre los negocios B2B y B2C?

    a. En los negocios B2B, dos empresas llevan a cabo transacciones de bienes o servicios, mientras que en los negocios B2C las empresas ofrecen productos y servicios al público en general.

    b. En los negocios B2B, las transacciones son más directas y enfocadas en la venta al público en general, mientras que en los negocios B2C las empresas trabajan en conjunto para lograr objetivos comunes.

    c. En los negocios B2B, las transacciones son más colaborativas y estratégicas, mientras que en los negocios B2C las empresas trabajan en conjunto para lograr objetivos comunes.

2. ¿Cuál es la diferencia entre los negocios B2B y B2C?

    a. Ventas indiferenciadas a empresas y clientes finales.
    b. Ventas exclusivas a empresas.
    c. Ventas diferenciadas entre empresas y clientes.

3. ¿Qué es la interoperabilidad en el contexto de los negocios digitales?

   a. La capacidad de compartir información entre diferentes dispositivos.
   b. La capacidad de intercambio y procesamiento de datos e información útil entre distintos sistemas, aplicaciones o componentes.
   c. La habilidad para realizar llamadas telefónicas internacionales.

4. ¿Cuál es un ejemplo de interoperabilidad en la vida cotidiana?

   a. Comprar en una tienda física.
   b. Realizar llamadas telefónicas internacionales sin preocuparse por estándares de señalización o cables transoceánicos.
   c. Utilizar una aplicación de mensajería instantánea.

5. ¿Qué significa B2B?

   a. *Business-to-business*.
   b. *Business-to-consumer*.
   c. *Business-to-government*.

6. ¿En qué se diferencia el enfoque B2B del enfoque B2C?

   a. B2B es más colaborativo y estratégico, mientras que B2C es más directo y enfocado en la venta al público en general.
   b. B2B se enfoca en ventas al por mayor y B2C en ventas al por menor.
   c. B2B implica transacciones entre empresas y B2C entre empresas y consumidores.

7. ¿Cuál es un ejemplo de venta exclusiva a empresas en el ámbito B2B?

   a. Venta de productos industriales.
   b. Venta de productos electrónicos.
   c. Venta de alimentos.

8. ¿Qué tipo de transacciones B2B es más común en pequeños comercios?

   a. Ventas exclusivas a empresas.
   b. Ventas indiferenciadas a empresas y clientes finales.
   c. Ventas diferenciadas entre empresas y clientes.

9. ¿Qué significa B2C?

   a. *Business-to-consumer*.
   b. *Business-to-business*.
   c. *Business-to-government*.

10. ¿Cuál es un aspecto crucial en el modelo de negocio B2C?

   a. La atención al cliente y soporte posventa.
   b. La capacidad de almacenamiento.
   c. La rapidez en las transacciones.

Ir a solucionario

Estimado estudiante, esta semana damos continuidad a la unidad 2 de la asignatura de Interoperabilidad Empresarial. Puntualmente, esta semana revisaremos el proceso de gestión de API y cuáles son las principales fases dentro del ciclo de vida.

2.2. Gestión de interfaces de programación de aplicaciones

Recapitulando, una API es una capacidad comercial que se ofrece a través de *Internet* para usuarios internos y externos. Se caracterizan por:

- Ser accesibles a través de la red y utilizar protocolos *web* estándar. (HTTP, GRPC, WebSocket, etc.).
- Tener interfaces bien definidas.
- Si está permitido, ser accesibles para entidades de terceros.

¿Por qué son necesarias?

En un ejemplo práctico, analicemos el siguiente escenario. Si consideramos el departamento de evaluaciones de la UTPL, podríamos obtener información acerca del número de alumnos que han rendido las evaluaciones en un ciclo académico o en un centro en particular. Si los datos están en un archivo, cuando solicitamos los datos, obtenemos un archivo grande con todos los datos. Pero, ¿qué pasa si necesitamos un conjunto específico de datos, como únicamente los alumnos del Centro Loja que han completado las evaluaciones del último semestre? O tenemos que extraerlos del archivo, o el departamento debe proporcionar un nuevo archivo con los datos. Si el Departamento de Evaluaciones tuviera una API para esto, podríamos enviar fácilmente la solicitud y obtener el conjunto de datos requerido.

En una empresa u organización, durante la etapa inicial de implementación de Interfaces de Programación de Aplicaciones (API), podría haber un número limitado de proveedores de API utilizados por los consumidores. A medida que el negocio se expande, tanto la cantidad de proveedores de API como el número de consumidores que hacen uso de ellas experimentan un crecimiento.

Como resultado, se vuelve muy difícil administrar todas estas interfaces debido a:

1. Uso de diferentes protocolos.
2. Uso de diferentes mecanismos de autenticación.
3. Falta de documentación.
4. Código heredado.

¿Por qué es importante la gestión?

La gestión adecuada es de suma importancia para garantizar la eficiencia, seguridad y escalabilidad de las aplicaciones y servicios que se basan en ella. Al realizar una gestión efectiva del API, los desarrolladores pueden optimizar el rendimiento, monitorear su uso, establecer políticas de acceso y garantizar la compatibilidad con futuras actualizaciones. Además, una gestión apropiada permite identificar y solucionar problemas rápidamente, así como prevenir el mal uso o ataques que puedan comprometer la integridad de los datos y la funcionalidad de las aplicaciones. En última instancia, una gestión de API bien ejecutada es esencial para mejorar la experiencia del usuario final y asegurar el éxito a largo plazo de cualquier proyecto digital. En resumen, una gestión centralizada nos permite:

- Establecer confianza y seguridad.
- Regular el uso compartido y los permisos.
- Monetizar el uso.

El desarrollo de interfaces de programación de aplicaciones generalmente lo realiza alguien que comprende los aspectos técnicos, las interfaces, la documentación, las versiones, etc., mientras que la administración generalmente la lleva a cabo alguien que comprende los aspectos comerciales de las interfaces.

La gestión introduce el concepto de autoservicio bajo demanda en el ámbito de integración y Arquitectura Orientada a Servicios (SOA). De esta manera, los desarrolladores pueden localizar fácilmente las API pertinentes, explorar sus funcionalidades, probarlas en línea, suscribirse, evaluarlas, generar claves de acceso e interactuar con los responsables de las mismas. Estos últimos, por su parte, pueden provisionar sus interfaces, compartir documentación, administrar claves de acceso y recopilar comentarios acerca del desempeño, calidad y utilización de la interfaz en cuestión.

Adicional, exponer los procesos, datos y servicios centrales como públicos nos permitirán:

- A otros mezclar sus API de formas innovadoras para crear nuevas soluciones.
- Aumentar su potencial de crecimiento y avances de asociación.
- Reducir la duplicación y el retrabajo.
- Aumentar el desarrollo colaborativo.

2.2.1. Ciclo de vida

El ciclo de vida de una API es un proceso integral que abarca desde la concepción hasta la retirada, pasando por múltiples etapas clave para garantizar un desarrollo sostenible y una implementación exitosa. En este proceso, debemos considerar aspectos esenciales como el diseño, la creación, la publicación, el mantenimiento y la evolución. Un enfoque estructurado y bien planificado en cada etapa permite a los desarrolladores crear API robustas y escalables que se adapten a las necesidades cambiantes del mercado y los usuarios. Además, al comprender el ciclo de vida de una API, se facilita la integración con otros sistemas y servicios, lo que resulta en una mayor interoperabilidad y colaboración entre aplicaciones y organizaciones en el ecosistema digital (Lane, 2022).

A continuación lo invito a revisar las fases principales que involucran el ciclo de vida de una API en la siguiente infografía.

Las fases principales que involucran el ciclo de vida de una API.

Semana 11

Estimado estudiante, al continuar con el estudio de la segunda unidad en la materia de Interoperabilidad Empresarial, esta semana analizaremos aspectos cruciales al diseñar nuestra API. Además, exploraremos diferentes enfoques para gestionar el control de versiones en nuestra API.

2.2.2. Diseño de una API: versionamiento compatibilidad

El diseño de una API es una de las tareas más importantes para el desarrollo de aplicaciones. La seguridad, el versionamiento y la compatibilidad son tres componentes clave para garantizar el éxito de la

API. La seguridad asegura que los usuarios no puedan acceder a los datos no autorizados, el versionamiento ayuda a los desarrolladores a identificar y mantener la compatibilidad de la API entre versiones y la compatibilidad garantiza que la API se pueda usar con otros sistemas.

Acorde, como menciona (Jin et al., 2021), las API deben estar alineadas con el negocio principal para maximizar los resultados. Ejemplos de empresas de *Software* como Servicio (SaaS) que tienen interfaces son GitHub, Salesforce y Stripe. Estas API permiten la construcción de productos conocidos como "integraciones de servicio" y funcionan mejor con contenido generado por el usuario. Sin embargo, hay una clara razón para no crear una plataforma para desarrolladores si la estrategia de API no está alineada con el negocio principal.

Una interfaz correctamente definida debe estar determinado por el problema a resolver y su valor. Si se trata de obtener una base de datos única o una funcionalidad compleja, podemos caer en el uso una API confuso, inconsistente y mal documentado. Por otro lado, una API correctamente definida ofrece claridad, flexibilidad, poder, capacidad de modificación y documentación.

2.2.3. Estándares

A continuación, repasaremos algunos puntos clave al momento de diseñar nuestras interfaces.

- ***Definir la API con respecto a recursos***: en este punto debemos poner atención a las entidades comerciales que vamos a exponer en API. Por ejemplo, en un sistema de biblioteca, las entidades principales podrían ser los estudiantes y los préstamos. La creación de un préstamo se puede lograr realizando una solicitud **HTTP POST** que contenga la información del préstamo. La respuesta **HTTP** indica si el pedido se realizó con éxito o no. Cuando sea posible, los URI de recursos deben basarse en sustantivos (el recurso) y no en verbos (Microsoft, 2023). En la tabla 3 podemos revisar dos ejemplos de URI definidas.

Tabla 3.
Ejemplos de una correcta nomenclatura para definir un recurso API

URI	Descripción
https://biblioteca-utpl.edu.ec/prestamos	OK
https://biblioteca-utpl.edu.ec/crear-prestamos	Evitar este tipo de nomenclatura

Nota. Quiñonez, F. 2023.

Adicional, otro factor que se debe tomar en cuenta, es evitar que una API represente nuestra estructura interna de la base de datos. Adicional las entidades a menudo se agrupan en colecciones (estudiantes, préstamos). En general, es útil utilizar sustantivos en plural para los URI que hacen referencia a colecciones. Es una buena práctica organizar los URI de colecciones y elementos en una jerarquía. Por ejemplo, */estudiantes* es la ruta a la colección de estudiantes y */estudiantes/4* es la ruta al estudiante con ID igual a 4. Este enfoque nos ayuda a mantener la API *web* intuitiva.

- ***Definir las operaciones API en términos de los métodos HTTP***

El protocolo HTTP nos define distintos métodos que asignan un significativo semántico a una solicitud. Los métodos HTTP que comúnmente son utilizados por la mayoría de la API *web* RESTFul son los que se exponen en la tabla 4.

Tabla 4.
Métodos HTTP utilizados para definir un API

Método	Descripción
GET	Obtiene una representación del recurso en el URI especificado. El cuerpo del mensaje de respuesta contiene los detalles del recurso solicitado.
POST	Crea un nuevo recurso en el URI especificado. El cuerpo del mensaje de la solicitud proporciona los detalles del nuevo recurso. También podemos utilizar este método para activar operaciones que no creen recursos.
PUT	Crear o reemplazar un recurso en el URI especificado. El cuerpo del mensaje de solicitud nos indica si el recurso se debe crear o actualizar.
PATCH	Realiza una actualización parcial de un recurso. El cuerpo de la solicitud especifica el conjunto de cambios que se aplicarán al recurso.
DELETE	Remueve el recurso especificado en el URI.

Nota. Quiñonez, F. 2023.

En la tabla 5 se presenta un resumen de las convenciones comunes adoptadas por la mayoría de las implementaciones RESTFul utilizando el ejemplo de la biblioteca.

Tabla 5.
Resumen de verbos utilizados en el ejemplo de préstamo de libros de una biblioteca

Recurso	POST	GET	PUT	DELETE
/estudiantes	Crea un nuevo estudiante	Obtiene todos los estudiantes	Actualización masiva de estudiantes	Remueve todos los estudiantes
/estudiantes/1		Obtiene el detalle del estudiante 1	Actualiza el detalle del estudiante 1 en caso de existir	Elimina el estudiante 1
/estudiantes/1/prestamos	Crea un nuevo préstamo para el estudiante 1	Obtiene todos los préstamos para el estudiante 1	Actualiza masivamente los préstamos del estudiante 1	Elimina todos los préstamos del estudiante 1

Nota. Quiñonez, F. 2023.

- ***Cumplir la semántica HTTP: a*** continuación, revisaremos algunas consideraciones ajustadas a la especificación HTTP.

  - Media *Types*

    Como se ha comentado, los clientes y servidores intercambian representaciones de recursos. Por ejemplo, en una solicitud POST, el cuerpo de la solicitud contiene una representación del recurso a crear. En una solicitud GET, el cuerpo de la respuesta contiene una representación del recurso obtenido.

    En el protocolo HTTP, los formatos se especifican mediante el uso de tipos de medios, también llamados tipos MIME. Para datos no binarios, la mayoría de las API *web* admiten JSON (tipo de medio = *application*/json) y posiblemente XML (tipo de medio = *application*/xml).

    A continuación, un ejemplo de solicitud POST que incluye datos en formato JSON:

```
POST https://biblioteca-utpl.edu.ec/prestamos HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 57

{"Id":1,"Name":"Felipe","Book":"Interoperabilidad","TimeDays"
:4}
```

En caso de que el servidor no soporte el medio, podría retornar un estado HTTP con el código 415 (*Unsupported Media Type*).

- Métodos *GET*

  Una petición correcta a un método GET normalmente retorna un estado HTTP 200. Si el recurso no es encontrado, devuelve una respuesta HTTP 204. Otro caso que podemos encontrar es en los métodos de búsqueda, en los cuales se puede devolver 204 (No *Content*) para los escenarios en los cuales la consulta no encuentre ningún recurso.

- Métodos *POST*

  En caso de que el método cree un recurso correctamente, la respuesta es 201 (*Created*). El cuerpo de la respuesta contiene una representación del recurso creado. En caso de que la petición no cumple con la especificación podemos recibir una respuesta HTTP 400 (*Bad Request*).

- Métodos *DELETE*

  En caso de que la operación sea correcta, recibiremos por lo general el código de respuesta HTTP 204 (No *Content*), indicando que el proceso se realizó con normalidad. En caso de que intentemos borrar un recurso que no se encuentre recibiremos el código HTTP 404.

2.2.4. Versionamiento

No es casual encontrar una API que permanezca sin sufrir cambios cuando esté productivo. Esto se debe que los requisitos empresariales cambian, podemos tener escenarios que requieran agregar nuevas colecciones de recursos, las relaciones entre recursos pueden subir modificación y escenarios donde la estructura de un recurso puede sufrir cambios. Los puntos anteriormente descritos pueden tener un grado de dificultad, pero

lo que en realidad genera mayor dificultad es pensar en los efectos que dichos pueden tener en las aplicaciones cliente que consumen la API (Microsoft, 2023).

El versionamiento nos permite que una API indique las funciones y recursos que ofrece, para que los clientes puedan en una petición especificar a qué versión están dirigidas las solicitudes.

A continuación, revisaremos 3 formas comunes para especificar la versión en las solicitudes de los clientes:

- ***Versionamiento por URI:*** **c**ada vez que existen cambios en la definición de un recurso, se debe agregar un número de versión en la URI de un recurso. Las versiones anteriores deben seguir funcionando acorde las definiciones anteriores. Por lo general este número de versión se agrega con la letra "v", un ejemplo es el que se presenta a continuación.

```
https://biblioteca-utpl.edu.ec/v2/estudiantes/2
```

- ***Versionamiento en la cadena de consulta: a*** diferencia del enfoque anterior, podemos definir la versión del recurso que necesitamos utilizando un parámetro en la cadena de consulta. Ejemplo:

```
https://biblioteca-utpl.edu.ec/estudiantes/2?version=2
```

- ***Versionamiento utilizando la cabecera:*** este tipo de versionamiento se logra utilizando un nombre de cabecera personalizado dentro de la petición a un recurso. Ejemplo:

```
POST https://biblioteca-utpl.edu.ec/prestamos HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: 57
Custom-Header: api-version=1

{"Id":1,"Name":"Felipe","Book":"Interoperabilidad","TimeDays":4}
```

Autoevaluación 4

Una vez completadas las primeras 3 semanas de estudio del segundo bimestre, se aconseja llevar a cabo una autoevaluación con el objetivo de verificar el aprendizaje obtenido. Al concluir la guía, se proporcionan las respuestas a dichas autoevaluaciones; no obstante, se insta a revisarlas únicamente después de haber registrado sus propias soluciones, permitiéndole de esta manera evaluar adecuadamente el nivel de conocimientos adquiridos.

Lea detenidamente cada una de las preguntas y seleccione la alternativa según corresponda.

1. ¿Cuál es la función principal de los tipos de medios (media *types*) en el protocolo HTTP?

    a. Indicar el tamaño del contenido en una solicitud HTTP.
    b. Especificar el formato de los datos intercambiados entre cliente y servidor.
    c. Establecer el tiempo de vida del contenido en caché.

2. ¿Cuál es el propósito del versionamiento en una API *web*?

    a. Facilitar la depuración de errores en el código de la API.
    b. Mejorar el rendimiento de la API al reducir la cantidad de datos transferidos.
    c. Permitir que las aplicaciones cliente especifiquen la versión de funciones y recursos que desean utilizar.

3. ¿Cuáles son las características que definen a las API?

    a. Son accesibles solo a través de la red y utilizan protocolos estándar.
    b. Tienen interfaces bien definidas.
    c. Ambas opciones anteriores son correctas.

4. ¿Cuál de las siguientes no es una razón por la que la gestión de API es importante?

   a. Establecer confianza y seguridad.
   b. Reducir el tiempo de desarrollo de aplicaciones.
   c. Ambas opciones anteriores son correctas.

5. ¿Qué se busca al exponer los procesos, datos y servicios centrales como API al público?

   a. Disminuir la colaboración entre desarrolladores.
   b. Aumentar el potencial de crecimiento y avances de asociación.
   c. Reducir la interoperabilidad entre aplicaciones y organizaciones.

6. ¿Qué etapa del ciclo de vida de una API se centra en garantizar que las operaciones relacionadas con una API estén correctamente definidas?

   a. Diseñar.
   b. Definir.
   c. Seguridad.

7. La especificación de OpenAPI proporciona un vocabulario común para describir:

   a. Los métodos de monetización de una API.
   b. La superficie de las solicitudes y respuestas de las API.
   c. Los aspectos comerciales de las API.

8. ¿Qué es un servidor simulado (*mock server*)?

   a. Un servidor que almacena datos sensibles de una API.
   b. Un servidor que replica la funcionalidad de una API antes de escribir código.
   c. Un servidor que se utiliza exclusivamente para el almacenamiento de archivos.

9. ¿Cuál de las siguientes etapas del ciclo de vida de una API se enfoca en establecer un proceso y enfoque formal para diseñar una API?

a. Diseñar.
b. Desarrollo y documentación.
c. Monitorear.

10. ¿Cuál de las siguientes etapas del ciclo de vida de una API se asegura de que exista un enfoque consistente para la gestión de identidad y acceso para cada API?

a. *Deploy*.
b. Seguridad.
c. Monitorear.

Ir a solucionario

Estimado estudiante, esta semana seguimos avanzando en la unidad 2 de la asignatura de Interoperabilidad Empresarial. A lo largo de la semana, estudiaremos la herramienta de gestión de API WSO2 y analizaremos los componentes principales que conforman la arquitectura de WSO2.

2.3. WSO2 Gestión de Api

WSO2 API Manager es una solución completamente de código abierto para la gestión de API de extremo a extremo en la nube, en el local o en entornos híbridos. Viene con una Licencia de *Software* Apache Versión 2.0 que lo hace gratuito para usar. Permite a los desarrolladores de API diseñar, publicar y gestionar el ciclo de vida de las API y al gestor de productos de API crear productos de API a partir de una o más API. Alberga un portal de desarrolladores de aplicaciones que ayuda a construir y gestionar una comunidad de desarrolladores para sus API. Su *gateway* de API nativo en la nube se usa para asegurar, enrutar, controlar y monitorizar el tráfico de sus API de manera escalable (WSO2, 2023).

Es momento de ver el siguiente video sobre la plataforma WSO2, el cual proporciona una visión general comprehensiva de esta valiosa herramienta.

Como podrá haber revisado en el video anterior, en los últimos años se ha experimentado un crecimiento exponencial en la adopción de microservicios a nivel empresarial. Por ello, resulta esencial contar con una herramienta eficiente de administración. WSO2 es una herramienta de administración de servicios *web* que permite gestionar el ciclo de vida de los mismos, proporcionando una administración óptima de los microservicios.

WSO2 API Manager es una plataforma de gestión de ciclo de vida completo para API. Nos ofrece una suite de herramientas para:

- Crear y publicar API.
- Publicitar API en la tienda.
- Versiones API.
- Administrar sus ciclos de vida.
- Monetizar el uso de API.
- Monitorear y analizar el uso de API.
- Implementar gobernanza y seguridad.
- Facilitar la participación de la comunidad y proporcionar puntos de extensión.

2.3.1. Arquitectura de WSO2

WSO2 API *Manager* está compuesto por un *Gateway* API y un espacio de colaboración donde los publicadores, API se reúnen con los consumidores API. Este espacio de colaboración consiste en una API *Devportal* y una API *Publisher*. A continuación, se da una breve descripción de cada uno de los componentes:

- **API *Gateways***: permite asegurar, proteger, administrar y escalar llamadas API.

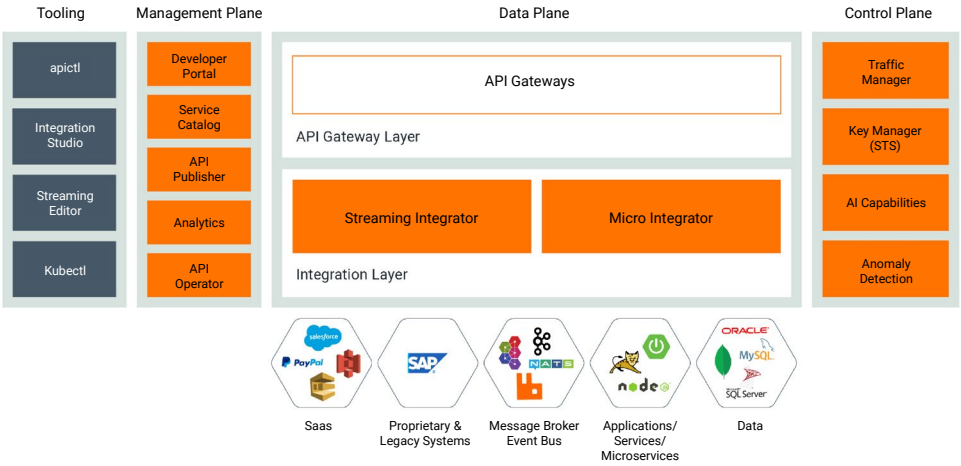
- **API *Publisher***: permite a los proveedores API publicar fácilmente sus API, compartir documentación, proporcionar claves API y recopilar comentarios sobre las características, la calidad y el uso de una API.

- **API *Developer* Portal**: proporciona un espacio para que los consumidores descubran la funcionalidad de la API, se suscriban a ellas, las evalúen y se comuniquen con los publicadores, API.

- ***Key Manager***: administra todas las operaciones relacionadas con los clientes, la seguridad y los *tokens* de acceso.

- ***Traffic Manager***: El Administrador de tráfico ayuda a los usuarios a regular el tráfico API, hacer que las API y las aplicaciones estén disponibles para los consumidores a diferentes niveles de servicio y proteger las API contra ataques de seguridad. El Administrador de tráfico cuenta con un motor de límites dinámicos de tasa para procesar políticas de límite de tasa en tiempo real, incluido el límite de tasa de solicitudes API.

Figura 16.
Arquitectura WSO2, principales componentes



Nota. Tomado de arquitectura de la plataforma WSO2 [Ilustración], por WSO2 2023, wso2 (https://apim.docs.wso2.com/en/4.2.0/assets/img/get_started/architecture/apim-architecture.png).

En la figura 16, podemos revisar la arquitectura de la plataforma WSO2. Adicional, el frontal del administrador de API consta de dos componentes principales.

- **API *Publisher*:** para crear y publicar API.
- ***Developer* Portal:** para que los consumidores de API descubran, se suscriban y prueben las API publicadas.

2.3.2. Configuración básica de WSO2

El archivo principal de configuración es:

<APIM_HOME>/conf/apim/repository/conf/deployment.toml

Todas las configuraciones se realizan a través de este archivo (nombre del *host*, configuraciones de bases de datos, configuración de almacenamiento de usuarios, configuración de almacenamiento de claves, entre otras). Un ejemplo de esta configuración se puede encontrar en la figura 17:

Figura 17.
Configuración básica de WSO2

```
[server]
hostname = "localhost"
#offset=0
base_path = "S$(carbon.protocol)://$fcarbon.host):$(carbon.management .port)"
#discard_empty_caches = false
server_role = "default"

[super admin]
username = "admin"
password = "admin"
create_admin_account = true

[user_store]
type = "database_unique_id"

[database.apim_db]
type = "mysql"
url = "jdbc:mysql://mysql:3306/WSO2AM_DB?autoReconnect=true&amp;useSSL=false"
username = "wso2carbon"
```

Nota. Quiñonez, F. 2023.

Estimados estudiantes, todas las posibles configuraciones las podemos encontrar en el sitio oficial de WSO2, en la sección de parametrización. En esta documentación se describe como ajustar para la parte del servidor, la sección de administración, la parte de la base de datos, la sección de JWT para la parte de autentificación, entre otras.

Actividades de aprendizaje recomendadas

- Instale localmente la herramienta WSO2 para navegar a través de los diferentes componentes de la misma. Para esto, siga los pasos detallados en la Guía oficial de WSO2 . De esta manera tendrá experiencia de despliegue del Api *Manager* y su configuración inicial. Puntos claves a tener en esta actividad:
    - Equipo con sistema operativo compatible (Windows, MacOS o Linux).
    - Acceso a *Internet* para descargar el *software*.
    - Adicional podría realizar esta instalación utilizando contenedores de *Docker*.

Estimado estudiante, continuamos avanzando en la unidad 2 de la asignatura de Interoperabilidad Empresarial. Durante esta semana, exploraremos cómo implementar mecanismos de seguridad en una API y, además, analizaremos el proceso de gestión de autenticación.

2.3.3. Implementación de seguridad para servicios *web*

La autenticación de una API es una forma para proteger el acceso a la API de accesos no identificados o anónimos. Lo anterior garantiza que la API esté segura y sea accesible para los consumidores que demuestren su identidad. WSO2 ofrece los siguientes mecanismos de autentificación para asegurar nuestra API:

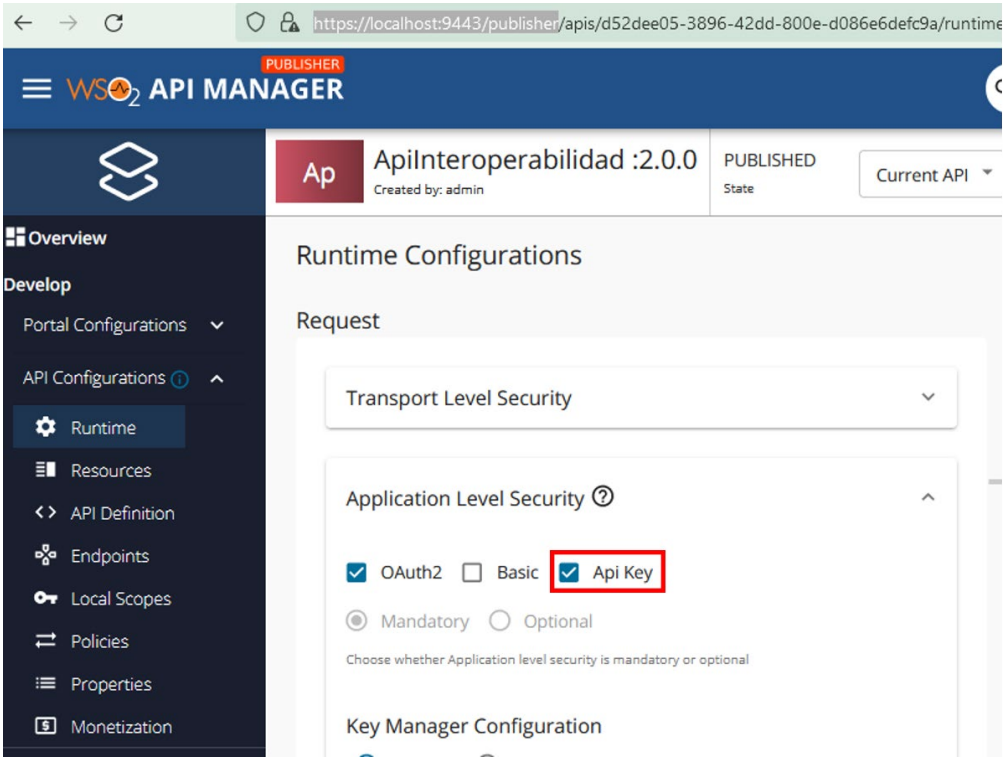
- OAuth2 Acces Tokens: es un estándar de facto el acceso a nuestra API REST. Todas las aplicaciones cliente que invoquen una API protegida por OAuth2 deben tener una suscripción válida a la API solicitada, y presentar un *token* de acceso al momento de invocarla. El encabezado de autorización HTTP es el método más habitual para proporcionar información de autenticación en las API REST. El *token* de acceso (seguido por *Bearer*) debe enviarse a través del encabezado de autorización para que la aplicación cliente autentique la API a la que se accede. El formato del encabezado es el que se muestra a continuación:

```
Authorization: Bearer NtBQkXoKElu0H1a1fQ0DWfo6IX4a
```

- API Keys: la API *Key* es la forma más sencilla de agregar seguridad en las aplicaciones. Podemos obtener una clave API para una aplicación cliente, el portal *Developer* de WSO2, esto utilizando la interfaz gráfica o podemos realizarlo utilizando la API propia de WSO2. Luego que se genere esta llave, las aplicaciones cliente podrán utilizarla en las diferentes peticiones. WSO2, cuando recibe una invocación a una API especifico, realiza dos validaciones básicas:
  - Validación de firma.
  - Validación de suscripción.

Para activar este tipo de mecanismo debemos ingresar al portal *Publisher* de WSO2 (*Portal Publisher*), y marcar la casilla API *Key*.

Figura 18.
Activar el mecanismo API Key para autentificación del API



Nota. Quiñonez, F. 2023.

- SSL mutuo: a diferencia de la autenticación SSL unidireccional habitual, en la que en cliente verifica la identidad del servidor, en este caso el SSL mutuo, el servidor valida la identidad del cliente, resultando una confianza entre ambas partes. Este sistema ofrece una seguridad muy sólida y evita que se hagan solicitudes al cliente para proporcionar su nombre de usuario y contraseña, tomando como referencia que el servidor siempre conozca los certificados que pertenecen al cliente.

- Autenticación básica: este es un esquema simple de autenticación, el cual dentro la petición contiene una cabecera de válida codificada en base 64 con el usuario y contraseña. Siempre es recomendable utilizar los mecanismos anteriormente descritos.

Actividades de aprendizaje recomendadas

- Revise la página de configuración API *Key* de WSO2, y siga los pasos detallados para configurar este método de autentificación, con lo anterior tendrá un acercamiento real de cómo agregar una capa de seguridad a nuestra API.

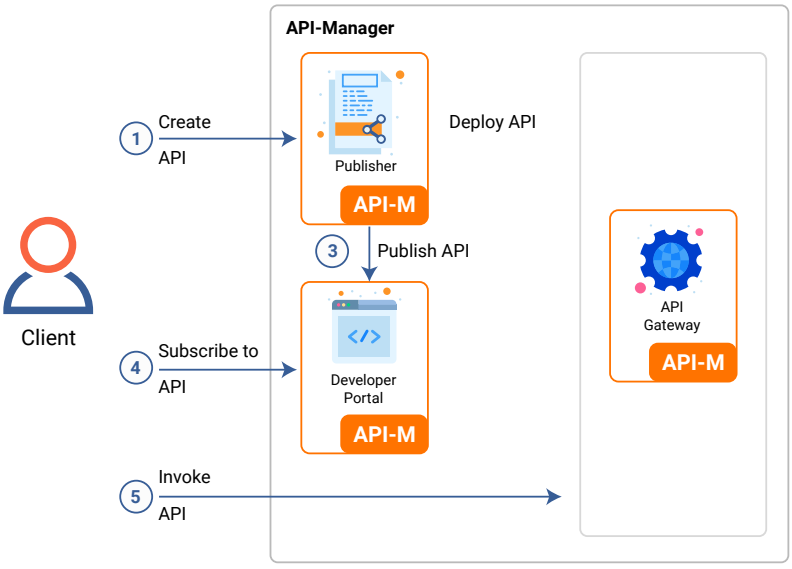
Semana 14

Estimado estudiante, esta semana seguimos avanzando en la unidad 2 de la asignatura de Interoperabilidad Empresarial. A lo largo de la semana, estudiaremos como publicar nuestra API utilizando la herramienta WSO2.

2.3.4. Publicación de API con WSO2

La herramienta WSO2 Api *Manager*, nos permite gestionar la publicación de un servicio *web* utilizando el módulo *Publisher*.

Figura 19.
Flujo para publicar un api utilizando WSO2



Nota. Tomado de una guía rápida de la plataforma WSO2 [Ilustración], por WSO2 2023, wso2 (https://apim.docs.wso2.com/en/4.2.0/assets/img/get_started/apim-qsg-diagram.png).

En la figura 19, podemos observar el flujo de publicaciones de un servicio *web*, dentro de los pasos tenemos:

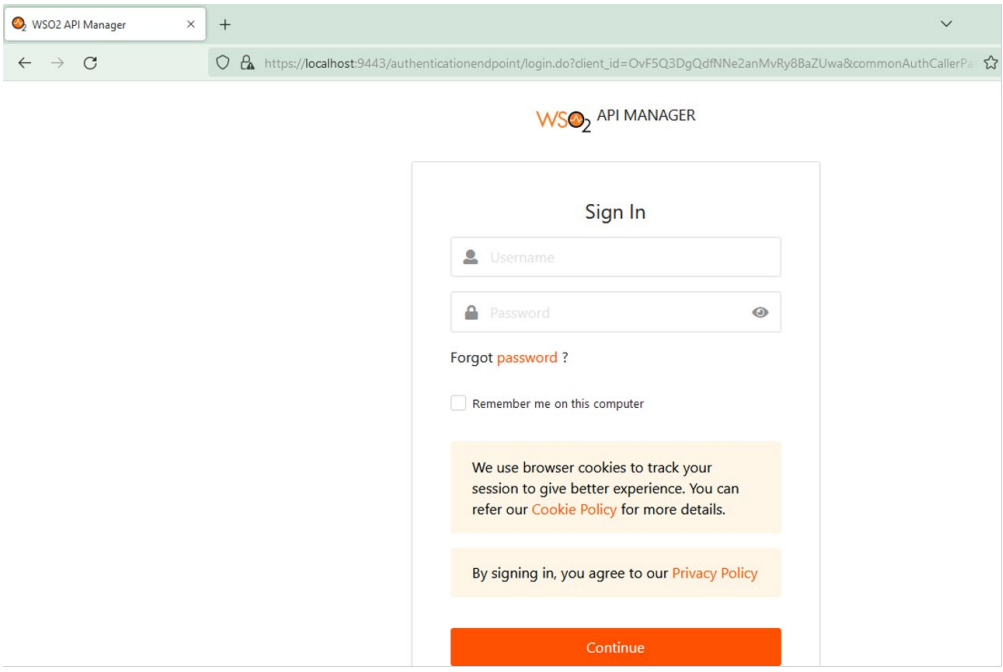
1. Crear y publicar una API utilizando el portal ***Publisher*.**
2. Desplegar la API en un ambiente ***Gateway***.
3. Publicar la API.
4. Subscribirse a la API desde el portal ***Developer*** y generar llaves para el acceso del mismo.
5. Invocar la API con las llaves generadas en el paso 4.

Para afianzar el flujo descrito vamos a realizar una publicación de una nueva API, para ello vamos a seguir los siguientes pasos:

1. Ingresa al portal utilizando de publicación.
2. Accede con las credenciales por defecto (figura 20):

    - Usuario: admin.
    - Clave: admin.

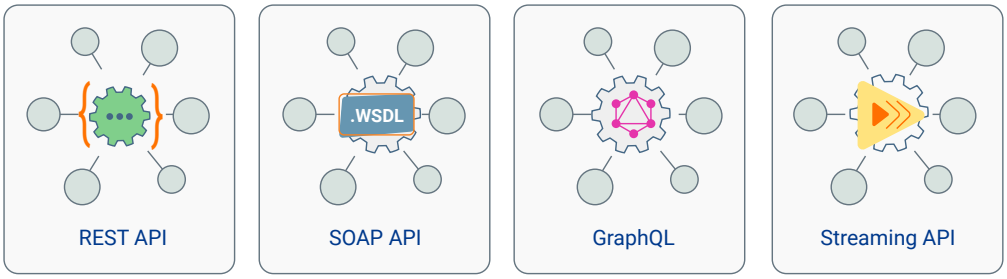
Figura 20.
Página de autentificación del portal Publisher



Nota. Quiñonez, F. 2023.

3. Acorde a la figura 21, seleccionamos la opción ***REST* API:**

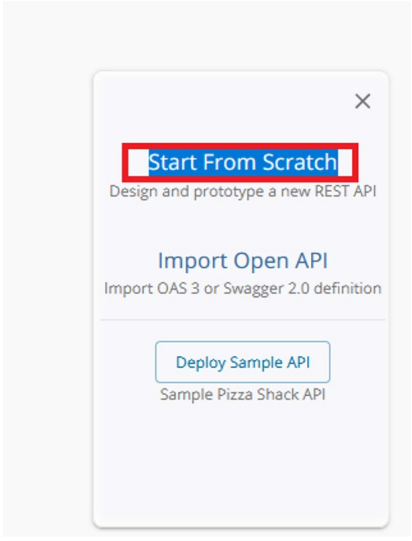
Figura 21.
Tipos api disponibles para crear dentro de la herramienta Publisher



Nota. Quiñonez, F. 2023.

4. Ahora seleccionamos iniciar desde *scratch*, acorde a la figura 22:

Figura 22.
Opción iniciar desde scratch

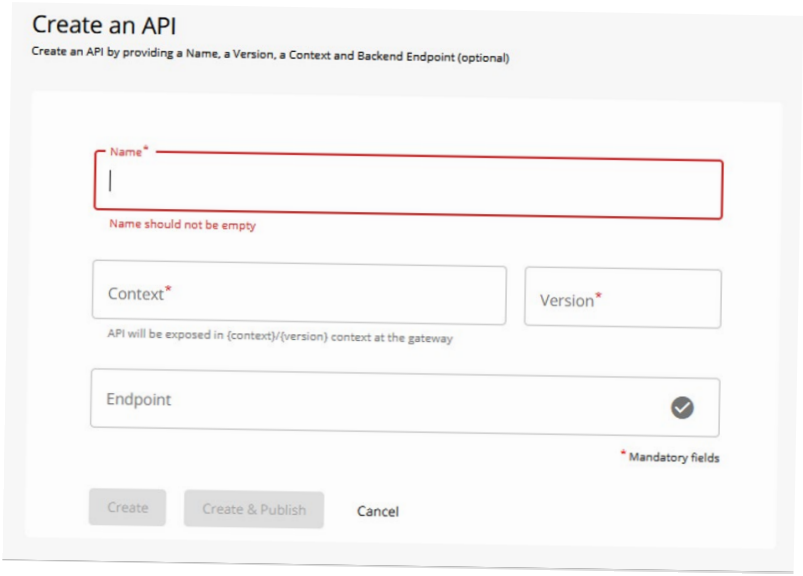


Nota. Quiñonez, F. 2023.

5. Ingrese los siguientes detalles, y damos click **Crear y Publicar** (Figura 23).

- Nombre: ApiInteroperabilidad
- Contexto: /apitest
- Versión: 1.0.0
- Endpoint: https://petstore3.swagger.io/api/v3/openapi.json

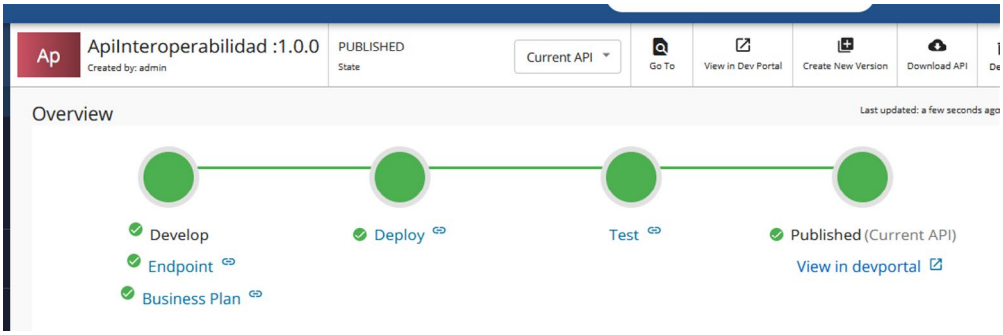
Figura 23.
Formulario para creación de un API



Nota. Quiñonez, F. 2023.

Esto publicará nuestro primer API en el Portal de desarrolladores, así como desplegarla en la Puerta de Enlace de API. Ahora tenemos una API REST segura con OAuth 2.0 lista para ser consumida. En la figura 24 se puede revisar el estado de nuestro API publicado.

Figura 24.
Estado del api publicado



Nota. Quiñonez, F. 2023.

Ahora, continuando con nuestra publicación, vamos a subscribirnos al API publicado anteriormente:

1. Ingresar al portal de desarrolladores (Figura 25):

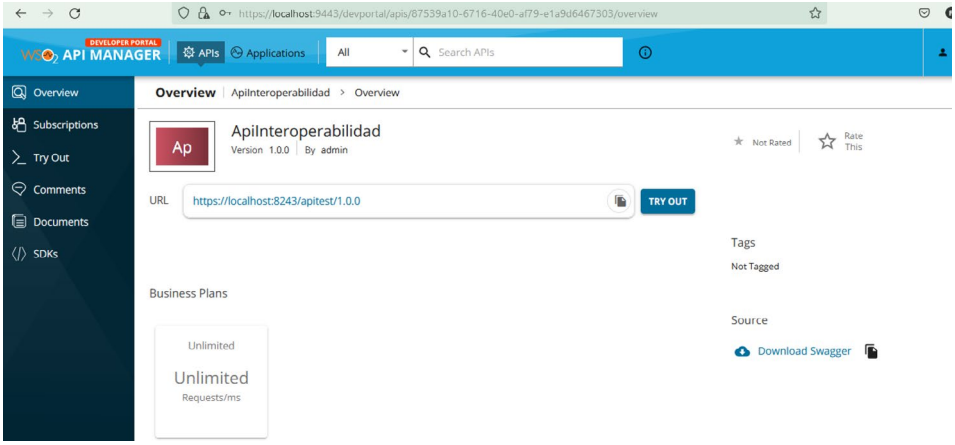
Figura 25.
Portal de desarrollo de la interfaz

Nota. Quiñonez, F. 2023.

2. Hacemos clic en **Iniciar sesión** e ingresamos con las credenciales admin/admin como sus credenciales para iniciar sesión en el Portal de desarrolladores.

3. Damos clic en el **ApiInteroperabilidad**, para ver detalles del mismo (Figura 26).

Figura 26.
Detalle de un api en el portal de desarrollo

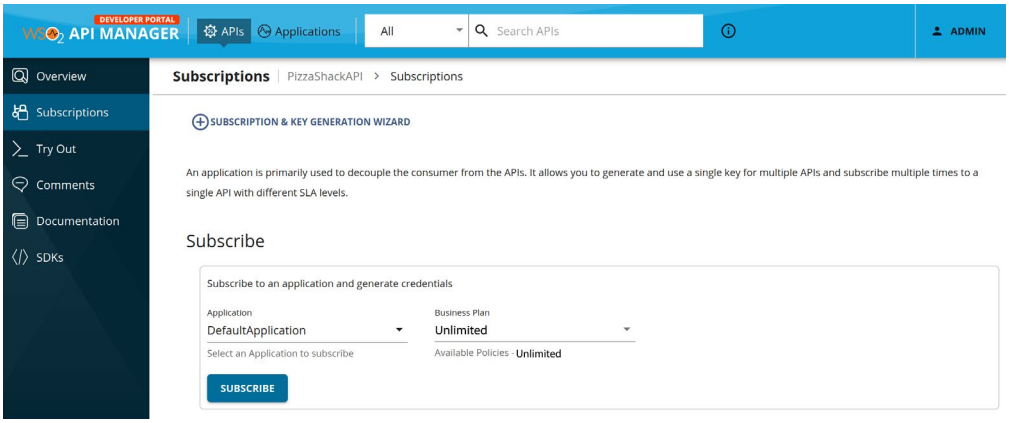


Nota. Quiñonez, F. 2023.

4. Damos clic en el menú derecho en la opción subscripciones (Figura 27).

Figura 27.

Sección de subscripción en el portal de desarrollo



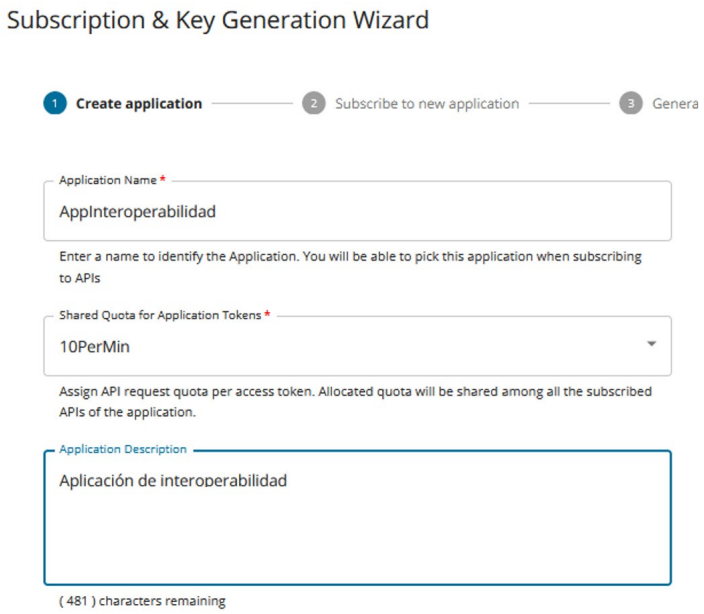
Nota. Quiñonez, F. 2023.

5. Clic en la sección **SUBSCRIPTION & KEY GENERATION WIZARD**, en la figura anterior.

6. El asistente le guiará a través de 5 pasos para registrar nuestra aplicación OAuth 2.0 que utilizará para consumir la API creada en la sección anterior. Ingresamos un nombre de nuestra aplicación, y damos clic en el botón siguiente, el resto de campos los dejamos con los valores por defecto.

Figura 28.

Sección de subscripción para la creación de aplicación consumidora.

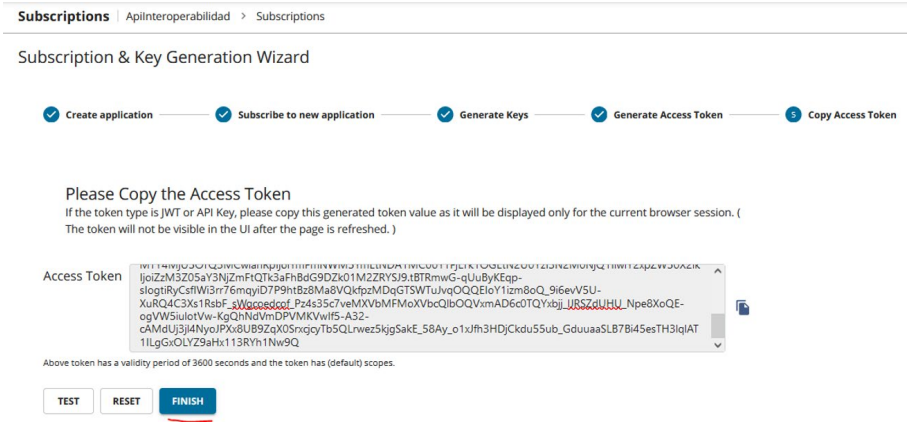


Nota. Quiñonez, F. 2023.

7. Los pasos 2, 3 y 4 los dejamos con los valores por defecto. Finalmente, le damos clic en **Finalizar** (Figura 29).

Figura 29.

Sección de generación de token para la creación de aplicación consumidora.

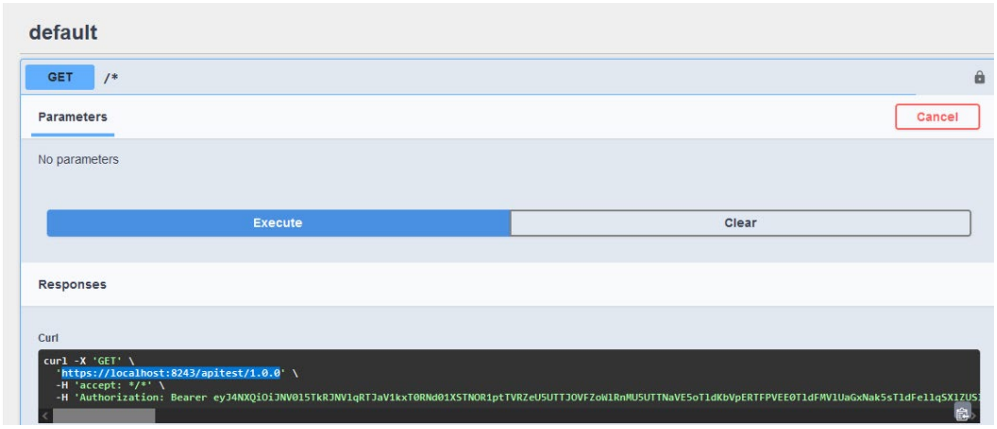


Nota. Quiñonez, F. 2023.

Con los pasos realizados anteriormente, estamos listos para validar nuestro servicio desde la aplicación consumidora.

1. Vamos a dar clic en **Try Out:**

Figura 30.
Herramienta para validar un API



Nota. Quiñonez, F. 2023.

Semana 15

Estimado estudiante, esta semana continuamos con el estudio de la unidad 2 de la asignatura de Interoperabilidad Empresarial. Esta semana, vamos a examinar cómo gestionar y publicar una nueva versión de una API utilizando la herramienta WSO2 API *Manager*. Abordaremos temas clave como la creación y actualización de API, la gestión de versiones y su importancia en un entorno empresarial.

2.3.5. Uso de WSO2 API manager para la gestión de versión

La habilidad de gestionar adecuadamente las versiones de API es esencial para mantener la interoperabilidad y la continuidad del negocio. WSO2 API *Manager* es una solución completa y flexible para la creación, publicación y gestión de API en un entorno empresarial.

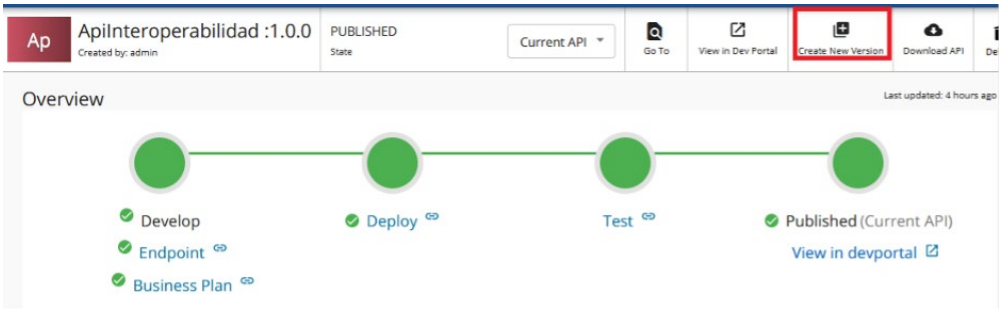
Estimados estudiantes, este proceso de publicar una nueva versión es necesario desarrollar cuando se desea modificar el comportamiento, mecanismo de autenticación, recursos, niveles de limitación, audiencias objetivo, etc., de una API publicada. Como menciona (WSO2, 2023), no se recomienda alterar una API publicada que tenga suscriptores conectados a ella.

Una vez creada que creemos nueva versión, generalmente se implementa como un prototipo para su promoción temprana. Un prototipo puede utilizarse para pruebas, sin necesidad de una suscripción, junto con las versiones publicadas de la API (WSO2, 2021). Después de un período de tiempo en el que se utiliza la nueva versión de la API en paralelo con las versiones anteriores, se puede publicar la API prototipada y deprecias las versiones más antiguas (Wso2, 2022).

En este momento vamos a ir paso a paso detallando como crear una nueva versión de nuestra API:

1. Ingresa al portal utilizando el portal de publicación.
2. Accede con las credenciales por defecto:
   a. Usuario: admin.
   b. Clave: admin.
3. A continuación, ingresamos a la sección de detalle de nuestra API (figura 31). Y realizamos un clic sobre el botón ***Create New Versión***.

Figura 31.
Sección para crear una nueva versión

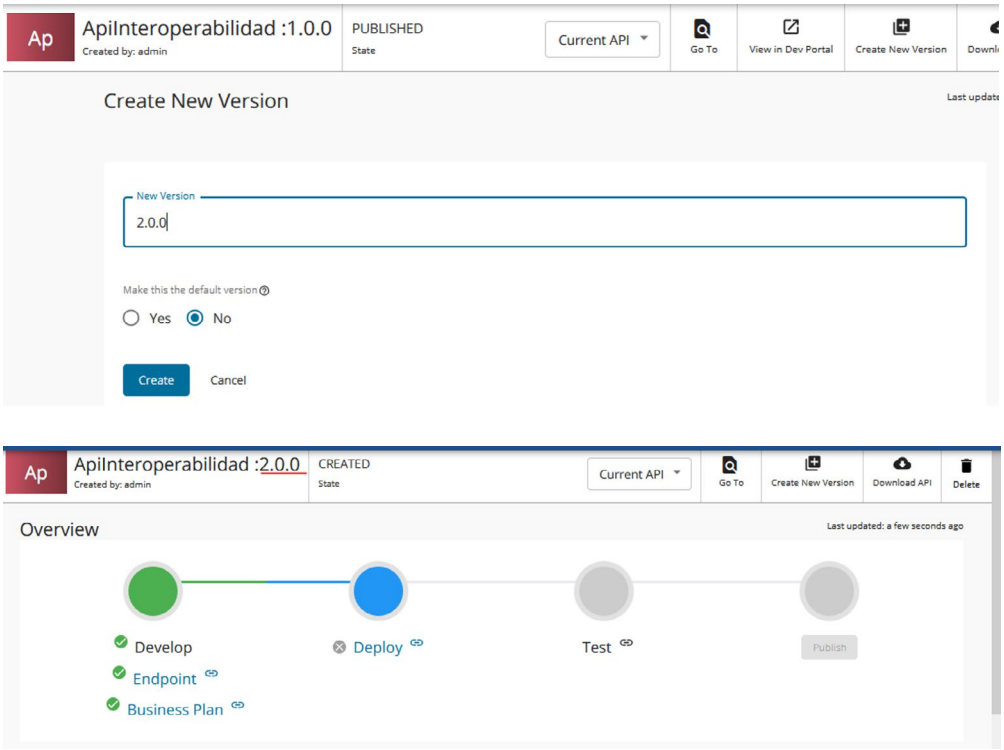


Nota. Quiñonez, F. 2023.

4. Ahora, ingresaremos la versión de la nueva versión (Figura 32), en este caso 2.0.0, y luego clic en el botón **Create**.

Figura 32.

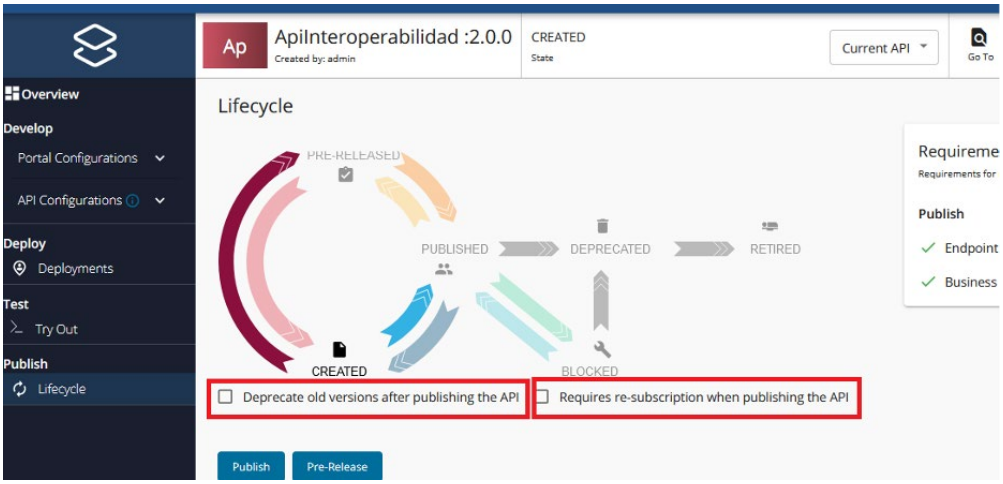
Sección para ingresar detalle de nuestra nueva versión de API.



Nota. Quiñonez, F. 2023.

Una vez que tenemos creada la nueva versión de nuestro api versión 2.0.0, vamos a publicar esta versión. Daremos clic en la opción **Lifecycle** (Figura 33).

Figura 33.
Sección detallada que muestra el ciclo de vida de un API



Nota. Quiñonez, F. 2023.

Nota: En este punto tenemos dos opciones de publicación. Podemos realizar un **Deprecated** de las versiones anteriores de nuestro Api, y adicional podemos hacer que las aplicaciones que están subscritas a una versión anterior, se subscriban automáticamente a la nueva versión.

Actividades de aprendizaje recomendadas

- Revise la sección de publicación de una nueva versión de API utilizando WSO2, y siga los pasos descritos. Con esta práctica obtendremos un acercamiento a la administración de versiones.

Autoevaluación 5

Una vez completado el estudio de las semanas 12, 13, 14 y 15 del segundo bimestre, se aconseja llevar a cabo una autoevaluación con el objetivo de verificar el aprendizaje obtenido. Al concluir la guía, se proporcionan las respuestas a dichas autoevaluaciones; no obstante, se insta a revisarlas únicamente después de haber registrado sus propias soluciones, permitiéndole de esta manera evaluar adecuadamente el nivel de conocimientos adquiridos.

Lea detenidamente cada una de las preguntas y seleccione la alternativa según corresponda.

1. ¿Qué es WSO2?

   a. Una herramienta de gestión de API.
   b. Un lenguaje de programación para el desarrollo *web*.
   c. Un sistema de gestión de bases de datos.

2. ¿Cuál es el principal propósito de la arquitectura de WSO2?

   a. Mejorar la eficiencia en la creación de sitios *web*.
   b. Facilitar la administración y seguridad de las API.
   c. Optimizar el rendimiento de aplicaciones de escritorio.

3. ¿Qué componente de la arquitectura de WSO2 es responsable de la publicación y administración de API?

   a. Almacenamiento de datos.
   b. *Gateway* de API.
   c. Publicador de API.

4. ¿Cuál es el propósito de la configuración básica en WSO2?

   a. Establecer la estructura de la base de datos.
   b. Personalizar la apariencia de la plataforma WSO2.
   c. Definir la configuración inicial para comenzar a utilizar WSO2.

5. ¿Qué componente de la arquitectura de WSO2 permite el acceso seguro a las API y gestiona las solicitudes entrantes?

   a. Almacenamiento de datos.
   b. *Gateway* de API.
   c. Publicador de API.

6. ¿Cuál es el primer paso para publicar una API en WSO2?

   a. Configurar el *Gateway* de API.
   b. Crear y diseñar la API.
   c. Establecer políticas de acceso y seguridad.

7. ¿Qué información es necesaria para crear una nueva API en WSO2?

   a. Nombre, contexto, versión y URL del punto final.
   b. Políticas de acceso y seguridad.
   c. Configuración del *Gateway* de API.

8. ¿Cómo se hace disponible una API publicada en WSO2 para los consumidores?

   a. A través del administrador de API.
   b. A través del publicador de API.
   c. A través de la tienda de API.

9. ¿Cuál es el enfoque recomendado para manejar diferentes versiones de una API en WSO2?

   a. Crear una nueva API para cada versión.
   b. Utilizar la misma API y modificarlo según sea necesario.
   c. Crear múltiples puntos finales para cada versión en el mismo API.

10. ¿Cómo pueden los consumidores identificar y acceder a diferentes versiones de una API en WSO2?

    a. A través del contexto y la versión del API en la URL.
    b. A través de un parámetro de cadena de consulta en la URL.
    c. A través de una cabecera personalizada en la solicitud.

Ir a solucionario

Semana 16

Actividades finales del bimestre

Durante este tiempo, es fundamental que se prepare para las evaluaciones presenciales. Para lograrlo, le sugiero revisar a fondo las actividades de aprendizaje correspondientes a la unidad 2, que aborda el tema de interoperabilidad en los negocios digitales. Cabe recordar que en esta unidad se trataron temas como la gestión de una API, el ciclo de vida y la administración de interfaces mediante la herramienta WSO2. No olvide realizar las autoevaluaciones 4, 5 y 6, ya que le serán de gran utilidad como material de estudio para el examen del segundo bimestre.

4. Solucionario

Autoevaluación 1		
Pregunta	**Respuesta**	**Retroalimentación**
1	c	El Open Group define la interoperabilidad como la capacidad de compartir información y servicios, intercambiar y usar información entre sistemas y proporcionar y recibir servicios de otros sistemas, permitiendo que estos funcionen juntos de manera eficiente.
2	b	La interoperabilidad empresarial se refiere a la capacidad de unidades organizacionales o procesos de negocio para comunicarse, cooperar y coordinarse entre sí, ya sea dentro de una gran empresa distribuida o dentro de una red empresarial.
3	c	Se identifican tres niveles de interoperabilidad: organizacional, semántica y técnica. La opción "funcional" no es un nivel de interoperabilidad.
4	b	La interoperabilidad reduce los costos para las organizaciones y los ciudadanos, ya que permite reutilizar recursos humanos, tecnológicos y logísticos, generando mayor valor.
5	c	En una arquitectura basada en eventos, **los sistemas de distribución de eventos** se encargan de enrutar los eventos a los consumidores apropiados. Los generadores de eventos crean los eventos, mientras que los consumidores de eventos los reciben y procesan.
6	b	En la arquitectura basada en microservicios, la mayoría de los componentes de los servicios individuales funcionan de forma separada entre sí, lo que permite una implementación independiente. Esto reduce el efecto que las modificaciones o actualizaciones de un componente tienen en otros servicios y componentes en comparación con una arquitectura monolítica.
7	a	La arquitectura de microservicios busca dividir la aplicación en módulos autocontenidos que se exponen a través de múltiples API, mientras que la arquitectura basada en eventos se refiere al diseño de sistemas mediante la transferencia de mensajes asíncronos llamados eventos entre los componentes.
8	c	El nivel de contexto permite entender la interoperabilidad de una empresa al visualizar la relación entre diferentes sistemas y su interacción con otros sistemas y usuarios.

Autoevaluación 1		
Pregunta	**Respuesta**	**Retroalimentación**
9	c	El diagrama de componentes en el modelo C4 detalla grupos de código dentro de un solo contenedor y representa abstracciones de la base de código.
10	b	El modelo C4 permite a los desarrolladores crear sistemas coherentes y escalables que mejoran la comunicación e intercambio de información entre diferentes sistemas empresariales.

Ir a la autoevaluación

Autoevaluación 2		
Pregunta	**Respuesta**	**Retroalimentación**
1	a	Los estándares y protocolos de interoperabilidad empresarial tienen como objetivo facilitar la colaboración y comunicación entre diferentes sistemas, aplicaciones y dispositivos, mejorando la eficiencia y productividad en los procesos empresariales.
2	b	El EDI tiene como objetivo principal permitir el intercambio electrónico de documentos y datos comerciales estructurados entre organizaciones, mejorando la eficiencia de procesos empresariales.
3	c	La estructura de un mensaje SOAP incluye una envoltura, un encabezado opcional y un cuerpo que contiene la información del mensaje, todo ello basado en XML.
4	b	La arquitectura REST se basa en protocolos y principios de la *web*, como HTTP y URI, para diseñar aplicaciones *web* escalables y eficientes.
5	a	El código de respuesta **HTTP 200 OK** indica que la solicitud ha sido procesada con éxito, y es comúnmente utilizado en servicios REST.
6	b	La integración de aplicaciones se enfoca en conectar aplicaciones y sistemas dispares dentro de una empresa, lo que permite una comunicación fluida y una gestión eficiente de la información.
7	b	La integración de datos implica la consolidación y sincronización de información proveniente de diferentes fuentes de datos, lo que permite a las organizaciones acceder a información coherente y actualizada en tiempo real.
8	c	Heroku es una plataforma en la nube que se ofrece como un S ervicio de P lataforma (PaaS), lo que permite a los desarrolladores construir, desplegar, gestionar y escalar aplicaciones *web* de manera rápida y eficiente sin preocuparse por la infraestructura subyacente.
9	b	Heroku es una plataforma que ayuda a los equipos de desarrollo en la implementación de prácticas de Integración Continua y Despliegue Continuo (CI/CD), ofreciendo un flujo de trabajo simplificado para la automatización de pruebas, integración y despliegue de aplicaciones. Esto facilita la detección temprana de errores y reduce el tiempo de lanzamiento al mercado.
10	c	El propósito principal de Azure API Management es gestionar el ciclo de vida integral de las API en un entorno empresarial.

Ir a la autoevaluación

Autoevaluación 3		
Pregunta	**Respuesta**	**Retroalimentación**
1	a	Los negocios B2B se refieren a las transacciones entre dos empresas, mientras que los negocios B2C se refieren a las transacciones entre empresas y consumidores finales.
2	a	Los proveedores en el ámbito B2B pueden ofrecer ventas exclusivas a empresas, lo que implica la oferta de productos o servicios únicamente a otras empresas, adaptándose a las necesidades de los clientes empresariales, incluyendo métodos y plazos de pago específicos.
3	b	La interoperabilidad es fundamental en un mundo digital y altamente interconectado para permitir la comunicación eficiente entre diferentes sistemas.
4	b	La interoperabilidad permite que diferentes sistemas y tecnologías trabajen juntos sin problemas, facilitando tareas cotidianas como las llamadas internacionales.
5	a	B2B se refiere a transacciones comerciales entre empresas, donde ambas partes buscan beneficiarse de las habilidades y recursos de la otra.
6	a	B2B implica relaciones comerciales de largo plazo y estrategias conjuntas, mientras que B2C se enfoca en satisfacer las necesidades inmediatas de los consumidores.
7	a	Las ventas exclusivas a empresas en el ámbito B2B suelen involucrar productos o servicios que solo son relevantes o accesibles para otras empresas.
8	b	Los pequeños comercios que no diferencian entre compradores empresariales y consumidores individuales realizan este tipo de transacciones.
9	a	B2C es un modelo de negocio en el que las empresas venden productos y servicios directamente a los consumidores finales.
10	a	En el modelo de negocio B2C, la satisfacción del cliente es fundamental para el éxito y crecimiento de la empresa. Un buen soporte, posventa y atención al cliente pueden marcar la diferencia en la percepción y fidelidad del consumidor.

Ir a la autoevaluación

Autoevaluación 4		
Pregunta	**Respuesta**	**Retroalimentación**
1	b	La media *types*, también llamados tipos MIME, se utilizan para especificar el formato de los datos en las solicitudes y respuestas HTTP, como JSON (application/json) y XML (application/xml).
2	c	El versionamiento en una API *web* permite a las aplicaciones cliente adaptarse a los cambios en la API, manteniendo la compatibilidad con versiones anteriores y permitiendo el uso de nuevas características y recursos.
3	c	Las API se caracterizan por ser accesibles a través de la red y utilizar protocolos *web* estándar, así como por tener interfaces bien definidas.
4	c	La gestión de API es importante para establecer confianza y seguridad, regular el uso compartido y los permisos, y para monetizar el uso, entre otros beneficios. No es una razón para aumentar la complejidad de las aplicaciones.
5	b	Exponer los procesos, datos y servicios centrales como API al público permite a otros crear nuevas soluciones, aumentar el potencial de crecimiento y avances de asociación, reducir la duplicación y el retrabajo, e incrementar el desarrollo colaborativo.
6	b	La etapa de "Definir" se centra en garantizar que las operaciones relacionadas con una API estén correctamente definidas, estableciendo las bases para diseñar y dar vida a una API de manera efectiva.
7	b	La especificación de OpenAPI proporciona un vocabulario común para describir la superficie de las solicitudes y respuestas de las API, así como para los *webhooks*, facilitando la integración entre productores y consumidores de la API.
8	b	Un servidor simulado (*mock server*) ayuda a replicar gran parte de la funcionalidad que una API tendría en producción antes de tener que escribir cualquier código, facilitando el diseño y la implementación de las API.
9	a	La fase de "Diseñar" del ciclo de vida se enfoca en establecer un proceso y enfoque formal para diseñar una API que ayude a establecer la consistencia de cada API que se implementa, utilizando patrones comunes de la industria y de la organización mientras se establecen prácticas conocidas para dar forma a la superficie y comportamientos de las API.

Autoevaluación 4		
Pregunta	**Respuesta**	**Retroalimentación**
10	b	La fase de “Seguridad” del ciclo de vida de la API se asegura de que exista un enfoque consistente para la gestión de identidad y acceso para cada API, implementando pruebas de seguridad adecuadas para garantizar que todas las API estén siendo escaneadas y protegidas de manera consistente.

Ir a la autoevaluación

Autoevaluación 5		
Pregunta	**Respuesta**	**Retroalimentación**
1	a	WSO2 es una plataforma de gestión de API que permite a las organizaciones diseñar, desarrollar, publicar y administrar API.
2	b	La arquitectura de WSO2 está diseñada para ayudar a las organizaciones en la gestión y protección de sus API, así como en la implementación de políticas de acceso y control.
3	c	El componente Publicador de API en WSO2 permite a los desarrolladores publicar, actualizar y administrar las API, así como aplicar políticas de acceso y control.
4	c	La configuración básica en WSO2 permite a los usuarios establecer las opciones iniciales necesarias para comenzar a trabajar con la plataforma, como la configuración de la base de datos y la conexión a otros sistemas.
5	b	El componente *Gateway* de API en WSO2 es responsable de garantizar el acceso seguro a las API y administrar las solicitudes entrantes de los clientes, aplicando políticas de seguridad y control de acceso.
6	b	El primer paso para publicar una API en WSO2 es crear y diseñar la API, definiendo sus recursos, métodos y parámetros, así como la información relevante para los consumidores del API.
7	a	Para crear una nueva API en WSO2, es necesario proporcionar información básica como el nombre del API, el contexto, la versión y la URL del punto final del servicio que se desea exponer.
8	c	Una vez que una API ha sido publicado en WSO2, se hace disponible para los consumidores en la Tienda de API, donde pueden descubrir, suscribirse y acceder a la documentación del API.
9	a	En WSO2, se recomienda crear una nueva API para cada versión, lo que permite mantener y gestionar de forma independiente las diferentes versiones y facilita a los consumidores la transición entre ellas.
10	a	En WSO2, las diferentes versiones de una API se identifican y acceden a través del contexto y la versión del API en la URL, lo que permite a los consumidores seleccionar fácilmente la versión deseada al realizar solicitudes a la API.

Ir a la autoevaluación

5. Referencias bibliográficas

Allen, J. (2006). The Unicode Standard / the Unicode Consortium. *Unicode, Inc.*, *5*, 42

Amazon. (2023). *Arquitectura basada en eventos*. https://aws.amazon.com/es/event-driven-architecture/

Box, D. (2000). *Simple Object Access Protocol (SOAP) 1.1*. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Brown, S. (2018). *The C4 model for visualising software architecture*. https://c4model.com/

Brown, S. (2022). *The C4 model for visualising software architecture*. https://c4model.com/

EDI para Walmart | Meade Willis Inc. (n.d.). Retrieved March 20, 2023, from https://www.meadewillis.com/es/food-beverage-edi/walmart-edi

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

FirstBit. (2023). *Implementamos UiPath la plataforma líder de RPA*. https://first-bit.co/uipath

Hiberus. (2023). *Hiberus Tecnología, Expertos en Consultoría de Negocio y Tecnología*. https://www.hiberus.com/

IEEE Standard Glossary of Software Engineering Terminology. (1990). *IEEE Std 610.12-1990*, 1–84. https://doi.org/10.1109/IEEESTD.1990.101064

Intel. (2021). *Qué son los microservicios y la arquitectura de microservicios? -...* https://www.intel.la/content/www/xl/es/cloud-computing/microservices.html

Ionos. (2020, April 15). *SOAP | Web services y recomendación de W3C*. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/soap-simple-object-access-protocol/

Jeremy H. (2022). *What Are Containerized Microservices? | DreamFactory Software – Blog*. https://blog.dreamfactory.com/what-are-containerized-microservices/

Jin, B., Sahni, S., & Shevat, A. (2021). *Designing Web APIs, BUILDING APIS THAT DEVELOPERS LOVE*.

Kotzé, P., & Neaga, I. (n.d.). *Towards an Enterprise Interoperability Framework*. Retrieved February 17, 2023, from http://interop-vlab.eu/

Kotzé, P., & Neaga, I. (2010). *Towards an enterprise interoperability framework*. 19–20. http://interop-vlab.eu

Lane, kin. (2022, January). *The 8-Point API Lifecycle Blueprint | Postman Blog*. https://blog.postman.com/api-lifecycle-blueprint/

Microsoft. (2023a, February 10). *Información general y conceptos clave de Azure API Management | Microsoft Learn*. https://learn.microsoft.com/es-es/azure/api-management/api-management-key-concepts

Microsoft. (2023b, March 27). *Web API design best practices – Azure Architecture Center | Microsoft Learn*. https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

Naser, A. (2021). *Gobernanza digital e interoperabilidad gubernamental: una guía para su implementación*. www.cepal.org/apps

Open Group. (2018). *Open Agile Architecture*TM. https://pubs.opengroup.org/architecture/o-aa-standard/software-architecture.html#_event_driven_architecture

¿Qué es la interoperabilidad y cómo puede lograrla mi empresa? (2019). https://nexusintegra.io/es/que-es-la-interoperabilidad-y-como-puede-lograrla-mi-empresa/

Sánchez, Fani. (n.d.). ¿Qué es el Business to business (B2B): negocio entre empresas? Retrieved April 26, 2023, from https://www.humanlevel.com/diccionario-marketing-online/business-to-business-b2b

Solace. (2022). *The Complete Guide to Event-Driven Architecture*. https://solace.com/what-is-event-driven-architecture/#who-uses-eda

Solano, M. (2020). *Qué es la Interoperabilidad | Blog EAE Online*. https://www.eaeprogramas.es/blog/negocio/tecnologia/que-es-la-interoperabilidad

The TOGAF Standard, Version 9.2 – Interoperability Requirements. (n.d.). Retrieved February 17, 2023, from https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap25.html

Wso2. (2022). *Create a New API Version – WSO2 API Manager Documentation 4.2.0*. https://apim.docs.wso2.com/en/latest/design/api-versioning/create-a-new-api-version/

WSO2. (2023). Architecture – WSO2 *API Manager Documentation 4.2.0*. https://apim.docs.wso2.com/en/latest/get-started/apim-architecture/

Young, M. (2018). *How to Build an Effective Initial Deployment Pipeline | by Michelle Young | Chatbots Life*. https://chatbotslife.com/how-to-build-an-effective-initial-deployment-pipeline-e21b7e382fb5