



UTPL
La Universidad Católica de Loja

Modalidad Abierta y a Distancia

Itinerario 2: Programación Integrativa

Guía didáctica



Facultad de Ingenierías y Arquitectura

Departamento de Ciencias de la Computación y Electrónica

Itinerario 2: Programación Integrativa

Guía didáctica

Carrera	PAO Nivel
▪ <i>Tecnologías de la información</i>	VIII

Autora:

López Vargas Jorge Afranio



D S O F _ 4 0 8 0

Asesoría virtual
www.utpl.edu.ec

Itinerario 2: Programación Integrativa

Guía didáctica

López Vargas Jorge Afranio

Universidad Técnica Particular de Loja



4.0, CC BY-NY-SA

Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

www.ediloja.com.ec

edilojainfo@ediloja.com.ec

Loja-Ecuador

ISBN digital - 978-9942-39-271-8



La versión digital ha sido acreditada bajo la licencia Creative Commons 4.0, CC BY-NY-SA: Reconocimiento-No comercial-Compartir igual; la cual permite: copiar, distribuir y comunicar públicamente la obra, mientras se reconozca la autoría original, no se utilice con fines comerciales y se permiten obras derivadas, siempre que mantenga la misma licencia al ser divulgada. <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>

28 sep. 2021

Índice

1. Datos de información.....	8
1.1. Presentación de la asignatura	8
1.2. Competencias genéricas de la UTPL	8
1.3. Competencias específicas de la carrera.....	8
1.4. Problemática que aborda la asignatura.....	8
2. Metodología de aprendizaje.....	10
3. Orientaciones didácticas por resultados de aprendizaje.....	11
Primer bimestre	11
Resultado de aprendizaje 1	11
Contenidos, recursos y actividades de aprendizaje	11
Semana 1	11
Unidad 1. Introducción a la integración de sistemas	11
1.1. Sistemas de información empresarial	13
1.2. Cómo reconocer cuando una organización necesita sistemas integrados	17
1.3. ¿Qué es la integración?.....	18
1.4. ¿Qué es la integración de sistemas?.....	19
1.5. Tipos de arquitecturas para integración de sistemas	24
Actividades de aprendizaje recomendadas	25
Semana 2	25
1.6. Sistemas distribuidos	27
1.7. Comunicación entre sistemas.....	28
Actividades de aprendizaje recomendadas	44
Autoevaluación 1	46
Resultado de aprendizaje 2.....	48
Contenidos, recursos y actividades de aprendizaje	48
Semana 3	48
Unidad 2. Integración centrada en aplicaciones, una integración funcional .	48
2.1. Codificación integrativa	51
2.2. Técnicas de programación integrativa.....	52

2.3. Arquitecturas orientadas a servicios	53
Actividades de aprendizaje recomendadas	65
Semana 4	65
Actividades de aprendizaje recomendadas	75
Semana 5	75
Actividades de aprendizaje recomendadas	86
Autoevaluación 2	87
Semana 6	89
Actividades de aprendizaje recomendadas	95
Semana 7	95
Actividades de aprendizaje recomendadas	101
Autoevaluación 3	102
Semana 8	103
Actividades finales del bimestre.....	103
Segundo bimestre	104
Resultado de aprendizaje 3	104
Contenidos, recursos y actividades de aprendizaje	104
Semana 9	104
Unidad 3. Integración e interoperabilidad de datos	104
3.1. Integración de datos	106
3.2. Tipos de integración.....	107
3.3. Técnicas de integración de datos	109
3.4. Ciclo de vida de la integración de sistemas	109
3.5. Los datos y sus tipos	115
Semana 10	118
Unidad 4. Integración de datos no estructurados	118

4.1. Extracción de datos no estructurados	119
4.2. Transformar datos no estructurados.....	121
Actividades de aprendizaje recomendadas	127
Autoevaluación 4	128
Semana 11	130
4.3. Integración de datos no estructurados en la práctica	130
4.4. Otros formatos de datos no estructurados una propuesta de trabajo	134
Actividades de aprendizaje recomendadas	139
Semana 12	140
Unidad 5. Integración de datos semiestructurados	140
5.1. JSON - JavaScript Object Notation	141
Actividades de aprendizaje recomendadas	154
Semana 13	154
5.2. Extensible Markup Language XML.....	154
Actividades de aprendizaje recomendadas	163
Semana 14	163
Unidad 6. Integración de datos estructurados	163
6.1. Modelos de base datos.....	165
6.2. Interactuando con datos estructurados desde un programa.....	167
Actividades de aprendizaje recomendadas	179
Autoevaluación 5	181
Semana 15	183
Unidad 7. Integración de datos en la práctica.....	183
7.1. Planteamiento del problema	183
7.2. La solución.....	184
Actividad de aprendizaje recomendada	189
Semana 16	189
Actividades finales del bimestre.....	189

4. Solucionario	191
5. Glosario.....	199
6. Referencias bibliográficas	202
7. Anexos	206



1. Datos de información

1.1. Presentación de la asignatura



1.2. Competencias genéricas de la UTPL

- Comunicación oral y escrita.

1.3. Competencias específicas de la carrera

- Implementar aplicaciones de baja, mediana y alta complejidad integrando diferentes herramientas y plataformas para dar solución a requerimientos de la organización.

1.4. Problemática que aborda la asignatura

La asignatura aborda uno de los problemas recurrentes dentro de las empresas de toda índole que poseen más de un sistema informático para la gestión de sus actividades, la integración, pero no solo a nivel de sistemas o aplicaciones sino también la integración de datos.

La integración a nivel de aplicaciones es un problema al que se le ha propuesto diferentes soluciones que, con el pasar del tiempo y el avance de la tecnología, han tomado diferentes formas tales como sockets, arquitecturas orientadas a servicios, SOAP, REST y recientemente microservicios, cada una de esas soluciones y las que vendrán buscan optimizar los recursos de una empresa.

Para la integración de los datos, los problemas se originan debido a los diferentes grupos de datos (no estructurados, semiestructurados y estructurados) y cómo integrarlos con el fin de realizar análisis de datos e inteligencia de negocios que permitan obtener una ventaja competitiva en un entorno empresarial en donde se necesita estar un paso delante de los demás.

Estos problemas cambian de forma con la llegada de los datos masivos, la denominada Big Data, ya que las fuentes de datos no solo están dentro de la empresa, sino que están fuera en forma de redes sociales en línea y los sistemas que las empresas necesitan también se encuentran fuera, eso hace que buscar soluciones a los problemas de integración sea un tema muy retador.



2. Metodología de aprendizaje

Para lograr el aprendizaje de la asignatura se propone el uso de varias metodologías de aprendizaje que se resumen a continuación.

A través de los diferentes materiales didácticos que están a su disposición, tales como el plan docente, el texto guía, el entorno virtual de aprendizaje y la interacción, tanto síncrona como asíncrona, con su docente tutor se usará el aprendizaje guiado.

Además, se usará el autoaprendizaje ya que usted debe leer, revisar diferentes materiales y responder a autoevaluaciones.

Finalmente, el aprendizaje basado en casos ya que tendrá que desarrollar actividades práctico-experimentales que le brindan la posibilidad de experimentar el trabajo real en un entorno controlado y dentro de un contexto definido.

Adicionalmente se recomienda:

- Dedicar el tiempo que se señala en el plan docente a cada uno de los componentes.
- Leer los anuncios que semanalmente se publican en el entorno virtual de aprendizaje.
- Participar en las actividades síncronas planificadas para cada bimestre.
- Comunicarse con su docente tutor a través de las tutorías permanentes que se dictan cada semana o a través de mensajería del EVA o correo electrónico.



3. Orientaciones didácticas por resultados de aprendizaje



Primer bimestre

Resultado de aprendizaje 1

- Diseña, desarrolla y prueba un programa de socket que se comunica entre dos servicios diferentes con los sockets TCP/IP y sockets de datagramas.

El resultado de aprendizaje se alcanza a través del estudio teórico de los principales conceptos para luego aplicarlos a través de la creación de programas que permitan experimentar con situaciones bastante reales y que son comunes dentro del contexto laboral.

Contenidos, recursos y actividades de aprendizaje



Semana 1

Unidad 1. Introducción a la integración de sistemas

Antes de iniciar con el desarrollo de los contenidos, es menester comprender el problema que actualmente existe en muchas empresas y que es materia de estudio en esta asignatura. Con este fin, realice una lectura de los siguientes párrafos.

La mayoría de las empresas poseen varios sistemas informáticos transaccionales (de aquí en adelante serán llamados simplemente sistemas) para realizar diversas funcionalidades. Por ejemplo, una universidad entre sus sistemas puede tener un sistema para la administración de su personal (sistema de recursos humanos) y un sistema en el cual, entre otras cosas, se asignan docentes a las asignaturas que se debe impartir (sistema académico).

Esos sistemas han sido desarrollados utilizando diferentes arquitecturas y tecnologías las cuales, usualmente, no han sido diseñadas para la integración y las empresas no pueden darse el lujo de dejarlas de usar o reemplazarlas de la noche a la mañana porque son de misión crítica, tampoco existe la posibilidad de volver a desarrollarlos desde cero, ya que el entorno empresarial es bastante competitivo.

Además, las empresas tienen la necesidad, de vez en cuando, de usar nuevas aplicaciones y sistemas. Esas nuevas soluciones generalmente usan arquitecturas y tecnologías modernas que difieren significativamente de las arquitecturas usadas por los sistemas antiguos. Los nuevos sistemas deben integrarse a los existentes, así como estos deben integrarse entre sí para cumplir con los objetivos de disponibilidad y accesibilidad a la información y a las funcionalidades.

Con la lectura de los párrafos anteriores de seguro notó que el término integración es el que más destaca. Antes de continuar, trate de responder a la pregunta, ¿qué es la integración? La respuesta a esta pregunta la encontrará al continuar con la lectura.

Dentro de las empresas que poseen varios sistemas es muy probable que se necesite *compartir información* entre sistemas. Usando el ejemplo anterior, es probable que el sistema de recursos humanos necesite conocer cuáles son las asignaturas que imparte un docente.

Además de compartir información, hoy en día, es muy común que las funcionalidades de un sistema se utilicen en otro, es decir, *compartir funcionamiento*. En el ejemplo de la universidad, imagine que para ingresar a los sistemas se necesita de la funcionalidad de autenticación (usuario y contraseña) que se implementa en el sistema académico, ya que este cuenta con una mejor tecnología de seguridad.

Detenga por un momento su lectura y trate de plantear dos problemas, uno para cada escenario (*compartir información* y *funcionalidad*) que puede enfrentar una organización como la universidad que se propone en el ejemplo.

De seguro no solo encontró dos problemas sino muchos más. A continuación, le propongo los siguientes problemas:

- Para el primer caso, compartir información. Inconsistencia de los datos, es muy probable que docentes que están en el sistema académico no consten en el de recursos humanos o viceversa, debido a que cada sistema maneja su propia base de datos con números de cédula diferentes para un mismo docente. Error común al digitar.
- En el segundo caso, compartir funcionalidad. Diversidad tecnológica. Los sistemas han sido desarrollados en lenguajes de programación diferentes y no pueden invocar funcionalidades entre sí.

A este tipo de problemas y a otros más se los conoce como problemas de integración de información y de aplicaciones, respectivamente, y se constituyen en el tema principal de estudio de nuestra asignatura.

Es momento de empezar a formalizar los temas de estudio para comprender aún más la integración. En la siguiente sección se presentará los conceptos fundamentales asociados con la integración.

1.1. Sistemas de información empresarial

La integración tiene diferentes promotores, posiblemente el más grande son los denominados sistemas de información empresarial. Esta afirmación es razón suficiente para realizar una revisión de este tipo de sistemas. Revise los siguientes párrafos que tienen como propósito recordarle algunos conceptos que seguro ya estudió en asignaturas predecesoras a esta.

Anteriormente señalamos que una empresa tiene varios sistemas informáticos que manejan ciertos datos de la organización, integrando todos sus sistemas las organizaciones ejecutan sus procesos de negocio; esa integración y coordinación se hace a través de los denominados Sistemas de Información Empresarial o Enterprise Information Systems (EISs). Una definición formal se presenta a continuación.

En Romero y Vernadat (2016) se define a los EISs como software para la gestión empresarial que abrancan módulos que dan soporte tanto a las áreas organizativas como a las funcionales. Sobre su estructura, los mismos autores sostienen que estos sistemas están construidos por computadoras, software, personas, procesos y datos.

Deténgase un momento y analice esa definición, intente encontrar la diferencia entre un EIS y un sistema informático, ya que como cualquier otro sistema estos procesan entradas, almacenan datos y producen resultados.

Para ayudarle a establecer las diferencias, revise lo que mencionan Moreno-Cevallos y Dueñas-Holguín (2018), un EIS va más allá del aspecto meramente computacional y llega a la organización de los sistemas informáticos y a la obtención de los datos almacenados en cada sistema informático para integrarlos, procesarlos y generar información que busca ayudar en la toma de decisiones, la correcta administración de la organización y brindarle una ventaja competitiva.

La Figura 1 muestra una síntesis de cómo un sistema de información organiza y obtiene información (tanto de sistemas transaccionales propios y otros sistemas como la Web y las redes sociales en línea) de los sistemas (que pasan a llamarse subsistemas) que una empresa tiene. Si una empresa tiene varios sistemas informáticos, esto no implica que tenga un sistema de información empresarial.

Figura 1.

Estructura de un sistema de información



Ya conoce la definición de un sistema de información empresarial y sabe diferenciarlo de cualquier otro sistema informático, ahora le propongo que revise cuál es el objetivo que persigue este tipo de sistemas.

En Peiró (2020) se menciona que el principal objetivo de un sistema de información es la gestión y la administración de los datos e información que lo componen, esto se complementa con lo que menciona Cobarsi-Morales (2011), un sistema de información empresarial aporta en tres aspectos, el primero contribuye en asegurar la eficacia (satisfacer necesidades), la eficiencia (productividad y ahorro) y la calidad, y mejora continua en las operaciones cotidianas (rutina diaria de la organización). Un segundo aspecto genera un entorno de trabajo cooperativo y proactivo. Un tercer y último aspecto prepara a la organización para asumir retos más allá de lo cotidiano (creatividad e innovación, cambios organizativos, crisis, etc.).

Luego de leer el párrafo anterior de seguro tiene claro que los EIS van más allá de las transacciones llegando a la administración de la organización. Siga estudiando sobre este interesante tema. En los siguientes párrafos conocerá la clasificación de los sistemas de información empresarial.

La clasificación la puede encontrar en Proaño et al (2018), ahí se menciona que los sistemas de información empresarial se clasifican en:

- Sistemas de planeación de recursos empresariales (Enterprise Resource Planning, ERP). Encargados de unificar todos los datos de cada área en un repositorio común con el propósito de facilitar el flujo de información, mejorando así la comunicación entre áreas.
- Sistemas de administración de la cadena de suministros (Supply Chain Management, SCM). Se encargan del manejo de las relaciones entre los proveedores y las empresas para garantizar el desarrollo eficiente de las actividades. Buscan invertir menos tiempo al costo más bajo posible.
- Sistemas de administración de las relaciones con el cliente (Customer Relationship Management, CRM). Se encargan de la administración de las relaciones entre la empresa y sus clientes ayudando a las empresas a identificar, atraer y conservar clientes.

Es necesario mencionar que esas categorías han evolucionado con el transcurso del tiempo y el avance en las tecnologías. Un trabajo que muestra

esa evolución es el realizado por Romero y Vernadat (2016) y que se ha resumido de la siguiente manera:

- ERP:
 - ERP/I, uso de la información dentro de la empresa (intraempresarial).
 - ERP/II o eERP usa información de otras empresas (interempresarial).
 - ERP/III, uso de información publicada en redes sociales.
- SCM:
 - SCM/I, centrado en inventarios, costes de producción, control de la capacidad y calidad.
 - SCM/II incluye información de mercado, gestión de pedidos para mejorar la disponibilidad del producto.
 - SCM/III, incluir información del cliente para mejorar su satisfacción.
 - SCM/IV, gestionar la demanda, distribución de productos terminados, logística para reparación, refabricación y/o reciclaje.
- CRM:
 - CRM/I, marketing basado en la gestión de contactos.
 - CRM/II, analizar la información que genera el cliente para mejorar el servicio.
 - CRM/III respalda a actividades de ventas, marketing automatizadas. Usa herramientas como emisión de tickets y administración de casos.

Para finalizar con esta sección sobre los sistemas de información empresarial, le propongo que revise la evolución histórica de los mismos. Con ese fin en mente revise los siguientes párrafos.

Según Romero et al (2016), los sistemas de información se pueden remontar a los inicios de la década de los 60, en donde su principal función era automatizar sistemas basados en actividades manuales y reemplazar a sistemas basados en papel.

Un resumen de la evolución de los sistemas de información se presenta en el recurso. Esa evolución está aplicada a las industrias de manufactura y su necesidad de integración.

Evolución histórica de los sistemas de información

En ese apartado revisó a una de las principales iniciativas que buscan la integración de los sistemas transaccionales que tiene una organización. En el siguiente apartado se presenta las características de las organizaciones que necesitan integración de sus sistemas.

1.2. Cómo reconocer cuando una organización necesita sistemas integrados

Si bien, existen muchos síntomas que señalan la necesidad de integración de sistemas, en este apartado se le presenta únicamente dos ya que probablemente los ha experimentado como técnico y como usuario de cualquier sistema.

Ingreso repetido de datos. Este primer síntoma es evidente cuando se debe ingresar repetidas veces los mismos datos, mostrando así que los diferentes sistemas no están interconectados entre sí y es responsabilidad de los usuarios repetir el ingreso. Esto sin lugar a duda causa malestar y pérdida de tiempo.

Alta latencia para acceso a la información. Este síntoma está relacionado con el tiempo que los usuarios deben esperar para que los cambios que se hace en un sistema se vean reflejados en los otros. Este tiempo puede variar de horas a días y como el anterior causa molestias en los usuarios y la pérdida de recursos.

Si usted, como parte de una organización, encuentra uno de esos síntomas o ambos no tenga la menor duda que se encuentra ante un problema de integración de sistemas de información que tal vez aún sea llevadero, pero con el pasar del tiempo se convertirá en una situación intolerable.

Es momento de acercarse un poco más a la integración. Es por ello que en el siguiente apartado estudiará el concepto de integración desde dos puntos de vista: empresarial y de sistemas.

1.3. ¿Qué es la integración?

Antes de definir la integración de sistemas, le invito a que estudie la definición de integración desde el punto de vista de una organización o empresa.

¿Qué es la integración empresarial?

Según Eii (2013, como se citó en Rubeš y Jandoš, 2013), la integración empresarial es la alineación de estrategias, procesos comerciales, sistemas de información, tecnologías y datos a través de los límites organizacionales para brindar una ventaja competitiva.

La anterior es una definición bastante amplia, para hacer un poco más explícita revise los siguientes párrafos que fueron propuestos por Romero y Vernadat (2016).

Los mencionados autores sostienen que, si bien cualquier proceso de integración es un desafío técnico, es ante todo organizativo, ya que se necesita analizar aspectos funcionales, información, recursos y de organización que necesitan ser integrados. Además, mencionan que la integración empresarial (IE) se preocupa de facilitar la información, el control y los flujos de materiales a través de las fronteras organizacionales mediante la conexión de todas las funciones necesarias y entidades funcionales heterogéneas (por ejemplo, sistemas de información, dispositivos, aplicaciones y personas) para mejorar la comunicación, cooperación y colaboración dentro de la empresa, para que esta se comporte como un todo integrado mejorando así su productividad general, flexibilidad y capacidad para la gestión del cambio o la reactividad.

Finalmente, sostienen que la integración empresarial se ocupa de estrategias, métodos, modelos y herramientas que tienen como objetivo consolidar y coordinar bases de datos y aplicaciones.

En cuanto a los niveles de integración, afirman que existen tres niveles de integración:

- Física, también conocida como de comunicación, que se ocupa de las interconexiones de sistemas y el intercambio de datos (interconexión de dispositivos físicos, computadores, bases de datos, etc.), todo a través de redes informáticas.

- De aplicaciones o nivel de cooperación, que trabaja en la interoperabilidad de aplicaciones de software y bases de datos en entornos informáticos heterogéneos.
- Comercial o nivel de colaboración, que se trata con la coordinación de funciones, procesos y personas que administran, controlan y monitorean las operaciones de la empresa.

De seguro notó que la integración es un proceso grande, en sí es un área de estudio para la que se ha desarrollado una disciplina denominada ingeniería de la integración empresarial que, aunque está fuera del alcance de esta asignatura, nos deja ver lo importante y lo extenso del tema.

Con la lectura de los párrafos anteriores es posible que comience a ver a la integración empresarial como un proceso en el que interviene una única empresa, pero la integración empresarial no es solo a lo interno de una empresa, también se puede dar entre dos o más empresas que necesitan coordinar sus negocios o trabajar juntas como una sola entidad (ejemplo: fusiones o adquisiciones de empresas) para aumentar su eficiencia o productividad y, por lo tanto, requieren una integración estrecha o completa de los procesos comerciales, los recursos y los sistemas de información asociados.

Ya conoce que la integración empresarial tiene diferentes aristas y que los sistemas es una de ellas. Es momento de conocer la definición de integración de sistemas.

1.4. ¿Qué es la integración de sistemas?

Esta es una pregunta que puede tener muchas respuestas, ya que no es un problema nuevo y muchas de las definiciones responden al contexto tecnológico que rodeaba al autor cuando elaboró su respuesta. Sin embargo, se ha seleccionado una definición bastante actual y libre del contexto tecnológico. Le invito a profundizar su conocimiento sobre este importante tema.

Lea detenidamente la siguiente definición.

Para Freire et al (2019), la integración de sistemas es el campo de investigación que aporta con metodologías, técnicas y herramientas

que apoyan al desarrollo de soluciones de integración que permitan que las diferentes aplicaciones funcionen de forma síncrona y posibiliten la incorporación de nuevas aplicaciones con el menor impacto en las existentes.

¿Qué le pareció la definición? De seguro interesante. Note cómo la definición es amplia, sin nombrar tecnologías concretas, dando a la integración de sistemas la categoría de campo de investigación y no solo un mero proceso.

Entonces, usted se encuentra estudiando un tema bastante amplio y que, a veces, puede tomar diferentes nombres. En Fazlollahi & Franke (2018) se listan varios sinónimos para el concepto de integración de sistemas, además, por cada uno de ellos se presenta en paréntesis su acrónimo en inglés: integración de aplicaciones empresariales (EAI), integración de aplicaciones (AI), integración de la cadena de valor (VCI), integración de la cadena de suministro (SCI), integración comercial extendida (EBI) e integración del comercio electrónico.

Si usted realiza una búsqueda de información sobre esta temática es muy probable que se encuentre con algún nombre o acrónimo del listado anterior, pero ya conoce que se tratan de diferentes formas de nombrar a un mismo concepto.

Es momento de conocer la clasificación de los sistemas de información integrados, un resumen muy completo se encuentra en el trabajo realizado por Romero y Vernadat (2016) que se expresa en los siguientes párrafos.

Los sistemas de información integrados pueden ser clasificados en tres clases:

- Sistemas interconectados. Son la forma de integración más débil, pero ampliamente utilizada, incluso en la actualidad. Los sistemas solo pueden intercambiar datos utilizando protocolos y esquemas de datos predefinidos a través de TCP / IP o de esquemas SQL que usan aplicaciones propietarias como DBLink de Oracle. Básicamente, el intercambio de información es posible trasmitiendo una copia de los datos desde un sistema a otro.
- Sistemas fuertemente acoplados. Integran todas las fuentes de datos mediante la creación de “mapeos” lógicos entre ellas, mediante

interfaces estandarizadas (codificadas de forma rígida) y esquemas globales predefinidos.

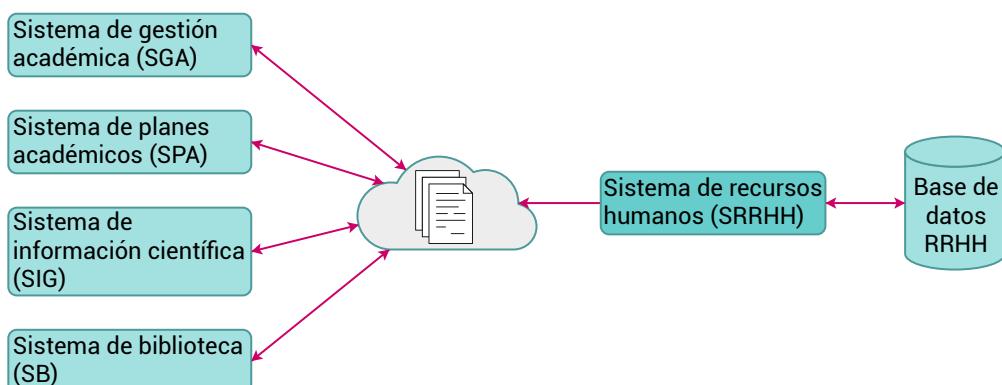
- Sistemas débilmente acoplados. Coordinan tanto las fuentes de datos y las aplicaciones de software, a través de un conjunto de esquemas federados, además de formatos y protocolos de intercambio de datos abiertos, preferiblemente formatos XML u otros formatos estándares. Un ejemplo de esta categoría son los sistemas que usan un “Enterprise Service Buses (ESB)” para el enrutamiento de mensajes, otro ejemplo son las arquitecturas orientadas a servicios o SOA, estos temas los estudiará más adelante.

Existe también la posibilidad de que los sistemas se encuentren en puntos intermedios si consideramos a las tres categorías como marcas en una escala, por ejemplo: un sistema de información integrado podría estar formado por varios sistemas transacciones interconectados y que se encuentren estrechamente acoplados entre sí.

Si analiza por un instante cada una de las tres categorías, de seguro se percató que las dos últimas categorías se pueden aplicar a diferentes sistemas e inclusive a los módulos que forman un sistema. Para afirmar la comprensión de las tres categorías revise las siguientes figuras, las mismas que amplían el ejemplo con el que inicio esta unidad.

Figura 2.

Ejemplo de un sistema interconectado



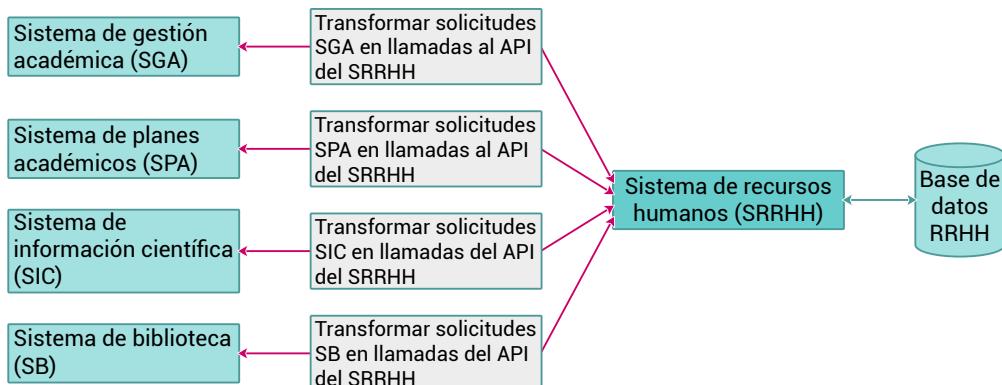
En la figura 2 el sistema de recursos humanos, periódicamente, genera una copia de sus datos en formatos como CSV y los comparte con el resto de

las aplicaciones que los necesitan, quienes a su vez utilizan servicios TCP/IP para obtener una copia de estos.

Antes de continuar, piense en alguna complicación que se puede presentar en este tipo de integración. Aquí le proponemos una: suponga que el tiempo de actualización de los archivos del sistema de recursos humanos es de 24 horas, entonces para que un docente tenga acceso a la biblioteca deberá esperar por lo menos un día hasta que el sistema de biblioteca actualice sus datos con la última nómina de personal.

Figura 3.

Sistema fuertemente acoplado

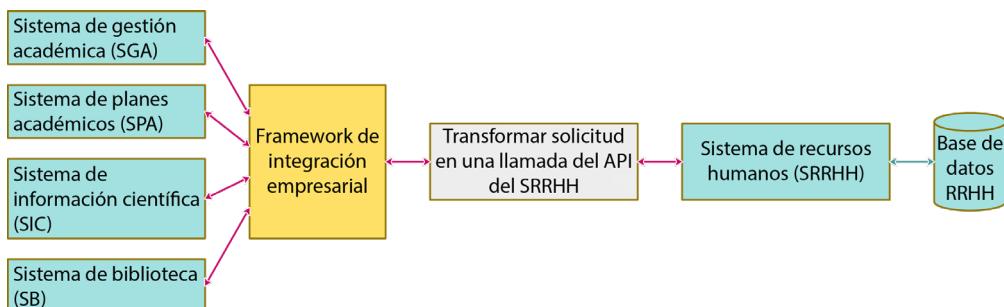


En la figura 3, cada sistema que necesite de la información de recursos humanos debe desarrollar una funcionalidad que consuma, haga llamadas a la interfaz de programación de aplicaciones (API) que ese sistema publica. Los sistemas de la parte izquierda deben conocer de forma anticipada la estructura de los datos que recibirá para así poder “mapearlos” con cada uno de sus esquemas.

Así como se hizo con la primera categoría, se presenta un inconveniente de esta categoría. Suponga que existe un cambio del sistema de recursos humanos, esto provoca que todos los sistemas que usan el API del sistema anterior deban actualizarse, lo que implica el consumo de recursos.

Figura 4.

Ejemplo de una sistema débilmente acoplado



Finalmente, en la Figura 4, aquí la comunicación entre los sistemas de la izquierda (consumidores) y el de la derecha (productor) no se hace directamente, sino que se utiliza un sistema intermediario, como un bus de servicios empresariales quien es el encargado de comunicar ambas partes liberando a los sistemas consumidores del desarrollo de funcionalidades específicas.

En este último ejemplo, si el sistema de recursos humanos cambia su API de consulta, el cambio únicamente se hace en el framework de integración, los otros sistemas seguirán funcionando sin verse afectados por los cambios, es más, los cambios pasaran desapercibidos.

Los frameworks de integración empresarial han evolucionado acorde con el avance de la tecnología, las primeras versiones tenían que lidiar con la diversidad que iba desde los protocolos de comunicación hasta los formatos de los datos que enviaban o eran capaces de recibir, un ejemplo lo podemos encontrar en Anstey y Ibsen (2018) en donde mencionan que la herramienta de integración denominada Camel soporta alrededor de 280 protocolos y tipos de datos.

Actualmente, gracias al desarrollo de las arquitecturas orientadas a servicios, Service Oriented Architecture (SOA), la diversidad de protocolos y tipos de datos se han reducido, ya que de forma generalizada se usa el protocolo TCP/IP para la comunicación y los datos se transmiten en formatos como XML y JSON.

Antes de finalizar, es conveniente señalar que, como menciona Kanade, 2019), generalmente las aplicaciones que se pretende integrar son aplicaciones con diferentes entornos de ejecución escritas en diferentes

lenguaje de programación y que operan bajo diferentes políticas. Estas características hacen que la integración sea una actividad compleja.

Una vez definida la integración de sistemas debe estudiar los tipos de integración que existen, este tema se desarrolla en el siguiente apartado.

1.5. Tipos de arquitecturas para integración de sistemas

De cara al usuario final, el resultado de la integración de sistemas es contar con un único sistema de información, es decir un único sistema en donde se encuentran todos los servicios (funcionalidades) que necesita y un único punto donde consultar toda la información que requiere para realizar su trabajo.

Este punto de vista se puede utilizar para clasificar a la integración de sistemas en cuatro grupos, así:

- Integración centrada en las aplicaciones.
- Integración centrada en los datos.
- Integración de procesos de negocio.
- Integración de presentación.

Es necesario mencionar que, dentro de la integración, muchos autores señalan que existe un quinto tipo, la integración Business-to-Business o B2B, que consiste en la integración de dos organizaciones, es decir una integración interorganizacional, pero esta integración se podría lograr con alguno de los tipos mencionados anteriormente, es por ello que queda fuera como tema de estudio, aunque más adelante se la mencionará, especialmente cuando se estudie la integración centrada en aplicaciones.

Para la asignatura los dos primeros tipos de integración son los más relevantes, es por eso que centrará su atención en ellos. A continuación, se dan algunos detalles de cada tipo de integración.

Dentro de la integración centrada en las aplicaciones, la integración se puede hacer para suministrar un único punto de acceso a las funcionalidades o con el fin de brindar servicios más completos (cooperación de funcionalidades) que cumplan con los requisitos de una empresa.

Antes de continuar, le invito a reflexionar sobre el desarrollo actual de aplicaciones. En estos días es muy común que las empresas utilicen servicios (funcionalidades) de otras empresas, por ejemplo: una plataforma de comercio electrónico utiliza una pasarela de pago para transacciones con tarjetas de crédito.

Tomando en cuenta el punto anterior, la integración centrada en aplicaciones puede ser entre aplicaciones a lo interno de la organización o entre aplicaciones que pertenecen a diferentes organizaciones. Aquí cabe el concepto B2B mencionado anteriormente.

Es momento de revisar la integración de datos. Este tipo de integración se refiere al acceso y uso de la información a través de un único punto de entrada permitiendo el desarrollo de soluciones OLAP (procesamiento analítico en línea), inteligencia de negocio o minería de datos, ya que aplicaciones de este estilo pueden usar los datos integrados para realizar su trabajo. En resumen, todas aquellas actividades que están relacionadas con toma de decisiones y la planificación se verán beneficiadas al contar con información integrada.

Ha concluido el desarrollo de los contenidos planificados para esta semana que tenían por objetivo presentar el campo de estudio de la asignatura. Para la siguiente semana estudiará la comunicación entre sistemas distribuidos.



Actividades de aprendizaje recomendadas

Una vez concluidos los contenidos planificados, se le recomienda realizar la instalación del entorno de desarrollo que se usará durante todo el bimestre. La guía de instalación está disponible en [Itinerario 2: Programación Integrativa](#) en el apartado Entorno de desarrollo.



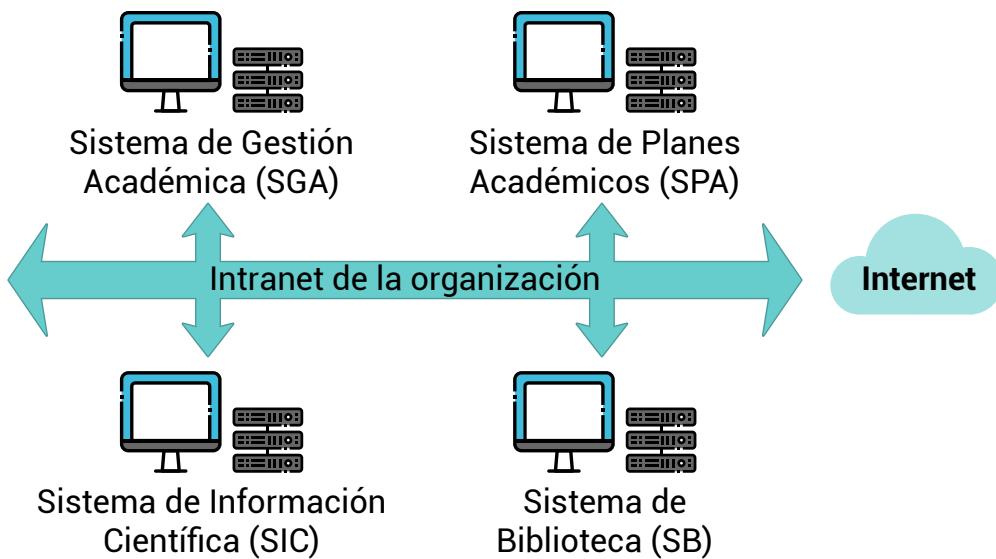
Semana 2

La semana anterior fue la primera de trabajo y tuvo como finalidad introducir los conceptos de integración de sistemas, además, se presentó a los sistemas de información empresarial (SIE) como uno de los principales promotores de la integración.

Para el desarrollo de los contenidos de esta semana, se propone el estudio del escenario que se describe a continuación.

Anteriormente se planteó un problema en donde una institución posee varios sistemas transaccionales. Adicionalmente, la institución posee una red de comunicaciones a la que se conectan todos los sistemas e inclusive les permite contar con una conexión a Internet. En la Figura 5 puede ver una representación de este esquema de trabajo.

Figura 5.
Sistemas conectados a la red



Suponga que el sistema de biblioteca posee una funcionalidad que permite verificar si una obra se encuentra disponible en el catálogo de libros físicos y/o digitales que posee la institución. Esta funcionalidad es utilizada por los sistemas de planes académicos y de información científica. En el primero, debido a que todos los planes académicos que se elaboran deben utilizar libros que se encuentran en el catálogo de biblioteca. Mientras que el segundo lo utiliza para verificar que la producción científica (*papers*) escrita por los docentes se encuentre publicada.

Con la información expuesta se plantea la siguiente pregunta, ¿cómo podrían los sistemas utilizar funcionalidades de otros sistemas?

Comience a desarrollar los contenidos teóricos que le ayudarán a dar respuesta a esta pregunta.

1.6. Sistemas distribuidos

En el escenario propuesto, de cara al docente, usuario de los sistemas de planes académicos y de información científica, se pretende dar la impresión de que se trata de un único sistema, aunque internamente existan más de uno funcionando de forma conjunta. A esto se lo conoce como un sistema distribuido.

Inicie el estudio de los conceptos estudiando la definición de programación y sistemas distribuidos, para ello debe realizar la siguiente lectura.

“1. Introducción” del texto base (Cardador, 2015), realice una lectura comprensiva hasta el tema “Características generales de las arquitecturas de servicios distribuidos”.

¿Comprendió las definiciones propuestas? Estoy seguro de que así fue. Complementando lo anterior, para que el usuario perciba que se trata de un único sistema, es necesario que los sistemas se integren para proporcionar cierto tipo de funcionamiento a los usuarios, en este caso concreto que integren su funcionamiento. Sin esta integración el usuario seguirá viendo dos sistemas independientes a los cuales debe acceder por separado para realizar su trabajo.

Antes de continuar con la lectura, es necesario comentar que el término RPC se estudiará más adelante, así que por el momento no se entrará en detalles.

Es momento de continuar con la lectura, ahora debe leer el tema “2. Características generales de las arquitecturas de servicios distribuidos” del texto base (Cardador, 2015). Trate de hacer una lectura rápida de este tema, ya que seguramente lo estudió en la asignatura de Sistemas Distribuidos.

Si bien son 4 características, tratar de alcanzar cada una ha provocado un gran esfuerzo y mucho trabajo, consecuencia de lo anterior es la gran cantidad de tecnologías que se han desarrollado y que giran alrededor de esas características. A manera de ejemplo, considere las ideas que giran alrededor de la computación paralela o paralelismo, existen varios modelos: basados en tareas, basados en actores, síncrona, asíncrona, etc., por mencionar algunas. Dentro del tema de la compartición de recursos aparecen modelos como cluster, grid o cloud.

Con lo anterior, de seguro tiene claro que el mundo de los sistemas distribuidos es basto y está en continuo desarrollo, puede empezar a sentirse abrumado con todo lo que se necesita conocer, pero no se preocupe, ya que no es necesario más detalles, recuerde que el propósito de esta asignatura es la integración.

Un tema clave tanto para los sistemas distribuidos como para los procesos de integración es la comunicación entre sistemas. Comprender cómo los sistemas se comunican, más allá de la idea de una red de computadores, es un tema importante ya que de ahí se derivan muchas otras formas diferentes de comunicación, pero que subyacen en un mismo modelo.

Para conocer este tema tan importante debe revisar el siguiente apartado. Recuerde hacer una lectura comprensiva del tema.

1.7. Comunicación entre sistemas

Una aplicación o sistema lleva a cabo varios tareas o procesos para cumplir con su propósito, en muchos casos estos procesos deben comunicarse entre sí con el fin de entregar datos para que ejecute alguna o más acciones; o recibir datos fruto de la ejecución de alguna funcionalidad de otro proceso. Si se trata de una aplicación en la que todos sus procesos se ejecutan en un mismo equipo (computador), la comunicación es sencilla, por ejemplo, se puede definir un espacio de memoria (buffer) compartido en donde los procesos pueden ir a entregar o buscar los datos que necesitan.

En una aplicación distribuida sus procesos se ejecutan en diferentes equipos (computadores), los mecanismos de comunicación no son tan sencillos y al no existir una memoria compartida es necesario utilizar otro tipo de mecanismo que permita la comunicación entre equipos.

Dentro de los sistemas distribuidos uno de los factores que ayuda a determinar el mecanismo de comunicación entre los diferentes equipos es el tipo de red que los conecta. Es así que, si el sistema distribuido está conectado por una red área local (LAN-Local Area Network) se puede optar por un modelo de comunicación en donde existe un cliente que envía un mensaje al servidor y este último envía de regreso una respuesta al cliente. Un ejemplo de esto mecanismo se presenta cuando se trabaja con un motor de base de datos conectado en red, los clientes serían todos los equipos

que necesitan interactuar con la base de datos y el servidor sería la base de datos, de hecho es muy común hablar del servidor de base de datos.

Otra alternativa muy empleada, dentro de una LAN, es el uso de llamadas a procedimientos remotos o RPC por sus siglas en inglés (Remote Procedure Call). Un ejemplo de este mecanismo, aunque llevado al mundo de la orientación a objetos, concretamente al lenguaje de programación Java, son los Enterprise Java Beans (EJB) que sintetizando su funcionamiento se puede decir lo siguiente: permiten tener un servidor de objetos remotos, es decir, se puede acceder a un objeto que fue creado por y existe en un servidor, y que está disponible para los clientes que lo necesiten.

En cambio, si el sistema distribuido está conectado por una red de área amplia (WAN) el mecanismo de comunicación utiliza protocolos de capas orientados hacia la conexión como OSI y TCP/IP, ya que se necesita un medio de transporte fiable que garantice la comunicación en lugar de, por ejemplo, su velocidad.

Independientemente del tipo de red, los sistemas distribuidos utilizan un sistema de mensajes (paso de mensajes) para la comunicación entre procesos (equipos), es decir, a través de la red de comunicación se transfieren mensajes. Estos mensajes pueden variar en forma y estructura, según la tecnología que se utiliza, pero su propósito se mantiene, el intercambio de datos.

Es momento de revisar este sistema de comunicación que también es conocido como un modelo de distribución. En los siguientes párrafos se describen los principales conceptos del paso de mensajes.

1.7.1. Paso de mensajes

El paso de mensajes o message passing es el paradigma más básico de comunicación dentro de las aplicaciones distribuidas, su uso es masivo, tanto así que el sistema distribuido más grande que actualmente existe, la Web, lo utiliza; resumiendo el funcionamiento de la Web se puede decir que un cliente, como un navegador Web, se comunica con un servidor de páginas Web a través del paso de mensajes. Le invito a profundizar su conocimiento sobre este tema.

Para comprender este mecanismo de comunicación es necesario que realice una lectura del tema 3.1.2 Comunicación entre aplicaciones de Montes (2015).

De seguro notó que la comunicación entre sistemas, a pesar de ser un tema bastante técnico, se basa en el proceso de comunicación entre seres humanos que probablemente ya lo estudió. Sin embargo, aparecen otros conceptos como paquete y protocolo. Posiblemente al concepto protocolo es necesario ampliarlo.

Para ampliar el concepto de protocolo le propongo la siguiente pregunta, ¿cuáles son los protocolos que existen para el paso de mensajes? Para responder a esta pregunta es necesario que revise el recurso de Montes (2015), esta vez en el punto 3.2 (lea hasta el tema Funcionamiento de la pila de protocolos).

De seguro encontró la respuesta a la pregunta propuesta anteriormente, recuerde que los protocolos que son de nuestro interés son aquellos que se encuentran a nivel de transporte, es decir, TCP y UDP, ya que las primeras comunicaciones entre sistemas las hará utilizando uno de esos protocolos.

Es momento de conocer el detalle del protocolo de control de transmisión o Transmission Control Protocol (TCP), ponga especial atención en las principales características de ese protocolo. Los detalles los encontrará con lectura del tema Protocolo TCP en Montes (2015).

Le recomiendo que tome nota del algoritmo que usa el protocolo para transmitir mensajes debido a que lo aplicará cuando construya programas que comuniquen dos sistemas.

Una vez revisado el protocolo TCP, es necesario avanzar en el desarrollo de los contenidos y revisar brevemente el protocolo UDP, también conocido como User Datagram Protocol o protocolo de datagramas de usuario. Continúe con la lectura del tema Protocolo UDP que propone Montes (2015).

Recuerde que las principales diferencias entre ambos protocolos son: a) la fiabilidad: TCP garantiza la entrega de los mensajes en el orden que fueron enviados; y b) la orientación o no a conexión: TCP es orientado a conexión, mientras que UDP no.

Existe una tendencia a pensar que el protocolo UDP es poco útil y se encuentra únicamente en aplicaciones antiguas, esto se debe a sus

características que algunos las ven como limitantes y al uso masivo de TCP como protocolo para muchos de los servicios que se ofrecen en Internet, pero algunos sistemas actuales usan UDP como se resume en la Tabla 1.

Tabla 1.
Aplicaciones de los protocolos TCP y UDP

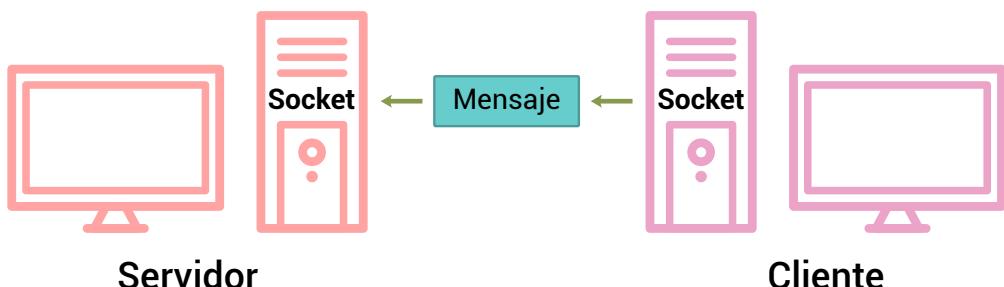
TCP	UDP
Web.	Tunneling/VPN.
SSH, FTP, telnet (administración remota, transferencia de archivos, conexiones remotas respectivamente).	Voz sobre IP.
SMTP (envío de correos).	Media streaming.
IMAP/POP (recibir correos).	Juegos.

Ahora que ya conoce los protocolos que se pueden utilizar para la comunicación, es necesario avanzar y estudiar una interfaz de programación de aplicaciones (API) sencilla que permite la comunicación entre diferentes aplicaciones que son parte de un sistema distribuido.

1.7.2. Sockets

Según la Figura 6, existen otros dos elementos que intervienen en la comunicación entre aplicaciones que se denominan sockets. Observe como existe un socket en cada equipo que interviene, es decir existirá un socket del cliente y otro del servidor.

Figura 6.
Comunicación entre aplicaciones



Para comprender en detalle los sockets es necesario que revise el siguiente recurso Montes (2015), para este tema es necesario que haga una lectura desde el tema 3.3., Sockets, hasta el tema 3.3.2., Programación con sockets.

Una vez hecha la lectura, es necesario destacar algunos puntos que son importantes cuando se implementa la comunicación entre aplicaciones a través de paso de mensajes:

1. De seguro notó cómo por cada tipo de protocolo TCP y UDP existe un tipo de socket, stream y datagram respectivamente.
2. Independientemente del tipo de sockets, siempre deberá programar dos sockets, uno para el cliente y otro para el servidor.
3. Según el tipo de protocolo, el algoritmo para la comunicación, tanto para el cliente como para el servidor, cambia.

Es momento de avanzar y pasar a la práctica, a la programación, en el siguiente apartado se describe cómo implementar sockets utilizando el lenguaje de programación Java.

1.7.3. Programación con sockets

En el apartado anterior estudió el concepto de sockets y revisó los algoritmos que se necesitan para programar sockets TCP y UDP. En este apartado revisará estos mismos conceptos, pero esta vez en el contexto de un lenguaje de programación.

Para comenzar, debe revisar el recurso Montes (2015) desde el tema Programación con sockets. Una recomendación, cuando se encuentre revisando el código trate de ir asociando las sentencias con los pasos que se describen en cada algoritmo.

¿Cómo estuvo la lectura? Me imagino que la encontró enriquecedora. ¿Observó que existen clases para el socket del servidor y del cliente? Estoy seguro de que así fue, si no, le invito a revisar nuevamente y fíjese en la descripción que se hace de las clases `socket` y `ServerSocket`.

Es momento de probar el código, para ello utilice su entorno de desarrollo y escriba el código que se encuentra en los ejemplos 3.2. y 3.3. También los puede encontrar en [Itinerario 2: Programación Integrativa](#) (ver directorio `sockets`).

Recuerde que el orden correcto de ejecución de los programas demanda que primero se ejecute el servidor y luego el cliente, clases `ServidorSocketStream` y `ClienteSocketStream` respectivamente.

A continuación, revise una tabla que integra el código presentado en el recurso con los algoritmos estudiados anteriormente. Inicie la revisión con los sockets tipo TCP tanto para el cliente (Tabla 2) y el servidor (Tabla 3).

Tabla 2.

Algoritmo y código para un cliente TCP

Paso	Código
Paso 1. Creación del socket.	Socket clienteSocket = new Socket();
Paso 2. Conexión con el servidor.	clienteSocket.connect(address); (address es un objeto que indica la dirección y puerto del socket en el servidor).
Paso 3. Envío de mensajes.	outStream.write(mensaje.getBytes()); (outStream es un objeto que permite enviar un flujo de bytes desde el cliente hacia el servidor).
Paso 4. Cierre de conexión.	clienteSocket.close();

Tabla 3.

Algoritmo y código para un servidor TCP

Paso	Código
Paso 1. Creación del socket.	ServerSocket serverSocket = new ServerSocket();
Paso 2. Asociar dirección y puerto.	serverSocket.bind(addr); (addr es un objeto que señala la dirección y puerto asignados al servidor).
Paso 3. Escuchar.	Las 2 sentencias anteriores llevan al socket a este estado.
Paso 4. Aceptar conexiones.	Socket clienteConexion = serverSocket.accept();
Paso 5. Recepción de mensajes.	inStream.read(mensaje); (inStream es un objeto que recibe los bytes que llegan al SeverSocket).
Paso 6. Cierre de conexión con el cliente.	clienteConexion.close();

Los programas que se utilizan en el libro son sencillos y utilizan un mecanismo de comunicación basado en bytes, es por ello que usan las clases abstractas `InputStream` y `OutputStream`, pero existen otras clases en las que se puede enviar/recibir mensajes que sean números o texto, esto lo verá más adelante. Además, es necesario señalar que dentro de la comunicación entre computadoras es posible que existan problemas, como por ejemplo, no se puede establecer comunicación con el servidor o con el cliente, es por esto que es necesario manejar excepciones del tipo `IOException` usando bloques `try/catch`.

Antes de ver la tabla que corresponde al protocolo UDP, es necesario hacer las siguientes puntualizaciones:

- Dentro del protocolo UDP no existe diferencia entre cliente y servidor, por lo que el algoritmo es el mismo tanto para el que envía como para el que recibe.
- Un emisor se puede convertir en receptor y un receptor en emisor.
- Los mensajes se transportan en paquetes que en Java son conocidos como DatagramPacket.
- Si el emisor quiere recibir un mensaje de vuelta del receptor, en el paquete debe incluir su dirección y puerto.

Si bien el código del emisor y del receptor lucen diferentes, en esencia es el mismo, su principal diferencia se da a nivel de sintaxis y el uso de diferentes métodos para realizar las mismas tareas. Con esto en mente revise la Tabla 4 que asocia el algoritmo para sockets UDP con el código tanto del cliente como del servidor.

Tabla 4.
Algoritmo y código para el protocolo UDP

Paso	Código
Paso 1. Creación del socket.	DatagramSocket datagramSocket = new DatagramSocket();
Paso 2. Asociación de dirección y puerto.	DatagramSocket datagramSocket = new DatagramSocket(address); DatagramPacket datagram = new DatagramPacket(mensaje.getBytes(),mensaje.getLength(),address, 5555);
Paso 3. Envío y recepción de mensajes.	datagramSocket.send(datagram); datagramSocket.receive(datagram);
Paso 4. Cierre de la conexión.	datagramSocket.close();

Los sockets son la base para cualquier otra forma de comunicación que se base en el paso de mensajes. Sobre ellos se han construido y se siguen construyendo otros mecanismos de comunicación en los que se necesite conectar equipos a través de una red de comunicación, todos estos mecanismos son abstracciones de mayor nivel que los sockets, es decir lo han tomado como base y han agregado otras características que permiten que el programador no se preocupe de tareas de “bajo nivel”.

Un ejemplo de una abstracción superior a los sockets es la llamada a procedimientos remotos o RPC por sus siglas en Inglés (Remote Procedure Call) en donde, de manera virtual, se oculta la red de comunicaciones y se aparenta una comunicación directa como si estuviera invocando a un procedimiento local y no a uno que se encuentra en otro equipo.

Muchas de las abstracciones posteriores a los sockets han cambiado la estructura del mensaje que se intercambia. Para comprender de mejor manera este tema es necesario avanzar y proponer un nuevo apartado.

1.7.4. Evolución de la comunicación

Anteriormente se explicó que el paso de mensajes es una de las estrategias más utilizadas cuando de comunicar sistemas distribuidos se trata. Además, estudió de forma teórica y práctica una implementación de este mecanismo de comunicación a través de sockets.

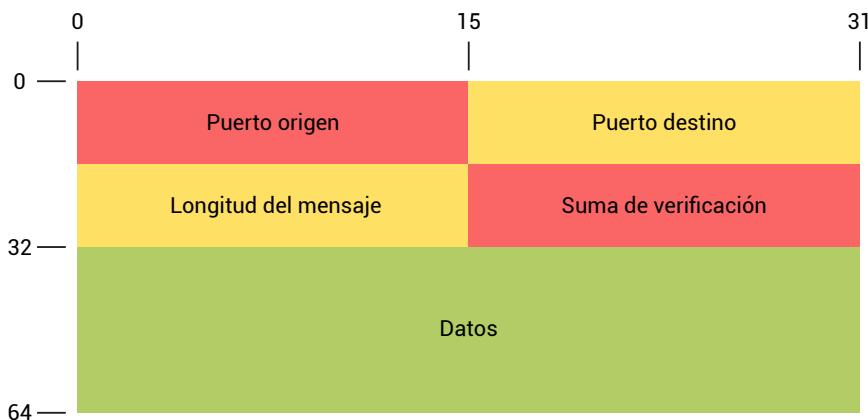
Hasta el momento, poco se dijo sobre el formato de los mensajes que se intercambian entre equipos dentro de un sistema distribuido. Obviamente, a través de una red de comunicaciones viajan bits, pero estos bits son organizados de cierta manera que responden al protocolo que se utiliza TCP o UDP.

En los siguientes párrafos estudiará superficialmente la estructura de los mensajes UDP y TCP para luego revisar la estructura de otros mensajes como los que se utilizan en otros protocolos, como en el protocolo de transferencia de hipertexto o HTTP que viene a constituirse en una abstracción de nivel superior a los sockets. Por último revisará la estructura de un mensaje SOAP (Simple Object Access Protocol).

Dentro del protocolo UDP un datagrama está formado por dos secciones, una de cabecera y otra de datos. En la cabecera existen 4 campos, 2 mandatarios (puerto de destino y longitud del mensaje) y 2 opcionales (puerto de origen y suma de verificación). Una representación visual se puede ver en la Figura 7.

Figura 7.

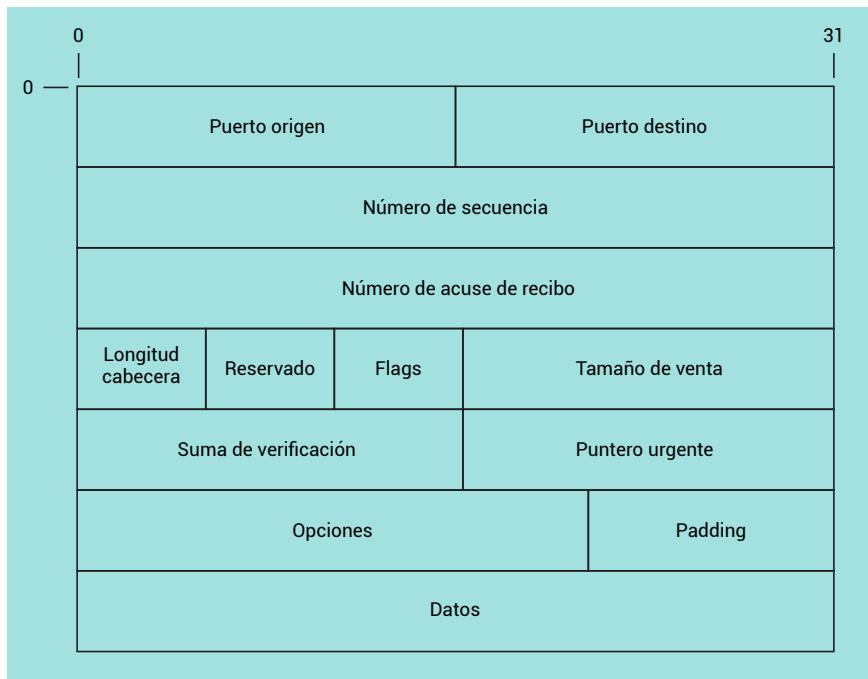
Estructura de un mensaje UDP



La estructura de un paquete TCP es de mayor complejidad que la del datagrama UDP, recuerde que se trata de un protocolo orientada a conexión en donde se garantiza la entrega de todos los paquetes, esas características ya le agregan varias secciones a la estructura del mensaje, esto se puede ver en la Figura 8.

Figura 8.

Estructura de un mensaje TCP



No es de interés para la asignatura profundizar en el estudio de la estructura de los mensajes, es por ello que no se dará más detalles y se le invita a utilizar los gráficos anteriores como punto de referencia que le permita comparar la evolución de los mensajes que se intercambian en sistemas distribuidos que usan un mecanismo diferente de comunicación al que usan los sockets.

Por encima del protocolo TCP/IP, que es en donde se utiliza los sockets, se encuentra por ejemplo el protocolo HTTP que es el protocolo que permite que uno de los servicios que brinda el Internet funcione. ¿Conoce usted el nombre de ese servicio?

El protocolo de transferencia de hipertexto, en inglés Hypertext Transfer Protocol, abreviado HTTP, permite que la Web funcione, es decir, es un protocolo que permite transferir información entre su navegador y un servidor web. El ejemplo anterior es posiblemente el más conocido, aunque reduce en exceso las capacidades y características de la Web.

El protocolo HTTP se basa en el paso de mensajes y funciona con dos tipos de mensajes, el primero es un mensaje de solicitud o *request* (el que envía el cliente al servidor) y el segundo, un mensaje de respuesta o *response* (el que envía el servidor al cliente). Revise ahora la estructura de estos mensajes, inicie por el mensaje de solicitud que se ve en la Figura 9.

Figura 9.

Mensaje de solicitud Request dentro de HTTP

GET /index.html HTTP/1.1	→ Línea de solicitud
User-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X) Host: j4loxa.com Accept: text/html,application/xhtml+xml,application/xml;	→ Encabezado de la solicitud
?param1=value1¶m2=value2	→ Cuerpo de la solicitud

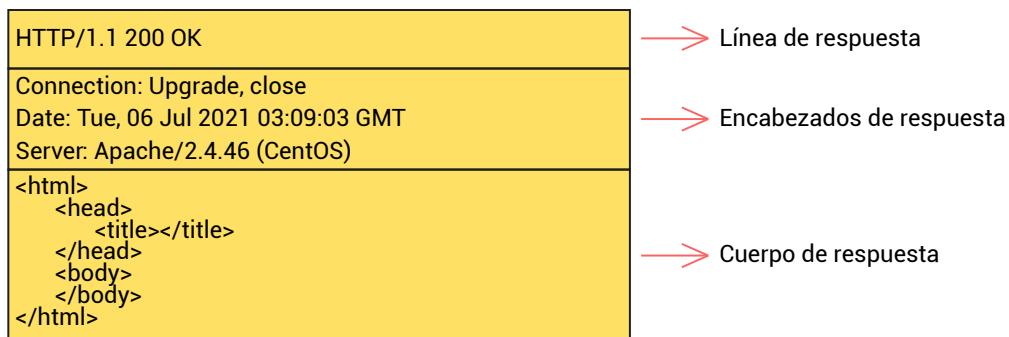
Como puede ver, este tipo de mensajes son muchos más elaborados que los podemos encontrar en un socket TCP o UDP. De forma general, este tipo de mensajes se divide en tres secciones, cada una de las secciones aportan información específica que se relaciona con la petición, es así que la primera señala qué recurso se está solicitando, la segunda son cabeceras (campos de información) que señalan en dónde está el recurso que se solicita e

identifican el cliente que hace la petición. La última sección se utiliza para enviar los datos al servidor.

Ahora revise un mensaje de respuesta, ver la Figura 10. Al igual que los de petición, tiene tres secciones cada una con información relacionada con la respuesta, la primera parte incluye información sobre el resultado de la petición. La segunda son cabeceras de respuesta, por ejemplo se identifica el servidor. La última incluye el cuerpo de la respuesta que puede ser el contenido (código html) de una página Web.

Figura 10.

Estructura del mensaje de respuesta Response de HTTP



Hasta la versión 1.1 del protocolo HTTP los mensajes se escribían en formato texto (como los que muestran las imágenes), pero esto cambió en la versión 2 en donde se utiliza un formato binario, aunque la semántica de los mensajes se mantienen.

Existen más detalles que analizar, pero no es el momento de hacerlo, retomará este tema más adelante cuando estudie los servicios Web. En los siguientes párrafos estudiará un nuevo formato de mensaje que se utiliza en la construcción de un tipo de servicios Web.

Dentro de la evolución del sistema de comunicación basado en mensajes se encuentran los mensajes que intercambian los servicios Web que se basan en el SOAP o Simple Object Access Protocol. Este tipo de mensajes se escriben utilizando el Lenguaje de Marcas Extendido XML.

Gráficamente los mensajes SOAP se pueden ver en los siguientes recuadros. Tanto los mensajes de petición como los de respuesta tienen la misma estructura formada por tres secciones, aunque en las figuras anteriores únicamente se visualizan dos. La primera, *Envelope* que describe el

contenido y forma que tiene que ser procesado el mensaje. Una segunda sección denominada de cabecera que identifica el contenido del mensaje. Y la última sección el cuerpo del mensaje que representa las llamas o respuestas a los servicios que ofrecen otros equipos.

Formato mensaje XML de solicitud

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/">
    <soap:Body>
        <getCourseDetails xmlns="http://utpl.edu.ec/ws">
            <courseld>827635</courseld>
        </getCourseDetails>
    </soap:Body>
</soap:Envelope>
```

Formato de un mensaje XML de respuesta

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/">
    <soap:Body>
        <getCourseDetailsResponse xmlns="http://utpl.edu.
ec/ws">
            <getPourseDetailsResult>
                <coursetName>Java apuntes
básicos</coursetName>
                <courseld>827635</courseld>
                <description>Conceptos básicos
sobre Java</description>
                <price>0</price>
                <active>true</active >
            </getourseDetailtDetailsResult>
        </getCourseDetailsResponse>
    </soap:Body>
</soap:Envelope>
```

En este caso, los mensajes se transportan a través del protocolo HTTP, es decir, se construyó un sistema de mensajes sobre otro, algo parecido a lo que hicieron con HTTP y los sockets.

No olvide que aunque existen mensajes HTTP, SOAP, etc., la comunicación entre equipos es a través del protocolo TCP/IP, es decir, la mayoría de las telecomunicaciones actuales continuarán utilizando sockets como el mecanismo de intercambio de mensajes.

Antes de finalizar, es necesario dar una respuesta al problema propuesto al inicio de este apartado, cómo obtener información del sistema de biblioteca. Para resolver este problema se utilizará comunicación a través de sockets, es decir, en el sistema de la biblioteca se creará un socket de tipo servidor y los otros sistemas que necesitan consultar los datos deben crear sockets cliente. Los clientes enviarán un código (ISBN o DOI) al servidor, quien consultará en su base de datos y devolverá verdadero (true) si el código existe y falso (false) en caso contrario.

Una vez hecha la explicación general del funcionamiento del algoritmo, es momento de presentar y explicar el código que se utilizó.

Inicie la revisión del código con la implementación del socket del servidor. Para mejorar la comprensión del código se lo presentará y analizará por los métodos que lo forman. El primer método por revisar será el *main* o también conocido como principal.

```
public static void main(String[] args) {  
    BibliotecaServer bibServer = new BibliotecaServer();  
    bibServer.start(5555);  
}
```

En el código anterior se muestra la creación de un objeto y la invocación al método *start*, el mismo que recibe, como parámetro, el número del puerto que se usará para levantar el socket de tipo servidor. El constructor que se utiliza es el constructor por defecto y no ejecuta ninguna sentencia, por lo que el siguiente método a analizar es el método *start*.

```
public void start(int port) {  
    InetSocketAddress addr = new InetSocketAddress("localhost",  
    port);
```

```

try {
    serverSocket = new ServerSocket();
    serverSocket.bind(addr);
    while (true) {
        handlerRequest(serverSocket.accept());
    }
} catch (IOException e) {
    e.printStackTrace();
    LOG.log(Level.SEVERE, e.getMessage());
}
}

```

Si compara este código con los anteriores, notará que aquí se agregó un ciclo repetitivo que invoca al método *handlerRequest*. De esta manera, el servidor siempre estará activo (ejecutándose) y cada cliente que se conecta (socket cliente) ejecutará el mencionado método.

Revise ahora el método *handlerRequest* que, al igual que el anterior, tiene varios elementos que lo diferencian de versiones más sencillas que ya revisó.

```

private void handlerRequest(Socket clientSocket) throws
IOException {
    LOG.log(Level.INFO, "Ready to query");
    DataOutputStream out = new
        DataOutputStream(clientSocket.getOutputStream());
    DataInputStream in =
        new DataInputStream(clientSocket.
    getInputStream());
    while (clientSocket.isConnected() && !clientSocket.isClosed()
    ) {
        try {
            String code = in.readUTF();
            LOG.log(Level.INFO, "code: " + code);
            out.writeBoolean(Repository.exists(code.
    toUpperCase()));
        } catch (EOFException eofex) {
            LOG.log(Level.INFO, "End message");
        }
    }
}

```

```

        break;
    }
}

in.close();
out.close();
clientSocket.close();
LOG.log(Level.INFO, "Closing everything");
}

```

En este método se utilizan las objetos de las clases *DataOutputStream* y *DataInputStream* que son clases que permiten leer y escribir datos de cualquier tipo de forma directa, sin convertirlos en bytes como lo vio anteriormente. Observe los métodos *readUTF* (para leer un *String*) y *writeBoolean* (para leer un valor lógico) que son ejemplo de lo mencionado.

Además, para terminar la ejecución del método se usa dos estrategias, la primera es la condición del *while* que verifica si el socket cliente está conectado y abierto; y la segunda estrategia es capturar la excepción del tipo *EOFException* que se produce cuando el cliente deja de enviar datos y es el momento adecuado para poner fin a la ejecución del ciclo repetitivo, en este caso usando la sentencia *break*.

Ese sería el código que debe ejecutarse como servidor, es necesario mencionar que esa clase debería tener acceso a los datos que maneja la biblioteca. Ese acceso se representa con la clase *Repository* y el método *exists*.

Para continuar con la revisión del código, es necesario revisar la otra parte, el socket cliente. Se aplicará la misma estrategia, revisar cada uno de los métodos que la conforman.

El primer método es el principal, cuyo código es el siguiente:

```

public static void main(String[] args) {
    SPAClient client = new SPAClient();
    client.startConnection("localhost", 5555);
    try {
        System.out.printf("El código ABC existe? %b\n",client.
verifyCode("ABC"));
    }
}

```

```

        System.out.printf("El código DEF existe? %b\n",client.
verifyCode("DEF"));

        System.out.printf("El código A1B existe? %b\n",client.
verifyCode("A1B"));

        client.stop();

    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}

```

Este es el punto de partida para la ejecución de todo el código del cliente, aquí se puede observar la creación de un objeto del tipo *SPAClient*, la llamada al método *startConnection* (como su nombre lo indica inicia la conexión con el servidor), varias invocaciones al método *verifyCode* que sería el método que envía el código que se necesitar verificar al servidor y, finalmente, se cierra el socket con el método *stop*.

Revise entonces la clase *SPAClient* para conocer su funcionamiento, comience con el código del método *startConnection*.

```

public void startConnection(String ip, int port) {
    try {
        clientSocket = new Socket(ip, port);
        in = new DataInputStream(clientSocket.
getInputStream());
        out = new DataOutputStream(clientSocket.
getOutputStream());
    } catch (IOException e) {
        LOG.log(Level.SEVERE, e.getMessage());
    }
    LOG.log(Level.INFO, "Connected");
}

```

Ese método recibe como parámetros la dirección IP y puerto donde se encuentra ejecutándose el servidor, lo siguiente que hace es crear un objeto del tipo socket y crear los flujos de entrada *DataInputStream* y salida *DataOutputStream* que, al igual que en el servidor, utiliza flujos de un nivel superior con el fin de evitar el trabajo directo con bytes. De seguro se

percató que el propósito de este método es de creación de todo lo necesario para el intercambio de datos.

El envío y recepción de los datos se hace en el método *verifyCode* que ha sido implementado así:

```
public boolean verifyCode(String code) throws IOException {  
    out.writeUTF(code);  
    return in.readBoolean();  
}
```

Pueden parecer muy pocas líneas, pero son las más importantes, la primera escribe (envía al servidor) el código a verificar y la segunda línea lee la respuesta que envía el servidor (un valor lógico).

Con las clases anteriores se resuelve el problema propuesto. Para ver el código completo debe revisar la siguiente página Web: [Itinerario 2: Programación Integrativa](#) (apartado semana 2), además del código completo encontrará otros recursos multimedia que le ayudarán a comprender de mejor manera el código.

De esta manera se termina esta semana en la cual se revisó, tanto teóricamente como de forma práctica, el mecanismo de comunicación básico para el intercambio de mensajes en un sistema distribuido. Lo que se viene es aún más interesante ya que lleva al paso de mensajes a otro nivel de abstracción que permite hacer tareas más complejas de forma más sencilla.



Actividades de aprendizaje recomendadas

Para esta semana las actividades recomendadas son de tipo práctico-experimental:

- Implementar el código que se encuentra en disponible en: [Itinerario 2: Programación Integrativa](#), en el directorio sockets.
- Revisar la descripción de la implementación del caso de estudio de esta semana disponible en [Itinerario 2: Programación Integrativa](#), apartado semana 2.

Ha concluido el desarrollo de los contenidos de la primera unidad y es momento de verificar los conocimientos adquiridos respondiendo a la siguiente autoevaluación.

A continuación, se presentan preguntas relacionadas con los contenidos desarrollados en la unidad 1. Esta autoevaluación le permitirá a usted reafirmar los conceptos estudiados.



Autoevaluación 1

En las siguientes preguntas deberá seleccionar verdadero o falso según corresponda:

1. () Un sistema transaccional es un sistema de información empresarial.
2. () Los problemas de integración se presentan en empresas u organizaciones que poseen al menos dos sistemas.
3. () La integración de sistemas transaccionales permite a las organizaciones ejecutar su proceso de negocio.
4. () Un sistema de información empresarial es un tipo de sistema informático.
5. () Los sistemas de información empresarial son sistemas que reciben como entrada los datos de registro de un usuario.
6. () Los sistemas transaccionales ayudan en la administración de la organización.
7. () Un ERP se encarga de facilitar el flujo de información para mejorar la comunicación entre áreas.
8. () Un CRM trabaja con los clientes de una empresa.
9. () La integración empresarial se centra en las tecnologías, datos y sistemas de información.
10. () La integración empresarial puede ser física, de aplicaciones y comercial.
11. () El proceso de comunicación que se utiliza en un sistema distribuido se basa en el intercambio de mensajes.

12. () Para crear un objeto de la clase ServerSocket, es necesario conocer la dirección IP y el puerto en donde se ejecutará el cliente.
13. () Dentro de una comunicación basada en sockets, el cliente necesita conocer su dirección IP y el puerto en donde está trabajando.
14. () Dentro de un sistema distribuido, cada uno de los sistemas que lo integran son independientes entre sí.
15. () Utilizar sockets, los mensajes entre cliente y servidor, son un flujo de bytes.
16. () Al usar sockets TCP se garantiza la recepción de los mensajes.
17. () Una característica de los sockets UDP es la rapidez en el envío/recepción de los mensajes.
18. () Al utilizar sockets UDP el código del cliente y del servidor son casi idénticos.
19. () Dentro del lenguaje de programación Java, los flujos de entrada se identifican porque generalmente en su nombre está el texto InputStream.
20. () La clase DataOutputStream es un flujo que únicamente trabaja con un flujo de bytes.

[Ir al solucionario](#)

Debido a su empeño y dedicación ha culminado esta unidad, no olvide que las respuestas a esta autoevaluación las encuentra al final de este documento. Es momento de continuar e iniciar el estudio de una nueva unidad.

Resultado de aprendizaje 2

- Describe cómo se utilizan los servicios web para integrar aplicaciones dispersas en una organización.

Contenidos, recursos y actividades de aprendizaje

El resultado de aprendizaje se alcanza a través del estudio teórico de los principales conceptos para luego aplicarlos a través de la creación de programas que permitan experimentar con situaciones bastante reales y que son comunes dentro del contexto laboral.



Semana 3

Unidad 2. Integración centrada en aplicaciones, una integración funcional

En la unidad anterior, uno de los temas que estudió fue la integración de sistemas de información y comprendió que es una necesidad, actual o futura, de todas las organizaciones. También revisó los tipos de integración y, brevemente, revisó dos de esos tipos, la integración centrada en aplicaciones y la centrada en los datos.

En esta nueva unidad ampliará el estudio de la integración centrada en aplicaciones o también conocida como integración funcional, y conocerá en detalle cómo se puede conseguir este tipo de integración.

2. Integración funcional

Antes de empezar con el desarrollo de los contenidos, es necesario repasar brevemente la situación actual de la mayoría de las empresas que Sawicki (2016) lo resumen así:

A lo largo de los años, las empresas han estado comprando software estándar y han desarrollado aplicaciones personalizadas con el objetivo de brindar soporte computacional para implementar y ejecutar sus procesos comerciales. Como resultado, no es difícil encontrar una empresa que tenga un ecosistema de software compuesto por aplicaciones que se crearon

utilizando diferentes tecnologías, modelos de datos, sistemas operativos y, en la mayoría de los casos, no fueron diseñados para intercambiar datos y compartir funcionalidades. Es común que, desde un punto de vista administrativo, dichos ecosistemas de software se consideren sistemas homogéneos y se espera que respalden los procesos comerciales de las empresas mediante el uso de datos y funciones actuales sin requerir demasiada inversión.

Una vez propuesto el problema, es necesario que empiece a estudiar la integración centrada en aplicaciones o conocida como integración funcional, con esto en mente lo invito a leer los siguientes párrafos.

La integración funcional es un tipo de integración que se enfoca en compartir funcionalidades entre sistemas, es decir, cada sistema comparte una parte de lo que sabe hacer, promoviendo así el reuso antes que un desarrollo desde 0 ahorrando tiempo y dinero.

El párrafo anterior intencionalmente es amplio y utiliza términos como funcionalidades y partes, pero varios autores coinciden que en este tipo de integración existen dos niveles de funcionalidades a compartir:

1. A nivel de métodos: se comparten métodos, es decir, funcionalidades concretas y, muchas veces, de propósito general que se usan dentro de un sistema y que otros sistemas pueden necesitar.
2. A nivel de interfaz de aplicación: también conocido como nivel de procesos; aquí lo que se comparte son implementaciones de la lógica de negocios, es decir, no se trata de compartir un método de propósito general, sino de compartir una funcionalidad que llame a la aplicación de un proceso de negocio completo.

Es probable que tenga algunas interrogantes sobre la clasificación anterior, por lo que le propongo los siguientes ejemplos: un método que permita validar un número de cédula corresponde al primer tipo; mientras que un método que genere una factura dado el número de cédula de un cliente y la orden de pedido es un ejemplo del segundo caso. El primer método puede ser clasificado de propósito general, muchos sistemas lo implementan, mientras que el segundo es propio para tipo de negocio, es decir, es una regla del negocio.

Si bien, ambos pueden ser vistos como métodos, pero el segundo se puede construir a partir de unir varios métodos y así crear una funcionalidad que responda a una tarea completa y propia dentro de una organización.

La implementación de cualquiera de los dos tipos de integración de funcionalidades se puede conseguir a través de llamadas interfaces de programación o API, por siglas en inglés, ya que estas hacen posible el acceso a la funcionalidad en un nivel programático sin el uso de una interfaz de usuario.

Hace años atrás los programadores no veían utilidad en las API, por lo tanto, la mayoría de las aplicaciones antiguas no las tienen, pero debido a la necesidad de integración entre sistemas eso cambió y hoy en día la gran mayoría de sistemas proporcionan servicios para que puedan ser utilizadas por otras aplicaciones, esos servicios se encuentran publicados como API.

Mediante el uso de API, es posible acceder a las funcionalidades de los sistemas, pero al principio no fue tan fácil, ya que las API publicadas por diferentes sistemas tenían diferentes formas para acceder a ellas, lo mismo sucedía con la tecnología que se debía emplear en usarlas, esto ha cambiado con la llegada de SOA, como lo verá más adelante.

Haga una pausa en su lectura y analice la importancia de esas interfaces de programación o API, especialmente en lo que a cambios se refiere y trate de responder a la pregunta, ¿es conveniente cambiar una API frecuentemente? Para responder a esta pregunta le invito a leer los siguientes párrafos.

Una API es similar a un contrato entre sistemas, en donde el sistema que publica el API se compromete a que, si recibe ciertos datos de entrada, generará una salida específica. Entonces, si los API permanecen sin cambios, los contratos no cambian, favoreciendo a la estabilidad de la integración. Es más, resulta más sencillo cambiar los sistemas que implementan una interfaz, sistemas que se ejecutan en segundo plano, que cambiar un contrato debido a que muchos sistemas pueden trabajar bajo ese contrato y un cambio provocaría que dejen de funcionar.

La integración funcional de aplicaciones se propone dos objetivos: 1) entender y usar las API para acceder a las funcionalidades necesarias y 2) enmascarar las diferencias tecnológicas entre las diferentes API.

Para llegar a conseguir esos objetivos han existido diferentes propuestas que han ido evolucionando con el desarrollo de la tecnología. Para conocer brevemente esa evolución le invito a revisar el siguiente apartado.

2.1. Codificación integrativa

La codificación integrativa se entiende como las diferentes técnicas o estrategias de programación que permiten que dos sistemas se integren, tanto a nivel de funcional como de datos.

En los siguientes párrafos se puede encontrar detalles que describen varias características de la codificación o programación integrativa.

La programación integrativa se ocupa tanto de las actividades y técnicas de integración que conectan diferentes componentes de la infraestructura de tecnologías de la información de una empresa, componentes como personas, aplicaciones, plataformas y bases de datos, con el objetivo de permitir una colaboración segura dentro de una aplicación, así como también entre aplicaciones.

Actualmente, la integración de aplicaciones no es únicamente puerta adentro, es así que la programación integrativa permite que una organización integre sus procesos con agentes externos creando así entornos dinámicos que atienden a los requisitos actuales y futuros.

Luego de la lectura anterior, de seguro tiene claro la importancia que la programación integrativa tiene actualmente, no solo porque permite a los sistemas cooperar entre sí (incluidos los legados), sino también porque abre la posibilidad de conectarse con sistemas externos que, sin lugar a duda, es un atributo necesario en un mundo hiperconectado como el actual.

Luego de definir a la codificación integrativa, el siguiente punto a conocer son las técnicas de programación integrativa. Es necesario mencionar que las técnicas que revisará son las relacionadas con las aplicaciones y giran alrededor de la comunicación entre sistemas distribuidos.

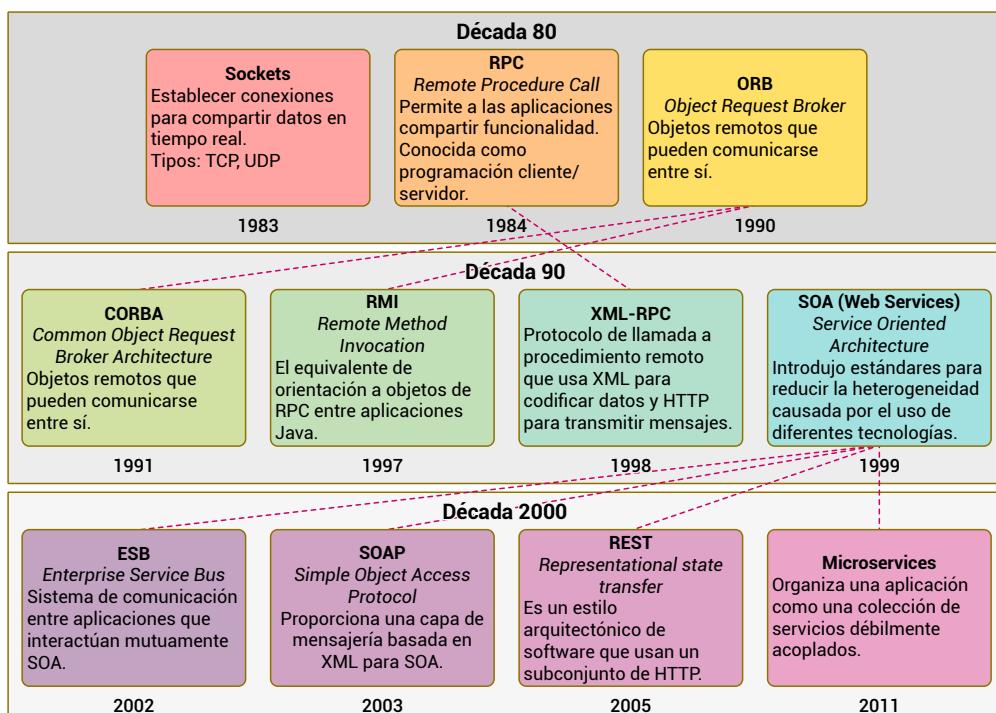
2.2. Técnicas de programación integrativa

Calleja (2013) menciona que las técnicas de programación integrativa tratan de resolver problemas de comunicación y funcionamiento de diferentes sistemas, estas tratan la problemática de unir de un modo coherente aplicaciones distintas, posiblemente en máquinas separadas, para dar una visión homogénea de un sistema.

Indudablemente, esas técnicas han cambiado con el transcurso del tiempo, siempre buscando la rapidez y el bajo costo para la integración. La Figura 11 muestra las diferentes técnicas de integración. Esas técnicas están relacionadas con la comunicación entre aplicaciones y trata de resumir tres décadas, si bien no trata de ser una descripción completa, presenta las técnicas que a criterio del autor son las más relevantes.

Las líneas punteadas que unen técnicas señalan que la técnica posterior, cronológicamente hablando, se basó en ideas de una anterior.

Figura 11.
Técnicas de integración de aplicaciones



Possiblemente se esté preguntando, ¿por qué la diversidad de técnicas? No existe una única respuesta, pero posiblemente una con mayor peso es la necesidad de comunicar sistemas que han sido desarrollados en diferentes lenguajes de programación.

En la unidad anterior, usted creó programas que se comunicaron entre sí utilizando sockets, pero los dos fueron desarrollados utilizando el mismo lenguaje de programación y sería una tarea bastante compleja tratar de comunicar esos mismos programas si, por ejemplo, el cliente se escribió en Python y el servidor en Java y viceversa.

Si regresamos a la Figura 11, en la década de los ochenta no era posible la comunicación entre sistemas desarrollados en diferentes lenguajes de programación, ya que ninguna de las técnicas brindaba esa posibilidad. Si bien era una necesidad latente, aún no se habían hecho los esfuerzos necesarios para alcanzar esa característica.

No es sino hasta la década de los 90, con la creación de CORBA, que se tuvo una primera aproximación para la comunicación entre sistemas desarrollados de forma heterogénea. Si bien esa aproximación no tuvo el impacto que se buscaba, posiblemente por su complejidad, fue un primer paso que prepararía el camino para la llegada de SOA que es la primera apuesta que llegó a comunicar sistemas escritos en lenguajes diferentes.

A partir de SOA, todas las demás técnicas soportan lenguajes heterogéneos, la variedad de técnicas responde al avance en las tecnologías como la omnipresencia de la Web (REST) y la evolución del software como un servicio (microservicios).

Dentro de esta asignatura hará un salto en el tiempo, revisó sockets (1983) y lo siguiente a revisar será arquitecturas orientadas a servicios SOA, por lo que le invito a revisar el siguiente apartado.

2.3. Arquitecturas orientadas a servicios

Como ya se mencionó, es la primera técnica que consiguió que aplicaciones implementadas utilizando diferentes lenguajes de programación se conecten, ejecuten funcionalidades entre sí e intercambien datos. Todo lo anterior se consiguió gracias al uso de estándares independientes de los lenguajes de programación. Estándares como HTTP y XML son la base de esa técnica.

Una de las primeras implementaciones de SOA que se utiliza ampliamente se basa en el protocolo conocido como SOAP. En el siguiente apartado lo estudiará a detalle.

2.3.1. Simple Object Access Protocol-SOAP

Para revisar este tema, es necesario revisar el texto base (Cardado, 2015), comience su lectura desde el punto 3, Modelo conceptual de las arquitecturas orientadas a servicios, detenga su lectura en el punto Componentes básicos de un documento XML.

De seguro las ideas que encontró en la lectura propuesta le parecen interesantes. Además de lo que se menciona en el texto, es necesario comentar que hoy en día SOAP es un estándar de la W3C (consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web). La versión más reciente de la especificación es la 1.2. (disponible en: [SOAP Version 1.2 Part 0: Primer \(Second Edition\)](#)).

Dentro de SOAP, uno de los conceptos más recurrentes es XML y necesario realizar una primera revisión de ese tema, por lo que le propongo la siguiente lectura de Cardado (2015) desde el tema Componentes básicos de un documento XML hasta el tema Funcionamiento de SOAP.

Luego de la revisión del texto, es necesario destacar algunos puntos sobre este tema. XML es un acrónimo de eXtensible Markup Language, utiliza una representación jerárquica para almacenar y transportar datos, y fue diseñado para ser leído por seres humanos y computadores. Al igual que SOAP, es una recomendación de la W3C.

Como se menciona en la lectura propuesta, XML se especializa en datos; existen otros lenguajes de marcas con otro propósito como HTML que se encarga de la visualización en un navegador, de una página Web y, muchas veces, tiende a mezclar datos con presentación. XML no tiene relación con la presentación de los datos, su relación es con representación de los datos.

En contexto con el tema de estudio, la integración de sistemas XML es independiente de los lenguajes de programación, es decir, un documento XML puede ser leído por cualquier programa, es más, un documento XML puede ser escrito por un programa Python y leído por un programa Java sin ningún inconveniente.

De seguro ya intuye la utilidad de XML dentro de la integración de sistemas, especialmente para los escritos en lenguajes de programación diferentes, pero antes de pasar a estudiar este tema, es necesario que conozca cómo se puede generar un archivo XML desde un programa, los siguientes párrafos le ayudarán a comprender este tema.

En la lectura propuesta, varios párrafos atrás se plantea un ejercicio práctico sobre la representación de información como tablas. A continuación, se presenta un programa Java que genera un archivo XML como solución al problema propuesto.

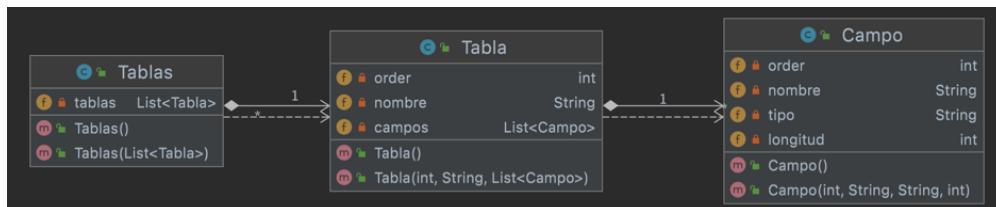
```
public static void main(String[] args) throws JAXBException {  
    JAXBContext context = JAXBContext.newInstance(Tablas.  
        class);  
    Marshaller marshaller = context.createMarshaller();  
    marshaller.setProperty(Marshaller.JAXB_FORMATTED_  
        OUTPUT, Boolean.TRUE);  
  
    Campo campo1 = new Campo(1, "Nombre", "Char", 50);  
    Campo campo2 = new Campo(2, "Apellidos", "Char", 100);  
    Campo campo3 = new Campo(3, "dni", "Char", 10);  
    List<Campo> camposT1 = List.of(campo1, campo2, campo3);  
  
    Tabla Tabla1 = new Tabla(1, "tabla1", camposT1);  
  
    List<Campo> camposT2 = List.of(  
        new Campo(1, "Código", "Int", 50),  
        new Campo(2, "Nombre", "Char", 100),  
        new Campo(3, "Descripción", "Char", 10));  
  
    Tabla Tabla2 = new Tabla(2, "tabla2", camposT2);  
  
    Tablas tablas = new Tablas(List.of(tabla1, tabla2));  
  
    marshaller.marshal(tablas, new File("tablas.xml"));  
}
```

Java es un lenguaje de programación orientado a objetos, es decir, todo se representará como clases y objetos. Para trabajar entre objetos Java y XML existe un framework denominado JAXB que, entre otras cosas, se encarga de transformar objetos a XML (operación denominada marshal) y de convertir los datos de un archivo XML en objetos Java (unmarshal).

De forma general, existen dos secciones en el código anterior, la primera es la representación de los datos a través de la creación de objetos de las clases: campo, tabla y tablas; la segunda parte es la de convertir los objetos de esas clases en XML (marshal), que es una tarea que se ejecuta en un línea de código. Si bien existen otras acciones, estas no se describen en este documento (más adelante encontrará el recurso en donde se hace una descripción completa).

Es necesario revisar las clases anteriormente mencionadas para comprender el funcionamiento del programa, inicie con el diagrama de clases.

Figura 12.
Diagrama de clases



En la Figura 12 se muestran las clases tablas, tabla y campo, y las relaciones entre estas. El diagrama se puede interpretar así: la clase tablas está compuesta por objetos del tipo tabla y esta, a su vez, por objetos del tipo campo. Es decir, a través de la relación de agregación se llega a representar la jerarquía en los datos que propone el ejercicio.

Haga una breve pausa y trate de responder a las preguntas, ¿cuál es el rol que cumplen las relaciones entre clases? Y, ¿qué función cumplen los objetos?

Contraste sus respuestas con las que a continuación le propongo. Las relaciones entre clases permiten representar la estructura de los datos que, como se dijo, es jerárquica, mientras los objetos representan los datos que se escribirán como XML.

Aún hay detalles por cubrir, así que avance con la revisión del código de la clase campo que posiblemente es la más representativa ya que contiene varios elementos que necesita conocer.

```
@XmlRootElement  
@XmlAccessorType(XmlAccessType.FIELD)  
public class Campo {  
    @XmlAttribute  
    private int order;  
    private String nombre;  
    private String tipo;  
    private int longitud;  
  
    public Campo() {  
    }  
  
    public Campo  
(int order, String nombre, String tipo, int longitud) {  
        this.order = order;  
        this.nombre = nombre;  
        this.tipo = tipo;  
        this.longitud = longitud;  
    }  
}
```

A simple vista, se trata de una clase Java sencilla, también denominada POJO (Plain Old Java Object), es decir, es una clase con atributos (orden, nombre, tipo y longitud) y dos constructores, uno de ellos el constructor por defecto y el otro que inicializa cada atributo a un valor que se pasa como parámetro.

Pero la clase utiliza anotaciones relacionadas con XML (ver las líneas que inician con `@XML`). Una anotación es una especie de metadato que proporciona datos sobre un programa, esos datos no forman parte del programa en sí. Además, esas anotaciones no tienen ningún efecto directo sobre el funcionamiento del código que anotan, para más detalles puede consultar el siguiente sitio [The Java™ Tutorials](#).

El propósito de las anotaciones que usan las tres clases que se usarán para representar la estructura en XML, se resume a continuación:

- `@XmlRootElement`: se utiliza para señalar qué la clase representa a un elemento de un documento XML.
- `@XmlAccessorType`: se usa para especificar la forma que se usará para trabajar con los atributos de la clase. Básicamente existen dos formas, la primera (`XmlAccessType.FIELD`) indica que se usarán los atributos de la clase, y la segunda (`XmlAccessType.PROPERTY`) que señala que se usarán los métodos get/set de cada atributo.
- `@XmlAttribute`: señala que un atributo dentro de Java será convertido a atributo dentro de XML.

De seguro se percató que las anotaciones ayudan a construir la estructura de un documento XML. Las otras dos clases, tablas y tabla, usan las mismas anotaciones descritas.

El documento XML que genera las clases anteriores es el siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tablas>
    <tabla order="1">
        <nombre>tabla1</nombre>
        <campos>
            <campo order="1">
                <nombre>Nombre</nombre>
                <tipo>Char</tipo>
                <longitud>50</longitud>
            </campo>
            <campo order="2">
                <nombre>Apellidos</nombre>
                <tipo>Char</tipo>
                <longitud>100</longitud>
            </campo>
            <campo order="3">
                <nombre>dni</nombre>
                <tipo>Char</tipo>
```

```

        <longitud>10</longitud>
    </campo>
</campos>
</tabla>
<tabla order="2">
    <nOMBRE>tabla2</nOMBRE>
    <campos>
        <campo order="1">
            <nOMBRE>Código</nOMBRE>
            <tipo>Int</tipo>
            <longitud>50</longitud>
        </campo>
        <campo order="2">
            <nOMBRE>Nombre</nOMBRE>
            <tipo>Char</tipo>
            <longitud>100</longitud>
        </campo>
        <campo order="3">
            <nOMBRE>Descripción</nOMBRE>
            <tipo>Char</tipo>
            <longitud>10</longitud>
        </campo>
    </campos>
</tabla>
</tablas>

```

Como puede ver, el resultado generado es diferente al que se muestra en el libro, esto debido a que se utilizó otros elementos, como los atributos que no se mencionan en el problema propuesto.

Hasta aquí está la revisión de código para generar archivos XML, en una unidad posterior aprenderá a leer y procesar estos archivos. Antes de continuar, es necesario mencionar que el código completo y otros recursos multimedia que le ayudarán a comprender todo el código lo puede encontrar en: [Itinerario 2: Programación Integrativa](#) (apartado semana 3).

Antes de finalizar esta unidad, le propongo estudiar el funcionamiento del protocolo SOAP, con este objetivo en mente realice una lectura comprensiva del tema Funcionamiento de SOAP, que se encuentra disponible en Cardado (2015), haga su lectura hasta el tema 3.2., Basado en recursos.

¿Qué le pareció la lectura? Ahora ya conoce el funcionamiento de un SOAP, pero, antes de continuar, es necesario comentar que existe un error en la figura con título Mensajes SOAP con HTML, el bloque que se encuentra sobre el texto Procesamiento del requerimiento tiene un nombre incorrecto, debería ser Cliente Java y no Cliente.NET, así quedaría en concordancia con el texto.

No olvide que SOAP utiliza el protocolo HTTP para enviar mensajes que están escritos en XML. Tal y como encontró en la lectura, la estructura general de un mensaje SOAP tiene 3 partes, esas partes son fijas y no se pueden cambiar, lo que si puede cambiar es el contenido de cada una de esas secciones.

Entre los pasos que describen el funcionamiento de SOAP está el uso de servidor UDDI (Universal Description, Discovery and Integration). Este servidor define mecanismos para publicar y encontrar información sobre servicios, es decir, se convierte en una especie de guía telefónica en donde se encuentran cada uno de los servicios web disponibles y la descripción de su funcionamiento, así los clientes pueden consultar los servicios disponibles y conocer cómo invocarlos.

Si bien, UDDI desempeña una función importante dentro del mundo SOAP, no es mandatario contar con un servidor de registro (no se necesita de uno para publicar o invocar a un servicio), esto provoca que muchas veces sea ignorado, pero las empresas con varios servicios SOAP, es decir, aquellas que basen su funcionamiento de forma parcial o total en esa arquitectura deben usar un servidor de registro. En su caso, al ser un programador, no necesita usarlo.

Dentro del funcionamiento de SOAP, recuerde que existe un algoritmo que se debe seguir para procesar un mensaje, un algoritmo de tres pasos. Finalmente, no olvide que SOAP es un protocolo que se puede extender a través de plugins, en la lectura propuesta se mencionan algunos de ellos.

Es momento de pasar a la parte práctica, en esta ocasión se hará la construcción de un servicio web SOAP utilizando el lenguaje de

programación Java, recuerde que aquí se hará una explicación resumida del código, es importante que revise los otros recursos publicados en [Itinerario 2: Programación Integrativa](#) (semana 3), ya que ahí se hace una explicación amplia del código utilizando otros recursos multimedia.

El problema a resolver es el mismo que se planteó al inicio de los contenidos de la semana 2 que se resumen así: se necesita consultar la existencia o no de un libro o publicación científica dentro de la biblioteca. En esta ocasión, únicamente se programará el servicio web (parte del servidor), no se programará el cliente, en su lugar se usará una herramienta que permite probar servicios web. Comience con el código.

```
@WebService  
public class QueryService {  
    @WebMethod  
    public boolean exists(String code) {  
        return Repository.exists(code.toUpperCase());  
    }  
}
```

La clase `QueryService` también utiliza anotaciones para hacer mucho más sencillo el trabajo con los servicios web SOAP. Las anotaciones utilizadas se pueden resumir así:

- `@WebService`: se utiliza a nivel de clase y su propósito es señalar que la clase implementa un servicio web.
- `@WebMethod`: se ubica en el método que será expuesto como una operación dentro de un servicio web. El método debe ser público y debe devolver algún valor.

Note como el método `exists` cumple con las características que señala la descripción de la anotación `WebMethod`. Además, ese método invoca a otro método, esta vez de la clase `Repository` que es una representación de cualquier mecanismo de persistencia de datos, como por ejemplo una estructura de datos, un archivo o una base de datos (relacional o NoSQL).

Así de sencillo es el código que se necesita para crear un servicio web SOAP, pero no es lo único que se necesita para poder ejecutarlo con el fin

de probarlo, también se necesita un servidor, el código del servidor es el siguiente:

```
public class ServerTest {  
    public static void main(String[] args) {  
        Endpoint.publish  
        ("http://localhost:8080/biblioteca/qryservice",  
         new QueryService());  
    }  
}
```

La clase *ServerTest* contiene el método *main*, es decir, es el punto de entrada para ejecutar el servicio web. La única sentencia que se utiliza se puede describir como la publicación en la URL `http://localhost:8080/biblioteca/qryservice` del servicio Web que está representado por un objeto de la clase *QueryService*. Aparentemente es algo sencillo, pero esa sencillez ocultan varias acciones que se describen a continuación.

1. Se levanta un servidor web en el puerto 8080.
2. Se habilita la ruta `/biblioteca/qryservice` que se convierte en el punto de acceso al servicio.
3. Asociación de la ruta del paso 2 con un objeto de la clase *QueryService*, es decir, al enviar un mensaje SOAP a esa ruta, el método *exists* será invocado.
4. Generar la descripción del servicio web a través de un archivo WSDL.

Cuando se ejecuta la clase *ServerTest* es posible ingresar a la URL [enlace](#), la misma que genera una página web con alguna descripción técnica del servicio web implementado. La Tabla 5 es un ejemplo de la página que se genera.

Tabla 5.*Información sobre el servicio web*

Punto Final	Información
Nombre de Servicio\:	Dirección\: http://localhost:8080/biblioteca/qryservice
Nombre de Puerto\:	WSDL\: http://localhost:8080/biblioteca/qryservice?wsdl
	Clase de Implantación\: ec.edu.utpl.pintegrativa.ws.soap.biblioteca.QueryService

Como se mencionó anteriormente, un elemento fundamental dentro de este tipo de servicio es el archivo WSDL que es una descripción del servicio, los métodos a los que se puede invocar, así como los parámetros que se deben enviar y el valor a recibir. La gran mayoría de herramientas y lenguajes de programación utilizan ese archivo para la construcción de clientes que consumen el servicio.

Para la construcción de un cliente que permita probar el funcionamiento del servicio creado, se usará la herramienta SoapUI que utiliza la URL del archivo WSDL para crear mensaje SOAP que se enviará al servicio y recibir otro mensaje de respuesta. Dentro del recurso [Itinerario 2: Programación Integrativa](#) existe el apartado Interpretar WSDL que utiliza el archivo que genera el servicio web que aquí se presenta para enseñar a leer archivos XML escritos en ese formato.

A continuación, se presentan ambos mensajes, el primero en presentarse es el mensaje de solicitud, es decir, el que envía el cliente al servicio.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:bib="http://biblioteca.soap.ws.pintegrativa.utpl.edu.ec/">
    <soapenv:Header/>
    <soapenv:Body>
        <bib:exists>
            <arg0>ABC</arg0>
        </bib:exists>
    </soapenv:Body>
</soapenv:Envelope>
```

En este caso, el valor del parámetro que se envía al servicio está encerrado en las etiquetas *arg0*.

El mensaje de respuesta, el que se envía desde el servidor al cliente como respuesta a la invocación al servicio web es el siguiente:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:existsResponse
            xmlns:ns2="http://biblioteca.soap.ws.pintegrativa.utpl.edu.ec/">
            <return>true</return>
        </ns2:existsResponse>
    </S:Body>
</S:Envelope>
```

El valor que devuelve la invocación al servicio es *true* señalando que el recurso bibliográfico sí existe en biblioteca.

Como se pudo percibir, programar un servicio no es una tarea compleja, tampoco lo es invocar a uno. Por ahora, no es necesario entrar en mayores detalles técnicos, por esta razón las siguientes líneas presentan un resumen como cierre del desarrollo de los contenidos de esta semana.

De esta manera termina el desarrollo de los contenidos para esta semana en la que estudió sobre la integración de aplicaciones a través de las arquitecturas orientadas a servicios utilizando SOAP. Además, aprendió a crear un servicio web y conoció cómo crear un cliente que lo consuma.

Para la próxima semana seguirá estudiando las arquitecturas orientadas a servicios, pero esta vez será utilizando servicios web basados en REST.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Prácticas con XML: [Itinerario 2: Programación Integrativa](#), revise la subsección práctica que está dentro de semana 3.
- Revisar la descripción de la implementación de un servicio web SOAP. Disponible en [Itinerario 2: Programación Integrativa](#), revise el apartado semana 3.
- Probar servicios web SOAP usando la herramienta SoapUI. Una guía está disponible en [Itinerario 2: Programación Integrativa](#), revise el apartado Interpretar WSDL.



Semana 4

En esta semana se continuará el estudio de las arquitecturas orientadas a servicios (SOA), pero, esta vez estudiará los servicios Web basados en REST que al igual que los basados en SOAP utiliza el protocolo HTTP para enviar mensajes, pero se diferencian en la forma como lo utilizan.

2.3.2. REST

Para introducir el tema, es necesario que realice una lectura del texto base (Cardado, 2015) del tema REST, detenga su lectura en el tema 5.2., Lenguajes de definición de servicios: el estándar WSDL.

Luego de la lectura, es necesario hacer énfasis en los siguientes temas:

REST no es una tecnología, sino que es una forma de utilizar una tecnología ya existente, en este caso la tecnología subyacente a la Web, es decir, REST utiliza las ideas y tecnologías que han permitido que la Web sea el sistema distribuido más grande construido hasta el día de hoy.

Recuerde que al trabajar con SOAP se habla de invocar a un método remoto, mientras que en REST se habla de acceder a un recurso. Sobre ese recurso se pueden invocar a un conjunto reducido de métodos, los proporcionados por el protocolo HTTP para hacer operaciones que dentro del mundo de las bases de datos se conocen como CRUD (Create, Read, Update y Delete).

Antes de avanzar, es necesario mencionar dos temas sobre REST, el primero es sobre su origen y el segundo sobre los motivos de su popularidad. En los siguientes párrafos se abordan estos temas, por favor, haga una lectura comprensiva de los mismos.

El origen de REST es una tesis doctoral de Roy Thomas Fielding, disponible en [Architectural Styles and the Design of Network-based Software Architectures](#), que propone utilizar los principios de la Web (recursos, representaciones, etc.) y la tecnología que los implementa (HTTP, URI, entre otros) como testimonio que la arquitectura de software se puede comprender a través de los estilos arquitectónicos y demostrar cómo los estilos pueden ser utilizados para guiar el diseño de una arquitectura de aplicaciones software basadas en la Red. Dentro de su tesis se concluye que la Web moderna es una instancia de una arquitectura que sigue el estilo REST.

Se podría decir, entonces, que se usó la Web para definir un estilo para construir aplicaciones distribuidas.

Es momento de hablar sobre su popularidad, debo aclarar que los siguientes párrafos corresponden a un criterio personal del autor de este documento que se sustenta en la experiencia y lectura de temas afines, que lo que busca es ir más allá de las comparaciones, que siempre son odiosas, y proponer un punto de vista diferente a los tradicionales, la mayoría de ellos basados en buscar las desventajas de algo para justificar el uso de algo nuevo.

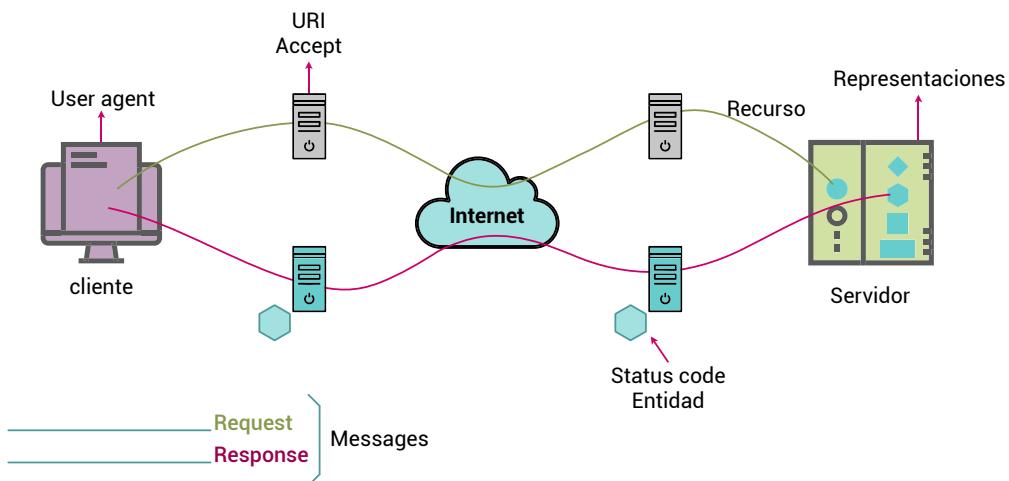
Cuando apareció SOAP, por allá a finales de la década de los 90, la mayoría de las empresas se enfrentaban al problema de integrar sus aplicaciones que se encontraban puertas a dentro, es decir, dentro de su propia intranet. Ese fue el nicho de SOAP. A la par del desarrollo de SOAP se encontraba creciendo el Internet y, entre ellos, uno de sus servicios más populares, la Web. Además, las empresas se vieron en la necesidad de traspasar sus fronteras (intranet) y salir a integrarse a sistemas que se encontraban fuera, es así como REST empezó a ganar popularidad.

Lo anterior no quiere decir que SOAP no se puede usar en Internet, recuerde que al igual que REST se basa en el protocolo HTTP, aún recuerdo cuando Google tenía un servicio SOAP para tener acceso a su servicio de búsqueda, con esto quiero decir que en un inicio se utilizaba SOAP para consumir servicios sobre el Internet, pero el surgimiento de nueva tecnología, lenguajes de programación y dispositivos provocó que SOAP ceda espacio a REST.

Sin lugar a duda, lo nuevo siempre trae mejoras, cubre fallas que lo anterior tenía y hace que la vida de los desarrolladores sea más productiva, pero los párrafos anteriores utilizan el entorno tecnológico en un espacio de tiempo como elementos que permiten que una idea o tecnología nueva relegue o reemplace a algo antiguo.

Luego de esta introducción, es necesario volver a estudiar algunos conceptos de REST, esta vez analizando su funcionamiento que se puede resumir en la Figura 13.

Figura 13.
Funcionamiento de REST



Tal y como muestra la figura anterior, el funcionamiento de REST es el mismo del protocolo HTTP y es el que se usa cuando se visita una página web, pero es necesario destacar algunos elementos que se pasan por alto.

Un cliente, que está identificado por la cabecera *user agent*, envía una solicitud (*request*) a través de las operaciones o métodos que brinda HTTP. Dentro de esa solicitud se incluye la *URI* (*Uniform Resource Identifier*) del

recurso que necesita y el formato que puede interpretar (cabecera *Accept*). A través de la infraestructura que brinda Internet, esa solicitud llega al servidor que posee el recurso, el servidor (generalmente un servidor web) analiza la información que se incluye en la solicitud y selecciona la representación del recurso que pide el cliente y prepara un mensaje de respuesta (*response*), este mensaje incluye el código de estado (*status code*) y la presentación del recurso que ahora recibe el nombre de *Entity* o Entidad. Al igual que se hizo con la solicitud, la respuesta usa Internet para llegar al cliente.

De seguro la descripción anterior le resulta un poco extraña, especialmente por los detalles que ocultan los navegadores cuando se los utiliza para visitar páginas web, pero la Web fue concebida para soportar no solo páginas web sino recursos y enviar representaciones de esos recursos. Aún debe tener varias preguntas sobre el funcionamiento de la Web bajo el estilo arquitectónico de REST, las respuestas las encontrará en el recurso disponible en [Itinerario 2: Programación Integrativa](#), revise el tema “La Web como una implementación de REST” dentro del apartado semana 4.

¿Qué le pareció el recurso propuesto? Estoy seguro de que le ayudó a despejar sus dudas relacionadas a la Web funcionando como una implementación de REST. A manera de resumen, revise las siguientes párrafos.

El protocolo HTTP soporta las operaciones GET, POST, PUT, DELETE, aunque un navegador web únicamente soporta GET y POST, el primero se usa cuando se solicita cualquier página web y el segundo cuando se envían formularios.

Generalmente se habla de la dirección de una página web o de su URL (Uniform Resource Locator). Las URL se constituyen en un subconjunto de las URI que se asocia a páginas web, pero al hablar de servicios se debe usar Uniform Resource Identifier.

Los clientes soportan ciertas representaciones de los recursos, HTTP usa como estándar los [tipos de datos MIME](#), básicamente es una forma de expresar los diferentes tipos de archivos que existen. Un navegador soporta (puede interpretar) múltiples tipos de archivos, pero por defecto trabaja con text/html o application/xhtml+xml, es decir, es una herramienta especializada en presentar páginas escritas usando HTML.

HTTP define una operación que se conoce con el nombre de negociación de contenido que es, básicamente, determinar si el servidor posee una representación del recurso solicitado por el cliente en el formato lo solicita. Recuerde la cabecera Accept que viaja en la solicitud y señala el tipo de dato MIME que el cliente soporta. En caso de no existir una representación del recurso en el tipo solicitado se debe informar, a través de los códigos de estado, que la representación solicitada no existe disponibilidad.

Los códigos de estado son números de tres dígitos que están agrupados así:

- Respuestas afirmativas (100 – 199)
- Respuestas satisfactorias (200 – 299)
- Redirecciones (300 – 399)
- Errores de los clientes (400 – 499)
- Errores de los servidores (500 – 599)

Estoy seguro de que al intentar visitar una página web, alguna vez se encontró con un error 404 que se presenta como página no encontrada, o un error 503 que se muestra como servicio no disponible. Ese comportamiento se debe a que al navegar se sigue un comportamiento estándar definido por HTTP.

Hasta aquí los detalles de REST, por ahora es necesario hacer una pausa en el desarrollo de la teoría detrás de ese concepto y pasar a la parte práctica.

Para ello, una vez más se usará el ejemplo propuesto semanas atrás y que tiene que ver con consultar la existencia de un recurso bibliográfico en la biblioteca, pero esta vez usando REST.

```
public class QueryBibResource {  
    public static void main(String[] args) {  
        get("/bibresource/:code", (request, response) -> {  
            Gson gson = new Gson();  
            var code = request.params(":code").  
                    toUpperCase();  
            var exists = Repository.exists(code);  
            BibResponse bibResponse = new  
                    BibResponse(code, exists);  
        });  
    }  
}
```

```
        if(exists) {
            response.status(200);
        } else {
            response.status(400);
        }
        response.type("application/json");

        return gson.toJson(bibResponse);
    });
}
}
```

El código anterior tiene algunas características que no se habían utilizado, como por ejemplo transformar un objeto Java a un formato bastante popular dentro de REST, como JSON, y que estudiará más adelante. Para que usted comprenda el código, a continuación se describen sus partes más importantes.

Recuerde que usted construye los servicios, es decir, se encuentra programando del lado del servidor. Cada solicitud REST tiene asociado un método GET para leer, DELETE para borrar, Put para actualizar y Post para crear, en este caso al ser consulta se usará GET. Cuando el cliente hace una solicitud GET al recurso que se encuentra identificado por la URI <http://localhost:4567/bibresource/> se ejecutará la función lambda que empieza por `get`, esa función recibe dos parámetros que representan la solicitud (`request`) y la respuesta (`response`).

Es necesario mencionar que Java, a partir de la versión 8, soporta el paradigma funcional a través de las funciones lambda. Dentro de ese paradigma es mucho más importante el qué se debe hacer antes que el cómo, es por esto que el código de programas funcionales es mucho más reducido y claro que el de otros paradigmas como el imperativo.

El servicio necesita el código del recurso bibliográfico que se está buscando, en este caso se envía utilizando una estrategia denominada “path param” o parámetro en ruta, para este ejemplo el parámetro recibe el nombre de “`:code`”. Posteriormente se lo recupera y almacena en variable `code`.

Al igual que se hizo anteriormente, se usa una clase, Repository, para representar al repositorio de recursos bibliográficos que posee la biblioteca y se verifica si existe o no un recurso a través del método `exists`, el resultado se almacena en la variable `exists`.

Con los valores de las variables `code` y `exists` se crea un objeto de la clase `BibResponse` que es una clase especial de Java, llamada formalmente registro, que se utiliza para modelar datos. El código de esa clase es el siguiente:

```
public record BibResponse(String code, boolean exist) {}
```

Como puede ver, es una clase que no necesita muchos detalles para funcionar, de esta forma el lenguaje trata de facilitar la programación ahorrando la escritura de código que puede verse como innecesario.

Lo siguiente que se hace es utilizar el valor de la variable `exists` para determinar el código de estado de la respuesta, si es verdadero se devolverá 200 que se interpreta como exitoso, caso contrario se devolverá 400, que se deberá interpretar como que el recurso buscado no existe.

Para finalizar se realizan dos acciones, la primera es agregar a la respuesta (`response`) una cabecera que informa el formato del contenido que se envía ("Content-Type") y se transforma del objeto `bibResponse` a JSON usando la librería `Gson`.

Para tener acceso a todo el código fuente, es necesario que visite el siguiente sitio Web: [Itinerario 2: Programación Integrativa](#). Cuando tenga el código completo, una de las primeras pruebas que puede hacer es abrir su navegador web (trate de usar FireFox), visitar la URL <http://localhost:4567/bibresource/ABC> y verá un resultado similar al siguiente:

Figura 14.

Resultado de invocar al servicio JSON usando un navegador web

The screenshot shows a web browser window with the following details:

- Address bar: localhost:4567/bibresource/ABC
- Toolbar buttons: back, forward, refresh, home.
- Header bar: JSON, Datos sin procesar, Cabeceras, Guardar, Copiar, Contraer todo, Expandir todo, Filtrar JSON.
- Content area:

```
code: "ABC"
exist: true
```

Firefox brinda un gran soporte a JSON, es por eso que tiene una forma especial de presentarlo, sin esa opción el JSON que el servidor envía es el siguiente:

```
{"code":"ABC","exist":true}
```

Ahora se va a modificar el código anterior para mostrar la negociación de contenido. Al código anterior, dentro del método main, se le agregará la siguiente función lambda:

```
before((request, response) -> {
    if (!request.headers("Accept").equalsIgnoreCase
        ("application/json")) {
        halt(406);
    }
});
```

El código anterior corresponde a un filtro que se ejecuta antes (before) de invocar al método solicitado (get). En ese filtro se evalúa si la cabecera Accept tiene el valor “application/json”, es decir, si el cliente soporta JSON. Si lo acepta se pasa el filtro y lo siguiente sería ejecutar la función get, caso contrario se detiene la ejecución y se devuelve como código de estado 406 Not Acceptable, el servidor no posee una representación en el formato que el cliente solicita.

Con este cambio, trate de usar nuevamente su navegador web para invocar al servicio, ¿qué resultado obtuvo? Es muy posible que su navegador se quedó en blanco y aparentemente no muestra ningún resultado. Ese comportamiento se debe a que los navegadores no están preparados para mostrar mensajes amigables a todos los códigos de estado de HTTP y 406 es uno de ellos.

Antes de continuar, trate de responder a la siguiente pregunta, ¿por qué el navegador web obtiene como código de estado 406? Estoy convencido de que encontró la respuesta, pero si no es así en el siguiente párrafo la encontrará.

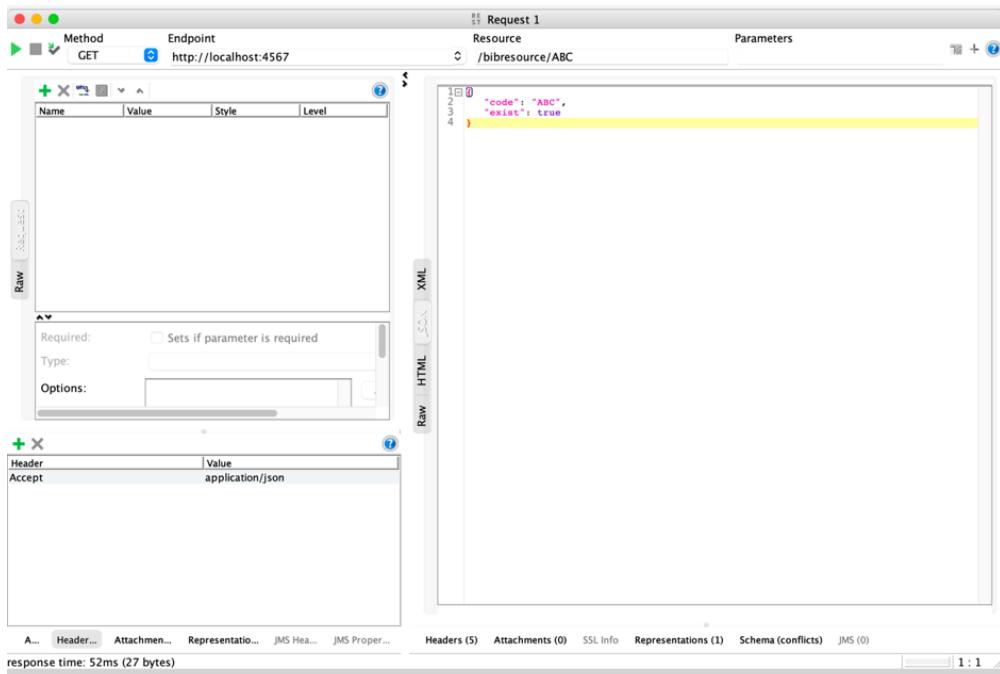
Como ya se mencionó, el navegador es una herramienta que trabaja con HTML y que generalmente dentro de su cabecera Accept se encuentran

valores como text/html o application/xhtml+xml, y no application/json, es por eso que no cumple con el filtro before y recibe como respuesta 406.

Para probar esta nueva implementación, es necesario usar otra herramienta que permita especificar las cabeceras que va a enviar. SoapUI es una buena alternativa ya que además de soportar SOAP soporta REST.

Figura 15.

Ejemplo de SoapUI para invocar a un servicio REST



Observe como en la esquina inferior izquierda está la opción de cabeceras configurada con Accept y el valor application/json, esta configuración en la solicitud hace que la misma supere el filtro y llegue a ser procesada. Acaba de experimentar con la negociación de contenido que generalmente es una acción que el browser no deja visualizar.

Espero que haya podido realizar este pequeño ejercicio, sino fue así, puede revisar el recurso disponible en [Itinerario 2: Programación Integrativa](#), debe ir a semana 4 y buscar el tema Negociación de contenido en donde encontrará recursos multimedia que le ayudarán a superar cualquier inconveniente.

Es momento de estudiar uno de los temas asociados a REST, los RESTful API que es un concepto que muchas veces llega a causar confusión, pero,

si es abordado correctamente, se convierte en una herramienta que ayuda en la construcción de servicios web. En el siguiente apartado estudiará este tema.

2.3.3. RESTful API

Alrededor de REST, como estilo arquitectónico, se pueden desarrollar diferentes aplicaciones, la Web es una de ellas y buscando separar el estilo arquitectónico con la creación servicios web que se basan en este, se definió el término RESTful API.

En este momento se debe estar haciendo alguna de las siguientes preguntas, ¿qué es RESTful API? ¿Cuáles son las diferencias entre REST y RESTful? Las respuestas a estas preguntas las encontrará en los siguientes párrafos, así que es necesario que haga una lectura comprensiva de los mismos.

Servicios Web RESTful o RESTful API son servicios web que se basan en REST para funcionar, es decir, utilizan los mismos conceptos que propone REST, pero esta vez para diseñar e implementar servicios web. Un ejemplo de esto fue el programa que se presentó en el apartado anterior.

Con esta premisa se puede decir que siempre que se hable de servicios web basados en REST se está hablando de una API RESTful. No olvide que una API es una Interfaz de Programación de Aplicaciones y es una forma simple de conectar e integrar sistemas. A diferencia de una aplicación, las API no cuentan con una interfaz de usuario (botones, ventanas, etc.), usualmente no son visibles, ya que su trabajo lo hacen por debajo, pero es fundamental para el funcionamiento de cualquier software.

La diferencia fundamental entre REST y RESTful, es que REST señala la forma en la que se pueden construir sistemas distribuidos, incluso sin el uso del protocolo HTTP, recuerde que el creador de REST lo presenta con un estilo definido para ayudar a crear y organizar sistemas distribuidos. Entonces los sistemas distribuidos que siguen ese estilo son denominados RESTful API.

Ahora mismo es difícil concebir a los servicios web REST sin HTTP, pero para tratar de entenderlo le propongo que haga una búsqueda en la Web con los términos “rest without HTTP” o “rest sin HTTP” y haga una lectura de lo que ahí puede encontrar. Si llega a sentirse sobrepasado por los resultados

arrojados, le sugiero leer Jain, A. (2019, Mayo 31) [REST - Stop calling your HTTP APIs as RESTful APIs](#).

Estoy seguro de que estos párrafos le ayudaron a comprender qué es RESTful API y la diferencia con REST. Es momento de avanzar en el desarrollo de los contenidos y que revise una de las tareas comunes que debe realizar cuando diseña servicios web RESTful, el diseño de una API para RESTful.

Para comprender los principios de diseño generales que usan para diseñar una RESTful API debe revisar el recurso [Itinerario 2: Programación Integrativa](#), en el apartado semana 4 encontrará el tema: Diseñando RESTful API.

No olvide que el correcto diseño de una RESTful API tiene injerencia directa en el uso de la misma. Además, existen diversos criterios para el diseño de las API, lo importante no es conocer todos los criterios, sino usar al menos uno.

De esta manera termina el desarrollo de los contenidos planificados para esta semana. En la próxima semana estudiará la última tecnología que pertenece a las arquitecturas orientadas a servicios y que a día de hoy es bastante popular.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Revisar el tema: La Web como una implementación de REST disponible en: [Itinerario 2: Programación Integrativa](#), en el apartado semana 4.
- Implementar el código fuente del caso desarrollado aquí y que se encuentra disponible en [Itinerario 2: Programación Integrativa](#).



Semana 5

Hoy en día uno de los temas más populares dentro del mundo del desarrollo de aplicaciones software de cualquier tipo son los microservicios, al igual que los temas anteriores es más un estilo arquitectónico antes que una

tecnología o un framework. Esa característica provoca que para estudiar microservicios se necesite de mucho tiempo, ya que por un lado se necesita estudiar el principio arquitectónico y por otro lado estudiar algún framework que siga esos principios y que permita implementar microservicios.

2.3.4. Microservicios

Es un estilo arquitectónico que ha ganado bastante popularidad durante los últimos años, especialmente en el desarrollo de aplicaciones en cloud, es decir, es un tema bastante popular, de hecho el término microservicios fue presentado en 2014 a través de una entrada en un blog. Si está interesado en leer ese artículo, aquí está la referencia: Lewis, J., Fowler, M. (2014, 25 marzo). [Microservices: a definition of this new architectural term](#). martinfowler.com.

El escrito de Lewis y Fowler sentó los conceptos básicos de lo que, en años posteriores, se empezó a utilizar como un estilo para desarrollar aplicaciones distribuidas.

Regrese por un instante al primer párrafo, ya que ahí se menciona otro tema bastante popular hoy en día, aplicaciones nativas cloud o cloud native applications, este tema lo estudiará más adelante, ahora inicie el estudio de esta tema revisando un resumen del estilo arquitectónico denominado microservicios.

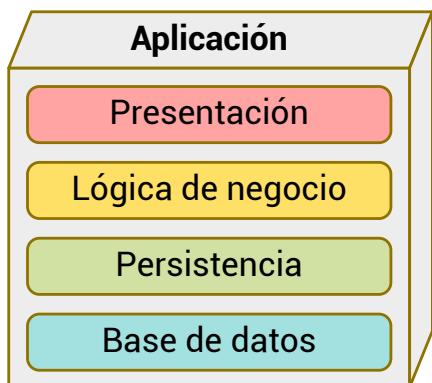
Para estudiar a los microservicios como un estilo arquitectónico es necesario que revise el recurso [Itinerario 2: Programación Integrativa](#), en el apartado semana 5 encontrará el tema Microservicios un estilo arquitectónico, haga una lectura del tema.

Ahora que conoce a los microservicios como un estilo arquitectónico es necesario recordar su contra parte: los monolitos. Un monolito es la forma más común de desarrollar aplicaciones, se desarrollaba una aplicación que implemente todos los requisitos de una empresa, esa aplicación se instalaba en una sola máquina aislada.

Las aplicaciones monolíticas tenían todo lo que el usuario necesitaba, interfaz de usuario, procesaban datos, los almacenaban en una base de datos, además, podían usar servicios web, es decir, eran parte de un sistema distribuido. Una representación visual de un monolito se muestra en la Figura 16.

Figura 16.

Una aplicación monolítica construida en capas



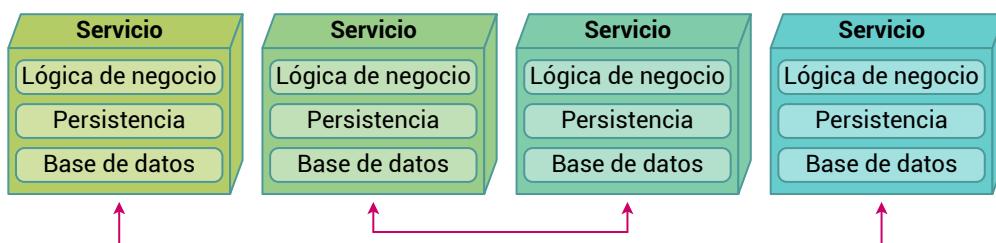
Como se puede ver en la figura anterior, una aplicación construida en capas puede ser un monolito, de hecho Newman (2021) clasifica a los monolitos en tres grupos, así:

- Monolitos de proceso único, en donde todo el código se implementa como un proceso único.
- Monolitos modulares, igual que el anterior, pero dividido en módulos que pueden trabajar de forma independiente.
- Monolitos distribuidos, es un sistema que está formado por varios servicios, pero que, por alguna razón, todo el sistema debe implementarse en conjunto.

Para concluir con la comparación, una aplicación monolítica es un todo integrado, mientras que en una aplicación basada en microservicios, cada microservicio es una parte independiente que tiene todo lo que necesita para funcionar, gráficamente se vería así:

Figura 17.

Una aplicación basada en microservicios



Cada servicio está provisto de todas las partes que necesita para poder operar de forma independiente, esto incluye hasta la base de datos y cualquier otro componente que necesite. Además, es posible que entre microservicios intercambien información para poder ejecutar su trabajo.

Estoy convencido de que el tema le parece interesante y cada vez conoce un poco más de este estilo arquitectónico. Ahora es necesario estudiar otro de los conceptos asociados a los microservicios, la computación en la nube o cloud computing, en los párrafos posteriores se presenta este tema.

El cloud computing, al igual que la mayoría de temas que se han tratado, es en realidad un tema extenso, pero una explicación bastante precisa y didáctica se encuentra en Celaya (2014), es por ello que debe hacer una lectura del tema: Concepto de Cloud, detenga su lectura en el tema “Lo que hemos aprendido”.

Con los conceptos en mente le propongo que haga el siguiente ejercicio, compare la computación en la nube y la forma tradicional de implementar aplicaciones y trate de encontrar las principales diferencias.

Estoy seguro de que pudo encontrar muchas diferencias, a continuación se presenta una explicación de la forma tradicional de implementar aplicaciones.

Tradicionalmente cada aplicación o sistema desarrollado era implementado en un servidor (hardware), además, según la importancia de la aplicación tenía un servidor exclusivo, esto determina que si se desarrolla una nueva aplicación esta necesitará un nuevo servidor. Asegurar que los servidores estén en funcionamiento y administrarlos trae consigo un incremento de las responsabilidades del personal técnico.

Cuando se trata de aplicaciones nuevas no es posible tener una certeza de qué tan grande debe ser el servidor, qué tan rápido, cuánta memoria, cuántas CPU. Por lo que, normalmente, se terminaba con un servidor grande y caro para una aplicación que apenas utiliza el 5% de esos recursos.

Ahora es necesario que revise la evolución al concepto de servidores físicos y llegue a estudiar el tema contenedores (containers) que es un tema que va de la mano con cloud computing.

Tomando en cuenta la forma tradicional de implementar aplicaciones, aparece la virtualización que permite tener múltiples aplicaciones

implementadas en un único servidor, pero cada una aislada de otras, así, si se desarrolla una nueva aplicación, no es necesario comprar un nuevo servidor, se puede reusar el existente. De esta forma los recursos del servidor son aprovechados de mejor manera.

En la virtualización existe una máquina virtual, cada aplicación obtiene un porcentaje de los recursos físicos reales del servidor. Cada máquina virtual necesita de un sistema operativo con licencia, así que, cuando tiene varias máquinas virtuales termina con varios sistemas operativos que administrar, actualizar, parchear, etc.

Una alternativa a la virtualización son los contenedores, un contenedor corre en un servidor físico sin la necesidad de virtualización, además, en lugar de instalar varios sistemas operativos por aplicación, se instalará solamente un sistema en el servidor y un contenedor por cada aplicación, por lo que cada aplicación inicia de forma rápida porque no es necesario iniciar un sistema operativo.

Dentro del mundo de los contenedores, actualmente **Docker** es una de las herramientas más populares. Entre sus conceptos más populares están las imágenes (Docker Image) que son plantillas para leer las instrucciones que permiten crear un contenedor (Docker container). Una imagen docker es una combinación de archivos de sistema y parámetros. Generalmente una imagen se construye basándose en otra y agregando ciertas personalizaciones.

Un contenedor (Docker container) es una instancia ejecutable de una imagen, es posible que existan varios contenedores que se construyeron con una misma imagen. Otra de las características de los contenedores es que estos permanecen aislados tanto entre ellos como del servidor sobre el cual se ejecutan.

Ejecutar un contenedor sobre un único servidor es una cosa, pero tener que ejecutar docenas o cientos de contenedores, manejarlos, reiniciarlos si fallan, incrementar sus recursos, etc., es otra historia, dado este escenario es que se necesita una tarea que se encargue de todo lo anterior.

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios facilitando la automatización y la configuración declarativa. Es decir, K8s, como también se conoce a Kubernetes, es una orquestador para aplicaciones que usan contenedores.

Dadas unas configuraciones, K8s es capaz de decidir el número de contenedores que la aplicación requiere, además, si es necesario crear otro contenedor ya sea por falta de CPU o memoria debido a una carga de trabajo alto o porque algún contenedor falla; así mismo puede deshacerse de contenedores si la carga disminuye. Esa capacidad de decisión lo consigue monitoreando el estado de los contenedores y sin la necesidad de intervención humana.

Docker y Kubernetes son dos temas bastante extensos y se escapan de los límites de esta asignatura. Por ahora, dejará estos temas ahí, pero si está interesado en continuar con el tema le recomiendo visitar las URL anteriormente suministradas.

Es momento de poner un alto en el desarrollo de los conceptos asociados a los microservicios como un estilo arquitectónico y pasar a la parte práctica y realizar una implementación de un microservicio.

Desarrollar un programa o aplicación que siga el estilo arquitectónico que promueve microservicios generalmente implica de un lenguaje de programación que a través de una librería o framework implemente esos principios. Recuerde que los lenguajes de programación son de propósito general, así que las librerías o frameworks son las encargadas de los temas específicos.

En la mayoría de los lenguajes de programación, generalmente, existen más de un framework para la implementación de algo específico, por ejemplo dentro del lenguaje Java para los microservicios es posible listar los siguientes frameworks:

- [Quarkus](#)
- [Helidon](#)
- [Micronauta](#)
- [Spring boot con Spring Cloud](#)
- [Vert.x](#)

El listado anterior no pretende, de ninguna forma, ser un listado completo de los frameworks que permiten desarrollar aplicaciones Java basadas en microservicios. La idea detrás del listado anterior es mostrar la popularidad de los microservicios a través de las herramientas que se han desarrollado para un lenguaje de programación.

A continuación, le invito a realizar la siguiente actividad.

Le propongo la siguiente actividad, considerando que Python es uno de los lenguajes más populares, realice una búsqueda en la Web sobre los frameworks Python para microservicios. Desde ya le adelanto que por lo menos encontrará 5 frameworks.

Como se ha venido haciendo dentro de esta asignatura, se usará Java como lenguaje de programación y Helidon como framework para desarrollar un microservicio. Así también se usará el mismo ejercicio que se ha venido usando.

Empiece la revisión del código con el microservicio que permite consultar si un recurso bibliográfico existe o no en la biblioteca. Revise detenidamente el código y luego lea la explicación adicional que encontrará en párrafos posteriores.

```
public class BibResourceService implements Service {  
    private BibRepository repository;  
    private static final JsonBuilderFactory JSON =  
        Json.createBuilderFactory(Collections.emptyMap());  
  
    public BibResourceService(BibRepository repository) {  
        this.repository = repository;  
    }  
  
    @Override  
    public void update(Routing.Rules rules) {  
        rules.get("/{code}", this::getResourceById);  
    }  
  
    private void getResourceById(ServerRequest request,  
        ServerResponse response) {  
        String code = request.path().param("code").  
            toUpperCase();  
        boolean exists = repository.exists(code);  
        if (exists) {  
            JsonObject returnObject = JSON.  
                createObjectBuilder()
```

```

        .add("code", code)
        .add("exist", exists)
        .build();
    response.send(returnObject);
} else {
    response.status(Http.Status.NOT_FOUND_404).send();
}
}
}
}

```

Estoy seguro de que notó que la clase *BibResourceService* implementa la interfaz *Service*, y tiene dos atributos *BibRepository* y *JSON*, el primero representa a la base de datos en donde se encuentran almacenados los recursos bibliográficos y el segundo es un mecanismo para crear objetos *JSON*.

Al implementar la interfaz *Service*, es mandatorio que se implemente el método *update* cuyo propósito es configurar las reglas con las que funcionará el servicio. En este caso, existe una única regla que señala que cuando se accede al microservicio usando el método *GET* y en la ruta se agrega un parámetro, denominado *code*, se ejecutará el método *getResourceById*.

Dentro del método *getResourceById* se encuentra toda la lógica que implementa el microservicio. Lo primero que se hace es recuperar el código que se envía como parte de la ruta, es decir, es un “path param”, una vez recuperado se transforma a mayúsculas y se almacena en la variable *code*. Lo siguiente es llamarla método *exists* de la clase *BibRepository* y asignarla a la variable *exists*.

Una vez recuperados los datos y si la variable *exists* es igual a *true*, se construye un objeto *JSON* que contendrá el código del recurso y *true*. Caso contrario no se devuelve contenido, únicamente se envía el código de estado 404 (no encontrado).

Ahora que ha concluido con la revisión del código, de seguro se ha percatado que el funcionamiento es igual al servicio RESTful que se presentó la semana anterior, con algunas diferencias que nacen del uso de

frameworks diferentes. Esto de seguro lo ha confundido. Los siguientes párrafos tratarán de despejar sus dudas.

Ambos servicios, los que se basan en REST como los que se basan en microservicios usan el protocolo HTTP para funcionar, esto provoca sus similitudes, pero analice las diferencias que ahí encontrará información relevante.

BibResourceService es un microservicio y, por lo tanto, contiene todos los componentes que necesita para ejecutarse, a diferencia del servicio RESTful que comparte los componentes con otras funcionalidades del sistema.

El microservicio desarrollado debe ser ligero, recuerde que su entorno natural es el cloud, en donde paga por lo que usa. Esto exige que los microservicios sean pequeños en tamaño (el microservicio desarrollado pesa 16 KB) y de fácil puesta en funcionamiento. Además, esto también demanda, por ejemplo, que se deje de usar Gson para representar un objeto Java como JSON.

Otras diferencias tienen que ver con los contenedores y su orquestación. Helidon escribe los archivos que se necesitan para: a) crear un contenedor que ejecute el microservicio y b) poder administrarlo usando Kubernetes. A continuación, se presentan ambos archivos.

El primer archivo que se presentará es el de configuración del contenedor de Docker, el archivo se genera automáticamente y se denomina Dockerfile.

```
# 1st stage, build the app
FROM maven:3.6-jdk-11 as build

WORKDIR /helidon

# Create a first layer to cache the "Maven World" in the local
repository.
# Incremental docker builds will always resume after that, unless
you update
# the pom
ADD pom.xml .

RUN mvn package -Dmaven.test.skip -Dclipselink.weave.skip
```

```
# Do the Maven build!
# Incremental docker builds will resume here when you change
sources
ADD src src
RUN mvn package -DskipTests

RUN echo "done!"

# 2nd stage, build the runtime image
FROM openjdk:11-jre-slim
WORKDIR /helidon

# Copy the binary built in the 1st stage
COPY --from=build /helidon/target/bibmicroservices.jar ./
COPY --from=build /helidon/target/libs ./libs

CMD ["java", "-jar", "bibmicroservices.jar"]
```

EXPOSE 8080

Le sugiero que lea los comentarios del archivo, las líneas que inician con un #, para comprender lo que sucede. Además, puede revisar el sitio web [¿Qué es Dockerfile y cómo crear una imagen de Docker?](#), le ayudará a comprender algunos comandos básicos que contiene el archivo.

El segundo archivo, el relacionado con Kubernetes, se denomina app.yaml y su contenido es el siguiente:

```
kind: Service
apiVersion: v1
metadata:
  name: bibmicroservices
  labels:
    app: bibmicroservices
spec:
  type: NodePort
```

```
selector:
  app: bibmicroservices
ports:
  - port: 8080
    targetPort: 8080
    name: http
---
kind: Deployment
apiVersion: apps/v1
metadata:
  name: bibmicroservices
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bibmicroservices
  template:
    metadata:
      labels:
        app: bibmicroservices
        version: v1
    spec:
      containers:
        - name: bibmicroservices
          image: bibmicroservices
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080
---

```

Dentro de Kubernetes, este archivo es una representación de un objeto Kubernetes, una introducción a su interpretación está disponible aquí [Entender los Objetos de Kubernetes](#)

Recuerde que una de las principales actividades dentro de los microservicios es el monitoreo del mismo, así se podrá realizar, vía Kubernetes, la administración automática del microservicio. Para cumplir con esta

característica, Helidon agrega dos conceptos denominados health y metrics. Ambos conceptos se traducen en URLs que devuelven información en formato JSON y texto plano, respectivamente, que puede ser interpretado por Kubernetes y decidir si es necesario levantar otro contenedor y dar de baja a alguno, por ejemplo. Para acceder a la información que arroja health se debe ingresar a la URL <http://localhost:8080/health> y <http://localhost:8080/metrics>.

El código completo lo encuentra en [Itinerario 2: Programación Integrativa](#) y una explicación extensa de cómo construir el microservicio está disponible en [Itinerario 2: Programación Integrativa](#), busque el apartado semana 5.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Poner en funcionamiento el código que implementa el caso de estudio desarrollado aquí y que se encuentra disponible en [Itinerario 2: Programación Integrativa](#).
- Revisar la explicación del caso implementado disponible en [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 5.

A continuación, se presentan preguntas relacionadas con los contenidos desarrollados en los temas 2.3.5. y 2.3.6. Esta autoevaluación le permitirá a usted reafirmar los conceptos estudiados.

Le invito a reforzar sus conocimientos participando en la siguiente autoevaluación:



Autoevaluación 2

En las siguientes preguntas seleccione verdadero o falso según corresponda:

1. () Con el uso de microservicios las aplicaciones se dividen en elementos más pequeños e independientes entre sí.
2. () ¿Una de las limitaciones de los microservicios es su compleja implementación?
3. () ¿En una aplicación, cada microservicio debe ser desarrollado bajo el mismo lenguaje de programación?
4. () ¿En una aplicación, cada microservicio tiene un nombre único?
5. () ¿Al fallar uno de los microservicios, falla toda la aplicación?
6. () ¿Los microservicios se pueden gestionar de manera independiente?
7. () ¿Los microservicios limitan la escalabilidad de una aplicación?
8. () ¿Una de las ventajas de los microservicios es el desarrollo de aplicaciones más estables?
9. () ¿En la capa de almacenamiento de una aplicación basada en microservicios se debe utilizar una única base de datos para todos los microservicios?
10. () Los contenedores son un ejemplo de una arquitectura de microservicios.
11. () ¿En la arquitectura de microservicios se pueden presentar más problemas de seguridad en comparación a las aplicaciones monolíticas?

12. () ¿Docker permite el uso de contenedores para crear entornos consistentes y eficientes de recursos para empaquetar servicios individuales?

[Ir al solucionario](#)

Aquí concluyen los contenidos de esta semana. Para la próxima semana estudiará cómo se puede implementar seguridad en aplicaciones basadas en SOAP.



Semana 6

En esta semana se desarrollará el tema de la seguridad en arquitecturas orientadas a servicios (SOA). Como usted conoce, se revisaron tres estilos arquitectónicos, SOAP, REST y microservicios, de los cuales los dos últimos tienen grandes similitudes, por esta razón se hará una revisión de la seguridad en SOAP, para luego pasar a revisar la seguridad de REST y microservicios, destacando alguna diferencia.

Una característica común de los tres estilos arquitectónicos es que se basan en el protocolo HTTP para su funcionamiento y, para Conceptos de seguridad (n.d), desde el punto de vista de la seguridad, un servicio web presenta los mismos problemas que cualquier otra aplicación web, por ejemplo:

- Robo de sesiones.
- SQL Injection.
- XML Injection.
- XPATH Injection.
- Denegación de Servicios (DoS).
- Cross Site Scripting (XSS).
- Etc.

Muchas de las técnicas que se usan para resolver los problemas en aplicaciones web son aplicables para los servicios web. Para conocer un poco más del tema, comience con el estudio de la seguridad para servicios Web SOAP.

2.3.5. Seguridad en servicios Web SOAP

En Conceptos de seguridad (n.d) se presenta un listado de las amenazas y ataques que los servicios SOAP pueden recibir. Dentro de las amenazas se menciona:

- Acceso no permitido.
- Alteración de la información en el canal de comunicación.
- Acceso inapropiado a los servicios.
- Denegación de servicios.

Mientras que para en los tipos de ataques se listan:

- Alteración de los mensajes.
- Ataques de confidencialidad.
- Hombre en el medio o “man-in-the-middle”.
- Suplantación de identidad (Spoofing).
- Denegación de servicio.
- Ataque de repetición..

Estas listas no son completas, además, son temas bastante activos y constantemente aparecen nuevas amenazas y tipos de ataques son constantes.

¿Qué se puede hacer para tratar de contener estas amenazas y ataques?
Para dar respuesta a esta pregunta le invito a revisar los siguientes párrafos.

La seguridad, en cualquier aplicación, es un tema de suma importancia, esa importancia se eleva aún más cuando existe un intercambio de datos que viajan por una red de datos. Los mensajes que intercambian los servicios web basados en SOAP están escritos en XML, que es un formato legible por cualquier persona y más aún por alguien con conocimientos técnicos.

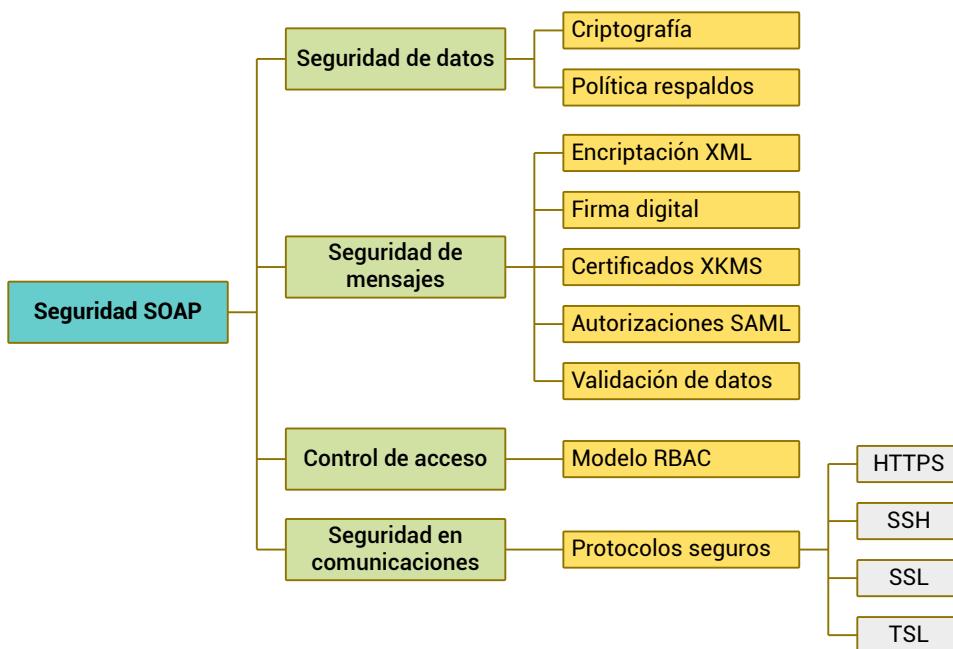
Estos y otros elementos son los que se deben hablar de la seguridad; para que adquiera una aprendizaje más profundo del tema le propongo que haga una lectura del tema 4.. Aspectos de seguridad en arquitecturas orientadas a servicios del libro base (Cardado, 2015).

La seguridad a nivel tecnológico es un campo extenso, lo que se hace en el texto base es una revisión teórica de los temas que se deben considerar cuando se habla de seguridad. Los temas que se proponen son: seguridad a nivel de datos, mensajes y comunicación y el control de acceso. En la Figura 18 se muestra un mapa conceptual que resume cada uno de estos temas.

Como un ejercicio le propongo que trate de emparejar las amenazas con uno o varios temas relacionados con la seguridad para servicios web SOAP. Así por ejemplo: alteración de los mensajes con seguridad de los mensajes, concretamente con firma digital.

Figura 18.

Seguridad en WS SOAP



Hasta aquí la revisión de este tema para las arquitecturas orientadas a servicios utilizando SOAP. En el siguiente apartado continuará estudiando la seguridad, pero en el centro estarán los servicios web que se basan en REST y microservicios.

2.3.6. Seguridad en servicios web RESTful y microservicios

Estos dos estilos arquitectónicos tienen mucho en común, ya que un microservicio puede ser considerado un servicio RESTful, pero un servicio RESTful no puede ser considerado microservicio. Considerando esta característica en este documento han sido agrupados para conversar sobre seguridad, por lo tanto, se hablará de la seguridad de un API y esa referencia abarca tanto a los API RESTful como microservicios.

Una buena fuente de información sobre este tema lo encontrará en Levin (2021), se sugiere al lector descargar el recurso (es de libre acceso) y utilizarlo como material de referencia. A continuación se presenta una adaptación de ese recurso.

Levin (2021) señala que es necesario considerar las siguientes recomendaciones antes de desarrollar un API.

- Protección de los datos. La protección de los datos que proporciona un API debe ser considerada siempre como una tarea de alta prioridad. Se necesita que se defina claramente las políticas para métodos como DELETE (eliminar un recurso) y PUT(actualizar un recurso). Únicamente los usuarios autenticados deben acceder a esos métodos y por cada llamada de ese tipo se debe guardar información que se conoce como de auditoría (qué se hizo, quién lo hizo, desde dónde se hizo, cuándo se hizo, etc.).
- Uso de Transport Layer Security (TLS) y Secure Sockets Layer (SSL), para la comunicación cliente-servidor, ya que son protocolos criptográficos que brindan seguridad en las comunicaciones a través de una red de computadores.
- Ataques Denial Of Service (DOS), son una posibilidad latente ya sea en APIs públicos como de acceso limitado si no se toman las medidas de seguridad adecuadas.
- ANTI-FARMING, que es el uso de un API por terceros para su beneficio. Esto generalmente se da en sitios web que ofrecen ofertas de todo tipo, desde vuelos hasta vehículos pasando por comestibles. Estos sitios se alimentan de muchas API, una de ellas puede ser la suya, estas llamadas pueden causar sobrecarga en sus servidores. Generalmente sucede cuando un API no tiene un mecanismo de autorización/autenticación implementado.

Antes de avanzar, trate de responder a la siguiente pregunta, ¿conoce claramente las definiciones de autorización y autenticación? Si es así, le invito a saltarse el siguiente párrafo, pero si tiene alguna duda, por favor lea el siguiente párrafo que tratará de resumir esos conceptos.

La autenticación trata de responder a la pregunta, ¿quién es usted? Generalmente se usa el mecanismo de nombre de usuario y contraseña para responder a esa pregunta. La autorización busca dar respuesta a la pregunta, ¿qué permisos tiene usted? La respuesta a esa pregunta determinará lo que un usuario, autenticado o anónimo, puede usar dentro de una aplicación o un API.

Levin (2021) hace una revisión de los principales tipos de autenticación, la mayoría de las API basan su seguridad en la tecnología subyacente, en este caso HTTPS, pero además se señala que deberían utilizar autenticación/

autorización basada en sesión que puede implementarse con Basic Auth, SAML, OAuth y JWT, pero claramente señala que las sesiones deberían enviarse como cabeceras y no como parámetros.

Es momento de estudiar las tecnologías de autenticación propuestas. Inicie con Basic Auth. Es un mecanismo que utiliza HTTP Basic authentication en donde el usuario debe suministrar nombre de usuario y contraseña los mismos que son codificados y almacenados en la cabecera con nombre Authorization. Este mecanismo debería combinarse con HTTPS, ya que el usuario y contraseña son únicamente codificados y el uso de TLS o SSL harán la encriptación.

Otra de las alternativas propuestas es SAML. Security Assertion Markup Language (SAML) se basa en XML para las tareas de autenticación y autorización. Utiliza dos entidades: un proveedor de servicios y un proveedor de identidades. La interacción entre esas entidades es la siguiente: un usuario solicita un servicio al proveedor de servicios, quien a su vez solicita confirmación de identidad al proveedor de entidad, en caso éxito el proveedor de servicios tiene acceso a la identidad del usuario. Con la confirmación recibida el proveedor de servicio puede tomar decisiones acerca del acceso autorizado a un usuario.

La siguiente alternativa a revisar es Oauth, creado en 2006, es un estándar abierto de autenticación que proporciona un flujo de trabajo de autorización a través de HTTP que se utiliza para autorizar dispositivos, servidores, aplicación y API con tokens de acceso en lugar de credenciales, así se puede leer datos de un usuario desde otra aplicación sin comprometer datos personales o confidenciales. Este mecanismo se hizo bastante popular gracias al uso de Google, Facebook, Microsoft, Twitter, Github, etc.

Finalmente, JSON Web Token (JWT) es un estándar abierto para la creación de tokens de acceso que permiten la propagación de identidad y privilegios. Así por ejemplo, un servidor podría generar un token indicando que un usuario tiene privilegios de administrador y proporcionarlo al cliente. El token se transmite como un objeto JSON y está firmado digitalmente.

Hasta aquí la revisión de los mecanismos de autenticación/autorización que usan cabeceras HTTP para enviar los datos, pero existe otra forma de autenticación/autorización que se basa en el envío de parámetros, aunque ha perdido bastante espacio con los API Keys.

Un API Key no es un estándar formal, sino una práctica común. Se basa en el uso de una o dos claves que se envía como un parámetro en cada llamada al API. Los API Keys tienen más que ver con la identidad del usuario que con la seguridad. Su uso generalizado está en controlar del número de llamadas a una API y también se puede usar como un mecanismo de monetización de las API, en lugar de bloquear llamadas, los clientes que pagan por usar el servicio tienen acceso al número de llamadas que han pagado.

Si bien, se mencionó que tanto los servicios web RESTful y microservicios se pueden agrupar dado su familiaridad, es necesario mencionar que existen diferencias en la implementación de la seguridad para cada estilo arquitectónico. En Richardson (2018) se menciona que no es posible imitar la seguridad de un monolito en los microservicios debido a:

1. Contexto de seguridad en memoria: los microservicios no comparten memoria, es por ello que no se puede utilizar un mecanismo de seguridad que se base en un contexto de seguridad en memoria. Un ejemplo de este esquema sería que una vez que se ingresa a una aplicación, la aplicación guarda en la memoria, del servidor o del cliente, que el usuario ha sido autenticado. Este mecanismo es bastante popular en frameworks de desarrollo web.
2. Sesiones centralizadas, si no es posible un contexto de seguridad en memoria, tampoco lo es una sesión en memoria. En teoría, varios microservicios podrían acceder a una sesión basada en una base de datos, excepto que violaría el principio de bajo acoplamiento.

Esas son las razones por las que en los microservicios se necesita de otros mecanismos como OAuth y JWT, tal y como se presentaron en párrafos anteriores.

Sin lugar a dudas, el tema de seguridad para los servicios web es un tema extenso, tan extenso que se han escrito varios libros sobre el mismo. Aquí se ha hecho una breve revisión teórica de esa temática. Se recomienda al lector que le interese el tema, buscar información en la Web que seguramente encontrará nuevas tecnologías que ayudan a llevar de mejor manera el tema de seguridad.

Antes de finalizar con el desarrollo de los contenidos de esta semana, es necesario revisar de forma práctica alguna de estas estrategias de seguridad estudiadas. Con esto en mente, revise el recurso disponible en

[Itinerario 2: Programación Integrativa](#), busque el apartado semana 6 y revise el tema prácticas que ahí encontrará varios ejemplos, se sugiere empezar por el ejemplo de JWT.

Ha concluido una semana más de estudios. La próxima semana será la última de este bimestre y se desarrollará un nuevo tema que se relaciona con los errores más comunes y buenas prácticas para evitarlos cuando se diseña e implementa un API RESTful.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Revisar la implementación de seguridad con JWT [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 6.



Semana 7

Los contenidos para esta semana tienen como tema principal presentar algunos de los errores más comunes que surgen al implementar una arquitectura orientada a servicios.

Debido a que se estudiaron tres posibles formas de implementar SOA, resultaría un apartado bastante extenso estudiar los errores más comunes en SOAP, RESTful y microservicios, es necesario delimitar el tema de estudio, por lo tanto, se propone estudiar desde el punto de vista del desarrollo de una API RESTful, que hoy en día es una tarea bastante común dentro de cualquier empresa.

En el siguiente apartado hará un estudio del tema descrito anteriormente, pero no solo de forma teórica, sino que al final encontrará la implementación práctica de los temas propuestos aquí. Además, se plantean las soluciones que resolverían esos problemas

2.3.7. Errores comunes al diseñar un API RESTful

En la actualidad, el diseñar un API RESTful es una tarea común dentro de cualquier empresa que utilice REST como su estilo arquitectónico. El correcto diseño de una API RESTful, no solo es útil para la empresa que lo

implementa, sino que también lo es para quienes consumen ese API, ya que de su correcto diseño se deriva su adopción.

Al diseñar una API RESTful se pueden cometer una serie de errores que muchas veces son recurrentes y que permiten que aparezcan los patrones de diseño que no son más que una explicación del problema y la descripción de su solución. Existe mucha bibliografía sobre los patrones de diseño para servicios RESTful que incluye libros enteros dedicados al tema.

Sin embargo, en este apartado se ha seleccionado los más comunes y de ninguna manera pretende ser una lista completa de todos los errores de diseño que se pueden dar cuando se trata de implementar un API. Además, presentar solo los problemas no es de gran utilidad, estos deberían acompañarse de una descripción de la solución y es justamente ese enfoque el que se empleará aquí.

Los problemas más comunes al diseñar un API RESTful son:

- Trabajar con formatos no estándar para la comunicación entre cliente y servidor.
- Pensar en métodos a invocar.
- Diseñar las URIs complejas de deducir.
- No respetar los código de estado del protocolo.
- No cuidar el ancho de banda utilizado.
- Despreocupación por la seguridad.
- No prestar atención al desempeño del servicio.
- Ausencia de una política clara en el manejo de versiones.

Una vez que se ha listado los errores más comunes en el desarrollo de un API RESTful, es necesario mencionar que estos son independientes del lenguaje de programación o framework que se usa para la implementación de servicios. Son más bien errores u omisiones de diseño antes que técnicos, obviamente se resuelven de forma técnica, se deberían contemplar en la fase de análisis y diseño.

Antes de pasar a revisar a detalle de cada uno de los errores le propongo un ejercicio, de los errores listados anteriormente seleccione uno y utilizando los conocimientos que posee trate de proponer una solución y luego compárela con la solución que encontrará más adelante.

Ahora le propongo revisar los detalles de cada uno de los errores listados y, sobre todo, poner atención en la solución que encontrará.

Trabajar con formatos no estándar para la comunicación entre cliente y servidor

Los mensajes que intercambian cliente y servidor pueden estar escritos en diferentes formatos. Muchos de esos formatos son abiertos, pero otros son específicos y se pueden abrir únicamente en herramientas con licencias de pago, por ejemplo, los archivos docx se abren en Microsoft Word que a su vez necesita una licencia de pago. Además, si el cliente es un programa, no una persona, este no podría abrirlo y leer su contenido.

Solución

Utilice formatos abiertos e independientes de las plataformas y que puedan ser manipulados por cualquier lenguaje de programación.

Un formato que se usa ampliamente es JSON, ya que este cumple con las características anteriormente mencionadas. Otro formato es XML, aunque este formato utiliza más caracteres que JSON para representar la información, que se traduce en mensajes que ocupan más bytes.

Técnicamente, utilice las cabeceras Accept y Content-Type para especificar el formato que el cliente soporta y que el servidor suministra, que en el caso de JSON será application/json y de XML es application/xml.

Pensar en métodos a invocar

Al diseñar un API, RESTful debe considerar que REST se basa en la noción de recurso, es decir, no debe pensar en acciones o métodos que necesita ejecutar sino en el recurso que necesita y en los métodos HTTP (GET, POST, PUT, DELETE), esa combinación sustituye al método. Esto afecta directamente en las URI que usará. Revise el siguiente ejemplo:

Suponga que necesita consultar un libro según su ISBN y ha desarrollado dos URI así:

- /getBookByISBN. Esta URI señala una acción, obtener un libro según su ISBN.
- /book/:ISBN. En este caso lo primero que señala la URI es el recurso libro (book) que tiene un ISBN.

Solución

Utilizar URI que incluyan sustantivos para señalar los recursos junto con el correcto uso de los métodos del protocolo HTTP así:

- GET para leer o recuperar recursos.
- POST para crear nuevos recursos.
- PUT para actualizar y.
- DELETE para borrar los recursos.

Diseñar las URI complejas de deducir

Las URI son el punto de acceso a los recursos. Un error bastante común es no diseñar ni mantener, en el tiempo, un esquema claro y fácil de deducir que permita a los clientes de una forma intuitiva encontrar otra URI que pueda necesitar.

Solución

Utilizar plantillas para la creación de URI, por ejemplo:

- /personas -> Mostar todas las personas.
- /personas/:id -> Mostrar una persona concreto.
- /personas/:id/emails ->Mostrar las diferentes direcciones de correo electrónico que una persona tiene.

De esta forma un cliente que parte del listado de personas puede deducir que si agrega el identificador de la persona puede acceder a ese recurso y que usando emails puede llegar a otro recurso. Dentro de este esquema hay que tener cuidado con los niveles de anidamiento que sean convenientes permitir.

No respetar los códigos de estado del protocolo

El protocolo HTTP es una protocolo basado en texto, fácilmente se puede enviar como respuesta un código que no existe dentro del mismo. Con base en la experiencia le puedo comentar que esto es común, en un trabajo que realicé encontré que algunos servidores enviaban códigos de estado como -1, 1000 o 0.

El principal motivo para cometer este error es tratar de personalizar los mensajes que se envían al usuario, esta personalización trae problemas

antes que soluciones. Recuerde que una fortaleza de REST es el uso de una interfaz estándar, si se cambia algo se pierde la estandarización.

Solución

Use los códigos de estado que propone el protocolo HTTP y personalice los mensajes, envíe un objeto JSON con los detalles del error, así el usuario, ya sea una persona o un programa, tendrá claro lo que ha sucedido.

No cuidar el ancho de banda utilizado

Este error se produce cuando en el API no se brinda posibilidades de filtrar y paginar los recursos devueltos, especialmente cuando se devuelven varios recursos. Por ejemplo, suponga que existe un servicio RESTful que le permite recuperar todas las representaciones de personas que viven en alguna parte del mundo. Esa consulta en teoría devolvería más de 7 mil millones de resultados, tomaría mucho tiempo enviar los datos y, si se usa una conexión de datos móvil, de seguro consumirá todos los megas disponibles.

Solución

Implementar opciones que permitan filtrar los datos por varios criterios, así como también paginar (dividir los datos en páginas de tamaño controlable).

Despreocupación por la seguridad

Su nombre es bastante descriptivo y posiblemente la causa más común sea la falta de tiempo que provoca que se trabaje más en el funcionamiento del API que en su seguridad.

Solución

Implementar uno de los mecanismos de seguridad que se cubrió anteriormente, tales como JWT o OAuth.

No prestar atención al desempeño del servicio

Este error se produce por falta de visión a futuro, le explico, en los primeros días de funcionamiento de un API son pocos los usuarios y la carga de trabajo del servidor es mínima, pero a futuro eso puede cambiar, se incrementan a cientos o miles los usuarios y la carga que ellos generan pueden llegar a provocar una denegación del servicio.

Solución

Existen varias alternativas de solución, desde balanceo de carga hasta otras más complejas, pero se puede empezar con cosas sencillas como alojar en caché (servidor) los datos que se consultan comúnmente, este mecanismo, aunque sencillo, ayuda a mejorar significativamente el tiempo de respuesta de un servicio.

Ausencia de una política clara en el manejo de versiones

Cualquier API es un contrato en donde un proveedor se compromete a funcionar de una manera, cualquier cambio en ese contrato traerá problemas a quienes usan el API. A partir de ese contrato los clientes construyen sus funcionalidades y recuerde que un API puede tener muchos clientes, así que un cambio puede hacer que todos esos clientes no puedan trabajar hasta implementar los cambios.

Solución

Establecer una política de versiones en donde todo lo nuevo va en una versión mientras se mantiene la versión anterior. Además de mantener las versiones anteriores se suele señalar una fecha límite (deadline) en la que esas versiones anteriores dejarán de funcionar.

De esta manera los clientes pueden empezar a hacer la transición de sus implementaciones a la nueva versión, minimizando el impacto que esto puede causarles.

Hasta aquí la revisión de los errores en el diseño de un API RESTful. Tal y como se mencionó, son errores en diseño que se deberían resolver en esa etapa, así los programadores tendrán claro lo que deben hacer desde el principio.

Mayor información la puede encontrar en Au-Yeung (2020), Yadav (2020) y Subramanian (2019).

Es momento de pasar a la parte práctica para diseñar y construir un API RESTful que implemente las soluciones a los errores analizados anteriormente. Comience revisando la descripción del problema.

Uno de los servicios de la biblioteca física de la UTPL es la reservación de salas de estudio. Actualmente existen 21 salas que se pueden reservar. El

horario para el uso de las salas empieza a las 07h00 y terminar a las 08h00, además una sala se puede reservar por un tiempo de máximo de 2 horas. Una detalle final es que una reserva se puede hacer para el día actual, no se puede hacer reservas en días venideros.

Con la descripción a este problema se propone desarrollar un API RESTful que simule el proceso de reservas de salas de estudio de la biblioteca. Entre las capacidades que tendrá el API están:

- Consultar las reservas por sala.
- Crear una nueva reserva.
- Modificar o eliminar una reserva.

El desarrollo completo del API está disponible en [Itinerario 2: Programación Integrativa](#), consulte el apartado semana 7.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Revisar el desarrollo de un API RESTful para el problema propuesto en [Itinerario 2: Programación Integrativa](#). consulte el apartado semana 7.

A continuación, se presentan preguntas relacionadas con los contenidos desarrollados las últimas semanas de clase. Esta autoevaluación le permitirá a usted reafirmar los conceptos estudiados.

Le invito a reforzar sus conocimientos participando en la siguiente autoevaluación:



Autoevaluación 3

En las siguientes preguntas seleccione verdadero o falso según corresponda:

1. () ¿Es un protocolo escrito en XML que nos sirve para el intercambio de información entre aplicaciones?
2. () ¿Dentro de los elementos de un mensaje SOAP, la cabecera es un elemento obligatorio?
3. () ¿En el cuerpo (body) de un mensaje SOAP contiene metadatos como la encriptación que se ha utilizado?
4. () ¿La seguridad en los servicios web de SOAP se lleva a cabo mediante la especificación WS-Security?
5. () ¿SOAP permite la interoperabilidad entre múltiples entornos?
6. () ¿Uno de los estilos de comunicación que admite SOAP es el RPC o llamada a procedimiento remoto?
7. () ¿Los mensajes SOAP son pequeños, por esta razón no ocupan mucha capacidad de procesamiento?
8. () Dado que SOAP está basado en XML, ¿es susceptible a diversos ataques y vulnerabilidades relacionados con XML?
9. () En un mensaje SOAP el elemento envelope es el elemento más importante y de mayor jerarquía dentro del documento XML y representa al mensaje que lleva almacenado dicho documento.
10. () El WSDL (Web Services Description Language) es un lenguaje basado en XML utilizado para describir la funcionalidad que proporciona un servicio web.

[Ir al solucionario](#)



Actividades finales del bimestre

Han transcurrido 8 semanas desde que inició el estudio de los temas relaciones con la integración y la integración de aplicaciones. Esta semana es una semana de preparación para la primera evaluación presencial. Con eso en mente se recomienda realizar las siguientes actividades:

- Revisar las respuestas a las autoevaluaciones propuestas durante todo el bimestre. Recuerde que al final de este documento encontrará las respuestas a las mismas junto con una breve sustentación de esta.
- Revisar la grabación del chat académico y sus recursos asociados que estarán disponibles en el EVA. Si no pudo participar del evento en vivo, recuerde que debe desarrollar la actividad suplementaria.
- Leer los comentarios que recibió su participación en el foro académico por parte de su docente tutor y sus compañeros. Esa retroalimentación es importante, ya que le ayudará conocer en donde debe reforzar.
- Realizar ejercicios y construir servicios SOAP, REST y microservicios, no olvide que la asignatura es práctica.
- Ingrese al entorno virtual de aprendizaje para leer los anuncios publicados, en ellos se incluirá instrucciones, actividades y ejercicios que deberá desarrollar en esta semana de preparación. Además, puede comunicarse con su tutor para resolver dudas e interactuar con sus compañeros para formar grupos de estudio.



Segundo bimestre

- Resultado de aprendizaje 3**
- Describe y contrasta los diferentes tipos de arquitecturas de integración de sistemas.

Contenidos, recursos y actividades de aprendizaje

El resultado de aprendizaje se alcanza a través del estudio teórico de los principales conceptos para luego aplicarlos a través del uso de herramientas que permitan experimentar con situaciones bastante reales y que son comunes dentro del contexto laboral.



Semana 9

Unidad 3. Integración e interoperabilidad de datos

Esta asignatura estudia dos tipos de integración, el primer tipo es la integración funcional, que comprende la comunicación entre sistemas con el fin de compartir funcionalidades, el enfoque empleado para este tipo de integración fueron las arquitecturas orientadas a servicios tal y como lo revisó el primer bimestre.

Para este segundo bimestre, continuará con la integración, pero el enfoque será la integración a nivel de datos. Hoy en día, la mayoría de las empresas cuentan con datos que provienen de sus sistemas, tanto internos como externos, y de otras fuentes como correos electrónicos y las redes sociales en línea. Si una empresa está en la capacidad de integrar todos esos datos con el propósito de analizarlos puede alcanzar una ventaja competitiva que le permita diferenciarse, de forma positiva, de sus competidores.

En esta unidad se presentarán los conceptos necesarios que le permitan comprender el problema de estudio y que le brindarán las bases conceptuales requeridas para estudiar el resto de las unidades.

Inicie este segundo bimestre con una revisión del concepto de integración de datos.

3. Integración de datos

Antes de pasar a las definiciones formales le propongo leer los siguientes párrafos que le ayudarán a tener una visión general de la importancia de los datos.

En Malcher, et al. (2016), se menciona que los datos son una fuente valiosa para las empresas. El uso adecuado de los mismos ha permitido a las empresas dejar a su competencia atrás, ayudándoles a ser eficientes en sus actividades, conocer a sus clientes y promover sus productos en función de hechos concretos y no en suposiciones. Además, los datos pueden convertirse en una fuente para la toma de decisiones, en fin, el poder de los datos es bastante grande.

El párrafo anterior es una visión a lo interno de una empresa, de los datos que una empresa tiene en sus bases de datos, pero hoy en día los datos relacionados a una empresa, a sus clientes, a sus proveedores e incluso de sus competidores se encuentran fuera, en la Web. El siguiente párrafo muestra esa otra fuente de datos y el impacto que ha causado.

En el 2017 una publicación de la prestigiosa revista semanal The Economist publicaba una sección especial titulada “The world’s most valuable resource is no longer oil, but data” (El recurso más valioso del mundo ya no es el petróleo, sino los datos) mostrando así la importancia de capturar la mayor cantidad de datos posible. Todos esos datos son suministrados por las personas en forma de publicaciones en redes sociales en línea de cualquier tipo.

Si bien, lo anterior puede ser visto como un riesgo, especialmente por la violación de la privacidad, existe también una parte positiva en la captación de los datos que se resumen en Bhageshpur (2019) en donde se mencionan varios ejemplos positivos de la captación y el uso masivo de los mismos puede ayudar en temas como pronosticar con mayor precisión cosechas, evitar incendios forestales o mejorar las estrategias de evacuación en caso de desastres naturales. También existen ejemplos relacionados con la salud (detección de cáncer), cuidado el medio ambiente, etc.

Sin lugar a duda una consecuencia de lo anterior es el gran volumen de datos que existen actualmente y que son accesibles a través del consumo de API que permiten acceso a los datos captados por las redes sociales en línea, como por ejemplo [Twitter](#), que, si bien no permite el acceso a todos

sus datos, sí es posible consultar cierto volumen de datos. Además, cada vez son más las empresas que siguen estrategias similares de captación de datos, una evidencia de esto es el gran número de aplicaciones para dispositivos móviles que se han desarrollado con doble propósito, por un lado, facilitar la experiencia de los usuarios y por el otro captar datos.

Esto ha permitido que la integración de datos sea un tema que necesita de tecnologías y técnicas del Big Data para poder utilizarla y estar en la capacidad de analizar para pasar de los datos a la información.

Para finalizar esta introducción, es necesario dar una visión de la investigación que se hace alrededor de la integración de los datos, en Daraio & Glänelz (2016) se muestran las tendencias sobre esta temática y se mencionan temas como open linked data (una forma de representar recursos enlazados), Big Data, Big Data Science, relevancia, precisión, credibilidad, accesibilidad y coherencia.

Luego de la lectura de los párrafos anteriores, de seguro tiene una visión clara de la importancia de los datos y de su necesidad de integración. Es momento de empezar a definir formalmente a la integración de los datos.

3.1. Integración de datos

Según Gartner Inc. (2016), es una disciplina que comprende prácticas, técnicas arquitectónicas y herramientas para lograr el acceso y entrega consistente de datos que provienen de diferentes fuentes y están estructurados de forma diferente, relacionados a una empresa.

La definición anterior es bastante amplia y general, para complementarla revise la definición que propone Halevy (2018), a pesar de que se centra en los sistemas de integración de información, es decir un tema concreto, aporta varios elementos que permite comprender de mejor forma el concepto general.

Los sistemas de integración de información ofrecen acceso uniforme a un conjunto de fuentes de datos autónomas y heterogéneas. Las fuentes pueden variar desde sistemas de bases de datos y sistemas heredados hasta formularios web, servicios web y archivos planos. No todos los datos están estructurados como en una base datos relacional. El número de

fuentes de datos en una aplicación de integración de información puede variar desde unas pocas hasta varios cientos.

Con esas dos definiciones, de seguro ha comprendido que el principal problema que la integración de datos trata de resolver es cómo acceder a los datos que son de una empresa, pero que se encuentran en diferentes sistemas, como si fueran silos aislados de datos. La integración de datos trata de romper esos silos.

Al romper esos silos, es cuando una empresa puede beneficiarse de sus datos integrados, al tener una visión integral del negocio es posible hacerlo más eficiente, identificar nuevas oportunidades, mejorar la toma de decisiones y la planificación futura.

De seguro han surgido algunas preguntas relacionadas con la integración de datos que probablemente en los siguientes párrafos encontrará las respuestas. En el siguiente apartado continuará con el estudio de la integración de datos y conocerá los diferentes tipos de integración que existen.

3.2. Tipos de integración

Así como muchos temas de las ciencias de la computación, no existe criterio universal sobre los tipos que existen. Una recopilación de algunos de esos tipos se encuentra en Data Integration (2021), por tal motivo antes de continuar debe realizar una lectura de ese tema.

Es muy probable que algunas de esas categorías le sean familiares. Otra categorización es la que propone Sreemathy (2020) y que se resume así:

Tipos de integración

Una característica que comparten esos cinco tipos de integración de datos es que los datos son transformados para así adaptarse a un único modelo de datos que ha sido diseñado con el objetivo de usarlo para realizar un tipo de análisis que se denomina inteligencia de negocio o BI (Business Intelligence). Es decir, los datos se transforman teniendo en mente qué preguntas (de negocio) pueden ayudar a responder.

Business Intelligence.- Análisis de información con el objetivo de mejorar y optimizar decisiones y desempeño, en Gartner (2016) puede encontrar una definición completa.

Existen otros tipos de integración que, si bien no se mencionan en ninguno de los recursos que revisó anteriormente, pero debe conocerlos para así conocer algo de las tendencias actuales.

Indudablemente la abundancia de datos es una de las características más relevantes de estos tiempos. Los tipos de integración tradicionales son poco útiles ante tal cantidad de datos que llegan al rango de los Zettabytes (1021 bytes).

Un data lake es un repositorio de datos que provienen de diferentes fuentes que se almacenan en sus formatos originales, es decir, no hay transformación a un modelo único. Las fuentes de datos van desde bases de datos relacionales, no relacionales, aplicaciones móviles, dispositivos de Internet de las cosas y redes sociales en línea.

Otra característica de un data lake es que provee de un motor de transformación de datos que permite transformar, limpiar e integrar los datos que almacena. Esta es una característica muy necesaria en este tipo de integración, ya que sin ella se convertiría en un agujero negro para los datos.

Finalmente, un tipo de integración que se destaca por su propósito es el data hub. Un data hub almacena la información principal de una empresa. Centraliza la información crítica de todas las aplicaciones de una empresa. Su objetivo es permitir compartir datos para el gobierno de una empresa. Se puede ver como un integrador de data warehouse y data lake.

De seguro encontró este tema bastante interesante y como punto final comentar que el data warehouse es uno de los tipos de integración más conocidos que ha sido ampliamente utilizado desde el 2000.

Hasta el momento, ha estudiado la definición de integración de datos y los tipos de integración, en el siguiente apartado iniciará el estudio de las técnicas de integración de datos que existen.

3.3. Técnicas de integración de datos

Las diferentes técnicas de integración de datos se encuentran descritas en Data Integration (2021), por lo tanto, es necesario realizar una revisión de ese material antes de continuar.

Estoy seguro de que la lectura le pareció interesante, recuerde que el orden de presentación de las diferentes técnicas se basa en el nivel de automatización de la integración, es así que la primera técnica es manual, mientras que la última tiende a ser 100% automática.

Información complementaria a las definiciones anteriores se muestran en [Información complementaria técnicas de integración](#), que es una adaptación de la información que proviene de 5 Data Integration Methods (n.d).

Ahora es necesario estudiar otro tema que es bastante importante y que pretende darle a conocer cuáles son los pasos por ejecutar dentro de un proyecto de integración de datos, es por ello que el siguiente tema es el ciclo de vida de la integración de datos.

3.4. Ciclo de vida de la integración de sistemas

El ciclo de vida de la integración de sistemas puede ser estudiado desde tres puntos de vista.

Le invito a profundizar su conocimiento sobre este importante tema.

El primero como un proyecto de desarrollo de software al que se le agregan ciertos elementos propios de la naturaleza del proyecto. En Reeve (2013) se presenta una descripción de la metodología en cascada para el desarrollo de sistemas, pero esta vez aplicada en la integración de datos. Dentro de la fase de análisis, que el autor la denominada (alcance), se hacen tareas como análisis de los datos que son necesarios, además de la identificación de fuentes tanto de origen como de destino.

En ese mismo trabajo se agrega un paso al ciclo tradicional dentro del cual se establecen perfiles para la fuentes de datos según el nivel de apertura de las fuentes, se argumenta que este paso es necesario para evitar futuros conflictos. Finalmente, en la fase de diseño se agregan dos actividades, la primera es detallar transformaciones y mapeos de los datos que son

acciones necesarias para la integración de los datos, y la segunda actividad, diseñar procesos de prueba periódicos para verificar que la integración se ha realizado con éxito.

El segundo enfoque es un enfoque más detallado y próximo a los procesos de integración, para estudiar este enfoque debe realizar una lectura del apartado El ciclo de vida de la integración de datos que hace Data Integration (2021).

Una vez hecha la lectura anterior, debe revisar el recurso interactivo que es una adaptación de la figura que se presenta en el recurso previamente mencionado.

Ciclo de vida de la integración de aplicaciones

Observe como el proceso que, propuesto, es un proceso iterativo, es decir, de desarrollo continuo, en el mismo se abarcan cuestiones mucho más específicas para la integración de datos, como por ejemplo estandarizar definiciones de datos o revisar temas de seguridad y privacidad, en fin, acciones mucho más cercanas al manejo de los datos.

El tercer punto de vista que se presentará es aún más técnico que el anterior y se basa en la forma de trabajo que varios autores la ven como una metodología de una de las herramientas que se utilizan para la integración de datos y que se denomina ETL, acrónimo de Extracción Transformación y carga (Load).

Es necesario que ponga especial énfasis en esta forma de trabajo, ya que será utilizada posteriormente para realizar varias prácticas de integración de datos y muchos de los conceptos que se utilizarán se sustentan en los conceptos que a continuación se presentan.

Reeve (2013) menciona que la principal función de la integración de datos es obtener datos desde donde esta se encuentra, cambiarla para hacerla compatible con el formato de destino y, finalmente, almacenarla en un nuevo repositorio. Esos tres pasos se denominan extract (extracción), transform (transformar) y load (carga), además, señala que toda integración de datos independientemente del tipo y técnica que use se resuelve con esas tres acciones básicas. A continuación, conocerá brevemente cada una de esas acciones.

Extract (Extraer)

Para poder extraer es necesario el acceso a los datos dentro de los sistemas o repositorios donde estos se encuentran. Se necesita una comprensión básica del formato y el significado de los datos para poder seleccionar los datos de interés que se seleccionarán.

Transform (Transformar)

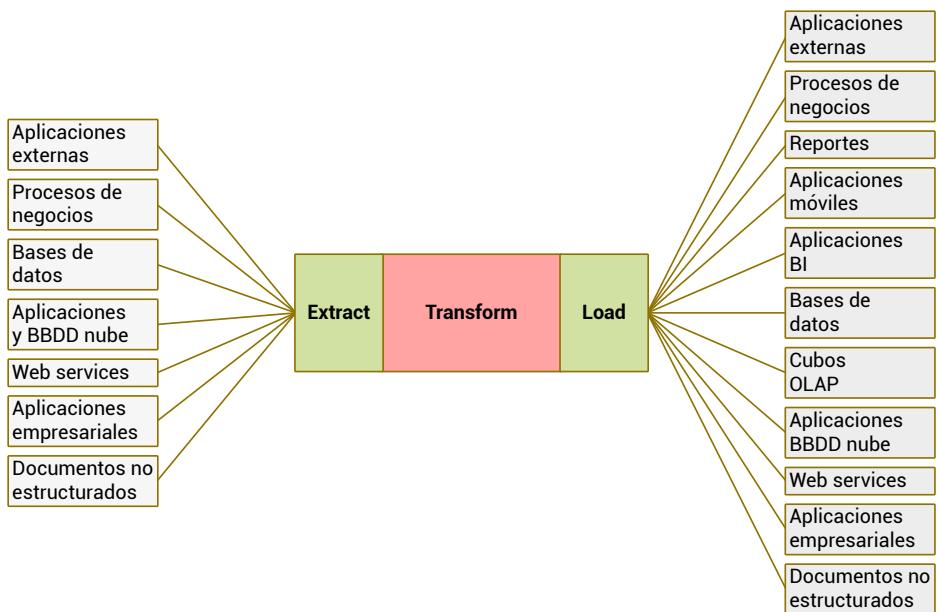
Transformar datos es un proceso que busca que los datos, de la fuente de origen, sean compatibles con la estructura de datos de destino, es un proceso que puede ser simple o complejo y que podría necesitar de información adicional. Aquí se ejecutan varias acciones, tales como mapeo (crear pares campo origen y su campo destino), lookup (buscar, devolver un valor a partir de una consulta), calcular (el dato a almacenar se obtiene al aplicar una fórmula, regla de negocio, etc.).

Load (Cargar)

Esta acción tiene que ver con llevar los datos transformados a la estructura de datos de destino. Existen dos enfoques para cumplir con esta acción, el primero es escribir código para almacenar directamente los datos en la fuente de destino, mientras que el segundo es utilizar el código existente para insertar valores en el repositorio de destino. Siempre que sea posible se debe preferir el segundo enfoque, debido a que ese código fue implementado para responder a las características del repositorio de destino.

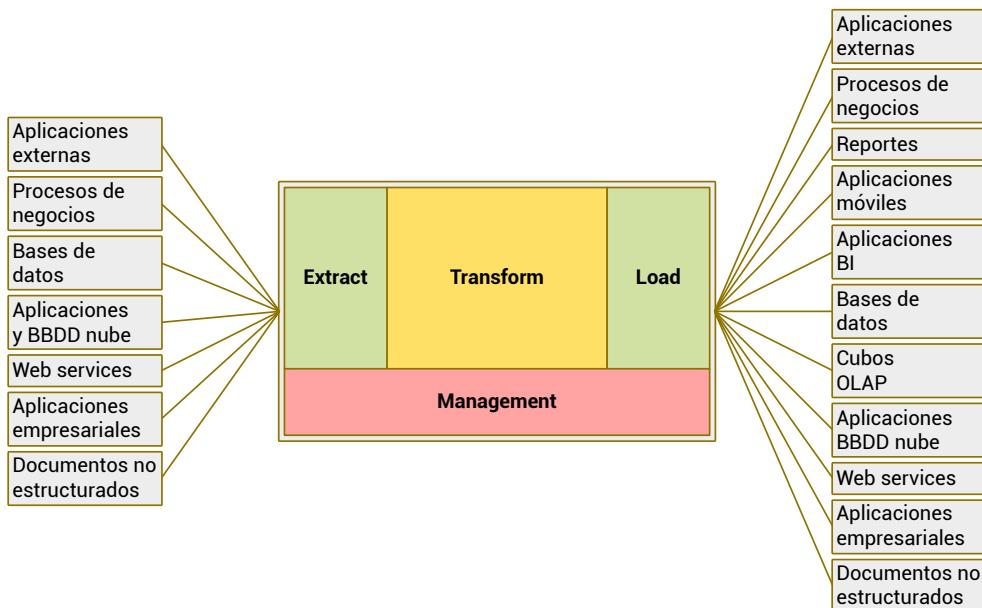
Con la descripción anterior ETL puede verse como la Figura 19, que según Sherman (2015) es la primera versión, en donde las herramientas que lo implementaban eran simples generadores de código con funcionalidades limitadas.

Figura 19.
ETL tradicional



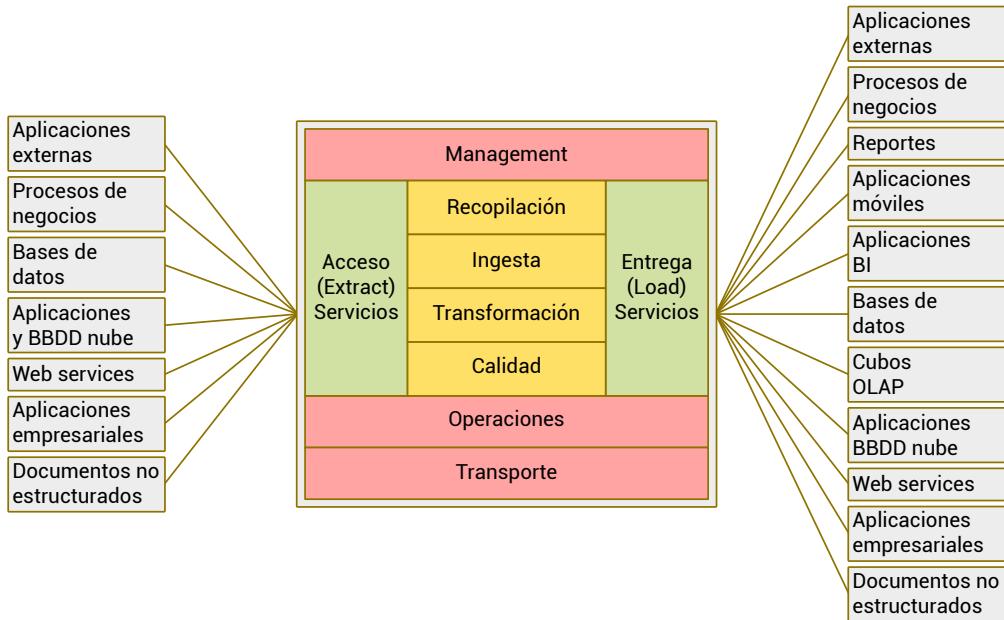
La Figura 20 muestra la segunda versión de ETL, en donde se agregaron soportes para la gestión de procesos operativos y desarrollo de software.

Figura 20.
Segunda versión de ETL



La imagen anterior no es un reflejo del proceso ETL moderno. La última versión se encuentra en la Figura 21 y que se describe a continuación.

Figura 21.
Última versión de ETL



Las herramientas modernas de ETL son paquetes completos que permiten integración en tiempo real, transmisión de mensajes, servicios web, procesamiento de eventos complejos y funcionalidades de virtualización de datos.

Esta versión moderna está formada por servicios de datos que, en la figura anterior, se representan por rectángulos y cada uno de ellos está formado por varios elementos cuya descripción es muy extensa y quedan fuera del alcance de este documento. Sin embargo, en los siguientes párrafos se hará una descripción de cada uno de los servicios generales.

Servicios de acceso (extract) y entrega (load), aunque se grafican en extremos opuestos, fundamentalmente utilizan el mismo conjunto de funciones que leen y escriben datos a lo largo de todo el flujo de integración de datos.

Una vez que los servicios de acceso recopilan los datos, los servicios de ingestá actualizan los repositorios de destino con los datos más recientes según las reglas definidas para, de esta manera, dar seguimiento a los datos que se crean y modifican.

La recopilación es un servicio que se encarga de crear perfiles de las fuentes de datos, perfilar las fuentes de datos es comprender los datos, conocer cómo se definen, su condición y cómo se almacenan y formatean.

Los servicios de transformación crean y completan el esquema (tablas, columnas y relaciones) de cada uno de los repositorios de datos de origen.

En todos los procesos existen errores, una vez que se llegan a detectar y comprender cuáles son y cómo corregirlos, es posible pasar a construir servicios de calidad para los datos. Esta es la base para futuras acciones.

Los servicios de gestión de procesos son el centro de comando y control para las tareas de integración de datos. Dependiendo de la herramienta de integración de datos que se utilice, las tareas pueden denominarse sesiones, flujos de trabajo, flujos, secuencias de trabajos o paquetes.

Los servicios de gestión de operaciones se emplean para supervisar y gestionar el diseño, el desarrollo y la ejecución de los procesos de integración de datos. Estos servicios deben hacer que el entorno de integración de datos sea eficiente en términos de utilización de recursos y eficaz en términos de lograr los objetivos comerciales y técnicos.

Las tecnologías de integración que se han etiquetado como servicios de transporte de datos permiten que la integración de datos se realice a través de una variedad de plataformas de integración. Esta lista de servicios de transporte de datos incluye:

- ETL y ELT (que se mencionan más adelante).
- Integración de aplicaciones empresariales, incluidos servicios ESB, SOA y web (o de datos).
- Servicios de mensajes empresariales que incluyen XML y JSON.
- Técnicas como virtualización de datos y federación de datos.

Como se mencionó, dentro de cada servicio existe un grupo de servicios especializados que ayudan a cumplir el objetivo propuesto en cada servicio general, en [Estructura moderna del proceso ETL](#) muestra un esquema de esa organización.

Si está interesado en leer el propósito de cada uno de los servicios que no se mencionaron en este texto debe revisar Sherman (2015) que hace una descripción amplia y detallada de cada uno de estos servicios.

Antes de continuar, es necesario comentar que dentro de Big Data se habla de ELT, es decir, extraer, cargar y transformar. Debido al volumen de datos la transformación se hace en la misma infraestructura que tiene la capacidad de procesar ese volumen.

Como pudo ver, el ciclo de vida de un proceso de integración de datos puede estudiarse desde el punto de vista del desarrollo de una aplicación en el que se incluyen ciertos pasos propios del propósito del proyecto. Otro punto de vista es abordar la integración de datos con una metodología propia y el tercero es llegar a utilizar la forma de trabajo de herramientas propias del proceso de integración de aplicaciones como las aplicaciones ETL.

Antes de continuar le planteo la siguiente pregunta, ¿con cuál de esos tres enfoques se queda?

Personalmente, creo que una mezcla de los tres ayudará a que la integración de datos llegue a cumplir con los objetivos propuestos y que iniciaron el proceso de integración. Las metodologías de desarrollo de sistemas son producto de la experiencia y buenas prácticas, y cualquier proyecto se puede ver beneficiado al usarla. Si a lo anterior le sumamos el trabajar de una manera propicia al tipo de proyecto, ayudará a comprender y manejar de mejor forma el proyecto y al agregarle los conceptos de una herramienta enriquecen aún más el proyecto y sobre todo que lo convierten en algo que se puede implementar.

Hasta el momento se ha utilizado el término datos, pero no se ha presentado una definición formal de lo que se entiende por datos, tampoco se ha dicho mucho de los tipos de datos, estos y otros detalles los encontrará en el siguiente apartado.

3.5. Los datos y sus tipos

Los datos son una de las primeras nociones que se adquiere desde pequeños, incluso antes de iniciar la formación académica se empieza a interactuar con datos debido, principalmente a que estamos rodeados de ellos y son el sustento de nuestra sociedad, pero ¿cuál es la definición formal de un dato?

Dar respuesta a esa pregunta no es nada sencillo, basta con realizar una búsqueda en algunos de los diccionarios en línea y podrá ver que cada uno

de ellos muestra definiciones diferentes, inclusive se utiliza como sinónimo de información. Es necesario entonces dejar clara cuál es la definición de información que se usará aquí.

En Merriam-Webster (n.d.) se define al dato como información fáctica (como mediciones o estadísticas) utilizada como base para el razonamiento, la discusión o el cálculo.

Esa definición deja claro que los datos provienen de los hechos y que se puede utilizar para otras acciones, como el razonamiento y el cálculo.

Es decir, es la materia prima para otros procesos. Así como lo señala la definición clásica de un dato, un dato es la unidad mínima de información que se necesita para crear información y, esta a su vez, para crear conocimiento.

Ahora ya tiene clara la noción de datos que es de nuestro interés, pero ¿cómo se clasifican los datos? Dentro de las ciencias de la computación y, en concreto, en la programación se habla de los tipos de datos, pero para la integración de datos existe otra clasificación que llega a determinar la forma de trabajar para integrar esos datos.

En Salina & Lemus (2017) se categorizan a los datos como estructurados y no estructurados. A continuación se presenta un resumen de esos tipos de datos.

Los datos estructurados son datos que son capaces de ser representados a través de estructuras predefinidas como arreglos, grafos, tablas, entre otros. Son datos cuya estructura es generalizada. Estos datos pertenecen al dominio de las bases de datos y son administrados por lenguajes como SQL.

Considerando lo anterior, los datos estructurados tienen un modelo de datos en donde se especifica un conjunto de características tales como el tipo de dato (entero, real, texto, etc.), longitud, valor inicial, la capacidad o no de admitir valores nulos, etc. De seguro recuerda el modelo entidad relación que se usa para la creación de una base datos, los datos estructurados responden a este tipo de modelos.

Estos modelos ayudan a que este tipo de datos sean fáciles de manipular, ya que generalmente se conoce qué esperar de ellos. Es necesario mencionar que estar estructurados no significa que sean simples. Hay cientos de

sistemas estructurados, cada uno con miles de tablas y cientos de miles de estructuras de datos. Y la complejidad de los datos va más allá de la estructura y el formato relativamente simples de los datos a la semántica de lo que realmente significa cada valor de los datos y cómo se relaciona con otros valores.

Otro tipo de dato son los datos no estructurados. Estos datos no tienen una estructura única predefinida, generalmente provienen del texto, correos electrónicos, publicaciones en blogs, imágenes, vídeos, que, hoy en día, son formatos en auge debido a las redes sociales en línea. Otra fuente son los sensores que hacen análisis analógico de señales como signos vitales, movimientos sísmicos, posicionamiento, procesos biológicos y químicos, entre otros.

Si bien, varios de los ejemplos mencionados anteriormente tienen cierta estructura, así como un correo electrónico tiene un remitente, un destinatario, un asunto, la información del cuerpo es más difícil de extraer de manera ordenada. Hasta hace relativamente poco tiempo, el procesamiento de datos no estructurados no era rentable debido a las dificultades que esta tarea acarreaba. Ahora, con el procesamiento del lenguaje natural (comprensión del lenguaje humano por parte de un computador), la tecnología de integración de datos puede comprender el sentimiento u otras ideas contenidas en el cuerpo de correos electrónicos o comunicaciones de redes sociales.

Una tercera categoría aparece, los datos semiestructurados que se refieren a los datos que se formatean de acuerdo con ciertas reglas aceptadas, pero que pueden variar en estructura dependiendo de los datos que proporcionen. Entre los ejemplos de datos semiestructurados, posiblemente los más conocidos son XML (Lenguaje de Marcado Extensible) y JSON (notación de objetos JavaScript). En esta categoría no existen filas ni columnas, pero los campos están marcados y los datos son identificables.

Generalmente, en los datos semiestructurados definen la estructura de datos y su significado dentro del mismo archivo y requiere que la aplicación comprenda cómo intercambiar datos entre diferentes sistemas.

Tanto los formatos no estructurados y los semiestructurados se almacenan en bases de datos NoSQL, ya que estas bases de datos son extremadamente flexibles y pueden manejar un amplio rango de formatos de datos. Inclusive, la mayoría de estas usan formatos no estructurados

para almacenar la información, por ejemplo: MongoDB es una base de datos que usa JSON como su formato de almacenamiento.

Para reforzar este tema le recomiendo revisar el siguiente recurso: LinkedIn Learning. (2021). [Fundamentos de big data: Técnicas y conceptos](#)-Datos no estructurados y semiestructurados. Este recurso es un vídeo de menos de 6 minutos en donde se explica los conceptos aquí expuestos.

Hasta aquí los contenidos para esta semana en donde se desarrollaron contenidos relacionados a la integración de datos, desde sus datos, técnicas, tipos y ciclo de vida, para finalizar con una breve presentación de los tipos de datos.

Como se mencionó anteriormente, estos tipos de datos determinan la forma de integración de los datos, con esto en mente las siguientes unidades se han diseñado para estudiar cómo integrar datos según su tipo y utilizando un lenguaje de programación y una herramienta ETL.



Semana 10

Unidad 4. Integración de datos no estructurados

Antes de iniciar con el desarrollo de los contenidos, le propongo una pregunta, ¿qué datos se puede obtener desde una fotografía? Trate de responder a esta pregunta antes de continuar con la lectura.

La pregunta anterior no especifica quién obtiene los datos, si se trata de una persona o de un computador. Si fuera una persona, muy probablemente hará una descripción de lo que está viendo en la fotografía, pero si ahora es un computador los datos serán diferentes y podrían estar asociados a las características del archivo (tipo de archivo, tamaño, dimensión, etc.)

En la actualidad, y gracias a los avances en la Inteligencia Artificial (IA), es muy probable que un computador pueda empezar a describir el contenido de la fotografía, podría identificar rostros, sitios, objetos, etc., es decir que hoy en día un computador puede obtener mucha más información de una fotografía que está dentro de los datos no estructurados, que los que antes podría obtener.

Así como sucede en el ejemplo anterior, hoy en día se utiliza muchas técnicas que provienen de la IA para la integración de datos no estructurados, especialmente en la etapa de transformación, es decir, se usa para transformarlos a un grupo de datos más amplios, por ejemplo, a un texto se puede transformar en una lista de elementos (entidades) que son mencionados en el texto y han sido identificados, tales como fechas, personas, lugares, números, cifras, etc. Otras técnicas pueden transformar el texto en un sentimiento positivo, negativo o neutro que es útil cuando se evalúan los criterios de una persona sobre un producto, servicio o institución. Y así se puede mencionar muchas transformaciones según del tipo de archivo.

Debido a lo extenso del tema y a la naturaleza de nuestra asignatura, es necesario dejar claro cuáles son los límites para este tipo de datos. El primero de ellos tiene que ver con el tipo de dato no estructurado a emplear, se usará únicamente texto, el segundo, y tal vez el más importante, se usarán servicios de terceros que implemente IA, es decir no se hará ninguna implementación propia de ninguna técnica de IA. El último límite que fija está en relación con los otros formatos de archivos de esta categoría, vídeo e imágenes, se usará una liberaría para obtener información de archivos que pertenecen a esas categorías.

En los siguientes apartados estudiará cómo se puede trabajar con tipos de datos no estructurados para realizar procesos de integración de este tipo de información. Considerando que es una tarea concreta, que no es parte de un proyecto, se usará el enfoque ETL como conductor del desarrollo de los contenidos. Comience con la primera fase de ese proyecto, la extracción.

4.1. Extracción de datos no estructurados

De forma general los datos provienen de dos tipos de fuentes, las internas y las externas. Dentro de las fuentes internas está todo el texto que una organización produce. Mientras que las fuentes externas generan texto que una organización recibe, hoy en día una gran fuente externa de texto son las redes sociales en línea en donde los usuarios o clientes pueden expresar abiertamente sus opiniones sobre una organización, un producto o un servicio, etc.

Dentro de las fuentes internas están, por ejemplo, los informes o comunicaciones que se intercambian entre diferentes departamentos,

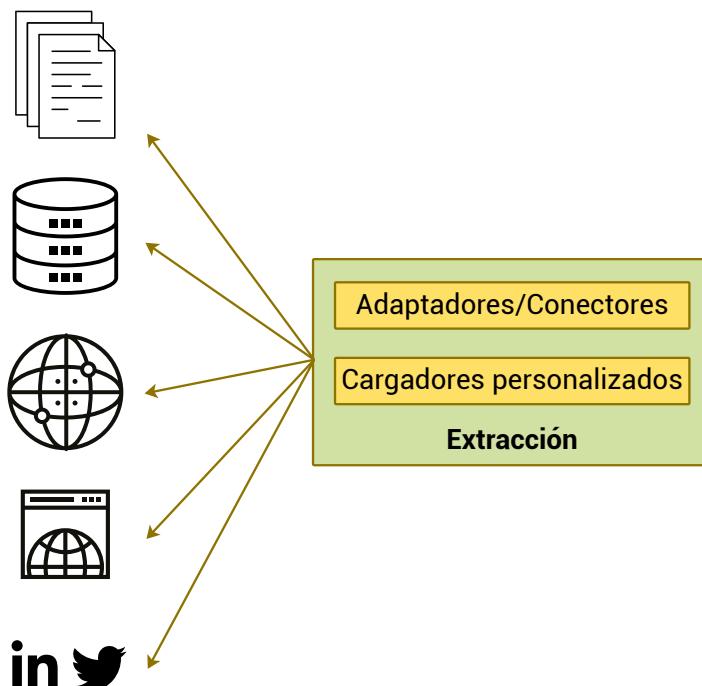
así como también los documentos de carácter legal que toda institución necesita para trabajar. Muchos de esos documentos se encuentran en poder de quienes los generan.

Además, con el arribo de las tecnologías de la información y comunicación los documentos, tanto internos como externos se encuentran en formatos digitales e inclusive aquellos que, por alguna razón no están en ese formato, son digitalizados generalmente con fines de preservarlos e inclusive disminuir el uso del papel. Las TIC agregan sus propios formatos, por ejemplo, correos electrónicos o páginas web.

Como puede ver, son diversas las fuentes de datos que poseen datos no estructurados en formato texto y son a esas fuentes a las que se debe acceder en la fase de extracción, es decir, tener acceso a documentos, emails, páginas web, etc., que ha sido identificados como relevantes para un proceso de integración de datos.

La Figura 22 muestra gráficamente algunos elementos que se necesitan para realizar la extracción de texto según sus fuentes. Ponga especial atención a los servicios que se agregan a la fase de extracción.

Figura 22.
Extracción de datos no estructurados



Dentro de la fase de extracción se necesitará construir mecanismos de acceso a cada una de las fuentes y necesitará utilizar los servicios de adaptadores/conectores y cargados personalizados para esa actividad. Los primeros se encargan de leer los datos, es decir, el texto. Algunos ejemplos de conectores:

- Librerías como JDBC, que permite a aplicaciones Java conectarse a base de datos relacionales.
- Apache POI, que es una librería Java que permite tener acceso a documentos del Microsoft Office utilizando el lenguaje de programación Java.
- Apache PDFBox, permite la manipulación (lectura/escritura) de archivos PDF.
- Clientes de APIs REST, que permiten tener acceso a fuentes como las redes sociales en línea.

Además, es muy probable que necesite desarrollar programas personalizados de carga de datos debido a que una o varias fuentes tienen estructuras propias (no estándares) y que necesitan del desarrollo de un software específico para tener acceso a esa fuente.

Antes de finalizar con este apartado, es necesario comentar que varias de las redes sociales en línea tienen API de acceso a sus datos, no todas, y las que permiten acceso, generalmente es a un grupo reducido de sus datos. Considera que el valor monetario de algunas de ellas no está dado por su tecnología, sino por los datos, razón por la cual no permiten acceso total a una de sus principales fuentes de ingreso.

Es momento de avanzar y estudiar la fase de transformación, en esta fase es donde se utilizará inteligencia artificial para transformar el texto en un grupo de datos que podrían ser más relevantes y, sobre todo, que se puedan emplear en actividades de análisis.

4.2. Transformar datos no estructurados

Como se mencionó anteriormente, los datos no estructurados que se usarán se basan en texto. En la fase anterior, extracción, se obtuvo el texto de alguna de las fuentes que se mencionaron anteriormente, en esta etapa se

transformará ese texto en otro formato que posiblemente sea mucho más útil para tareas de análisis de datos o inteligencia de negocios. Comience el desarrollo de este tema analizando el siguiente texto que se obtuvo de Díaz, Y. (2020, Diciembre 8) [Loja conmemora 472 años de fundación](#).

“El 8 de diciembre, Loja celebra 472 años de fundación, en este marco el Municipio de Loja realizó la colocación de ofrendas en el monumento al capitán Alonso de Mercadillo, ubicado en la plaza San Francisco.

Con civismo se desarrolló este acto, en donde elementos del cuerpo de Bomberos y agentes de control municipal realizaron el marco de honor. Con el ingreso de la bandera se inició el programa.

Jeremy Maita, alumno de la escuela municipal Ecológica, ofreció la conferencia alusiva a la fecha, destacando las dos fundaciones de Loja. La primera que se realizó en 1546 en el valle de Garrochamba y la segunda el 8 de diciembre de 1548, en el valle de Cuxibamba, a cargo del capitán Alonso de Mercadillo con el nombre de Inmaculada Concepción de Loja.

Loja fue considerada como fortaleza porque desde este lugar partieron los conquistadores hacia nuevos rumbos. “Son 472 años de esta bella ciudad que nos desafía a diario, tierra de artistas, intelectuales, gente genuina que ama y defiende su cultura”, enfatizó el niño Jeremy Maita.

Un total de trece ofrendas se ubicaron, que corresponde a la Fundación Cívica San Sebastián, Banco del Ecuador, Universidad Nacional, Universidad Técnica, Celec, Consejo Nacional Electoral, Corte Nacional de Justicia, Policía Nacional, Ejército, Consulado del Perú, Prefectura, Gobernación y Municipio de Loja.”

El texto, así como otros formatos, tiene como finalidad la comunicación entre seres humanos, es decir, ha sido diseñado y construido para ser entendido e interpretado por personas, que si bien desde varias décadas atrás se utiliza la tecnología para su producción y distribución, la interpretación de este sigue siendo una tarea exclusiva de los seres humanos.

Por ejemplo, usando el texto anterior le resultará bastante sencillo determinar cuál es el mensaje que la nota trata de hacer llegar a todos sus lectores. Si ese mismo texto se ingresa a un software como un procesador de texto como Microsoft Word, este nos dará algunos datos, tales como

números de palabras, si la ortografía y la gramática es correcta, pero será incapaz de comprender el propósito del texto.

En los últimos años, gracias al desarrollo de la inteligencia artificial, se han hecho grandes avances en el procesamiento del texto y al parecer que se acerca al día en donde un computador podrá comprender el mensaje que está dentro de un texto. Pero mientras ese día llegue, es posible utilizar la IA en tareas como la transformación de datos. En este apartado se hará una breve presentación de la minería de texto que es el proceso que se utilizará.

4.2.1. Minería de texto para transformar texto

Para Barila (2021) la minería de texto o text mining es el proceso o práctica de examinar un texto con el objetivo de obtener información de calidad que se utiliza para encontrar información nueva, previamente desconocida, a partir de diferentes datos no estructurados. La minería de texto utiliza una variedad de metodologías de análisis, una de ellas es el Procesamiento de Lenguaje de Natural o NLP. Las tareas específicas de la minería de texto son recuperación de información (Information Retrieval), extracción de información (Information Extraction), resumen de texto (Text Summarization), categorización de texto (Text Categorization) y agrupamiento (Text Clustering).

De las tareas mencionadas, posiblemente la extracción de información es la que resulta más interesante para los procesos de transformación. Antes de presentar el tema, regrese a la nota de prensa de la conmemoración de la fundación de Loja y trate de encontrar nombres de personas, fechas, números, instituciones y nombres de ciudades. Por ejemplo, una institución es Universidad Técnica.

Estoy convencido que la tarea no fue complicada. La extracción de información trata de hacer algo parecido a la tarea propuesta. Inicie con el estudio de este tema.

Barila (2021) dice que la extracción de información (IE) es el proceso de extracción automática de información específica de un lenguaje natural no estructurado o semiestructurado. Esta tarea reconoce entidades como nombres de personas, organizaciones, ubicaciones geográficas y búsqueda de relaciones entre entidades. La información extraída se almacena en bases de datos como plantillas y está disponible para su uso posterior. En general, esta actividad se basa en el procesamiento del lenguaje natural.

Las principales tareas de IE incluyen Reconocimiento de entidades nombradas (NER), Resolución de correferencia (CO), Extracción de relaciones (RE) y Extracción de eventos (EE).

El objetivo de NER es la detección y clasificación de tipos de entidades, como nombres de personas, organizaciones, ubicaciones, expresiones de tiempo, expresiones de moneda, cantidades, porcentajes. El desempeño de esta tarea se ve afectado por ciertos factores como el idioma.

Con esto se puede afirmar que la minería de texto, a través de la extracción de información, se puede emplear sobre información no estructurada basada en texto para transformarlo un grupo diverso de entidades.

Obviamente este proceso no alcanza el 100% de efectividad, ya que algunas entidades no serán reconocidas y otras posiblemente agrupadas en una categoría que no les corresponde, pero es una alternativa viable en constante evolución y, que gracias a las APIs web, está al alcance de todos.

Es momento de hacer un ejercicio que demuestre el funcionamiento de esta aproximación. En el siguiente apartado se presenta brevemente un ejercicio en donde se usará el reconocimiento de entidades sobre la nota de prensa que se presentó anteriormente.

4.2.2. Ejemplo

Para este ejemplo se usará el API web que proporciona la herramienta [Meaning Cloud](#). Además, posee una interfaz web que permite hacer pruebas de la misma funcionalidad que brinda el API web, pero con la facilidad que trae una interfaz gráfica de usuario. Si bien, para utilizarlo se necesita de un registro, el proceso no es complejo. A continuación, se muestra una figura con el resultado del proceso.

Figura 23.

Resultado de aplicar reconocimiento entidades

The diagram illustrates the process of entity recognition. It starts with a large blue arrow pointing left, labeled "El 8 de diciembre, Loja celebra 472 años de fundación, en este marco el Municipio de Loja realizó la colocación de ofrendas en el monumento al Capitán Alonso de Mercadillo, ubicado en la plaza San Francisco. Con civismo se desarrolló este acto, en donde elementos del Cuerpo de Bomberos y Agentes de Control Municipal realizaron el marco de honor. Con el ingreso de la bandera se inició el programa. Jeremy Maita, alumno de la escuela municipal Ecológica, ofreció la conferencia alusiva a la fecha, destacando las dos fundaciones de Loja. La primera que se realizó en 1546 en el valle de Garrochamba y la segunda el 8 de diciembre de 1548, en el valle de Cuxibamba, a cargo del capitán Alonso de Mercadillo con el nombre de Inmaculada Concepción de Loja." followed by a smaller blue arrow pointing right. This is followed by a table showing the results of entity recognition:

Entry type	Relevance	Form	Type	User defined
Entity	100	Loja	City	
Entity	66	Municipio de Loja	Top	
Entity	66	Capitán Alonso de Mercadillo	Top	
Entity	66	Jeremy Maita	FirstName	
Entity	33	Consulado del Perú	Top	
Entity	33	Policía Nacional	Police	
Entity	33	Nacional de Justicia	Top	
Entity	33	Nacional Electoral	Top	
Entity	33	Celce	Top	
Entity	33	Universidad Técnica	University	
Entity	33	Universidad Nacional	University	
Entity	33	Banco del Ecuador	BankingCompany	
Entity	33	Fundación Cívica San Sebastián	Top	
Entity	33	Inmaculada Concepción de Loja	Top	
Entity	33	valle de Cuxibamba	Address	
Entity	33	valle de Garrochamba	Address	
Entity	33	Agentes de Control Municipal	Top	
Entity	33	Cuerpo de Bomberos	ProfessionalAssociation	
Entity	33	plaza San Francisco	Address	
Concept	100	fundación	Top	
Time Expression		8 de diciembre		
Time Expression		1546		
Time Expression		8 de diciembre de 1548		
Quantity Expression		472 años de fundación		
Quantity Expression		las dos fundaciones de Loja		
Quantity Expression		trece ofrendas		

Como puede ver en la Figura 23, son varias las entidades que se han reconocido y clasificado (según una [ontología](#) definida por los creadores de la herramienta). También se puede observar que muchas de las entidades se podrían considerar como incompletas, tales como Consejo Nacional Electoral o Corte Nacional de Justicia, la herramienta los reconoce como Nacional Electoral y Nacional de Justicia respectivamente.

Algunas entidades que se encuentran en el texto no ha sido reconocidas, así como Ejército, Prefectura, Gobernación o Escuela Ecológica Municipal. A pesar de estas complicaciones, la información que se puede extraer del texto es amplia. Además, muchas de estos API permiten realizar ciertas personalizaciones, de tal manera que si el texto de entrada está enmarcado en un tema se puede usar información relacionada a este para mejorar el proceso de reconocimiento de entidades.

Así como Meaning Cloud, hoy en día existen muchas aplicaciones que poseen API web que implementan procesos de IA y que permiten que cualquier aplicación pueda utilizarlos. La gran mayoría de esas aplicaciones trabajan con un esquema de pago dentro del cual un número de invocaciones son gratuitos y, superado ese límite, se debe pagar para seguir usando el API.

Incluso grandes empresas como IBM, Google y Microsoft han desarrollado sus iniciativas para que, a través de su infraestructura, se pueda utilizar sus servicios de IA que siguen el patrón de consultas descritos en el párrafo anterior.

Aunque también existe la alternativa gratuita, un ejemplo es [DBpedia Spotlight](#) que se basa en datos enlazados, para hacer en reconocimiento de entidades y otras tareas de extracción de información. Esta herramienta va más allá de solo encontrar entidades, adicionalmente enlace la entidad con recurso disponible en la nube de Linked Data que abre la posibilidad de tener acceso a muchos otros datos, de esta forma se consigue reconocer entidades enriquecidas.

El resultado de anotar el nombre que le da spotlight al proceso de reconocimiento y enriquecimiento, a la nota de prensa sobre la fundación de Loja se muestra a continuación en la Figura 24.

Figura 24.

Un ejemplo de anotación de texto usando DBpedia Spotlight

The screenshot shows the DBpedia Spotlight annotation interface. At the top, there are settings for 'Confidence' (set to 0.35), 'Language' (set to Spanish), and options for 'n-best candidates' (unchecked) and buttons for 'SELECT TYPES...' and 'ANNOTATE'. The main area contains a block of text in Spanish about the founding of Loja, with various entities highlighted in blue underlines. Below the text is a 'BACK TO TEXT' button.

Confidence: 0.35 Language: Spanish
 n-best candidates [SELECT TYPES...](#) [ANNOTATE](#)

El [8 de diciembre](#), [Loja](#) celebra 472 años de fundación, en este marco el Municipio de [Loja](#) realizó la colocación de ofrendas en el monumento al Capitán [Alonso de Mercadillo](#), ubicado en la plaza [San Francisco](#). Con civismo se desarrolló este acto, en donde elementos del Cuerpo de Bomberos y Agentes de Control Municipal realizaron el marco de honor. Con el ingreso de la [bandera](#) se inició el programa. [Jeremy](#) Maita, alumno de la escuela municipal Ecológica, ofreció la conferencia alusiva a la fecha, destacando las dos fundaciones de [Loja](#). La primera que se realizó en 1546 en el [valle](#) de Garrochamba y la segunda el [8 de diciembre](#) de 1548, en el [valle](#) de Cuxibamba, a cargo del capitán [Alonso de Mercadillo](#) con el nombre de [Inmaculada Concepción](#) de [Loja](#). [Loja](#) fue considerada como fortaleza porque desde este lugar partieron los conquistadores hacia nuevos rumbos. "Son 472 años de esta bella [ciudad](#) que nos desafía a diario, tierra de artistas, intelectuales, gente genuina que ama y defiende su [cultura](#)", enfatizó el niño [Jeremy](#) Maita. Un total de trece ofrendas se ubicaron, que corresponde a la Fundación Cívica [San Sebastián](#), Banco del [Ecuador](#), [Universidad Nacional](#), Universidad Técnica, Colec, [Consejo Nacional Electoral](#), [Corte Nacional de Justicia](#), [Policía Nacional](#), Ejército, [Consulado del Perú](#), Gobernación y Municipio de [Loja](#).

[BACK TO TEXT](#)

Las entidades reconocidas se convierten en enlaces que llevan a recursos con más información para así enriquecer aún más los datos no estructurados.

Es momento de que trate de usar alguna de estas herramientas por su cuenta. Busque algún texto, una buena fuente son las notas de prensa, y utilice al menos una de las herramientas antes mencionadas.

De esta forma se pone fin al desarrollo de los contenidos de esta semana, en donde se presentó algunos detalles del proceso de integración de datos no estructurados, concretamente el texto. Se presentó y desarrolló una propuesta basada en IA para hacer reconocimiento de entidades y así transformar el texto en una lista de entidades que se pueden utilizar para realizar análisis de datos.

En la próxima unidad usará un lenguaje de programación para la construcción de un pequeño programa que permitirán reconocer entidades a partir de un texto. Además, se presentará una propuesta para trabajar con otros formatos de datos no estructurados.



Actividades de aprendizaje recomendadas

Se recomienda realizar la siguiente actividad.

- Busque una nota de prensa y analícelo usando [DBpedia Spotlight](#).

Le invito a reforzar sus conocimientos participando en la siguiente autoevaluación:

A continuación, se presentan preguntas relacionadas con los contenidos desarrollados en las últimas semanas. Esta autoevaluación le permitirá a usted reafirmar los conceptos estudiados.



Autoevaluación 4

En las siguientes preguntas seleccione verdadero o falso según corresponda:

1. () Una de las principales ventajas que ofrece la API REST, es que siempre es independiente del tipo de plataformas o lenguajes.
2. () ¿Los objetos en REST siempre se manipulan a partir de la URI?
3. () ¿En un sistema REST, la URI cambia a lo largo del tiempo?
4. () ¿La operación GET es una operación solo de lectura?
5. () ¿El método PATCH está pensado para la creación de nuevos recursos?
6. () ¿Una de las bases de los servicios REST es su sistema por capas?
7. () ¿REST es un tipo de lenguaje de programación para el desarrollo web?
8. () Una de las desventajas principales del uso de una API REST reside en la independencia que proporciona frente a cualquier consumidor.
9. () Una de las características principales de los servicios Web REST es el uso explícito de métodos HTTP.
10. () Los servicios REST utilizan una técnica basada en el uso de hipermedios como motor para poder indicar las diferentes acciones.
11. () Una API RESTful es un servicio que funciona como un estándar para compartir información, en un sistema de doble vía: ¿consulta y respuesta?

12. () ¿El código de error HTTP 403 Forbidden, indica cuando el servidor web no encuentra la página o recurso solicitado?

[Ir al solucionario](#)



4.3. Integración de datos no estructurados en la práctica

La semana anterior se presentó el tema de la integración de datos no estructurados y se mostró algunas herramientas para realizar el reconocimiento de entidades. Para esta semana continuará con el mismo tema, pero se usará un lenguaje de programación para construir un cliente básico que consuma una API web para realizar el reconocimiento de entidades. Finalmente, se presenta, de forma práctica, una propuesta para trabajar con otros formatos de datos no estructurados como audio y vídeo que se basa en el uso de metadatos.

Antes de comenzar, es necesario mencionar que el código completo se encuentra disponible en [Itinerario 2: Programación Integrativa](#) y una explicación detallada con recursos multimedia se encuentra en [Itinerario 2: Programación Integrativa](#), bajo el tema semana 11.

4.3.1. Construcción de un ejemplo

Actualmente son varios los API que permiten realizar reconocimientos de entidades en texto, también existen frameworks especializados en temas de IA, ambas alternativas son factibles. Una diferencia sustancial, entre ambas, es el nivel de conocimiento de IA que cada alternativa exige. Los API son construidos pensando en hacer las tareas más sencillas sin demasiado conocimiento sobre IA. Mientras que un framework demanda más conocimientos de IA. Sin lugar a duda, los frameworks son mucho más flexibles que un API.

En este ejemplo se usará un API web para realizar la extracción de entidades desde un texto, de hecho, es la misma herramienta que se usó la semana pasada, pero esta vez se deja de lado la interfaz web y se usa el lenguaje de programación Java.

Un requisito para poder trabajar en este apartado es crear una cuenta de usuario en MeaningCloud, existen varias opciones, pero la gratuita será suficiente, ya que ofrece 20000 consultas mensuales y 2 consultas por segundo. Con ese usuario lo siguiente que necesitará es una clave (API Key)

o User license key), ya que cada solicitud que se envía al API se debe incluir ese valor.

Todos los pasos anteriores los encontrará en detalle en [Itinerario 2: Programación Integrativa](#), baja el tema semana 11. Es momento de revisar el código Java que hace la llamada, se han omitido varios elementos y se presenta únicamente la parte medular.

```
String mensaje = """
    El 8 de diciembre, Loja celebra 472 años de fundación, ...
    """;
String msjEncoded = URLEncoder.encode(mensaje, "UTF-8");
HttpResponse<String> response =
    Unirest.post("http://api.meaningcloud.com/topics-2.0")
        .header("content-type", "application/x-www-form-urlencoded")
        .body("key=API_KEY&lang=es&txt="+msjEncoded+"&tt=ectmn")
        .asString();
System.out.println(response.getBody());
```

Del código anterior, es necesario comentar que utiliza una cliente liviano denominado Unirest que se encarga de las llamadas al API REST que proporciona MeaningCloud. La llamada (request) se hace a través del método POST y se agrega la cabecera Content-Type para señalar que los datos de la solicitud viajan en el cuerpo (body) como una lista de tuplas (clave/valor) concatenadas con & y sus valores se encuentran codificados.

La siguiente acción es construir el contenido a enviar, utilizando el método body se agrega el API key (clave key), el idioma del texto (clave lang), el texto (clave txt) y la lista de los tipos de tópicos a extraer (clave tt).

De la lista anterior de tuplas es necesario explicar un poco más la lista de tipos de tópicos (tt=ectmn). Según la [documentación](#), cada letra tiene un significado así:

- e: entidades nombradas.
- c: conceptos.
- t: expresiones de tiempo.
- m: expresiones relacionadas a monedas.
- n: expresiones relacionadas a cantidades.

Como puede ver, las entidades son un tipo de tópico que la herramienta puede extraer. Esta es una característica propia de la herramienta, posiblemente otras API no tendrán la misma capacidad, inclusive existirán otras que tendrán muchas más capacidades.

Estoy seguro de que implementar ese código será tarea compleja. Ahora revise la respuesta que devuelve el servicio REST que está utilizado.

La respuesta es un objeto JSON, este tema se mencionó brevemente en el primer bimestre, más adelante se hará una revisión detallada del mismo que tiene varios atributos que corresponden con cada uno de los tipos de tópicos que se describió párrafos arriba. Un ejemplo de la salida, modificada para reducirla, se muestra a continuación:

```
{  
    "entity_list": [  
        {  
            "form": "Loja",  
            "id": "1755074ccc",  
            "sementity": {  
                "class": "instance",  
                "fiction": "nonfiction",  
                "id": "ODENTITY_CITY",  
                "type": "Top>Location>  
GeoPoliticalEntity>City"  
            },  
            "relevance": "100"  
        },  
        {  
            "form": "fundación",  
            "id": "302fc322a0",  
            "sementity": {  
                "class": "class",  
                "fiction": "nonfiction",  
                "id": "ODENTITY_TOP",  
                "type": "Top"  
            }  
        }  
    ],  
    "concept_list": [  
        {  
            "form": "fundación",  
            "id": "302fc322a0",  
            "sementity": {  
                "class": "class",  
                "fiction": "nonfiction",  
                "id": "ODENTITY_TOP",  
                "type": "Top"  
            }  
        }  
    ]  
}
```

```
        },
        "relevance": "100"
    }
],
"time_expression_list": [
{
    "form": "8 de diciembre",
    "normalized_form": "|||||12|8||||",
    "actual_time": "2021-12-08",
    "precision": "day",
    "inip": "3",
    "endp": "16"
},
],
"money_expression_list": [
],
"quantity_expression_list": [
{
    "form": "472 años de fundación",
    "amount_form": "472",
    "numeric_value": "472",
    "unit": "año",
    "inip": "32",
    "endp": "52"
},
],
"quotation_list": [
{
    "form": "Son 472 años de esta bella ciudad que nos desafía a diario, tierra de artistas, intelectuales, gente genuina que ama y defiende su cultura",
    "inip": "866",
    "endp": "1005"
}
]
```

La salida completa la encontrará en [Itinerario 2: Programación Integrativa](#), bajo el título semana 11. Considerando que aún no conoce a detalle a JSON haré una descripción simple y apoyado en los conocimientos que usted tiene de los conceptos de la orientación a objetos.

Los atributos de ese objeto son: entity_list, concept_list, time_expresion_list, money_expression_list, quantity_expression_list, quation_list. Esos atributos representan a un grupo o lista de otros objetos que cambian su estructura según el tipo de tópico al que pertenecen.

Dentro de cada atributo se dejó un objeto para que usted pueda ver su estructura, así por ejemplo, el único objeto de la lista entity_list tiene los siguientes atributos: form, id, sementity y relevance. De seguro notó que sementity es otro objeto que tiene sus propios atributos.

Esta información es clave para comprender la respuesta que devuelve esa API web. Como ejercicio revise la Figura 23 y asocie las columnas que ahí se muestran con alguno de los atributos de la respuesta en JSON. Es necesario comentar que el JSON de la respuesta tienen mayor cantidad de detalles que los que se ve en esa figura.

Por ahora no es necesario más detalles, más adelante se retomará el tema JSON y se mostrará un mecanismo para obtener los datos de esa forma de representar información. Es momento de continuar con otro tema.

Ha visto cómo se puede construir un pequeño programa para trabajar con texto, un tipo de dato no estructurado, pero existen otros tipos dentro de esa categoría, por ejemplo audio, vídeo, imágenes, etc. ¿Cómo se puede trabajar con esos tipos de datos? Una respuesta a esa pregunta se muestra en el siguiente apartado.

4.4. Otros formatos de datos no estructurados una propuesta de trabajo

Como se mostró anteriormente, para el texto se utilizó IA para transformar texto en un grupo de datos. Ese mismo mecanismo se podría aplicar a otros formatos como vídeo, imágenes, audio, etc. Y así como se hizo con el texto, usar algún API o librería basada en IA para obtener datos, pero existen algunas alternativas no tan potentes como IA, pero que pueden ayudar a extraer información, no del contenido, de esos datos no estructurados.

En este apartado se propone utilizar los metadatos para transformar una imagen en una lista de atributos que son interesantes y que pueden ayudar en el proceso de integración. Es necesario entonces introducir el concepto de metadato.

4.4.1. Metadatos

Los metadatos son datos que describen otros datos, un ejemplo clásico de metadatos son las fichas que se utilizan en una biblioteca con el fin de buscar rápidamente una obra. Es decir, los metadatos contienen información útil, esa información puede variar según el tipo de recurso que se utiliza.

Dentro de las ciencias de la computación, los metadatos han sido utilizados, como por ejemplo para describir documentos, como las páginas web a través de las MetaTags, para realizar clasificaciones de fotografías, para describir archivos de audio, por citar unos pocos ejemplos. Además, se utilizan metadatos para describir cualquier tipo de recurso, como lo propone la Web Semántica o como lo implementa Google [Schema.org](#) y otros grandes de las TIC, usar metadatos (vocabularios) para estructurar las páginas web con el objetivo de mejorar, por ejemplo, los motores de búsqueda. Es colaboración hombre/máquina, son seres humanos que describen recursos en un formato que las máquinas pueden entender.

Como pudo leer en esa breve introducción, los metadatos son un mecanismo para estructurar la información no estructurada. Considerando el poder de los metadatos y su propósito es posible utilizarlos en la integración de datos no estructurados. Es decir, se puede obtener esos metadatos de archivos con datos no estructurados y transformar un grupo de datos que pueden utilizarse para tareas como análisis.

Para comprender mejor esta propuesta, en el siguiente apartado se muestra de forma práctica cómo se puede obtener metadatos de varios datos no estructurados.

4.4.2. Obtener metadatos desde datos no estructurados

Los metadatos que se pueden extraer desde un archivo de imagen son muchos. Un ejemplo se muestra a continuación en la Figura 25.

Figura 25.

Metadatos de una imagen



Metadata 1		Metadata 2		Metadata 3		Metadata 4	
Metadata name	Value	Metadata name	Value	Metadata name	Value	Metadata name	Value
Acquisition-Vector	U+0.09 vs. 0.19 backward	Exif GP10/Lens Model	iPhone X5 back dual camera 6mm f/2.4	GP10/Geo Latitude Ref	-79° 12' 31.30"	MP3ImageWidth	4000
Component-1	V component: Quantization-table 0, Sampling-factors 2x2/2x1	Exif GP10/Lens Specification	4.25-mm f/1.9-2.4	GP10/Geo Longitude Ref	W	MP3ImageHeight	3024
Component-2	Cb component: Quantization-table 1, Sampling-factors 1x1/1x1	Exif GP10/Recording Mode	Multi-sensor	GP10/Geo Speed	0.01 km/h	MP3Model	Apple
Component-3	Cr component: Quantization-table 1, Sampling-factors 1x1/1x1	Exif GP10/Scene Type	Normal	GP10/Geo Street Ref	none	MP3ModelName	iPhone X5
Component-Type	Baseline	Exif GP10/Sensor Type	Directly photographed image	GP10/Time Stamp	11/19/2018 00:00 (UTC)	MP3Orientation	1
Content-Type	Image/png	Exif GP10/Sensor Width	One-chip color area sensor	ICC/Color Colorant	(0,101,1,0000, 0,3744)	MP3ResolutionUnit	inch
Data-Precision	8 bits	Exif GP10/Sensor Height	4000 pixels	ICC/Color TRC	pare (0x701201) 32 bytes	MP3Software	12.4.1
Interlaced	False	Exif GP10/Subject Location	1319/2007 1270/2112	ICC/Colorimetry	0x00000000	MP3SoftwareVersion	7.0.0
Interlace-Method	None	Exif GP10/White Balance Auto	Auto white balance	ICC/CIE Colorspace	0x00000000	MP3Thumbnail	12.0
Exif/0/0/0/0/0/0/0	2018-09-04T01:29:51	Exif GP10/White-Balance-Auto	none	ICC/CIE Colorspace Name	0x00000000	MP3ThumbnailX	10
Exif/0/0/0/0/0/0/1	Apple	Exif GP10/White-Balance-Auto-Mode	RGB	ICC/Color Space	0x00000000	MP3ThumbnailY	358 (value)
Exif/0/0/0/0/0/0/2	iPhone X5	Exif GP10/White-Balance-Mode	Auto	ICC/Color Space Name	0x00000000	MP3ThumbnailZ	1
Exif/0/0/0/0/0/0/3	Top-left (size: horizontal / normal)	Exif GP10/Thumbnail-Resolution	15000 bytes	ICC/Green Colorant	(0,200, 0,0000, 0,0476)	MP3Unkown1	100
Exif/0/0/0/0/0/0/4	Int	Exif GP10/Thumbnail-Size	15000 bytes	ICC/Green Colorant	pare (0x701201) 32 bytes	MP3Unkown2	1
Exif/0/0/0/0/0/0/5	12.4.1	Exif GP10/Thumbnail-Offset	2110 bytes	ICC/Media White Point	(0,0001, 1,0000)	MP3Unkown3	4
Exif/0/0/0/0/0/0/6	72 dots per inch	Exif GP10/Thumbnail-Resolution	72 dots per inch	ICC/Profile Platform	Apple Computer, Inc.	MP3Unkown4	100
Exif/0/0/0/0/0/0/7	72 dpi	Exif GP10/Thumbnail-Orientation	Top	ICC/Profile Version	0x00000000	MP3Unkown5	100
Exif/0/0/0/0/0/0/8	Exif/0/0/0/0/0/0/0	Exif GP10/Thumbnail-Space	none	ICC/Profile Copyright	Copyright Apple Inc., 2017	MP3Unkown6	4161/002572
Exif/0/0/0/0/0/0/9	Center of pixel array	Exif GP10/Thumbnail-Time-Original	2018-09-04T01:29:51	ICC/Profile Date/Time	2017/07/17 13:23:32	MP3Unkown7	1
Exif/0/0/0/0/0/0/10	0.04	Exif GP10/Thumbnail-Width	4780 pixels	ICC/Profile Description	Display P3	MP3Unkown8	0
Exif/0/0/0/0/0/0/11	Undefined	Exif GP10/Thumbnail-Height	3412 pixels	ICC/Profile Name	0x00000000	MP3Unkown9	110/200400000
Exif/0/0/0/0/0/0/12	UnDefined	Exif GP10/Thumbnail-Depth	4.0	ICC/Red Colorant	(0,01, 0,01, 0,0000)	MP3Unkown10	394
Exif/0/0/0/0/0/0/13	YDQQ	Exif GP10/Thumbnail-Media-File-Number	125	ICC/Red TRC	pare (0x701201) 32 bytes	MP3Unkown11	3
Exif/0/0/0/0/0/0/14	Unknown (2)	Exif GP10/Thumbnail-Modifed-Date	2018-09-04T01:29:51	ICC/Signature	none	MP3Unkown12	7354/0234
Exif/0/0/0/0/0/0/15	2018-09-04T01:29:51	Exif GP10/Thumbnail-Size	3011010 bytes	ICC/Version	4.0.0	MP3Unkown13	1
Exif/0/0/0/0/0/0/16	Exif/0/0/0/0/0/0/0	Exif GP10/Thumbnail-Flags	Valid	ICC/X/Y/Z Value	0.984 1.0001	Value	180/274 seconds
Exif/0/0/0/0/0/0/17	Exif/0/0/0/0/0/0/1	Exif GP10/Thumbnail-Image-Height	4000 pixels	ICC/XY Resolution	4000 pixels	Version	1.1
Exif/0/0/0/0/0/0/18	4000 pixels	Exif GP10/Thumbnail-Image-Width	301101 pixels	ICC/Z Resolution	300 pixels	X Resolution	300 dpi
Exif/0/0/0/0/0/0/19	301101 pixels	Exif GP10/Thumbnail-Media-File-Number	1	ICC/XY Units	0x00000000	Y Resolution	300 dpi
Exif/0/0/0/0/0/0/20	1	Exif GP10/Thumbnail-Offset	79	ICC/XY Units	0x00000000	Z Resolution	0
Exif/0/0/0/0/0/0/21	Auto exposure	GPS/GPS-Altitude	2147.83 meters	ICC/Photo ID	13754003	EXIF Value Count	10
Exif/0/0/0/0/0/0/22	Program	GPS/GPS-Altitude Ref	Sea level	ICC/Number of Components	4 (unknown values)	EXIF Value	org.apple.xmp.parser.DefaultParser
Exif/0/0/0/0/0/0/23	Normal	GPS/GPS-Stamp	2018/09/04	ICC/Number of Tables	0x00000000	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/24	17:17 sec	GPS/GPS-Datum	WGS_1984	ICC/Number of Tables	0x00000000	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/25	0.04	GPS/GPS-Degrees	182.29 degrees	ICC/Row Units	inch	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/26	0.04	GPS/GPS-Degrees-Bearing	182.29 degrees	ICC/Row Units	0x00000000	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/27	Pixel off the line, auto	GPS/GPS-Degrees-Fractional	1.22 meters	ICC/Scale	1000000000	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/28	1.00	GPS/GPS-Degrees-Ref	182.29 degrees	ICC/Thermal Height Pixels	0	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/29	6 mm	GPS/GPS-Direction	True direction	ICC/Thermal Width Pixels	0	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/30	52 mm	GPS/GPS-Direction-Ref	4° 47' 46.2"	ICC/Unknown	0	EXIF Value	0x00000000
Exif/0/0/0/0/0/0/31	Exif/0/0/0/0/0/0/0	Exif GP10/Unknown		ICC/Unknown	0	EXIF Value	0x00000000

Son alrededor de 150 metadatos los que se obtuvieron, aunque muchos de esos valores son pocos conocidos, sobre todo si no se está relacionado con el tema de la fotografía, existen algunos metadatos conocidos como latitud, longitud y altitud. Además, los metadatos no describen el contenido de la fotografía, por lo contrario, son datos asociados al archivo en donde se almacena la imagen y a la cámara que se utilizó para capturarla.

Los metadatos varían según su tipo, es decir, las imágenes tendrán un grupo de metadatos, los videos otros, así como también los basados en texto, inclusive se puede tener diferentes metadatos por cada subtipo, por ejemplo, entre imágenes JPG y PNG pueden existir diferentes metadatos.

Considerando que los metadatos de una imagen son poco conocidos, le propongo revisar los metadatos que se pueden obtener desde un archivo PDF. En el documento [Metadatos de un archivo PDF](#) se muestran los metadatos que se obtuvieron desde un archivo PDF.

Con estos dos ejemplos, se ha mostrado que los metadatos se pueden utilizar para transformar un archivo con datos no estructurados en un grupo de datos que lo describen y que algunos de esos datos se podrían utilizar para realizar análisis que pueden ayudar a quien los utiliza.

Existe mucho más por estudiar en relación con los metadatos, pero ahora es necesario pasar a la parte práctica y de experimentación en donde se

aprenderá a obtener metadatos utilizando un lenguaje de programación. Avance al siguiente apartado.

4.4.3. Extraer metadatos a través de un programa

En el apartado anterior se presentaron los metadatos de dos tipos de datos no estructurados, imagen y texto. Esa metadata reside dentro del archivo en donde se almacena, es decir, existe un grupo de bits que se dedican para el almacenamiento de datos. Ese espacio se encuentra bien delimitado para así evitar confundirlo con el espacio de los datos que son el propósito del archivo.

A nivel de los lenguajes de programación, existen varias librerías especializadas para la obtención de los metadatos según el tipo de archivo con el que trabaja. Esto es una señal de la diversidad de los metadatos. Para esta semana se usará la librería [Apache Tika](#) que es un framework que entre sus características permite obtener metadatos desde diferentes archivos.

El código que se utilizó para obtener los metadatos que corresponden a la imagen que se encuentra en Figura 25 es el que se muestra a continuación:

```
BodyContentHandler handler = new BodyContentHandler();
Metadata metadata = new Metadata();
FileInputStream inputstream = new FileInputStream(new
File(RUTA_ARCHIVO));
ParseContext pcontext = new ParseContext();

AutoDetectParser autoParser = new AutoDetectParser();
autoParser.parse(inputstream, handler, metadata, pcontext);
System.out.println("Metadatos:");
String[] metadataNames = metadata.names();
for(String name : metadataNames) {
    System.out.println(name + ": " + metadata.get(name));
}
```

El código anterior utiliza objetos propios de Apache Tika entre los que se destacan *AutoDetectParser* y *Metadata*. La primera tiene como objetivo determinar, de forma automática, cuál es el tipo de contenido del archivo para seleccionar la forma de leerlo. En la segunda clase, *Metadata*, su

nombre señala ya su propósito, almacenar todos los metadatos encontrados en el archivo.

Recuerde, para que el código anterior funcione se debe proporcionar la ruta del archivo con lo imagen, es decir, un valor a la constante RUTA_ARCHIVO.

De seguro encontró el código bastante sencillo y fácil de comprender, pero le invito a revisar todos los pasos que se necesitan para implementar el programa, descritos a detalle se encuentran en [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 11.

Ahora revise el código que se usó para extraer los metadatos de un archivo PDF. Es necesario mencionar que Apache Tika, además de obtener metadatos, puede leer el contenido de los archivos que se basan en texto, si a esto se lo combina con lo visto la semana pasada (reconocimiento de entidades) se puede construir un proceso de integración interesante, pero antes de más detalles se presenta y comenta brevemente el código:

```
BodyContentHandler handler = new BodyContentHandler(-1);
PDFParser parser = new PDFParser();
Metadata metadata = new Metadata();
ParseContext pcontext = new ParseContext();
InputStream stream = new FileInputStream(new File(RUTA_
ARCHIVO));
parser.parse(stream, handler, metadata, pcontext);
System.out.printf("Document Content:\n%s\n", handler);
System.out.println("Document Metadata:");
String[] metadataNames = metadata.names();
for (String name : metadataNames) {
    System.out.printf("%s: %s\n", name, metadata.get(name));
}
```

El código es bastante parecido al anterior, eso demuestra la consistencia del framework utilizado. Los elementos a destacar de esta son la clase PDFParser y BodyContentHandler. PDFParser es una clase especializada para manejar archivos PDF que además de obtener los metadatos permite leer el contenido de un archivo PDF. La clase BodyContentHandler se crea con un parámetro-1 para señalar que debe leer todo el contenido del archivo.

Para el manejo de los metadatos, el procedimiento es el mismo que se utilizó para las imágenes.

Recuerde que el código completo y la descripción en detalle del código anterior lo encontrará en [Itinerario 2: Programación Integrativa](#), dentro de los contenidos del apartado semana 11.

Luego de revisar estos ejemplos integración, se presenta un escenario de uso para la integración de datos no estructurados. Imagine que está digitalizando los documentos de una organización. Muchos de esos datos son archivos de texto, podría entonces usar el texto para extraer entidades mencionadas en el texto y los metadatos para conocer, por ejemplo, quién fue el autor del documento. Con esa información podría hacer un análisis y llegar a determinar relaciones entre las personas mencionadas en un documento y los autores de ese documento. Así podría determinar la pertinencia de esa relación. Esto es solo un escenario, de seguro usted puede pensar en otros más.

Hasta aquí el desarrollo de los contenidos para esta semana en donde se expuso de forma práctica el trabajo con datos no estructurados y se presentó una posibilidad de integración a través del uso de IA y metadatos.

La próxima semana estudiará la integración de datos semiestructurados, pero antes de continuar se recomienda que realice las siguientes actividades.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Implementar el código que se encuentra en [Itinerario 2: Programación Integrativa](#).
- Crear una cuenta en MeaningCloud. El proceso está descrito en [Itinerario 2: Programación Integrativa](#). Busque el apartado semana 11.
- Revisar la explicación de la implementación del caso desarrollado en la unidad, disponible en [Itinerario 2: Programación Integrativa](#), bajo el tema semana 11.



Unidad 5. Integración de datos semiestructurados

Continuando con la integración de los diferentes tipos de datos, es momento de estudiar los datos semiestructurados, así como se lo hizo anteriormente estudiara la integración desde el punto de vista teórico y sobre todo desde el práctico.

Como se mencionó anteriormente, los datos semiestructurados se refieren a los datos que se formatean de acuerdo con ciertas reglas aceptadas, pero que pueden variar en estructura dependiendo de los datos que se proporcionen. Entre los ejemplos de datos semiestructurados posiblemente XML (Lenguaje de Marcado Extensible), CSV(Comma Separated Values) y JSON (notación de objetos JavaScript) los más conocidos, existen también otros formatos que no son ampliamente conocidos como por ejemplo Avro, ORC, Parquet.

Lin & et al. (2018) sostienen que los datos semiestructurados tienen las siguientes características:

1. Tienen cierta estructura, pero la estructura y el contenido están mezclados.
2. El conjunto de datos puede estar compuesto de elementos heterogéneos y la información puede estar representada por diferentes tipos de datos.
3. No existe un modelo de datos predefinido, los datos tienen una estructura irregular, por lo que las restricciones por tipo de dato no existe. Es decir, un atributo puede tener valores numéricos en una instancia y en otra el mismo atributo puede tener valores de cada tipo de texto.

Considerando las características anteriores, para el almacenamiento de este tipo de datos no se puede usar un mecanismo basado en base de datos relaciones (donde se tiene un modelo de datos definido). La tendencia actual es almacenarlos en base de datos NoSQL.

Recuerde que en este tipo de datos la definición de la estructura, también conocido como modelo, de los datos y su significado se encuentran dentro del archivo, a diferencia de las bases de datos en donde la definición de la estructura se encuentra por separado. La definición del modelo de datos se hace a través de etiquetas o ciertas marcas, así se puede diferenciar la semántica de los datos de su contenido.

En lo referente al procesamiento de datos, los métodos que directamente procesan los datos semiestructurados son pocos, la mayoría del procesamiento consiste en transformar datos semiestructurados en estructurados.

Para conocer acerca de las ventajas y desventajas de los datos semiestructurados lea el recurso Ortiz (2021) que hace una buena explicación del tema. Entre las ventajas a destacar está la simplicidad para representar relaciones jerárquicas complejas. Mientras que entre las desventajas se encuentra que al no existir unas restricciones, es muy fácil tener datos incorrectos.

Hasta aquí esta breve descripción de los datos semiestructurados, es momento de pasar a estudiar una de las representaciones más populares de este tipo de datos, los archivos JSON que se desarrolla en el siguiente apartado.

5.1. JSON - JavaScript Object Notation

Uno de los formatos de datos más populares hoy en día es JSON que es un acrónimo de notación de objetos JavaScript. Es un formato ligero de intercambio de datos de fácil escritura y lectura, además, para un computador es bastante sencillo de analizar. Otra característica bastante importante es que es independiente del lenguaje de programación por lo que se convierte en una alternativa viable a XML.

Para entrar en los detalles y sobre todo conocer la sintaxis de JSON, debe revisar el siguiente recurso: ECMA.(nd). [Introducción a JSON](#). Revise hasta el tema Objeto y regrese a este documento para que ponga en práctica los conceptos que ahí se presentan.

Antes de ir a la parte práctica, es necesario recalcar que los tipos de datos de los atributos de un objeto JSON están determinados por el valor. Revise el siguiente ejemplo para aclarar este tema.

Utilizando el diagrama de objetos de la izquierda, se construyó un objeto JSON equivalente, el resultado se muestra a la derecha.

Figura 26.

Objeto de la clase estudiante en Java

<u>jorgeCabrera :Estudiante</u>	{
nombre = Jorge	"nombre": "Jorge",
apellido = Cabrera	"apellido": "Cabrera",
edad = 7	"edad": 7,
peso = 45.8	"peso": 45.8,
mayorEdad = false	"mayor_edad": false
	}

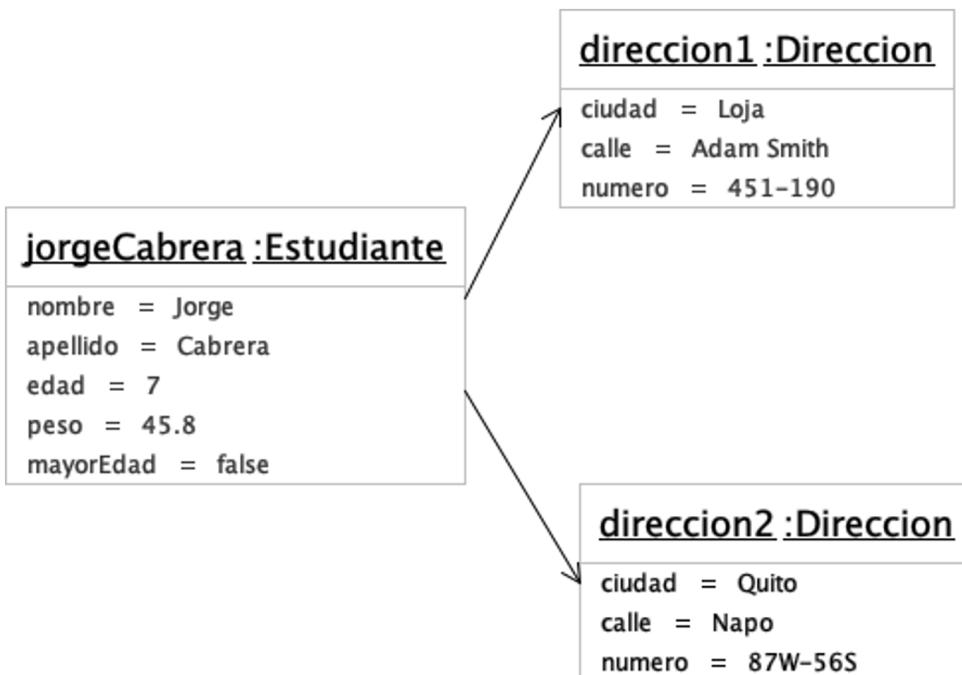
Observe las diferencias entre ambas implementaciones, además, la forma o estilo de nombrar los atributos cambian, un ejemplo es el atributo mayor de edad, en Java se usa el estándar propuesto, mientras que en JSON se usa el guion bajo para separar las palabras. Observe también que como se cumple lo dicho sobre los datos semiestructurados no existe un modelo de datos, el modelo está mezclado con los datos, es así que para determinar el tipo de dato de un atributo se usa el valor del atributo, así se puede decir que peso es de tipo real, ya que 45.8 es de ese tipo de dato.

Para continuar es tiempo de revisar el recurso anterior, esta vez lea el tema arreglos. Una vez que concluya la revisión, regrese para continuar con la explicación de la parte práctica.

Suponga ahora que un estudiante puede tener varias direcciones y que cada dirección tiene ciertos atributos como los que muestra la Figura 27.

Figura 27.

Diagrama de objetos con relaciones



La representación en JSON se hace de la siguiente forma:

```
{  
    "nombre": "Jorge",  
    "apellido": "Cabrera",  
    "edad": 7,  
    "peso": 45.8,  
    "mayor_edad": false,  
    "direcciones": [  
        {  
            "ciudad": "Loja",  
            "calle": "Adam Smith",  
            "numero": "451-190",  
        },  
        {  
            "ciudad": "Quito",  
            "calle": "Napo",  
        }  
    ]  
}
```

```

        "numero": "87W-56S",
    }
]
}

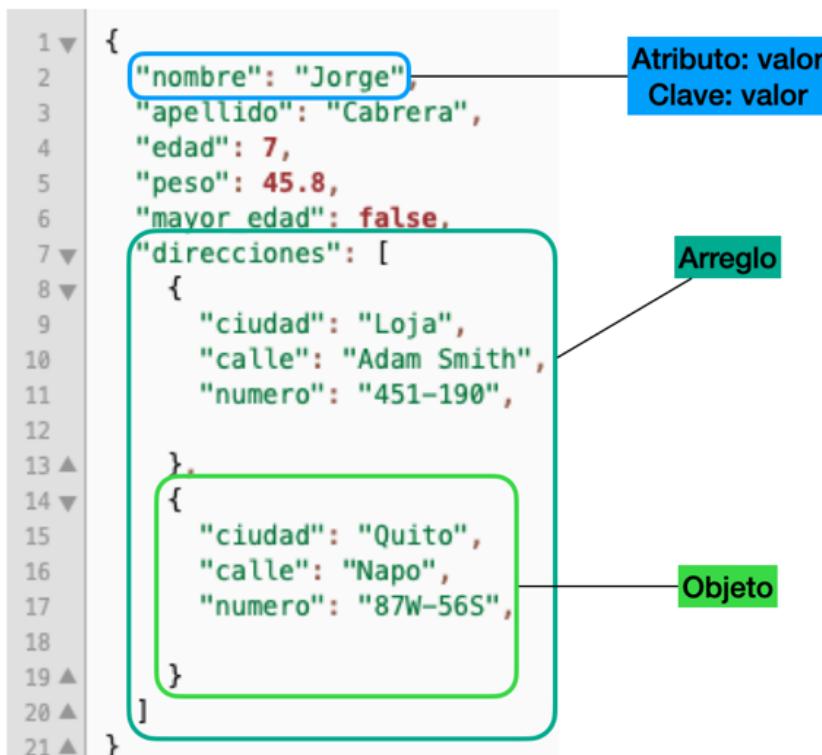
```

Observe cómo la relación entre estudiante y dirección se representa a través de un arreglo, ya que un estudiante puede tener varias direcciones, dentro del arreglo se representa a cada dirección como un objeto JSON con sus respectivos datos.

Antes de continuar, revise la Figura 28 que describe los elementos que se han creado. Note la naturaleza jerárquica de JSON, un objeto que contiene un arreglo de objetos.

Figura 28.

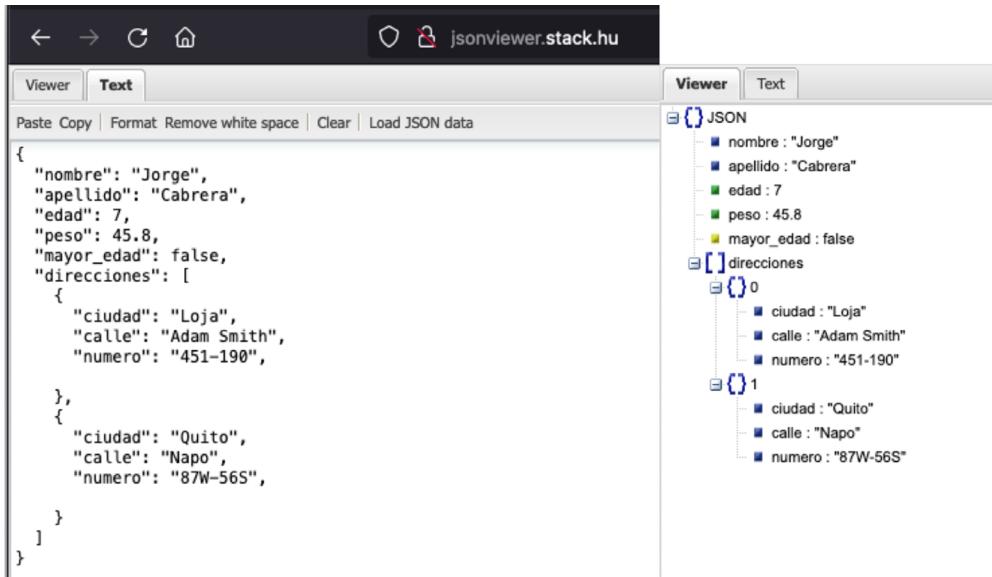
Descripción gráfica de los datos creado en JSON



Como pudo experimentar, la sintaxis de JSON no es complicada de comprender y con la práctica se puede evitar los errores de sintaxis que son los más comunes en este formato de datos. Una herramienta que es

bastante útil para trabajar con JSON es [JSONViewer](#) ya que, entre otras cosas, permite formatear sintaxis, verificar y visualizar datos escritos en JSON. En la Figura 29 se muestra la herramienta.

Figura 29.
JSONViewer con sus dos opciones



En la representación visual (derecha) se puede ver la jerarquía de JSON, además del uso de llaves y corchetes para señalar los tipos de los elementos, objeto y arreglo, JSON respectivamente, mientras que los atributos o pares clave/valor tiene un cuadrado de color según su tipo de dato, azul para texto, verde para números y amarillo para valores lógicos. Hoy en día existen muchas herramientas similares a la que aquí se presentó, esto lo puede verificar con una búsqueda en la Web.

Ahora que ya conoce cuál es la sintaxis de JSON, es conveniente que estudie cómo se puede trabajar los datos JSON desde el lenguaje de programación, parte de estos temas, la creación, ya lo revisó brevemente cuando estudió los servicios web basados en REST, es por ello que se revisará brevemente. Un segundo tema que estudiar es la lectura de los datos que están representados como JSON, este es un tema nuevo e importante, sobre todo en procesos de integración de datos. En los siguientes apartados se presentan estos temas.

5.1.1. Escritura de JSON

Dentro de los lenguajes de programación, generalmente existen varias alternativas para realizar una misma tarea, y la escritura de JSON no es la excepción.

Le invito a profundizar su conocimiento sobre estas alternativas.

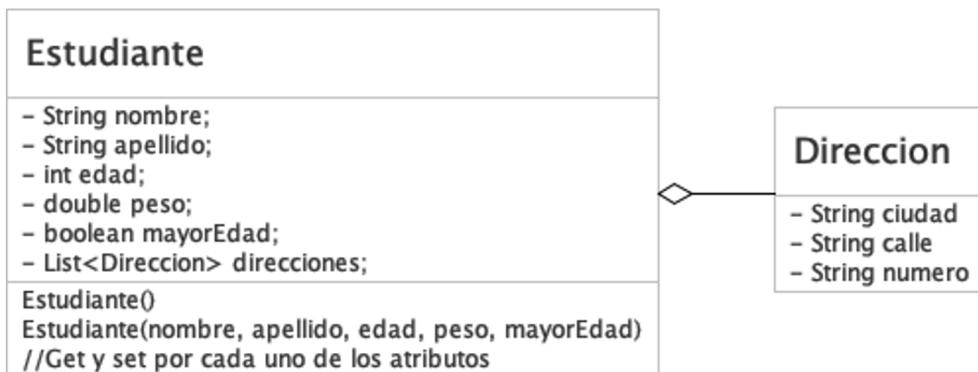
En los apartados REST y microservicios conoció dos formas de producir información en formato JSON.

La primera forma que estudió es la creación de objetos JSON a través de la transformación de clases y objetos Java en elementos JSON, a este proceso se lo denomina serialización que se basa en la idea de “mapear” los datos de un objeto Java en un objeto JSON. Este tema es mucho más fácil de comprender si se lo estudia de forma práctica.

El código que se usará se basa en el mismo que se usó anteriormente para explicar la sintaxis de JSON (Estudiante y Dirección). El diagrama de clases que se utilizó es el siguiente:

Figura 30.

Diagrama de clases del ejercicio



La relación entre las clases es una relación de agregación que se resuelve creando el atributo **direcciones** en la clase **Estudiante**, así se señala la multiplicidad de la relación (un estudiante puede tener varias direcciones).

Una vez revisado el diagrama de clases, revise el código de las clases, inicie con la clase **Estudiante**.

```
public class Estudiante {  
    private String nombre;  
    private String apellido;  
    private int edad;  
    private double peso;  
    @SerializedName(value="mayor_edad")  
    private boolean mayorEdad;  
    private List<Direccion> direcciones;  
  
    public Estudiante(String nombre, String apellido, int edad,  
double peso, boolean mayorEdad) {  
        this.nombre = nombre;  
        this.apellido = apellido;  
        this.edad = edad;  
        this.peso = peso;  
        this.mayorEdad = mayorEdad;  
    }  
    public void setDirecciones(List<Direccion> direcciones) {  
        this.direcciones = direcciones;  
    }  
}
```

La clase es sencilla, tiene sus atributos, un constructor para asignar un valor a cada atributo y el método *setDirecciones* que asigna un valor al atributo *direcciones*. Como puede ver, es una clase bastante común, excepto por la anotación *SerializedName* que se utiliza para especificar cuál debe ser el nombre de ese atributo cuando se serialice en JSON. Esta anotación es necesaria únicamente en los atributos cuyos nombres en Java difieren a los nombres que se usará en JSON.

Ahora se presenta el código de la clase *Direccion*, esta clase es un tipo nuevo de clase que se denomina record y que busca evitar escribir mucho código para crear una clase que solo tiene atributos.

```
public record Direccion(String ciudad, String calle, String numero) {  
}
```

Este es el código que representa al modelo de datos que se va a utilizar para generar el JSON. Recuerde que el código completo está disponible en [Itinerario 2: Programación Integrativa](#)

Ahora debe estudiar el código que permite crear un objeto de la clase estudiante con dos direcciones y el método que le permite serializar un objeto. Comience con la creación de los datos.

```
private static Estudiante generateData() {  
    Estudiante est = new Estudiante("Jorge", "Cabrera", 7, 45.8,  
    false);  
    List<Direccion> direcciones = List.of(  
        new Direccion("Loja", "Adam Smith", "451-190"),  
        new Direccion("Quito", "Napo", "87W5-6S")  
    );  
    est.setDirecciones(direcciones);  
    return est;  
}
```

El código anterior se puede resumir así: en primer lugar crea un objeto de la clase Estudiante, para luego crear una lista de direcciones con el fin de asignarlas al estudiante y finalmente devolver el estudiante creado con sus direcciones.

La parte medular de la serialización se encuentra en el siguiente método:

```
public static void main(String[] args) {  
    Gson gson = new Gson();  
    //Gson gson = new GsonBuilder().setPrettyPrinting().create();  
    var estudiante = generateData();  
  
    String estJSON = gson.toJson(estudiante);  
    System.out.println(estJSON);  
}
```

Se utiliza la librería *Gson* para serializar un objeto de la clase estudiante. Lo primero es crear un objeto del tipo *Gson*, luego se obtiene un estudiante con

todos sus datos, finalmente se serializa (convertir) el objeto estudiante a cada texto escrito según la sintaxis de JSON.

De seguro se está preguntando sobre la línea que se encuentra comentada, esa línea crea un objeto del tipo Gson, pero configurada para que el JSON que se genera se encuentre formateado y sea visualmente agradable como lo verá a continuación. Usted puede seleccionar cualquiera de las dos formas de creación.

Finalmente el JSON que se creó es el siguiente:

```
{  
    "nombre": "Jorge",  
    "apellido": "Cabrera",  
    "edad": 7,  
    "peso": 45.8,  
    "mayor_edad": false,  
    "direcciones": [  
        {  
            "ciudad": "Loja",  
            "calle": "Adam Smith",  
            "numero": "451-190"  
        },  
        {  
            "ciudad": "Quito",  
            "calle": "Napo",  
            "numero": "87W5-6S"  
        }  
    ]  
}
```

Es idéntico al que se creó manualmente, observe también cómo el atributo que señala si el estudiante es mayor de edad cumple con lo especificado en el ejercicio original, gracias a la anotación sobre el atributo *mayorEdad* de la clase *Estudiante*.

Recuerde que el código completo, así como una explicación detallada del mismo se encuentra en [Itinerario 2: Programación Integrativa](#), bajo apartado semana 12.

Esta forma de generar JSON es bastante popular dentro del lenguaje de programación Java, si bien no es la única es bastante sencilla y evita la creación manual de JSON que es un tarea que puede ser engorrosa, especialmente cuando el JSON a escribir es extenso.

Si quiere conocer otra forma le sugiero revisar la Práctica 1 que se encuentra en [Itinerario 2: Programación Integrativa](#), bajo apartado semana 12.

Ahora ya refrescó sus conocimientos sobre la escritura de JSON desde el lenguaje de programación Java. Debe continuar estudiando la forma para leer JSON, tema que se presenta a continuación.

5.1.2. Lectura de JSON

Al igual que la escritura, existen varias formas de leer los datos que se han escrito usando la sintaxis JSON. La que estudiará aquí es el proceso *binding* que consiste en el proceso contrario de la serialización, es decir, convertir elementos JSON a objetos Java.

Una característica del *binding* es que necesita crear las clases Java y su jerarquía previamente, ya que los objetos JSON se convertirán en objetos Java y en ese lenguaje de programación no se puede tener un objeto sin antes haber creado una clase. Esto tiene un impacto directo en el tiempo de desarrollo, debido a que la primera acción a ejecutar es conocer los objetos JSON y su jerarquía para luego replicarlos, pero dentro de Java.

Con esa breve introducción, es tiempo de presentar el código que se usará para la lectura JSON a través del proceso de *binding*. Aquí es necesario mencionar que las clases *Estudiante* y *Dirección* deben tener ciertas características para poder ser parte de un proceso de *binding*, el primero es tener un constructor por defecto y los métodos set y get de cada uno de los atributos. Esas características traen como consecuencia que *Dirección* ya no sea un record, sino una clase común de Java.

El código de la clase *Estudiante* es el siguiente:

```
public class Estudiante {  
    private String nombre;  
    private String apellido;  
    private int edad;  
    private double peso;  
    @SerializedName(value = "mayor_edad")  
    private boolean mayorEdad;  
    private List<Direccion> direcciones;  
  
    public Estudiante() {  
    }  
  
    public Estudiante(String nombre, String apellido, int edad,  
double peso, boolean mayorEdad) {  
        this.nombre = nombre;  
        this.apellido = apellido;  
        this.edad = edad;  
        this.peso = peso;  
        this.mayorEdad = mayorEdad;  
    }  
    //Set y get de todos los atributos  
}
```

Note cómo se sigue usando la anotación *SerializedName*, así *Gson* conocerá a qué atributo Java se debe asignar el valor del atributo *es_mayor* que se encuentra en JSON.

El código para la clase Dirección es el siguiente

```
public class Direccion {  
    private String ciudad;  
    private String caller;  
    private String numero;
```

```
public Direccion() {  
}  
  
public Direccion(String ciudad, String caller, String numero) {  
    this.ciudad = ciudad;  
    this.caller = caller;  
    this.numero = numero;  
}  
//Get y set de todos los atributos.  
}
```

Al igual que la clase *Estudiante*, cumple con las características que el proceso de *binding* exige, constructor por defecto, get y set de cada atributo, así se puede usar en el proceso de convertir JSON a Java.

Una vez que conoce la estructura de las clases del modelo, es necesario que revise el código para leer JSON a través de *binding*.

```
public static void main(String[] args) {  
    String json = """"  
    {  
        "nombre": "Josué",  
        "apellido": "Piedra",  
        "edad": 7,  
        "peso": 35.5,  
        "mayor_edad": false,  
        "direcciones": [  
            {  
                "ciudad": "Loja",  
                "calle": "El Salvador",  
                "numero": "488-19"  
            },  
            {  
                "ciudad": "Guayaquil",  
                "calle": "Juan Tanca  
Marengo",  
                "numero": "123-654"  
            }  
        ]  
    }  
}"""
```

```
        }
    ]
}

Gson gson = new Gson();
Estudiante est = gson.fromJson(json, Estudiante.class);
System.out.printf("Apellido: %s\nNombre:%s\n",
    est.getApellido(),
    est.getNombre());
}
```

En esta clase, lo primero que debió notar es que los datos escritos en JSON están asignados a una variable, de hecho podría venir desde un archivo o desde un servicio web. Se crea el objeto gson y se llama al método fromJson señalando en dónde están los datos en JSON y a qué clase se debe convertir esos datos. Para verificar el correcto funcionamiento se cambiaron los datos que están dentro del JSON.

Con el proceso de *binding* se puede aprovechar las características de Java para el manejo de objetos, no olvide que Java es orientada a objetos, por lo tanto, las clases y los objetos son conceptos fundamentales para ese lenguaje.

Como ya conoce, el código completo está disponible en [Itinerario 2: Programación Integrativa](#) y una explicación detallada del proceso de creación del programa en [Itinerario 2: Programación Integrativa, debe buscar semana 12.](#)

Si quiere conocer otra forma de trabajo debe revisar la Práctica 2 que se encuentra en bajo apartado Semana 12.

Con este tema se cierra el desarrollo de los contenidos de esta semana que tuvo por objetivo presentar la integración de los datos semiestructurados a través de la manipulación de datos escritos en JSON. Además, puede ver y experimentar con la lectura/escritura de JSON usando un lenguaje de programación.

Para la siguiente semana estudiará cómo leer información de otro formato de datos semiestructurado.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Implementar el código desarrollado en esta unidad, disponible en [Itinerario 2: Programación Integrativa](#).
- Revisar las explicaciones de la construcción de cada una de las aplicaciones elaboradas, disponible en [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 12.



Semana 13

5.2. Extensible Markup Language XML

Si bien este tema se lo presentó en el primer bimestre cuando se habló de SOAP, es necesario retomarlo y estudiar cómo se puede leer el contenido de un archivo XML, ya que anteriormente estudió cómo escribir o convertir objetos Java en ese formato.

Como ya se mencionó, XML es un formato de datos semiestructurados que se usa ampliamente como un medio para intercambiar información, no solamente en los mensajes SOAP, sino que también como un formato para compartir información, así por ejemplo, el Servicio de Rentas Internas del Ecuador (SRI) utiliza XML para la facturación electrónica, como usuario de seguro ha recibido facturas electrónicas en su correo electrónico y uno de los adjuntos que recibe de seguro es un archivo XML que representa a una factura, pero escrita en una estructura predefinida. Ese archivo XML está diseñado como un medio de integrar información.

Además, muchas herramientas bastante populares han utilizado como base para construir sus formatos de almacenamiento, por ejemplo, Microsoft Word utiliza XML en sus archivos docx. Otras herramientas también lo utilizan, principalmente aquellas en donde los datos que se pretende almacenar tienen una estructura jerárquica que fácilmente se puede representar en XML.

Al igual que sucede con JSON, existen muchas formas de leer un archivo XML, pero de forma general existen dos grandes grupos. En el primer grupo busca recorrer la estructura del XML a través de una manipulación directa del Modelo de Objetos del Documento o DOM que permite acceder y manipular documentos XML. El segundo grupo se basa en la idea de transformar el contenido del XML en un grupo de objetos que pertenecen a clases previamente construidas.

Antes de pasar a revisar un ejemplo de cada una de las implementaciones debe revisar los datos XML con los que se trabajará en ambas implementaciones. Así como se hizo con JSON se trabajará con un estudiante que puede tener varias direcciones, una representación de esa información es la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<estudiante id="321">
    <apellido>Piedra</apellido>
    <direcciones>
        <direccion type="main">
            <calle>El Salvador</calle>
            <ciudad>Loja</ciudad>
            <numero>488-19</numero>
        </direccion>
        <direccion type="alt">
            <calle>Juan Tanca Marengo</calle>
            <ciudad>Guayaquil</ciudad>
            <numero>123-654</numero>
        </direccion>
    </direcciones>
    <edad>7</edad>
    <mayorEdad>false</mayorEdad>
    <nombre>Francisco</nombre>
    <peso>35.5</peso>
</estudiante>
```

Se han agregado cierta información que no estaba cuando se trabajó con JSON, en primer lugar, el estudiante tendrá una identificación que en XML

se ha representado como el atributo id de la etiqueta estudiante. El segundo cambio está en cada una de las direcciones, en este caso se agregó un atributo type (tipo) a cada dirección (ver etiqueta dirección) que señala cuál es la dirección principal y cuál es la dirección alterna. Esos son los cambios que se hicieron a los datos.

Es momento de pasar al código y revisar dos ejemplos, el primero aquel que transforma el XML en objetos Java y el segundo que usa el DOM para leer el XML. Comenzará revisando un ejemplo del primer tipo.

5.2.1. XML a objetos Java

La principal idea de esta aproximación es convertir los datos de XML a objetos Java. Esos objetos deben pertenecer a clases previamente creadas y usan ciertas anotaciones para señalar su relación con la estructura del archivo XML.

Esta aproximación es la misma que usó en el primer bimestre, pero en sentido contrario, ya que en ese bimestre fue desde Java a XML y ahora es desde XML a Java.

Ahora revise el código comience con el método principal.

```
public static void main(String[] args) throws JAXBException {  
    String xmlData = """";  
        <?xml version="1.0" encoding="UTF-8"?>  
        ...  
        """;  
    JAXBContext jaxbContext = JAXBContext.  
newInstance(Estudiante.class);  
    Unmarshaller unmarshaller = jaxbContext.  
createUnmarshaller();  
    StringReader reader = new StringReader(xmlData);  
  
    Estudiante estudiante = (Estudiante) unmarshaller.  
unmarshal(reader);  
    viewEstudianteData(estudiante);  
}
```

Algunos elementos a destacar. Para optimizar espacio, no se presenta todo el código de la variable *xmlData*, pero ahí va el mismo contenido que se presentó al finalizar la primera sección de semana. Otro elemento a destacar es el uso de la clase *Unmarshaller* que será la encargada de convertir el XML en objetos Java, esto se hace a través del método *unmarshal* que recibe como parámetro el contenido XML a procesar. Finalmente, se transforma a un objeto de la clase *Estudiante* a través del proceso denominado casting. La última sentencia del método principal es invocar a otro método que se encarga de obtener los datos de los objetos creados (estudiante y dirección) para generar la siguiente salida.

Datos del estudiante:

Id: 321

Piedra, Francisco

Edad: 7 años. Peso: 35,50KG. Mayor de edad: false

Direcciones

=====

Tipo: main

Ciudad: Loja

Calle: El Salvador

Número: 488-19

=====

Tipo: alt

Ciudad: Guayaquil

Calle: Juan Tanca Marengo

Número: 123-654

Ahora revise el código de las clases que reciben los datos desde XML. Inicie con la clase *Dirección*.

```
@XmlRootElement(name = "direccion")
@XmlAccessorType(XmlAccessType.FIELD)
public class Direccion {
    @XmlAttribute
    private String type;
    private String calle;
    private String ciudad;
```

```
private String numero;  
  
    public Direccion() {  
    }  
}
```

Es una clase sencilla en donde se usan anotaciones para establecer la correspondencia con los datos en XML. Así, XmlRootElement señala que los datos que corresponden a esta clase están bajo la etiqueta XML dirección. XmlAccessorType por su lado señala que todos los atributos de la clase se vinculan con las propiedades que están dentro de la etiqueta dirección en XML. Aquí un comentario adicional, esto es posible gracias a que tanto en XML como en Java se usan los mismos nombres. Finalmente, la anotaciónXmlAttribute señala que el valor para el atributo type en XML está representado como un atributo y no como una propiedad.

No olvide que estas mismas anotaciones se utilizaron anteriormente para construir un ejemplo que representaba a campos y tablas, esto en el primer bimestre en el apartado SOAP.

Ahora revise la clase que representa al estudiante, esta clase tiene un atributo que representa a la relación con la clase dirección, recuerde que un estudiante tiene varias direcciones. A continuación el código:

```
@XmlRootElement(name = "estudiante")  
@XmlAccessorType(XmlAccessType.FIELD)  
public class Estudiante {  
    @XmlAttribute  
    private String id;  
    private String nombre;  
    private String apellido;  
    private int edad;  
    private double peso;  
    private boolean mayorEdad;  
    @XmlElementWrapper(name = "direcciones")  
    @XmlElement(name = "direccion")  
    private List<Direccion> direcciones;
```

```
public Estudiante() {  
}  
}
```

Al igual que la clase anterior, se emplean las mismas anotaciones para señalar la correspondencia con los datos en XML, únicamente señalar que la relación con las direcciones se resuelve con una lista de objetos de tipo Dirección. Esa lista se marca con las anotaciones *XmlElementWrapper* que señala cuál es el nombre de la propiedad XML en donde se encuentran los datos, mientras que *XmlElement* señala cuál es el contenido de esa lista y así el lenguaje conoce cómo proceder (igual que con la clase *Dirección*).

Estas anotaciones se usaron también cuando se trabajó en la escritura desde objetos Java a XML.

Como pudo observar, el trabajar de esta manera es sencillo, pero demanda construir un conjunto de clases para así tener una correspondencia directa con el contenido del archivo XML. Además, cada clase debe contar con las anotaciones necesarias para que esa correspondencia quede clara.

Recuerde que aquí se presentan extractos del código, para acceder al código completo puede ir al repositorio que se encuentra en [Itinerario 2: Programación Integrativa](#). También, recuerde que una explicación completa del código la encontrará en [Itinerario 2: Programación Integrativa](#).

Hasta aquí el contenido de este tema, debe avanzar y revisar otra forma de trabajo.

5.2.2. Acceso a través de DOM

Document Object Model o DOM (Modelo de Objetos del Documento) es un modelo que representa a cualquier documento XML y HTML como un árbol de nodos en donde cada nodo representa una parte del documento (MDN Web Docs, n.d.).

Como se menciona, la definición anterior se usa en la estructura jerárquica del contenido XML para crear un árbol y luego poder recorrer sus ramas y hojas. Con esta forma de representación y con otros API se puede tener acceso directo a los datos que están dentro del XML, aunque es necesario conocer la ruta o camino que se debe transitar para llegar a un sitio que puede ser un elemento, un atributo, una propiedad o un valor.

Uno de los API que generalmente se combina con DOM es XPath que es un lenguaje de ruta XML, es una sintaxis que sirve para seleccionar elementos y atributos de un documento XML navegando su estructura en forma de árbol (Altova, n.d.).

Es momento de ver este par de API en acción, nuevamente se usará el mismo contenido XML definido para esta semana. Además, debe conocer que el propósito del código es presentar el valor del elemento apellido. El código que cumple con lo solicitado es el siguiente:

```
public static void main(String[] args) throws Exception {  
    String xmlData = """";  
        <?xml version="1.0" encoding="UTF-8"?>  
        ..  
        """;  
    String xPathExpression = "/estudiante/apellido";  
  
    DocumentBuilderFactory factory = DocumentBuilderFactory.  
newinstance();  
    DocumentBuilder builder = factory.newDocumentBuilder();  
    InputSource is = new InputSource(new  
StringReader(xmlData));  
  
    Document document = builder.parse(is);  
  
    XPath xPath = XPathFactory.newInstance().newXPath();  
  
    NodeList nodos =  
        (NodeList) xPath.evaluate(xPathExpression,  
document, XPathConstants.NODESET);  
    for(int i = 0; i < nodos.getLength(); i++) {  
        System.out.printf("%s: %s\n",  
            nodos.item(i).getNodeName(),  
            nodos.item(i).getTextContent());  
    }  
}
```

Antes de empezar con el análisis del código, es necesario comentar que el contenido de la variable `xmlData` ha sido truncado por temas de espacio, pero su contenido es el que se presentó al inicio de esta semana.

Como puede ver, con esta forma de trabajo se necesitan varios objetos de diferentes clases, sería demasiado extenso explicar cada uno de los ellos en este documento, es por ello que una explicación detallada ha sido elaborada en [Itinerario 2: Programación Integrativa](#), concretamente en el apartado semana 13.

Aquí se mencionará el funcionamiento de XPath. Lo primero que debe hacer es definir la ruta que debe seguir, en este caso se especifica en la variable `xPathExpression`. Para este ejemplo se señala que debe ir a la raíz (elemento estudiante) luego dirigirse a los contenidos de esta, concretamente al elemento apellido. Esa expresión se escribe como si se trata de una ruta de archivos.

Con esa expresión se crea una lista de nodos que se recorre y se presenta el nombre del nodo y su valor. Así se podría definir el programa anterior.

Aquí no se dará mayores detalles del funcionamiento del programa anterior, para encontrarlos debe visitar el recurso que se mencionó anteriormente.

Es muy probable que ahora mismo se esté preguntando, ¿cuál de las dos formas de lectura debo usar? La respuesta a esta pregunta no es sencilla y como casi todos los temas en ciencias de la computación, dependerá del problema que debe resolver. Le propongo dos problemas diferentes para que usted seleccione la técnica que le parece correcta.

Problema 1:

Debe obtener el total de las facturas electrónicas que le han llegado a su correo en los últimos 7 días. Este pedido no se volverá a repetir, ya que el dato solo se necesitará una única vez.

Problema 2:

Por cada una de las facturas electrónicas que le han llegado en los últimos 30 días, debe obtener lo siguiente: el nombre, dirección y RUC del establecimiento que emitieron las facturas, además del total a pagar con y sin impuestos. Así mismo se debe obtener un listado de los productos adquiridos clasificados en dos grupos, los que pagan IVA y los que no. Esta

información se requiere al finalizar cada vez y se usará para la proyección de gastos y descuentos en el impuesto a la renta.

Con la descripción de ambos problemas y su conocimiento sobre las técnicas para leer datos escritos en formato XML, trate de seleccionar una técnica para cada problema y sobre todo argumentando su decisión.

Una vez hecho el ejercicio anterior, se muestra y argumenta la selección al problema hecha por su tutor.

Para el problema 1, se cree que la técnica 2 (manipulación de DOM) es la mejor alternativa, los criterios que se usaron son: a) únicamente se pide un valor; b) el procesamiento no es complejo (sumar); y c) la tarea no se volverá a repetir. Con base en estas consideraciones se opta por construir una aplicación que manipule el DOM, aunque puede verse como tediosa de iniciar (por todos los objetos que se debe crear) resulta sencilla al momento de obtener los datos.

Mientras que para el problema 2, se considera que la técnica 1 es la mejor los criterios que se usaron son: a) es una tarea que se repite mensualmente y seguirá así por mucho tiempo; b) necesita procesar muchos datos y c) se necesitan diferentes datos. Es decir, se está dispuesto a construir todas las clases Java que representarán la estructura del XML de una factura que será una tarea compleja, pero es un esfuerzo que se pagará con el tiempo.

Como puede ver con los ejemplos anteriores, es necesario buscar equilibrios entre tiempo, esfuerzo y recompensa para seleccionar la estrategia de lectura que resuelva de mejor manera el problema propuesto.

Hasta aquí el desarrollo de los contenidos de esta unidad en donde revisó cómo leer datos semiestructurados que provienen de un archivo XML. Conoció dos técnicas bastante comunes y útiles para usar un lenguaje de programación en la tarea de lectura.

El próximo tema a estudiar será la integración con el tercer y último tipo de dato, los estructurados.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Implementar el código desarrollado en esta unidad, disponible en [Itinerario 2: Programación Integrativa](#).
- Revisar las explicaciones de la construcción de cada una de las aplicaciones elaboradas, disponible en [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 13.



Semana 14

Unidad 6. Integración de datos estructurados

El último tipo de dato que estudiará son los datos estructurados, en este apartado se recordará algunos detalles de los datos estructurados y pondrá en práctica lo que aprendió en otras asignaturas de la carrera, especialmente lo relacionado con base de datos, porque recuerde que los datos estructurados por excelencia son los que se almacenan en un motor de base de datos relacional.

Comience con una pequeña revisión de los datos estructurados.

Los datos estructurados son datos que responden a un modelo de datos bien definido, ese modelo describe información sobre los datos, así por ejemplo, el modelo de datos establece el tipo de dato (entero, real, cadena, etc.) al que pertenece el dato, su longitud, su valor inicial, si se permite o no valores nulos. Es decir, el modelo de datos es un grupo de metadatos.

Antes de avanzar le planteo la siguiente pregunta, ¿cuál es el modelo que ha estudiado que permite describir lo descrito anteriormente? Piense su respuesta enfocándose a los conocimientos que adquirió en base de datos.

Una respuesta a la pregunta anterior puede ser el modelo físico, ya que este modelo permiten hacer lo descrito y más.

Gracias a la existencia de esos modelos se considera que la información que se almacena en una base de datos son datos estructurados. Entre las características que los datos estructurados tienen se pueden listar los propuestos por GeekforGeeks (2019) listado a continuación:

- Los datos se ajustan al modelo de datos y tienen una estructura fácilmente identifiable.
- Los datos se almacenan en forma de filas y columnas.
- Los datos están bien organizados gracias a la definición del formato y el significado de estos.
- Los datos residen en campos fijos dentro de un registro o archivo.
- Las entidades similares se agrupan para formar relaciones o clases.
- Las entidades del mismo grupo tienen los mismos atributos.
- Fácil acceso y consulta, por lo que los datos pueden ser utilizados fácilmente por otros programas.
- Los datos son fáciles de procesar y analizar.

Como puede ver, estos tipos de datos tienen muchas ventajas, especialmente para el procesamiento y análisis de datos que es una de las tareas que hoy en día son de gran utilidad y una actividad cada vez más necesaria por cualquier empresa actual.

Otra característica que resalta sobre los datos estructurados es que cuentan con un mecanismo de consulta que permite un fácil acceso a los mismos. A nivel de base de datos relacionales, ese mecanismo de consulta es el SQL lenguaje de consultas estructurado que, además de consultar, también permite crear el modelo de datos.

SQL es un lenguaje estándar que funciona que todos los motores de base de datos soportan, aunque algunos de los motores de base de datos tienen sus propias personalizaciones con las que pretenden enriquecerlo para aprovechar las características del motor de base de datos que lo implementan. Como ya mencionó SQL tiene un conjunto de sentencias que permiten definir el modelo de base, a esas sentencias se las denomina DDL (Data Definition Language). Otro grupo son las sentencias que permiten manipular los datos y son denominadas DML (Data Manipulation Language).

Hasta aquí esta presentación del tema, en las próximas secciones se presentará brevemente un ejercicio que le ayudará a recordar sus conocimientos de base datos para luego ponerlos en práctica con un lenguaje de programación, especialmente, con las sentencias DML.

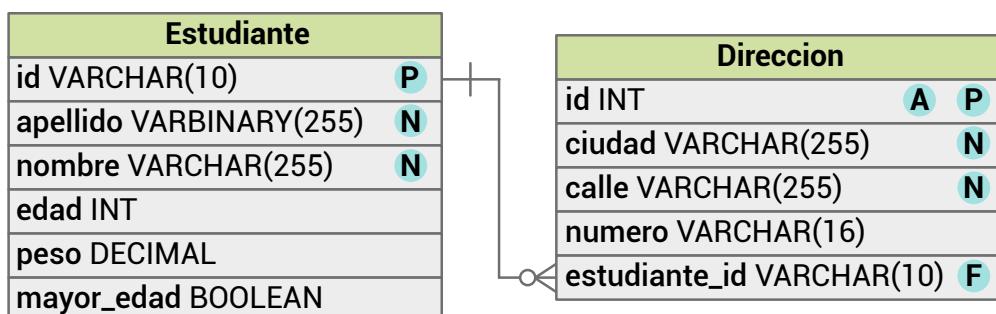
6.1. Modelos de base datos

Antes de construir algún modelo de base de datos se necesita de un problema a resolver. El problema es el mismo que se ha trabajado las últimas semanas, un estudiante tiene varias direcciones. Los datos del estudiante son identificación, apellido, nombre, si es o no mayor de edad, edad y peso, mientras que para la dirección se usará ciudad, calle y número. Además, un estudiante puede tener 1 o más direcciones.

Con ese problema en mente, se construirá un modelo físico para crear las tablas que ayuden a resolver el problema. La Figura 31, muestra el diagrama. Para este problema se utilizará la base de datos MySQL.

Figura 31.

Diagrama físico



Se crean dos tablas, la primera, estudiante y la segunda, dirección. Observe cómo cada columna de la base de datos tiene su nombre y su tipo de dato, algunos de ellos tienen marcas como P para señalar que el valor es una clave primaria o N que señala que la columna no admite valores nulos. Mientras F señala clave foránea o A para indicar que es un campo autoincremental.

A partir de ese modelo se generó las siguientes sentencias DML para la creación de las tablas. Las sentencias son:

```
CREATE TABLE ESTUDIANTE
(
    id VARCHAR(10),
    apellido VARCHAR(255) NOT NULL,
    nombre VARCHAR(255) NOT NULL,
```

```
edad INT,  
peso DECIMAL,  
mayor_edad BOOLEAN,  
PRIMARY KEY (id)  
);  
CREATE TABLE DIRECCION  
(  
id INT AUTO_INCREMENT,  
ciudad VARCHAR(255) NOT NULL,  
calle VARCHAR(255) NOT NULL,  
numero VARCHAR(16),  
estudiante_id VARCHAR(10),  
PRIMARY KEY (id)  
);  
ALTER TABLE DIRECCION ADD FOREIGN KEY estudiante_id_idxfk  
(estudiante_id) REFERENCES ESTUDIANTE (id);
```

Las siguientes sentencias DML permiten agregar datos a cada tabla. Recuerde que, según la relación entre las tablas, primero se debe insertar en la tabla Estudiante para luego en la tabla Dirección. Las sentencias son:

```
INSERT INTO ESTUDIANTE VALUES ("321", "PIEDRA",  
"FRANCISCO", 7, 35.4, TRUE)  
INSERT INTO DIRECCION(`type`, `ciudad`, `calle`, `numero`,  
`estudiante_id`) VALUES ("main", "Loja", "Adam Smith", "451-190",  
"321");  
INSERT INTO DIRECCION(`type`, `ciudad`, `calle`, `numero`,  
`estudiante_id`) VALUES ("alta", "Guayaquil", "Juan Tan  
Camarengo", "87W-56S", "321");
```

Observe cómo para la tabla Estudiante no se señalan las columnas y los valores se insertan según el orden en el que se crearon las columnas. Esto no sucede con la tabla Dirección, en donde es necesario especificar las columnas, ya que la columna ID no se debe insertar considerando que es un campo autoincremental.

Antes de concluir con este tema, se mostrará un ejemplo que muestra cómo hacer una consulta sobre esa pequeña base de datos que pretende

encontrar el apellido de un estudiante, la calle y ciudad de la dirección marcada como *main*. La sentencia es:

```
SELECT e.`apellido`, d.`ciudad`, d.`calle`  
FROM DIRECCION d, ESTUDIANTE e  
WHERE d.`type` = "main"  
AND d.`estudiante_id` = e.`id`
```

No se entrará en más detalles, ya usted conoce del tema. Lo siguiente que debe hacer es revisar las estrategias para interactuar con base de datos a través de un programa.

6.2. Interactuando con datos estructurados desde un programa

En este apartado usará el lenguaje de programación Java para interactuar con datos estructurados que se encuentran almacenados en una base de datos. El lenguaje de programación a usar será Java, antes de pasar al código, es necesario mencionar que estudiará tres formas de interactuar con las bases de datos relacionales.

Dentro del ecosistema de la mayoría de los lenguajes de programación existen diferentes formas para interactuar con una base de datos que generalmente se encuentran en diferentes niveles de abstracción. Uno de esos niveles de programación será el menor de nivel de abstracción que generalmente hace de base para que sobre él se creen otros niveles.

Dentro de Java, el nivel más bajo de abstracción se denomina JDBC. En el siguiente apartado se presenta este tema desde un enfoque práctico.

6.2.1. JDBC

Es un acrónimo de Java Database Connectivity y, como se mencionó anteriormente, es la forma de trabajo de más bajo nivel sobre la cual se han creado otros niveles que conocerá más adelante.

A pesar de ser una de las formas más básicas es posiblemente la más utilizada, principalmente porque es la primera que se emplea cuando se aprende a utilizar bases de datos relacionales desde el lenguaje Java.

Ahora es momento de pasar a la práctica, pero si quiere conocer más detalles sobre JDBC le recomiendo leer (Behler, 2020), además, le recuerdo que más adelante encontrará las instrucciones para acceder a una descripción detallada de la práctica que encontrará a continuación.

Así como se indicó en el inicio de esta tema, se usará MySQL con las tablas y datos que se mostraron en el apartado anterior Modelos de base de datos. El propósito del siguiente código es realizar una consulta SQL y obtener sus datos. El código que se usará es el siguiente:

```
public static void main(String[] args) {  
    String dbURL = "jdbc:mysql://localhost/P_INTEGRATIVA";  
    String selectQry = """";  
        SELECT e.`apellido`, d.`ciudad`, d.`calle`  
        FROM DIRECCION d, ESTUDIANTE e  
        WHERE d.`type` = "main"  
        AND d.`estudiante_id` = e.`id`  
    """;  
    String frmtOutput = """";  
        Apellido: %s  
        Ciudad: %s  
        Calle: %s  
    """;  
  
    try (Connection conn = DriverManager.getConnection(dbURL,  
    "root", "")) {PreparedStatement selectStatement = conn.  
    prepareStatement(selectQry);  
    ResultSet rs = selectStatement.executeQuery();  
  
    while (rs.next()) {  
        System.out.printf(frmtOutput,  
            rs.getString("apellido"),  
            rs.getString("ciudad"),  
            rs.getString("calle"));  
    }  
}
```

```
        } catch (SQLException sqlException) {
            sqlException.printStackTrace();
        }
    }
```

El código no es extenso ni complejo, sin embargo, es necesario enfatizar en algunas líneas de código que son importantes.

Dentro de JDBC para conectarse a una base de datos se necesita de una “url” o cadena de conexión que cambia según el motor de base de datos que emplee. En esta práctica se encuentra definida en la variable *dbURL*, la última parte de la ruta señala el nombre de la base de datos creada anteriormente *P_INTEGRATIVA*.

La siguiente línea de código importante es obtener la conexión que se hace con el método *getConnection*, en ese método se usa la cadena de conexión y el nombre de usuario y contraseña que pertenecen a una cuenta que tiene acceso a la base de datos. En el código el usuario es root y no el usuario no tiene contraseña.

Una vez ejecutada esa línea de código tendrá una conexión abierta con la base de datos y podrá enviar y recibir mensajes. El paso siguiente es crear un objeto del tipo *PreparedStatement* que permite enviar sentencias SQL a la base de datos, en esta práctica la sentencia a ejecutar está almacenada en la variable *selectQry*.

Para obtener los datos se debe enviar a ejecutar la sentencia a través del método *executeQuery*, el mismo que devuelve una *ResultSet* que una especie de tabla (filas y columnas) en donde vienen los resultados de ejecutar la sentencia SQL.

Finalmente se recorre el *ResultSet* fila a fila con el método *next()* y por cada fila se accede a cada columna a través del método *getString* y su nombre. Aquí es importante mencionar que existe un gran variedad de métodos *get**, generalmente una por tipo de dato, ya que según el tipo de dato al que pertenece la columna se debe usar el método le corresponde.

Esta forma de trabajo se aleja un poco de una de las características principales de Java, ser orientado a objetos, especialmente en los resultados, ya que el resultado de una consulta SQL no es un grupo de objetos, sino de valores “sueltos” a los que se debe llegar de forma

individual. Obviamente que con cada uno de esos valores se puede construir un objeto, pero es una tarea que puede consumir tiempo y que se aleja de los temas importante como la programación de una lógica de negocio.

Ese distanciamiento ha provocado que se construyan varias librerías que buscan automatizar el proceso de extracción de los datos. Una de esas librerías las estudiará más adelante, pero antes de avanzar un recordatorio.

Una guía completa del código hecha con herramientas multimedia la encontrará en [Itinerario 2: Programación Integrativa](#). Además, el código fuente lo encontrará en [Itinerario 2: Programación Integrativa](#).

Una vez concluido este tema, revise el siguiente apartado en donde se presenta otra forma de trabajo que usa JDBC como base para crear otro modelo de abstracción.

6.2.2. JDBI

JDBI es una pequeña librería que se construye sobre JDBC y que pretende que los programas Java puedan trabajar con una base de datos de una manera más sencilla y cubrir algunas deficiencias de JDBC. En Dehler (2020) encontrará una explicación adicional sobre esta librería.

La intención de este documento no es convertirlo en un experto en cada una de las librerías que se ha presentado, sino brindarle una visión de varias alternativas para que usted las conozca y si es de su interés profundice en una de ellas. Por esta razón la descripción de la librería no es amplia.

Con la aclaración hecha, es momento de pasar a la parte práctica. En esta práctica seguirá usando el problema propuesto en el apartado anterior. El código es el siguiente:

```
public static void main(String[] args) {  
    String dbURL = "jdbc:mysql://localhost/P_INTEGRATIVA";  
    String selectQry = """";  
        SELECT e.`apellido`, d.`ciudad`, d.`calle`  
        FROM DIRECCION d, ESTUDIANTE e  
        WHERE d.`type` = "main"  
        AND d.`estudiante_id` = e.`id`  
        """;
```

```
String frmtOutput = """
    Apellido: %s
    Ciudad: %s
    Calle: %s
""";
Jdbi jdbi = Jdbi.create(dbURL, "root", "");
List<StudentAddress> data = jdbi.withHandle(handle ->
    handle.createQuery(selectQry)
        .mapToBean(StudentAddress.class)
        .list());
for (StudentAddress sa : data) {
    System.out.printf(frmtOutput,
        sa.getApellido(),
        sa.getCiudad(),
        sa.getCalle());
}
}
```

Debido a que se usará la misma base de datos y el mismo problema, algunas variables son las mismas y mantienen el mismo propósito, en este caso están las variables *dbURL*, *selectQry* y *frmtOutput*.

Dentro de JDBI, la primera sentencia es crear un objeto de la clase *Jdbi* usando la cadena de conexión y las credenciales de acceso (usuario y contraseña). Con ese objeto se usa una expresión lambda (*jdbi.withHandle*) para ejecutar sentencias SQL, mapear cada fila del resultado en un objeto de la clase *StudentAddress* y convertirlos en una lista que se almacena en la variable *data*.

El siguiente paso es recorrer la lista con los datos y generar una salida en pantalla que muestre los datos recuperados.

Recuerde que una descripción completa de la construcción del programa está disponible en [Itinerario 2: Programación Integrativa](#).

Antes de continuar le propongo que revise cada una de las soluciones y trate de ver varias similitudes y diferencias. Luego continúe con la lectura de los siguientes párrafos.

Dentro de las similitudes de ambas soluciones, está que ambas usan algunas variables como la cadena de conexión, la sentencia de consulta y el formato de salida. Esto se debe a que ambas soluciones trabajan con el mismo problema y los mismos datos.

Si hablamos de las diferencias, que van más allá de la sintaxis, se encuentra la capacidad de JDBI para transformar automáticamente los resultados en objetos de una clase y agruparlos en una lista. Esa misma tarea en JDBC es posible, pero requiere varias líneas de código.

Lo anterior es posible sólo si los atributos de la clase StudentAddress coinciden exactamente con los nombres de las columnas seleccionadas en el consulta SQL, además la clase debe tener un constructor por defecto y los métodos get y set por cada atributo, es decir, ser un tradicional POJO.

Possiblemente se esté preguntando sobre el código de la clase StudentAddress, es por ello que su código se muestra a continuación:

```
public class StudentAddress {  
    private String apellido;  
    private String ciudad;  
    private String calle;  
    public StudentAddress() {  
    }  
    //Métodos get y set por cada atributo  
}
```

De esta forma se concluye con esta alternativa que permite extraer datos estructurados desde una base de datos relacional.

La siguiente y última alternativa se encuentra en lo más alto del nivel de abstracción que tiene como base JDBC y se denomina JPA, algunos detalles se muestran en el siguiente apartado.

6.2.3. Java Persistence API o JPA

Esta tercera y última alternativa es mucho más abstracta que la anterior, tanto así que, si es bien empleada, se puede ganar independencia del motor de base de datos, eso quiere decir que podrá cambiar su base de datos sin la necesidad de cambiar el código relacionado a la base de datos.

Ese nivel de independencia tiene un costo que en este caso se traduce en incremento del número de clases a usar, ya que se basa en el concepto de Mapeo Objeto-Relación también conocido como ORM en donde básicamente una tabla de una base de datos relacional se mapea a un clase y cada fila de la clase se mapea en un objeto.

Sin lugar a duda, este es el mecanismo más utilizado cuando se trata de trabajar con aplicaciones transaccionales, ya que permite la independencia del sistema gestor de base datos, así una aplicación puede cambiar su base de datos por otra más potente o gratuita sin entrar en cambios de la aplicación.

Possiblemente no sea demasiado útil en procesos de integración de datos, ya que las tareas previas, en especial la creación del mapeo, pueden consumir mucho tiempo que se podría usar en otras actividades más relacionadas a la integración que al correcto mapeo tabla clase.

Pero es una alternativa que debe conocer especialmente porque usa para la construcción de servicios SOAP, REST y microservicios que, como lo estudió anteriormente, es la forma más común de integración de aplicaciones de hoy en día.

Luego de la breve introducción, es momento de revisar el código que permite interactuar con una base de datos usando JPA. Veamos.

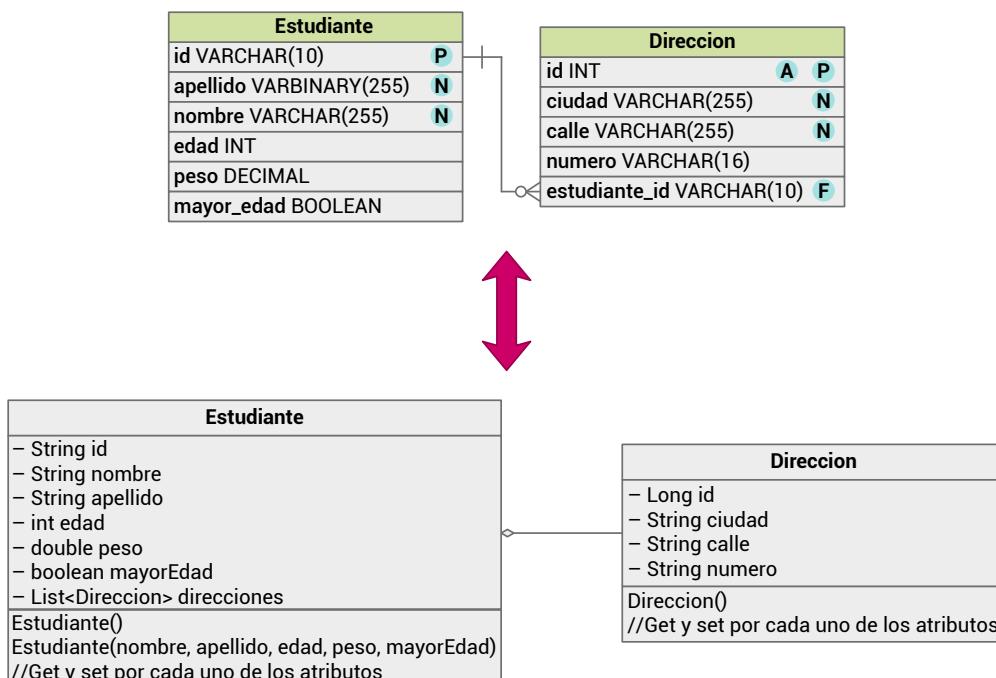
Para su mejor comprensión se ha dividido en tres partes, la primera relacionada al mapeo, la segunda a la interacción con la base de datos y finalmente la consulta de datos.

Mapeo

Dentro de JPA el mapeo es la forma de expresar cómo dos modelos pueden integrarse. Dentro de JPA se utilizan anotaciones para indicar cómo se hará la correspondencia entre los elementos de la base de datos con los elementos de las clases Java.

Figura 32.

Representación del mapeo OR



La Figura 32 muestra una representación del mapeo entre tablas y clases Java, es necesario mencionar que el mapeo se puede hacer en ambos sentidos, es decir, desde las tablas de una base de datos generar las clases y desde las clases Java generar las tablas.

En su caso, ya cuenta con una base de datos y sus tablas, por lo que el mapeo se hará desde la base de datos a Java. Dentro de los entornos de desarrollo integrado es muy común encontrar herramientas que ayudan en estos procesos.

Es momento de pasar al código, comience revisando el código de la clase *Dirección*.

```
@Entity  
public class Direccion {  
    @Id  
    private int id;  
    private String type;  
    private String ciudad;
```

```
private String calle;  
private String numero;  
@ManyToOne  
@JoinColumn(name = "estudiante_id",  
referencedColumnName = "id")  
private Estudiante estudiante;  
//Get y Set por cada atributo
```

La clase Dirección hace uso de anotaciones para configurar el mapeo hacia la tabla con el mismo nombre. JPA utiliza un principio que se denomina configuración por excepción, es decir, muchas configuraciones se hacen sin la necesidad de alguna acción, a menos que explícitamente se señale algo.

Luego de esta primera vista de la clase, es necesario detallar las anotaciones que la clase usa y que se resumen en el siguiente listado:

- *Entity*, señala que la clase es una entidad, es decir, representa a una tabla en una base de datos.
- *Id*, se utiliza para señalar que el atributo representa a la clave primaria de la tabla.
- *ManyToOne*, se utiliza para indicar que existe una relación de muchos a uno, muchas direcciones pertenecen a un estudiante.
- *JoinColumn*, señala algunas características de la relación con la clase *Estudiante*. Las propiedades *name* y *referencedColumnName* especifican las columnas que participan en la relación. La propiedad *name* señala el nombre de la columna en la tabla *Dirección*, mientras que, *referencedColumnName* señala el nombre de la columna de la clase *Estudiante*.

Probablemente se esté preguntado cómo se hace el mapeo del resto de atributos al resto de las columnas de la tabla. La respuesta es la configuración por excepción, al no existir ninguna anotación JPA determina que el nombre del atributo es exactamente igual al nombre de la columna, lo mismo sucede con los tipos de datos. De esa forma se consigue escribir menos código para conseguir su funcionamiento.

Un ejemplo más de la configuración por defecto se da con el nombre de la tabla, en la clase no se especifica el nombre de la tabla a la que corresponde

la clase. Esto no es necesario, siempre y cuando el nombre de la clase sea el mismo que el de la tabla.

Una vez revisada la primera clase, debe avanzar y revisar la clase Estudiante.

```
@Entity  
public class Estudiante {  
    @Id  
    private int id;  
    private String code;  
    private String apellido;  
    private String nombre;  
    private Integer edad;  
    private Integer peso;  
    @Column(name = "mayor_edad")  
    private boolean mayorEdad;  
    @OneToMany(mappedBy = "estudiante")  
    private Collection<Direccion> direcciones;  
  
    //Get y Set por cada atributo
```

Comparada con la clase anterior, no es muy diferente y la diferencia son dos anotaciones que se explican a continuación:

- *Column*, se usa cuando existe una diferencia en los nombres del atributo y la columna. Este es un caso en el que la configuración por excepción no funciona, es el programador quien debe señalar el nombre de la columna contra la que se mapear el atributo.
- *OneToMany*, se utiliza para señalar la relación uno a muchos que existe entre las tablas. El valor de la propiedad *mappedBy* señala el nombre del atributo de la clase Dirección que contiene más detalles de la relación.

Así mismo, el resto de atributos se mapean por excepción con las demás columnas de la tabla Estudiante.

Ambas clases muestran únicamente una porción del código, pero el resto del código son los métodos get y set de los atributos. Para ver el código fuente debe ir a [Itinerario 2: Programación Integrativa](#).

Interacción con la base

Dentro del ecosistema JPA existen varios cambios con respecto a las técnicas anteriores, ya puedo ver algunos, pero aún existen más, entre esos cambios está la manera de interactuar con los datos.

El código que generalmente se usa tiene la siguiente forma:

```
EntityManagerFactory emFactory =  
    Persistence.createEntityManagerFactory("default");  
EntityManager em = emFactory.createEntityManager();
```

El *EntityManager* es una clase Java que tiene dos funcionalidades, la primera es abrir y mantener una conexión con la base de datos y la segunda es mantener en memoria a las entidades (clases marcadas con la anotación *Entity*) y sincronizarlas con la base de datos.

En *EntityManager* se obtiene a través de la clase *EntityManagerFactory* que se configura a través de la unidad de persistencia que es un archivo XML con los datos necesarios para interactuar con una base de datos. El contenido del archivo es el siguiente:

```
<persistence-unit name="default">  
    <class>ec.edu.utpl.pintegrativa.jpa.entity.Direccion</class>  
    <class>ec.edu.utpl.pintegrativa.jpa.entity.Estudiante</class>  
    <properties>  
        <property name="hibernate.connection.url"  
value="jdbc:mysql://localhost/P_INTEGRATIVA"/>  
        <property name="hibernate.connection.driver_class"  
value="com.mysql.cj.jdbc.Driver"/>  
        <property name="hibernate.connection.username"  
value="root"/>  
        <property name="hibernate.connection.password"  
value="" />  
    </properties>  
</persistence-unit>
```

La unidad de persistencia tiene el nombre *default* y contiene las clases marcadas como entidades, además, en las propiedades se configura la cadena de conexión, nombre de usuario y contraseña.

Esta forma de trabajo es clave para lograr la independencia de la base de datos, ya que si necesita cambiar de base de datos, debe cambiar la unidad de persistencia, no el resto del código.

Consulta de datos

Como ya conoce, el EntityManager cumple un rol fundamental dentro de JPA, una de sus funciones es generar la consultas SQL para interactuar con la base datos, esta es otra característica para lograr la independencia de la base de datos, no dejar al programador que escriba consultas directamente a la base de datos, sino tener un elemento que lo genera en su lugar. Así se evita dependencias con la base de datos.

Es muy probable que se encuentre confundido y preguntándose, ¿SQL es estándar porque usarlo me hace dependiente a un motor de base datos? La respuesta es que cada motor de base de datos tiene su propia implementación de SQL que es diferente entre bases de datos. Es por eso que para lograr independencia se debe evitar escribir SQL.

El EntityManager conoce con qué base de datos se conecta y puede escribir el SQL adecuado a esa base de datos. Además, las consultas que generan usan recomendaciones y buenas prácticas que ayudan a mejorar su desempeño. Existen otras ventajas de usar EntityManager como por ejemplo la seguridad, ya que previene problemas comunes al trabajar con una base de datos, como la inyección de [SQL](#).

Es momento del código que muestra cómo consultar los datos de la tabla Estudiante.

```
TypedQuery<Estudiante> qry =  
    em.createQuery("SELECT e FROM Estudiante e", Estudiante.  
    class);  
List<Estudiante> estudiantes = qry.getResultList();  
for (Estudiante est : estudiantes) {  
    System.out.println(est.getApellido());  
    for(Direccion d : est.getDirecciones()) {
```

```
        System.out.printf("Ciudad: %s\n", d.getCiudad());
        System.out.printf("Calle: %s\n", d.getCalle());
        System.out.printf("Número: %s\n", d.getNumero());
    }
}
```

El código anterior utiliza el EntityManager para crear una consulta JPQL que obtiene todas las filas de la tabla Estudiante y las mapea como objetos de esa clase. Luego se recorre la lista y por cada estudiante se presenta su apellido y la lista de direcciones que posee.

Antes de continuar, es necesario aclarar que JPQL es un lenguaje de consultas para JPA similar al SQL, pero que en realidad no hace referencia a tablas, sino a clases, así se mantiene la independencia de la base de datos.

Una explicación completa del código y recursos multimedia que le ayudarán a comprender su funcionamiento está en [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 14.

Ha revisado tres técnicas que permite extraer datos estructurados. De las tres técnicas mencionadas anteriormente, la que más se utiliza dentro de la integración de datos es la JDBC, mientras que para integración de aplicaciones a través de servicios es JPA. JDBI también se puede usar en tareas de integración de datos, posiblemente en proyectos pequeños y concretos en donde no se necesite software especializado para la integración.

De esta forma termina el desarrollo de los contenidos de esta penúltima semana de actividades en donde conoció cómo obtener datos estructurados desde una base de datos relacional utilizando tres técnicas diferentes. Para la próxima semana podrá lo aprendido en práctica e integrará datos estructurados y semiestructurados de diversas fuentes.



Actividades de aprendizaje recomendadas

Se recomienda realizar las siguientes actividades:

- Implementar el código desarrollado en esta unidad, disponible en [Itinerario 2: Programación Integrativa](#).

- Revisar las explicaciones de la construcción de cada una de las aplicaciones elaboradas, disponible en [Itinerario 2: Programación Integrativa](#), bajo el apartado semana 14.

A continuación, se presentan algunas preguntas que debe responder. Esta autoevaluación le permitirá a usted reafirmar los conceptos estudiados.

Le invito a reforzar sus conocimientos participando en la siguiente autoevaluación:



Autoevaluación 5

En las siguientes preguntas seleccione verdadero o falso según corresponda:

1. () ¿Los datos correctamente administrados son fuente para la toma de decisiones?
2. () ¿Los datos externos provienen de las redes sociales en línea?
3. () Actualmente, los datos que las empresas necesitan únicamente se encuentran en sus bases de datos internas.
4. () ¿La captación de datos puede verse solo como una estrategia cuestionable?
5. () ¿Open Linked data es una forma de representar recursos y enlazarlos?
6. () Los datos son independientes y heterogéneos es por eso que se necesita de procesos de integración.
7. Juntar físicamente los datos define a:
 - a. Consolidación.
 - b. Propagación.
 - c. Virtualización.
 - d. Federación.
 - e. Warehousing.
8. Copiar datos en varios lugares para su posterior sincronización describe a:
 - a. Consolidación.
 - b. Propagación.
 - c. Virtualización.
 - d. Federación.
 - e. Warehousing.

9. A través de una interfaz única proporcionar hace a los datos:
- a. Consolidación.
 - b. Propagación.
 - c. Virtualización.
 - d. Federación.
 - e. Warehousing.
10. Usa una base de datos virtual para almacenar los datos:
- a. Consolidación.
 - b. Propagación.
 - c. Virtualización.
 - d. Federación.
 - e. Warehousing.
11. Comprende limpieza, reformatear y almacenar datos en un único repositorio:
- a. Consolidación.
 - b. Propagación.
 - c. Virtualización.
 - d. Federación.
 - e. Warehousing.

[Ir al solucionario](#)



Unidad 7. Integración de datos en la práctica

A lo largo de la asignatura ha creado programas usando el lenguaje de programación Java para aprender sobre diferentes conceptos relacionados a la integración de aplicaciones y de datos, pero aún no ha tenido la posibilidad de experimentar un proceso de integración. Esa ausencia se cubre esta semana.

Para esta semana se presentará un problema de integración de datos en donde usará una herramienta ETL para realizar ese proceso.

Como usted bien conoce, las herramientas están en constante cambio y evolucionan con el tiempo, algunas veces esas herramientas desaparecen o son reemplazadas por otras que son mejores. Para evitar que esta sección, que depende de una herramienta, se quede estancada y deje de ser útil porque usa alguna herramienta antigua, se ha diseñado para que la mayoría del contenido relacionada con el problema y su solución se encuentren en un entorno dinámico de rápida actualización que permita evolucionar y ayudar de forma efectiva en su formación.

Los contenidos teóricos de base en general no cambian rápidamente, se mantienen, así que esos contenidos son los que aquí encontrará.

Comenzará a estudiar esta sección con el planteamiento del problema, luego pasará a la solución paso a paso de este.

7.1. Planteamiento del problema

En un proceso de integración de datos, es muy común que se encuentre con diferentes tipos de datos que debe integrar. Esos datos pueden ser estructurados, semiestructurados o no estructurados. Además, esos datos pueden venir de diferentes fuentes, así, por ejemplo, archivos JSON, XML, CSV o de diferentes bases de datos, como por ejemplo Oracle, MySQL, Postgres y posiblemente alguna base de datos no relacional como Neo4J o MongoDB.

El problema que se planteará estará apegado a la realidad, por lo que deberá integrar datos de diferente tipo, exceptuando los no estructurados, por su complejidad quedan fuera de este ejercicio. Esos datos tendrán diferentes fuentes, dejando de lado las bases de datos no relacionales, ya que demandan de una infraestructura que aún no es muy común en nuestro entorno educativo.

En resumen, el problema a resolver se basa en datos estructurados y semiestructurados que provienen de base de datos, servicios web y archivos.

La descripción del problema se encuentra en [Itinerario 2: Programación Integrativa](#), bajo el título semana 15.

Ahora que conoce el problema, es necesario comenzar a estudiar cómo implementar la solución. Recuerde que el problema gira alrededor de diversas fuentes que deben estructurarse como una única fuente para realizar análisis posteriores.

7.2. La solución

Como en la mayoría de los proyectos informáticos una solución tiene dos partes, la primera es la solución a nivel de diseño y la segunda es la solución a nivel de implementación, ya sea desarrollando un software o utilizando alguno.

Comience a estudiar la solución a nivel de diseño.

7.2.1. Asignación e intercambio de modelos de datos-Mapping

Cuando se enfrenta a un problema de integración de datos que son de diferente tipo y provienen de diferentes fuentes, una de las primeras acciones es estudiar las fuentes y llegar a establecer una forma o medio de integración, es decir, las diferentes fuentes deben tener información común para poder ser integradas, caso contrario la integración no es factible.

Esa información común debe tener el mismo significado, aunque puede estar identificada de diferente manera, lo que implica asignar e intercambiar modelos de datos que es un proceso que se denomina mapping. Para conocer sobre este proceso haga una lectura de los siguientes párrafos.

Fatima (2018) define al mapeo de datos o data mapping como el proceso de extraer datos de una o varias fuentes de origen y hacerlos coincidir con los datos de destino, es una forma de homogenizar los datos para hacerlos más accesibles a quienes toman las decisiones.

Un ejemplo sencillo de data mapping, suponga que tiene información sobre la provincia en donde reside una persona, es información es digitada por el usuario, usted podrá encontrar valores como LOJA, Loja, loja, se toma la decisión de que los datos serán almacenados como LOJA. Este proceso, aunque pequeño, es necesario para la integración de datos y no solo es necesaria la acción, sino también la documentación del proceso, pero eso lo verá más adelante.

Ahora avance en el desarrollo de los contenidos y estudie los pasos para el mapeo de datos.

Para Yaddow (n.d), el mapeo de datos puede tener los siguientes pasos:

Pasos para el mapeo de datos

Este proceso debe ser repetible y fiable, es decir, cada vez que se ejecute todos los paso anteriores, sobre un mismo conjunto de datos, los resultados deben ser los mismos, si existen alguna diferencia es muy probable que está cometiendo alguna omisión.

Con el conocimiento de los pasos del proceso, es importante que conozca los tipos de mapeo de datos que existen, para ello haga una lectura del tema Técnicas Comunes de Mapeo de Datos que propone (Fatima, 2018).

De las técnicas comunes, es necesario mencionar que la técnica automática de mapeo en realidad es un tipo de técnica manual, pero asistida por un software que ayuda a la construcción del mapeo. También se debe comentar que dentro de la técnica mapeo de esquemas se realizan muchos trabajos de investigación buscando mejorarla y así ayudar al mapeo, sobre todo por la gran cantidad de datos a los que se tiene acceso hoy en día.

Antes de avanzar, es importante mencionar que para que un proceso sea repetible, es necesario que este se encuentre correctamente documentado. Si hace una búsqueda en la Web usted encontrará varias plantillas que muestran cómo documentar correctamente este proceso. Lo importante no es seleccionar una plantilla que es promocionada como la mejor, lo importante es contar con una y, sobre todo, con el compromiso de

mantenerla y actualizarla. Ya luego con el uso podrá quitarle o agregarle información.

Una de las mejores formas de comprender cualquier proceso es a través de un ejemplo, así que a continuación se plantea un problema sencillo que mostrará cómo documentar y ejecutar un proceso de mapeo. El ejercicio que verá a continuación se basa en el descrito en [What is ETL Mapping Document ?A Real Time Example](#).

Suponga que debe integrar información del personal administrativo de una institución, los datos tienen la siguiente forma:

Tabla 6.

Fuente de datos de origen

Nro.	Emp_Primer_Nombre	Emp_Apellido	Emp_Nro	Emp_Departamento
1	Juan	Pérez	2001	Ventas
2	Flor	Gómez	2002	Ventas
3	Andrés	Granda	2003	Contabilidad
4	Sebastián	López	2004	Contabilidad
5	Vanessa	Armijos	2005	Recursos humanos

Con esa información se elabora un documento de mapeo en donde se describen la información que los autores del recurso la consideran importante. El documento está compuesto de los siguientes datos:

- Indicador de mapeo (Valores A: AGREGAR, D: Eliminar, C: Cambiar).
- Descripción del cambio (se utiliza para indicar cambios en el mapeo).
- Indicador de clave (indica si el campo es clave principal o no).
- Tabla de origen / Nombre de archivo.
- Nombre del campo de origen.
- Tipo de dato del campo de origen.
- Longitud del campo de origen.
- Descripción del campo de origen (la descripción se utilizará como metadatos para el usuario final).
- Regla.
- Nombre de la tabla de destino.
- Nombre del campo de destino.
- Tipo de dato del destino.
- Longitud del campo en destino.
- Comentarios adicionales.

Para comprender la forma en la que se construye el documento es importante mencionar que la tabla en donde se ubican los datos de origen se denomina Empleado_Adm y que se necesita traer los datos a una nueva tabla que se denomina EMPLEADO.

Los datos primer nombre, apellido, número no se modifican, el único que debe cambiar es departamento, así si el departamento es Ventas ubicar el valor V, si es igual a contabilidad ubicar C y si es recursos humanos reemplazar ese valor por R.

A continuación, un ejemplo del documento de mapeo [Tabla de documentación del mapeo](#)

La tabla muestra un ejemplo de la documentación que el proceso de mapeo debe tener. Luego de revisar esa tabla, ¿cree usted que le hace falta algo?

A criterio personal, le hace falta una correcta administración de los cambios, es decir, si por algún motivo alguna de las filas cambia, es decir, un mapeo cambia, los datos anteriores se pierden y no se podrá conocer sobre el cambio original. Pero esto es cuestión de criterios.

Hasta aquí esta sección, recuerde que la descripción a la solución del problema está disponible en [Itinerario 2: Programación Integrativa](#) bajo el título semana 15.

Con estos conocimientos sobre mapeo en mente, avance al siguiente apartado para conocer el procedimiento a seguir.

7.2.2. ETL

Los procesos de integración de cambio se pueden hacer con escritura de código o con el uso de una herramienta. La escritura del código está bien cuando el proceso de integración es pequeño y no se ejecutará más de una vez, caso contrario se debe seleccionar una herramienta basada en ETL que tenga las características que se necesitan según el proceso de integración.

La idea actual dentro del ETL es contar con una herramienta visual que permita definir y ejecutar un flujo de trabajo. La mayoría de las herramientas cuentan con diferentes procesos, representados como bloques, dedicados a una tarea exclusiva que reciben unas entradas y que generan salidas. Las entradas pueden provenir de otro proceso o de una fuente de datos y las

salidas se pueden convertir en nuevas entradas para otros procesos, así se construye un Pipeline o flujo de trabajo.

En el mercado, actualmente existen muchas herramientas ETL que es difícil seleccionar una, sobre todo debido a la dinámica de constante cambio de estas herramientas. Es por eso que en este apartado se indicará, en términos generales, cuáles son los pasos a ejecutar y dónde encontrar el detalle que estará en un documento externo que permita, entre otras cosas, cambiar de versión de una herramienta y si es necesario cambiar herramienta, así se busca ofrecer una mejor experiencia de aprendizaje.

Los paso a seguir y dónde encontrar los recursos son:

1. Instalación de la herramienta. Pasos disponibles en [Itinerario 2: Programación Integrativa](#), semana 15, bajo el título Instalación.
2. Descripción de la herramienta. Antes de empezar a trabajar debe conocer la herramienta que usará. Pasos disponibles en [Itinerario 2: Programación Integrativa](#), semana 15, bajo el título Introducción a la herramienta ETL.
3. Configuración. En este paso se indican cuáles son y en dónde están las fuentes de datos a utilizar. Pasos disponibles en [Itinerario 2: Programación Integrativa](#), semana 15, bajo el título Configuración.
4. Integración de datos semiestructurados. Aquí se construirán los flujos que se encargarán de los datos semiestructurados que en su mayoría serán archivos. Pasos disponibles en [Itinerario 2: Programación Integrativa](#), semana 15, bajo el título Integrando datos semiestructurados.
5. Integración de datos estructurados. Aquí se construirán los flujos que se encargarán de los datos estructurados que en su mayoría serán tablas de diferentes bases de datos. Pasos disponibles en [Itinerario 2: Programación Integrativa](#), semana 15, bajo el título Integrando datos estructurados.
6. Ejecución. Será el último paso que ejecutar y tiene como finalidad generar resultados de la ejecución del ciclo. Pasos disponibles en [Itinerario 2: Programación Integrativa](#), semana 15, bajo el título Ejecución.



Actividad de aprendizaje recomendada

- Implementar el ejercicio de integración de datos de diferentes fuentes que se describió anteriormente.



Semana 16



Actividades finales del bimestre

Han finalizado las ocho semanas de desarrollo de contenidos y estudio planificadas para este bimestre en donde revisó los temas relacionados con la integración de datos. Esta semana es una semana de preparación para la segunda evaluación presencial. Con eso en mente se recomienda realizar las siguientes actividades:

- Revisar las respuestas a las autoevaluaciones propuestas durante todo el bimestre. Recuerde que al final de este documento encontrará las respuestas a las mismas junto con una breve sustentación de estas.
- Revisar la grabación del chat académico y sus recursos asociados que estarán disponibles en el EVA. Si no pudo participar del evento en vivo, recuerde que debe desarrollar la actividad suplementaria.
- Leer los comentarios que recibió su participación en el foro académico, por parte de su docente tutor y de sus compañeros. Esta retroalimentación es importante, ya que le ayudará a conocer en dónde debe reforzar.
- Realizar ejercicios y construir programas para obtener datos no estructurados, semiestructurados y estructurados, no olvide que la asignatura es práctica.
- Ingrese al entorno virtual de aprendizaje para leer los anuncios publicados, en ellos se incluirá instrucciones, actividades y ejercicios que deberá desarrollar en esta semana de preparación. Además,

puede comunicarse con su tutor para resolver dudas e interactuar con sus compañeros para formar grupos de estudio.

Hasta aquí el desarrollo de los contenidos de la asignatura. Espero que haya disfrutado del contenido tanto como el autor disfrutó escribirlo. Si tiene alguna observación o sugerencia, por favor, póngase en contacto con su tutor, él será el encargado de hacer llegar al autor de esta guía sus valiosos comentarios.



4. Solucionario

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
1	F	Los sistemas empresariales buscan la integración de varios sistemas, mientras que los transaccionales se dedican a una tarea concreta.
2	V	Siempre que existan dos o más sistemas se pueden presentar problemas de integración.
3	V	La integración busca la optimización y el compartir recursos que ayudan a optimizar los sistemas transaccionales.
4	V	Un sistema empresarial es un tipo de sistema informático.
5	F	Los sistemas de información empresarial pueden recibir diferentes entradas y una de ellas puede ser la información de los usuarios.
6	F	Los sistemas de transacciones son sistemas que apoyan a la parte operativa y no a la administración.
7	V	Un ERP ayuda a la comunicación entre aplicaciones.
8	V	Un CRM se encarga de la administración de las relaciones con los clientes.
9	F	La integración es un proceso integral que no solo incluye a los sistemas informáticos y su entorno cercano.
10	V	Son los tres tipos de integración.
11	V	El intercambio de mensajes es la forma más usada para comunicar sistemas distribuidos.
12	F	No se necesita conocer esa información, se puede configurar posteriormente.
13	F	El cliente generalmente necesita conocer la información del servidor.
14	V	La independencia es una característica de los sistemas distribuidos.
15	V	Los sockets son de bajo nivel, por lo tanto, se comunican con elementos de bajo nivel.
16	V	Una característica del protocolo TCP/IP es garantizar el arribo de los mensajes.
17	V	Comparado frente al protocolo TCP/IP, UDP es mucho más rápido.
18	V	No existen grandes diferencias entre ambos códigos.
19	V	Input se asocia con la entrada.

Autoevaluación 1

Pregunta	Respuesta	Retroalimentación
20	F	DataOutputStream es una abstracción de mayor nivel que trabaja con tipos de datos y no con bytes.

[Ir a la
autoevaluación](#)

Autoevaluación 2		
Pregunta	Respuesta	Retroalimentación
1	V	A diferencia de un enfoque tradicional, los microservicios son elementos independientes que funcionan en conjunto para llevar a cabo las mismas tareas.
2	F	Los microservicios tienen como ventaja la facilidad de implementación, debido a que las aplicaciones basadas en microservicios son más modulares, su implementación es más ágil y sencilla que cuando se trataba de una aplicación monolítica.
3	F	Los microservicios son independientes de lenguajes concretos. Cada microservicio puede desarrollarse en un distinto lenguaje de programación y apoyar a diversas tecnologías de almacenamiento.
4	V	Cada microservicio tiene un nombre único para que pueda resolver su ubicación; un registro de servicios se ocupa de recopilar un directorio.
5	F	En caso de error, un microservicio puede reiniciarse en otra máquina para seguir estando disponible, evitando la pérdida de datos y manteniendo su coherencia. Lo que quiere decir que, si una parte falla, no afectará a toda la aplicación. Algo que sí ocurre con un desarrollo tradicional.
6	V	En el caso de los microservicios, tanto su implementación, actualización y escalado se realizan de manera independiente, así como el control de versiones y el almacenamiento de los estados de los servicios.
7	F	Los microservicios se pueden replicar con facilidad, lo cual permite que una aplicación pueda crecer con mayor rapidez y mejor servicio.
8	V	En el caso de los microservicios, cada servicio es autónomo, por lo tanto, los desarrolladores pueden dar mantenimiento y actualizar cada función de forma independiente sin afectar al resto.
9	F	La capa del almacenamiento de datos proporciona un mecanismo de persistencia, como un motor de almacenamiento de base de datos, archivos log, etc. Cada microservicio dispone de su propia base de datos persistente independientemente para cada microservicio.
10	V	Los contenedores son un ejemplo notable de arquitectura de microservicios, ya que le permite concentrarse en el desarrollo de los servicios sin tener que preocuparse por las dependencias. Las aplicaciones nativas de la nube de hoy en día suelen crearse como microservicios mediante contenedores.

Autoevaluación 2		
Pregunta	Respuesta	Retroalimentación
11	F	La arquitectura de microservicios puede aliviar algunos problemas de seguridad que surgen con las aplicaciones monolíticas. Los microservicios simplifican el monitoreo de seguridad porque las diversas partes de una aplicación están aisladas. Una brecha de seguridad podría ocurrir en una sección sin afectar otras áreas del sistema.
12	V	Docker es una herramienta que permite el uso de contenedores en las instalaciones o en la nube pública o privada.

Ir a la
autoevaluación

Autoevaluación 3		
Pregunta	Respuesta	Retroalimentación
1	V	Se utiliza SOAP para conectarse a un servicio e invocar métodos remotos, aunque puede ser utilizado de forma más genérica para enviar cualquier tipo de contenido.
2	F	La cabecera o en inglés Header, es un elemento opcional, donde se puede incluir información sobre el mensaje.
3	F	Los metadatos se llevan en el encabezado de la solicitud SOAP, mientras que en el cuerpo del mensaje se encuentran los datos en sí.
4	V	En SOAP con el WS-Security se estandariza la forma de proteger y transferir los mensajes usando identificadores únicos llamados tokens.
5	V	SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dichos estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.
6	V	La llamada a procedimiento remoto (RPC) realiza una invocación de una operación que devuelve un resultado. Normalmente se utiliza con la codificación SOAP.
7	F	Los mensajes SOAP suelen ser grandes, ya que deben contener la información que las aplicaciones y los clientes necesitan para analizar los datos que contienen y ejecutar la lógica apropiada. Cuanto más grande es el mensaje, más procesamiento requiere del servidor.
8	V	SOAP es susceptible a diversos ataques y vulnerabilidades relacionados con XML, y más vulnerable aún a los ataques asociados a su protocolo de capa de transporte, normalmente HTTP.
9	V	En el sobre (envelope), se describe el mensaje a quién va dirigido, y cómo debe procesarse. El sobre incluye las definiciones de tipos que se usarán en el documento. Contiene una cabecera de forma opcional, y el cuerpo del mensaje.
10	V	Una descripción WSDL (fichero WSDL) de un servicio web proporciona una descripción entendible por la máquina de la interfaz del servicio web, indicando cómo se debe llamar al servicio, qué parámetros espera, y qué estructuras de datos devuelve.

Ir a la
autoevaluación

Autoevaluación 4		
Pregunta	Respuesta	Retroalimentación
1	V	La API REST siempre se adapta al tipo de sintaxis o plataformas con las que se esté trabajando, lo que ofrece gran libertad al momento de cambiar o probar nuevos entornos dentro del desarrollo.
2	V	La URI es el único elemento que identifica cada recurso de ese sistema REST. La URI facilita acceder a la información para su modificación o borrado, por ejemplo, para compartir su ubicación exacta con terceros.
3	F	La URI no cambia a lo largo del tiempo, ya que la implementación de la arquitectura es la que gestiona los servicios, localiza los recursos, negocia las representaciones, y envía respuestas con los recursos solicitados.
4	V	La operación GET se utiliza para "recuperar" información específica del servidor.
5	F	El método PATCH permite la actualización parcial de un recurso. El método POST es el encargado de la creación de nuevos recursos.
6	V	Los servicios REST se basan en un sistema por capas que permiten hacer uso de servidores intermedios para mejorar la escalabilidad del sistema, y que al mismo tiempo permite el balance de cargas y ofrecer un almacenamiento compartido.
7	F	REST es un tipo de arquitectura de desarrollo web para realizar una comunicación entre el cliente y el servidor.
8	F	La independencia que proporciona frente a cualquier consumidor es su ventaja principal, ya que permite que una misma API REST sea consumida por infinidad de clientes sea cual sea su naturaleza y que el cambio a cualquier otro tipo de consumidor no provoque impacto alguno en ella.
9	V	REST se apoya totalmente en el estándar HTTP y en el uso de métodos como GET, PUT, POST, DELETE, entre otros. Que son indicados en la cabecera HTTP por parte del cliente.
10	V	El uso de hipermedios sirve para explicar la capacidad de una interfaz de desarrollo de aplicaciones para proporcionar al cliente y al usuario los enlaces adecuados, y ejecutar acciones concretas sobre los datos.
11	V	Cuando se consulta una API RESTful se deben especificar parámetros de consulta para que el servicio sepa lo que queremos consultar, basado en una estructura previamente definida provista por el API por medio de una documentación.

Autoevaluación 4

Pregunta	Respuesta	Retroalimentación
12	F	El código 403 Forbidden, indica que la solicitud fue legal, pero el servidor se rehúsa en responder dado que el cliente no tiene los privilegios para hacerla.

[Ir a la
autoevaluación](#)

Autoevaluación 5		
Pregunta	Respuesta	Retroalimentación
1	V	Al analizar los datos los resultados ayudan a la toma de decisiones.
2	V	Hoy en día las redes sociales en línea son las fuentes de datos más grandes que existen.
3	F	Muchos de los datos se encuentran fuera de otras empresas y en las redes sociales en línea.
4	F	La captación de datos puede usarse para la salud, por ejemplo.
5	V	Open Linked Data describe recursos y los enlaza para permitir su enriquecimiento.
6	V	Sin esas características no existiría la necesidad de integración.
7	Consolidación	La descripción es parte de su definición.
8	Propagación	La descripción es parte de su definición.
9	Virtualización	La descripción es parte de su definición.
10	Federación	Técnicamente puede verse como un tipo de virtualización.
11	Warehousing	La descripción es parte de su definición.

[Ir a la autoevaluación](#)



5. Glosario

AI: Integración de Aplicaciones

API: Interfaz de Programación de Aplicaciones

B2B: Business To Business

CORBA: Common Object Request Broker Architecture

CRM: Administración de las relaciones con el cliente

CRUD: Create, Read, Update y Delete.

CSV: Valores Separados por Comas

DDL: Data Definition Language

DML: Data Manipulation Language

EAI: Integración de aplicaciones empresariales

EBI: Integración comercial extendida

EBS: Enterprise Services Bus

EJB: Enterprise Java Beans

EIS: Sistemas de Información Empresarial

ERP: Planeación de Recursos Empresariales

ETL: Extracción Transformación y Cargas

HTTP: Protocolo de Transferencia de HiperTexto

IA: Inteligencia Artificial

IE: Integración empresarial

IP: Internet Protocol

JDBC: Java Data Base Connectivity

JPA: Java Persistence API

JPQL: Java Persistence Query Language

JSON: JavaScrtip Object Notation

K8S: Kubernetes

LAN: Local Areal Network

MIME: MULTipurpose Internet Mail Extensions

NLP: Procesamiento de Lenguaje de Natural

NoSQL: No solo SQL

OLAP: Procesamiento Analítico en Línea.

OSI: Open System Interconnection

REST: Representational state transfer

RPC: Remote Procedure Call

SCI: Integración de la cadena de suministro

SCM: Administración de la cadena de suministros

SIE: Sistemas de Información Empresarial

SOA: Arquitectura Orientada a Servicios

SOAP: Simple Object Access Protocol

SQL: Structured Query Language

TCP: Protocolo de Control de Transmisiones

UDP: User Datagram Protocol

URL: Uniform Resource Locator

URI: Uniform Resource Identifier

VCI: Integración de la cadena de valor

WAN: Wide Area Network

WSDL: Web Services Description Language,

XML: Lenguaje de Marcas eXtendido



6. Referencias bibliográficas

- Atova (n.d). Herramientas de desarrollo Xpath: Sencillas herramientas para editar y depurar código XPath. <https://www.altova.com/es/xpath-tools>
- Au-Yeung, J., Donava, R. (2020, marzo 2). *Best practices for REST API design*. The Overflow. <https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/#h-nesting-resources-for-hierarchical-objects>
- Barila, A., Danubianu, M., & Turcu, C. (2021). Towards Useful Information from Unstructured Data Mining. EIRP Proceedings, 16(1).
- Behler, M. (2020, diciembre 12). JDBC – a short guide. [Marcobehler.com](https://www.marcobehler.com/guides/jdbc).
<https://www.marcobehler.com/guides/jdbc>
- Bhageshpur, K. (2019, noviembre 15). Data Is The New Oil -- And That's A Good Thing. Forbes. Recuperado julio 20, 2021, desde <https://www.forbes.com/sites/forbestechcouncil/2019/11/15/data-is-the-new-oil-and-thats-a-good-thing/?sh=658f3bfe7304>
- Calleja, M. A. (2013). *Tecnologías y programación integrativas*. Editorial Universitaria Ramon Areces.
- Anstey, J., & Ibsen, C. (2018). *Camel in Action*. Manning Publications.
- Cobarsi-Morales, J. (2011). Sistemas de información en la empresa. Editorial UOC. <https://elibro.net/es/lc/bibliotecaupl/titulos/33493>
- Conceptos de seguridad en los servicios Web (n.d). Marco de Desarrollo de la Junta de Andalucía. Recuperado julio 20, 2021, desde <http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/211>
- Celaya Luna, A. (2014). *Cloud: herramientas para trabajar en la nube*. Editorial ICB. <https://elibro.net/es/lc/bibliotecaupl/titulos/56046>
- Daraio, C., & Glänzel, W. (2016). *Grand challenges in data integration—State of the art and future perspectives: An introduction*. *Scientometrics*, 108(1), 391-400.

Data Integration. (2021, Julio 8). Complete Encyclopedia of IT management. https://cio-wiki.org/wiki/Data_Integration

5 Data Integration Methods and Strategies. (n.d). Talend. <https://www.talend.com/resources/data-integration-methods/>

Fatima, N. (2018, diciembre 9). Understanding Data Mapping and Data Modeling Techniques. Astera. <https://www.astera.com/type/blog/understanding-data-mapping-and-its-techniques/>

Fazlollahi, A., & Franke, U. (2018). Measuring the impact of enterprise integration on firm performance using data envelopment analysis. *International Journal of Production Economics*, 200, 119-129.

Freire, D., Frantz, R., Roos-Frantz, F., & Sawicki, S. (2019). *Optimization of the Size of Thread Pool in Runtime Systems to Enterprise Application Integration: A Mathematical Modelling Approach*. TEMA (São Carlos), 20(1), 169-188. Epub June 03, 2019. <https://www.scielo.br/j/tema/a/pNHkzFSvnHsR3XQyvP8BJBS/?format=html>

Gartner Inc. (2016). Gartner IT Glossary. <https://www.gartner.com/en/information-technology/glossary/>

GeeksForGeeks (2019, abril 15). What is structured data. <https://www.geeksforgeeks.org/what-is-structured-data/>

Kanade, S. M. (2019). *Extending the Enterprise Using Enterprise Application Integration (EAI) Technologies for the Cloud*. *International Journal of Applied Evolutionary Computation*, 10(2), 37–42. doi:10.4018/ijaec.2019040105

Levin, G. (2021, julio 27). REST API Security. DZone, <https://dzone.com/asset/download/242331>

Lin, Y., Jun, Z., Hongyan, M., Zhongwei, Z., & Zhanfang, F. (2018). A method of extracting the semi-structured data implication rules. *Procedia computer science*, 131, 706-716.

Malcher, M., Curstis, B., Lawles, C. (2016). *Oracle Data Integration: Tools for Harnessing Data*. McGraw-Hill Education.

MDN Web Docs.(n.d). DOM. <https://developer.mozilla.org/es/docs/Glossary/DOM>

Merriam-Webster. (n.d.). Data. In Merriam-Webster.com dictionary. <https://www.merriam-webster.com/dictionary/data>

Montes Sánchez, J. y Campos Sánchez, A. (2015). *Programación de servicios y procesos*. RA-MA Editorial. <https://elibro.net/es/lc/bibliotecaupl/titulos/62499>

Moreno-Cevallos, J. R., & Dueñas-Holguín, B. L. (2018). Sistemas de información empresarial: la información como recurso estratégico. *Dominio de las Ciencias*, 4(1), 141-154.

Newman, S. (2021). *Building microservices*. " O'Reilly Media, Inc.".

Ortiz, A. (2021, abril 12). Datos semiestructurados, definición, qué son, tipos; ventajas y desventajas. PCWeb. https://pcweb.info/datos-semiestructurados-definicion-que-son-tipos-ventajas-y-desventajas/#Datos_semi-estrucurados

Peiró, R (05 de mayo, 2020). *Sistema de información*. <https://economipedia.com/definiciones/sistema-de-informacion.html>

Proaño, M. F., Orellana, S. Y., & Martillo, I. O. (2018). Los sistemas de información y su importancia en la transformación digital de la empresa actual. *Revista Espacios*, 39(45).

Reeve, A. (2013). *Managing data in motion: data integration best practice techniques and technologies*. Newnes.

Richardson, C. (2018). *Microservices patterns: with examples in Java*. Simon and Schuster.

Romero, D., & Vernadat, F. (2016). Enterprise information systems state of the art: Past, present and future trends. *Computers in Industry*, 79, 3-13.

Rubeš, M., & Jandoš, J. (2013). On Enterprise Integration. *Systémová Integrace*, 20(3).

Salinas, S. O., & Lemus, A. C. (2017). *Data warehouse and big data integration*. Int. Journal of Comp. Sci. and Inf. Tech, 9(2), 1-17.

- Sawicki, S., Frantz, R. Z., Fernandes, V. M. B., Roos-Frantz, F., Yevseyeva, I., & Corchuelo, R. (2016). Characterising enterprise application integration solutions as discrete-event systems. In *Handbook of Research on Computational Simulation and Modeling in Engineering* (pp. 261-288). IGI Global.
- Sherman, R. (2015). *Chapter 12–data integration processes*. Business intelligence guidebook, 301-333.
- Sreemathy, J., Nisha, S., & RM, G. P. (2020, March). Data integration in ETL using Talend. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 1444-1448). IEEE.
- Subramanian, H., & Raj, P. (2019). *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web API*. Packt Publishing Ltd.
- Yadav, A. (2020, agosto 19). *A Look at REST API Design Patterns*. DZone. <https://dzone.com/articles/a-look-at-rest-api-design-patterns>
- Yaddow, W. (n.d.). The Importance of Data Mapping for Data Integration Projects. DataManagementU. <https://www.ewsolutions.com/the-importance-of-data-mapping-for-data-integration-projects/>



7. Anexos

Anexo 1 · Información complementaria de técnicas de integración

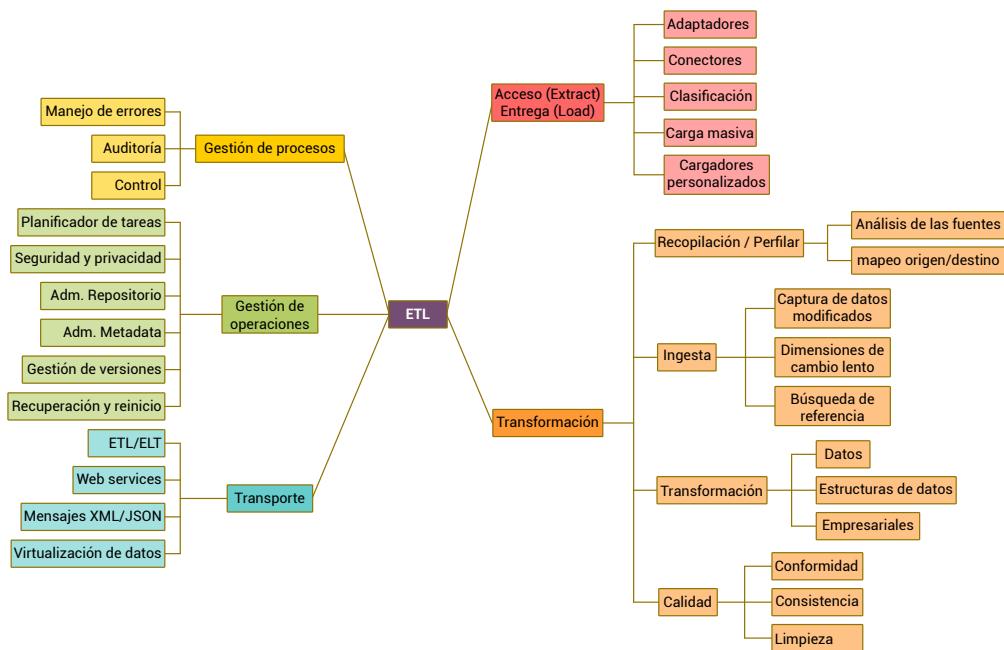
Técnica integración	Pros	Contras	Escenario de uso
Manual. Todo se realiza a “mano” desde la recopilación de los datos hasta la limpieza y la presentación.	<ul style="list-style-type: none">Costo reducido: requiere de poco mantenimiento, ya que integra pocas fuentes de datos.Gran libertad: el usuario tiene control total sobre la integración.	<ul style="list-style-type: none">Menor acceso: se requiere de alguien con experiencia para orquestar la integración.Difícil de escalar: si el proyecto crece se requiere cambiar el código de cada integración.Mayor margen de error: propias de las implementaciones manuales.	Combinar datos para un análisis básico con una pequeña cantidad de fuentes de datos.

Técnica integración	Pros	Contras	Escenario de uso
Basada en aplicaciones.	<ul style="list-style-type: none"> ▪ Procesos simplificados: una aplicación hace todo el trabajo. 	<ul style="list-style-type: none"> ▪ Acceso limitado: se requiere conocimientos técnicos para mantener la aplicación. 	Para empresas que tienen fuentes de datos en la nube (cloud) y en bases de datos internas.
Los programas localizan, recuperan, limpian e integran los datos.	<ul style="list-style-type: none"> ▪ Intercambio de información: la transferencia de datos entre sistemas y departamentos se simplifica. ▪ Uso de pocos recursos: debido a la automatización los gerentes y analistas pueden dedicarse a otros proyectos. 	<ul style="list-style-type: none"> ▪ Resultados inconsistentes: no es un enfoque estándar por lo que los resultados varían según la empresa que ofrece el servicio. ▪ Configuración complicada: diseñar aplicaciones 100% automatizadas no es sencillo, siempre se requiere conocimientos técnicos. ▪ Gestión complicada de datos: acceder a diferentes sistemas puede comprometer la integridad de los datos. 	El nivel de análisis es más complicado.
Middleware.	<ul style="list-style-type: none"> ▪ Mejora la transmisión de datos: el software realiza la integración de datos de la misma manera una y otra vez. ▪ Acceso fácil entre sistemas: el software está construido para facilitar la comunicación entre los sistemas. 	<ul style="list-style-type: none"> ▪ Menos acceso: el middleware debe ser implementado y mantenido por personal técnico. ▪ Funcionalidad limitada: el middleware solo puede funcionar con ciertos sistemas. 	Automatizar la integración de datos entre sistemas heredados (antiguos) y los actuales.

Técnica integración	Pros	Contras	Escenario de uso
Acceso uniforme.	<ul style="list-style-type: none"> ▪ Menores requisitos de almacenamiento: no es necesario crear un nuevo repositorio de datos. 	<ul style="list-style-type: none"> ▪ Desafíos de la integridad de los datos: acceder a fuentes tan diversas puede comprometer la integridad. 	En empresas que necesitan acceder a una gran variedad de sistemas con formatos muy diversos. El nivel de análisis es complejo.
Accede a grupos de datos bastante heterogéneos y los presenta de forma uniforme.	<ul style="list-style-type: none"> ▪ Fácil acceso a los datos: esta técnica funciona bastante bien con diversidad de datos. ▪ Vista simplificada: se crea una apariencia de datos uniformes. 	<ul style="list-style-type: none"> ▪ Sobrecarga de sistemas: si existe gran volumen de datos y alta frecuencia de uso de los mismos 	

Técnica integración	Pros	Contras	Escenario de uso
Almacenamiento común.	<ul style="list-style-type: none"> ▪ Carga reducida: al no manejar consultas constantes a diferentes fuentes. 	<ul style="list-style-type: none"> ▪ Aumento de costos de almacenamiento: tener una copia de los datos significa más recursos. 	En empresas con los suficientes recursos. Este sería el mejor enfoque.
Igual al anterior, pero implica la creación de una copia de los datos en un repositorio común.	<ul style="list-style-type: none"> ▪ Mayor control sobre la gestión de los datos: acceder a los datos de una única fuente mejora la integridad de los datos. ▪ Apariencia de datos más limpia: la copia almacenada permite ejecutar muchas consultas mientras se mantiene la uniformidad en la apariencia de los datos. ▪ Mejor análisis de datos: mantener una copia de los datos permite ejecutar consultas sofisticadas sin comprometer la integridad de los datos. 	<ul style="list-style-type: none"> ▪ Mayores costos de mantenimiento: este enfoque requiere expertos técnicos para configurar, supervisar y mantener la integración. 	

Anexo 2 · Estructura moderna del proceso ETL



Anexo 3 · Metadatos de un archivo PDF

Nombre metadato	Valor
pdf:unmappedUnicodeCharsPerPage	0
pdf:PDFVersion	1.4
pdf:docinfo:title	Plan docente de ANÁLISIS DE REDES SOCIALES
xmp:CreatorTool	LOD-UTPL, j4loxa
pdf:hasXFA	false
access_permission:modify_annotations	true
access_permission:can_print_degraded	true
dc:creator	JORGE AFRANIO LÓPEZ VARGAS
dcterms:created	2019-01-16T21:26:54Z
dcterms:modified	2019-01-16T21:26:54Z
dc:format	application/pdf; version=1.4
pdf:docinfo:creator_tool	LOD-UTPL, j4loxa
access_permission:fill_in_form	true
pdf:docinfo:keywords	UTPL, Plan académico, Plan, LOD-UTPL, ANÁLISIS DE REDES SOCIALES, ARQUITECTURA, ELECTRÓNICA Y TELECOMUNICACIONES, GEOLOGÍA Y MINAS, INGENIERÍA CIVIL, SISTEMAS INFORMÁTICOS Y COMPUTACIÓN, Abr/2018-Ago/2018, LIBRE CONFIGURACIÓN, Nro. Créditos: 3
pdf:docinfo:modified	2019-01-16T21:26:54Z
pdf:encrypted	false
dc:title	Plan docente de ANÁLISIS DE REDES SOCIALES
cp:subject	Plan docente de ANÁLISIS DE REDES SOCIALES (ARQUITECTURA, ELECTRÓNICA Y TELECOMUNICACIONES, GEOLOGÍA Y MINAS, INGENIERÍA CIVIL, SISTEMAS INFORMÁTICOS Y COMPUTACIÓN-UTPL)
pdf:docinfo:subject	Plan docente de ANÁLISIS DE REDES SOCIALES (ARQUITECTURA, ELECTRÓNICA Y TELECOMUNICACIONES, GEOLOGÍA Y MINAS, INGENIERÍA CIVIL, SISTEMAS INFORMÁTICOS Y COMPUTACIÓN-UTPL)
pdf:hasMarkedContent	false
Content-Type	application/pdf
pdf:docinfo:creator	JORGE AFRANIO LÓPEZ VARGAS

Nombre metadato	Valor
dc:subject	UTPL, Plan académico, Plan, LOD-UTPL, ANÁLISIS DE REDES SOCIALES, ARQUITECTURA, ELECTRÓNICA Y TELECOMUNICACIONES, GEOLOGÍA Y MINAS, INGENIERÍA CIVIL, SISTEMAS INFORMÁTICOS Y COMPUTACIÓN, Abr/2018-Ago/2018, LIBRE CONFIGURACIÓN, Nro. Créditos: 3
pdf:producer	iText® 5.5.0 ©2000-2013 iText Group NV (AGPL-version)
access_permission:extract_for_accessibility	true
access_permission:assemble_document	true
xmpTPg:NPages	12
pdf:hasXMP	false
pdf:charsPerPage	2004
access_permission:extract_content	true
access_permission:can_print	true
meta:keyword	UTPL, Plan académico, Plan, LOD-UTPL, ANÁLISIS DE REDES SOCIALES, ARQUITECTURA, ELECTRÓNICA Y TELECOMUNICACIONES, GEOLOGÍA Y MINAS, INGENIERÍA CIVIL, SISTEMAS INFORMÁTICOS Y COMPUTACIÓN, Abr/2018-Ago/2018, LIBRE CONFIGURACIÓN, Nro. Créditos: 3
access_permission:can_modify	true
pdf:docinfo:producer	iText® 5.5.0 ©2000-2013 iText Group NV (AGPL-version)
pdf:docinfo:created	2019-01-16T21:26:54Z

Es necesario mencionar que dentro del entorno de los metadatos no existe el concepto de metadatos obligatorios, todos los metadatos son opcionales, es así que el número de metadatos va a cambiar entre archivos. Para obtener la tabla anterior se utilizó un PDF rico en metadatos que se generaban para un proyecto de la UTPL relacionado con el tema de los metadatos.

Anexo 4 · Tabla de documentación del mapeo

Indicador de mapeo del cambio	Descripción del cambio	Indicador de clave	Origen					Regla de negocio	Nombre de tabla	Destino			
			Tabla/archivo	Nombre del campo	Tipo de dato del campo	Longitud del cambio	Descripción del campo			Nombre del campo de	Tipo de dato	Longitud del campo	Comentario
A	NA	Empleado_Adm	Empleado	Emp_Primer Nombre	Texto	50	Primer nombre del empleado	Directo	EMPLEADO	EMP_P_NOMBRE	Varchar	255	
A	NA	Empleado_Adm	Empleado	Emp_Apellido	Texto	50	Primer apellido del empleado	Directo	EMPLEADO	EMP_APELLIDO	Varchar	255	
A	PK	Empleado_Adm	Empleado	Emp_Nro	Número	10	Número del empleado	Directo	EMPLEADO	EMP_NUM	Varchar	15	
A	NA	Empleado_Adm	Empleado	Emp_Departamento	Texto	50	Departamento del empleado	Si dep = V Si dep = C Si dep = R	VENTAS	EMP_DEP	Varchar	1	

