SIMULACRO SISTEMAS WEB II

Contenido

[1 APUNTES PARA EL EXÁMEN 3](#_Toc197616867)

[2 Introducción a los Servicios Web 3](#_Toc197616868)

[2.1 Definición de servicio web y API. 3](#_Toc197616869)

[2.2 Ventajas y desventajas de utilizar APIs. 4](#_Toc197616870)

[2.3 Tipos de servicios web (REST, SOAP, etc.) y sus diferencias. 5](#_Toc197616871)

[2.4 Webhooks: qué son y cómo funcionan (SÍ entra en el examen). 5](#_Toc197616872)

[2.5 Descripción y diferencias entre WSDL, WADL y SOAP. 6](#_Toc197616873)

[2.6 NOTA: La historia de los servicios web NO entra. 7](#_Toc197616874)

[3 JSON 7](#_Toc197616875)

[3.1 ESTE TEMA NO ENTRA EN PARTE TEÓRICA, SOLO PARTE PRÁCTICA 7](#_Toc197616876)

[4 REST 8](#_Toc197616877)

[4.1 Teoría completa sobre APIs REST (entra en el examen). 8](#_Toc197616878)

[4.2 Qué es REST, sus propiedades, arquitectura RESTful. 8](#_Toc197616879)

[4.3 Constraints (restricciones) del estilo REST: importante. 8](#_Toc197616880)

[4.4 Principios REST: orientación a recursos, uniformidad, etc. 9](#_Toc197616881)

[4.5 Muy importante: repasar las diapositivas de la 46 a la 53. – RESTful 9](#_Toc197616882)

[Servicios RESTful – Nivel 2 Servicios RESTful – Nivel 3 10](#_Toc197616883)

[10](#_Toc197616884)

[**BUENAS PRÁCTICAS EN REST** 10](#_Toc197616885)

[4.6 Ejercicio importante: página 64 (podría aparecer en el examen). 10](#_Toc197616886)

[4.7 Webhooks: también SÍ entran. 10](#_Toc197616887)

[4.8 - Escalando la API: concepto 10](#_Toc197616888)

[4.9 NOTA: YAML y OpenAPI NO entran en teoría (solo parte práctica). 11](#_Toc197616889)

[5 NoSQL 11](#_Toc197616890)

[5.1 Toda la teoría hasta la página 24 entra en el examen. 11](#_Toc197616891)

[**BASE – Brewer** 12](#_Toc197616892)

[5.2 Conceptos clave: modelos de datos NoSQL, diferencias con bases relacionales, ventajas. 12](#_Toc197616893)

[5.3 MongoDB 13](#_Toc197616894)

[5.4 Lo que está después de la página 24 es parte práctica. 14](#_Toc197616895)

[6 XML 14](#_Toc197616896)

[6.1 Validación de XML: se puede pedir si un XML está bien formado. 14](#_Toc197616897)

[6.2 DTD y XSD: definición, diferencias, ventajas e inconvenientes. 14](#_Toc197616898)

[6.3 Ejercicios importantes: especialmente los de conversión entre DTD y XSD. 15](#_Toc197616899)

[6.4 XPath: repasar ejercicios (excepto el de Web Scraping) – Pag 106. 16](#_Toc197616900)

[6.5 XQuery: entender ejemplos y ejercicios, especialmente de las páginas 123 y 124. 16](#_Toc197616901)

[6.6 XSLT: también entra. 16](#_Toc197616902)

[6.7 NOTA: Node.js NO entra. 18](#_Toc197616903)

# APUNTES PARA EL EXÁMEN

**PARTE TEÓRICA (4p)**

No hay parte tórica en el simulacro

**PARTE PRÁCTICA ORDENADOR (6p)**

PARTE I - API (Proyecto con NodeJS, Con OpenAPI)

El de api un proyecto node js y saber usar open api, implementar rutas de la api etc, modificar el openapi o añadir cosas, añadir cosas a node

PARTE II - MONGODB (Script para instalar mongo y colección para acerlo automáticamente)

El de mongo tendremos un script para arrancar en nuestra maquina para instalar mongo y nos dara una coleccion. Y nos hará unas preguntas como en las diapos. El comando solo no el resultado

# Introducción a los Servicios Web

## Definición de servicio web y API.

**API**

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de protocolos y definiciones que permite la comunicación entre diferentes componentes de software. Nos permite usar métodos para invocar (GET, POST, …), Parámetros (Path Parameterers, etc.), Valores de retorno (Código HTTP (200,404, etc.), Formao de datos en la respuesta (JSON, XML, etc.)

* API local: Interacción en la propia máquina. Ej: API del S.O., Biblioteca de C
* API remota (servicio web): Interacción entre máquinas diferentes, comunicación a través de una red. Ej: API de Twitter
* Proveedor(provider): La entidad que ofrece sus servicios a través de la API
* Consumidor(consumer): La entidad que solicita los servicios

**SERVICIO WEB**

[Un servicio web es una tecnología que permite la comunicación y el intercambio de información entre diferentes aplicaciones y sistemas a través de Internet.](https://www.bing.com/ck/a?!&&p=df736d445ab00afef01959c2c04cf7e29497471d18d9ad9afb48924cd97edaabJmltdHM9MTc0NTc5ODQwMA&ptn=3&ver=2&hsh=4&fclid=21b98c16-2025-6ef4-3999-990721626f3e&psq=QUE+ES+UN+SERVICIO+WEB&u=a1aHR0cHM6Ly93ZWJjb3Jwb3JhdGl2YS5lcy9xdWUtZXMtdW4tc2VydmljaW8td2ViLXktZWplbXBsb3Mv&ntb=1)

Captura de pantalla con letras

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

## Ventajas y desventajas de utilizar APIs.

**VENTAJAS API**

* Interoperabilidad:
  + Independientes de la plataforma y del lenguaje
* Estandarización:
  + Las APIs son contratos que no deberían romperse
  + El proveedor promete no cambiar la API
  + Protocolos basados en texto, que hacen que sea más fácil acceder a su contenido y entender su funcionamiento
* Ubicuidad
  + Los servicios web se comunican utilizando HTTP, XML y JSON, por lo tanto, cualquier dispositivo que soporte estas tecnologías puede implementar o acceder a estos servicios
* Encapsulamiento
  + Information hiding
  + Ninguna de las partes conoce la implementación de la otra
* Integración y composición de servicios
  + Reutilización de servicios
  + Creación de servicios más complejos

**DESVENTAJAS API**

* Soporte de transacciones
  + Por detrás de CORBA u otros estándares
* Rendimiento
  + Por su uso de texto y XML, por detrás de RMI, DCOM o CORBA
* Problemas derivados del uso de HTTP
* Interfaces “inmutables”
* Respuesta no garantizada

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.APLICACIONES WEB**

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

## Tipos de servicios web (REST, SOAP, etc.) y sus diferencias.

* SOAP
  + Simple Object Access Protocol, Es un protocolo
  + Expone operaciones que representan lógica
  + Las operaciones se muestran con WSDL
* REST
  + RepresentationalStateTransfer
  + Es una arquitectura de software
  + Se desarrolló paralelamente a HTTP/1.1
  + Cada URL representa un objeto
  1. Interfaz de usuario gráfica, Texto, Aplicación

     El contenido generado por IA puede ser incorrecto.**DIFERENCIAS**

Interfaz de usuario gráfica, Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Tipos de Servicios Web REST

REST APIs should be independent of the protocol(Stateless, Cacheable, Uniform Interface, Idempotent)

* RESTful
  + Orientadas al recurso
  + La función va en el método HTTP
* RPC-Style
  + Orientadas al proceso
  + Toda la información va en el “paquete”, el encapsulamiento no cambia
* REST-RPC Hybrid
  + Mix

## Webhooks: qué son y cómo funcionan (SÍ entra en el examen).

* 1. **Webhooks (ppt)**
* user-definedcallbacksmadewithHTTP
* Mensajes automatizados resultado de un evento concreto
* Mensajes específicos
* Para comunicación uno a uno entre dos appsUsados por ejemplo en CI/CD

**Webhooks (Copilot)**

Los **webhooks** son una forma de comunicación entre aplicaciones que permite enviar datos automáticamente cuando ocurre un evento específico.

* ¿Qué son los webhooks?
  + Son **callbacks definidos por el usuario** que se ejecutan mediante **HTTP**.
  + Permiten el envío de **mensajes automatizados** en respuesta a eventos concretos.
  + Se utilizan para **comunicación uno a uno** entre dos aplicaciones.
* ¿Cómo funcionan?
  + **Evento activador**: Una aplicación detecta un evento (por ejemplo, una compra en una tienda online).
  + **Solicitud HTTP:** La aplicación envía una solicitud POST a una URL predefinida.
  + **Procesamiento:** La aplicación receptora recibe los datos y ejecuta una acción (por ejemplo, actualizar el inventario).
  + **Automatización:** Todo ocurre en tiempo real sin intervención manual.
* Casos de uso
  + **Integración en CI/CD**: Notificar a un sistema de despliegue cuando hay cambios en el código.
  + **E-commerce**: Enviar confirmaciones de pago o actualizaciones de pedidos.
  1. Los webhooks son una **alternativa eficiente a las API tradicionales**,eliminan la necesidad de realizar consultas constantes y permiten una comunicación más rápida y automatizada.

## Descripción y diferencias entre WSDL, WADL y SOAP.

**WSDL (Se utiliza para describir servicios web. Definir la interfaz pública de un servicio web, especificando cómo los clientes pueden interactuar con él)**

* Web Services Description Language
* Basado en XML
* **Describe la interfaz pública de los servicios web**
* Un cliente puede conectarse a un servicio web para leer el WSDL y determinar qué funciones están disponibles en el servidor

**WADL (Se utiliza para describir aplicaciones web basadas en HTTP, especialmente servicios web REST)**

* Web Application Description Language, Basado en XML
* Describe la interfaz pública de los servicios web.
* WSDL en principio no estaba diseñado para REST al no admitir distintos métodos HTTP que POST, por ello surgió WADL
* Menos flexible que WSDL, pero más ligero y fácil de usar
* Sin embargo, para servicios REST no es necesaria una descripción por lo que su uso no está muy extendido

**Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**

## NOTA: La historia de los servicios web NO entra.

# JSON

## ESTE TEMA NO ENTRA EN PARTE TEÓRICA, SOLO PARTE PRÁCTICA

# REST

## Teoría completa sobre APIs REST (entra en el examen).

* REST: REpresentationalStateTransfer
* Independiente del protocolo
  + Soporte de URIs
  + Principalmente usado con HTTP
* Estilo de arquitectura
  + No es un estándar (no hay RFC)
  + No hay "hardrules"

## Qué es REST, sus propiedades, arquitectura RESTful.

**PROPIEDADES**

* Heterogeneidad
* Escalabilidad
* Evolvability
* Visibilidad
* Reliability
* Eficiencia

**ARQUITECTURA**

* Características principales:
  + Sigue el paradigma cliente-servidor
  + Sin estado
  + Sistema basado en capas
  + Interfaz uniforme y desacoplada.
  + Hiperenlace como motor
  + Cacheable
* Orientado al recurso

Perspectivas en el proceso de diseño de la arquitectura

* Empieza con nada
  + Construye una arquitectura con componentes familiares
  + Hasta que cumpla las necesidades de la intención del propio sistema
  + ¿Enfatiza en creatividad y una visión **unbounded?**
* Empieza con las necesidades del sistema como un agujero
  + ¿Empieza sin restricciones?
  + ¿Incrementa la identidad y aplica restricciones?
  + Enfócate en restricciones y en entender el contexto del sistema

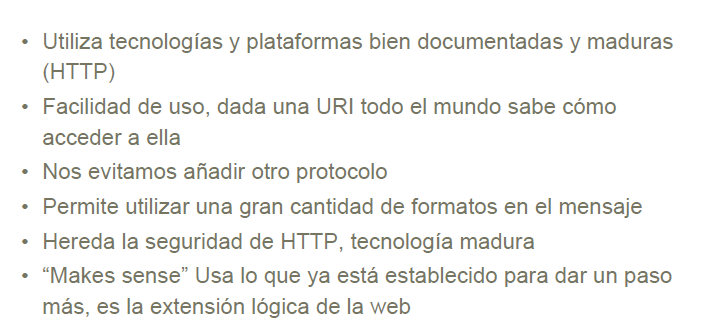
## Constraints (restricciones) del estilo REST: importante.

**CONSTRAINTS**

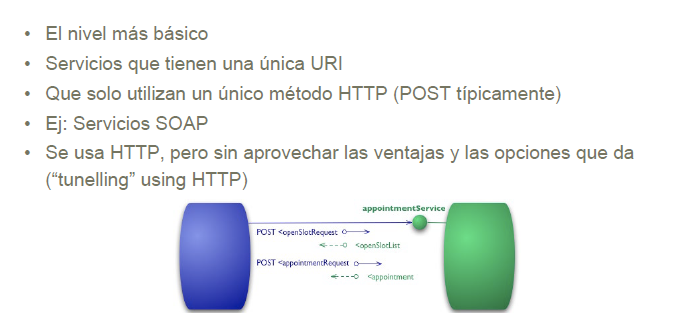
* Client-Server
* Stateless
* Cacheable
* Layered System
* Codeo on Demand
* Uniform Interface

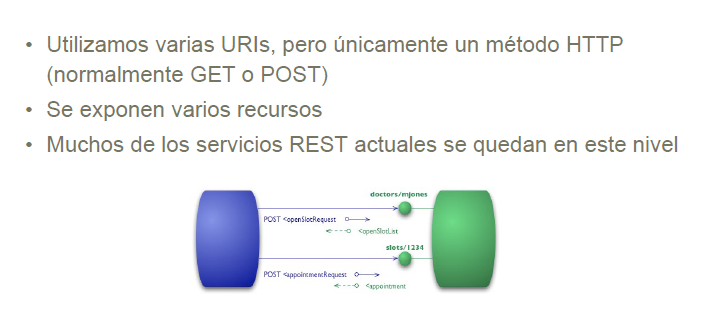
## Principios REST: orientación a recursos, uniformidad, etc.

* Interfaz Uniforme
  + Los recursos (URL, Formatos, Métodos) se identifican de forma unívoca por su URI
  + Servicios permiten leer, crear, modificar y eliminar recursos
  + Operaciones se corresponden con los métodos HTTP (GET, POST, PUT, DELETE, etc.)
  + Códigos estado corresponden a los de HTTP (200, 404, etc.)
* Orientado al recurso
  + CON HTTP el recurso se presenta en el cuerpo del mensaje
  + El tipo de formato se especifica mediante el tipo de contenido
  + Cliente puede negociar con Accept
  + Content-Type (text/plain, text/html, application/xml, application/json)
  + Contrato de servicio WADL(~= WSDL de SOAP, MIRAR PPT diap 32 para más información)
* Sin estado
* HATEOAS
* Seguridad en REST
* VENTAJAS DE REST



## Muy importante: repasar las diapositivas de la 46 a la 53. – RESTful

**Servicios RESTful – Nivel 0 Servicios RESTful – Nivel 1**

****

## Servicios RESTful – Nivel 2 Servicios RESTful – Nivel 3

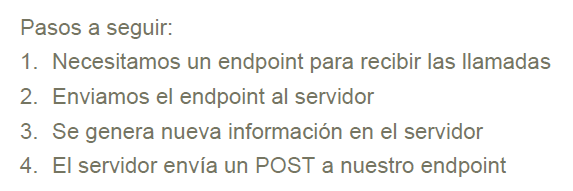
## 

**BUENAS PRÁCTICAS EN REST**

* Nombrado sustantivos no verbos
* Múltiples URLs por recurso
* API debe ser intuitiva a la hora de definir las asociaciones
* No sobrecargar al cliente, usar paginación si el contenido es muy grande
* Recordar que las peticiones GET tienen un tamaño máximo
* Servicio tiene muchos argumentos o con valores de tamaño variable se debe utilizar POST o PUT
* Usar correctamente los métodos y códigos de error HTTP
* Para filtrar, paginar, etc. -> añaden parámetros GET
* Aplicar HATEOAS
* Pensar como el consumidor va a realizar las operaciones necesarias

## Ejercicio importante: página 64 (podría aparecer en el examen).

## Webhooks: también SÍ entran.



**Webhooks DEMO**

## - Escalando la API: concepto

* Scaling Troughput
  + Número de llamadas a la API por segundo
  + Encontrar cuellos de botella (Recolectar datos de uso, Monitorizar la capacidad)
  + Soluciones
    - Análisis de código
    - Añadir recursos
    - Indexar las DB
* Evolving API Design
  + Introduciendo nuevos patrones de acceso a datos
  + Añadiendo nuevos métodos API
  + Apoyando bulk Endpoints
  + Añadiendo nuevos filtros(Search filter, date filter, order filter
* Paginación
  + Evitamos devolver demasiados datos
  + Offset-Based Pagination
  + Cursor-Based Pagination
* Rate-Limiting APIs
  + Limitamos el número de llamadas a nuestra API
  + Evitamos ataques DoS y spam
  + Podemos permitir límites globales o específicos por endpoints
  + Tenemos que medir el tráfico
  + Puede que nos interese permitir picos de tráfico
  + Quizás tengamos excepciones
* Hay que informar al cliente del número de peticiones restantes
  + Indicarlo en las cabeceras
  + Una API que pueda consultarlo

## NOTA: YAML y OpenAPI NO entran en teoría (solo parte práctica).

# NoSQL

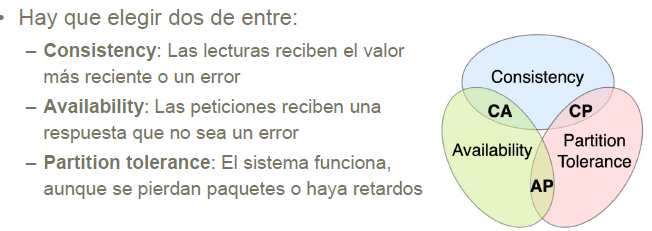
## Toda la teoría hasta la página 24 entra en el examen.

**INTRODUCCIÓN**

* Aparecieron en los años 60 las BBDD No Relacionales
* NoSQL (Not Only SQL)
* Surgieron como soluciones específicas para las necesidades de grandes empresas
* Resolver el “impedance mismatch” (Conversión de clases y objetos a una tabla, encapsulación, herencia, tipos de datos, etc.)

**ACID**

* Atomicity: la transacción se ejecuta completa o no se ejecuta
* Consistency: las transacciones mantienen la validez
* Isolation: no importa el orden en el que se ejecuten las transacciones
* Durability: cuando la transacción se ha completado se guarda en memoria no volátil

**CAP – Teorema de Brewer**

**BASE – Brewer**

Versión ACID de NoSQL

Propiedades:

* Basic availability: el sistema está disponible en caso de fallo
* Softstate: el estado puede cambiar sin interacciones
* Eventual consistency: después de un tiempo sin inputs, el sistema será consistente

## Conceptos clave: modelos de datos NoSQL, diferencias con bases relacionales, ventajas.

**Qué es y para qué se usa**

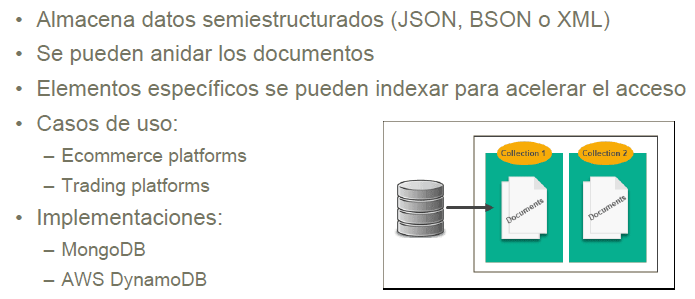
* **NoSQL** (Not Only SQL) es una categoría de sistemas de gestión de bases de datos que no se basan en el modelo relacional tradicional. En lugar de utilizar tablas y relaciones, las bases de datos NoSQL emplean diferentes modelos de datos, como documentos, grafos, columnas y pares clave-valor. Estas bases de datos están diseñadas para manejar grandes volúmenes de datos, proporcionar alta disponibilidad y escalar horizontalmente.

**Características NoSQL**

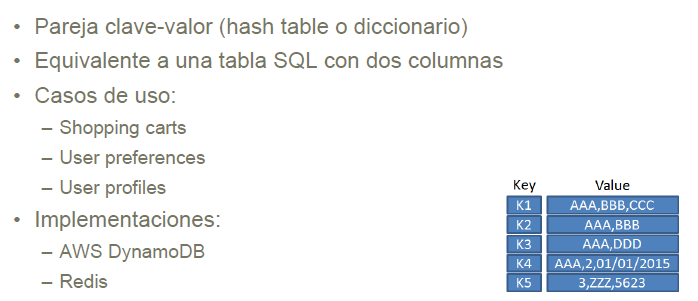
* No hay un modelo relacional
* No usan SQLAlguno puede tener un lenguaje parecido (ej. Cassandra’sCQL)
* Pensado para sistemas distribuidos (clusters)
* Schemaless: puedes añadir datos sin tener que definir primero la estructura
* Open source

**Tipos de bases de datos NoSQL**

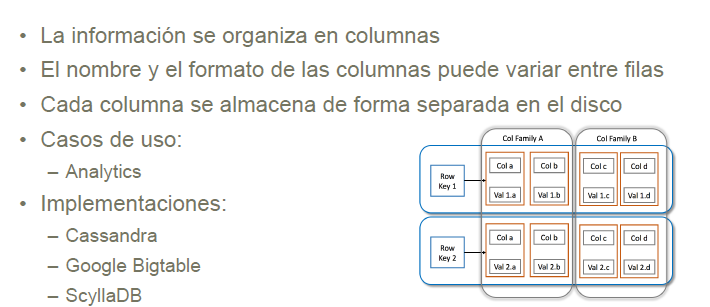
* Document Databases



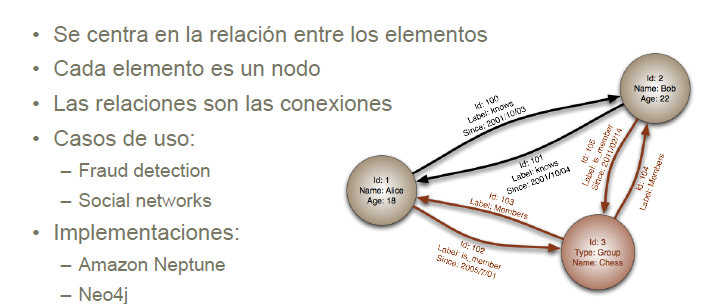
* Key-value Stores



* Wide-column store



* Graph Databases



**Polyglot persistence**

* Usar diferentes tecnologías en función de las necesidades
* No usar una misma base de datos para todo
* Basado en Polyglotprogramming

## MongoDB

**Características**

* Ad-hoc queries(MQL)
* Índices
* Replicación
* Load balancing
* Pipelines de agregación
* Transacciones

**Conceptos**

* Document: a way to organize and store data as a set of field-value pairs
* Field: a unique identifier for a datapoint
* Value: data related to a given identifier
* Collection:
  + An organized store of documents in MongoDB
  + Usually with common fields between documents
  + There can be many collections per database
  + There can be many documents per collection
* Replica Set:
  + A few connected machines that store the same data
  + They ensure that if something happens to one of the machines the data will remain intact
  + Comes from the word replicate - to copy something
* Instance: a single machine locally or in the cloud, running a certain software, in our case it is the MongoDB database
* Cluster: group of servers that store your data

## Lo que está después de la página 24 es parte práctica.

# XML

## Validación de XML: se puede pedir si un XML está bien formado.

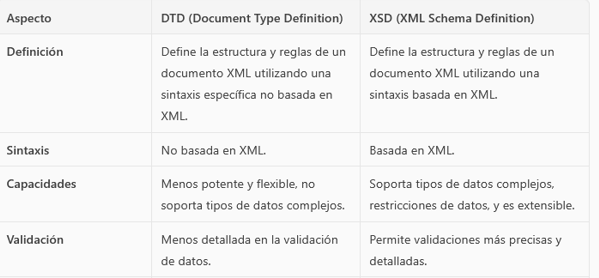
## DTD y XSD: definición, diferencias, ventajas e inconvenientes.

**DTD**

* “DocumentTypeDescriptor”
* Descriptor de tipo de documento
* Estándar para validar XML.
  + Lenguaje que proporciona una gramática para especificar: Elementos, Atributos, Anidado, Ordenación, Número de apariciones

**XSD**

* “XML SchemaDefinition”, También conocido simplemente como "XML Schema"
* Estándar para validar XML (esp1, esp2), Basado en XML
  + Lenguaje que proporciona una gramática para especificar: Elementos, orden, anidación y número de apariciones, Atributos, Tipos de datos y valores por defecto



Tabla

El contenido generado por IA puede ser incorrecto.

## Ejercicios importantes: especialmente los de conversión entre DTD y XSD.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

## XPath: repasar ejercicios (excepto el de Web Scraping) – Pag 106.

## XQuery: entender ejemplos y ejercicios, especialmente de las páginas 123 y 124.

**XQuery**

* Lenguaje para realizar consultas sobre XML
* Similar en concepto a SQL
* Versión 3.1 en 2017
* Cada expresión opera sobre y devuelve una secuencia de elementos:Documento XML o StreamXML
* XPathes uno de los tipos de expresiones que soportaPodemos incluir XPathdentro de XQuery
  1. **Diagrama

     El contenido generado por IA puede ser incorrecto.**

## XSLT: también entra.

**XSLT (Extensible Stylesheet Language Transformations)** es un lenguaje de transformación de documentos XML. Permite transformar un documento XML en otro formato, como HTML, texto plano, o incluso otro documento XML. XSLT es parte de la familia de tecnologías XSL (Extensible Stylesheet Language), que también incluye XSL-FO (Formatting Objects) para la presentación de documentos.

**Características de XSLT**

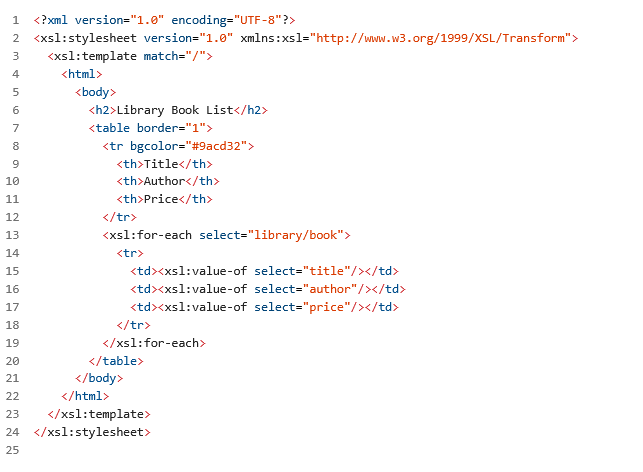
1. **Transformación de Documentos**: XSLT se utiliza para transformar la estructura y el contenido de un documento XML en otro formato.
2. **Plantillas**: Utiliza plantillas para definir cómo se deben transformar los elementos y atributos del documento XML.
3. **XPath**: XSLT utiliza XPath para navegar y seleccionar partes del documento XML que se van a transformar.
4. **Extensible**: Permite la creación de funciones y plantillas personalizadas para realizar transformaciones complejas.

**Ejemplo de XSLT**

Supongamos que tenemos el siguiente documento XML:

Texto

El contenido generado por IA puede ser incorrecto.



Queremos transformarlo en un documento HTML que muestre una lista de libros. Aquí tienes un ejemplo de XSLT para lograrlo:

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.VENTAJAS E INCONVENIENTES**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

## NOTA: Node.js NO entra.

# Microservicios