Tema 9: Servicios REST

¿Cuál de las siguientes características NO es típica de los servicios web? 0 puntos

- ○ Accesibilidad a través de protocolos estándar
- ○ Independencia de lenguaje y plataforma
- ◉ Acoplamiento fuerte entre cliente y servidor
- ○ Autodescripción y localización

¿Cuál es la principal diferencia entre servicios web y tecnologías como CORBA y RMI? 1 punto

- ○ Los servicios web utilizan tecnologías propietarias
- ◉ Los servicios web se basan en protocolos estándar como HTTP y TCP
- ○ Los servicios web requieren un entorno Java para funcionar
- ○ Los servicios web no permiten el uso de XML o JSON para transmitir datos

¿Qué objetivo cumplen los servicios web al utilizar estándares como XML y JSON? 1 punto

- ◉ Interoperabilidad entre aplicaciones
- ○ Mayor seguridad frente a ataques externos
- ○ Control centralizado del flujo de datos
- ○ Aislamiento total entre clientes y servidores

¿Cuál es una ventaja de usar servicios web en comparación con otras tecnologías de computación distribuida? 1 punto

- [ ] Mayor rendimiento y velocidad
- [x] Mayor interoperabilidad y modularidad
- [ ] Menos consumo de recursos de red
- [ ] Menor necesidad de seguridad

Los servicios web permiten el uso de HTTP como protocolo de aplicación. ¿Qué ventaja conlleva esta característica? 1 punto

- [x] Permite la interacción entre diferentes plataformas a través de Internet
- [ ] Aumenta la velocidad de transmisión de datos
- [ ] Reduce el consumo de ancho de banda
- [ ] Simplifica la creación de interfaces gráficas

Los servicios web se basan en estándares abiertos para lograr la interoperabilidad. ¿Cuál de los siguientes NO es un estándar utilizado en servicios web? 1 punto

- [ ] SOAP
- [x] CORBA
- [ ] WSDL
- [ ] UDDI

¿Qué característica de los servicios web permite una fácil escalabilidad y modularidad del sistema? 1 punto

- [ ] Uso de XML y JSON para transmitir datos
- [x] Desacoplamiento entre cliente y servidor
- [ ] Interacciones síncronas entre componentes
- [ ] Dependencia de tecnologías específicas

¿Qué característica de los servicios web reduce la necesidad de cambiar reglas de filtrado en firewalls? 1 punto

- [x] Uso de HTTP como protocolo de transporte
- [ ] Uso de servicios web para todas las comunicaciones internas
- [ ] La modularidad del sistema
- [ ] El empleo de tecnologías específicas para cada entorno

¿Cuál de las siguientes no es una desventaja de los servicios web? 1 punto

- [ ] Bajo rendimiento comparado con otros modelos de computación distribuida
- [ ] Riesgos de seguridad debido a la visibilidad de URIs públicas
- [x] Dificultad para adaptarse a estándares abiertos
- [ ] Complejidad en el desarrollo sin herramientas adecuadas

¿Qué principio guía el diseño de servicios web para garantizar su independencia de plataforma y lenguaje? 1 punto

- ○ Utilización de protocolos propietarios
- ◉ Uso de estándares abiertos para la comunicación
- ○ Control centralizado de todas las interacciones
- ○ Implementación única para cada lenguaje de programación

¿Qué es un servicio web SOAP? 1 punto

- ◉ Un servicio web que utiliza Simple Object Access Protocol para la comunicación
- ○ Un servicio web que está basado en JavaScript y JSON
- ○ Un servicio web diseñado específicamente para navegadores
- ○ Un servicio web que usa solo métodos GET y POST para la comunicación

¿Cuál de las siguientes afirmaciones es cierta respecto a los servicios web RESTful? 1 punto

- ◉ Utilizan HTTP como protocolo de comunicación
- ○ Requieren un servidor dedicado para funcionar
- ○ Son siempre síncronos
- ○ Solo utilizan el método POST para la comunicación

¿Qué tipo de servicios web se centra en el uso de recursos y métodos HTTP para interactuar con ellos? 1 punto

- (●) Servicios web RESTful
- ( ) Servicios web SOAP
- ( ) Servicios web CORBA
- ( ) Servicios web DCOM

¿Cuál es la función de un WSDL (Web Services Description Language) en el contexto de servicios web SOAP? 1 punto

- (●) Describir la estructura y operaciones de un servicio web SOAP
- ( ) Proporcionar la URL del servicio web
- ( ) Determinar qué clientes pueden acceder al servicio web
- ( ) Gestionar el tráfico entre servicios web

¿Qué se recomienda al diseñar una jerarquía de URLs para una API? 1 punto

- (●) Mantenerla simple y coherente para un uso intuitivo
- ( ) Usar números aleatorios como identificadores únicos
- ( ) Incluir información sensible en las URLs
- ( ) Evitar el uso de versiones en la estructura de las URLs

En el diseño de una API, ¿por qué se recomienda no usar el método GET para realizar cambios en los recursos? 1 punto

- [x] Porque GET se almacena en caché y no debe cambiar el estado de un recurso
- [ ] Porque GET es más lento que otros métodos HTTP
- [ ] Porque GET solo admite un parámetro
- [ ] Porque GET es el único método que no es seguro

¿Por qué es importante incluir la versión de la API en la URL? 1 punto

- [x] Para gestionar cambios en la API sin romper la compatibilidad con versiones anteriores
- [ ] Para ayudar a los motores de búsqueda a indexar mejor la API
- [ ] Para aumentar el rendimiento de la API
- [ ] Para evitar el uso de métodos HTTP inseguros

¿Cuál es una buena práctica para proporcionar capacidades de filtrado y ordenado en una API? 1 punto

- [x] Permitir que los usuarios utilicen parámetros de consulta para especificar criterios de filtrado y orden
- [ ] Usar solo rutas fijas para todos los recursos
- [ ] Hacer que el filtrado y ordenado solo sea posible en el servidor
- [ ] Evitar el uso de filtros y ordenaciones para simplificar la API

¿Cuál de las siguientes es una práctica recomendada para especificar el formato de datos en una API? 1 punto

- [x] Usar cabeceras HTTP para indicar el formato de respuesta
- [ ] Forzar siempre el uso de XML
- [ ] Solo permitir formatos binarios
- [ ] No especificar el formato, permitiendo que el cliente adivine

¿Qué es la Iniciativa OpenAPI y por qué es relevante para el diseño de APIs? 1 punto

- [x] Una comunidad que proporciona especificaciones para estandarizar el diseño, documentación y consumo de APIs REST
- [ ] Una empresa que ofrece servicios de diseño de APIs
- [ ] Un grupo que se especializa en pruebas de seguridad para APIs
- [ ] Un consorcio de desarrolladores que crean APIs propietarias

¿Cuál es una práctica recomendada para el uso de identificadores únicos (IDs) en las URLs de una API? 1 punto

- [x] Usar un formato que sea consistente y fácil de recordar
- [ ] Usar un hash aleatorio para mejorar la seguridad
- [ ] Incluir información sensible como el número de seguro social o DNI
- [ ] Reutilizar identificadores de otras aplicaciones para mantener la consistencia

¿Qué enfoque es recomendable para gestionar los errores en una API? 1 punto

- [x] Devolver códigos de error HTTP apropiados para cada tipo de error
- [ ] Devolver siempre un código 500 para todos los errores
- [ ] No enviar información de errores al cliente para evitar filtraciones de datos
- [ ] Devolver un código 200 y manejar errores en el cliente

¿Cuál de las siguientes es una práctica recomendada para gestionar versiones de una API? 1 punto

- [x] Incluir la versión en la URL para facilitar la compatibilidad con versiones anteriores
- [ ] Devolver siempre la última versión para mantener la simplicidad
- [ ] Forzar a los clientes a usar la versión más reciente
- [ ] No permitir el uso de versiones antiguas para evitar confusiones

¿Qué es una práctica recomendada para gestionar permisos y autorizaciones en una API? 1 punto

- [x] Usar tokens de acceso para controlar el acceso a recursos específicos
- [ ] Permitir acceso abierto para reducir la complejidad de la API
- [ ] Usar solo autenticación basada en IP para controlar el acceso
- [ ] Deshabilitar la autenticación para mejorar el rendimiento

¿Cuál es la principal ventaja de usar tokens para el control de acceso en una API REST? 1 punto

- ○ Permiten el uso de credenciales únicas y reutilizables para múltiples servicios
- ○ Los tokens son fáciles de compartir entre diferentes usuarios
- ◉ Permiten el control granular del acceso y son útiles para autenticación sin estado
- ○ Los tokens son más seguros que las contraseñas

¿Qué característica clave debe tener un sistema de tokenización para control de acceso? 1 punto

- ◉ Los tokens deben ser únicos por usuario y válidos para sesiones cortas
- ○ Los tokens deben ser persistentes para reutilización en múltiples sesiones
- ○ Los tokens deben ser de fácil reproducción para mejorar la interoperabilidad
- ○ Los tokens deben tener una clave de descifrado común entre múltiples servicios

En un sistema basado en tokens, ¿cómo se asegura la autenticidad de un token? 1 punto

- ◉ Usando una clave secreta conocida solo por el servidor para firmar los tokens
- ○ Aplicando una clave secreta común que el cliente y el servidor comparten
- ○ Haciendo que el token sea un simple ID que el servidor puede validar
- ○ Utilizando un método de cifrado que garantiza que solo el cliente puede desencriptar el token

¿Qué es un "token de actualización" (refresh token) en un sistema de control de acceso mediante tokens? 1 punto

- [x] Un token que permite a un usuario obtener un nuevo token de acceso sin volver a autenticarse
- [ ] Un token que solo se usa para reiniciar la sesión del usuario
- [ ] Un token que se envía al servidor para obtener un nuevo token cuando el original expira
- [ ] Un token que puede ser reutilizado indefinidamente para evitar expiración

¿Por qué es importante establecer un tiempo de expiración para los tokens de acceso? 1 punto

- [x] Para minimizar el impacto en caso de que un token sea comprometido
- [ ] Para garantizar que el usuario tenga que volver a autenticarse con frecuencia
- [ ] Para reducir la carga en el servidor y evitar acumulación de tokens
- [ ] Para asegurarse de que los tokens no se usen en exceso

¿Cuál es un riesgo potencial al usar tokens para control de acceso? 1 punto

- [ ] Si un token es robado, puede usarse para acceder a recursos protegidos hasta que expire
- [ ] Los tokens pueden ser reutilizados indefinidamente si no se controlan adecuadamente
- [ ] Los tokens pueden ser interceptados y reutilizados en ataques de "replay"
- [x] Todos los anteriores

En un sistema de control de acceso mediante tokens, ¿cuál es el principal propósito del token de acceso? 1 punto

- [x] Autenticar al usuario y otorgar permisos para acceder a recursos específicos
- [ ] Almacenar datos sensibles que el servidor puede usar para identificar al usuario
- [ ] Actuar como contraseña para que el servidor permita el acceso
- [ ] Asegurar que el usuario se autentique cada vez que haga una solicitud

¿Qué hace que los tokens sean útiles para sistemas de autenticación sin estado? 1 punto

- [x] El servidor no necesita retener información de sesión entre solicitudes
- [ ] Los tokens son únicos y no reutilizables
- [ ] Los tokens son almacenados en el lado del cliente para evitar sobrecarga del servidor
- [ ] Los tokens son validados por el cliente antes de ser enviados al servidor

¿Cuál de los siguientes es un problema potencial al compartir tokens entre diferentes aplicaciones? 1 punto

- [ ] Puede dar lugar a riesgos de seguridad si no se implementa correctamente
- [ ] Los tokens compartidos pueden ser utilizados por usuarios no autorizados
- [ ] Puede crear una falta de control sobre quién tiene acceso a qué recursos
- [x] Todas las anteriores

En el modelo de autenticación basado en tokens, ¿cómo se protege un token de acceso? 1 punto

- [x] Almacenando el token de forma segura y transmitiéndolo solo a través de canales cifrados
- [ ] Encriptando el token en cada solicitud para garantizar su seguridad
- [ ] Requiriendo una autenticación secundaria para usar el token
- [ ] Asegurándose de que el token no expire para mantener su seguridad

¿Qué situación puede requerir el uso de un "token de actualización" (refresh token)? 1 punto

- [x] Cuando el token de acceso ha expirado y se necesita uno nuevo sin que el usuario vuelva a autenticarse
- [ ] Cuando el usuario necesita cambiar su contraseña
- [ ] Cuando el servidor quiere extender la duración de la sesión
- [ ] Cuando el cliente solicita una operación que requiere mayor seguridad

¿Qué método se utiliza comúnmente para garantizar la autenticidad y la integridad de un token? 1 punto

- [x] Firmar el token con una clave secreta utilizando algoritmos criptográficos
- [ ] Cifrar el token con una clave que solo el servidor conoce
- [ ] Generar un hash del token y compararlo con un valor conocido
- [ ] Utilizar certificados digitales para validar el token

¿Por qué es importante evitar que los tokens se reutilicen fuera de contexto? 1 punto

- ○ Para prevenir ataques de repetición ("replay attacks")
- ○ Para evitar que un token comprometido sea usado para obtener acceso no autorizado
- ○ Para mantener la integridad del sistema de autenticación
- ◉ Todas las anteriores

¿Qué significa Intercambio de Recursos de Origen Cruzado (CORS)? 1 punto

- ◉ Permite solicitudes HTTP entre diferentes dominios de manera segura
- ○ Limita las solicitudes HTTP a un solo origen para mayor seguridad
- ○ Controla el acceso a recursos mediante certificados SSL
- ○ Solo permite solicitudes HTTP entre servidores de un mismo origen

¿Cuál es el propósito del encabezado HTTP "Access-Control-Allow-Origin" en el contexto de CORS? 1 punto

- ◉ Especificar qué dominios tienen permiso para acceder a un recurso
- ○ Indicar los métodos HTTP permitidos para acceder a un recurso
- ○ Definir qué encabezados HTTP pueden ser usados durante la solicitud
- ○ Indicar el tiempo de expiración para un recurso compartido

Si el encabezado "Access-Control-Allow-Origin" está configurado como "*", ¿qué implica? 1 punto

- (●) Cualquier origen puede acceder al recurso
- ( ) Solo los orígenes con certificados SSL pueden acceder al recurso
- ( ) Ningún origen tiene permiso para acceder al recurso
- ( ) Solo orígenes específicos tienen permiso para acceder al recurso

¿Cuál es el riesgo de usar "Access-Control-Allow-Origin: *" en un servidor de producción? 1 punto

- (●) Puede aumentar el riesgo de ataques de origen cruzado
- ( ) Limita las solicitudes solo a dominios seguros
- ( ) No es compatible con la mayoría de los navegadores
- ( ) Puede afectar negativamente al rendimiento del servidor

¿Qué significa el encabezado "Access-Control-Allow-Credentials: true"? 1 punto

- (●) El servidor permite que las credenciales (como cookies) se incluyan en solicitudes de origen cruzado
- ( ) Solo solicitudes con credenciales autenticadas son permitidas
- ( ) Permite que el servidor incluya credenciales adicionales en la respuesta
- ( ) Indica que el servidor tiene credenciales cifradas para recursos cruzados

¿Qué es un "preflight request" en el contexto de CORS? 1 punto

- [x] Una solicitud de comprobación enviada por el navegador antes de realizar una solicitud HTTP de origen cruzado
- [ ] Una solicitud que permite al servidor validar credenciales antes de acceder a un recurso
- [ ] Una solicitud que evita solicitudes malintencionadas a través de CORS
- [ ] Una solicitud HTTP que solo se utiliza para verificar la validez de las credenciales del usuario

En una solicitud preflight, ¿qué encabezado indica los métodos HTTP permitidos para acceder a un recurso? 1 punto

- [x] Access-Control-Allow-Methods
- [ ] Access-Control-Allow-Headers
- [ ] Access-Control-Allow-Origin
- [ ] Access-Control-Allow-Credentials

¿Por qué es importante usar encabezados CORS en aplicaciones web? 1 punto

- [x] Para permitir solicitudes HTTP seguras entre diferentes orígenes
- [ ] Para evitar ataques de denegación de servicio (DDoS)
- [ ] Para garantizar que las solicitudes sean encriptadas
- [ ] Para restringir el acceso a recursos en el mismo dominio

¿Cuál de los siguientes es un principio fundamental del estilo arquitectónico REST? 1 punto

- [ ] Uso de HTTP para todas las comunicaciones
- [x] Enfoque en recursos en lugar de funciones
- [ ] Implementación de estado persistente en el servidor
- [ ] Uso de servicios de seguridad avanzados como OAuth

¿Qué característica clave diferencia a REST de otros estilos de arquitectura como SOAP? 1 punto

- [ ] REST utiliza XML exclusivamente para transmitir datos
- [x] REST es sin estado, mientras que SOAP puede ser con o sin estado
- [ ] REST requiere un esquema de seguridad avanzado
- [ ] REST tiene una estructura de servicio monolítica

¿Cuál es el propósito del principio HATEOAS (Hypermedia As The Engine Of Application State) en REST? 1 punto

- [x] Proveer hipervínculos para guiar a los clientes a través de las interacciones con el servidor
- [ ] Establecer una interfaz uniforme para todos los recursos
- [ ] Implementar un sistema de capas para mejorar la escalabilidad
- [ ] Hacer que los recursos sean autodescriptivos

En una API RESTful, ¿cuál es la diferencia clave entre los métodos HTTP POST y PUT? 1 punto

- [x] POST crea recursos nuevos, mientras que PUT puede actualizar o reemplazar recursos existentes
- [ ] POST siempre reemplaza recursos existentes, mientras que PUT crea nuevos recursos
- [ ] POST es para solicitudes sin estado y PUT es para solicitudes con estado
- [ ] POST se usa para operaciones de lectura y PUT para operaciones de escritura

En el diseño REST, ¿qué se entiende por "protocolo sin estado"? 1 punto

- [x] El servidor no retiene información sobre solicitudes pasadas; cada solicitud es independiente
- [ ] El servidor guarda el estado de las solicitudes en una base de datos para referencia futura
- [ ] Los clientes deben proporcionar un estado persistente para mantener el contexto
- [ ] Los datos se transmiten sin formato para evitar el almacenamiento de estado

¿Cuál de las siguientes opciones describe mejor el sistema de capas en la arquitectura REST? 1 punto

- [x] Las solicitudes y respuestas pasan por múltiples capas que proporcionan funcionalidades específicas
- [ ] El sistema de capas implementa una estructura jerárquica para el control de acceso
- [ ] El sistema de capas permite la duplicación de datos para mayor seguridad
- [ ] Las capas permiten definir reglas de negocio y roles de usuario para solicitudes REST

¿Cómo puede un servicio RESTful mejorar el rendimiento del cliente mediante el uso de caché? 1 punto

- [x] Al permitir que los recursos sean almacenados en caché, se reducen las solicitudes repetitivas al servidor
- [ ] Al enviar solo datos comprimidos, se optimiza la carga de la red
- [ ] Al usar servidores intermedios para reducir la carga del servidor principal
- [ ] Al permitir que los clientes almacenen información de sesión para futuras solicitudes