APIs

Sistemas y Tecnologías Web (CC3062) - 2026

APIs

Semestre 02, 2026

Contexto

Las aplicaciones no viven solas

- Una app necesita datos.
- Esos datos suelen estar en otro sistema.
- ¿Cómo se comunican?

Ejemplo cotidiano

- App del clima.
- No tiene satélites propios.
- Consulta datos a otro sistema.

¿Cómo permitimos que dos sistemas distintos hablen entre sí sin compartir todo su código interno?

Definición

- API = Application Programming Interface.
- Conjunto de reglas para que dos programas se comuniquen.
- Define qué se puede pedir.
- Define cómo se debe pedir.
- Define qué se responde.

Analogía del restaurante

- Cliente = aplicación que pide.
- Menú = documentación.
- Camarero = API.
- Cocina = backend.
- Plato = respuesta.

Importancia

Integración

- Sistemas distintos pueden trabajar juntos.
- No importa el lenguaje.
- No importa la plataforma.

Innovación

- No construímos todo desde cero.

- Usamos servicios existentes.

Ejemplos reales

- Login con Google.
- Pagos en línea.
- Mapas embebidos.
- Envío de correos automáticos.

Conceptos

Cliente

- Programa que hace la solicitud.
- Puede ser web, móvil o servidor.

Servidor

- Programa que recibe solicitudes.
- Procesa la lógica.
- Devuelve una respuesta.

Solicitud (Request)

- Mensaje que envía el cliente.
- Incluye datos necesarios.

Respuesta (Response)

- Mensaje que envía el servidor.

- Incluye datos o errores.

Backend y API

Backend

- Parte interna del sistema.
- Maneja base de datos.
- Ejecuta reglas de negocio.

Relación

- La API es la puerta de entrada.
- El backend hace el trabajo real.
- La API oculta la complejidad.

Tipos de APIs

Según quién puede usarlas

- Públicas.
- Privadas.
- Internas.
- De socios.

APIs públicas

- Disponibles para desarrolladores externos.
- Ejemplo: API de clima.

APIs privadas

- Uso interno dentro de una empresa.
- No están expuestas al público.

Comunicación

Basadas en HTTP

- Funcionan sobre la web.
- Usan el modelo cliente-servidor.

La idea básica

- El cliente envía una solicitud.
- Esa solicitud incluye una acción.
- Esa acción se expresa con un verbo HTTP.

Verbos HTTP

Definición

- Es la intención de la solicitud.
- Indica qué queremos hacer con los datos.
- No define la arquitectura completa.

GET

- Se usa para obtener información.
- No debería modificar datos.

- Ejemplo: obtener un recurso.

POST

- Se usa para enviar información.
- Generalmente crea un nuevo recurso.
- Ejemplo: crear un usuario.

PUT

- Reemplaza completamente un recurso existente.
- Envía todos los datos actualizados.

PATCH

- Actualiza parcialmente un recurso.
- Solo envía los campos que cambian.

DELETE

- Elimina un recurso.
- Ejemplo: borrar un usuario.

Resumen

| Verbo | Intención principal | | ------ | ---------- | | GET | Obtener datos | | POST | Crear datos | | PUT | Reemplazar datos | | PATCH | Modificar datos | | DELETE | Eliminar datos |

Códigos de respuesta

¿Qué indican?

- Resultado de la solicitud.
- Éxito o error.

Familias principales

| Código | Significado | | ------ | ------------------ | | 2xx | Éxito | | 4xx | Error del cliente | | 5xx | Error del servidor |

Ejemplos importantes

- 200 → Todo bien.
- 404 → No encontrado.
- 500 → Error interno.

Problemas comunes

Falta de documentación

- No sabes qué puedes pedir.
- No sabes qué esperar.

Errores mal manejados

- Respuestas confusas.
- Difícil depurar.

Cambios sin versión

- Rompen aplicaciones existentes.

Buenas prácticas

Documentación clara

- Explicar endpoints.
- Explicar parámetros.
- Explicar respuestas.

Versionado

- Ejemplo: /api/v1/usuarios
- Permite evolucionar sin romper.

Manejo correcto de errores

- Usar códigos adecuados.
- Mensajes claros.

Seguridad

- Autenticación.
- Autorización.
- No exponer datos sensibles.

Herramientas para probar APIs

¿Por qué necesitamos herramientas?

- No siempre hay interfaz gráfica.
- Necesitamos enviar solicitudes manualmente.

Herramientas comunes

- Postman.
- Hoppscotch.

¿Qué permiten hacer?

- Enviar solicitudes.
- Ver respuestas.
- Probar errores.