Integración y Mundo Real

Conectando sistemas y desplegando aplicaciones 🌐

Probabilidad y Estadística

Aprendiendo con diversión 🎓

25 de noviembre de 2025

¿Qué vamos a aprender?

Nota importante

¡Hoy conectamos nuestro código con el mundo real!

Objetivo 1: Entender qué son las APIs y cómo funcionan

Objetivo 2: Conocer los verbos HTTP (GET, POST, PUT, DELETE)

Objetivo 3: Aprender sobre autenticación y seguridad

Objetivo 4: Comprender el ciclo de vida del software

Ejemplo

Como conectar piezas de LEGO: cada API es una pieza que encaja con otras

¿Qué es una API?

Nota importante

Application Programming Interface (Interfaz de Programación)

Es un 'contrato' que define cómo comunicarse con un sistema

Permite que diferentes aplicaciones hablen entre sí

Ejemplo

Como un menú de restaurant: lista lo que puedes pedir y cómo pedirlo

Tu app pide datos → API procesa → API responde

$$Ejemplo : App del clima pide temperatura ß API responde '22C'$$

REST: El Estándar de las APIs

REST = Representational State Transfer

Es un estilo arquitectónico para crear APIs web

Nota importante

Principios de REST:

- Cliente-Servidor separados
- Sin estado (cada petición es independiente)
- URLs representan recursos

Ejemplo

```
/api/usuarios    Lista de usuarios
/api/usuarios/123   Usuario con ID 123
/api/facturas    Lista de facturas
```

Verbos HTTP: Acciones sobre Recursos

Verbo	Acción	Ejemplo
GET	Leer/Consultar	Consultar saldo
POST	Crear	Crear nueva empresa
PUT	Actualizar	Actualizar datos de usuario
DELETE	Eliminar	Borrar factura

Nota importante

Son como verbos en una oración: GET obtiene, POST crea, PUT modifica, DELETE elimina

Ejemplo: Petición GET

Problema

Consultar el saldo de una billetera digital

Ejemplo

```
 Peticin :
GET /api/billeteras/12345/saldo

Respuesta (JSON):
{
  "billetera_id": 12345,
  "propietario": "Juan Prez",
  "saldo": 150000,
  "moneda": "CLP"
}
```

Nota importante

Ejemplo: Petición POST

Problema

Crear una nueva empresa en el sistema

Ejemplo

```
 Peticin :
POST /api/empresas

Body (JSON):
{
  "nombre": "Tech Solutions S.A.",
  "rut": "76.123.456-7",
  "email": "contacto@techsolutions.cl"
}

Respuesta:
{
  "id": 789,
```

Ejemplos: PUT y DELETE

Ejemplo

Actualizar email de un usuario (PUT):

```
PUT /api/usuarios/123
Body: {"email": "nuevo@email.com"}
Respuesta: {"mensaje": "Usuario actualizado"}
```

Ejemplo

Eliminar una factura (DELETE):

```
DELETE /api/facturas/456
Respuesta: {"mensaje": "Factura eliminada"}
```

Nota importante

Autenticación y Seguridad

Nota importante

¡Proteger la billetera digital es crítico!

No cualquiera puede consultar saldos o hacer transferencias

Método	Descripción	Seguridad
Usuario/Contraseña	Login tradicional	★★
Tokens	Código temporal	★★★
JWT	Token con información	★★★★
OAuth	Login con Google/Facebook	★★★★★

Autenticación con Tokens

Nota importante

Como una pulsera de evento: la muestras para entrar 🎟

1 Usuario hace login con usuario/contraseña

2 Servidor valida credenciales

3 Servidor genera un token único

4 Cliente guarda el token

5 Cada petición incluye el token en el header

Ejemplo

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

El token expira después de cierto tiempo ⏱

JWT: JSON Web Token

Token que contiene información codificada

Estructura: Header.Payload.Signature

Ejemplo

```
Payload (decodificado):
{
  "usuario_id": 123,
  "email": "juan@email.com",
  "rol": "admin",
  "exp": 1700000000
}
```

Nota importante

La firma garantiza que el token no fue modificado

Servidor verifica la firma antes de confiar en el token

Consumo de APIs Externas

Problema

Integrar con servicios de impuestos del SII (Chile)

Ejemplo

```
import requests

# Consultar datos de empresa por RUT
rut = "76.123.456-7"
url = f"https://api.sii.cl/empresas/{rut}"
headers = {"Authorization": "Bearer tu_token_aqui"}

response = requests.get(url, headers=headers)

if response.status_code == 200:
    datos = response.json()
    print("Razon Social:", datos["razon_social"])
else:
```

Ejemplo: Integración con Banco

Problema

Procesar pago con tarjeta de crédito

Ejemplo

```
# API del banco procesador
POST /api/pagos

Body:
{
  "monto": 50000,
  "tarjeta": {
    "numero": "4532-****-****-1234",
    "cvv": "***",
    "expiracion": "12/25"
  },
  "descripcion": "Compra en tienda online"
}
```

Ciclo de Vida del Software

Nota importante

El viaje del código desde la idea hasta producción

1. Planificación: Definir qué construir
2. Desarrollo: Escribir el código
3. Testing: Probar que funcione
4. Despliegue: Publicar en internet
5. Mantenimiento: Corregir bugs y agregar features

Ejemplo

Es un ciclo continuo, no un proceso lineal

Control de Versiones: Git

Nota importante

¡Guardar el historial de cambios del código!

Git es como 'Ctrl+Z' infinito para tu proyecto

Comando	Acción
git init	Iniciar repositorio
git add .	Preparar cambios
git commit -m 'mensaje'	Guardar cambios
git push	Subir a servidor
git pull	Descargar cambios

Ejemplo

Cada commit es una 'foto' del proyecto en ese momento

Branching: Trabajar en Paralelo

Ramas (branches) permiten trabajar en features sin afectar el código principal

Ejemplo

```
main (produccin)


feature-login
feature-pagos
```

Nota importante

Desarrollas en tu rama, luego haces merge a main cuando está listo ✔

Evita conflictos cuando varios desarrolladores trabajan juntos

Testing: Asegurar la Calidad

Nota importante

¡Probar antes de enviar a producción!

Pruebas Unitarias: Probar funciones individuales

Ejemplo

```
def test_calcular_iva():
    precio = 100000
    resultado = calcular_iva(precio)
    assert resultado == 19000, "IVA incorrecto"
```

Si el cálculo financiero está mal, ¡puede ser desastroso!

Testaprobado ß *Código confiable*

Tipos de Testing

Tipo	Qué Prueba	Ejemplo
Unitarias	Funciones aisladas	$\text{test}_c alcular_i va()$
Integración	Componentes juntos	API + Base de Datos
E2E	Flujo completo	Login → Compra → Pago
Carga	Rendimiento	10,000 usuarios simultáneos

Nota importante

Más tests = Menos bugs en producción

Despliegue: Publicar el ERP

Nota importante

Poner el sistema en internet para que usuarios accedan

Opción	Descripción	Ejemplo
Servidor Propio	Comprar y mantener	Data center físico
VPS	Servidor virtual	DigitalOcean, Linode
Cloud	Infraestructura escalable	AWS, Azure, GCP
PaaS	Plataforma gestionada	Heroku, Vercel

Ejemplo

Cloud es como Uber: pagas solo lo que usas

CI/CD: Despliegue Continuo

CI = Continuous Integration (Integración Continua)

CD = Continuous Deployment (Despliegue Continuo)

Ejemplo

Flujo automatizado:

```
1. Desarrollador hace push a Git
2. CI ejecuta tests automaticamente
3. Si tests pasan, Build del proyecto
4. CD despliega a servidor
5. Aplicacion actualizada en minutos
```

Nota importante

De código a producción sin intervención manual

Monitoreo y Logs

Una vez en producción, hay que vigilar el sistema

Aspecto	Herramienta	Qué Monitorea
Logs	Elasticsearch	Errores y eventos
Performance	New Relic	Tiempo de respuesta
Uptime	Pingdom	¿Está funcionando?
Errores	Sentry	Crashes y excepciones

Nota importante

Si el ERP cae, necesitas saberlo INMEDIATAMENTE

¡Tu Turno de Practicar!

Problema

Diseñar una API REST para un sistema de biblioteca

Endpoints necesarios:

- GET /api/libros → Listar todos los libros
- GET /api/libros/:id → Obtener un libro específico
- POST /api/libros → Agregar nuevo libro
- PUT /api/libros/:id → Actualizar libro
- DELETE /api/libros/:id → Eliminar libro

Nota importante

Define qué datos envía y recibe cada endpoint en formato JSON

¡Resumen de lo Aprendido!

- APIs permiten que sistemas se comuniquen
- REST usa verbos HTTP: GET, POST, PUT, DELETE
- Autenticación con tokens (JWT) protege recursos
- Consumir APIs externas integra servicios (bancos, impuestos)
- Git guarda historial de cambios del código
- Testing asegura que el código funcione correctamente
- Despliegue pone la aplicación en internet
- CI/CD automatiza el proceso de prueba y despliegue

Nota importante

¡Ahora puedes crear, integrar y desplegar sistemas reales!

El Camino del Programador

Nota importante

¡Has completado el viaje desde cero hasta sistemas profesionales!

- Nivel 1: Entendiste cómo funcionan las computadoras
- Nivel 2: Aprendiste los fundamentos (variables, if, loops)
- Nivel 3: Organizaste datos y creaste funciones
- Nivel 4: Diseñaste arquitecturas escalables
- Nivel 5: Conectaste con el mundo real

Ejemplo

Ahora puedes crear un ERP completo desde cero

Nota importante

¡La programación es una aventura sin fin! Sigue aprendiendo y construyendo

¡Sigue aprendiendo!