PRUEBAS CON APIs

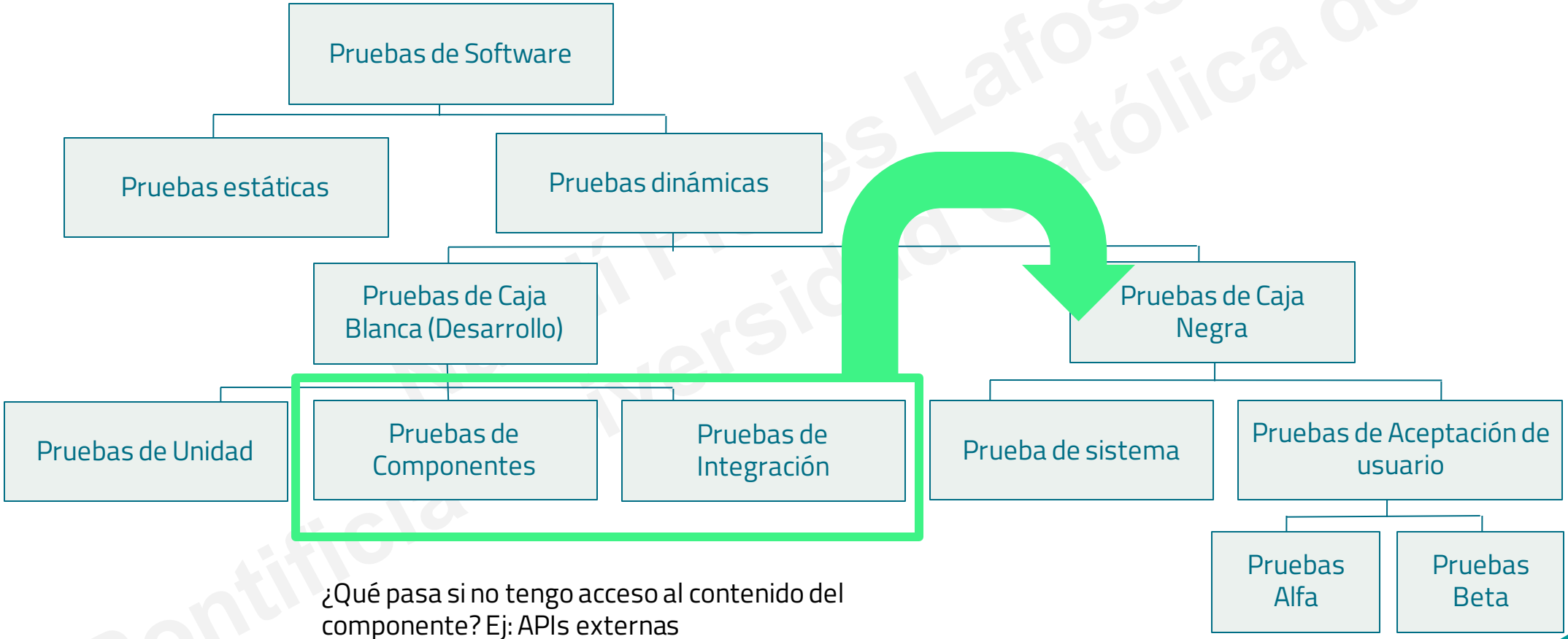
OBJETIVOS

- Identificar la diferencia de necesidad de pruebas con interfaz gráfica y sin ella.
- Establecer estrategias para las pruebas de APIs.
- Conocer tipos de pruebas enfocadas en API y cuándo aplicarlas.

1

REPASO DE PRUEBAS DE COMPONENTES

Árbol de tipos de Pruebas (General)



¿Qué pasa si no tengo acceso al contenido del componente? Ej: APIs externas

2

ENFOQUE Y DESAFÍOS

¿Qué es un API?

- Application Programming Interface
- Grupo de protocolos que permiten comunicar diferentes componentes de software para transferir datos.

Estilos de API más populares

REST (Representational State Transfer):

Los recursos se acceden por un endpoint (url) y se usan los métodos de HTTP como GET, POST, PUT y DELETE. Comúnmente usa JSON.

Ejemplo Stripe:

```
POST /v1/customers                                                cURL

curl https://api.stripe.com/v1/customers \
  -u "sk_test_4eC39Hq...arjtT1zdp7dc:" \
  -d "metadata[order_id]"=6735
```

RESPONSE


```
{
  "object": "search_result",
  "url": "/v1/customers/search",
  "has_more": false,
  "data": [
    {
      "id": "cus_4QFJOjw2pOmAGJ",
      "object": "customer",
      "address": null,
      "balance": 0,
      "created": 1405641735,
      "currency": "usd",
      "default_source": "card_14HOpG2eZvKYlo2Cz4u5AJG5",
      "delinquent": false,
      "description": "someone@example.com for Coderwall",
      "discount": null,
      "email": null,
      "invoice_prefix": "7D11B54",
      "invoice_settings": {
        "custom_fields": null,
        "default_payment_method": null,
        "footer": null,
        "rendering_options": null
      },
      "livemode": false,
      "metadata": {
        "foo": "bar"
      },
      "name": "fakename",
      "next_invoice_sequence": 25,
      "phone": null,
      "preferred_locales": [],
      "shipping": null,
      "tax_exempt": "none",
      "test_clock": null
    },
    {...},
    {...}
  ]
}
```

Estilos de API más populares

- **SOAP (Simple Object Access Protocol)**

XML para transferir mensajes altamente estructurados entre cliente y servidor.

Se usa mucho en sistemas legados, suele ser algo más lento.

Ejemplo NumberToWords:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <NumberToWords xmlns="http://www.dataaccess.com/webservicesserver/">
      <ubiNum>500</ubiNum>
    </NumberToWords>
  </soap:Body>
</soap:Envelope>'
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <NumberToWordsResponse xmlns="http://www.dataaccess.com/webservicesserver/">
      <NumberToWordsResult>string</NumberToWordsResult>
    </NumberToWordsResponse>
  </soap12:Body>
</soap12:Envelope>
```

Diferencia de enfoque al hacer pruebas de APIs

- No se cuenta con interfaz gráfica.
- El usuario no suele tener acceso directo a ellas.
- Son de caja negra (hay documentación).
- Es conveniente usar software específico/automatizado.

Tres consideraciones en Pruebas de API

- Especificación de interfaz y documentación
- Seguridad
- Desempeño

3

ESPECIFICACIÓN Y DOCUMENTACIÓN

Especificación del API

Antes de poder generar nuestros casos de prueba, debemos conocer su especificación y comportamiento.
¿Cómo se comporta la interfaz cuando...

- ...no hay parámetros?
- ...hay un parámetro correcto con valor correcto?
- ...hay un nombre de párametro incorrecto?
- ...hay parámetros sin valor?
- ...hay un valor de parámetro incorrecto?
- ...hay una combinación correcta de parámetros?

Especificación del API

Sobre su resultado:

- ¿Cuál es el formato de respuesta si no especifica un formato?
- ¿Cuál es el formato de respuesta tanto para condiciones de éxito o error?
- ¿Cuál es el código de respuesta tanto para condiciones de éxito o error?
- ¿Cuál es la respuesta del API para métodos, cabeceras, URLs inesperados?

Documentación del API

- La Documentación es la descripción del contrato del API.
- Debería responder (si no todas) la mayoría de las preguntas; pero es necesario asegurarse de ello.
- Verificar la correspondencia entre la versión de la documentacion y la del API.
- Ejemplos de documentaciones:
  - https://docs.stripe.com/api
  - https://docs.github.com/es/rest?apiVersion=2022-11-28

Pruebas de contrato

- Son pruebas para confirmar que la especificación del API corresponde a lo esperado por nuestro sistema.

- Debe Incluir:
  - Validación del código y formato de respuesta: 200 en REST y un JSON válido, por ejemplo
  - Verificación de estructura: Contiene objetos, cabeceras y propiedades URL requeridas
  - Validación de datos enviados: Validar que cumplen lo esperado
  - Consistencia de datos: Verifica que la data enviada correponde a la respuesta

Prueba de contrato - Beneficios

- Colaboración: El proveedor y consumidor entiende la especificación de forma temprana.
- Detección de incompatibilidad temprana: Se identifica los problemas y desviaciones antes de que la integración se haga compleja.
- Fiabilidad aumentada: Permite que proveedor y consumidor cumplan los términos del contrato y así asegurar un comportamiento precible de la interfaz.

Nota: Algunos proveedores ponen a disposición plantillas de contrato para sus consumidores.

Prueba de contrato - Desventajas

- Las Pruebas de contrato no deberían usarse para probar escenarios completos.
  - Por ello mismo, un comportamiento complejo tal vez no sería probado desde el inicio y su complejidad relegada a otro momento.

- Las puebas de contrato no deben usarse para probar ni la seguridad ni el desempeño del API.

Prueba de contrato - Ejemplo

```
Contract.make {
        request {
                description('Get a list of all the attendees at a conference')
                method GET()
                url '/conference/1234/attendees'
                headers {
                        contentType('application/json')
                        }
                }
        response {
                status OK()
                headers {
                        contentType('application/json')
                        }
                body(
                value: [
                        $(
                                id: 123456,
                                givenName: 'James',
                                familyName: 'Gough'
                        ),
                        $(
                                id: 123457,
                                givenName: 'Matthew',
                                familyName: 'Auburn'
                        )
                        ]
                )
                }
        }
```

Pruebas unitarias en APIs

- Sirven para realizar las pruebas de que el endpoint de la API responde a nuestros casos de prueba específicos.
- Ayuda a localizar los casos de error.
- Podemos usar clases de equivalencia, considerando los parámetros del API.

4

SEGURIDAD Y DESEMPEÑO

Pruebas de Autenticación y Autorización

- Necesarias si el recurso es privado o limitado.
- Las APIs pueden usar un juego de llaves mutuas (intercambio de llaves) o tokens usando OAuth/OpenID.
- Las pruebas deben enfocarse en el acceso correcto a sólo la información que nos corresponde.

Otras pruebas (si el API es nuestra)

Si nosotros somos los proveedor del API deberíamos considerar:

- Pruebas para evitar API Fuzzing (Envío de parámetros aleatorios para conseguir información de la estructura de base de datos o sistema de archivos)
- Pruebas para protegerse de injección maliciosa o malformada:
  - Malformada: Estructura dentro de estructura para saturar el sistema al des-anidar.
  - Contenido: Códigos ingresados como texto plano. Revisar que no se recibe: SQL, javascript, shell, XQuery, python, etc.

Pruebas de Desempeño

- Realizar **pruebas en condiciones normales**, para descubrir cuellos de botella.
- Realizar **pruebas con carga artificial**, para obtener métricas de comportamiento: tiempo total, latencia, errores obtenidos, uso de CPU.
- Realizar **pruebas de estrés**, para determinar los puntos de quiebre.
- Realizar **pruebas de remojo (Soak)** manteniendo el sistema activo por largos periodos, para analizar si se generan errores de memoria o inestabilidad.

5

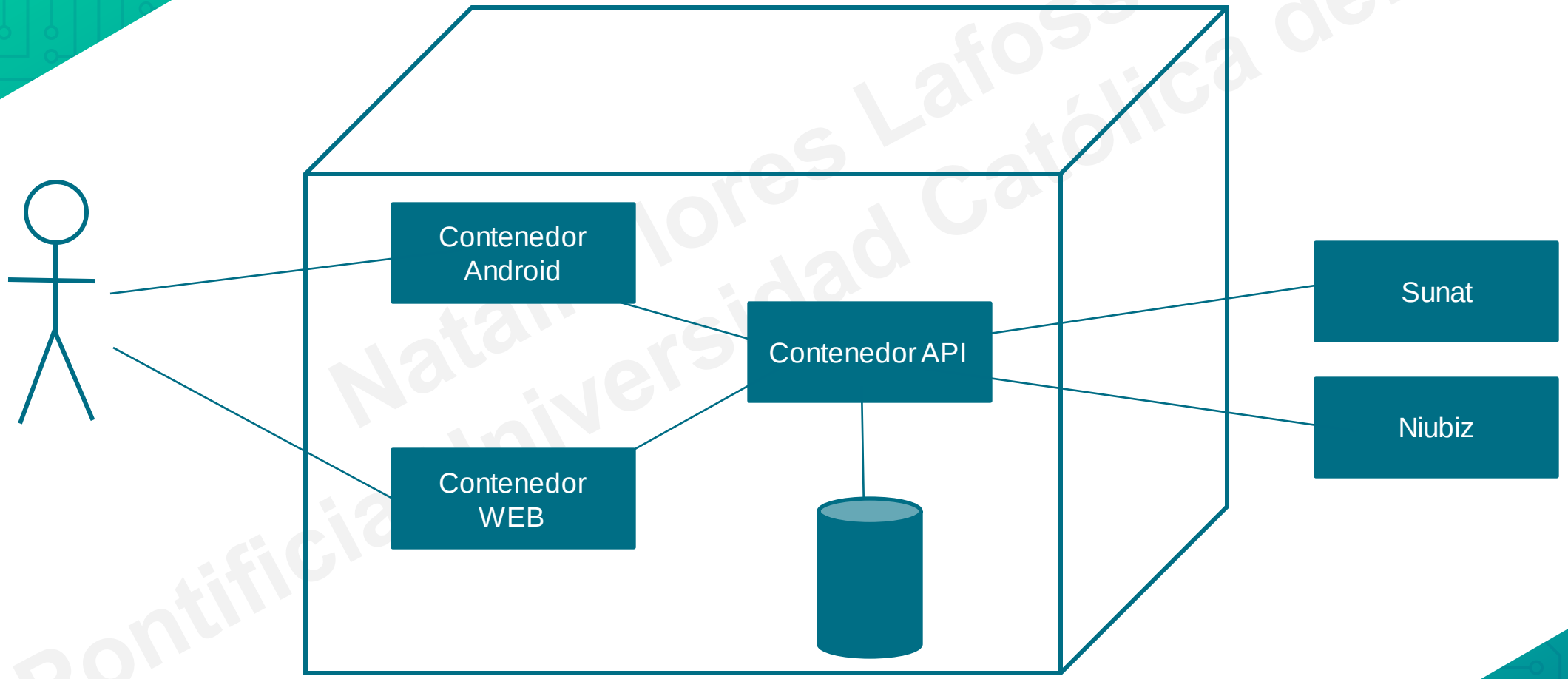
PRUEBAS PUNTO A PUNTO

Unimos todo

Pruebas punto a punto

- Son pruebas de escenarios completos, equivalente a las pruebas de sistema.
- **Es importante ver como los diferentes servicios funcionan en combinación.**
- Es importante analizar si alguna de las pruebas anteriores no consideró parte de alguno de los escenarios.

Prueba punto a punto - Ejemplo



Consideraciones finales

- En cada contexto debe evaluarse qué pruebas deben realizarse para cada API (tanto si somos proveedores o consumidores).

- En muchos casos, serán necesarias herramientas que automaticen las pruebas, ya que no se cuenta con interfaz gráfica.
  - Algunas herramientas (para curiosos):
    - Postman,
    - Pact.io,
    - Jmeter , etc.

6

MINI-CASOS

Aplicando

Mini-caso 01

Maricarmen es parte del equipo que ha desarrollado un sistema que utiliza el API de Niubiz y además se conecta a la SUNAT para realizar la facturación electrónica. Antes de ponerlo en producción, el equipo decidió realizar una prueba automatizada de carga, usando su tarjeta de crédito. Sin embargo, la tarjeta se bloqueó por comportamiento sospechoso.

¿Qué sugerencias podrían darle al equipo de Maricarmen?
¿Cómo habrían planteado las pruebas?

Mini-caso 02

Carlos Alberto ha desarrollado una API para la generación de mapas en un videojuego y está documentando su contrato/especificación.

Un mapa se envía en la siguiente forma JSON

En texto plano:

{"Mapa":"\"Nuevo Oriente\"","version":3.1,"tierra1":[{"arbol1":{"hoja1": 123,"hoja2": 124},"colina":{"flor":{"petalo1": 187,"petalo2": 125,"petalo3": 167,"petalo4": 120}}}]}


```
{
  "Mapa": "\"Nuevo Oriente\"",
  "version": 3.1,
  "tierra1": [
    {
      "arbol1": {
        "hoja1": 123,
        "hoja2": 124
      },
      "colina": {
        "flor": {
          "petalo1": 187,
          "petalo2": 125,
          "petalo3": 167,
          "petalo4": 120
        }
      }
    }
  ]
}
```

Mini-caso 02

Ha pedido a sus compañeros que lancen diferentes llamadas a su endpoint para comprobar su comportamiento.
Una de las llamadas recibidas tiene la siguiente forma (para guardar un nuevo mapa):

{'Mapa':'Nuevo Oriente',version:'3.1',tierra1:[{arbol1:{hoja1:123,hoja2:124},colina:{flor{petalo:187,petalo:125,167}}}]}

Analizar las diferencias con la especificación.

- ¿Tiene la forma adecuada?
- ¿Los parámetros en cadena están encomillados?
- ¿Los caracteres de escape necesarios están incluidos?

Mini-caso 02

Otra llamada recibida tiene la siguiente forma (para guardar un nuevo mapa):

{"Mapa":"'; DROP TABLE maps;
DROP TABLE mapas;","version":3.1,"tierra1":[{"arbol1":{"hoja1": 123,"hoja2":
124},"colina":{"flor":{"petalo1": 187,"petalo2": 125,"petalo3": 167,"petalo4": 120}}}]}

- ¿Qué está queriendo hacer esta llamada?
- ¿Cómo evitarlo?

7

REFERENCIAS

BIBLIOGRAFÍA

- De, Brajesh. "API Management" p153-165. Berkeley, CA: Apress, 2017.
- Gough, James; Bryant, Daniel y Auburn, Matthew "Mastering API Architecture" p27-52. Sebastopol, CA: O'Reilly, 2023
- Postman "Api Plataform" https://www.postman.com/api-platform/ 2024

Créditos:

- Plantilla de la presentación por SlidesCarnival
- Diseño del fondo Hero Patterns