**Section 1: Introduction**

1. **Introducción**

* Se tienen tres capas: Front end, Bussiness logic and Back end
* Front end: generalmente son hechas en HTML, JavaScript, CSS.. se pueden testear con Selenium por ejemplo
* Business Logic: Generalmente hechas Java, C#, python , C++
* Back end: generalmente hechas en MySQL, Oracle server.
* The front presenta la información al usuario por eso se le puede llamar Capa de presentación Presentation layer
* The Back end se puede llamar Data layer
* In the mittle, the business layer se puede llamar Aplication layer
* El testing de API’s se basa en la capa de aplicación. Para este testing hay varias aplicaciones como Postman, Soap UI…
* El front end se le puede llamar como el cliente
* El Back y el busines se puede tener como el server ya que no podemos verlo. Solo interactuar con el por medio del Front end
* API’s Applications programming interface:

Generalmente esta en Java archive (JAR) las cuales tienen diferentes clases que están unidas. Esto es implementado por los developers.

Basado en las request la API va a responder

Las APIS pueden ser testeadas con diferentes inputs. Si se quiere testear el JAR no es necesario por ejemplo usar un browser.

Una API no es un web service, pero un webservice puede ser una API

|  |  |
| --- | --- |
| **API’s** | **Webservice** |
| No se necesita un URL | Formato XML |
| No se necesita un browser aunque se podría usar | Pegarle al URL ya que está en un webserver. Se necesita una URL para acceder a ellas |

* Hay 2 tipos de webservices

SOAP: Simple Object Access protocol 🡪 es más Viejo, soporta solo XML

REST: Representational state transfer 🡪 es más nuevo y mas popular. Soporta JSON, XML, HMTL

1. **Overview on Webservices and API Testing:**

* WSDL 🡪 Web services description language.
* UDDI 🡪 Universal Description discovery and integration
* Service providers van a tener todos los servicios en un solo lugar, este lugar es el UDDI.
* Los consumidores de los servicios los consumen usando WSDL.
* El cliente manda por el WSDL y recibe la respuesta del UDDI.
* EL UDDI es donde el consumidor puede encontrar el servicio. Todos los proveedores de servicios mantendrán los servicios allí.
* **Soap & Restful services:**
* Soap (simple object access protocol): Se basa en XML, manda el request en XML y lo recibe también en XML.

Es especialmente diseñado para aplicaciones web

* **REST:** suportan todo tipo de aplicaciones, web, de escritorio, mobiles, por esto son mas populares.

Soporta XML, JSON, HTTP

Es más rápido y soporta más tipos de aplicaciones

* En HTTP toda la información enviada a través de la red va sin encriptar, lo que puede causar que alguien vea lo que se está transmitiendo como los usuarios y contraseñas
* En HTTPS toda la información está encriptada lo que hace más difícil que alguien pueda ver lo que se envía, solo el transmisor y el receptor pueden ver y descifrar lo que se envía.
* Todos los servicios REST usan HTTP para la comunicación
* Para los SOAP se usa una request XML
* **Herramientas para el testing de estos servicios:**

Recordemos que no todos las API´s son webservices pero todos los webservices son Api’s

Para las Api´s no es necesario tener internet porque se pueden probar localmente en el Jar file.

Webservices --> requieren internet, se deben testear enviando una request HTTP (URL)

**Soap UI** – Soporta ambos protocolos Soap y Rest. Inicialmente no tenían REST pero ahora si se puede probar ambos. Se puede correr cierta automatización

**Postman** – Se puede hacer manual y automation testing. Para testear todo tipo de Apis, webservices, etc

**Katalon**  - Soporta web applications e internamente corre con Selenium. Se puede hacer varios tipos de testing

**Jmeter** – Especial para performance Testing

**Rest assuerd** – Se usa para hacer testing automatizado, se descargan unos paqueres de llava y se corre para poder realizar el testing de Apis y Webservices

**Assertible, Apigee, Tricentis, Karate** – son otras herramientas para el en testing de APis y webservices que se pueden usar, pero no son tan conocidas y usadas.

Algunos webrowser proveen herramientas para probar Apis

**Api’s Testing Process:**

1. Revisión de las especificaciones de la Api: que hacen, como es el proceso interno, lenguaje etc.
2. Identificar como se accede, identificar los parámetros, respuestas que se van a recibir, etc.
3. Escribir los casos de prueba, especificar el framework, escribir los scripts de automatización.
4. Ejecutar los casos de prueba manual/automatizado
5. Reportar los defectos al desarrollador

**Rest Services:**

* Hay 4 tipos de operaciones que normalmente se usan como usuario, el usuario puede, insertar datos, actualizar datos, borrar datos o seleccionar los datos.

Insert = Post

Update = PUT

Delete = Delete

Select = Get

* Esto se puede llevar a una serie de operaciones llamadas CRUD, esto se usa en las operaciones de Base de datos

Create 🡪 Insert = Post

Read 🡪 Select = Get

Update 🡪 Update = PUT

Delete 🡪 Delete = Delete

* **Domain:** El servidor físico donde el sitio web está alojado
* **URI:** El identificador que mapea los archivos en el servidor
* **Query String:** Parte del GET que pasa ciertos valores para personalizar el output

Escala de tiempo

Descripción generada automáticamente

Para crear una request HTTP se necesita:

1. URL
2. Headers
3. Payload - Puede ser en JSON, XML, HTTP

La Respuesta HTTP tiene:

1. Status code - 200 ok
2. Payload de respuesta
3. Mensaje string de exito

<https://reqres.in/>

Pagina donde hay muchas request para probar y ver como funcionan.

Hay varios estatus 200 para el status

200 🡪 Get successfull

201🡪 Post successfull

204🡪 Delete successfull