Consumo de API REST

API REST (Parte I)

{desafío} latam_

Construir una aplicación Android que consume un servicio REST actualizando la interfaz de usuario, acorde al lenguaje Kotlin y a la librería Retrofit

- Unidad 1:
  Acceso a datos en Android
- Unidad 2:
  Consumo de API REST
- Unidad 3:
  Testing
- Unidad 4:
  Distribución del aplicativo Android

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Distinción cliente/servidor*
- *JSON*
- *HTTP Requests*
- *Verbos HTTP*
- *Partes de un request HTTP*
- *Códigos de respuesta*

¿Cómo crees que se realiza la comunicación entre una App en Android y un servicio en Internet?

{desafío} latam_

/* Distinción Cliente/Servidor */

Cliente/Servidor

- Un **cliente** es una computadora o programa que solicita servicios o recursos de un servidor. El cliente realiza una solicitud al servidor y el servidor responde con la información o el servicio solicitado. Por ejemplo, un navegador web es un cliente que solicita páginas web de un servidor web y el servidor web responde enviando las páginas web solicitadas al cliente.

- Un **servidor**, por otro lado, es una computadora o programa que proporciona servicios o recursos a los clientes. El servidor recibe las solicitudes de los clientes y responde con la información o el servicio adecuado. Por ejemplo, un servidor web es un servidor que proporciona páginas web a los navegadores web.

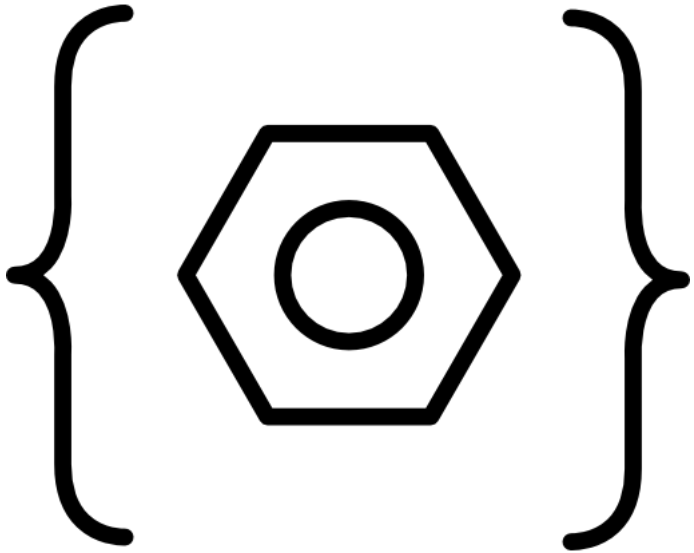
Cliente/Servidor

En una arquitectura cliente-servidor, el cliente y el servidor se comunican entre sí mediante un protocolo específico, como **HTTP** o **FTP**. El cliente envía una solicitud al servidor usando el protocolo apropiado y el servidor responde con la información o el servicio apropiado. Esto permite una separación de preocupaciones entre el cliente y el servidor y permite escalabilidad y flexibilidad en el sistema.

Es importante tener en cuenta que esta definición se basa en una comprensión general de los términos y, en algunos casos, la diferencia entre cliente y servidor puede ser menos clara, por ejemplo, en una red peer-to-peer, cada nodo puede actuar como cliente y servidor.

/* JSON */

JSON



- JSON (Notación de objetos de JavaScript) es un formato de intercambio de datos liviano que es fácil de leer y escribir para los humanos y fácil de analizar y generar para las máquinas. Es un formato estándar abierto que utiliza texto legible por humanos para transmitir objetos de datos que consisten en pares de atributo-valor.

- Un objeto JSON es una colección de pares clave-valor, donde cada clave es una cadena y cada valor puede ser un número, una cadena, un booleano, un valor nulo, una matriz u otro objeto JSON. Los objetos JSON están encerrados entre llaves { } y separados por comas.

JSON - ejemplo

Aquí hay un ejemplo de un objeto JSON:

```
{
    "name": "John Smith",
    "age": 30,
    "address": {
        "street": "123 Main St",
        "city": "Anytown",
        "state": "CA",
        "zip": "12345"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "555-555-5555"
        },
        {
            "type": "work",
            "number": "555-555-5555"
        }
    ]
}
```

JSON - ejemplo

JSON también se puede utilizar para representar una matriz de objetos; en este caso, la matriz se incluye entre corchetes [ ] y se separa con comas.

```
[
  { "name": "John Smith", "age": 30, "address": "123 Main St" },
  { "name": "Jane Doe", "age": 25, "address": "456 Park Ave" }
]
```

JSON se usa ampliamente en servicios web, API e intercambio de datos entre sistemas porque es liviano, fácil de leer y escribir, y las computadoras le pueden analizar fácilmente. Muchos lenguajes de programación proporcionan bibliotecas para analizar y generar JSON, lo que lo convierte en una opción popular para el intercambio de datos.

/* HTTP Requests */

HTTP Requests

HTTP (Protocolo de transferencia de hipertexto) es un protocolo utilizado para transmitir datos a través de Internet. Es la base de la World Wide Web y se utiliza para la comunicación entre navegadores web y servidores web.

Una solicitud HTTP es un mensaje enviado por un cliente (como un navegador web) a un servidor (como un servidor web) para solicitar un recurso o servicio específico. La solicitud contiene varios componentes, que incluyen:

A. **Un método:** la acción que el cliente quiere realizar. Los métodos más utilizados son `GET`, `POST`, `PUT` y `DELETE`.

B. **Un URI:** la ubicación del recurso o servicio al que el cliente quiere acceder.

C. **Encabezados:** información adicional sobre la solicitud, como el tipo de contenido que se envía, el idioma preferido y las credenciales de autenticación.

D. **El cuerpo de un mensaje:** datos que el cliente envía al servidor, como el envío de un formulario o una carga JSON.

Este es un ejemplo de una solicitud HTTP GET:


```
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Language: en-US
User-Agent: Mozilla/5.0
```

La primera línea de la solicitud se denomina línea de solicitud y contiene el método (`GET`), el URI (`index.html`) y la versión de HTTP que se utiliza (`HTTP/1.1`). Las líneas restantes son encabezados que brindan información adicional sobre la solicitud.

Este es un ejemplo de una solicitud HTTP GET:

Cuando el servidor recibe una solicitud HTTP, la procesa y envía una respuesta HTTP al cliente, la respuesta contiene un código de estado, encabezados y un cuerpo de mensaje. El código de estado indica el resultado de la solicitud, como éxito (`200 OK`) o falla (`404 No encontrado`).

Las solicitudes y respuestas HTTP generalmente se envían a través de conexiones TCP/IP y no tienen estado, lo que significa que el servidor no retiene ninguna información sobre el cliente entre solicitudes. Esto permite escalabilidad y flexibilidad en el sistema, pero también significa que se pueden necesitar mecanismos adicionales, como cookies y sesiones, para mantener interacciones con el estado de la conexión.

/* Verbos HTTP */

Verbos HTTP - (HTTP Methods)

Los verbos HTTP, también conocidos como métodos HTTP, son los comandos que un cliente envía a un servidor para solicitar una acción específica. Los verbos HTTP más utilizados son:

- **GET:** solicita que el servidor devuelva la representación actual de un recurso específico. Este es el verbo más común utilizado para recuperar datos de un servidor.
- **POST:** solicita que el servidor acepte la representación adjunta y la almacene como un nuevo recurso. Este verbo se usa para enviar datos al servidor, como cuando se completa un formulario o se carga un archivo.
- **PUT:** solicita que el servidor acepte la representación adjunta y la almacene como el recurso especificado. Este verbo es similar a POST, pero normalmente se usa para actualizar un recurso existente.
- **DELETE:** solicita que el servidor elimine el recurso especificado.
- **HEAD:** igual que GET pero devuelve solo los encabezados y no el cuerpo de la respuesta.
- **OPTIONS:** solicita que el servidor devuelva los métodos HTTP que admite.
- **PATCH:** solicita que el servidor aplique un conjunto de modificaciones al recurso especificado.

Verbos HTTP

Todos estos verbos son parte de la especificación HTTP/1.1, y algunos otros verbos como CONNECT, TRACE y otros que existen, pero no se usan mucho. Estos verbos indican la acción que se desea realizar en el recurso identificado. Mediante el uso del verbo apropiado, los clientes pueden crear, leer, actualizar y eliminar recursos en el servidor.

Es importante tener en cuenta que no todos los servidores admiten todos los verbos y que diferentes verbos pueden tener diferentes comportamientos o restricciones según el servidor.

/* Partes de un request HTTP */

Partes de un request HTTP

Una solicitud HTTP es un mensaje enviado por un cliente a un servidor para solicitar un recurso o servicio específico. Consta de varias partes:

1. Línea de solicitud: primera línea de la solicitud que contiene el método, el URI y la versión de HTTP que se está utilizando.

2. Encabezados: información adicional sobre la solicitud, como el tipo de contenido que se envía, el idioma preferido y las credenciales de autenticación.

3. Cuerpo del mensaje: datos que el cliente envía al servidor, como el envío de un formulario o una carga JSON. No todas las solicitudes tienen un cuerpo de mensaje.

Partes de un request HTTP

Utilicemos el ejemplo anteriormente mostrado:

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Language: en-US
User-Agent: Mozilla/5.0
```

En este ejemplo, la línea de solicitud es "`GET /index.html HTTP/1.1`", los encabezados son "`Host: www.example.com`", "`Accept-Language: en-US`" y "`User-Agent: Mozilla/5.0`" y no hay cuerpo de mensaje en esta solicitud.

Partes de un request HTTP

La línea de Solicitud se compone de:

- **Método**: La acción que el cliente quiere realizar. Los métodos más utilizados son `GET`, `POST`, `PUT` y `DELETE`.
- **URI**: La ubicación del recurso o servicio al que el cliente quiere acceder.
- **Versión HTTP**: la versión del protocolo que se está utilizando. `HTTP/1.1` es la versión más común en uso.
- **Los encabezados o Headers**: proporcionan información adicional sobre la solicitud. Son pares clave-valor separados por dos puntos. Los encabezados pueden incluir información como el tipo de contenido, el agente de usuario, el host, la codificación y más.

El cuerpo del mensaje se usa para enviar datos al servidor, puede contener datos como parámetros de formulario, carga útil JSON y más. El cuerpo del mensaje no está presente en todas las solicitudes, por ejemplo, la solicitud GET no tiene un cuerpo de mensaje.

/* Códigos de respuesta */

Códigos de respuesta

Los códigos de respuesta HTTP, también conocidos como códigos de estado HTTP, son números de tres dígitos que devuelve un servidor en respuesta a una solicitud HTTP. Estos códigos indican el resultado de la solicitud y brindan información adicional sobre el estado del recurso solicitado.

El primer dígito del código de estado define la clase de respuesta. Las clases más comunes son:

- **1xx (Informativo)**: Se recibió la solicitud, proceso continuo.
- **2xx (Successful)**: la solicitud se recibió, comprendió y aceptó correctamente.
- **3xx (Redireccionamiento)**: la solicitud necesita más acciones para completarse, como redirigir al cliente a una ubicación diferente.
- **4xx (Error del cliente)**: la solicitud contiene una sintaxis incorrecta o el servidor no puede cumplirla.
- **5xx (Error del servidor)**: el servidor no pudo cumplir con una solicitud válida.

Códigos de respuesta

Algunos de los códigos de respuesta HTTP más comunes son:

- **200** OK: La solicitud fue exitosa y el servidor ha devuelto la información solicitada.
- **201** Creado: La solicitud fue exitosa y el servidor ha creado un nuevo recurso.
- **204** Sin contenido: la solicitud fue exitosa pero no hay información para devolver.
- **400** Solicitud incorrecta: la solicitud contiene una sintaxis incorrecta o el servidor no puede cumplirla.
- **401** No autorizado: la solicitud requiere autenticación.
- **403** Prohibido: El servidor entiende la solicitud pero se niega a autorizarla.
- **404** No encontrado: No se pudo encontrar el recurso solicitado.
- **500** Error interno del servidor: se produjo un error en el servidor y no se pudo completar la solicitud.

Estos códigos son solo algunos ejemplos de los muchos códigos de respuesta HTTP que existen. Cada código representa un significado específico y es importante manejarlos de la manera adecuada.

{desafío} latam_

Es importante tener en cuenta que estos códigos son un estándar y no todos los servidores devolverán los mismos códigos para los mismos errores, además, se pueden agregar nuevos códigos a medida que evoluciona el estándar.

¿Cuáles son los diferentes tipos de solicitudes HTTP y cuál es el propósito de cada una?

Próxima sesión...

- *REST, RESTful, RESTless*
- *Estándares REST para clientes*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales