FIAP GRADUAÇÃO

SISTEMAS DE INFORMAÇÃO

Desenvolvimento Mobile, Games e Internet das Coisas

PROF. Kassiano Resende
profkassiano.resende@fiap.com.br

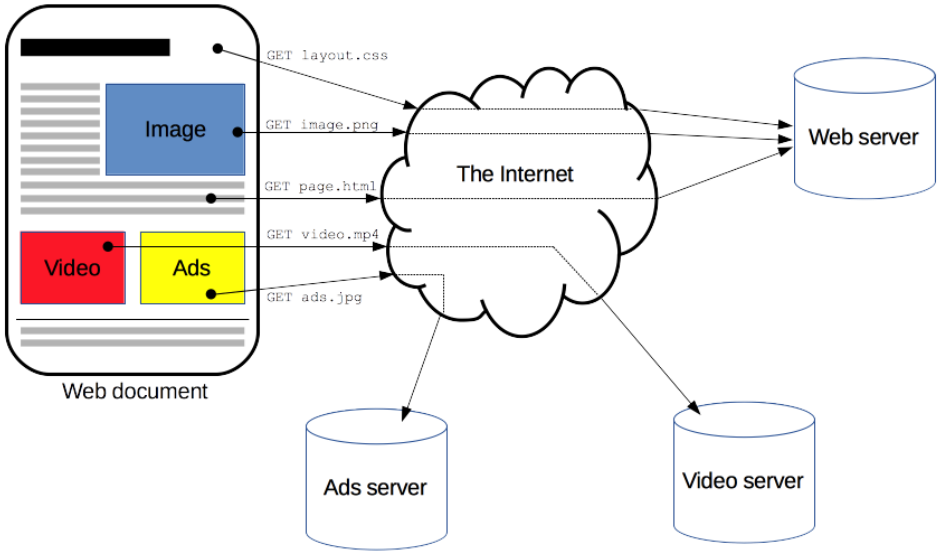
HTTP com Retrofit

- Protocolo HTTP
- Verbos HTTP
- Serviços REST
- JSON
- Retrofit

Protocolo HTTP

Protocolo HTTP

HTTP é um protocolo que permite a obtenção de recursos, tais como documentos HTML. É a base de qualquer troca de dados na Web e uma arquitetura cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web.
Um documento completo é reconstruído a partir dos diferentes sub-documentos obtidos, como por exemplo texto, descrição do layout, imagens, vídeos, scripts e muito mais.



Verbos HTTP

O protocolo HTTP possui alguns verbos que indicam ao servidor o desejo do cliente de alguma ação seja executada. O protocolo possui diversos verbos, vou abordar os 4 principais **GET, POST, PUT, DELETE**

GET

O verbo GET é utilizado para obter um recurso do servidor, por ex: quando acessamos uma página de internet atráves do browser estamos fazendo uma chamada GET daquela página e o servidor nos devolve o HTML dela.

Verbos HTTP

POST

O verbo POST é utilizado para enviar informações ao servidor, geralmente enviar uma entidade que causará uma mudança de estado do lado do servidor.

Ex: Preencher um formulário de cadastro, ao submeter o formulário será feita uma chamada POST para o servidor enviando os dados daquele formulário.

Verbos HTTP

PUT

O verbo PUT é utilizado para criar ou atualizar um recurso no servidor.

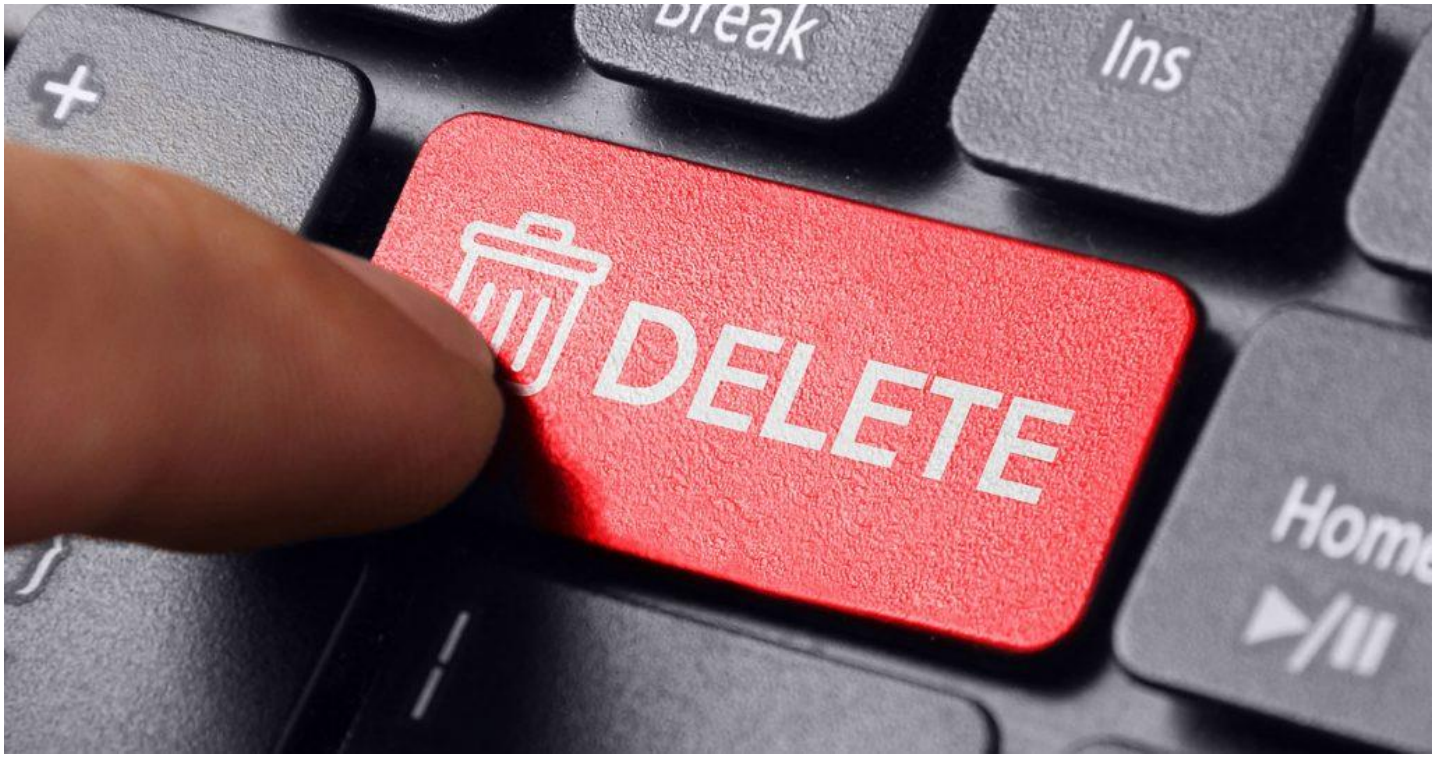
Seu uso é muito parecido com o POST, poderíamos utilizar o PUT em um formulário também para inserir um novo registro ou atualizar.

Com a diferença que o método PUT é idempotente. Portanto, se você enviar uma nova tentativa de solicitação várias vezes, isso deve ser equivalente a uma única modificação de solicitação.

Verbos HTTP

DELETE

O verbo DELETE é utilizado para deletar um recurso do servidor.



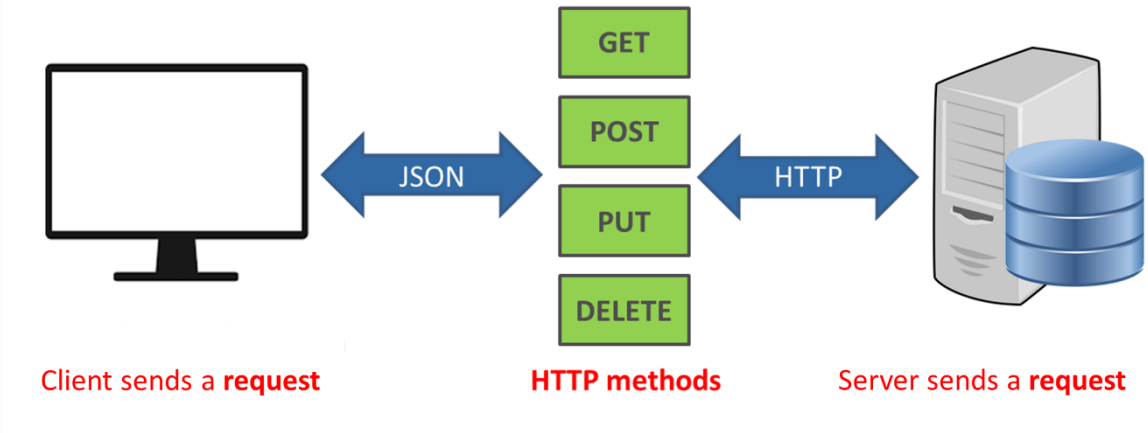
Serviço REST

Serviço REST

Um serviço REST é basicamente um servidor que irá escutar os verbos: GET, POST, PUT e DELETE.

A palavra REST vem de: *Representational State Transfer*

A ideia é criar um serviço com regras definidas que proverá dados para outras aplicações.



Arquitetura de Rotas (EndPoints)

Uma característica de uma API REST é o padrão de Rotas. Por exemplo. Vamos supor que temos a seguinte **URL Base** para nosso serviço: **http://exemplo.rest.com**

E queremos expor métodos de acesso a uma entidade ***user.*** *As rotas seriam:*

GET – Retorna todos usuários
*{baseUrl}**/users***

GET – Retorna um único usuário por ID
*{baseUrl}**/user/{id}***

PUT – Atualiza um usuário por ID
*{baseUrl}**/user/{id}***

DELETE – Deleta um usuário por ID
*{baseUrl}**/user/{id}***

Formato JSON

JSON (JavaScript Object Notation) é um formato de dados leve, fácil para humanos ler e escrever. É fácil para as máquinas analisar e gerar.

JSON -> Kotlin

```kotlin
data class User(
    val nome: String,
    val idade: Int,

    @SerializedName("frutas_preferidas")
    val frutasPreferidas: List<String>
)
```

Retrofit

Retrofit

É a lib cliente HTTP mais utilizada no Android.

- Facilita integrações com serviços REST
- Facilita testes unitários
- Possui uma arquitetura robusta
- Funciona com Java e Kotlin

https://square.github.io/retrofit/

Configurar dependências

```
//Retrofit
implementation 'com.squareup.retrofit2:retrofit:2.7.1'
implementation 'com.squareup.retrofit2:converter-gson:2.7.1’


//Viewmodel injection
implementation "androidx.activity:activity-ktx:1.1.0"

//coroutines
implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.3.0-
alpha06"
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.0-
alpha06"
```

Configurando a interface de serviço

```
interface Servico {

    @GET("users")
    suspend fun obterTodosUsuarios() : List<User>

    @GET("user/{id}")
    suspend fun obterUsuarioPorId(@Path("id") userId: Int) : User

    @PUT("user/{id}")
    suspend fun atualizarUsuario(@Path("id") userId: Int)

    @DELETE("user/{id}")
    suspend fun deletarUsuario(@Path("id") userId: Int)

}
```

Instanciar o serviço

```
val retrofit = Retrofit.Builder()
    .baseUrl("http://exemplo.rest.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build()

val service: Servico = retrofit.create(Servico::class.java)
```

Caso de Uso

```
class UserViewModel : ViewModel() {


    val usersLiveData = MutableLiveData<List<User>>()

    fun fetchUsers() {
        viewModelScope.launch(Dispatchers.IO) {
            val result = service.obterTodosUsuarios()
            usersLiveData.postValue(result)
        }
    }

}
```

CodeLab

THE WAY WE ARE