API Rate Limiting

Handling HTTP 429 with OkHttp interceptors

API Rate Limiting

Public APIs cap requests per time window to ensure fair access. Exceeding the limit returns HTTP 429.

The Problem
Too many requests too fast and the server rejects you with HTTP status 429 (Too Many Requests)

Common Limits
USGS Water Data
50 req/hr (anonymous)

Reddit
10 req/min (no auth)

The Solution
OkHttp Interceptor detects 429, waits, and retries — transparent to your app code and ViewModel

Handle rate limiting in the network layer with an interceptor, not scattered across every API call site.

OkHttp Interceptor — Retry on 429

An interceptor inspects every response in the HTTP pipeline.
Wire it in with OkHttpClient.Builder.addInterceptor().

FloodWatch — intercept()

```
override fun intercept(
    chain: Interceptor.Chain
): Response {
    val response =
        chain.proceed(chain.request())
    readRateLimit(response)
    if (response.code == 429) {
        response.close()
        return countdownAndRetry(chain)
    }
    return response
}
```

FloodWatch — countdownAndRetry()

```
private fun countdownAndRetry(
    chain: Interceptor.Chain
): Response {
    for (i in retryDelaySec downTo 1) {
        retryStatus.postValue(
          "Rate limited — retry ${i}s…")
        Thread.sleep(1000)
    }
    retryStatus.postValue(null)
    return chain.proceed(chain.request())
}
```

close() the old response before retrying — OkHttp allows one active response.
Thread.sleep runs on an OkHttp IO thread, not the main thread.

Reading Rate-Limit Headers

Many APIs include rate-limit metadata in response headers.
Use the server's reset time instead of guessing a fixed delay.

Reddit — intercept()

```
override fun intercept(
    chain: Interceptor.Chain): Response {
    val response = chain.proceed(chain.request())
    readRateLimit(response)
    if (response.code == 429) {
        val resetSec = response
            .header("X-Ratelimit-Reset")
            ?.toFloatOrNull()?.toInt() ?: 15
        response.close()
        return countdownAndRetry(chain, resetSec)
    }
    return response
}
```

Reddit — readRateLimit()

```
val rateLimitRemaining =
    MutableLiveData<Int?>()

private fun readRateLimit(
    response: Response) {
    response.header("X-Ratelimit-Remaining")
        ?.toFloatOrNull()?.toInt()?.let {
            rateLimitRemaining.postValue(it)
        }
}
```

The server tells you exactly when to retry via headers.
Expose the remaining quota via LiveData to keep the UI informed.