

In one line: URLSession moves bytes; **you** judge them: it throws only on **transport** failure — a 404/500 arrives as success, so check the **status**, then **decode**. Three layers, three error types, and UI back on the **main** thread.

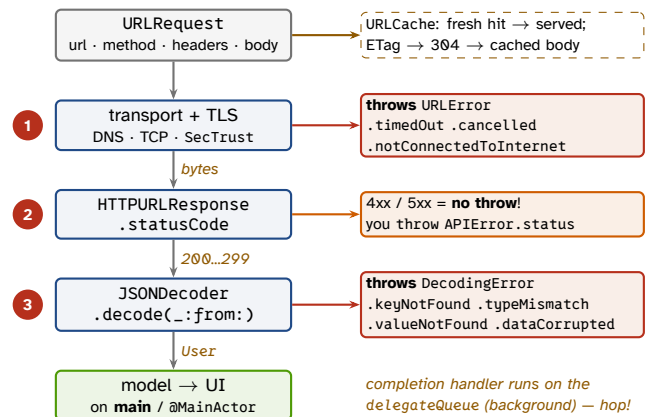
How it works

- **URLRequest:** url, httpMethod, setValue(_:forHTTPHeaderField:), httpBody, timeoutInterval, cachePolicy.
- **Session:** URLSession.shared (no delegate) or URLSession(configuration:delegate:delegateQueue:). Configs: .default (disk cache, cookies) · .ephemeral (memory only) · .background(withIdentifier:) (system daemon; download/upload only, no dataTask).
- **Tasks:** dataTask · downloadTask (to a file) · uploadTask. Born *suspended* — nothing happens until .resume().
- **async** (iOS 15): try await session.data(for: req) / data(from: url) → (Data, URLResponse); cancel = cancel the Task.
- **Threads:** the completion handler runs on the session's delegateQueue — a *background serial* OperationQueue, not main. UI: DispatchQueue.main.async {...} or @MainActor.
- **Three error layers**, in order: **1 transport** NSError; **2 HTTP** cast to HTTPURLResponse, require 200 ... 299; **3 decode** DecodingError (carries context.codingPath — log it).
- **Codable keys:** enum CodingKeys: String, CodingKey { case fullName = "full_name" } or keyDecodingStrategy = .convertFromSnakeCase. Pick **one**. Dates: default .deferredToDate (seconds since 2001!) — set .iso8601.
- **TLS:** ATS demands HTTPS; URLSession validates chain + hostname *by default* — you only add code to be *stricter* (pinning, below the picture). No-code pinning: NSPinnedDomains under NSAppTransportSecurity in Info.plist.
- **Caching:** URLLCache.shared (memory + disk) obeys Cache-Control; ETag → If-None-Match → **304** is transparent (you get the cached body). cachePolicy: .useProtocolCachePolicy (default) · .reloadIgnoringLocalCacheData · .returnCacheDataElseLoad (stale OK) · .returnCacheDataDontLoad (offline). *ElseLoad ignores freshness — serves stale data.*

Example

```
func load(_ url: URL) async throws -> User {
    var req = URLRequest(url: url, timeoutInterval: 15)
    req.setValue("application/json",
        forHTTPHeaderField: "Accept")
    // 1 transport: throws NSError (offline, TLS...)
    let (data, resp) = try await URLSession.shared
        .data(for: req)
    // 2 HTTP: 404 / 500 do NOT throw -- check!
    guard let http = resp as? HTTPURLResponse,
        (200...299).contains(http.statusCode)
    else { throw APIError.badStatus(resp) }
    // 3 decode: throws DecodingError
    return try decoder.decode(User.self, from: data)
}
```

Picture — one request, three gates



Pinning at gate 1 (URLSessionDelegate): (a)

urlSession(_:didReceive:completionHandler:) gets a URLAuthenticationChallenge; (b) if protectionSpace.authenticationMethod == NSURLAuthenticationMethodServerTrust, take protectionSpace.serverTrust (a SecTrust); (c) SecTrustEvaluateWithError (keep default validation), then SHA-256 of the server's public key ∈ your pins? (d) yes → .useCredential, URLCredential(trust:); no → .cancelAuthenticationChallenge; other methods → .performDefaultHandling.

Interview traps

- ! **"A 500 lands in catch."** No — its HTML body then fails at layer 3 as a misleading DecodingError. Check status *before* decoding.
- ! **Completion handler on main?** No — background delegateQueue; UIKit there = Main Thread Checker hit.
- ! **Both key mechanisms:** with .convertFromSnakeCase the JSON key becomes `fullName` *before* matching, so a raw value "full_name" never matches. Acronyms: `user_id` → `userId`, never `userID`.
- ! CodingKeys is all-or-nothing: an omitted property needs a default value. A default does *not* make a missing JSON key OK (keyNotFound); use an optional or decodeIfPresent.
- ! Pinning needs *your own* session with a delegate — URLSession.shared has none. Pin the **public key** (survives renewal) and ship ≥ 2 pins (current + backup).
- ! Forgot task.resume() — nothing happens, no error. One session per request = no connection reuse + leaked delegate (session retains it until invalidateAndCancel).

Remember

"Wire, Status, Shape." Did bytes arrive (URLError)? Were they good (statusCode)? Do they fit (DecodingError)? Then **Main**.

Likely questions

1. Throws on 404? — No; check HTTPURLResponse.statusCode.
2. Completion thread? — delegateQueue (background); hop to main.
3. Three error layers? — NSError · HTTP status · DecodingError.
4. How do you pin? — delegate didReceive challenge → SecTrust → key hash.
5. snake_case JSON? — .convertFromSnakeCase or CodingKeys, not both.