NETWORKING

Gleb Arkhipov, iOS Platform Lead at STRV

STRV

“How did the investor know Apple’s stock was going to go up?“

He had incider information.

ASYNCHRONICITY

01

DELEGATES

```
protocol CharacterUpdateDelegate {
    func didUpdate(characters: [Character])
}

extension CharactersListStore: CharacterUpdateDelegate {
    func didUpdate(characters: [Character]) {
        self.characters = characters
    }
}
```

CLOSURES

```
extension CharactersListStore {
    func fetch() {
        getCharacters(completion: { characters in
            self.characters = characters
        })
    }

    func getCharacters(completion: @escaping ([Character]) -> Void) {
        completion([Character.mock])
    }
}
```

COMBINE

```
extension CharactersListStore {
    func fetch() {
        getCharacters().sink { characters in
            self.characters = characters
        }
        .store(in: &cancellable)
    }

    func getCharacters() -> AnyPublisher<[Character], Never> {
        Just([Character.mock]).eraseToAnyPublisher()
    }
}
```

ASYNC/AWAIT

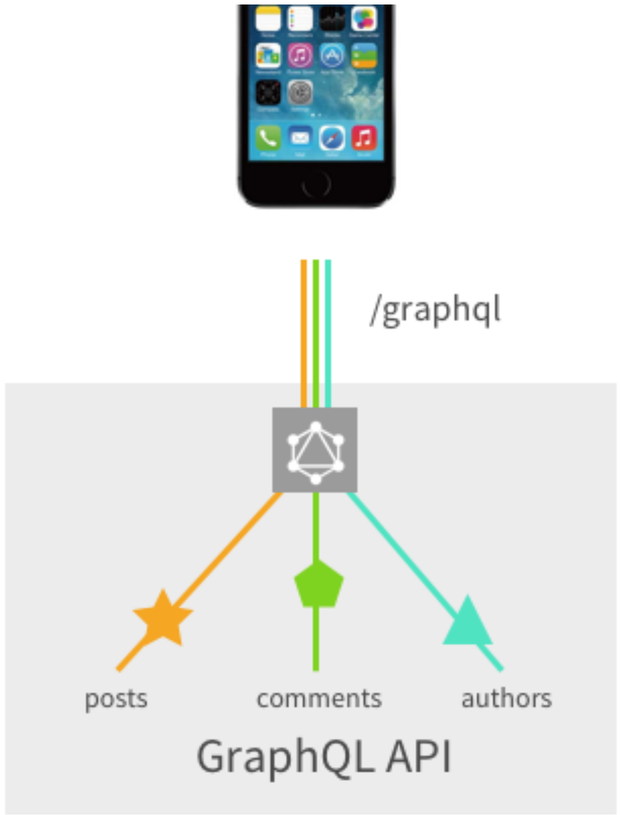
```
extension CharactersListStore {
    func fetch() async {
        let characters = await getCharacters()
        self.characters = characters
    }

    func getCharacters() async -> [Character] {
        [Character.mock]
    }
}
```

NETWORKING PROTOCOLS 02

REST

- HTTP
- Client <-> Server
- Stateless
- Uniform interface



GRAPHQL

Define types, fields & functions

```
type Query{
    me: User
}

type User{
    id: ID
    name: String
}

function Query_me(request){
    returnrequest.auth.user;
}

function User_name(user){
    returnuser.getName();
}
```

Run the query

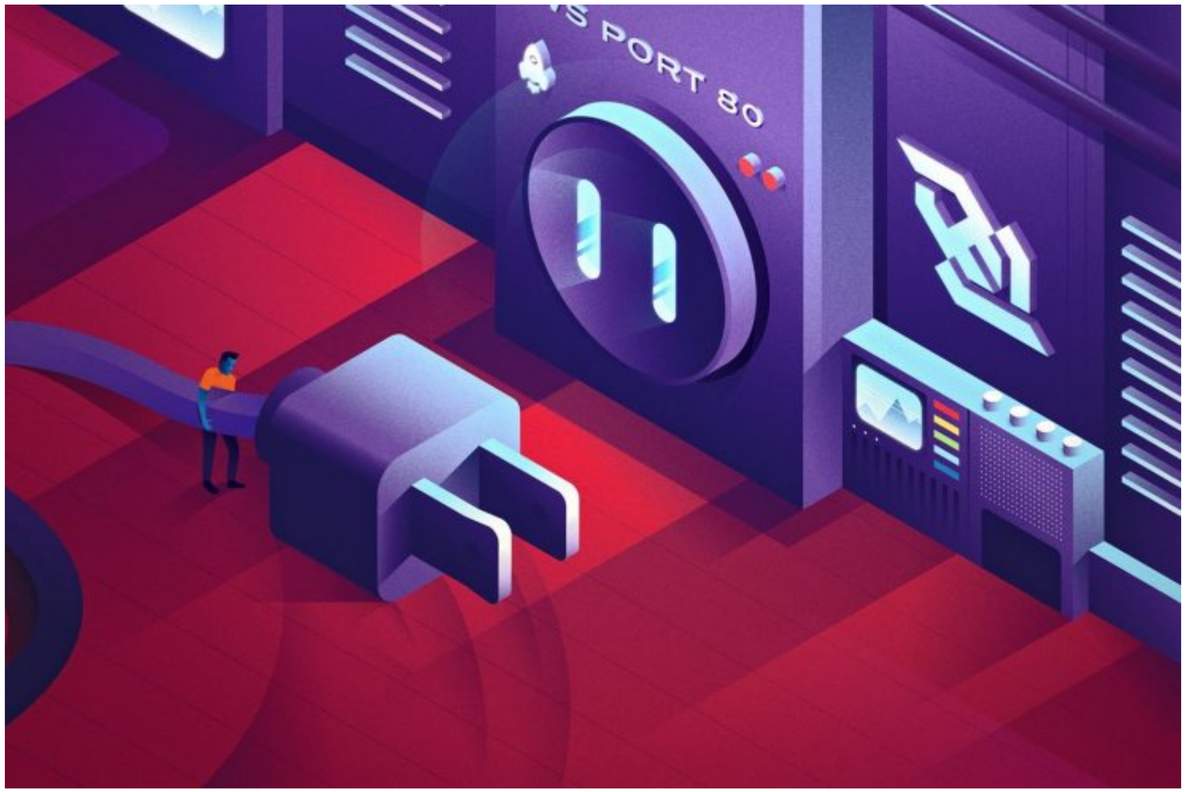
```
{
    me{
        name
    }
}
```

Return results

```
{
    "me":{
        "name":"John Doe"
    }
}
```

REAL TIME COMMUNICATION

- Web sockets (URLSessionWebSocketTask)
- Firebase
- WebRTC
- MQTT



URLSESSION

04

CONFIGURATION

- Special sessions
- Setting common parameters
- Internet connection type
- Caching

```
let configuration = URLSessionConfiguration.default
configuration.allowsCellularAccess = false
configuration.timeoutIntervalForRequest = 30
configuration.httpAdditionalHeaders = ["User-Agent": "MyApp 1.0"]

let session = URLSession(
    configuration: configuration,
    delegate: nil,
    delegateQueue: nil
)
```

URLREQUEST

- Different parameters
- Custom headers
- Overrides session defaults
- Same code across the app

```
let url = URL(string: "https://www.strv.com")!

var request = URLRequest(url: url)
request.httpMethod = "GET"
request.timeoutInterval = 10
request.cachePolicy = .useProtocolCachePolicy
request.addValue(
    "Bearer 123",
    forHTTPHeaderField: "Authorization"
)
```

DATA TASK

- Asynchronous data task
- Closure, Combine, async/await variants
- Upload/download tasks
- Returns Data & URLResponse

```
do {
    let (data, response) = try await session.data(for: request)
    print(data)
    print(response)
} catch {
    print(error)
}
```

CODABLE

05

JSON REPRESENTATION

- Typical server response
- Other file formats can be implemented
- Different date formats
- Custom case conversion


```
▼ 0 {12}
    id  : 42
    name : Big Head Morty
    status : unknown
    species : Human
    type : Human with giant head
    gender : Male
  ▼ origin {2}
      name : unknown
      url : value
  ▼ location {2}
      name : Citadel of Ricks
      url : https://rickandmortyapi.com/api/location/3
    image : https://rickandmortyapi.com/api/character/avatar/42.jpeg
  ▼ episode [1]
      0 : https://rickandmortyapi.com/api/episode/22
    url : https://rickandmortyapi.com/api/character/42
    created : 2017-11-05T10:15:53.349Z
```

SWIFT REPRESENTATION

- Struct
- Nesting other structs
- Computed vars

```
struct Character {
    let id: Int
    let name: String
    let species: String
    let type: String
    let gender: String
    let origin: CharacterOrigin
    let location: CharacterLocation
    let imageUrl: URL
    let episodeUrls: [URL]

    var episodeIds: [Int] {
        episodeUrls
            .compactMap {
                Int($0.lastPathComponent)
            }
    }
}
```

DECODABLE

```
enum CodingKeys: String, CodingKey {
    case id
    case name
    case species
    case type
    case gender
    case origin
    case location
    case imageUrl = "image"
    case episodeUrls = "episode"
}
```

```
struct Character: Decodable {
    let id: Int
    let name: String
    let species: String
    let type: String
    let gender: String
    let origin: CharacterOrigin
    let location: CharacterLocation
    let imageUrl: URL
    let episodeUrls: [URL]

    var episodeIds: [Int] {
        episodeUrls
            .compactMap {
                Int($0.lastPathComponent)
            }
    }
```

APIMANAGER

06

MANAGER LAYER

APIMANAGER PROTOCOL

- Consumes any request
- That returns Decodable model
- Is async
- Either throws or returns model

```
protocol APIManaging {
    func request<T: Decodable>(
        _ endpoint: Endpoint
    ) async throws -> T
}
```

CONFIGURATION

- URLSession configuration
- OAuth
- Decoder
- Upload
- Download
- Environments

```
final class APIManager: APIManaging {
    private lazy var urlSession: URLSession = {
        let config = URLSessionConfiguration.default
        config.timeoutIntervalForRequest = 30
        config.timeoutIntervalForResource = 30

        return URLSession(configuration: config)
    }()

    private lazy var dateFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateFormat = "yyyy-MM-dd"

        return formatter
    }()

    private lazy var decoder: JSONDecoder = {
        let decoder = JSONDecoder()

        decoder.keyDecodingStrategy = .convertFromSnakeCase
        decoder.dateDecodingStrategy = .formatted(dateFormatter)

        return decoder
    }()
```

ROUTER

07

ROUTER PROTOCOL

- Abstraction of URLRequest
- Allows to define default behaviour
- Ability to mock an endpoint
- Customisable

```
protocol Endpoint {
    var path: String { get }
    var method: HTTPMethod { get }
    var urlParameters: [String: Any]? { get }
    var headers: [String: String]? { get }

    func asRequest() throws -> URLRequest
}
```

IMPLEMENTATION

- Each request is bound to a router
- Using default implementation
- Ability to customise each request
- Extendable

```
enum CharactersRouter {
    case getCharacters(page: Int?)
    case getCharacter(id: Character.ID)
}

extension CharactersRouter: Endpoint {
    var path: String {
        switch self {
        case .getCharacters:
            return "character"
        case let .getCharacter(id):
            return "character/\(id)"
        }
    }

    var urlParameters: [String: Any]? {
        switch self {
        case let .getCharacters(.some(page)):
            return ["page": page]
        case .getCharacters, .getCharacter:
            return nil
        }
    }
}
```

STATE MANAGEMENT

08

STATE DEFINITION

```
enum State: Equatable {
    case initial
    case loading
    case finished(loadingMore: Bool)
    case failed
}
```

STATE CHANGES

```
do {
    let response: PaginatedResponse<Character> = try await
        apiManager.request(endpoint)

    currentResponseInfo = response.info
    characters += response.results

    state = .finished(loadingMore: false)
} catch {
    Logger.log("\(error)", .error)

    state = .failed
}
```

LIVE CODING

THANK YOU!

STRV

QUESTIONS

STRV