

Сетевой слой и хранение данных в iOS

Никита Сабынин
скромный iOS-разработчик



Каким должен быть
сетевой слой?



Субъективное мнение того, кто хочет получать удовольствие от работы



- ▶ Какой должна быть сеть?
- ▶ Реализация должна быть скрыта за абстракцией

Субъективное мнение того, кто хочет получать удовольствие от работы



- ▶ Какой должна быть сеть?
- ▶ Реализация должна быть скрыта за абстракцией
- ▶ Архитектурно сетевой слой не должен влиять на приложение

Субъективное мнение того, кто хочет получать удовольствие от работы



- ▶ Какой должна быть сеть?
- ▶ Реализация должна быть скрыта за абстракцией
- ▶ Архитектурно сетевой слой не должен влиять на приложение
- ▶ Изменения в требованиях, контракте с бэкендом, не должны ломать приложение

Субъективное мнение того, кто хочет получать удовольствие от работы



- ▶ Какой должна быть сеть?
- ▶ Реализация должна быть скрыта за абстракцией
- ▶ Архитектурно сетевой слой не должен влиять на приложение
- ▶ Изменения в требованиях, контракте с бэкендом, не должны ломать приложение
- ▶ Сетевой слой и связанную функциональность можно разрабатывать без бэкенда

Какие варианты
встречаются в дикой
природе?

GOD OBJECT

```
class PomolimsyaBratki {

    static let shared = PomolimsyaBratki()
    private init() {}

    weak var delegate: SomeDelegate

    // MARK: - Auth
    func authorizeUser(login: String, password: String) { }

    // MARK: - Some business logic
    func createOrder(info: Order, items: [Items]) { }
    func deleteOrder(info: Order) { }
    func stealGreyPants(whichPants: Pants, fromWho: Person) { }

    // MARK: - Permissions
    func updatePermissions(
        id: String,
        to newPermissions: Permissions
    ) {}

    // MARK: - Some other fun shit
    func shootDeveloperInTheLeg(
        developerModel: Developer,
        forWhat: ReasonToBeHappy
    ) { }

}
```



Network manager

```
protocol NetworkManaging {
    func fetch<T: Decodable>(from endpoint: Endpoint) async throws -> T
}

final class NetworkManager: NetworkManaging {
    static let shared = NetworkManager()
    private let session: URLSession

    private init(session: URLSession = .shared) {
        self.session = session
    }

    func fetch<T: Decodable>(from endpoint: Endpoint) async throws -> T {
        let request = try endpoint.urlRequest()
        let (data, response) = try await session.data(for: request)

        guard let httpResponse = response as? HTTPURLResponse else {
            throw NetworkError.invalidResponse
        }

        try validateResponse(httpResponse)

        do {
            let decoder = JSONDecoder()
            return try decoder.decode(T.self, from: data)
        } catch {
            throw NetworkError.decodingFailed
        }
    }
}
```



Endpoint protocol

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

extension Endpoint {
    func urlRequest() throws -> URLRequest {
        let url = baseURL.appendingPathComponent(path)
        var request = URLRequest(url: url)
        request.httpMethod = method.rawValue
        request.allHTTPHeaderFields = headers

        if let parameters = parameters {
            if method == .get {
                var components = URLComponents(url: url, resolvingAgainstBaseURL:
false)
                components?.queryItems = parameters.map { URLQueryItem(name:
$0.key, value: "\($0.value)") }
                request.url = components?.url
            } else {
                request.httpBody = try JSONSerialization.data(withJSONObject:
parameters)
            }
        }

        return request
    }
}
```



Error Handling

```
enum NetworkError: Error {
    case invalidResponse
    case decodingFailed
    case clientError(Int)
    case serverError(Int)
    case unknownError(Int)
}

extension NetworkError: LocalizedError {
    var errorDescription: String? {
        switch self {
        case .invalidResponse:
            return "Invalid response received from the server."
        case .decodingFailed:
            return "Failed to decode the response data."
        case .clientError(let statusCode):
            return "Client error occurred. Status code: \(statusCode)"
        case .serverError(let statusCode):
            return "Server error occurred. Status code: \(statusCode)"
        case .unknownError(let statusCode):
            return "An unknown error occurred. Status code: \(statusCode)"
        }
    }
}
```



Есть вариант
поинтереснее?

Model controller

Доклад: Архитектура сетевого слоя от Романа Бусыгина



Рецепт приготовления вкусной сети



- ▶ Нужно понять, что нам требуется

Рецепт приготовления вкусной сети



- ▶ Нужно понять, что нам требуется
- ▶ Описываем это протоколом

Рецепт приготовления вкусной сети



```
protocol VkusnoITochkaMenuProviding {
    func provideMenu(for vkusnoID: String)
}

protocol WeatherReportProviding {
    func provideWeatherReport(for city: String)
}

protocol PhotoUploading {
    func uploadPhoto(_ data: Data, attributes: PhotoAttributes)
}
```

Рецепт приготовления вкусной сети



- ▶ Нужно понять, что нам требуется
- ▶ Описываем это протоколом
- ▶ Делаем 2 реализации протокола: `stub` и `real deal(prod)`

```
class VkusnoITochkaMenuProvider: VkusnoITochkaMenuProviding {
    func provideMenu(for vkuskoID: String) { }
}

class StubVkusnoITochkaMenuProvider: VkusnoITochkaMenuProviding {
    func provideMenu(for vkuskoID: String) { }
}

class AppFactory {
    func makeVkusnoITochkaMenuProvider() ->
        VkusnoITochkaMenuProviding {
        isTesting
        ? StubVkusnoITochkaMenuProvider()
        : VkusnoITochkaMenuProvider()
    }
}
```

Рецепт приготовления вкусной сети



- ▶ Нужно понять, что нам требуется
- ▶ Описываем это протоколом
- ▶ Делаем 2 реализации протокола: stub и real deal(prod)
- ▶ Продовая реализация model controller

```
class VkusnoITochkaMenuProvider: VkusnoITochkaMenuProviding {
    private let api: APIClient

    init(
        apiClient: APIClient,
        cache: MenuCache
    ) { }

    func provideMenu(for vkuskoID: String) {
        let request = VkusnoITochkaMenuRequest(id: vkuskoID)
        api.performRequest(request)
    }
}
```

Рецепт приготовления вкусной сети



- ▶ Нужно понять, что нам требуется
- ▶ Описываем это протоколом
- ▶ Делаем 2 реализации протокола: stub и real deal(prod)
- ▶ Продовая реализация model controller
- ▶ API client

```
class APIClient {
    private let httpClient: HTTPClient

    init(env: someAPIEnv, httpClient: HTTPClient) { }

    func performRequest(_ request: YourRequest) {
        let httpRequest = makeHTTPRequest(apiRequest: request)
        httpClient.perform(httpRequest)
    }

    private func makeHTTPRequest(apiRequest: YourRequest) -> URLRequest { }
```

Рецепт приготовления вкусной сети



- ▶ Нужно понять, что нам требуется
- ▶ Описываем это протоколом
- ▶ Делаем 2 реализации протокола: stub и real deal(prod)
- ▶ Продовая реализация model controller
- ▶ API client
- ▶ HTTP client

```
protocol HTTPClient {
    func perform(_ httpRequest: URLRequest) -> HTTPResponse
}

class HTTPClient {
    private let urlSession: URLSession

    init(urlSession: URLSession) { }

    func perform(_ httpRequest: URLRequest) -> HTTPResponse {
        let task = urlSession.dataTask(with: URLRequest(httpRequest))

        task.resume()
    }
}

extension URLRequest {
    init(_ httpRequest: URLRequest) {}
}
```

Какие плюшки мы получаем?

Плюсы подхода



- ▶ Становится возможным без перепроектирования:
 - ▷ Комфортный переезд с URLSession, AF etc
 - ▷ Локальное изменение протокола для endpoint
 - ▷ Изменение количества запросов для выполнения сценария

Мониторинг сетевых проблем

Какие бывают сетевые проблемы?



- ▶ Разные окружения QA / Production
- ▶ Балансировка – разные кластеры
- ▶ Авторизация
- ▶ Ошибки на бэке

Теперь про мобилки



- ▶ Смена канала
- ▶ Смена базовых станций
- ▶ Нестабильность сети

Реальный мир

>

Эмуляция

>

XCode

Как с этим жить?
Логи.

Запрос URLSession

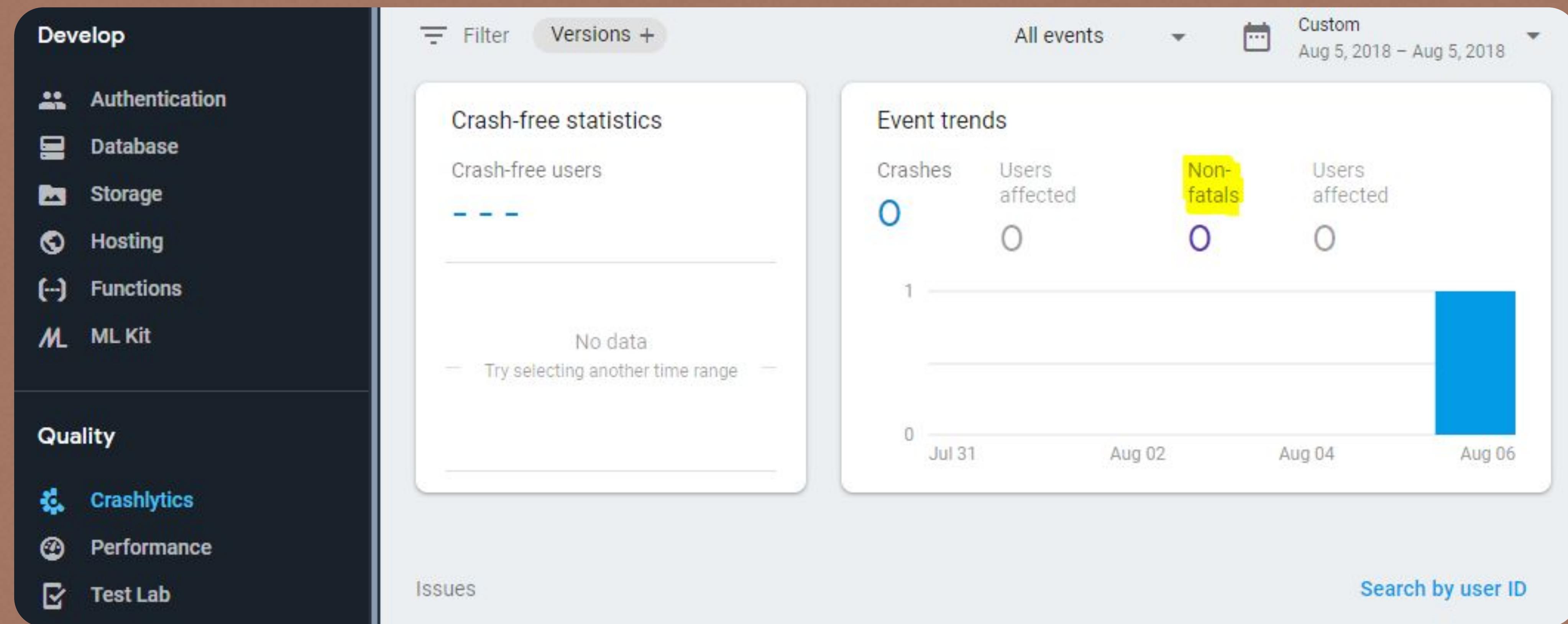


```
URLSession.shared.dataTask(with: url) { data, response, error in
    guard let data = data, error == nil else {
        ???
        return
    }

    do {
        let page = try JSONDecoder().decode(MoviePage.self, from: data)
        completion(page)
    } catch _ {
        ???
    }.resume()
}
```

Crashlytics

Crashlytics. Non-fatal

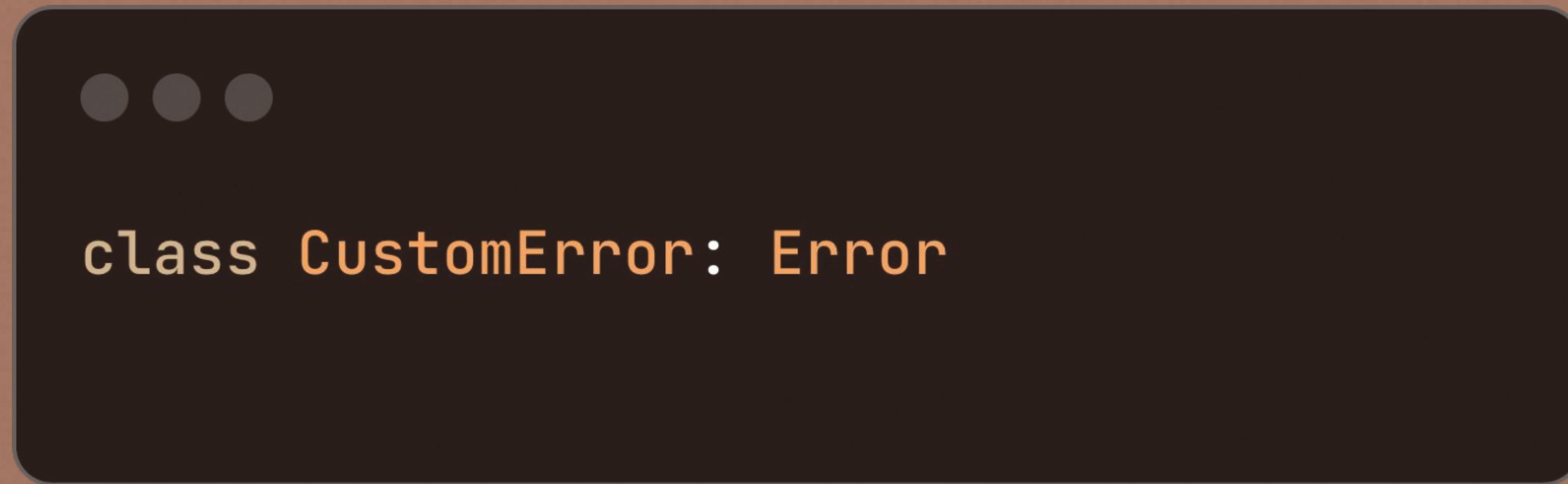


Crashlytics. API



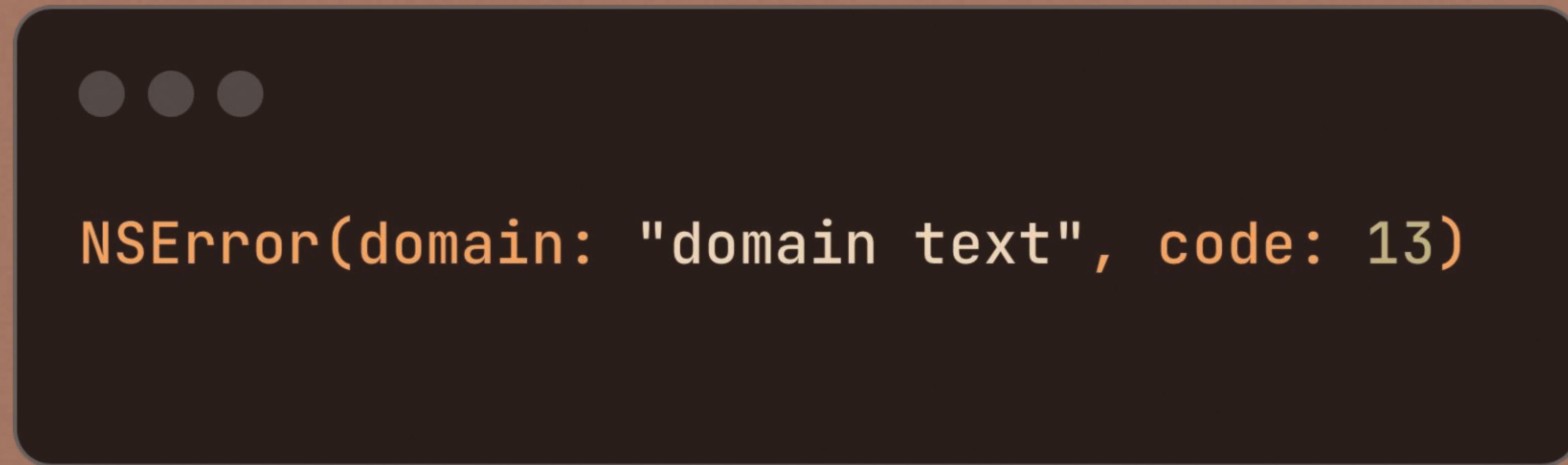
```
Crashlytics.crashlytics().record(error: Error)
```

Crashlytics. CustomError



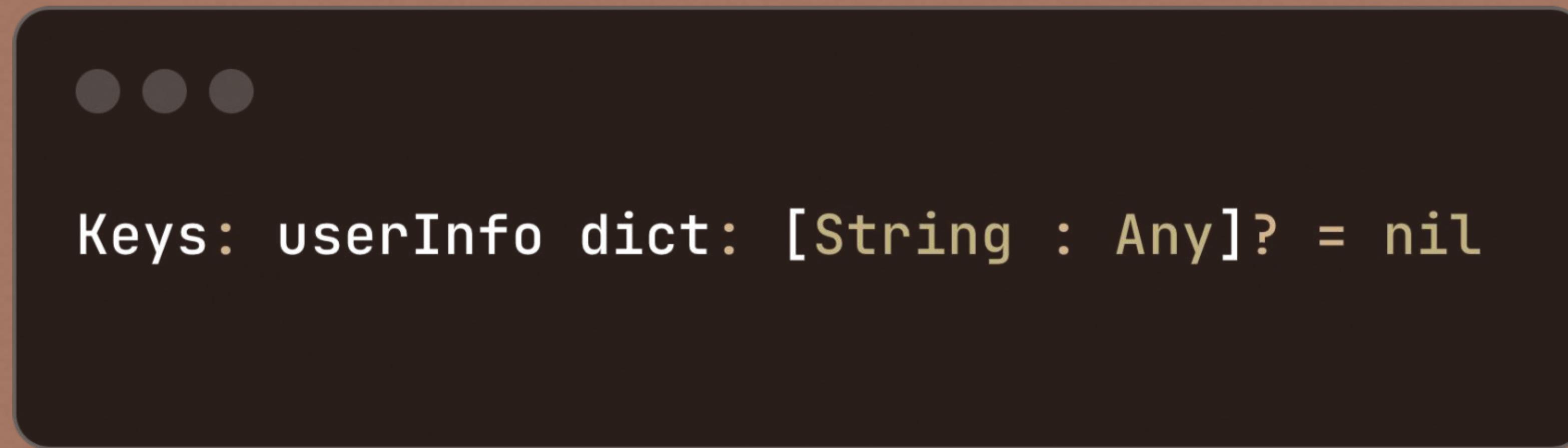
matches.CustomError	Non-fatal	1.0 – 1.0	3	1
1				

Crashlytics. NSError



A screenshot of a Crashlytics dashboard. At the top, there are filter buttons for "Filter issues" and "Issue state = "Open"" with an "X" button, and a search bar with the placeholder "Search issue title, subtitle or keys". To the right is a vertical ellipsis menu. Below the header is a table with columns: Issues, Event type, Versions, Events, and Users. A single row is visible, showing "domain text" in the Issues column (which is highlighted with a red border), "Non-fatal" in the Event type column, "1.0 – 1.0" in the Versions column, "3" in the Events column, and "1" in the Users column.

Crashlytics. NSError

A screenshot of the Crashlytics interface. At the top, there are tabs: "Stack trace", "Keys" (which is highlighted with a red border), "Logs", and "Data". Below the tabs is a search bar with the placeholder "Filter keys". A table follows, listing error details:

Key	Value
count	1
hasAuth	1
nserror-code	0
nserror-domain	matches parse error
paramters	photo:false&contacts:true&favourite:true
url	https://us-central1-matches-3813a.cloudfunctions.net/match

Crashlytics. Результат



```
...  
Crashlytics.crashlytics().record(  
    error: NSError(  
        domain: "parsing error",  
        code: 0,  
        userInfo: [  
            "hasAuth": hasAuth,  
            "count": someStuff.count,  
            "url": url,  
            "parameters": filter.queryParams()  
                .map { $0 + ":" + $1 }.joined(separator: "&")  
        ]  
    )  
)
```

Crashlytics. Логи



...

```
Crashlytics.crashlytics().log("some kind of info")
```

Crashlytics. Логи



Stack trace Keys **Logs** Data

Filter logs [Download .log](#)

#	Time (newest first)	Source	Log
1	11:51:22.687 pm	☰	sceneDidBecomeActive
2	11:51:21.221 pm	☰	sceneWillResignActive
3	11:51:19.389 pm	☰	sceneDidBecomeActive
4	11:51:17.946 pm	☰	sceneWillResignActive
5	11:51:16.944 pm	☰	sceneDidBecomeActive
6	11:51:14.721 pm	☰	sceneWillResignActive
7	11:50:56.694 pm	☰	sceneDidBecomeActive
8	11:50:56.312 pm	☰	didFinishLaunchingWithOptions

Если вы активно всё логируете,
можно упереться
в ограничения Crashlytics

Лучше уметь собирать
специальные ошибки

Stack trace	Keys	Logs	Data
Filter keys			
Key	Value		
activeLog	L:11:51 A:11:51 B:11:51 A:12:23		
count	1		

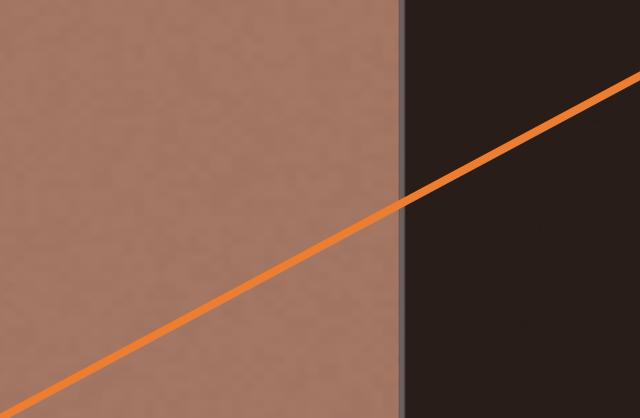
Так что же логировать?

Что логировать? Транспортная ошибка



```
URLSession.shared.dataTask(with: url) { data, response, error in
    guard let data = data, error == nil else {
        ???  
        return
    }

    do {
        let page = try JSONDecoder().decode(MoviePage.self, from: data)
        completion(page)
    } catch _ {
        ???
    }.resume()
}
```



Что логировать?



- ▶ А ЧТО МЫ МОЖЕМ ПРОИГНОРИРОВАТЬ?
 - ▷ NSURLErrorCanceled = -999
 - ▷ NSURLErrorTimedOut = -1001
 - ▷ NSURLErrorNetworkConnectionLost = -1005
 - ▷ NSURLErrorNotConnectedToInternet = -1009

**Эти ошибки могут быть нам важны,
но лучше их собирать
на другом логическом уровне**

Что логировать? Ошибка парсинга



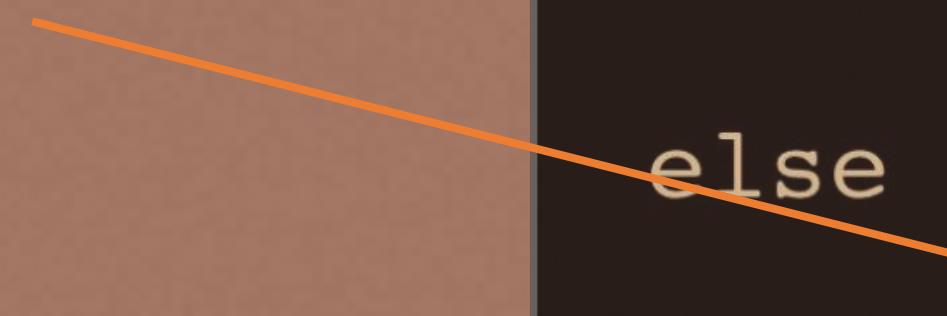
```
URLSession.shared.dataTask(with: url) { data, response, error in
    guard let data = data, error == nil else {
        ???
        return
    }

    do {
        let page = try JSONDecoder().decode(MoviePage.self, from: data)
        completion(page)
    } catch _ {
        ???
    }.resume()
}
```

Что логировать? Ошибка валидации



```
guard  
    someStuff.count > 0,  
    someStuff.error == nil,  
    someStuff.session != nil,  
  
else {  
    ???  
    return  
}
```



Осторожнее с логами,
мои чуваки!

Parameters: login:kurkin password:**magic12345**

Response: { chatid: 343453, pinned: false, last_message: {
 “msg”: “Жена в командировке. Мечтаю тебя увидеть”}
}

Воркшоп: Мониторинг сетевых проблем от Дмитрия Куркина



Как меня найти (если нужно)



Связь со мной



@humble_tech

Телеграм-канал



@нет_телеграм
_канала

Никита Сабынин
Ответит на твои вопросы

?any questions?