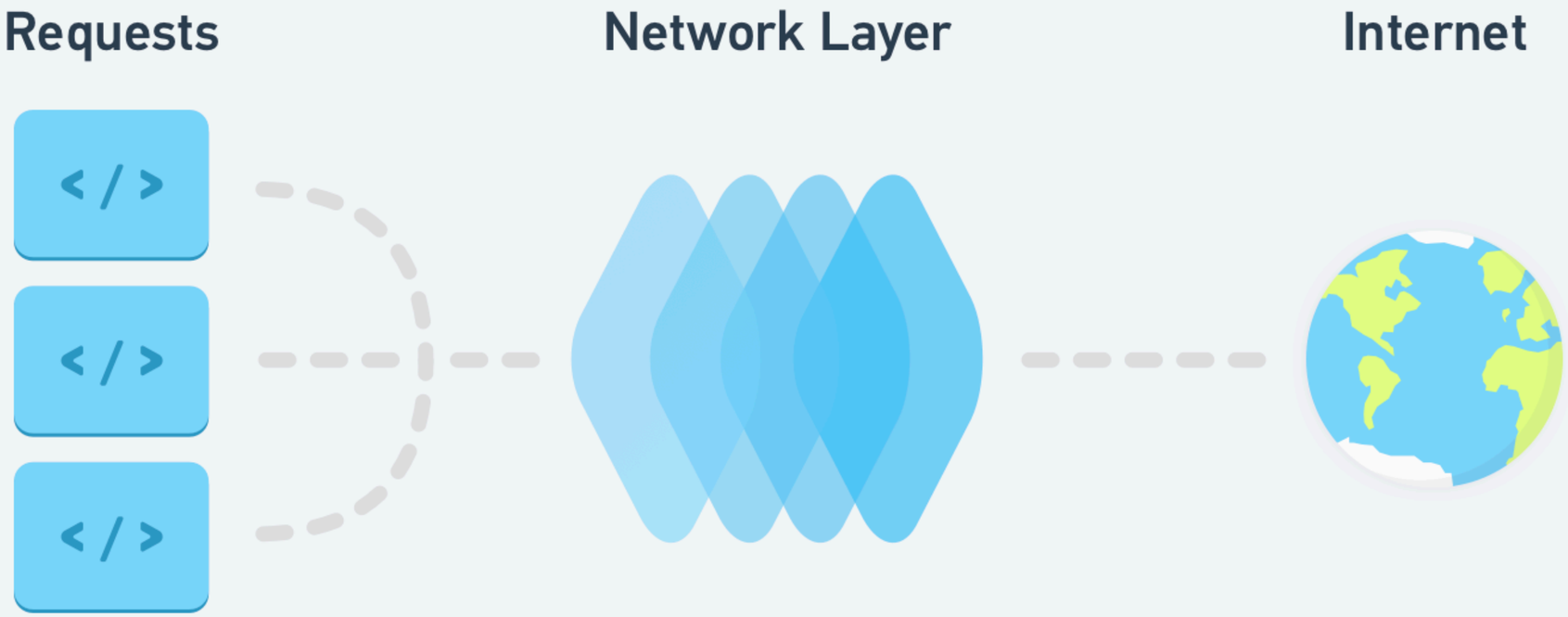
Network Layer

@alexandercg


Requests
Network Layer
Internet
< / >
< / >
< / >

```
POST /calculator.asmx HTTP/1.1
Host: www.dneonline.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <Add xmlns="http://tempuri.org/">
            <intA>int</intA>
            <intB>int</intB>
        </Add>
    </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <AddResponse xmlns="http://tempuri.org/">
            <AddResult>int</AddResult>
        </AddResponse>
    </soap:Body>
</soap:Envelope>
```

```
- (void)request:(NSString *)value {

    NSString *param = [value stringByReplacingOccurrencesOfString: @"&" withString:@"&amp;"];
    NSString *soapBody = [NSString stringWithFormat:
                          @"<?xml version=\"1.0\" encoding=\"utf-8\"?>\n"
                          "<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xmlns:xsd=\"http://
www.w3.org/2001/XMLSchema\" xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">\n"
                          "<soap:Body>\n"
                          "<VerificaQr xmlns=\"http://tempuri.org/\">\n"
                          "<cadenaQr>%@</cadenaQr>\n"
                          "</VerificaQr>\n"
                          "</soap:Body>\n"
                          "</soap:Envelope>", param];

    NSURL *url = [NSURL URLWithString:@"http://ws2014.cfdis.mx/WSVerifica.asmx"];
    NSString *msgLength = [NSString stringWithFormat:@"%lu", (unsigned long)[soapBody length]];

    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url];
    [request addValue:@"text/xml; charset=utf-8" forHTTPHeaderField:@"Content-Type"];
    [request addValue:@"http://tempuri.org/VerificaQr" forHTTPHeaderField:@"SOAPAction"];
    [request addValue:msgLength forHTTPHeaderField:@"Content-Length"];
    [request setHTTPMethod:@"POST"];
    [request setHTTPBody:[soapBody dataUsingEncoding:NSUTF8StringEncoding]];


    NSURLConnection *theConnection = [[NSURLConnection alloc] initWithRequest:request delegate:self];

    if(theConnection){
        webData = [NSMutableData data];
    }
    else{
        NSLog(@"theConnection is NULL");
    }
}
```

```
-(void) connectionDidFinishLoading:(NSURLConnection *) connection {
    //NSLog(@"DONE. Received Bytes: %lu", (unsigned long)[webData length]);
    NSString *theXML = [[NSString alloc]
                        initWithBytes: [webData mutableBytes]
                        length:[webData length]
                        encoding:NSUTF8StringEncoding];
    //---shows the XML---
    //NSLog(@"%@", theXML);

    [self verificarRespuesta:theXML];

}
```

sudz-c

código fuente limpio desde sus servicios web

Escriba la dirección web del WSDL para convertir

Cargar un WSDL

http://www.dneonline.com/calculator.asmx?WSDL

Elija un espacio de nombres para el código generado

SDZ

asegura definiciones de clases únicas (recomendado)

Elija el tipo de paquete de códigos para crear

Objective-C para iOS (Recuento Automático de Referencias)

Generar

Hemos incluido todo lo que necesita para comenzar.
vicios web con unas pocas líneas de código.
arrolladores para implementar servicios web en su aplicación.

Calculator.iPhone

- Examples
  - SudzCExamples_Prefix.pch
  - SudzCExamples-Info.plist
  - SudzCExamplesAppDelegate.h
  - SudzCExamples.xcodeproj
  - main.m
  - MainWindow.xib
  - SudzCExamplesAppDelegate.m
- WSDL
  - calculator.wsdl
- Source
  - Examples
  - TouchXML
  - Generated
  - Soap
- Documentation
  - tutorial
  - assets
  - framework
    - SDZCalculator.html
  - classes
  - toc
    - index.html

```objc
- (void)run {
  // Create the service
  SDZCalculator* service = [SDZCalculator service];
  service.logging = YES;
  // service.username = @"username";
  // service.password = @"password";


  // Returns int
  /* Adds two integers. This is a test WebService. ©DNE Online */
  [service Add:self action:@selector(AddHandler:) intA: 0 intB: 0];

  // Returns int
  /*  */
  [service Divide:self action:@selector(DivideHandler:) intA: 0 intB: 0];

  // Returns int
  /*  */
  [service Multiply:self action:@selector(MultiplyHandler:) intA: 0 intB: 0];

  // Returns int
  /*  */
  [service Subtract:self action:@selector(SubtractHandler:) intA: 0 intB: 0];
}
```

```swift
enum ApiURL {
    static let dev = "https://some-base-url.com/"
    //static let staging = ""
    //static let production = ""
}

enum ApiEndpoint {
    static let userProfile  = "api/user"
}

struct UserRequest: Request {
    typealias ResponseObject = User

    func build() -> URLRequest {
        let url = URL(string: "\(ApiURL.dev)\(ApiEndpoint.userProfile)")!
        return URLRequest(url: url)
    }
}

class UserInteractor: Interactor {
    static let shared = UserInteractor()

    var user:User?

    func getUser(block: @escaping ResultBlock) {
        let userRequest = UserRequest()

        APIClient().perform(userRequest) { (response: Result<User, Error>) in
            switch response {
            case let .success(user):
                print("Here's the user: \(user)")
                UserInteractor.shared.user = user
                block(user)
            case let .failure(error):
                print("Oh no, an error: \(error)")
            }
        }

    }
}
```

Requirement

You are given an api RESTful with the first endpoint to load a model

Observations Endpoints

Overview:

- `/observations` (GET, POST)
- `/observations/{id}` (GET, PUT, DELETE)

```swift
struct Observation: Codable {
    var id: String
    var createdBy: String?
    var createdTime: Date?
    var content: String?
    var media: [URL]?
    var tags: [String]?
    var location: String?
    var likeCount: Int
    var isLiked: Bool
    var commentCount: Int

    // Internal logic
    //  . . . .

}
```

ALAMOFIRE

Elegant Networking in Swift

```swift
final class NetworkServiceWithAlamofire {

    private let baseURL: String = "https://your-base-url.com/api"

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

    func getObservations(completion: @escaping (Result<[Observation], Error>) -> Void) {

        let observationEndpoint: String = "/observations"

        Alamofire.request("\(baseURL)\(observationEndpoint)", method: .get).responseJSON { (response) in

            guard response.data != nil else { return }

            var observations: [Observation]?
            do {
                observations = try JSONDecoder().decode([Observation].self, from: response.data!)
            } catch {
                print("Could not parse JSON: \(error)")
                completion(.failure(error))
            }

            if let observations = observations {
                completion(.success(observations))
            } else {
                completion(.failure(response.error!))
            }
        }
    }

}
```

```swift
func post(observation: Observation, completion: @escaping (Result<Observation, Error>) -> Void) {
    // Implementation with Alamofire
    // ....
}

func put(observation: Observation, completion: @escaping (Result<Observation, Error>) -> Void) {
    // Implementation with Alamofire
    // ....
}

func delete(observation: Observation, completion: @escaping (Result<Bool, Error>) -> Void) {
    // Implementation with Alamofire
    // ....
}
```

URLSession

The URLSession class and related classes provide an API for downloading data from and uploading data to endpoints indicated by URLs. The API also enables your app to perform background downloads when your app isn't running or, in iOS, while your app is suspended. A rich set of delegate methods support authentication and allow your app to be notified of events like redirection.

Using the URLSession API, your app creates one or more sessions, each of which coordinates a group of related data transfer tasks. For example, if you're creating a web browser, your app might create one session per tab or window, or one session for interactive use and another for background downloads. Within each session, your app adds a series of tasks, each of which represents a request for a specific URL (following HTTP redirects, if necessary).

```swift
final class NetworkServiceWithURLSession {

    private let baseURL: String = "https://your-base-url.com/api"

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

    func fetchObservations(completion: @escaping (Result<[Observation], Error>) -> Void) {

        let observationEndpoint: String = "/observations"
        var urlRequest = URLRequest(url: URL(string: "\(baseURL)\(observationEndpoint)")!)
        urlRequest.httpMethod = "GET"

        let task = URLSession(configuration: .default).dataTask(with: urlRequest) { data, response, error in
            guard error == nil else {
                completion(.failure(error!))
                return
            }

            guard let httpResponse = response as? HTTPURLResponse else {
                completion(.failure(error!))
                return
            }

            guard (200...299).contains(httpResponse.statusCode) else {
                completion(.failure(error!))
                return
            }

            // do you parsing logic

            completion(.success([]))
        }
        task.resume()

    }

}
```

```
func post(observation: Observation, completion: @escaping (Result<Observation, Error>) -> Void) {
    // Implementation with URLSession
    // ....
}

func put(observation: Observation, completion: @escaping (Result<Observation, Error>) -> Void) {
    // Implementation with URLSession
    // ....
}

func delete(observation: Observation, completion: @escaping (Result<Bool, Error>) -> Void) {
    // Implementation with URLSession
    // ....
}
```

URLSession + Generic-ish

Some *Generic code* enables you to write flexible, reusable functions and types that can work with any type, subject to requirements that you define. You can write code that avoids duplication and expresses its intent in a clear, abstracted manner. Generics are one of the most powerful features of Swift, and much of the Swift standard library is built with generic code.

In fact, you've been using generics throughout the *Language Guide*, even if you didn't realize it. For example,
Swift's Array and Dictionary types are both generic collections. You can create an array that holds Int values, or an array that holds String values, or indeed an array for any other type that can be created in Swift. Similarly, you can create a dictionary to store values of any specified type, and there are no limitations on what that type can be.

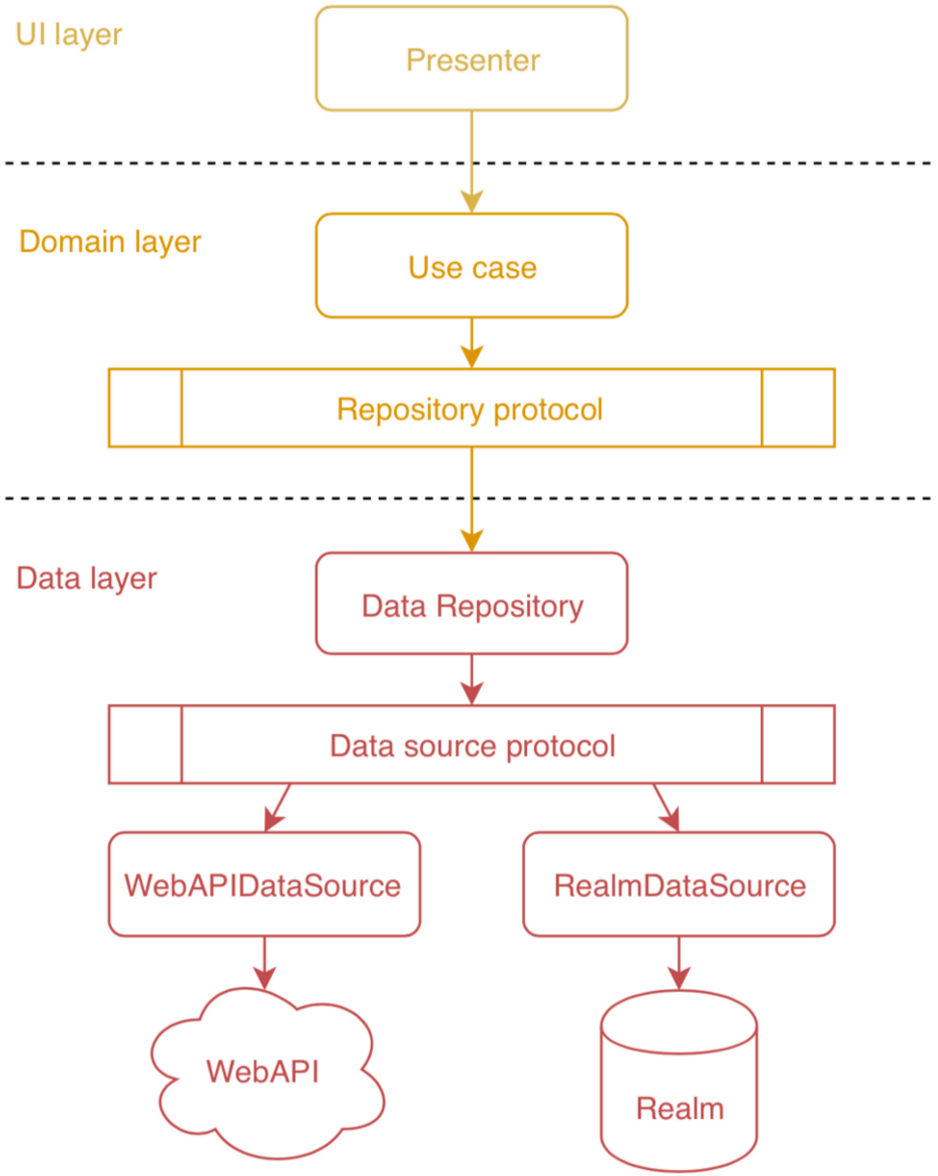
```swift
func getData(from originalRequest: URLRequest,
                completion: @escaping ( _ data: Data?, _ response: HTTPURLResponse?, _ error: APIError?) -> Void) {

        var request = originalRequest
        request.httpMethod = "GET"
        addAuthorizationHeader(to: &request)

        let task = URLSession(configuration: urlSessionConfiguration).dataTask(with: request) { data, response, error in
            guard error == nil else {
                completion(nil, nil, APIError.connectionError(error: error!))
                return
            }

            guard let httpResponse = response as? HTTPURLResponse else {
                completion(nil, nil, APIError.noResponse)
                return
            }

            guard (200...299).contains(httpResponse.statusCode) else {
                completion(nil, httpResponse, APIError.serverError(statusCode: httpResponse.statusCode))
                return
            }

            completion(data, httpResponse, nil)
        }
        task.resume()
    }
```

```swift
func getObject<T: Decodable>(from request: URLRequest, type: T.Type, consultCache: Bool,
                             completion: @escaping (_ value: T?, _ error: APIError?) -> Void) -> T? {

    let jsonDecoder = JSONDecoder()
    jsonDecoder.dateDecodingStrategy = .iso8601

    // First, check for a cached response and return it immediately if we have it in cache
    if consultCache, let cachedResponse = urlCache.cachedResponse(for: request) {
        if let obj = try? jsonDecoder.decode(T.self, from: cachedResponse.data) {
            return obj
        }
    }

    // The object wasn't available from cache, so go fetch it
    getData(from: request) { data, response, error in
        guard error == nil else {
            completion(nil, error)
            return
        }

        guard let mimeType = response!.mimeType, mimeType == "application/json" else {
            completion(nil, APIError.unexpectedMIMEType(mimeType: response!.mimeType ?? ""))
            return
        }

        guard let data = data else {
            completion(nil, APIError.noData)
            return
        }

        // Decode the data as JSON into the specified type
        do {
            let obj = try jsonDecoder.decode(T.self, from: data)
            completion(obj, nil)
        }
        catch let error {
            print("Decoding error occurred while decoding: \(data)")
            let sourceString = String(data: data, encoding: .utf8) ?? ""
            completion(nil, APIError.decodingError(error: error, sourceString: sourceString))
        }
    }

    return nil
}
```

```
func getObservations(completion: @escaping (_ observations: [Observation]?, _ error: APIError?) -> Void ) {
    guard let url = makeFullURL(string: "/observations"),
          let urlRequest = URLRequest(url: url) else { return }

    _ = getObject(from: request, type: type, consultCache: false, completion: completion)
}
```

```swift
func getUsers(completion: @escaping (_ observations: [User]?, _ error: APIError?) -> Void ) {
    guard let url = makeFullURL(string: "/users"),
          let urlRequest = URLRequest(url: url) else { return }

    _ = getObject(from: request, type: type, consultCache: false, completion: completion)
}

func getComments(completion: @escaping (_ observations: [Comment]?, _ error: APIError?) -> Void ) {
    guard let url = makeFullURL(string: "/comments"),
          let urlRequest = URLRequest(url: url) else { return }

    _ = getObject(from: request, type: type, consultCache: false, completion: completion)
}

func getPosts(completion: @escaping (_ observations: [Post]?, _ error: APIError?) -> Void ) {
    guard let url = makeFullURL(string: "/posts"),
          let urlRequest = URLRequest(url: url) else { return }

    _ = getObject(from: request, type: type, consultCache: false, completion: completion)
}
```

URLSession + Generic + Repository



```
class ObservationRepository {

    func getAll() -> [Observation] {
        // Code that returns from API
    }

    func get( identifier:Int ) -> Observation? {
        // API code
    }
    func create( article:Observation ) -> Bool {
        // API code
    }

    func update( article:Observation ) -> Bool {
        // API code
    }

    func delete( article:Observation ) -> Bool {
        // API code
    }
}

let observationRepo = ObservationRepository()

let observations = observationRepo.getAll()

let observation = observationRepo.get(identifier: 1)

observationRepo.create(article: observation)
observationRepo.delete(article: observation)
```

```swift
protocol ObservationRepository {
    func getAll() -> [Observation]
    func get( identifier:Int ) -> Observation?
    func create( article:Observation ) -> Bool
    func update( article:Observation ) -> Bool
    func delete( article:Observation ) -> Bool
}

class WebObservationRepository: ObservationRepository { }
class LocalObservationRepository: ObservationRepository { }


let webObservationRepo = WebObservationRepository()
webObservationRepo.getAll() // -> implementation to get data from the web


let localObservationRepo = LocalObservationRepository()
localObservationRepo.getAll() // -> implementation to get data from an offline file
```

Demo

Model Repository