```swift
import Foundation
import UIKit

// Model
struct Records: Codable {
    let records: [Furniture]?
}

struct Furniture: Codable {
    let id: String
    let fields: Fields
}

struct Fields: Codable {
    let name: String
    let type: String?
    let images: [FurnitureImage]?
    enum CodingKeys: String, CodingKey {
        case name = "Name"
        case type = "Type"
        case images = "Images"
    }
}

struct FurnitureImage: Codable {
    let url: String
}

struct Response: Codable {
    let id: String
    let deleted: Bool
}

struct ErrorResponse: Codable {
    let error: String
}

enum RequestType: String {
    case get = "GET"
    case post = "POST"
    case delete = "DELETE"
}

enum CustomError: Error {
    case requestError
    case statusCodeError
    case parsingError
}


// Request Factory
protocol RequestFactoryProtocol {
    func createRequest(urlStr: String, requestType: RequestType, params:
     [String]?) -> URLRequest
    func getFurnitureList(callback: @escaping ((errorType: CustomError?,
     errorMessage: String?), [Furniture]?) -> Void)
    func deleteFurniture(with id: String, callback: @escaping ((errorType:
     CustomError?, errorMessage: String?), Response?) -> Void)
}

private let furnitureUrlStr =
 "https://api.airtable.com/v0/appurgriHL535VVAP/Furniture"


class RequestFactory: RequestFactoryProtocol {
    internal func createRequest(urlStr: String, requestType: RequestType,
     params: [String]?) -> URLRequest {
        var url: URL = URL(string: urlStr)!
        if let params = params {
            var urlParams = urlStr
            for param in params {
                urlParams = urlParams + "/" + param
            }
            print(urlParams)
            url = URL(string: urlParams)!
        }

        var request = URLRequest(url: url)
        request.timeoutInterval = 100
        request.httpMethod = requestType.rawValue

        let accessToken = "keyuGTkgeGQoidxs6"
        request.setValue("Bearer \(accessToken)", forHTTPHeaderField:
         "Authorization")

        return request
    }

    func getFurnitureList(callback: @escaping ((errorType: CustomError?,
     errorMessage: String?), [Furniture]?) -> Void) {
        let session = URLSession(configuration: .default)
        let task = session.dataTask(with: createRequest(urlStr:
         furnitureUrlStr,
                                                        requestType: .get,
                                                        params: nil)) {
                                                   (data, response, error)
                                                   in
            if let data = data, error == nil {
                if let responseHttp = response as? HTTPURLResponse {
                    if responseHttp.statusCode == 200 {
                        if let response = try?
                         JSONDecoder().decode(Records.self, from: data) {
                            callback((nil, nil), response.records)
                        }
                        else {
                            callback((CustomError.parsingError, "parsing
                             error"), nil)
                        }
                    }
                    else {
                        callback((CustomError.statusCodeError, "status
                         code: \(responseHttp.statusCode)"), nil)
                    }
                }
            }
            else {
                callback((CustomError.requestError,
                 error.debugDescription), nil)
            }
        }
        task.resume()
    }

    func deleteFurniture(with id: String, callback: @escaping ((errorType:
     CustomError?, errorMessage: String?), Response?) -> Void) {
        let session = URLSession(configuration: .default)
        let task = session.dataTask(with: createRequest(urlStr:
         furnitureUrlStr,
                                                        requestType:
                                                   .delete,
                                                        params: [id])) {
                                                   (data, response, error)
                                                   in
            if let data = data, error == nil {
                if let responseHttp = response as? HTTPURLResponse {
                    if responseHttp.statusCode == 200 {
                        if let response = try?
                         JSONDecoder().decode(Response.self, from: data) {
                            callback((nil, nil), response)
                        }
                        else if let response = try?
                         JSONDecoder().decode(ErrorResponse.self, from:
                         data) {
                            callback((CustomError.requestError,
                             response.error), nil)
                        }
                        else {
                            callback((CustomError.parsingError, "parsing
                             error"), nil)
                        }
                    }
                    else {
                        callback((CustomError.statusCodeError, "status
                         code: \(responseHttp.statusCode)"), nil)
                    }
                }
            }
            else {
                callback((CustomError.requestError,
                 error.debugDescription), nil)
            }
        }
        task.resume()
    }
```

```
}


// Controller
let requestFactory = RequestFactory()

requestFactory.getFurnitureList { (errorHandle, furnitures) in
    if let _ = errorHandle.errorType, let errorMessage =
     errorHandle.errorMessage {
        print(errorMessage)

    }
    else if let list = furnitures, let furniture = list.last {
        print(furniture.id)
    }
    else {
        print("Houston we got a problem")
    }
}

requestFactory.deleteFurniture(with: "rec00x6nIiwAmQjhB") { (errorHandle,
 response) in
    if let _ = errorHandle.errorType, let errorMessage =
     errorHandle.errorMessage {
        print(errorMessage)
    }
    else if let reponse = response {
        print(reponse.deleted)
    }
    else {
        print("Houston we got a problem")
    }
}
```