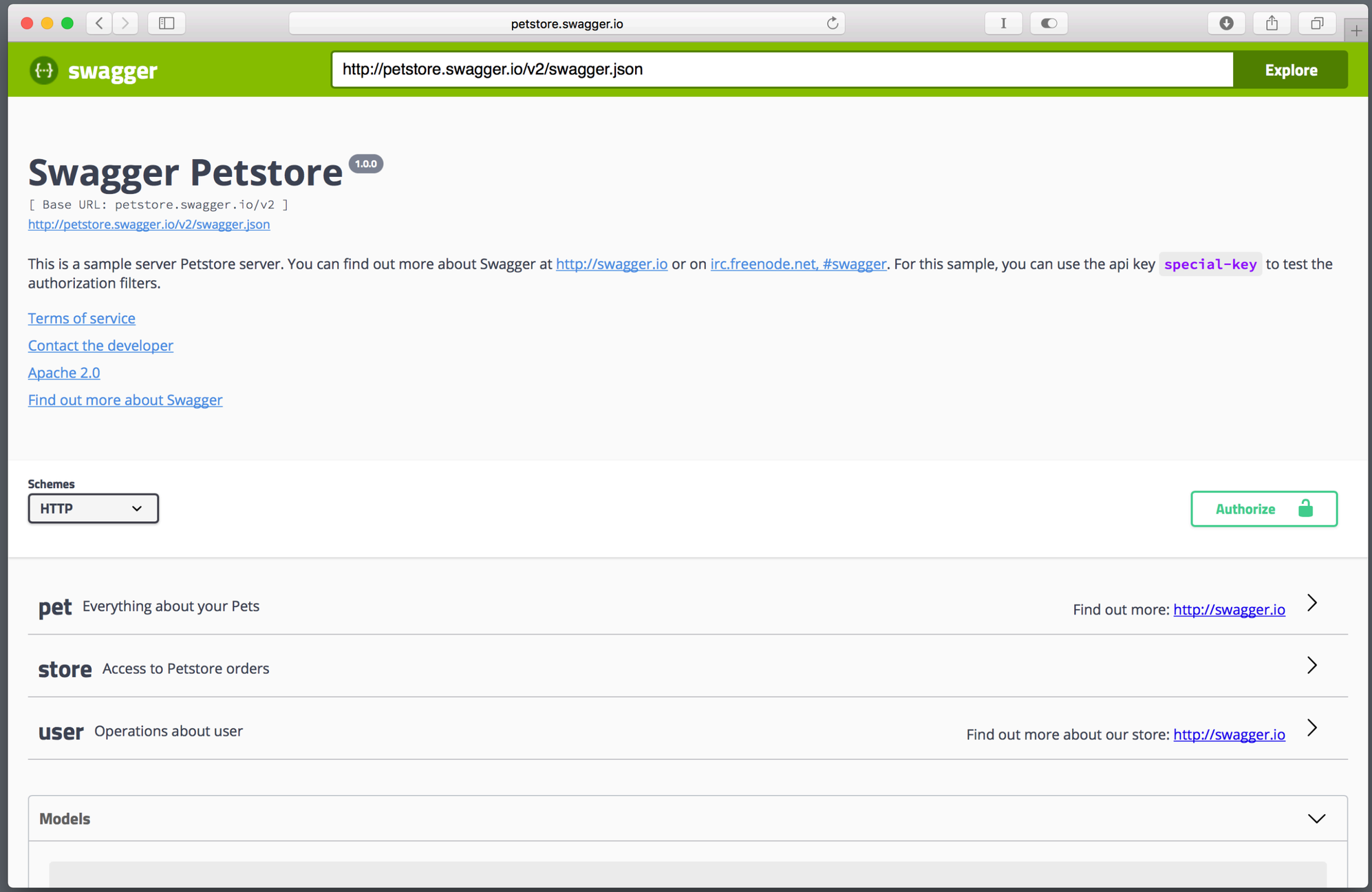
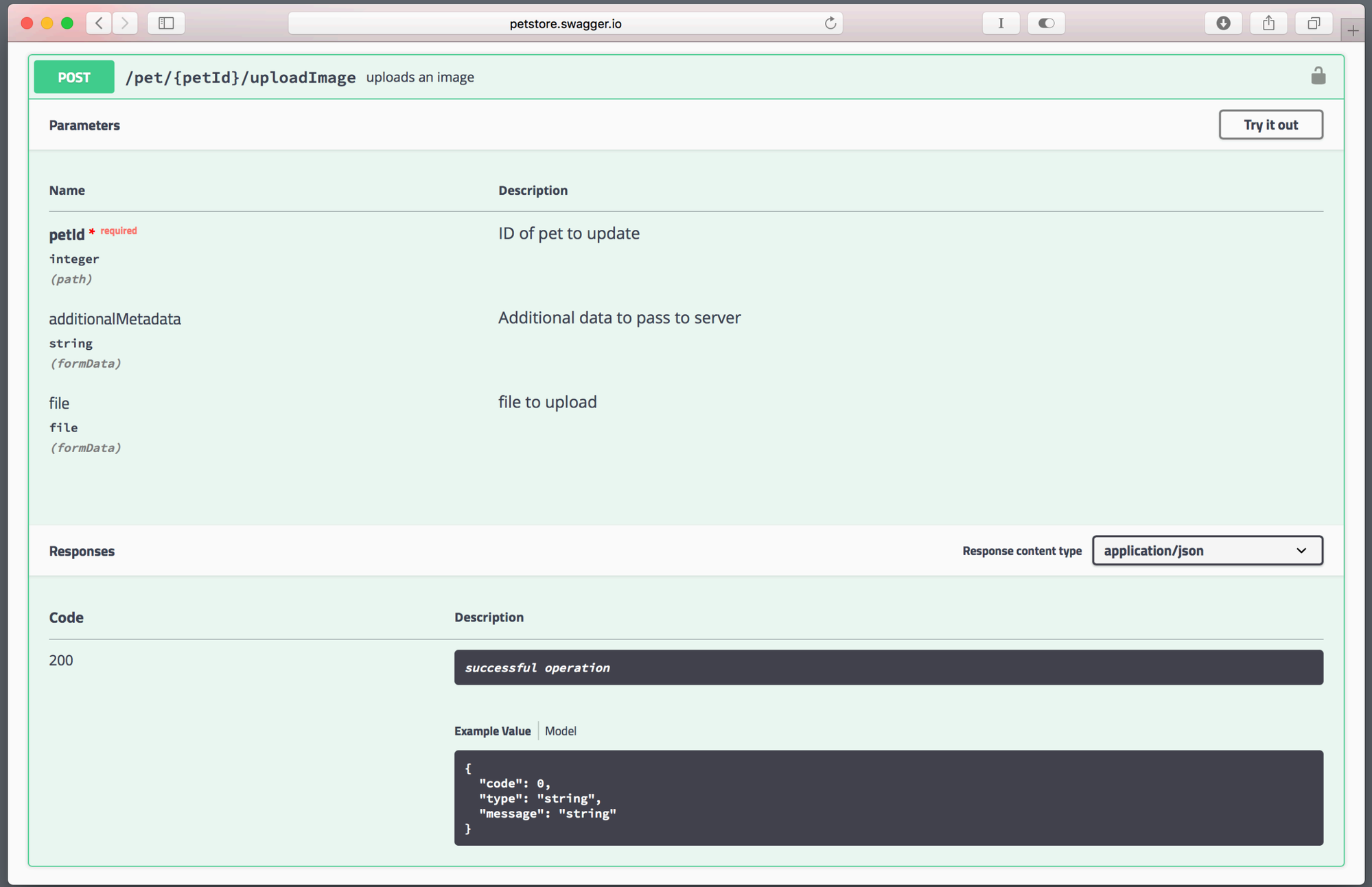
Networking


petstore.swagger.io
swagger
http://petstore.swagger.io/v2/swagger.json
Explore
Swagger Petstore
1.0.0
[Base URL: petstore.swagger.io/v2]
http://petstore.swagger.io/v2/swagger.json
This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key special-key to test the authorization filters.
Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger
Schemes
HTTP
Authorize
pet Everything about your Pets
Find out more: http://swagger.io
store Access to Petstore orders
user Operations about user
Find out more about our store: http://swagger.io
Models


petstore.swagger.io
POST
/pet/{petId}/uploadImage
uploads an image
Parameters
Try it out
Name
Description
petId * required
integer
(path)
ID of pet to update
additionalMetadata
string
(formData)
Additional data to pass to server
file
file
(formData)
file to upload
Responses
Response content type
application/json
Code
Description
200
successful operation
Example Value
Model
{
"code": 0,
"type": "string",
"message": "string"
}

```
ConfigurationLogic *logic = [ConfigurationLogic currentConfiguration];

[NetworkFetcher  callService:@"/users"
                        host:[logic baseURL]
                      method:GET
                    postData:nil
                  bodyInJSON:NO
                    hasCache:YES
             timeoutInterval:[logic servicesTimeout]
              successHandler:successHandler
              failureHandler:failureHandler];
```

```
func lookStockFor(symbol: String, handler: @escaping Handler) {
    let parameters = ["input": symbol]
    let request = networkFetcher.request(
        .GET,
        "http://dev.markitondemand.com/api/v2/lookup.json"
        parameters: parameters,
        encoding: .URL
    )

    request.responseJSON { result in
        guard result.error == nil {
            handler(nil, result.error)
            return
        }
        let symbols: [StockSymbols] = JSONParse(result.json)
        handler(symbols, nil)

    }
}
```

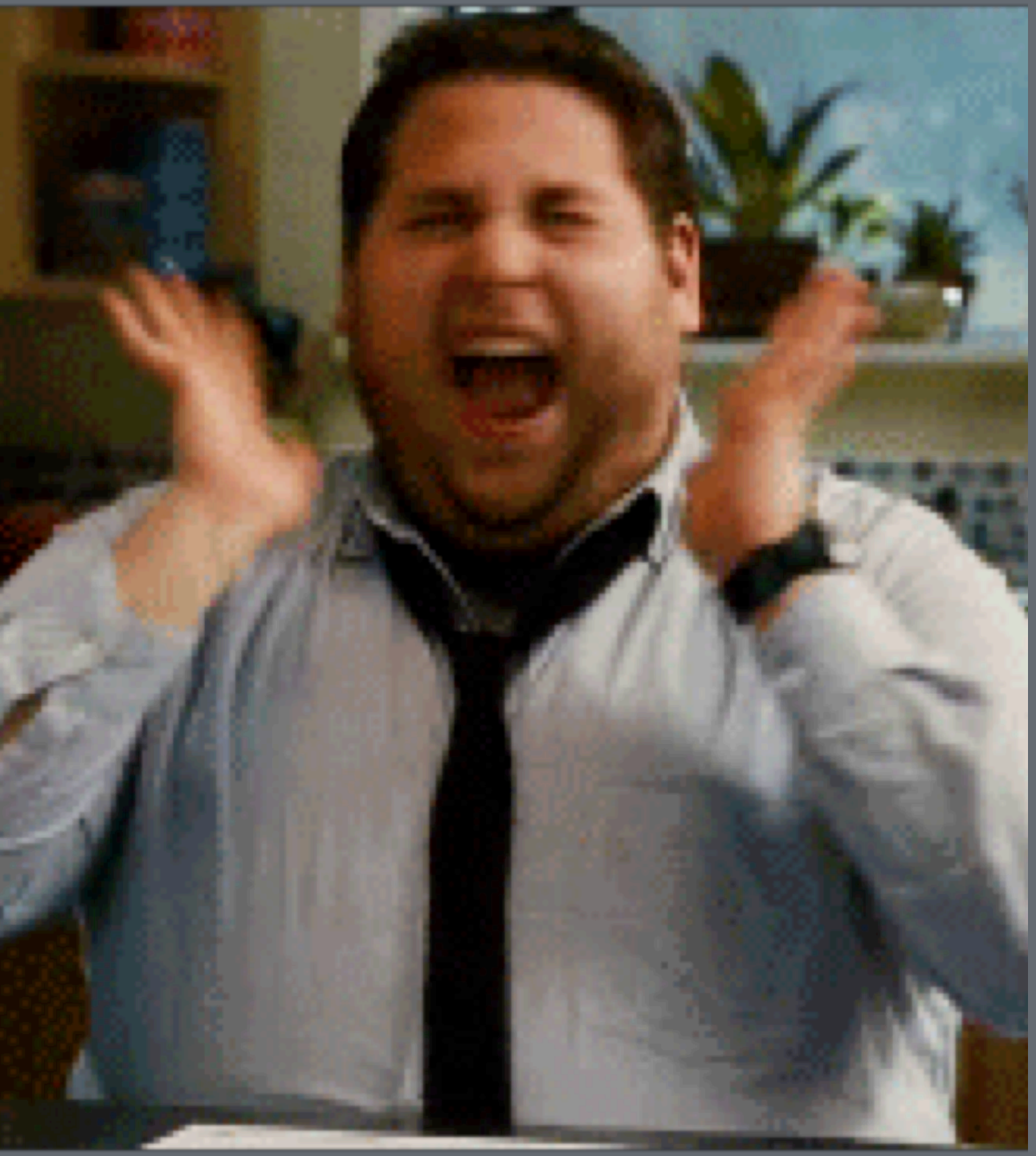
Objectives:

1. Composability
2. Sensitive Default Values
3. Readability
4. Self Documented
5. No `Alamofire`

More protocols

```swift
public protocol Endpoint {

    /// The path for the request
    var path: String { get }

    /// The HTTPMethod for the request
    var method: HTTPMethod { get }

    /// Optional parameters for the request
    var parameters: [String : Any]? { get }

    /// The HTTP headers to be sent
    var httpHeaderFields: [String : String]? { get }
}
```

```swift
//  This is the default implementation for Endpoint
extension Endpoint {
    public var method: HTTPMethod {
        return .GET
    }

    public var parameters: [String : AnyObject]? {
        return nil
    }

    public var httpHeaderFields: [String : String]? {
        return nil
    }
}
```



To Xcode!

What we've built:

- Thin abstraction on top of `URLSession`
- Protocol oriented
- A way to bind `Request` to `Response`
- Automatic parsing using `Decodable`

Improvements:

- Add URL encoding for parameters
- Dynamic signing of request
- Cache support
- Multipart upload
- Custom response validation