# GitHub Lite with GraphQL

Issarapong

Growth Session #9 - November 16-17 2017

# Intro

GitHub Lite

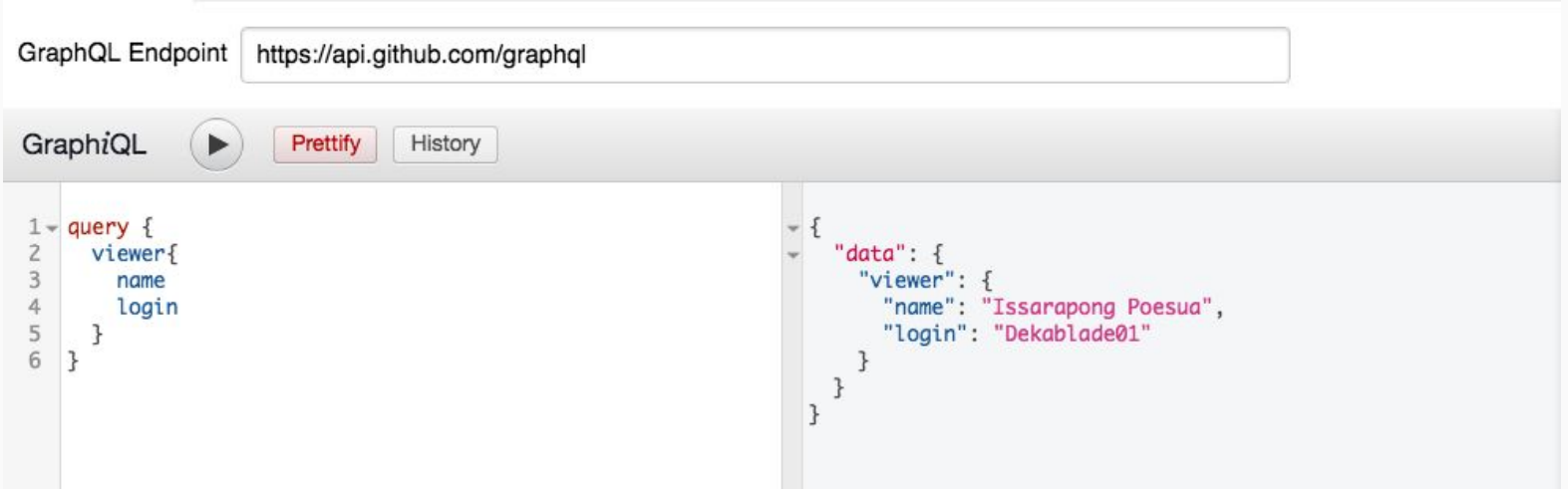
- GitHub Oauth
- GitHub GraphQL V4
- Apollo GraphQL iOS Framework
  - Request
  - Response
  - ViewModels

# Achievements

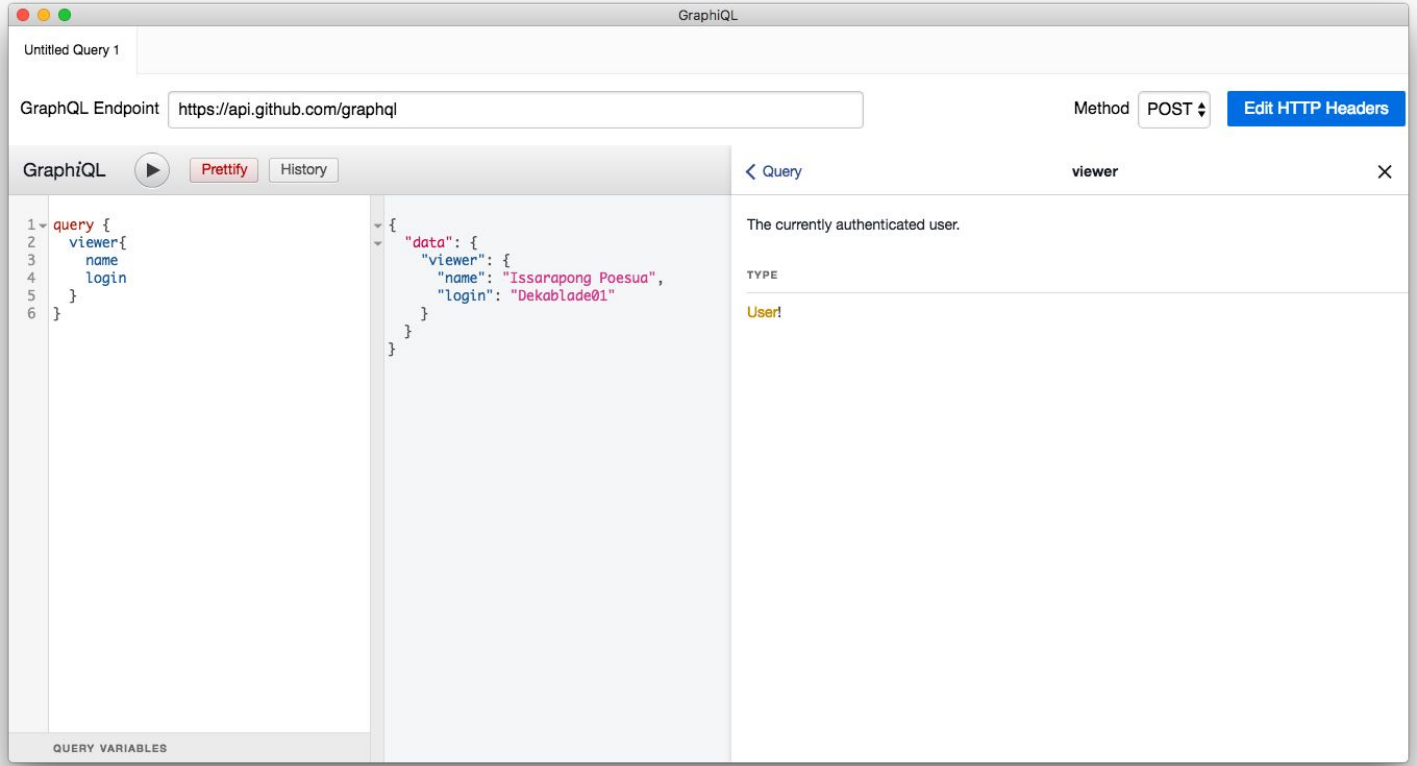
- GitHub Oauth In Signing In
  - Use GitHub Oauth code to get GitHub Personal Access Token
- Current User Profile With Name, Username, Avatar
- List Current User Repositories ( Query Request )
- List Pull Requests in Repository ( Query Request )
- Rename Repository ( Mutation )

# GraphQL

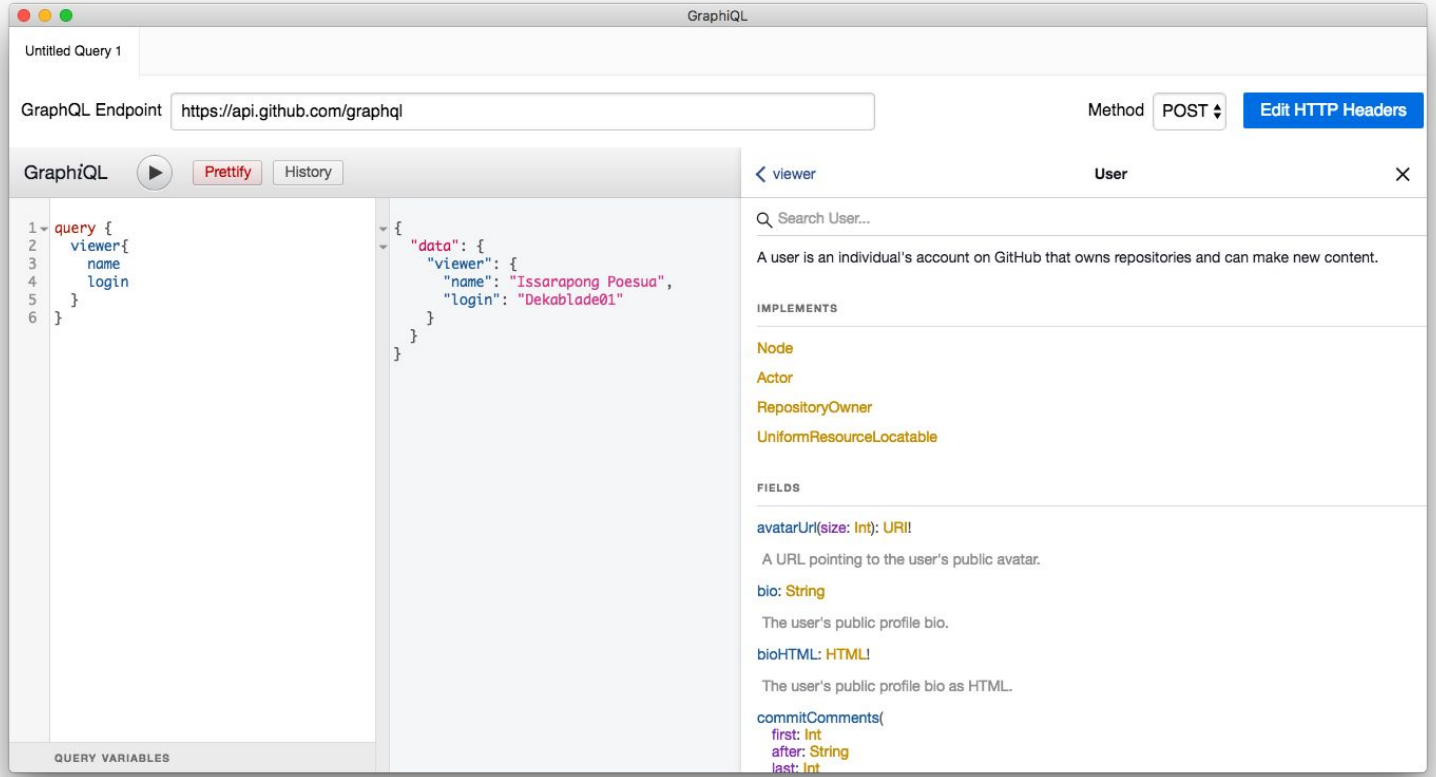
- GraphQL is a query language for APIs and a runtime for fulfilling queries with GraphQL query schema which include in provider document. and single endpoint.
- Client can configure response model as client wish.



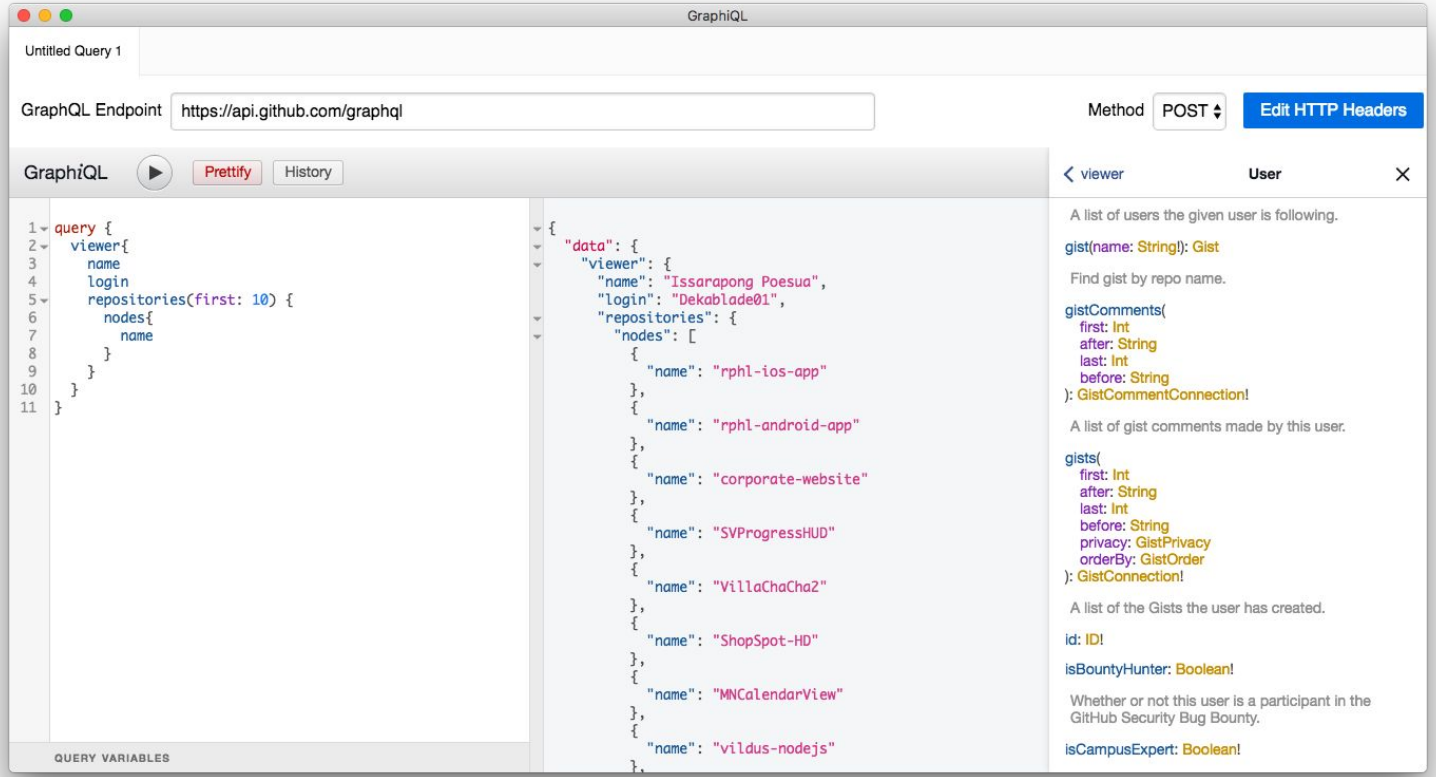
# GraphQL


GraphiQL
Untitled Query 1
GraphQL Endpoint
https://api.github.com/graphql
Method
POST
Edit HTTP Headers
GraphiQL
Prettify
History
query {
viewer{
name
login
}
}
{
"data": {
"viewer": {
"name": "Issarapong Poesua",
"login": "Dekablade01"
}
}
}
QUERY VARIABLES
Query
viewer
The currently authenticated user.
TYPE
User!

# GraphQL


GraphiQL
Untitled Query 1
GraphQL Endpoint
https://api.github.com/graphql
Method
POST
Edit HTTP Headers
GraphiQL
Prettify
History
query {
viewer{
name
login
}
}
{
"data": {
"viewer": {
"name": "Issarapong Poesua",
"login": "Dekablade01"
}
}
}
QUERY VARIABLES
viewer
User
Search User...
A user is an individual's account on GitHub that owns repositories and can make new content.
IMPLEMENTS
Node
Actor
RepositoryOwner
UniformResourceLocatable
FIELDS
avatarUrl(size: Int): URI!
A URL pointing to the user's public avatar.
bio: String
The user's public profile bio.
bioHTML: HTML!
The user's public profile bio as HTML.
commitComments(
first: Int
after: String
last: Int

# GraphQL


GraphiQL
Untitled Query 1
GraphQL Endpoint
https://api.github.com/graphql
Method
POST
Edit HTTP Headers
GraphiQL
Prettify
History
query {
viewer{
name
login
repositories(first: 10) {
nodes{
name
}
}
}
}
{
"data": {
"viewer": {
"name": "Issarapong Poesua",
"login": "Dekablade01",
"repositories": {
"nodes": [
{
"name": "rphl-ios-app"
},
{
"name": "rphl-android-app"
},
{
"name": "corporate-website"
},
{
"name": "SVProgressHUD"
},
{
"name": "VillaChaCha2"
},
{
"name": "ShopSpot-HD"
},
{
"name": "MNCalendarView"
},
{
"name": "vildus-nodejs"
},
QUERY VARIABLES
viewer
User
A list of users the given user is following.
gist(name: String!): Gist
Find gist by repo name.
gistComments(
first: Int
after: String
last: Int
before: String
): GistCommentConnection!
A list of gist comments made by this user.
gists(
first: Int
after: String
last: Int
before: String
privacy: GistPrivacy
orderBy: GistOrder
): GistConnection!
A list of the Gists the user has created.
id: ID!
isBountyHunter: Boolean!
Whether or not this user is a participant in the GitHub Security Bug Bounty.
isCampusExpert: Boolean!

# GraqhQL Schema

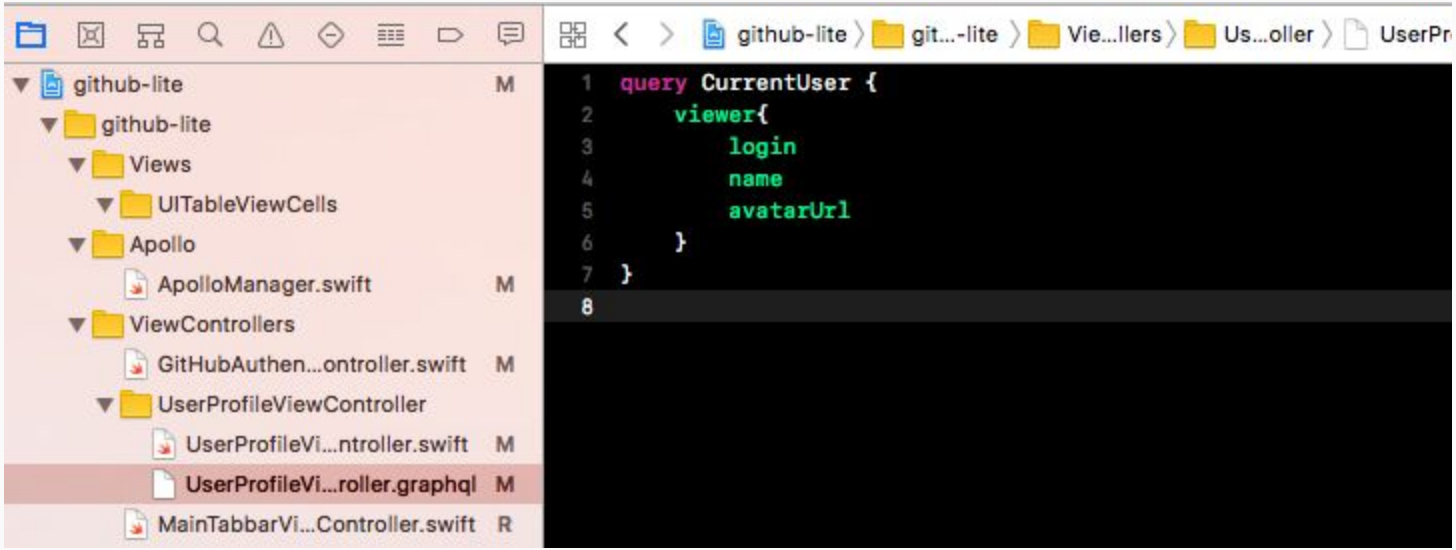
iOS Developer need to download GraqhQL Schema from the API Endpoint

```
`apollo-codegen download-schema __SIMPLE_API_ENDPOINT__ --output schema.json`
```

- Apollo iOS
  - iOS GraphQL framework which allow iOS developer to Query & Mutation to the GraqhQL API
- Apollo-codegen
  - Apollo code generator to let iOS developer to write the query code in .graqhql and generate to API.swift ( requesting code )
  - Apollo-codegen will generate all models you query in .graphql. ( models code )
  - You can make sure that you parse JSON with correct key and all keys
  - `apollo-codegen`

# Write The QueryCode In GraqhQL Query

# Write The QueryCode In GraqhQL Query



# Generating SwiftModel with apollo-codegen

```swift
public final class CurrentUserQuery: GraphQLQuery {
  public static let operationString =
    "query CurrentUser {\n  viewer {\n    __typename\n    login\n    name\n    avatarUrl\n  }\n}"

  public init() {
  }

  public struct Data: GraphQLSelectionSet {
    public static let possibleTypes = ["Query"]

    public static let selections: [GraphQLSelection] = [
      GraphQLField("viewer", type: .nonNull(.object(Viewer.selections))),
    ]

    public var snapshot: Snapshot

    public init(snapshot: Snapshot) {
      self.snapshot = snapshot
    }

    public init(viewer: Viewer) {
      self.init(snapshot: ["__typename": "Query", "viewer": viewer.snapshot])
    }

    /// The currently authenticated user.
    public var viewer: Viewer {
      get {
        return Viewer(snapshot: snapshot["viewer"]! as! Snapshot)
      }
      set {
        snapshot.updateValue(newValue.snapshot, forKey: "viewer")
      }
    }

    public struct Viewer: GraphQLSelectionSet {
      public static let possibleTypes = ["User"]

      public static let selections: [GraphQLSelection] = [
        GraphQLField("__typename", type: .nonNull(.scalar(String.self))),
        GraphQLField("login", type: .nonNull(.scalar(String.self))),
        GraphQLField("name", type: .scalar(String.self)),
        GraphQLField("avatarUrl", type: .nonNull(.scalar(String.self))),
      ]

      public var snapshot: Snapshot

      public init(snapshot: Snapshot) {
        self.snapshot = snapshot
      }
```

# Requesting with Swift

```
private func getUserData() {
    SVProgressHUD.show()
    guard let apollo = ApolloManager.shared.apolloClient else { return }
    apollo.fetch(query: CurrentUserQuery()) { [weak self] (result, error) in
        if (error != nil) {

        }

        if let result = result, let data = result.data {
            self?.setUserInfoToViewController(user: data.viewer)
            SVProgressHUD.dismiss()
        }
    }
}
```

# Requesting with Swift

```
private func getUserData() {
    SVProgressHUD.show()
    guard let apollo = ApolloManager.shared.apolloClient else { return }
    apollo.fetch(query: CurrentUserQuery()) { [weak self] (result, error) in
        if (error != nil) {

        }

        if let result = result, let data = result.data {
            self?.setUserInfoToViewController(user: data.viewer)
            SVProgressHUD.dismiss()
        }
    }
}
```

```
    public init(viewer: Viewer) {
      self.init(snapshot: ["__typename": "Query", "viewer": viewer.snapshot])
    }
```

# Requesting with Swift

```
private func getUserData() {
    SVProgressHUD.show()
    guard let apollo = ApolloManager.shared.apolloClient else { return }
    apollo.fetch(query: CurrentUserQuery()) { [weak self] (result, error) in
        if (error != nil) {

        }

        if let result = result, let data = result.data {
            self?.setUserInfoToViewController(user: data.viewer)
            SVProgressHUD.dismiss()
        }
    }
}
```

```
    public init(viewer: Viewer) {
      self.init(snapshot: ["__typename": "Query", "viewer": viewer.snapshot])
    }
```

# Pro & Cons

- No need to create Model for any GraqlQL Request
- Any Model Structure you want.
- Apollo-codegen is not working properly

  ⇒ Generated invalid API.swift

# Thanks!

Contact Nimbl3

hello@nimbl3.com

399 Sukhumvit Road, Interchange 21
Klongtoey nua, Wattana
Bangkok 10110

20th Floor, Central Tower
28 Queen's Road
Central, Hong Kong

nimbl3.com