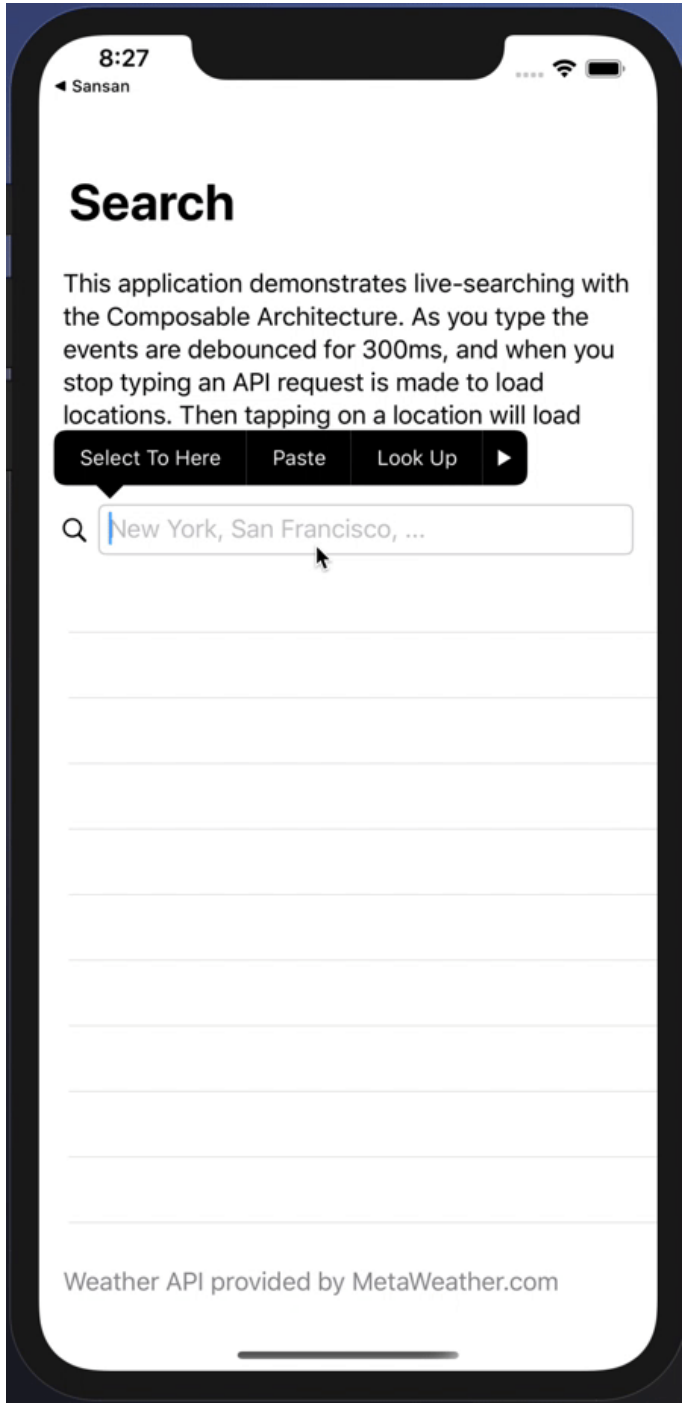
TCA Examples の Search プロジェクトから知る TCA のテストの便利さ

自己紹介

- アイカワ（@kalupas0930）
- 名刺管理サービスを作っている
  新卒 iOS エンジニア
- 函館出身です
- 最近は Flutter, 機械学習の勉強をしてます
- SwiftUI と Combine 最近勉強し始めました

今回紹介する題材

- TCA の Exmaples の Search アプリ
  - 地名を入力する
  - 300ms 何も打たない
  - API Request が飛んで、該当する地名があれば表示される
  - 表示された地名をタップすると、その地域の天気情報が見れる
- Search アプリの Test
  - TCA の テストサポート機能
  - テストを書くのが楽・テスト結果もわかりやすい


8:27
Sansan
Search
This application demonstrates live-searching with the Composable Architecture. As you type the events are debounced for 300ms, and when you stop typing an API request is made to load locations. Then tapping on a location will load
Select To Here
Paste
Look Up
New York, San Francisco, ...
Weather API provided by MetaWeather.com

ファイルツリー

- 全体のファイルツリー

```
\Search
|---\Search.xcodeproj
|---\Search // 今回は主にここと
|---\SearchTests // ここを紹介します
|--- README.md
```

まずは Search 自体について

Search のファイルツリー

```
\Search
|--- SearchView.swift // TCA の色々な要素* が詰め込まれています
|--- ActivityIndicator.swift // ただの ActivityIndicator
|--- SceneDelegate.swift // SearchView の初期化
|--- WeatherClient.swift // Model と API client の実装
|--- Info.plist
|--- Assets.xcassets
```

TCA の色々な要素*： State, Action, Environment, Reducer, View

WeatherClient ∅ models

```
struct Location: Decodable, Equatable {
  var id: Int
  var title: String
}
```

WeatherClient ∅ models

```
struct LocationWeather: Decodable, Equatable {
  var consolidatedWeather: [ConsolidatedWeather]
  var id: Int

  struct ConsolidatedWeather: Decodable, Equatable {
    var applicableDate: Date
    var maxTemp: Double
    var minTemp: Double
    var theTemp: Double
    var weatherStateName: String?
  }
}
```

WeatherClient の API client interface

```
struct WeatherClient {
  var searchLocation: (String) -> Effect<[Location], Failure>
  var weather: (Int) -> Effect<LocationWeather, Failure>

  struct Failure: Error, Equatable {}
}
```

Effectの説明〜〜〜〜〜〜〜〜〜

WeatherClient の API implementation

```
extension WeatherClient {
  static let live = WeatherClient(
    searchLocation: { query in
      // ...
    },
    weather: { id in
      // ...
    })
}
```

テスト用に利用することになる Mock API implementation も
ありますがそちらは後ほど紹介します

WeatherClient ∅ API implementation

```
extension WeatherClient {
  static let live = WeatherClient(
    searchLocation: { query in
      var components = URLComponents(string: "https://www.metaweather.com/api/location/search")!
      components.queryItems = [URLQueryItem(name: "query", value: query)]

      return URLSession.shared.dataTaskPublisher(for: components.url!)
        .map { data, _ in data }
        .decode(type: [Location].self, decoder: jsonDecoder)
        .mapError { _ in Failure() }
        .eraseToEffect()
    },
    weather: { id in
      // ...
    })
}
```

WeatherClient ∅ API implementation

```
extension WeatherClient {
  static let live = WeatherClient(
    searchLocation: { query in
      // ...
    },
    weather: { id in
      let url = URL(string: "https://www.metaweather.com/api/location/\(id)")!

      return URLSession.shared.dataTaskPublisher(for: url)
        .map { data, _ in data }
        .decode(type: LocationWeather.self, decoder: jsonDecoder)
        .mapError { _ in Failure() }
        .eraseToEffect()
    })
}
```

SearchView ∅ State, Action

```
struct SearchState: Equatable {
  var locations: [Location] = []
  var locationWeather: LocationWeather?
  var locationWeatherRequestInFlight: Location?
  var searchQuery = ""
}

enum SearchAction: Equatable {
  case locationsResponse(Result<[Location], WeatherClient.Failure>)
  case locationTapped(Location)
  case locationWeatherResponse(Result<LocationWeather, WeatherClient.Failure>)
  case searchQueryChanged(String)
}
```

SearchView ∂ Environment

```
struct SearchEnvironment {
  var weatherClient: WeatherClient
  var mainQueue: AnySchedulerOf<DispatchQueue>
}
```

SearchView ∅ Reducer

```
let searchReducer = Reducer<SearchState, SearchAction, SearchEnvironment> {
  state, action, environment in
  switch action {
  case .locationsResponse(.failure):
  case let .locationsResponse(.success(response)):
  case let .locationTapped(location):
  case let .searchQueryChanged(query):
  case let .locationWeatherResponse(.failure(locationWeather)):
  case let .locationWeatherResponse(.success(locationWeather)):
  }
}
```