모바일 프레임워크

장 단점 및 비교 정리

Commax 인턴

오 수찬

목차

서론 1

연구분야 1.1

동기 1.2

목적 1.3

관련연구 2

본론: 모바일 프레임워크별 장단점 및 특징 3

모바일 프레임워크의 종류 및 사용량 3.1

모바일 프레임워크별 장단점 및 특징 3.2

결론 4

참고 자료 5

1. 서론

1.1 연구 분야

* 모바일 프레임워크별 제공하는 기능의 특징과 장단점 및 차이점.

1.2 동기

* 프레임워크의 사용은 일반화 되어있다. 웹, 모바일 등 다양한 분야에 다양한 프레임워크가 적용되어 있다. 이처럼 프레임워크가 넓게 사용되는 주 골자는 재사용성이 향상되고 유지보수 비용이 감소된다는 점이다. 특히 요즈음 사용량이 증가하여 40억명 이상의 유저가 사용하는 모바일 디바이스의 경우 다양한 프레임워크가 범람하고 있다. 그렇기에 그 종류와 특징을 파악하여 유연한 선택을 지원하고자 하였다.

1.3 목적

* 모바일 프레임워크의 종류와 특징 및 구조를 파악.
* 모바일 프레임워크별 장단점을 파악하고 어떤 서비스에 적합한지 파악.

2. 관련연구

우선 프레임워크란 무엇일까? 프레임워크란 소프트웨어의 구체적인 부분에 해당하는 설계와 구현의 재사용성을 위해서 일련의 협업화 된 형태로 클래스 들을 제공하는 것. 시스템 개발에 일관적인 틀을 주어서 개발된 SW에 통일성을 보장해 줌. 덕분에 개발 과정이 간단해지지만 자유가 제한됨.

핵심 로직에만 집중할 수 있도록 기본적인 기능(라이브러리)을 제공해 줌과 동시에 틀을 갖추고 있어 프레임워크가 정의한 틀에 맞춰 기능을 구현해야 함.

쉽게 말하자면 뼈대. 라이브러리\*와 차이점은 프레임워크는 라이브러리에 기본 틀(설계된 구조와 그에 따른 규칙)에 더해 작업을 호출할 수 있는 제어권이 추가된 개념이라고 생각하면 된다.

장점은 대표적으로 효율적이라는 점이 있다. 프레임워크를 사용하면 그렇지 않았을 때에 대비해 개발시간, 비용 절약, 생산성 증대의 효과를 낳는다. 또한 퀄리티가 향상된다. 버그 발생 가능성을 처리해줄 뿐 아니라 다수 사용자에 의한 검증을 통해 구성되어 있기 때문에 일관된 품질을 보장한다. 마지막으로 유지보수가 용이하다. 담당자가 바뀌어도 코드가 체계적이기 때문에 유지보수가 안정적이다.

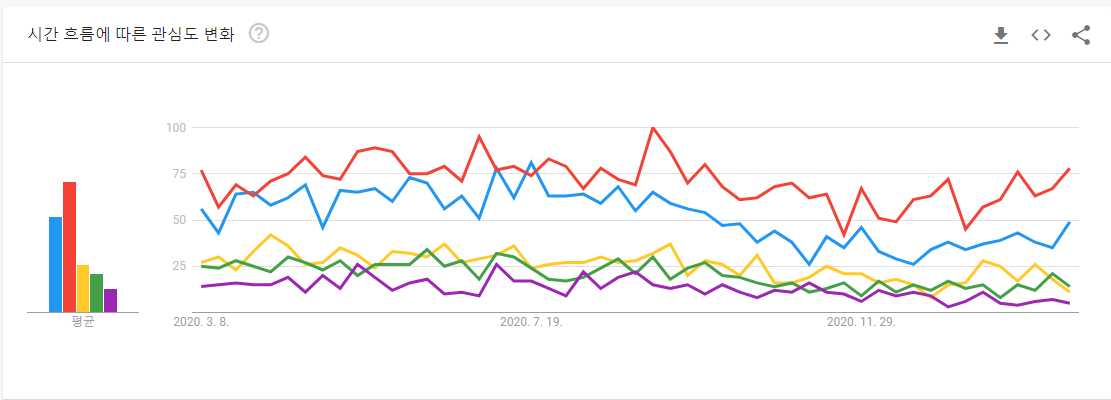
반면 단점으로는 학습시간이 길다는 점이 있다. 프레임워크를 처음 사용하는 경우 코드를 습득하고 이해하는데 시간이 걸린다. 또한 제약 사항이 존재하기 때문에 자유롭고 유연한 개발에 한계가 있다.

이러한 단점이 존재함에도 불구하고 프레임워크가 주는 이점이 많기 때문에 오늘날 프레임워크는 모바일 분야 이외에도 다양한 분야에 널리 사용되고 있다.

3. 본론: 모바일 프레임워크별 장단점 및 특징

**3.1 모바일 프레임워크의 종류 및 사용량**

모바일 프레임워크는 대부분 오픈소스로 제공된다. 고로 시장 점유율에 관한 자료가 존재하지 않는다. 그렇기 때문에 검색양과 사용자의 관심의 정도로 사용량을 대신 유추해 볼 수 있다.



위의 자료는 구글 트렌드에서 제공하는 검색 빈도자료다. 가장 많은 검색이 있을 때를 100으로 잡고 비교한 자료. 인터넷, 통신 카테고리의 검색양을 비교한 자료로 붉은색은 Flutter, 푸른색은 React, 노란색은 Xamarin, 녹색은 Ionic, 마지막으로 보라색은 Cordova를 나타낸다.

2020년 12월 기준으로 Flutter가 가장 큰 관심을 받는 모바일 프레임워크로 나타난다. 20년 중반에 비해서 관심도가 약간 낮아진 모습을 보이지만 그래도 꾸준히 가장 높은 관심을 받는 프레임워크이다. React가 그 뒤를 잇는 모바일 프레임워크이다. 중반에 비해 관심도가 떨어진 모습을 보이는 것은 동일하나 여전히 아래 3개의 프레임워크에 비해서 높은 관심을 받고 있다. 나머지 3개의 모바일 프레임워크 Xamarin, Ionic Cordova의 경우 비슷한 관심을 받고 있다고 할 수 있다. 3개의 프레임워크 모두 관심이 늘거나 줄지 않고 꾸준히 어느정도 선에서 유지되고 있다는 점에서 확고한 사용자층을 보유하고 있다고 볼 수 있다.

**3.2 모바일 프레임워크별 장단점 및 특징**

현재 가장 많은 관심을 받고 있는 위의 5가지 모바일 프레임워크의 장단점 및 특징을 파악하여 개발하고자 하는 앱의 성격에 맞는 모바일 프레임워크를 선택하는 현명한 기준에 대해 알아보자.

3.2.1 Flutter

구글에서 선보인 모바일 프레임워크로 네이티브 앱에서 웹의 특성을 포함할 수 있는 특징을 가지고 있다. Dart라는 자체 개발한 언어를 사용한다는 점이 특징적인 점이다. 네이티브 수준의 빠른 퍼포먼스와 뛰어난 내장 UI가 장점이지만 아직 제작된 라이브러리 수가 부족하고 정리된 학습자료가 부족하다는 단점 또한 존재한다. 라이브러리나 학습 자료의 문제는 사용자가 늘어나면 개선될 여지가 충분한 부분이다. 이외에도 아래 나열할 여러가지 특징들을 가진다.

1. 핫 리로딩기능

Flutter가 제공하는 편의성 중 하나로 컴파일 하지 않고 코드에 변동사항을 미리 볼 수 있는 기능을 말한다. 덕분에 개발자들은 컴파일 시간을 기다리지 않고 UI를 빠르게 구축하거나 코드를 실험해볼 수 있다.

1. 내장 위젯을 이용한 UI

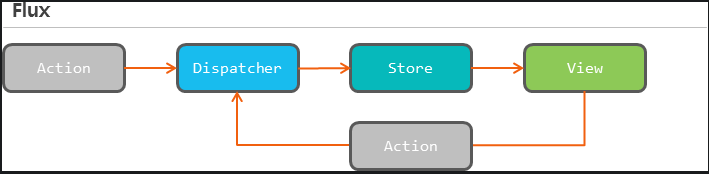
내장 위젯을 사용하기 때문에 플랫폼 의존도가 현저히 낮아지는 장점을 낳는다. 뿐만 아니라 재사용성까지 높아지는 장점을 가진다. 반면 아직 API의 개수가 모자라서 표현하고자 하는 UI를 직접 만들어야 되는 경우가 발생한다는 단점 또한 존재한다.

1. OEM의존성이 없다

Webview나 OEM위젯을 사용하지 않는다. 그 대신 고성능의 자체적 엔진을 사용해서 위젯을 그리는 방식을 사용한다.

3.2.2 React

페이스북과 인스타그램에서 UX향상을 목적으로 만든 라이브러리(정확히는 모바일 프레임워크가 아니다). React는 자바스크립트 UI 라이브러리이다. 단지 UI생성의 작업만 하며 캡슐화를 통해 재사용성을 높여준다. 이렇게 단순한 형태이기 때문에 여러 자바 스크립트 프레임워크와 같이 쓰일 수 있다. 이러한 형태를 가지게 된 이유는 개발 측에서 프레임워크에 많이 사용되는 MVC패턴은 확장성이 떨어지고 큰 프로젝트에 적합하지 않다는 결론을 내리고 Flux라는 새로운 시스템 아키텍처를 적용해서 라이브러리 형태로 개발하게 되었다.



Flux는 아래와 같은 구조를 가진다. Store부분이 MVC의 Model의 역할을 하고 Dispatcher가 Control, View가 View의 역할을 한다. 기존 MVC의 경우 Control이 Model에 접근해 수정, 삽입, 삭제의 과정을 거친 뒤 Model과 View를 더해 반환했던 것과 달리 단방향으로 Control이 Model을 업데이트 하면 View에 적용되는 방식이다. 또 View를 통해 사용자 입력이 들어오면 Control을 거쳐 Model이 수정되는 방식을 가진다.

이외에도 아래 나열된 특징을 가진다.

1. 템플릿을 사용하지 않는다

자바스크립트라는 강력한 언어를 사용하기 때문에 템플릿을 사용하지 않을 수 있다. 대신 JSX(XML과 비슷한 자바 스크립트 확장 문법)를 사용할 수 있다. 덕분에 자바스크립트에서 HTML과 자바스크립트 변수 및 속성 값을 사용할 수 있게 된다. 또한 테스트 코드는 컴파일 하지 않은 상태에서 동작을 확인할 수 있다. 이런 방식은 로직과 뷰를 분리하여 확장 및 유지보수가 용이 해진다.

1. Virtual DOM

가상 DOM을 사용한다. 이는 DOM Tree에 의해 관리되어지는데 변동사항이 생겼을 때 페이지 전체를 다시 그리는 것이 아니라 변경이 생긴 부분만 다시 그릴 수 있게 해준다. 덕분에 반응속도가 빠르며 새로 고침 없이 변경내용을 적용할 수 있다.

1. 데이터가 단방향으로 움직인다

데이터 흐름이 단방향이 되면 데이터 흐름을 이해하기 쉬워진다. 단기적으로 작은 프로젝트의 경우 이러한 장점은 크게 다가오지 않을 수 있으나 프로젝트가 커질수록 개발 및 확장비용을 절감해 준다.

3.2.3 Xamarin

엄밀히 말하면 프레임워크는 아니고 마이크로소프트 소유의 하이브리드 모바일 앱을 만들기 위한 오픈소스 플랫폼이다. .NET과 C#을 이용하며 각종 라이브러리와 API가 제공된다. 하이브리드 플랫폼 답게 안드로이드, iOS, Windows등 다양한 환경에서 구동이 가능한 앱을 개발할 수 있다. 주로 미국에서 사용되며 6만명이 넘는 개발자와 사용자가 존재한다. MS사가 제공하는 만큼 고객지원이 뛰어나며 자료가 풍부한 점도 장점이다. 마지막으로 이클립스와 같은 현대적 IDE를 사용하여 개발이 가능하다는 점도 사용자에게 큰 편의성으로 다가온다.

이외의 특징은 아래와 같다.

1. 기본 플랫폼 SDK들에 대해 바인딩 되어있다

기본 플랫폼 SDK들에 바인딩 되어있기 때문에 탐색 및 사용이 용이하고 런타임 오류가 줄어든다.

1. 네이티브 급 성능

Xamarin을 이용해 개발 시 네이티브 앱으로 빌드 되기 때문에 성능이 안드로이드나 iOS의 네이티브 앱 급 성능을 보여준다. 또 플랫폼 테스팅과 앱 성능 관찰이 용이하다.

1. 오픈 소스 라이브러리 사용 제한

플랫폼에 의해 제공되는 컴포넌트와 .NET 오픈소스 리소스만 사용이 가능하다는 단점 이 존재한다. 하지만 오늘날 Xamarin 라이브러리에도 수천개의 UI, 차트, 그래프 등이 존재하기 때문에 큰 약점이 되지 않을 수 있다.

3.2.4 Ionic

하이브리드 어플리케이션을 쉽게 만들 수 있는 프레임워크로 HTML5 API를 사용할 수 있는 컴포넌트와 크로스 플랫폼 빌드가 가능한 Cordova가 기반이 되는 프레임워크다. Cordova환경을 제공하고 Cordova플러그인을 사용할 수 있기 때문에 Cordova를 사용하던 개발자들은 더 쉽게 배울 수 있다.

이외의 특징은 아래와 같다.

1. AngularJS를 기반으로 만들어졌다

덕분에 SPA를 MVC나 MVVM패턴으로 개발할 수 있다.

1. 네이티브에 가까운 UI컴포넌트를 제공한다

네이티브는 아니지만 네이티브와 유사한 UI컴포넌트를 제공하기 때문에 사용자에게 친숙하게 다가갈 수 있다.

1. HTML로 UI를 만들고 JavaScript로 데이터 처리가 가능하다

이러한 방식이 주는 최고의 장점은 웹으로 작성된 코드를 거의 그대로 사용할 수 있다는 점이다.

1. 네이티브 앱에 비해서 성능이 떨어진다

네이티브로 동작하거나 가상환경 위에서 동작하는 하이브리드 앱과 달리 성능이 떨어진다는 단점이 존재한다.

3.2.5 Cordova

Apache 재단에서 지원하는 하이브리드 앱 개발 프레임워크이다. 안드로이드, iOS, Window phone등 8개 플랫폼을 동시 지원할 뿐만 아니라 PhoneGap, Ionic등의 기반이 된 프레임워크이다. Node.js플랫폼을 기반으로 하고 있으며 HTML5, CSS3와 같은 웹 표준 또한 지원한다.

이외엔 아래와 같은 특징을 가진다.

1. 쉽게 배울 수 있다

오래된 하이브리드 앱 프레임워크인 만큼 다양한 커뮤니티가 존재하고 많은 사용자가 존재하기 때문에 관련 정보도 많은 편이고 또 웹 기술을 기반으로 하기 때문에 기존 웹 개발을 하던 사람이라면 쉽게 배울 수 있다.

1. 네이티브 API에 접근이 가능하다

카메라, 연락처 등 기기에서 제공하는 기본 API에 쉽게 접근하여 사용할 수 있다.

1. 네이티브 앱에 비해서 성능이 떨어진다
2. 네이티브로 동작하거나 가상환경 위에서 동작하는 하이브리드 앱과 달리 성능이 떨어진다는 단점이 존재한다.
3. 모든 플랫폼에서 동일하게 동작하지 않는다

최고의 단점 중 하나 같은 코드로 모든 플랫폼에서 동일한 동작이 보장되지 않는다. UI뿐만 아니라 동작부분도 플랫폼 별로 다를 수 있어 신중하게 코드를 작성해야 한다.

위의 프레임워크들로 만들어진 대표적인 앱들은 아래와 같다. 자신의 UX에 비추어 각 모바일 프레임워크로 제작된 대표적인 앱들을 살펴보면 각 프레임워크의 특징과 장단점에 대해서 좀더 직관적인 인식을 가질 수 있다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 프레임워크 | Flutter | React | Xamarin | Ionic | Cordova |
| 어플리케이션 | 알리바바  구글애드  리플랙틀리 | 페이스북  인스타그램  에어비앤비 | 애저  유피에스  알래스카항공 | 마켓워치  스워킷  인스턴트 팟 | 위키피디아  헬스 탭  트립케이스 |

4. 결론.

전세계적으로 40억명이 넘는 모바일 디바이스 사용자 존재한다. 개인이나 기업이 더 많은 고객을 확보하고 더 큰 이득을 추구하기 위해서는 모바일 사용자를 무시할 수 없게 되었다는 뜻이다. 그렇기 때문에 모바일 사용자를 위한 개발은 선택이 아닌 필수가 됐다.

하지만 모바일 플랫폼은 너무나 다양하다 대표적으로 안드로이드, iOS, Window등이 대표적이다. 때문에 모바일 개발비용을 줄이는 것은 모바일 개발을 위해서는 필수적인 과제가 됐다. 이러한 문제를 해결해 줄 수 있는 방안이 바로 모바일 프레임워크이다.

하지만 모바일 프레임워크 또한 너무 많은 종류와 특성이 존재한다. 하지만 오늘날은 주로 하이브리드 앱 개발을 위한 프레임 워크나 플랫폼이 주류가 되는 분위기다.

네이티브 앱을 개발하는 경우 성능은 더 뛰어나고 플랫폼이 제공하는 API에 접근이 편하다는 장점이 있으나 플랫폼 별로 따로 개발을 진행해야 하고 구조나 새로운 언어를 배워야 될 수고까지 존재한다. 반면 하이브리드 앱을 개발하는 경우 한번의 개발로 수많은 플랫폼에 적용될 수 있고 또 본인이 익숙한 개발 환경을 사용할 수도 있으며 프레임워크의 종류에 따라 사용하는지에 따라서 성능도 네이티브와 유사할 수 있다.

하이브리드 모바일 프레임워크를 선택할 시에는 본인이 개발하려는 앱의 특성에 따라 성능이 중요한지 아니면 쉽게 배울 수 있고 많은 자료와 라이브러리가 존재하여 학습이 쉬운 프레임워크가 중요한지 반응속도가 중요한지 기존 코드를 최대한 많이 재활용할 수 있는 방식이 중요한지 등의 기준에 따라서 위에 정리된 자료들을 참고하여 결정할 수 있겠다.

5. 참고자료.

- 모바일 프레임워크 사용량 관련

<https://trends.google.com/trends/explore?cat=13&q=React%20Native,Flutter,Xamarin,Ionic,Cordova>

https://brunch.co.kr/@jowlee/101

* 모바일 프레임워크 특징 관련

<https://m.blog.naver.com/dsz08082/221842323069>

<https://brunch.co.kr/@jowlee/101>

http://blog.wishket.com/2020%EB%85%84-%EB%8C%80%EC%9C%A0%ED%96%89%EC%A4%91%EC%9D%B8-%EB%AA%A8%EB%B0%94%EC%9D%BC-%EC%95%B1%EA%B0%9C%EB%B0%9C-%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC-7%EA%B0%80%EC%A7%80/

<https://coozplz.me/2016/01/05/cordova-pros-cons/>

<https://velog.io/@kimtaeeeny/Ionic-%EC%95%84%EC%9D%B4%EC%98%A4%EB%8B%89-vs-React-Native-%EB%A6%AC%EC%95%A1%ED%8A%B8-%EB%84%A4%EC%9D%B4%ED%8B%B0%EB%B8%8C-feat.-%ED%95%98%EC%9D%B4%EB%B8%8C%EB%A6%AC%EB%93%9C-%EC%95%B1-FE-study12>

<https://ithub.tistory.com/70>

<https://blog.gaerae.com/2016/04/hello-react.html>

https://seokjun.kim/time-to-stop-react/