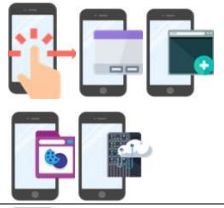
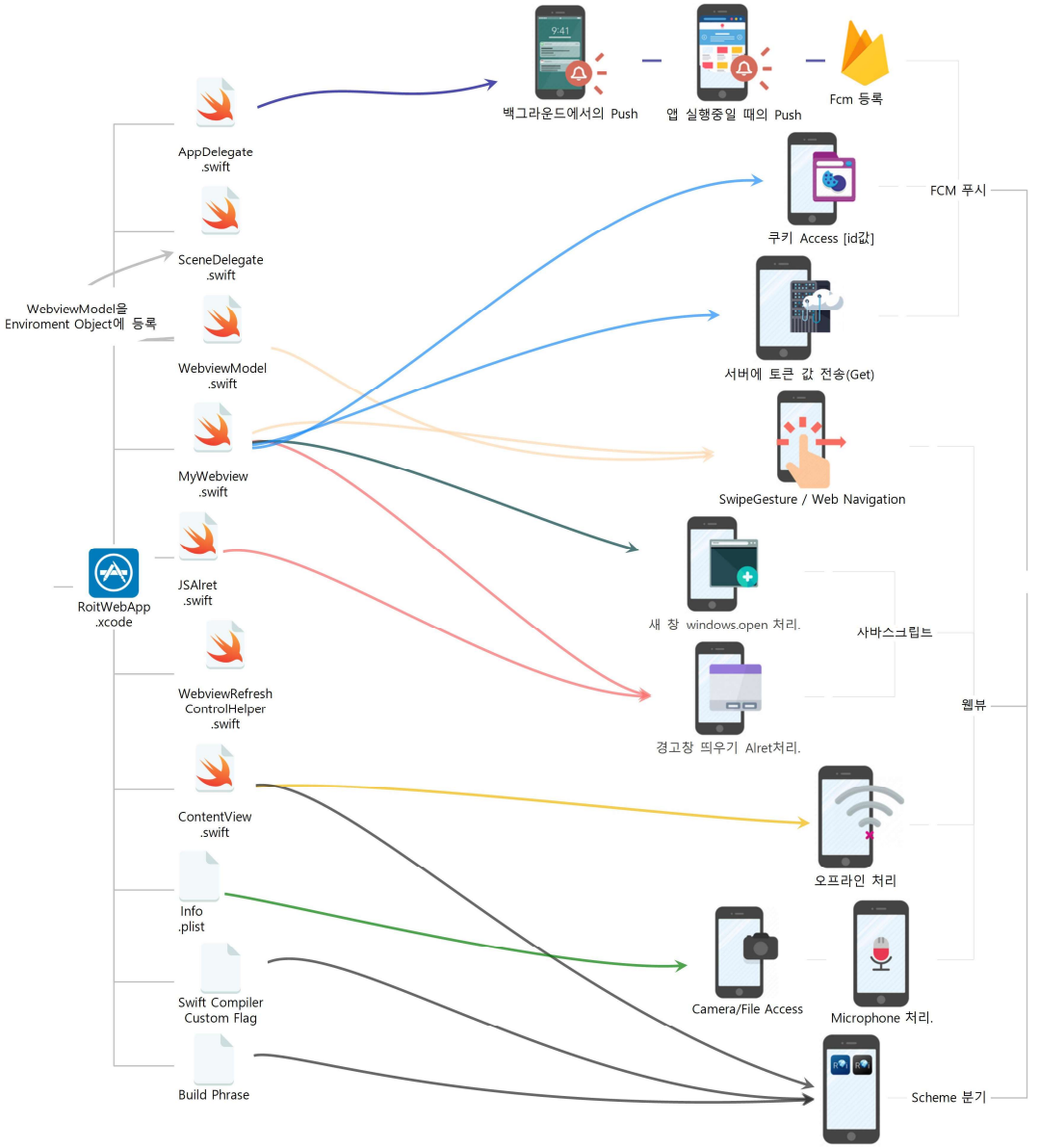
프로젝트 구조

파일명	내용	
Appdelegate .swift (기본 내장 파일)	파이어베이스 등록. 파이어베이스 토큰 정보 값 가진다. 백그라운드 때 푸시이벤트 리시버 앱 실행 중 푸시이벤트 리시버가 있다.	
ScenDelegate .swift (기본 내장 파일)	여기서 webviewModel을 환경변수(enviromentObject)에 등록하면, 프로젝트 전역에서 호출가능하다.	
ContenView .swift (기본 내장 파일)	UI화면 창이다. Custom Flag 설정된 Debug(bool)로 url 설정, 오프라인 때의 UI설정 등을 한다.	
MyWebview .swift	SwipeGesture처리를 한다. 자바스크립트 Alert처리를 한다. 웹 페이지의 캐시 정보를 가져온다. DB서버에 FCM 토큰 정보를 전송한다. -추후 규모가 커지면 get/post통신에 대한 기능이 분리가 필요할 수 있다.	
WebviewModel .swift	webview의 이벤트를 받아 처리한다. Back, Forward, Refresh, url 정보 변경 등이 설정되어있다.	
Webview Refresh ControlHelper .swift	웹뷰 리프레시 이벤트를 받아 처리해준다.	
JsAlret .swift	자바스크립트 알람을 받아 처리해준다.	
info .plist	카메라 마이크 파일저장 권한설정을 해준다.	
RoitWebApp .xcode	Swift Compiler Custom Flags - Build Phrase -	



1. 웹뷰

가. SwipeGesture / Web Navigation

0

Ios SwipeGesture에 따라

웹페이지의 Navigation을 제어하는 기능.

새로고침 아래로

좌우Swipe 앞 뒤로가기

```
import Foundation
import UIKit

class WebViewRefreshControlHelper{

    var refreshControl : UIRefreshControl?
    var viewModel : WebViewModel?

    @objc func didRefresh(){
        print("WebViewRefreshControlHelper - didRefresh() called")
        guard let refreshControl = refreshControl,
        let viewModel = viewModel else {
            print ("refreshControl, viewModel이 없다.")
            return
        }

        DispatchQueue.main.asyncAfter(deadline: .now()+0.8, execute:{
            print("리프레시 액션이 들어왔다.")
            //리프레시 하라고 알려주기.
            viewModel.webNavigationSubject.send(.REFRESH)
            refreshControl.endRefreshing()
        })
    }
}
```

```
import Foundation
import Combine

typealias Web_Nav = WebViewModel.NAVIGATION

class WebViewModel : ObservableObject{

    var LoginCookie_Checked=false;
    enum NAVIGATION {
        case BACK,FORWARD,REFRESH
    }

    var webNavigationSubject = PassthroughSubject<Web_Nav,Never>()
    var jsAlertEvent = PassthroughSubject<JsAlert, Never>()
    var webSiteTitleSubject = PassthroughSubject<String, Never>()
}
```

나. 자바스크립트

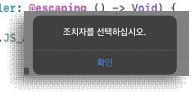
1) 경고창 띄우기 Alret처리.

웹상에서의 javascript 이벤트 중 alert창은 웹뷰에서 동작하지 않는다.

따라서 javascript alert 이벤트를 발생시

Native에서 해당 창을 재발생시켜줘야한다.

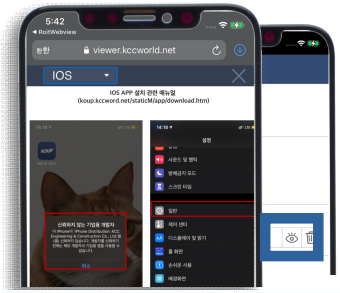
```
func webView(_ webView: WKWebView, runJavaScriptAlertPanelWithMessage message: String,
             initiatedByFrame frame: WKFrameInfo, completionHandler: @escaping () -> Void) {
    print("webView runJavaScriptAlertPanelWithMessage")
    self.myWebView.viewModel.jsAlertEvent.send(JsAlert(message, .JS_
    completionHandler()
}
```



2) 새 창 windows.open 처리.

웹의 특정url을 새 팝업 또는 탭에서 생성하는 명령(window.open)은 웹뷰에서 동작하지 않는다.

이 경우 새 url을 호출해주는 것으로 해결할 수 있다.




```
//새창열기 javascript window.open() 처리.
func webView(_ webView: WKWebView, createWebViewWith configuration: WKWebViewConfiguration,
             for navigationAction: WKNavigationAction, windowFeatures: WKWindowFeatures) -> WKWebView? {

    let loadUrl : String = navigationAction.request.url!.absoluteString
    if (loadUrl.contains("https://") || loadUrl.contains("http://")) {
        if let aString = URL(string:(navigationAction.request.url?.absoluteString )!) {
            UIApplication.shared.open(aString, options:[:], completionHandler: { success in
            })
        }
    } else {
        print("else"+loadUrl)
    }
    return nil
}
```

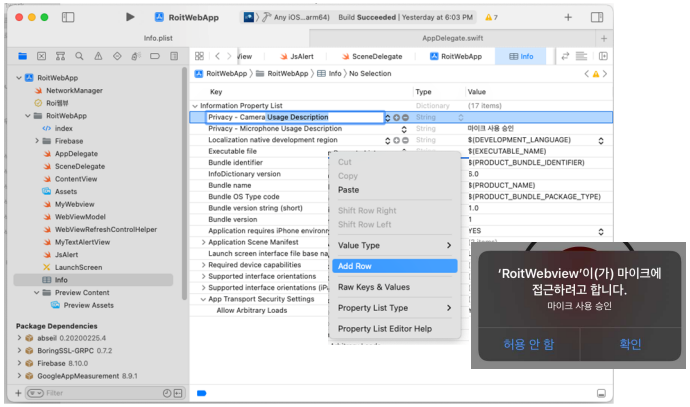
다. 오프라인 처리

기본적인 HTML 페이지는 offline 환경일 때,
에러 페이지로 표시된다.
ServiceWorker.js에서 각 파일을 캐시에 등록하면
offlline일 때도 저장된 웹 화면이 디스플레이 된다.
react나 vue같은 프론트엔드 라이브러리에서 기본적으로 지원된다.
 이 방식으로 처리할 경우 offline을 대비한 별도의 HTML페이지를 만들 필요가 없고, sqlite와 같은 내장DB와 연결하는 작업도 필요가 없다. 코드가 경량화 된다.

1) Microphone/Camera/File Access

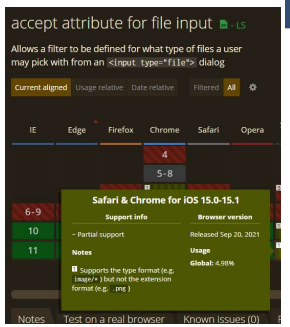
Info.plist 파일에서 권한을 넣고 상세설명 문자열을 지정해주면 된다.
앱 실행 때 권한승인여부 창을 띄우지 않더라도,
해당 미디어(녹음/카메라)를 사용시 권한승인창이 호출된다.

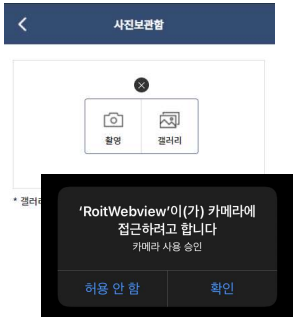
Privacy- Camera Usage Description
Privacy- Microphone Usage Description



라. Camera/File Access

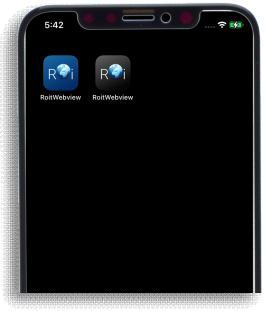
Ios safari는 <input file=type>를 지원한다.
같은 webkit을 사용하는 wkwebview도 지원된다.
별도의 Native 코드없이
단 input file type 의 지정에서 "img/" 와 같이 종류(사진/파일)를 지정할 수 있다.
파일 확장자를 fix할 수는 없다.

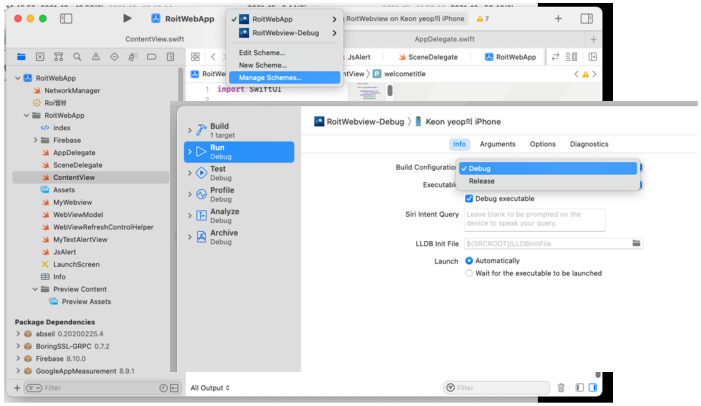


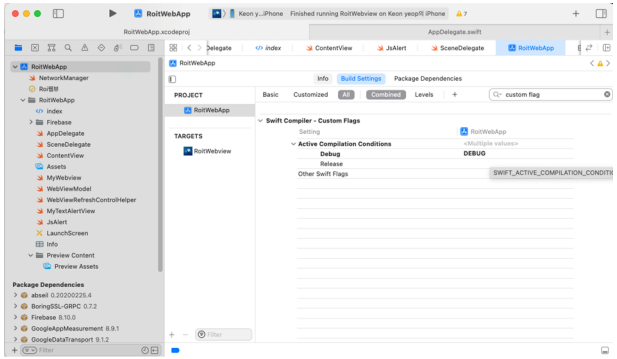


2. Scheme 분기(개발/상용)

*상세한 설명필요
Ios에서는 개발버전/상용버전을 Scheme으로 구분한다.
(Android Build Flavor)
-유료/프로/베타 버전 등 모두 여기서 등록한다.
- 각 버전마다 번들ID, 앱 이름(Display Name), 앱 아이콘(App Icon)을 다르게 지정할 수 있으며,
- Custom Flag에 전처리명령어를 등록하면 코드에서 그 버전에 따라 수정할 수 있다.







```
#if DEBUG
    @State var url : String = "http://koupdev.kccworld.net"
    @State var welcometitle : String = "개발용 웹 페이지"
#else
    @State var url : String = "https://koup.kccworld.net"
    @State var welcometitle : String = "운영 웹 페이지"
#endif

    var body: some View {

        ZStack{ //좌에서 우로 UI 배치
            VStack{ //위에서 아래로 UI 배치
                MyWebview(urlToLoad: url) //웹뷰
                #if DEBUG //디버그 모드일때만, 개발용 표시하기.
                    Text(welcometitle)
                    .font(.largeTitle)
                    .onAppear {}
                #endif
```

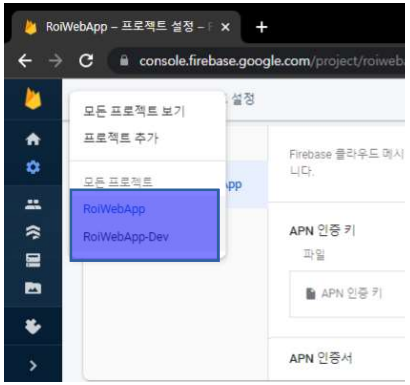
이 웹뷰앱에서는 최초설치 페이지를 위와같이 지정했다.
contentview.swift의 body에서 control들이 배치되는데
Debug 여부에 따라 생성을 달리할 수 있다.

3. FCM 푸시

가. FCM -

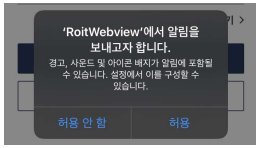
빌드 분기 넣기

빌드 버전(개발/상용)에 따른 푸시메시지를 따로 만든다.
파이어베이스 동일계정 내에 프로젝트를 중복 생성.
(그래야 APi키를 받을 수 있다.)



나. Fcm 등록

```
//fcm 등록 토큰을 받았을때.
func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
    token = fcmToken!
    print("AppDelegate - Firebase registration token: \(String(describing: fcmToken))")
}
```



Appdelegate - FCM 등록. 토큰(Push고유 값)이 발급된다.

다. 앱 실행중일 때의 Push

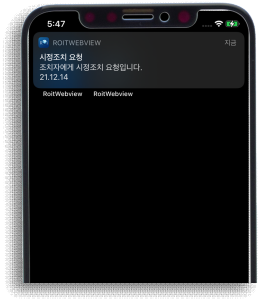
Push 메시지는
앱실행 중일 때도 발생하고,
앱을 꺼두었을 때도 발생한다.
아래의 코드에서 구분되어 처리된다.

1) 앱 실행중일 때의 Push

AppDelegate내의 ----

```
//푸시메시지가 앱이 켜진상태에서 나올때.
func userNotificationCenter(_ center: UNUserNotificationCenter, will
    notification: UNNotification, withCompletionHandler completionHa
    @escaping (UNNotificationPresentationOptions) -> Void) {
      let userInfo = notification.request.content.userInfo
    completionHandler([.banner, .sound, .badge])
    print("푸시메시지가 앱이 켜진상태에서 받았을때")
}
```

2) 백그라운드에서의 Push



AppDelegate내의 ----

```
//푸시메시지를 받았을 때.
func userNotificationCenter(_ center: UNUserNotificationCenter, didF
    response: UNNotificationResponse, withCompletionHandler completi
    @escaping () -> Void) {
        let userInfo = response.notification.request.content.userInf
    print("didRecieve : userinfo", userInfo)
    completionHandler()
}
```

라. 쿠키 Access [id값]

웹의 쿠키정보를 활용하는 방법.
특정 웹페이지/URL일 때의
이벤트를 발생시키면 별도의 함수호출 없이도 통신이 가능한다.
여기서는
index.html 웹페이지에서 intro페이지로 넘어가는 시점에
로그인정보(userid)가 쿠키에 저장되고
이 때 앱에서 쿠키에 저장된 로그인 정보를 불러온다.
또한 Appdelegate에서 저장된 Token값을 가져와 get명령을 날린다.

```
func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
    webView.evaluateJavaScript("window.location.href") { (response, error) in
        if let title = response as? String {
            if(self.myWebView.viewModel.LoginCookie_Checked)//로그인 후 1회만 작동.
            {
                //로그인전, 로그아웃 화면(::index경로)에서 초기화한다.
                if(title == "http://koupdev.kccworld.net/" || title=="https://koup.kccworld.net/")
                {   self.myWebView.viewModel.LoginCookie_Checked=false }
            }
            else if(title == "http://koupdev.kccworld.net/intro" || title=="https://koup.kccworld.net/intro"){
                //쿠키생성된 후 (::Intro 페이지)에서 KCC 서버로 토큰을 전송한다.
                let dataStore = WKWebsiteDataStore.default()
                dataStore.httpCookieStore.getAllCookies({ (cookies) in
                    for cookie in cookies
                    {
                        if(cookie.name=="write")
                        {
                            let usercode = cookie.value
                            let delegate = UIApplication.shared.delegate as! AppDelegate //등록된 토큰.
                            self.SaveToken_to_KCC(IsDev:true, token:delegate.token, user:usercode)
                            self.myWebView.viewModel.LoginCookie_Checked=true
                        }
```

마. 서버에 토큰 값 전송(Get)

get/post 서버와 통신.

```
func SaveToken_to_KCC(IsDev:Bool, token:String, user: String)
{
    let url_dev = IsDev ? "http://koupdev.kccworld.net/api/setuserdevice" : "https://koup.kccworld.net/api/setuserdevice"
    let param = "?user=\(user)&type=i&token=\(token)&version=0.2.3&build=20211111&number=&imei=&model=iPhone&carrier=&manufacturer=Apple&sender=APNS"
    let url = URL(string: (url_dev+param))!

    var request = URLRequest(url: url)
    request.httpMethod = "GET"
    request.setValue("0", forHTTPHeaderField: "Content-Length")
    request.setValue("*/*", forHTTPHeaderField: "Accept")
    request.setValue("gzip, deflate, br", forHTTPHeaderField: "Accept-Encoding")
    request.setValue("keep-alive", forHTTPHeaderField: "Connection")

    print(request)
    let task = URLSession.shared.dataTask(with: request) { data, response, error in
        guard let data = data, error == nil else {
            print(error?.localizedDescription ?? "No data")
            return
        }
        let responseJSON = try? JSONSerialization.jsonObject(with: data, options: [])
        if let responseJSON = responseJSON as? [String: Any] {
            print(responseJSON)
        }
                }
    task.resume()
}
```

4. 빌드·배포

가. 아이콘 작업

www.figma.com에서 AppIcon을 생성할 수 있다.
생성된 파일을 https://appicon.co/에서
Ios 파일로 전환할 수 있다.

나. 오픈마켓 등록

다. 웹 설치 - Enterprise 계정 Inhouse 빌드[1)]

1) 사원 100명 이상 고객사(사원 100명이상) 전용 앱 개발 시. 로이 명의로는 불가.