**Part 1**

**앱만들기**

1. workspace만들기 (새폴더생성)
2. file – openfolder > 새폴더선택
3. 프로젝트생성

* expo init project명

폴더구조

APP

-first

- .expo-shared (엑스포 공유파일)

- assets (파일저장소(이미지, 폰트, 비디오, 사운드 파일))

- node\_modules(react module파일)

- .gitignore(git ignore 파일)

- App.js(앱 실행시 눈으로 보이는 화면.)

- app.json(코드에 속하지 않는 앱의 일부들을 구성)

- babel.config.js(환경설정파일)

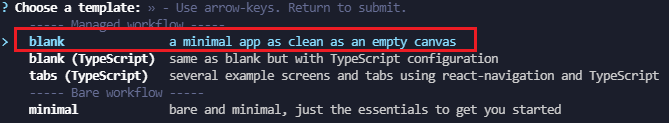
- package-lock.json

- package.json



명령어 - expo init first

4.템플릿 선택



blank - 간단합 앱을 만들 수 있는 빈 템플릿

blank(TypeScript) – 타입스크립트로 구성될수있는 빈 템플릿

tabs(TypeScript) – react-navigation 및 type스크립트를 사용하는 몇가지 예제화면 및 탭

minimal – 최소한의 필수적인 것 들로만 구성된 템플릿

5. cd 해당프로젝트로 들어간다.

- cd 프로젝트명

6. 기본 node\_modules에 없는 사용할 modules추가

- webview 패키지 (webview사용을위한 modules)

* expo install react-native-webview
* 스타일 컴포넌트( Javascript 파일 내에서 CSS를 사용할 수 있게 해주는 modules)
  + npm install styled-components
* 구글 & 페이스북 로그인
  + expo install expo-google-sign-in
  + expo install expo-facebook

구글로그인 & 페이스북 로그인 필요없는 기본 깡통어플.

App.js

*import \* as React from 'react'; // 리액트 기본 문법사용 import*

*import { WebView } from 'react-native-webview'; // webview 사용을위한 impoirt*

*import styled from 'styled-components/native' // style css 사용을위한 import*

*const SafeContainer = styled.SafeAreaView` // iphone 노치디자인 오류방지 백그라운드*

*flex: 1;*

*background-color: #264361;*

*`;*

*export default class App extends React.Component {*

*render() {*

*return (*

*<SafeContainer>*

*<WebView*

*source={{ uri: 'http://www.mozac.io/' }} // 사용할 링크*

*style={{ marginTop: 20 }} // 인라인 스타일링*

*/>*

*</SafeContainer>*

*);*

*}*

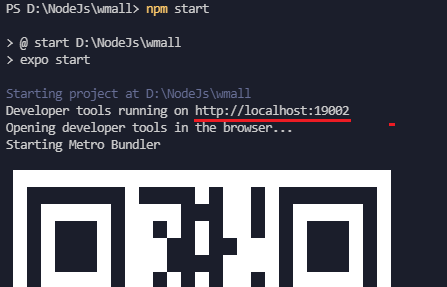
*}*

1. 복사 붙여넣기.

* 위 코드를 복사 붙여넣기한다

1. 주석제거 (주석까지 코드로 인식한다.)
2. 인터페이스 실행

* npm start



IOS

해당 주소(빨간줄)로된 URL창이 새로 열리는데, 왼쪽 하단의 qr코드나, vscode 콘솔창의 qr코드를 찍으시면 됩니다.

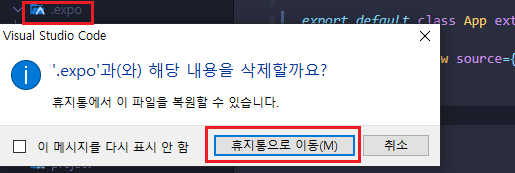
안드로이드

안드로이드의경우 어플에 들어가 Scan Qr Code를 통해 해당 큐알을 찍으면 아래 게이지바가 진행되며, 아이폰은 카메라로 QR을 찍어 어플에 들어가면 게이지바가 진행됩니다.

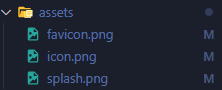
게이지가 완료되고 다시한번 QR을찍으면 정상적으로 어플리케이션의 모습을 미리 보실 수 있습니다.

\*테스트를 진행후 AFK를 만들예정이면 .expo폴더를 지워준다.(connect 오류 방지)

- .expo에 로그파일이 남아서 충돌가능성이 있다.



예제파일의 img들을 assets폴더에, 복사 붙여넣기 한다.



favicon 64 x 64 파비콘이미지

icon 512 x 512 앱아이콘이미지

splash 1242 x 2436 로딩이미지

**app.json**

{

"expo": {

"name": "앱만들기", *// 앱 이름*

"slug": "appSystem",

"version": "1.0.0",

"orientation": "portrait",

"icon": "./assets/icon.png", *// 앱아이콘*

"splash": {

"image": "./assets/splash.png", *// 첫 로딩 이미지*

"resizeMode": "contain",

"backgroundColor": "#2d2e46" *// 여백 색*

},

"updates": {

"fallbackToCacheTimeout": 0

},

"assetBundlePatterns": ["\*\*/\*"],

"ios": {

"bundleIdentifier": "io.nodehome.education", *//는패키지명(중복x)*

"buildNumber": "1.0.0", *// 버전*

"supportsTablet": true

},

"android": {

"package": "io.nodehome.education", *// 패키지명 (중복x)*

"versionCode": 1 *// 버전*

},

"web": {

"favicon": "./assets/favicon.png"

}

}

}

복 붙 후, 주석을 삭제한다.

(JSON은 데이터만 취급한다. 주석을 달게되면 그것까지 데이터로 취급해버림.)

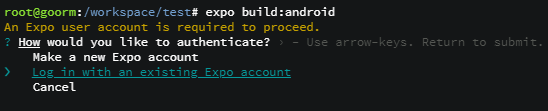
Android

expo build:android

IOS

expo build:ios

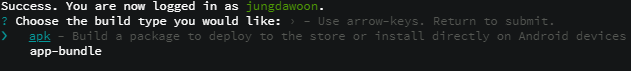




방향키로 이미 아이디가 있다는 선택지를 가고 엔터를 쳐줍니다.



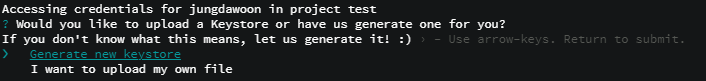
아이디 비번 입력 후 엔터



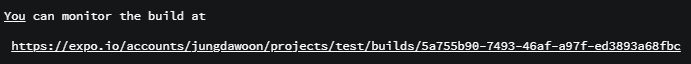
apk : 기본적은 apk파일

apk-bundle : apk와 비슷하지만 코드, 리소스, cpu 아키텍처와 메타데이터를 압축한 파일

apk를 만드는것이므로 apk선택 후 엔터

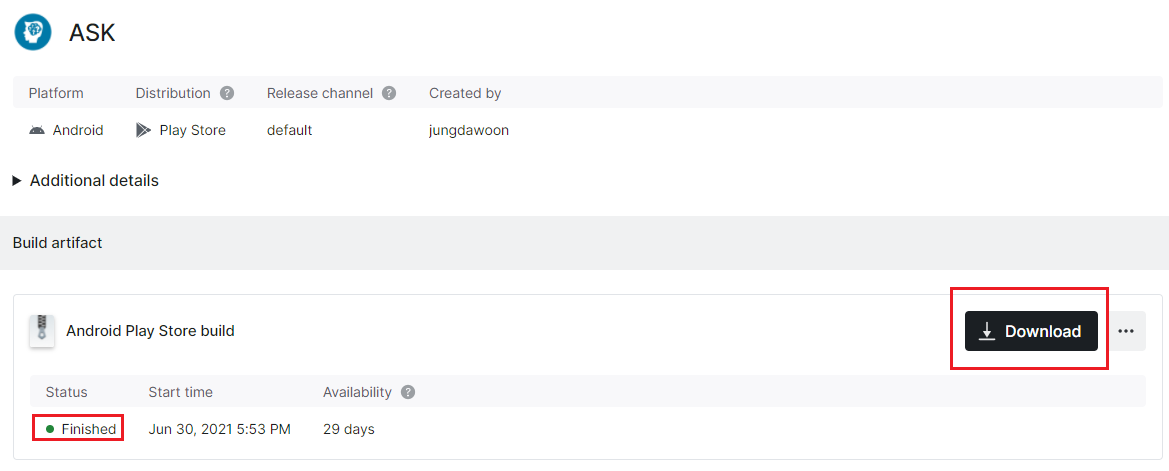


진행중에



이런식으로 링크가 나오는데 여기서 진행과정을 확인할 수 있으며, 완료되면 apk다운로드가 가능하다.

마지막으로 generate new keystore 선택후 기다려주기만 하면 됩니다. (약20분~30분 걸렸음.)



다운로드하게되면 어플을 다운받을 수 있는 apk가 완성됩니다.

**Part 2**

**API\_PACKAGE**

폴더구조

APP

-first

- .expo-shared (엑스포 공유파일)

- assets (파일저장소(이미지, 폰트, 비디오, 사운드 파일))

-fonts

-images

- adaptive-icon.png

- favicon.png

- icon.png

- splash.png

- node\_modules(react module파일)

- src

- components (부가 기능들 관련된 컴포넌트)

- appstatusbar.js – 상단백그라운드

- index.js – 실행파일

- navigations(화면 전환과 관련된 컴포넌트)

- MainTab.js – 어플 메인구성화면

- index.js – 실행파일

- screens(어플의 화면 단위 컴포넌트)

- Home.js – 홈아이콘클릭시 나타날 화면

- index.js – 실행파일

- Settings.js- 세팅아이콘클릭시 나타날 화면

- theme

- theme.js

- .gitignore(git ignore 파일)

- App.js(앱 실행시 눈으로 보이는 화면.)

- app.json(코드에 속하지 않는 앱의 일부들을 구성)

- babel.config.js(환경설정파일)

- package-lock.json

- package.json

1. Sample 파일 열어서 복사 붙여넣기
2. imgaes에 프로젝트에 맞는 이미지로 수정
3. app.json에 test문구 프로젝트명으로 수정
4. modules추가

- npm install

5. npm start실행

6. bulid진행

- Android

-expo build:android

-IOS

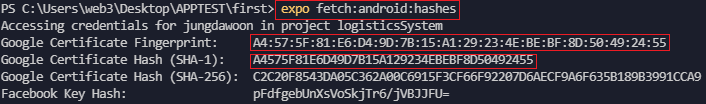
-expo build:ios

**Part 3**

**구글 및 페이스북 로그인**

* **구글로그인**
* firebase설정

터미널 expo fetch:android:hashes



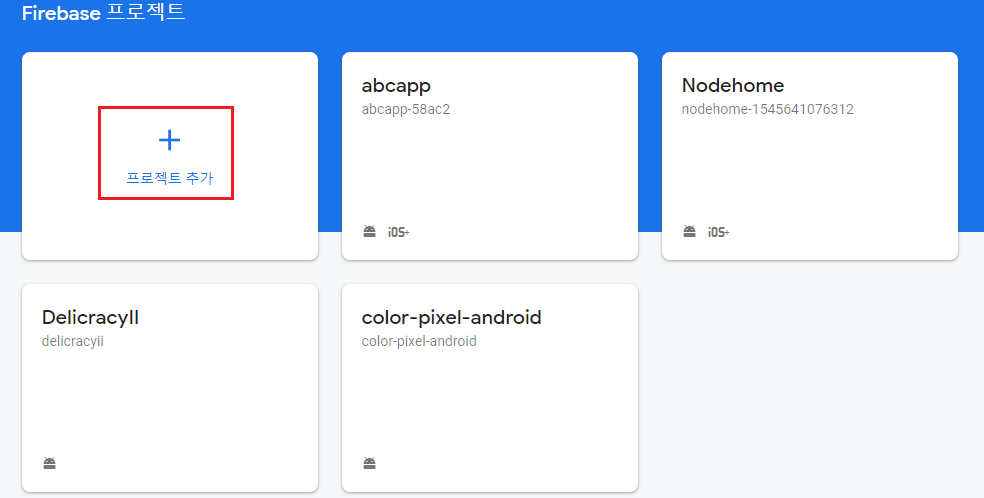
※참고사이트 : <https://docs.expo.dev/versions/latest/sdk/google-sign-in/#googlesigninoptions>

firebase (https://console.firebase.google.com/u/1/?pli=1)

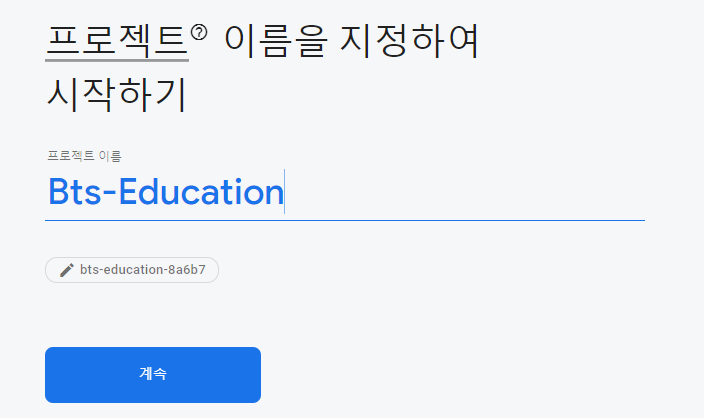
ID: app.net.4intel@gmail.com

PW: 4intel123ok

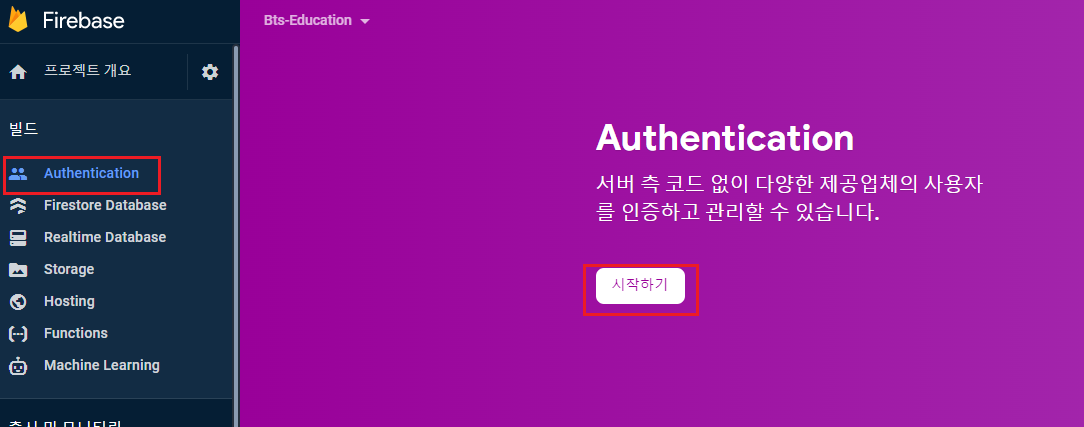
firebase로그인 새프로젝트 만들기.

****

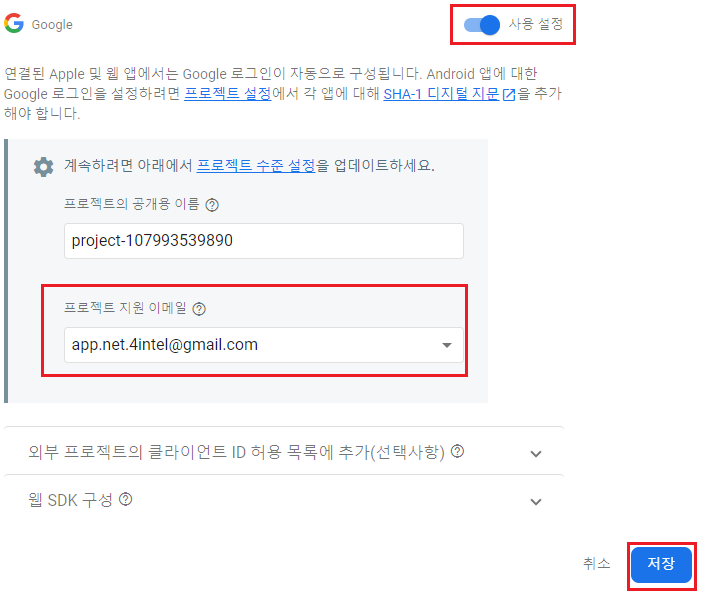
프로젝트 이름 입력 후 계속

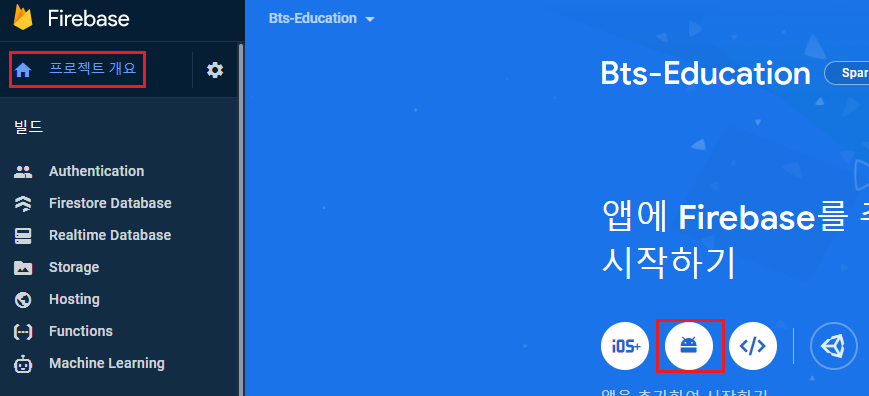
****

왼쪽 Authentication 클릭 후 시작하기를 누른다.



로그인 제공업체에서 구글을 클릭한 후, 사용 설정을 ON으로 설정 한 뒤, 이메일을 선택 후 저장한다.





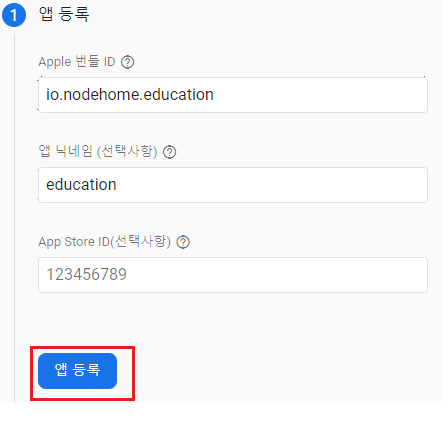
android package name과 fingerprint는 build를 해야 사용할 수 있어, 1.0.0버전을 사전에 빌드했으므로 expo를 통해 가져올 수 있다.

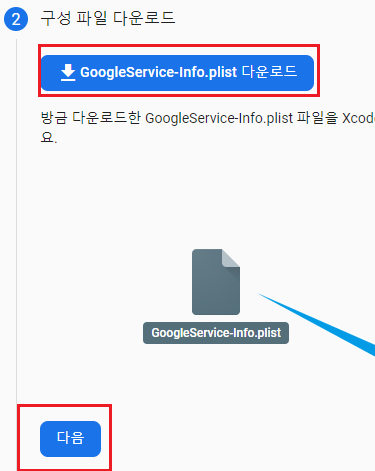




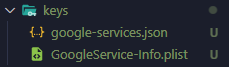
다음 마침

다시 앱추가하여, IOS를 선택.





keys라는 폴더를 만들어주어 해당 두파일을 저장합니다.



함수형으로 만든다.

App.js

*import* App *from* "./src/App";

*export* *default* App;

/src/App.js

*import* React, { useEffect } *from* 'react';

*import* { WebView } *from* 'react-native-webview';

*import* styled *from* 'styled-components/native'

*import* \* *as* GoogleSignIn *from* "expo-google-sign-in";

*import* \* *as* Facebook *from* "expo-facebook";

const SafeContainer = styled.SafeAreaView`

flex: 1;

background-color: #264361;

`;

const App = () => {

*// 구글 로그인*

useEffect(() => {

this.initAsync();

});

initAsync = async () => {

*await* GoogleSignIn.initAsync({

clientId:

"107993539890-gcaft094ptl3t7vqs2sta8g0ov4jqthi.apps.googleusercontent.com",

});

};

async function signInWithGoogleAsync(webviewRef) {

*try* {

*await* GoogleSignIn.askForPlayServicesAsync();

const result = *await* GoogleSignIn.signInAsync();

*if* (result.type === "success") {

result.loginchk = "google";

webviewRef.postMessage(JSON.stringify(result));

} *else* {

webviewRef.postMessage("실패");

}

} *catch* ({ message }) {

alert("login: Error:" + message);

}

}

*// 페이스북로그인*

async function signInWithFacebookAsync(webviewRef) {

console.log(">>>>>in facebook login");

*try* {

*await* Facebook.initializeAsync({

appId: "351218286764011",

});

const { type, token } = *await* Facebook.logInWithReadPermissionsAsync({

permissions: ["public\_profile"],

});

*if* (type === "success") {

const response = *await* fetch(

`https://graph.facebook.com/me?fields=id,name,email&access\_token=${token}`

);

const userInfo = *await* response.json();

userInfo.token = token;

userInfo.loginchk = "facebook";

*//console.log(userInfo);*

webviewRef.postMessage(JSON.stringify(userInfo));

} *else* {

webviewRef.postMessage(type);

}

} *catch* ({ message }) {

alert(`Facebook Login Error: ${message}`);

}

}

*return* (

<SafeContainer>

<WebView

source={{

uri: "https://bts-education.nodehome.io/",

}}

javaScriptEnabled={true}

injectedJavaScript={

`

var dtype =

'` +

Platform.OS +

`';

window.localStorage.setItem("dtype", dtype );

`

}

style={{ marginTop: 20 }} *// 인라인 스타일링*

ref={(ref) => (this.webviewRef = ref)}

onMessage={(event) => {

var loginchk = event.nativeEvent.data;

console.log(loginchk);

*if* (loginchk == "google") {

signInWithGoogleAsync(this.webviewRef);

} *else* *if* (loginchk == "facebook") {

signInWithFacebookAsync(this.webviewRef);

}

}}

/>

</SafeContainer>

);

};

*export* *default* App;

app.json 붙여넣기

{

"expo": {

"name": "앱만들기",

"slug": "appSystem",

"version": "1.0.2",

"orientation": "portrait",

"icon": "./assets/icon.png",

"splash": {

"image": "./assets/splash.png",

"resizeMode": "contain",

"backgroundColor": "#2d2e46"

},

"updates": {

"fallbackToCacheTimeout": 0

},

"assetBundlePatterns": ["\*\*/\*"],

"facebookScheme": "fb351218286764011",

"ios": {

"bundleIdentifier": "io.nodehome.education",

"config": {

"googleSignIn": {

"reservedClientId": "com.googleusercontent.apps.107993539890-gcaft094ptl3t7vqs2sta8g0ov4jqthi"

}

},

"googleServicesFile": "./keys/GoogleService-Info.plist",

"buildNumber": "1.0.2",

"supportsTablet": true

},

"android": {

"package": "io.nodehome.education",

"googleServicesFile": "./keys/google-services.json",

"config": {

"googleSignIn": {

"certificateHash": "a4575f81e6d49d7b15a129234ebebf8d50492455"

}

},

"versionCode": 1

},

"web": {

"favicon": "./assets/favicon.png"

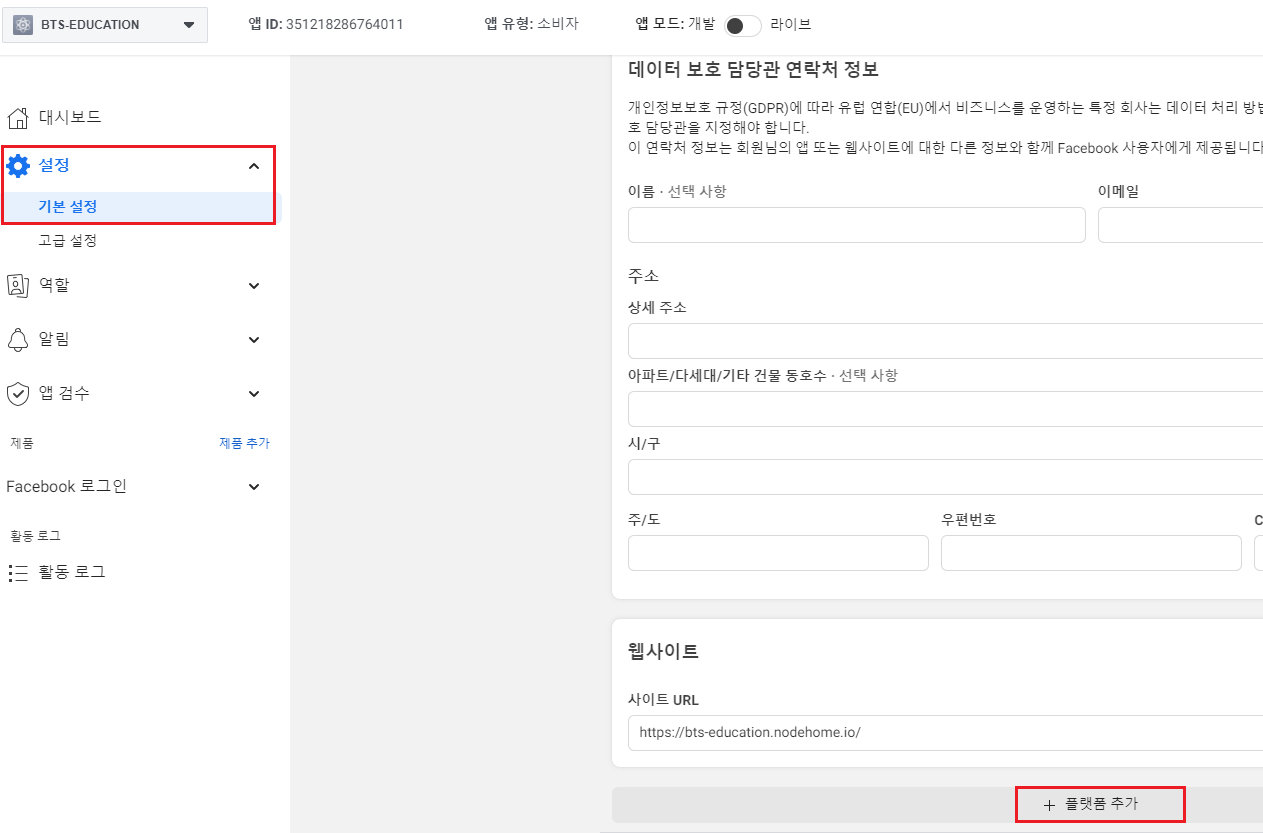
}

}

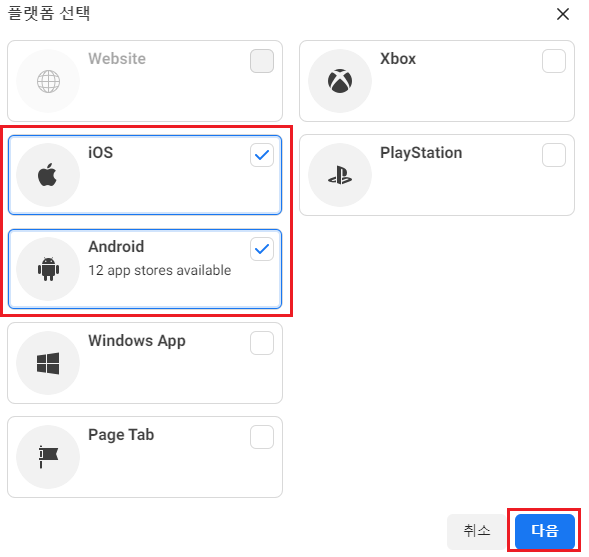
}

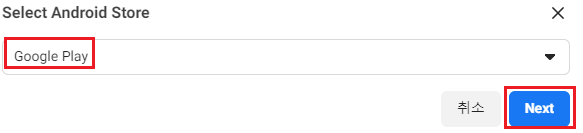
* **페이스북 로그인**

※참고사이트 : <https://docs.expo.dev/versions/latest/sdk/facebook/>



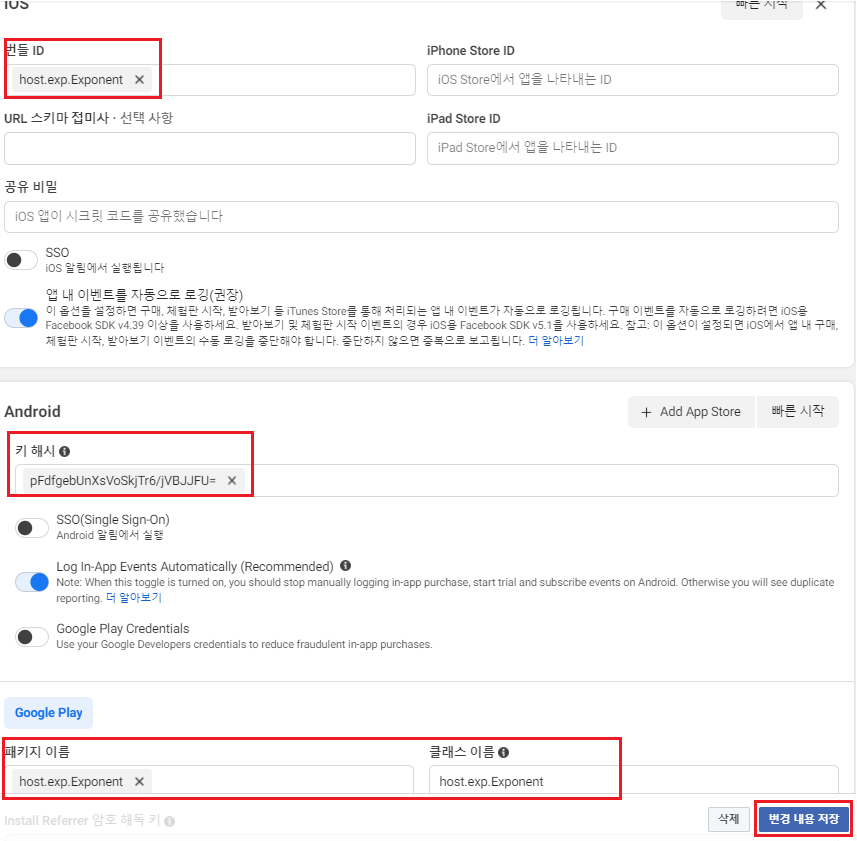
Android, IOS 추가





bundleID값은 host.exp.Exponent를 준다.

Android의 암호 해독키는 자동으로 생긴다.



키해시는 앞전에 했던, expo fetch:android:hashes 의 facebook key hashes의 값을 가져온다.

app.json에



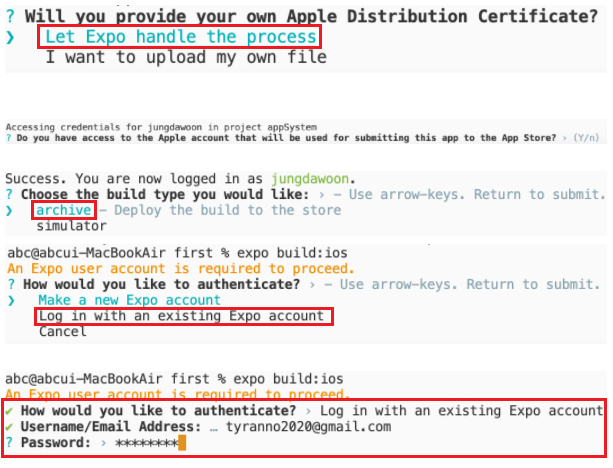
해당코드를 추가한다. fb + facebook 앱아이디

App.js속 페북의 appId도 변경해준다.



==========================================================

IOS 배포전 테스트버전



나머지 다음에 나오는것들은 YES, Y 만 눌러주면 된다. -> 완료 후 expo 사이트에서, build한걸 다운로드 한다.

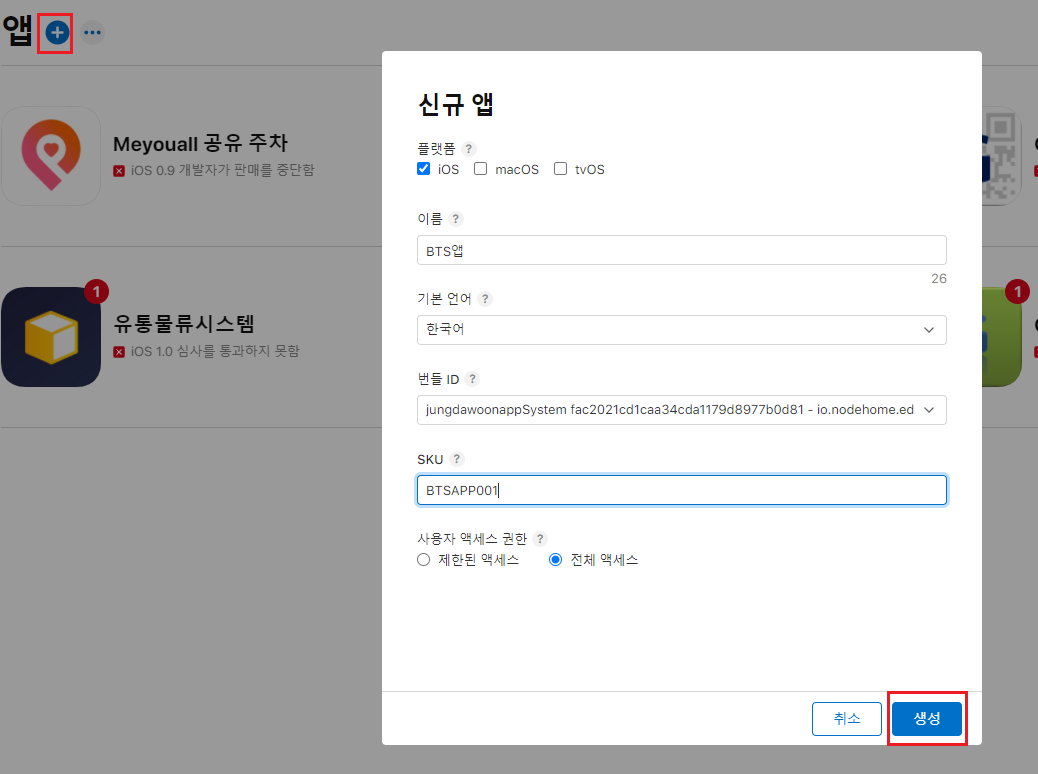
※기존 identifiers가 해당 pakage명이 있다면, 신규앱 생성시 해당패키지명이 생기지 않으므로 주의해야한다.

======================================================================

apple store (https://appstoreconnect.apple.com/)

ID: hs0chae@naver.com

PW: Jakeen2784

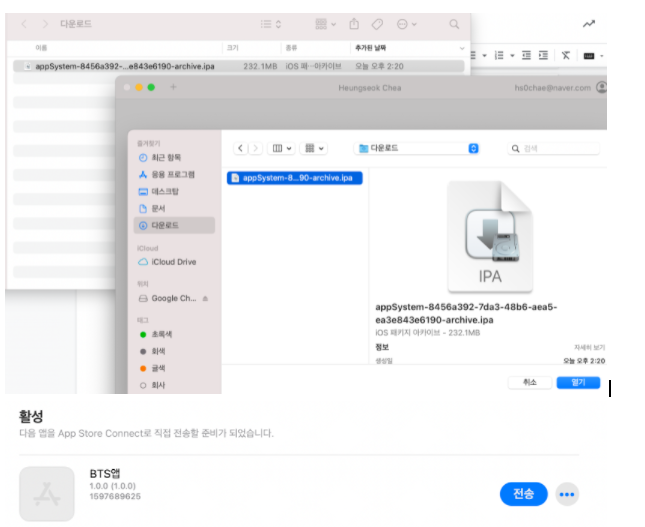


+버튼을 눌러 신규앱을 만들어준후 생성합니다.

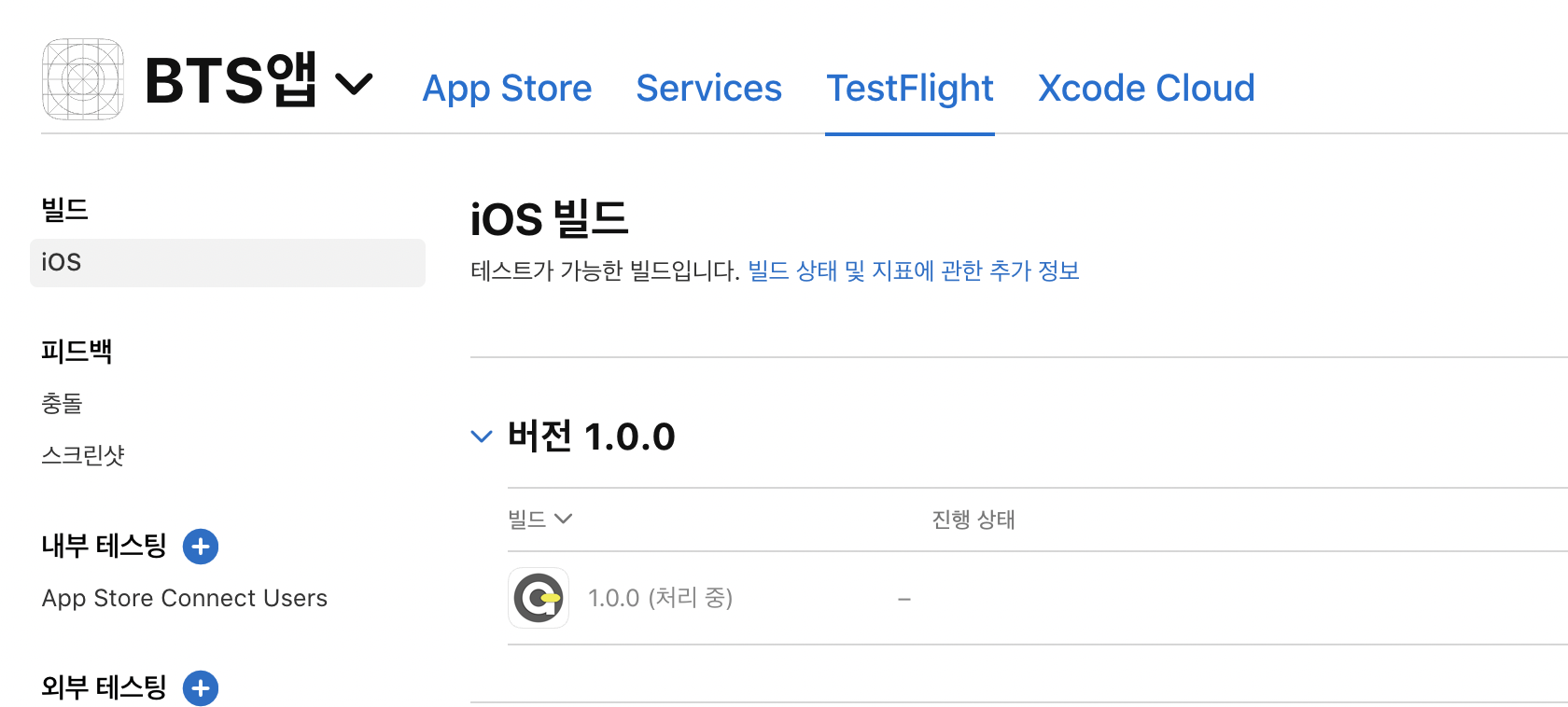
(번들id: 최근배포한 ios패키지명, SKU : 내가 마음대로적는다. 앱 고유 이름)

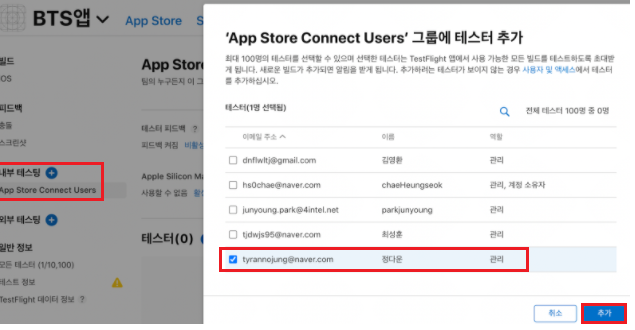
맥북의

Transporter 앱으로 해당 파일을 전송합니다.



전송은 금방 되지만 , App Store Connect까지 전달되기 30분이상소요.





배포참고

<https://devsrkim.tistory.com/entry/%EC%95%B1-%EC%8A%A4%ED%86%A0%EC%96%B4-%EC%95%B1-%EB%B0%B0%ED%8F%AC-%ED%95%98%EA%B8%B0>