앱개발 4주차

4주차 배울것

[앱과 서버]우리 앱에서의 서버

[앱과 서버]날씨 서버 외부 API - 휴대폰 위치 가져오기

[앱과 서버]날씨 서버 외부 API - API 적용

[서버리스]서버를 사용만 하자! 서버리스(Serverless)

[파이어베이스]파이어베이스(firebase) 소개

[파이어베이스]파이어베이스를 앱에 연결하기

[파이어베이스]파일 스토리지

[파이어베이스]리얼타임 데이터베이스 > 설정

[파이어베이스]리얼타임 데이터베이스 > 설전체 데이터 읽기

[파이어베이스]리얼타임 데이터베이스 > 특정 데이터 읽기

[파이어베이스]리얼타임 데이터베이스 > 쓰기

4주차 끝-숙제 설명

**[앱과 서버]우리 앱에서의 서버**

* 앱과 서버
* 서버리스(Serverless)
* 파이어베이스(구글)
* API
* JSON(list + Dictionary Data Type)
* API 사용 방식: 주소 이용 API, 함수 이용 API

**[앱과 서버]날씨 서버 외부 API - 휴대폰 위치 가져오기(함수 이용)**

openweathermap을 이용한 구글 API이용,현재 위치의 날씨정보 가져오기

**앱 위치정보 권한 설정(Expo 활용)**

터미널을 이용하여 expo-location을 설치

| expo install expo-location |
| --- |
| C:\Users\kkk\Desktop\sparta\_study\[프로젝트 폴더]>expo install expo-location  Installing 1 SDK 44.0.0 compatible native module using Yarn.  > yarn add expo-location@~14.0.1  yarn add v1.22.18  [1/4] Resolving packages...  [2/4] Fetching packages...  [3/4] Linking dependencies...  [4/4] Building fresh packages...  success Saved lockfile.  success Saved 2 new dependencies.  info Direct dependencies  └─ expo-location@14.0.2  info All dependencies  ├─ expo-location@14.0.2  └─ unimodules-task-manager-interface@7.1.1  Done in 7.34s. |

Main.js에 날씨 가져오기도구 적용(함수 이용)

| import \* as Location from "expo-location";  const getLocation = async () => {  //수많은 로직중에 에러가 발생하면  //해당 에러를 포착하여 로직을 멈추고,에러를 해결하기 위한 catch 영역 로직이 실행  try {  //자바스크립트 함수의 실행순서를 고정하기 위해 쓰는 async,await  await Location.requestForegroundPermissionsAsync();  const locationData= await Location.getCurrentPositionAsync();  console.log(locationData)  console.log(locationData['coords']['latitude'])  console.log(locationData['coords']['longitude'])  } catch (error) {  //혹시나 위치를 못가져올 경우를 대비해서, 안내를 준비합니다  Alert.alert("위치를 찾을 수가 없습니다.", "앱을 껏다 켜볼까요?");  }  } |
| --- |

이 경우, 사용자 앱 실행시 [위치 정보 허용 여부] 팝업이 나타나고, “예” 선택시 앱이 실행된다.

추후 앱 등록시에도 고객 위치정보 허용 여부를 묻게 된다

* async() : 비동기식 함수 호출하라
* await : 비동기식일지라도 순서대로 호출하라

결과 Console에는 다음과 같이 locationdata, latitude,longitude 내용이 차례대로 출력된다

| Object {  "coords": Object {  "accuracy": 20,  "altitude": 47.80000305175781,  "altitudeAccuracy": 1.5090075731277466,  "heading": 0,  "latitude": 37.4854791,  "longitude": 126.9695079,  "speed": 0,  },  "mocked": false,  "timestamp": 1652166449629,  }  37.4854791  126.9695079 |
| --- |

**[앱과 서버]날씨 서버 외부 API - API 적용(주소 이용)**

**주소 요청 API(axios) 이용 도구**

터미널을 이용하여 **axios** 를 설치

axios는 주소 방식 API를 이용하기 위해 사용하는 도구

| (tip)구글링 후 “npm axios”를 검색하면  axios 사용법을 확인할 수 있다 |
| --- |
| <https://www.npmjs.com/package/axios> |

**\*\* yarn을 이용하여 설치함에 주의**

| yarn add axios |
| --- |
| C:\Users\kkk\Desktop\sparta\_study\[프로젝트 폴더]>yarn add axios  yarn add v1.22.18  [1/4] Resolving packages...  [2/4] Fetching packages...  [3/4] Linking dependencies...  warning "react-native > react-native-codegen > jscodeshift@0.11.0" has unmet peer dependency "@babel/preset-env@^7.1.6".  [4/4] Building fresh packages...  success Saved lockfile.  success Saved 6 new dependencies.  info Direct dependencies  └─ axios@0.27.2  info All dependencies  ├─ asynckit@0.4.0  ├─ axios@0.27.2  ├─ combined-stream@1.0.8  ├─ delayed-stream@1.0.0  ├─ follow-redirects@1.15.0  └─ form-data@4.0.0  Done in 4.97s. |

| **외부 주소 API를 이용할 시에는**  **해당 API에 회원가입 후 Key를 발급받아 이용하는 것이 원칙임** |
| --- |

날씨 AP를 가져오는 코드는 다음과 같다

| (중략)  import \* as Location from "expo-location";  import axios from "axios"  export default function MainPage({navigation,route}) {  const [state,setState] = useState([])  const [cateState,setCateState] = useState([])  //날씨 데이터 상태관리 상태 생성!  const [weather, setWeather] = useState({  temp : 0,  condition : ''  })  const [ready,setReady] = useState(true)  useEffect(()=>{  navigation.setOptions({  title:'나만의 꿀팁'  })  setTimeout(()=>{  //헤더의 타이틀 변경  getLocation()  setState(data.tip)  setCateState(data.tip)  setReady(false)  },1000)      },[])  const getLocation = async () => {  //수많은 로직중에 에러가 발생하면  //해당 에러를 포착하여 로직을 멈추고,에러를 해결하기 위한 catch 영역 로직이 실행  try {  //자바스크립트 함수의 실행순서를 고정하기 위해 쓰는 async,await  await Location.requestForegroundPermissionsAsync();  const locationData= await Location.getCurrentPositionAsync();  console.log(locationData)  console.log(locationData['coords']['latitude'])  console.log(locationData['coords']['longitude'])  const latitude = locationData['coords']['latitude']  const longitude = locationData['coords']['longitude']  const API\_KEY = "cfc258c75e1da2149c33daffd07a911d";  const result = await axios.get(  `http://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lon=${longitude}&appid=${API\_KEY}&units=metric`  );  console.log(result)  const temp = result.data.main.temp;  const condition = result.data.weather[0].main    console.log(temp)  console.log(condition)  setWeather({  temp,condition  })  } catch (error) {  Alert.alert("위치를 찾을 수가 없습니다.", "앱을 껏다 켜볼까요?");  }  }  const category = (cate) => {  if(cate == "전체보기"){  //전체보기면 원래 꿀팁 데이터를 담고 있는 상태값으로 다시 초기화  setCateState(state)  }else{  setCateState(state.filter((d)=>{  return d.category == cate  }))  }  }  //data.json 데이터는 state에 담기므로 상태에서 꺼내옴  // let tip = state.tip;  let todayWeather = 10 + 17;  let todayCondition = "흐림"  //return 구문 밖에서는 슬래시 두개 방식으로 주석  return ready ? <Loading/> : (  /\*  return 구문 안에서는 {슬래시 + \* 방식으로 주석  \*/  <ScrollView style={styles.container}>  <StatusBar style="light" />  {/\* <Text style={styles.title}>나만의 꿀팁</Text> \*/}  <Text style={styles.weather}>오늘의 날씨: {weather.temp + '°C ' + weather.condition} </Text>  <TouchableOpacity style={styles.aboutButton} onPress={()=>{navigation.navigate('AboutPage')}}>  <Text style={styles.aboutButtonText}>소개 페이지</Text>  </TouchableOpacity> |
| --- |

코드를 살펴보면

| **관련 모듈 import** |
| --- |
| import \* as Location from "expo-location";  import axios from "axios" |
| **날씨 데이터 state 생성** |
| const [weather, setWeather] = useState({  temp : 0,  condition : ''  }) |
| **useEffect에서 위치/날씨 가져오기(getLocation())** |
| useEffect(()=>{  navigation.setOptions({  title:'나만의 꿀팁'  })  //뒤의 1000 숫자는 1초를 뜻함  //1초 뒤에 실행되는 코드들이 담겨 있는 함수  setTimeout(()=>{  //헤더의 타이틀 변경  getLocation()  setState(data.tip)  setCateState(data.tip)  setReady(false)  },1000)  },[]) |
| **getLocation() 함수** |
| const getLocation = async () => {  //수많은 로직중에 에러가 발생하면  //해당 에러를 포착하여 로직을 멈추고,에러를 해결하기 위한 catch 영역 로직이 실행  try {  //자바스크립트 함수의 실행순서를 고정하기 위해 쓰는 async,await  await Location.requestForegroundPermissionsAsync();  const locationData= await Location.getCurrentPositionAsync();  console.log(locationData)  console.log(locationData['coords']['latitude'])  console.log(locationData['coords']['longitude'])  const latitude = locationData['coords']['latitude']  const longitude = locationData['coords']['longitude']  const API\_KEY = "cfc258c75e1da2149c33daffd07a911d";  const result = await axios.get(  `http://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lon=${longitude}&appid=${API\_KEY}&units=metric`  );  console.log(result)  const temp = result.data.main.temp;  const condition = result.data.weather[0].main    console.log(temp)  console.log(condition)  //오랜만에 복습해보는 객체 리터럴 방식으로 딕셔너리 구성하기!!  //잘 기억이 안난다면 1주차 강의 6-5를 다시 복습해보세요!  setWeather({  temp,condition  })  } catch (error) {  //혹시나 위치를 못가져올 경우를 대비해서, 안내를 준비합니다  Alert.alert("위치를 찾을 수가 없습니다.", "앱을 껏다 켜볼까요?");  }  } |
| 처음 화면 뿌리기 전 초기값 지정 |
| let todayWeather = 10 + 17;  let todayCondition = "흐림" |
| 화면에 온도, 날씨 출력 |
| <Text style={styles.weather}>오늘의 날씨: {weather.temp + '°C ' + weather.condition} </Text> |

**[서버리스]서버를 사용만 하자! 서버리스(Serverless)**

**[파이어베이스]파이어베이스(firebase) 소개**

**파이어베이스**

구글에서 만든 서비스로, 다음과 같은 서비스를 제공하고 있다.



파이어베이스 이용 방법은 다음과 같은 순서로 진행한다

* 파이어베이스 회원 가입(google 계정)
* 파이어베이스 프로젝트 생성
* 파이어베이스 활성화

| **파이어베이스 회원 가입(구글 계정 가입되어 있으면 바로 연결)** |
| --- |
|  |
| 프로젝트 생성 |
| 프로젝트 시작하기  - 프로젝트 이름: sparta-myhoneytip-kim (앱 명과 동일하게 하는 것이 좋음)  - Google Analytics: OK |
| 파이어베이스 프로젝트 홈 |
|  |

**[파이어베이스]파이어베이스를 앱에 연결하기**

**앱에 Firebase를 추가하여 시작하기**

| 유형 선택 |
| --- |
| iOS, Andrios, 웹 중 택일(웹 선택) |
| 프로젝트명 입력 |
|  |
| npm 사용 |
|  |
| 프로젝트 설정 확인/연결정보 확인  하단에 Key값이 나타남(보안 유지) |
|  |

**expo firebase 설치**

터미널 이용, expo firebase 설치

| expo install firebase |
| --- |
| C:\Users\kkk\Desktop\sparta\_study\sparta-myhoneytip-kim>expo install firebase  Installing 1 other package using Yarn.  > yarn add firebase  yarn add v1.22.18  [1/4] Resolving packages...  [2/4] Fetching packages...  [3/4] Linking dependencies...  success Saved lockfile.  success Saved 49 new dependencies.  info Direct dependencies  └─ firebase@9.8.1  info All dependencies  .  .  (중략)  .  .  ├─ tmp@0.2.1  ├─ websocket-driver@0.7.4  └─ websocket-extensions@0.1.4  Done in 35.00s. |

**firebaseConfig.js 생성**

프로젝트 폴더 root에 firebaseConfig.js를 생성 후 복붙

이 파일은 firebase 서비스에 접속할 Key 파일이다.(공개는 금물!!!)

key 부분을 방금 생성한 Key 로 대체한 후 저장한다.

드디어 이제 앱에서 firebase를 이용할 준비가 되었다.

[파이어베이스]파일 스토리지

앱에 이용할 이미지들을 firebase storage에 저장한다.

이 경우 앱에 이미지를 저장할 필요가 없어지므로 앱이 한층 가벼워지게 된다.

| firebase storage 시작하기(좌측 storage 메뉴 선택 후 시작) |
| --- |
|  |
| 폴더 생성 후 파일 업로드하기 |
|  |

이미지 서버가 완성되었다. 정말 편하고 쉽게 이미지 서버를 갖게 되었다.

**[파이어베이스]리얼타임 데이터베이스 > 설정**

리얼타임 데이터베이스, JSON 리스트, 딕셔너리 구조,

* 이미지 저장: 파일 저장소 스토리지
* JSON 데이터: 리얼타임 데이터베이스(firebase)

| 리얼타임데이터베이스 설정 |
| --- |
|  |
| firebaseconfig.js내에 database URL 저장 |
| projectId: "sparta-myjhoneytip-kim",  databaseURL:"https://sparta-myjhoneytip-kim-default-rtdb.asia-southeast1.firebasedatabase.app/", |
| Realtime Database 설정 변경  Read/write 모두 true로 변경 후 “게시” |
|  |
| JSON 데이터 쓰기  데이터>...>JSON 불러오기 후 프로젝트 폴더>data.json 선택>등록 |
|  |

이제 리얼타임 데이터베이스에 데이터(json)를 등록하고, 앱에서 사용할 준비가 되었다.

정말 편하다.

**[파이어베이스]리얼타임 데이터베이스 > 전체 데이터 읽기**

기존 로컬 JSON 데이터로 가져오던 것을 리얼타임 데이터베이스로 이동, 이를 가져오게끔 변경

| **MainPage.js 수정**  기존 data.json에서 가져오던 것을, firebase 에서 가져오도록 변경  나머지 페이지 처리는 동일함 |
| --- |
| import {firebase\_db} from "../firebaseConfig"  useEffect(()=>{  navigation.setOptions({  title:'나만의 꿀팁'  })  //뒤의 1000 숫자는 1초를 뜻함  //1초 뒤에 실행되는 코드들이 담겨 있는 함수  setTimeout(()=>{  firebase\_db.ref('/tip').once('value').then((snapshot) => {  console.log("파이어베이스에서 데이터 가져왔습니다!!")  let tip = snapshot.val();    setState(tip)  setCateState(tip)  getLocation()  setReady(false)  });  // getLocation()  // setState(data.tip)  // setCateState(data.tip)  // setReady(false)  },1000)  },[]) |

**[파이어베이스]리얼타임 데이터베이스 > 특정 데이터 읽기**

카드 상세 페이지 이동시 전체 데이터를 넘겨주지 않고 키값을 전달, firebase에서 특정 데이터를 조회하는 형태로 변경

| **Card.js 수정**  **DetailPage 호출부분 수정** |
| --- |
| <TouchableOpacity style={styles.card} onPress={()=>{navigation.navigate('DetailPage',{idx:content.idx})}}> |
| DetailPage.js 수정  해당 키를 가진 레코드만을 firebase 에서 가져오도록 변경.  나머지 화면 처리는 전과 동일함 |
| //넘어온 데이터는 route.params에 들어 있습니다.  const { idx } = route.params;  firebase\_db.ref('/tip/'+idx).once('value').then((snapshot) => {  let tip = snapshot.val();  setTip(tip)  }); |

**[파이어베이스]리얼타임 데이터베이스 > 쓰기**

Firebase 레코드 추가하기

* 사용자 고유 ID 가져오기
* 꿀팁 정보 가져오기
* 등록용 레코드 생성
* 데이터 등록

**악몽의 시작**

Firebase에 앱을 연동하니 못보던 에러들이 속출한다. 분명 개발환경에 문제인 것은 알겠으나, 아직 문제 해결 수준은 초보라 힘들다.

결국 2주차 수업으로 돌아가, 프로젝트 등록부터 다시 시작했다. 하지만 결과는 마찬가지.

옛날 같았으면 며칠을 헤매었을 거다.

하지만 슬랙, 구글, github가 있으니 이전과는 달랐다.

전 세계 사용자가 비슷한 문제를 등록하고 Known-Error를 등록해 놓았다.

하나 하나 문제를 해결해 나갔지만 쉽지는 않다. 당연할테지.

스파르타 코딩 클럽의 슬랙 채널에 문제를 올렸다.

3시간만에 답변이 올라왔고, 답변을 기초로 github에 등록된 Known-Error를 적용하니 화면이 작동되는 것이 아닌가?

얼마만에 느껴보는 뿌듯함인가?

서둘러 슬랙 게시판에 답변을 올리고, 공유를 했다. 이게 github의 정신일테지?