NATIVE WEATHER

11조

2024-1
크로스플랫폼 프로그래밍

프로젝트 구성원

202184048 임준혁	발표자	→ 디자인, 메인 화면 구현, 기능 통합
202004092 이성현	PPT	→ 탭 및 기능 구현
202284024 신명재	PPT	
201904038 민태호	PPT	→ API 호출 컴포넌트, 데이터 파싱
202284011 김연재	시연영상	


개요
기능
회고

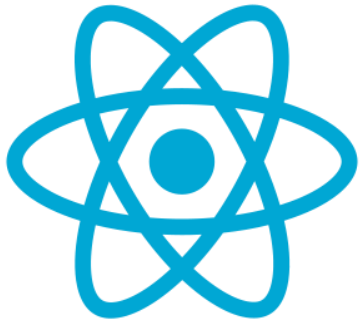
개요

기능

회고

Native Weather : 기상 정보 앱

- 학습 내용을 최대한 활용하여 개인 역량 강화에 초점을 둠
- OPEN API를 사용하여 기상청 공공데이터에 쉽게 접근 가능
- 사용자가 실시간으로 기온, 날씨, 대기질 정보를 큰 틀에서 파악 가능



+

OPEN API

개요

기능

회고

기능 : 홈

① 앱 및 개발자 정보 팝업

② 화면 이동 버튼 (직관적)

③ 바텀 네비게이션 바

앱 내 모든 상태관리 : useState 사용

개요

기능

회고

기능 : 홈

```
<NavigationContainer>
  <Tab.Navigator
    screenOptions={({ route }) => ({
      tabBarIcon: ({ color, size }) => {
        let iconName;
        if (route.name === 'Home') {
          iconName = 'home';
        } else if (route.name === 'Temperature') {
          iconName = 'thermometer';
        } else if (route.name === 'Weather') {
          iconName = 'cloud';
        } else if (route.name === 'Dust') {
          iconName = 'partly-sunny';
        }
        return <Icon name={iconName} size={size} color={color} />;
      },
    })}
    tabBarOptions={{
      activeTintColor: 'tomato',
      inactiveTintColor: 'gray',
    }}
  >
```

Tab.Navigator

하단 탭 네비게이션을 제공

screenOptions = { }

각 탭 버튼에 대한 아이콘, 스타일을 설정

tabBarOption = { }

활성, 비활성 화면을 색상으로 구분

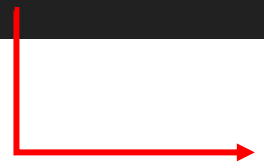
기능 : 홈

개요

기능

회고

```
<Tab.Screen
  name="Home"
  component={HomeStack}
  options={{ tabBarLabel: '홈' }} // 아래 네비게이션의 라벨 설정
/>
<Tab.Screen name="Temperature" component={TempStack} options={{ tabBarLabel: '기온' }} />
<Tab.Screen name="Weather" component={WeatherStack} options={{ tabBarLabel: '날씨' }} />
<Tab.Screen name="Dust" component={DustStack} options={{ tabBarLabel: '대기질' }} />
```



Tab.Screen

Tab.Navigator 내부에 사용하여 각 화면을 구성

개요

기능

회고

기능 : 홈

```
<Stack.Navigator initialRouteName="HomeMain">
  <Stack.Screen name="HomeMain" component={HomeScreen} options={{ headerShown: false }} />
</Stack.Navigator>
```

```
<Stack.Navigator>
  <Stack.Screen name="Temp" component={TempScreen} options={{ headerShown: false }} />
  <Stack.Screen name="TempDetail" component={TempDetailScreen} options={{ headerShown: false }} />
</Stack.Navigator>
```

```
<Stack.Navigator>
  <Stack.Screen name="Weather" component={WeatherScreen} options={{ headerShown: false }} />
  <Stack.Screen name="WeatherDetail" component={WeatherDetailScreen} options={{ headerShown: false }} />
</Stack.Navigator>
```

```
<Stack.Navigator>
  <Stack.Screen name="Dust" component={DustScreen} options={{ headerShown: false }} />
</Stack.Navigator>
```

```
const Stack = createNativeStackNavigator();
```

화면을 Stack으로 관리하기 위한
createNativeStackNavigator()

개요

기능

회고

스택 네비게이션의 필요성

화면 전환

- 기능에 따라 화면 구분
- 자유로운 화면 전환

네비게이션
히스토리 관리

- 이전 화면 전환 가능
- 다른 지역 화면 확인 가능

코드 조직화

- 화면을 스택으로 그룹화
- 코드베이스 유지가 깔끔

개요

기능

회고

기능 : API 호출 / 기온, 날씨

```
const { data } = await axios.get(
  'http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0/getVilageFcst', {
    params: {
      serviceKey: 'V7RZpsZ3goxaM                                    1vKburVUVbFU
      numOfRows: 1000,
      pageNo: 1,
      dataType: 'JSON',
      base_date: dateString,
      base_time: timeString,
      nx: userNx,
      ny: userNy,
    },
  }
);
```

날짜와 시간을 바탕으로 데이터를 호출

지역 코드

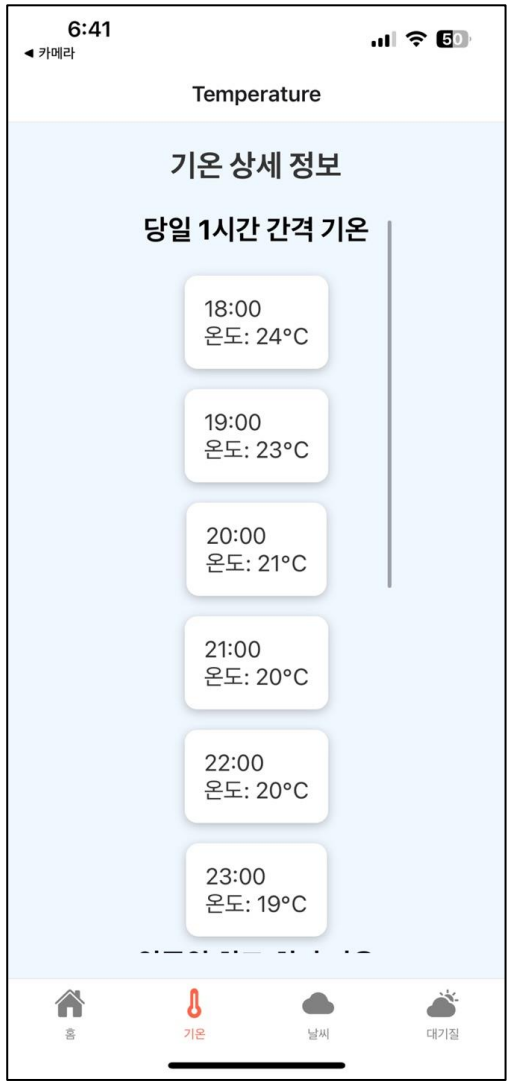
개요

기능

회고

기능 : 기온

지역 선택



기능 : 기온

```
const data = [
  { id: '1', name: '서울특별시', nx: 60, ny: 127 },
  { id: '2', name: '경기도', nx: 61, ny: 120 },
  { id: '3', name: '강원도', nx: 73, ny: 134 },
  { id: '4', name: '충청북도', nx: 69, ny: 106 },
  { id: '5', name: '충청남도', nx: 51, ny: 112 },
  { id: '6', name: '전라북도', nx: 63, ny: 89 },
  { id: '7', name: '전라남도', nx: 58, ny: 74 },
  { id: '8', name: '경상북도', nx: 89, ny: 90 },
  { id: '9', name: '경상남도', nx: 91, ny: 77 },
  { id: '10', name: '제주특별자치도', nx: 53, ny: 38 },
];
```

해당 지역 코드를 입력값으로 사용 → API 데이터 호출

개요

기능

회고

기능 : 기온

데이터 리스트

리스트 요소 컨테이너

스타일 적용

리스트 타이틀

```
return (
  <FlatList
    data={data}
    renderItem={(props) => renderItem({ ...props, navigation })}
    keyExtractor={(item) => item.id}
    contentContainerStyle={styles.container}
    ListHeaderComponent={() => (
      <Text style={styles.title}>지역 선택</Text>
    )}
  />
);
```

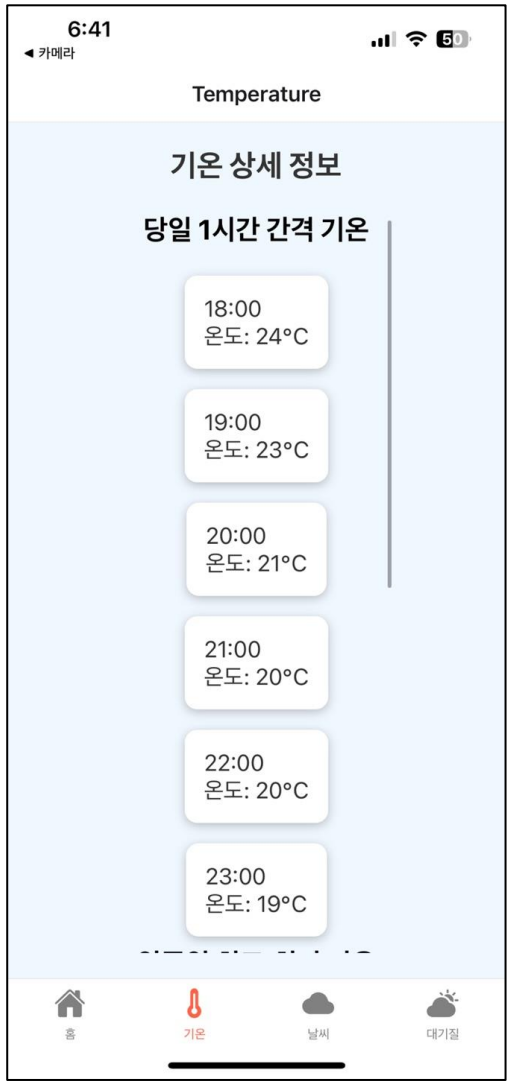
많은 양의 데이터를 리스트로 출력 할 수 있는 “FlatList” 컴포넌트 사용

개요

기능

회고

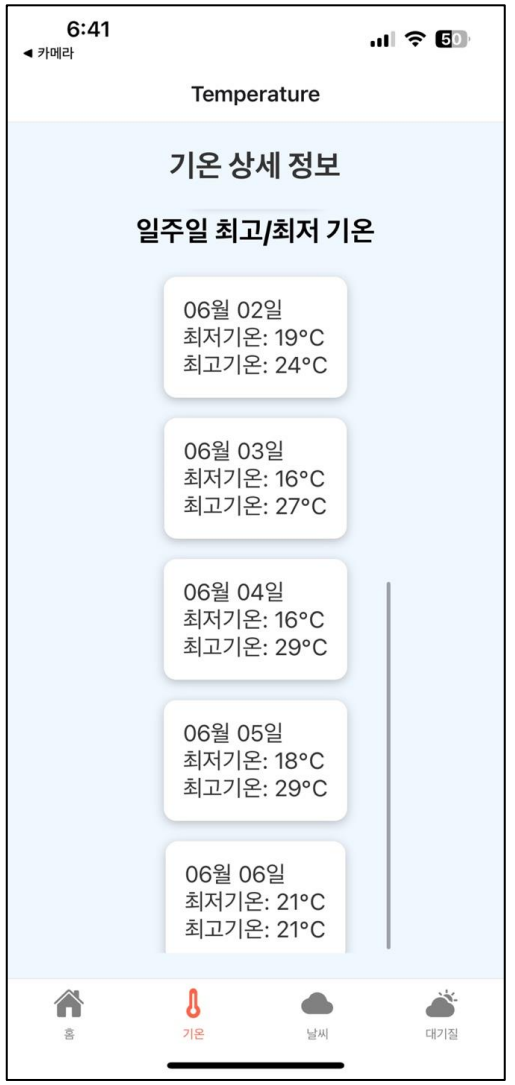
기능 : 기온



사용 당일 시간별 기온

사용 당일 이후 최고/최저 기온

기온 상세 정보도 FlatList 사용



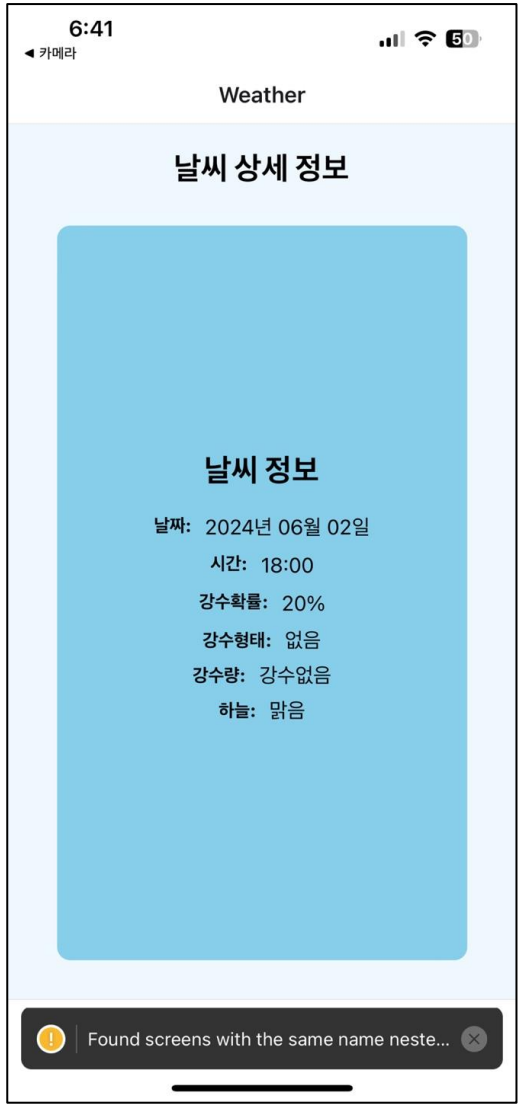
개요

기능

회고

기능 : 날씨

지역 선택



강수 형태

- 없음
- 비/눈
- 비
- 눈

하늘 상태

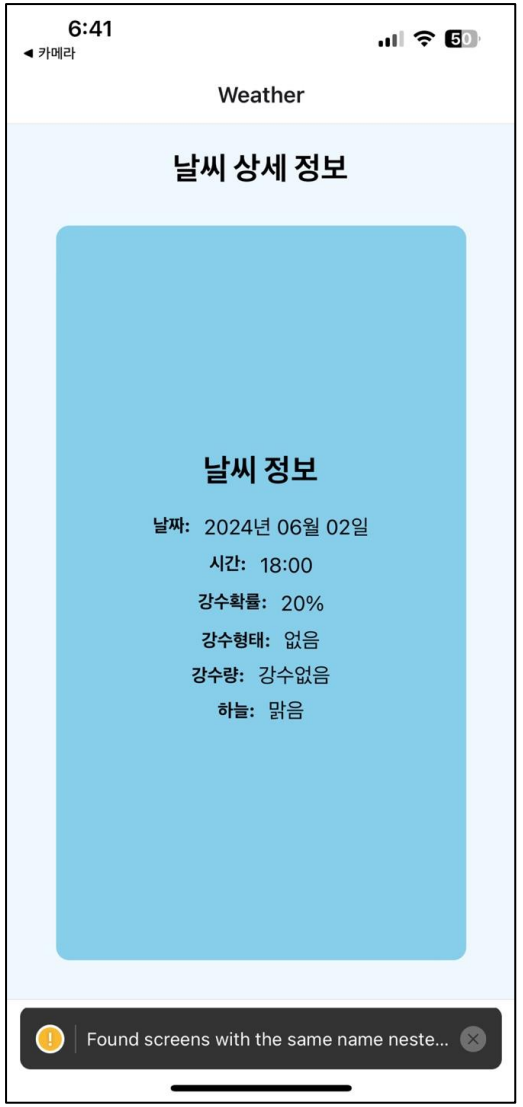
- 맑음
- 흐림
- 구름 많음

개요

기능

회고

기능 : 날씨



하늘 상태

맑음 구름 많음 흐림

BackgroundColor

개요

기능

회고

기능 : API 호출 / 대기질

```
const fetchAirQuality = async () => {
  try {
    const { data } = await axios.get(
      'http://apis.data.go.kr/B552584/ArpltnInforInqireSvc/getMsrstnAcctoRltmMesureDnsty', {
        params: {
          serviceKey: 'V7RZpsZ3goxaM6p+ssykr                                    UVbFUdARFRk+T9Tf
          returnType: 'json',
          numOfRows: 100,
          pageNo: 1,
          stationName: '송파구',
          dataTerm: 'DAILY',
          ver: '1.0',
        },
      }
    );
```

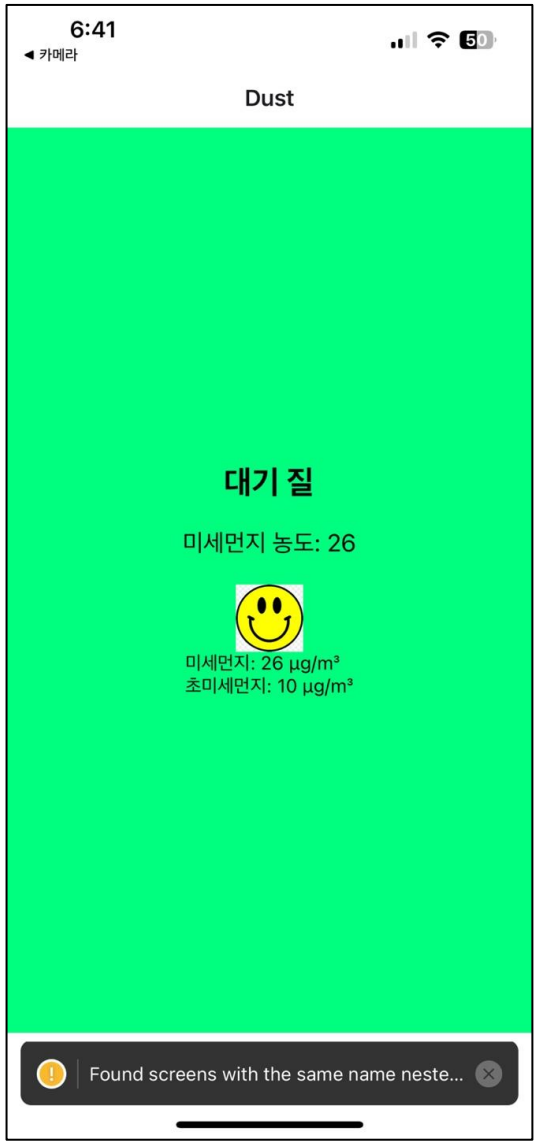
미세먼지 측정소 이름

개요

기능

회고

기능 : 대기질



※ 지역 제한 : 서울특별시 송파구

농도에 따른 UI 변화

~ 30	30 ~ 80	80 ~	Value
			Image
			BackgroundColor

개요

기능

회고

시연 영상

개요

기능

회고

파트 별 회고 : API 호출 컴포넌트

Good ↑

OPEN-API를 사용하여
데이터 호출 및 가공을 해보는
좋은 경험을 함

Bad ↓

미세먼지 정보도 지역 별로
구현했다면 좋았을 듯

API호출 부분을 기능별 컴포넌트로
나누지 않고 통합 했다면 좋았을 듯

개요

기능

회고

파트 별 회고 : 메인 화면 구현, 기능 통합

Good ↑

팀원들이 구현한 컴포넌트 및
화면들을 통합하며 제대로 된
프로젝트 협업 경험을 한 느낌

Bad ↓

한 학기 동안 많은 부분을 배웠는데
적용하지 못한 기술이 있어
다양한 방법으로 구현할 수 있도록 하고싶음

개요

기능

회고

파트 별 회고 : 탭, 화면 별 기능 구현

Good ↑

OPEN-API 파트와 소통하며
개발하는 좋은 경험을 함

백엔드와 협업하는 느낌이라
좋았음

Bad ↓

React Native를 처음 다뤄보면서
어려운 부분이 있었음

강의 중 배운 내용들을 프로젝트에서
전부 사용해보지 못해 아쉬움

질의응답

감사합니다