Balance Place

서울시 인구 현황과 날씨를
Balance Place에서
한번에 확인하자!

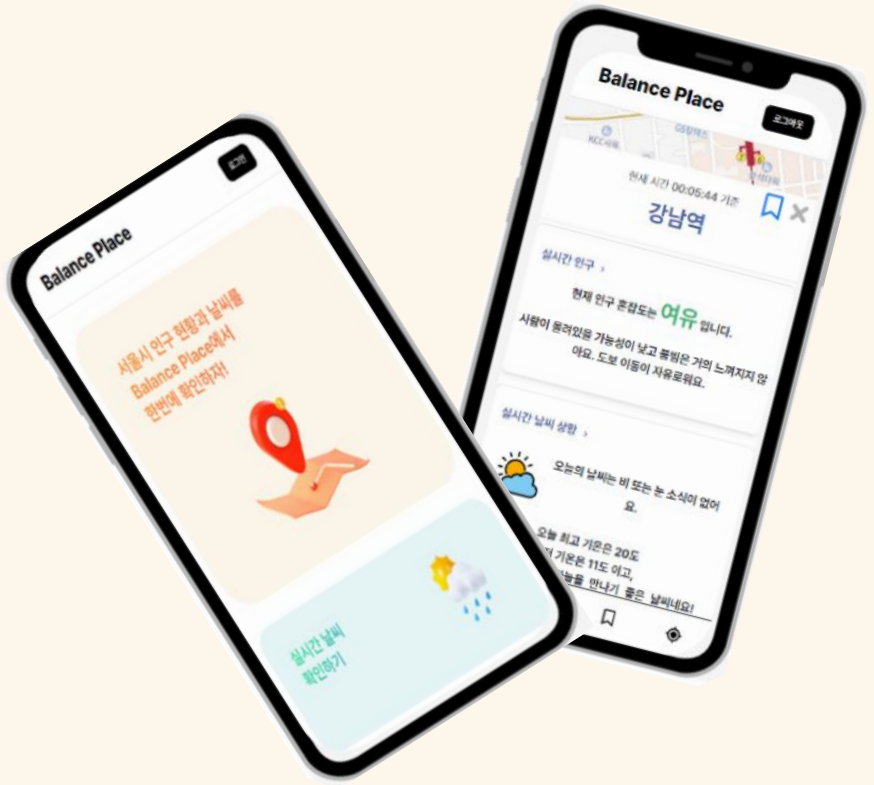
Balance Place

서비스 아키텍쳐

주요기능

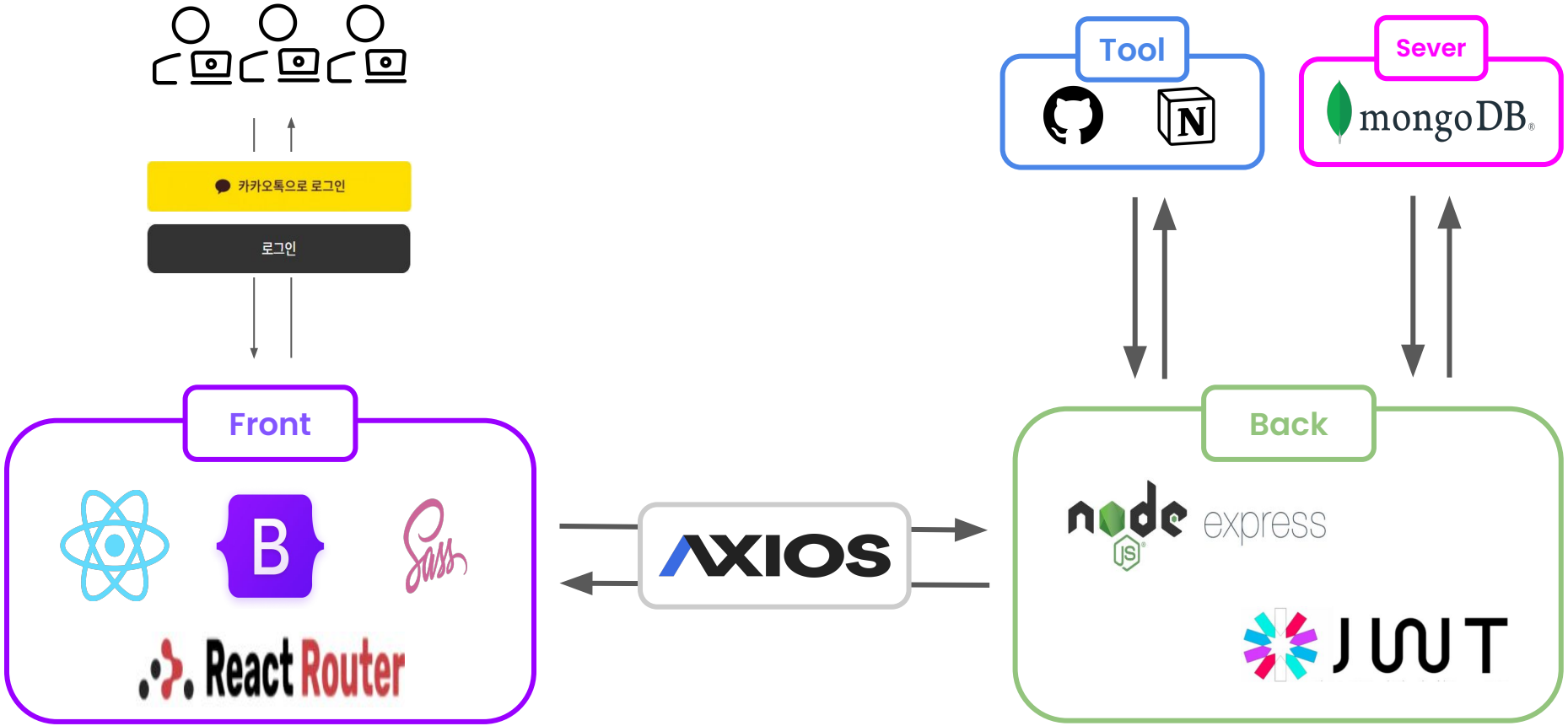
기술 커뮤니케이션

기술명세



Balance Place

서비스 아키텍쳐



Balance Place

주요 기능

홈

- 소셜로그인 (카카오톡)
- 50개 지역구 카테고리별 분류

댓글

- 마커 클릭시 해당 장소 자동 저장
- 사용자가 위치한 곳에 대한 후기 작성

정보

- 서울시 공공API 를 활용
- 지역구 내에 핫 스팟 실시간 날씨, 인구분포 정보 제공

위치

- geoFence 기능
- geoLocation API로 사용자 위치 추적

북마크

- 주로가는 장소 즐겨 찾기 형태로 저장해 해당 장소의 인구분포와 날씨 파악
- 북마크 등록 및 삭제 구현

알림

- 회원가입 시 입력한 전화번호를 찾아 메시지로 알림 기능
- 카카오 로그인한 계정에 나에게 보내는 메시지로 알림 기능

Balance Place

서비스 소개

프로젝트 전	선택지	프로젝트 적용
짧은 기간 내에 반응형 까지 완성 목표	• **Bootstrap** 적용 : 패키지 설치에 따른 반응형 컨테이너 적용 • **React Media Query** 적용 : **Viewport** 너비를 설정해서 그에 따른 다른 스타일 적용	다양한 방법으로 구현하기 위해서 디자인 충돌 나지 않는 선에서 **Bootstrap** 과 **React Media Query** 사용해서 반응형 구축

프로젝트 전

서울시 주요 **50**곳의 공공데이터 **api** 가져오기

가져온 공공데이터 가공

선택지

read-excel-file
엑셀파일 읽을 수있는 모듈

xml12js모듈의의 **parseString** 메소드를 이용하여 **json**형식으로 변환

프로젝트 적용

엑셀파일에 지역명, 위도, 경도, 이미지링크 등 필요 데이터 정리 후 **read-excel-file** 모듈을 사용하여 데이터를 배열 형식으로 담아 프론트 요청에 응답

xml2js모듈의 **parseString** 메소드를 이용하여 **json**형식으로 변환 후 공공데이터 중 **kt**의 실시간 인구밀도 데이터, 지역이름, 당일 날씨, 시간별 날씨 데이터 가공

```
    return (
      <AppContext.Provider
        value={{
          isSidebarOpen,
          sidebarCategory,
          wantMyLocation,
          endPoint,
          openSidebar,
          closeSidebar,
          isNeedMyLocation,
          changeEndPoint,
        }}
      >
        {children}
      </AppContext.Provider>
    );
```

문제

컴포넌트간 공유해야할 상태값이 많아짐

```
import { AppProvider } from "./components/MapPage/Context";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <BrowserRouter>
    <NavermapsProvider ncpClientId={NAVER_MAP_API}>
      <AppProvider>
        <App />
      </AppProvider>
    </NavermapsProvider>
  </BrowserRouter>
);
```

해결

useGlobalContext를 사용하여
App Provider 컴포넌트에서 정의한
상태값과 함수를, **AppContext.Provider**로
감싸인 자식 컴포넌트에서 사용할 수 있다.

```
const getData = async () => {
  const res = await axios.get('http://localhost:4000/nameData');
  if (res.status !== 200) console.log('데이터 수신 실패');

  // 전체 분류 데이터
  const allDataBase = res.data.Arr;
  // // 첫번째 전체 데이터
  const firstNameData = res.data.Arr[0].name;
  const firstImgData = res.data.Arr[0].img;
  const firstLatitudeData = res.data.Arr[0].latitude;
  const firstLogitudeData = res.data.Arr[0].longitude;

  setAllDataBase((cur) => allDataBase);
  setAllData((cur) => firstNameData);
  setAllImgData((cur) => firstImgData);
  setAllLatitudeData((cur) => firstLatitudeData);
  setAllLongitudeData((cur) => firstLogitudeData);
};

useEffect(() => {
  getData();
}, []);
```

문제

-백엔드가 라우팅해놓은은 주소로 요청 날리기
- 응답으로 받아온 데이터를 사용하기

해결

async-await 구문 과 **axios** 를 활용하여 요청

useState를 사용하여 응답 데이터 정의

```jsx
<ul>
  {bookmarks.length > 0 ? (
    bookmarks.map((el, idx) => (
      <li key={idx}>
        <div>
          <div>{el?.area}</div>
          <div>실시간 날씨🔥 - {el?.weather?.pcp_msg}</div>
          <div>
            실시간 인구 혼잡도 👥 -{" "}
            <span // 붐빔도 레벨로 컬러 색상 지정
              style={{ fontSize: "18px" }}
              className={`report-crowded ${
                el?.data?.AREA_CONGEST_LVL[0] === "여유"
                  ? "green"
                  : el?.data?.AREA_CONGEST_LVL[0] === "보통"
                  ? "yellow"
                  : el?.data?.AREA_CONGEST_LVL[0] === "약간 붐빔"
                  ? "orange"
                  : "red"
              }`}
            >
              {data?.AREA_CONGEST_LVL[0]}
            </span>{" "}
          </div>
          <button onClick={() => handleBookmarkDelete(idx)}>
            삭제
          </button>
```

```jsx
// 북마크 삭제
const handleBookmarkDelete = (idx) => {
  const newBookmarks = [...bookmarks];
  newBookmarks.splice(idx, 1);
  setBookmarks(newBookmarks);
  // 삭제버튼이 눌리면 북마크 아이콘 초기화
  if (bookMarkIcon) {
    setbookMarkIcon(false);
  }
};
```

```
//각 마커 아래에 표시되어있는 원의 바운더리를 구하기 위해 그냥 중심좌표와 반지름이 똑같은 원을 새로 만들어버렸다.
//왜냐하면 네이버맵스 Circle컴포넌트는 프로퍼티로 getBounds 메서드를 제공하지 않기 때문이다
// makeMarkerBoundary 함수 안에서 상속받은 좌표값을 파라미터로 받아서 활용
const getboundary = () => {
  locationData?.map((item) => {
    const boundary = new navermaps.Circle({
      map: map,
      fillOpacity: 0,
      fillColor: null,
      strokeColor: null,
      center: new navermaps.LatLng(item[0], item[1]),
      radius: 400,
    }).getBounds();

    const path = [
      new navermaps.LatLng(boundary._ne.y, boundary._sw.x), // 왼쪽 위
      new navermaps.LatLng(boundary._ne.y, boundary._ne.x), // 오른쪽 위
      new navermaps.LatLng(boundary._sw.y, boundary._ne.x), // 오른쪽 아래
      new navermaps.LatLng(boundary._sw.y, boundary._sw.x), // 왼쪽 아래
    ];
    checkMyLocationByGeofence(path, item[2]);
  });
};
```

```
<Circle
  key={index}
  center={new navermaps.LatLng(coordinate[0], coordinate[1])}
  radius={400}
```

문제**1**

geofence를 위해서는 검사를 위한 구역이 설정되어 있어야함

문제**2**

Circle컴포넌트에서 **getBounds** 메소드 지원 **x**

해결

Circle메소드로 중심축과 반지름이 똑같은 원을 새로 그려주고 **getBounds**메소드 사용 및 **geofence**검사를 위한 구역 설정

```
const checkMyLocationByGeofence = (path, area) => {
  navigator.geolocation.getCurrentPosition(
    (position) => {
      const { latitude, longitude } = position.coords;

      const testX = 126.93689331765; //longitude
      const testY = 37.5578078217728; //latitude

      if (
        testX > path[0].x &&
        testX < path[1].x &&
        testY < path[1].y &&
        testY > path[2].y
      ) {
        console.log('찾았다', area);
        if (isLogin === '로그인' && area !== previousArea) {
          sendKakaoAccessToken(area);
          previousArea = area;
        } else {
          console.log('로그인이 필요합니다');
        }
      }
    },
    (error) => {
      console.log(error);
    },
    { timeout: 5000, maximumAge: 0 }
  );
};
```

- **geofence**를 위한 검사 로직 구현

- 이전 지역과 비교하여 지역이 같으면 알림서비스 함수 실행 **x**

```
const loginUser = async (req, res) => {
  const { email, password } = req.body;

  //알림 기능을 위한 전역변수 변경
  userID = email;
  isNormalUserLogined = true;
  try {
    // body에 담아서 보내준 email을 db에서 확인
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(403).json({
        loginSuccess: false,
        message: '해당되는 이메일이 없습니다.',
      });
    }

    // 해싱 암호화한 비밀번호 대조
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(403).json({
        loginSuccess: false,
        message: '비밀번호가 틀렸습니다.',
      });
    }

    const token = jwt.sign({ email: user.email }, ACCESS_SECRET, {
      expiresIn: '7d',
    });
    user.token = token;
    await user.save();

    return res
      .status(200)
      .json({ loginSuccess: true, email: user.email, token });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ error: 'something wrong' });
```

```
const registerUser = async (req, res) => {
  try {
    const { email, phone, password, passwordCheck } = req.body;
    // 빈값이 오면 팅겨내기
    if (
      email === '' ||
      phone === '' ||
      password === '' ||
      passwordCheck === ''
    ) {
      return res.json({ registerSuccess: false, message: '정보를 입력하세요' });
    }
    const sameEmailUser = await User.findOne({ email });
    if (sameEmailUser !== null) {
      return res.json({
        registerSuccess: false,
        message: '이미 존재하는 이메일입니다',
      });
    }
    const samePhoneUser = await User.findOne({ phone });
    if (samePhoneUser !== null) {
      return res.json({
        registerSuccess: false,
        message: '이미 존재하는 닉네임입니다.',
      });
    }
    // 솔트 생성 및 해쉬화 진행
    const salt = await bcrypt.genSalt(saltRounds);
    const hashedPassword = await bcrypt.hash(password, salt);
    const user = new User({
      email,
      phone,
      password: hashedPassword,
    });
    await user.save();
    return res.json({ registerSuccess: true });
```

```
app.use("/data", dataRouter);
```

↓

```
router.post('/getdata', fetchData);

module.exports = router;
```

↓

```
//프론트에서 요청body에 담아 보낸 지역엔드포인트를 변수에 저장
//마커 클릭으로 받아오는 지역엔드포인트는 문자 앞에 공백이 하나 있기때문에 trim()을 통해 공백제거
const END_POINT = req.body.point.trim();

console.log(END_POINT);

const AREA_END_POINT = `http://openapi.seoul.go.kr:8088/${DATA_API_KEY}/xml/citydata/1/5/${END_POINT}`;

try {
  const response = await fetch(AREA_END_POINT, {
    method: 'GET',
    headers: {
      'Content-Type': 'application/xml',
    },
  });

  const rawData = await response.text();
```

```
  const result = await new Promise((resolve, reject) => {
    parseString(rawData, (err, result) => {
      if (err) reject(err);
      else resolve(result);
    });
  });
  //KT의 실시간 인구밀도 데이터
  const liveData =
    result['SeoulRtd.citydata']['CITYDATA'][0].LIVE_PPLTN_STTS[0]
      .LIVE_PPLTN_STTS[0];
  //지역 이름
  const areaName = result['SeoulRtd.citydata']['CITYDATA'][0].AREA_NM;
  //당일 전체적인 날씨
  const dayWeather =
    result['SeoulRtd.citydata']['CITYDATA'][0].WEATHER_STTS[0]
      .WEATHER_STTS[0];
  //당일 시간별 날씨
  const timeWeather =
    result['SeoulRtd.citydata']['CITYDATA'][0].WEATHER_STTS[0].WEATHER_STTS[0]
      .FCST24HOURS[0];

  //프론트에서 필요한 데이터만 추린 모델
  const model = {
    area_name: areaName[0],
    live_data: liveData,
  };

  const weatherModel = {
    temperature: dayWeather.TEMP[0],
    sen_temperature: dayWeather.SENSIBLE_TEMP[0],
    min_temperature: dayWeather.MIN_TEMP[0],
    max_temperature: dayWeather.MAX_TEMP[0],
    pcp_msg: dayWeather.PCP_MSG[0],
    air_idx: dayWeather.AIR_IDX[0],
    fcst_24hours: timeWeather,
  };
```

```
const findPhoneNumber = async (req, res) => {
  const kakaoAccessToken = req.body.kakao_access_token;
  const area = req.body.area;

  //이메일 형식의 유저 아이디를 컨트롤러 상단의 전역변수에서 받아와서 DB에서 해당 유저정보를 가져옴
  const user = await User.findOne({ email: userID }).select('phone');
  const phone = user?.phone;

  if (isNormalUserLogined || userID !== undefined || kakaoAccessToken) {
    try {
      // 핸드폰 번호가 존재하면 알림문자전송 모듈에 인자로 전달
      if (phone || kakaoAccessToken) {
        simpleNotification(phone, kakaoAccessToken, area);
      }
    } catch (err) {
      console.error(err.message);
      res.status(500).json('알 수 없는 오류가 발생했습니다.');
    }
  } else {
    const errorMessage =
      '로그인 미들웨어에서 처리가 안됐거나 카카오엑세스 토큰값 확인 불가';
    console.error(errorMessage);
    res.status(500).json(errorMessage);
  }
};
```

- 로그인 미들웨어에서 변경시켜주는 전역변수로 유저 전화번호 조회

- 카카오 로그인, 일반 로그인 모두 보낼 수 있도록 처리

감사합니다