프로젝트 기술서

<table>
<tr><td>프로젝트 명</td><td colspan="2">SafeHaven
(재난 상황에 따른 대피소 안내 서비스)</td><td>개발기간</td><td>2023.10.04. ~
2023.12.01</td></tr>
<tr><td rowspan="6">구현기술</td><td colspan="2">Java, Spring Boot, JPA</td><td rowspan="2">개발 장소</td><td rowspan="2">한국직업전문학교</td></tr>
<tr><td colspan="2">React, HTML, CSS3, JavaScript</td></tr>
<tr><td colspan="2">SQL</td><td rowspan="3">투입인원</td><td rowspan="3">3명</td></tr>
<tr><td colspan="2">python</td></tr>
<tr><td colspan="2">ELK</td></tr>
<tr><td colspan="2">오픈API(Google Maps, 공공데이터)</td></tr>
<tr><td rowspan="2">환경</td><td>개발 환경</td><td>OS</td><td>FrameWork</td><td>DataBase</td></tr>
<tr><td>Visual Studio Code

Eclipse

ElasticSearch Dev Tools</td><td>Windows 10 & 11</td><td>Spring Boot, React</td><td>MariaDB</td></tr>
<tr><td>Project시연 Github 링크</td><td colspan="4">https://github.com/companykim/portfolio
https://github.com/companykim/portfolio-server</td></tr>
<tr><td rowspan="2">프로젝트 소개</td><td colspan="4">[프로젝트의 시작]

1. 사전조사
지난 6월 대한민국에 공습 경보가 발의된 적이 있음. 또한, 우리나라에 계속해서 지진과 같은 재난이 발생하고 있기에 주위에 대피소가 어딨는지 미리 파악하고 있어야 한다. 하지만 우리는 내 위치 주변에 있는 대피소가 어딨는지 잘 알지 못한다.

2. 프로젝트 개발방향
우리 SafeHaven팀은 총 2가지의 큰 목표를 개발 방향으로 정하고 프로젝트에 임하였다.

1) 대피소 경로 안내
재난 상황에서 개인이 해당 지역에서 발생한 재난 유형에 따라 인접 대피소를 신속하게 찾을 수 있도록 안내

2) 대피 요령 및 대피 물품 대비
재난 상황 별 대피 요령 및 현재 위치를 기점으로 구호 물품을 구비 가능한 장소를 지도에 표현</td></tr>
<tr><td colspan="4">[프로젝트의 주요기능]

1. 공공데이터 API를 통해 대피소 위치 정보를 DB에 저장하기 위한 서버 제작

2. 대피소 위치 정보를 불러와 구글맵에서 마커로 표시

3. 내 위치 정보를 불러와 마커로 표시하고 내 위치에 있는 슈퍼마켓, 약국, 병원과 같은 주변 장소들을 마커로 표시</td></tr>
</table>

<table>
<tr><td></td><td>

4. 구글 장소 라이브러리로 장소명을 검색하고 클릭하면 그 검색한 장소가 마커로 표시됨.

5. 내 위치에서 선택한 도착지까지의 경로를 안내하는 기능

6. 네이버 뉴스 기사를 크롤링하고 그 목록을 DB에 저장

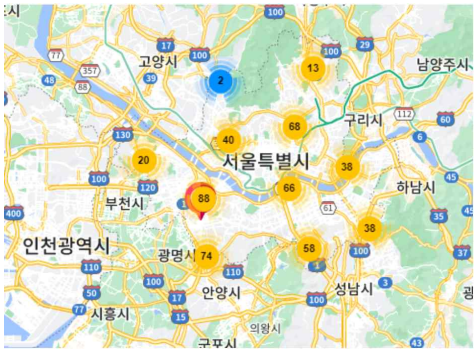
7. DB에 있던 기사 목록을 ELk에 불러와 재난과 관련된 뉴스 기사를 자동 검색하는 API 구현

</td></tr>
<tr><td>본인의 역할</td><td>

[구성 및 흐름]

[프로젝트 구현 결과]

1. 대피소 위치 정보를 DB에 저장할 서버 제작 & 지도 화면에 대피소 마커 구현
DB에 저장된 대피소 위치 정보를 불러와 React로 지도 화면과 마커를 구현
또한, 대피소를 유형별로 나누어 유형에 맞는 대피소를 탐색하도록 구현함.



<table>
<tr><td>View</td><td>[ShelterMarkersGoogle.js]

```
// 파라미터: 함수에서 정의되어 사용되는 변수
export default function ShelterMarkersGoogle({ center, zoomLv, scale, clusterer, setDestPoint }) {
    const displayLv = zoomLv * parseInt(100 / 14)
    const halfBoundary = scale * 100000
    const [usageType, setUsageType] = useState(null);
    const shelterUri = `http://localhost:8080/shelter/${usageType}/${center.latitude}/${center.longitude}/${displayLv}/${halfBoundary}`;
```

(...생략...)</td></tr>
</table>

</td></tr>
</table>

```
// 대피소 목록 마커 출력
function RenderSuccess(shelterList) {
    return (
        <>
            {shelterList?.map((shelter) => (
                <>
                    <MarkerF
                        position={shelter.shelterId}
                        onClick={() => handleActiveShelter(shelter)}
                        onRightClick={() => handleInfo(shelter)}
                        icon={{
                            url: "https://cdn-icons-png.flaticon.com/128/5587/5587696.png", // 마커이미지의 주소
                            scaledSize: {
                                width: 30,
                                height: 30,
                                } // 마커이미지의 크기입니다
                        }}
                        clusterer={clusterer}
                    />
```

(...생략...)

```
return (
    <>
    <div style={{ position: 'absolute', top: '5px', left: '5px', zIndex: 1000 }}>
       <DropdownButton id="dropdown-basic-button" title="대피소 유형" onSelect={setShelterUsetype}>
           <Dropdown.Item eventKey="지진-옥외">지진-옥외</Dropdown.Item>
           <Dropdown.Item eventKey="지진-실내">지진-실내</Dropdown.Item>
           <Dropdown.Item eventKey="이재민임시">이재민 임시</Dropdown.Item>
       </DropdownButton>
    </div>

       {usageType &&
           <Fetch uri={shelterUri} renderSuccess={RenderSuccess} />
       }
    </>
);
```

[ShelterMapGoogle.js]

```
{/* 클러스터링 */}
<MarkerClusterer>
  {clusterer =>
    <ShelterMarkersGoogle center={curPos} zoomLv={zoomLv} scale={50} clusterer={clusterer} setDestPoint={setDestPoint} />
  }
</MarkerClusterer>
```

WebClient

```
@Service
@PropertySource("classpath:application.properties")
public class WebClient4Shelter {
    // 지진-옥외
    private String seoulKey = [redacted];

    @Autowired
    private ShelterService shelterService;

    public void loadShelter() {
        WebClient webClient = WebClient.builder().baseUrl("http://openapi.seoul.go.kr:8088")
                .defaultHeader(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE).build();

        String result = webClient.get().uri("/" + seoulKey + "/json/TlEtqkP/1/1").retrieve().bodyToMono(String.class)
                .block();

        Map<String, Object> map = this.jsonToMap(result);

        int seoulShelterTotalCount = (int) ((Map) map.get("TlEtqkP")).get("list_total_count");
        int step = 500;
        for (int i = 1; i < seoulShelterTotalCount; i += step) {
            String subUri = "/" + seoulKey + "/json/TlEtqkP/" + i + "/" + (i + step);
            result = webClient.get().uri(subUri).retrieve().bodyToMono(String.class).block();
            map = this.jsonToMap(result);

            List seoulShelters = (List) ((Map) map.get("TlEtqkP")).get("row");

            List<ShelterVO> seoulSheltersVo = new ArrayList<>();

            for (Object obj : seoulShelters) {
                Map shelter = (Map) obj;
                ShelterId id = new ShelterId(Float.parseFloat((String) shelter.get("YCORD")),
                        Float.parseFloat((String) shelter.get("XCORD")));
                ShelterVO shelterVO = new ShelterVO(id, (String) shelter.get("EQUP_NM"),
                        (String) shelter.get("LOC_SFPR_A"), "지진-옥외");

                seoulSheltersVo.add(shelterVO);
```

Controller

```
@RestController // Container에 담기도록 지정
@RequestMapping("/shelter")
public class ShelterController {
    @Autowired
    private ShelterService shelterService;

    @Autowired
    private WebClient4Shelter webClient4Shelter;

    // /shelter/
    // halfBoundary : meter 단위로
    @GetMapping("/{useType}/{lat}/{lng}/{displayLv}/{halfBoundary}")
    public ResponseEntity<List<ShelterVO>> get(@PathVariable String useType, @PathVariable float lat,
            @PathVariable float lng, @PathVariable int displayLv, @PathVariable int halfBoundary) {
        List<ShelterVO> result = shelterService.listShelter(useType, lat, lng, displayLv, halfBoundary);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }
```

VO

```java
@Getter
@Entity
@Table(name="t_shelter")
@NoArgsConstructor
public class ShelterVO {
    @EmbeddedId
    private ShelterId shelterId;
    private String name;
    private String address;
    private String useType;
    private int displayLv;

    public ShelterVO(ShelterId id, String name, String address, String useType) {
        shelterId = id;
        this.name = name;
        this.address = address;
        this.useType = useType;
        displayLv = (int) ((Math.random() * (100-1 + 1)) +1); // 관리자가 설정해서 저장
    }

    @Override
    public String toString() {
        return "ShelterVO [shelterId=" + shelterId + ", name=" + name + ", address=" + address + ", useType=" + useType
                + ", displayLv=" + displayLv + "]";
    }
}
```

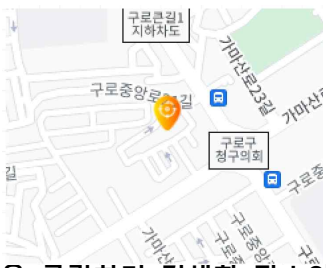
Service

```java
@Service
public class ShelterService {
    @Autowired
    private ShelterRepository shelterRepository;

    public int uploadShelter(List<ShelterVO> listShelterVO) {
        // 중복 데이터
        int dupCount = 0;
        for (ShelterVO obj : listShelterVO) {
            try {
                shelterRepository.save(obj);
            } catch (Exception e) {
                dupCount++;
            }
        }
//      System.out.println("Shelter 입력 중 " + dupCount + "건의 중복 데이터 발생하여 이는 무시함");
        return 0;
    }
```

Repository

```java
public interface ShelterRepository extends JpaRepository<ShelterVO, String>{
    @Query(nativeQuery = true, value = "select * from T_shelter "
            + "where use_type like :useType "
            + "and lat < :northBound "
            + "and lat > :southBound "
            + "and lng > :westBound "
            + "and lng < :eastBound "
            + "and display_lv >= :displayLv")
    List<ShelterVO> findShelter(String useType, float westBound, float eastBound,
            float northBound, float southBound, int displayLv);


}
```

2. 장소를 검색하고 그 결과를 지도에 마커로 구현



장소명을 검색하고 그것을 클릭하면 검색한 장소의 마커가 지도 하면에 출력되도록 구현함.

View

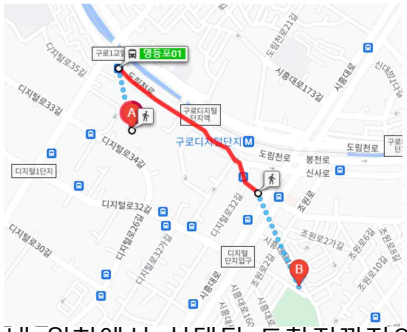
[AutoSearchPlaces.js]

```javascript
// 장소 자동완성
export default function AutoSearchPlaces({ setPlacelating }) {

    const autoCompleteRef = useRef()
    const inputRef = useRef()

    const options = {
        componentRestrictions: { country: "kr" }, // 예측 결과가 한국 시설로만 제한
        fields: ["address_components", "geometry", "icon", "name"],
        types: ["establishment"]
    }

    useEffect(() => {
        autoCompleteRef.current = new window.google.maps.places.Autocomplete(
            inputRef.current,
            options
        );
        autoCompleteRef.current.addListener("place_changed", async function () {
            const place = await autoCompleteRef.current.getPlace();
            console.log("위도", place.geometry.location.lat());
            console.log("경도", place.geometry.location.lng());
            setPlacelating({ lat: place.geometry.location.lat(), lng: place.geometry.location.lng() })
        });
    }, []);

    return (
        <div>
            <input ref={inputRef} placeholder='장소를 입력하세요...' />
            <span></span>
        </div>
    );
```

[ShelterMapGoogle.js]

```
{/* 검색한 장소 띄우는 마커 */}
{placelating &&
  //console.log(placelating)
  <MarkerF
    position={{
      lat: placelating.lat,
      lng: placelating.lng
    }}

    icon={{
      url: "https://cdn-icons-png.flaticon.com/128/6781/6781647.png",
      scaledSize: {
        width: 30,
        height: 30
      }
    }}
  >
  </MarkerF>
}
```

3. 경로 탐색 기능



내 위치에서 선택된 도착지까지의 경로를 탐색하는 기능을 구현함.

View

[Direction.js]

```
useEffect(() => {
    pointCounter.current = 0;
    console.log(origin, destination);
}, [origin.lat, origin.lng, destination.lat, destination.lng]);

// 경로 출력
const directionsCallback = (result, status) => {
    console.log(result, status);
    if (status === 'OK' && pointCounter.current === 0) {
        pointCounter.current += 1;
        setDirections(result);
    }
}

return (
    <>
        <DirectionsService
            options={{ origin, destination, travelMode: 'TRANSIT' }}
            callback={directionsCallback}
        />

        <DirectionsRenderer directions={directions} options={LineStyles} />

    </>
);
```

[ShelterMapGoogle.js]

```
{swichDirections && destPoint ?
  <ViewDirections origin={{ lat: 37.4854799, lng: 126.8981862 }} destination={destPoint} />
  : <></>
}

<NearPlaces lat={37.4854799} lng={126.8981862} radius={2000} setDestPoint={setDestPoint} />
```

4. 뉴스 자동 검색 API

DB에 저장되어 있던 뉴스 기사들을 불러와 ELK를 통해 재난 키워드에 맞는 기사들을 추출해 화면에 출력되게 함.

View

```
export default function Home() {
    const listNewsUri = 'http://localhost:8080/elastic/anonymous';
```

(...생략...)

<table>
<tr><td>기사
경주 7년 만에 또 지진...첨
경북 경주서 새벽잠 깨운 규
전국 곳곳서 "여기도 흔들
"건물이 좌우로 흔들렸다"...
"2016년보다 진도 작지만 강한 지진, 수능 또 지진, 수민 불안
경주 7년 만에 또 지진... 새벽 굉음에 주민들 '화들짝'
새벽 경북 경주서 규모 4.0
[속보] 경북 경주에서 규모</td><td>

```
<table>
    <thead>
        <tr>
            <th>기사</th>
            <th>날짜</th>
        </tr>
    </thead>
    <tbody>
        <Fetch uri={listNewsUri} renderSuccess={RenderSuccess} />
    </tbody>
</table>
```

(...생략...)

```
function RenderSuccess(newsList) {
    console.log("newsList", newsList);

    // 데이터가 없거나 hits가 없다면 빈 배열을 반환
    const hits = newsList?.hits?.hits || [];

    return (
        <>
            {hits.map((hit, index) => (
                <tr key={index}>
                    <td>{hit._source.newstitle}</td>
                    <td>{hit._source.newsdate}</td>
                </tr>
```

</td></tr>
<tr><td>WebClient</td><td>

```
@Service
public class WebClient4News {
    public String callElasticSearch(String urlParam) {
        urlParam = "/news*/_search?q=newstitle=\"지진\" and newstitle=\"홍수\" and newstitle=\"해일\"";
        WebClient webClient = WebClient.builder().baseUrl("http://localhost:9200")
                .defaultHeader(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE).build();
        String result = webClient.get().uri(urlParam).retrieve().bodyToMono(String.class).block();
        return result;
    }
}
```

</td></tr>
<tr><td>Controller</td><td>

```
@RequestMapping("/elastic")
public class NewsController {
    @Autowired
    private WebClient4News service;

    @GetMapping("/anonymous")
    public ResponseEntity<String> adapt(String url) {
        return new ResponseEntity<>(service.callElasticSearch(url), HttpStatus.OK);
    }
```

</td></tr>
<tr><td>Config</td><td>

```
input {
  jdbc {
    jdbc_driver_library => "C:/logstash-7.10.1/lib/mariadb-java-client-3.1.4.jar"
    jdbc_driver_class => "org.mariadb.jdbc.Driver"
    jdbc_connection_string => "jdbc:mariadb://localhost:3306/bbdb?allowMultiQueries=true"
    jdbc_user => "[redacted]"
    jdbc_password => "[redacted]"
    #실행빈도 (1분마다)
     schedule => "*/1 * * * *"
    statement => "SELECT newstitle, newsdate FROM news_crawling"
    clean_run => true
  }
}
filter {
  mutate {
    add_field => {
      "id" => "%{newstitle}_%{newsdate}"
    }
  }
}

output {
  elasticsearch {
      hosts => ["localhost:9200"]
      index => "news"
      document_id => "%{id}"
  } stdout {
   codec => rubydebug
  }
}
```

</td></tr>
</table>

프로젝트 결과분석

React-Native를 학습해서 모바일에서도 대피소를 안내받을 수 있도록 구현하고 싶음.
경로를 탐색할 때 여러 경로가 검색되게 하여 그중 최단 거리를 고를 수 있는 기능을 구현하고 싶음.
기사가 화면에 출력될 때 경고 알림을 설정하고 싶음.