

상 장

우수

작품명 : 행사 플래닛

성 명 : 이승표

위 학생은 제29회 정보통신융합공학부
종합학술제에서 위와 같이 우수한 성적을
거두었기에 이에 상장을 수여함.

2023년 12월 12일

목원대학교 정보통신융합공학부장



행사 Planet

제29회 재학생 (3학년) 발표회 - 목원대 정보통신공학과

정보통신공학과
1960039 이승표

A solid red vertical bar.

목차

1. 작품 소개
2. 작품 설계
3. 작품 구현

1. 행사 플래닛 소개 - 구현 내용

전국에 있는 모든 행사, 축제를 한눈에 알아볼 수 있는 웹 사이트
구현 내용

DB

- 행사 추가, 수정, 삭제
- 회원가입을 이용한 계정 추가, 수정, 삭제

크롤링

- 최근에 열린 행사목록 확인 기능

API

- 필터를 통해 월별 또는 특정 지역의 행사 확인 기능
- 키워드로 원하는 행사의 검색 기능
- 지도를 통한 행사장의 위치 확인 기능

1. 행사 플래닛 소개 - 구현 내용

전국에 있는 모든 행사, 축제를 한눈에 알아볼 수 있는 웹 사이트
구현 내용

소켓

- 소켓을 이용해 DB에 행사를 추가, 수정, 삭제했을 때 실시간으로 달력에 보여지는 기능

VUEX

- VUEX를 통해 로그인 사용자의 이름과 로그인상태 데이터를 공유하고 사용.

1. 행사 플래닛 소개 - 개발 환경

Server-side (Backend)

node.js

Express Framework

Database (mongoDB)

Client-side (Frontend)

html, css, javaScript

Vue.js Framework

1. 행사 플래닛 소개 - 시스템 구성도

- **Legacy Server**

- 한국관광공사_국문 관광정보 서비스_GW
- Google Maps
- OpenWeatherMap
- Etc

- **Server**

- node.js

- **Client**

- Browser/PC

- **DataBase**

- MongoDB

1. 행사 플래닛 소개 - 관리자 시나리오

행사 확인 서비스 시나리오

- 계정 관리 : 관리자 계정들의 정보들을 삭제하거나 수정할 수 있습니다.
- 이달 진행중인 축제, 행사목록 확인 기능

1. 행사 플래닛 소개 - 사용자 시나리오

행사 확인 서비스 시나리오

1. 사용자는 가장 최근에 업데이트 된 행사들을 간략히 확인 가능
2. 지역필터를 통해 원하는 지역의 행사들만 보기 가능
3. 특정지역의 특정한 달에 어떠한 행사들이 진행되는지 캘린더 형태로 확인가능
4. 검색된 행사들이 어디서 열리는지 확인 가

2. 작품 설계 - 데이터 흐름

- 데이터 흐름

- ① 한국관광공사에서 제공하는 관광정보 API에서 정보들 가져온다
- ② 최근 행사정보들은 한국관광공사 또는 네이버에서 크롤링해 가져온다
- ③ 사용자 접속 시 크롤링한 데이터를 보여주고 필터 설정 시 API에서 필요한 데이터만 수집해 가져온다.
- ④ 행사를 클릭할 시 해당 행사의 장소를 구글 맵으로 보내 지도에 표시해준다.
- ⑤ 사용자가 달력태그에서 특정한 지역 선택 시 API를 통해 가져온 그 지역의 행사 정보와 관리자가 추가한 행사들의 데이터를 DB에서 뽑아와 달력에 표현

2. 작품 설계 - 관리자 UI 설계

- 화면에서 수정할 수 있도록 표시할 주요 항목
 - 관리자 계정 테이블
 - 이름
 - 아이디
 - 비밀번호
 - 주소
 - 행사 테이블
 - 번호
 - 이름
 - 장소
 - 시작일
 - 마감일
 - 내용

2. 작품 설계 - 사용 UI 설계

- 화면에 표시할 주요 항목
 - 최신 올라온 행사 정보
 - 행사 명
 - 위치
 - 시작일
 - 마감일
 - 행사 위치를 볼 수 있는 지도
 - 해당 월의 행사 계획을 볼 수 있는 달력
 - 검색을 통한 행사 필터 목록
 - 카테고리를 이용한 지역 행사 필터 목록

3. 작품 구현 - 메인 페이지

문화체육관광부 홈페이지에서
최근 행사들의 전체 데이터를
크롤링을 해 페이지마다 5개씩
보여준다.



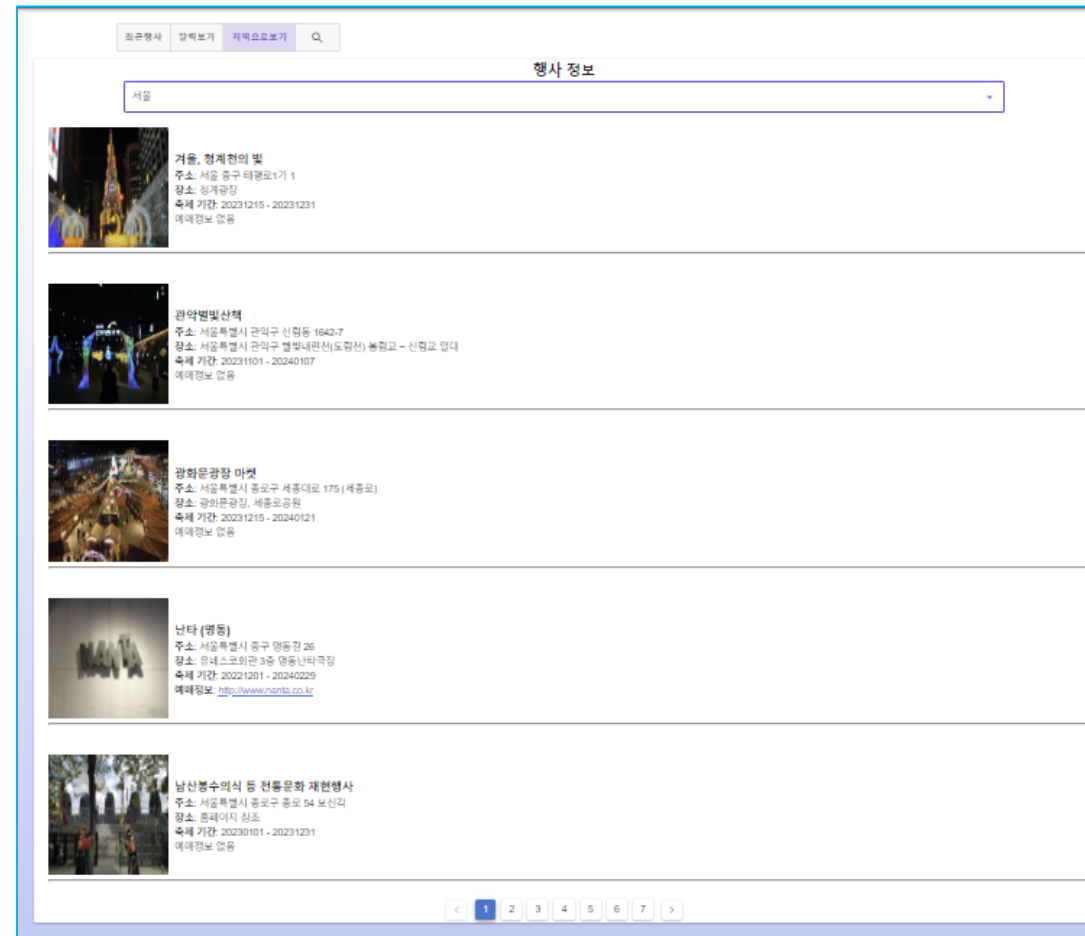
3. 작품 구현 - 메인 페이지

문화체육관광부 홈페이지에서
최근 행사들의 전체 데이터를
크롤링을 해 페이지마다 5개씩
보여준다.

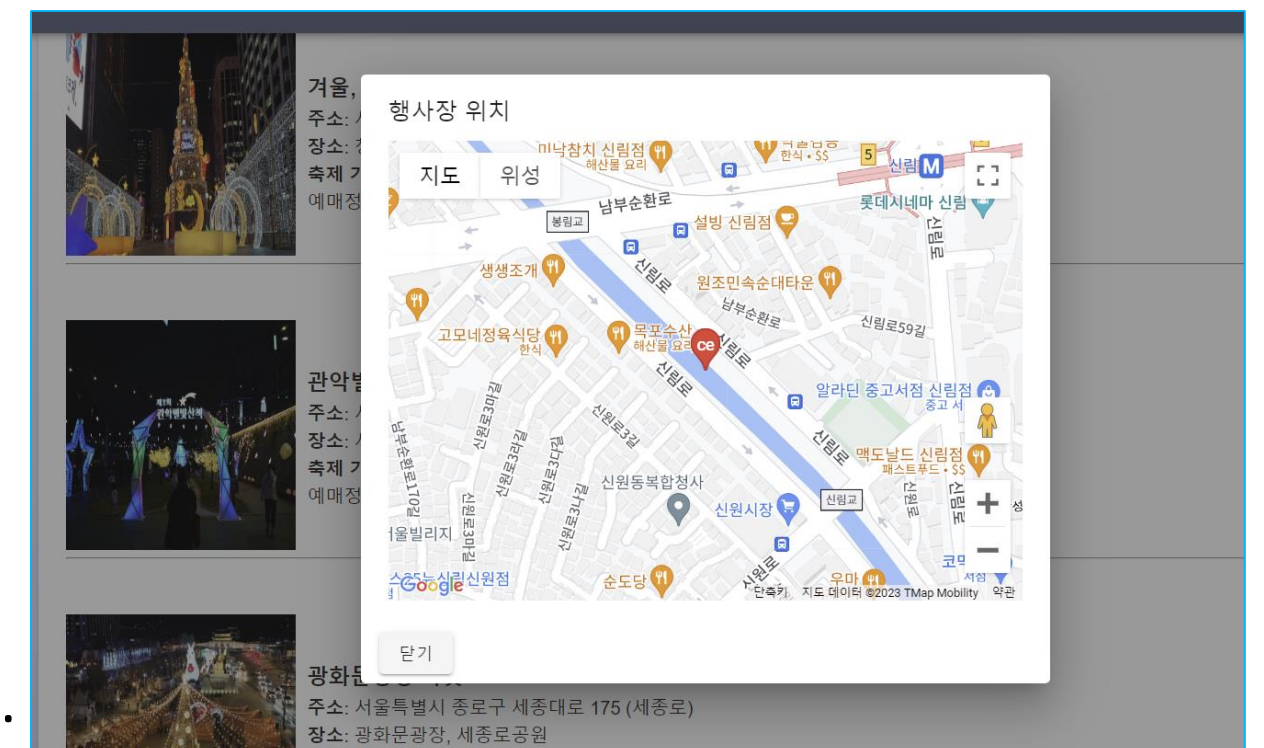


3. 작품 구현 - 지역으로 보기

지역을 선택시 해당 지역의
행사들이 전부 보여진다.



행사 그림을 누르면
행사장의 위치가 표시된다.



3. 작품 구현 - 달력

달력보기에서 지역을 대전으로 골랐을 때,
축제이름이라는 이벤트는 관리자가 임시로 넣은 대전의 행사데이터

11월 행사내용

최근행사

달력보기

지역으로보기

Q

대전

< 11월 2023 >

MONTH

SUN	MON	TUE	WED	THU	FRI	SAT
29	30	31	Nov 1	2	3	4
대전 행 축제			유성문화전시장			
5	6	7	8	9	10	11
유성문화전시장						
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	Dec 1	2
					유성문화 크리스마스 축제	

12월 행사내용

최근행사

달력보기

지역으로보기

Q

대전

< 12월 2023 >

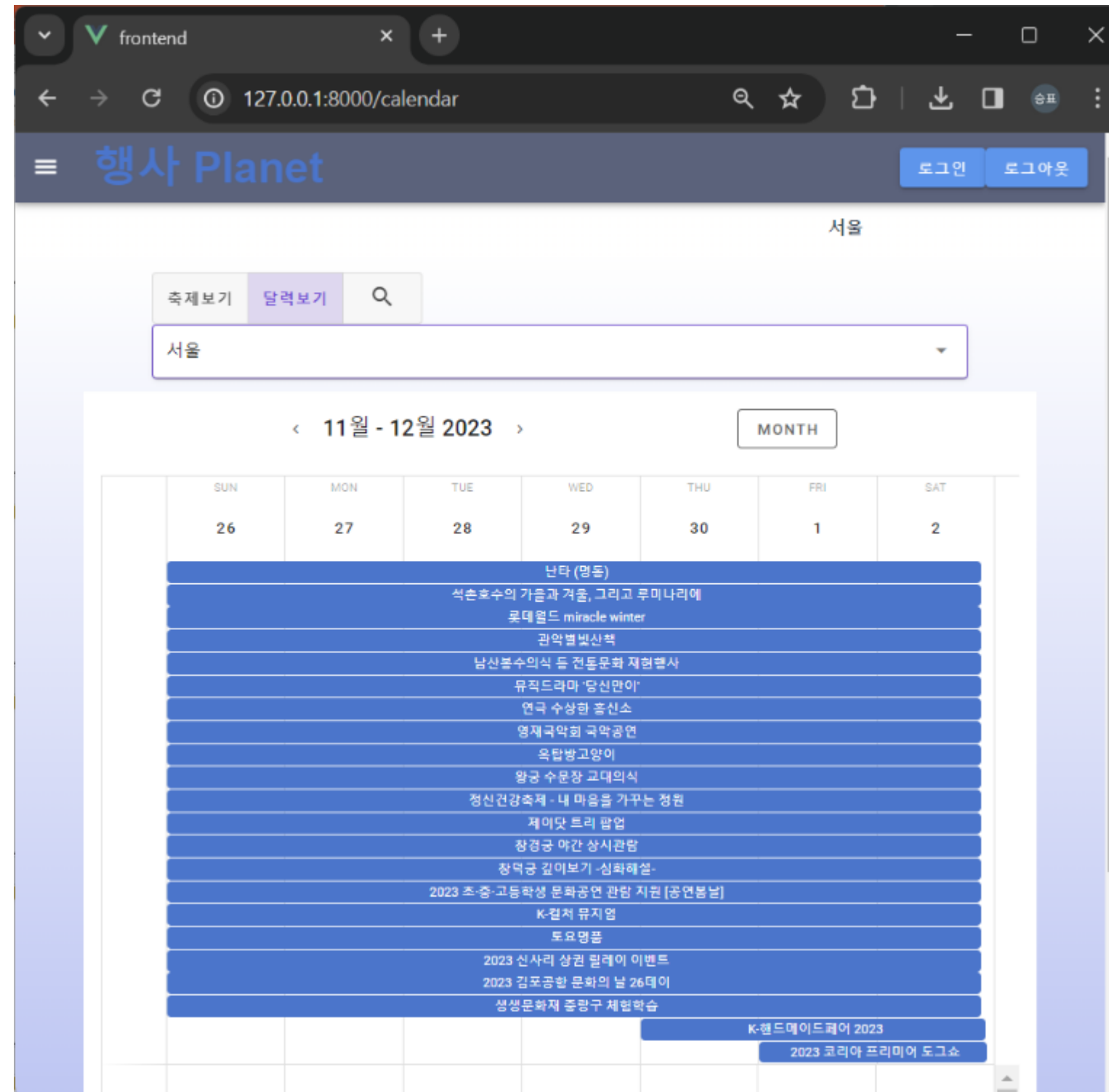
MONTH

SUN	MON	TUE	WED	THU	FRI	SAT
26	27	28	29	30	Dec 1	2
					유성온천 크리스마스 축제	
3	4	5	6	7	8	9
유성온천 크리스마스 축제		축제이름				
10	11	12	13	14	15	16
17	18	19	20	21	22	23

달력을 움직이면서 월별 데이터를 한눈에 확인이 가능하다.

3. 작품 구현 - 달력

행사들이 많아 more이라고 나올때 클릭시 그 날짜의 행사들을 한눈에 볼 수 있다.



3. 작품 구현 - 키워드에 누들이라고 타이핑 했을 때

최근행사


달력보기

지역으로보기


Q

키워드를 입력해주세요

키워드
누들



강릉누들축제
주소: 강원특별자치도 강릉시 경강로 2111 임당시장
장소: 월화거리 일원
축제시간: 20231027 ~ 20231029
예매정보 없음



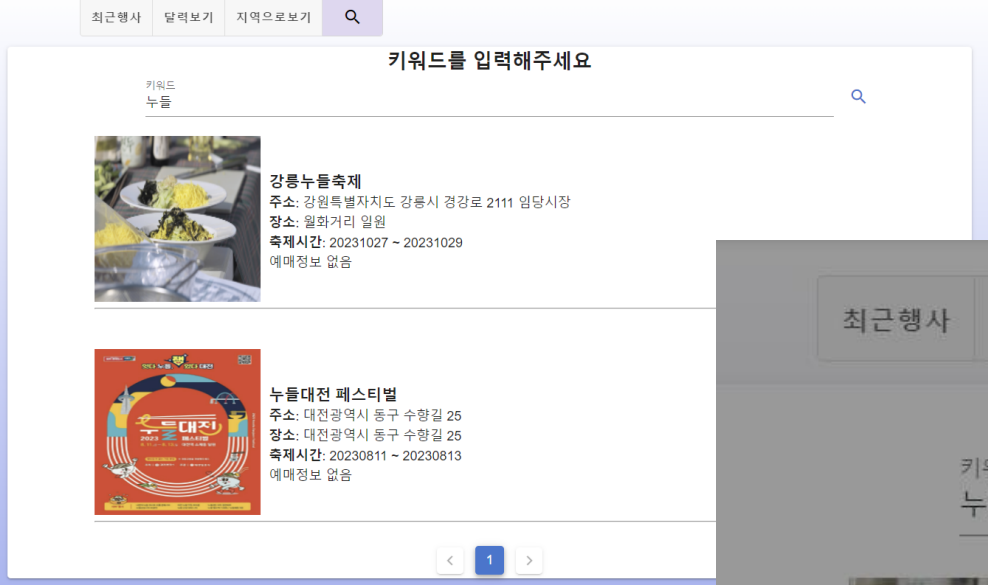
누들대전 페스티벌
주소: 대전광역시 동구 수향길 25
장소: 대전광역시 동구 수향길 25
축제시간: 20230811 ~ 20230813
예매정보 없음

<

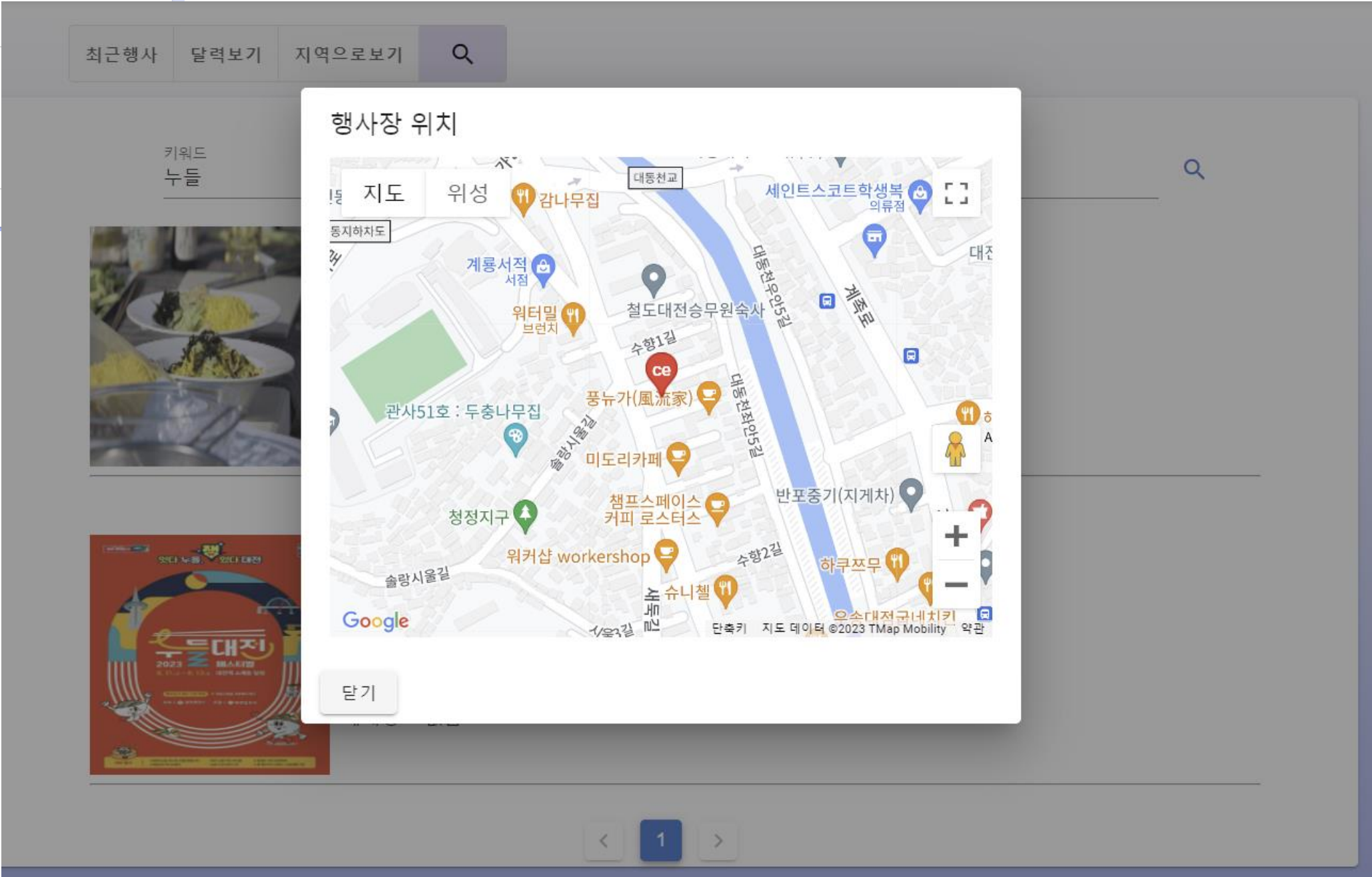
1

>

100



그림을 클릭했을 시 행사장 위치 표시



3. 작품 구현 - 관리자 페이지

관리자 화면에선 로그인을 해야 테이블과 달력에 내용이 표시된다.

로그인을 안한 상태

행사 Planet

로그인회원가입

최근행사달력보기지역으로보기

추가

번호	축제이름	장소	시작일	종료일	축제내용	등록일	관리
No data available							

Rows per page: 10 - < >

< 12월 2023 >

MONTH

SUN	MON	TUE	WED	THU	FRI	SAT
26	27	28	29	30	Dec 1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23

로그인을 한 상태
DB에 저장되어있는 계정의 이름으로
환영문구가 출력된다

행사 Planet

이승표님 안녕하세요로그아웃

최근행사달력보기지역으로보기

추가

번호	축제이름	장소	시작일	종료일	축제내용	등록일	관리
4	축제이름	3	2023-12-05T00:00:00.000Z	2023-12-06T00:00:00.000Z	축제내용	2023-12-05T12:02:58.165Z	

Rows per page: 10 1-1 of 1 < >

< 12월 2023 >

MONTH

SUN	MON	TUE	WED	THU	FRI	SAT
26	27	28	29	30	Dec 1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23

3. 작품 구현 - 회원가입

로그인 버튼 오른쪽에 회원가입 버튼을 누르면
입력칸이 뜨고 가입을 할 수 있다.



The screenshot shows the '행사 Planet' website interface. At the top right, there are two buttons: '로그인' (Login) and '회원가입' (Sign Up). Below these, a modal form is displayed with the following fields:

- 이름 (Name)
- 아이디 (ID)
- 패스워드 (Password)
- 거주지 (Residence)

At the bottom right of the modal, there are two buttons: '취소' (Cancel) and '확인' (Confirm).

3. 작품 구현 - 가입된 계정들 확인

관리자 페이지와 마찬가지로
로그인을 해야 테이블이 보인다.

≡ 행사 Planet

이승표님 안녕하세요 [로그아웃](#)

최근 행사

달력 보기

지역으로 보기

🔍

이름	아이디	패스워드	거주지	등록일	관리	
이승표	lsp	1234	대전	2023-12-06T09:08:51.027Z	수정	삭제
홍길동	asdf	1234	유성	2023-12-10T19:12:34.201Z	수정	삭제

Rows per page:

10

1-2 of 2

<

>

3. 작품 구현 - API

<u>메소드</u>	경로	설명
GET	/events	이벤트 검색
GET	/events/ <u>eventNum</u>	특정 이벤트 검색
POST	/events	새로운 이벤트 추가
PUT	/events/ <u>eventNum</u>	특정 이벤트 수정
DELETE	/events/ <u>eventNum</u>	특정 이벤트 삭제
GET	/ <u>eventCrawling</u>	최근 이벤트를 <u>크롤링</u> 해와서 보여줌
GET	/users	존재하는 관리자 계정을 보여줌
DELETE	/users/:id	계정 삭제
POST	/users	계정 추가
PUT	/users/:id	계정 수정

3. 작품 구현 - Backend 간단한 코드 설명

라우터

```
app.get('/events', authenticateUser, events.findAll); //관리자페이지에서 볼 수 있음 이벤트 검색
app.get('/userevents', events.findAll); // 사용자가 볼 수 있음
app.get('/events/:eventNum', events.findOne); //특정 이벤트 검색
app.post('/events',authenticateUser, events.create); //새로운 이벤트 추가
app.put('/events/:eventNum',authenticateUser, events.update); //특정 이벤트 업데이트
app.delete('/events/:eventNum',authenticateUser, events.delete); //특정 이벤트 삭제
```

```
app.post('/users', users.create); //계정 추가
app.get('/users',authenticateUser, users.findAll); //모든 계정 조회
app.get('/users/:Id', users.findOne); //특정 계정 조회
app.put('/users/:Id', users.update); //특정 계정 수정
app.delete('/users/:Id', users.delete); //특정 계정 삭제
```

```
app.get('/eventCrawling', crawling.eventCrawling); //크롤링한 데이터 res.send
```

로그인을 해야 사용할 수 있는 기능들은
라우터에 authenticateUser 설정

크롤링 라우터를 설정해서 경로에 접근 시
크롤링 데이터를 전송

```
exports.eventCrawling = async (req, res) => {
  try {
    let totalPages = 10;
    let events = [];

    for (let page = 1; page <= totalPages; page++) {
      const url = `https://www.mcst.go.kr/kor/s_culture/festival/festivalList.jsp?pSeq=&pRo=&pCurrentPa`;

      const response = await axios.get(url);
      const $ = cheerio.load(response.data);

      const $bodyList = $('ul.mediaWrap.color01').children('li');

      $bodyList.each(function (i, elem) {
        const imgSrc = $(this).find('img').attr('src');
        const fullImgSrc = `https://www.mcst.go.kr${imgSrc}`;

        events.push({
          title: $(this).find('div.text > p').text(),
          location: $(this).find('ul.detail_info li:nth-child(1)').text(),
          period: $(this).find('ul.detail_info li:nth-child(2)').text(),
          picture: fullImgSrc
        });
      });
    }

    const data = events.filter(event => event.title);

    if (data.length === 0) {
      return res.json({ error: '행사 데이터 크롤링 실패 혹은 데이터가 없습니다.' });
    }

    console.log('이벤트크롤링 데이터보냄', data);
    res.json({ events: data });
  } catch (error) {
```

Backend의 크롤링.js

3. 작품 구현 - Backend 간단한 코드 설명

소켓

```
// 새로운 이벤트 만들기
exports.create = (req, res) => {
  $socket.emit('newEvent', {          //새로운 데이터가 업로드되면 소켓을 이용하여 클라이언트에 전달
    eventNum: req.body.eventNum,
    eventName: req.body.eventName,
    eventLocation: req.body.eventLocation,
    eventStartDay: req.body.eventStartDay,
    eventEndDay: req.body.eventEndDay,
    eventContent: req.body.eventContent
  });
  const contact = new eventData({
    eventNum: req.body.eventNum,
    eventName: req.body.eventName,
    eventLocation: req.body.eventLocation,
    eventStartDay: req.body.eventStartDay,
    eventEndDay: req.body.eventEndDay,
    eventContent: req.body.eventContent
  });
};
```

DB에 새로운 이벤트를 추가할 시
실시간으로 달력에 이벤트를 추가하기
위해 소켓을 사용해 emit(전송)

```
//특정 이벤트 업데이트
exports.update = (req, res) => {
  eventData.findOneAndUpdate({ eventNum: req.params.eventNum },
    { eventNum: req.body.eventNum, eventName: req.body.eventName,
    { new: true }
  )
  .then(contact => {
    if (!contact) {
      return res.status(404).send({
        message: req.params.eventNum +
        "에 해당하는 축제가 없습니다."
      });
    }
    res.send(contact);
    $socket.emit('newEvent', {          //새로운 데이터가 업로드
      eventNum: req.body.eventNum,
      eventName: req.body.eventName,
      eventLocation: req.body.eventLocation,
      eventStartDay: req.body.eventStartDay,
      eventEndDay: req.body.eventEndDay,
      eventContent: req.body.eventContent
    });
  })
  .catch(err => {
```

DB에 있는 이벤트를 수정할 시 실시간으로
달력을 수정하기 위해 소켓을 사용해
emit(전송)

3. 작품 구현 - Backend 간단한 코드 설명

소켓

```
//특정 이벤트 삭제
exports.delete = (req, res) => {
  eventData.findOneAndDelete({ eventNum: req.params.eventNum })
    .then(contact => {
      if (!contact) {
        return res.status(404).send({ message: req.params.eventNum + "에 해당하는 축제가 없습니다." })
      }

      // 소켓을 사용하여 클라이언트에게 이벤트 삭제를 알림
      $socket.emit('deleteEvent', { eventNum: req.params.eventNum });

      res.send({ message: "정상적으로 " + req.params.eventNum + " 장치가 삭제되었습니다." })
    })
}
```

DB에 이벤트를 삭제할 시 실시간으로
달력에 이벤트를 지우기 위해 소켓을 emit

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트

```
▼ components
  ▼ Calendar.vue
  ▼ Calendar2.vue
  ▼ EventAPI.vue
  ▼ GoogleMap.vue
  ▼ KeywordSearch.vue
  ▼ recentEvent.vue
  ▼ ShowEventTable.vue
```

캘린더 : 로그인 했을 때 관리자 페이지에 보여질 달력

캘린더2 : 로그인 하지 않았을 때 볼 수 있는 달력

EventAPI : eventAPI에서 행사 데이터들을 가져오는 컴포넌트

GoogleMap : 이벤트 클릭시 지도에 표시하기 위한 컴포넌트

KeywordSearch : 키워드로 검색 시에 해당 데이터를 API에서 가져오는 컴포넌트

recentEvent : backend에 요청을 보내 크롤링한 데이터를 가져오는 컴포넌트

ShowEventTable : 로그인 했을 때 관리자 페이지에서 행사 테이블을 볼 수 있는 컴포넌트

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(Calendar)

```
init() {
  axios.get(this.urlinfo) //서버에 요청하기
    .then((res) => {
      for (let i = 0; i < res.data.length; i++) {
        const SDAY = new Date(res.data[i].eventStartDay);
        const EDAY = new Date(res.data[i].eventEndDay);
        const formatSDAY = SDAY.toISOString().split('T')[0];
        const formatEDAY = EDAY.toISOString().split('T')[0];

        this.DataBaseEvents.push({
          name: res.data[i].eventName,
          start: formatSDAY,
          end: formatEDAY,
        });
      }
    })
    .catch((err) => {
      console.log(err);
    });
  console.log(res.data.length); //성공시
}
```

<http://127.0.0.1:8000/events> 의 경로로 요청을 보내
DB에 있는 데이터 가져와서 달력에 출력

```
created() {
  this.init();
}

this.$socket.on('newEvent', (data) => {
  //if (this.$route.path === '/adminpage') {
  this.DataBaseEvents = [];
  this.init();
  console.log(data);

  // eventStartDay와 eventEndDay가 정의되어 있고 문자열인 경우에만 처리
  if (data.eventStartDay && data.eventEndDay) {
    const SDAY = new Date(data.eventStartDay);
    const EDAY = new Date(data.eventEndDay);
    const formatSDAY = SDAY.toISOString().split('T')[0];
    const formatEDAY = EDAY.toISOString().split('T')[0];
    console.log(this.DataBaseEvents);
  } else {
    console.error('eventStartDay 또는 eventEndDay가 적절한 형식이 아닙니다.~');
  }
  //}
});

// 클라이언트에서 'deleteEvent' 이벤트를 리스닝
this.$socket.on('deleteEvent', (data) => {
  this.events = [];
  this.DataBaseEvents = [];
  this.init();
});
```

소켓을 이용해 새로운 이벤트가 추가, 수정, 삭제
되었을 때 실시간으로 달력을 갱신

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(Calendar2)

```
watch: {  
  selectedItem() {  
    this.getSelectedItem(this.selectedItem);  
  },  
},
```

지역을 선택했을 때 API 데이터를 가져오고
또한 /userevents 경로로 요청을 보내 DB에 있는
데이터를 가져와 두개의 데이터를 합쳐 달력에
출력한다.

```
created() {  
  const socket = this.$socket;  
  this.$socket.on('newEvent', (data) => {  
    //if (this.$route.path === '/adminpage') {  
    console.log(data);  
    // eventStartDay와 eventEndDay가 정의되어 있고 문자열인 경우에만 처리  
    if (data.eventStartDay && data.eventEndDay) {  
      const SDAY = new Date(data.eventStartDay);  
      const EDAY = new Date(data.eventEndDay);  
      const formatSDAY = SDAY.toISOString().split('T')[0];  
      const formatEDAY = EDAY.toISOString().split('T')[0];  
  
      this.DataBaseEvents.push({  
        name: data.eventName,  
        start: formatSDAY,  
        end: formatEDAY,  
      });  
      console.log(this.DataBaseEvents)  
    } else {  
      console.error('eventStartDay 또는 eventEndDay가 적절한 형식이 아닙니다.~');  
    }  
  });  
  
  // 클라이언트에서 'deleteEvent' 이벤트를 리스닝  
  this.$socket.on('deleteEvent', (data) => {  
    this.events = [];  
    this.DataBaseEvents = [];  
  });  
},
```

소켓을 이용해 새로운 이벤트가 추가, 수정, 삭제
되었을 때 실시간으로 달력을 갱신

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(EventAPI)

```
// 선택된 축제 장소가 구글맵에 표시되게 하기 위해 이벤트버스로 주소 전송
handleImageClick(index) {

  // 구글맵에 주소를 보내기위한 이벤트버스
  this.$bus.$emit('sendEventLocation', index.addr1);

},
```

API를 통해 주소, 장소, 축제기간, 예매정보 등을 가져와 보여준다.

```
mounted() {

  this.$bus.$on('sendEventLocation', this.getLocation);

},
```

또한 이미지 클릭시 이벤트 버스로 구글맵에 주소를 보내 위치를 띄운다.

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(GoogleMap)

```
mounted() {  
  this.$bus.$on('sendEventLocation', this.getLocation);  
},
```

이벤트 버스로 행사장의 주소를 받아온다

```
this.geocoder.geocode({ address: this.city }, (results, status) => {  
  if (status === 'OK' && results[0].geometry) {  
    const location = results[0].geometry.location;  
    this.map.setCenter(location);  
  } else {  
    console.error('지오코딩 실패:', status);  
  }  
});
```

받아온 주소를 지오코더를 사용해 위도, 경도로 바꾼 후 그 좌표를 센터값으로 설정한다.

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(KeywordSearch)

```
<v-row>
  <v-col offset="1" cols="10">
    <v-text-field v-model="keyword" label="키워드" hide-details></v-text-field>
  </v-col>

  <v-col cols="1" class="d-flex align-center" style="padding: 0; ">
    <v-btn @click="performSearch" color="primary" icon><v-icon>mdi-magnify</v-icon></v-btn>
  </v-col>
</v-row>
```

검색칸에 키워드를 입력 후 검색버튼을 누르면 해당 키워드를 받아온다.

```
const encodedKeyword = encodeURIComponent(this.keyword);
const serviceKey = 'bmDxI%2F3KJw7bCGRhEHG%2F3wiUzRAy5FJ0%2BUJwAXV%2BQpXSus4MwYaDw8iBnhKtrsLUtyD3dxsFZ%2FVAiSTiSyFg%3D%3D';

const response1 = await this.axios.get(
  `http://apis.data.go.kr/B551011/KorService1/searchKeyword1?MobileOS=WIN&MobileApp=WebTest&pageNo=${this.currentPage}&numOfRows=50&contentType=15&keyword=${encodedKeyword}&_type=
`);
```

받은 키워드를 URL형식으로 인코드 후 쿼리스트링 형식으로 집어넣어 API를 호출한다.

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(recentEvent)

```
data() {  
  return {  
    events: [], // 전체 이벤트 데이터  
    displayedEvents: [], // 현재 페이지에 표시할 이벤트 데이터  
    itemsPerPage: 5, // 페이지 당 표시할 이벤트 수  
    currentPage: 1, // 현재 페이지  
    urlinfo: 'http://127.0.0.1:8000/eventCrawling',  
  };  
},  
mounted() {  
  // 백엔드 API로 요청을 보냅니다.  
  axios.get(this.urlinfo, { withCredentials: true }) //서버에 요청하기  
    .then((res) => {  
      console.log(res.data); //성공시  
      this.events = res.data.events;  
      console.log(this.events);  
      this.updateDisplayedEvents();  
    })  
    .catch((err) => {  
      alert('에러 발생: ' + err); //에러 발생  
    });  
},
```

라우터에서 설정한 경로로 get 요청을 보내 크롤링 데이터를 받아온다.

3. 작품 구현 - frontend 간단한 코드 설명

컴포넌트(ShowEventTable)

```
created() {  
  
  const fetchData = async () => {  
    try {  
      // 비동기 요청, 데이터 로딩이 완료될 때까지 대기  
      const res = await axios.get(this.urlinfo, { withCredentials: true });  
  
      // 데이터 로딩 완료 후 작업 수행  
  
      if (res.data.length > 0) {  
        this.items = res.data;  
      }  
      console.log('데이터들어감', res);  
  
      console.log('들어감', this.items);  
    } catch (err) {  
      // 에러 처리  
      alert('에러 발생: ' + err);  
    }  
  };  
  
  // async 함수 호출  
  fetchData();  
},
```

관리자 페이지에서 보여질 행사 테이블

```
btnClick($event) {  
  this.dialog = false;  
  if ($event.target.innerHTML === "확인") {  
    if (this.dialogTitle === "추가") {  
      axios.post(this.urlinfo, {  
        eventNum: this.eventInfo.eventNum,  
        eventName: this.eventInfo.eventName,  
        eventLocation: this.eventInfo.eventLocation,  
        eventStartDay: this.eventInfo.eventStartDay,  
        eventEndDay: this.eventInfo.eventEndDay,  
        eventContent: this.eventInfo.eventContent,  
        register_date: this.eventInfo.register_date  
      }, { withCredentials: true })  
        .then(() => {  
          axios.get(this.urlinfo, { withCredentials: true }) //서버에 요청하기  
            .then((res) => {  
              //console.log(res.data); //성공시  
              if (this.isUserLogin) {  
                this.items = res.data;  
                alert("행사 데이터 추가 성공");  
              }  
              else {  
                alert("로그인이 필요합니다.");  
              }  
            })  
            .catch((err) => {  
              alert('에러 발생: ' + err); //에러 발생  
            });  
        })  
        .catch((err) => {  
          alert('에러 발생: ' + err); //에러 발생  
        });  
    }  
  }  
}
```

Vuex로 사용자가 로그인한 상태인지 확인하는 변수를 만들어 그 변수를 가지고 로그인되어있다면 테이블의 추가, 수정, 삭제가 가능하게 구현

3. 작품 구현 - VUEX(store의 index.js)

```
export default new Vuex.Store({
  // 공유할 데이터
  state: {
    username: '',
    isLogin: false,
  },
  getters: {
    isLogin(state) {
      return state.username !== '';
    },
    username: (state) => state.username,
  },
  // mutations은 state의 값을 바꿀수 있는 유일한 방법이다.
  // mutations의 첫번째 인자는 state고, 두번째 인자는 호출할때 넘기는 값을 의미한다.
  // username을 받아서 state가 호출될때 username을 넘기겠다 뜻
  mutations: {
    setUsername(state, username) {
      state.username = username;
    },
    setLoginState(state, payload) {
      state.isLogin = payload;
    },
  },
});
```

Username : 로그인 했을 때 사용자의 계정에 맞는 이름이 들어갈 변수

isLogin : 로그인을 한 상태인지 bool형태로 체크하는 변수

```
console.log(req.isAuthenticated());
req.session.save((err) => {
  if (err) {
    console.error(err)
    next(err)
  } else {
    res.send({ code: 1, name: user.name });
  }
});
```

Backend의 passport-session.js에서 로그인 처리시 로그인정보가 일치하면 코드로 1과 username을 전송

```
}).then((res) => {
  if (res.status === 200) {
    // 로그인 성공시 처리해줘야할 부분
    if (res.data.code === 0) { //로그인 실패
      alert('로그인 실패');
      //this.$router.push('/login');
    } else if (res.data.code === 2) { //접근제한 페이지
      alert('접근제한 페이지');
      //this.$router.push('/login');
    } else if (res.data.code === 1) {
      alert('로그인 성공');

      //this.$bus.$emit('deviceSelected', res.data.code);
      this.$store.commit('setUsername', res.data.name); // 이름 보내준거
      this.$store.commit('auth/setLoginState', true); // 로그인 상태 보내준거
    }
  }
});
```

Views의 Login.vue에서 로그인 성공 시 username과 isLogin에 값을 전달.

3. 작품 구현 - APP.VUE

```
<v-spacer></v-spacer><!--오른쪽으로 붙이기 위해 씬-->
<template v-if="isUserLogin">
  <span>{{ getUsername }}님 안녕하세요</span>
  <v-btn class="blue" style="margin-left: 8px;" @click="logout">로그아웃</v-btn>
</template>

<template v-if="!isUserLogin">
  <router-link to="/login"><v-btn class="blue" style="margin-right: 8px;">로그인</v-btn></router-link>
  <router-link to="/subscribe"><v-btn class="blue" style="margin-right: 8px;">회원가입</v-btn></router-link>
</template>
```

VUEX를 사용해 로그인한 상태이면 유저이름과 로그아웃 버튼이 보이게 하고,
로그인하지 않은 상태라면 로그인과 회원가입 버튼이 보이게 한다.