SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Vue.js

<알림>

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로! Enabling People

8장. Vue.js 프로젝트

챕터의 포인트

- 주문 관리 웹앱 제작

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

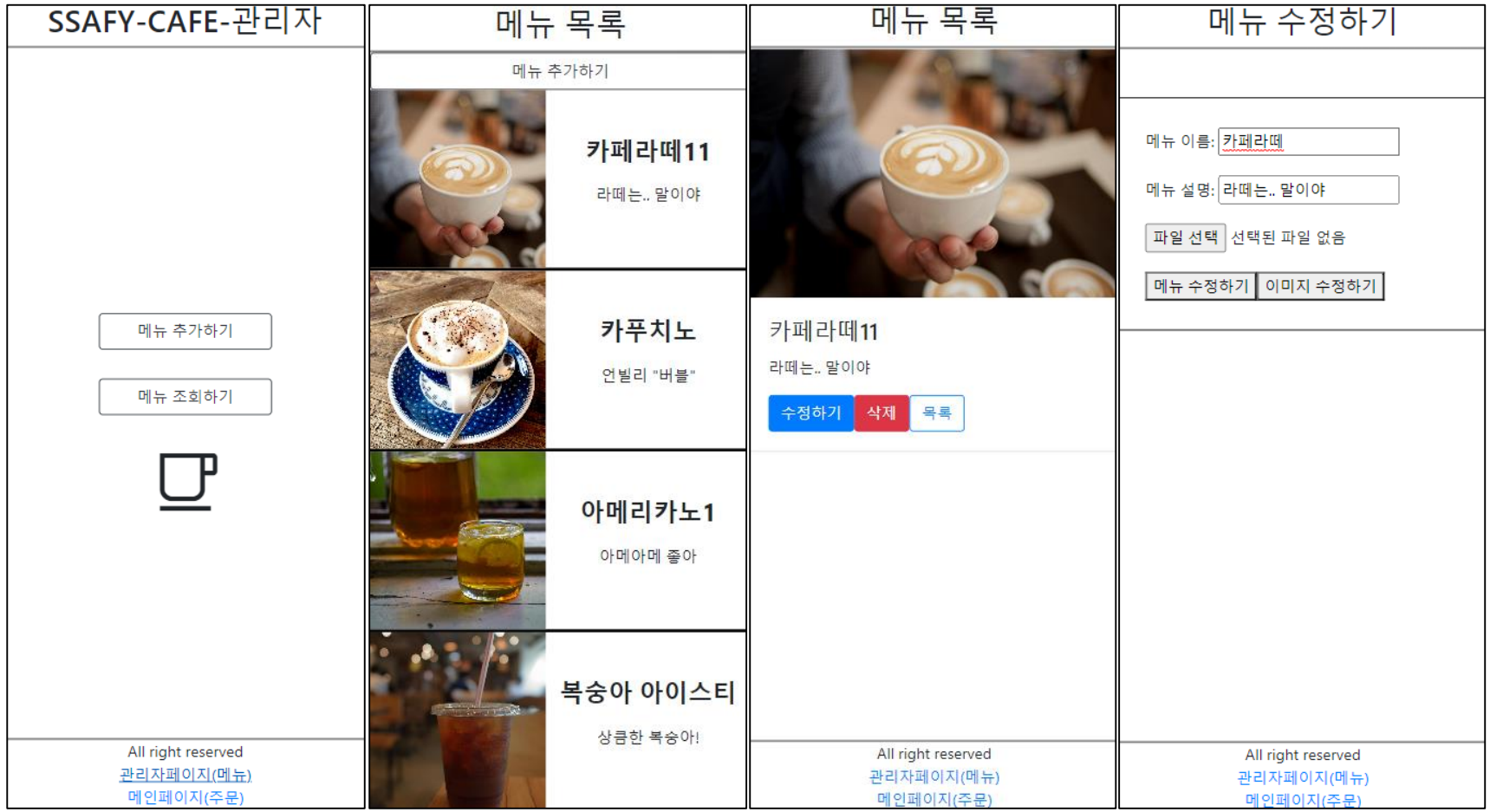
주문 관리 웹앱 제작

프로젝트

- 백엔드
  - Node.js 시간에 만든 백엔드 API
- 프론트엔드
  - Vue.js 활용

Node.js 주간에 만든 주문관리 API 를 활용

- 카페 주문관리 웹앱 제작



API 명세서 - menus

- menus
  - GET /api/menus
    - 메뉴 전체 조회
    - GET /api/menus/:id
      - 메뉴 일부 조회
  - POST /api/menus
    - 메뉴 등록
  - PATCH /api/menus/:id
    - 메뉴 수정
  - POST /api/menus/:id/image
    - 메뉴 이미지 수정
  - DELETE /api/menus/:id
    - 메뉴 삭제

API 명세서 - orders

- orders
  - GET /api/orders
    - 주문 전체 조회
  - GET /api/orders/:id
    - 주문 내역 조회
  - POST /api/orders
    - 주문 하기
  - PATCH /api/orders/:id
    - 주문 수정하기
  - DELETE /api/orders/:id
    - 주문 삭제하기

GET /api/menus

- 모든 메뉴 목록을 가져온다.
- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - [{

```
"id": 1,
"name": "아이스 아메리카노",
"description": "여름엔 아아가 진리",
“image_src": "/public/ice-americano.jpg"
```

  }, …]
- 실패 시
  - { success : false, message: ”전체 메뉴 목록 조회에 실패하였습니다.” } 리턴

GET /api/menus/:id

- 메뉴 목록중 id에 해당하는 한가지를 가져온다.
- 성공 시, 메뉴 객체 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - {

```
"id": 1,
"name": "아이스 아메리카노",
"description": "여름엔 아아가 진리",
“image_src": "/public/ice-americano.jpg"
```

  }
- 실패 시
  - { success : false, message: ” 메뉴 조회에 실패하였습니다.” } 리턴

POST /api/menus

- 메뉴를 추가한다.
- 클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.
  - {
    "name": "아이스 아메리카노",
    "description": "여름엔 아아가 진리",
    “file”: “파일 첨부”

    }

- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - {
    success:true,
    message: “메뉴 등록에 성공하셨습니다.”
    }
- 실패 시
  - {success : false, message: ”메뉴 등록에 실패하였습니다.” } 리턴

PATCH /api/menus/:id

- id를 받아와서 이미지를 제외한 메뉴를 수정한다.
  - 이미지는 POST 요청으로 보내야 한다.

- 클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.
  - {
    "name": "아이스 아메리카노",
    "description": "여름엔 아아가 진리",
    }

- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - {
    success:true,
    message: “메뉴 정보 수정에 성공하셨습니다.”
    }
- 실패 시
  - {success : false, message: ”메뉴 정보 수정에 실패하였습니다.” } 리턴

POST /api/menus/:id/image

- id를 받아와서 이미지만 수정한다
  - 이미지는 POST 요청으로 보내야 하므로 POST로 처리한다.

- 클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.
  - {
    “file": “file 정보",
    }

- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - {
    success:true,
    message: “메뉴 이미지 수정에 성공하셨습니다.”
    }
- 실패 시
  - {success : false, message: ”메뉴 이미지 수정에 실패하였습니다.” } 리턴

DELETE /api/menus/:id

- id에 해당되는 menus를 삭제한다.
- 요청시 보내는 JSON 데이터는 존재하지 않으며 :id로 삭제만 수행
- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - {
    success: true,
    message: “메뉴 삭제에 성공하셨습니다.”
    }
- 실패 시
  - {success : false, message: ”메뉴 삭제에 실패하였습니다.” } 리턴

GET /api/orders

- 모든 주문 목록을 가져온다.
- 성공 시, 주문 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - [{
    "id": 1,
    "name": "아이스 아메리카노",
    "quantity": 1,
    “request_detail": "뜨겁게 만들어주세요"
    }, …]

  힌트: JOIN을 사용해 menus_id 에 해당하는 name을 가져와야한다.
- 실패 시
  - { success: false, message: ”전체 주문 목록 조회에 실패하였습니다.” } 리턴

POST /api/orders

- 새로운 주문을 추가한다.
- 클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.
  - {
    “menus_id”: 1
    "quantity": 1,
    “request_detail": “뜨겁게 주세요"
    }
- 성공 시, 다음 형태의 JSON 을 리턴한다.
  - {
    success : true,
    id: 1 // 주문 번호이자 post시 생성된 orders의 id
    }
- 실패 시
  - {success : false, message: ”주문에 실패하였습니다.” } 리턴

GET /api/orders/:id

- 주문 내역을 가져온다.
- 성공 시, 주문 내역 객체를 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
  - {

```
"id": 1,
"name": "아이스 아메리카노",
"quantity": 1,
“request_detail": "뜨겁게 만들어주세요"
}
```

  힌트: JOIN 사용해 menus_id 에 해당하는 name 가져와야한다.
- 실패 시
  - { success: false, message: ”주문 내역 조회에 실패하였습니다.” } 리턴

PATCH /api/orders/:id

- 기존 주문 내역을 수정한다.
- 클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.
  - {
    “menus_id”: 1
    "quantity": 1,
    “request_detail": “뜨겁게 주세요"
    }
- 성공 시, 다음 형태의 JSON 을 리턴한다.
  - {
    success : true,
    message: “주문 수정에 성공하셨습니다.”
    }
- 실패 시
  - {success : false, message: ”주문 수정에 실패하였습니다.” } 리턴

DELETE /api/orders/:id

- 기존 주문 내역을 삭제한다.
- 성공 시, 다음 형태의 JSON 을 리턴한다.
  - {

    success : true,

    message: “주문 삭제에 성공하셨습니다.”

    }
- 실패 시
  - {success : false, message: ”주문 삭제에 실패하였습니다.” } 리턴

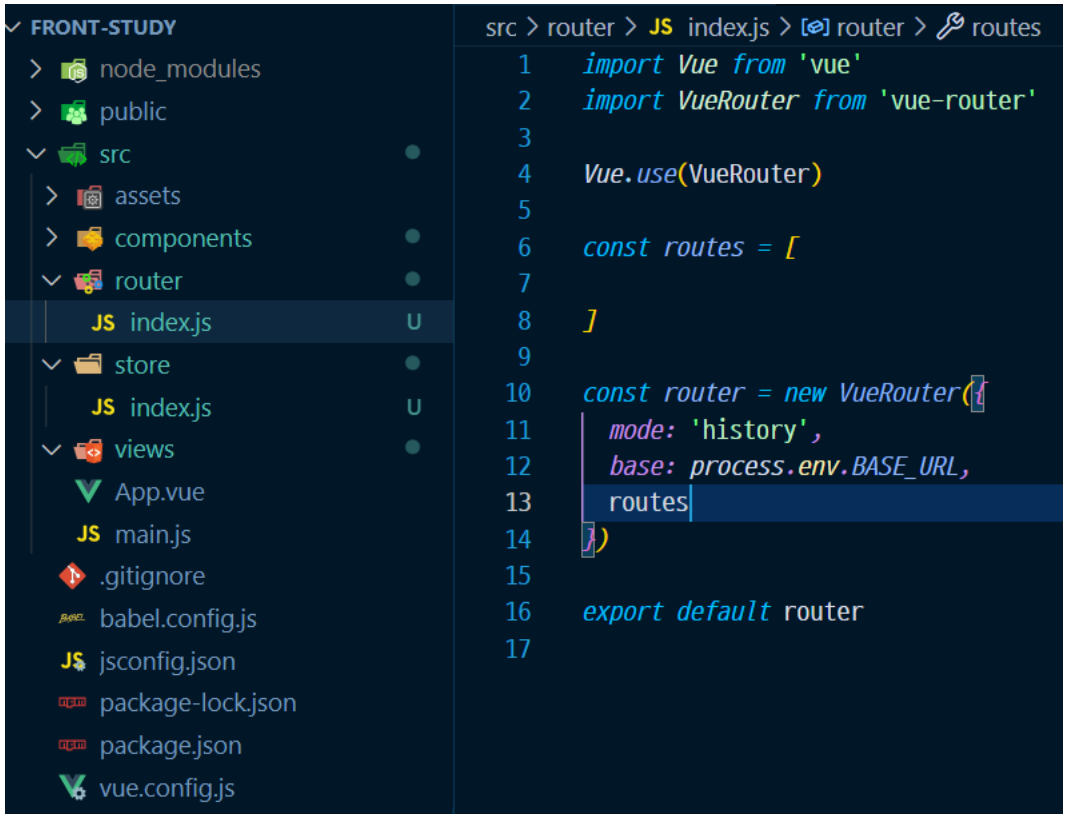
필요 라이브러리

- 기존 vue-cli
  - router, vuex 포함
- npm i axios bootstrap-vue bootstrap@4.6.1
  - boostrap을 버전4로 설치하는 이유
    - Vue-boostrap이 현재 boostrap 버전4과 호환

```
npm i axios bootstrap-vue bootstrap@4.6.1
```

기초 셋팅

- 지울 부분
  - /components/HelloWorld.vue 삭제
  - /views/AboutView.vue, /views/HomeView.vue 삭제
  - router/index.js
    - import HomeView from “../views/HomeView.vue” 삭제
    - rotues = [ ]
      - 빈 배열의 형태로 만들기



Vue-bootstrap 셋팅

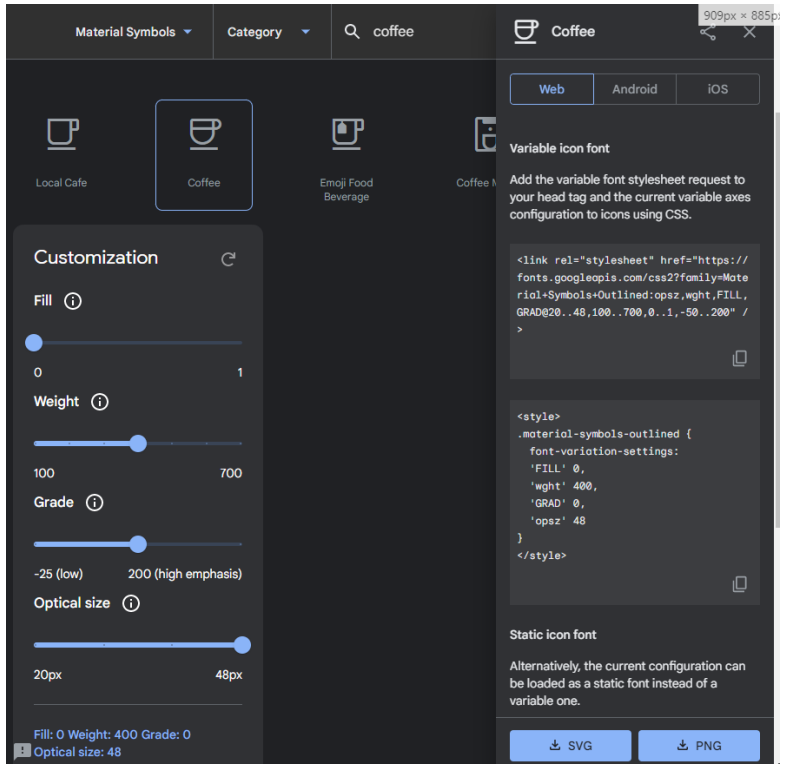
- src/main.js 수정
  - boostrap을 위한 셋팅 추가

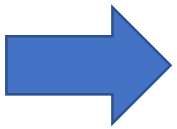
src > JS main.js > ...

```
1   import Vue from 'vue'
2   import App from './App.vue'
3   import router from './router'
4   import store from './store'
5
6   Vue.config.productionTip = false
7
8   import { BootstrapVue } from "bootstrap-vue";
9   import "bootstrap/dist/css/bootstrap.css";
10  import "bootstrap-vue/dist/bootstrap-vue.css";
11  Vue.use(BootstrapVue);
12
13  new Vue({
14    router,
15    store,
16    render: h => h(App)
17  }).$mount('#app')
```

public/index.html 수정

- Google Fonts 접속
  - ICON 클릭
  - Coffee 검색 후 나오는 <link> 부분 복사
  - public/index.html 에 해당 link 삽입





```
public > index.html > ...
1  <!DOCTYPE html>
2  <html lang="">
3    <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width,initial-scale=1.0">
7      <link rel="icon" href="<%= BASE_URL %>favicon.ico">
8      <title><%= htmlWebpackPlugin.options.title %></title>
9      <link
10     rel="stylesheet"
11     href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@48,400,0,0"
12   />
```

App.vue 수정하기

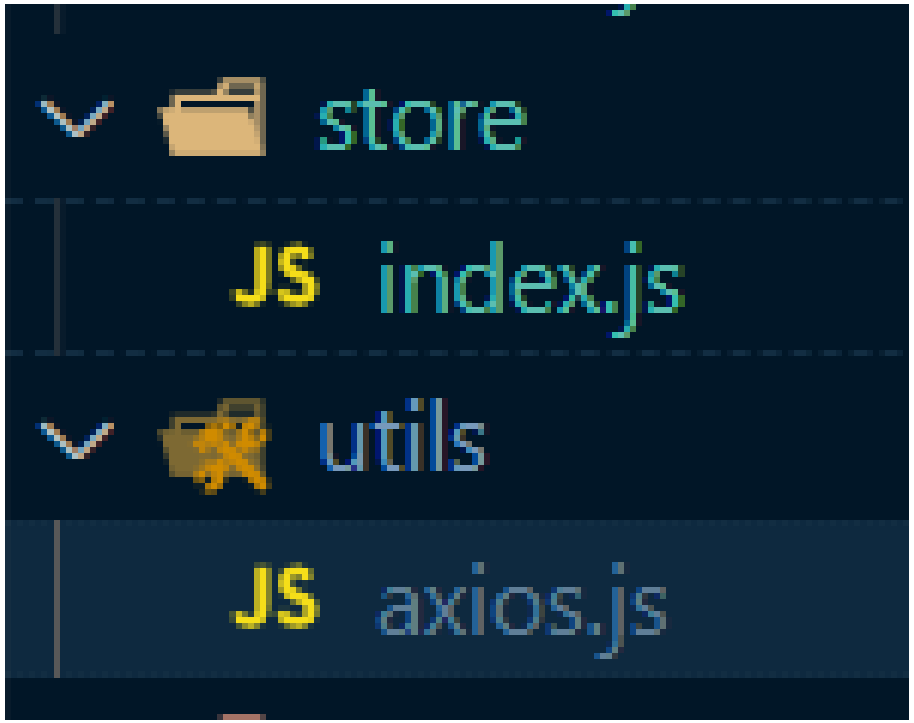
- router-view를 제외한 모든 내용 제거
  - script 추가
  - style 제거

src > App.vue > Vetur > {} "App.vue"

```vue
<template>
  <div id="app">

    <router-view/>

  </div>
</template>
<script>
export default {
  data(){
    return {

    }
  }
};
</script>
<style>

</style>
```

utils 폴더 생성

- API 명세표를 기반으로 axios 요청 생성



axios.js 수정

- DOMAIN
  - 미리 띄워둔 서버의 도메인 주소
    - IP도 가능하고 도메인이름도 가능
  - axios.create
    - 해당 request 요청을 보낼때
      정의된 내용을 바탕으로 요청이 보내진다.
    - ex) request.get("/hello-world")
      - http://ssafy.o-r.kr:8080/api/hello-world
      - baseURL + /api가 기본적으로 요청이 붙는다

```
import axios from "axios";

const DOMAIN = "http://ssafy.o-r.kr:8080";

const request = axios.create({
    baseURL: `${DOMAIN}/api`,
});
export const api = {}
```

axios.js 수정 – menus

- 각 api에 menus 객체를 생성
  - findAll
    - 전체 메뉴
  - findOne
    - 한가지 메뉴
  - create
    - 메뉴 등록
  - update
    - 메뉴 수정
  - updateImage
    - 메뉴 이미지 수정

```
export const api = {
    menus: {
        // http://ssafy.o-r.kr:8080/api/menus 요청과 같다.
        findAll: () => request.get("/menus"),
        findOne: (id) => request.get(`/menus/${id}`),

        create: (name, description, file) => {
            const formData = new FormData();
            formData.append('name', name);
            formData.append('description', description);
            formData.append('file', file);
            return request.post(`/menus`, formData, {
                headers: {
                    "Content-Type": "multipart/form-data"
                }
            })
        },
        // 메뉴 수정
        update: (id, name, description) => request.patch(`/menus/${id}`, {
            name: name,
            description
        }),
        // 메뉴 이미지 수정

        updateImage: (id, file) => {
            const formData = new FormData();
            formData.append('file', file)
            return request.post(`/menus/${id}/image`, formData, {
                headers: {
                    "Content-Type": "multipart/form-data"
                }
            })
        },
        // 메뉴 삭제
        delete: (id) => request.delete(`/menus/${id}`)
    },
```

파일 첨부를 포함한 요청시 주의 사항

- 파일 첨부시 기존 axios.post('/주소', { key:value }) 형식과 다르게 지켜줄 부분들이 있다.
- 파일첨부는 POST 요청만 가능!
- new FormData() 활용
  - 파일첨부시 FormData 생성및 formData.append('키','값 ')형식으로 넣어줘야한다.
  - headers 옵션
    - Content-Type: multipart/form-data 활용

```
create: (name, description, file) => {
    const formData = new FormData();
    formData.append('name', name);
    formData.append('description', description);
    formData.append('file', file);
    return request.post(`/menus`, formData, {
        headers: {
            "Content-Type": "multipart/form-data"
        }
    })
},
```

axios.js 수정 – orders

- 각 api에 orders 객체를 생성
  - findAll
    - 전체 주문내역 조회
  - findOne
    - 한가지 주문내역 조회
  - create
    - 주문 등록
  - update
    - 주문 수정

```
orders: {
    // 주문목록 조회
    findAll: () => request.get("/orders"),
    // 주문 조회
    findOne: (id) => request.get(`/orders/${id}`),
    // 주문하기
    create: (menus_id, quantity, request_detail) => request.post(`/orders`, {
        menus_id,
        quantity,
        request_detail
    }),
    // 주문 내역 수정하기
    update: (id, menus_id, quantity, request_detail) => request.patch(`/orders/${id}`, {
        menus_id: menus_id,
        quantity,
        request_detail
    }),
    // 주문 내역 삭제하기
    delete: (id) => request.delete(`/orders/${id}`)
}
```

라우터 설계

- 메뉴
  - /admin/menus
    - 메뉴 전체 조회
  - /admin/menus/register
    - 메뉴 등록
  - /admin/menus/register/:id
    - 메뉴 수정
    - id에 해당되는 메뉴를 수정
  - /admin/menus/:id
    - 메뉴 상세 조회
    - id에 해당되는 메뉴를 조회

라우터 설계

- 주문
  - /orders
    - 주문 전체 조회
  - /orders/register
    - 주문 등록
  - /orders/register/:id
    - 주문 수정
    - id에 해당되는 주문을 수정
  - /orders/:id
    - 주문 상세 조회
    - id에 해당되는 order를 조회

라우터에 필요한 views 작성하기

views 폴더 생성

- admin-menu 폴더
  - Main.vue
    - admin 메인 화면
  - List.vue
    - 메뉴 리스트
  - Register.vue
    - 메뉴 등록/수정
  - Detail.vue
    - 메뉴 상세

```
<template>
  <div>
    admin 메인 화면
  </div>
</template>

<script>


export default {
  data(){
    return {

    }
  }
};
</script>

<style scoped>

</style>
```

라우터에 필요한 views 작성하기

views 폴더 생성

- order 폴더
  - Main.vue
    - 메인 화면
  - List.vue
    - 주문 리스트
  - Register.vue
    - 주문 등록/수정
  - Detail.vue
    - 주문 상세


```vue
<template>
  <div>
    메인 화면
  </div>
</template>

<script>


export default {
  data(){
    return {

    }
  }
};
</script>

<style scoped>

</style>
```

라우터와 views 매핑작업

router/index.js 수정

- 각 정의한 파일들 import로 가져오기

```
import Vue from 'vue'
import VueRouter from 'vue-router'

import OrderMain from '../views/order/Main.vue'
import OrderList from "../views/order/List.vue"
import OrderRegister from "../views/order/Register.vue"
import OrderDetail from "../views/order/Register.vue"

import AdminMenuMain from "../views/admin-menu/Main.vue"
import AdminMenuList from "../views/admin-menu/List.vue"
import AdminMenuRegister from "../views/admin-menu/Register.vue"
import AdminMenuDetail from "../views/admin-menu/Detail.vue"
```

라우터와 views 매핑작업

router/index.js 수정

- 앞에서 정의한
  라우터에 따라 라우터 작성

```
const routes = [
  {
    path: '/',
    name: 'home',
    component: OrderMain
  },
  {
    path: '/orders',
    name: 'orders',
    component: OrderList
  },

  {
    path: '/orders/register',
    name: 'order-register',
    component: OrderRegister
  },
  {
    path: '/orders/register/:id',
    name: 'order-update',
    component: OrderRegister
  },

  {
    path: '/orders/:id',
    name: 'order-detail',
    component: OrderRegister
  },
```

```
// admin
{
  path:'/admin/home',
  name: 'menus-home',
  component: AdminMenuMain
},

{
  path:'/admin/menus',
  name: 'menus',
  component: AdminMenuList
},

{
  path:'/admin/menus/register',
  name: 'menus-register',
  component: AdminMenuRegister
},
{
  path: '/admin/menus/:id',
  name: 'menus-detail',
  component: AdminMenuDetail
},
{
  path:'/admin/menus/register/:id',
  name: 'menus-update',
  component:AdminMenuRegister
},

]
```

라우터 접근확인

- 각 localhost에 접근해서 접속이 잘되는지 체크
  - 다른 localhost인 경우 앞의 포트만 바꿔서 체크하기
  - http://localhost:8080/orders
  - http://localhost:8080/orders/register
  - http://localhost:8080/orders/register/3
  - http://localhost:8080/orders/3
  - http://localhost:8080/admin/menus
  - http://localhost:8080/admin/menus/register
  - http://localhost:8080/admin/menus/register/3
  - http://localhost:8080/admin/menus/3

App.vue 수정하기

template 수정

- header 부분
- footer 부분 추가

```vue
<template>

  <div id="app">
    <header>
      <h2>SSAFY-CAFE</h2>
    </header>

    <router-view />

    <footer>
      <div>All right reserved</div>
      <router-link to="/admin/home"> 관리자페이지(메뉴)</router-link>
      <br/>
      <router-link to="/"> 메인페이지(주문)</router-link>
    </footer>
  </div>

</template>
```

App.vue 수정하기

style

- css 수정

```
header,
footer {
  border-top: 2px solid grey;
  border-bottom: 2px solid grey;
}

#app {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

#app > header {
  /* flex:1; */
  text-align: center;
}

#app > div {
  /* header랑 footer만 제외하고 늘어난다 */
  flex: 1;
}

#app > footer {
  text-align: center;
  /* flex: 1; */
}
```

store 수정하기

store/index.js

- state를 변경해주는 mutations 작성하기
- 해당 title을 활용해 App.vue에 있는 SSAFY-CAFE 부분을 변경

src > store > JS index.js > ...

```
1   import Vue from "vue";
2   import Vuex from "vuex";
3
4   Vue.use(Vuex);
5
6   export default new Vuex.Store({
7     state: {
8       title:"SSAFY-CAFE"
9     },
10    mutations: {
11
12      SET_TITLE(state, data) {
13        state.title = data;
14      }
15    },
16  });
```

기존의 text 형식의 header 변경

- $store.state.title
  - vuex의 store에 저장되어있는 state의 title을 가져온다
- this.$store.commit("SET_TITTLE", "SSAFY-CAFE")
  - state를 변경시키기 위한 mutation SET_TITLE에 인자로 SSAFY-CAFE를 넣어준다.

```
<div id="app">

  <header>
    <h2>{{$store.state.title}}</h2>
  </header>

  <router-view/>
```

```
export default {
  data(){
    return {

    }
  },
  created(){
    this.$store.commit("SET_TITLE", "SSAFY-CAFE")
  }
};
</script>
```

views/order/Main.vue 수정

Main.vue 수정

- template
  - vue-bootstrap을 활용해 컴포넌트 삽입
- script
  - TITLE을 SSAFY-CAFE로 지정

```vue
<template>
  <div>
    <div class="main-container" >

      <div >
        <router-link to="/orders/register">
          <b-button class="order-button"  variant="outline-dark">주문하기</b-button>
        </router-link>
      </div>

      <div >
        <router-link to="/orders">
          <b-button class="order-button"  variant="outline-dark"
            >주문 내역 조회하기</b-button
          >
        </router-link>
      </div>

      <div class="icon-wrapper">
        <span class="material-symbols-outlined"> local_cafe </span>
      </div>
    </div>
  </div>
</template>
<script>


export default {
  components: {

  },

  async created() {
    this.$store.commit("SET_TITLE", "SSAFY-CAFE");
  },

};
</script>
```

views/order/Main.vue 수정

Main.vue 수정

- css 수정
- 테스트하기
  - 주문하기/주문내역조회
    라우터로 잘 이동되는지 테스트
  - footer 관리자페이지/메인페이지
    테스트

```
<style scoped>

.main-container{
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%);
}
.main-container>div{
  text-align: center;
  margin-top: 30px;
}

.icon-wrapper>span{
  font-size:80px;
}

.order-button{
  min-width: 180px;
}
</style>
```

views/admin/Main.vue 수정

방금 작성한 views/order/Main.vue 내용 복사

- /admin-menu/Main.vue 에 붙여넣는다.
- 기존에 주문 추가하기 수정
  - rotuer-link를 menus에 맞게 수정
- 주문목록 조회하기 수정
  - rotuer-link를 menus에 맞게 수정
- script 내용 추가
  - SET_TITLE을 활용해 이번에는 SSAFY-CAFE-관리자로 변경

```
<script>
export default {
  components: {},

  async created() {
    this.$store.commit("SET_TITLE", "SSAFY-CAFE-관리자");
  },
};
</script>
```

```
<div>
  <router-link to="/admin/menus/register">
    <b-button class="order-button" variant="outline-dark"
      >메뉴 추가하기</b-button
    >
  </router-link>
</div>

<div>
  <router-link to="/admin/menus">
    <b-button class="order-button" variant="outline-dark"
      >메뉴 조회하기</b-button
    >
  </router-link>
</div>

<div class="icon-wrapper">
  <span class="material-symbols-outlined"> local_cafe </span>
</div>
</div>
```

views/admin/Main.vue 수정

테스트

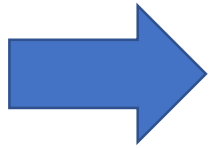
- 변경한 라우터 링크 또한 잘 동작하는지 확인
- SSAFY-CAFE-관리자로 변경되었는지 확인

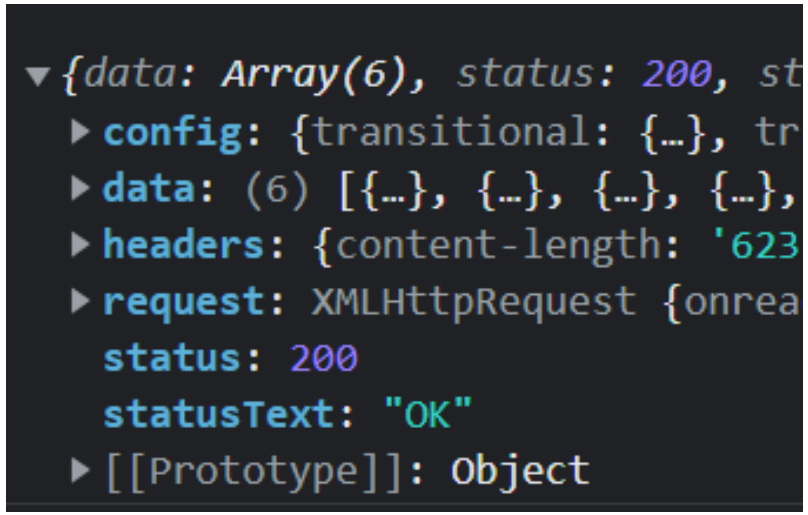
메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- 해당 라우터에 도착시 미리 설계해둔 axios.js 에 정의된 요청을 보낸다.
- resul의 data값을 data의 menus에 넣는다.

```vue
<template>
  <div>
      메뉴 리스트
  </div>
</template>

<script>
import { api } from '@/utils/axios';

export default {
  data() {
    return {
      menus: []
    };
  },
  async created(){
    this.$store.commit("SET_TITLE", "메뉴 목록");
    const result = await api.menus.findAll();
    console.log(result);
    this.menus = result.data;
  },

};
```





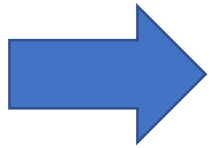
console.log(result)

메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- data의 menus를 기반으로 v-for을 활용해 해당 data를 Vue에서 보여지게 한다.

```
<template>
  <div>
    <div v-for="menu in menus" :key="menu.id">
      <div class="menu-container">
        <div class="menu-image"></div>

        <div class="menu-info-wrapper">
          <h2 class="menu-name">{{ menu.name }}</h2>
          <p class="menu-description">{{ menu.description }}</p>
        </div>

      </div>
    </div>
  </div>
</template>
```



카페라떼

라떼는.. 말이야

카푸치노

언빌리 "버블"

아메리카노

아메아메 좋아

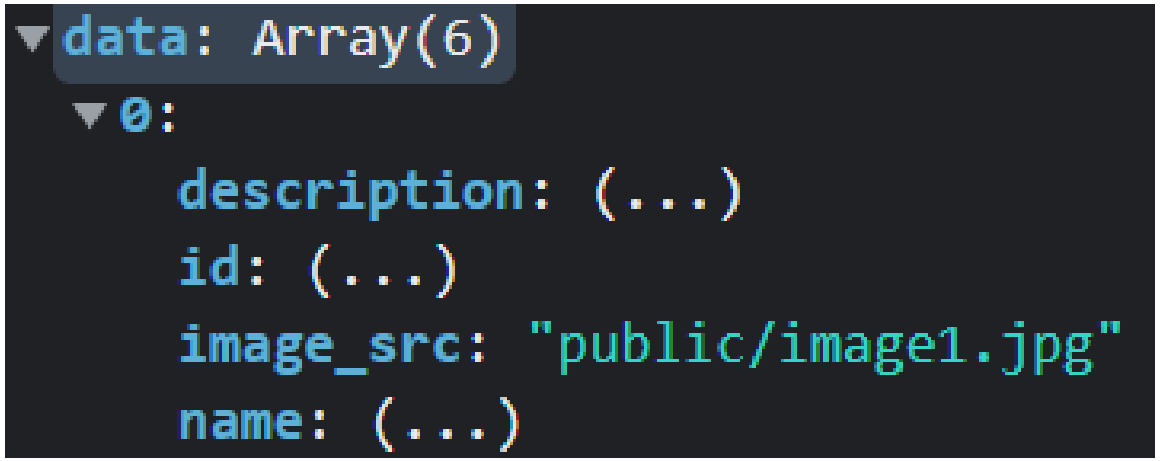
복숭아 아이스티

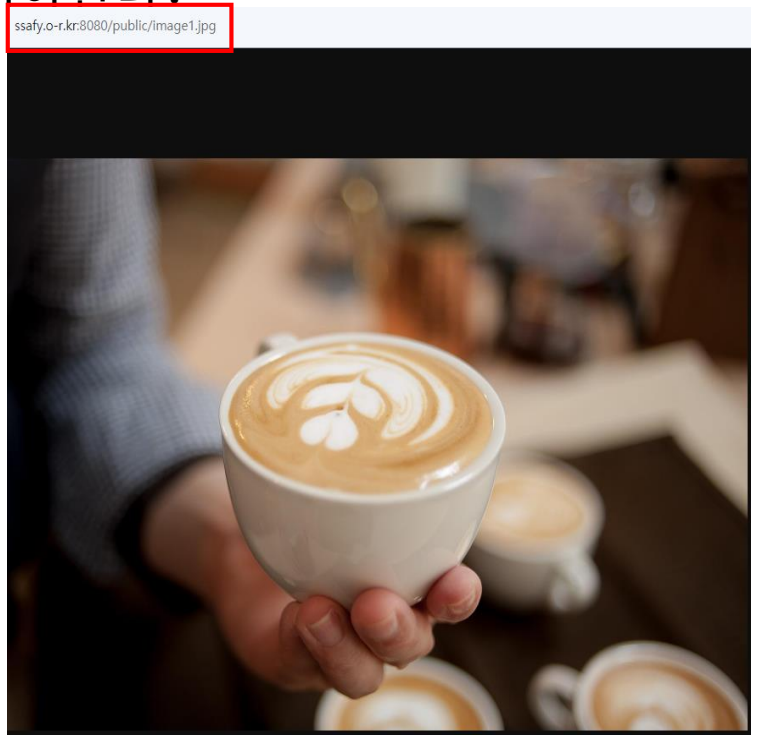
상큼한 복숭아!

메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- 이미지 넣기
  - 현재 menus에 담겨있는 image_src를 보면
    public/image1.jpg 형식으로 되어있다.
  - 이미지에 접근하기 위해서는
    백엔드의 주소 http;//host:8080/public/image1.jpg 형식으로 되어야한다.





메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- setImage 메서드 생성
  - image_src를 받아오면 호스트 주소와 포트를 붙여 리턴

```
<script>
import { api } from "@/utils/axios";

export default {
  data() {
    return {
      menus: [],
    };
  },
  async created() {
    this.$store.commit("SET_TITLE", "메뉴 목록");
    const result = await api.menus.findAll();
    console.log(result);
    this.menus = result.data;
  },

  methods:{
    setImage(image_src){
      return `http://ssafy.o-r.kr:8080/${image_src}`
    }
  }
};
</script>
```

메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- template 수정
  - 기존 class=menu-image 파트 수정
    - :style을 추가 해서 해당 menu의 image_src를 setImage를 통해 변환해서 전달

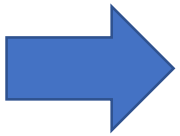
```vue
<template>
  <div>
    <main>
      <router-link to="/admin/menus/register">
          <b-button class="order-button w-100"  variant="outline-dark">메뉴 추가하기</b-button>
      </router-link>
      <div @click="moveDetail(menu.id)" v-for="menu in menus" :key="menu.id">
        <!-- {{ li }} -->
        <div
          class="menu-container"
        >


        <div class="menu-image" :style="`background-image:url(${setImage(menu.image_src)})`"></div>

          <div class="menu-info-wrapper">
            <h2 class="menu-name">{{ menu.name }}</h2>
            <p class="menu-description">{{ menu.description }}</p>
          </div>
        </div>
      </div>


    </main>
  </div>
</template>
```

메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- style 수정
  - css 추가 후 결과 확인

```
<style>
.menu-container{
    display: flex;
    align-items: center;
    border-bottom: 3px solid □□black;
}

.menu-info-wrapper{
  margin: 0 auto;
  text-align: center;
}

.menu-name {
  font-size: 25px;
  font-weight: bold;

}

.menu-description {
  padding-top: 10px;

}
.menu-image{
  background-size: cover;
  background-position: center;
  width:180px;
  height: 180px;
}
</style>
```

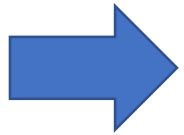


메뉴 리스트 조회 구현

/admin/menus (admin-menu/List.vue)

- 메뉴 추가 하기 router-link 구현
  - 메뉴 추가하기 를 눌러 이동 테스트

```
<template>
  <div>
    <main>
      <router-link to="/admin/menus/register">
          <b-button class="order-button w-100"  variant="outline-dark">메뉴 추가하기</b-button>
      </router-link>

      <div @click="moveDetail(menu.id)" v-for="menu in menus" :key="menu.id">
        <!-- {{ li }} -->
        <div
          class="menu-container"
        >

         <!-- 컴포넌트로 빼기 -->
        <div class="menu-image" :style="`background-image:url(${setImage(menu.image_src)})`"></div>

          <div class="menu-info-wrapper">
            <h2 class="menu-name">{{ menu.name }}</h2>
            <p class="menu-description">{{ menu.description }}</p>
          </div>
        </div>
      </div>


    </main>
  </div>
</template>
```



메뉴 추가 구현

/admin/menus/register (admin-menu/Register.vue)

- 기본 코드 작성
- name, description
  - v-model로 받아온다
- file
  - file 정보는 v-model에 담기지 않고 v-on:change 함수안에 담는
  - fileChange
    - file의 값을 this.file에 저장
- create 함수
  - 메뉴 작성을 위한 함수

```
FileList {0: File, length: 1}
  0: File {name: 'image2.jpg', la
  length: 1
  [[Prototype]]: FileList
```

e.target.files

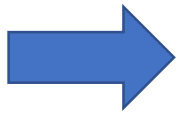
```
<template>
  <div>
    <div class="form-wrapper">
      <div>메뉴 이름: <input v-model="name" type="text" /></div>
      <div>메뉴 설명: <input v-model="description" type="text" /></div>
      <input v-on:change="fileChange" type="file" />

      <button @click="create">메뉴 추가하기</button>
    </div>

    <div class="image-wrapper"></div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      name: null,
      description: null,
      file: null,
    };
  },
  async created(){
    this.$store.commit("SET_TITLE", "메뉴 추가하기");
  },
  methods:{
    fileChange(e){
      console.log(e.target.files);
      this.file = e.target.files[0];
    },
    create(){
      console.log("메뉴 추가하기");
    }
  }
};
</script>

<style scoped>
</style>
```

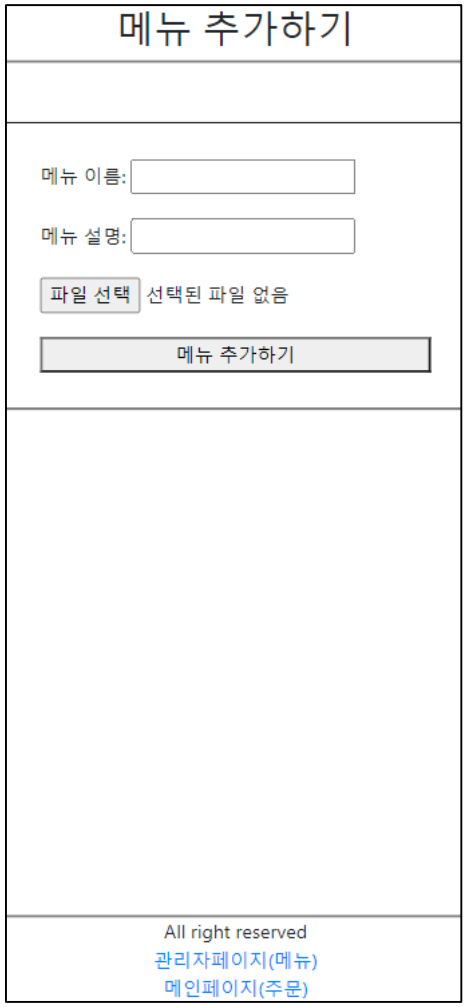
메뉴 추가 구현

/admin/menus/register (admin-menu/Register.vue)

- css 작성
- 결과 확인

```
<style>
.form-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 50px;
  border: 1px solid □□black;
  padding: 20px;
}

.form-wrapper > * {
  margin: 10px;
}

.image-wrapper {
  margin-top: 30px;
}
</style>
```





메뉴 추가 구현

/admin/menus/register (admin-menu/Register.vue)

- 이미지 미리보기 구현
  - 업로드 한 file을 URL.createObjectURL로 임시 URL을 생성 후 img의 src에 대입

```
<template>
  <div>
    <div class="form-wrapper">
      <div>메뉴 이름: <input v-model="name" type="text" /></div>
      <div>메뉴 설명: <input v-model="description" type="text" /></div>
      <input v-on:change="fileInput" type="file" />

      <div v-if="$route.params.id">
        <button @click="update">메뉴 수정하기</button>
        <button @click="updateImage">이미지 수정하기</button>
      </div>

      <button v-else @click="create">메뉴 추가하기</button>

    </div>

    <div class="image-wrapper" v-if="file">
      <img :src="setURL(file)" width="100%" />
    </div>
  </div>

</template>
```

```
methods:{
  fileChange(e){
    console.log(e.target.files);
    this.file = e.target.files[0];
  },
  setURL(file) {
    console.log(file);
    const imageURL = URL.createObjectURL(file);
    console.log(imageURL);
    return imageURL;
  },

  create(){
    console.log("메뉴 추가하기");
  }
}
```

메뉴 추가 구현

/admin/menus/register (admin-menu/Register.vue)

- 이미지 미리보기 테스트

메뉴 추가 구현

/admin/menus/register (admin-menu/Register.vue)

- 업로드 구현
  - api import
    - 자동완성으로 사용하다보면 @가 나오는데
    - @는 src를 의미한다.
  - api에 이름, 상세 설명, 파일까지 전달
    - 요청 시 나오는 message의 값을 alert로 띄우기
  - this.$router.push
    - 특정 라우터로 이동하는 메서드
    - this.$router.push("/admin/menus")
    - 요청에 성공하면 /admin/menus로 이동한다.
      - 작성 -> 성공시 메뉴 리스트로 이동

```
▼data:
   message: "메뉴 등록에 성공하였습니다."
   success: true
  ▶[[Prototype]]: Object
```

```
import { api } from '@/utils/axios';
export default {
```

```
async create() {
  if (!this.name || !this.description || !this.file) {
    alert("빈 값이 있습니다 내용을 전부 작성해주세요");
  }
  const result = await api.menus.create(
    this.name,
    this.description,
    this.file
  );
  console.log(result);
  // 요청 성공
  if (result.data.success) {
    alert(result.data.message);
    this.$router.push("/admin/menus");
  }
  // 요청 실패
  if (!result.data.success) {
    alert(result.data.message);
  }
},
```

메뉴 추가 구현

/admin/menus/register (admin-menu/Register.vue)

- 구현 테스트

메뉴 디테일 구현

/admin/menus (admin-menu/List.vue)

- 메뉴 디테일로 이동
  - 메뉴 중 한가지 클릭 -> 메뉴 디테일 라우터 /admin/menus/:id 로 이동

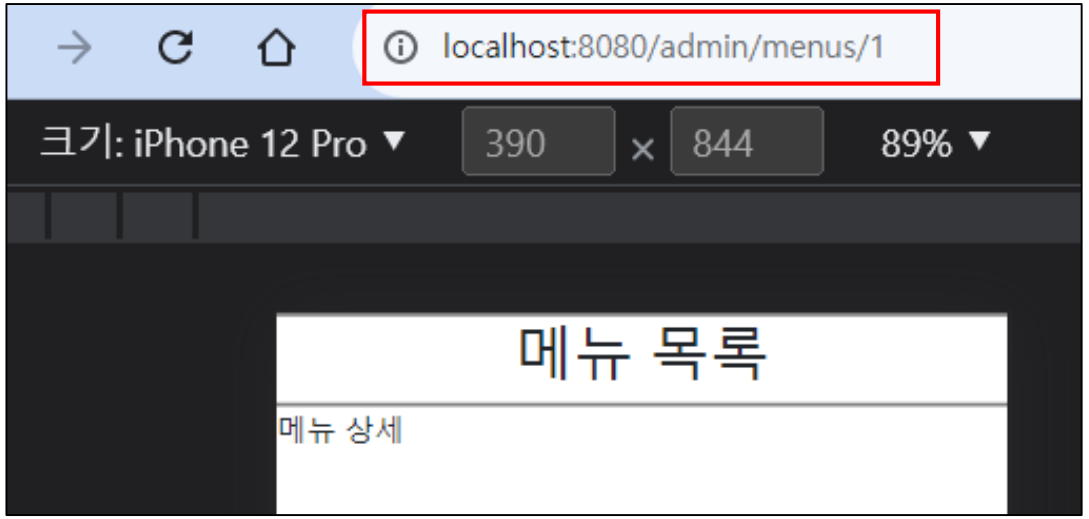
```vue
<template>
  <div>
    <router-link to="/admin/menus/register">
      <b-button class="order-button w-100" variant="outline-dark"
        >메뉴 추가하기</b-button
      >
    </router-link>
    <div @click="moveDetail(menu.id)" v-for="menu in menus" :key="menu.id">
      <div class="menu-container">
        <div
          class="menu-image"
          :style="`background-image:url(${setImage(menu.image_src)})`"
        ></div>

        <div class="menu-info-wrapper">
          <h2 class="menu-name">{{ menu.name }}</h2>
          <p class="menu-description">{{ menu.description }}</p>
        </div>
      </div>
    </div>
  </div>
</template>
```

```vue
methods: {
  setImage(image_src) {
    return `http://ssafy.o-r.kr:8080/${image_src}`;
  },
    moveDetail(id){
    this.$router.push(`/admin/menus/${id}`)
  }
},
```

메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

- /admin/menus 에서 특정 메뉴 클릭시 /admin/menus/:id로 오는지 체크

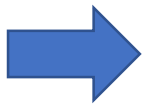


메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

- this.$route
  - 현재 라우터에 대한 정보가 담겨있다.
  - :id에 대한 정보는 this.$route.params.id에 담겨있다.

```vue
<template>
  <div>

  </div>
</template>

<script>
export default {
    data(){
        return{
            data: {}
        }
    },
    async created(){

        console.log(this.$route);
        console.log(this.$route.params);

    },

}
</script>

<style>

</style>
```



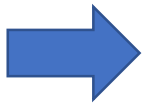
```
▼{name: 'menus-detail', meta: {…}, path: '/admin/menus/1', hash: '', query: {…}, …}
   fullPath: "/admin/menus/1"
   hash: ""
  ▶matched: [{…}]
  ▶meta: {}
   name: "menus-detail"
  ▶params: {id: '1'}
   path: "/admin/menus/1"
  ▶query: {}
  ▶[[Prototype]]: Object
▼{id: '1'} i
   id: "1"
  ▶[[Prototype]]: Object
```

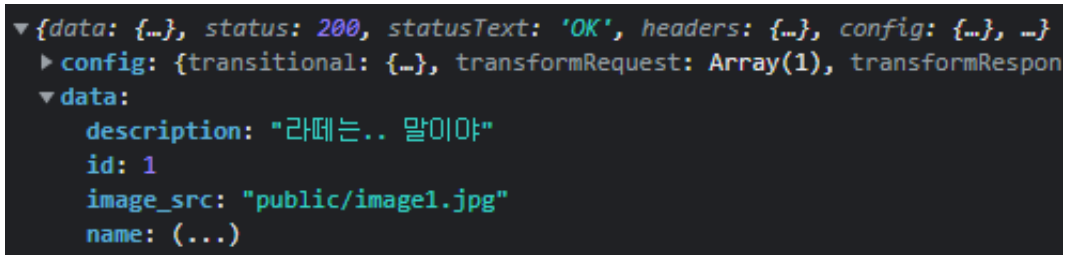
메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

- this.$route.params.id 를 활용해 api 요청 (/menus/:id)
  - 받아온 result.data 를 this.data에 넣는다.

```vue
<template>
  <div>

  </div>
</template>

<script>
import { api } from '@/utils/axios';
export default {
    data(){
        return{
            data: {}
        }
    },
    async created(){

        console.log(this.$route);
        console.log(this.$route.params);
        const result = await api.menus.findOne(this.$route.params.id);
        console.log(result);
        this.data = result.data;
    },

}
</script>

<style>

</style>
```





메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

- template 및 script 구현
  - vue-bootstrap의 card 활용
    - img-src는 setImage 함수를 통해
      http://호스트:포트/public/image.jpg 형태로 변환되어 보
  - 수정하기 클릭 -> /admin/menus/register/:id 로 이동
  - 목록 클릭 -> /admin/menus 로 이동

```vue
<template>
  <div>
    <b-card
      v-if="data"
      :title="data.name"
      :img-src="setImage()"
      img-alt="Image"
      img-top
      tag="article"
      class="mb-2 detail-card"
    >
      <b-card-text>
        {{ data.description }}
      </b-card-text>

      <b-button @click="moveRegister" href="#" variant="primary"
        >수정하기</b-button
      >
      <b-button @click="deleteMenu" href="#" variant="danger">삭제</b-button>
      <b-button @click="moveList" href="#" variant="outline-primary"
        >목록</b-button
      >
    </b-card>
  </div>
</template>
```

```vue
<script>
import { api } from "@/utils/axios";
export default {
  data() {
    return {
      data: {},
    };
  },
  async created() {
    console.log(this.$route);
    console.log(this.$route.params);
    const result = await api.menus.findOne(this.$route.params.id);
    console.log(result);
    this.data = result.data;
  },

  methods: {
    setImage() {
      return `http://ssafy.o-r.kr:8080/${this.data.image_src}`;
    },

    moveRegister() {
      this.$router.push(`/admin/menus/register/${this.$route.params.id}`);
    },
    async deleteMenu() {
      console.log("삭제 구현")
    },
    moveList() {
      this.$router.push("/admin/menus");
    },
  },
};
</script>
```

메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

- 수정하기, 목록 이동 테스트

메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

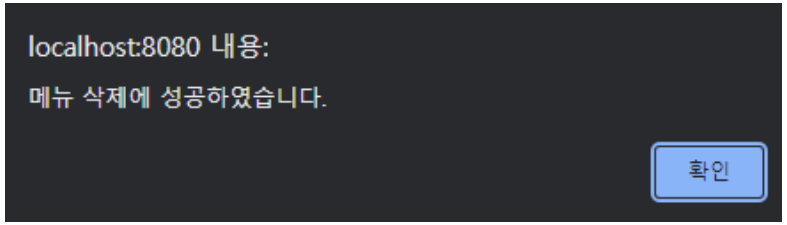
- 삭제 하기 구현
  - 삭제에 성공하는 경우 삭제 후 /admin/menus 라우터로 이동

```
methods: {
  setImage() {
    return `http://ssafy.o-r.kr:8080/${this.data.image_src}`;
  },

  moveRegister() {
    this.$router.push(`/admin/menus/register/${this.$route.params.id}`);
  },
  async deleteMenu() {
    const confirmResult = confirm("삭제하시겠습니까?");
    if (confirmResult) {
      const result = await api.menus.delete(this.$route.params.id);
      alert(result.data.message);
      this.$router.push("/admin/menus");
    }
  },
  moveList() {
    this.$router.push("/admin/menus");
  },
},
```

메뉴 디테일 구현

/admin/menus/:id (admin-menu/Detail.vue)

- 삭제 테스트



메뉴 수정 구현

/admin/menus/register/:id (admin-menu/Register.vue)

- 기존 Register.vue에 내용을 추가하기
  - update 모드인지 update 모드가 아닌지 정한다.
  - this.$route.params.id 가 존재하면 update 모드
    - ex) /admin/menus/register/3 – update 모드
    - ex) /admin/menus/register - 일반 등록

메뉴 수정 구현

/admin/menus/register/:id (admin-menu/Register.vue)

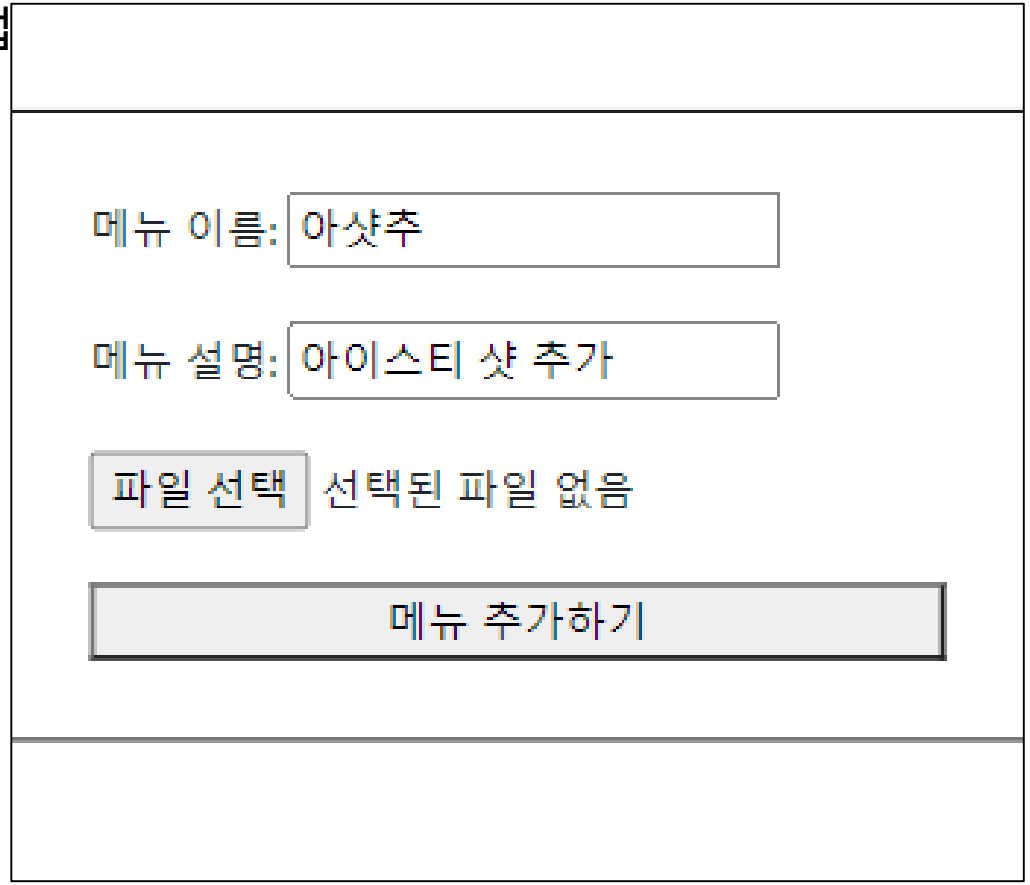
- 기존 Register.vue에 내용을 추가하기
  - async created 수정
  - this.$route.params.id
    - :id 가 존재한다면 update 모드
      - title을 메뉴 수정하기로 변경
    - 그것이 아니라면 일반 메뉴 추가하기
      - 메뉴 추가하기로 변경
  - :id로 /api/menus/:id 요청보내기
    - 해당 요청을 통해 받아온 값을 data의 name,description에 넣어준다.

```vue
<script>
import { api } from "@/utils/axios";
export default {
  data() {
    return {
      name: null,
      description: null,
      file: null,
    };
  },
  async created() {
    console.log(this.$route);

    if (this.$route.params.id) {
      this.$store.commit("SET_TITLE", "메뉴 수정하기");
      const result = await api.menus.findOne(this.$route.params.id);
      console.log(result);
      this.name = result.data.name;
      this.description = result.data.description;
    } else {
      this.$store.commit("SET_TITLE", "메뉴 추가하기");
    }
  },
```

메뉴 수정 구현

/admin/menus/register/:id (admin-menu/Register.vue)

- 기존 name,description이 변경되는지 체크
  - menus/:id 에 해당되는 값을 가져와 기존 코드 업



메뉴 수정 구현

/admin/menus/register/:id (admin-menu/Register.vue)

- 템플릿 수정
  - $route.params.id가 존재하는 경우 v-if로 수정 모드를 존재하지 않는경우 일반 메뉴 추가 모드로 보여준다.
  - update 함수
    - name 과 description을 업데이트하는 함수
    - update 완료 시 /admin/menus/:id 로 이동
  - updateImage 함수
    - 이미지만 업데이트하는 함수

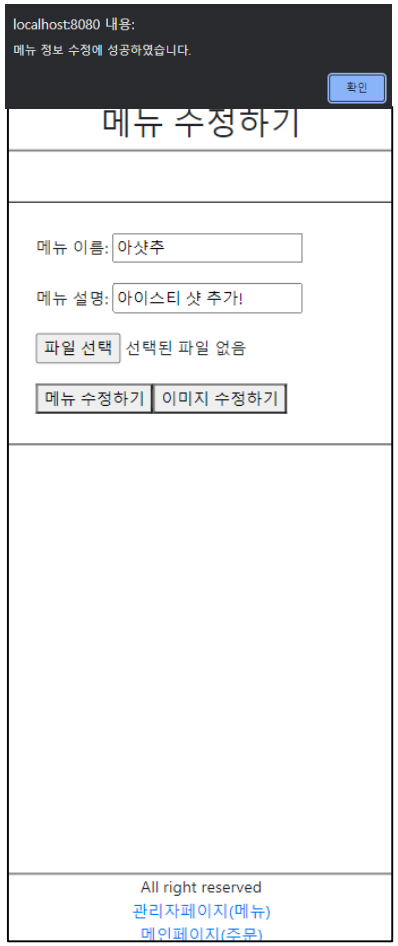
```vue
<template>
  <div>
    <div class="form-wrapper">
      <div>메뉴 이름: <input v-model="name" type="text" /></div>
      <div>메뉴 설명: <input v-model="description" type="text" /></div>
      <input v-on:change="fileInput" type="file" />

      <div v-if="$route.params.id">
        <button @click="update">메뉴 수정하기</button>
        <button @click="updateImage">이미지 수정하기</button>
      </div>

      <button v-else @click="create">메뉴 추가하기</button>

    </div>

    <div class="image-wrapper" v-if="file">
      <img :src="setURL(file)" width="100%" />
    </div>
  </div>


</template>
```

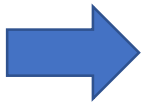
```js
async update() {
  // update;
  const result = await api.menus.update(this.$route.params.id, this.name, this.description);
  console.log(result);
  alert(result.data.message);
  this.$router.push(`/admin/menus/${this.$route.params.id}`)
},

async updateImage(){
  const result = await api.menus.updateImage(this.$route.params.id,this.file);
  console.log(result);
  alert(result.data.message);
}
```

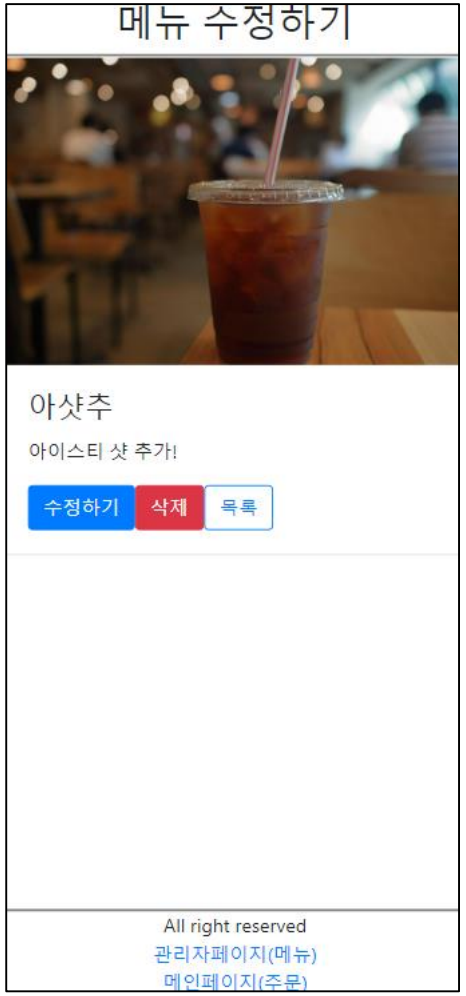
메뉴 수정 구현

/admin/menus/register/:id (admin-menu/Register.vue)

- 업데이트 체크



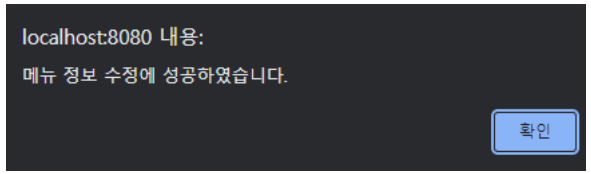




메뉴 수정 구현

/admin/menus/register/:id (admin-menu/Register.vue)

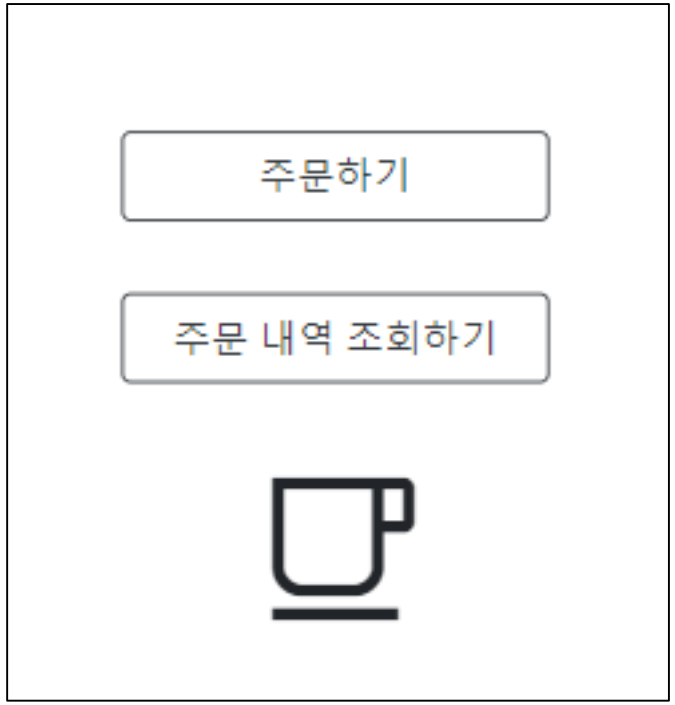
- 업데이트 이미지 체크
- 이미지 업데이트는 라우터 이동이 없기 때문에 메뉴 수정하기를 한번 더 눌러준다.



도전 - 주문하기

주문하기를 구현해보자

- 미리 정의해둔 라우터 활용
- API 명세서및 utils/axios.js 활용

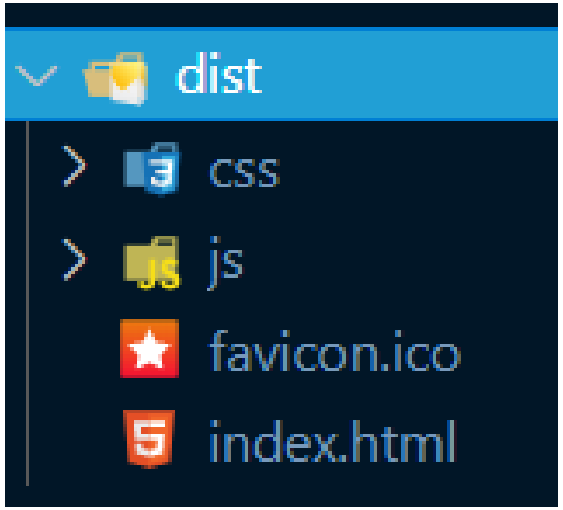


Vue 배포하기

npm run build

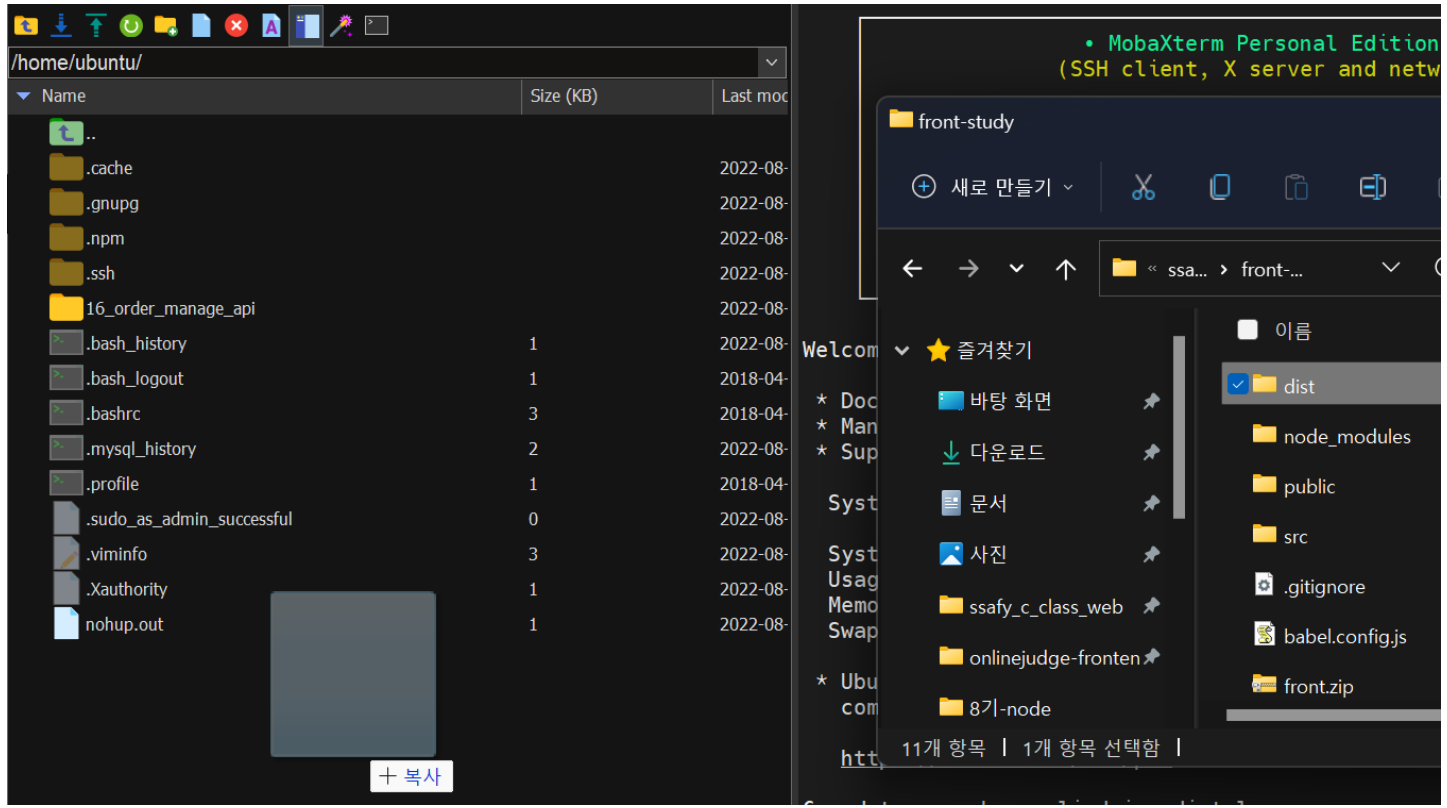
- build 명령어 실행으로 dist 폴더 생성
- 해당 dist 폴더를 AWS에 넣는 방식

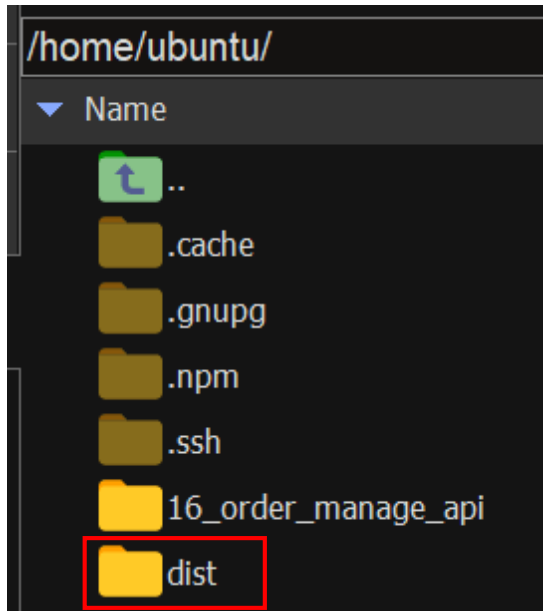
```
webpack performance recommendations:
You can limit the size of your bundles by using import() or require.ensure to lazy load some parts of your application.
For more info visit https://webpack.js.org/guides/code-splitting/

  File                                 Size                                    Gzipped

  dist\js\chunk-vendors.92b754f7.js    602.99 KiB                              165.66 KiB
  dist\js\app.d416bfd7.js              10.01 KiB                               3.37 KiB
  dist\css\chunk-vendors.c632c9ea.css  214.54 KiB                              31.62 KiB
  dist\css\app.afe355fc.css            1.17 KiB                                0.45 KiB

  Images and other types of assets omitted.
  Build at: 2022-08-24T04:26:16.834Z - Hash: 958686703bcb19d5 - Time: 28307ms

 DONE  Build complete. The dist directory is ready to be deployed.
 INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```



MobaXterm 접속

- 해당 dist 폴더 를 드래그해서 AWS 서버에 집어넣기





nginx 수정하기

- nginx가 깔려있다는 조건
  - nginx가 깔려있지 않는경우
    - sudo apt update
    - sudo apt install nginx
  - nginx가 깔린 후
    - sudo nano /etc/nginx/sites-available/default
    - root /home/ubuntu/dist;
    - try_files $uri $uri/ /index.html; 로 수정
      - ;이 빠지면 에러가 나니 주의
    - 저장 후 Ctrl + O Enter
      Ctrl + X

```
root /home/ubuntu/dist;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
}
```

Vue 배포하기

nginx reload

- sudo service nginx reload
- 해당 IP 혹은 도메인 주소 접속
  - 접속확인 테스트
  - 배포 완료

```
ubuntu@ip-172-31-30-8:/etc/nginx/sites-available$ sudo service nginx reload
ubuntu@ip-172-31-30-8:/etc/nginx/sites-available$
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미