KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

프론트엔드 준비

[KB] IT's Your Life

KB국민은행

프론트엔드 준비

프로젝트 만들기

vsc에

명령창에 터미널에

09 Vue+Spring₩scoula> npm init vue frontend

Vue.js - The Progressive JavaScript Framework

✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit Testing? ... No / Yes
✓ Add an End-to-End Testing Solution? » No
✓ Add ESLint for code quality? ... No / Yes

ㅊㅇ ㄹ

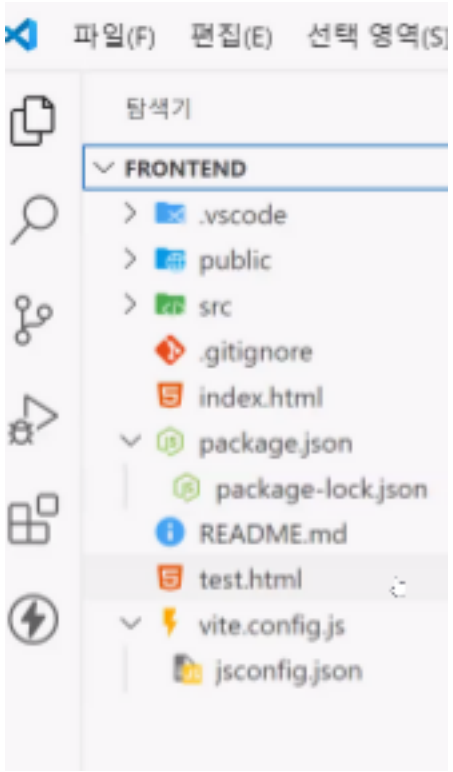
Scaffolding project in ...09 Vue+Spring₩scoula₩frontend...

Done. Now run:

cd frontend
npm install
npm run dev

프론트엔드 준비

- 기본 라이브러리 추가
  - npm install
  - npm install bootstrap@5 axios moment



프론트엔드 준비

index.html (fontawesom CDN 추가)

서버에 배포될때
/resources/index.html
에 위치할 예정.

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css"
      integrity="sha512-Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve5UzT7ecDGkiQ=="
      crossorigin="anonymous"
      referrerpolicy="no-referrer"
    />
    <title>Scoula App</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

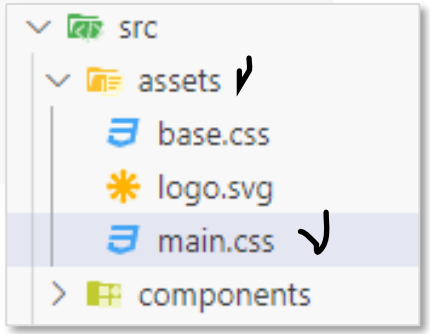
<link
rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css"
integrity="sha512-Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve
crossorigin="anonymous"
referrerpolicy="no-referrer"
/>

KB국민은행

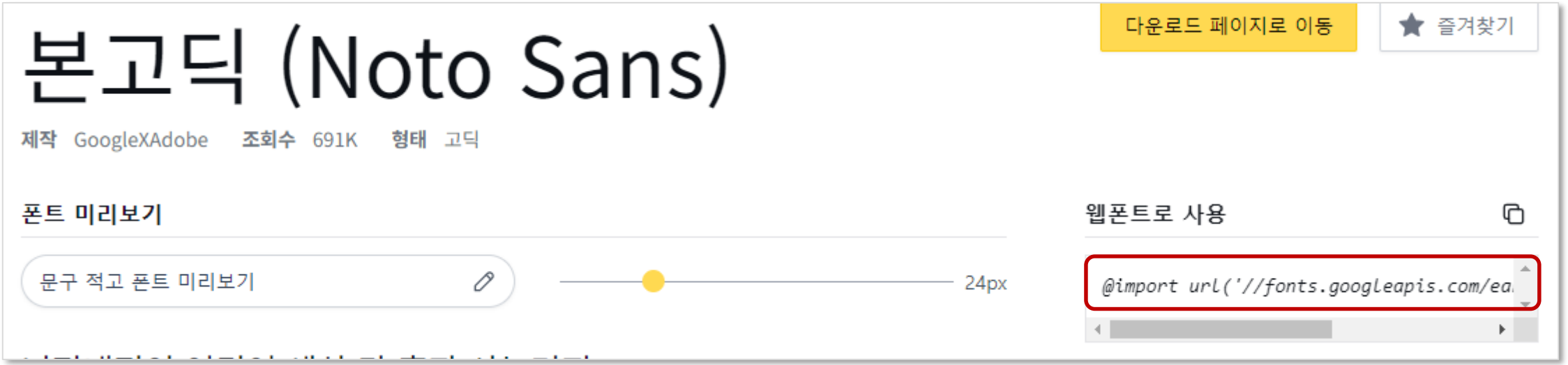
assets/main.css

```
@import
url('https://fonts.googleapis.com/css2?family=Noto+Sans+KR:wght@100;200;300;400;500;600;700;800;900&display=swap');

* {
  font-family: 'Noto Sans KR';
}
```

```
@import url('https://fonts.googleapis.com/css2?family=Noto+Sans+KR:wght@100;200;300;400;500;600;700;800;900&display=swap');
*{
  font-family: 'Noto Sans KR';
}
```



- webfont 설정
  - https://noonnu.cc/ 에서 noto 검색



프론트엔드 준비

main.js 엔트리포인트

```
import './assets/main.css';
import 'bootstrap/dist/css/bootstrap.css';

import { createApp } from 'vue';
import { createPinia } from 'pinia';

import App from './App.vue';
import router from './router';

const app = createApp(App);
app.use(createPinia());
app.use(router);

app.mount('#app');
```

전역 적용

토큰,유저정보 등 저장

미들웨어로 등록

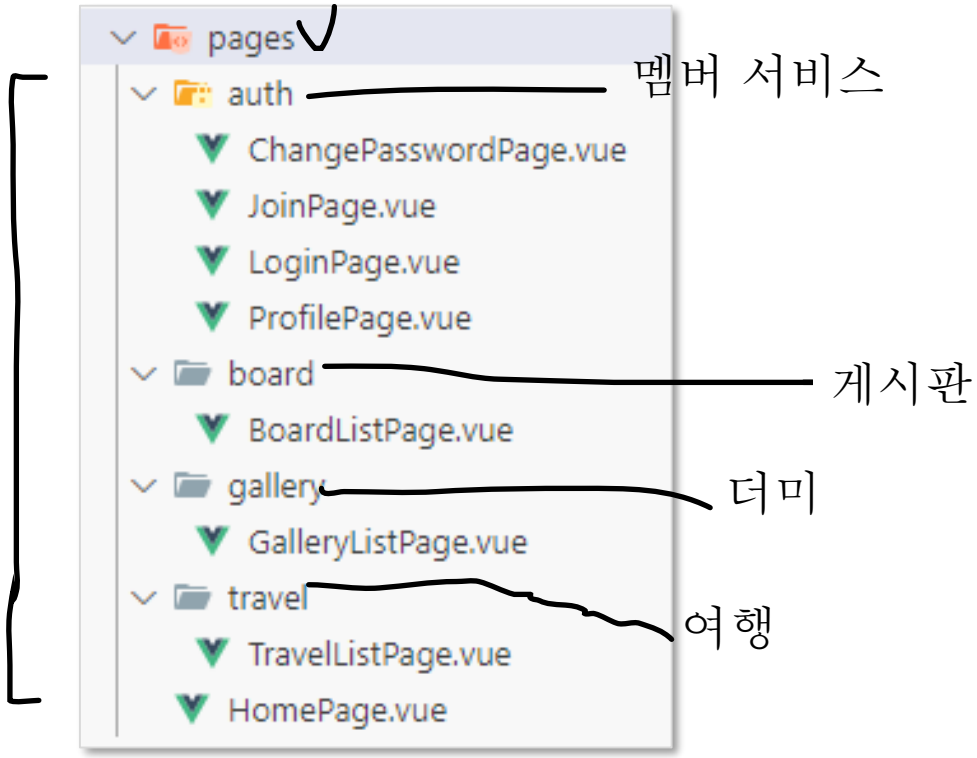
라우팅이 되도록 커스터마이징하자

App.vue

```
<script setup>
import { RouterView } from 'vue-router';
</script>

<template>
    <RouterView />
</template>

<style scoped></style>
```

프론트엔드 준비

기본 페이지 컴포넌트



라우터 테이블을 구성해야하는데

한테이블에 모든것을 구성하면 너무 많음

각각을 모듈화하자

- 기존 views 폴더는 삭제

프론트엔드 준비

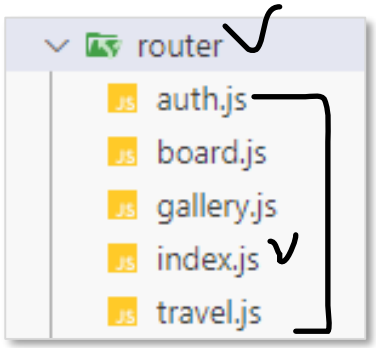
pages/HomePage.vue

```
<script setup>
</script>

<template>
  <h1>첫 번째 페이지</h1>
</template>
```

- 나머지 모든 페이지 → 타이틀만 구성

프론트엔드 준비

라우팅 준비



- 기능별로 라우팅 설정을 모듈화
  - index: 기본 라우팅
  - auth: login, logout, join 라우팅
  - board: 게시글 관련 라우팅
  - travel: 추천 여행지 관련 라우팅
  - gallery: 갤러리 관련 라우팅

디렉터리명을 모듈명으로 호출했을때 동작되는 index.js

라우팅 설정

router/auth.js 멤버 서비스 용.

```
export default [
  {
    path: '/auth/login',
    name: 'login',
    component: () => import('../pages/auth/LoginPage.vue'),
  },
  {
    path: '/auth/join',
    name: 'join',
    component: () => import('../pages/auth/JoinPage.vue'),
  },
  {
    path: '/auth/profile',
    name: 'profile',
    component: () => import('../pages/auth/ProfilePage.vue'),
  },
  {
    path: '/auth/changepassword',
    name: 'changepassword',
    component: () => import('../pages/auth/ChangePasswordPage.vue'),
  },
];
```

이런 표현은 지연 임포트.
필요할때 임포트 하겠다. 기동될때 말고

페이지 이동시
경로로 이동한 경우와 이름으로 이동하는 경우가 있다.
후자가 권장된다. 경로는 언제든지 바뀔수있어서

router/board.js

```
export default [
  {
    path: '/board/list',
    name: 'board/list',
    component: () => import('../pages/board/BoardListPage.vue'),
  },
];
```

router/travel.js

```
export default [
  {
    path: '/travel/list',
    name: 'travel/list',
    component: () => import('../pages/travel/TravelListPage.vue'),
  },
];
```

router/gallery.js

```
export default [
  {
    path: '/gallery/list',
    name: 'gallery/list',
    component: () => import('../pages/gallery/GalleryListPage.vue'),
  },
];
```

router/index.js router모듈 구성

```
import { createRouter, createWebHistory } from 'vue-router';
import HomePage from '../pages/HomePage.vue';
import authRoutes from './auth';
import boardRoutes from './board';
import travelRoutes from './travel';
import galleryRoutes from './gallery';

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    { path: '/', name: 'home', component: HomePage },
    ...authRoutes,
    ...boardRoutes,
    ...travelRoutes,
    ...galleryRoutes,
  ],
});

export default router;
```

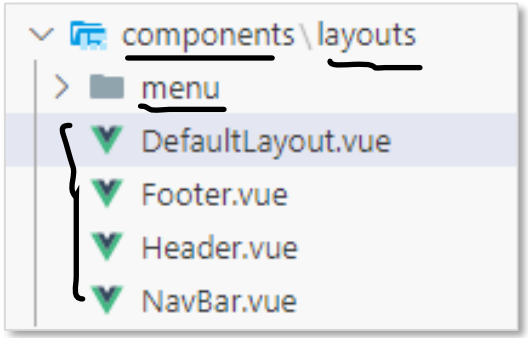
펼침연산자다!

npm run dev로 테스트해봐. 직접 url을 입력하여 테스트해보자
프론트엔드 포트는 5173, 백엔트 포트는 8088.
=> 나중에 origin문제가 발생한다!
=>우리가 security에서 cors허용 설정을 했쪼? origin안따지겠다고

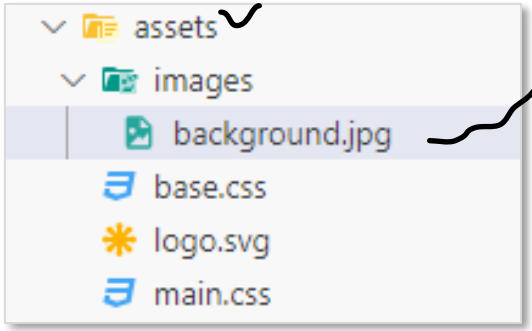
프론트엔드 준비

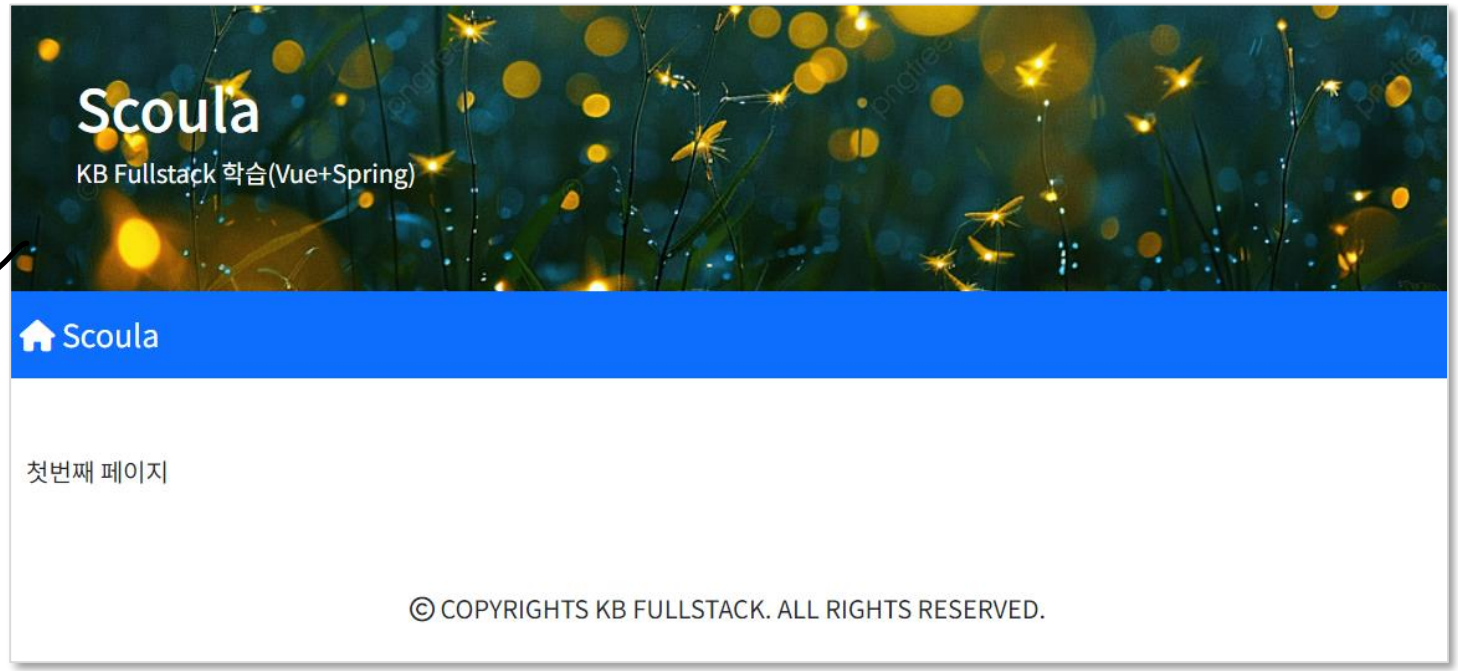
기본 레이아웃

- 기존 components/의 내용 모두 삭제
- 다음 구조로 컴포넌트 작성



- 헤더의 배경 이미지 준비



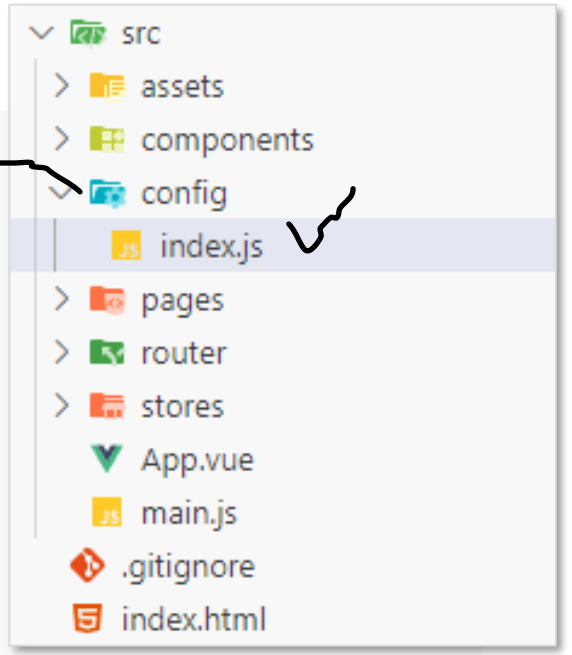


프론트엔드 준비

config/index.js

```
export default {
  title: 'Scoula',    // 메인 타이틀
  subtitle: 'KB Fullstack 학습(Vue+Spring)', // 서브 타이틀
  menus: [  // 메인 메뉴 구성 정보
    {
      title: '게시판',
      url: '/board/list',
      icon: 'fa-solid fa-paste',
    },
    {
      title: '여행',
      url: '/travel/list',
      icon: 'fa-solid fa-plane-departure',
    },
    {
      title: '갤러리',
      url: '/gallery/list',
      icon: 'fa-regular fa-images',
    },
  ],
```

폴더명이
모듈명이됨.
config모듈



메뉴에서 쓰일
제목과 같이 대응할 url 및 아이콘들 묶어놓음


```
export default {
  title: 'Scoula',  // 메인 타이틀
  subtitle: 'KB Fullstack 학습(Vue+Spring)', // 서브 타이틀
  menus: [  // 메인 메뉴 구성 정보
    {
      title: '게시판',
      url: '/board/list',
      icon: 'fa-solid fa-paste',
    },
    {
      title: '여행',
      url: '/travel/list',
      icon: 'fa-solid fa-plane-departure',
    },
    {
      title: '갤러리',
      url: '/gallery/list',
      icon: 'fa-regular fa-images',
    },
  ],
```

config/index.js

```
  accoutMenus: {    // 인증 관련 메뉴 정보
    login: {
      url: '/auth/login',
      title: '로그인',
      icon: 'fa-solid fa-right-to-bracket',
    },

    join: {
      url: '/auth/join',
      title: '회원가입',
      icon: 'fa-solid fa-user-plus',
    },
  },
};
```

```
accoutMenus: {  // 인증 관련 메뉴 정보
    login: {
      url: '/auth/login',
      title: '로그인',
      icon: 'fa-solid fa-right-to-bracket',
    },
    join: {
      url: '/auth/join',
      title: '회원가입',
      icon: 'fa-solid fa-user-plus',
    },
  },
};
```

프론트엔드 준비

components/layouts/Header.vue

아까 만진 config모듈

```vue
<script setup>
import config from '@/config';
</script>
<template>
  <div class="jumbotron p-5 bg-primary text-white">
    <h1>{{ config.title }}</h1>
    <p>{{ config.subtitle }}</p>
  </div>
</template>
<style scoped>
.jumbotron {
  background-image: url('@/assets/images/background.jpg');
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  color: white;
  padding: 2rem;
}
</style>
```

source를 기준으로 절대경로로 표현하겠다

✅ Q13. 다음은 Vite 설정에서 절대 경로 alias를 설정한 코드이다. '@'는 어느 디렉토리를 가리키는가?

```
resolve: {
  alias: {
    '@': fileURLToPath(new URL('__________', import.meta.url))
  }
}
```

./src

components/layouts/NavBar.vue

ref와 reactive 기억하세요
기본 참조타입

```
<script setup>
import { reactive, computed } from 'vue';
import config from '@/config';

let state = reactive({ isNavShow: false });

let navClass = computed(() => (state.isNavShow ? 'collapse navbar-collapse show' : 'collapse navbar-collapse'));

const toggleNavShow = () => (state.isNavShow = !state.isNavShow);
</script>
```

프론트엔드 준비

components/layouts/NavBar.vue

```
<template>
  <nav class="navbar navbar-expand-sm bg-primary navbar-dark">
    <div class="container-fluid">
      <router-link class="navbar-brand" to="/">
홈버튼 <i class="fa-solid fa-house"></i>
이네    Scoula
      </router-link>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#collapsibleNavbar"
햄버거          @click="toggleNavShow">
버튼이네  <span class="navbar-toggler-icon"></span>
      </button>
      <div :class="navClass" id="collapsibleNavbar">
          <!-- 추후 작업 예정 -->
      </div>
    </div>
  </nav>
</template>
<style></style>
```

href와 대응

SPA에서는 a태그 사용하면 안돼.
router-link사용해라

앞서 정의한 computed 속성

<template>
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">

<router-link class="navbar-brand" to="/">

Scoula
</router-link>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#collapsibleNavbar"
@click="toggleNavShow">

</button>
<div :class="navClass" id="collapsibleNavbar">
<!-- 추후 작업 예정 -->

</nav>
</template>
<style></style>

프론트엔드 준비

components/layouts/Footer.vue

```
<script></script>
<template>
  <div class="my-5 p-3 text-center">
      <i class="fa-regular fa-copyright"></i>
      COPYRIGHTS KB FULLSTACK. ALL RIGHTS RESERVED.
  </div>
</template>
```


```
<script></script>
<template>
  <div class="my-5 p-3 text-center">
      <i class="fa-regular fa-copyright"></i>
      COPYRIGHTS KB FULLSTACK. ALL RIGHTS RESERVED.
  </div>
</template>
```

components/layouts/DefaultLayout.vue

```
<script setup>
import Header from './Header.vue';
import NavBar from './NavBar.vue';
import Footer from './Footer.vue';
</script>

<template>
  <div class="container">
    <Header />
    <NavBar />
    <div class="content my-5 px-3">
      <slot></slot>
    </div>
    <Footer />
  </div>
</template>
```

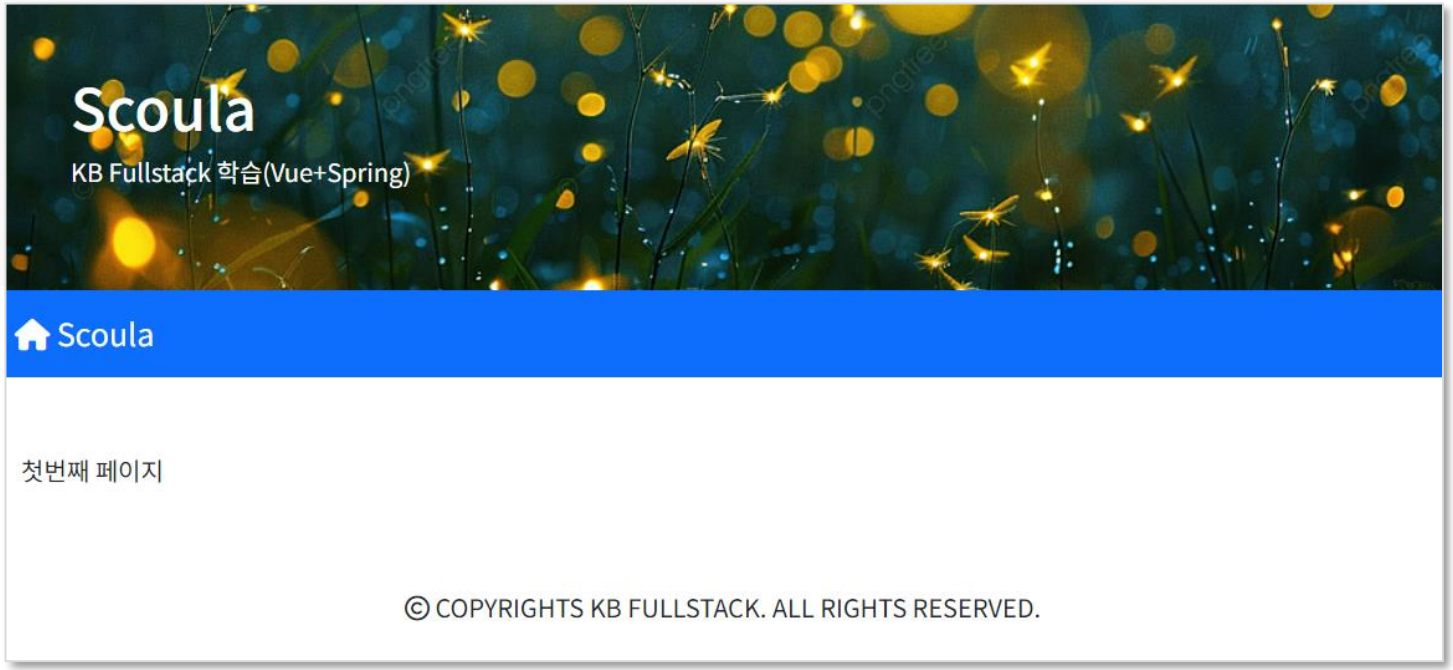
자식은 위치 결정만, 내용은 부모가 결정하는 slot기억하죠잉

App.vue

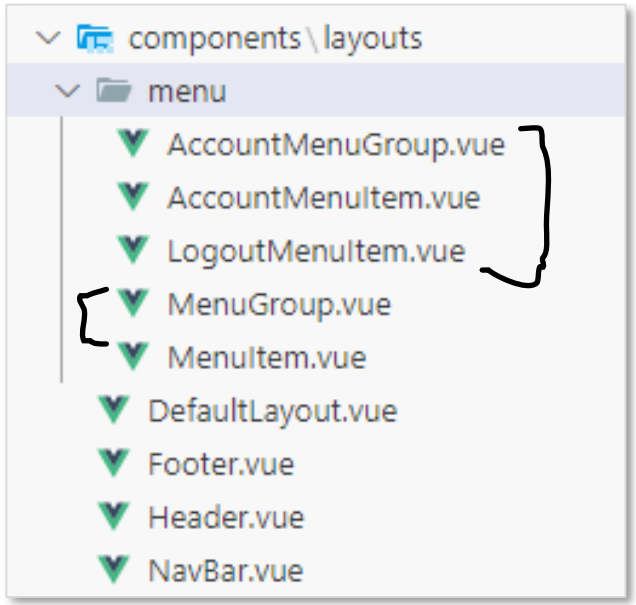
```
<script setup>
import { RouterView } from 'vue-router';
import DefaultLayout from './components/layouts/DefaultLayout.vue';
</script>

<template>
  <DefaultLayout>
    <RouterView />
  </DefaultLayout>
</template>

<style scoped></style>
```

부모가 내용 결정

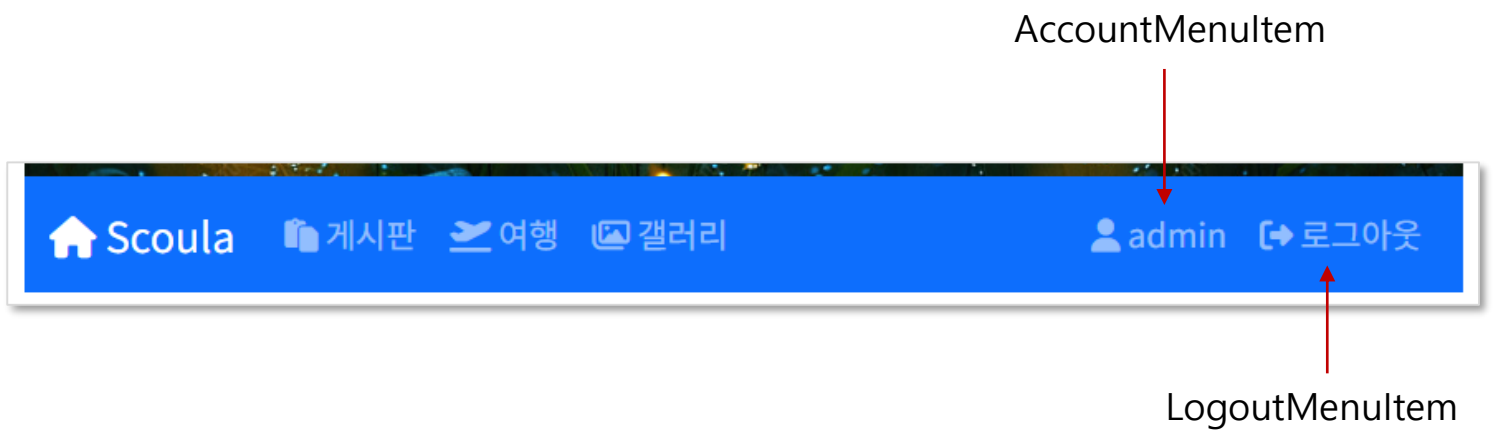
DefaultLayout.vue의 slot의 내용.



메뉴의 구성


components \ layouts
menu
AccountMenuGroup.vue
AccountMenuItem.vue
LogoutMenuItem.vue
MenuGroup.vue
MenuItem.vue
DefaultLayout.vue
Footer.vue
Header.vue
NavBar.vue


MenuGroup
AccountMenuGroup
MenuItem
애도menuitem
Scoula
게시판
여행
갤러리
로그인
회원가입


AccountMenuItem
Scoula
게시판
여행
갤러리
admin
로그아웃
LogoutMenuItem

components/layouts/menu/MenuItem.vue

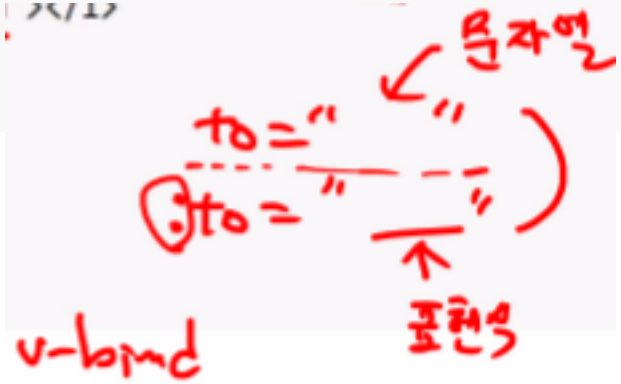
```
<script setup>
const props = defineProps({
  menu: { Type: Object, required: true },
});
</script>

<template>
  <li class="nav-item">
    <router-link class="nav-link" :to="menu.url">
      <i :class="menu.icon"></i>
      {{ menu.title }}
    </router-link>
  </li>
</template>
<style></style>
```

나중에 이렇게 쓰임
<MenuItem 정의한프로퍼티=값>

config모듈의 menus배열중에 하나인 객체가 들어갈 예정

템플릿에서는
속성명으로 바로 접근 가능

js에서는 props로 접근 가능.



components/layouts/menu/MenuGroup.vue

```
<script setup>
import MenuItem from './MenuItem.vue';
const props = defineProps({
  menus: { Type: Array, required: true },
});
</script>

<template>
  <ul class="navbar-nav">
    <MenuItem v-for="menu in menus" :menu="menu"  />
  </ul>
</template>
```

config모듈에서 menus배열

키 속성 줘야하는거 권장. :key="" 특히 변화가 많은 목록인 경우

components/layouts/menu/AccountMenuItem.vue

```
<script setup>
defineProps({ username: String });
</script>

<template>
  <li class="nav-item">
    <router-link class="nav-link" to="/auth/profile">
      <i class="fa-solid fa-user"></i>
      {{ username }}
    </router-link>
  </li>
</template>
```

components/layouts/menu/LogoutMenuItem.vue

```
<script setup>
import { useRouter } from 'vue-router';

const router = useRouter();
const logout = (e) => {
  // 로그아웃
  router.push('/');
};
</script>

<template>
  <a href="#" class="nav-link" @click.prevent="logout">
    <i class="fa-solid fa-right-from-bracket"></i>
    로그아웃
  </a>
</template>
```

useRouter, useRoute 구분 잘해라잉

router모듈의 export. /router/index.js.

pinia 상태관리 코드 들어갈 예정

useRoute는 현재 라우트

이벤트 수식어로 한정
이번 수식어는 default event
핸들러를 막는 것

components/layouts/menu/AccountMenuGroup.vue

```
<script setup>
import { computed } from 'vue';
import MenuItem from './MenuItem.vue';
import AccountMenuItem from './AccountMenuItem.vue';
import LogoutMenuItem from './LogoutMenuItem.vue';
import config from '@/config';

const { login, join } = config.accoutMenus;

const islogin = computed(() => false); // 임시: 로그인하지 않음
const username = computed(() => '');  // 임시: 사용자명 없음
</script>
```

아직 pinia가 없기에

components/layouts/menu/AccountMenuGroup.vue

```
<template>
  <ul class="navbar-nav ms-auto">
    <template v-if="islogin">
      <AccountMenuItem :username="username" />
      <LogoutMenuItem />
    </template>
    <template v-else>
      <MenuItem :menu="login" />
      <MenuItem :menu="join" />
    </template>
  </ul>
</template>
```

components/layouts/NavBar.vue

```
<script setup>
import { reactive, computed } from 'vue';
import config from '@/config';
import MenuGroup from './menu/MenuGroup.vue';
import AccountMenuGroup from './menu/AccountMenuGroup.vue';
…
</script>
<template>
  <nav class="navbar navbar-expand-sm bg-primary navbar-dark">
    <div class="container-fluid">
      …
      <div :class="navClass" id="collapsibleNavbar">
        <!-- 추후 작업 예정 -->
        <MenuGroup :menus="config.menus" />
        <AccountMenuGroup />
      </div>
    </div>
  </nav>
</template>
<style></style>
```

좌측메뉴, 우측메뉴

프론트엔드 준비

api 연결을 위한 proxy 서버 설정 및 배포 설정

- 백엔드 서버로 요청을 포워딩할 proxy 설정

- npm run build 명령시 결과 생성 폴더를 백엔드의 resources 폴더로 지정

frontend 배포본
배포할 시점에서 배포해보자.
npm run build

=>frontend/dist/에 다 생성됨.

해당 파일들을 서버에 배포하면된다.

프론트엔드 배포본과 백엔드 배포본이 있다면 어덯게 구성하는 것이 좋을까?
2가지 방법이 있다.
1. 프론트엔드 배포를 위한 웹서버(아파치,nginx)를 구성 백엔드도 따로 구성. 백엔드는 관련 기능에 따라 나눌수 있다.

그래서 클라이언트에서 각각 기능에 따라 다른 백엔드 서버와 연동시켜서 운용할 수 있다. MSA도 이것의 연장선이다

2번째 방법. 서버는 지금 하나. 백엔드 서버에 프론트엔드를 배포하는 것.
브라우저가 요청을 보내고 index.html, 관련 파일들을 받아옴.
운용하다가 BL이 필요할때 서버에 api통신요청을 보냄.

webapp>resources의 절대경로를 복사하고
npm run build 의 결과의 경로를 복사한 경로로 지정.
dist아래에 들어있는 파일들이 webapp/resources/아래에 결과가 들어가

vite.config.js

```
import { fileURLToPath, URL } from 'node:url';
…
export default defineConfig({
  plugins: [vue()],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url)),
    },
  },
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:8080',
      },
    },
  },
  build: {
    outDir: 'C:/KB_Fullstack/09_Spring+Vue/scoula/backend/src/main/webapp/resources',
  },
});
```

앞서 말한 설정

복사한 절대 경로

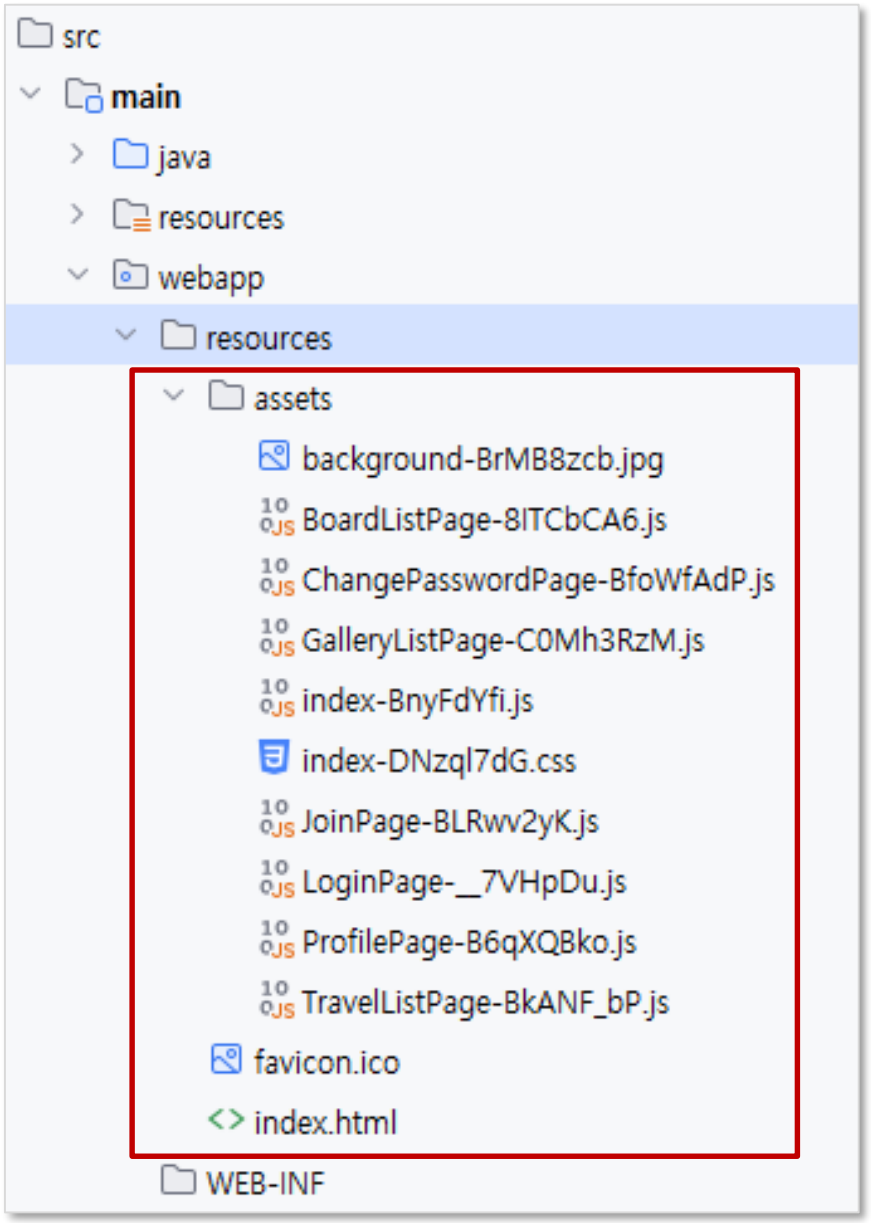
기존 만들어진 dist디렉토리는 삭제하자

프론트엔드 준비

- **npm run build**
  - backend의 resources 폴더에 결과물 생김

짜잔

후에 톰캣을
기동시키면
완성ㅇㅇㅇ



백엔드 기동

http://localhost:8080/