KB금융그룹 | 국민의 평생 금융파트너

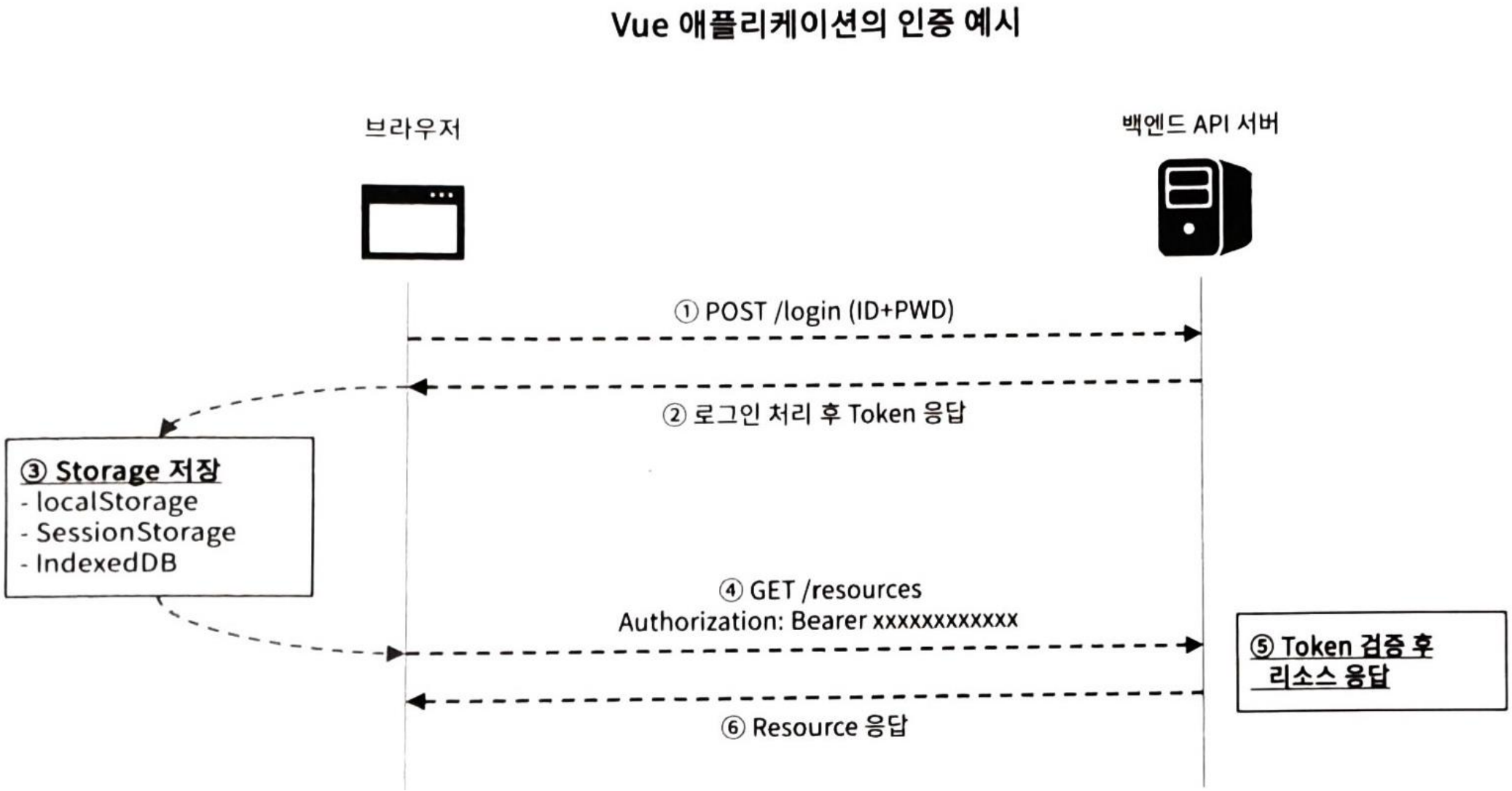
2025년 상반기 K-디지털 트레이닝

라우팅과 인증 처리

[KB] IT's Your Life

KB 국민은행

토큰 기반 인증 개요


Vue 애플리케이션의 인증 예시
브라우저
백엔드 API 서버
① POST /login (ID+PWD)
② 로그인 처리 후 Token 응답
③ Storage 저장
- localStorage
- SessionStorage
- IndexedDB
④ GET /resources
Authorization: Bearer xxxxxxxxxxx
⑤ Token 검증 후
리소스 응답
⑥ Resource 응답

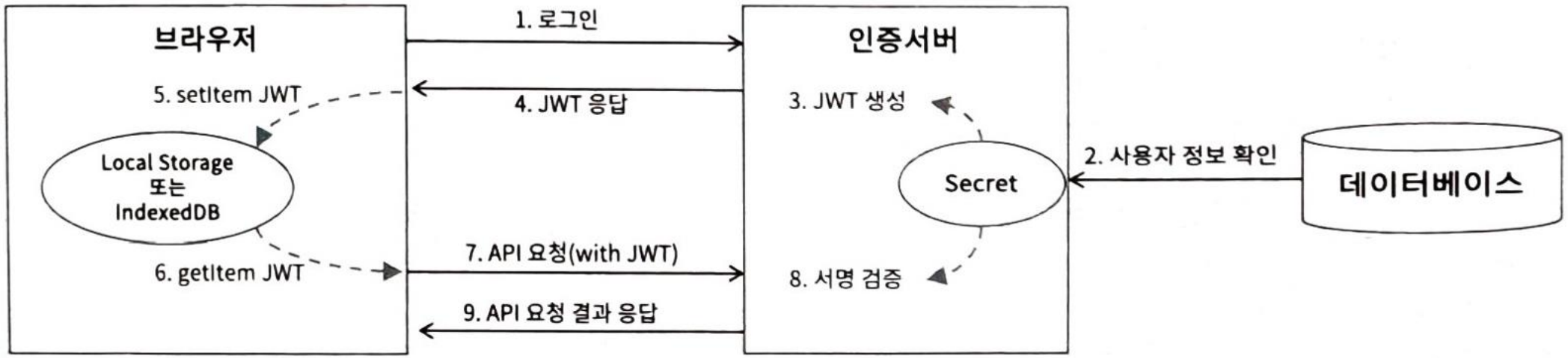
토큰 기반 인증

JWT(JSON Web Token)

- Token 정보를 별도로 백엔드 쪽 데이터베이스 등에 저장하지 않아도 됨
- Header.Payload.Signature 영역으로 구분
  - 세번째 서명 영역

```
Signature = HMAC( ( Base64(Header) + "." + Base64(Payload) ), Secret)
세 번째 영역 서명 = Base64(Signature)
```

- JWT를 이용한 인증 흐름도



토큰 기반 인증

- 백엔드에서 생성한 token의 저장
  - 클라이언트에서 저장
  - 저장소
    - IndexedDB, localStorage, sessionStorage 등을 사용해서 브라우저 저장소에 저장
- 저장소에 저장된 Token을 서버에 요청 보낼때 매번 같이 전송
  - Authorization HTTP Header 또는
  - HTTP Cookie에 실어 전송

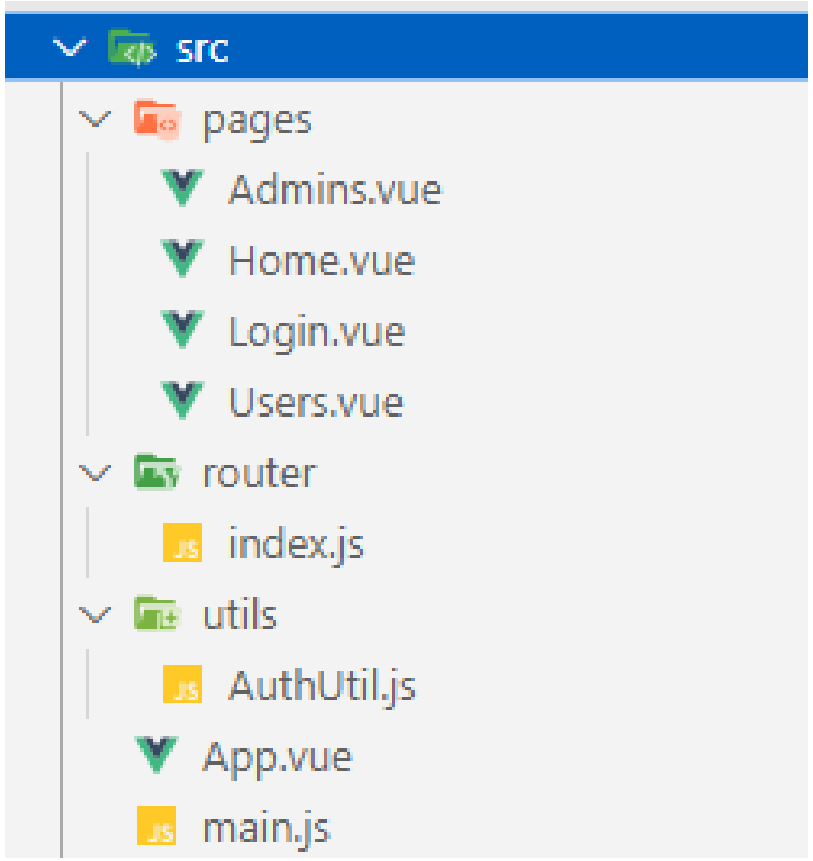
내비게이션 가드를 이용한 로그인 화면 전환

- 브라우저에 보유한 토큰이 있는지 확인하고 토큰이 없거나 파기된 토큰(expired token)인 경우
  - Login 화면의 경로로 자동 전환
- 로그인이 된 후에는 다시 로그인 전에 접근하려 했던 경로로 이동

→ 내비게이션 가드로 구현 가능

프로젝트 생성

```
npm init vue router-auth-test
...
Router 설치 → Yes
...
cd router-auth-test
npm install
```



라우팅 정보

- /, /login : 누구나 접근 가능한 퍼블릭 액세스 페이지
- /users, /admins는 지정한 역할(role)을 가진 사용자만 접근 가능하도록 설정

경로	라우트명	컴포넌트	설명
/	home	Home	홈 화면. 누구나 접근 가능(everybody)
/login	login	Login	로그인 화면. 누구나 접근 가능(everybody)
/users	users	Users	사용자 전용 화면. users 역할(role)을 가진 사용자만 접근 가능
/admins	admins	Admins	관리자 전용 화면. adimins 역할(role)을 가진 사용자만 접근 가능

src/utils/AuthUtil.js

```
const staticUsers = [
    { userid:"user1", password:"1234", roles: ["users"] },
    { userid:"user2", password:"1234", roles: ["users"] },
    { userid:"admin", password:"1234", roles: ["users","admins"] },
]

const pathsToRoles = [
    { path:"/", roles: ["everybody"] },
    { path:"/login", roles: ["everybody"] },
    { path:"/users", roles: ["users","admins"] },
    { path:"/admins", roles: ["admins"] },
]
```

src/utils/AuthUtil.js

```
//userInfo가 null이면 로컬 스토리지 삭제
const setUserInfo = (userInfo) => {
    if (userInfo && userInfo.authenticated) {
        window.localStorage.setItem("userInfo", btoa(JSON.stringify(userInfo)))
    } else {
        window.localStorage.removeItem("userInfo")
    }
}

const getUserInfo = () => {
    let strUserInfo = window.localStorage.getItem("userInfo");
    if (!strUserInfo) {
        return { authenticated: false};
    } else {
        return JSON.parse(window.atob(strUserInfo));
    }
}
```

src/utils/AuthUtil.js

```
const loginProcess = (userid, password, successCallback, failCallback) => {
    //이 부분은 백엔드 API 인증 서버와 HTTP로 통신하여 인증 처리해야 함.
    const user= staticUsers.find((u) => u.userid===userid && u.password===password);
    if (user) {
        let userInfo = { authenticated:true, userid:user.userid, roles: user.roles };
        setUserInfo(userInfo);
        successCallback();
    } else {
        if (failCallback) failCallback();
    }
}

const logoutProcess = (callback) => {
    setUserInfo(null);      //로컬 스토리지 삭제
    callback();
}
```

src/utils/AuthUtil.js

```
//경로와 사용자 정보의 role을 기반으로 접근 허가 여부 결정(true/false)
const isMatchToRoles = (reqPath) => {
    //{ path:"/", roles: ["everybody"] }
    const path = pathsToRoles.find((pr)=>pr.path === reqPath);
    //경로가 없다면 접근 불가
    if (!path)  return false;

    const userInfo = getUserInfo();
    //인증되지 않았다면 everybody 가 지정된 경로만 접근 가능
    if (userInfo.authenticated === false) {
        return path.roles.find((p)=>p==="everybody") ? true : false;
    } else {
        //인증이 되었다면 userInfo의 roles와 path.roles에 동일한 것이 있어야 함.
        let isAccessible = false;
        if (path.roles.indexOf('everybody') > -1) {
            isAccessible = true;
        } else {
            for (let i=0; i < userInfo.roles.length; i++) {
                let role = userInfo.roles[i];
                const index = path.roles.indexOf(role);
                if (index >= 0) {
                    isAccessible = true;
                    break;
                }
            }
        }
```

src/utils/AuthUtil.js

```
        return isAccessible;
    }
}

export { isMatchToRoles, loginProcess, logoutProcess, getUserInfo };
```

src/pages/Login.vue

```
<template>
    <div>
        <h2>로그인</h2>
        사용자 : <input type="text" v-model="info.userid" /><br />
        암호  : <input type="password" v-model="info.password" /><br />
        <br />
        <button @click="login">로그인</button>
    </div>
</template>

<script>
import { reactive } from 'vue';
import { useRoute, useRouter } from 'vue-router';
import { loginProcess } from '@/utils/AuthUtil.js';
```

src/pages/Login.vue

```
export default {
    name : "Login",
    setup() {
        const router = useRouter();
        const currentRoute = useRoute();
        const fromname = currentRoute.query.fromname;

        const info = reactive({ userid:"", password:"" });

        const successCallback = () => {
            if (fromname) router.push({ path: fromname })
            else router.push({ name:'home' })
        }
        const failCallback = () => {
            alert('로그인 실패');
        }

        const login = ()=> {
            loginProcess(info.userid, info.password, successCallback, failCallback)
        }

        return { info, login };
    }
}
</script>
```

src/pages/Users.vue

```
<template>
    <div>
        <h2>Users</h2>
        <p>users 역할이 있어야만 접근할 수 있는 페이지</p>
        <p>사용자 : {{userInfo.userid}}</p>
        <p>사용자의 역할 : [ {{userInfo.roles.join(', ')}} ]</p>
        <button @click="logout">로그아웃</button>
    </div>
</template>

<script>
import { getUserInfo, logoutProcess } from '@/utils/AuthUtil.js'
import { useRouter } from 'vue-router';

export default {
    name : "Users",
    setup() {
        const router = useRouter();
        const userInfo = getUserInfo();
        const logout = () => {
            logoutProcess(()=>{
                router.push({ name:'home' });
            })
        }
        return { userInfo, logout }
    }
}
</script>
```

src/pages/Home.vue

```
<template>
    <div>
        <h2>Home</h2>
        <p>인증되지 않아도 접근 가능한 페이지</p>
        <div v-if="data.userInfo.authenticated">
            <p>사용자 : {{data.userInfo.userid}}</p>
            <p>사용자의 역할 : [ {{data.userInfo.roles.join(', ')}} ]</p>
            <button @click="logout">로그아웃</button>
        </div>
    </div>
</template>
```

src/pages/Home.vue

```
<script>
import { getUserInfo, logoutProcess } from '@/utils/AuthUtil.js'
import { useRouter } from 'vue-router';
import { reactive } from 'vue';

export default {
    name : "Home",
    setup() {
        const router = useRouter();
        const data = reactive({ userInfo: getUserInfo() })
        const logout = () => {
            logoutProcess(()=>{
                data.userInfo = {};
                router.push({ name:'home' });
            })
        }
        return { data, logout }
    }
}
</script>
```

src/pages/Admins.vue

```
<template>
    <div>
        <h2>Admins</h2>
        <p>admins 역할이 있어야만 접근할 수 있는 페이지</p>
        <p>사용자 : {{userInfo.userid}}</p>
        <p>사용자의 역할 : [ {{userInfo.roles.join(', ')}} ]</p>
        <button @click="logout">로그아웃</button>
    </div>
</template>

<script>
import { getUserInfo, logoutProcess } from '@/utils/AuthUtil.js'
import { useRouter } from 'vue-router';

export default {
    name : "Admins",
    setup() {
        const router = useRouter();
        const userInfo = getUserInfo();
        const logout = () => {
            logoutProcess(()=>{
                router.push({ name:'home' });
            })
        }
        return { userInfo, logout }
    }
}
</script>
```

src/router/index.js

```
import { createRouter, createWebHistory } from 'vue-router';
import { isMatchToRoles } from '@/utils/AuthUtil.js';
…

const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/login', name:'login', component: Login },
        { path: '/users', name:'users', component: Users },
        { path: '/admins', name:'admins', component: Admins },
    ]
})

router.beforeEach((to)=>{
    if (!isMatchToRoles(to.path)) {
        return { name:'login', query: { fromname:to.name } };
    }
})

export default router;
```

src/main.js

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

const app = createApp(App)
app.use(router)
app.mount('#app')
```

src/App.vue

```
<template>
  <div class="container">
    <div>
      <router-link to="/">Home</router-link> 
      <router-link to="/login">Login</router-link> 
      <router-link to="/users">Users</router-link> 
      <router-link to="/admins">Admins</router-link> 
    </div>
    <hr />
    <div>
      <router-view />
    </div>
  </div>
</template>

<script>
export default {
  name : "App"
}
</script>

<style>
.container { margin:10px; }
</style>
```

Local Storage에 저장된 토큰