앞에 들어야 함.

2025년 상반기 K-디지털 트레이닝

회원관리-정보 수정(백엔드)

[KB] IT's Your Life

로그인 한 사애에서 받을 수 잇는 서비스에 대하여

정보수정과 비번 변경 둘다 로그인 한 상태에서 해야함 ==> JWT 필요함. JWT를 리퀘스트 헤더에 심는 것은 axios interceptor로 구현했죠?

그 기능을 오늘 자연스럽게 사용할 예정!

우리가 신경써야하는 것은 비번 변경.

기존 비번 맞는지 확인후 encoder의 matches->새로운 비번 암호화 후 업데이트.

★ KB 국민은행

MemberUpdateDTO.java

package org.scoula.member.dto; @Data @NoArgsConstructor @AllArgsConstructor public class MemberUpdateDTO { private String username; private String password; private String email; MultipartFile avatar; public MemberVO toVO() { return MemberVO.builder() .username(username) .email(email)

.build();

하지만 localStorage에서 복원이 됐더라도 JWT가 만료 됐어도 로그인 처리가 된다. 개선해야함. 복원 과정에서 JWT 유효성 검사를 해야함 토큰이 만기가 됐는지 확인해야함.

클라에서 직접 할수도 있고

서버에게 물어볼 수 있다. 토큰 검사 URL을 만들고 이에 맞는 컨트롤러 동작을 정의하고 요청 보내는 형식으로

이건 해당 실습에서는 안할거야 해봐^^

```
member
controller
dto
ChangePasswordDTO
MemberDTO
MemberJoinDTO
MemberUpdateDTO
exception
mapper
service
```

BL에서는 pw 검사해야하고 (DB에서 꺼내고 비교) 일치X시 사용자 예외 정의한거 발생시킬거야

☑ 정보의 수정

- 해당 비밀번호가 맞는 경우에만 수정
- o 비밀번호가 틀리면 PasswordMissMatchException 발생

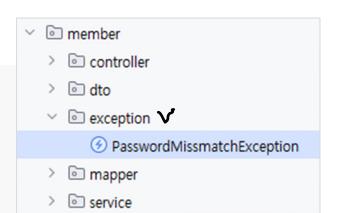
sql쿼리문 where로 password=방금 암호화로 찾을 수 없을까 없다. 매번 암호화된 것은 달라지니까

=말고 matches로 일치하는지 검사해야함

PasswordMissmatchException.java

```
package org.scoula.member.exception;

public class PasswordMissmatchException extends RuntimeException{ 런타임 에러 상속 public PasswordMissmatchException() { super("비밀번호가 일치하지 않습니다."); ✔
}
```



MemberMapper.java

```
public interface MemberMapper {
    ...
    int update(MemberVO member);
}
```

MemberMapper.xml

```
<update id="update">
    UPDATE tbl_member
    SET
    email = #{email},
    update_date = now()
    WHERE username =#{username}
</update>
```

MemberService.java

```
public interface MemberService {
    ...
    MemberDTO update(MemberUpdateDTO member);
}
```

MemberServiceImpl.java

```
@Override
public MemberDTO update(MemberUpdateDTO member) {
    MemberVO vo = mapper.get(member.getUsername()); 디비에서 멤버 정보
    if(!passwordEncoder.matches(member.getPassword(),vo.getPassword())) { // 비밀번호 일치 확인
        throw new PasswordMissmatchException(); 평문, 암호문
    }

mapper.update(member.toVO());
saveAvatar(member.getAvatar(), member.getUsername()); multiPart, username 인자
    return get(member.getUsername());
}
```

의문 : 기능에 비해서 sql문을 너무 많이 호출하는거 아닌가 ==> 성능에 무리가 가지 않을까? 걱정마 mysql은 cache를 운영한다. select한 결과들은 다 캐싱해 놓기 때문에 아이 좋아.

MemberController

```
@PutMapping("/{username}")
public ResponseEntity<MemberDTO> changeProfile(MemberUpdateDTO member) {
   return ResponseEntity.ok(service.update(member));
}
```



2025년 상반기 K-디지털 트레이닝

회원관리-정보 수정(프론트엔드)

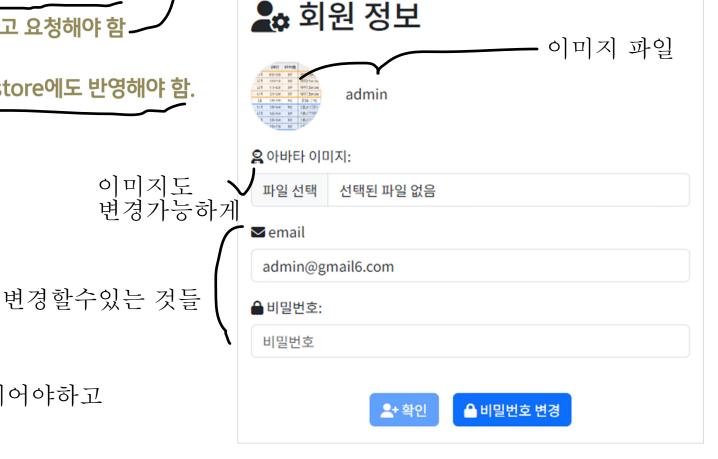
[KB] IT's Your Life



,헤더에 JWT를 추가하는.

회원관리-정보 수정

- ☑ 인터셉터가 적용된 axios를 이용
 - 인증 헤더를 추가하고 요청해야 함.
 - 수정 요청 성공 후 store에도 반영해야 함.



바이너리 데이터도 전송되니까 multiPartencoding요청이어야하고 formdata로

🗹 api/authApi.js

```
import api from '@/api'; 우리가 만든 api. == 인터셉터 적용된 axios
const headers = { 'Content-Type': 'multipart/form-data' };
export default {
 async update(member) {
                                  폼을 객체화한 formData이용
  const formData = new FormData();
  formData.append('username', member.username);
  formData.append('password', member.password);
  formData.append('email', member.email);
  if (member.avatar) {
   formData.append('avatar', member.avatar);
  const { data } = await api.put(`${BASE_URL}/${member.username}`, formData, headers);
  console.log('AUTH PUT: ', data); 경로변수를 이용한 url경로
  return data;
```

백엔드는 영역이 명확하게 나뉘는데 작업이. 프론트엔드는 명확하지 않다 그 구분이 가이드는 있다. 컴포넌트는 출력에만 신경써라. BL은 분리해서 운용해라.

stores/auth.js

```
export const useAuthStore = defineStore('auth', () => {

const changeProfile = (member) => {
    state.value.user.email = member.email;
    localStorage.setItem('auth', JSON.stringify(state.value));
};

load();
return { state, username, email, isLogin, changeProfile, login, logout, getToken };
});
```

```
<script setup>
import authApi from '@/api/authApi';
import { useAuthStore } from '@/stores/auth';
import { computed, reactive, ref } from 'vue';
const auth = useAuthStore();
                                                                 이미지 태그의 src속성에 바인딩될 예정
const avatar = ref(null);
const avatarPath = \alpha / api/member/{auth.username}/avatar;
const member = reactive({
 username: auth.username,
 email: auth.email,
 password: ",
 avatar: null,
});
const error = ref('');
const disableSubmit = computed(() => !member.email || !member.password);
```

```
const onSubmit = async () => {
 if (!confirm('수정하시겠습니까?')) return;
 if (avatar.value.files.length > 0) {
  member.avatar = avatar.value.files[0]; ~
 try {
  await authApi.update(member);
  error.value = ";
  auth.changeProfile(member); V
  alert('정보를 수정하였습니다.');
                                  passwordMissmatchException
 } catch (e) {
  error.value = e.response.data;
</script>
```

```
<div class="mb-3 mt-3">
 <label for="email" class="form-label">
  <i class="fa-solid fa-envelope"></i>
  email
 </label>
 <input type="email" class="form-control" placeholder="Email" id="email" v-model="member.email" />
</div>
<div class="mb-3">
 <label for="password" class="form-label">
  <i class="fa-solid fa-lock"></i>
  비밀번호:
 </label>
 <input type="password" class="form-control" placeholder="비밀번호" id="password" v-model="member.password" />
</div>
<div v-if="error" class="text-danger">{{ error }}</div>
```