API DOCS

홈

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
서비스 전체 누적 메시지 수	황정주		ALL	GET	api/v1/home/total-count	서비스 전체 누적 메시지 수를 반환한다.		{ status: "success", message: "서비스 전체 누적 메시지 수 반환합니다.", data: { totalHeartCount : "서비스 누적 메시지 수" { }		☑		https://www.notion.so/70f84ee4c7b64ab7bfc5cebda6e97637

회원 관리

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
관리자용 일반 로그인	황정주		ALL	POST	api/v1/auth/guests/admin/login	accessToken이 필요한 경우 accessToken 사용하기 위한 api이다. **id는 user table PK**이고 현재 우리 user table에는 aaaa, bbbb, cccc가 있습니다. 만약 더 많은 사용자가 필요하다면 user 테이블에 insert하고 사용하면 될 것 같습니다.	RequestBody { id : "aaaa" }	{ status: "success", message: "테스트용 로그인 성공", data: { userId : "aaaa", nickname : "김싸피", accessToken : "fas22kkdj23isldk22" { }	{ status: "fail", message: "설명", data: { } } 에러코드 404와 에러메시지 반환 404 : 해당 사용자가 DB에 존재하지 않음.	☑	userId : user 고유코드	https://www.notion.so/4aac8f5cb4654633b584b176e3d09f8a
소셜 회원가입 & 로그인	황정주		ALL	GET	api/v1/auth/guests/social/{provider}?code={code}	인가코드를 파라미터로 받아와서 카카오나 구글로부터 user 정보를 받아오고 가공하여 user 정보를 반환한다. provider : kakao나 google이 들어간다. code : 인가코드 **최초 회원가입 시 :** isFirst를 **true**로 반환 **그 외에는 false**로 반환	PathVariable { provider : "kakao" or "google" } RequestParam { code : "4be900cf-d2e7-4d16-abc6-eafc27f4a7e0" }	{ status: "success", message: "소셜 로그인 성공", data: { userId : "550e8400-e29b-41d4-a716-446655440000", nickname : "김싸피", statusMessage : "메롱", accessToken : "fas22kkdj23isldk22", isFirst : true or false } }	{ status: "fail", message: "설명" or "pause" or "out", data: { } } 에러코드 404와 에러메시지 반환 404 : 카카오나 구글로부터 user 정보 받아오지 못했다는 의미 **일시 정지 회원**의 경우 : **pause**를 반환 **영구 정지 회원**의 경우 : **out**을 반환	☑	userId : user 고유 코드	https://www.notion.so/56f6e6bf92b64a06b5d550e805575259d
로그아웃	황정주	Authorization : access token	USER	PATCH	api/v1/auth/users/logout	로그아웃 버튼 눌렀을 시 호출된다.		{ status: "success" message: "로그아웃 성공", data: { } }		☑		https://www.notion.so/873b1ca5e761469d9f45b38bfadbcce5
트위터 redirecURL 받아오기	황정주		ALL	GET	api/v1/auth/guests/twitter/redirect-url	트위터 로그인 호출 순서 1 트위터 로그인을 할 수 있는 url을 반환한다. 반환받은 url로 바로 redirect 해야 한다.		{ status: "success", message: "트위터 로그인 redirect URL 얻어오기 성공!", data: { redirectUrl : "https://api.twitter.com/oauth/authenticate?oauth_token=Bm6OqwAAAAABnZIeAAABiAm" } }		☑		https://www.notion.so/redirecURL-4179864e03d3452290fcd251899c5c09
상태메시지 수정	황정주		USER	PATCH	api/v1/auth/users/status-message	변경된 상태메시지를 DB에 반영한다.	RequestBody { statusMessage : "모두 화이팅~!" }	{ status: "success", message: "상태메시지 변경 성공", data: { statusMessage : "변경된 상태메시지" } }	{ status: "fail", message: "상태메시지 변경을 실패했습니다.", data: { } }	☑		https://www.notion.so/75e255d167f347f68bfe9b95f09a5c5e
트위터 회원가입 & 로그인	황정주		ALL	POST	api/v1/auth/guests/twitter/user-info	트위터 로그인 호출 순서 2 바디에 담긴 oauthToken, oauthVerifier로 트위터로부터 user 정보를 받아오고 가공하여 user 정보를 반환한다. **최초 회원가입 시 :** isFirst를 **true**로 반환 **그 외에는 false**로 반환	RequestBody { oauthToken : "NPcudxy0yU5T3tBzho7i", oauthVerifier : "uw7NjWHT6OJ1MpJOXsHf" }	{ status: "success", message: "트위터 로그인 성공!!", data: { userId : "5qjtalw0vc", nickname : "하팅163", statusMessage : "메롱", accessToken : "fas22kkdj23isldk22", isFirst : true or false } }	{ status: "fail", message: "설명" or "pause" or "out", data: { } } 에러코드 404와 에러메시지 반환 404 : 카카오나 구글로부터 user 정보 받아오지 못했다는 의미 **일시 정지 회원**의 경우 : **pause**를 반환 **영구 정지 회원**의 경우 : **out**을 반환	☑	userId : user 고유 코드	https://www.notion.so/773571edbe4e4413a36e2768b68e44e0
닉네임 수정	황정주	Authorization : access token	USER	PATCH	api/v1/auth/users/nickname	변경된 닉네임을 DB에 반영한다.	RequestBody { nickname : "jjda" }	{ status: "success", message: "닉네임 변경 성공", data: { nickname : "변경된 닉네임" } }	{ status: "fail", message: "닉네임 변경을 실패했습니다.", data: { } }	☑		https://www.notion.so/871000fae9fe4478cbbe75711ad5f061c
하트판 주인 정보 조회	황정주		ALL	GET	api/v1/auth/guests/{userId}	하트판 주인의 정보를 반환한다.	PathVariable { userId : "52516f07-1f4f-494d-9516-2903f1677fbf" }	{ status : "success", message : "하트판 주인 정보 반환합니다", data: { nickname : "jj", statusMessage : "하트 줘", messageTotal : 20 } }	{ status: "fail", message: "존재하지 않는 하트판입니다.", data: { } } 에러코드 404와 에러메시지 반환 404 : 존재하지 않는 하트판	☑	messageTotal : 하트판 주인이 받은 누적 메시지 수	https://www.notion.so/bd8732c85b4b4789acf2d03d5cb1e1c4
access token 재발급	황정주	Authorization : access token	USER	GET	api/v1/auth/users/access-token	access token 재발급 할 때 호출한다.		{ status: "success", message: "access token 재발급 성공", data: { accessToken : "새로 발급한 access token" { }	에러코드 401과 에러메시지 반환 401 : 인증되지 않은 사용자라는 의미	☑		https://www.notion.so/access-token-ff2328df80f048fe9973a1cdc6239a16

하트판

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
특정 메시지 신고하기	임영묵	Authorization : access token	USER	POST	api/v1/messages/{messageId}/reports	특정 메시지를 신고한다. Access token에서 받아온 userId가 해당 메시지의 receiver_id와 동일하지 않으면 401 에러를 발생한다. **메시지의 sender_id가 없으면 (발신자가 비로그인 유저이면)** **isLoggedInUser를 false로 해서 반환한다.** 메시지의 sender_id를 통해 발신인의 report_count를 +1 해주고 3회, 5회 신고횟수에 따라 status를 업데이트 하고, blocked_user 테이블에 해당 유저를 추가한다. 메시지의 isReported를 true로 업데이트 해 준다. 신고 테이블에 해당 건을 추가한다.	PathVariable { messageId: "long" } RequestBody { content, }	{ status: "success", message: "메시지가 성공적으로 신고되었습니다.", **data: {** **isLoggedInUser: true/false** **}** }	에러코드 401와 에러메시지 반환 401 : 권한 없음	☑		https://www.notion.so/490c33b245c34528a33b5557489e6702

받은 메시지 상세보기	임영묵	Authorization : access token	USER	GET	api/v1/messages/received/detail/{messageId}	본인 유저가 받은 메시지의 상세 내용을 볼 수 있다. Access token에서 받아온 userId가 해당 메시지의 receiver_id와 동일하지 않으면 401 에러를 발생한다.	PathVariable { messageId: “long” }	{ status: “success”, message: “특정 메시지의 상세 내용을 조회합니다.”, data: { messageId, title, heartId, heartName, heartUrl, emojiId, emojiName, emojiUrl, createdDate, expiredDate, isRead, isReported, isStored, content, shortDescription, } }	에러코드 401와 에러메시지 반환 401 : 권한 없음	☑	위 24시간 내 받은 메시지 리스트에서 찾을 수도 있지만, 받은유저와 메시지 아이디로 특정 메시지를 반환한다는 것에 의미를 둔다.	https://www.notion.so/6cc014d668fd4a0c83adbe388f648812
메시지 삭제	임영묵	Authorization : access token	USER	DELETE	api/v1/messages/{messageId}	메시지를 삭제한다. DB 상에서는 삭제하지 않고 isActive를 false로 바꿔주기만 한다.	PathVariable { messageId: “long” }	{ status: “success”, message: “메시지가 성공적으로 삭제되었습니다.”, }		☑		92495c49-2f45-4fc7-afcb-976393926bfb
메시지 발송 (웹소켓)	임영묵		USER		ws	위에서 메시지를 성공적으로 발송했다는 백엔드 서버 응답을 확인한 후 웹소켓 요청을 보낸다 (send-message). 웹소켓 요청은 대상 유저의 userId로 보내고, data는 메시지 발송 후 반환된 data 객체를 그대로 다시 보낸다. 이 신호를 받은 클라이언트(receive-message)는 로그인 된 유저의 받은메시지를 관리하는 자료구조에 받은 객체를 추가하면 된다.	emit: ‘send-message’, on: ‘receive-message’ 내용: - 보낼 유저의 userId, - data: { messageId, title, heartId, heartName, heartUrl, emojiId, emojiName, emojiUrl, createdDate, expiredDate, isRead }	data: { messageId, title, heartId, heartName, heartUrl, emojiId, emojiName, emojiUrl, createdDate, expiredDate, isRead }		☑		https://www.notion.so/172a747f97aa4ac7bcfccff3c28e0101
24시간 내 받은 메시지 조회	임영묵	Authorization : access token	USER	GET	api/v1/messages/received/{userId}	본인 유저가 24시간 내에 받은 메시지를 조회한다. Access token에서 받아온 userId가 해당 메시지 URI 경로의 userId와 동일하다면 본인, 아니면 다른 유저로 취급한다. expired_date가 지난 메시지들은 isActive를 false로 바꿔 db에 저장하고 반환할 메시지 리스트에서 제외한다. 반환값의 isRead는 본인일때만 보내준다. ********** **알림구현** 1) 현재 페이지가 로그인된 유저 본인의 페이지라면: - 본 API 한 번만 호출 → 받은 정보로 isRead = false인것만 알림창에 띄우면 된다. 2) 현재 페이지가 로그인된 유저와 다른 사람의 페이지라면: - 본 API **2번 호출** → 한번은 userId에 현재 페이지 주인의 id를 넣고, 한번은 로그인된 유저 본인의 id를 넣어서 호출하면, 2번째 호출에서 알림에 필요한 정보를 받을 수 있다. 3) 현재 유저가 로그인 되어있지 않을 때: - 본 API 한 번만 호출 → 알림 구현 안하고 그냥 화면에 하트판 주인의 하트들을 띄워주면 된다. **********	PathVariable { userId: “string” }	{ status: “success”, message: “유저의 24시간 내 메시지를 조회했습니다.”, data: { messageList: [{ messageId, title, heartId, heartName, heartUrl, emojiId, emojiName, emojiUrl, createdDate, expiredDate, isRead }, …] } }		☑	유효기간 만료된 메시지들을 이 요청을 보낼때 업데이트 시켜주는게 맞나 싶다.	https://www.notion.so/24-2b25d0cedb924c78ac5c3ab03d14f0c6
메시지 발송	임영묵	Authorization : access token	USER	POST	api/v1/messages	메시지를 발송한다. Access token에서 받아온 userId가 해당 메시지의 sender_id와 동일하지 않으면 401 에러를 발생한다. receiverId가 존재하지 않으면 400 에러를 발생한다. heartId가 존재하지 않으면 400 에러를 발생한다. receiverId에 해당하는 유저의 message_total을 +1 해준다. 메시지 테이블에 해당 메시지를 새로 추가한다.	RequestBody { heartId, senderId, receiverId, title, content, }	{ status: “success”, message: “메시지가 성공적으로 발송되었습니다.”, data: { messageId, heartId, heartName, heartUrl, isRead, **isCheckSender** } }	에러코드 401와 에러메시지 반환 401 : 권한 없음 400 : 잘못된 요청	☑	이 요청을 보내고 receiverId 이름의 socket.io 방에 ‘send-message 통신을 emit 해야한다. 그럼 receiverId에 해당하는 유저가 접속해 있을 경우 알림과 함께 메시지 정보를 받는다.	https://www.notion.so/953f97fd9d164d0391122cbf717c83f6
웹소켓 접속 (웹소켓)	임영묵		USER		ws	유저가 로그인 되면 최상위 컴포넌트에서 소켓통신 연결을 한다. 연결과 함께 ‘join-room’ 요청을 ‘emit’해 해당 유저의 userId를 같이 보낸다. 이 userId와 유저의 socket.id를 node.js에서 매핑시켜 향후 메시지를 받을때 쓸 수 있게 한다. 페이지가 새로고침 되더라도 유저가 로그인 되어있는지 확인하고, 되어있으면 바로 소켓통신을 해줘야 한다 (새로고침하면 웹소켓 연결 다시 해줘야함)	emit: ‘join-room’ 내용: - 로그인 된 유저의 userId			☑		https://www.notion.so/ab487ece696e4412bf7091a3e4c1b53c
받은 메시지에 반응하기	임영묵	Authorization : access token	USER	POST	api/v1/messages/{messageId}/emojis/{emojiId}	특정 메시지에 반응을 추가한다. Access token에서 받아온 userId가 해당 메시지의 receiver_id와 동일하지 않으면 401 에러를 발생한다. **emojiId를 -1로 주면 emoji를 해제한다.**	PathVariable { messageId: “long”, emojiId: “long”, }	{ status: “success”, message: “반응이 성공적으로 추가되었습니다.”, data: { emojiUrl, **senderId** } }	에러코드 401와 에러메시지 반환 401 : 권한 없음	☑		https://www.notion.so/dd4b6801834f41a485d6f027a71c092c

영구 보관함

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
영구 보관 메시지 리스트 조회	손민혁	Authorization : access token	USER	GET	api/v1/messages/inbox	회원의 영구 보관된 메시지 리스트를 조회한다.	PathVariable -	{ "status": "success", "message": "영구 보관 메시지 리스트 조회를 성공했습니다.", "data": { "inboxList": [{ "messageId": 33, "title": "someTitle", "messageContent": "someContent", "heartId": 2, "heartUrl": "test.com", "emojiId": "1", "emojiName": **null**, "emojiUrl": **null**, "createdDate": "2023-04-26T01:26:53" }, …] } }	404: 해당 유저 없음	☑	messageId: 메시지 아이디 messageTitle: 메시지 제목 messageContent: 메시지 내용 heartId: 하트 아이디 heartImage: 하트 이미지 URL emojiUd: 이모지 Id emojiName: 반응 이름 emojiUrl: 반응 이미지 URL createdDate: 메시지 받은 날짜	https://www.notion.so/474a1b58700f4463bc6da355697b4a0a

영구 보관 메시지 삭제	손민혁	Authorization : access token	USER	DELETE	api/v1/messages/inbox/{messageId}	회원의 영구 보관된 메시지 하나를 삭제한다	PathVariable messageId: 메시지 아이디	{ status: “success”, message: “영구 보관 메시지 삭제를 성공했습니다.”, data: null }	404: 해당 유저 또는 해당 메시지 없음	☑		https://www.notion.so/580998a3a63f40b59887b6ef836264ae
보낸 메시지 리스트 조회	손민혁	Authorization : access token	USER	GET	api/v1/messages/sent	회원이 보낸 메시지 리스트를 조회한다.	PathVariable -	{ "status": "success", "message": "보낸 메시지 리스트 조회를 성공했습니다.", "data": { "sentMessageList": [{ "messageId": 234, "title": "낭낭", "heartId": 5, "heartName": "애정", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_red_1.svg", "emojiId": null, "emojiName": null, "emojiUrl": null, "createdDate": "2023-05-01T10:46:40", "expiredDate": "2023-05-02T10:46:40" },] } }	404: 해당 유저 없음	☑	보낸 메시지에 대해 상대방의 반응이 없다면 emojiId, emojiName, emojiUrl이 null 값이다.	https://www.notion.so/8cb1eeaa1211427fa24686d6a994f77f
영구 보관 메시지 저장	손민혁	Authorization : access token	USER	POST	api/v1/messages/inbox/{messageId}	회원이 받은 메시지를 영구 보관함에 저장한다.	PathVariable messageId: 메시지 아이디	{ status: “success”, message: “메시지 영구 보관 저장을 성공했습니다.”, data: null }		☑	data는 항상 null이 반환됩니다.	https://www.notion.so/b7129b95f5404f469628a701c140d673
보낸 메시지 상세 조회	손민혁	Authorization : access token	USER	GET	api/v1/messages/sent/{messageId}	회원이 보낸 메시지 하나를 상세 조회한다	PathVariable messageId: 메시지 아이디	{ "status": "success", "message": "보낸 메시지 상세 조회를 성공했습니다.", "data": { "messageId": 241, "title": "ㅎㅎ", “shortDescription” : “나 너 좋아하냐 ?” "content": "ㅎㅎ", "heartId": 1, "heartName": "호감", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_yellow_1.svg", "emojiId": null, "emojiName": null, "emojiUrl": null, "createdDate": "2023-05-01T23:21:17", "expiredDate": "2023-05-02T23:21:17", “isStored” : null } }	404: 해당 유저 또는 해당 메시지 없음 400: messageId가 숫자 타입이 아닐 경우	☑		https://www.notion.so/c4c03e77da0f4f41994b38f2806227db
영구 보관 메시지 상세 조회	손민혁	Authorization : access token	USER	GET	api/v1/messages/inbox/{messageId}	회원의 영구 보관된 메시지 하나를 상세 조회한다	PathVariable messageId: 메시지 아이디	{ "status": "success", "message": "영구 보관 메시지 상세 조회를 성공했습니다.", "data": { "messageId": 4, "heartId": 2, "heartName": "테스트하트", "heartUrl": "test.com", "messageContent": "someContent", "createdDate": "2023-04-25T06:01:59", "emojiUrl": null, "emojiName": null } }	404: 해당 유저 또는 해당 메시지 없음	☑		https://www.notion.so/cf469eb9f4b84bf8a1ae189bf476930b

하트

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
(도감용) 하트 리스트 조회	손민혁		ALL	GET	api/v1/hearts	도감용 모든 하트 리스트를 조회한다.	PathVariable -	{ "status": "success", "message": "도감용 하트 리스트 조회에 성공했습니다.", "data": { "heartList": [{ "heartId": 1, "name": "호감하트", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_yellow_1.svg", "type": "DEFAULT", "isLocked": false, conditons: null }, { "heartId": 2, "name": "응원하트", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_blue_1.svg", "type": "DEFAULT", "isLocked": false },] } }	404: 해당 유저 없음	☑	heartId - 하트 PK name - 하트 이름 heartUrl - 하트 이미지 주소 type - 하트 타입(DEFAULT/SPECIAL) isLocked - 잠금 여부 도감용 하트 리스트 조회 - 로그인 유저: 모든 하트를 볼 수 있다. 단, 내가 획득하지 못한 하트는 잠겨있다. - 비 로그인 유저: 모든 하트를 볼 수 있다. 단, 기본 하트를 제외한 모든 하트는 잠겨있다.	https://www.notion.so/2c482e34e6e243a5859f87eaa9577b00
(메시지용) 하트 리스트 조회	손민혁		ALL	GET	api/v1/hearts/user-hearts	메시지용 모든 하트 리스트를 조회한다.	PathVariable -	"status": "success", "message": "메시지용 하트 리스트 조회에 성공했습니다.", "data": { "heartList": [{ "heartId": 1, "name": "호감하트", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_yellow_1.svg", "type": "DEFAULT", "shortDescription": "짧은 설명", "isLocked": false }, { "heartId": 2, "name": "응원하트", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_blue_1.svg", "type": "DEFAULT", "shortDescription": "짧은 설명", "isLocked": false },] } }		☑	메시지용 하트 리스트 조회 - 로그인 유저: 기본 하트 + 내가 획득한 하트만 볼 수 있다. - 비 로그인 유저: 기본하트 만 볼 수 있다. 단, 설렘 하트와 애정 하트는 잠겨있다.	https://www.notion.so/c267e76991324a71b92df870a63be9bb

도감 하트 상세 조회	손민혁		ALL	GET	api/v1/hearts/{heartId}	상세 하트 정보를 조회한다.	PathVariable heartId	{ "status": "success", "message": "도감 하트 상세 조회에 성공했습니다.", "data": { "heartId": 7, "name": "무지개", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_rainbow_1.svg", "shortDescription": "내 모든 마음을 담아서!", "longDescription": "상대방을 위한 여러 마음들이 모여 탄생한 무지개 하트. \n하트를 받은 상대의 마음도 찬란한 무지개빛으로 물들인다.", "type": "SPECIAL", "acqCondition": "모든 기본 하트 전송", "isLocked": true, "conditions": [{ "heartId": 1, "name": "호감", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_yellow_1.svg", "currentValue": 1, "maxValue": 1 }, { "heartId": 2, "name": "응원", "heartUrl": "https://heart-ing.s3.ap-northeast-2.amazonaws.com/heart/heart_blue_1.svg", "currentValue": 1, "maxValue": 1 },] } }		☑	비로그인 유저의 경우 conditions 가 null 값이 날라간다.	https://www.notion.so/76d21d0013b24f8ea8bb158289b6e46c
하트 획득	손민혁		USER	POST	api/v1/hearts/user-hearts/{heartId}	스페셜 하트를 획득합니다.	PathVariable heartId	{ "status": "success", "message": "하트 획득에 성공했습니다.", "data": null }		☑		https://www.notion.so/988a20fbbbb0464a8ea90d52f995d9ec

데이터 마이그레이션

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
유저 보낸 하트 수	손민혁		ADMIN	GET	/api/v1/migration/userSentHeart	mysql to redis로 유저가 보낸 하트 수를 가져온다.	-	{ "status": "success", "message": "Redis 유저 보낸 하트 정보를 MySQL과 동기화에 성공했습니다.", "data": null }		☑		https://www.notion.so/3de4181d8ff541f3b621e3b723a4ce6c
하트 정보	손민혁		ADMIN	GET	/api/v1/migration/heartInfo	mysql to redis로 하트 정보를 가져온다.	-	{ "status": "success", "message": "Redis 하트 정보를 MySQL과 동기화에 성공했습니다.", "data": null }		☑		https://www.notion.so/c40ee97360d04272a35f666aa07903ba

알림

API 이름	담당자	Header	권한	Method	URI	API 기능 설명	매개변수	성공 반환값/의미	에러 반환값/의미	구현	비고	Page URL
유저의 알림 갖고오기	임영묵	Authorization : access token	USER	GET	/api/v1/notifications	유저의 24시간 내 알림을 가져온다. 알림은 24시간을 기준으로 시간이 지났을 시 비활성화 처리하고 갖고오지 않는다. **알림 종류(R: 받은 하트 E: 보낸 하트 H: 도감)**		{ "status": "success", "message": "유저의 알림 조회에 성공하였습니다.", data: { notificationList: [{ notificationId, userId, content, createdDate, expiredDate, type, isChecked, isActive notificationId, heartName, heartUrl, messageId, createdDate, type, isChecked, }, …] } }		☑		https://www.notion.so/3f1cac13dc0b465cae27183d24ce1431
알림 읽기	임영묵	Authorization : access token	USER	POST	/api/v1/notifications/{notificationId}	해당 notificationId를 읽음 처리 한다	PathVariable { notificationId: “number” }	{ status: “success”, message: “해당 알림이 성공적으로 읽음처리 되었습니다.”, }		☑		https://www.notion.so/f7161fb89a9d43ae9032ad607e03b2a5