Protocol22-API

- □□
- Protocol22-API □□
- Protocol22-API □□ □□ □□
- [JavaScript] Protocal22-API □□□□
- [WebApp] Protocal22-API □□□□ (PHP)
- [WebApp] Protocal22-API □□□□ (GoLang)
- OwnerKey□ □□□ □□□ □□ □□

개요

“

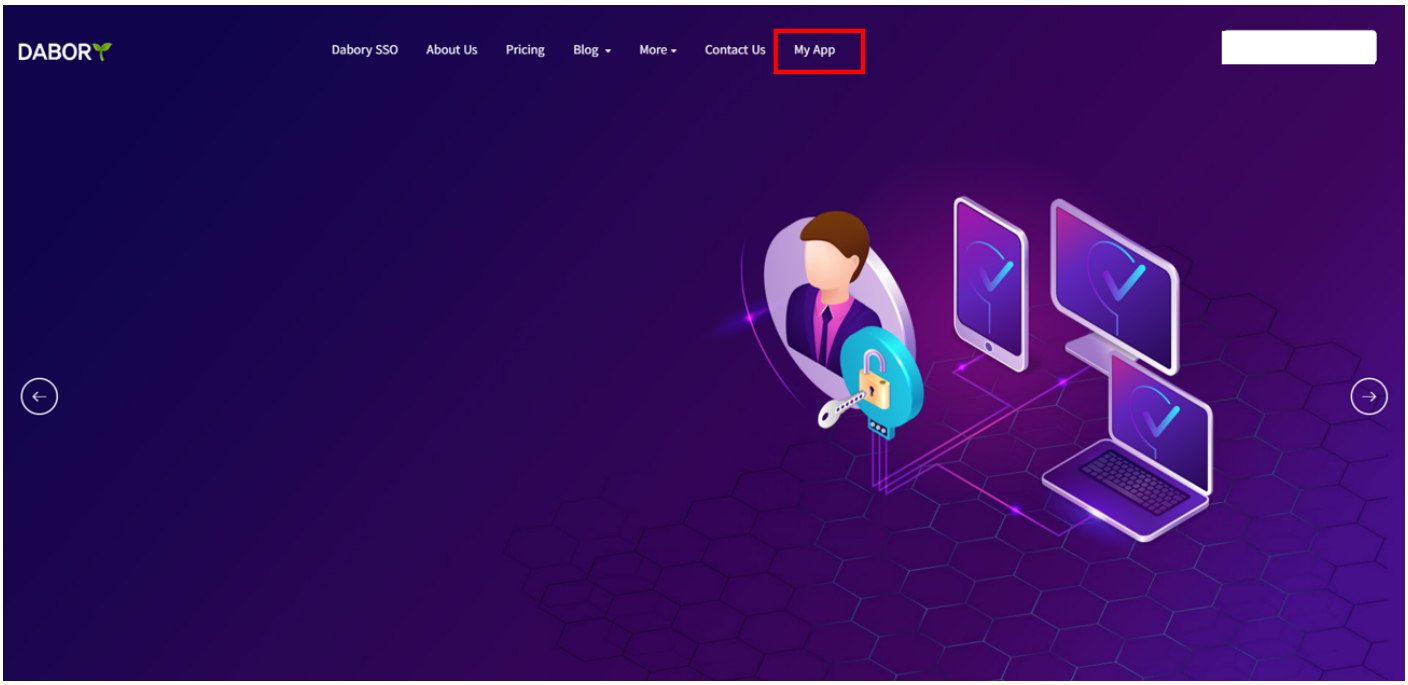
PROTOCOL22-API

PROTOCL22는 다보리SSO에서 OwnerKey를 활용하여 MAIN APP과 GUEST APP간의 API를 공유하여 특정 API에 접근하는 방식입니다. OwnerKey란?
PROTOCAL22-API를 사용하기 위해서는 OwnerKey 발급이 필요합니다.
여기서 OwnerKey는 APP 등록시 main APP에 등록된 API 서버와 DB 접근 권한이 설정된 암호화 KEY입니다.
발급된 OwnerKey를 guest APP에 저장하여 PROTOCL22-API를 사용할 수 있습니다.

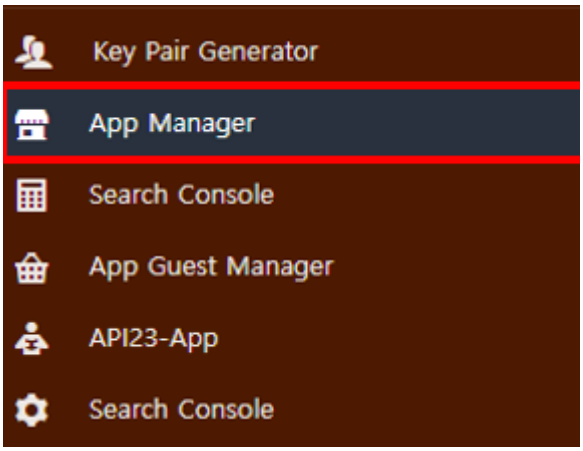
“

OwnerKey 발급방법

1. main app을 등록한 dabory 계정으로 로그인합니다.
2. guest app 등록을 위해서 My App 메뉴를 클릭합니다.



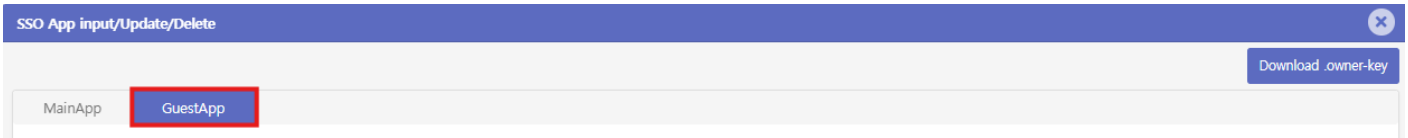
3. □□ □□□□ App Manager□ □□□□□.



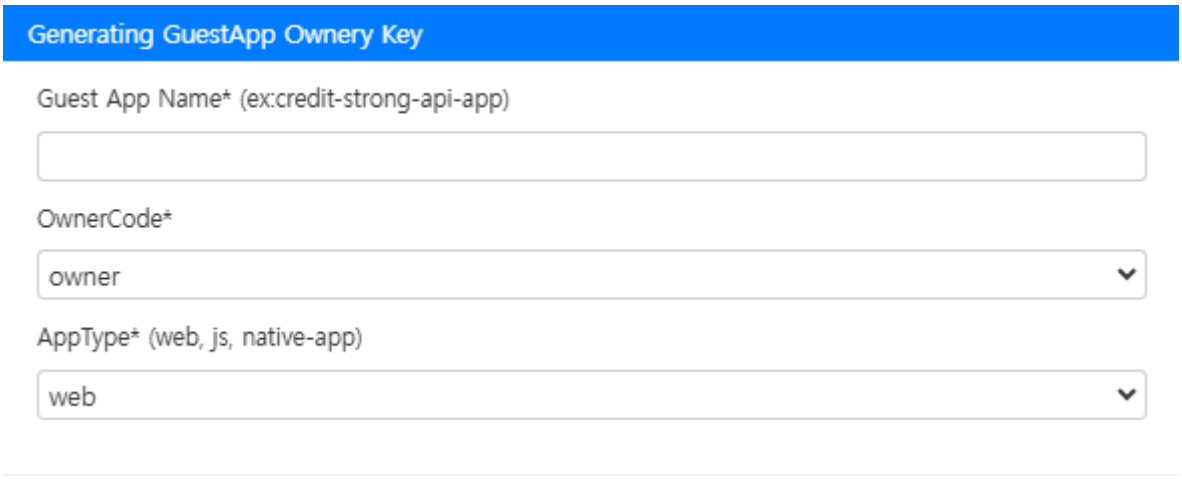
4. □□□ app□ Owner key□ □□□ app□ □□□□□.



5. GuestApp □□ □□□□□.



6. a□□ c□□ □□ □ □□□□□.



□ Guest App Name : Owner Key□ □□

□ OwnerCode : Owner Key□ □□□□

- owner : □□ api □□
- keypair :
- dbupdate :
- limited : □□ api□ □□

□ AppType : guest app□ type

7. Download .owner-key □□□ □□□□ Owner key□ download □□□.

8. MainApp □□ □□□ save □□□□□ □□□□□.

9. □□□ .owner-key□ □□□□ frontend □□□□□ open□□□.

10. .env □□□ MAIN_API_OWNER_KEY □□□ □□□□ □□□ owner key□ □□ □□□□□.

> “
>
> # Protocol22-API □□

1. Javascript (js □□)

2. GO (webapp □□)

3. Laravel (webapp □□)

js방식과 app방식의 차이점

구분	**Javascript**	**WebApp**
호출	프론트엔드(브라우저)	백엔드(Laravel, Go)
목적	Guest APP에서 직접 API 요청	서버간 api 요청 (Backend to Backend)
호출방식	$.fn.ownerKeyLinker.ownerApp() 호출	callOwnerApi() 호출
OwnerKey 저장	스크립트에 직접입력	특정 변수에 저장후 session 활용
인증방식	OnwerKey를 직접 포함하여 API 요청	GateToken을 세션에서 불러와 API 요청
보안	상대적으로 낮음 (클라이언트에서 호출)	상대적으로 높음(서버에서 요청)
사용예제	<script> 태그로 js 라이브러리 로드후 호출	Laravel, Go 등의 서버 코드에서 API 호출

Protocol22-API □□

Protocol22-API□ □□□□ □□□□ □□□□ □ □□ API □□□□ □□□□□□. □ □□ URL□ □□□□, □□ □□, □□□□ □□ □□ □□□□ □□□□ □□□□□, □ □□□□ □□ □ □□□□ □

□□ URL	□□□	□□ □□	□□
/printing-curr-price	post	json	
/printing-curr-price	post	json	
/projects	get	json	
/projects	post	json	
/projects	post	json	
/projects/{id}	get	json	
/projects	delete	json	
/after-base64	post	json	
/send-images	post	json	
/text-send	post	json	
/gate-token-get	post	json	
/call-owner-api	post	json	
/owner-token-get	post	json	
/auth/send-emailConfirm	post	json	
/auth/send-smsConfirm	post	json	
/auth/signup	post	json	

/item-url-scrap	post	json	
/gate-token-get-test	post	json	
/call-api	post	json	
/send/text	post	json	
/send-mail	post	json	
/test-send-mail	post	json	
/elasticsearch	post	json	
/elasticsearch-dsl	post	json	

Protocol22-API □□ □□ □□

Javascript

1. ownerApp

JavaScript□□ API□ □□□ □□ ownerApp() □□□ □□□ □ □□□□. ajax□ □□□□ API □□□ □□ □ □□□□□.

```
$.fn.ownerKeyLinker.ownerApp = function (url, request, callback, async = true) {
    getParameter();

    $.ajax({
        // ${$.fn.ownerKeyLinker.serverUrl} : script □□□□ url□ □□□ serverUrl
        url: `${$.fn.ownerKeyLinker.serverUrl}/dabory-app/${url}`, //ex. https://www.daborysso.com/dabory-app/auth/signup
        data: JSON.stringify(request),
        beforeSend: function (xhr) {
            xhr.setRequestHeader('OwnerKey', $.fn.ownerKeyLinker.ownerKey);
        },
        method: 'POST',
        dataType: 'json',
        async: async,
    })
    .done(function (json) {
        callback(json);
    })
    .fail(function (json, textStatus, errorThrown) {
        callback(json);
        iziToast.error({ title: 'Error', message: 'Dabory Owner App Error' });
    });
};
```

url : main app□ □□□/dabory-app/api url (ex. www.daborysso.com/dabory-app/auth/signup)

request : □□□ □□□ □□□□□ (JSON□□)

callback : □□□ □□□ □□□□

async : □□□ □□/□□□ □□ (□□□ true -> □□□ □□)

beforeSend: □□ □□□ setRequestHeader□ □□□ OwnerKey□ header□ □□.

2. ownerApi

JavaScript□□ API□ □□□ □□ ownerApi() □□□ □□□ □ □□□□. ajax□ □□□□ API □□□ □□ □ □□□□□.

```
$.fn.ownerKeyLinker.ownerApi = function (url, request, callback, async = true) {
    getParameter(); // $.fn.ownerKeyLinker.serverUrl□ $.fn.ownerKeyLinker.ownerKey□ □□□□ □□

    $.ajax({
        url: `${$.fn.ownerKeyLinker.serverUrl}/dabory-app/call-owner-api`,
        data: JSON.stringify(request),
        beforeSend: function (xhr) {
            xhr.setRequestHeader('OwnerKey', $.fn.ownerKeyLinker.ownerKey);
            xhr.setRequestHeader('Url', url);
        },
        method: 'POST',
        dataType: 'json',
        async: async,
    })
    .done(function (json) {
        callback(json);
    })
    .fail(function (json, textStatus, errorThrown) {
        callback(json);
        iziToast.error({ title: 'Error', message: 'Dabory Owner API Error' });
    });
};
```

url : main app□ □□□/dabory-app/call-owner-api (ex. www.daborysso.com/dabory-app/call-owner-api)

request : □□□ □□□ □□□□□ (JSON□□)

callback : □□□ □□□ □□□□

async : □□□ □□/□□□ □□ (□□□ true -> □□□ □□)

beforeSend: □□ □□□ setRequestHeader□ □□□ OwnerKey□ url□ header□ □□□□.

OwnerApi vs OwnerApp

두 함수는 모두 다보리 API와 통신하는 역할을 하지만 요청을 처리하는 방식과 목적이 다릅니다.

두 함수에서 호[illegible] 목록은

OwnerApp :

함수의 이름에서 알 수 있듯이 Js를 [illegible] 증코드나 이메일 인증코드 발송 API를 호출하여 게스트 앱에서 해당 인증코드를 활용할 때 사용할 수 있습니다.

OwnerApi :

게스트앱에서 메인앱의 API를 호출하여 필요한 데이터를 반환받을 때 활용할 수 있습니다. 예를들면 메인앱에 저장된 비밀번호 정책이나 가입한 회원 목록, 메인앱으로 회원가입 하기, 메인앱에서 회원 중

[JavaScript] Protocal22-API 사

시작하기 전에

이 문서는 Dabory Protocol22-API for JavaScript를 이용해 **Main App**과 **Guest App**을 개발할 때 **JavaScript**를 활용한 클라이언트 측 **API** 호출 방식

예제실행전에 아래 준비사항들을 반드시 준비해주시기 바랍니다.

- ☐ script 다운로드
- ☐ app 등록하기

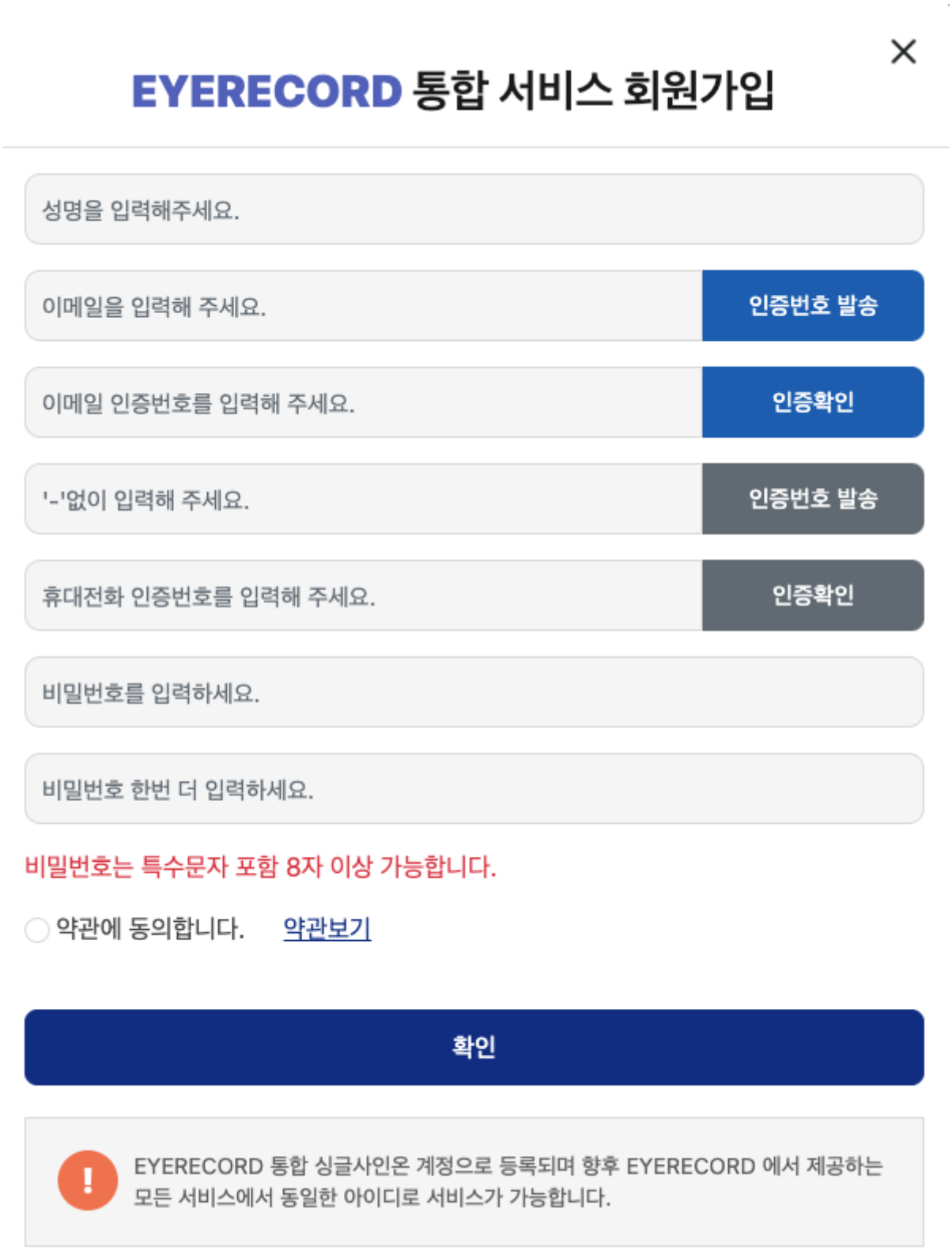
JavaScript 방식의 **API** 호출 흐름

1. 클라이언트 측 JavaScript에 **Owner Key** 및 **MAIN APP**의 **URL** 정보를 설정

2. JavaScript에서 ownerApp() 함수를 사용하여 API 요청을 보냅니다.

3. API 응답을 받아 클라이언트 측에서 데이터를 처리합니다.

> “
>
> # JavaScript 사용예제 (회원가입 구현)



script □□□ □□□□

javascript □□□□□ script src □□□ □□□□ url□ ServerUrl□ OwnerKey□ □□ □□□□□.

```
<script src="/js/dabory.owner-key-
linker.js?serverUrl=https://www.daborysso.com&ownerKey=zEUhgZEpPbFpz/cIghtYCsoFlNi8JX3Lh45ank1s+w2P
```

```
GuK4phOxmemgKuuIJ/OQZlrpdSzlK03MpBLBSo1o537brbDPKaq9ydK8QK8JgEAYGKSNzD8W6dJhSIjPfxLJu+cma0w
cAMDmSV7DMhwwgiwVb2ADWu+QOVKpU+VmIR1TBIfT00EPpuW1Ju+jop8WcBEJRrpEAGLK5jcRnAGSXaDTumqEV
sLRSq87hVqRJGImIE7Bf68Ix4b2PqgVhTUY1Bpalw==""></script>
```

<script src="/js/dabory.owner-key-linker.js?serverUrl=&ownerKey=""></script>

serverUrl : MAIN APP□ □□□□□□ □□□□□.

ownerKey : MAIN_APP □□□ □□□□ OwnerKey□ □□□□□.

API □□ □□(OwnerApp)

JavaScript□□ API□ □□□ □□ ownerApp() □□□ □□□ □ □□□□. ajax□ □□□□ API □□□ □□ □ □□□□□.

```
$.fn.ownerKeyLinker.ownerApp = function (url, request, callback, async = true) {
    getParameter();

    $.ajax({
        // ${$.fn.ownerKeyLinker.serverUrl} : script □□□□ url□ □□□ serverUrl
        url: `${$.fn.ownerKeyLinker.serverUrl}/dabory-app/${url}`, //ex. https://www.daborysso.com/dabory-
app/auth/signup
        data: JSON.stringify(request),
        beforeSend: function (xhr) {
            xhr.setRequestHeader('OwnerKey', $.fn.ownerKeyLinker.ownerKey);
        },
        method: 'POST',
        dataType: 'json',
        async: async,
    })
    .done(function (json) {
        callback(json);
    })
    .fail(function (json, textStatus, errorThrown) {
        callback(json);
        iziToast.error({ title: 'Error', message: 'Dabory Owner App Error' });
    });
};
```

url : API □□□ □□□□□(ex. www.daborysso.com/dabory-app/auth/signup)

request : □□□ □□□ □□□□□ (JSON□□)

callback : □□□ □□□ □□□□

async : □□□ □□/□□□ □□ (□□□ true -> □□□ □□)

beforeSend: □□ □□□ setRequestHeader□ □□ OwnerKey□ header□ □□□□.

API □□ □□(OwnerApi)

JavaScript□□ API□ □□□ □□ ownerApi() □□□ □□□ □ □□□□. ajax□ □□□□ API □□□ □□ □ □□□□□.

```
$.fn.ownerKeyLinker.ownerApi = function (url, request, callback, async = true) {
    getParameter(); // $.fn.ownerKeyLinker.serverUrl□ $.fn.ownerKeyLinker.ownerKey□ □□□□ □□

    $.ajax({
        url: `${$.fn.ownerKeyLinker.serverUrl}/dabory-app/call-owner-api`,
        data: JSON.stringify(request),
        beforeSend: function (xhr) {
            xhr.setRequestHeader('OwnerKey', $.fn.ownerKeyLinker.ownerKey);
            xhr.setRequestHeader('Url', url);
        },
        method: 'POST',
        dataType: 'json',
        async: async,
    })
    .done(function (json) {
        callback(json);
    })
    .fail(function (json, textStatus, errorThrown) {
        callback(json);
        iziToast.error({ title: 'Error', message: 'Dabory Owner API Error' });
    });
};
```

url : API □□□ □□□□□(ex. www.daborysso.com/dabory-app/call-owner-api)

request : □□□ □□□ □□□□□ (JSON□□)

callback : □□□ □□□ □□□□

async : □□□ □□/□□□ □□ (□□□ true -> □□□ □□)

beforeSend: 요청 이전에 setRequestHeader를 통해 OwnerKey를 header에 넣습니다.

OwnerApi 함수는 OwnerApp 함수와는 다르게 api 서버에 요청할 api url을 header에 넣습니다.

OwnerApi vs OwnerApp

두 함수는 모두 다보리 API와 통신하는 역할을 하지만 요청을 처리하는 방식과 목적이 다릅니다.

두 함수에서 사용할 수 있는 API 목록은

	OwnerApp()	OwnerApi()
차이점	게스트앱에서 메인앱에 구현된 백엔드 API를 호출할때 (메인앱의 백엔드에 로직 구현 필요)	게스트앱에서 메인앱의 백엔드 API를 거쳐 api 서버에 구현된 api를 호
호출 흐름	guest (frontend)-> main (backend) -> guest(frontend)	guest (frontend)-> main (backend) -> api 서버 (backend) -> guest(frontend)

ownerApi 호출 예제

- API를 호출하여 비밀번호 정책 (passwd-policy) 데이터를 가져오는 예제입니다.

```
$.fn.ownerKeyLinker.ownerApi(
    'remote-signup-setup-get',
    {
        'SetupCode': 'passwd-policy',
        'BrandCode': 'member'
    },
    function (response) {
        window.passwdPolicy = response;
        $('#policy-desc-em').text(window.passwdPolicy['PolicyDesc']);
        console.log(window.passwdPolicy);
        $('#join').modal('show');
    }
);
```

첫번째 인자값 : **api** 서버의 **url**

두번째 인자값 **: api** 서버 요청시 **body**에 포함될 **request** 객체

□□ □□ : response□ □□□□ □□ callback □□

ownerApp □□ □□

- API□ □□□□ □□ □□□□ (auth/send-smsConfirm)□ □□□□ □□□□□.

```
$.fn.ownerKeyLinker.ownerApp('auth/send-smsConfirm', {
    "MobileNo": $('#mobile-no-txt').val(),
}, function (response) {
    if (response['error']) {
        $($this).prop('disabled', false);
        SmsNumber = null;
        iziToast.error({ title: 'Error', message: response['message'] });
    } else {
        SmsNumber = response['data'];
        iziToast.success({ title: 'Success', message: '□□□□ □□ □□' });
    }
});
```

- API□ □□□□ □□ □□□□ (auth/send-emailConfirm)□ □□□□ □□□□□.

```
$.fn.ownerKeyLinker.ownerApp('auth/send-emailConfirm', {
    "Email": $('#email-txt').val(),
}, function (response) {
    if (response['error']) {
        $($this).prop('disabled', false);
        EmailNumber = null;
        iziToast.error({ title: 'Error', message: '□□□□ □□ □□' });
    } else {
        EmailNumber = response['data'];
        iziToast.success({ title: 'Success', message: '□□□□ □□ □□' });
    }
});
```

API□ □□□□ □□□□□ □□□□ □□□□□.

```
$.fn.ownerKeyLinker.ownerApp('auth/signup', {
    "Email": $('#email-txt').val(),
    "Password": $('#password-txt').val(),
    "MobileNo": $('#mobile-no-txt').val(),
    "FirstName": $('#first-name-txt').val(),
}, function (response) {
    if (response['error']) {
        if (response['code'] === 701) {
            iziToast.info({
                title: 'Info',
                message: '□□ □□□□ □□ □□□ □□ □□□□□ □□□□□□□. □□□ □□□ □□ □□ □□□□ □□□.'
            });
        } else if (response['validator']) {
            showErrorMessages(response['message']);
        } else {
            iziToast.info({ title: 'Info', message: response['message'] });
        }
        click_submit_btn(self, false);
    } else {
        iziToast.success({ title: 'Success', message: $('#action-completed').text() });
        $('#join').modal('hide');
    }
});
```

□□□ □□□ : main app□ □□□ api□ url□ □□□□□.

□□□ □□□ : main app□ □□□ api url□□□.

□□□ □□□ : response□ □□□□ □□□ callback □□□□□.

response['error'] : api□ response status□ 200□ □□ □□ respons['error']□ □□□□□.

[WebApp] Protocal22-API 사용예 (PHP)

> “
>
> # [WebApp] PROTOCOL22-API 사용

시작하기 전에

이 **Dabory OwnerKey API for WebApp**은 **Main App**과 **Guest App**을 개발할 때 필요한 참고 정보를 제공하며 **PHP Laravel**을 활용한 **WebApp** 방식 **PROTOCOL22-API**를 호출하는 방법을 설명합니다.

WebApp 방식의 Owner Key API 호출 흐름

1. .env 파일에 환경변수를 저장합니다.

2. Laravel callOwnerApi() 함수를 통해 Owner APP에 등록된 API 서버로 요청을 보냅니다.

3. 세션에 저장된 GateToken과 frontUrl 정보를 활용하여 API 요청을 보냅니다.

4. API 응답을 받아 데이터를 처리합니다.

WebApp 사용예제

- env 파일에 환경변수 저장

먼저 .env 파일에 OwnerKey 및 APP 정보를 설정해야합니다.

변수명은 다운로드 받은 그대로 동일하게 작성해야 하지만 설정할 MAIN APP이 2개 이상이라면

개수의 맞게 변수 끝에 _ 숫자를 입력하여 변수명을 네이밍합니다.

```
# [protocol22]
OWNER_APP_NAME=YOUR_OWNER_APP_NAME  # Owner APP의 이름 (대소문자 구분)
GUEST_APP_OWNER_KEY=YOUR_OWNER_KEY  # Guest APP에서 사용하는 OwnerKey 값
GUEST_APP_OWNER_URL=YOUR_OWNER_URL  # Owner APP의 URL 주소 (ex.www.daborysso.com)

OWNER_APP_NAME2=YOUR_OWNER_APP_NAME_2
GUEST_APP_OWNER_KEY2=YOUR_OWNER_KEY_2
GUEST_APP_OWNER_URL2=YOUR_OWNER_URL_2

OWNER_APP_NAME3=YOUR_OWNER_APP_NAME_3
GUEST_APP_OWNER_KEY3=YOUR_OWNER_KEY_3
GUEST_APP_OWNER_URL3=YOUR_OWNER_URL_3
```

OwnerKey?

- OwnerKey는 **GUSET APP과 MAIN APP의 API**
- WebApp

- Laravel Owner API

callOwnerApi()

```
public function callOwnerApi($request)
    {
        // 전달된 'type'에서 숫자 추출 (guest[1] → 1)
        preg_match('/\d+/', $request['type'], $matches);
        $index = isset($matches[0]) ? (int) $matches[0] : null;

        // 세션에서 API URL 및 GateToken 가져오기
        if ($index !== null) {
            $apiUrl = session("OwnerGateToken.guest")[$index]['frontUrl'] ?? null;
            $gateToken = session("OwnerGateToken.guest")[$index]['gateToken'] ?? null;
        }

        // API 요청 URL 설정
        $request['url'] = $apiUrl . '/' .$request['url'];
        $request['headers'] = ['GateToken' => $gateToken];
        return $this->callApi($request);
    }
```

- API □□ □□

API□ □□□□ remote-signup-setup-get api□ □□□□ □□□□ □□□□□.

```
$officeInfo = $this->callApiService->callOwnerApi([
    'url'  => 'remote-signup-setup-get', // api □□□ □□□ api url
    'data' => [  // □□□ data
        'SetupCode' => 'passwd-policy',
        'BrandCode' => 'member',
    ],
    'type' => 'guest[1]' // env□ □□□□ owner app□ □□□ session□ □□□□ □□□ index□ □□
]);
```

□□ Dabory Protocol22-API for WebApp□□□□□□□**API**□□□□.

[WebApp] Protocal22-API 사용예[(GoLang)

> “
>
> ## [WebApp] Protocal22-API 사용예제2

시작하기 전에

이 문서는 Dabory OwnerKey API for WebApp을 사용해 main app과 guest app 개발시 필요한 참고 정보를 안내합니다.

WebApp 사용예제

2. WebApp 사용 예제

□□ GateToken 발급 스케줄러

```
package main
func main() {
□// 새로운 고루틴(Goroutine) 실행: 백그라운드에서 주기적으로 실행됨
□go func() {
□□for {
□□□// 1. UpdateGateTokenIfNeeded() 함수 호출 (GateToken 발급 및 업데이트)
□□□err := controllers_func.UpdateGateTokenIfNeeded()
```

```
			
			// 2. 만약 에러가 발생하면 로그를 남김
			if err != nil {
				e.ErrLog("UpdateGateTokenIfNeeded Error", err)
			}

			// 3. 2초 동안 대기 후 반복 (즉, 2초마다 실행됨)
			time.Sleep(2 * time.Second)
		}
	}()

	// 주의: main 함수가 종료되면 고루틴도 종료됨
	select {} // main 함수가 종료되지 않도록 무한 대기
}
```

"OwnerKey"를 활용한 GateToken, BackUrl 반환

Host의 dabory-app/gate-token-get을 호출 -> Host -> main_api (GateToken) 반환

```

// GateToken을 요청하는 함수
func UpdateGateTokenIfNeeded() error {
	myOwnerKey := "My_OwnerKey" //OwnerKey (GateToken 발급 주체 선정을 위해 사용)

	// GateToken을 요청 (OwnerKey를 사용하여 요청)
	err := locals.GuestGateTokenGets(myOwnerKey)
	if err != nil {
		fmt.Println("GateToken 요청 실패:", err)
		return err
	}

	return nil
}



type AppApi struct {
	ApiUrl    string
	GateToken string
}
```

```
var GAppApis [2]AppApi // GateToken, ApiUrl □□ □□

func GuestGateTokenGets(myOnwerKey string) error {
□appType := 0 //Dbupdate
□// 0:keypair, 1:Dbu  // "SsoConnString" Must be Deprecated

□var err error

□if myOnwerKey != "" {
□□// e.LogStr("lskdjqfals", e.LogFuncName()+"; GuestGateTokenOwner with OwnerKey: "+myOnwerKey)
□□_, _, err = GuestGateTokenOwner(appType, "https://host_domain_URL.com", myOnwerKey) // app type ->
OwnerCode
    // Host Domain URL, OwnerKey□ □□□□□ □□
□□if err != nil {
□□□return e.ErrLog(e.FuncRun("23rfsr3qrase", e.CurrFuncName()), err)
□□}
□}

□return nil
}

func GuestGateTokenOwner(appType int, pivotUrl string, ownerKey string) (string, string, error) { // HOST
URL/dabory-app ->  pivot URL
□var appTypeCode string
□if appType == 0 {
□□appTypeCode = "keypair"
□} else if appType == 1 {
□□appTypeCode = "dbupdate"
□}

□if GAppApis[appType].GateToken != "" { // □□□□ □□ □□ (□□ GateToken□ □□ □□)
□□// fmt.Println("34092ujfa : "+"Using GateToken in ARRAY to access : ", GAppApis[0].GateToken)
□} else { // (GateToken□ □□ □□ □□ □□)
□□fmt.Println("34092ujfa : " + "isn't exist GateToken in ARRAY to access ")

□□vGt := &GateTokenGetReq{
□□□OwnerKey: ownerKey,
□□}
□□bodyBytes, _ := json.Marshal(vGt)
```

□□frontUrl := pivotUrl + "/dabory-app/gate-token-get"
□□msgBytes, staInt, err := HttpResponseSimplePost("POST", frontUrl, bodyBytes)
// Strong -> Host Lalavel□ Http □□ -> Lalavel -> main_api□ GateToken □□
□□e.LogStr("23rfsr3qr-GateTokenGetReq", e.LogFuncName()+"; Raravel frontUrl : "+frontUrl)
□□if err != nil {
□□□return "", "", e.ErrLog(e.FuncRun("45425fd34sd-The HTTP request "+frontUrl, e.CurrFuncName()), err)
□□}

□□if staInt != 200 {
□□□TrackFailure() // □□ □□ □ □□ □□ □□
□□□fmt.Println("GateToken □□ □□:", err, "(□□ 1□□ □ □□ □□:", len(failTimestamps), ")", "□□ Code : ", staInt)

□□□// □□ 1□□ □□ □□ □□□ maxFailures(3) □□□□ □□□□ □□
□□□if len(failTimestamps) >= maxFailures {
□□□□fmt.Println("1□□ □ GateToken □□□", maxFailures, "□ □□ -> □□□□□ □□")
□□□□os.Exit(1) // □□□□ □□
□□□}

□□□return "", "", errors.New(e.FuncRun("87ty344ra3-Request Fail "+string(msgBytes), e.CurrFuncName()))
□□}

□□failTimestamps = []int64{} // □□ □ □□ □□ □□□

□□ret := &struct {
□□□ApiUrl   string
□□□GateToken string
□□}{}

□□if err := json.Unmarshal(msgBytes, ret); err != nil { // GateToken□ BackendUrl□ ret □□□□ □□
□□□return "", "", e.ErrLog(e.FuncRun("45425fd34sd-Json Format "+frontUrl, e.CurrFuncName()), err)
□□}
// □□ □□ GateToken□ □□ □□□ □□
□□GAppApis[appType].ApiUrl = ret.ApiUrl
□□GAppApis[appType].GateToken = ret.GateToken
□□e.OkLog("Just Added GateToken in ARRAY to access AppType: " + appTypeCode)
□□e.OkLog("Just Added GateToken in ARRAY to access AppType: " + ret.GateToken)
□□e.OkLog("Just Added GateToken in ARRAY to access AppType: " + ret.ApiUrl) // main_api URL

□}

```
□return GAppApis[appType].ApiUrl, GAppApis[appType].GateToken, nil
}
```

□□ □□□□ GateToken□ □□□□ main_api □□

```
// □□ API □□□ □□□□ □□
func process() {
□// 1. GateToken□ API URL □□□□
□apiUrl := locals.GAppApis[0].ApiUrl    // API □□□ □□ □□ URL
□gateToken := locals.GAppApis[0].GateToken // API □□ □ □□□ □□ □□

□// 2. □□□ API □□□□□ □□ (credit-page □□)
□mainApiURL := apiUrl + "/credit-page"

□// 3. □□ □□ □□ (□□□ □□□)
□requestBody := map[string]interface{}{
□□"PageVars": map[string]interface{}{
□□□"Fields":     "credit_no",
□□□"Limit":      1,
□□□"Desc":       "id",
□□□"MyFilter":   "",
□□□"QueryCnt":   0,
□□□"Query":      "",
□□□"Asc":        "",
□□□"Offset":     0,
□□□"ReturnJson": "",
□□},
□}

□// 4. API □□ □□ (GateToken □□)
□Body, err := RequestWithGateToken(mainApiURL, gateToken, requestBody)
□if err != nil {
□□// □□ □□ □ □□ □□ □□
□□e.ErrLog("API □□ □□: ", err)
□□return
□}
```

```
□// 5. □□ □□□ □□
□fmt.Println("API □□ □□□:", string(Body))
}
```

□□ □□□ GateToken□□ □□ (□□□ GateToken □ □□)

```
// GateToken□ □□□□□ API □□□ □□□ □□
func RequestWithGateToken(mainApiURL string, gateToken string, requestBody map[string]interface{}) ([]byte,
error) {
□// 1. HTTP □□ □□□ (□□ GateToken □□)
□resp, body, err := SendHttpRequest(mainApiURL, gateToken, requestBody)
□if err != nil {
□□fmt.Println("HTTP □□ □□:", err)
□□return nil, err
□}
□defer resp.Body.Close()

□fmt.Println("□□ □□:", string(body)) // API □□ □□

□// 2. □□ □□□ 505□□ -> GateToken □□ (□□□ GateToken □□ □□)
□if resp.StatusCode == 505 {
□□fmt.Println("GateToken □□! □□□ GateToken □□ □□..")

□□// □□ □□□ GateToken □□□
□□locals.GAppApis[0] = locals.AppApi{}

□□// □□□□□ □□□ GateToken□ □□□ □□□ □□ (□□ 5□□□ □□)
□□for locals.GAppApis[0].GateToken == "" {
□□□fmt.Println("□□□ GateToken □□ □..")
□□□time.Sleep(5 * time.Second)
□□}

□□fmt.Println("□□□ GateToken □□ □□!")

□□// □□□ GateToken□□ API □□□
□□newGateToken := locals.GAppApis[0].GateToken
□□resp, body, err := SendHttpRequest(mainApiURL, newGateToken, requestBody)
```

□□if err != nil {
□□□fmt.Println("HTTP □□ □□:", err)
□□□return nil, err
□□}
□□defer resp.Body.Close()

□□// □□□ □□ □□ □□
□□if resp.StatusCode == 200 {
□□□e.OkLog("□□□ GateToken□ □□□ HTTP □□ □□!")
□□□return body, nil
□□} else {
□□□return nil, fmt.Errorf("□□□ GateToken □□ □□, □□ □□: %d", resp.StatusCode)
□□}
□}

□// □□ GateToken□□ □□ □□□ □□
□e.OkLog("□□ GateToken□ □□□ HTTP □□ □□!")
□return body, nil
}

// HTTP □□□ □□□ □□
func SendHttpRequest(mainApiURL string, gateToken string, requestBody map[string]interface{})
(*http.Response, []byte, error) {
□// 1. JSON □□□□ □□
□jsonData, err := json.Marshal(requestBody)
□if err != nil {
□□fmt.Println("JSON □□ □□:", err)
□□return nil, nil, err
□}

□// 2. HTTP □□ □□ (POST □□)
□req, err := http.NewRequest("POST", mainApiURL, bytes.NewBuffer(jsonData))
□if err != nil {
□□fmt.Println("HTTP □□ □□ □□:", err)
□□return nil, nil, err
□}

□// 3. HTTP □□ □□ (GateToken □□)
□req.Header.Set("Content-Type", "application/json")
□req.Header.Set("GateToken", gateToken)

```
□// 4. HTTP □□ □□
□client := &http.Client{}
□resp, err := client.Do(req)
□if err != nil {
□□fmt.Println("HTTP □□ □□:", err)
□□return nil, nil, err
□}

□// 5. □□ □□□ □□
□body, err := io.ReadAll(resp.Body)
□if err != nil {
□□fmt.Println("□□ □□ □□ □□:", err)
□□return nil, nil, err
□}

□// 6. □□ □□ □□
□fmt.Println("□□ □□:", resp.StatusCode)
□fmt.Println("□□ □□:", string(body))

□return resp, body, nil
}
```

OwnerKey를 활용한 게시판 연동 예제

> 시작하기 전에

이 문서는 OwnerKey□□하 **Guest App**에□**Owner App**의 게시판 데이터 연동할 때 필요한 참고정보를 제공 및 설명합니다.

OwnerKey와 SSO는 뭐예요?

1. **OwnerKey** = 나만의 열쇠 (공유 가능한 친구링크)

- 내가 만든 콘텐츠나 앱을 다른 웹사이트에서도 쉽게 붙여서 쓸 수 있게 해주는 열쇠
- 예: 내 사이트에 있는 "예약 시스템"을 다른 가게 주인이 자신의 홈페이지에 붙일 수 있음
  → 하지만 관리 권한은 여전히 내게 있음

2. **SSO (Single Sign-On)** = 한번 로그인하면 여러 사이트 친구집도 들락날락

- 내 이메일/계정 하나로 여러 사이트(다보리로 만든)에서 자유롭게 이용 가능
- 마치 네이버 계정으로 웹툰, 쇼핑, 카페 다 들어갈 수 있는 것처럼

□□ 더 쉽게 말하면 이렇습니다:

> "Dabory는 모든 가게, 회사,나블로그, 커뮤니티가 서로 친구처럼 연결되는 마을
> 마을에서 누가 만든 물건(앱, 기능)을 누구나 쓸 수 있고,
> 각자 가게를 운영하면서도, 필요한 건 서로 공유하며 살아갈 수 있죠."

□□ 플랫폼 독점이 아니라, 모두의 플랫폼

- 쿠팡/네이버 같은 플랫폼은 모든 트래픽을 자기 플랫폼에 모아 수수료를 받습니다
- 반면 Dabory는 각자의 웹사이트가 자립형 플랫폼이 될 수 있도록 도와줍니다.
- 쇼핑몰, 블로그, ERP, 고객관리, 다 따로 만들 필요 없이,
  한 시스템 안에서 자유롭게 조합해 쓸 수 있게 한 것이죠.

□ □□□□,

기존 플랫폼	**Dabory Composable**

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

DC(Dabory Composable)[illegible] **OwnerKey** [illegible] [illegible] **SEO** [illegible] **SEO** [illegible]:

[illegible] 1. [illegible] SEO [illegible]

1.1 [illegible] [illegible] [illegible]

- OwnerKey[illegible]
- [illegible]
- [illegible] A [illegible] B [illegible]

1.2 [illegible] [illegible](Authority Sharing)

- [illegible]
- DC [illegible]
- [illegible]

1.3 [illegible] [illegible]

- [illegible] OwnerKey[illegible]
- [illegible] **(link diversity)** [illegible]

[illegible] 2. [illegible] SEO [illegible]

2.1 [illegible] [illegible] (Canonical) [illegible]

- [illegible] **canonical**
- [illegible]
- [illegible] **SEO**

2.2 [illegible](Topic Cluster) [illegible]

- [illegible]
- [illegible] **E-E-A-T (Experience, Expertise, Authoritativeness, Trustworthiness)**
- [illegible]

2.3 [illegible]

- OwnerKey 구조는 1분 내 [illegible] 빠→□
- 블로그, 상품, 이벤트 등을 빠르게 클러스터 전체에 배포 가능
- 구글이 선호하는 신선한 콘텐츠**(fresh content)**□□ 짧아져 순위 향상에 유리

□ 3. 다대다 클러스터로의 확장 효과

3.1 네트워크 효과 (Network SEO Effect)

- 각 사이트 [illegible] **SEO** [illegible] 아니라,
- 마치 분산형 콘텐츠 **CDN**처럼 웹에 광범위한 존재감을 형성
- 구글 알고리즘 상에서는 "업계 전체의 권위 있는 리소스 그룹"으로 인식 가능성 증가

3.2 경쟁이 아닌 협업 기반 SEO 전략

- 기존에는 동일 업종 간 SEO 경쟁이 치열했으나, DC 구조는 **경쟁이 아닌 동맹(Cluster SEO)**으로 방향 전환
- 협업**SEO**□ [illegible] 이어짐

□ □□ □ □□ □□

DC OwnerKey 기반 [illegible] **P2P**형 **SEO** 협업 구조

- 콘텐츠 중심의 상호 권위 강화
- 백링크의 품질과 신뢰성 향상
- 주제별 클러스터링을 통한 전문성 상승
- 검색엔진이 선호하는 시맨틱 연결 구조 강화

를 실현합니다.

> □□□ [illegible] 참여 사이트가
>
> - 고유한 컨텐츠 생산 → 친구 사이트에 공유
> - canonical 태그 정확히 적용
> - 제품/서비스와 연관된 블로그 글에 키워드 삽입 및 내부 링크 최적화
>
>   를 동시에 진행하면 SEO 시너지 효과를 최대화할 수 있습니다.

예제실행전에 아래 준비사항들을 반드시 준비해주시기 바랍니다.

□ 오너 사이트로부터 제공받은 OwnerKey (env 파일에 저장)

□ css 파일(디자인 커스터마이징의 경우)

☐ app □□□□

☐ owner app□ □□□ □□

☐ env □□□ □□□ OWNER_APP_NAME

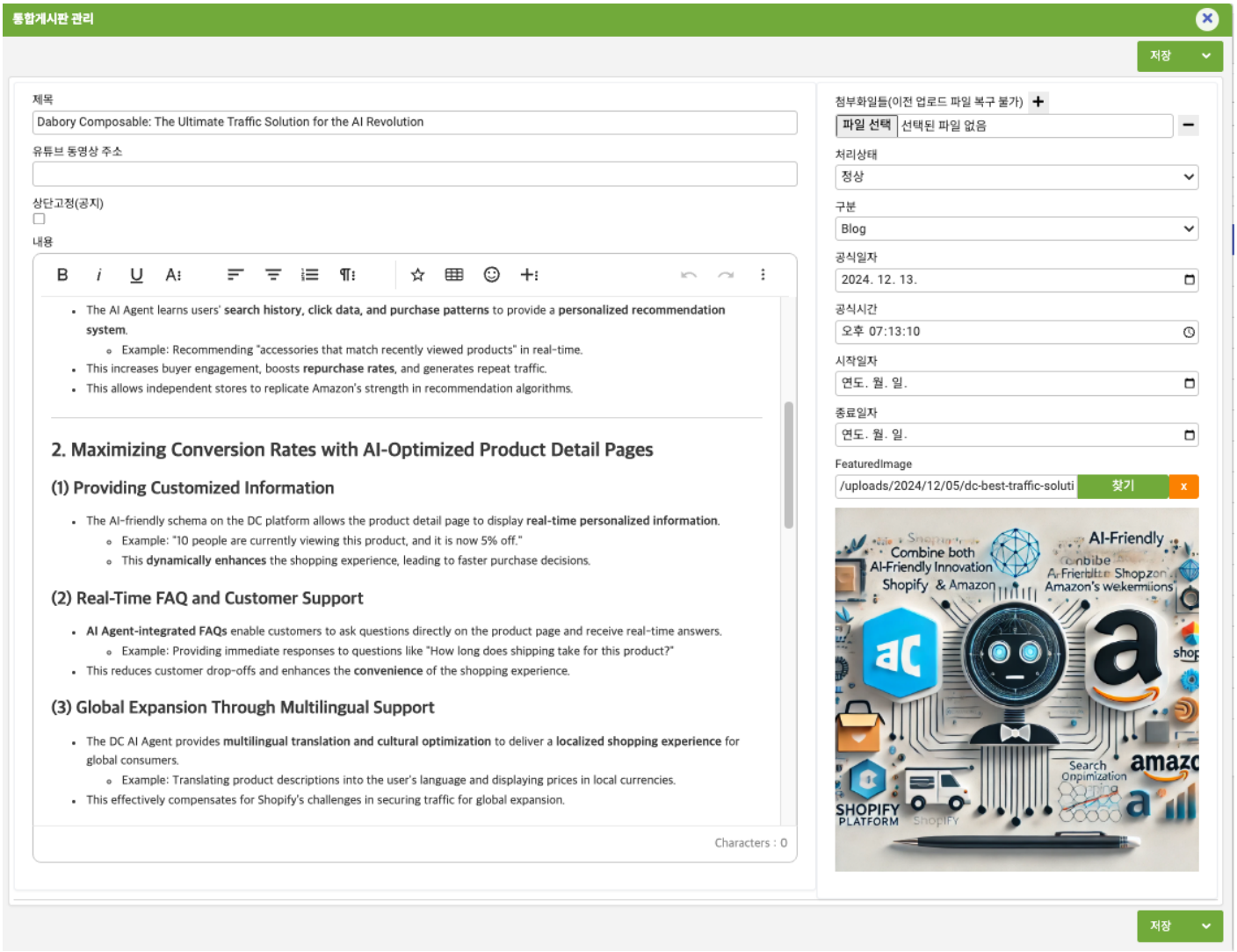
□□□ □□□ □□

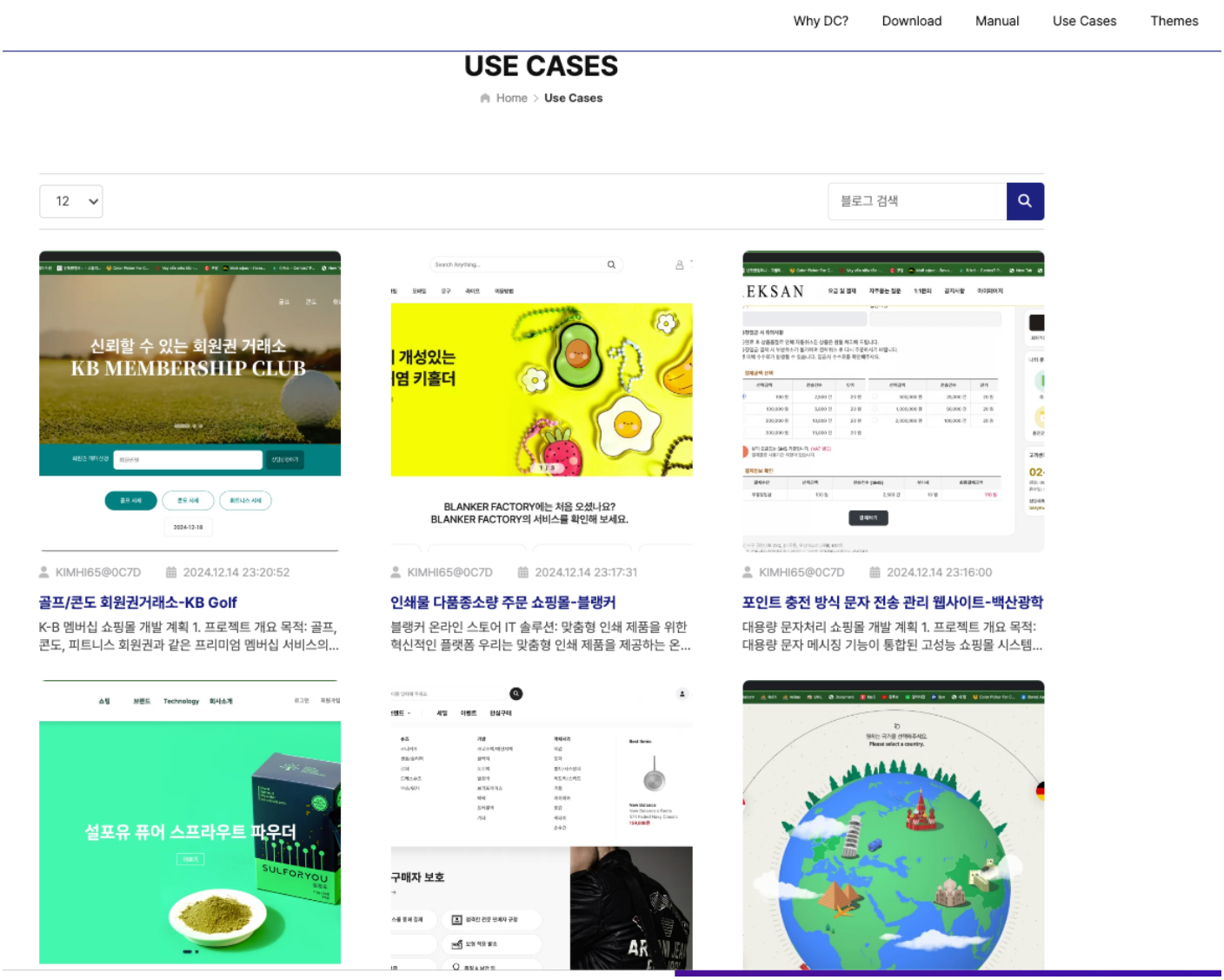
1. □□□□ OwnerKey□ env □□□ □□□□□.

2. owner app□ □□□ □□□ □□□□ guest app□ □□□□□.

3. main_menu□ □□□□□.

□□□ □□□□





□ □□□□ daboryposable □□□ USE CASES □□□(□□□)□□□. □ □□□□ factory □□□□□ □□□□□□□□.

- env □□□ □□□ □□

□□ .env □□□ OwnerKey□ □□□□□□□□. □□ □□□ □□□□□ □□ □□ □□□□□□ □□□□ □□□ □□ □□ □□ _□□□ □□□□□ □□□ □□ □□□□ □□□□□□. Frier

```
# [Friend Keys]
OWNER_APP_NAME_0=YOUR_OWNER_APP_NAME  # Owner APP□ □□ (□□□□ □□)
GUEST_APP_OWNER_KEY_0=YOUR_OWNER_KEY  # Guest APP□□ □□□□ OwnerKey □□
GUEST_APP_OWNER_URL_0=YOUR_OWNER_URL  # Owner APP□ URL □□ (ex.https://composable.daboryhost.com)

OWNER_APP_NAME_1=YOUR_OWNER_APP_NAME_2
GUEST_APP_OWNER_KEY_1=YOUR_OWNER_KEY_2
GUEST_APP_OWNER_URL_1=YOUR_OWNER_URL_2
```

```
OWNER_APP_NAME_2=YOUR_OWNER_APP_NAME_3
GUEST_APP_OWNER_KEY_2=YOUR_OWNER_KEY_3
GUEST_APP_OWNER_URL_2=YOUR_OWNER_URL_3
```

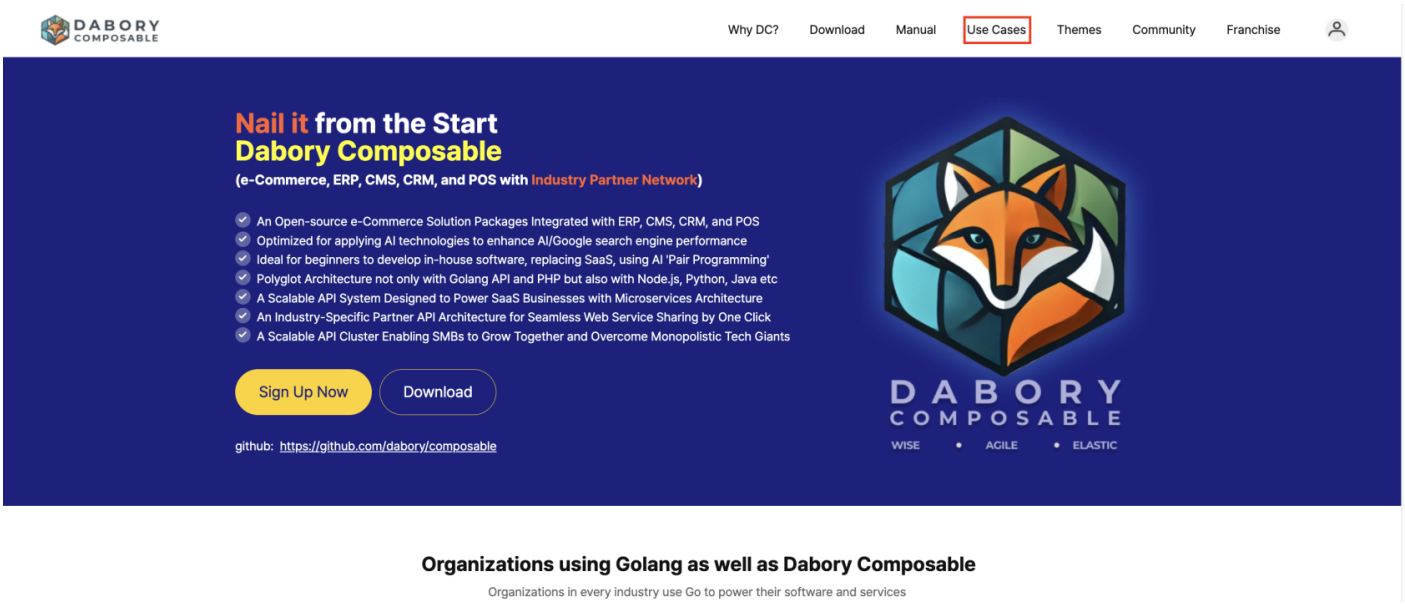
□OwnerKey□?

- OwnerKey□**GUSET APP□ MAIN APP□ API**□□□□□□□□ □
- WebApp □□□□□ OwnerKey□□□□ API □□ □ □□□ □□□□ □□□□□.

- □□□ □□□□□ □□□ URL □□

□□ □□□□□□□ □□□□□□ □□□ **url**□ □□□□ □□□ □□□□□ □□ □□□ **url**□ □□□ □ □□□□.

(1). □□□ □□□□ composable□□ Use Cases□ □□□□□.



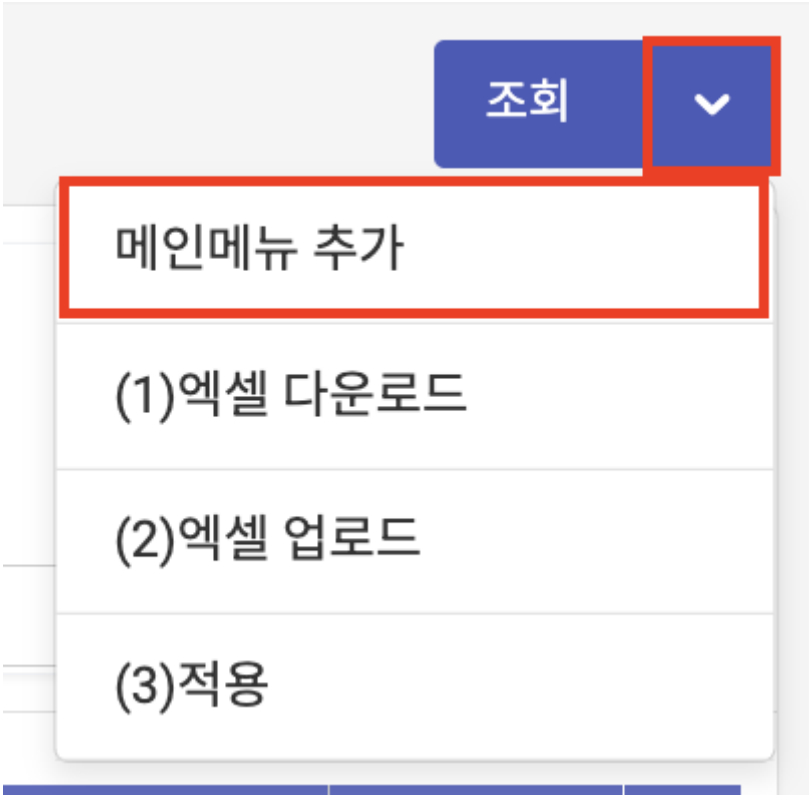
(2). /bbs/list/use-cases□ □□□□ □□□□ □□□□□□.



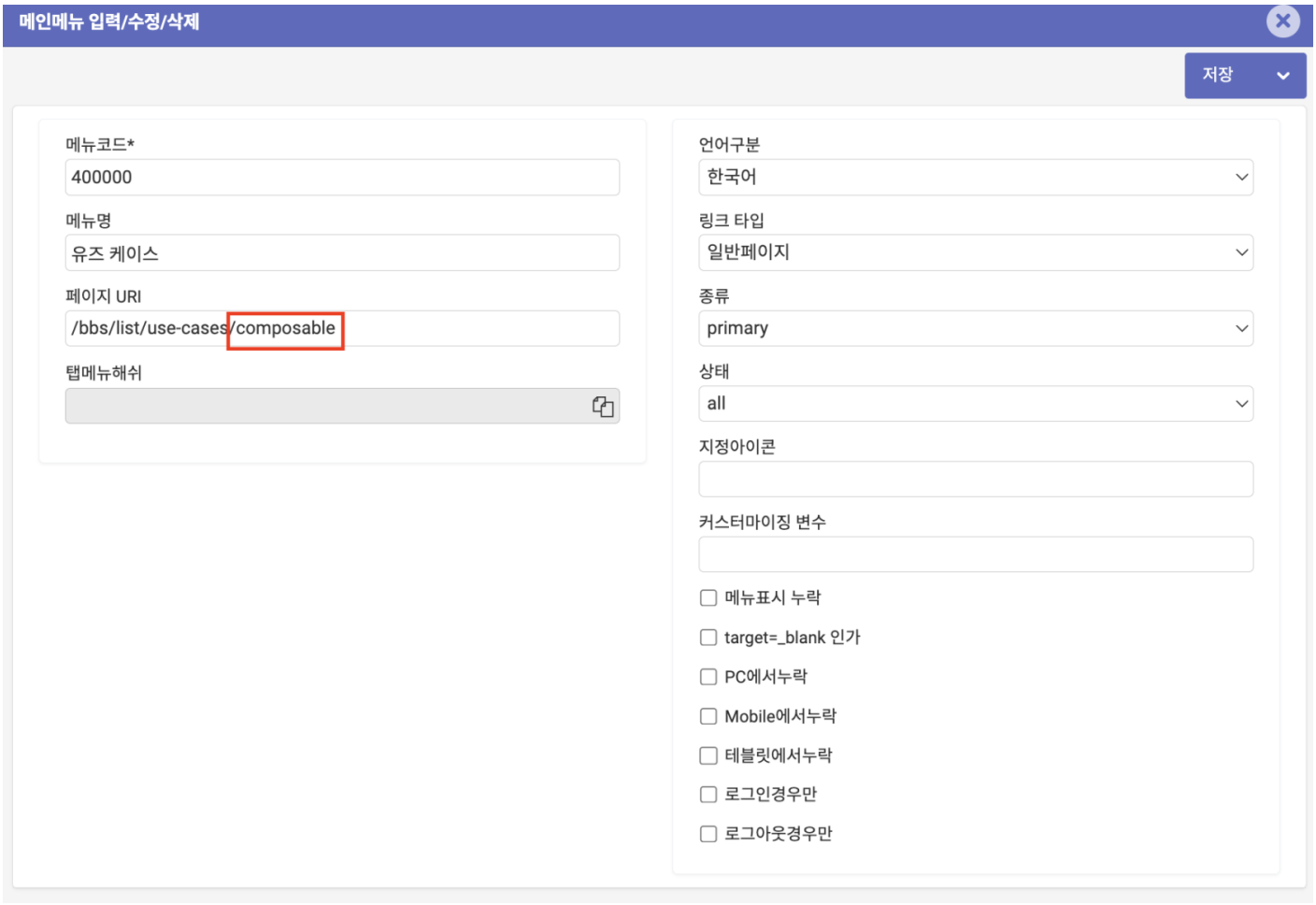
- 마이(Guest) 홈페이지 메뉴 등록
(1) 나의 홈페이지 관리자 보드로 로그인 합니다.
(2) 설정과 로그(바둑판메뉴) - 수퍼관리자(공통)- 메뉴가져오기(주의) - 홈페이지 메뉴 불러오기 메뉴로 진입합니다.


슈퍼 유저 메뉴
관리자 계정
구분설정 관리
잔액/재고/적립금 설정
메뉴 확장과 API 관리
시스템 관리(주의)
바둑판 메뉴 불러오기
사용자 메뉴 불러오기
홈페이지 메뉴 불러오기
회원(마이앱) 메뉴 불러오기
통합 설정 불러오기
거래구분 불러오기
품목구분 불러오기
게시판 구분 불러오기
표준 품목옵션 불러오기
회계계정 불러오기
DB Update
Encryption Keys

(2). 우측 상단의 메인메뉴 추가 버튼을 클릭합니다.



(3). 입력창을 모두 입력한 뒤 저장버튼을 클릭하여 저장합니다.
페이지URI 에 프렌드 사이트의 게시판URI와 .env의 OWNER_APP_NAME 을 아래와 같이 결합하여 저장합니다.



[redacted] [redacted] [redacted] [

□□□□□□□□□□ □□□□ □□ □□□ □□□□□□.

□□□□□□□□□□ □□□□ □□ □□□ □□□□□□.

(4). □□□□□ □□□□ □□□ □□□□ □□□□□□ □□□□□.



(5). composable□ □□□ □□□ □□□□□□□.

