

Kết nối website với MARZI

Website của khách hàng có thể viết bằng PHP, WordPress, Node.js, C#, Python hay công nghệ nào khác. Để app MARZI hiện được sản phẩm, đơn hàng của website đó, lập trình viên chỉ cần làm một lớp nhỏ trên website gọi là **connector**: vài đường dẫn HTTPS trả về JSON theo một chuẩn chung. Tài liệu này mô tả chuẩn đó.

Tóm tắt trong 30 giây

1) Website mở các đường dẫn /v1/. . . trả JSON. 2) MARZI gọi vào đó, mỗi yêu cầu có chữ ký HMAC-SHA256 tính bằng một khóa bí mật chung. 3) Website khai báo ở GET /v1/capabilities mình làm được những phần nào, nên có thể làm dần từng bước, **bắt đầu chỉ với sản phẩm**. 4) Có sẵn bản mẫu chạy được, bộ kiểm tra tự động và đặc tả OpenAPI để bắt đầu nhanh.

1. Bức tranh tổng thể



- App **không** nói chuyện trực tiếp với website. Chỉ máy chủ MARZI gọi connector.
- Website **không bao giờ** phải gọi ngược lại MARZI.
- Sản phẩm, giá, đơn hàng và tài khoản vẫn **nằm ở website**. MARZI không giữ bản sao đơn hàng hay sản phẩm của website, chỉ nhớ danh sách khả năng (capabilities) trong vài phút.
- Tài khoản đăng nhập app là chính tài khoản của website: người mua dùng tài khoản thành viên, chủ cửa hàng dùng tài khoản quản trị. Connector tự kiểm mật khẩu bằng cách của website; MARZI không bao giờ lưu mật khẩu đó.

2. Chọn cách kết nối

CÁCH	DÙNG KHI	TÌNH TRẠNG
A. Connector trên website (tài liệu này)	Khách đã có website riêng với sản phẩm và đơn hàng, công nghệ gì cũng được.	Đã có Chạy thật với website PHP của VIETFOOD.
B. MARZI giữ dữ liệu	Khách chưa có website, hoặc chỉ cần app.	Đang xây Nơi lưu sản phẩm và đơn đã có; công cụ nhập sản phẩm trên trình duyệt chưa xong.
C. Plugin dựng sẵn (WordPress/WooCommerce, các nền tảng bán hàng)	Cài là chạy, không cần lập trình.	Chưa có Chỉ là ý định; hiện làm theo cách A.

Website bmweb (PHP)? Có bộ cài sẵn, không cần viết mã.

Nếu website của khách chạy khung **bmweb** (các bảng `table_products` / `table_product` , `table_orders` / `table_order` ...), hãy tải **marzi-bmweb-connector.zip**: chép lên website, điền khóa bí mật, khai báo trong bảng điều khiển là chạy. Chạy được trên **PHP 7.0, 7.4 và 8.x**, tự nhận cả hai thế hệ bmweb (v1 và v2), có trang tự kiểm tra và hướng dẫn tiếng Việt trong gói (`README.md`).

Vậy câu trả lời cho "gửi tài liệu API cho bên họ rồi họ tự kết nối"?

Đúng, đó chính là cách A. Bạn gửi tài liệu này (kèm file đặc tả `openapi.yaml` , bản mẫu và bộ kiểm tra) cho lập trình viên của công ty/đại lý. Họ viết connector bằng ngôn ngữ họ quen, chạy bộ kiểm tra cho tới khi đạt, rồi đại lý nhập địa chỉ và khóa bí mật vào bảng điều khiển MARZI. Không bên nào phải sửa app.

3. Yêu cầu tối thiểu

- **HTTPS** với chứng chỉ hợp lệ (đuôi chứng chỉ trung gian). Địa chỉ phải là website công khai trên internet: MARZI từ chối địa chỉ IP nội bộ, `localhost` , tên `.local` / `.internal` và một số cổng dịch vụ (SSH, cơ sở dữ liệu, thư...). Dùng cổng 443 mặc định là đủ.

- **Không chuyển hướng.** MARZI không đi theo chuyển hướng (301/302), nên địa chỉ nhập vào phải là địa chỉ cuối cùng (đúng `https` , đúng `www` hoặc không, đúng dấu `/`).
- Trả **JSON UTF-8** với `Content-Type: application/json` , không BOM, không in thêm cảnh báo hay HTML.
- Trả lời trong **10 giây** và không quá **5 MB** mỗi lần. Luôn giới hạn `limit` (tối đa 100).
- Đồng hồ máy chủ website **đúng giờ** (lệch dưới 5 phút), vì chữ ký có thời hạn.
- Một **khóa bí mật** dài, ngẫu nhiên (từ 32 ký tự). Bảng điều khiển MARZI có nút "Tạo khóa" tạo sẵn khóa 48 ký tự.
- Địa chỉ nhập vào bảng điều khiển là **thư mục gốc của connector**, ví dụ `https://www.tenmien.com/marzi-connector` ; MARZI nối thêm `/v1/...` phía sau.

4. Khóa bí mật và chữ ký

Mọi yêu cầu MARZI gửi đi đều có hai header. Connector **phải kiểm cả hai** trước khi làm bất cứ việc gì, kể cả đọc dữ liệu.

HEADER	GIÁ TRỊ
X-Marzi-Timestamp	Giờ gửi, tính bằng giây từ 1970 (unix), ví dụ <code>1700000000</code> .
X-Marzi-Signature	Chuỗi hex (chữ thường) của <code>HMAC-SHA256(khóa bí mật, nội dung ký)</code> .

Nội dung ký gồm bốn dòng nối bằng ký tự xuống dòng `\n` (không có dòng trống ở cuối):

```
{timestamp}
{METHOD}                (GET hoặc POST, chữ hoa)
{đường dẫn + query}      (ví dụ /v1/products?page=1&limit=20)
{sha256 hex của body}    (GET không có body: dùng sha256 của chuỗi rỗng)
```

Bốn quy tắc dễ sai

1. **Đường dẫn ký là đường dẫn không có thư mục chứa connector.** Nếu connector nằm ở `https://site.com/marzi-connector` thì website nhận `/marzi-connector/v1/products` nhưng MARZI ký `/v1/products` . Hãy cắt tiền tố thư mục trước khi kiểm chữ ký và trước khi chọn route (bản mẫu có biến `BASE_PATH` , bản PHP có cấu hình `base_path`).
2. **Ký đúng nguyên văn query** như nhận được (thứ tự tham số, mã hóa %), đừng sắp xếp lại.

3. **Hash body bằng byte gốc** nhận được, đừng đọc JSON rồi dựng lại chuỗi.
4. **So sánh bằng hàm so sánh thời gian cố định** (`hash_equals` , `timingSafeEqual` , `hmac.compare_digest` ...), không dùng `==` . Từ chối nếu `|giờ hiện tại - timestamp| > 300` giây để chặn phát lại.

Giá trị mẫu để tự kiểm tra bản cài đặt của bạn

```
secret      = test-secret-key-0123456789abcdef
timestamp   = 1700000000
method      = POST
uri         = /v1/auth/verify?x=1
body        = {"username":"khach", "password":"mat khau"}

signature   = 6555990f154cc9a91c0fb5182bfb991bebcca1fed0a1f137a11f0be74d5913d7

(GET /v1/products?page=1&limit=20, body rỗng, cùng khóa và giờ)
signature   = b0c9e960e037ae0e9266c7e7c1c20e45fa8e451ae84d83b586d8df671534adb7
```

Nếu hàm của bạn trả đúng hai chuỗi này thì phần chữ ký đã đúng.

Mã mẫu

Ba đoạn dưới đây đã được chạy thử và cho đúng giá trị mẫu. Đoạn C# viết theo cùng công thức nhưng **chưa được biên dịch và chạy thử**, hãy so với giá trị mẫu trước khi dùng.

Node.js

JAVASCRIPT

```
// Node.js 18+. Check or make a MARZI signature.
const crypto = require('crypto');

/** hex( HMAC-SHA256( secret, timestamp \n METHOD \n path?query \n sha256hex(body) ) ); body
is the raw text (" for GET). */
function sign(secret, timestamp, method, uri, body) {
    const bodyHash = crypto.createHash('sha256').update(body).digest('hex');
    return crypto.createHmac('sha256',
secret).update(`${timestamp}\n${method}\n${uri}\n${bodyHash}`).digest('hex');
}

/** True when the request really comes from MARZI. `uri` is the path + query WITHOUT the
folder your connector lives in. */
function isFromMarzi(secret, headers, method, uri, body, maxSkewSeconds = 300) {
    const timestamp = String(headers['x-marzi-timestamp'] || '');
    const given = String(headers['x-marzi-signature'] || '').toLowerCase();
    if (!/^\\d+$/ .test(timestamp) || !given) return false;
    if (Math.abs(Date.now() / 1000 - Number(timestamp)) > maxSkewSeconds) return false; // too
old: someone may be replaying it
    const expected = Buffer.from(sign(secret, timestamp, method, uri, body));
    const actual = Buffer.from(given);
    return expected.length === actual.length && crypto.timingSafeEqual(expected, actual); //
never compare with ===
}

module.exports = { sign, isFromMarzi };

if (require.main === module) {
    // Test vector from the documentation: must print
6555990f154cc9a91c0fb5182bfb991bebcca1fed0a1f137a11f0be74d5913d7
    console.log(sign('test-secret-key-0123456789abcdef', '1700000000', 'POST',
'/v1/auth/verify?x=1', '{"username":"khach","password":"mat khau"}'));
}
```

Python

PYTHON

```
"""Python 3. Check or make a MARZI signature (standard library only)."""
import hashlib
import hmac
import time

def sign(secret: str, timestamp: str, method: str, uri: str, body: bytes) -> str:
    """hex( HMAC-SHA256( secret, timestamp \\n METHOD \\n path?query \\n sha256hex(body) ) );
    body is the raw bytes (b"" for GET)."""
    body_hash = hashlib.sha256(body).hexdigest()
    payload = f"{timestamp}\\n{method}\\n{uri}\\n{body_hash}".encode()
    return hmac.new(secret.encode(), payload, hashlib.sha256).hexdigest()

def is_from_marzi(secret: str, headers: dict, method: str, uri: str, body: bytes,
max_skew_seconds: int = 300) -> bool:
    """True when the request really comes from MARZI. `uri` is the path + query WITHOUT the
    folder your connector lives in."""
    timestamp = str(headers.get("X-Marzi-Timestamp", ""))
    given = str(headers.get("X-Marzi-Signature", "")).lower()
    if not timestamp.isdigit() or not given:
        return False
    if abs(time.time() - int(timestamp)) > max_skew_seconds:
        return False # too old: someone may be replaying it
    return hmac.compare_digest(sign(secret, timestamp, method, uri, body), given) # never
compare with ==

if __name__ == "__main__":
    # Test vector from the documentation: must print
    6555990f154cc9a91c0fb5182bfb991bebcca1fed0a1f137a11f0be74d5913d7
    print(sign("test-secret-key-0123456789abcdef", "1700000000", "POST", "/v1/auth/verify?
x=1", '{"username":"khach","password":"mat khau"}'.encode()))
```

PHP

PHP

```
<?php
// PHP 7.4+. Check or make a MARZI signature.

/** hex( HMAC-SHA256( secret, timestamp \n METHOD \n path?query \n sha256hex(body) ) ); $body
is the raw text ('' for GET). */
function marzi_sign(string $secret, string $timestamp, string $method, string $uri, string
$body): string
{
    return hash_hmac('sha256', $timestamp . "\n" . $method . "\n" . $uri . "\n" .
hash('sha256', $body), $secret);
}

/** True when the request really comes from MARZI. $uri is the path + query WITHOUT the
folder your connector lives in. */
function marzi_is_genuine(string $secret, string $uri, int $maxSkewSeconds = 300): bool
{
    $timestamp = $_SERVER['HTTP_X_MARZI_TIMESTAMP'] ?? '';
    $given = strtolower($_SERVER['HTTP_X_MARZI_SIGNATURE'] ?? '');
    if (!ctype_digit($timestamp) || $given === '') {
        return false;
    }
    if (abs(time() - (int)$timestamp) > $maxSkewSeconds) {
        return false; // too old: someone may be replaying it
    }
    $body = file_get_contents('php://input') ?: '';
    return hash_equals(marzi_sign($secret, $timestamp, $_SERVER['REQUEST_METHOD'], $uri,
$body), $given); // never compare with ==
}

if (PHP_SAPI === 'cli' && realpath($_SERVER['SCRIPT_FILENAME']) === __FILE__) {
    // Test vector from the documentation: must print
    6555990f154cc9a91c0fb5182bfb991bebcca1fed0a1f137a11f0be74d5913d7
    echo marzi_sign('test-secret-key-0123456789abcdef', '1700000000', 'POST',
'/v1/auth/verify?x=1', '{"username":"khach","password":"mat khau"}'), "\n";
}
```

C# (.NET 6+)

CSHARP

```
// C# (.NET 6+). Check or make a MARZI signature.
// NOTE: this file was written to match the other languages and the test vector, but it has
// NOT been compiled or run by
// the MARZI team. Check it against the test vector in the documentation before you rely on
// it.
using System;
using System.Security.Cryptography;
using System.Text;

public static class MarziSignature
{
    /// hex( HMAC-SHA256( secret, timestamp \n METHOD \n path?query \n sha256hex(body) ) );
    body is the raw bytes (empty for GET).
    public static string Sign(string secret, string timestamp, string method, string uri,
    byte[] body)
    {
        var bodyHash = Convert.ToHexString(SHA256.HashData(body)).ToLowerInvariant();
        var payload = $"{timestamp}\n{method}\n{uri}\n{bodyHash}";
        using var hmac = new HMACSHA256(Encoding.UTF8.GetBytes(secret));
        return
        Convert.ToHexString(hmac.ComputeHash(Encoding.UTF8.GetBytes(payload))).ToLowerInvariant();
    }

    /// True when the request really comes from MARZI. `uri` is the path + query WITHOUT the
    folder your connector lives in.
    public static bool IsFromMarzi(string secret, string? timestamp, string? signature,
    string method, string uri, byte[] body, int maxSkewSeconds = 300)
    {
        if (string.IsNullOrEmpty(timestamp) || string.IsNullOrEmpty(signature) ||
        !long.TryParse(timestamp, out var seconds)) return false;
        if (Math.Abs(DateTimeOffset.UtcNow.ToUnixTimeSeconds() - seconds) > maxSkewSeconds)
        return false; // too old: maybe a replay
        var expected = Encoding.ASCII.GetBytes(Sign(secret, timestamp, method, uri, body));
        var actual = Encoding.ASCII.GetBytes(signature.ToLowerInvariant());
        return CryptographicOperations.FixedTimeEquals(expected, actual); // never compare
        with ==
    }

    // Test vector from the documentation: Sign("test-secret-key-0123456789abcdef",
    "1700000000", "POST", "/v1/auth/verify?x=1",
    // Encoding.UTF8.GetBytes($"{username}:" + "khach", "password" + "mat khau"}))
    // must return 6555990f154cc9a91c0fb5182bfb991bebccafed0a1f137a11f0be74d5913d7
}
```

WordPress

Viết connector thành một plugin nhỏ: đăng ký route bằng `register_rest_route('marzi/v1', ...)` hoặc dùng một file PHP riêng ngoài WordPress. Lưu ý hai điều: (1) kiểm chữ ký trên **đường dẫn đã bỏ tiền tố** (`/wp-json/marzi`) và trên **body thô** (`$request->get_body()`); (2) tắt cache trang cho các đường dẫn này và cho phép chúng đi qua plugin bảo mật/tường lửa (Wordfence, Cloudflare...).

5. Làm theo từng giai đoạn

Connector khai báo ở `GET /v1/capabilities` những khả năng nó có. MARZI chỉ bật tính năng tương ứng, nên bạn có thể giao từng phần. Khuyến nghị làm theo thứ tự:

#	KHẢ NĂNG (CAPABILITIES)	NGƯỜI DÙNG THẤY GÌ	ĐƯỜNG DẪN
1	<code>products.read</code> , <code>categories.read</code>	App người mua xem sản phẩm, danh mục, tìm kiếm.	3
2	<code>auth.verify</code> , <code>orders.read</code>	Đăng nhập bằng tài khoản website, giá thành viên, xem lịch sử đơn.	4
3	<code>orders.create</code>	Giỏ hàng và đặt hàng ngay trong app.	2 (cộng phương thức thanh toán)
4	<code>staff.verify</code> , <code>staff.orders</code> , <code>staff.customers</code> , <code>staff.products</code>	App quản lý: chủ cửa hàng đăng nhập, xem và lọc đơn, khách hàng, sản phẩm.	11
5	<code>staff.orders.write</code> , <code>staff.products.write</code>	Đổi trạng thái đơn, đánh dấu đã thanh toán, bật/tắt và sửa nhanh giá sản phẩm.	4
6	<code>pages.read</code>	Trang giới thiệu, tin tức, khuyến mãi trong app người mua.	4
7	<code>auth.qr</code>	Người dùng đăng nhập trang web của khách bằng cách quét mã QR bằng app MARZI (mục 6.6).	3 + trang đăng nhập của website

Giai đoạn 1 là đủ để có app chạy

Chỉ cần `/v1/capabilities`, `/v1/products`, `/v1/products/{id}` (và `/v1/categories` nếu muốn lọc theo danh mục). Phần còn lại thêm sau, mỗi lần thêm nhớ cập nhật danh sách capabilities.

6. Chi tiết từng đường dẫn

Quy ước chung: `id` luôn là **chuỗi** (`"12"`, không phải số 12); tiền là **số** (không phải chuỗi) theo đơn vị nhỏ nhất của loại tiền cửa hàng đang dùng (VND, KRW không có số lẻ); thời gian theo **ISO 8601 UTC**; chữ nhiều ngôn ngữ ở dạng `{"vi": ..., "en": ..., "kr": ...}` (tiếng Hàn là `kr`, không phải `ko`; chưa dịch thì lặp lại tiếng Việt); trường không có giá trị trả `null`, không bỏ trường. Phân trang bằng `page` (từ 1) và `limit` (tối đa 100). Các ví dụ dưới đây là câu trả lời thật của bản mẫu, đã rút gọn danh sách còn một phần tử.

6.1 Khả năng **bắt buộc**

GET `/v1/capabilities`

JSON

```
{
  "connector": "marzi-sample",
  "contract": 1,
  "capabilities": [
    "products.read",
    "categories.read",
    "auth.verify",
    "orders.read",
    "orders.create",
    "staff.verify",
    "staff.orders",
    "staff.orders.write",
    "staff.customers",
    "staff.products",
    "staff.products.write",
    "pages.read"
  ]
}
```

6.2 Sản phẩm: `products.read` , `categories.read`

GET `/v1/products?page=1&limit=20&q=&category=&member=`

THAM SỐ	Ý NGHĨA
<code>q</code>	Tìm trong tên và mã sản phẩm.
<code>category</code>	Lọc theo <code>id</code> danh mục.
<code>member</code>	Có khi người mua đã đăng nhập: trả giá thành viên cho họ (nếu website có).

JSON

```
{
  "items": [
    {
      "id": "1",
      "code": "BAN001",
      "names": {
        "vi": "Bánh su kem",
        "en": "Cream puffs",
        "kr": "슈크림"
      },
      "description": "Vỏ giòn, nhân kem vani béo nhẹ.",
      "price": 199000,
      "regularPrice": 35000,
      "variants": [
        {
          "id": "1-6",
          "name": "Hộp 6 cái",
          "price": 199000,
          "stock": 40
        },
        {
          "id": "1-12",
          "name": "Hộp 12 cái",
          "price": 359000,
          "stock": 25
        }
      ],
      "image": "p1.png",
      "stock": 60
    }
  ],
  "total": 9
}
```

- **price** : giá bán **cho người đang xem**. Sản phẩm bán theo quy cách (variants): giá thấp nhất trong các quy cách. **0** nghĩa là "liên hệ cửa hàng".
- **regularPrice** : giá gốc, dùng để gạch giá khi đang giảm.
- **image** : **chỉ tên file**, MARZI ghép với thư mục ảnh bạn nhập ở bảng điều khiển (ví dụ <https://site.com/upload/product/>).
- **variants** : danh sách quy cách (hộp 6, hộp 12, size S/M/L...). Không có thì trả **[]** . **stock** của quy cách là **null** nếu không theo dõi tồn kho.
- Chỉ trả sản phẩm **đang bán** (không trả sản phẩm đã ẩn).

- **url** (không bắt buộc): trang của sản phẩm này trên website, là đường dẫn dưới địa chỉ website (ví dụ `/san-pham/ban001`) hoặc địa chỉ https đầy đủ **trên chính website đó**. Dùng cho link chia sẻ: nếu cửa hàng chọn "chuyển thẳng sang website" ở bảng điều khiển, người chưa cài app bấm vào link sẽ được đưa tới trang này; nếu không có **url** , họ thấy trang giới thiệu sản phẩm của MARZI. Địa chỉ ở website khác, hoặc không phải https, bị bỏ qua.

Một sản phẩm có quy cách:

JSON

```
{
  "id": "1",
  "code": "BAN001",
  "names": {
    "vi": "Bánh su kem",
    "en": "Cream puffs",
    "kr": "슈크림"
  },
  "description": "Vỏ giòn, nhân kem vani béo nhẹ.",
  "price": 199000,
  "regularPrice": 35000,
  "variants": [
    {
      "id": "1-6",
      "name": "Hộp 6 cái",
      "price": 199000,
      "stock": 40
    },
    {
      "id": "1-12",
      "name": "Hộp 12 cái",
      "price": 359000,
      "stock": 25
    }
  ],
  "image": "p1.png",
  "stock": 60
}
```

`GET /v1/products/{id}?member=` trả một sản phẩm như trên; không có hoặc đã ẩn thì trả **404**.

Sản phẩm có nhiều lựa chọn (không bắt buộc)

Có hai cách, **website tự chọn cách hợp với dữ liệu của mình**, không cần cài đặt gì ở bảng điều khiển; app nhận ra qua dữ liệu được trả về, và hai cách dùng lẫn trong một cửa hàng được.

- **Quy cách trong một sản phẩm** (`variants` , ở trên): hợp khi các lựa chọn chỉ khác giá, ví dụ hộp 6 / hộp 12, 250g / 500g.
- **Mỗi lựa chọn là một sản phẩm riêng**, gom bằng `group` : hợp khi mỗi lựa chọn có **ảnh, tồn kho, giá khuyến mãi riêng** hoặc có **nhiều chiều** (hương vị × size, màu × cỡ). Mỗi sản phẩm vẫn là một sản phẩm đầy đủ như trên, thêm một trường `group` :

JSON

```
{
  "id": "14",
  "code": "DRK104",
  "names": {
    "vi": "Trà trái cây",
    "en": "Fruit tea",
    "kr": "과일 티"
  },
  "description": "Trà thanh mát, chọn hương vị và size theo ý bạn.",
  "price": 39000,
  "regularPrice": 46000,
  "variants": [],
  "image": "tra-dau.png",
  "stock": 18,
  "group": {
    "id": "tra-trai-cay",
    "options": [
      {
        "name": "Hương vị",
        "value": "Dâu",
        "names": { "vi": "Hương vị", "en": "Flavor", "kr": "맛" },
        "values": { "vi": "Dâu", "en": "Strawberry", "kr": "딸기" }
      },
      { "name": "Size", "value": "M" }
    ]
  }
}
```

- `group.id` : mã nhóm, giống nhau ở mọi sản phẩm của nhóm.
- `group.options` : lựa chọn của **sản phẩm này**; tên (`name`) và thứ tự giống nhau ở mọi sản phẩm của nhóm. `names` / `values` (`vi` , `en` , `kr`) là bản dịch, không bắt buộc.
- Tổ hợp không bán thì **không tạo sản phẩm**; app tự làm mờ. Hết hàng thì cứ để `stock: 0` .
- Hai tham số thêm cho `GET /v1/products` : `collapse=1` = mỗi nhóm chỉ trả **một** sản phẩm (lựa chọn đầu tiên) kèm `group.count` (số lựa chọn) và `group.priceFrom` (giá thấp nhất), để danh sách không

lập; `group=<mã>` = trả mọi sản phẩm của nhóm đó (tối đa 50), không gom. Website không dùng nhóm thì bỏ qua cả hai.

- Nếu website chưa gom khi nhận `collapse=1`, MARZI vẫn chỉ giữ một sản phẩm cho mỗi nhóm trong từng trang kết quả, nhưng tổng số và phân trang có thể lệch: nên làm đúng.

Bộ kiểm tra `check.js` tự kiểm thêm hai điểm trên khi website có trả `group`.

GET `/v1/categories`

JSON

```
{
  "items": [
    {
      "id": "banh",
      "names": {
        "vi": "Bánh ngọt",
        "en": "Pastries",
        "kr": "베이커리"
      }
    },
    {
      "id": "do-uong",
      "names": {
        "vi": "Đồ uống",
        "en": "Drinks",
        "kr": "음료"
      }
    },
    {
      "id": "qua-tang",
      "names": {
        "vi": "Quà tặng",
        "en": "Gifts",
        "kr": "선물"
      }
    }
  ]
}
```

6.3 Người mua: `auth.verify`, `orders.read`, `orders.create`

POST `/v1/auth/verify`, body `{"username":"...", "password":"..."}`. Đúng thì trả thông tin người mua; sai trả **401** với `"error":"invalid_credentials"`.

JSON

```
{
  "memberId": "1",
  "username": "khachmau",
  "name": "Nguyễn Thị Lan",
  "email": "lan@example.com",
  "phone": "0900000001"
}
```

GET /v1/members/{memberId}/orders?page=1&limit=20 : đơn của người đó, mới nhất trước. GET /v1/members/{memberId}/orders/{id} : một đơn; đơn của **người khác phải trả 404**.

JSON

```
{
  "id": "1001",
  "code": "MAU1001",
  "status": "pending",
  "paid": false,
  "createdAt": "2026-10-17T03:08:45.000Z",
  "fullName": "Nguyễn Thị Lan",
  "phone": "0900000001",
  "address": "Số 12, đường Hoa Mai, Quận Mầu",
  "note": "Giao giờ hành chính",
  "subtotal": 429000,
  "shippingFee": 0,
  "total": 429000,
  "lines": [
    {
      "productId": "1",
      "name": "Bánh su kem (Hộp 12 cái)",
      "unitPrice": 359000,
      "quantity": 1
    },
    {
      "productId": "4",
      "name": "Cà phê sữa đá (Size M)",
      "unitPrice": 35000,
      "quantity": 2
    }
  ]
}
```

`status` là trạng thái rút gọn hiển thị cho người mua: `pending` , `confirmed` , `packing` , `shipping` , `delivered` , `cancelled` . Website có trạng thái riêng thì ánh xạ về giá trị gần nhất (không rõ: `pending`).

GET `/v1/payment-methods` :

JSON

```
{
  "items": [
    {
      "id": "1",
      "names": {
        "vi": "Thanh toán khi nhận hàng",
        "en": "Cash on delivery",
        "kr": "현금 결제"
      }
    },
    {
      "id": "2",
      "names": {
        "vi": "Chuyển khoản",
        "en": "Bank transfer",
        "kr": "계좌 이체"
      }
    }
  ]
}
```

POST `/v1/members/{memberId}/orders` : đặt hàng. **App chỉ gửi "mua gì, bao nhiêu", không bao giờ gửi giá.** Website tự tính giá, kiểm tồn kho, tính phí giao hàng và tổng tiền, rồi trả đơn vừa tạo (giống `GET .../orders/{id}`).

JSON

```
{
  "items": [
    {
      "productId": "1",
      "variantId": "1-6",
      "quantity": 2
    },
    {
      "productId": "4",
      "variantId": "4-m",
      "quantity": 1
    }
  ],
  "fullName": "Nguyễn Thị Lan",
  "phone": "0900000001",
  "address": "Số 12, đường Hoa Mai, Quận Mẫu",
  "note": "Giao giờ hành chính",
  "paymentMethodId": "1"
}
```

Nếu không nhận đơn (hết hàng, thiếu thông tin...) trả **422** kèm `message` bằng tiếng Việt để hiểu, app sẽ hiện nguyên văn cho người mua:

JSON

```
{
  "error": "invalid_order",
  "message": "Giỏ hàng đang trống"
}
```

Đơn phải tạo trọn vẹn hoặc không tạo gì.

Nếu bảng đơn của website không hỗ trợ giao dịch (ví dụ MyISAM), hãy tự xóa đơn dở khi ghi dòng sản phẩm lỗi. Đừng để lại đơn thiếu sản phẩm.

6.4 App quản lý: `staff.*`

Chủ cửa hàng đăng nhập bằng tài khoản quản trị của website. **Connector phải áp đúng phân quyền của website** cho từng tài khoản.

POST `/v1/staff/verify` , body như `auth/verify` . Sai mật khẩu: **401**. Đúng nhưng tài khoản không được xem đơn: **403**. `canEdit=false` nghĩa là chỉ được xem, không được sửa.

JSON

```
{
  "staffId": "1",
  "username": "chushop",
  "name": "Chủ cửa hàng Mẫu",
  "canEdit": true
}
```

Mọi đường dẫn dưới `/v1/staff/{staffId}/...` phải **kiểm lại quyền của staffId đó ở mỗi lần gọi** (đừng mặc định đã kiểm lúc đăng nhập). Không đủ quyền: **403**.

JSON

```
{
  "error": "forbidden",
  "message": "This account may not manage orders"
}
```

Đơn hàng

YÊU CẦU	VIỆC LÀM
GET <code>.../order-statuses</code>	Các trạng thái đơn mà cửa hàng tự đặt (số <code>id</code> do website quy định).
GET <code>.../orders</code>	Danh sách đơn. Tham số: <code>page</code> , <code>limit</code> , <code>status</code> (id), <code>q</code> (tìm mã/tên/điện thoại), <code>since</code> (sau thời điểm, unix), <code>from</code> (từ, tính cả), <code>to</code> (đến, không tính), <code>paid</code> (<code>1</code> hoặc <code>0</code>).
GET <code>.../orders/stats?since=</code>	Số đơn mới và số đơn theo từng trạng thái, để app báo "có đơn mới".
GET <code>.../orders/{id}</code>	Một đơn (đầy đủ thông tin người đặt).
POST <code>.../orders/{id}/status</code>	Body <code>{"statusId": 2}</code> . Trả đơn sau khi đổi. Sai id: 422. Yêu cầu <code>staff.orders.write</code> .
POST <code>.../orders/{id}/payment</code>	Body <code>{"paid": true}</code> (phải là true/false). Trả đơn sau khi đổi. Yêu cầu <code>staff.orders.write</code> .

amount là tổng của *mọi* đơn khớp bộ lọc, không chỉ trang hiện tại

Màn đơn hàng của app hiện "số đơn" và "doanh thu" của bộ lọc đang chọn. **total** là số đơn khớp, **amount** là tổng doanh thu của chúng (bỏ qua đơn đã hủy). Tính trong cơ sở dữ liệu, đừng cộng các dòng của một trang.

JSON

```
{
  "items": [
    {
      "id": "1001",
      "code": "MAU1001",
      "status": "pending",
      "paid": false,
      "createdAt": "2026-10-17T03:08:45.000Z",
      "fullName": "Nguyễn Thị Lan",
      "phone": "0900000001",
      "address": "Số 12, đường Hoa Mai, Quận Mầu",
      "note": "Giao giờ hành chính",
      "subtotal": 429000,
      "shippingFee": 0,
      "total": 429000,
      "lines": [
        {
          "productId": "1",
          "name": "Bánh su kem (Hộp 12 cái)",
          "unitPrice": 359000,
          "quantity": 1
        },
        {
          "productId": "4",
          "name": "Cà phê sữa đá (Size M)",
          "unitPrice": 35000,
          "quantity": 2
        }
      ],
      "statusId": 1,
      "statusNames": {
        "vi": "Mới đặt",
        "en": "New",
        "kr": "신규"
      },
      "paymentMethodNames": {
        "vi": "Thanh toán khi nhận hàng",
        "en": "Cash on delivery",
        "kr": "현금 결제"
      },
      "memberId": "1",
      "email": "lan@example.com",
      "introCode": null
    }
  ],
  "total": 12,
```

```
"amount": 3656000
```

```
}
```

JSON

```
{
  "items": [
    {
      "id": 1,
      "names": {
        "vi": "Mới đặt",
        "en": "New",
        "kr": "신규"
      }
    },
    {
      "id": 2,
      "names": {
        "vi": "Đã xác nhận",
        "en": "Confirmed",
        "kr": "확인됨"
      }
    },
    {
      "id": 3,
      "names": {
        "vi": "Đang đóng gói",
        "en": "Packing",
        "kr": "포장 중"
      }
    },
    {
      "id": 4,
      "names": {
        "vi": "Đang giao",
        "en": "Shipping",
        "kr": "배송 중"
      }
    },
    {
      "id": 5,
      "names": {
        "vi": "Đã giao",
        "en": "Delivered",
        "kr": "배송 완료"
      }
    },
    {
      "id": 6,
      "names": {
        "vi": "Đã hủy",
```

```
    "en": "Cancelled",
    "kr": "취소됨"
  }
}
]
```

JSON

```
{
  "newSince": 11,
  "byStatus": {
    "1": 2,
    "2": 1,
    "3": 1,
    "4": 1,
    "5": 6,
    "6": 1
  },
  "now": 1792210127
}
```

Khách hàng

GET `.../customers?page=&limit=&q=&sort=` với `sort` là `spent` (chi tiêu), `orders` (số đơn), `recent` (mua gần đây) hoặc `joined` (mới tham gia). Số điện thoại trong danh sách phải **che bớt** (`090****001`). GET `.../customers/{id}` trả đầy đủ thông tin kèm 5 đơn gần nhất.

JSON

```
{
  "items": [
    {
      "id": "2",
      "name": "Trần Văn Minh",
      "phoneMasked": "090****002",
      "orders": 2,
      "spent": 1486000,
      "lastOrderAt": "2026-10-16T08:08:45.000Z",
      "joinedAt": "2026-08-18T04:08:45.000Z"
    }
  ],
  "total": 6
}
```

Sản phẩm (cho chủ cửa hàng)

GET `.../products?page=&limit=&q=&visible=` (`visible` là `1` hoặc `0` ; ở đây **trả cả sản phẩm đang ẩn**), GET `.../products/{id}` .

JSON

```
{
  "items": [
    {
      "id": "1",
      "code": "BAN001",
      "names": {
        "vi": "Bánh su kem",
        "en": "Cream puffs",
        "kr": "슈크림"
      },
      "image": "p1.png",
      "visible": true,
      "regularPrice": 35000,
      "salePrice": 0,
      "variants": [
        {
          "id": "1-6",
          "name": "Hộp 6 cái",
          "price": 199000,
          "salePrice": 0
        },
        {
          "id": "1-12",
          "name": "Hộp 12 cái",
          "price": 389000,
          "salePrice": 359000
        }
      ],
      "stock": 60
    }
  ],
  "total": 10
}
```

POST `.../products/{id}/visibility` với `{"visible": false}` để ẩn/hiện. POST `.../products/{id}` để sửa nhanh: **chỉ trường nào có mặt mới đổi**; kiểm tra hết trước rồi mới ghi, một trường sai thì không đổi gì (422). Giá trong khoảng 0 đến 1.000.000.000; tên 1 đến 255 ký tự. Cả hai đều yêu cầu `staff.products.write` .

JSON

```
{
  "names": {
    "vi": "Bánh su kem vani"
  },
  "regularPrice": 36000,
  "salePrice": 0,
  "variants": [
    {
      "id": "1-6",
      "price": 199000,
      "salePrice": 189000
    }
  ]
}
```

6.5 Trang nội dung: `pages.read`

Các phần của website có thể hiện thành trang trong app (Giới thiệu, Tin tức, Khuyến mãi...). `kind` là `single` (một bài duy nhất) hoặc `list` (danh sách bài). `key` chỉ gồm chữ thường và dấu gạch ngang. `count = 0` nghĩa là chưa có gì để hiện, app sẽ ẩn mục đó.

YÊU CẦU	VIỆC LÀM
<code>GET /v1/pages</code>	Danh sách các mục.
<code>GET /v1/pages/{key}</code>	Bài duy nhất của mục <code>single</code> , kèm <code>contents</code> (HTML).
<code>GET /v1/pages/{key}/articles?page=&limit=</code>	Danh sách bài của mục <code>list</code> , không kèm <code>contents</code> .
<code>GET /v1/pages/{key}/articles/{id}</code>	Một bài kèm <code>contents</code> .

JSON

```
{
  "items": [
    {
      "key": "about",
      "kind": "single",
      "names": {
        "vi": "Giới thiệu",
        "en": "About us",
        "kr": "회사 소개"
      },
      "count": 1
    },
    {
      "key": "news",
      "kind": "list",
      "names": {
        "vi": "Tin tức",
        "en": "News",
        "kr": "뉴스"
      },
      "count": 2
    },
    {
      "key": "promotion",
      "kind": "list",
      "names": {
        "vi": "Khuyến mãi",
        "en": "Promotions",
        "kr": "프로모션"
      },
      "count": 1
    }
  ]
}
```

JSON

```
{
  "id": "1",
  "names": {
    "vi": "Khai trương chi nhánh mới",
    "en": "Khai trương chi nhánh mới",
    "kr": "Khai trương chi nhánh mới"
  },
  "summaries": {
    "vi": "Chúng tôi vừa mở thêm một cửa hàng.",
    "en": "Chúng tôi vừa mở thêm một cửa hàng.",
    "kr": "Chúng tôi vừa mở thêm một cửa hàng."
  },
  "image": "upload/news/n2.png",
  "publishedAt": "2026-10-12T04:08:45.000Z",
  "contents": {
    "vi": "<p>Nội dung bài viết mẫu.</p>",
    "en": "<p>Nội dung bài viết mẫu.</p>",
    "kr": "<p>Nội dung bài viết mẫu.</p>"
  }
}
```

`image` của bài viết là **đường dẫn trên website** (ví dụ `upload/news/x.jpg`), khác với ảnh sản phẩm chỉ là tên file; MARZI ghép với "Địa chỉ website" ở bảng điều khiển.

6.6 Đăng nhập website bằng mã QR: `auth.qr`

Người mua hoặc chủ cửa hàng đã đăng nhập trong app MARZI có thể **đăng nhập trang web của khách bằng cách quét một mã QR** hiện trên trang đăng nhập (giống đăng nhập web của Zalo, WhatsApp hay Naver). Phần "tạo phiên đăng nhập" nằm trên website của bạn; MARZI chỉ báo lại "người này đã đồng ý".

1 Trang đăng nhập của website

tạo một `sid` ngẫu nhiên (ít nhất 128 bit, 16 đến 128 ký tự `A-Z a-z 0-9 - _`) và một `token` riêng chỉ trình duyệt đó biết. Trang hiện mã QR chứa `marzi://login?shop=<mã cửa hàng>&sid=<sid>` và bắt đầu hỏi website "đã có ai đồng ý chưa?" bằng `token` (cứ 1 đến 2 giây).

2 Người dùng quét mã

bằng app MARZI (mục "Đăng nhập website bằng mã QR" trong Tài khoản hoặc Cài đặt).

3 MARZI hỏi website

`GET /v1/qr-login/{sid}` để biết mã còn hiệu lực không và ai đang đăng nhập (trình duyệt, địa chỉ IP, giờ). App hiện thông tin đó cho người dùng kiểm tra.

4 Người dùng bấm Đăng nhập.

MARZI gọi `POST /v1/qr-login/{sid}/approve` kèm `role` và `userId` (chính là `memberId` hoặc `staffId` website đã trả lúc đăng nhập). Bấm Từ chối thì MARZI gọi `.../reject`.

5 Website đánh dấu phiên đã đồng ý

, tiêu hủy `sid` (chỉ dùng một lần).

6 Lần hỏi tiếp theo của trang

(kèm `token`) thấy "đã đồng ý": website **tự tạo phiên đăng nhập** cho trình duyệt đó bằng cơ chế của website (cookie, session...).

Mã trong QR có dạng:

JSON

```
{
  "note": "what the QR code says",
  "link": "marzi://login?shop=mau-demo&sid=r-MGJd0ElohtJMPzHHH29xTaUGudxGbX"
}
```

GET /v1/qr-login/{sid} : mô tả mã (404 nếu không có, đã hết hạn, đã dùng hoặc bị từ chối). Việc xem **không được** làm ai đăng nhập.

JSON

```
{
  "sid": "r-MGJd0ElohtJMPzHHH29xTaUGudxGbX",
  "expiresAt": "2026-10-17T05:38:14.969Z",
  "site": "Cửa hàng Mẫu",
  "requester": {
    "ip": "127.0.0.1",
    "device": "Trình duyệt trên máy tính",
    "place": null,
    "startedAt": "2026-10-17T05:36:14.969Z"
  },
  "roles": [
    "member",
    "staff"
  ]
}
```

TRƯỜNG	Ý NGHĨA
<code>expiresAt</code>	Hết hạn (ISO 8601 UTC). Nên chỉ khoảng 2 phút.
<code>site</code>	Tên website hiện trên màn xác nhận.
<code>requester</code>	Ai đang mở trang đăng nhập: <code>ip</code> , <code>device</code> (ví dụ "Chrome trên Windows"), <code>place</code> (nơi, nếu biết), <code>startedAt</code> . App hiện nguyên văn để người dùng nhận ra có đúng là mình không. Trường không biết thì trả <code>null</code> .
<code>roles</code>	Loại tài khoản trang này nhận: <code>member</code> (người mua), <code>staff</code> (quản trị). Trang đăng nhập khách chỉ để <code>["member"]</code> ; trang quản trị chỉ để <code>["staff"]</code> .

POST /v1/qr-login/{sid}/approve :

JSON

```
{
  "role": "member",
  "userId": "1"
}
```

JSON

```
{  
  "ok": true  
}
```

Mã không còn: **404**. Không muốn cho tài khoản này đăng nhập bằng cách này (ví dụ quản trị bị cấm): **422** kèm lời giải thích; MARZI hiện lời đó cho người dùng.

Điều bắt buộc để không bị chiếm phiên

- **Trình duyệt nhận phiên bằng token riêng** (không có trong mã QR). Nếu để ai biết **sid** cũng nhận được phiên, một người lạ nhìn thấy mã QR có thể đăng nhập thay người đã quét.
- **sid ngẫu nhiên, dùng một lần, ngắn hạn** (khoảng 2 phút). Sau khi đồng ý, từ chối hoặc hết hạn: xóa.
- **Trả thật requester** (địa chỉ IP và trình duyệt của người mở trang). Đây là thứ duy nhất giúp người dùng phát hiện một trang giả mạo hiện hộ mã QR của bạn rồi dụ họ quét.
- **Chỉ tin lời gọi có chữ ký MARZI** (mục 4). Mọi đường dẫn `/v1/qr-login/...` cũng phải qua bước kiểm chữ ký như các đường dẫn khác.
- **Giới hạn tạo mã** theo địa chỉ IP trên trang đăng nhập (chống tạo hàng loạt) và giới hạn số lần hỏi trạng thái.
- **Tài khoản quản trị** có quyền cao: cân nhắc chỉ cho đăng nhập bằng mã QR nếu quản trị đó có thêm bước xác nhận, hoặc chỉ dùng cho người mua.

Website **bmweb v2** không cần tự làm: bộ cài `marzi-bmweb-connector.zip` có sẵn trang `qr-login.php` và các đường dẫn trên (xem mục "Đăng nhập website bằng mã QR" trong `README.md` của gói).

Trang đăng nhập mẫu của bản mẫu chạy tại `http://127.0.0.1:8083/demo/login` (mở trong trình duyệt, quét bằng app đã nối cửa hàng `mau-demo`). Mã nguồn nằm trong `server.js` (phần "QR sign-in") và `demo/login.html`.

Việc của app và MARZI

App chỉ hiện nút "Đăng nhập website bằng mã QR" khi website khai báo `auth.qr`. MARZI chặn mã của cửa hàng khác, giới hạn 20 lần thử mỗi phút cho mỗi tài khoản, và luôn dùng danh tính trong token của người dùng (app không thể nói "đăng nhập tôi thành người khác"). Website không cần làm gì thêm cho các việc này.

6.7 Đơn từ sàn MARZI (không cần làm thêm gì)

Sàn Marzi Market lấy sản phẩm của cửa hàng qua **đúng những đường dẫn đã có** (`products.read` , `categories.read`) và, với cửa hàng đã ký thỏa thuận phí sàn, **gửi đơn qua** `POST /v1/members/{memberId}/orders` (khả năng `orders.create`) như một đơn của một thành viên. Không có đường dẫn hay trường mới.

- **Thành viên nhận đơn sàn:** cửa hàng tạo trên website một thành viên riêng; chủ hệ thống MARZI nhập **mã thành viên** đó (`memberId`) vào thỏa thuận. MARZI đặt đơn dưới thành viên ấy. Họ tên, số điện thoại và địa chỉ giao hàng trong đơn là của **người mua thật**, không phải của thành viên.
- **Ghi chú của đơn** bắt đầu bằng `[Sàn MARZI MK-12345678]` . Đơn người mua đã chuyển khoản vào tài khoản của MARZI có thêm chữ `- ĐÃ THANH TOÁN qua MARZI, KHÔNG thu tiền khi giao` trong ngoặc: website nên hiển thị ghi chú này cho nhân viên và không thu tiền lần nữa khi giao.
- **Phương thức thanh toán:** MARZI chọn trong `GET /v1/payment-methods` phương thức có tên chứa "tiền mặt", "khi nhận hàng", "COD" hoặc "cash"; không có thì lấy phương thức đầu tiên. Nên có sẵn một phương thức như vậy.
- **Website tự tính giá, phí giao, giảm giá** như mọi đơn; MARZI lấy con số website trả về làm tiền hàng khi tính phí sàn. Đơn bị từ chối (hết hàng, lỗi 422...) thì MARZI giữ đơn lại và ghi lý do, cửa hàng không bị mất khách.
- **Trạng thái:** người mua trên sàn xem trạng thái đơn qua `GET /v1/members/{memberId}/orders/{id}` (khả năng `orders.read`) với cùng thành viên; hãy cập nhật trạng thái đơn trên website như bình thường.
- **Cờ "đã thanh toán":** khi người mua thanh toán bằng mã QR (hoặc chuyển khoản trong app cửa hàng), MARZI **chưa** ghi cờ đã thanh toán về website. Khả năng này sẽ được thêm sau dưới dạng tùy chọn, không làm hỏng connector đã cài.

7. Mã lỗi

Khi lỗi, trả mã HTTP phù hợp và JSON `{"error": "mã_ngắn", "message": "giải thích"}` :

MÃ	KHI NÀO	MARZI LÀM GÌ
200	Thành công.	Dùng dữ liệu.
400	Body POST không phải JSON hợp lệ.	Báo lỗi tạm thời.
401	Chữ ký sai/thiếu/quá cũ (<code>unauthorized</code>), hoặc sai tên đăng nhập/mật khẩu ở <code>auth/verify</code> và <code>staff/verify</code> (<code>invalid_credentials</code>).	Với hai đường dẫn đăng nhập: coi là "sai mật khẩu". Với đường dẫn khác: báo cửa hàng tạm thời không dùng được.
403	Tài khoản không đủ quyền (<code>forbidden</code>).	Hiện <code>message</code> cho chủ cửa hàng.
404	Không có thứ được hỏi, hoặc là của người khác (<code>not_found</code>).	Coi là "không tồn tại".
422	Yêu cầu không hợp lệ về nghiệp vụ (hết hàng, trạng thái lạ, giá sai...).	Hiện <code>message</code> nguyên văn cho người dùng, nên viết bằng tiếng Việt để hiểu.
501	Tính năng ghi chưa bật.	Báo "cửa hàng tạm thời không dùng được". Vì vậy đừng khai báo khả năng ghi (<code>orders.create</code> , <code>staff.*.write</code>) khi chưa bật.
5xx , quá 10 giây, JSON hỏng	Lỗi của website.	Báo "cửa hàng tạm thời không dùng được".

JSON

```
{
  "error": "unauthorized",
  "message": "Bad signature"
}
```

Đừng lẫn hai loại 401

Vì MARZI coi mọi 401 ở đường dẫn đăng nhập là "sai mật khẩu", một connector ký sai khóa sẽ khiến mọi người dùng thấy "sai mật khẩu". Khi gặp tình huống này, đừng kiểm tài khoản trước: dùng nút

Thử kết nối hoặc bộ kiểm tra (mục 8) để thấy lỗi chữ ký.

8. Kiểm thử

1 Chạy bản mẫu để xem đúng chuẩn trông thế nào.

BASH

```
cd mau
MARZI_SECRET=khoa-thu-nghiem-dai-it-nhat-16-ky-tu node server.js
# mở cổng 8083, dữ liệu giả, tài khoản: khachmau / chushop (mật khẩu Mau@12345)
```

2 Chạy bộ kiểm tra tự động

(chỉ cần Node.js 18 trở lên, không cài gì thêm). Nó gọi connector đúng như MARZI gọi, kiểm chữ ký, kiểm hình dạng JSON từng đường dẫn, và **chỉ đọc**, không đặt đơn hay sửa gì.

BASH

```
node check.js https://www.tenmien.com/marzi-connector KHOA_BI_MAT \
--member tai-khoan-nguoi-mua:mat-khau --staff tai-khoan-quan-tri:mat-khau
```

Kết quả mẫu (rút gọn):

```
Chữ ký và an toàn
[đạt] Từ chối yêu cầu ký sai khóa
[đạt] Từ chối yêu cầu quá cũ (chống phát lại)
[đạt] Từ chối yêu cầu không có chữ ký
Sản phẩm (products.read)
[đạt] GET /v1/products
[LỖI] GET /v1/products/{id} - product.price: là string ("35000"), cần number
Kết quả: 24 đạt, 1 lỗi.
```

3 Nhập vào bảng điều khiển MARZI

(mục "Nguồn dữ liệu" của khách hàng): địa chỉ connector, khóa bí mật, bấm **Thử kết nối**. Thành công sẽ liệt kê các khả năng website cung cấp.

4 Thử bằng tay trong app

những việc bộ kiểm tra không tự làm vì có ghi dữ liệu: đặt một đơn thử, đổi trạng thái đơn, đánh dấu đã thanh toán, bật/tắt và sửa giá một sản phẩm thử.

Với website thật, nên tạo **tài khoản thử riêng** cho người mua và cho quản trị để dùng khi kiểm tra.

9. Lỗi thường gặp

TRIỆU CHỨNG	NGUYÊN NHÂN THƯỜNG GẶP
"Website từ chối khóa bí mật" trong bảng điều khiển	Khóa hai bên khác nhau (thừa dấu cách/xuống dòng khi dán); connector nằm trong thư mục con mà chưa cắt tiền tố; đồng hồ máy chủ lệch quá 5 phút; ký trên query đã bị sắp xếp lại; hash body sau khi đã dựng lại JSON.
Bộ kiểm tra báo "chuyển hướng 301/302"	Địa chỉ nhập là <code>http://</code> trong khi website ép <code>https://</code> , hoặc thiếu/thừa <code>www</code> , hoặc thiếu dấu <code>/</code> cuối. Dùng đúng địa chỉ cuối cùng.
"không phải JSON hợp lệ"	PHP/ASP in cảnh báo hay lỗi ra trước JSON; file có BOM; trang lỗi HTML của website; ký tự lạ khi xuất tiếng Việt (cần UTF-8).
"là string, cần number"	Giá hoặc số lượng bị trả thành chuỗi (<code>"35000"</code>). Trả số. <code>id</code> thì ngược lại, phải là chuỗi.
Gọi từ máy bạn được, từ MARZI thì không	Tường lửa, Cloudflare ("Under Attack", kiểm tra bot) hay plugin bảo mật chặn lời gọi giữa hai máy chủ. Cho phép các đường dẫn <code>/v1/...</code> của connector đi qua. Hãy hỏi chủ hệ thống MARZI địa chỉ IP máy chủ khi cần đưa vào danh sách cho phép.
Ảnh sản phẩm không hiện	Connector trả cả đường dẫn thay vì chỉ tên file; hoặc "Thư mục ảnh" ở bảng điều khiển sai; hoặc ảnh chỉ mở được khi đã đăng nhập website.
Chậm hoặc báo tạm thời không dùng được	Truy vấn không có chỉ mục; không giới hạn <code>limit</code> ; trả quá 5 MB; quá 10 giây.
Giờ đơn lệch nhiều tiếng	Trả giờ địa phương không kèm múi giờ. Dùng ISO 8601 UTC (kết thúc bằng <code>Z</code>).

10. Bảo mật cho connector

- **Khóa bí mật không nằm trong thư mục web công khai** và không đưa vào git. Đặt trong file cấu hình ngoài thư mục gốc web hoặc biến môi trường. Không ghi khóa, mật khẩu hay nội dung body đăng

nhập vào log.

- **Kiểm chữ ký trước tiên, cho mọi yêu cầu**, kể cả đọc sản phẩm. Từ chối mọi thứ khác với 401.
- **Giá do website quyết định**. Khi nhận đơn, tự tra giá, quy cách, tồn kho từ dữ liệu của website. Không tin giá hay tổng tiền từ bên gọi.
- **Tách quyền cơ sở dữ liệu**. Dùng tài khoản **chỉ đọc** cho các đường dẫn đọc, và một tài khoản riêng quyền tối thiểu cho việc ghi đơn (như connector của VIETFOOD đang làm). Dùng câu lệnh có tham số, không ghép chuỗi SQL.
- **Chỉ trả đúng phần được phép**: người mua chỉ thấy đơn của chính mình (đơn người khác: 404); danh sách khách hàng che số điện thoại; sản phẩm ẩn không lộ cho người mua.
- **Không để lộ chi tiết lỗi** (đường dẫn file, câu SQL, stack trace). Ghi vào log nội bộ và trả **500** với thông báo chung.
- **Chỉ HTTPS**, giữ chứng chỉ còn hạn. Cập nhật PHP/Node/.NET/Python và thư viện đều đặn.
- Khi đổi khóa bí mật: đổi ở **cả hai nơi** cùng lúc (website và bảng điều khiển MARZI, nút "Đổi địa chỉ / khóa").

11. Tải về

FILE	NỘI DUNG
openapi.yaml	Đặc tả OpenAPI 3 của toàn bộ chuẩn (27 đường dẫn, 29 thao tác). Dùng để sinh khung máy chủ (ví dụ openapi-generator cho ASP.NET, FastAPI, Express) hoặc để xem trong Swagger/Postman. Đã được đối chiếu với bản mẫu: mọi câu trả lời của bản mẫu đều khớp đặc tả.
mau/server.js	Connector mẫu bằng Node.js (không cần thư viện), dữ liệu giả: sản phẩm, đơn, khách hàng, trang nội dung. Đọc mã để xem cách kiểm chữ ký, định tuyến và trả lời.
mau/check.js	Bộ kiểm tra tự động (mục 8), có cả phần đăng nhập bằng mã QR.
mau/demo/login.html	Trang đăng nhập mẫu có mã QR (dùng thư viện tạo QR qrcode.js, giấy phép MIT).
sign.js · sign.py · sign.php · MarziSignature.cs	Hàm ký và kiểm chữ ký bằng Node.js, Python, PHP, C#.
mau/test-vector.json	Giá trị mẫu để thử hàm ký của bạn (mục 4).
marzi-connector-mau.zip	Tất cả các file trên trong một gói.
marzi-bmweb-connector.zip	Connector PHP dựng sẵn cho website bmweb (PHP 7.0, 7.4, 8.x; bmweb v1 và v2): chép lên website, điền một tệp cấu hình, khai báo trong bảng điều khiển. Có trang tự kiểm tra (<code>selftest.php</code>) và hướng dẫn cài đặt tiếng Việt (<code>README.md</code>) trong gói.

Về hợp đồng

Chuẩn hiện tại là phiên bản **1** (khai báo ở trường `contract` của `/v1/capabilities`). Hãy cứ khai báo đúng danh sách khả năng website làm được; khả năng lạ MARZI sẽ bỏ qua.