MUHAMMET ŞAFAK — VISUALS

The 7 layers of a predictable API

An API that never surprises its client is the consistency of seven small decisions.

Muhammet Şafak • October 3, 2026



FLOW

Four stops

01 **Resource and status code** Layers 1 and 2

02 **List endpoints** Layer 3: pagination, filtering, sorting

03 **Response and error** Layers 4 and 5

04 **Version and contract** Layers 6 and 7



SEVEN LAYERS

Every layer is a consistency decision

Resource The URL is noun-based and plural; the HTTP method carries the action.

List page, per_page, sort, order: the same names on every endpoint.

Error Errors come back as error.code, error.message, error.details.

OpenAPI The contract is written before the code.

Status Success is 2xx, client error is 4xx, server error is 5xx.

Envelope Every response has the same skeleton: success, data, message, errors.

Version A breaking change calls for a new version.

The method carries the action, not the URL

METHOD	MEANING	EXAMPLE
GET	**Read**	`GET /users/42`
POST	**Create**	`POST /users`
PUT	**Replace entirely**	`PUT /users/42`
PATCH	**Update partially**	`PATCH /users/42`
DELETE	**Delete**	`DELETE /users/42`

LAYER 2 · STATUS CODE

The status code tells the outcome

CLASS	CODES	MEANING
2xx	200 · 201 · 204	Succeeded; created; no content
4xx	400 · 401 · 403 · 404 · 422	Bad data, identity, permission, not found, validation
5xx	500	Unexpected error on the server side

LAYER 3 · PAGINATION

The identity of a page

2

current_page

last_page: 15

20

per_page

Upper limit: min($adet, 100)

21–40

from – to

Records on this page

287

total

Total records

EXAMPLE FROM THE SOURCE · OFFSET

Deeper pages, costlier OFFSET.

To reach page ten thousand, the database has to read through and skip 200,000 records; one solution is cursor pagination.

LAYER 4 · RESPONSE ENVELOPE

Every response has the same skeleton

The client reads the same four fields on every endpoint: `success`, `data`, `message`, `errors`.

response.json

```
{
"success": true,
"data": {},
"message": null,
"errors": null
}
```

LAYER 4 · RESPONSE ENVELOPE

Lists use the same envelope

A paginated list comes back as `data.items` and `data.meta`; the envelope does not change.

response.json

```
{
"success": true,
"data": {
  "items": [],
  "meta": {
    "current_page": 1,
    "per_page": 15,
    "total": 243,
    "last_page": 17
  }
},
"message": null,
"errors": null
}
```

LAYER 5 · ERROR CONTRACT

An error is a contract too

`code` is fixed for machines, `message` can change for humans, `details` carries extra context.

error.json

```
{
"error": {
  "code": "VALIDATION_FAILED",
  "message": "Gönderilen veriler doğrulanamadı.",
  "details": [
    {
      "field": "email",
      "message": "Geçerli bir e-posta adresi giriniz."
    },
    {
      "field": "phone",
      "message": "Telefon numarası zorunludur."
    }
  ]
}
}
```

LAYERS 4 AND 5

Envelope and error rules

ERROR WITH 200 — Returning 200 + "error": true for errors works but goes against REST; caching and logging break.

FORMAT — Error codes in uppercase with underscores: QUOTA_EXCEEDED.

CODES — New codes can be added, old codes should not be removed.

SCALE — In a small service with a single client, a status code and a short message are enough.

Three strategies, URI preferred

STRATEGY	EXAMPLE	TRADE-OFF
URI	`GET /api/v1/users`	Clearly visible in logs and proxies; multiplies URLs
Header	`X-API-Version: 2`	URLs stay clean; a missing header creates ambiguity
Query	`GET /api/users?version=2`	Practical at the start; proxies and CDNs may not see it

At a basic level: which change calls for a new version?

Compatible

- Adding a new field to the response.
- Adding an optional parameter.
- Enriching error messages.

Breaking

- Removing or renaming an existing field.
- Adding a required parameter.
- Changing the response structure or the status code.

DEPRECATION

Give at least 6–12 months' notice.

For an API open to external clients, a deprecation period of at least 6-12 months is reasonable.

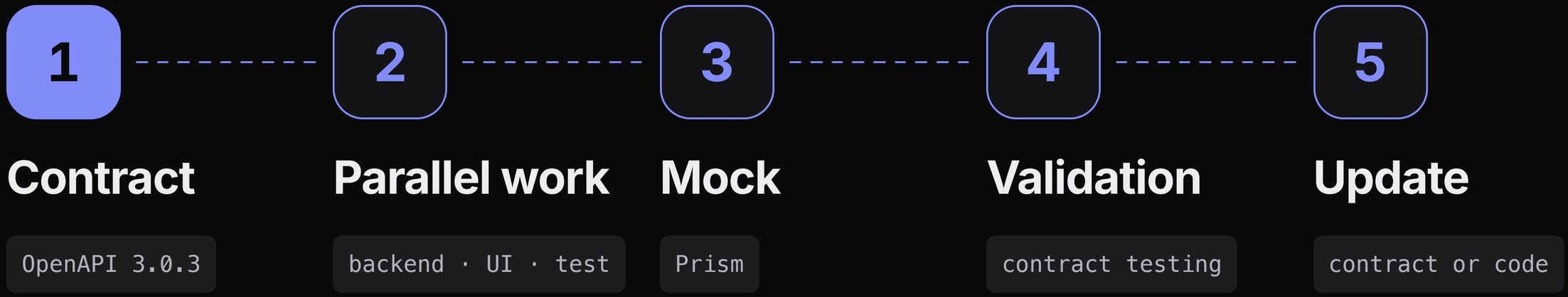
LAYER 6 · RFC 8594 (2019) · RFC 9745 (2025)

Announce a version's lifespan with a header

HEADER	DOCUMENT	WHAT IT SAYS
`Deprecation: @1780272000`	RFC 9745	When it was deprecated; the value is a date, true is invalid
`Sunset: Sat, 31 Oct 2026 23:59:59 GMT`	RFC 8594	When it will stop responding
`Link: <…/docs/migrate-v2>; rel="deprecation"`	RFC 9745	The address of the migration guide

LAYER 7 · CONTRACT-FIRST

Contract first, code later



LAYER 7 · OPENAPI

The contract is written before the code

The schema excerpt of the example contract. The `/orders` endpoint restricts the `status` filter to `pending`, `completed`, `cancelled`.

openapi.yaml

```yaml
components:
schemas:
  Order:
    type: object
    required: [id, status, total]
    properties:
      id:
        type: integer
      status:
        type: string
      total:
        type: number
```

LAYER 7

When does contract-first pay off?

SMALL PROJECT	One developer, small project: contract-first can be overkill.
MANY CONSUMERS	Web, mobile, third parties: it definitely saves time.
LONG LIFE	In an API that will live for years, the contract is a decision record.

Thank you

muhammetsafak.com

Seven layers, seven promises of consistency. The details are in the six source posts.

