

HTTPS et CORS

HTTPS : non négociable

En HTTP, tout circule en clair : l'URL, les headers, le body. Sur le Wi-Fi de l'école, n'importe qui avec Wireshark peut lire votre clé d'API, votre token ou votre mot de passe.

HTTPS, c'est HTTP dans un tunnel **TLS**, qui apporte :

- la **confidentialité** : personne ne peut lire les échanges ;
- l'**intégrité** : personne ne peut les modifier en route ;
- l'**authentification du serveur** : le certificat prouve que vous parlez bien à `api-ritif.example`.

Avec Let's Encrypt, un certificat est gratuit et se renouvelle automatiquement. Il n'y a plus aucune excuse.

Pour une API, ne redirigez pas HTTP vers HTTPS : **refusez** les requêtes HTTP. Quand la redirection arrive, la clé d'API est déjà passée en clair, et le client ne s'en rendra jamais compte.

CORS

La politique de même origine

Par sécurité, un navigateur interdit au JavaScript d'une page de lire les réponses d'une autre **origine**. L'origine, c'est le trio **protocole + domaine + port** :

Page	Appelle	Même origine ?
<code>https://bar.example</code>	<code>https://bar.example/v1/cocktails</code>	☐
<code>https://bar.example</code>	<code>https://api.bar.example/v1/cocktails</code>	☐ domaine différent
<code>http://localhost:8080</code>	<code>http://localhost:3000/v1/cocktails</code>	☐ port différent

Page	Appelle	Même origine ?
http://bar.example	https://bar.example/v1/cocktails	☐ protocole différent

Sans cette règle, n'importe quel site que vous visitez pourrait appeler, avec vos cookies, l'API de votre banque.

Le mécanisme CORS

CORS (*Cross-Origin Resource Sharing*) permet à une API d'**autoriser** certaines origines. Le navigateur envoie l'origine de la page dans le header `Origin`, et l'API répond si elle l'accepte :

```
GET /v1/cocktails HTTP/1.1
Origin: http://localhost:5173

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:5173
```

Pour les requêtes « non simples » (un `DELETE`, du JSON, un header `X-API-Key`...), le navigateur envoie d'abord une requête **preflight** en `OPTIONS` pour demander la permission :

```
OPTIONS /v1/orders HTTP/1.1
Origin: http://localhost:5173
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type, x-api-key

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:5173
Access-Control-Allow-Methods: GET, POST, DELETE
Access-Control-Allow-Headers: content-type, x-api-key
```

CORS est une protection **du navigateur**. curl, Bruno ou un script Python ne l'appliquent pas : c'est pour ça que tout marchait jusqu'ici. Et c'est pour ça que CORS ne protège **pas** votre API contre un attaquant : seule l'authentification le fait.

Manip : provoquer une erreur CORS

API-ritif n'autorise qu'une seule origine. Saurez-vous trouver laquelle ?

1. Créez un fichier `index.html` :

```
<!doctype html>
<meta charset="utf-8">
<title>Test CORS</title>
<button id="go">Charger la carte</button>
<pre id="out"></pre>
<script>
  document.getElementById("go").onclick = async () => {
    const out = document.getElementById("out");
    try {
      const res = await fetch("http://api-ritif.portlatch.com/v1/cocktails");
      out.textContent = JSON.stringify(await res.json(), null, 2);
    } catch (e) {
      out.textContent = "Erreur : " + e.message;
    }
  };
</script>
```

2. Servez-le sur le port 8080 : `python3 -m http.server 8080`, ouvrez

<http://localhost:8080> et cliquez. Lisez l'erreur dans la console des DevTools.

3. La même requête avec curl fonctionne. Comparez les headers de réponse avec et sans `Origin` :

```
curl -i http://api-ritif.portlatch.com/v1/cocktails -H "Origin:
http://localhost:8080"
```

4. Indice : les développeurs front qui utilisent Vite connaissent bien le port `5173`. Testez cette origine avec curl, puis servez la page sur ce port.

5. Modifiez le script pour passer une commande (`POST /v1/orders` avec du JSON et votre `X-API-Key`). Dans l'onglet Réseau, repérez la requête `OPTIONS` envoyée avant le `POST`.

Configurer CORS sur son API

```
from fastapi.middleware.cors import CORSMiddleware

app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:5173"],
    allow_methods=["*"],
```

```
allow_headers=["*"],  
)
```

`Access-Control-Allow-Origin: *` autorise tout le monde. C'est acceptable pour une API publique en lecture seule, jamais pour une API qui utilise des cookies.

Revision #4

Created 2 October 2026 20:00:56 by Thibaud FRICHET

Updated 13 October 2026 22:31:38 by Thibaud FRICHET