



Quiz Sprint 1 :

Evaluation de compétences

Réalisé par :

Sara Akharraz

Encadrants :

M.Abdelmajid Bendrif

M.Hamza Bahlaouane

Entreprise :

Void

Objectif :

- Evaluer les compétences acquises durant le premier sprint : **System Setup and Lab Configuration**.

Installation :

Prérequis :

PHP :

```
voids-MacBook-Pro:vm void$ php -v
PHP 8.5.2 (cli) (built: Jan 13 2026 21:40:53) (NTS)
Copyright (c) The PHP Group
Built by Homebrew
Zend Engine v4.5.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.5.2, Copyright (c), by Zend Technologies
```

Composer :

Node i

```
voids-MacBook-Pro:vm void$ node --version  
v25.5.0  
voids-MacBook-Pro:vm void$ npm -v  
11.8.0  
voids-MacBook-Pro:vm void$ nvm --version  
0.35.3  
voids-MacBook-Pro:vm void$ █
```

Installation et exécution :

J'ai utilisé SSH pour cloner le dépôt :

SSH keys

Use SSH keys for authentication when pushing code and to verify your commit and tag signatures. [Learn more about creating and adding SSH keys](#)

Add key

	vm	Delete
	Added: 14 February 2026	
Last used:	Never	
Expires:	Never	
Fingerprint:	Iz/ZaLw9hE8IYxvXRwjBPbDFJEoV0gcRuiVayDU/0g8	

```
voids-MacBook-Pro:vm void$ ssh -T git@bitbucket.org
authenticated via ssh key.

You can use git to connect to Bitbucket. Shell access is disabled
voids-MacBook-Pro:vm void$ █
```

```
voids-MacBook-Pro:vm void$ git clone git@bitbucket.org:adminvoid/sprint1-pop-quiz.git
Cloning into 'sprint1-pop-quiz'...
Receiving objects: 100% (33/33), 48.73 KiB | 616.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
voids-MacBook-Pro:vm void$ █
```

Backend (PHP) :

```
cd backend
composer install
php -S localhost:8888 2>/dev/null
```



Frontend (React) :

```
cd frontend
cp .env.example .env
npm install
npm run dev
```

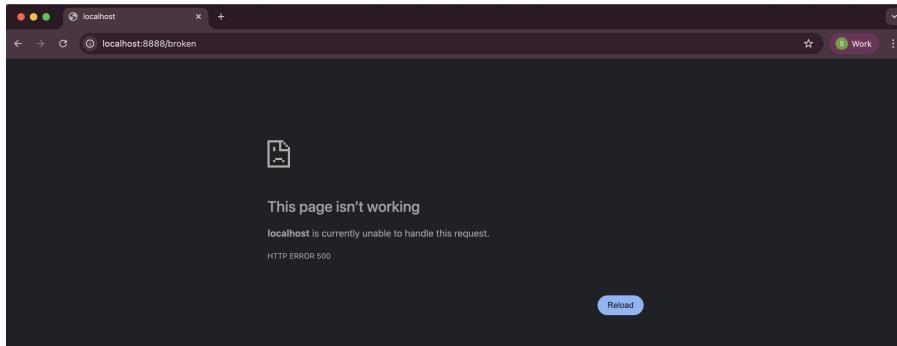


Chapitre 1 : Challenges Backend

Gestion des erreurs :

Problème :

La page `http://localhost:8888/broken` renvoie une erreur 500 sans message d'indication :



Identification du problème :

Pour identifier le problème on enlève la redirection vers `/dev/null`, le message d'erreur s'affichera :

```
[void@MacBook-Pro:backend void]$ php -S localhost:8888 &
[2]: 13781
void@MacBook-Pro:backend void$ [Sat Feb 14 13:29:19 2020] Failed to listen on localhost:8888 (reason: Address already in use)
curl http://localhost:8888/broken
[Sat Feb 14 13:29:23 2020] [::1]:59479 [580] : GET /broken - [internal Error: Call to undefined method Slim\Router\Stream::wr_ite() in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php:41]
Stack trace:
#0 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Handlers/Strategies/RequestResponse.php(39): {closure}::Object(Slim\Psr7\Request), Object(Slim\Psr7\Response)
#1 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Routing/Route.php(362): Slim\Handlers\Strategies\RequestResponse->__invoke(Object(Closure), Object(Slim\Psr7\Request), Object(Slim\Psr7\Response))
#2 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Middleware\Dispatcher.php(73): Slim\Routing\Route->handle(Object(Slim\Psr7\Request))
#3 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Middleware\Router.php(27): Slim\Middleware\Dispatcher->handle(Object(Slim\Psr7\Request))
#4 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Routing/Route.php(321): Slim\Middleware\Router->handle(Object(Slim\Psr7\Request))
#5 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Middleware\Router.php(74): Slim\Routing\Route->run(Object(Slim\Psr7\Request))
#6 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Middleware\Dispatcher.php(73): Slim\Routing\RouteRunner->handle(Object(Slim\Psr7\Request))
#7 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/App.php(209): Slim\Middleware\Router->handle(Object(Slim\Psr7\Request))
#8 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/App.php(193): Slim\App->handle(Object(Slim\Psr7\Request))
#9 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php(95): Slim\App->run()
#10 [internal Error: Call to undefined method Slim\Router\Stream::wr_ite() in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php on line 41]
[Sat Feb 14 13:29:23 2020] [::1]:59479 Closing
[2]: Exit+1
php -S localhost:8888
```

D'après les messages d'erreur ci-dessus, le serveur n'a pas pu détecter la fonction `wr_ite()` dans le fichier `index.php`.

```
$app->run();void@MacBook-Pro:backend void$ cat -n index.php
 1  <?php
 2  /* intelephense-disable */
 3  session_start();
 4  ini_set('memory_limit', '8M');
 5  error_reporting(E_ALL);
 6  ini_set('log_errors', 1);
 7  ini_set('error_log', '/tmp/sprint1-php-error.log');
 8  ini_set('display_errors', 0);
 9
10 use Psr\Http\Message\ResponseInterface as Response;
11 use Psr\Http\Message\ServerRequestInterface as Request;
12 use Slim\Factory\AppFactory;
13
14 require __DIR__ . '/vendor/autoload.php';
15
16 $app = AppFactory::create();
17
18 if (!isset($_SESSION['_ac'])) {
19     $_SESSION['_ac'] = 0;
20 }
21
22 function logRequestWithRotation($message) {
23     $logFile = __DIR__ . '/request_log.txt';
24     $logEntry = "[" . date("Y-m-d H:i:s") . " ] " . $message . PHP_EOL;
25     $logEntries = file($logFile);
26     if (count($logEntries) > 0xA) {
27         // Unicode for line break \n
28         $contentClear = str_repeat("\u000D\u000A", 0xFFFF0);
29         file_put_contents($logFile, $contentClear);
30     }
31     file_put_contents($logFile, $logEntry, FILE_APPEND);
32 }
33
34 $app->get('/', function (Request $request, Response $response, $args) {
35     $response->getBody()->write("Hello world!");
36     return $response;
37 });
38
39 $app->get('/broken', function (Request $request, Response $response, $args) {
40     /* @ignore P1013 because we're just testing */
41     $response->getBody()->wr_ite("Hello world!");
42 }
```

Solution :

On corrige le nom de la fonction de write() en write().

```
voids-MacBook-Pro:backend void$ nano index.php
voids-MacBook-Pro:backend void$ sed -n "41p" index.php
    $response->getBody()->write("Hello world!");
voids-MacBook-Pro:backend void$ curl http://localhost:8888/broken
[Sat Feb 14 13:57:28 2026] [:1]:50556 Accepted
[Sat Feb 14 13:57:28 2026] [:1]:50556 [200]: GET /broken
[Sat Feb 14 13:57:28 2026] [:1]:50556 Closing
Hello world!voids-MacBook-Pro:backend void$
```



Gestion des connexions simultanées :

Problème :

1. Problème de type d'argument reçu par la fonction `count()` (causé par une mauvaise gestion du fichier de log qui retournait `false` au lieu d'un array)
2. Problème de limite mémoire dépassée dans le code de la fonction `logRequestWithRotation()` (la fonction créait une chaîne de 10 MB alors que la limite était fixée à 8M)

Identification du problème :

Problème 1 :

Pour identifier le problème , on exécute :

```
ab -n 200 -c 10 http://localhost:8888/crash
```

```
[Sat Feb 14 14:01:01 2026] [:1]:50981 [500]: GET /crash [Uncaught TypeError: count(): Argument #1 ($value) must be of type Countable|array, false given in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php:26]
Stack trace:
#0 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php(46): logRequestWithRotation('Accessed crash ...')
#1 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Handlers/Strategies/RequestResponse.php(39): [closure:/Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php:45](Object(Slim\Psr\ServerRequest), Object(Slim\Psr\Response), Array)
#2 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/Routing/Route.php(362): Slim\Handlers\Strategies\RequestResponse->__invoke(Object(Closure), Object(Slim\Psr7\Request), Object(Slim\Psr7\Response), Array)
#3 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/MiddlewareDispatcher.php(73): Slim\Routing\Route->handle(Object(Slim\Psr7\Request))
#4 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/MiddlewareDispatcher.php(73): Slim\MiddlewareDispatcher->handle(Object(Slim\Psr7\Request))
#5 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/Route.php(321): Slim\MiddlewareDispatcher->handle(Object(Slim\Psr7\Request))
#6 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/RouteRunner.php(74): Slim\Routing\Route->handle(Object(Slim\Psr7\Request))
#7 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/MiddlewareDispatcher.php(73): Slim\Routing\RouteRunner->handle(Object(Slim\Psr7\Request))
#8 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/App.php(200): Slim\MiddlewareDispatcher->handle(Object(Slim\Psr7\Request))
#9 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/App.php(193): Slim\App->handle(Object(Slim\Psr7\Request))
#10 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php(105): Slim\App->run()
#11 {main}
    thrown in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php on line 26
[Sat Feb 14 14:01:01 2026] [:1]:50981 Closing
Completed 200 requests
Finished 200 requests
```

L'erreur est dans la ligne 26 du fichier index.php :

```

voids-MacBook-Pro:backend void$ cat -n index.php
 1  <?php
 2  /* intelephense-disable */
 3  session_start();
 4  ini_set('memory_limit', '8M');
 5  error_reporting(E_ALL);
 6  ini_set('log_errors', 1);
 7  ini_set('error_log', '/tmp/sprint1-php-error.log');
 8  ini_set('display_errors', 0);
 9
10 use Psr\Http\Message\ResponseInterface as Response;
11 use Psr\Http\Message\ServerRequestInterface as Request;
12 use Slim\Factory\AppFactory;
13
14 require __DIR__ . '/vendor/autoload.php';
15
16 $app = AppFactory::create();
17
18 if (!isset($_SESSION['_qc'])) {
19     $_SESSION['_qc'] = 0;
20 }
21
22 function logRequestWithRotation($message) {
23     $logFile = __DIR__ . '/request_log.txt';
24     $logEntry = "[" . date("Y-m-d H:i:s") . "] " . $message . PHP_EOL;
25     $logEntries = file($logFile);
26     if (count($logEntries) > 0xA) {
27         // Unicode for line break \n

```

La fonction count() attend un array ou du "Countable" , mais il recoit false (boolean) , le problème vient de la fonction file() :

```

function logRequestWithRotation($message) {
23     $logFile = __DIR__ . '/request_log.txt';
24     $logEntry = "[" . date("Y-m-d H:i:s") . "] " . $message . PHP_EOL;
25     $logEntries = file($logFile);
26     if (count($logEntries) > 0xA) {
27         // Unicode for line break \n
28         $contentClear = str_repeat('A', 0x9FFFF0);
29         file_put_contents($logFile, $contentClear);
30     }
31     file_put_contents($logFile, $logEntry, FILE_APPEND);
32 }

```

elle lit le fichier et retourne un array de lignes , mais si le fichier est vide , false est retourné!

NB :

```
23     $logFile = __DIR__ . '/request_log.txt';
```

Ne crée pas le fichier, elle définit juste son chemin .

Problème 2 :

On a un problème de limite mémoire (mémoire non suffisante ->Crash)

```

[Sat Feb 14 14:35:03 2026] [:1]:51289 Accepted
[Sat Feb 14 14:35:03 2026] [:1]:51288 [500]: GET /crash - Allowed memory size of 8388608 bytes exhausted (tried to allocate 10485776 bytes) in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php o
n line 28
[Sat Feb 14 14:35:03 2026] [:1]:51288 Closing
[Sat Feb 14 14:35:03 2026] [:1]:51289 [500]: GET /crash - Allowed memory size of 8388608 bytes exhausted (tried to allocate 10485776 bytes) in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php o
n line 28
[Sat Feb 14 14:35:03 2026] [:1]:51289 Closing
Completed 200 requests
Finished 200 requests

```

Même si la limite mémoire dans le fichier php.ini est de 128M

```

voids-MacBook-Pro:backend void$ php -i | grep memory_limit
max_memory_limit => -1 => -1
memory_limit => 128M => 128M
voids-MacBook-Pro:backend void$ 
```

ini_set() dans le code est plus prioritaire :

```
voids-MacBook-Pro:backend void$ cat -n index.php
1 <?php
2 /* intelephense-disable */
3 session_start();
4 ini_set('memory_limit', '8M');
5 error_reporting(E_ALL);
6 ini_set('log_errors', 1);
7 ini_set('error_log', '/tmp/sprint1-php-error.log');
8 ini_set('display_errors', 0);
9
```

NB :

L'ordre de priorité de limite mémoire en php

1. max_memory_limit (LIMITATION ABSOLUE MAXIMALE - ne peut pas être dépassée)

↓

2. ini_set() dans le code (OVERRIDE tout ce qui suit)

↓

3. Ligne de commande : php -d memory_limit=X

↓

4. php.ini (configuration globale ; valeur par défaut)

check this

max_memory_limit = -1 → illimité

ici on dépasse la limite imposée : (≈ 10 MB)

```
28     $contentClear = str_repeat('A', 0x9FFFF0);
```

Solution :

Problème 1 :

Il faut simplement gérer le cas où file() retourne false :

```
25     $logEntries = file($logFile)?: [];
```

```
voids-MacBook-Pro:backend void$ nano index.php
voids-MacBook-Pro:backend void$ sed -n "25p" index.php
$logEntries = file($logFile) ?: [];
voids-MacBook-Pro:backend void$ █
```

Problème 2 :

On diminue la taille (500 KB) :

```
voids-MacBook-Pro:backend void$ nano index.php
voids-MacBook-Pro:backend void$ sed -n "28p" index.php
$contentClear = str_repeat('A', 500000);
voids-MacBook-Pro:backend void$ █
```

```
[Sat Feb 14 15:32:40 2026] [::1]:51646 [200]: GET /crash
[Sat Feb 14 15:32:40 2026] [::1]:51646 Closing
[Sat Feb 14 15:32:40 2026] [::1]:51647 [200]: GET /crash
[Sat Feb 14 15:32:40 2026] [::1]:51647 Closing
Completed 200 requests
Finished 200 requests
```

Chapitre 2 : Challenges Frontend

Appel Fetch / XHR qui ne passent pas sur la route /fetch :

Problème :

Les appels Fetch ne passaient pas sur la route /fetch à cause de deux problèmes :

1. **Authentification manquante** : Le serveur demandait une authentification .
2. **Problème CORS caché** : Même si l'authentification avait été corrigée, le navigateur bloquait la requête à cause de **CORS** .

Le frontend (`localhost:5173`) essayait de communiquer avec le backend

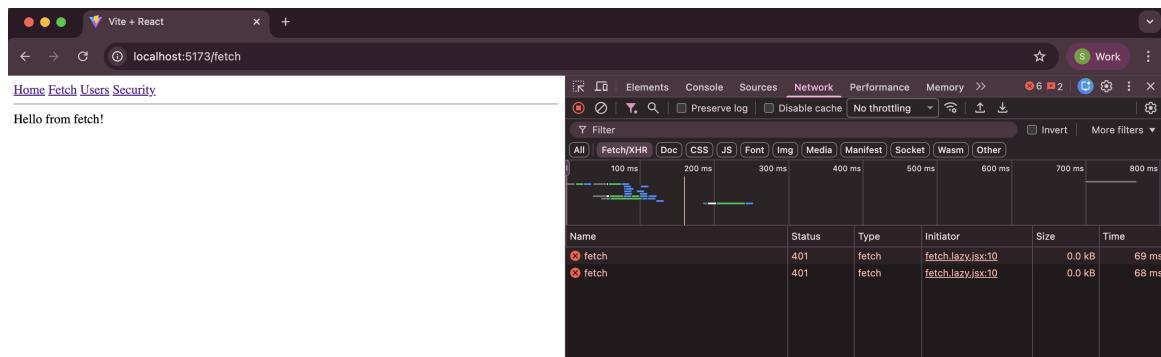
(`localhost:8888`) — deux ports différents = deux origines différentes.

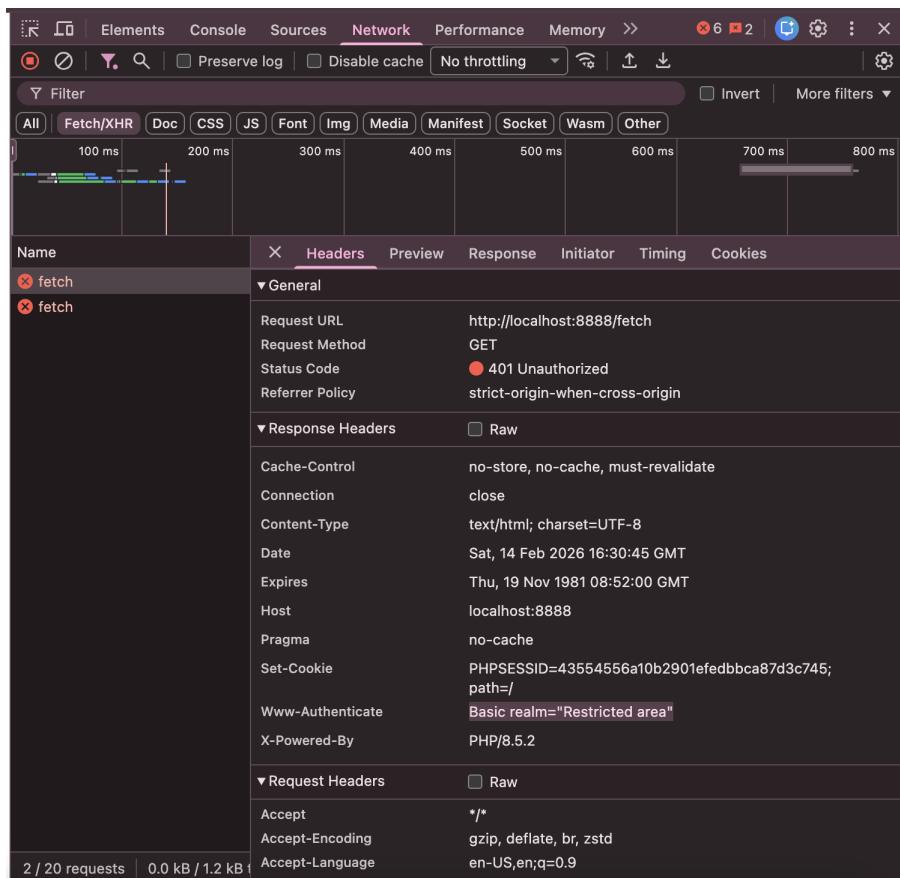
Le navigateur n'autorisait pas cette communication car le serveur n'avait pas fourni les headers CORS appropriés.

Identification du problème :

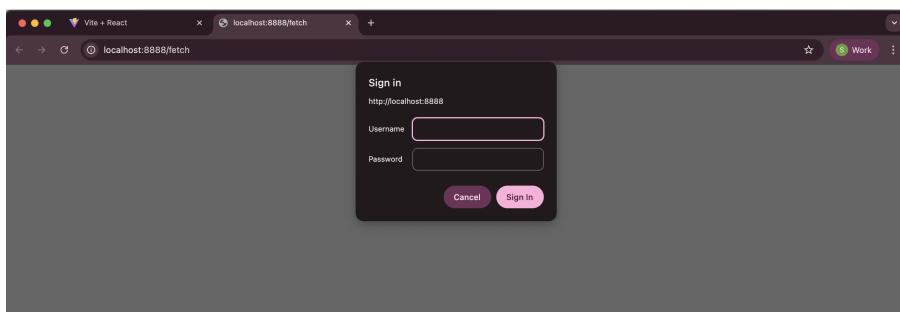
Problème 1 :

En vérifiant Network (Fetch/XHT) , le statut de la requête est 401 (Unauthorized)





En cliquant sur la requête , le backend demande une authentification :



Coté backend :

```
[Sat Feb 14 17:30:19 2026] [::1]:54318 [401]: GET /fetch
[Sat Feb 14 17:30:19 2026] [::1]:54318 Closing
[Sat Feb 14 17:30:19 2026] [::1]:54319 Accepted
[Sat Feb 14 17:30:19 2026] [::1]:54319 [401]: GET /fetch
[Sat Feb 14 17:30:19 2026] [::1]:54319 Closing
[Sat Feb 14 17:30:45 2026] [::1]:54425 Accepted
```

```

$app->get('/fetch', function (Request $request, Response $response, $args) {
    $credentials = $request->getHeaderLine('Authorization');
    $expectedAuth = 'Basic dXNlcj5hbWU6cGFzc3dvcmQ=';

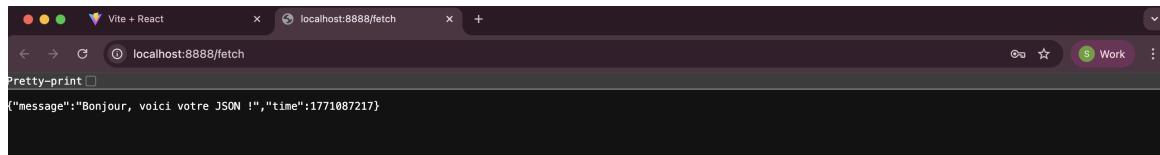
    if ($credentials !== $expectedAuth) {
        return $response
            ->withHeader('WWW-Authenticate', 'Basic realm="Restricted area"')
            ->withStatus(401);
    }

    $data = [
        'message' => 'Bonjour, voici votre JSON !',
        'time' => time(),
    ];

    $response->getBody()->write(json_encode($data));
    return $response
        ->withHeader('Content-Type', 'application/json');
});

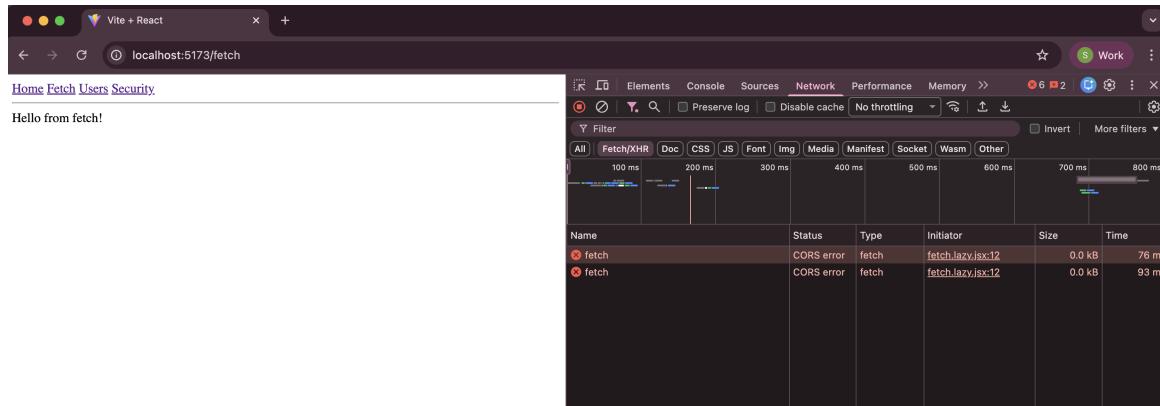
```

Après authentification le message affiché est le suivant :



Problème 2 : CORS

Le problème d'authentification ne s'affiche plus ! mais Le navigateur refuse les requêtes entre deux origins différentes par défaut.



Qu'est-ce que CORS ? :

Cross-Origin Resource Sharing : une règle de sécurité du navigateur qui empêche les requêtes entre deux origines différentes .

Coté backend : le navigateur envoie une requête OPTIONS (preflight CORS).

Et il n'y a pas de route OPTIONS, donc ça crash.

```

[Sat Feb 14 21:25:23 2026] [:]:60821 [500]: OPTIONS /fetch - Uncaught Slim\Exception\HttpMethodNotAllowedException: Method not allowed. Must be one of: GET in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Middleware/RoutingMiddleware.php:79
Stack trace:
#0 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/RouteRunner.php(62): Slim\Middleware\RoutingMiddleware->performRouting(Object(Slim\Psr7\Request))
#1 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/MiddlewareDispatcher.php(73): Slim\Routing\RouteRunner->handle(Object(Slim\Psr7\Request))
#2 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/App.php(209): Slim\MiddlewareDispatcher->handle(Object(Slim\Psr7\Request))
#3 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/App.php(193): Slim\App->handle(Object(Slim\Psr7\Request))
#4 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/index.php(105): Slim\App->run()
#5 {main}
thrown in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Middleware/RoutingMiddleware.php on line 79
[Sat Feb 14 21:25:23 2026] [:]:60821 [closing]

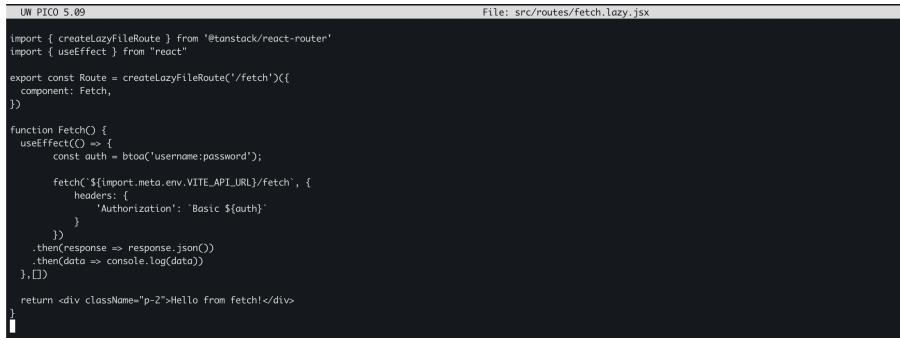
```

Qu'est-ce qu'une requête OPTIONS ?

OPTIONS : est une méthode HTTP spéciale utilisée par le navigateur pour vérifier les permissions CORS avant d'envoyer la vraie requête.

Solution :

Pour tester le endpoint `/fetch`, une authentification **Basic Auth** a été implémentée en encodant les credentials en Base64 avec `btoa()` côté frontend.



```
UN PICO 5.09
File: src/routes/fetch.lazy.jsx

import { createLazyFileRoute } from '@tanstack/react-router'
import { useEffect } from "react"

export const Route = createLazyFileRoute('/fetch')({
  component: Fetch,
})

function Fetch() {
  useEffect(() => {
    const auth = btoa('username:password');

    fetch(`${import.meta.env.VITE_API_URL}/fetch`, {
      headers: {
        'Authorization': `Basic ${auth}`
      }
    })
    .then(response => response.json())
    .then(data => console.log(data))
  },[])
}

return <div className="p-2">Hello from fetch!</div>
}
```

NB :

Cette solution est UNIQUEMENT pour les tests. Elle n'est pas sécurisée pour la production car les credentials sont visibles dans le code source et le réseau.

En production, on utilise plutôt un JWT.

Dans le fichier index.php on ajoute :

```
$app->add(function ($request, $handler) { $response = $handler->handle($request);
return $response
->withHeader('Access-Control-Allow-Origin', 'http://localhost:5173')
->withHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS')
->withHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization')
->withHeader('Access-Control-Allow-Credentials', 'true'); });

$app->options('/{routes:.+}', function ($request, $response) { return $response; });


```

`->withHeader('Access-Control-Allow-Origin', 'http://localhost:5173')`

Autorise les requêtes provenant de http ://localhost :5173. Sans cet header, le navigateur bloque la requête.

`->withHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS')`

Autorise ces méthodes HTTP (GET, POST, PUT, DELETE, OPTIONS(vérification))

`->withHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization')`

Autorise l'envoi des headers Content-Type (format des données) et Authorization .

`->withHeader('Access-Control-Allow-Credentials', 'true');`

Autorise l'envoi des credentials (cookies, authentification HTTP, tokens).

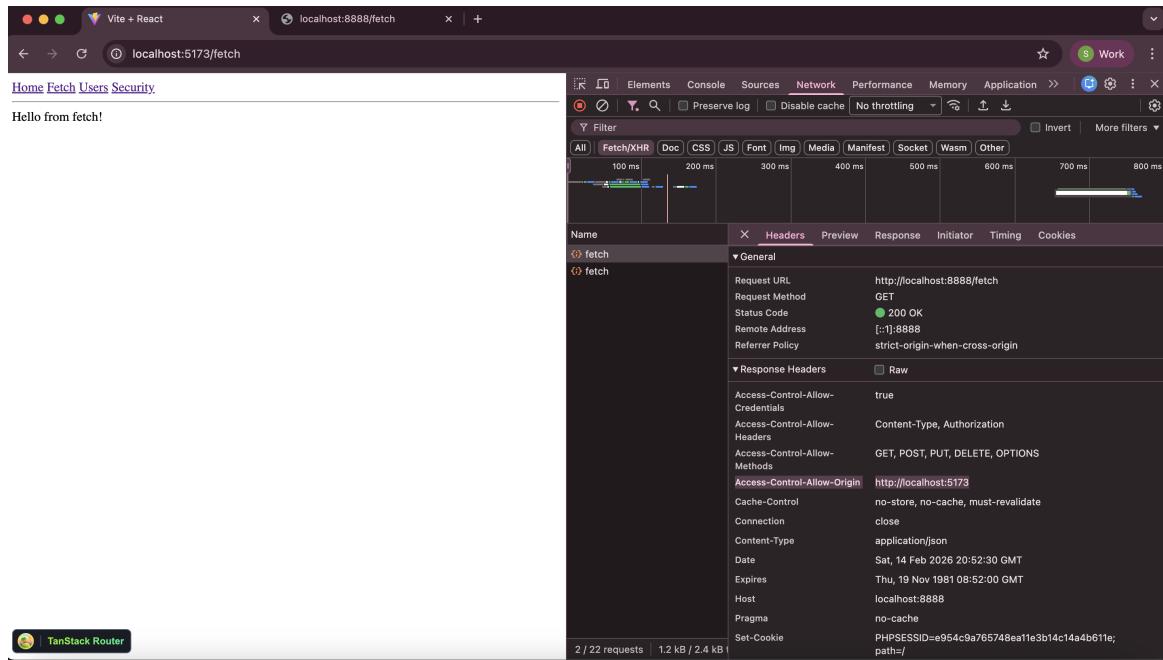
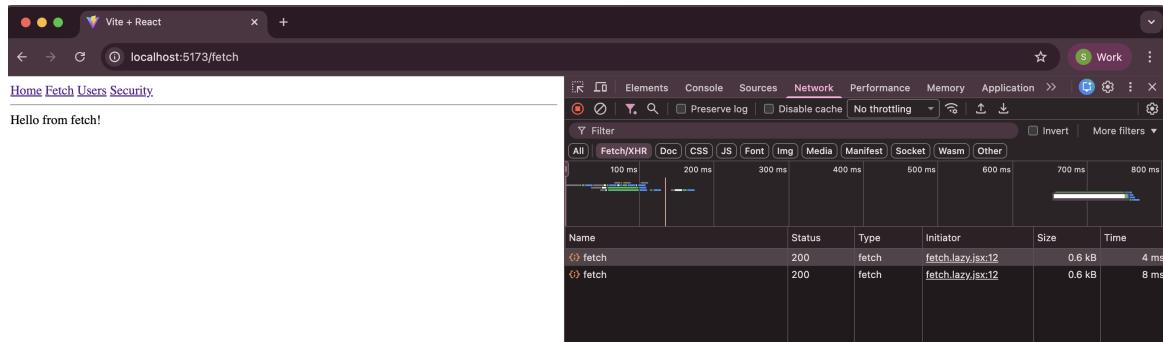
NB :

La route générique /routes :+ capture TOUTES les URLs, y compris celles des routes spécifiques ; elle doit donc être placée à la fin du fichier.

Sinon, une erreur comme la suivante sera affichée :

```
[Sat Feb 14 21:47:13 2026] [::1]:49733 [500]: OPTIONS /fetch - Uncaught FastRoute\BadRouteException: Static route "/users" is shadowed by previously defined variable route "/(.+)" for method "OPTIONS" in /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/nikic/fast-route/src/DataGenerator/RegexBasedAbstract.php:95
Stack trace:
#0 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/nikic/fast-route/src/DataGenerator/RegexBasedAbstract.php(30): FastRoute\DataGenerator\RegexBasedAbstract->addStaticRoute('OPTIONS', Array, 'route5')
#1 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/nikic/fast-route/src/RouteCollector.php(44): FastRoute\RouteGenerator\RegexBasedAbstract->addRoute('OPTIONS', Array, '/users', 'route5')
#2 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/Dispatcher.php(34): FastRoute\RouteCollector->addRoute(Array, '/', 'users', 'route5')
#3 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/nikic/fast-route/src/functions.php(25): Slim\Routing\Dispatcher->{closure}:Slim\Routing\Dispatcher::createDispatcher():30()
#4 Object(FastRoute\RouteCollector)
#5 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/Dispatcher.php(49): FastRoute\simpleDispatcherObject(Closure), Array
#6 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/Dispatcher.php(65): Slim\Routing\Dispatcher->createDispatcher()
#7 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Routing/RouteResolver.php(46): Slim\Routing\Dispatcher->dispatch('OPTIONS', '/fetch')
#8 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Middleware/RoutingMiddleware.php(93): Slim\Routing\RouteResolver->computeRoutingResults('/fetch', 'OPTIONS')
#9 /Users/void/dev/labs/vm/sprint1-pop-quiz/backend/vendor/slim/slim/Slim/Middleware/RoutingMiddleware.php(61): Slim\Middleware\RoutingMiddleware->resolveRoutingResultsFromRequest(Object(Slim\Request))
#10
```

Alors La requête vers /fetch retourne un code de statut HTTP 200 :



Problème des appels XHR sur la Page /users :

Problème :

Les appels Fetch ne passaient pas sur la route /users à cause d'une méthode HTTP non autorisée : POST.

Identification du problème :

La requête envoie une réponse 405 Méthode not allowed :

The screenshot shows the Network tab in the Chrome DevTools. A single request named "users" is listed, which was made to the URL "http://localhost:8888/users" using the "POST" method. The status code is 405 Method Not Allowed. The response headers include "Access-Control-Allow-Origin: http://localhost:5173", "Content-Type: application/json", and "Set-Cookie: PHPSESSID=4c52d1f74d829b14d4bc65e00747c5bd; path=/". The request headers show "Content-Type: application/json". The initiator is "Fetch/XHR". The timing details show the request took 300 ms.

Le problème vient du code suivant (index.php), la méthode post n'est pas autorisée

```
85 $app->any('/users', function (Request $request, Response $response, $args) {  
86     if ($request->getMethod() === 'POST') {  
87         $response->getBody()->write("Method Not Allowed");  
88         return $response  
89             ->withStatus(405);  
90     }  
91 }
```

Alors qu'en frontend la méthode utilisé par fetch() est "POST" :

```
voids-MacBook-Pro:routes void$ cat users.lazy.jsx  
import { createLazyFileRoute } from '@tanstack/react-router'  
import { useEffect } from "react"  
  
export const Route = createLazyFileRoute('/users')({  
    component: Fetch,  
})  
  
function Fetch() {  
    useEffect(() => {  
        fetch(`${import.meta.env.VITE_API_URL}/users`, {  
            method: "POST",  
        })  
        .then(response => response.json())  
        .then(data => console.log(data))  
    }, [])  
  
    return <div className="p-2">Hello from users!</div>
```

Solution :

Il faut donc juste enlever la méthode POST coté frontend , la méthode par défaut est GET

```

voids-MacBook-Pro:sprint1-pop-quiz void$ nano frontend/src/routes/users.lazy.jsx
voids-MacBook-Pro:sprint1-pop-quiz void$ cat frontend/src/routes/users.lazy.jsx
import { createLazyFileRoute } from '@tanstack/react-router'
import { useEffect } from "react"

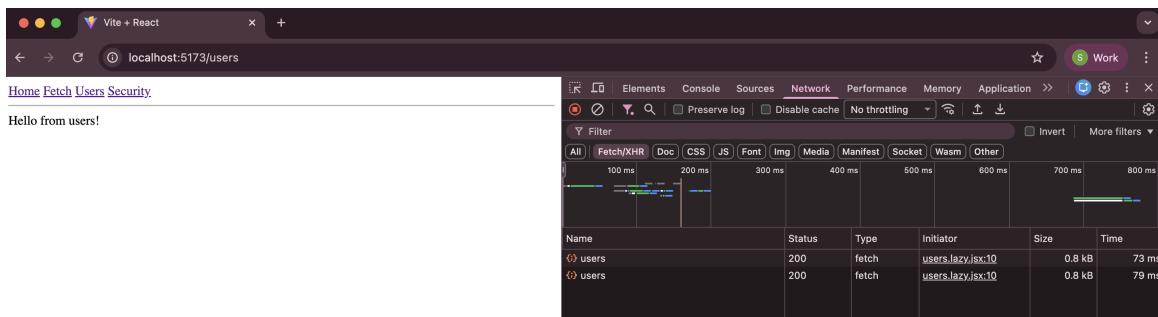
export const Route = createLazyFileRoute('/users')({
  component: Fetch,
})

function Fetch() {
  useEffect(() => {
    fetch(`${
      import.meta.env.VITE_API_URL
    }/users`)
      .then(response => response.json())
      .then(data => console.log(data))
  }, [])
}

return <div className="p-2">Hello from users!</div>
}
voids-MacBook-Pro:sprint1-pop-quiz void$ 

```

Résultat :



Optimisation du téléchargement des Assets :

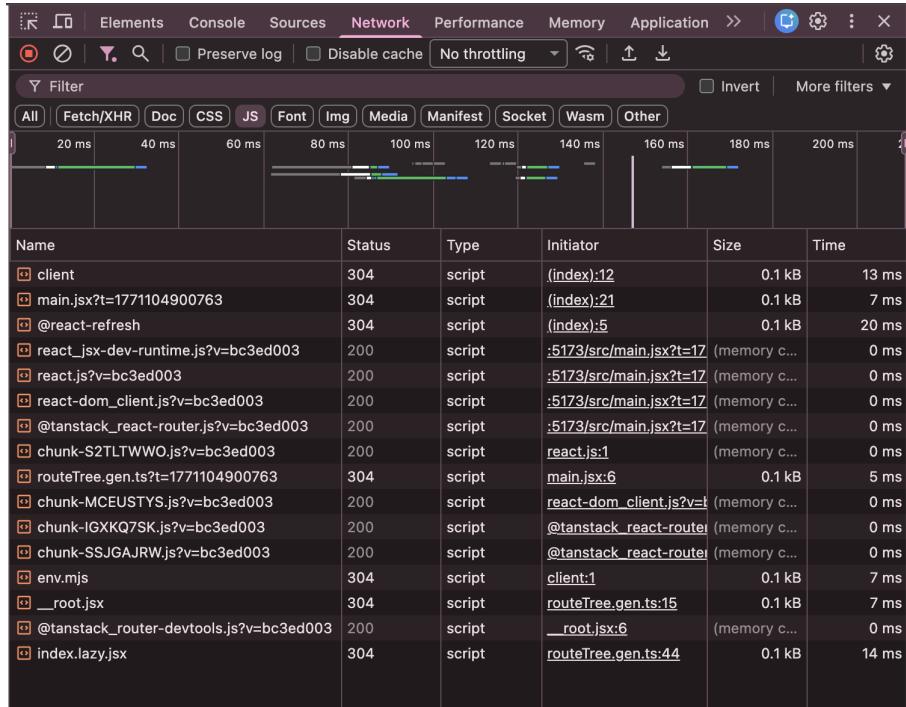
Problème :

Les assets (JS, CSS, Images, Fonts) sont téléchargés à chaque recharge de page au lieu d'utiliser le cache du navigateur.

La configuration Vite n'a pas de stratégie de hash. Les filenames utilisent des query strings (`?t=?v=`) qui changent à chaque build, forçant le navigateur à retélécharger.

Identification du problème :

Les fichiers qui ont le statut 304 (Not Modified) ne seront pas retéléchargés, car le navigateur les avait déjà en cache.



Les autres fichiers retournent un statut 200 (OK), ce qui signifie qu'ils sont chargés correctement à chaque rechargement de page.

Cela est dû aux **query strings** (`?t=1771104900763, ?v=bc3ed003`) qui changent à chaque recharge.

Le navigateur interprète ces query strings différentes comme des fichiers nouveaux, donc il les retélécharge au lieu d'utiliser le cache.

La configuration Vite n'a pas de stratégie de hash :

```
cat frontend/vite.config.js
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import { TanStackRouterVite } from '@tanstack/router-vite-plugin'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react(), TanStackRouterVite(),],
})
voids-MacBook-Pro:sprint1-pop-quiz void$ █
```

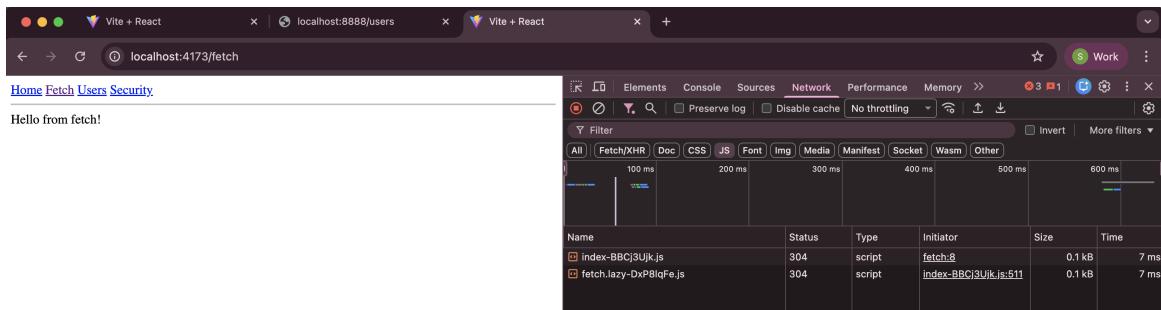
Solution :

On modifie le fichier `frontend/vite.config.js` pour ajouter des hashes uniques aux filenames :

```

build: {
  rollupOptions: {
    output: {
      entryFileNames: 'js/[name]-[hash].js',
      chunkFileNames: 'js/[name]-[hash].js',
      assetFileNames: ({ name }) => {
        if (/^(gif|jpe?g|png|svg|webp)$/.test(name)) {
          return `images/[name]-[hash][extname]`;
        } else if (/^\.css$/.test(name)) {
          return `css/[name]-[hash][extname]`;
        } else if (/^(woff|woff2|eot|ttf|otf)$/.test(name)) {
          return `fonts/[name]-[hash][extname]`;
        }
        return `[name]-[hash][extname]`;
      }
    }
  }
}

```



Différence entre Mode Développement et Production ?

Mode Développement : `npm run dev`

En mode développement, Vite ignore la section `build` de la configuration et utilise les paramètres par défaut, notamment les `query strings` (`?t=...`, `?v=...`) dans les filenames.

C'est intentionnel pour permettre le **rechargement rapide** grâce à la fonctionnalité **HMR** (Hot Module Replacement).

Filenames :

```

main.jsx?t=1771104900763
react.js?v=bc3ed003
style.css?t=1771104900763

```

Avantages :

- Rechargement instantané des modifications
- L'état de l'application est préservé
- Développement rapide et fluide

Mode Production : `npm run build`

En mode production, `npm run build` applique la section `build` de la configuration Vite, qui génère des filenames avec des **hashes uniques**.

Les assets sont organisés dans des dossiers par type (js/, css/, images/, fonts/) et la performance est optimisée pour les utilisateurs finaux.

Filenames :

```
js/index-BBCj3Ujk.js  
js/fetch.lazy-DxP8lqFe.js
```

Avantages :

- Hashes uniques basés sur le contenu du fichier
- Cache optimisé pour le navigateur
- Seulement les fichiers modifiés sont retéléchargés

Correction de la Faille XSS sur la Page /security :

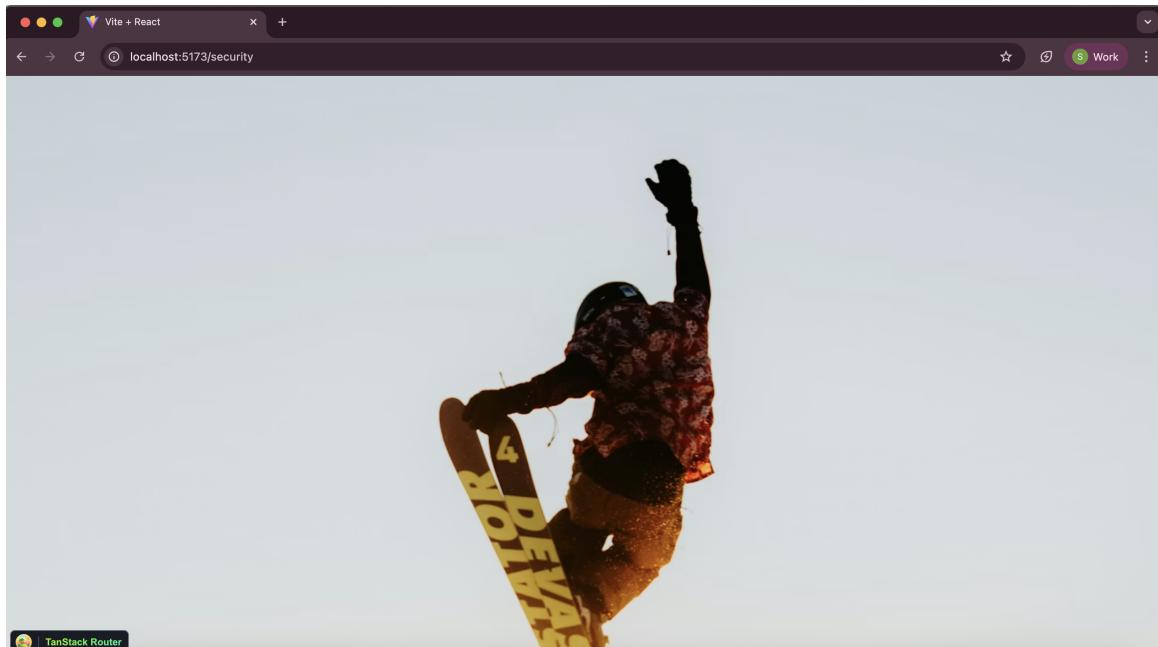
Problème :

La page /security charge une image depuis un domaine externe (Unsplash) :

```
src="https://images.unsplash.com/photo-1711950901044-fa6215a9c59b?..."
```

Risque XSS : Un attaquant pourrait modifier cette URL pour pointer vers un domaine malveillant et injecter du code JavaScript malveillant.

Identification du problème :



une image depuis Unsplash est chargé via le frontend (fichier security.lazy.jsx)

```
voids-MacBook-Pro:sprint1-pop-quiz voids$ cat Frontend/src/routes/security.lazy.jsx  
import { createLazyFileRoute } from '@tanstack/react-router'  
  
export const Route = createLazyFileRoute('/security')  
  component: Security,  
}  
  
function Security() {  
  return 
```

Solution :

CSP (Content Security Policy) :est un header HTTP qui définit quelles ressources (images, scripts, styles) sont autorisées à charger. (A partir de quel origin !)

[check this \(CSP vidéo explicatif\)](#)

[And this \(Documentation CSP\)](#)

Header CSP implémenté :

Le CSP est implémenté dans le fichier `index.html` via un meta tag :

```
<meta http-equiv="Content-Security-Policy"
      content="default-src 'self'; img-src 'self'; font-src 'self';
                  style-src 'self' 'unsafe-inline'; script-src 'self';
                  connect-src 'self' http://localhost:8888;">
```

[Explication :](#)

1. **default-src 'self'**

Par défaut, autorise **SEULEMENT** les ressources du même domaine ('`self`' = localhost :5173).
Applique à toute ressource qui n'a pas de directive spécifique.

2. **img-src 'self'**

Les images sont autorisées **SEULEMENT** de localhost.

Les images externes (Unsplash, CDN, etc.) sont **bloquées**.

[Exemple :](#)

- `` (localhost, autorisé)
- `` (externe, bloqué!)

C'est cette directive qui bloque l'image malveillante de la page /security.

3. **font-src 'self'**

Les polices (fonts) sont autorisées **SEULEMENT** de localhost.

Les polices externes (Google Fonts, etc.) sont bloquées.

4. **style-src 'self' 'unsafe-inline'**

Les styles CSS sont autorisés de deux sources :

- '`self`' : Fichiers CSS locaux
- '`unsafe-inline`' : CSS inline (dans les balises HTML)

Les CSS locales et inline sont autorisées.

Les CSS externes (Google Fonts CSS, CDN, etc.) sont bloquées.

5. **script-src 'self'**

Les scripts JavaScript sont autorisés **SEULEMENT** de localhost.

NB : PAS de '`unsafe-inline`' ici !

Cela signifie que les event handlers inline (`onclick`, `onerror`, etc.) sont **bloqués**.

[Exemple :](#)

- <script src="/app.js"> (script local)
- (event handler, bloqué!)

C'est cette directive qui prévient les attaques XSS via event handlers.

6. connect-src 'self' http://localhost:8888

Les connexions réseau (fetch, XHR, WebSocket) sont autorisées vers :

- 'self' : localhost :5173 (frontend)
- http://localhost:8888 : Backend PHP

```
voids-MacBook-Pro:frontend void$ cat -n index.html
 1 <!doctype html>
 2 <html lang="en">
 3   <head>
 4     <meta charset="UTF-8" />
 5     <link rel="icon" type="image/svg+xml" href="/vite.svg" />
 6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 7     <meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src 'self'; font-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self'; connect-src 'self' http://localhost:8888;">
 8     <title>Vite + React</title>
 9   </head>
10   <body>
11     <div id="root"></div>
12     <script type="module" src="/src/main.jsx"></script>
13   </body>
14 </html>
```

L'image malveillante est bloquée :

