Pop Quiz - Sprint 1

Stage PFE – Entreprise **VOID**

Stagiaire : Hibat Allah Rguiti

15 février 2026

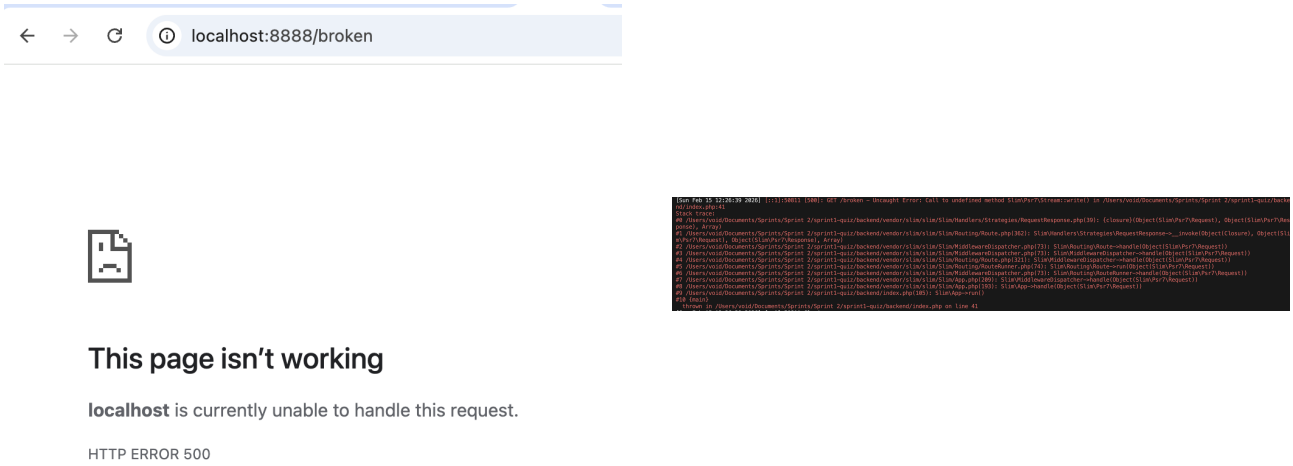
1. Introduction

Ce rapport présente les solutions apportées lors du Sprint 1 - Pop Quiz. L'exercice est divisé en deux chapitres : Backend et Frontend. Chaque problème rencontré est analysé, expliqué et résolu.

2. Chapitre 1 : Backend - Gestion des erreurs et optimisation des performances

2.1. Problème 1 : Page /broken renvoie une erreur 500

L'accès à `http://localhost:8888/broken` provoquait une erreur 500.



Analyse : La route '/broken' contenait une faute de frappe dans la méthode PHP , le mot write contient un caractère Unicode caché :

```
\$response->getBody()->write("Hello world!");
```

Cette ligne générait une erreur fatale.

Solution : Correction de la méthode :

```
$response->getBody()->write("Hello world!");
```

Résultat : La page renvoie désormais correctement "Hello world!" sans erreur 500.

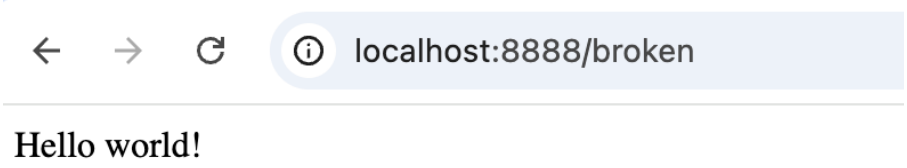
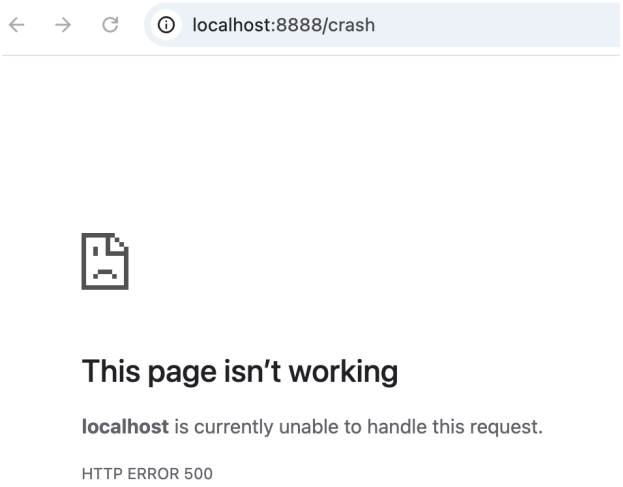


FIGURE 1 – Blocage de l'image externe par la CSP

2.2. Problème 2 : Gestion des connexions simultanées sur /crash

Sous test de charge avec ApacheBench (‘ab -n 200 -c 10‘), le serveur crashait ou avait des lenteurs.



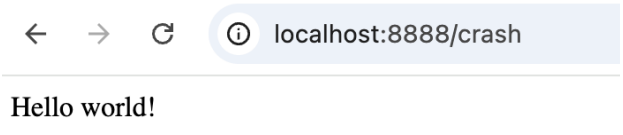

```
void@192 sprint1-quiz % ab -n 200 -c 10 http://localhost:8888/crash

This is ApacheBench, Version 2.3 <$Revision: 1923142 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests


Server Software:
Server Hostname:        localhost
Server Port:            8888

Document Path:          /crash
Document Length:        0 bytes
```

Analyse : La fonction `logRequestWithRotation()` écrivait dans un fichier et tentait de lire un grand nombre de lignes à chaque requête, ce qui surchargeait la mémoire.

Solution : - Limiter la taille du fichier de log avec `filesize` plutôt que de charger toutes les lignes. - Implémentation sécurisée :

```php
if (file_exists($logFile) && filesize($logFile) > 50000) {
    file_put_contents($logFile, "");
}
file_put_contents($logFile, $logEntry, FILE_APPEND);
```

Résultat : Le test ApacheBench montre 0 requêtes échouées et des temps de réponse stables.




```
void@192 sprint1-quiz % ab -n 200 -c 10 http://localhost:8888/crash

This is ApacheBench, Version 2.3 <$Revision: 1923142 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests


Server Software:
Server Hostname:        localhost
Server Port:            8888

Document Path:          /crash
Document Length:        12 bytes

Concurrency Level:      10
Time taken for tests:   0.084 seconds
Complete requests:      200
Failed requests:        0
Total transferred:      68600 bytes
HTML transferred:       2400 bytes
Requests per second:    2393.38 [#/sec] (mean)
Time per request:       4.178 [ms] (mean)
Time per request:       0.418 [ms] (mean, across all concurrent requests)
Transfer rate:          801.69 [Kbytes/sec] received
```

3. Chapitre 2 : Frontend - Résolution de problèmes XHR, optimisation du cache et sécurité

3.1. Problème 1 : Appels XHR sur /fetch bloqués par CORS et 401

La console affichait :

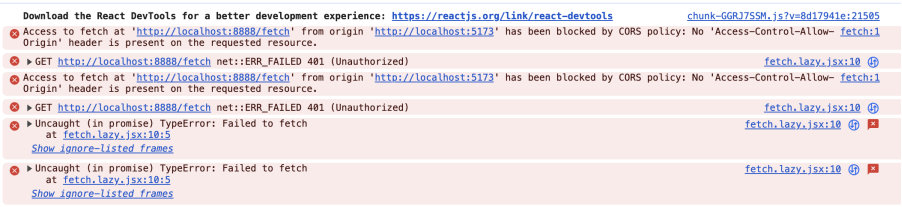


FIGURE 2 – Blocage de l'image externe par la CSP

Analyse : - Le backend nécessite une authentification Basic. - Le navigateur bloque la requête cross-origin car aucun header 'Access-Control-Allow-Origin' n'était envoyé.

Solution : - Ajout de headers CORS dans `index.php` :

```
header('Access-Control-Allow-Origin: http://localhost:5173');
header('Access-Control-Allow-Methods: GET, POST, OPTIONS');
header('Access-Control-Allow-Headers: Authorization, Content-Type');
```

- Gestion de la requête OPTIONS (preflight) :

```
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    http_response_code(200);
    exit;
}
```

De plus, avant certains appels (ex : avec Authorization), le navigateur envoie une requête `OPTIONS` dite *preflight*. Cette requête doit retourner un statut `200 OK`, sinon l'appel principal est refusé.

3.2. Problème 2 : Appels XHR sur /users échouent avec 405

`POST http://localhost:8888/users 405 (Method Not Allowed)`

Analyse : La route '/users' était configurée pour refuser les POST.

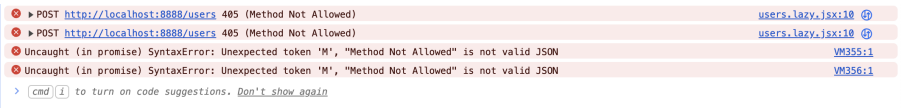


FIGURE 3 – Blocage de l'image externe par la CSP

Solution : Changer l'appel dans le frontend de 'POST' à 'GET'.

3.3. Problème 3 : Optimisation du téléchargement des assets

Observation : Lors des rechargements de page, les fichiers statiques (JavaScript, CSS, images) semblaient être retéléchargés systématiquement.

— Lorsque l'option `Disable cache` est activée dans les DevTools, toutes les requêtes retournent un statut **200 OK**, ce qui signifie que les fichiers sont rechargés entièrement.

@react-refresh	200	script	security:5	62.8 kB	3 ms
env.mjs	200	script	client:1	4.3 kB	4 ms
react_jsx-dev-runtime.js?v=8d17941e	200	script	main.jsx:18	36.2 kB	6 ms
react.js?v=8d17941e	200	script	main.jsx:1	0.5 kB	4 ms
react-dom_client.js?v=8d17941e	200	script	main.jsx:2	1.4 kB	4 ms
@tanstack_react-router.js?v=8d17941e	200	script	main.jsx:3	4.1 kB	7 ms
routeTree.gen.ts?t=1771192203529	200	script	main.jsx:6	5.5 kB	4 ms
chunk-CE4FKTVB.js?v=8d17941e	200	script	react.js?v=8d17941e:3	79.4 kB	3 ms
__root.jsx	200	script	routeTree.gen.ts:15	5.5 kB	2 ms
chunk-GGRJ7SSM.js?v=8d17941e	200	script	react-dom_client.js?v=8d17941e:3	925 kB	11 ms
chunk-J5MTLWTH.js?v=8d17941e	200	script	@tanstack_react-router.js?v=8d17	135 kB	8 ms
chunk-WDSN7AAI.js?v=8d17941e	200	script	@tanstack_react-router.js?v=8d17	36.6 kB	9 ms
@tanstack_router-devtools.js?v=8d17941e	200	script	__root.jsx:2	98.3 kB	7 ms
security.lazy.jsx	200	script	routeTree.gen.ts:34	4.0 kB	3 ms
users.lazy.jsx?t=1771192203529	200	script	routeTree.gen.ts:29	5.0 kB	4 ms
fetch.lazy.jsx?t=1771191946790	200	script	routeTree.gen.ts:39	5.2 kB	3 ms
index.lazy.jsx	200	script	routeTree.gen.ts:44	3.9 kB	3 ms

FIGURE 4 – Blocage de l'image externe par la CSP

— Lorsque le cache du navigateur est activé, certaines requêtes retournent un statut **304 Not Modified**, indiquant que le navigateur réutilise une version déjà stockée localement.

Name	Status	Type	Initiator	Size	Time
client	304	script	fetch:12	0.2 kB	2 ms
main.jsx?t=1771192203529	304	script	fetch:21	0.2 kB	2 ms
react_jsx-dev-runtime.js?v=8d17941e	200	script	main.jsx:1	(memory cache)	0 ms
react.js?v=8d17941e	200	script	main.jsx:2	(memory cache)	0 ms
react-dom_client.js?v=8d17941e	200	script	main.jsx:3	(memory cache)	0 ms
@tanstack_react-router.js?v=8d17941e	200	script	main.jsx:4	(memory cache)	0 ms
@react-refresh	304	script	fetch:5	0.2 kB	1 ms
env.mjs	304	script	client:1	0.2 kB	1 ms
chunk-CE4FKTVB.js?v=8d17941e	200	script	react-dom_client.js:6	(memory cache)	0 ms
routeTree.gen.ts?t=1771192203529	304	script	main.jsx:5	0.2 kB	1 ms
chunk-GGRJ7SSM.js?v=8d17941e	200	script	react-dom_client.js:3	(memory cache)	0 ms
chunk-J5MTLWTH.js?v=8d17941e	200	script	@tanstack_react-router.js:105	(memory cache)	0 ms
chunk-WDSN7AAI.js?v=8d17941e	200	script	@tanstack_react-router.js:106	(memory cache)	0 ms
__root.jsx	304	script	routeTree.gen.ts:15	0.2 kB	2 ms
@tanstack_router-devtools.js?v=8d17941e	200	script	__root.jsx:2	(memory cache)	0 ms
fetch.lazy.jsx?t=1771191946790	304	script	routeTree.gen.ts:39	0.2 kB	2 ms

FIGURE 5 – Blocage de l'image externe par la CSP

Cependant, ce comportement par défaut reste limité car aucun cache explicite n'est défini pour les assets statiques.

Solution : Afin d'optimiser les performances, il est recommandé de configurer un cache côté serveur :

```
1 header('Cache-Control: max-age=31536000, immutable');
```

Explication :

— Les assets peu modifiés (fonts, images, CSS) peuvent être mis en cache longtemps pour éviter des téléchargements inutiles.

— Lors des déploiements quotidiens, les fichiers JavaScript changent souvent : l'utilisation de noms de fichiers avec hash (ex : `app.8fd3.js`) permet de forcer le navigateur à récupérer automatiquement la nouvelle version.

— En environnement de développement, désactiver le cache permet de tester les changements immédiatement.

3.4. Problème 4 : Correction de la faille XSS sur /security

Observation : La page `/security` chargeait une image provenant d'un domaine externe (`unsplash.com`). Même si cela semble inoffensif, cela représente un risque de sécurité réel.

Problème de sécurité Une attaque **XSS (Cross-Site Scripting)** consiste à injecter du contenu malveillant (scripts ou ressources externes) dans une page web. Si un attaquant parvient à modifier une source externe ou à injecter une URL non contrôlée, cela peut permettre :

— l'exécution de code JavaScript malveillant,
— le vol de cookies ou de sessions,
— l'injection de contenu frauduleux.

Solution : mise en place d'une CSP Pour empêcher le chargement de ressources non autorisées, une politique de sécurité des contenus (**CSP**) a été ajoutée côté backend :

```
Content-Security-Policy: default-src 'self'; img-src 'self';
```

Cette directive impose que seules les ressources provenant du domaine local (`localhost`) soient acceptées. Ainsi, toute image ou script externe est automatiquement bloqué par le navigateur.

Résultat : Le navigateur empêche désormais le chargement de toute image qui ne provient pas de l'hôte local, réduisant fortement les risques de XSS.

FIGURE 6 – Blocage d'une image externe grâce à la politique CSP

4. Conclusion

Ce quiz de Sprint 1 a permis de mettre en pratique plusieurs compétences essentielles liées à la configuration et au diagnostic d'une application. :

— identifier et corriger des erreurs backend (HTTP 500) en analysant les logs PHP,
— tester la robustesse d'un serveur avec Apache Benchmark et comprendre l'impact des écritures disque sur les performances,
— résoudre des problèmes de communication frontend/backend liés au CORS et aux requêtes preflight `OPTIONS`,
— comprendre le fonctionnement du cache navigateur (statuts 200 vs 304) et mettre en place une stratégie adaptée pour les assets statiques,
— renforcer la sécurité d'une application web en bloquant les ressources externes via une politique CSP, afin de prévenir les attaques XSS.