Sprint 1 - Pop Quiz

Rapport de Solutions

Sadiki Abdelkarim

1 Backend - PHP

1.1 Erreur 500 sur /broken

Problème: Caractère Unicode invisible (U+200B zero-width space) dans `wr420013ite`.

Solution:

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

1.2 Crash sur /crash

Problème: Allocation de 10MB (`0x9FFFF0` bytes) avec limite de 8MB + lecture inefficace du fichier de log.

Solution optimisée:

```
function logRequestWithRotation($message) {
    $logFile = __DIR__ . '/request_log.txt';
    $logEntry = "[" . date("Y-m-d␣H:i:s") . "]␣" . $message . PHP_EOL;

    //Check if file exists before reading
    if (file_exists($logFile)) {
        $logEntries = file($logFile);
    } else {
        $logEntries = [];
    }

    // If more than 10 lines, clear the file
    if (count($logEntries) > 10) {
        file_put_contents($logFile, ''); // Clear file properly
    }

    // Add new log entry
    file_put_contents($logFile, $logEntry, FILE_APPEND);
}
```

2 Frontend - React

2.1 Appels XHR sur /fetch

Problème: Manque d'authentification Basic Auth requise par le backend. et CORS

Solution:

```
$app->add(function ($request, $handler) {

    $origin = $request->getHeaderLine('Origin');

    // Allow specific origins (edit if needed)
    $allowedOrigins = [
        'http://localhost:5173',
    ];

    if (in_array($origin, $allowedOrigins)) {
        header("Access-Control-Allow-Origin: $origin");
    }

    header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
    header("Access-Control-Allow-Headers: Content-Type, Authorization");
    header("Access-Control-Allow-Credentials: true");

   // Handle preflight (OPTIONS)
    if ($request->getMethod() === 'OPTIONS') {
        $response = new \Slim\Psr7\Response();
        return $response->withStatus(200);
    }

    return $handler->handle($request);
});
```

2.2 Appels XHR sur /users

Problème: Méthode POST refusée (405) alors que GET est attendu.

Solution:

```
fetch(`${import.meta.env.VITE_API_URL}/users`, {
  method: "GET" // Correction: GET au lieu de POST
})
```

2.3 Optimisation Assets

Problème: Assets retéléchargés à chaque reload. Déploiements quotidiens avec changements JS fréquents.

Solution - vite.config.js:

```
export default defineConfig({
  plugins: [react()],
  build: {
    rollupOptions: {
      output: {
        entryFileNames: 'assets/[name].[hash].js',
        chunkFileNames: 'assets/[name].[hash].js',
        assetFileNames: (assetInfo) => {
          const ext = assetInfo.name.split('.').pop();
          if (/png|jpe?g|svg|gif|webp/i.test(ext))
            return `assets/img/[name].[hash][extname]`;
          if (/woff2?|ttf|otf/i.test(ext))
            return `assets/fonts/[name].[hash][extname]`;
          if (/css/i.test(ext))
            return `assets/css/[name].[hash][extname]`;
          return `assets/[name].[hash][extname]`;
        }
      }
    }
  }
})
```

Configuration Cache Headers (PHP):

```
$app->add(function ($request, $handler) {
    $response = $handler->handle($request);
    $path = $request->getUri()->getPath();

    if (preg_match('/\.(js|css)$/', $path)) {
        // 1 jour - deploiement quotidien
        $response = $response->withHeader(
            'Cache-Control',
            'public, max-age=86400, must-revalidate'
        );
    } elseif (preg_match(
        '/\.(woff2?|ttf|otf)$/', $path)) {
        // 1 an - fonts rarement modifies
        $response = $response->withHeader(
            'Cache-Control',
            'public, max-age=31536000, immutable'
        );
    } elseif (preg_match(
        '/\.(png|jpe?g|gif|svg)$/', $path)) {
        // 1 semaine - images
        $response = $response->withHeader(
            'Cache-Control',
            'public, max-age=604800'
        );
    }
    return $response;
});
```

Stratégie:

- JS/CSS: hash + cache 1j (cache busting automatique)
- Fonts: cache 1 an immutable
- Images: cache 1 semaine
- HTML: no-cache

2.4 Faille XSS sur /security

Problème: Image externe (Unsplash) crée un vecteur d'attaque XSS.

Solution - CSP Headers (index.php):

```
$app->add(function ($request, $handler) {
    $response = $handler->handle($request);

    $csp = [
        "default-src␣'self'",
        "script-src␣'self'␣'unsafe-inline'",
        "style-src␣'self'␣'unsafe-inline'",
        "img-src␣'self'␣http://localhost:*
␣␣␣␣␣␣␣␣␣␣␣␣https://localhost:*",
        "font-src␣'self'␣data:",
        "connect-src␣'self'␣http://localhost:*",
        "frame-ancestors␣'none'",
        "base-uri␣'self'",
        "form-action␣'self'"
    ];

    return $response->withHeader(
        'Content-Security-Policy',
        implode('; ', $csp)
    );
});
```

Correction React - security.lazy.jsx:

```
function Security() {
  return (
    <div className="p-4">
      <h2>Security Page</h2>
      <p>Page securisee avec CSP. Images
      externes bloquees.</p>
      <img
        src="/assets/placeholder.jpg"
        alt="Securite"
      />
      <div className="mt-4 p-4 bg-blue-100">
        <strong>Protection CSP:</strong>
        Ressources non autorisees bloquees.
      </div>
    </div>
  );
}
```

Protection CSP:

- `default-src 'self'`: Origine unique par défaut
- `img-src`: Images localhost uniquement
- `frame-ancestors 'none'`: Anti-clickjacking
- `connect-src`: XHR/Fetch vers origine uniquement