RAPPORT DE STAGE

Diplôme BTS – Services Informatiques aux Organisations option SLAM
2023

ARTHUR Félix-Vincent
Du 2 janvier 2023 au 24 février 2023
Tuteur de stage : OLOA Jérôme

Description de l'entreprise :

GSV communication est une entreprise française de communication et de marketing digital qui propose des services de création de sites web, de développement d'applications mobiles, de référencement SEO, de community management et de stratégie digitale.

Établissement/Formation

Lycée Fulbert
BTS – Services Informatiques
aux Organisations option SLAM

Entreprise d'accueil :

AG2R La Mondiale
12 Rue Edmond Poillot,
28000, Chartres
Tel : 09.74.50.12.34

SOMMAIRE

I. Introduction :

A. Résumé

Dans le cadre de mes études en informatique (BTS Services Informatiques aux Organisations) au Lycée Fulbert, je dois effectuer un stage en entreprise d'une durée de 2 mois, du 2 janvier au 24 février 2023. Ce stage me donnera l'opportunité d'appliquer les compétences et les connaissances que j'ai acquises au cours de mon année et demi de formation. Effectuer mon stage au sein d'une entreprise telle que GSV COMMUNICATION me permettra également d'apprendre et de découvrir de nouvelles choses.

B. Présentation de l'entreprise :

GSV Communication est une entreprise française spécialisée dans la communication et le marketing digital. Elle propose des services de

création de sites web, de développement d'applications mobiles, de référencement SEO, de community management et de stratégie digitale. Fondée en janvier 2008, l'entreprise est basée à Chartres et composée d'une équipe de professionnels du web et de la communication. GSV Communication se positionne comme un partenaire stratégique pour les entreprises souhaitant améliorer leur visibilité en ligne et leur présence sur les réseaux sociaux.

Leurs services de création de sites web incluent la conception et le développement de sites web professionnels sur mesure, adaptés aux besoins de leurs clients, ainsi que la création de sites e-commerce et de landing pages. GSV Communication propose également des services de référencement naturel (SEO) pour optimiser le positionnement des sites web de leurs clients sur les moteurs de recherche.

En ce qui concerne le marketing digital, GSV Communication propose des services de community management pour gérer la présence de ses clients sur les réseaux sociaux, ainsi que des services de publicité en ligne tels que le référencement payant (SEA) et la publicité sur les réseaux sociaux. En somme, GSV Communication est une agence de communication numérique qui offre une gamme complète de services pour aider les entreprises à renforcer leur présence en ligne et à améliorer leur visibilité sur le web.

C. Objectifs du stage :

Mon stage à GSV Communication a été divisé en deux parties. La première partie consistait à me former sur un nouveau framework PHP appelé Laminas, tout en travaillant sur de nouvelles technologies telles que jQuery et Ajax. Grâce aux connaissances acquises au cours du premier mois, j'ai pu travailler sur des projets clients en appliquant les compétences acquises durant ma formation.

Au cours de mon travail, j'ai pris en charge les aspects non-développement du projet, tels que la préparation des fichiers CSV, l'importation de données, la création de la base de données, le formatage des fichiers avec le bon encodage, ainsi que d'autres tâches similaires.

D. Outils et technologies utilisés :

Tout au long de mon stage, j’ai utilisé les outils suivants :

- Microsoft Office :
- Adobe Dreamweaver
- Visual Studio Code
- FileZilla
- PuTTY
- PhpMyAdmin

Et les technologies suivantes :

- HTML
- CSS
- JS
- La bibliothèque JS => AJAX
- PHP
- SQL

II. Les missions :

A. Découverte et appropriation de Laminas Framework :

Laminas (anciennement connu sous le nom de Zend Framework) est un framework open source pour le développement d'applications web en PHP. Basé sur une architecture modulaire, il permet aux développeurs de choisir les composants nécessaires à leur application.

Laminas offre un contrôle total sur le projet, sans moteur de template comme TWIG dans le framework Symfony ou ORM comme Doctrine. Cette absence de limitation offre une grande flexibilité pour intégrer des plug-ins et des services selon les besoins.

Grâce à son architecture modulaire MVC, Laminas est utilisé par de grandes entreprises telles que BBC et BNP Paribas. Sa sécurité intégrée et sa documentation complète et de qualité offrent aux développeurs une solution répondant à leurs besoins pour tout projet.

Laminas est conçu pour être extensible et facile à utiliser. Les développeurs peuvent facilement créer des modules et des plugins pour étendre les fonctionnalités de base. En outre, Laminas dispose d'une grande communauté de développeurs qui fournissent un support technique et des ressources en ligne pour les utilisateurs.

En somme, grâce à ses qualités, j'ai pu facilement prendre en main ce nouveau framework.

B. Fonctionnement de Laminas :

Le fonctionnement de Laminas est assez simple à comprendre en effet, un projet Laminas est décomposé en plusieurs parties qu’on appelle module. Chaque module est considéré comme une fonctionnalité et peut être utilisé dans d'autres modules, ce qui permet une grande modularité et une réutilisation facile de code.

Il est également possible d'utiliser des modules tiers déjà développés et disponibles sur internet pour ajouter des fonctionnalités à notre projet sans avoir à les coder nous-mêmes.

Par défaut, un projet Laminas contient le module Application qui représente le site web que l'utilisateur utilisera. Nous pouvons créer d'autres modules pour décomposer notre projet et ajouter des fonctionnalités spécifiques, comme un module Admin pour les administrateurs du site.

Chaque module contient trois dossiers : config, src et views. C'est dans le dossier src que nous allons coder notre fonctionnalité en utilisant les différents composants fournis par Laminas tels que les contrôleurs, les modèles, et les formulaires. Les dossiers config et views contiennent respectivement les fichiers de configuration et les fichiers de vue utilisés par le module.

C. Création d'un projet E-commerce :

1. Création des modules :

Les premières semaines de stage, j'ai suivi les tutoriels de base disponibles sur le site officiel de Laminas. Ces tutoriels m'ont permis de comprendre le fonctionnement de Laminas et de créer mes propres modules.

Mon maître de stage m'a demandé de créer un module "Admin" qui nous permettra de contenir l'ensemble de notre projet. J'ai donc créé un dossier "Admin" et le fichier Module.php. Par défaut, Laminas dispose d'un ModuleManager (ANNEXE 1).

Ce Manager va chercher dans chaque module un fichier nommé Module.php. Dans ce fichier, il faut créer la fonction getConfig() qui va indiquer la configuration du module (ANNEXE 2).

Ensuite, à la racine de notre projet, il faut modifier les informations dans le fichier composer.json. En effet, il faut indiquer dans la section autoload que notre module doit être chargé dans le projet (ANNEXE 3).

Après avoir créé la fonction getConfig(), il faut créer un nouveau fichier "module.config.php" à la racine du module. Il faut ajouter les informations nécessaires pour que le ServiceManager puisse créer les instances dont il a besoin, ainsi que déclarer l'emplacement des templates de notre module (ANNEXE 4).

Enfin, il faut indiquer dans le projet que nous souhaitons ajouter un nouveau module (ANNEXE 5).

2. Création des routes et des contrôleurs :

Après avoir créé le module Admin, j'ai ajouté les routes de mon projet. Une route est un nom que l'on entre dans l'URL et qui pointe vers une partie de notre projet.

Par exemple, j'ai créé une route /color (ANNEXE 6). Comme on peut le voir dans l'annexe, la route que j'ai spécifiée est un peu spéciale. En effet, j'ai créé une route de telle sorte qu'elle puisse prendre des paramètres. Il faut savoir que ces paramètres sont facultatifs.

Analysons la route :

En premier lieu, je lui ai donné un nom : "admin_color". Ensuite, dès lors qu'il y a /admin_color dans l'URL, la route peut prendre deux paramètres non obligatoires qui sont "action" et "id". Voici des exemples d'utilisation des routes :

- /admin_color/index
- /admin_color/add
- /admin_color/edit/1
- /admin_color/delete/1

Comme on peut le voir, les routes sont toutes différentes et ne prennent pas tous les paramètres à chaque fois.

Pour continuer, je lui donne des règles (contraintes) pour les paramètres grâce aux expressions régulières (REGEX) (ANNEXE 7).

Pour finir, je lui donne une action par défaut si la route commence par "admin_color".

Passons maintenant aux contrôleurs :

Un contrôleur est responsable de gérer les requêtes de l'utilisateur, de traiter les données envoyées par l'utilisateur et de déterminer quelle vue doit être affichée en réponse à cette requête.

Le contrôleur est donc chargé de contrôler le flux d'exécution de l'application en récupérant les données nécessaires à partir des modèles et en appelant les vues pour afficher les données. Il peut également prendre en charge la gestion des erreurs et la redirection vers les pages d'erreur appropriées.

Pour la route "admin_color", j'ai créé un contrôleur ColorController.php.

Dans ce contrôleur, j'ai créé les fonctions permettant de faire le CRUD. Le CRUD permet d'ajouter, de modifier, de supprimer et de consulter quelque chose. Ces fonctions seront codées un peu plus tard (ANNEXE 8).

Comme on peut le voir dans le fichier, chaque nom de fonction se termine par "Action". Cela signifie que Laminas est capable d'identifier la fonction adéquate en fonction des paramètres envoyés dans l'URL. Toutefois, rien ne nous empêche de créer nos propres fonctions qui n'ont rien à voir avec les actions.

Pour bien comprendre le système de noms et d'URL, voici une image qui peut vous aider (ANNEXE 9).

3. Création de la base de données :

Après avoir fait les routes, les contrôleurs et configuré les fichiers des modules, j'ai créé une base de données MySQL à l'aide de l'outil phpMyAdmin. Pour ce faire, j'ai créé un schéma UML pour modéliser mes classes et les liaisons (ANNEXE 10).

Ensuite, j'ai créé les tables pour mon projet. Dans un premier temps, mon maître de stage m'a demandé de faire les tables sans les liaisons pour comprendre et assimiler le fonctionnement de Laminas.

Pour continuer, j'ai créé pour chaque table de la base de données un fichier avec le nom de la table et un fichier avec Table dans le dossier "Model" qui se trouve dans le dossier src de mon module.

Exemple : Color.php et ColorTable.php

Color.php : est un fichier qui représente la base de données. C'est un peu comme le miroir de la base de données. C'est dans ce fichier qu'on indique les types de données, la longueur, les contraintes pour les champs de la base de données.

ColorTable.php : est un fichier qui nous permet de faire des requêtes SQL grâce aux fonctions qui se trouvent dans la documentation. Par exemple, nous avons les fonctions suivantes :

- fetchAll()
- getColor($id)
- saveColor(Color $color)
- deleteColor($id)

Nous avons la base de données, les fonctions pour faire les requêtes SQL et les contrôleurs pour les traitements. Il faut maintenant faire la connexion à la base de données sur Laminas.

Pour ce faire, il faut créer un fichier global.php dans le dossier config à la racine du projet. Dans ce fichier, il faut mettre les informations permettant la connexion à la BDD (ANNEXE 11).

4. Configuration des fichiers module :

Cette étape est assez simple et relativement courte. En effet, après avoir créé le contrôleur et les fichiers permettant de faire les requêtes SQL, il faut modifier le fichier « Module.php » pour que le ServiceManager puisse configurer les TableGateway et injecter les bons TableGateway dans les contrôleurs.

Exemple :

Le contrôleur ColorController a besoin de ColorTable pour effectuer le CRUD. Il faut donc configurer le fichier Module.php pour donner les bonnes instances de classes (ANNEXE 12).

5. Création des formulaires :

Passons maintenant aux formulaires. Un formulaire est un fichier PHP où l'on spécifie les types de champs (input) à récupérer, en indiquant si un champ peut être vide ou non, etc. Ce fichier se trouve dans le dossier Form, situé dans le dossier src du module (ANNEXE 13).

Grâce à ce formulaire, on peut l'utiliser pour différentes actions telles que l'ajout ou la modification.

De plus, il faut ajouter des fonctions dans l'entité Color, c'est-à-dire Color.php, pour pouvoir effectuer des vérifications avant d'envoyer le formulaire (inputFilter).

Les input filters sont des mécanismes qui permettent de valider, filtrer et normaliser les données entrées par l'utilisateur avant de les utiliser dans une application. Les input filters sont configurés pour chaque champ de formulaire et peuvent être utilisés pour effectuer des validations simples et complexes. Les input filters sont importants pour améliorer la sécurité et la qualité de l'application.

6. Ajout des méthodes dans les contrôleurs :

Cette partie est également simple car la documentation de Laminas nous fournit le code permettant de faire le CRUD. Nous avons donc :

- indexAction
- addAction
- editAction
- deleteAction

7. Création des vues :

Pour finir, il faut créer les vues, la particularité avec Laminas c'est que l'extension pour les vues c'est .phtml.

Dans les vues, libre à nous de faire le design que l'on veut, mais le mieux est de faire le code PHP permettant de générer les formulaires comme indiqué dans l'annexe (ANNEXE 14).

D. Création d'un mini jeu Mario :

1. Le squelette du jeu :

Après quelques semaines, mon maître de stage m'a proposé de créer un jeu en utilisant HTML, CSS, JS et JQuery.

J'ai donc ajouté un code HTML basique et un peu de CSS pour obtenir un rendu convenable.

2. Découverte de la bibliothèque JQuery :

Développer un mini Mario avait pour but de me faire découvrir la bibliothèque JQuery. En effet, ne connaissant pas JQuery, c'était pour moi l'occasion de manipuler cette dernière afin de découvrir d'autres technologies.

J'ai donc créé un système pour animer mon personnage avec des sprites et j'ai mis en place le système pour pouvoir avancer, reculer et sauter. Pour finir, j'ai ajouté des blocs en tant qu'obstacles pour simuler les collisions dans le jeu (ANNEXE 15).

E. Ajout de fonctionnalités et amélioration du projet E-commerce :

1. Mise en place de la technologie AJAX :

Après avoir manipulé jQuery, mon maître de stage m'a demandé d'afficher la liste des produits en AJAX sur le projet e-commerce. En effet, j'avais plus de 1500 produits en base de données et charger l'ensemble des produits à chaque fois prenait beaucoup de temps et dégradait surtout l'expérience utilisateur.

AJAX (Asynchronous JavaScript and XML) est une technologie web qui permet à une page web de mettre à jour dynamiquement une partie de son contenu sans recharger toute la page. Elle utilise des requêtes HTTP asynchrones pour récupérer des données du serveur et des scripts JavaScript pour mettre à jour l'interface utilisateur en temps réel.

Je suis donc allé sur la documentation jQuery pour comprendre et mettre en place la technologie AJAX sur mon e-commerce.

2. Création des routes :

Pour faire l'affichage en AJAX, j'ai fait une nouvelle route sur mon projet Laminas ce qui nous permet de récupérer le contenu généré en AJAX et de le placer sur la bonne page.

Exemple : sur la page /ajaxLoadProduct je vais afficher le html permettant d'afficher les produits 6 par 6.

3. Création des contrôleurs :

Pour faire de l'AJAX, j'ai fait un nouveau contrôleur qui va me permettre de faire les traitements nécessaires pour afficher le bon nombre de produits (ANNEXE 16).

4. Création du script JS :

Après avoir fait la route et le contrôleur, je suis passé sur le script JS.

J'ai commencé par faire une fonction onScroll en JS pour pouvoir afficher les produits 6 par 6. Ensuite, j'ai fait une fonction qui va récupérer l'ensemble des produits qui s'appelle fetchAllLimit et c'est la fonction onScroll qui va appeler la fonction fetchAllLimit (ANNEXE 17).

F. Les projets clients :

1. Optimisation du code :

Après quelques semaines de stage, mon maître de stage m'a fait travailler sur les projets clients.

J'ai travaillé sur un site web permettant de calculer l'empreinte carbone pour Carrefour. Ma mission était d'optimiser une fonction, en effet, celle-ci prenait énormément de temps à s'exécuter car les requêtes SQL étaient complexes et nous devions importer un fichier CSV en base de données.

Avant d'optimiser le code, j'ai commencé par comprendre le fonctionnement de la fonction car ce n'était pas mon projet mais celui de mon maître de stage. De plus, ce projet était assez complexe.

Ensuite, j'ai effectué des tests pour voir à quel moment la fonction prenait beaucoup de temps.

En essayant de comprendre et de faire des tests, je me suis aperçu que faire des commentaires dans le code était plus qu'utile. En effet, cela permet à une personne qui n'a jamais travaillé sur un projet de s'approprier celui-ci plus rapidement.

2. Optimisation des requêtes SQL :

Après avoir optimisé le code, mon maître de stage m'a demandé d'optimiser une requête SQL qui prenait également trop de temps.
La requête prenait approximativement 9 secondes en effet, on effectue des calculs directement dans la requête SQL sur des millions d'enregistrement.

Malheureusement, je ne peux montrer des photos du code et les requêtes pour des raisons de confidentialité.

3. Formatage des fichiers CSV :

Suite à l'optimisation du code et des requêtes SQL, j'ai appris les autres aspects du développeur, celles de formatter les fichiers dans le but d'importer dans la base de données. Pour ce faire, j'ai utilisé Excel pour formatter les fichiers CSV.

4. Affichage en AJAX :

Vers la fin du stage mon maître de stage m'a demandé de passer l'affichage des produits d'un site client en AJAX. Encore une fois, avant de me lancer, je regarde le code en global pour comprendre et m'adapter à ce nouveau projet.

Après plusieurs heures, je me suis retrouvé bloqué car l'AJAX ne fonctionnait pas, j'ai donc demandé de l'aide et finalement j'ai réussi à le faire fonctionner à moitié mais malheureusement mon maître de stage était obligé de le reprendre par manque de temps.

III. Présentation et justification de la méthodologie de travail suivie :

Pour pouvoir effectuer les missions que mon tuteur m'attribue, je commence par comprendre la mission en elle-même, notamment l'objectif

de celle-ci. Dans un premier temps, j'essaie de la faire sans aide pour voir si je peux la réussir sans assistance. Si je n'y parviens pas, je passe quelques minutes à consulter la documentation en ligne ou à rechercher des exemples sur internet.

Ensuite, après mes recherches, je passe aux tests pour remplir les missions qui m'ont été attribuées. Si je ne réussis pas, même avec l'aide des documentations, je demande à mon maître de stage des pistes de recherche sans pour autant lui demander la réponse. Très souvent, j'arrive à me débrouiller pour finir ma mission, mais lorsque c'est très difficile, je consulte la documentation.

Durant ces 8 semaines de stage, il m'est arrivé plusieurs fois de rester sur une ou plusieurs erreurs pendant plusieurs heures avec Laminas. Pour débugger ces problèmes, je peux passer plusieurs heures à comprendre les erreurs, mais quand j'ai vraiment beaucoup de mal, je demande de l'aide à mon tuteur de stage ainsi qu'à ses autres collègues.

Je trouve que ma méthode de travail permet d'apprendre plein de choses tout en continuant les missions qu'on m'a confiées. Lorsque j'ai des problèmes, je peux compter sur les documentations en ligne ou sur l'aide des personnes dans l'entreprise, même si elles sont dans un autre domaine.

IV. Synthèse des compétences métiers développées lors du stage :

Tout au long de mon stage, j’ai mobilisé les connaissances acquises en première année et en deuxième année dans le but de terminer les missions qu'on m'a confiées. J’ai appris à travailler en groupe, à communiquer et à m’exprimer au sein de mon équipe. J’ai également appris à coder en respectant les normes et les bonnes pratiques enseignées par mes collègues. De plus, j’ai découvert de nouvelles technologies pour travailler sur des projets qui m'étaient inconnus. En effet, un bon développeur n'est pas celui qui connaît tout, mais celui qui comprend et trouve des solutions aux problèmes.

Par ailleurs, j’ai pu expérimenter le rythme du monde professionnel et les exigences qui en découlent. Chaque tâche doit être effectuée correctement dans les moindres détails, non seulement pour terminer les missions, mais également pour les développeurs qui interviendront sur le code par la suite.

V. Bilan professionnel et personnel quant à l’expérience vécue en guise de conclusion :

Pour conclure, ce stage de deuxième année a été une expérience très enrichissante, à la fois d'un point de vue personnel et professionnel. Mon intégration au sein de l'entreprise GSV COMMUNICATION s'est très bien passée.

Tout au long de mon stage, j'ai remarqué que développer est certes très important, mais ce qui est le plus important, c'est la communication et le partage d'informations avec les autres personnes pour le bon avancement des missions/projets. En effet, j'ai été amené à travailler avec différentes personnes et expliquer son point de vue est très important pour travailler ensemble.

D'un point de vue professionnel, pouvoir travailler en toute autonomie m'a beaucoup aidé. Lorsque j'avais des problèmes, je devais me débrouiller tout seul, ce qui m'a permis de devenir plus autonome.

D'un point de vue personnel, ce stage m'a permis de consolider mes connaissances en développement.

Enfin, ce stage m'a également permis de découvrir de nouvelles technologies telles que le cloud, ce qui a renforcé mes connaissances en réseau et en développement.

VI. Annexes :

ANNEXE 1 - Présentation d'un module et ses dossiers :

```
projet/
    /module
        /Admin
            /config
            /src
                /Controller
                /Form
                /Model
            /view
```

Légende : Image montrant un module et ses sous-dossiers

ANNEXE 2 – Fichier Module.php :

```php
namespace Admin;

use Laminas\ModuleManager\Feature\ConfigProviderInterface;

class Module implements ConfigProviderInterface
{
    public function getConfig()
    {
        return include __DIR__ . '/../config/module.config.php';
    }
}
```

Légende : Image montrant la fonction getConfig() du fichier Module.php

ANNEXE 3 – Configuration autoload :

```
"autoload": {
    "psr-4": {
        "Application\\": "module/Application/src/",
        "Admin\\": "module/Admin/src/"
    }
},
```

Légende : Image montrant la configuration de la section autoload du fichier composer.json

ANNEXE 4 - Création du fichier module.config.php :

```
namespace Admin;

use Laminas\ServiceManager\Factory\InvokableFactory;

return [
    'controllers' => [
        'factories' => [
            Controller\AlbumController::class => InvokableFactory::class,
        ],
    ],
    'view_manager' => [
        'template_path_stack' => [
            'admin' => __DIR__ . '/../view',
        ],
    ],
];
```

Légende : présentation du fichier module.config.php

ANNEXE 5 – Indiquer l'utilisation du module :

```
return [
    'Laminas\Form',
    'Laminas\Db',
    'Laminas\Router',
    'Laminas\Validator',
    'Application',
    'Admin',
];
```

Légende : image montrant l'indication du module permettant l'utilisation de ce dernier.

ANNEXE 6 – Exemple de route :

```
'admin_color' => [
    'type'    => Segment::class,
    'options' => [
        'route' => '/admin_color[/:action[/:id]]',
        'constraints' => [
            'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
            'id'     => '[0-9]+',
        ],
        'defaults' => [
            'controller' => Controller\ColorController::class,
            'action'     => 'index',
        ],
    ],
],
```

Légende : capture d'une route depuis le fichier module.config.php

ANNEXE 7 – Les contraintes dans les routes :

```
'constraints' => [
    'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
    'id'     => '[0-9]+',
],
```

Légende : les contraintes des route grâce au REGEX.

ANNEXE 8 – Création du contrôleur :

```
1  namespace Admin\Controller;
2
3  use Laminas\Mvc\Controller\AbstractActionController;
4  use Laminas\View\Model\ViewModel;
5
6  class ColorController extends AbstractActionController
7  {
8      public function indexAction()
9      {
10     }
11
12     public function addAction()
13     {
14     }
15
16     public function editAction()
17     {
18     }
19
20     public function deleteAction()
21     {
22     }
23 }
```

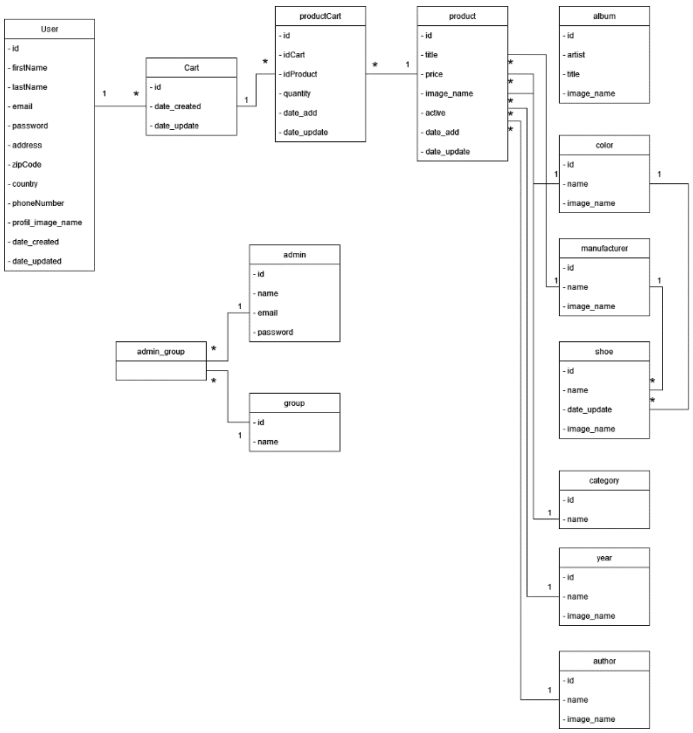
Légende : image montrant le ColorController avec des fonctions permettant de faire le CRUD vide.

ANNEXE 9 – Appel des fonctions grâce aux url :

URL	Method called
http://felix.gsvcom.fr/color	Admin\Controller\ColorController::indexAction
http://felix.gsvcom.fr/color/add	Admin\Controller\ColorController::addAction
http://felix.gsvcom.fr/color/edit	Admin\Controller\ColorController::editAction
http://felix.gsvcom.fr/color/delete	Admin\Controller\ColorController::deleteAction

Légende : tableau permettant de comprendre les fonctions appelé en fonction de l'url.

ANNEXE 10 – Schéma UML du projet e-commerce :



Légende : Capture du schéma UML du projet e-commerce

ANNEXE 11 – Connexion à la base de données :

```
return [
    'db' => [
        'driver' => 'Pdo',
        'username' =>'
        'password' =>
        'dsn' => 'mysql:dbname=felix;host=localhost;port=3306;charset=utf8',
    ],
];
```

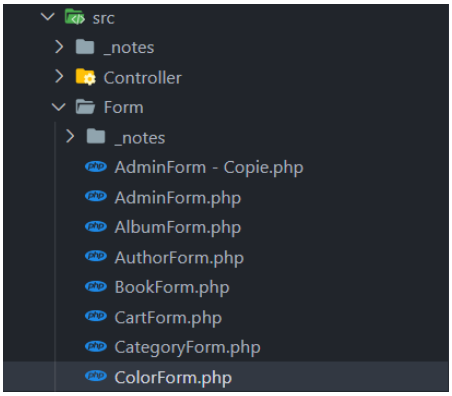
Légende : configuration du fichier global.php pour la connexion à la base de données.

ANNEXE 12 – Configuration des factories :

```php
    public function getControllerConfig()
    {
        return [
            'factories' => [
                Controller\ColorController::class => function($container) {
                    return new Controller\ColorController(
                        $container->get(Model\ColorTable::class)
                    );
                },
            ],
        ];
    }
}
```

Légende : Capture montrant la configuration des factories pour donner les bonnes instances de classes.

ANNEXE 13 – Emplacement des Form :



Légende : emplacement des fichiers form pour chaque model.

ANNEXE 14 – Exemple code pour afficher les couleurs :

```php
<?php

$title = 'My colors';
$this->headTitle($title);
?>
<h1><?= $this->escapeHtml($title) ?></h1>
<p>
    <a href="<?= $this->url('color', ['action' => 'add']) ?>">Add new color</a>
</p>

<table class="table">
<tr>
    <th>Title</th>
</tr>
<?php foreach ($colors as $color) : ?>
    <tr>
        <td><?= $this->escapeHtml($color->title) ?></td>
        <td>
            <a href="<?= $this->url('color', ['action' => 'edit', 'id' => $color->id]) ?>">Edit</a>
            <a href="<?= $this->url('color', ['action' => 'delete', 'id' => $color->id]) ?>">Delete</a>
        </td>
    </tr>
<?php endforeach; ?>
</table>
```

Légende : code permettant d’afficher les couleurs

ANNEXE 15 – Extrait de code du jeu Mario :

```
function moveRight(countRight)
{
    let mario = $("#mario").position();
    $('#mario img').attr('src','../img/move_right/frame-'+countRight+'.png');
    $('#mario').css({left:mario.left+7});
}
function moveLeft(countLeft)
{
    let mario = $("#mario").position();
    $('#mario img').attr('src','../img/move_left/frame-'+countLeft+'.png');
    $('#mario').css({left:mario.left-7});
}
function jumpUpRight()
{
    let mario = $("#mario").position();
    $("#mario").animate({top:mario.top - 100},'fast');
    $('#mario img').attr('src','../img/jump-right.png');
    $(document).keydown(false);
    $("#mario").animate({top:mario.top + 100},'fast');
    //$('#mario img').attr('src','../img/stand-up.png');
}
```

```
function checkCollision(obstacles, mario)
{
    position = mario.position();
    var marioLeft = position.left;
    var marioTop = position.top;
    var marioRight = marioLeft + mario.width();
    var marioBottom = marioTop + mario.height();
    var marioWidth = mario.width();
    var marioHeight = mario.height();
    //console.log(marioRight);
    obstacles.forEach(obstacle =>
    {
        var obstacleLeft = obstacle.getPositionX();
        var obstacleTop = obstacle.getPositionY();
        var obstacleRight = obstacleLeft + obstacle.getWidth();
        var obstacleBottom = obstacleTop + obstacle.getHeight();
        if (marioRight>obstacleLeft)
        {
            $('#mario').css({left:position.left-10});
            console.log(true);
        }
        else if(marioBottom>obstacleTop)
        {
            console.log(true);
        }
    });
}
```

```
$( document ).ready(function()
{
    //Initialisation des variables
    var countRight = 1;
    var countLeft = 1;
    var obstacles = new Array();
    //Chargement des obstacles
    for(let NumObstacle = 1 ; NumObstacle < 3 ; NumObstacle++)
    {
        obstacles.push(new Obstacle($('.obstacle'+NumObstacle)));
    }
    //console.log(obstacles);

    //Detection des touches
    $(document).keydown(function(e){
        if (e.which == 37) {

            checkCollision(obstacles,$('#mario'));
            //left
            countLeft = countLeft + 1;
            if(countLeft == 12)
            {
                countLeft = 1;
                moveLeft(countLeft);
            }
            moveLeft(countLeft);
           //return false;
        }
        if (e.which == 32) {
            checkCollision(obstacles,$('#mario'));
            let position = $('#mario').position();
            jumpUpRight();
            state = true

            //up
            //return false;
        }
```

ANNEXE 16 – La fonction indexAction de AjaxController :

```
public function indexAction()
{
    $offset = $_POST['id'];
    $products = _array($this->productTable->fetchAllLimit($offset,0,'id'));
    return new ViewModel(compact('products'));
}
```

Légende : capture montrant le code permettant d'afficher les produits 6 par 6.

ANNEXE 17 - Code JS pour faire de l'AJAX :

```
var count = 3;
function onscroll(){
    if ($(window).scrollTop() >= ($(document).height() - $(window).height() - 200)) {
            count = count + 6;
            $.ajax({
            type: 'POST',
            cache: false,
            dataType: 'html',
            data: {id: count},
            url: '/ajax',
            success: function(data) {
                $('.ajax_container').html(data);
                initFunctions();

            }
        });
    }
}
```

```
function loadProducts()
{
    $.ajax({
        type: 'POST',
        cache: false,
        dataType: 'html',
        data: {id: count},
        url: '/ajax',
        success: function(data)
        {
            $('.ajax_container').html(data);
            initFunctions();
        }
    });
}
```

```
public function fetchAllLimit($limit = 3, $offset = 0, $order = "id")
{
    $this->offset = $offset;
    $this->limit = $limit;
    $this->order = $order;

    $resultSet = $this->tableGateway->select(function (Select $select){
        $select->join(array('A' => 'author'), $this->table . '.id_author= A.id', ['author_name'=>'name'], 'left');
        $select->join(array('Y' => 'year'), $this->table . '.id_year= Y.id', ['year_name'=>'name'], 'left');
        $select->join(array('C' => 'color'), $this->table . '.id_color= C.id', ['color_name'=>'name'], 'left');
        $select->join(array('M' => 'manufacturer'), $this->table . '.id_manufacturer= M.id', ['manufacturer_name'=>'name'], 'left');
        $select->join(array('CA' => 'category'), $this->table . '.id_category= CA.id', ['category_name'=>'name'], 'left');

        $select->limit($this->limit);
        $select->offset($this->offset);

        $select->order($this->order);

        //s($this->tableGateway, $select);

    });
    return $resultSet;
    //return $this->tableGateway->select();
}
```

Légende : Extrait de code montrant les fonctions permettant de faire l'affichage en AJAX.