Compte Rendu (Sprint 4 : Agora BO sur Symfony)

Valentin, Yenis, Marco, Marvin

Projet réalisé : Transformation de la structure du site d'administration de la MJC AGORA de Libreville en projet Symfony (site créé durant la Mission 1 et intégration de Twig dans la Mission 3).

Contexte de la mission

Après la deuxième mission où nous avons intégré Twig comme moteur de templates pour améliorer la séparation entre la logique et la présentation, nous avons poursuivi avec une nouvelle étape : l'intégration du framework Symfony.

L'objectif de cette évolution est de bénéficier d'une structure plus robuste, d'un routage intégré, d'un meilleur système de gestion des dépendances (via Composer), et d'une architecture conforme aux standards modernes du développement PHP.
Cette transition permet aussi de centraliser la configuration, de renforcer la sécurité et d'améliorer la maintenabilité du projet sur le long terme.

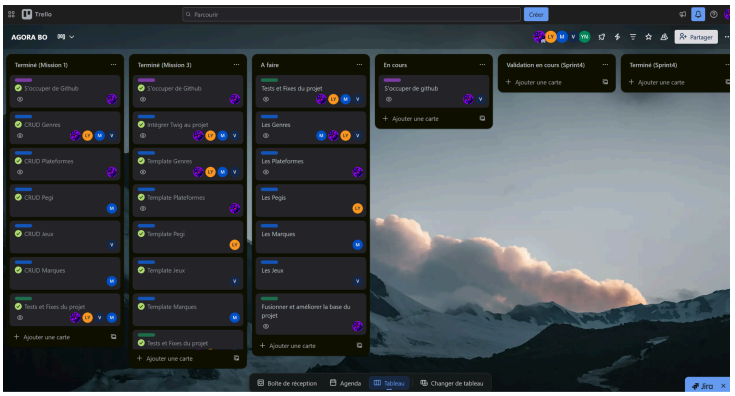
Organisation

Nous utilisons **Trello**, **Github**, **Google Docs** pour le compte rendu : chaque étape est définie sur le projet Trello pour avoir une vision globale de l'avancée de la mission afin de s'organiser le mieux possible. Pour Github nous utilisons la méthode du "pusher" pour éviter les conflits lors des push.

Nous nous sommes partagés ainsi les tâches :

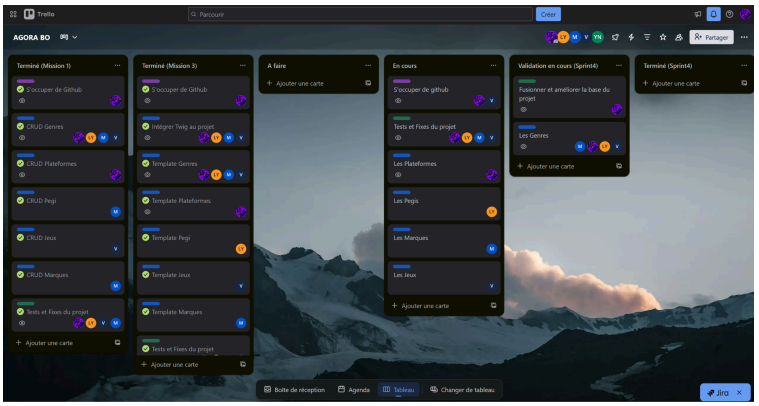
Création et amélioration du projet : **Tout le groupe**
Contrôleur et vue des Genres : **Tout le groupe**
Contrôleur et vue des Jeux : **Valentin**
Contrôleur et vue des Marques : **Marco**
Contrôleur et vue des Pegis : **Yenis**
Contrôleur et vue des Plateformes : **Marvin**

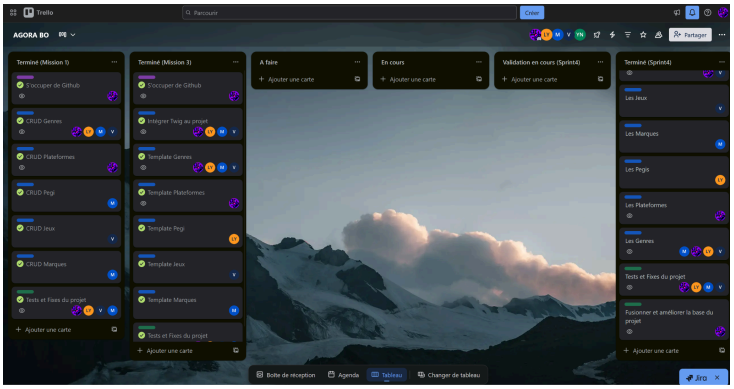
Trello au début



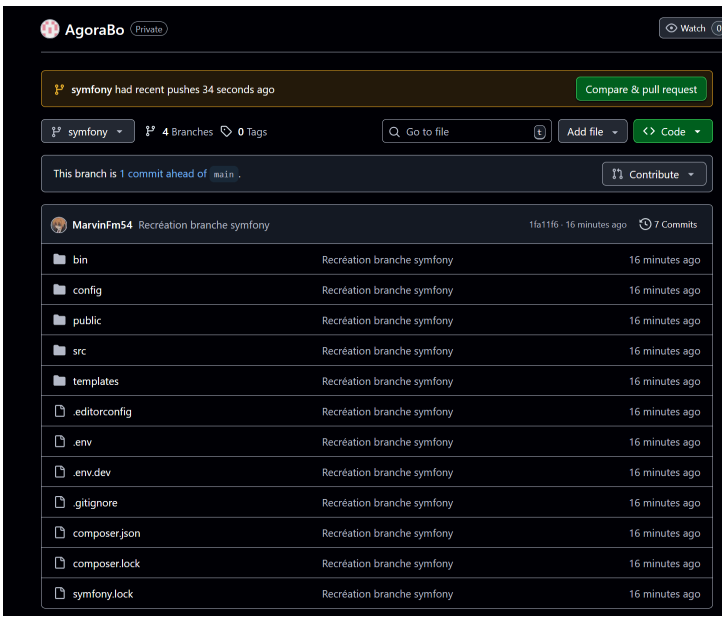
Trello au milieu du projet

Trello à la fin du projet





Branche GitHub pour le projet Symfony



Technologies utilisées

- **Back-end** : PHP
- **Front-end** : Twig, HTML, Bootstrap, JS
- **Base de données** : MySQL
- **Architecture** : MVC
- **Outils** : Visual Studio Code, GitHub, Apache, Xampp/Wamp

Pourquoi Symfony ?

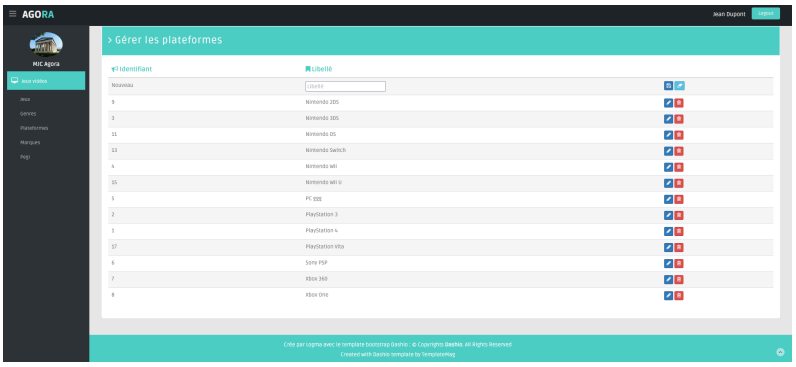
Symfony est un framework PHP moderne et structuré qui facilite le développement d'applications web robustes et maintenables. Il offre une architecture claire basée sur le modèle MVC, un système de routage intégré, une gestion efficace des dépendances grâce à Composer et une forte sécurité intégrée.

En l'adoptant, nous gagnons en productivité grâce à ses nombreux composants réutilisables et à la standardisation du code. De plus, Symfony s'intègre parfaitement avec Twig pour séparer la logique du code et la présentation, tout en garantissant une meilleure évolutivité du projet sur le long terme.

Cas d'utilisation

<u>Affichage des plateformes :</u>

→ L'utilisateur (admin) choisit « Plateformes » dans le menu.



→ Le lien “Plateformes” dans le menu mène à “/plateforme/”. Cette route est écrite dans le Controller des Plateformes (PlateformesController.php)

```php
#[Route(path: '/plateformes', name: 'plateformes_afficher')]
4 references | 0 overrides
public function index(SessionInterface $session): Response
{
    if (!$session->has(name: 'idUtilisateur')) {
        $session->set(name: 'idUtilisateur', value: 'autoUser');
        $session->set(name: 'role', value: 'admin');
    }
    $db = PdoJeux::getPdoJeux();
    return $this->afficherPlateformes(db: $db, idPlateformeModif: -1, idPlateformeNotif: null, n…'rien');
}
```

→ Le contrôleur récupère les données via le modèle (dans la classe PdoJeux, appelée avec “use App\Model\PdoJeux;”).

```php
src > Model > PdoJeux.php > ...
<?php

namespace App\Model;

/**
 *  AGORA
 *  ©  Logma, 2019
 * @package default
 * @author MD
 * @version    1.0
 * @link       http://www.php.net/manual/fr/book.pdo.php
 *
 * Classe d'accès aux données.
 * Utilise les services de la classe PDO
 * pour l'application AGORA
 * Les attributs sont tous statiques,
 * $monPdo de type PDO
 * $monPdoJeux qui contiendra l'unique instance de la classe
 */

use \PDO;
use \PDOException;
use \Exception;

72 references | 0 implementations
class PdoJeux
{
```

```php
use App\Model\PdoJeux;

4 references | 0 implementations
class PlateformesController extends AbstractController
{
```

→ Une fois traitée, les données sont transmises à un template Twig (gererPlateformes.html.twig) *(Déjà créée dans la Mission 3).*

→ Twig affiche proprement la liste des plateformes avec une boucle “{% for plateforme in plateformes %}”.

```twig
{% for plateforme in tbPlateformes %}
    <tr>
        <form action="{% if plateforme.identifiant == idPlateformeModif %}{{ path('plateformes_validermodifier') }}{% else %}{{ path('plateformes_demander
            <td>
                {{ plateforme.identifiant }}
```