LYCÉE ROBERT SCHUMAN METZ

AgoraBo

CHYLAK KILYAN / ENES / ANTHONY / NOÉ

26/09/2024

Table des matières

1 Expliquer la technologie Symfony et faire un rappel sur TWIG et MVC

- Symfony est un framework basé sur le modèle MVC qui permet de réaliser des sites complexes rapidement, mais de façon structurée et avec un code clair et maintenable. SensioLabs est le créateur du framework PHP Symfony. Fabien Potencier est le fondateur et chef de projet Symfony. Symfony fédère plus de 600 000 développeurs Symfony à travers le monde

- Twig est un moteur de template qui a été créé par SensioLabs, les créateurs du framework Symfony. On le retrouve nativement dans les frameworks Symfony, mais il peut être installé sur la majorité des frameworks ainsi que dans un environnement PHP.
  Un moteur de template est un outil de modèle structurel qui simplifie la syntaxe pour assurer une bonne maintenabilité d'une application web. Il permet de dissocier la partie présentation (HTML) de la partie programmation. Les fichiers templates sont des fichiers qui comportent la mise en page des pages, (et aussi des e-mails, flux RSS, …).
- L'architecture **MVC** (Modèle-Vue-Contrôleur) est un modèle de conception couramment utilisé dans le développement d'applications web et logicielles. Elle permet de séparer les différentes responsabilités d'une application en trois composants distincts : le **Modèle**, la **Vue**, et le **Contrôleur**. Cette séparation facilite la maintenance, l'évolution et la réutilisation du code.

2 Présenter le contexte et le besoin du client

Les besoins de la MJC changent, l'application AgoraBo sera progressivement complétée avec de nouvelles fonctionnalités. Afin d'en faciliter la maintenabilité et l'évolutivité, l'ESN Logma a décidé d'utiliser le framework Symfony pour refondre l'application. En vue d'aligner au mieux l'application Agora avec les attentes de la MJC et de mettre met l'accent sur le retour d'informations des utilisateurs tout au long du processus de développement, l'ESN Logma a fait le choix d'utiliser le framework de gestion de projets Agile qui aide les équipes à structurer et à gérer leur travail. Les nouvelles demandes de la MJC vont faire l'objet de sprints qui est la terminologie utilisée dans le cadre de Scrum. Ce sprint va consister à réaliser cette refonte de l'application.

3 La stratégie et les outils permettant d'organiser l'équipe + Github

Scrum : Une méthodologie agile utilisée pour organiser et gérer les projets. Chaque Sprint a un objectif précis, et les tâches sont réparties entre les membres de l'équipe. L'idée est de travailler de manière efficace pour livrer les fonctionnalités du projet rapidement et régulièrement.
Git : Un système de contrôle de version distribué utilisé pour suivre les modifications du code, collaborer en équipe, et permettre un développement parallèle. Elle est utilisée en ligne de commande.
GitHub : est une plateforme en ligne qui héberge des dépôts Git. Elle permet de stocker, partager et collaborer sur des projets Git à distance.
Trello : Un outil de gestion de projet utilisé pour organiser les tâches. Dans Trello, vous pouvez utiliser des tableaux pour suivre l'état des tâches (TODO, En cours, À Valider, Terminé).
BRODE SUR LE SUJET, PARLE DE CE QUE TU CONNAIS DE TRELLO

4 stratégie de répartition des tâches et de granularité des tâches

Chaque membre de l'équipe se voit attribuer des tâches en fonction de ses compétences et des priorités du sprint

Pour avancer plus rapidement, chacun du groupe gère une table et lorsque la personne finit sa table elle aide ses coéquipiers à finir leur table.

- Noe gère la table Jeux (controller,modèle,vue)
- Enes gère la table Pegi (controller,modèle,vue)
- Anthony gère la table Marques(controller,modèle,vue)
- Kilyan gère la table Plateforme(controller,modèle,vue)

5 Notre stratégie de validation des tâches

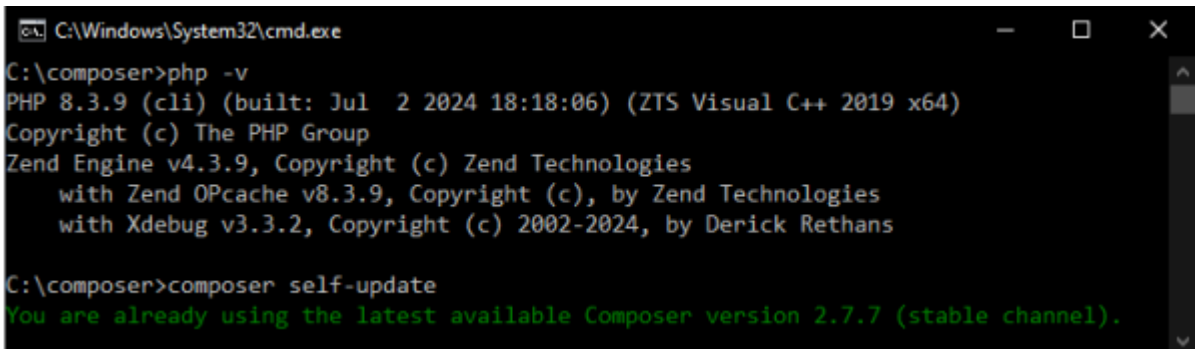
La stratégie de validation des tâches dans un projet est cruciale pour garantir la qualité et la stabilité du code

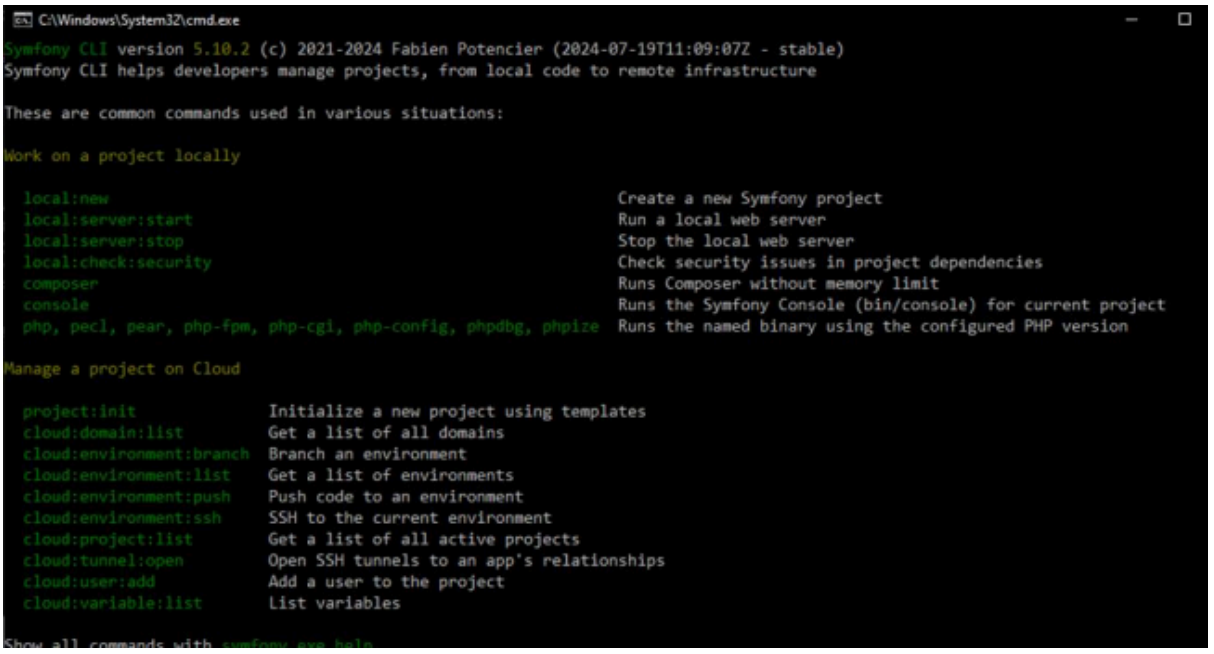
La revue de code est un processus par lequel un membre de l'équipe examine le code écrit par un autre membre avant que celui-ci ne soit fusionné dans la branche principale du projet.
exécution du code

6 présenter votre environnement symfony

- **Version de Symfony :**

  Symfony 7.1.3
  PHP 8.3.9





- **Structure d'un Projet Symfony :**

  `src/` : Le dossier où se trouvent le code source et les contrôleurs.

  `templates/` : Les fichiers Twig pour les vues.

  `config/` : Tous les fichiers de configuration pour les services, les routes, et les paramètres généraux de l'application.

  `public/` : Point d'entrée du site, où se trouve le fichier `index.php` (gère toutes les requêtes HTTP).

  `var/` : Contient les fichiers générés automatiquement comme les logs, les caches, etc.

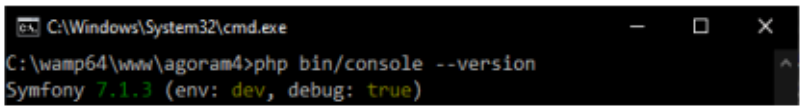
  `vendor/` : Les dépendances installées via Composer, comme Symfony lui-même, Twig, Doctrine, etc.

- **Commandes :**

symfony serve -no-tls : Pour lancer un serveur local

composer install : Installer les dépendances via Composer.

php bin/console –version : affichera la version actuelle de Symfony installée sur votre projet.



php bin/console make:controller : Générer un contrôleur.

php bin/console debug:router : Voir la liste des routes disponibles

symfony new lucky : Créez le squelette de l'application Symfony

- **Fichiers de Configuration :**

config/services.yaml : Définit les services utilisés dans l'application.

config/routes.yaml : Définit les routes disponibles.

config/packages/* : Contient les configurations des bundles et des paquets supplémentaires (comme Twig, Doctrine, etc.).

7 Explication complète du use case ajouter jeu, l'utilisation des bibliothèques de Symfony

Lorsque l'utilisateur ajout un jeu dans l ‘application voici le processus de l'architecture :

- Dans le controller Jeux Une Route est défini spécialement pour l’ajout de jeux

```
#[Route('/jeux/ajouter', name: 'jeux_ajouter')]

Cela signifie que lorsque l'utilisateur accède  aux chemin
/jeux/ajouter, la méthode ajouter du controller est appellé

#[Route('/jeux/ajouter', name: 'jeux_ajouter')] public
function ajouter(SessionInterface $session, Request $request)
{ $db = PdoJeux::getPdoJeux(); // Vérifie si le formulaire a
été soumis (vérification de la présence de 'txtRefJeu') if
(!empty($request->request->get('txtRefJeu'))) { // Appelle la
méthode du modèle pour ajouter un jeu avec les données
```

```
soumises $idJeuNotif = $db->ajouterJeu(
$request->request->get('txtRefJeu'),
$request->request->get('txtIdPlateformeJeu'),
$request->request->get('txtIdPegiJeu'),
$request->request->get('txtIdGenreJeu'),
$request->request->get('txtIdMarqueJeu'),
$request->request->get('txtNomJeu'),
$request->request->get('prixJeu'),
$request->request->get('txtDateParutionJeu') ); $notification
= 'Ajouté'; } // Rappelle la méthode privée afficherJeux pour
actualiser la liste des jeux return $this->afficherJeux($db,
-1, $idJeuNotif, $notification); }
```

Par exemple pour RefJeu Le contrôleur vérifie si le formulaire a été soumis en cherchant si la requete est présente $request->request->get('txtRefJeu').

- **Appel du modèle** :

  le contrôleur appelle la méthode ajouterJeu du modèle PdoJeux pour ajouter le jeu dans la base de données.

  ```
  public function ajouterJeu( string $refJeu, int $idPlateforme,
  int $idPegi, int $idGenre, int $idMarque, string $nom, float
  $prix, string $dateParution ) { $requete = 'INSERT INTO
  jeu_video (refJeu, idPlateforme, idPegi, idGenre, idMarque,
  nom, prix, dateParution) VALUES (:refJeu, :idPlateforme,
  :idPegi, :idGenre, :idMarque, :nom, :prix, :dateParution)';
  $stmt = PdoJeux::$monPdo->prepare($requete);
  $stmt->bindParam(':refJeu', $refJeu);
  $stmt->bindParam(':idPlateforme', $idPlateforme);
  $stmt->bindParam(':idPegi', $idPegi);
  $stmt->bindParam(':idGenre', $idGenre);
  $stmt->bindParam(':idMarque', $idMarque);
  $stmt->bindParam(':nom', $nom); $stmt->bindParam(':prix',
  $prix); $stmt->bindParam(':dateParution', $dateParution);
  $stmt->execute(); return PdoJeux::$monPdo->lastInsertId(); }
  ```

La Notification :

- Une notification est stockée dans la variable $notification pour informer l'utilisateur que l'ajout a été effectué avec succès.

- **Affichage des jeux** :

  La méthode afficherJeux est appelée pour afficher à nouveau la liste des jeux avec la nouvelle entrée ajoutée.

4. Vue

La vue est responsable de l'affichage des données à l'utilisateur.
Le contrôleur appelle la méthode render pour afficher la page Twig.

5. Bibliothèques Symfony utilisées

- **SessionInterface** : Utilisé pour gérer les sessions et vérifier si l'utilisateur est connecté.
- **Request** : Gère les données de requête HTTP pour traiter les soumissions de formulaires.
- **Twig** : Utilisé pour afficher les vues, ici, pour rendre le formulaire et afficher la liste des jeux.

.