STAGE 2ème année de Célia Baylet :

PIXINE

2ème semaine du 20 au 24 janvier 2025

20/01/2025

-Aujourd'hui pour mon nouveau ticket je dois rendre automatique la désactivation des formateurs rattachés à un organisme de formation inactif. Nous avons fait des tests fonctionnels pour vérifier que ce que l'on voulait faire fonctionne correctement. Lorsqu'un organisme de formation est passé inactif tous ses formateurs doivent être désactivés.

21/01/2025

-J'ai rajouté un 'tooltip' ou une infobulle en français, à côté d'un bouton activer/désactiver. Un tooltip est un message qui permet d'afficher des explications du contenu au passage de la souris.

Le 'tooltip' est créer dans la vue et on rajoute un lien pour récupérer le message qui est dans un fichier .yaml

*-Dans les mêmes locaux que Pixine il y a aussi **KOOX** productions qui est une agence créative de productions audiovisuelles ils ont donc décidé de faire un tournage pour la carte de vœux de cette année avec pixine.*

22/01/2025

-Aujourd'hui j'ai écrit un test fonctionnel puis on a testé notre code créé précédemment avec Julien.

-Dans mon nouveau ticket à côté des dates des formations je devais rajouter pour chacune si elles étaient faites en présentiel en visio ou mixte (présentiel + visio).

23/01/2025

-Aujourd'hui nous avons fait un point avec l’équipe développement normalement c'est tous les mardis mais exceptionnellement cette semaine c'était jeudi.

-J'ai avancé le même ticket qu’hier, fait du front end, du css car il fallait que le format (présentiel visio mixte) soit bien aligné comme les autres points, qu'il y ait les bons espaces etc…

24/01/2025

-J’ai continué mon ticket, il fallait appeler un ‘component’ dans la vue twig, pour faire une alternative color c'est à dire que ‘mixed’(mixte) soit en rouge obligatoirement. Pour présentiel et visio pas d’obligation particulière.

-J'ai fait une nouvelle branche pour mon nouveau ticket il fallait rajouter une civilité Mme/M. sur les formulaires d'ajout et de modification d'un formateur. Je dois donc créer une liste déroulante pour que l’utilisateur choisisse la civilité. Pour la modification, il faut la noter dans un label qui contient les informations du formateur.

-J'ai pu voir la carte de vœux mais elle n'est pas encore diffusée, elle le sera sur leur site ;)

-Cet après-midi Maximilien m'a aidé à refaire une migration pour le champ ‘civility’ de mon ticket *(voir annexe)

Fiche annexe + Vocabulaire :

-Maximilien m'avait expliqué dans la semaine que sur javascript il n'y a pas de ‘type’ et c'est quelque chose qui est important pour la sécurité. Aujourd'hui il existe ‘typescript’ c'est Java script mais avec type donc plus sécurisé.

*champ ‘civility’ migration :

make env-php : pour faire une migration il faut d'abord se connecter au docker

php bin/console make:migration

On voit alors dans le terminal s’afficher le nom du fichier de migration créer, un control + clic dessus permet de l'ouvrir.

On supprime tout ce qui a été créé dans les fonctions up et down.

On ne modifie jamais le down car on risque de supprimer des données à un moment donné.

Dans le up, on fait un ‘alter table’ pour modifier ce que l’on souhaite.

ALTER TABLE trainer ADD civility…

Comment on column title IS\’(DC2Type :civility)\’

Cette 2eme ligne sert à commenter que dans la base de données le champ est appelé title mais il sera appelé civility.

Ensuite on fait la commande **make db-migrate** pour finaliser la migration.

On peut faire un clique droit puis ‘go to definition’, c’est pratique pour savoir à quoi correspond une fonction qui est appeler ailleurs.

Différence margin – padding :

-**margin** : C'est l'espace **extérieur** entre l'élément et les autres éléments. Il crée une distance entre un élément et son environnement.
Exemple : éloigner une boîte d'une autre boîte.

- **Margin = espace externe (autour)**.

-**padding** : C'est l'espace **intérieur** entre le contenu de l'élément (texte, image, etc.) et ses bords.
Exemple : ajouter un espace entre le texte et la bordure de la boîte.

- **Padding = espace interne (dedans)**.

Class …. Extends … implements … :

extends : Sert à *hériter* d'une classe parente. La classe enfant obtient toutes les propriétés et méthodes de la classe parente et peut les personnaliser ou les étendre.

implements : Sert à *appliquer* un ou plusieurs *interfaces*. Une interface est un contrat qui impose à la classe d'implémenter certaines méthodes.

Le ‘repository’ c'est un fichier qui est entre l'entité et la base de donnée. Le repository utilise des requêtes SQL ou un ORM pour convertir les objets en données manipulables par la base (et inversement).

find() **:** (dans un repository) sert à récupérer une entité en base de données par son **ID**.

dump(‘stop’) : affiche la valeur passée ('stop') dans la console ou sur la page pour inspection. C'est utile pour déboguer.

ctrl + shift + / : pour commenter dans tt les langages/décommenter

make env-php pour ouvrir le docker du projet

.vendor/bin/simple-phpunid-d memory-limit = 3000M
Chemin_acces_fichiers

Cette commande configure l'exécution de **PHPUnit** (outil de test) avec une limite de mémoire définie et un chemin d'accès pour les fichiers à tester. Voici l'explication des parties :

1. **.vendor/bin/simple-phpunit**
    - Chemin vers l'exécutable **PHPUnit** installé via Composer (**gestionnaire de dépendances** pour PHP)
    - Symfony utilise une version de PHPUnit gérée par simple-phpunit.
2. **-d memory_limit=3000M**
    - Définit la limite de mémoire PHP à **3000 Mo** pour éviter les erreurs liées à une consommation excessive de mémoire pendant les tests.
3. **Chemin_acces_fichiers**
    - Spécifie où se trouvent les fichiers ou tests à exécuter.

1- créer une branche sur la branche dev sur sublime merge.

2-Répondre au ticket.

3-Faire des tests visuels grâce à localhost. Mais aussi des tests fonctionnels en les écrivant.

4-la commande : make phpcsfixer && make phpstan

-make phpcsfixer

- Exécute **PHP CS Fixer**, un outil qui corrige automatiquement les problèmes de style et formatage du code selon des règles définies.

-make phpstan

- Lance **PHPStan**, un outil d'analyse statique pour détecter les erreurs potentielles dans le code (ex. types incorrects, variables non définies).

5-Ensuite on va faire un commit sur sublime merge donc on clique sur ‘stage all’ puis un ‘push --force’.

(6)-si il y a plusieurs commits alors on fait la commande : git rebase -i HEAD~x (x nombre de commit à regrouper)

7-Puis on va faire une ‘pull request’ sur bitbucket, dans your work -> extranet (le nom du projet) -> pull requests -> create pull request (pr)

Et on pousse notre branche sur dev

Pour finir sur pipelines on peut observer si notre PR est ‘in progress’, ‘successful’ ou ‘failed’.

le rôle des contrôleurs est de vérifier les entrées-sorties :

Les **contrôleurs** (controllers) gèrent la logique de traitement des requêtes HTTP, en récupérant les données, en appelant les services appropriés, et en renvoyant les réponses (généralement des vues ou des données). Ils servent de médiateurs entre les utilisateurs et l'application.

Make env-start : demarre le projet

Make env-start-worker pour démarrer l’environnement + est utilisé qd on change une vue twig

make env-restart-worker : permet de redémarrer le worker pour appliquer les changements récents effectués dans l'application.

Dans le projet il y a des fichiers .yaml qui servent à stocker des messages pour éviter de surcharger le code.

les fichiers **.yaml** sont utilisés pour **stocker des messages de traduction** ou des chaînes de texte (comme des erreurs, messages utilisateur, etc.) afin de **séparer le contenu de la logique du code**.

Factory : Une **factory** est un outil ou une classe utilisée pour **générer de fausses données** (par exemple, pour les tests ou le développement).

Fixtures : Les **fixtures** sont des jeux de données pré-définis qui sont utilisés pour remplir une base de données avec des valeurs de test ou des données initiales dans une application. Elles sont souvent utilisées dans des environnements de développement ou de test pour simuler des données réelles.

Les tests sont souvent créés dans des **contrôleurs de test** pour séparer la logique de test du code principal de l'application. Cela permet de :

1. **Isoler les tests** : Tester les contrôleurs et autres composants sans perturber l'application en production.
2. **Simuler des requêtes** : Les contrôleurs de test peuvent simuler des requêtes HTTP pour tester les réponses des actions des contrôleurs sans avoir à passer par un serveur réel.

Cela facilite les tests unitaires et fonctionnels, tout en assurant que les modifications ne cassent pas la logique existante.

Un **crawler HTML** (ou **web crawler**) est un programme automatisé qui explore des pages web pour en extraire des informations. Il navigue de page en page, suit les liens, et récupère des données comme des textes, des images, des métadonnées ou des liens.

Make db : **make db** est souvent utilisée dans des projets Symfony pour gérer la base de données, et dans ce cas, elle peut être associée à la recréation des données de test ou fausses données à l'aide des **fixtures**.