Table des matières

I. Contexte

Le sprint 3 du projet "Maya" a été défini par M. Vigne, le Product Owner (PO). Les principales fonctionnalités à développer au cours de ce sprint étaient les suivantes :

- Gérer les produits/recettes.
- Réaliser des recherches multi-critères sur les produits à afficher par pages.
- Afficher la liste des catégories sous forme de card sur la page d'accueil.
- Ajouter la pagination et la recherche multi-critères à la gestion des clients et fournisseurs.
- Ajouter la pagination aux objets de gestion de l'application.

II. Itinéraire

1) Gérer les produits/recettes avec Symfony (formulaires, Doctrine)

Chaque membre de l'équipe a suivi un tutoriel pour gérer les produits avec Symfony, en utilisant des formulaires et Doctrine ainsi que des entités et des controller afin de gérer toutes les opérations. Un étudiant a été désigné pour mettre à jour le dépôt GitHub à la fin de la mission.

```php
#[Route('/modifier/{id<\d+>}', name: 'modifier')]

public function modifier(Produit $produit = null, Request $request, EntityManagerInterface $entityManager): Response
{
    $form = $this->createForm(ProduitType::class, $produit);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // cas où le formulaire  a été soumis par l'utilisateur et est valide
        //pas besoin de "persister" l'entité : en effet, l'objet a déjà été retrouvé à partir de Doctrine ORM.
        $entityManager->flush();
        $this->addFlash(
            'success',
            'Le produit '.$produit->getLibelle().' a été modifié.'
        );

        return $this->redirectToRoute('produit_index');
    }
    // cas où l'utilisateur a demandé la modification, on affiche le formulaire pour la modification
    return $this->render('produit/modifier.html.twig', [
        'form' => $form->createView(),
        'produit' => $produit
    ]);
}
```

Fonction permettant de modifier le produit la route sera utiliser par la suite de le template.

2) Recherches multi-critères sur les produits

De la même manière que pour la Mission 1, chaque membre de l'équipe a suivi un tutoriel pour implémenter des recherches multi-critères sur les produits. Qui ce fait via un formulaire symfony qui va récupérer les données rechercher et effectuer une requête SQL qui sera envoyer à la vue afin d'afficher les résultat rechercher.

```php
2 références | 0 overrides
public function findByNom(FournisseursRecherche $fournisseursRecherche): Array
{
    // le "p" est un alias utilisé dans la requête
    $qb = $this->createQueryBuilder('f')
        ->orderBy('f.nom', 'ASC');

    if ($fournisseursRecherche->getNom()) {
        $qb->andWhere('f.nom LIKE :nom')
            ->setParameter('nom', $fournisseursRecherche->getNom().'%');
    }

    $query = $qb->getQuery();
    return $query->execute();
}
```

3) Développement des autres fonctionnalités du sprint 3

Les fonctionnalités restantes du sprint, à l'exception de la pagination, ont été développées. Trello a été utilisé pour la répartition des tâches, et GitHub pour le développement collaboratif. Chaque membre de l'équipe a développé et testé la user story qui lui était attribuée avant de l'ajouter au dépôt GitHub une fois validée.

4) Mettre en œuvre la pagination sur les produits

Une mission individuelle a été effectuée pour mettre en œuvre la pagination sur les produits, suivant un tutoriel. En utilisant le bundle knp paginator qui est utiliser dans le repository et dans le controller pour être traiter dans la vue

```
<div class="navigation">
    {{ knp_pagination_render(lesProduits) }}
</div>
```

Les paramètre de la pagination sont définies dans le repository et dans les paramètre du bundle.

III. Annexes et Précisions

L'affichage des catégories dans la page d'accueil suit le principe des cards de Bootstrap. Les catégories peuvent être associées à une image, et ces informations sont dynamiquement récupérées depuis la base de données. Des liens utiles pour la mise en œuvre sont fournis. Et un couleurs aléatoire son données au carte Bootstrap via le controller.