Démarche de conception d'une User Story d'une application symfony

Un exemple avec l'application Equida : lister les chevaux d'une vente

Contexte

L'application Equida est une application permettant de gérer les ventes aux enchères des chevaux de courses. La société Equida organise environ dix ventes aux enchères par an sur deux sites :

- l'établissement « Elie de Brignac » situé à proximité de l'hippodrome de Deauville ;
- le manège « Boussac », situé sur l'hippodrome de Saint Cloud.

L'offre de vente aux enchères est variée, elle s'organise en catégories : vente mixte de février, vente d'été, vente d'automne, vente d'élevage, etc.

Lors d'une vente aux enchères, Equida propose différents types de chevaux :

- des yearlings, pur-sang anglais âgés d'un an ;
- des inédits, pur-sang anglais de deux ans n'ayant pas encore participé à une course mais déjà travaillés sur le plan de la condition physique ;
- des chevaux à l'entraînement, pur-sang arabes de plus de deux ans ayant déjà participé à des courses ;
- des étalons, chevaux dédiés à la reproduction ;
- des poulinières, juments dédiées à la reproduction.

Actuellement l'application permet de lister les ventes (passées et à venir).

L'application est opérationnelle. La configuration à la base de données est réalisée dans le fichier .env

Expression du besoin

Une nouvelle fonctionnalité (user story) doit être ajoutée à l'application Equida. Il faut ajouter la possibilité à l'utilisateur de l'application de consulter la liste les chevaux proposés lors d'une vente avec, pour chaque cheval, le prix de la mise de départ, les informations du vendeur. Si un cheval n'a pas été vendu, il pourra être remis aux enchères lors d'une prochaine vente. C'est la notion de Lot (précisé dans le cahier des charges).

Modélisation / Diagramme de classes

Le diagramme de classe existe déjà (classes en rose). Pour répondre au besoin exprimé, il faut le faire évoluer comme ci-dessous : classes en jaune.

Wireframes

On partira de la page existante « lister les ventes » et on ajoutera sur chaque vente, un lien qui affichera une page reprenant les informations de la vente et listant les chevaux sélectionnés pour la vente sur laquelle on aura cliquée.

Préalables sur l'application symfony

Cette nouvelle fonctionnalité sera développée en local à partir de la dernière version de l'application publiée sur gitHub. On créera une nouvelle branche nommée consulter_vente_chevaux pour implémenter cette nouvelle fonctionnalité. A chaque nouvelle unité de code ajoutée ou modifiée, un commit sera réalisé.

Couche Model

- l'entity Race dont nous aurons besoin existe déjà
- création de l'entity Lot + relation ManyToOne vers Vente
- création de l'entity Vendeur
- Création de l'entity Cheval + relation MTO vers Vendeur + relation MTO vers Race
- Reprendre l'entity Lot pour y ajouter la relation MTO vers Cheval Commit

L'id de chaque entity sera créé automatiquement ainsi que tous les getters/setters des propriétés.

Commandes cli symfony nécessaires :

```
php bin/console make:entity
```

Base de données

Il faut faire évoluer la structure de la base de données en fonction du nouveau Model.

- La commande cli symfony

  ```
  php bin/console make:migration
  ```

  permet de créer un fichier avec les instructions sql de modification de la base.

- La commande

  ```
  php bin/console doctrine:migrations:migrate
  ```

  permet d'exécuter ces instructions.

- Il faut vérifier en base de données les modifications réalisées (création des tables, des clés primaires et clés étrangères; par exemple , la table Cheval a pour Pk id int auto_increment, une Fk vers la Pk de TypeCheval, une Fk vers la Pk de Vendeur).
- Ajouter ensuite des données pertinentes et en nombre suffisant dans les nouvelles tables : donc ajouter des chevaux avez leur vendeur et leur type, ajouter des lots avec le cheval et la vente liés.

Couche Controller

- le contrôleur VenteControlleur existe déjà. Il faut y ajouter la méthode permettant de consulter une vente
- Signature de la méthode :
  - nom : getVente
  - paramètres : manager de type EntityManagerInterface + idVente de type int
  - visibilité : public
  - type de retour : Response
- adapter la route associée *(dans fichier .env ou en annotation des méthodes dans le contrôleur)*:
  - nom de la route : app_vente_show
  - url : /vente/show/{idVente}
- Corps de la méthode
  - récupérer dans un objet vente, les données d'une vente avec la méthode find du repository prenant en paramètre l'idVente
  - renvoyer vers la vue qui sera dans le dossier templates/vente et qui sera nommée show.html.twig en fournissant l'objet vente récupéré ci-dessus.

Couche View

- Reprendre la vue permettant de lister les ventes (template/vente/list.html.twig) pour ajouter, sur chaque vente, un lien vers la route app_vente_show, en passant le paramètre de l'id de vente
- Dans le dossier template/vente, créer la vue show.html.twig
  - afficher les informations basiques d'une vente en récupérant la vente passée en paramètre dans le contrôleur (vente/nom, vente.CategVente.libelle, vente.dateDebut)
  - créer un tableau des chevaux qui sera construit en faisant une boucle sur les lots d'une vente ({% for leLot in vente.lots %} (affichage des informations d'un cheval avec leLot.cheval.nom, leLot.cheval.vendeur.nom ,etc...)
  - implémenter le design commun aux pages de l'application