

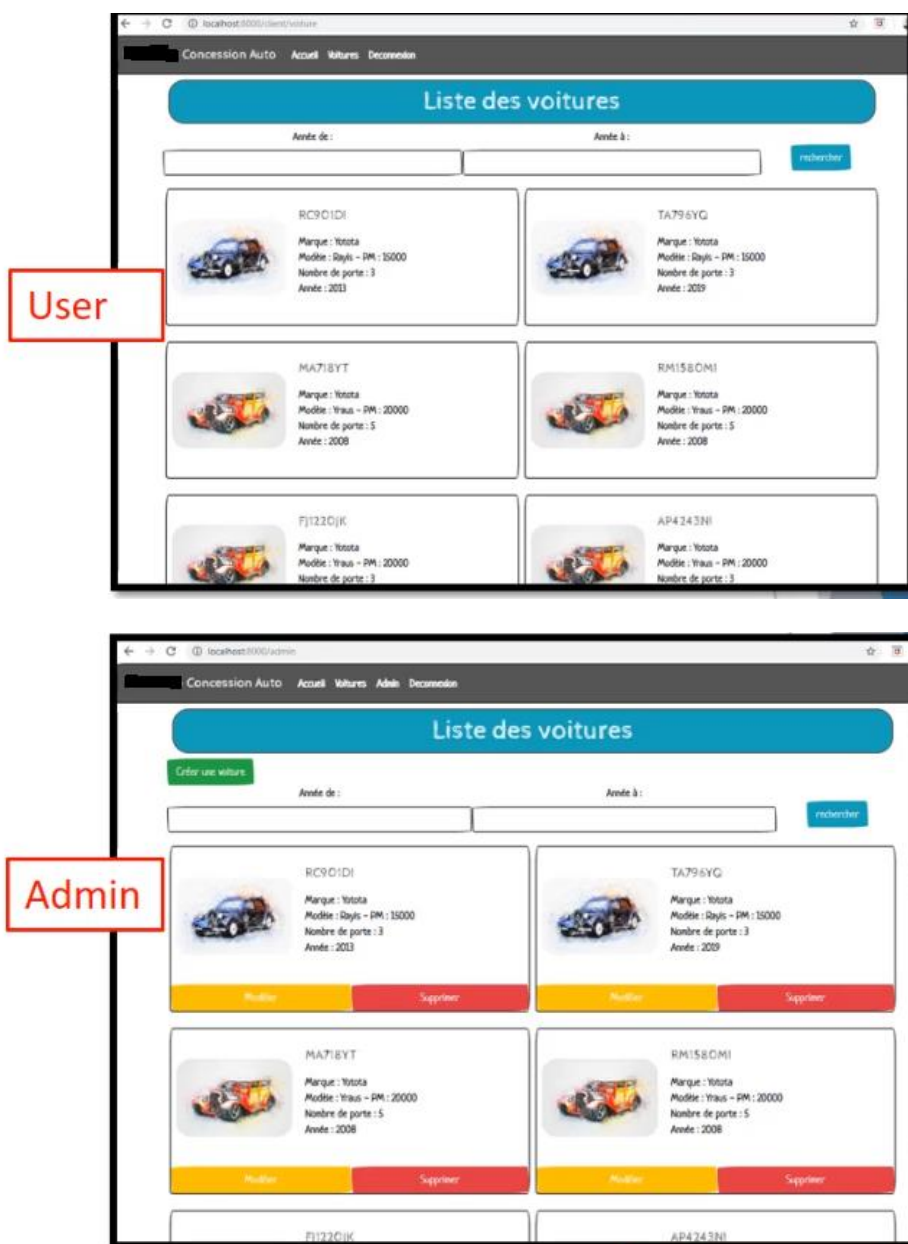
TP-2023-24 : Symfony

L'application à mettre en place permet de gérer un parc de voitures. Cette application reprend tout ce que vous avez appris dans le cours et servira d'exercice d'application.

Cependant, quelques fonctionnalités supplémentaires, comme l'utilisation des filtres sur l'année permettant d'aller sélectionner des voitures entre une année minimum et une année maximum, l'ajout d'un système de pagination, utilisation du service **Faker** permettant de gérer des données aléatoires et une explication de la mise en ligne sur Internet.

L'application est composée de deux interfaces **User** et **Admin**.

vous pouvez utiliser Bootstrap : <https://bootswatch.com> et utiliser le thème Sketchy.

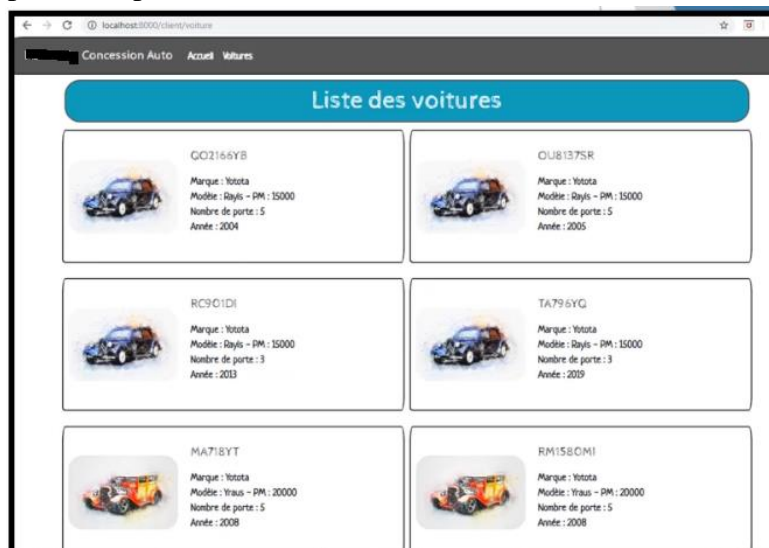


Pour mettre en place cette application, il est nécessaire de suivre les étapes suivantes

1- Création du projet

- 2- Mettre en place le MCD composé des entités suivantes :
 - **Voiture** (immatriculation, nbPortes : nombre de portes, année),
 - **Modèle** (libellé, prixMoyen, image),
 - **Marque** (libellé).
- 3- Création de la BD et des tables à partir du MCD ci-dessus :
- 4- Données Fixtures (remplir les tables avec des données). Pour cela on vous demande d'utiliser le composant Faker (à installer) qui permet de générer de manière aléatoire des valeurs. Ce bundle ne sera utilisé que pour les voitures.
On essaiera d'utiliser des données plus ou moins cohérentes (des vraies valeurs, c'est-à-dire à vous-mêmes de donner des valeurs plus ou moins exactes).
- 5- On utilisera les informations suivantes :
 - Marques : 2
 - Modèles : 5
 - nbPortes (nombre de portes) : 3 ou 5
 - L'immatriculation est de la forme : XX1234XX (XX étant des lettres). Pour cela utiliser une expression régulière (fonction) que l'on trouve dans la documentation de Faker.
 - Générer 3 à 5 voitures pour chaque modèle (quels sont le maximum et le minimum de voitures de ce parc ?).
 - L'année : comprise entre 1999 et 2019

Pour l'installation et l'utilisation de Faker se rendre sur le lien ci-dessous :
<https://github.com/fzaninotto/Faker>
- 6- Affichage des voitures
 - Créer un controller : global
 - Créer la vue correspondante pour afficher toutes les voitures :



- 7- Mise en place d'une pagination
 - <https://github.com/KnpLabs/KnpPaginatorBundle>
 - Installer le bundle knp-paginator-bundle
 - Créer le fichier knp-paginator.yaml
 - Copier/coller l'exemple dans votre fichier
 - Modifier le controller des voitures et le «Repository»

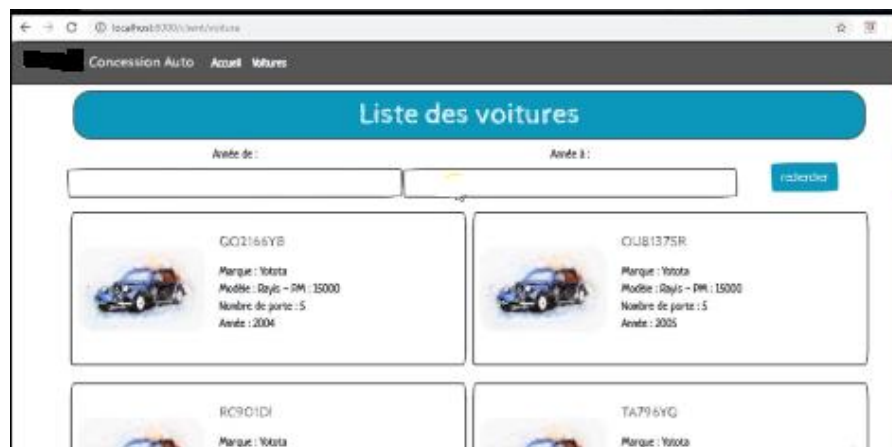
- 8- Mettre en place deux filtres sur l'année permettant d'aller sélectionner des voitures entre une année minimum et une année maximum.



Il faut générer un nouveau formulaire permettant de sélectionner ces voitures. Pour cela, créer une entité **RechercheVoiture** qui sera lié à un formulaire spécifique à définir. Cette entité sera créée à la main dans le dossier **Entity** et non en ligne de commande et ne doit pas être présente en BD.

N'oublier pas de rajouter dans votre fichier de configuration **Twig.yml** le thème de **Bootstrap**.

Ensuite vous rajouterez à l'intérieur de votre page le formulaire avec les deux champs input de type **number** qui permettront de récupérer l'année minimum et l'année maximum.



Enfin, envoyer l'information au Repository pour qu'il puisse la traiter.

```
$voitures = $pageInterf->paginate(
    $vRepo->findAllWithPagination($rechercheVoiture),
    $request->query->getInt('page', 1),
    6
);
```

```

public function findAllWithPagination(RechercheVoiture $recherche) : Query{
    $req = $this->createQueryBuilder('v');
    if($recherche->getMinAnnee()){
        $req = $req->andWhere("v.annee > :min")
        ->setParameter('min', $recherche->getMinAnnee());
    }
    if($recherche->getMaxAnnee()){
        $req = $req->andWhere("v.annee < :max")
        ->setParameter('max', $recherche->getMaxAnnee());
    }

    return $req->getQuery();
}

```

9- Validation des données sur la recherche : Ajout de validation

Ajouter quelques éléments de validation sur le formulaire pour par exemple vérifier qu'on a saisi les valeurs comme il se doit c'est-à-dire que l'année minimum ne soit supérieurs à l'année maximum :

```

<?php
namespace App\Entity;

use Symfony\Component\Validator\Constraints as Assert;

class RechercheVoiture {

    /**
     * @Assert\LessThan(propertyPath="maxAnnee", message="doit être plus petit que l'année Max")
     */
    private $minAnnee;

    /**
     * @Assert\GreaterThan(propertyPath="minAnnee", message="doit être plus grand que l'année Min")
     */
    private $maxAnnee;

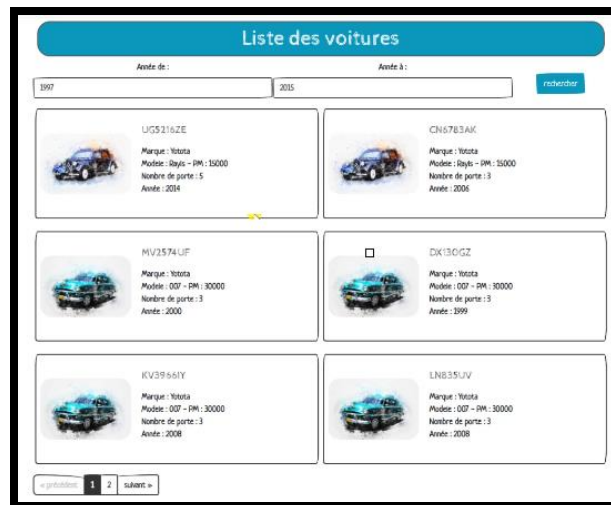
    public function getMinAnnee(){
        return $this->minAnnee;
    }
    public function getMaxAnnee(){
        return $this->maxAnnee;
    }
    public function setMinAnnee(int $annee){
        $this->minAnnee = $annee;
        return $this;
    }
    public function setMaxAnnee(int $annee){
        $this->maxAnnee = $annee;
        return $this;
    }
}

```

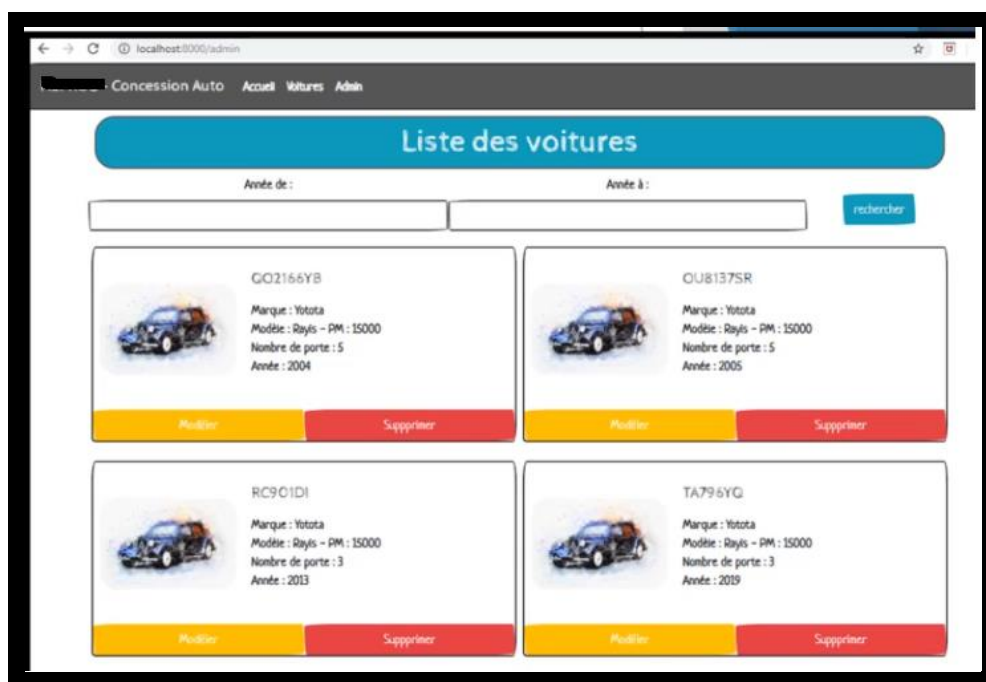
The screenshot shows a web interface for searching cars. The title is "Liste des voitures". Below the title, there are two input fields for the search criteria: "Année de :" and "Année à :". The "Année de :" field contains the value "2010" and has a red error message "ERREUR doit être plus petit que l'année Max". The "Année à :" field contains the value "2000" and has a red error message "ERREUR doit être plus grand que l'année Min". A "rechercher" button is located to the right of the input fields. The input fields have a red border and a red 'x' icon next to the error messages.

10- Mise en place d'un CRUD

L'objectif est de rajouter sur cette page dès lors qu'on est sur la partie administrative, les boutons **Modifier** et **Supprimer** ainsi qu'un bouton permettant d'ajouter un véhicule :



Pour effectuer cela, créer un nouveau controller sur la partie admin et générer une fonction spécifique permettant d'avoir le Controller d'administration de véhicules.



- Créer un nouveau Controller sur la partie Admin et générer une fonction spécifique permettant d'avoir votre Controller d'administration des véhicules.
- Dans le cadre de la partie administration, on enverra une information à la vue qui sera : *admin = true* (AdminController) et pour la page non administration (VoitureController), on enverra *admin = false*.

Ainsi on pourra réaliser des paramétrages supplémentaires dans le template en testant simplement si on l'information **admin** = true alors on fournira les deux boutons **Modifier** et

Supprimer.

Donc, la création de ce Contrôleur doit permettre l'apparition du menu Admin dans la barre.

- * Placer les deux boutons **Modifier** et **Supprimer** dans le template (**voitures.html.twig**).

- * Afficher le contenu des deux menus : Admin et Voitures.

- * Placer le lien pour le bouton Ajouter

11- Créer un formulaire qui permet d'**ajouter** et de **modifier** un véhicule. Pour cela, générer le formulaire via la ligne de commande et l'afficher dans un nouveau template à créer pour celui-ci.



Modification de la voiture immatriculée : OU8137SR

Immatriculation
OU8137SR

Nb portes
5

Année
2005

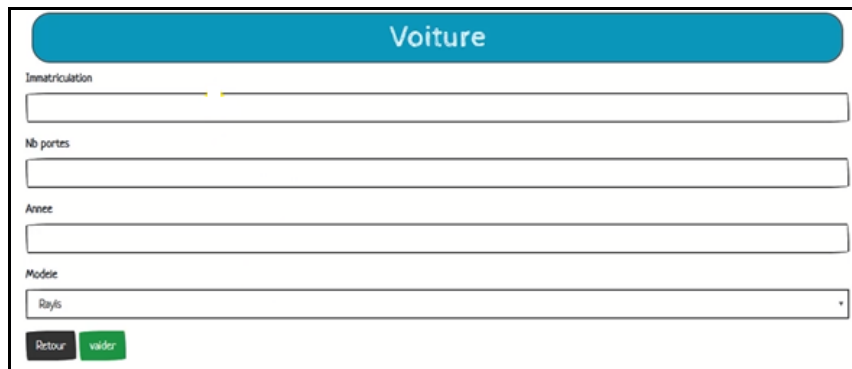
Modèle
Rayis

Retour Valider

Pour l'ajout, il suffit de créer une nouvelle route au niveau du formulaire ci-dessus qui ira pointer sur /admin/creation et indiquer qu'elle peut être null dans la fonction **modification** et si c'est le cas, il faudra alors créer une.

- Ajouter cette route dans **voitures.html.twig**, en la nommant (route nommée).

En cliquant sur le bouton Ajouter :



Voiture

Immatriculation

Nb portes

Année

Modèle
Rayis

Retour valider

Il est nécessaire d'afficher un message confirmant que l'action est réalisée avec succès.

12- Pour la suppression d'un véhicule

- Supprimer le code du bouton supprimer dans **voitures.html.twig** et créer à la place un formulaire et continuer les étapes de l'opération de suppression.