# JavaScript / Web Development - TP3

## Introduction

Dans ce TP, nous allons développer un site Web Frontend qui permettra d'ajouter des paris avant de lancer une course.

Nous allons donc utiliser le serveur développé dans le TP 2. Si vous n'avez pas implémenté toutes les fonctionnalités lors du précédent TP, vous pouvez utiliser la correction fournie.

Pour cela, nous allons utiliser différents outils/librairies :

- VueJS
- Element Plus
- Day.js

Le symbole  indique que vous devez faire quelque chose.

## Prérequis

Outils nécessaires :

- un navigateur récent (Firefox, Chromium)
- un éditeur de texte
- node (version >= 14.20.1)
- npm (version >= 6.14.17)

Démarrez tout d'abord le serveur backend en vous plaçant dans le dossier correspondant et en lançant la commande :

```
npm start
```

Le serveur écoute par défaut sur le port 3000.

## Démarrage

Pour information, ce projet a été initialisé en utilisant la commande create-vue. Nous utilisons le développement en mode Composant mono fichier (chaque fichier .vue est un composant).

Vous trouverez un dossier **phe-frontend** contenant :

- un fichier `package.json`
- un fichier `src/App.vue` contenant le code de notre page principale

Les autres fichiers ne seront pas manipulés dans ce TP :

- fichier `index.html` : page principale du site
- fichier `vite.config.js` : configuration Vite
- fichier `src/main.js` : fichier principal de configuration de VueJS

Placez-vous dans le répertoire **phe-frontend** et lancez la commande `npm install`. Cette commande va télécharger et installer les dépendances nécessaires pour notre TP.

## Lancer le site web en mode développement

Lancer la commande suivante : `npm run dev`, le rendu sera disponible à l'adresse http://localhost:5173. La page se recharge automatiquement lorsque vous modifiez votre code.

## Objectifs

Nous avons développé lors du TP 2 des routes pour effectuer les opérations suivantes :

- lister tous les paris (GET /paris)
- générer 10 paris aléatoirement (POST /paris/generate)
- ajouter un pari (POST /paris)
- supprimer un pari (DELETE /paris/:id)

Nous allons donc implémenter ces fonctionnalités dans notre page. La plupart des éléments visuels sont déjà présents dans la page, il faut ajouter la dynamique/logique Javascript et les balises VueJS.

La partie authentification et gestion de l'utilisateur ne fait pas partie de ce TP.

## Afficher les paris déjà présents dans la base de données

Notre composant Vue est séparé en 3 parties :

- template
- script
- style

Pour afficher les paris, nous allons le faire en deux étapes.

Premièrement, remarquez qu'il y a deux variables initialisées :

- `form` qui contiendra les données du formulaire
- `paris` qui est une liste initialisée avec de faux paris

Le premier objectif va être d'afficher les faux paris dans le template.

Utilisez la directive v-for pour afficher les paris (un pari par balise `el-card`). N'oubliez pas la balise :key sur la propriété `rowid` des paris.

Remplacez ensuite l'auteur, le cheval et la date par les propriétés du pari grâce à Mustache `{{ }}`.

L'affichage de la date n'est pas très esthétique. Créez une nouvelle fonction pour afficher la date avec le format suivant `Le DD/MM/YYYY à HH:mm` à l'aide de Day.js.

Maintenant que l'affichage est fonctionnel, nous aimerions travailler avec les vrais paris et pas ceux par défaut.

Dans la méthode `chargerParis`, utilisez **fetch**, comme vu au TP1, pour faire un appel au backend et récupérer les paris. Dans le traitement de la réponse, stocker la liste des paris dans la variable `paris`.

Ajoutez l'appel à cette méthode dans le "hook" `onMounted` pour déclencher le chargement des paris dès l'affichage de la page.

Une fois que tout fonctionne, supprimez les paris par défaut.

## Permettre l'ajout d'un pari

Nous allons maintenant travailler sur le formulaire d'ajout de pari. Pour cela, il faut utiliser le 2-way binding de VueJS.

Liez le champ `el-input` d'id **auteur** avec la variable `form.auteur`.

Faites de même avec le champ `el-select` d'id **cheval**.

Liez la soumission du formulaire `el-form` à la méthode `ajouterPari` et vérifiez via un `console.log` que la variable `this.form` est bien remplie. Lorsque l'utilisateur saisit des informations et clique sur le bouton **Parier**, vous devriez avoir les informations dans cette variable.

Ensuite, utilisez **fetch** pour faire un appel HTTP de type POST vers `http:/localhost:3000/paris` en transmettant les informations `auteur` et `cheval`.

Dans le traitement de la réponse, appelez la méthode `chargerParis` pour recharger les paris.

Vous pouvez aussi implémenter la méthode `onReset`, déclenchée au clic sur le bouton Reset et qui vide le formulaire (donc la variable `form`).

## Améliorer l'affichage des paris

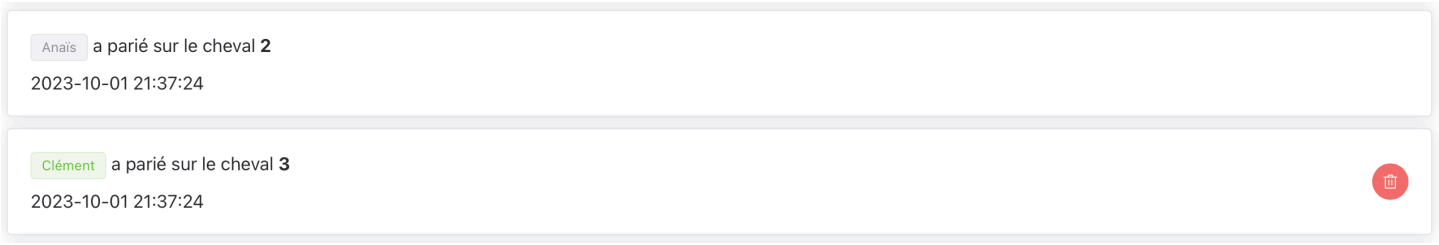
Maintenant que l'utilisateur a renseigné son nom, ce serait plus intéressant de mettre en évidence ses paris par rapport aux paris des autres utilisateurs.

Pour cela, nous allons utiliser v-if

Copiez tout le contenu du template qui contient l'affichage du pari. Ajoutez un `v-if` pour tester si l'auteur est celui du formulaire.

Si c'est le cas, modifiez la couleur du tag (`type="success"`) et ajouter une icône poubelle.

Si ce n'est pas le cas, n'affichez pas l'icône poubelle.




## Bouton pour générer 10 paris

Gestion des évènements

Liez le clic sur le bouton **Générez des paris** à la méthode `genererPari` pour appeler le backend sur l'opération de génération des paris.

Pensez à recharger les paris une fois l'appel terminé.

## Permettre la suppression d'un pari

Sur chaque pari dont je suis l'auteur, une icône permet de supprimer un pari.

Liez le clic sur l'icône poubelle à la méthode `supprimerPari` en passant en paramètre l'id du pari. Implémentez l'appel au serveur backend pour supprimer le pari de la base de données.

Dans la suite de l'appel, vous avez le choix entre recharger les paris depuis le backend ou supprimer le pari en javascript dans `paris`.

## Résultat

Vous avez développé en moins de 4 heures une page web dynamique capable de communiquer avec un serveur backend via des requêtes HTTP. L'utilisation de VueJS permet une manipulation facile du DOM (structure de la page) et un rendu dynamique et agréable pour l'utilisateur.

Dans le prochain TP, nous verrons comment créer plusieurs pages, découper notre page en composants, gérer l'authentification et la notion d'état.

## Conclusion

Dans ce TP, nous avons vu :

- comment développer un composant VueJS
- le développement de méthodes contenant de la logique
- l'utilisation de directives (v-if, v-for, v-model, ...)
- la gestion des évènements (actions utilisateur)

## J’ai fini plus tôt ou je veux en faire plus

Vous pouvez ajouter les fonctionnalités suivantes :

- ajouter des contrôles lors de l’ajout d’un pari (Nom et cheval obligatoire)
- afficher un message d’erreur s’il y a un problème de validation du formulaire ou de communication avec le backend (https://bootstrap-vue.js.org/docs/components/alert/)
- empêcher la modification du nom après la soumission du premier pari