**Site dynamique**

**Projet Caviste**

## Étapes de développement

**1 Introduction**

Mise en place d'une page Web qui permet d'afficher une liste de vins à partir d'une maquette fonctionnelle, tout en intégrant des fonctionnalités dynamiques via des librairies et des frameworks.

### Acteurs :

1. **Utilisateur non authentifié**
2. **Utilisateur authentifié**
3. **Système/API**

**Prérequis :**

* Connaissance des langages HTML5, CSS3 et JavaScript.
* Familiarité avec le framework Bootstrap et les librairies jQuery et jQueryUI.
* Maquette fonctionnelle pour guider la conception du prototype HTML.

### **2 Objectifs Principaux** :

**Objectifs du projet :**

* Integration d'une API RESTful pour Caviste.
* Développement d'une application cliente "MonCellier".

**3. Portée du projet**

**a. Application Web Caviste** :

* **API RESTful** accessible via caviste/api

**b. Site client MonCellier** :

* Application frontend accessible via moncellier.localhost/index.htm.

**4. Spécifications techniques :**

**a. Technologies et outils :**

* Frontend : Framework HTML Twitter Bootstrap, librairies jQuery/JQueryUI/Ajax.

**b. Fonctionnalités de l'API RESTful** :

* Répond aux commandes HTTP (GET, POST, PUT, DELETE...).

**d. Fonctionnalités du client MonCellier** :

* Consultation du catalogue de vins.
* Selectionner un vin pour afficher le détail
* Rechercher des vins par nom
* Trier la liste par cépage
* Filtrer la liste par pays et année
* Gestion d'une cave à vin : recherche, ajout, modification, suppression de bouteilles.
* Ajouter, supprimer, modifier des commentaires
* Modifier, supprimer une note personnelle
* Modifier, supprimer les détails des vins

**5. Développement :**

**Réalisation du client Web** :

**I. Spécifications techniques :**

1. **Ossature du document HTML5 :**

* Établir une base structurée en HTML5 pour assurer la compatibilité avec les standards actuels.

1. **Intégration des librairies et frameworks :**

* Incorporer jQuery et jQueryUI pour ajouter des fonctionnalités interactives.
* Utiliser le framework Bootstrap pour la mise en forme et le responsive design.
* Intégrer les polices de caractères depuis Google Fonts pour enrichir le design typographique.
* Ajouter les icônes FontAwesome pour embellir l'interface.

1. **Mise en place du prototype HTML :**

* Se baser sur la maquette fonctionnelle fournie pour l'implémentation.
* Adopter le système de grille de Bootstrap pour garantir un affichage adaptatif sur divers dispositifs.

1. **Intégration des composants visuels :**

* Insérer des widgets et composants proposés par Bootstrap et jQueryUI.
* Ajouter d'autres composants nécessaires (boutons, formulaires, etc.) provenant de diverses librairies.

1. **Intégration du script JavaScript externe :**

* Incorporer un script externe pour gérer les fonctionnalités dynamiques.
* Dans ce script, définir l'URL de base du service Web.

1. **Interactions avec l'API :**

* Établir une connexion avec l'API via une requête Ajax de type GET.
* Récupérer la liste des vins et traiter les données reçues.

1. **Affichage des données :**

* Présenter la liste des vins récupérés dans la page Web de manière structurée et esthétique.

### 6 . Cas d'utilisation :

#### Pour tout utilisateur :

1. **Consultation du catalogue de vins** :

* **Acteur principal** : Utilisateur non authentifié
* **Scénario** : L'utilisateur accède à la page principale, et le système affiche la liste des vins disponibles.

1. **Visualiser les détails d'un vin spécifique** :

* **Acteur principal** : Utilisateur non authentifié
* **Scénario** : L'utilisateur clique sur un vin particulier, et le système présente les détails complets de ce vin.

1. **Rechercher des vins par nom** :

* **Acteur principal** : Utilisateur non authentifié
* **Scénario** : L'utilisateur saisit le nom ou une partie du nom d'un vin dans la barre de recherche, et le système affiche les vins correspondants.

1. **Trier la liste de vins par cépage** :

* **Acteur principal** : Utilisateur non authentifié
* **Scénario** : L'utilisateur sélectionne l'option de tri par cépage, et le système trie et affiche la liste de vins en fonction du cépage.

1. **Filtrer la liste de vins par pays et année** :

* **Acteur principal** : Utilisateur non authentifié
* **Scénario** : L'utilisateur sélectionne un pays et/ou une année depuis les filtres, et le système affiche la liste des vins correspondant aux critères sélectionnés.

#### Pour l'utilisateur authentifié :

1. **Gestion d'une cave à vin** :

* **Acteur principal** : Utilisateur authentifié
* **Scénarios** :
* **Ajout de bouteille** : L'utilisateur ajoute une bouteille à sa cave.
* **Modification de bouteille** : L'utilisateur modifie les détails d'une bouteille existante.
* **Suppression de bouteille** : L'utilisateur supprime une bouteille de sa cave.

1. **Gestion des commentaires** :

* **Acteur principal** : Utilisateur authentifié
* **Scénarios** :
* **Ajout de commentaire** : L'utilisateur ajoute un commentaire sur un vin.
* **Modification de commentaire** : L'utilisateur modifie un de ses commentaires précédemment ajoutés.
* **Suppression de commentaire** : L'utilisateur supprime un de ses commentaires.

1. **Gestion des notes personnelles** :

* **Acteur principal** : Utilisateur authentifié
* **Scénarios** :
* **Ajout/Modification de note** : L'utilisateur ajoute ou modifie une note personnelle pour un vin.
* **Suppression de note** : L'utilisateur supprime une note personnelle qu'il a précédemment ajoutée pour un vin.

1. **Modifier les détails d'un vin** :

* **Acteur principal** : Utilisateur authentifié
* **Scénario** : L'utilisateur peut modifier les détails d'un vin pour, par exemple, corriger une erreur ou mettre à jour une information.

**6. Critères d'acceptation**

* La page doit s'afficher correctement sur divers dispositifs et navigateurs.
* Toutes les librairies et frameworks doivent être correctement intégrés.
* La liste des vins doit être affichée de manière fluide et conforme aux données de l'API.
* Les interactions définies (ex : hover, clic) doivent fonctionner sans erreurs.

**7. Livrables :**

* Fichier HTML principal.
* Dossiers contenant les ressources (CSS, JS, images, etc.).
* Documentation décrivant l'architecture de la page et les choix techniques effectués.

**8.Contraintes** :

* Respecter l'architecture REST pour le service web.
* Assurer la compatibilité avec les navigateurs les plus courants.
* Veiller à la sécurité et à la performance de l'application

### Liste des librairies JavaScript

* Graphique
  + Chart.js – <https://www.chartjs.org/>
* Data visualisation
  + D3.js – <https://d3js.org/>
* Animation
  + Granim.js – <https://sarcadass.github.io/granim.js/>
  + Animate On Scroll Library – <https://michalsnik.github.io/aos/>
  + Slick (Carroussel) – <https://kenwheeler.github.io/slick/>
* Tableau de données
  + Datatable.js – <https://datatables.net/>
* Gestion de formulaire
  + Parsley (Validation) – <http://parsleyjs.org/>
  + Algolia Places (autocomplétion) – <https://community.algolia.com/places/>
* Notification
  + Popper.js – <https://popper.js.org/>
* Data Access Object
  + TaffyDB (JavaScript Database functionality) – <http://taffydb.com/>
* Cartographie
  + Open Layers – <https://openlayers.org/>
  + Leaflet (Cartographie) – <https://leafletjs.com/>