Rapport Projet Programmation Web
Gestion de Recettes de Cuisine

Bogdan GORELOV
Université Paris-Saclay

4 mai 2025

Table des matières

1 Introduction

Ce rapport présente le développement d'une application web visant à gérer un catalogue de recettes de cuisine, dans le cadre d'un projet d'initiation à la programmation web. L'objectif principal était de mettre en œuvre les technologies fondamentales du web (HTML, CSS, JavaScript, PHP) tout en mettant un accent particulier sur l'utilisation d'AJAX (Asynchronous JavaScript and XML) pour créer une interface utilisateur dynamique et réactive.

L'application permet la gestion de recettes multilingues (français/anglais) et intègre un système de rôles utilisateurs (Cuisinier, Chef, Traducteur, Administrateur) avec des permissions distinctes. Les données sont stockées et manipulées via des fichiers JSON, simulant une base de données simple. Ce projet a servi de plateforme d'apprentissage pour la manipulation du DOM, les interactions client-serveur asynchrones, la gestion de sessions PHP et les bases de la structuration d'un projet web.

2 Architecture et Rôle des Fichiers

L'application adopte une architecture client-serveur classique :

- **Client (Navigateur)** : Gère l'interface utilisateur (HTML/CSS) et l'interactivité (JavaScript/jQuery). Il initie des requêtes AJAX vers le serveur pour récupérer des données ou déclencher des actions sans recharger la page entière.
- **Serveur (PHP)** : Traite les requêtes reçues du client, interagit avec les fichiers de données JSON (lecture/écriture), gère les sessions utilisateur (authentification, rôles) et renvoie des réponses (souvent au format JSON) au client.

2.1 Découpage du Code

Le projet a été découpé en plusieurs fichiers afin de promouvoir la modularité, la réutilisabilité et la maintenabilité :

- **Séparation des préoccupations** : HTML pour la structure, CSS pour la présentation, JavaScript pour l'interactivité client, PHP pour la logique serveur.
- **Modularité** : Chaque fichier PHP traitant une requête AJAX (`auth.php`, `comment.php`, etc.) a une responsabilité unique, facilitant le débogage et l'évolution.
- **Réutilisabilité** : Le fichier `header.php` contient la structure commune (navigation, entête, pied de page) et le script JS global (gestion de la langue, authentification, fonction `showMessage`). Le fichier `dynamic_fields.js` est réutilisé par les pages de création et de modification de recettes.
- **Lisibilité** : Un découpage logique rend le code plus facile à lire et à comprendre qu'un unique fichier monolithique.

2.2 Rôle des Fichiers Principaux

Fichiers PHP (Interface) : `index.php` (accueil, liste recettes), `recipe.php` (détail recette), `create_recipe.php` / `modify_recipe.php` (formulaires recette), `translate_recipe.php` (interface traduction), `admin.php` (panel admin), `profile.php` (profil utilisateur). Ces fichiers génèrent la structure HTML initiale et incluent souvent du JavaScript spécifique.

Fichiers PHP (Backend/AJAX) : `auth.php` (login/signup), `comment.php` (ajout commentaire), `like_recipe.php` (gestion des likes), `update_users.php` (màj/suppression utilisateur, demande rôle), `update_recipes.php` (validation recette), `remove_recipe.php` (suppression recette), `logout.php`. Ils reçoivent les requêtes AJAX, traitent les données et renvoient une réponse JSON.

Fichiers JavaScript :

— Script dans `header.php` : Logique globale (chargement traductions, changement langue, authentification, `showMessage`, appel `initializePageContent`).
— Scripts spécifiques (dans chaque page PHP interface) : Fonction `initializePageContent` pour charger le contenu dynamique propre à la page (liste recettes, détails recette, formulaire, etc.) et attacher les écouteurs d'événements spécifiques.
— `dynamic_fields.js` : Gère l'ajout/suppression dynamique et synchronisé des champs ingrédients/étapes/minuteurs dans les formulaires.

Fichiers de Données : `recipes.json` (stockage des recettes), `users.json` (stockage utilisateurs/rôles/mdp hashés), `data.json` (traductions fr/en).

Fichiers CSS : `styles.css` (mise en forme visuelle de l'application).

3 Fonctionnalités Clés et Mécanismes Techniques

3.1 Appels AJAX

L'utilisation intensive d'AJAX est un point central du projet pour fluidifier l'expérience utilisateur. Le mécanisme général est le suivant :

1. Une action utilisateur (clic sur un bouton, soumission de formulaire interceptée) déclenche une fonction JavaScript.
2. Le JavaScript (via jQuery `$.ajax`, `$.getJSON` ou `$.post`) envoie une requête HTTP (GET ou POST) asynchrone vers un fichier PHP dédié sur le serveur. Les données nécessaires (ID recette, texte commentaire, identifiants, etc.) sont envoyées avec la requête.
3. Le fichier PHP serveur reçoit la requête, accède aux données POST/GET, vérifie la session et les permissions si nécessaire, lit/modifie les fichiers JSON appropriés, et prépare une réponse.
4. Le serveur renvoie la réponse au client, généralement au format JSON (ex : `{'success': true, 'likeCount': 5}`).
5. La fonction JavaScript (dans le callback `success`, `done` ou `complete` de l'appel AJAX) reçoit la réponse JSON.
6. Le JavaScript met à jour dynamiquement le DOM (l'interface HTML) en fonction de la réponse, sans recharger toute la page (ex : met à jour le compteur de likes, ajoute un commentaire à la liste, affiche un message de succès/erreur via `showMessage`).

Exemples concrets d'utilisation d'AJAX :

— **Authentification** (`header.js` ↔ `auth.php`) : Envoi login/mdp, réception statut succès/erreur et infos utilisateur. Rechargement nécessaire après login pour mettre à jour l'état global de la session et l'interface.
— **Likes** (`index.js`/`recipe.js` ↔ `like_recipe.php`) : Envoi ID recette, mise à jour du compteur et de l'état du bouton like.
— **Commentaires** (`recipe.js` ↔ `comment.php`) : Envoi ID recette, texte, et potentiellement image via `FormData`. Ajout dynamique du nouveau commentaire à la liste.
— **Gestion Admin** (`admin.js` ↔ `update_users.php`/`update_recipes.php`) : Modification rôles/mdp, suppression utilisateur, validation recette. Mise à jour directe du tableau HTML affiché.
— **Demande de Rôle** (`profile.js` ↔ `update_users.php`) : Mise à jour du rôle de l'utilisateur courant vers "DemandeChef/Traducteur". Mise à jour de l'affichage du rôle et de l'état des boutons.
— **Chargement de Données** (`index.js`, `admin.js`, etc. ↔ `*.json`) : Récupération initiale des listes de recettes ou utilisateurs via `$.getJSON`.
— **Traduction** (`header.js` ↔ `data.json`) : Chargement des traductions au démarrage et lors du changement de langue.

3.2 Gestion Dynamique des Champs (`dynamic_fields.js`)

Pour éviter de dupliquer le code JavaScript gérant l'ajout et la suppression des champs d'ingrédients, d'étapes et de minuteurs dans les pages `create_recipe.php` et `modify_recipe.php`, cette logique a été externalisée dans le fichier `dynamic_fields.js`.

Ce script fonctionne ainsi :

- Il attache des écouteurs d'événements aux boutons "Ajouter..." (`#add-ingredient`, `#add-step`, etc.).
- Au clic, il génère le code HTML nécessaire pour un nouveau champ (ou une paire de champs EN/FR pour ingrédients/étapes) avec les bons attributs `name` (ex : `ingredients[N][name]`) et l'ajoute au conteneur approprié (`#ingredients-container`, etc.).
- Pour la suppression, il utilise la délégation d'événements sur `document` pour écouter les clics sur les boutons ayant la classe `.remove-field`.
- Lorsqu'un bouton "supprimer" est cliqué, il identifie le champ parent à supprimer et lit son attribut `data-sync-type` ('ingredient', 'step' ou 'timer').
- Si le type est 'ingredient' ou 'step', il détermine le conteneur "frère" (EN ou FR) et supprime également le champ correspondant à la même position dans ce conteneur frère, assurant la synchronisation.
- Après chaque suppression, il appelle une fonction `reindexFields` qui parcourt les champs restants dans le(s) conteneur(s) affecté(s) et met à jour l'index numérique dans leurs attributs `name` (ex : `ingredients[2][name]` devient `ingredients[1][name]`). Ceci est crucial pour que le serveur PHP reçoive un tableau correctement indexé lors de la soumission du formulaire.

Cette approche centralisée simplifie la maintenance et assure un comportement cohérent sur les deux pages de formulaire.

4 Fonctionnalités Envisagées Non Implémentées

Au cours du développement, l'idée d'ajouter une fonctionnalité permettant d'uploader une photo pour chaque étape individuelle d'une recette a été envisagée. L'objectif était d'améliorer la clarté et la visualisation du processus de cuisson pour l'utilisateur final.

Cependant, cette fonctionnalité n'a pas été implémentée pour deux raisons principales :

1. **Complexité et Temps** : L'intégration d'un système d'upload multiple (une image par étape), la gestion du stockage de ces images sur le serveur, et leur association correcte avec chaque étape dans la structure JSON auraient considérablement augmenté la complexité et le temps de développement requis, dépassant le cadre d'un projet d'initiation.
2. **Stockage Disque** : Permettre l'upload de nombreuses images pour chaque recette aurait pu entraîner une consommation d'espace disque importante sur le serveur, ce qui n'était pas souhaitable pour une application basée sur des fichiers JSON et sans gestion avancée du stockage.

Il a donc été décidé de conserver uniquement l'upload d'une image principale par recette (via URL) et d'une image optionnelle par commentaire.

5 Difficultés Rencontrées

Le développement de ce projet a présenté plusieurs défis :

Intégration de la Traduction

L'application a initialement été conçue pour une seule langue (l'anglais). L'ajout ultérieur de la fonctionnalité multilingue (français/anglais) a nécessité une refonte significative. Il a fallu :

- Modifier la structure des données JSON (`recipes.json`) pour inclure les champs spécifiques à la langue (`nameFR`, `ingredientsFR`, `stepsFR`).
- Créer un fichier de traduction centralisé (`data.json`).
- Intégrer un mécanisme JavaScript (dans `header.php`) pour charger dynamiquement les traductions et mettre à jour l'interface via des attributs `data-translate`.
- Adapter toutes les fonctions JavaScript générant du HTML (`initializePageContent` dans chaque page) pour utiliser les traductions et choisir les bons champs de données selon la langue sélectionnée.
- Le débogage des erreurs de chargement de traductions ou des clés manquantes, qui pouvaient parfois "casser" l'exécution des scripts, a été particulièrement chronophage.

Gestion des Erreurs

Bien qu'une gestion basique des erreurs soit présente (via la fonction `showMessage` pour les retours AJAX), une gestion exhaustive n'a pas été implémentée. Cela aurait nécessité d'anticiper et de traiter des dizaines de cas : erreurs réseau lors des appels AJAX, fichiers JSON corrompus ou manquants, permissions de fichiers incorrectes sur le serveur, données invalides soumises par les formulaires, conflits d'accès concurrents aux fichiers JSON (si plusieurs utilisateurs modifient en même temps), etc. Compte tenu de l'objectif d'apprentissage des bases, il a été choisi de ne pas approfondir cet aspect très consommateur en temps.

Sécurité

La sécurisation d'une application web est un domaine vaste. Pour ce projet d'initiation, seules les protections les plus fondamentales ont été mises en place :

- Utilisation systématique de `htmlspecialchars` lors de l'affichage de données provenant de l'utilisateur ou des fichiers JSON, afin de prévenir les attaques XSS (Cross-Site Scripting).
- Hashage sécurisé des mots de passe lors de l'inscription et vérification via `password_hash` et `password_verify`.

D'autres aspects (protection CSRF, validation approfondie des entrées serveur, configuration sécurisée du serveur) n'ont pas été abordés.

Manipulation des Données (Champ Quantité)

Le fichier `recipes.json` initial contenait souvent la quantité et l'unité dans un seul champ texte (ex : "1/2 cup"). Pour l'interface de traduction (`translate_recipe.php`), il était nécessaire de séparer la partie numérique (non traduisible) de l'unité (traduisible). Cela a nécessité la création de la fonction PHP `splitQuantityLabel` utilisant une expression régulière (`preg_match`) pour extraire ces deux parties, ajoutant une complexité imprévue à la gestion des ingrédients.

Évolution de la Structure du Code

Au fil du développement, la structure du code et l'organisation des fichiers ont été modifiées et réorganisées plusieurs fois pour améliorer la logique ou intégrer de nouvelles fonctionnalités. Ce processus itératif, bien que normal, peut parfois conduire à des incohérences temporaires ou à des duplications mineures de code avant refactorisation finale.

6 Conclusion

Ce projet de gestion de recettes de cuisine a constitué une expérience d'apprentissage très riche sur les fondamentaux du développement web. La mise en œuvre des technologies HTML, CSS, JavaScript (avec jQuery) et PHP a permis de solidifier la compréhension de leur interaction dans une application dynamique.

L'utilisation intensive d'AJAX a été particulièrement formatrice. Elle a démontré concrètement comment créer des interfaces utilisateur fluides et réactives, où les actions (liker, commenter, rechercher, gérer) peuvent être effectuées sans nécessiter de rechargement complet de la page, améliorant significativement l'expérience utilisateur.

Le choix d'utiliser des fichiers JSON comme unique source de stockage de données, bien que limitatif pour une application à grande échelle (notamment en termes de performance et de gestion des accès concurrents), s'est avéré pédagogique. Il a permis de visualiser et de manipuler directement la structure des données et de comprendre les opérations de lecture/écriture serveur de manière simple.

Les difficultés rencontrées, notamment l'ajout tardif de la traduction et la gestion limitée des erreurs, ont souligné l'importance d'une conception initiale réfléchie et de l'anticipation des besoins futurs. L'implémentation, même basique, de mesures de sécurité comme le hashage des mots de passe et la protection contre les injections XSS via `htmlspecialchars`, a mis en lumière l'importance cruciale de cet aspect dans tout développement web.

Ce projet constitue une base solide pour aborder des technologies plus avancées. Ayant déjà une expérience avec des frameworks PHP comme Laravel lors de projets académiques précédents, les compétences acquises ici sur les mécanismes sous-jacents (requêtes HTTP, AJAX, manipulation DOM, gestion sessions PHP) seront précieuses pour mieux comprendre et exploiter ces outils plus puissants. L'importance d'une approche structurée et sécurisée du développement web, même pour des projets d'apprentissage, est la leçon principale retenue.