Cahier des Charges Simplifié - MVP Recettes de Cuisine (JavaScript Vanilla)

🎯 Version Ultra-Simple pour Dev Junior

Principe : "JavaScript pur, HTML simple, PHP basique"

---

1. OBJECTIF DU MVP

Une simple app de recettes où les utilisateurs peuvent :

- Se connecter
- Ajouter des recettes
- Voir les recettes des autres
- Mettre en favoris
- Laisser des commentaires

C'est tout pour commencer !

---

2. STACK TECHNIQUE ULTRA-SIMPLIFIÉE

Frontend

```
JavaScript Vanilla (ES6+)
├── HTML5 simple
├── CSS simple ou Tailwind CSS via CDN
├── Pas de bundler (Vite, Webpack, etc.)
├── Pas de framework JavaScript
└── LocalStorage pour certaines données
```

Backend

```
PHP simple (pas d'architecture complexe)
├── Un seul fichier api.php au début
├── PDO pour la base de données
└── Sessions PHP (pas de JWT pour commencer)
```

Base de données

```
MariaDB avec seulement 4 tables :
├── users
├── recipes
```

```
├── favorites
└── comments
```

---

3. STRUCTURE SIMPLIFIÉE

Frontend (HTML/JS/CSS)

```
/
├── index.html          // Page d'accueil
├── login.html          // Connexion/Inscription
├── add-recipe.html     // Ajouter une recette
├── recipe.html         // Voir une recette
├── my-recipes.html     // Mes recettes
├── assets/
│   ├── css/
│   │   └── style.css   // Styles globaux
│   └── js/
│       ├── app.js      // JavaScript principal
│       ├── auth.js     // Gestion authentification
│       ├── recipes.js  // Gestion recettes
│       └── utils.js    // Fonctions utilitaires
└── api/
    ├── config.php      // Connexion BDD
    ├── api.php        // Toutes les routes
    └── uploads/       // Dossier photos
```

---

4. FONCTIONNALITÉS ESSENTIELLES

Phase 1 : Le Strict Minimum (1 mois)

🔐 Authentification Super Simple

```
// Dans auth.js - Gestion simple avec localStorage
const Auth = {
    login: async function(email, password) {
        const response = await fetch('api/api.php?action=login', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({email, password})
        });

        if (response.ok) {
            const user = await response.json();
            localStorage.setItem('user', JSON.stringify(user));
            window.location.href = 'index.html';
        }
```

```
    },

    logout: function() {
        localStorage.removeItem('user');
        window.location.href = 'login.html';
    },

    isLoggedIn: function() {
        return localStorage.getItem('user') !== null;
    }
};
```

Recettes Basiques

```
// Structure minimale d'une recette
const recipe = {
    title: "Pâtes carbonara",
    ingredients: "Pâtes, œufs, lardons...", // Simple textarea
    instructions: "1. Faire cuire...",      // Simple textarea
    cooking_time: "30 min",
    photo: "url_photo.jpg"                  // Une seule photo
};
```

---

5. PAGES HTML SIMPLES

1 index.html - Page d'accueil

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mes Recettes</title>
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <header>
        <h1>Mes Recettes</h1>
        <nav>
            <button id="btn-add-recipe">Ajouter une recette</button>
            <button id="btn-my-recipes">Mes recettes</button>
            <button id="btn-logout">Déconnexion</button>
        </nav>
    </header>

    <main>
        <div id="recipes-container">
            <!-- Les recettes seront chargées ici -->
```

```
        </div>
    </main>

    <script src="assets/js/utils.js"></script>
    <script src="assets/js/auth.js"></script>
    <script src="assets/js/recipes.js"></script>
    <script src="assets/js/app.js"></script>
</body>
</html>
```

2 login.html - Connexion/Inscription

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Connexion - Mes Recettes</title>
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <div class="auth-container">
        <!-- Formulaire de connexion -->
        <form id="login-form" class="auth-form">
            <h2>Connexion</h2>
            <input type="email" id="login-email" placeholder="Email" required>
            <input type="password" id="login-password" placeholder="Mot de passe"
required>
            <button type="submit">Se connecter</button>
            <p><a href="#" id="show-register">Pas de compte ? S'inscrire</a></p>
        </form>

        <!-- Formulaire d'inscription (masqué par défaut) -->
        <form id="register-form" class="auth-form" style="display: none;">
            <h2>Inscription</h2>
            <input type="text" id="register-username" placeholder="Nom
d'utilisateur" required>
            <input type="email" id="register-email" placeholder="Email" required>
            <input type="password" id="register-password" placeholder="Mot de
passe" required>
            <button type="submit">S'inscrire</button>
            <p><a href="#" id="show-login">Déjà un compte ? Se connecter</a></p>
        </form>
    </div>

    <script src="assets/js/auth.js"></script>
    <script>
        // Gestion basique des formulaires
        document.getElementById('login-form').addEventListener('submit', async (e)
=> {
            e.preventDefault();
```

```
            const email = document.getElementById('login-email').value;
            const password = document.getElementById('login-password').value;
            await Auth.login(email, password);
        });

        // Basculer entre connexion et inscription
        document.getElementById('show-register').addEventListener('click', (e) =>
{
            e.preventDefault();
            document.getElementById('login-form').style.display = 'none';
            document.getElementById('register-form').style.display = 'block';
        });
    </script>
</body>
</html>
```

3 add-recipe.html - Ajouter une recette

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajouter une recette</title>
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <header>
        <h1>Ajouter une recette</h1>
        <a href="index.html">← Retour</a>
    </header>

    <main>
        <form id="recipe-form" enctype="multipart/form-data">
            <input type="text" id="recipe-title" placeholder="Titre de la recette"
required>

            <textarea id="recipe-ingredients" placeholder="Ingrédients..."
rows="5" required></textarea>

            <textarea id="recipe-instructions" placeholder="Instructions..."
rows="8" required></textarea>

            <input type="text" id="recipe-time" placeholder="Temps de cuisson (ex:
30 min)" required>

            <input type="file" id="recipe-photo" accept="image/*">

            <button type="submit">Ajouter la recette</button>
        </form>
    </main>
```

```
    <script src="assets/js/auth.js"></script>
    <script>
        // Vérifier si connecté
        if (!Auth.isLoggedIn()) {
            window.location.href = 'login.html';
        }

        // Gestion du formulaire
        document.getElementById('recipe-form').addEventListener('submit', async
(e) => {
            e.preventDefault();

            const formData = new FormData();
            formData.append('title', document.getElementById('recipe-
title').value);
            formData.append('ingredients', document.getElementById('recipe-
ingredients').value);
            formData.append('instructions', document.getElementById('recipe-
instructions').value);
            formData.append('cooking_time', document.getElementById('recipe-
time').value);

            const photo = document.getElementById('recipe-photo').files[0];
            if (photo) {
                formData.append('photo', photo);
            }

            try {
                const response = await fetch('api/api.php?action=add_recipe', {
                    method: 'POST',
                    body: formData
                });

                if (response.ok) {
                    alert('Recette ajoutée avec succès !');
                    window.location.href = 'index.html';
                }
            } catch (error) {
                alert('Erreur lors de l\'ajout de la recette');
            }
        });
    </script>
</body>
</html>
```

---

6. JAVASCRIPT PRINCIPAL

assets/js/app.js

```
// JavaScript principal pour index.html
document.addEventListener('DOMContentLoaded', function() {

    // Vérifier si connecté
    if (!Auth.isLoggedIn()) {
        window.location.href = 'login.html';
        return;
    }

    // Charger les recettes au démarrage
    loadRecipes();

    // Gestion des boutons
    document.getElementById('btn-add-recipe').addEventListener('click', () => {
        window.location.href = 'add-recipe.html';
    });

    document.getElementById('btn-my-recipes').addEventListener('click', () => {
        window.location.href = 'my-recipes.html';
    });

    document.getElementById('btn-logout').addEventListener('click', () => {
        Auth.logout();
    });
});

// Fonction pour charger et afficher les recettes
async function loadRecipes() {
    try {
        const response = await fetch('api/api.php?action=recipes');
        const recipes = await response.json();

        const container = document.getElementById('recipes-container');
        container.innerHTML = '';

        recipes.forEach(recipe => {
            const recipeCard = createRecipeCard(recipe);
            container.appendChild(recipeCard);
        });

    } catch (error) {
        console.error('Erreur lors du chargement des recettes:', error);
    }
}

// Créer une carte de recette
function createRecipeCard(recipe) {
    const card = document.createElement('div');
    card.className = 'recipe-card';
    card.innerHTML = `
        <img src="${recipe.photo || 'assets/images/default-recipe.jpg'}"
alt="${recipe.title}">
        <h3>${recipe.title}</h3>
```

```
            <p>⏱️ ${recipe.cooking_time}</p>
            <p>Par ${recipe.username}</p>
        `;

        card.addEventListener('click', () => {
            window.location.href = `recipe.html?id=${recipe.id}`;
        });

        return card;
    }
```

assets/js/utils.js

```
// Fonctions utilitaires
const Utils = {
    // Récupérer paramètre URL
    getUrlParameter: function(name) {
        const urlParams = new URLSearchParams(window.location.search);
        return urlParams.get(name);
    },

    // Formater une date
    formatDate: function(dateString) {
        const date = new Date(dateString);
        return date.toLocaleDateString('fr-FR');
    },

    // Afficher un message d'erreur
    showError: function(message) {
        alert(message); // Simple pour commencer
    },

    // Afficher un message de succès
    showSuccess: function(message) {
        alert(message); // Simple pour commencer
    }
};
```

---

7. CSS SIMPLE

assets/css/style.css

```
/* Reset basique */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

```css
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f4f4f4;
}

/* Header */
header {
    background: #fff;
    padding: 1rem;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    display: flex;
    justify-content: space-between;
    align-items: center;
}

header h1 {
    color: #e74c3c;
}

nav button {
    background: #3498db;
    color: white;
    border: none;
    padding: 0.5rem 1rem;
    margin-left: 0.5rem;
    cursor: pointer;
    border-radius: 4px;
}

nav button:hover {
    background: #2980b9;
}

/* Container principal */
main {
    max-width: 1200px;
    margin: 2rem auto;
    padding: 0 1rem;
}

/* Cartes de recettes */
#recipes-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 1rem;
}

.recipe-card {
    background: white;
    border-radius: 8px;
    overflow: hidden;
```

```
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    cursor: pointer;
    transition: transform 0.2s;
}

.recipe-card:hover {
    transform: translateY(-2px);
}

.recipe-card img {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

.recipe-card h3 {
    padding: 1rem 1rem 0.5rem;
    color: #2c3e50;
}

.recipe-card p {
    padding: 0 1rem 1rem;
    color: #7f8c8d;
}

/* Formulaires */
.auth-container {
    max-width: 400px;
    margin: 2rem auto;
    background: white;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

.auth-form input,
.auth-form textarea {
    width: 100%;
    padding: 0.75rem;
    margin-bottom: 1rem;
    border: 1px solid #ddd;
    border-radius: 4px;
}

.auth-form button {
    width: 100%;
    padding: 0.75rem;
    background: #e74c3c;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

```css
.auth-form button:hover {
    background: #c0392b;
}

/* Formulaire recette */
#recipe-form {
    max-width: 600px;
    margin: 0 auto;
    background: white;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

#recipe-form input,
#recipe-form textarea {
    width: 100%;
    padding: 0.75rem;
    margin-bottom: 1rem;
    border: 1px solid #ddd;
    border-radius: 4px;
}

#recipe-form button {
    width: 100%;
    padding: 1rem;
    background: #27ae60;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 1.1rem;
}

#recipe-form button:hover {
    background: #229954;
}

/* Responsive */
@media (max-width: 768px) {
    header {
        flex-direction: column;
        text-align: center;
    }

    nav {
        margin-top: 1rem;
    }

    #recipes-container {
        grid-template-columns: 1fr;
    }
}
```

8. BACKEND PHP (Identique)

api/api.php

```php
<?php
session_start();
header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, OPTIONS');
header('Access-Control-Allow-Headers: Content-Type');

require_once 'config.php';

$action = $_GET['action'] ?? '';

switch($action) {
    case 'login':
        $input = json_decode(file_get_contents('php://input'), true);
        $email = $input['email'];
        $password = $input['password'];

        $stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();

        if ($user && password_verify($password, $user['password'])) {
            $_SESSION['user_id'] = $user['id'];
            echo json_encode([
                'id' => $user['id'],
                'username' => $user['username'],
                'email' => $user['email']
            ]);
        } else {
            http_response_code(401);
            echo json_encode(['error' => 'Email ou mot de passe incorrect']);
        }
        break;

    case 'recipes':
        $stmt = $pdo->query("
            SELECT r.*, u.username
            FROM recipes r
            JOIN users u ON r.user_id = u.id
            ORDER BY r.created_at DESC
        ");
        echo json_encode($stmt->fetchAll());
        break;

    case 'add_recipe':
        if (!isset($_SESSION['user_id'])) {
```

```
            http_response_code(401);
            exit(json_encode(['error' => 'Non connecté']));
        }

        $title = $_POST['title'];
        $ingredients = $_POST['ingredients'];
        $instructions = $_POST['instructions'];
        $time = $_POST['cooking_time'];

        // Upload photo simple
        $photo = '';
        if (isset($_FILES['photo'])) {
            $photo = 'uploads/' . time() . '_' . $_FILES['photo']['name'];
            move_uploaded_file($_FILES['photo']['tmp_name'], $photo);
        }

        $stmt = $pdo->prepare("
            INSERT INTO recipes (user_id, title, ingredients, instructions,
cooking_time, photo)
            VALUES (?, ?, ?, ?, ?, ?)
        ");
        $stmt->execute([$_SESSION['user_id'], $title, $ingredients, $instructions,
$time, $photo]);

        echo json_encode(['success' => true]);
        break;
}
?>
```

---

9. BASE DE DONNÉES (Identique)

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(255) UNIQUE,
    password VARCHAR(255),
    username VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE recipes (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    title VARCHAR(255),
    ingredients TEXT,
    instructions TEXT,
    cooking_time VARCHAR(50),
    photo VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

```
CREATE TABLE favorites (
    user_id INT,
    recipe_id INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, recipe_id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (recipe_id) REFERENCES recipes(id)
);

CREATE TABLE comments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    recipe_id INT,
    user_id INT,
    content TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (recipe_id) REFERENCES recipes(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

---

10. PLANNING RÉALISTE

Semaine 1-2 : Base HTML/CSS

- [ ] Créer les 5 pages HTML
- [ ] CSS basique responsive
- [ ] Structure des dossiers

Semaine 3-4 : JavaScript + Auth

- [ ] Système de login en JavaScript
- [ ] Navigation entre pages
- [ ] LocalStorage pour session

Semaine 5-6 : Recettes

- [ ] Affichage des recettes
- [ ] Ajout de recettes avec photos
- [ ] Page détail recette

Semaine 7-8 : Finalisation

- [ ] Favoris et commentaires
- [ ] Tests et corrections
- [ ] Amélioration CSS

💡 AVANTAGES DU JAVASCRIPT VANILLA

☑ **Plus simple à comprendre** : Pas de syntaxe de framework à apprendre ☑ **Pas de compilation** : Ouvrir le fichier HTML dans le navigateur ☑ **Débogage facile** : Console du navigateur directement ☑ **Léger** : Pas de dépendances ☑ **Apprentissage des bases** : Comprendre le DOM et les APIs natives

---

Cette version vanilla JavaScript est encore plus accessible et permet de bien comprendre les fondamentaux avant de passer aux frameworks !