Une image contenant logo

Description générée automatiquement

**EatFit**

**Rapport TPI et documentation technique**

**Avril – Mai 2023**

**Almeida Costa Lucas**

**Maitre d’apprentissage : Katia Mota Stroppolo**

**Table des matières**

[**Introduction** 4](#_Toc135138932)

[**Rappel de l'énoncé** 4](#_Toc135138933)

[Description de l’application 4](#_Toc135138934)

[Organisation 5](#_Toc135138935)

[Livrables 5](#_Toc135138936)

[Matériel et logiciels à disposition 5](#_Toc135138937)

[**Méthodologie** 5](#_Toc135138938)

[Planification en 6 étapes 5](#_Toc135138939)

[**Planification** 7](#_Toc135138940)

[Planification prévue 7](#_Toc135138941)

[Planification effective 7](#_Toc135138942)

[Product backlog 7](#_Toc135138943)

[**Analyse organique** 11](#_Toc135138944)

[**Front-end** 11](#_Toc135138945)

[**Api (Interface de Programmation d’Applications)** 11](#_Toc135138946)

[**Back-end** 12](#_Toc135138947)

[Architecture de l’api 12](#_Toc135138948)

[**Classes** 12](#_Toc135138949)

[Architecture Front 15](#_Toc135138950)

[**Classes** 16](#_Toc135138951)

[Base de données 17](#_Toc135138952)

[**Informations techniques** 17](#_Toc135138953)

[**MLD** 18](#_Toc135138954)

[Tables 18](#_Toc135138955)

[**Généralités concernant l'implémentation** 22](#_Toc135138956)

[Langages (versions) 22](#_Toc135138957)

[Environnement (local et/ou distant) 23](#_Toc135138958)

[**Plan de test** 23](#_Toc135138959)

[Périmètre 23](#_Toc135138960)

[Environnement 23](#_Toc135138961)

[Scénarios (les tests, les données et leurs évolution) 23](#_Toc135138962)

[Evolution des tests 30](#_Toc135138963)

[**Conclusion** 32](#_Toc135138964)

[Difficultés rencontrées 32](#_Toc135138965)

[Variantes de solutions et choix 32](#_Toc135138966)

[Améliorations possibles 32](#_Toc135138967)

[Bilan personnel 32](#_Toc135138968)

[Remerciements 32](#_Toc135138969)

[**Bibliographie** 32](#_Toc135138970)

[**Annexes** 33](#_Toc135138971)

[Conversations ChatGPT 33](#_Toc135138972)

[**Graphique en js** 33](#_Toc135138973)

[Glossaire 45](#_Toc135138974)

# **Introduction**

Ce document est un rapport présentant différents aspects de la conception du projet Eatfit. Ce projet a été réalisé dans le cadre du *Travail pratique individuel* (TPI) durant la session de mai 2023. Il a pour but de valider mes compétences acquises pendant la formation *Informaticien CFC* dispensée à l’école d’informatique du CFPT au Petit-Lancy.

EatFit est un site web de gestion de recettes culinaires permettant aux utilisateurs de rechercher, créer, modifier et partager des recettes en fonction de leur rôle (visiteur non-authentifié, membre authentifié ou administrateur technique). Les utilisateurs peuvent également gérer leur profil, consulter l'historique de leurs recettes consommées et visualiser un graphique montrant la répartition des types d'aliments consommés.

# **Rappel de l'énoncé**

*Les informations suivantes sont extraites du cahier de charges du TPI.*

## **Description de l’application**

L'application est un site web de gestion de recettes culinaires qui offre aux utilisateurs une plateforme interactive pour explorer, créer et partager des recettes. Elle est conçue pour être utilisée par trois types d'utilisateurs : les visiteurs non-authentifiés, les membres authentifiés et les administrateurs techniques.

Fonctionnalités pour les visiteurs non-authentifiés :

* Rechercher des recettes par nom, catégorie ou date d'ajout.
* Filtrer les recettes par catégorie ou type d'aliments.
* Lire la liste des recettes triées par ordre de publication, du plus récent au plus ancien.
* Sélectionner une recette et afficher les détails, incluant le titre, la catégorie, l'auteur, la durée de préparation, la difficulté, l'évaluation moyenne, les commentaires et les images.
* Lire les informations sur le fonctionnement du site.
* Créer un nouveau compte ou se connecter à un compte existant.

Fonctionnalités pour les membres authentifiés :

* Accéder à tous les droits des visiteurs non-authentifiés.
* Afficher des détails supplémentaires sur les recettes, tels que les instructions de préparation, le nombre de calories et la répartition des types d'aliments.
* Gérer un historique de recettes consommées (ajouter, lire, modifier, supprimer) avec des informations sur la date et le moment de la consommation.
* Afficher un graphique montrant la répartition des types d'aliments consommés sur différentes périodes (1 semaine, 1 mois, 1 an ou toute la période enregistrée).
* Ajouter de nouveaux types d'aliments.
* Gérer ses propres recettes (créer, lire, modifier, supprimer).
* Évaluer et commenter les recettes.
* Gérer son propre profil (lire, modifier, supprimer).

Fonctionnalités pour les administrateurs techniques :

* Gérer la base de données via l'interface d'administration avec un rôle administrateur.
* Maintenir le code source de l'application.

Le site web est structuré de manière logique, avec un découpage en user stories pour faciliter la compréhension et l'organisation du travail. La méthodologie suivie comporte six étapes : s'informer, planifier, décider, réaliser, contrôler et évaluer. Les tests fonctionnels et de sécurité sont réalisés pour s'assurer que les fonctionnalités et les droits des utilisateurs sont correctement implémentés et gérés.

## **Organisation**

|  |  |
| --- | --- |
| **Élève** | **Maîtresse d’apprentissage** |
| Lucas Almeida Costa  <lucas.almdc@eduge.ch> | Katia Mota Stroppolo  <katia.motastroppolo@edu.ge.ch> |

|  |  |
| --- | --- |
| **Experts** | |
| Brian Nydegger  <briannydegger@gmail.com> | Borys Folomietow  <borys@folomietow.ch> |

## **Livrables**

* Résumé du TPI sur une page A4
* Rapport de projet (réalisé en continu ; livraison intermédiaire à la fin de la première semaine, livraison finale à la fin du projet)
* Code source de l'application en annexe
* Manuel utilisateur
* Journal de travail (disponible pour consultation à tout moment par les experts et l'enseignante)

## **Matériel et logiciels à disposition**

* Un PC standard école, 2 écrans
* Windows 10
* Visual Studio code
* MySQLWorkbench
* DBeaver
* Suite Office
* Serveur Web, avec PHP

# **Méthodologie**

## **Planification en 6 étapes**

**1. S’informer**

Lors de la première étape de mon projet, j'ai effectué une analyse approfondie du cahier des charges afin de comprendre toutes les fonctionnalités à implémenter pour l'application de gestion de recettes culinaires. J'ai également consulté diverses ressources en ligne et des forums de discussion pour m'informer sur les meilleures pratiques et les technologies à utiliser. Lorsque nécessaire, j'ai sollicité l'aide de ma formatrice pour obtenir des éclaircissements sur certains aspects du projet.

**2. Planifier**

J'ai préparé un planning de travail détaillé pour organiser les différentes tâches à réaliser et déterminer les délais associés. J'ai divisé le projet en plusieurs user stories, en tenant compte des différents types d'utilisateurs et des fonctionnalités à implémenter pour chacun d'eux. Pour chaque user story, j'ai attribué un niveau de priorité, allant de P1 (bloquant) à P4 (secondaire), afin de déterminer l'ordre d'exécution des tâches.

**3. Décider**

Tout au long du projet, j'ai dû prendre des décisions concernant l'architecture de l'application, les technologies à utiliser et les solutions à adopter pour résoudre des problèmes spécifiques. J'ai consigné ces décisions dans mon journal de bord, en expliquant les raisons qui m'ont conduit à opter pour certaines options plutôt que d'autres.

**4. Réaliser**

Après avoir pris les décisions appropriées, j'ai procédé à l'implémentation des fonctionnalités dans le code source de l'application. J'ai travaillé de manière itérative, en développant et en testant chaque fonctionnalité individuellement avant de passer à la suivante.

**5. Contrôler**

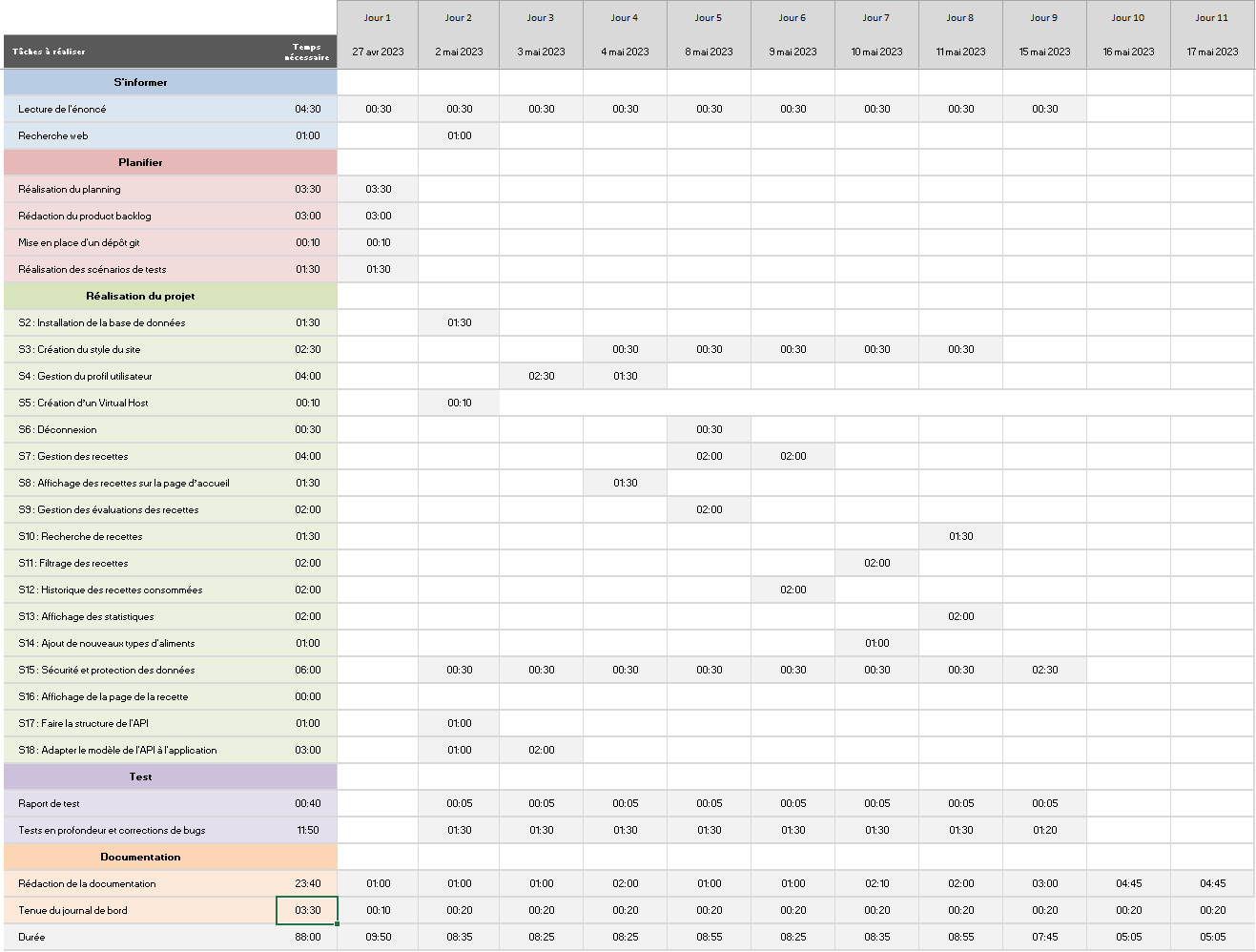
À chaque étape de réalisation, j'ai effectué des tests rigoureux pour m'assurer que chaque fonctionnalité fonctionnait correctement et répondait aux exigences du cahier des charges. J'ai documenté ces tests dans mon rapport de projet, en décrivant les scénarios de test et les résultats obtenus. J'ai également réalisé des tests de compatibilité sur différents navigateurs pour m'assurer que l'application fonctionnait correctement.

**6.Evaluer**

Enfin, j'ai procédé à une évaluation globale du projet, en examinant les réussites et les points à améliorer. J'ai rédigé des bilans journaliers dans mon journal de bord pour analyser les progrès réalisés et identifier les éventuels obstacles rencontrés. Dans la partie Conclusion de mon rapport de projet, j'ai présenté une auto-évaluation complète, en abordant les aspects positifs et négatifs du travail accompli et en proposant des pistes d'amélioration pour l'avenir.

# **Planification**

## **Planification prévue**



## **Planification effective**

## **Product backlog**

|  |  |
| --- | --- |
| Nom | S1 : Utilisation d’un dépôt Git |
| Description | En tant que développeur, je dois utiliser un dépôt Git afin d’enregistrer et accéder à toutes les versions de mon projet. |
| Critère d’acceptation | Un dépôt Git public est créé, et les modifications du projet sont régulièrement poussées sur ce dépôt. |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S2 : Installation de la base de données |
| Description | En tant que développeur, je dois installer une base de données. Pour ce faire, j’utilise le script SQL mis à disposition par mon formateur. |
| Critère d’acceptation | La base de données est créée ainsi qu’un utilisateur qui a les droits que sur cette base de données. |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S3 : Création du style du site |
| Description | En tant que développeur, je dois avoir un site visuel pour me concentrer seulement sur le back-end de celui-ci. |
| Critère d’acceptation | Les pages mentionnées dans les consignes du TPI sont créées et disposent des éléments requis. |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S4 : Gestion du profil utilisateur |
| Description | En tant qu'utilisateur identifié, je peux afficher et gérer mon profil (lire, modifier, supprimer). |
| Critère d’acceptation | Les modifications apportées au profil sont enregistrées et appliquées.  Tests : |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S5 : Création d’un Virtual Host |
| Description | En tant que développeur, je dois créer un Virtual Host qui me permettra d’utiliser un « router ». |
| Critère d’acceptation | Un Virtual Host est créé, permettant d'utiliser un "router" pour naviguer entre les différentes pages du site.  Test : |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S6 : Déconnexion |
| Description | En tant qu’utilisateur identifié, je peux me déconnecter de mon compte. |
| Critère d’acceptation | L'utilisateur identifié est déconnecté, et la session est terminée.  Test : |
| Priorité | P3 : Important |

|  |  |
| --- | --- |
| Nom | S7 : Gestion des recettes |
| Description | En tant qu’utilisateur identifié, je peux ajouter, modifier ou supprimer des recettes. |
| Critère d’acceptation |  |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S8 : Affichage des recettes sur la page d’accueil |
| Description | En tant qu’utilisateur identifié, je dois voir les recettes par catégories (petit déjeuner, entrée, etc…) |
| Critère d’acceptation | Les recettes sont affichées sur la page d'accueil, classées par catégorie (petit déjeuner, entrée, etc.). |
| Priorité | P2 : Critique |

|  |  |
| --- | --- |
| Nom | S9 : Gestion des évaluations des recettes |
| Description | En tant qu’utilisateur identifié, je peux évaluer de 1 à 5 et commenter, je peux aussi modifier et supprimer l’évaluation |
| Critère d’acceptation |  |
| Priorité | P3 : Important |

|  |  |
| --- | --- |
| Nom | S10 : Recherche de recettes |
| Description | En tant qu'utilisateur, je peux rechercher des recettes en fonction de leur nom, catégorie ou date d'ajout. |
| Critère d’acceptation | La recherche retourne les résultats correspondants aux critères de recherche.  Tests : |
| Priorité | P2 : Critique |

|  |  |
| --- | --- |
| Nom | S11 : Filtrage des recettes |
| Description | En tant qu'utilisateur, je peux filtrer les recettes par catégorie ou type d'aliments. |
| Critère d’acceptation | Les recettes sont filtrées en fonction des critères sélectionnés.  Tests : |
| Priorité | P2 : Critique |

|  |  |
| --- | --- |
| Nom | S12 : Historique des recettes consommées |
| Description | En tant qu'utilisateur identifié, je peux consulter et gérer l'historique de mes recettes consommées. |
| Critère d’acceptation | L'historique affiche correctement les recettes consommées et permet leur gestion (ajouter, lire, modifier, supprimer).  Tests : |
| Priorité | P2 : Critique |

|  |  |
| --- | --- |
| Nom | S13 : Affichage des statistiques |
| Description | En tant qu'utilisateur identifié, je peux afficher un graphique basé sur les données de l'historique des recettes consommées, montrant la répartition des proportions de types d'aliments consommés sur différentes périodes. |
| Critère d’acceptation | Le graphique est généré en fonction des données de l'historique et affiche les répartitions correctes pour les périodes sélectionnées.  Tests : |
| Priorité | P3 : Important |

|  |  |
| --- | --- |
| Nom | S14 : Ajout de nouveaux types d'aliments |
| Description | En tant qu'utilisateur identifié, je peux ajouter de nouveaux types d'aliments. |
| Critère d’acceptation | Les nouveaux types d'aliments sont ajoutés et peuvent être utilisés pour filtrer les recettes  Tests : |
| Priorité | P3 : Important |

|  |  |
| --- | --- |
| Nom | S15 : Sécurité et protection des données |
| Description | En tant que développeur, je dois m'assurer que le site est sécurisé et protège les données des utilisateurs. |
| Critère d’acceptation | Les données sont stockées et transmises de manière sécurisée, et les vulnérabilités courantes (injections SQL, XSS, CSRF, etc.) sont traitées.  Tests : |
| Priorité | P4 : Secondaire |

|  |  |
| --- | --- |
| Nom | S16 : Affichage de la page de la recette |
| Description | En tant qu’utilisateur, je veux regarder une recette détaillée |
| Critère d’acceptation | Tests : |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S17 : Faire la structure de l'API |
| Description | En tant que développeur, je dois créer la structure de base de l'API pour gérer les requêtes entre l'application et la base de données. |
| Critère d’acceptation | L'API est correctement structurée avec les routes, les contrôleurs et les modèles nécessaires pour gérer les requêtes.  Tests : |
| Priorité | P1 : Bloquant |

|  |  |
| --- | --- |
| Nom | S18 : Adapter le modèle de l'API à l'application |
| Description | En tant que développeur, je dois adapter le modèle de l'API pour qu'il corresponde aux besoins de l'application et aux relations entre les différentes entités (utilisateurs, recettes, évaluations, etc.). |
| Critère d’acceptation | Le modèle de l'API est adapté à l'application, et les relations entre les entités sont correctement gérées.  Tests : |
| Priorité | P1 : Bloquant |

# **Analyse organique**

L'application EatFit se compose de trois principales composantes : le front-end, l'API et le back-end.

### **Front-end**

Le front-end est l'interface utilisateur de l'application EatFit. Il s'agit de la partie de l'application qui est directement visible et utilisable par les utilisateurs. Le front-end peut être une interface web, une application mobile, ou tout autre client qui communique avec l'API.

Il est conçu pour fournir une expérience utilisateur fluide et intuitive, permettant aux utilisateurs de naviguer facilement à travers les différentes recettes, de les noter et de les ajouter à leur historique personnel.

### **Api (Interface de Programmation d’Applications)**

L'API est le pont entre le front-end et le back-end. Elle est conçue pour accepter les requêtes provenant du front-end, traiter ces requêtes en fonction de la logique métier définie dans le back-end, puis renvoyer les réponses appropriées au front-end.

L'API d'EatFit est basée sur le protocole HTTP et suit les principes de l'architecture REST. Elle comprend un ensemble de points d'accès (ou "routes") qui correspondent à différentes actions que le front-end peut demander, comme la création d'une nouvelle recette ou la récupération d'une liste de recettes.

*Image : https://player.support.brightcove.com/getting-started/learning-guide-using-rest-apis.html*

Une image contenant diagramme, texte, capture d’écran, conception

Description générée automatiquement

### **Back-end**

Le back-end est le cœur de l'application EatFit, où se trouve la logique métier et où les données sont stockées et récupérées. Il est composé de plusieurs composants, y compris les contrôleurs, les modèles et la base de données.

Les contrôleurs reçoivent les requêtes de l'API, effectuent les actions appropriées en utilisant les modèles, puis renvoient les réponses à l'API. Les modèles sont des représentations des structures de données de l'application, et ils contiennent les méthodes pour interagir avec la base de données.

La base de données est le dépôt où toutes les données de l'application sont stockées. Cela inclut les recettes, les utilisateurs, les notations et autres données pertinentes.

Cette architecture permet à EatFit d'être à la fois robuste et flexible, capable de s'adapter à divers clients front-end et de gérer une large gamme de requêtes d'utilisateurs.

## **Architecture de l’api**

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

### **Classes**

Pour implémenter les diverses fonctionnalités de l’api j’ai créé les classes suivantes :

Eatfit\Api\Controllers\**UserController**

Gère les opérations CRUD (Create, Read, Update, Delete) pour les utilisateurs, ainsi que les fonctionnalités d'authentification.

Eatfit\Api\Controllers\**RecipeController**

Responsable des opérations CRUD pour les recettes, y compris les fonctionnalités de recherche et de filtrage.

Eatfit\Api\Controllers\**RatingController**

Gère le CRUD pour les évaluations des recettes.

Eatfit\Api\Controllers\**HistoryController**

Eatfit\Api\Core\Db\**Database**

Fournit une interface pour se connecter à la base de données et effectuer des requêtes.

Eatfit\Api\Core\Db\**SqlResult**

Fournit une interface pour traiter les résultats des requêtes SQL.

Eatfit\Api\Core\**ApiValue**

Gère les réponses de l'API, y compris la codification des erreurs et la fourniture de messages d'erreur.

Eatfit\Api\Core\**Application**

Classe principale de l'application qui gère l'initialisation, le routage, et la gestion des erreurs.

Eatfit\Api\Core\**EnvLoader**

Charge les variables d'environnement à partir d'un fichier .env pour une utilisation dans l'application.

Eatfit\Api\Core\**Model**

Classe de base pour tous les modèles, fournit des fonctionnalités communes pour la gestion des données d'entités comme les utilisateurs et les recettes.

Eatfit\Api\Core\**Request**

Gère les requêtes entrantes, extrait les informations à partir des en-têtes, des paramètres de requête et des données POST.

Eatfit\Api\Core\**Router**

Gère le routage des requêtes vers les contrôleurs appropriés en fonction de l'URL et de la méthode HTTP.

Eatfit\Api\Models\**User**

Modèle pour les utilisateurs, gère les interactions avec la base de données pour les informations des utilisateurs.

Eatfit\Api\Models\**Recipe**

Modèle pour les recettes, gère les interactions avec la base de données pour les informations des recettes.

Toutes ces classes travaillent ensemble pour fournir les fonctionnalités de l'application EatFit. Elles définissent la structure de l’API, permettant une séparation claire des préoccupations et une organisation logique du code.

Routes

GET '/' : Cette route renvoie une liste de toutes les routes disponibles dans l'application.

GET '/history' : Cette route renvoie l'historique de l'utilisateur connecté. Elle utilise la méthode 'read' du contrôleur 'HistoryController'.

POST '/history' : Cette route ajoute une entrée à l'historique de l'utilisateur connecté. Elle utilise la méthode 'create' du contrôleur 'HistoryController'.

PUT '/history' : Cette route met à jour une entrée de l'historique de l'utilisateur connecté. Elle utilise la méthode 'update' du contrôleur 'HistoryController'.

DELETE '/history' : Cette route supprime une entrée de l'historique de l'utilisateur connecté. Elle utilise la méthode 'delete' du contrôleur 'HistoryController'.

POST '/food\_type' : Cette route crée un nouveau type de nourriture. Elle utilise la méthode 'addFoodType' du contrôleur 'RecipeController'.

GET '/food\_types' : Cette route renvoie tous les types de nourriture disponibles. Elle utilise la méthode 'getFoodTypes' du contrôleur 'RecipeController'.

GET '/categories' : Cette route renvoie toutes les catégories de recettes disponibles. Elle utilise la méthode 'getCategories' du contrôleur 'RecipeController'.

PUT '/login' : Cette route est utilisée pour authentifier un utilisateur. Elle utilise la méthode login du contrôleur UserController, qui prend un email et un mot de passe en entrée et renvoie un jeton d'accès si les informations sont correctes.

GET '/userById' : Cette route récupère les détails d'un utilisateur à partir de son jeton d'accès. Elle utilise la méthode getUserByToken du contrôleur UserController.

POST '/history' : Cette route ajoute une recette à l'historique d'un utilisateur. Elle utilise la méthode addHistory du contrôleur RecipeController, qui prend l'ID de la recette comme argument.

GET '/user' : Cette route renvoie les détails de l'utilisateur actuellement connecté. Elle utilise la méthode read du contrôleur UserController, qui prend l'email et le mot de passe de l'utilisateur comme arguments.

POST '/user' : Cette route crée un nouvel utilisateur. Elle utilise la méthode create du contrôleur UserController, qui prend l'email, le mot de passe, la confirmation du mot de passe et le nom d'utilisateur comme arguments.

PUT '/user' : Cette route met à jour les détails de l'utilisateur actuellement connecté. Elle utilise la méthode update du contrôleur UserController, qui prend l'email, le mot de passe et le nom d'utilisateur comme arguments.

DELETE '/user' : Cette route supprime l'utilisateur actuellement connecté. Elle utilise la méthode delete du contrôleur UserController.

GET '/recipe' : Cette route renvoie les détails de toutes les recettes disponibles. Elle utilise la méthode read du contrôleur RecipeController, qui peut prendre des filtres de recherche comme arguments.

POST '/recipe' : Cette route crée une nouvelle recette. Elle utilise la méthode create du contrôleur RecipeController, qui prend les détails de la recette comme arguments.

PUT '/recipe' : Cette route met à jour une recette existante. Elle utilise la méthode update du contrôleur RecipeController, qui prend l'ID de la recette et les nouveaux détails comme arguments.

DELETE '/recipe' : Cette route supprime une recette existante. Elle utilise la méthode delete du contrôleur RecipeController, qui prend l'ID de la recette comme argument.

GET '/rating' : Cette route renvoie les évaluations de toutes les recettes. Elle utilise la méthode read du contrôleur RatingController.

POST '/rating' : Cette route crée une nouvelle évaluation pour une recette. Elle utilise la méthode create du contrôleur RatingController, qui prend le score de l'évaluation, le commentaire et l'ID de la recette comme arguments.

PUT '/rating' : Cette route met à jour une évaluation existante. Elle utilise la méthode update du contrôleur RatingController, qui prend l'ID de l'évaluation, le nouveau score et le nouveau commentaire comme arguments.

DELETE '/rating' : Cette route supprime une évaluation existante. Elle utilise la méthode delete du contrôleur RatingController, qui prend l'ID de l'évaluation comme argument.

POST '/food\_type' : Cette route ajoute un nouveau type de nourriture. Elle utilise la méthode addFoodType du contrôleur RecipeController, qui prend le nom du nouveau type de nourriture comme argument.

Chaque route est liée à une fonction spécifique dans un contrôleur spécifique, qui effectue une action spécifique en fonction de la demande de l'utilisateur. Cette architecture permet de garder votre code organisé et facile à comprendre.

## **Architecture Front**

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

### **Classes**

Eatfit\Site\Controllers\HistoryController : Contrôleur pour gérer l'historique des recettes consultées/utilisées par un utilisateur.

Eatfit\Site\Controllers\**RecipeController** : Contrôleur pour gérer les opérations relatives aux recettes (par exemple, création, détails, mise à jour, suppression).

Eatfit\Site\Controllers\**SiteController** : Contrôleur principal de l'application, probablement pour gérer les vues et les opérations générales du site.

Eatfit\Site\Controllers\**UserController** : Contrôleur pour gérer les opérations relatives à l'utilisateur (par exemple, connexion, déconnexion, inscription, profil).

Eatfit\Site\Core\Exception\**ForbiddenException** : Une exception personnalisée pour gérer les scénarios d'accès interdit.

Eatfit\Site\Core\Exception\**NotFoundException** : Une exception personnalisée pour gérer les scénarios ou la page n’existe pas.

Eatfit\Site\Core\Form\**BaseField** : Classe de base pour les champs de formulaire.

Eatfit\Site\Core\Form\**Field** : Classe pour représenter un champ de formulaire spécifique.

Eatfit\Site\Core\Form\**Form** :Permet de construire une balise <form> et de créer des champs.

Eatfit\Site\Core\Middlewares\**AuthMiddleware** : Middleware pour gérer l'authentification des utilisateurs.

Eatfit\Site\Core\Middlewares\**BaseMiddleware** : Classe de base pour tous les middlewares.

Eatfit\Site\Core\**Application** : Classe principale de l'application qui pourrait initialiser et configurer l'application.

Eatfit\Site\Core\**Controller** : Classe de base pour tous les contrôleurs.

Eatfit\Site\Core\**Model** : Classe de base pour tous les modèles.

Eatfit\Site\Core\**Request** : Classe pour gérer les requêtes entrantes.

Eatfit\Site\Core\**Response** : Classe pour gérer les réponses sortantes.

Eatfit\Site\Core\**Router** : Classe pour gérer le routage de l'application.

Eatfit\Site\Core\**Session** : Classe pour gérer les sessions des utilisateurs.

Eatfit\Site\Core\**View** : Classe pour gérer les vues de l'application.

Eatfit\Site\Core\Models\**FoodType** : Modèle pour représenter un type d'aliment.

Eatfit\Site\Core\Models\**History** : Modèle pour représenter l'historique de consultation/utilisation des recettes par un utilisateur.

Eatfit\Site\Core\Models\**LoginForm** : Modèle pour représenter le formulaire de connexion.

Eatfit\Site\Core\Models\**ProfileModel** : Modèle pour représenter le profil d'un utilisateur.

Eatfit\Site\Core\Models\**Rating** : Modèle pour représenter une évaluation.

Eatfit\Site\Core\Models\**Recipe** : Modèle pour représenter une recette.

Eatfit\Site\Core\Models\**User** : Modèle pour représenter un utilisateur.

Routes :

'/' : La page d'accueil du site.

'/recipe' : Page pour créer une nouvelle recette.

'/recipe/detail/{idRecipe}' : Page de détail d'une recette spécifique.

'recipe/edit/{idRecipe}' : Page pour modifier une recette spécifique.

'recipe/delete/{idRecipe}' : Route pour supprimer une recette spécifique.

'rating/delete/{idRating}' et 'rating/update/{idRating}' : Routes pour supprimer et mettre à jour une évaluation spécifique.

'/food\_type' : Page pour ajouter un nouveau type d'aliment.

'/login' et '/logout' : Pages pour se connecter et se déconnecter.

'/register' : Page pour s'enregistrer en tant que nouvel utilisateur.

'/profile' et '/profile/{method}/' : Pages pour afficher et mettre à jour le profil d'un utilisateur.

'/addToHistory/{idRecipe}' and '/history' : Pages pour ajouter une recette à l'historique d'un utilisateur et pour afficher l'historique.

'/history/delete' et '/history/delete/{idConsumedRecipe}' : Routes pour supprimer tout l'historique ou une recette spécifique de l'historique d'un utilisateur.

## **Base de données**

### **Informations techniques**

EatFit utilise une base de données MySQL pour stocker les informations des recettes, les utilisateurs et l’historique de chaque recette consommée. J’ai utilisé InnoDB comme moteur de stockage et utf8\_general\_ci comme interclassement.

### **MLD**

Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

## **Tables**

**categories**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idCategory | Int | Non |  |
| name | Varchar (255) | Non |  |
| Image\_path | Varchar (255) | Non |  |

Dans cette table :

* Le champ « idCategory » est la clé primaire
* Le champ name permet de stocker le nom de la catégorie (ex : Petit déjeuner)
* Le champ image\_path permet de stocker l’image par défaut de la catégorie

**consumed\_recipes**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idConsumedRecipe | Int | Non |  |
| consumption\_date | Date | Non |  |
| idUser | Int | Non |  |
| idRecipe | Int | Non |  |
| idCategory | Int | Non |  |

Dans cette table :

* Le champ "idConsumedRecipe" est la clé primaire qui identifie chaque enregistrement de recette consommée.
* Le champ "consumption\_date" permet de stocker la date à laquelle la recette a été consommée.
* Le champ "idUser" est une clé étrangère faisant référence à l'utilisateur qui a consommé la recette.
* Le champ "idRecipe" est une clé étrangère faisant référence à la recette consommée.
* Le champ "idCategory" est une clé étrangère faisant référence à la catégorie à laquelle la recette appartient.

**food\_types**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idFoodType | Int | Non |  |
| name | Varchar (255) | Non |  |

Dans cette table :

* Le champ "idFoodType" est la clé primaire qui identifie chaque type de nourriture.
* Le champ "name" permet de stocker le nom du type de nourriture.

**ratings**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idRating | Int | Non |  |
| score | Int | Non |  |
| comment | Text | Oui | NULL |
| idUser | Int | Non |  |
| idRecipe | Int | Oui |  |

Dans cette table :

* Le champ "idRating" est la clé primaire qui identifie chaque enregistrement de notation.
* Le champ "score" permet de stocker la note attribuée à une recette.
* Le champ "comment" permet de stocker un commentaire facultatif associé à la notation.
* Le champ "idUser" est une clé étrangère faisant référence à l'utilisateur qui a donné la note.
* Le champ "idRecipe" est une clé étrangère faisant référence à la recette notée.

**recipes**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idRecipe | Int | Non |  |
| title | Int | Non |  |
| preparation\_time | Text | Oui | NULL |
| difficulty | Int | Non |  |
| instructions | Int | Oui |  |
| calories | Int | Non |  |
| created\_at | timestamp | Oui | CURRENT\_TIMESTAMP |
| idImage | Int |  |  |
| idUser | Int | Non |  |

Dans cette table :

* Le champ "idRecipe" est la clé primaire qui identifie chaque recette.
* Le champ "title" permet de stocker le titre de la recette.
* Le champ "preparation\_time" permet de stocker le temps de préparation de la recette.
* Le champ "difficulty" permet de stocker le niveau de difficulté de la recette.
* Le champ "instructions" permet de stocker les instructions de préparation de la recette.
* Le champ "calories" permet de stocker le nombre de calories de la recette.
* Le champ "created\_at" permet de stocker la date de création de la recette.
* Le champ "idImage" est une clé étrangère faisant référence à l'image associée à la recette.
* Le champ "idUser" est une clé étrangère faisant référence à l'utilisateur qui a créé la recette.

**recipe\_categories**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idRecipe | Int | Non |  |
| idCategory | Int | Non |  |

Dans cette table :

* Le champ "idRecipe" est une clé étrangère faisant référence à une recette.
* Le champ "idCategory" est une clé étrangère faisant référence à une catégorie à laquelle la recette appartient.

**recipe\_food\_types**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idRecipe | Int | Non |  |
| idFoodType | Int | Non |  |
| percentage | Decimal | Non |  |

Dans cette table :

* Le champ "idRecipe" est une clé étrangère faisant référence à une recette.
* Le champ "idFoodType" est une clé étrangère faisant référence à un type de nourriture associé à la recette.
* Le champ "percentage" permet de stocker le pourcentage de ce type de nourriture dans la recette.

**users**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idUser | Int | Non |  |
| username | Varchar (255) | Non |  |
| email | Varchar (255) | Non |  |
| password | Varchar (255) | Non |  |
| token | Varchar(400) | Non |  |
| expiration | Int | Non |  |

Dans cette table :

* Le champ "idUser" est la clé primaire qui identifie chaque utilisateur.
* Le champ "username" permet de stocker le nom d'utilisateur.
* Le champ "email" permet de stocker adresse e-mail de l'utilisateur.
* Le champ "password" permet de stocker le mot de passe de l'utilisateur de manière sécurisée.
* Le champ "token" permet de stocker le jeton d'authentification de l'utilisateur.
* Le champ "expiration" permet de stocker la durée de validité du jeton d'authentification.

**recipes\_images**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idImage | Int | Non |  |
| path | Varchar (255) | Non |  |
| idRecipe | Int | Non |  |

Dans cette table :

* Le champ "idImage" est la clé primaire qui identifie chaque image associée à une recette.
* Le champ "path" permet de stocker le chemin d'accès à l'image de la recette.
* Le champ "idRecipe" est une clé étrangère faisant référence à la recette notée.

**recipes\_details\_view**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| recipe\_id | Int | Non |  |
| recipe\_title | Varchar (255) | Non |  |
| recipe\_instructions | Int | Non |  |
| difficulty | Enum | Non |  |
| calories | Int | Non |  |
| created\_at | Timestamp | Non |  |
| preparation\_time | Int | Non |  |
| creator\_user | Varchar | Non |  |
| creator\_id | Int | Non |  |
| image\_paths | Text | Non |  |
| categories | Text | Non |  |
| foodtype\_with\_percentages | Text | Non |  |
| average\_rating | Decimal | Non |  |

Dans cette view:

* Le champ "recipe\_id" est la clé unique qui identifie chaque recette.
* Le champ "recipe\_title" stocke le titre de chaque recette.
* Le champ "recipe\_instructions" stocke les instructions pour la recette.
* Le champ "difficulty" représente le niveau de difficulté de la recette.
* Le champ "calories" donne le nombre de calories de la recette.
* Le champ "created\_at" enregistre la date et l'heure de création de la recette.
* Le champ "preparation\_time" donne le temps de préparation de la recette.
* Le champ "creator\_user" stocke le nom d'utilisateur du créateur de la recette.
* Le champ "creator\_id" est une clé étrangère qui fait référence à l'utilisateur qui a créé la recette.
* Le champ "image\_paths" stocke les chemins d'accès aux images associées à la recette.
* Le champ "categories" pourrait stocke les catégories associées à la recette.
* Le champ "foodtype\_with\_percentages" stocke les types d'aliments utilisés dans la recette avec leurs pourcentages respectifs.
* Le champ "average\_rating" donne la note moyenne de la recette.

**recipes\_foodtypes\_view**

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | Type | Null | Défaut |
| idRecipe | Int | Non |  |
| foodtypes\_with\_percentage | Varchar (255) | Non |  |

Dans cette table :

* Le champ "idRecipe" est la clé primaire qui identifie chaque recette.
* Le champ "foodtypes\_with\_percentage" stocke les types d'aliments utilisés dans la recette avec leurs pourcentages respectifs.

# **Généralités concernant l'implémentation**

## **Langages (versions)**

* PHP (version 8.2)
* MySQL (8.0.31)
* Apache (2.4.54.2)
* HTML5
* CSS3

## **Environnement (local et/ou distant)**

J’utilise l’environnement de développement local Wampserver - 3.3.0 – 64bit

# **Plan de test**

## **Périmètre**

Le plan de test pour le projet EatFit couvrira toutes les fonctionnalités clés de l'application, y compris la gestion des profils d'utilisateurs, la gestion des recettes, les évaluations de recettes, la recherche de recettes et le filtrage des recettes. Les tests fonctionnels seront réalisés pour vérifier le bon fonctionnement et la convivialité de l'interface utilisateur. Les tests de performance seront également effectués pour assurer la réponse rapide de l'application aux actions de l'utilisateur. En outre, la sécurité de l'application sera testée pour garantir la protection adéquate des données personnelles des utilisateurs.

## **Environnement**

Les tests seront effectués dans divers environnements pour assurer la compatibilité et la performance de l'application. Cela comprend les environnements de développement, de test, de staging et de production. Les tests seront également réalisés sur différents navigateurs (Chrome, Firefox, Safari, Edge) pour assurer une expérience utilisateur optimale sur tous les navigateurs.

## **Scénarios (les tests, les données et leurs évolution)**

|  |  |
| --- | --- |
| Nom | 1.1 Création d'un Virtual Host |
| User story | S5 : Création d'un Virtual Host |
| Situation | Le développeur a créé un Virtual Host pour permettre l'utilisation d'un "router" pour naviguer entre les différentes pages du site. |
| Résultats obtenus | Virtual host crée, peut être utilisé pour le site |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 2.1 Structure de l'API |
| User story | S17 : Faire la structure de l'API |
| Situation | Le développeur crée la structure de base de l'API pour gérer les requêtes entre l'application et la base de données. Par exemple, il pourrait mettre en place des endpoints pour les opérations CRUD (Création, Lecture, Mise à jour, Suppression) sur les utilisateurs, les recettes et les évaluations. |
| Résultats obtenus | La structure de base de l'API est mise en place et prête à être utilisée par l'application. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 2.2 Adapter le modèle |
| User story | S18 : Adapter le modèle de l'API à l'application |
| Situation | Le développeur adapte le modèle de l'API pour qu'il corresponde aux besoins de l'application et aux relations entre les différentes entités (utilisateurs, recettes, évaluations, etc.). |
| Résultats obtenus | Le modèle de l'API est adapté pour correspondre aux besoins de l'application. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.1 Création d’un profil utilisateur |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur non identifié crée un nouveau compte en entrant «Username, [email@gmail.com](mailto:email@gmail.com) ,test1234.» le nouveau compte est enregistré dans la base de données |
| Résultats obtenus | Retourne les informations de l’utilisateur avec un message de succès |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.1.2 Création d’un profil utilisateur - Echec |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur non identifié tente de créer un nouveau compte avec une adresse e-mail ou un pseudo qui est déjà utilisé(e). |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'adresse e-mail est déjà utilisée. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.2 Modification d’un profil utilisateur |
| User story | S4 : Gestion du profil utilisateur |
| Situation | Un utilisateur existant modifie les détails de son profil, comme son pseudo, son adresse e-mail ou son mot de passe. |
| Résultats obtenus | Les modifications sont enregistrées dans la base de données. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.2.2 Modification d’un profil utilisateur - Echec |
| User story | S4 : Gestion du profil utilisateur |
| Situation | Un utilisateur existant tente de modifier son adresse e-mail en une qui est déjà utilisée. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'adresse e-mail est déjà utilisée. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.3 Suppression d’un profil utilisateur |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur identifié peut supprimer son compte. |
| Résultats obtenus | Retourne un message de succès |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.3.2 Suppression d’un profil utilisateur - Echec |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur non identifié tente de supprimer un compte. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour supprimer un compte. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 3.4 Lecture d’un profil utilisateur |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur identifié peut accéder à ses informations. |
| Résultats obtenus | Retourne les informations de l’utilisateur |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.1 Création d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut créer une recette. Par exemple, un utilisateur peut créer une recette "Poulet à la moutarde" en renseignant le titre, le temps nécessaire, les instructions, la difficulté, les calories, la date de création et le créateur. |
| Résultats obtenus | La recette est créée avec succès et est visible dans la liste des recettes. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.1.2 Création d’une recette - Echec |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur non identifié tente de créer une recette. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour créer une recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.2 Modification d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut modifier sa recette. Par exemple, l'utilisateur pourrait changer le temps de préparation de sa recette "Poulet à la moutarde" de 45 minutes à 60 minutes. |
| Résultats obtenus | La recette est mise à jour avec succès et les modifications sont visibles dans la description de la recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.2.2 Modification d’une recette - Echec |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur non identifié tente de modifier une recette. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour modifier une recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.3 Suppression d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut supprimer sa recette. Par exemple, l'utilisateur pourrait décider de supprimer sa recette "Poulet à la moutarde" de sa liste de recettes. |
| Résultats obtenus | La recette est supprimée avec succès et n'est plus visible dans la liste des recettes. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.3.2 Suppression d’une recette - Echec |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur non identifié tente de supprimer une recette. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour supprimer une recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.4 Lecture d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut accéder à sa recette. Par exemple, l'utilisateur pourrait consulter les détails de sa recette "Poulet à la moutarde". |
| Résultats obtenus | L'utilisateur peut consulter les détails de sa recette avec succès. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.5 Recherche d’une recette |
| User story | S10 : Recherche de recettes |
| Situation | L'utilisateur peut rechercher des recettes en fonction de leur nom, catégorie ou date d'ajout. Par exemple, l'utilisateur pourrait rechercher toutes les recettes qui contiennent le mot "Poulet" dans leur nom. |
| Résultats obtenus | Les recettes correspondant aux critères de recherche de l'utilisateur sont affichées avec succès. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 4.6 Filtrage des recettes |
| User story | S11 : Filtrage des recettes |
| Situation | L’utilisateur identifié peut filtrer les recettes par catégorie ou type d'aliments. Par exemple, l'utilisateur pourrait choisir de n'afficher que les recettes avec protéines. |
| Résultats obtenus | Les recettes correspondant aux critères de filtrage de l'utilisateur sont affichées avec succès. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.1 Création d’une évaluation de recette |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L’utilisateur identifié peut évaluer de 1 à 5 et commenter(facultatif). Par exemple, après avoir cuisiné et goûté le "Poulet à la moutarde", l'utilisateur pourrait donner une note de 4 et laisser un commentaire comme "Très bon, mais un peu trop épicé pour moi". |
| Résultats obtenus | L'évaluation est enregistrée dans la base de données. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.1.2 Création d’une évaluation de recette - Echec |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L'utilisateur non identifié tente de créer une évaluation de recette. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour créer une évaluation. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.2 Modification d’une évaluation de recette |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L’utilisateur peut modifier la note ou le commentaire s’il n’y en avait pas avant il peut en ajouter. Par exemple, après une seconde préparation de la recette "Poulet à la moutarde", l'utilisateur décide de modifier sa note de 4 à 5 et d'ajouter un commentaire "Parfait avec moins de piment". |
| Résultats obtenus | L'évaluation de la recette est modifiée avec succès et les changements sont visibles sur la page de la recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.2.2 Modification d’une évaluation de recette - Echec |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L'utilisateur non identifié tente de modifier une évaluation de recette.  Un utilisateur identifié tente de modifier le titre de la recette et une recette avec ce nom existe déjà. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour modifier une évaluation.  Un message d’erreur est affiché indiquant que le titre est déjà utilisé. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.3 Suppression d’une évaluation de recette |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L’utilisateur identifié peut supprimer l’évaluation. Par exemple, l'utilisateur décide de supprimer son évaluation sur la recette "Poulet à la moutarde" pour des raisons personnelles. |
| Résultats obtenus | L'évaluation est supprimée avec succès et n'est plus visible sur la page de la recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.3.2 Suppression d’une évaluation de recette - Echec |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L'utilisateur non identifié tente de supprimer une évaluation de recette. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour supprimer une évaluation. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 5.4 Lecture d’une évaluation de recette |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L’utilisateur identifié peut voir les commentaires d’une recette. Par exemple, l'utilisateur peut lire les commentaires laissés par d'autres utilisateurs sur la recette "Poulet à la moutarde". |
| Résultats obtenus | L'utilisateur peut lire avec succès les évaluations et commentaires d'autres utilisateurs sur la recette. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 6.1 Historique des recettes consommées |
| User story | S12 : Historique des recettes consommées |
| Situation | L’utilisateur identifié peut accéder à son historique et voir les dernières recettes consommées. Par exemple, l'utilisateur peut voir consommé la recette "Poulet à la moutarde" il y a deux jours. |
| Résultats obtenus | L'utilisateur peut accéder avec succès à son historique et voir les recettes qu'il a consommées récemment. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 6.1.2 Historique des recettes consommées - Echec |
| User story | S12 : Historique des recettes consommées |
| Situation | Un utilisateur non identifié tente d'accéder à l'historique des recettes consommées. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour accéder à l'historique des recettes consommées. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 7.1 Création du style du site |
| User story | S3 : Création du style du site |
| Situation | Le développeur a créé le style visuel du site, en se concentrant sur l'aspect esthétique et la convivialité de l'interface utilisateur. |
| Résultats obtenus | Le site a un aspect esthétiquement agréable et une interface utilisateur intuitive. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 8.1 Connexion à un compte utilisateur |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur tente de se connecter avec un nom d'utilisateur et un mot de passe corrects. |
| Résultats obtenus | L'utilisateur est connecté avec succès et est redirigé vers la page d'accueil. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 8.2 Échec de la connexion à un compte |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur tente de se connecter avec un nom d'utilisateur et un mot de passe incorrects. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que les informations d'identification sont incorrectes. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 9.1 Ajout d’un nouveau type de nourriture |
| User story | S14 : Ajout de nouveaux types d'aliments |
| Situation | L’utilisateur rentre un type de nourriture dans le champ. |
| Résultats obtenus | Le type est enregistré et un message de succès est affiché. |
| Statut | OK |

|  |  |
| --- | --- |
| Nom | 9.1.2 Ajout d’un nouveau type de nourriture - Echec |
| User story | S14 : Ajout de nouveaux types d'aliments |
| Situation | Un utilisateur non identifié tente d'ajouter un nouveau type de nourriture.  Un utilisateur identifié tente d’ajouter un type existant. |
| Résultats obtenus | Un message d'erreur est affiché indiquant que l'utilisateur doit être identifié pour ajouter un nouveau type de nourriture.  Un message d’erreur est affiché indiquant que le type existe déjà. |
| Statut | OK |

## **Evolution des tests**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N° test** | **J1** | **J2** | **J3** | **J4** | **J5** | **J6** | **J7** | **J8** | **J9** | **J10** | **J11** |
| 1.1 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 2.1 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 2.2 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.1 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.1.2 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.2 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.2.2 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.3 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.3.2 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.4 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.1 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.1.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.2.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.3 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.3.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.4 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4.5 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ |
| 4.6 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.1 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.1.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.2.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.3 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.3.2 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5.4 | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 6.1 | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 6.1.2 | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 7.1 | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8.1 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ |
| 8.2 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ |
| 9.1 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 9.1.2 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ |

# **Conclusion**

## **Difficultés rencontrées**

Bien que je n'aie pas rencontré de problèmes majeurs durant mon projet, une difficulté que j'ai rencontrée concernait la planification des tâches. En rétrospective, chaque tâche aurait dû bénéficier de plus de temps alloué. Par exemple, lorsque je progressais rapidement sur la gestion des utilisateurs, je n'avais pas initialement envisagé le temps nécessaire pour développer le front-end correspondant à cette gestion. Ainsi, plutôt que de prévoir 4 heures pour une tâche, j'aurais peut-être dû prévoir 6 heures (par exemple). Cette expérience a souligné l'importance d'une planification précise et réaliste dans le cadre de la gestion de projet.

## **Variantes de solutions et choix**

|  |  |  |  |
| --- | --- | --- | --- |
| Choix et variantes | | | |
| Sujet | + | - | Conclusion |
| API ou Site classique | | | |
| API | Facilité de réutilisation pour d'autres plateformes (ex : app mobile) | Moins de familiarité avec la création d'API | Choix pour le projet actuel : API pour EatFit |
| Meilleure séparation entre la logique back-end et front-end | Temps de développement potentiellement plus long |
| Facilite l'intégration avec d'autres services et applications | Nécessite une documentation plus complète |
| Site classique | Plus familier avec la création de sites classiques | Limité aux navigateurs Web |
| Moins de documentation nécessaire | Difficulté d'intégration avec d'autres services |
| Temps de développement potentiellement plus court | Moins modulaire et moins évolutif |

## **Améliorations possibles**

## **Bilan personnel**

## **Remerciements**

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet. Je remercie tout particulièrement Mme Mota, mes formateurs et mes collègues pour leurs précieux conseils et leur soutien tout au long de ce projet.

# **Bibliographie**

Les liens que j’ai le plus consulté pendant mon TPI sont :

* <https://www.php.net/>
* <https://unsplash.com/> (pour les images)
* <https://stackoverflow.com>
* <https://www.w3schools.com>
* <https://codepen.io/bensassi/pen/dWBdwe>
* Avant de consulter, [ChatGPT](https://chat.openai.com/) j'ai utilisé ce site pour rechercher des exemples de graphiques et comprendre comment les implémenter dans mon projet. <https://www.chartjs.org/docs/latest/>
* <https://sql.sh/>

# **Annexes**

## **Conversations ChatGPT**

### **Graphique en js**

Peux tu me faire un graphique camembert en js

ChatGPT

Malheureusement, en tant que modèle de texte AI, je ne peux pas créer des graphiques directement. Cependant, je peux vous montrer comment vous pouvez créer un graphique à secteurs (ou graphique camembert) en utilisant JavaScript et une bibliothèque populaire appelée Chart.js.

Voici un exemple de code que vous pouvez utiliser :

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Graphique Camembert en JavaScript</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<canvas id="myChart"></canvas>

<script>

var ctx = document.getElementById('myChart').getContext('2d');

var myChart = new Chart(ctx, {

type: 'pie',

data: {

labels: ['Rouge', 'Bleu', 'Jaune', 'Vert', 'Violet', 'Orange'],

datasets: [{

label: '# de votes',

data: [12, 19, 3, 5, 2, 3],

backgroundColor: [

'rgba(255, 99, 132, 0.2)',

'rgba(54, 162, 235, 0.2)',

'rgba(255, 206, 86, 0.2)',

'rgba(75, 192, 192, 0.2)',

'rgba(153, 102, 255, 0.2)',

'rgba(255, 159, 64, 0.2)'

],

borderColor: [

'rgba(255, 99, 132, 1)',

'rgba(54, 162, 235, 1)',

'rgba(255, 206, 86, 1)',

'rgba(75, 192, 192, 1)',

'rgba(153, 102, 255, 1)',

'rgba(255, 159, 64, 1)'

],

borderWidth: 1

}]

},

options: {

responsive: true,

}

});

</script>

</body>

</html>

Dans cet exemple, nous créons un graphique à secteurs avec six sections différentes, chacune ayant une couleur différente. Les valeurs de chaque section sont spécifiées dans le tableau data et les étiquettes correspondantes sont spécifiées dans le tableau labels.

N'oubliez pas d'inclure Chart.js dans votre fichier HTML en utilisant la balise <script> dans l'en-tête de votre document. Vous pouvez télécharger Chart.js et l'inclure localement, ou vous pouvez utiliser un CDN comme dans l'exemple ci-dessus.

User

peux tu le faire plus petit

ChatGPT

Bien sûr, pour rendre le graphique plus petit, vous pouvez simplement définir la largeur et la hauteur du canvas. Par exemple, vous pouvez définir la largeur et la hauteur à 200 pixels comme ceci :

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Graphique Camembert en JavaScript</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<canvas id="myChart" width="200" height="200"></canvas>

<script>

var ctx = document.getElementById('myChart').getContext('2d');

var myChart = new Chart(ctx, {

type: 'pie',

data: {

labels: ['Rouge', 'Bleu', 'Jaune', 'Vert', 'Violet', 'Orange'],

datasets: [{

label: '# de votes',

data: [12, 19, 3, 5, 2, 3],

backgroundColor: [

'rgba(255, 99, 132, 0.2)',

'rgba(54, 162, 235, 0.2)',

'rgba(255, 206, 86, 0.2)',

'rgba(75, 192, 192, 0.2)',

'rgba(153, 102, 255, 0.2)',

'rgba(255, 159, 64, 0.2)'

],

borderColor: [

'rgba(255, 99, 132, 1)',

'rgba(54, 162, 235, 1)',

'rgba(255, 206, 86, 1)',

'rgba(75, 192, 192, 1)',

'rgba(153, 102, 255, 1)',

'rgba(255, 159, 64, 1)'

],

borderWidth: 1

}]

},

options: {

responsive: false, // pour désactiver le redimensionnement automatique

}

});

</script>

</body>

</html>

Notez l'ajout des attributs width="200" et height="200" à la balise <canvas>. J'ai également modifié l'option responsive à false pour désactiver le redimensionnement automatique en fonction de la taille de la fenêtre.

User

J'aimerai récupérer les données d'une api

ChatGPT

Bien sûr, vous pouvez récupérer les données d'une API en utilisant la fonction fetch() en JavaScript. Voici comment vous pouvez le faire :

javascript

Copy code

fetch('URL\_DE\_VOTRE\_API')

.then(response => response.json())

.then(data => {

// Utilisez vos données ici

console.log(data);

})

.catch(error => console.error('Erreur:', error));

Vous pouvez remplacer 'URL\_DE\_VOTRE\_API' par l'URL de l'API que vous souhaitez utiliser.

Cela dit, pour mettre à jour votre graphique avec les données récupérées, vous pouvez remplacer la partie de la récupération des données dans le script du graphique camembert comme suit :

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Graphique Camembert en JavaScript</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<canvas id="myChart" width="200" height="200"></canvas>

<script>

fetch('URL\_DE\_VOTRE\_API')

.then(response => response.json())

.then(data => {

// Ici, je suppose que 'data' est un objet avec deux propriétés : 'labels' et 'datasets'

var ctx = document.getElementById('myChart').getContext('2d');

var myChart = new Chart(ctx, {

type: 'pie',

data: data,

options: {

responsive: false,

}

});

})

.catch(error => console.error('Erreur:', error));

</script>

</body>

</html>

Dans cet exemple, je suppose que l'API renvoie un objet JSON avec les propriétés labels et datasets conformément à la structure attendue par Chart.js. Vous devrez adapter cette partie du code en fonction de la structure réelle des données renvoyées par votre API.

User

je peux récupérer des données du php ?

ChatGPT

Oui, vous pouvez certainement récupérer des données d'un script PHP. Le PHP peut générer des données dynamiques qui peuvent être consommées par votre application JavaScript.

Pour ce faire, vous pouvez effectuer une requête HTTP à votre script PHP en utilisant la fonction fetch() de JavaScript, puis utiliser la réponse pour alimenter votre graphique.

Voici comment vous pouvez le faire :

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Graphique Camembert en JavaScript</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<canvas id="myChart" width="200" height="200"></canvas>

<script>

fetch('votre\_script.php')

.then(response => response.json())

.then(data => {

// Ici, je suppose que 'data' est un objet avec deux propriétés : 'labels' et 'datasets'

var ctx = document.getElementById('myChart').getContext('2d');

var myChart = new Chart(ctx, {

type: 'pie',

data: data,

options: {

responsive: false,

}

});

})

.catch(error => console.error('Erreur:', error));

</script>

</body>

</html>

Dans votre script PHP, assurez-vous que vous générez une réponse JSON valide. Voici un exemple simple de ce que pourrait être votre script PHP :

php

Copy code

<?php

header('Content-Type: application/json');

$data = [

'labels' => ['Rouge', 'Bleu', 'Jaune', 'Vert', 'Violet', 'Orange'],

'datasets' => [

[

'label' => '# de votes',

'data' => [12, 19, 3, 5, 2, 3],

'backgroundColor' => [

'rgba(255, 99, 132, 0.2)',

'rgba(54, 162, 235, 0.2)',

'rgba(255, 206, 86, 0.2)',

'rgba(75, 192, 192, 0.2)',

'rgba(153, 102, 255, 0.2)',

'rgba(255, 159, 64, 0.2)'

],

'borderColor' => [

'rgba(255, 99, 132, 1)',

'rgba(54, 162, 235, 1)',

'rgba(255, 206, 86, 1)',

'rgba(75, 192, 192, 1)',

'rgba(153, 102, 255, 1)',

'rgba(255, 159, 64, 1)'

],

'borderWidth' => 1

]

]

];

echo json\_encode($data);

?>

Veuillez remplacer 'votre\_script.php' par l'URL de votre script PHP. Aussi, n'oubliez pas de configurer correctement votre serveur pour permettre les requêtes CORS si votre script PHP et votre page HTML ne sont pas sur le même domaine.

User

JE peux pas faire plus simple j'ai une variable qui contient le pourcentage et une qui contient le nom

ChatGPT

D'accord, si vous avez déjà les variables en PHP, vous pouvez les incorporer directement dans votre script JavaScript de la manière suivante:

Supposons que vous ayez une variable $nom pour le nom et une variable $pourcentage pour le pourcentage en PHP. Vous pouvez les utiliser dans votre script JavaScript comme ceci :

php

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Graphique Camembert en JavaScript</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<canvas id="myChart" width="200" height="200"></canvas>

<script>

<?php

$nom = 'Exemple'; // Remplacez par votre vraie variable

$pourcentage = 75; // Remplacez par votre vraie variable

?>

var ctx = document.getElementById('myChart').getContext('2d');

var myChart = new Chart(ctx, {

type: 'pie',

data: {

labels: ['<?php echo $nom; ?>', 'Autres'],

datasets: [{

data: [<?php echo $pourcentage; ?>, <?php echo (100 - $pourcentage); ?>],

backgroundColor: [

'rgba(255, 99, 132, 0.2)',

'rgba(54, 162, 235, 0.2)'

],

borderColor: [

'rgba(255, 99, 132, 1)',

'rgba(54, 162, 235, 1)'

],

borderWidth: 1

}]

},

options: {

responsive: false,

}

});

</script>

</body>

</html>

Ici, nous utilisons les valeurs PHP directement dans le script JavaScript en les intégrant dans le code source avec <?php echo $variable; ?>.

Notez que ce code doit être exécuté sur un serveur PHP pour fonctionner correctement, car le PHP est un langage de script côté serveur. Si vous ouvrez ce fichier directement dans un navigateur sans passer par un serveur PHP, le code PHP ne sera pas exécuté.

User

Fais aussi une fonction qui génere des couleurs aléatoires pour automatiser le graphique

ChatGPT

Bien sûr, voici comment vous pouvez générer des couleurs aléatoires en JavaScript :

javascript

Copy code

function generateRandomColor() {

var letters = '0123456789ABCDEF';

var color = '#';

for (var i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

return color;

}

Cette fonction génère une couleur hexadécimale aléatoire.

Et voici comment vous pouvez l'incorporer dans votre script de graphique :

php

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Graphique Camembert en JavaScript</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<canvas id="myChart" width="200" height="200"></canvas>

<script>

function generateRandomColor() {

var letters = '0123456789ABCDEF';

var color = '#';

for (var i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

return color;

}

<?php

$nom = 'Exemple'; // Remplacez par votre vraie variable

$pourcentage = 75; // Remplacez par votre vraie variable

?>

var ctx = document.getElementById('myChart').getContext('2d');

var myChart = new Chart(ctx, {

type: 'pie',

data: {

labels: ['<?php echo $nom; ?>', 'Autres'],

datasets: [{

data: [<?php echo $pourcentage; ?>, <?php echo (100 - $pourcentage); ?>],

backgroundColor: [

generateRandomColor(),

generateRandomColor()

],

borderColor: [

generateRandomColor(),

generateRandomColor()

],

borderWidth: 1

}]

},

options: {

responsive: false,

}

});

</script>

</body>

</html>

Dans ce code, generateRandomColor() est appelé à chaque fois que vous avez besoin d'une nouvelle couleur pour backgroundColor et borderColor.

## **Glossaire**

|  |  |
| --- | --- |
| Terme métier | |
| Recette  (Recipe) | Instructions étape par étape pour préparer un plat spécifique. |
| Catégorie de recette  (Category) | Classification des recettes en fonction de leur type, de leur moment de consommation ou de leurs ingrédients principaux (par exemple, petit déjeuner, déjeuner, dîner, dessert). |
| Difficulté (Difficulty) | Niveau de complexité ou d'effort requis pour préparer une recette, généralement classé en facile, moyen ou difficile. |
| Évaluation  (Rating) | Appréciation ou jugement porté sur une recette par les utilisateurs, sous forme de note sur une échelle de 1 à 5. |
| Commentaire (Comment) | Retour d'information textuel d'un utilisateur sur une recette, généralement lié à l'évaluation donnée. |
| Historique des recettes consommées (History) | Liste des recettes réalisées et consommées par un utilisateur, avec des informations sur les dates et les moments de consommation. |

|  |  |
| --- | --- |
| Terme technique | |
| Front-end | Partie d'un système informatique ou d'une application avec laquelle l'utilisateur interagit directement. |
| Back-end | Partie d'un système informatique ou d'une application qui gère et traite les opérations en arrière-plan. Le back-end est responsable du fonctionnement et du traitement des données, qui sont ensuite envoyées au front-end pour être affichées à l'utilisateur. |
| Utilisateur (User) | Personne qui utilise l'application ou le système informatique. Dans le contexte de ce projet, cela peut se référer à toute personne qui utilise l'application de recettes, que ce soit pour parcourir les recettes, les noter, les commenter, ou les ajouter à leur historique de recettes consommées. |
| API | Une API est un ensemble de règles et de protocoles pour la construction et l'interaction avec des logiciels. Dans le contexte de ce projet, l'API permet à différentes parties de l'application (par exemple, le front-end et le back-end) de communiquer entre elles de manière efficace et sécurisée. Une API peut également permettre à l'application de recettes d'interagir avec d'autres logiciels ou services externes. |
| JSON | JSON est un format de données léger utilisé pour le stockage et le transport de données. |
| MLD | Le MLD est un modèle de données qui représente la structure d'une base de données de manière logique et indépendante du système de gestion de base de données utilisé. Il décrit la structure de la base de données en termes de tables, de champs et de relations entre les tables. Dans le contexte de ce projet, le MLD a été utilisé pour concevoir la structure de la base de données pour l'application de recettes. |