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](#_Toc134022781)

[**Rappel de l'énoncé** 4](#_Toc134022782)

[**Description de l’application** 4](#_Toc134022783)

[**Organisation** 5](#_Toc134022784)

[**Livrables** 5](#_Toc134022785)

[**Matériel et logiciels à disposition** 5](#_Toc134022786)

[**Méthodologie** 5](#_Toc134022787)

[**Planification en 6 étapes** 5](#_Toc134022788)

[**Planification** 7](#_Toc134022789)

[**Planification prévue** 7](#_Toc134022790)

[**Planification effective** 7](#_Toc134022791)

[**Product backlog** 7](#_Toc134022792)

[**Analyse organique** 11](#_Toc134022793)

[**Base de données** 15](#_Toc134022794)

[**Informations techniques** 15](#_Toc134022795)

[**MCD** 16](#_Toc134022796)

[**MLD** 16](#_Toc134022797)

[**Dictionnaire de données** 16](#_Toc134022798)

[**Architecture** 18](#_Toc134022799)

[**Structure (Découpage du projet et organisation des dossiers : MVC ou autres)** 18](#_Toc134022800)

[**Classes** 13](#_Toc134022801)

[**Généralités concernant l'implémentation** 18](#_Toc134022802)

[**Langages (versions)** 18](#_Toc134022803)

[**Environnement (local et/ou distant)** 18](#_Toc134022804)

[**Framework et outils externes** 18](#_Toc134022805)

[**Composants externes** 18](#_Toc134022806)

[**Bootstrap** 18](#_Toc134022807)

[**Font Awesome** 18](#_Toc134022808)

[**Analyse des fonctionnalités majeures (Critères et fonctionnalités de CdC)** 18](#_Toc134022809)

[**Plan de test** 18](#_Toc134022810)

[**Périmètre** 18](#_Toc134022811)

[**Environnement** 19](#_Toc134022812)

[**Scénarios (les tests, les données et leurs évolution)** 19](#_Toc134022813)

[**Conclusion** 23](#_Toc134022814)

[**Difficultés rencontrées** 23](#_Toc134022815)

[**Variantes de solutions et choix** 23](#_Toc134022816)

[**Améliorations possibles** 23](#_Toc134022817)

[**Bilan personnel** 23](#_Toc134022818)

[**Remerciements** 23](#_Toc134022819)

[**Bibliographie** 23](#_Toc134022820)

[**Annexes** 23](#_Toc134022821)

[**Glossaire** 23](#_Toc134022822)

[**Résumé tpi** 24](#_Toc134022823)

[**Résumé TPI** 24](#_Toc134022824)

# **Introduction**

Ce document est un rapport présentant différents aspects de la conception du projet Interlude. 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**

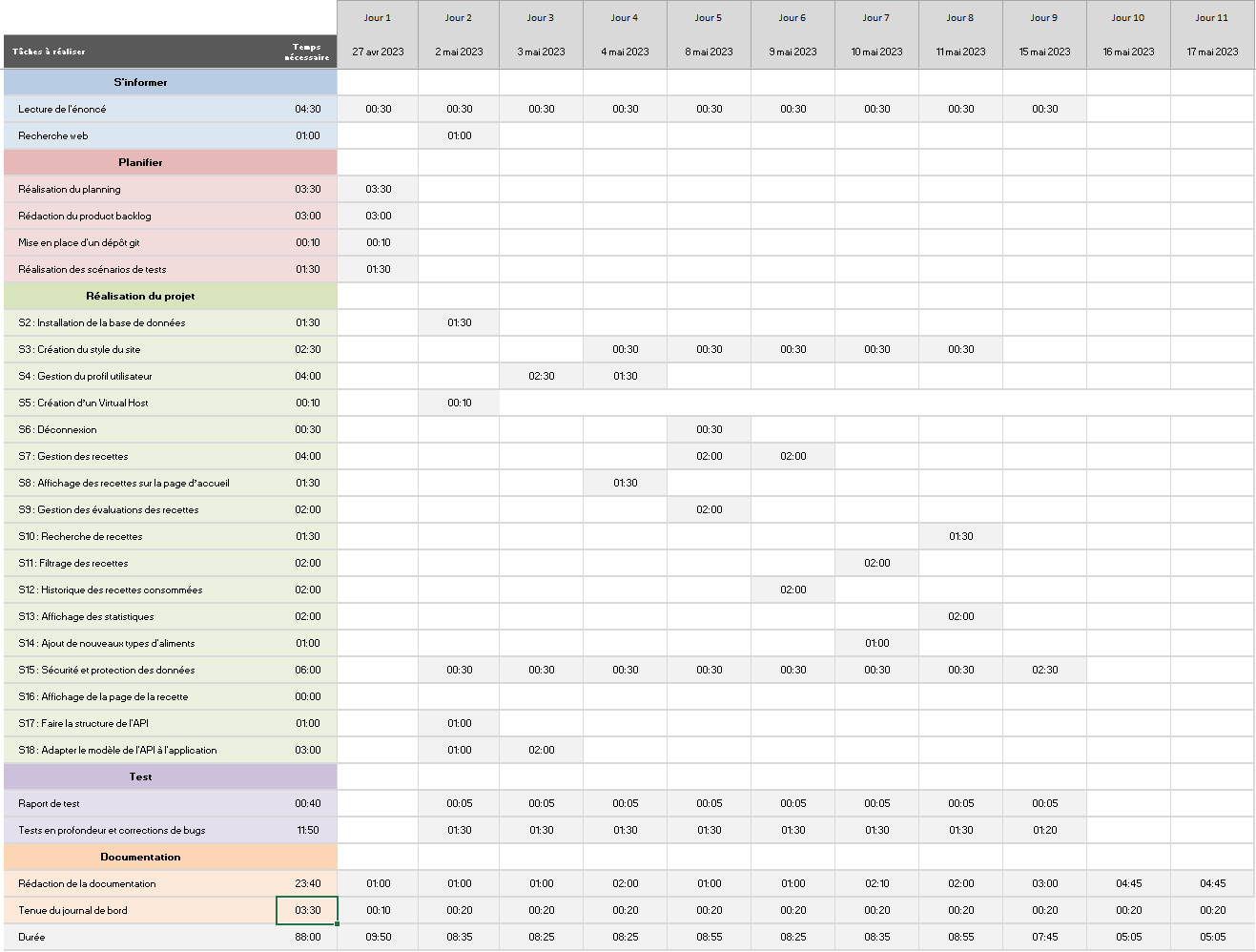
À chaque étape de réalisation, j'ai effectué des tests rigoureux pour m'assurer que chaque fonctionnalité fonctionnait correctement et répondait aux exigences du cahier des charges. J'ai documenté ces tests dans mon rapport de projet, en décrivant les scénarios de test et les résultats obtenus. J'ai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

## Architecture de l’API

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.

### Back-end

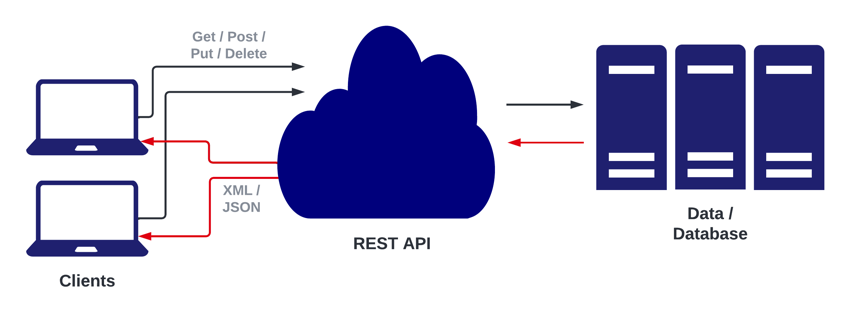
Le back-end est le cœur de l'application EatFit, où se trouve la logique métier et où les données sont stockées et récupérées. Il est composé de plusieurs composants, y compris les contrôleurs, les modèles et la base de données.

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.

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



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**

Cette classe n'a pas été détaillée dans votre liste, mais on peut supposer qu'elle gère les opérations CRUD pour les évaluations des recettes.

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

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.

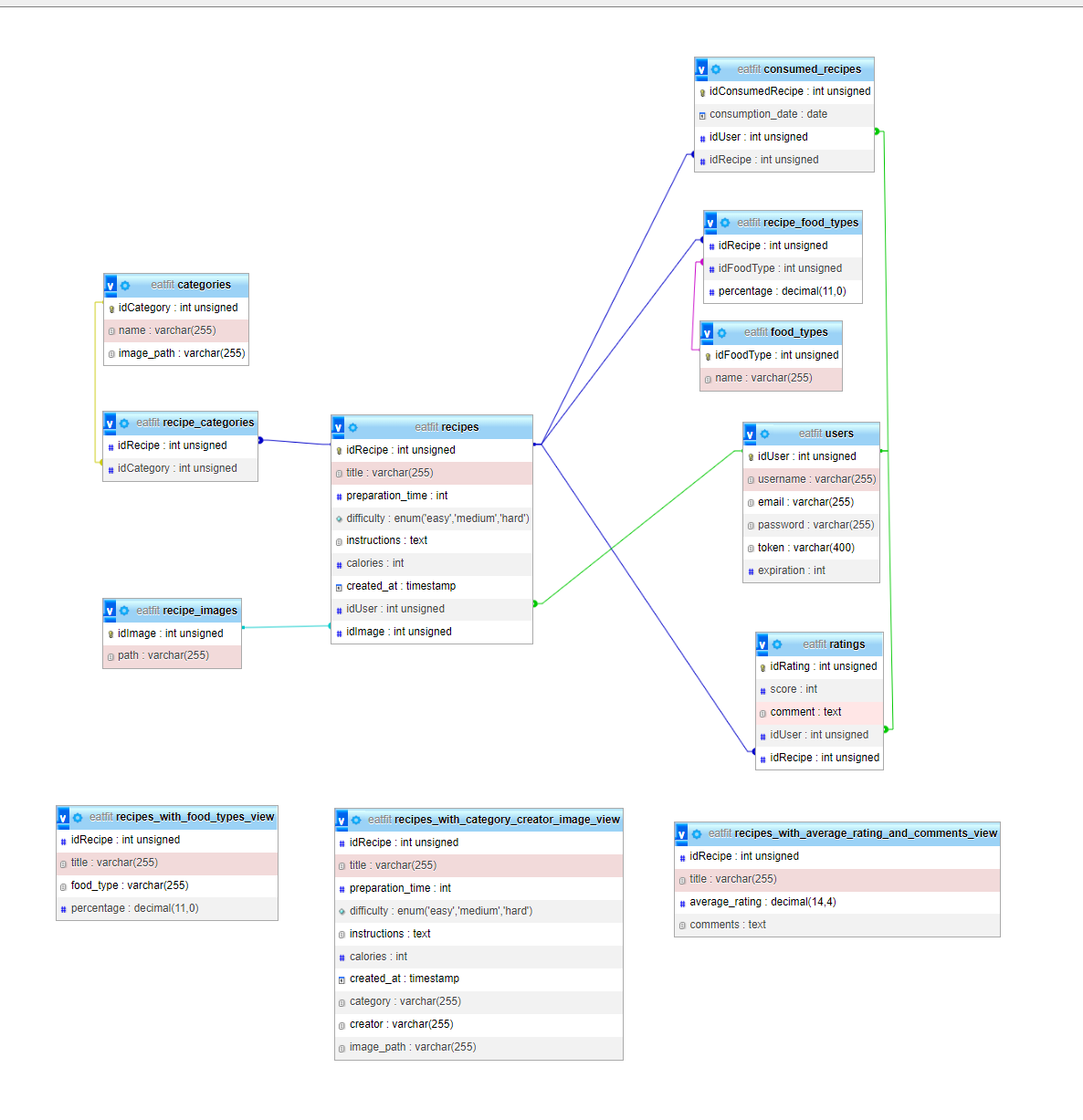
## **Base de données**

### **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.

### **MCD**

### **MLD**



### **Dictionnaire de données**

**categories**

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

**consumed\_recipes**

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

**food\_types**

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

**ratings**

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

**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 |  |

**recipe\_categories**

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

**recipe\_food\_types**

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

**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 |  |

**recipe\_images**

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

(Pas sur de garder les vues)

### **Architecture**

### **Structure (Découpage du projet et organisation des dossiers : MVC ou autres)**

# **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

# **Framework et outils externes**

## **Composants externes**

### **Bootstrap**

### **Font Awesome**

# **Analyse des fonctionnalités majeures (Critères et fonctionnalités de CdC)**

# **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 | 2.2 Structure de l'API / Adapter le modèle |
| User story | S17 : Faire la structure de l'API / S18 : Adapter le modèle de l'API à l'application |
| Situation | Le développeur a créé la structure de base de l'API pour gérer les requêtes entre l'application et la base de données. Le développeur a adapté 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 | Structure du modèle fait / |
| 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 | 3.2 Modification d’un profil utilisateur |
| User story | S4 : Gestion du profil utilisateur |
| Situation | L'utilisateur identifié peut modifier son compte. |
| Résultats obtenus | Retourne un message de succès et ce qui a été modifié |
| 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 | 3.4 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. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 4.2 Modification d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut modifier sa recette. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 4.3 Suppression d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut supprimer sa recette. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 4.4 Lecture d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut accéder à sa recette. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 4.5 Lecture d’une recette |
| User story | S7 : Gestion des recettes |
| Situation | L'utilisateur identifié peut accéder à sa recette. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 4.6 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. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 4.7 Filtrage des recettes |
| User story | S11 : Filtrage des recettes |
| Situation | L’utilisateur identifié peut filtrer les recettes par catégorie ou type d'aliments. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| 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). |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| 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. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| 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. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 5.4 Lecture d’une évaluation de recette |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L’utilisateur identifié voir les commentaires d’une recette. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 5.5 Lecture d’une évaluation de recette |
| User story | S9 : Gestion des évaluations des recettes |
| Situation | L’utilisateur identifié voir les commentaires d’une recette. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| 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. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 7.1 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. |
| Résultats obtenus |  |
| Statut |  |

|  |  |
| --- | --- |
| Nom | 8.1 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 |  |
| Statut |  |

## Évolution des tests

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N° test** | **J1** | **J2** | **J3** | **J4** | **J5** | **J6** | **J7** | **J8** | **J9** | **J10** | **J11** |
| 1.1 | ✘ | ✔ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 2.1 | ✘ | ✔ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 3.1 | ✘ | ✔ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 4.1 | ✘ | ✘ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 5.1 | ✘ | ✘ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 6.1 | ✘ | ✘ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 7.1 | ✘ | ✘ | ✔ | ✔ | ✔ |  |  |  |  |  |  |
| 7.2 | ✘ | ✘ | ✘ | ✔ | ✔ |  |  |  |  |  |  |
| 7.3 | ✘ | ✘ | ✘ | ✔ | ✔ |  |  |  |  |  |  |
| 7.4 | ✘ | ✘ | ✘ | ✘ | ✔ |  |  |  |  |  |  |
| 8.1 | ✘ | ✘ | ✘ | ✘ | ✔ |  |  |  |  |  |  |
| 8.2 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 8.3 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 8.4 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 9.1 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 9.2 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 9.3 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 10.1 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |
| 13.1 | ✘ | ✘ | ✘ | ✘ | ✘ |  |  |  |  |  |  |

# **Conclusion**

## **Difficultés rencontrées**

## **Variantes de solutions et choix**

## **Améliorations possibles**

## **Bilan personnel**

## **Remerciements**

# **Bibliographie**

# **Annexes**

## **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 | 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 | Retour d'information textuel d'un utilisateur sur une recette, généralement lié à l'évaluation donnée. |
| Historique des recettes consommées | Liste des recettes réalisées et consommées par un utilisateur, avec des informations sur les dates et les moments de consommation. |
|  |  |

Proportions de types d'aliments : Répartition des différents types d'aliments présents dans une recette, tels que les produits sucrés, les protéines, les produits laitiers, les matières grasses, les fruits et légumes, les féculents, et les produits céréaliers et légumineuses.

Graphique de répartition des types d'aliments : Représentation visuelle des proportions de types d'aliments consommés par un utilisateur sur une période spécifique (par exemple, une semaine, un mois, un an ou toute la période enregistrée).

Rôles d'utilisateur : Catégorisation des utilisateurs du site en fonction de leurs droits d'accès et de leurs fonctionnalités disponibles (par exemple, visiteur non-authentifié, membre authentifié, administrateur technique).

Vue : Représentation d'une table ou d'un ensemble de tables dans une base de données qui montre des données spécifiques sous un certain format.

LEFT JOIN : Opération SQL qui combine des lignes de deux ou plusieurs tables en fonction d'une condition de jointure, et inclut toutes les lignes de la table à gauche et les lignes correspondantes de la table de droite.

TPI : Travaux Pratiques Individuels, un projet d'évaluation en milieu scolaire pour tester les compétences pratiques des étudiants.

Terme technique

|  |  |
| --- | --- |
| Back-end |  |
| Utilisateur  (Anglais : user) | Personne qui utilise l’application |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## **Résumé tpi**

# **Résumé TPI**