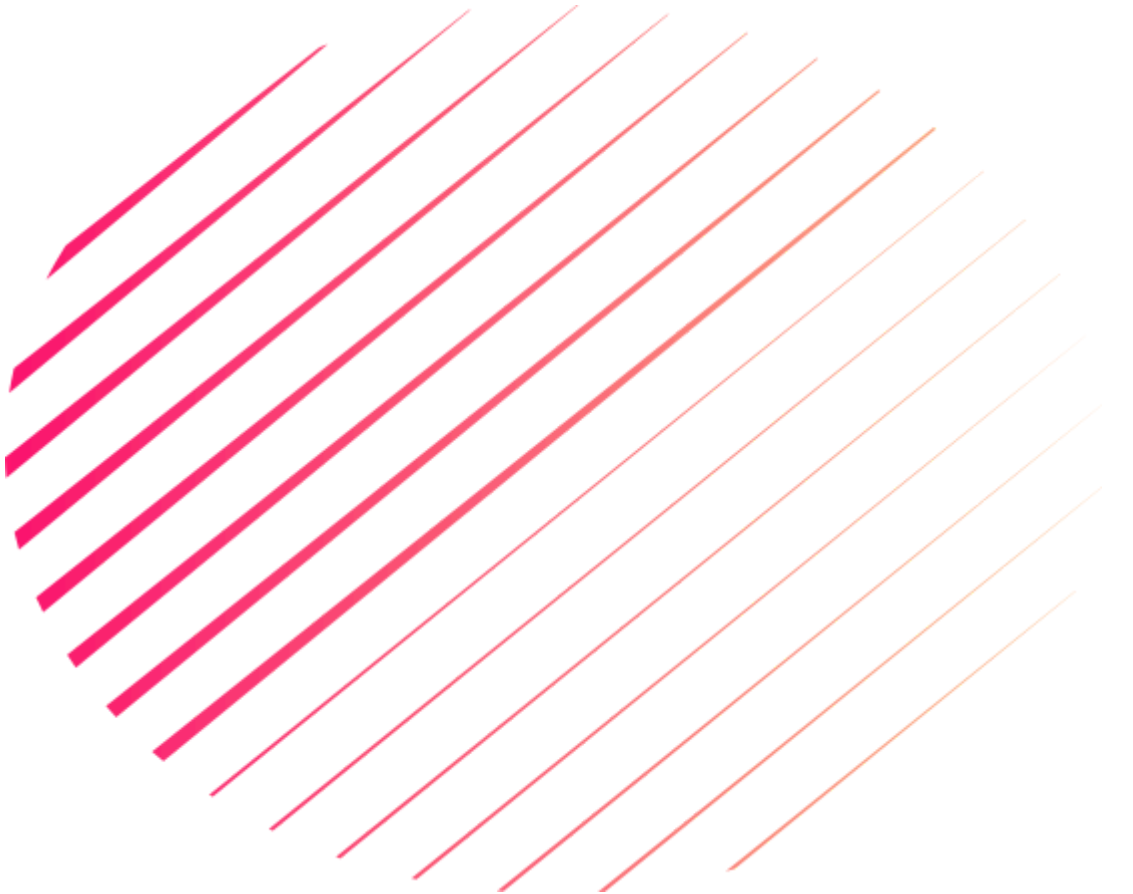


RAPPORT DE PROJET

« MANGER »



KHEDIR WASSIM
HADJARSİ MOHAMED EL FATEH
MOUAHEB ELIAS

BEAUREPÈRE MATHIAS
BOULAABI AHMED

Mise en place de Scrum.....	3
Utilisation de Git.....	4
Difficultés rencontrées.....	5
Technologies utilisées.....	7
PHP.....	7
GitHub.....	7
Discord.....	7
Bootstrap.....	8
MySQL.....	8
Composer.....	8
Ajax.....	9
Structure du projet.....	10
Organisation.....	10
Sprint 1.....	10
Sprint 2.....	13
Sprint 3.....	15
Fonctionnalités.....	18
Utilisateurs.....	18
Nutritionnistes.....	19
Administrateurs.....	19
Futur du projet.....	19
Conclusion.....	20

Introduction

Dans le cadre du cours "Architecture N-tiers", notre équipe a eu pour mission de développer une application web permettant aux nutritionnistes de suivre la nutrition de leurs clients, ainsi qu'aux utilisateurs de créer et sauvegarder des recettes. Ce projet avait pour objectif de mettre en pratique les concepts d'architecture n-tiers et de développement d'applications web modernes en utilisant le modèle MVC (Modèle-Vue-Contrôleur) avec PHP.

L'idée principale derrière ce site web était de fournir une plateforme conviviale et complète pour faciliter le suivi nutritionnel des patients par les professionnels de la santé, tout en offrant aux utilisateurs lambda un espace pour partager et conserver leurs recettes préférées.

Ce rapport a pour objectif de présenter le déroulement de ce projet, depuis la planification initiale jusqu'à la livraison finale, en mettant l'accent sur les choix techniques, les défis rencontrés et les leçons tirées de cette expérience de développement en groupe.







Méthode de travail

Mise en place de Scrum

En accord avec ce que le sujet préconisait, une méthode semblable à Scrum a été utilisée le long de ce projet, respecter la méthode Scrum telle quelle pouvant s'avérer compliqué dans un contexte de projet scolaire, où la gestion du temps est tout autre, et les membres de l'équipe sont moins expérimentés. Manquant d'assez d'expérience pour pouvoir évaluer correctement le temps que pourraient prendre les différentes tâches, nous avons opté pour des sprints de 1 mois, correspondant ainsi au délai avant une Release.

Pour jouer le rôle de Scrum Master, même si la méthode appliquée n'est qu'un substitut de Scrum, Elias Mouaheb a été choisi. Tout le long du projet, il se chargeait d'organiser des réunions, plus souvent en début et fin de sprint qu'en son milieu, avec les réunions de fin de sprint servant certes à se mettre d'accord sur les objectifs du prochain, mais aussi voir ce qui était à améliorer par rapport au sprint précédent. Il s'assurait que le code respectait bien les

PSR, était bien documenté, en expliquant aux différents membres du groupe comment s’y prendre (avec phpDocumentor), créait les issues sur GitHub, et expliquait au groupe la convention de nommage des branches qu’il faudrait utiliser sur GitHub. Les autres membres du groupe ont pu ainsi se concentrer plus efficacement sur le code.

Branch	Updated
main 	 yesterday
chore/elias-improve-autoloader 	 yesterday
fix/verification-plan-modif 	 3 days ago

Exemples de branches créés

Nous nous réunissions aussi après ou avant des cours pour faire un récapitulatif des tâches de chacun concernant le projet, dans ce qui pourrait se rapprocher d’un Daily Scrum, sauf que non journalier. L’utilisation d’idées clés de la méthode Scrum aura eu un effet bénéfique sur le déroulement du projet, tant par le système de sprint et le Product Backlog allant avec que par la communication quasi constante permettant une vraie transparence.

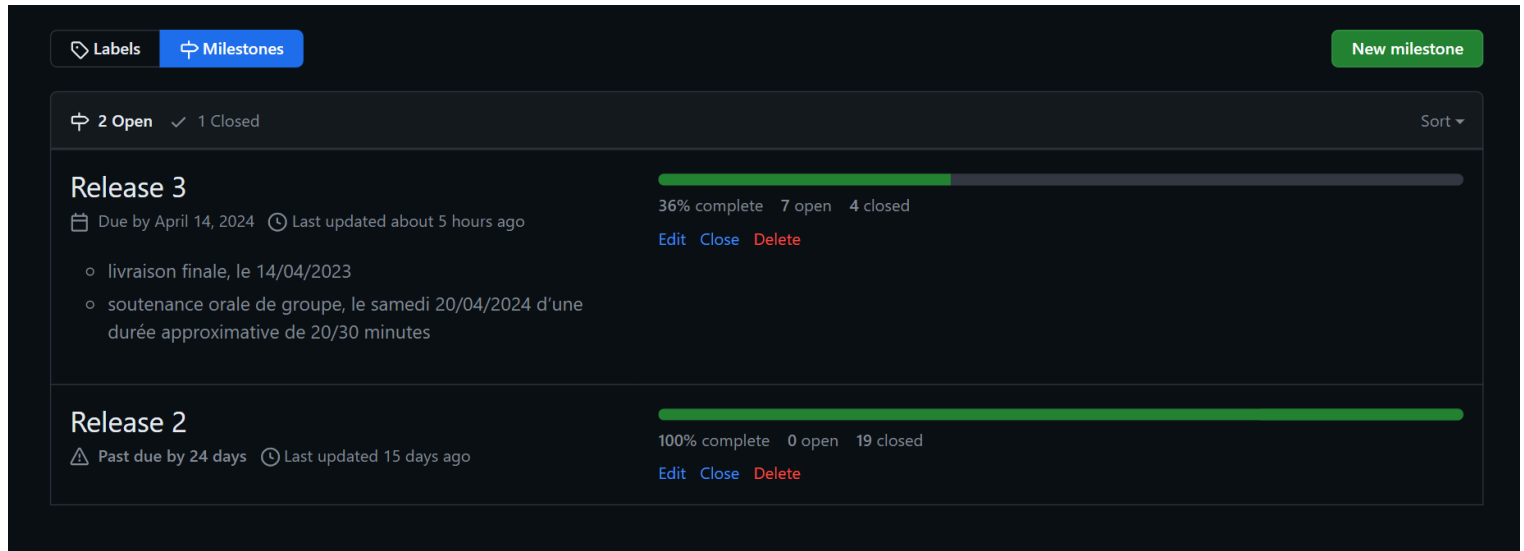
Utilisation de Git

GitHub a été utilisé principalement pour 2 aspects dans ce projet. Le premier étant le système de branche, pour lequel nous avons décidé de respecter une convention de nommage « type/raison-de-la-branche », comme par exemple « feature/make-login-responsive », en accord avec le "Git Feature Branch Workflow". Chacun travaillait donc sur sa branche, puis une fois sa tâche terminée, faisait une Merge Request, de manière à ce que d’autres membres puissent vérifier le contenu modifié, avant de merge dans la branche main. Le second grand intérêt de GitHub a été les issues, utilisées comme tâches dans le cadre de ce projet.

Pour la gestion des sprints, deux étiquettes supplémentaires ont été créées, “à faire” pour marquer les tâches du sprint actuel, et “backlog” pour les tâches de sprints futurs. D’autres tags ont été créés pour spécifier le type de tâche comme “nutritionniste” pour celles se rapportant aux nutritionnistes.

Toutes les tâches étaient liées à des milestones, une autre fonctionnalité de GitHub. Ces milestones représentaient dans notre cas les différentes Release, Release qui ont ensuite pu être publiées grâce au propre système de Release GitHub. Enfin, nous avons profité du

README que GitHub met en évidence sur les dépôts pour avoir facilement accès aux différentes conventions à respecter en codant afin de les garder en tête.



Les Release comme milestones

Difficultés rencontrées

En attendant qu'un dépôt GitHub nous soit attribué, un autre dépôt avait été utilisé, afin de perdre le moins de temps possible. C'est pour ça que le nombre de commits, de tâches et de branches sur le dépôt officiel au début du projet ne reflète pas l'entière réalité. Bien que cette manière de faire ait permis un gain de temps, cela a aussi entraîné des difficultés au moment de faire la transition vers le dépôt officiel.

L'arrivée d'un nouveau membre au cours du premier sprint nous aura aussi forcé à repenser la répartition du travail, en plus du fait que plus le nombre de membres d'un groupe augmente, plus il devient compliqué de s'organiser, notamment en ce qui concerne les réunions.

La plus grande difficulté à surmonter aura été le manque d'expérience de notre groupe, par rapport à PHP d'une part, et la méthode Scrum d'une autre. Concernant PHP, les TDs ont permis de progresser et de se familiariser avec l'environnement, par exemple Composer, ou l'implémentation du modèle MVC. Le manque d'expérience Scrum par contre n'a pu être pallié que grâce à de la documentation et aux cours.

Déroulement

Au début du projet, le premier objectif a été de définir les différentes fonctionnalités à implémenter pendant sa durée. Après avoir éclairci quelques doutes avec le client pour avoir une idée plus précise du résultat attendu, nous avons commencé à prévoir les sprints. Ne maîtrisant encore que peu PHP, le premier sprint et donc la première Release, avait comme objectif de pouvoir livrer une interface de connexion fonctionnelle avec inscription, une page d'accueil, ainsi qu'une autre page permettant de créer des recettes.

En outre, ce sprint visait à mettre en place une architecture MVC cohérente, et une base de données utilisable et sensée. Ce dernier point a été l'objet de plusieurs réunions pour se mettre d'accord, car la base de données serait une partie importante du projet, et qui se devait donc de fonctionner comme on l'entendait. Pendant ce temps, les différents membres du groupe avaient aussi comme tâche d'apprendre les bases de l'architecture MVC, et notamment avec PHP.

Les tâches ont été attribuées en fonction de l'aisance des membres, à mesure que les connaissances PHP des membres s'amélioraient. Par exemple Ahmed Boulaabi a vite compris comment utiliser Ajax avec PHP et jQuery, et a donc pu s'occuper de la partie backend de l'interface de connexion. Il était aussi très important de maintenir la communication, pour que personne ne s'isole et s'enfonce si bloqué dans sa tâche, surtout au début du projet. Le rôle du Scrum Master a été de s'assurer que tous les membres étaient bien sur leurs tâches. A cette fin, pouvoir échanger avant ou après un cours s'est avéré très utile.

Ce premier sprint servait donc à poser les bases dont nous allions constamment nous servir. C'est aussi au cours du premier sprint que Composer a commencé à être utilisé, principalement pour son autoloader.

Le second sprint, après une réunion du sprint précédent (pour voir ce qui s'était bien passé, ou moins bien passé, vérifier si l'architecture MVC était bien respectée, etc.), portait sur l'aspect nutritionniste, avec l'ajout d'un dashboard pour les nutritionnistes, et des fonctionnalités allant avec. Encore une fois, la clé a été l'organisation pour se mettre d'accord sur les différents aspects des fonctionnalités, et leurs implications, afin d'éviter tout malentendu.

Le même principe a été suivi pour la troisième et dernière Release, avec une attention encore une fois toute particulière à la transparence, afin d'être certain que tous les membres aient le même cap.

Pour chaque Release est paru un guide utilisateur pour permettre une prise en main facile par le client, ainsi qu'une documentation générée par phpDocumentor et un script pour mettre en place la base de données.

Architecture

Technologies utilisées

PHP

En conformité avec les demandes du sujet, nous avons utilisé PHP, un langage qui nous a permis de bien prendre en main le modèle MVC, grâce à sa facilité d'apprentissage et sa documentation accessible. De plus, PHP offre un excellent support pour l'interaction avec les bases de données grâce à PDO, qui a été utilisé tout au long de ce projet.

GitHub

GitHub nous a permis de bénéficier des avantages du contrôle de version avec Git, tout en offrant des fonctionnalités supplémentaires pour le travail d'équipe. L'utilisation des tâches et branches proposées par Git nous a permis de faciliter le travail en équipe, et la sûreté du code source, en travaillant sur nos propres branches. La coordination permis par GitHub a été très utile pour voir le travail de chacun.

Discord

Discord a eu plusieurs intérêts dans le cadre de ce projet. Le premier est celui d'avoir une messagerie instantanée classique permettant de communiquer avec n'importe quel membre du groupe. Un autre avantage était de pouvoir prévoir des réunions en s'adressant à tous les membres du groupe à la fois, ou même de faire une réunion à distance si plusieurs membres ne peuvent pas se déplacer.

Discord dispose aussi d'une fonctionnalité de partage d'écran facile à mettre en place pour aider un autre membre du groupe s'il rencontrait un problème dans son code. Enfin, le fait de pouvoir alerter tous les membres du groupe en utilisant la commande « @everyone » rendait simple le fait de notifier le groupe de toute manipulation à faire sur le code après avoir pull la dernière version de la branche main par exemple, comme mettre à jour la base de données, ou l'autoloader de composer.

Bootstrap

Afin d'accélérer le développement de l'interface utilisateur de notre application et de garantir une expérience cohérente sur différents appareils, nous avons décidé d'utiliser Bootstrap, un puissant framework front-end open-source.

MySQL

Nous avons choisi MySQL dans notre projet PHP MVC pour sa popularité et sa stabilité, avec l'utilisation de PDO pour une gestion de base de données flexible. Composer a facilité la gestion des dépendances, y compris le pilote MySQL pour PDO, simplifiant ainsi le processus de développement.

Composer

Composer est un outil de gestion de dépendances et de construction pour les applications PHP. Il a été très utile dans ce projet en tant que facilitateur, et de plusieurs manières. Au fur et à mesure que notre maîtrise de PHP s'améliorait, le fichier Composer.json évoluait pour apporter plus encore au projet.

Il a permis l'ajout facile de package, on peut penser entre autres à "dotenv" pour utiliser les variables d'environnement, ô combien utiles pour manier une base de données.

Un autre apport non négligeable est l'autoloader mentionné plus haut, pour permettre de lier facilement les différentes parties de l'architecture MVC, et rend la création de nouvelles classes assez simple. Parmi les autres apports de Composer, on peut penser aux fichiers globaux qui permettent d'utiliser des variables ou des fonctions depuis n'importe où dans le code, sans avoir à ajouter un import pour chaque utilisation. Une dernière utilisation de Composer que nous avons mise en place concerne les scripts, fussent-ils pour lancer le server, mettre à jour la base de données après modification du fichier SQL ou même générer la documentation avec phpDocumentor « as code ».


```

1  {
2      "name": "groupe06/arch-ntiers",
3      "description": "Application permettant un suivi des repas",
4      "type": "project",
5      "require": {
6          "php": ">=7.3.12",
7          "vlucas/phpdotenv": "*"
8      },
9      "autoload": {
10         "psr-4": {
11             "Manger\\Controller\\": "app/Controller",
12             "Manger\\Model\\": "app/Model",
13             "Manger\\View\\": "app/View",
14             "Manger\\Helpers\\": "app/Helpers",
15             "Manger\\": "app",
16             "PHPMailer\\PHPMailer\\": "app/PHPMailer/src/",
17             "Config\\": "Config/"
18         },
19         "files": [
20             "app/View/Utils/global.php",
21             "Config/Globals.php"
22         ]
23     },
24     "config": {
25         "sort-packages": true
26     },
27     "scripts": {
28         "post-install-cmd": [
29             "php init_database.php"
30         ],
31         "post-update-cmd": [
32             "php init_database.php"
33         ],
34         "phpdoc": "phpDocumentor -d app Config -t doc --ignore app/PHPMailer/",
35         "serve": "php -S localhost:8000 -t ../"
36     },
37     "minimum-stability": "stable",
38     "prefer-stable": true
39 }

```

Composer.json final

Ajax

Ajax (JavaScript asynchrone et XML) est une technique de développement web qui permet aux applications web de communiquer avec un serveur en arrière-plan, sans recharger toute la page. Cette méthode met à jour le contenu de manière dynamique, permettant une expérience utilisateur plus fluide et réactive.

Nous avons utilisé Ajax dans notre projet MVC PHP pour améliorer l'interaction utilisateur et l'efficacité. Cela permet des mises à jour de contenu en temps réel sans rechargement de page, accélérant l'application et améliorant la satisfaction des utilisateurs en rendant les opérations telles que la recherche d'utilisateurs ou de recettes et les soumissions de formulaires plus fluides.

Structure du projet

Ce projet a été fait en respectant l'architecture Modèle-Vue-Contrôleur, permettant ainsi d'avoir une séparation des préoccupations réussie. Elle comporte en son centre le fichier `index.php`, qui reçoit les requêtes, et appelle le routeur pour s'en occuper. De cette manière, le programme reste dans l'`index`. Le routeur se charge ensuite de transférer les requêtes au bon contrôleur prévu, ou vers une page d'erreur si la requête est inconnue. Le contrôleur appelle ensuite le modèle si nécessaire, par exemple pour toute manipulation de la base de données.

Une classe appelée `Database` facilite d'ailleurs son utilisation, permettant un maniement de l'objet PDO plus simple et récupérant les variables d'environnement stockées dans le fichier `.env`. Une fois que le modèle a renvoyé son résultat au contrôleur, ce dernier gère l'affichage, soit en appelant la vue qui utilisera les fichiers PHP stockés dans les dossiers `template` et `components`, soit en répondant à la requête Ajax en envoyant les résultats de la requête en JSON, que l'Ajax dans le fichier javascript éponyme se chargera d'utiliser pour modifier l'affichage. Les requêtes envoyées au routeur en passant par l'`index` sont aussi envoyées utilisant Ajax, notamment la fonction `performAjaxRequest`.

Répartition des tâches

Organisation

La répartition des tâches lors des sprints ci-dessous est décrite membre par membre, avec à chaque fois le nom du membre concerné mis en valeur.

Sprint 1

Pour le premier sprint, **Elias Mouaheb** a géré la mise en place de Composer, donc modifié le code pré-existant afin d'utiliser les namespaces proposés par Composer, ainsi que placés les variables globales du projet dans un fichier à part pour que Composer permette de les utiliser facilement depuis n'importe quel autre fichier. Un autre enjeu était d'expliquer le fonctionnement et l'utilité de Composer aux autres membres du groupe.

Il s'est ensuite occupé de la première version du routeur, avant que Ahmed Boulaabi ne l'améliore. Le principal objectif d'Elias lors de ce sprint particulièrement était de s'assurer de la bonne communication au sein du groupe, pour avoir des bases solides.

Il a ensuite aidé Wassim lors de la création de recettes, et géré le cas des URL fausses devant être redirigées. Puis il a créé le script permettant de mettre à jour facilement la base de données à partir du fichier SQL, et celui qui installe toutes les dépendances requises sur Ubuntu pour lancer le projet.

Il a aussi enlevé certaines redondances pour améliorer la qualité du code, puis a décrit comment la documentation se ferait le long de ce projet aux membres du groupe, soit avec phpDocumentor, dont un exemple d'utilisation était disponible sur le README. Enfin, il s'est chargé de générer la documentation après s'être assuré que toute fonction et toute classe était documentée, et a créé la Release correspondant au premier sprint.

De son côté, **Hadjarsi Mohamed El Fateh** a fait appel à Figma pour créer une ligne directrice Web Design et Branding unique et invitante afin d'assurer la cohérence des pages en termes de design et d'expérience utilisateur (UX). Grâce à son expérience dans le développement Web, il a écrit le DBML (Database Markup Language) pour la base de données et a généré un schéma solide que nous n'avons pas eu beaucoup à modifier depuis, ce qui nous a fait gagner beaucoup de temps. Il a également présenté un plan complet sur la façon dont chaque partie du site Web devrait fonctionner, facile à suivre et qui a permis d'éliminer une grande partie des conjectures.

Il a ensuite écrit le squelette MVC ainsi que les vues PHP, le CSS et les pages et composants réutilisables suivants : En-tête, Connexion, Registre, Pages de réinitialisation du mot de passe, Page d'accueil/Tableau de bord, Pages de paramètres utilisateur. Et pour le backend, il a écrit les fonctions nécessaires aux pages de paramètres et a rendu les pages fonctionnelles.

Après que Fateh ait réalisé la conception des pages de connexion, d'inscription et d'accueil, ainsi que l'architecture de base de données, **Ahmed Boulaabi** a pris le relais pour mettre en œuvre cette architecture dans PHPMyAdmin en utilisant SQL.

Ahmed a ensuite initié le développement du backend en créant les modèles et les contrôleurs pour la connexion et l'inscription. Il a utilisé le hachage PHP pour sécuriser les mots de passe, assurant ainsi la confidentialité des informations des utilisateurs.

En outre, il a mis en place la fonctionnalité de réinitialisation des mots de passe en utilisant PHPMailer pour envoyer des e-mails de réinitialisation. Chaque jeton de réinitialisation est unique et n'est utilisable qu'une seule fois, en plus d'avoir une expiration, afin de renforcer la sécurité du processus.

En parallèle, Ahmed a également développé la première version du fichier `ajax.js`. Cette contribution a facilité la communication entre les vues et les contrôleurs, améliorant ainsi l'expérience utilisateur en permettant une interaction plus fluide et dynamique avec l'application.

Enfin, il a ajouté au modèle MVC conçu les différents composants requis pour le formulaire de premier login, fussent-ils backend ou frontend. Lorsque les utilisateurs se connectent pour la première fois, ils doivent remplir un formulaire pour calculer les calories nécessaires en fonction de divers critères tels que le régime alimentaire, l'âge et le sexe. Ahmed a veillé à ce que ce formulaire soit fonctionnel et précis, offrant ainsi une expérience utilisateur personnalisée dès le début de leur utilisation de l'application.

De plus, il a rendu le header dynamique, dépendant de l'état de connexion de l'utilisateur, offrant une expérience utilisateur plus personnalisée.

Wassim Khedir, principalement orienté vers le développement mobile, a été confronté à l'apprentissage de PHP pour la première fois. Conscient de ce qui allait être demandé de lui au cours de ce projet, il a commencé à travailler sur les designs des pages pour l'ajout et l'affichage des recettes.

Grâce aux TD et à des ressources en ligne telles que des tutoriels YouTube, Wassim a progressé dans son apprentissage de PHP, réalisant deux mini-projets CRUD dans le seul but d'apprendre ce langage. Étant déjà familier avec l'architecture MVC grâce à son expérience dans le développement mobile avec framework Flutter, il a trouvé la transition vers cette architecture dans le contexte PHP relativement aisée.

Il a créé des pages flexibles, adaptées à différentes tailles d'écrans, et mis en place un contrôleur avec son modèle correspondant pour gérer l'ajout et l'affichage des recettes. Wassim a également modifié la base de données en ajoutant, par ailleurs, une auto-incrémentation à la table des recettes. De plus, il a effectué une modification dans la fonction "execute" du fichier "database.php" dans le but de changer le format de récupération des données de la base de données. En remplaçant `FETCH_ASSOC` par `FETCH_OBJ`, il a souhaité récupérer les données sous forme d'objets plutôt que sous forme de tableaux associatifs.

Lorsque Wassim a partagé son travail avec Ahmed et a comparé leurs approches, il a rapidement compris l'utilité d'AJAX, même si cela représentait un départ difficile pour lui. Cependant, il a rapidement entrepris d'apprendre et de comprendre l'utilisation d'AJAX, ainsi que la manière d'afficher des parties de la page dynamiquement grâce à cette technologie. Il est important de noter qu'AJAX a entraîné plusieurs changements dans les méthodes de travail de Wassim. Il a dû consacrer du temps pour apprendre ces nouvelles méthodes et a même dû réajuster son approche de travail.

Enfin, **Mathias Beaupère** a pu travailler sur différents contrôleurs lors de ce premier sprint, ainsi que nettoyer du code pour le rendre plus maintenable.

Sprint 2

Pour le second sprint, **Elias Mouaheb** devait implémenter le système de notifications permettant de lier les clients aux nutritionnistes, et vice-versa. Il fallait donc commencer par ajouter une icône sur le dashboard pour ouvrir une fenêtre modale elle aussi implémentée pour l'occasion.

Cliquer sur les noms apparaissant permet de leur envoyer une notification, correspondant à une entrée dans la table notifications créé dans la base de données pour l'occasion, avec comme champs les ids de l'utilisateur envoyant la notification et celui de l'utilisateur à la réception, ainsi qu'un nombre indiquant l'état de la notification.

Il a fallu aussi créer une autre table dans la base de données pour représenter les liens entre les clients et les nutritionnistes. Ajax a été pratique pour changer l'affichage de la notification directement en se basant sur son état en cliquant, en plus de stocker dans la base de données. Une autre icône permettant de voir les notifications reçues a dû être ajoutée à côté, avec elles aussi une requête Ajax permettant de les accepter ou les décliner, sachant qu'accepter va créer une relation entre le client et le nutritionniste impliqués dans la base de données.

De plus, pour rendre l'utilisation de l'architecture MVC plus claire encore au sein du projet, Elias a renommé certains fichiers et modifié le fichier Composer.json en accord avec ces noms, se débarrassant d'un certain nombre de redondances présentes dans le projet, en particulier en rassemblant les différents "define" dans un même fichier accessible depuis n'importe quel autre fichier grâce à Composer.

Enfin, il a créé le guide utilisateur pour cette Release, s'est assuré que les différentes fonctions et classes étaient bien documentées, généré la documentation correspondante avec phpDocumentor, et créé la Release 2.

Dans le même temps, **Hadjarsi Mohamed El Fateh**, a créé la base de la page Planning qui gère le plan de repas de l'utilisateur et son fichier CSS, a écrit le composant Modal réutilisable, et la recherche fonctionnelle de recettes, et pour cela il a également écrit la méthode Debounce qui retarde l'exécution d'une fonction. jusqu'à ce qu'une période déterminée d'inactivité soit écoulée.

Celle-ci est particulièrement utile dans les barres de recherche, où cela empêche un appel à la base de données de se déclencher après chaque frappe, réduisant ainsi les requêtes inutiles du serveur et améliorant les performances.

Fateh a également écrit les fonctions backend PHP pour la barre de recherche de recettes, écrit la logique derrière les paramètres du plan (Période, Durée et Nom) tout comme l'affichage dynamique de chaque date en fonction des paramètres, ainsi que l'ajout de recettes au conteneur de chaque date, et finalement le stockage de toutes ces données dans localStorage ainsi que leur récupération pour la persistance des données.

Il s'est également assuré que tous ces composants seraient non seulement fonctionnels mais aussi plaisant aux yeux des utilisateurs.

Ahmed Boulaabi a réalisé plusieurs tâches essentielles pour améliorer la fonctionnalité et l'expérience utilisateur du site web. Il a conçu un tableau de bord pour les nutritionnistes, permettant l'accès aux listes de clients avec des cartes clients codées par couleur en fonction de leur régime alimentaire. De plus, il a élaboré un tableau de bord administrateur pour gérer les utilisateurs.

Il a développé des fonctionnalités CRUD (Create, Read, Update, Delete) pour la gestion des listes d'utilisateurs dans le tableau de bord administrateur, assurant ainsi une gestion efficace des données utilisateur. Il a également implémenté une page d'erreur 404 personnalisée pour améliorer l'expérience utilisateur en cas de page non trouvée.

Ahmed a ajouté une fonctionnalité de mode sombre pour les tableaux de bord administrateur et nutritionniste, garantissant ainsi une expérience utilisateur plus agréable.

De plus, il a mis en place un contrôle d'accès basé sur les rôles pour restreindre l'accès des utilisateurs à des pages spécifiques en fonction de leur rôle attribué. Cette fonctionnalité a renforcé la sécurité du système en limitant l'accès aux informations sensibles.

Wassim Khedir a effectué une revue de la première version du projet. Ensuite, il a amélioré la liste des recettes, en ajoutant quelques fonctionnalités pour offrir une meilleure expérience utilisateur. Parallèlement, il a résolu les problèmes de flexibilité découverts lors des tests de la première version. Wassim optimisa ainsi la structure du code en éliminant les redondances inutiles présentes dans certains contrôleurs et modèles, et intégra le contrôleur "recipesController" dans "UserController" et "recipesModel" dans "UserModal", après quoi il a apporté les changements nécessaires pour assurer la cohérence et le bon fonctionnement du système.

Une autre optimisation faite par Wassim a été la correction d'une erreur qui se produisait lorsque aucune notification n'était présente dans la base de données.

Bien que Wassim ait mis du temps à comprendre la logique utilisée par Fateh dans la page de planification, il démarre sa tâche en créant un bouton “Ajouter un planning” dans la page planning et continue son travail jusqu'à ce qu'il ajoute avec succès un nouveau planning associé à l'utilisateur dans les données de la base de données. La tâche suivante consistait à récupérer et afficher le planning de l'utilisateur.

Puisque le tableau de bord ne disposait que d'un bouton pour accéder à la planification, un système de vérification a été mis en place pour déterminer si l'utilisateur dispose d'un plan lorsqu'il ouvre la page de planification. Wassim a géré la situation dans laquelle l'utilisateur avait un plan, la page du plan sera modifiée pour afficher le plan de l'utilisateur avec le nom, le nombre de calories et l'image des recettes et afficher un bouton pour modifier le plan. Le backend de cette tâche était un défi majeur car il impliquait la récupération d'informations provenant de trois tables distinctes. L'objectif était d'associer correctement ces données pour afficher les recettes au bon endroit avec les informations appropriées.

En plus, il s'est occupé de l'ajout de recette par un nutritionniste dans la base de données.

À la fin, Wassim a tenté de travailler sur la modification de planning, mais sans succès. Il a rencontré des difficultés car le changement de période ou de durée, ainsi que l'ajout de recettes au plan, entraînaient le rafraîchissement de la page. Ce rafraîchissement forçait l'AJAX à récupérer le plan original, annulant ainsi les modifications apportées.

Mathias Beaupère a principalement travaillé sur le tableau de bord administrateur, en parallèle avec Ahmed Boulaabi. Qui plus est, Mathias a contribué au développement de la page de gestion des recettes. Il a ajouté des fonctionnalités permettant l'ajout et la suppression de recettes, facilitant ainsi les modifications de la base de données directement depuis le tableau de bord administrateur. Il a également travaillé sur la conception des icônes de la barre d'en-tête du site. Mathias a également travaillé sur l'intégration de l'affichage des statistiques du site sur la page d'administrateur.

Sprint 3

La fonctionnalité principale implémentée par **Elias Mouaheb** lors du troisième sprint est la messagerie entre un nutritionniste et ses clients. Il s'agit d'ajouter une bulle de discussion sur l'écran d'accueil connecté pour tout utilisateur qui affiche le dernier message échangé avec son nutritionniste.

En cliquant sur cette bulle, une division modale apparaît, permettant de voir la totalité de la discussion et d'y contribuer. Grâce à Ajax, les messages envoyés apparaissent directement, en plus d'être stockés dans la base de données. Lorsqu'un nutritionniste se connecte, cette bulle n'apparaît pas, et il doit plutôt aller sur le dashboard propre aux nutritionnistes, développé par Ahmed Boulaabi, pour avoir accès à sa liste de conversations avec ses différents clients. Dans les deux cas, la date d'envoi du message apparaît sous chaque message, permettant de contextualiser la conversation.

Il a aussi aidé les membres du groupe à gérer les conflits de Merge, amélioré le Composer.json pour rendre l'autoloader plus efficace et corrigé des problèmes survenant à la connexion dans certains cas.

Hadjarsi Mohamed El Fateh a écrit les fonctions PHP et l'Ajax pour récupérer les données sur les utilisateurs liées au nutritionniste et calculer la progression de leur plan, ce qui nous donne également des statistiques telles que le nombre total d'utilisateurs pour ce nutritionniste, le nombre de plans terminés et non terminés. Il a réécrit l'ensemble du tableau de bord du nutritionniste pour afficher les statistiques et les utilisateurs gérés par le nutritionniste.

Le tableau de bord utilisateur a aussi été réécrit pour afficher les informations du plan telles que le nom et la progression, ainsi qu'un graphique dynamique pour suivre la consommation calorique quotidienne, un affichage dynamique du plan de repas d'aujourd'hui et un historique des jours passés. Il a également permis de basculer l'état d'une recette en "consommé" via le tableau de bord et de le refléter dans la base de données, finalisant le tableau de bord et le transformant en une partie entièrement fonctionnelle car il s'agit de l'interface principale de l'utilisateur avec l'application et elle est à l'avant-garde.

Il a ajouté au travail de Wassim sur la modification du plan, la possibilité d'afficher l'icône de suppression et la fonctionnalité permettant de supprimer une recette du plan, et a aidé certains membres de l'équipe à réaliser leurs tâches. Finalement, Fateh a effectué une vérification avant la Release pour s'assurer que les différentes fonctionnalités n'avaient aucun problème. Ce faisant, il a remarqué des défaillances concernant l'affichage de la liste des recettes depuis le dashboard de nutritionniste, qu'il a corrigé tout en améliorant la structure du code et l'affichage de la liste, rendant cette dernière bien plus compréhensible au regard. Il a aussi corrigé divers autres soucis d'affichage qui se produisaient dans certaines situations.

En utilisant les bases posées par Fateh, **Ahmed Boulaabi** a ensuite réécrit entièrement le tableau de bord du nutritionniste pour intégrer de nouvelles fonctionnalités. Cela inclut la capacité de trier les clients en fonction de leur progression dans leur plan, de n'afficher que les six meilleurs clients dont les plans sont presque terminés, et d'attribuer dynamiquement des

couleurs aux cartes des clients en fonction de leurs objectifs. Ces améliorations ont considérablement enrichi l'expérience utilisateur et la fonctionnalité globale du tableau de bord.

Il a permis le développement de fonctionnalités cruciales pour optimiser le tableau de bord du nutritionniste. Il a conçu une fonctionnalité permettant d'afficher de manière exhaustive la liste des clients, même pour ceux qui n'ont pas encore de plan en cours. En parallèle, il a intégré des boutons de suppression et de modification pour chaque client dans la liste, fournissant ainsi une interface pour une gestion efficace des clients. De plus, Ahmed a implémenté la logique nécessaire pour permettre une suppression de client.

Finalement, Il a créé une section dédiée pour gérer les demandes des nutritionnistes, conçu une interface intuitive pour afficher les requêtes des aspirants nutritionnistes, et mis en place une fonction pour récupérer les demandes des nutritionnistes de la base.

Wassim Khedir s'est concentré sur la modification du système pour permettre aux nutritionnistes de voir les plans de leurs clients. Suivant la structure utilisée par Ahmed, Wassim a d'abord mis en place la méthode pour récupérer l'ID du client visé, puis a réutilisé la page de planning existante tout en la transformant en un composant adaptable pour différents clients. Ces ajustements ont nécessité plusieurs mises à jour et modifications telles que le changement de la variable de stockage local, l'obligation d'envoyer l'id de l'utilisateur et l'obligation de supprimer la liste de recettes dans le stockage local.

Wassim est retourné à son défi initial, qui était de rendre possible la modification des plans par l'utilisateur. Finalement, il a réussi à trouver une solution. Ce n'était pas facile à réaliser car il devait permettre l'ajout, l'affichage et la mise à jour de tout cela dans la même page, et il n'était pas possible de les diviser en différentes pages. Ainsi, il a implémenté tout le backend nécessaire pour assurer le succès de cette modification.

Après, il a suivi la même logique avec cependant quelques modifications pour rendre un nutritionniste capable de modifier les plannings de ses clients en y ajoutant des recettes et modifiant le nom du plan, la durée et la période. Il a suivi la même logique que Fateh a utilisée pour rendre le nutritionniste capable de supprimer des recettes depuis le plan d'un client.

De plus, Wassim a amélioré la fonctionnalité d'insertion d'une recette dans le plan en récupérant directement l'image et le nombre de calories correct associé à la recette.

A la fin, il a finalisé son travail par une révision globale, l'ajout de la documentation sur les différentes fonctions créées, et quelques corrections concernant l'ajout de recettes.

Mathias Beaupère a travaillé sur la finalisation de la page d'administrateur, notamment en ajoutant des fonctionnalités de modification des utilisateurs et des recettes. De plus, il a intégré une page de demande de promotion, accessible depuis les paramètres, qui permet à un utilisateur régulier de demander une promotion pour devenir nutritionniste. Cet

onglet de promotion n'est accessible qu'aux utilisateurs réguliers et est donc indisponible pour les nutritionnistes et les administrateurs. La gestion de ces demandes, visible sur la page d'administrateur, comprend l'acceptation, qui change automatiquement le rôle du demandeur en nutritionniste, ou le refus, qui supprime la demande du tableau de bord administrateur.

Fonctionnalités

Les fonctionnalités sont expliquées plus en détail dans les différents Guides Utilisateurs trouvables dans le dossier du même nom sur le dépôt GitHub.

Utilisateurs

- Peut réinitialiser le mot de passe
- Peut s'inscrire
- Peut se connecter avec e-mail et mot de passe
- Peut modifier les détails et les paramètres du compte
- Peut se fixer un objectif (prendre du poids, perdre du poids, perdre du poids rapidement)
- Un objectif calorique est calculé pour l'utilisateur en fonction de ses informations (en utilisant le poids, la taille et l'objectif)
- Peut créer/modifier un plan de repas
- Peut définir des paramètres pour le plan de repas tels que la période et la durée
- Peut ajouter des repas/recettes à chaque jour du plan
- Peut rechercher des repas/recettes
- Peut ajouter/créer des recettes personnalisées
- Peut demander un nutritionniste
- Peut discuter avec son nutritionniste
- Peut suivre sa consommation et son plan depuis le tableau de bord
- Peut basculer l'état « consommé » de n'importe quel repas de son plan
- Peut se déconnecter

Nutritionnistes

A accès aux fonctionnalités des utilisateurs classiques

Peut voir des statistiques sur ses clients

Peut créer/modifier un plan pour ses clients

Peut discuter avec ses clients

Peut ajouter/faire de nouvelles recettes

Peut ajouter des clients

Peut changer l'interface en mode sombre ou clair

Administrateurs

A accès aux fonctionnalités des utilisateurs classiques

Peut gérer les utilisateurs, y compris leur création, modification et suppression.

Peut gérer les recettes, notamment en ajoutant de nouvelles recettes, en les modifiant ou en les supprimant.

Peut gérer les demandes des nutritionnistes, en examinant, approuvant ou rejetant les demandes soumises pour devenir nutritionniste.

Peut accéder à des statistiques sur l'utilisation de la plateforme et sur les données des utilisateurs.

Peut changer l'interface en mode sombre ou clair

Bilan

Futur du projet

Il est probable qu'avec une meilleure maîtrise de Scrum et PHP dès le début du projet, débouchant sur une planification plus réaliste, d'autres fonctionnalités auraient pu être implémentées.

Parmi ces fonctionnalités, un système qui diviserait les recettes en ingrédients avec chacun leurs propres valeurs nutritives, avec un degré de précision plus haut pourrait être développé. Celui-ci serait plus utile que lorsque nous prenons les valeurs nutritives des recettes directement. Il en va de même pour une fonctionnalité de séparation des recettes par type ou catégorie, chose qui avait été mentionnée lors des premières réunions, mais a été mis de côté pour se focaliser sur d'autres aspects.

Une autre fonctionnalité qui aurait gagné à être mise en place est un système de génération de liste de courses par rapport à une recette qu'un utilisateur souhaiterait préparer et les ingrédients dont il dispose déjà. Ces ajouts auraient permis d'aller encore plus loin dans le sens de l'expérience utilisateur. Le site, bien que pensé pour être utilisé depuis un ordinateur, aurait aussi à bénéficier de l'ajout de media queries, pour le rendre utilisable même depuis d'autres sources.

Conclusion

Au début du projet, le manque d'expérience des membres de notre groupe quant à PHP, au modèle MVC et à la méthodologie Scrum était évident. Cette expérience nous a permis de plonger dans ces concepts de manière concrète, en acquérant une compréhension plus profonde et des compétences pratiques qui nous ont été bénéfiques pour mener à bien ce projet. De plus, la gestion de ce projet nous a également apporté une expérience précieuse en matière de planification, de collaboration et de coordination d'équipe.

Cette acquisition de compétences ne se limite pas seulement à ce projet, mais sera également utile à long terme dans notre parcours professionnel. Avoir cette expérience au début du projet aurait permis une meilleure planification, et certainement donc une version plus aboutie du livrable final. Nonobstant, la concrétisation du projet correspond aux attentes fixées. C'est donc avec fierté que le groupe 6 soumet sa version du projet « Manger ».

En approchant de la fin du projet, nous restons conscients des opportunités d'amélioration à explorer. L'expérience acquise et les leçons tirées ouvrent la voie à de futures itérations, où une planification plus affinée et l'intégration de fonctionnalités supplémentaires pourraient encore enrichir l'expérience utilisateur.