



Crok'eco

Projet de Programmation 2

BATATAY Mallory
KEGLO Partice

Informatique
Faculté des Sciences
Université de Montpellier

2024 - 2025



Résumé

Crok'eco est une application collaborative permettant à chacun de s'informer sur l'impact écologique d'un plat.

Remerciements

Nous tenons à exprimer notre sincère gratitude envers toutes les personnes qui ont contribué à la réussite de ce projet. Leur soutien et leur encadrement ont été d'une importance capitale tout au long de la création de cette application.

Nous souhaitons exprimer nos sincères remerciements à M. Eric Bourreau pour nous avoir permis de travailler sur ce sujet. Sa confiance et son soutien nous ont permis de le réaliser avec succès. Nous lui sommes reconnaissants pour cette opportunité qui nous a permis d'acquérir une expérience en génie logiciel.

Nous souhaitons tout particulièrement remercier Ewen PHILIPOT, pour nous avoir aider à initier le projet malgré qu'il soit parti au cours de cette année scolaire. Ses connaissances approfondies nous ont permis d'avoir une base solide pour l'application.

Nous tenons également à exprimer notre gratitude envers Mme Elisabeth BAERT, responsable de la licence 3 Informatique, pour nous avoir permis de réaliser ces projets durant le second semestre.

Enfin, nous aimeraisons remercier chaleureusement toutes les personnes qui nous ont aidés dans la création de l'application ainsi que dans la rédaction de ce rapport et qui ont relu ce dernier. Leur soutien et leurs précieux commentaires ont grandement amélioré la qualité de ce travail.

Table des matières

1	Introduction	4
2	Spécification du sujet	5
2.1	Enjeux climatiques	5
2.2	Approche du sujet	6
3	Gestion du projet	7
4	Choix technique	8
5	Architectures	9
5.1	Architectures de l'application	9
5.2	Modèle Dynamique	10
6	Application	11
6.1	Base de données	11
6.1.1	Recherche	11
6.1.2	Modélisation	12
6.1.3	Implémentation	13
6.2	Application utilisateur	14
6.2.1	Analyse d'un menu	14
6.2.2	Historique	17
6.2.3	Recherche	17
6.2.4	Ajout de plat et vote	18
6.2.5	Design	20
6.3	Backend	20
6.4	Fonctionnalités non implémenté	20
6.5	Statistiques	21
7	Conclusion	21
	Bibliographie	22
8	Annexes	23

1 Introduction

Nous réalisons ce projet dans le cadre de notre troisième année en Licence Informatique. Le projet a débuté en décembre 2024 et nous a accompagnés tout au long de notre 2e semestre. Le sujet que nous avons choisi est celui de Monsieur Bourreau. Le but du projet est de créer une application permettant de noter l'impact écologique avec une couleur. Celle-ci peut être de couleur Verte, Orange ou Rouge, respectivement d'une empreinte carbone faible à élevée, de la même manière que le Nutri-Score.



Ce projet a été réalisé par KEGLO Patrice, BARATAY Mallory et PHILIPOT Ewen. PHILIPOT Ewen ayant arrêté la Licence avant les vacances de février, nous avons réalisé la majeure partie du projet à deux.

2 Spécification du sujet

2.1 Enjeux climatiques

Selon l'INSEE, en 2018, l'alimentation a représenté 22% de l'empreinte carbone de la France [1]. Que ce soit lié au transport de la nourriture, à l'élevage des animaux, à la quantité d'eau utilisée tout au long de la production, il y a beaucoup de facteurs polluants pour amener la nourriture dans nos assiettes. Une majorité de la population n'est pas informée sur ce sujet et ne pense pas à l'impact que sa nourriture a sur la planète.

Des solutions existent déjà pour réduire l'émission de gaz à effet de serre liée à notre alimentation. En effet, tous les aliments ne se valent pas en matière d'impact environnemental. Certains aliments tels que la viande et le poisson ont un impact écologique bien plus important que les fruits, légumes ou céréales. Le fait de consommer des produits de saison et locaux permet aussi de réduire l'émission due à notre alimentation.

Entre 2009 et 2019, la quantité de viande consommée a baissé de 5 % en France métropolitaine selon l'INSEE [2]. On constate donc une volonté en France de consommer plus écologiquement, et la demande concernant des applications permettant de se renseigner devient de plus en plus grande. On note également une augmentation de projets allant dans ce sens, que ce soit avec la nourriture, mais également d'autres produits du quotidien comme les cosmétiques et les textiles, qui sont des domaines eux aussi très polluants.

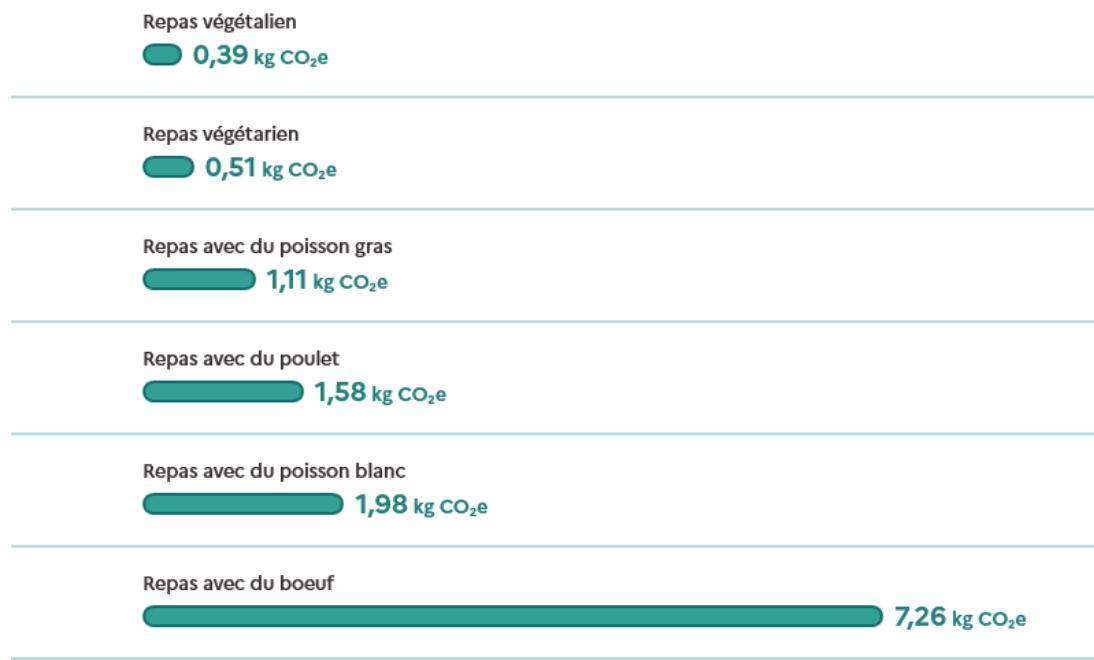


Image 1. – Emission de kg de CO₂ en fonction du type de repas

Source: ADEME [3]

2.2 Approche du sujet

Afin d'aider la population à faire des gestes écologiques, il est important de lui donner les outils permettant d'atteindre ses objectifs. La composition d'un plat étant la première cause de l'impact écologique de celui-ci, il serait intéressant de pouvoir être informé de l'empreinte écologique du plat que l'on souhaite manger. Afin de pallier ce problème, nous avons imaginé plusieurs solutions :

Afin de proposer une solution aux étudiants, dans un premier temps, il serait possible de s'accorder avec le Crous ou les gérants des restaurants de la Faculté des Sciences pour faire afficher sur les téléviseurs l'impact écologique de chaque plat. L'affichage pourrait se faire à l'aide d'une pastille de couleur correspondant au degré de pollution.

Sinon, nous pourrions afficher le résultat à l'aide d'un QR Code. Le QR Code redirigerait vers une page qui détaillerait l'empreinte carbone de chaque plat.

Ces deux solutions avaient déjà été envisagées l'année dernière, mais sans succès, car elle présente un problème. En effet, elle nécessite un accord avec les gérants des restaurants qui sont assez réticents à voir l'impact écologique de leurs plats révélé. De plus, même s'il avait été possible d'avoir l'accord pour les restaurants de la Faculté des Sciences, ces solutions demanderaient énormément d'efforts pour être déployées à l'échelle de la France, car elles nécessiteraient l'accord de chaque restaurant.

Nous avons donc choisi de faire une application collaborative qui calcule l'impact écologique de tous les plats en scannant leurs noms sur un menu. Le but de cette dernière méthode est de permettre à tout le monde de participer à améliorer l'écologie en rendant l'application collaborative. Cette méthode peut fonctionner dans n'importe quel restaurant traditionnel et peut être étendue aux selfs et aux cantines scolaires.

3 Gestión del proyecto

Le projet s'est déroulé en 4 grandes étapes, comme le décrit l'Image 2 ci-dessous, qui sont, la mise en place de la base de données, le développement des différentes fonctionnalités de l'application, la création du serveur distant ainsi qu'une base de données permettant de stocker et d'effectuer des mises à jour de données et enfin l'implémentation complète et finition du design de l'application ainsi que les tests de fonctionnalités.

Nous avons initié le projet en décembre 2024, en choisissant les différents outils que nous allions utiliser et en créant une maquette de l'application sur Figma.

La deuxième étape a été de préparer une première version la base de données de l'application avant de commencer à implémenter les différentes pages et les fonctionnalités de l'application.

Vers la fin du projet, nous avons ajouté un serveur afin que l'application ne fonctionne pas qu'en local, puis améliorer le design de l'interface. Pour finir, nous avons ajouté des tests à l'application.

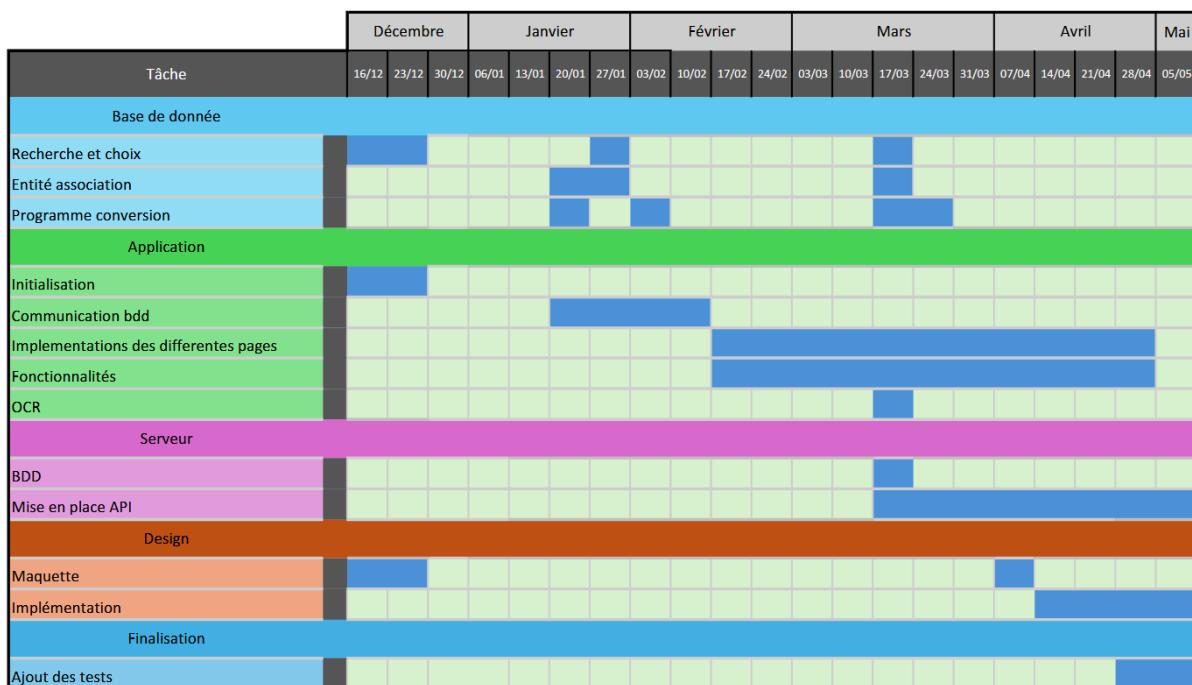


Image 2. – Diagramme de Gantt

4 Choix technique

Le développement de l'application a été réalisé avec Expo, un framework open-source basé sur React Native et TypeScript, un sur-ensemble typé de JavaScript, afin de bénéficier d'un typage statique, d'une meilleure lisibilité du code et d'une maintenance facilitée.

Le développement avec Expo nous permet d'avoir une application multi-plateformes (Android et iOS) ainsi qu'une version web.

Outils et bibliothèques principaux :

- Pandas(Python) pour la gestion de données en CSV
- Text-recognition de react-native-ml-kit , pour la reconnaissance de text sur une image
- Crypto-js pour encrypter les messages et assurer une sécurité dans le transfert de données entre notre serveur et l'application
- Netinfo de react-native-community pour mettre en place des mise à jour automatique de données entre l'application de notre serveur, en permettant de connaitre l'état de l'utilisateur (Connecté ou non Connecté)
- fastify pour mettre en place une api pour notre backend.
- Yup qui permet de vérifier les formulaires
- Jest pour tester les différentes fonctionnalités de notre application
- ainsi que d'autres librairies d'expo et de react-native.

L'application a été testé à la fois sur des simulateurs Android via Android Studio et sur des appareils physiques pour s'assurer d'une bonne compatibilité et d'une expérience utilisateur fluide.

Nous avons utilisé GitHub comme plateforme de gestion de version tout au long du projet afin de collaborer efficacement, et d'avoir un historique clair des modifications. Les branches ont été utilisées pour séparer le développement des différentes fonctionnalités, ce qui a facilité l'intégration progressive des différentes parties de l'application dans la branche principale (Branche MAIN).

Les données que nous utilisons pour notre base de données proviennent toutes du programme de collecte de données AGRIBALYSE 2.0.

5 Architectures

5.1 Architectures de l'application

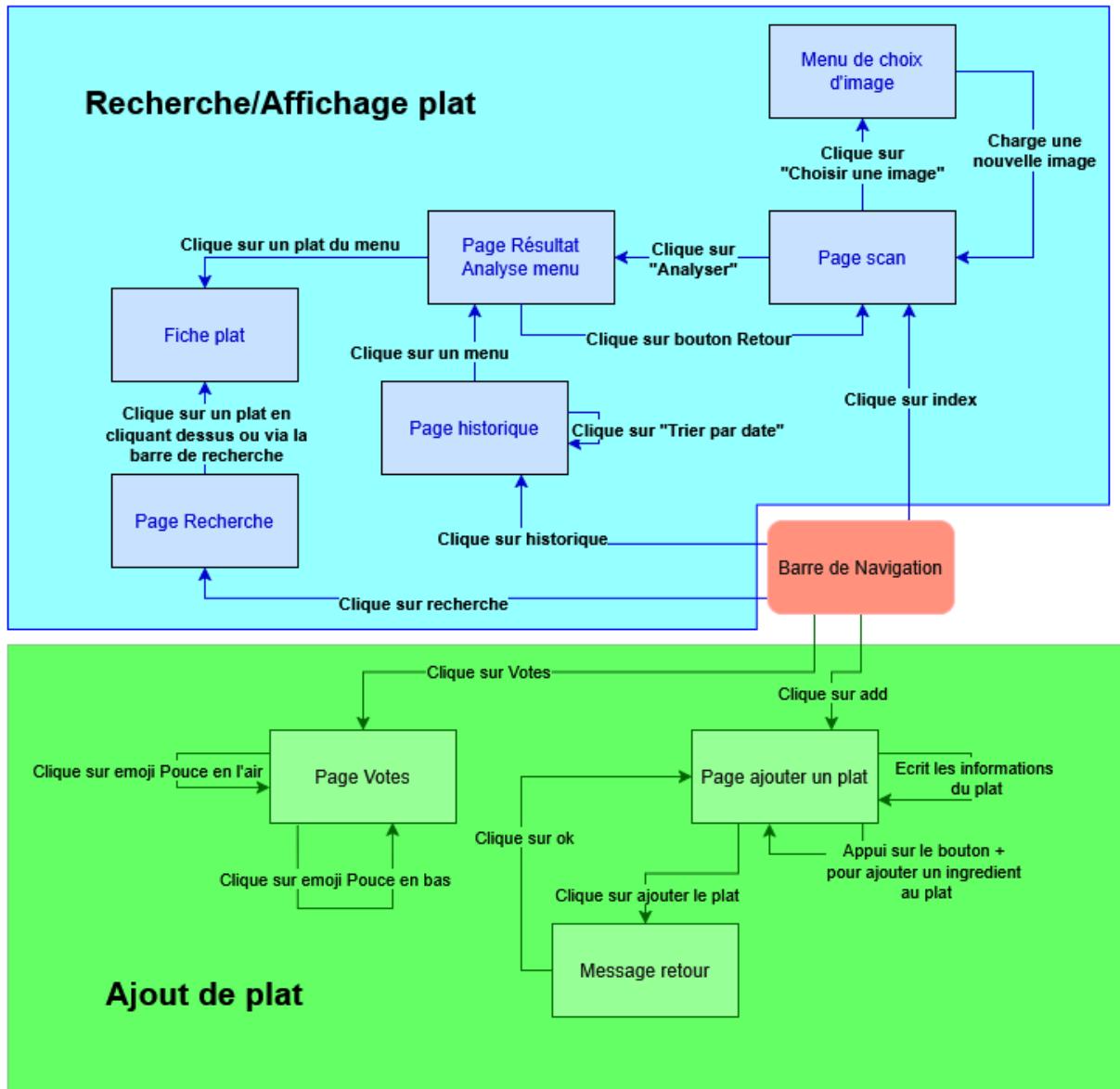


Image 3. – Architecture de l'application

L'Image 3 illustre l'architecture de notre application du point de vue de l'utilisateur. L'application s'organise autour de la barre de navigation qui permet d'atteindre les 5 pages principales (Index, Historique, Recherche, Votes et Add). Ces 5 pages peuvent être divisées en 2 catégories. Les pages permettant de rechercher et d'afficher la composition d'un plat et la partie permettant d'ajouter un plat à la base de données.

Dans la partie de recherche et d'affichage de plat, on retrouve dans un premier temps la page Index(- voir Chapitre 6.2.1). Cette page est la première qui s'ouvre lorsque l'on ouvre l'application. Elle a pour but de lancer l'analyse d'une image. Elle se compose de deux boutons. Le premier bouton, « Choisir une image », ouvre la médiathèque du téléphone pour que l'utilisateur sélectionne une image. Quand une image est choisie un menu s'ouvre pour proposer à l'utilisateur de modifier l'image en la rognant par exemple. Ensuite, l'image sélectionnée apparaît sur la page Index. L'utilisateur peut ensuite ap-

puyer sur le bouton « Analyser ». La page Historique permet de visualiser les différents menus que l'utilisateur a analysés et la page Recherche (voir Chapitre 6.2.3) permet de rechercher la composition et l'impact écologique d'un plat.

Dans la partie d'ajout de plat, la page Add permet d'ajouter un nouveau plat à la base de données en renseignant les différents ingrédients qui le composent (voir Chapitre 6.2.4) et la page Votes permet de dire si l'on pense qu'un plat ajouté par un utilisateur mérite d'être ajouté à la base de données final en cliquant sur un pouce en l'air ou en bas.

5.2 Modèle Dynamique

Lorsque l'utilisateur se sert de l'application, celle-ci communique avec le serveur afin de traiter les requêtes émises. Le diagramme de séquence (Image 4) ci-dessous illustre les échanges entre l'application et le serveur en fonction des actions réalisées par l'utilisateur.

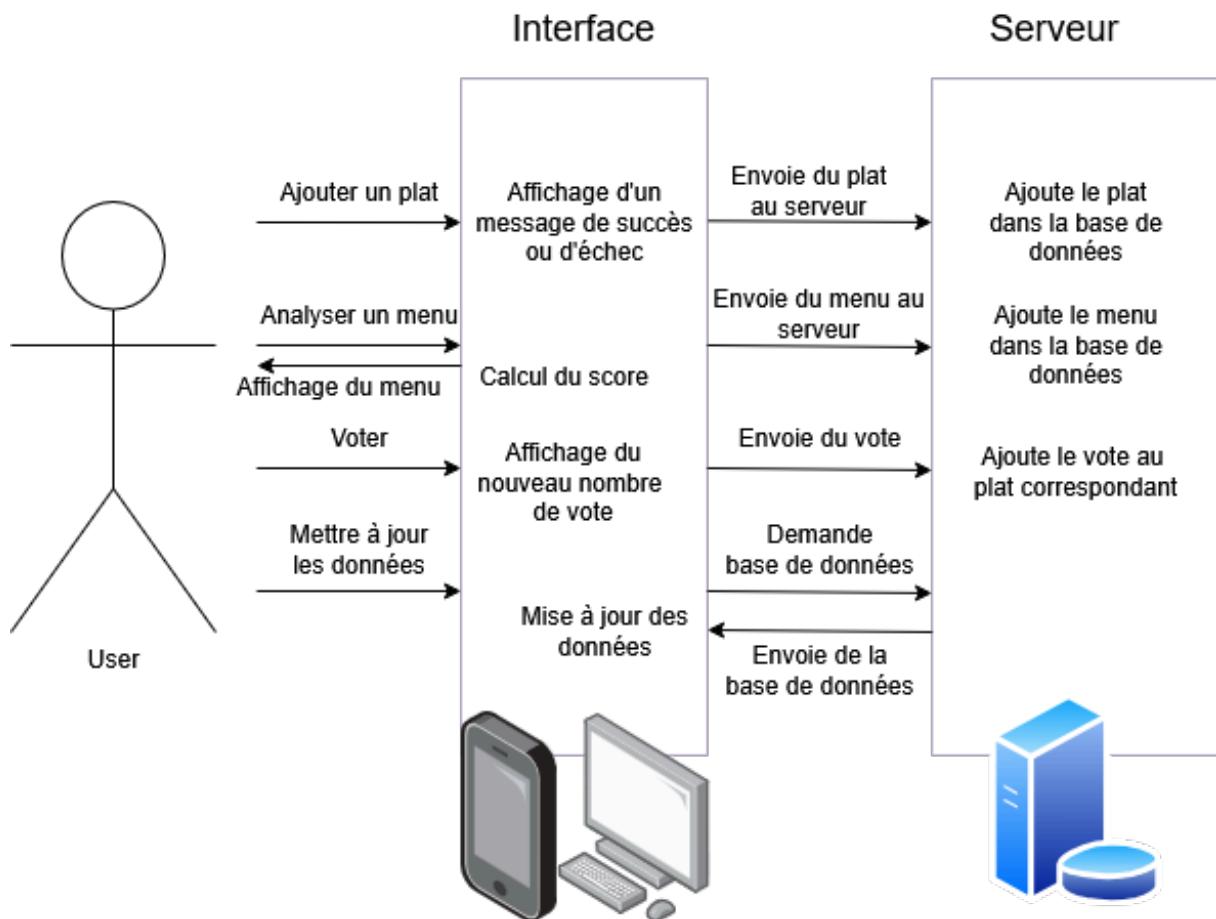


Image 4. – Diagramme de séquence

Les différentes requêtes présentes sur ce diagramme sont détaillés dans le Chapitre 6.2.

6 Application

6.1 Base de données

6.1.1 Recherche

Afin de connaître l'impact écologique d'un plat, nous avons choisi dans un premier temps de se servir de la base de données fournie par l'ADEME. La base de l'ADEME sur la consommation CO₂ est une immense base regroupant tous les types d'émission de gaz à effet de serre tel que celle due au textile, à l'industrie ou autres, ainsi que toutes les émissions liées à l'alimentation.



Image 5. – Logo AGRIBALYSE

En inspectant la base de données, nous avons remarqué que toutes les informations liées à la nourriture provenaient de deux bases de données qui sont AGRIBALYSE et AGRIBALYSE 2.0. AGRIBALYSE est un programme collectif et innovant qui met à disposition des données de référence sur les impacts environnementaux des produits agricoles et alimentaires à travers une base de données construite selon la méthodologie des Analyses du Cycle de Vie. Il est possible de se servir du site web d'AGRIBALYSE pour connaître l'impact environnemental d'un aliment ou bien de télécharger leur base de données.

Nous avons donc téléchargé la base de données concernant dans un premier temps uniquement les plats ayant nécessité une transformation. Cette base de données était disponible au format CSV. Afin de traiter, et de rendre les données utilisables, nous avons codé un programme python servant à initialiser une base de données en SQL comportant tous les plats décrit dans le CSV.

Le CSV était construit de la manière suivante : par plat présent dans la base, il y avait une ligne pour chaque ingrédient. Cela signifie qu'on retrouve l'impact écologique qu'a un ingrédient dans chaque plat le comportant, mais pas l'impact différé en fonction de la proportion de cet aliment dans le plat.

En analysant plus précisément nos besoins, nous avons remarqué un manque de plats trop important dans la base de données actuelle. Pour résoudre ce problème, nous avons changé une nouvelle fois de base de données, cette fois-ci comportant uniquement les ingrédients avec l'impact écologique associé par kilo d'aliment. Les données étaient fournies sous forme d'un fichier Excel comportant plusieurs onglets, dont un dédié uniquement aux ingrédients. À chaque ingrédient, est associé un code Ciqual [4] qui sert d'identifiant unique.

Pour connaître le poids de chaque aliment dans un plat et pour remplir la table SQL des plats nous avons choisi de faire confiance au utilisateur de l'application, même si ceci sera vérifié lors de sa validation. Une page de l'application permet d'enregistrer la composition d'un plat (voir Chapitre 6.2.4). Connaissant le poids de chaque aliment et son impact écologique, il devient alors possible de calculer l'impact environnemental de chaque ingrédient au sein d'un plat, mais également l'impact écologique global du plat lui-même. Pour cela, on doit donc réaliser la somme de l'impact de chaque ingrédient dans le plat et faire un produit en croix pour le ramener à un kilogramme de nourriture.

6.1.2 Modélisation

Pour l'application, nous avions besoin d'une base de données regroupant d'une part les informations du CSV fournit par AGRIBALYSE ainsi que les nouvelles données tel que les plats ou les menus.

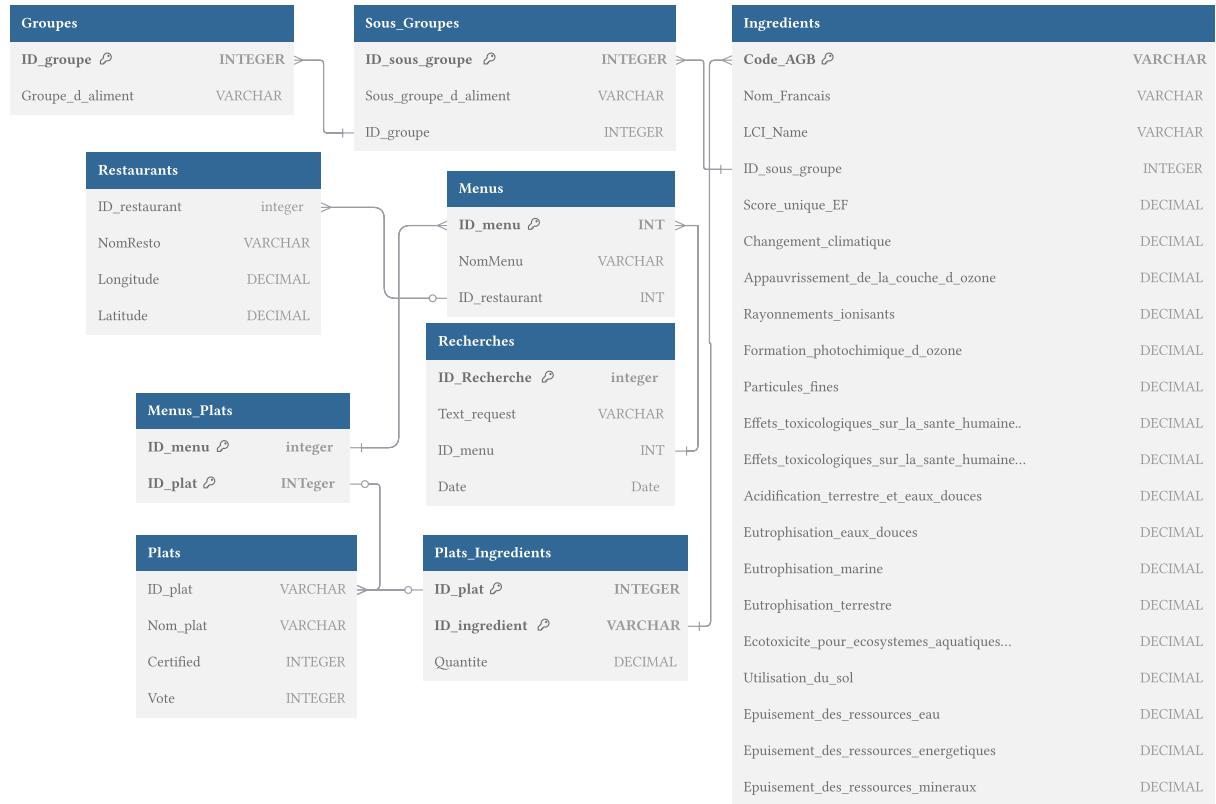


Image 6. – Schéma du modèle Entité Association de la base de données final

Nous avons donc défini un modèle Entité-Association (E-A) afin de représenter la structure de notre base de données. L'Image 6 met en évidence les différentes tables que comportent la base ainsi que les liaisons entre elles.

Les données que nous récupérons via le fichier Excel sont rangées dans la table ingrédients. Afin de trier les ingrédients par catégories, on retrouve les tables groupes et sous-groupes. Chaque ingrédient a une clé étrangère qui mène vers un sous-groupe lequel possédant à son tour une clé étrangère menant vers la table des groupes.

Ensuite, les plats sont composés de plusieurs ingrédients et un ingrédient peut être présent dans plusieurs plats. On a donc une table intermédiaire entre les plats et les ingrédients. Cette table est composée d'une clé étrangère menant vers l'ID des ingrédients.

Après avoir modélisé les deux bases de données, on rédige leur structure dans un fichier .sql qui servira pour leurs implémentations (voir Chapitre 6.1.3).

6.1.3 Implémentation

Afin de convertir les données d'AGRIBALYSE dans notre base de données décrites dans la partie précédente, nous avons choisi de coder un programme Python. Le but du programme est de créer à partir de zéro la base de données.

Pour cela, on commence par créer le fichier qui contiendra la base de données. Ce fichier aura une extension .db et s'il était déjà existant, il est remplacé. Ensuite, on importe le fichier .sql contenant l'agencement des tables décrit dans la partie Modélisation(voir Chapitre 6.1.2). Cela permet d'initialiser toutes les tables, mais elles sont encore vides.

Ensuite, il faut remplir trois tables avec les données fournies par AGRIBALYSE. Les tables à remplir sont les groupes, les sous-groupes et les ingrédients. Les données sont dans un tableau Excel comportant plusieurs pages. On commence donc par extraire la bonne page du Excel. Après cela, on commence le remplissage des trois tables par celle des groupes à cause des contraintes de clé étrangère. Pour remplir la deuxième table, parce qu'on a besoin de la clé primaire de la table des groupes qui viennent d'être ajoutée à la base de données, on doit d'abord faire une requête à la base de données. On range les indices et le nom du groupe dans un dictionnaire. Avec les informations du tableau Excel, on associe le nom des sous-groupes avec l'indice de son groupe correspondant. Enfin, on ajoute le dictionnaire résultant à la base de données. On recommence ensuite la même opération afin d'associer les indices des sous-groupes avec les ingrédients.

Ce programme permet à la fois de créer la base de données pour l'application de l'utilisateur et pour le serveur. La seule différence entre les deux tables et qu'on n'utilise pas le même fichier .sql.

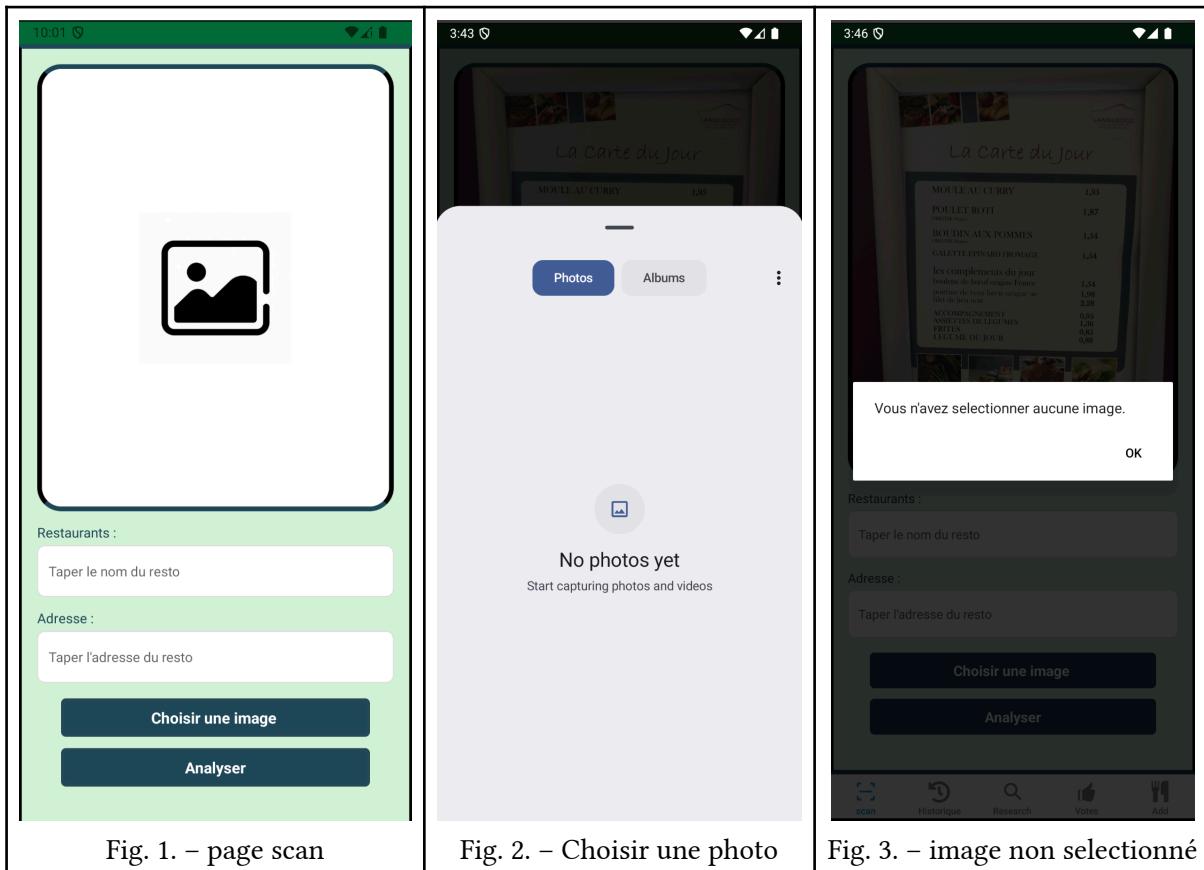
Avec ce programme Python, si la base de données d'AGRIBALYSE venait d'être mise à jour, il suffirait d'exécuter le programme pour obtenir la nouvelle base. Une amélioration possible pour ce programme serait de récupérer les autres données des anciennes bases pour les rajouter aux nouvelles.

6.2 Application utilisateur

Dans cette partie, nous allons décrire et expliquer les différentes parties ainsi que les différentes fonctionnalités implémentées tout en montrant leur fonctionnement avec des images.

6.2.1 Analyse d'un menu

Cette fonctionnalité est visible sur la première page de notre application, elle consiste, dans un premier temps, à reconnaître le texte contenu sur une image choisie par l'utilisateur depuis sa galerie décrit par la figure Fig. 2. Pour cela l'utilisateur appuis sur le bouton **Choisir une image**. Par défaut, il y a une image (voir Fig. 1). Les images de sa galerie s'affichent puis il clique sur celle qu'il veut analyser. Si une fois les images de la galerie affichées, il (l'utilisateur) ne clique sur aucune image et referme l'affichage, un message d'alerte s'affiche indiquant qu'aucune image n'a été sélectionnée (voir Fig. 3), sinon l'image est mise à jour (voir Fig. 4).



L'utilisateur peut rentrer le nom du restaurant dont il va analyser le menu ainsi que l'adresse s'il l'a. Ces deux informations ne sont pas obligatoires.

Une fois ceci fait, il peut cliquer sur le bouton **Analyser**. On effectue l'analyse textuelle de l'image et on récupère le nom des plats du menu ou du moins le texte reconnu. Une fois les noms récupérés, on vérifie pour chaque plat, s'il existe dans notre base de données, pour pouvoir ensuite récupérer la liste des ingrédients, leurs quantités dans le plat et effectuer le calcul de score. En fonction du score du plat, on lui attribue une couleur qui sera la même que celle de la pastille qui sera affichée pour ce plat, comme l'indique le Tableau 8.

Le score d'un plat est la somme des Score unique EF [5] (EF = Environmental Footprint) = Somme (impact catégorie * facteur de pondération) de chacun de ses ingrédients.

Intervale de score	Couleur	Niveau d'Impact
score ≥ 0 et score ≤ 1	Verte	Faible
score > 1 et score ≤ 5	Orange	Moyen
score > 5	Rouge	Elevé
Plat non reconnu	Noire	-

Tableau 8. – Tableau descriptif de l'attribution des couleurs

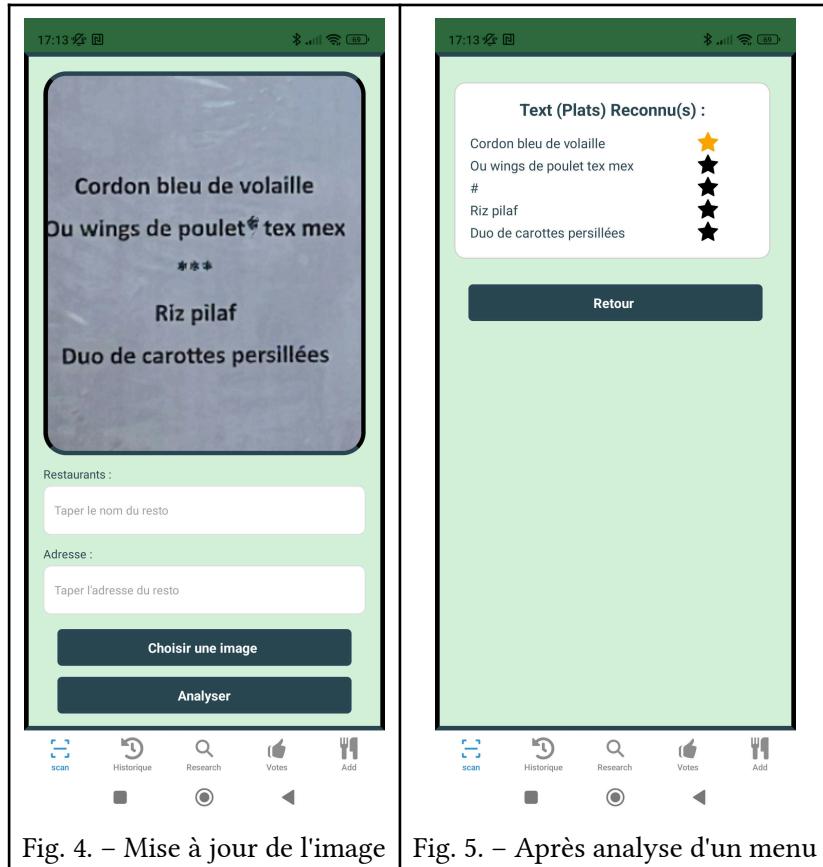


Tableau 9. – Update image et resultat de l'analyse

Avant l'affichage du texte ou des plats reconnu(s), les informations renseignées sont stockées dans la base de données locale de l'application. Ces données sont aussi envoyées au serveur si l'utilisateur est connecté à internet et donc au serveur ou attend qu'il le soit pour les envoyées. Cette fonctionnalité est implementée par l'algorithme Fig. 21 (voir Annexe Chapitre 8).

Après l'analyse, on affiche la liste des plats avec une pastille en forme d'étoile juste devant le nom ayant une couleur descriptif de l'impact écologique du plat comme l'indique la Fig. 5. Quand un plat n'existe pas dans notre base de données, l'étoile est de couleur noire.

On peut cliquer sur chaque ligne ou plat. Et si le plat existe dans notre base de données, on est reconduit sur une autre page affichant les informations du plat ainsi que les différents ingrédients qui le composent avec leur pourcentage comme décrit la Fig. 6.

Si le plat n'existe pas, alors le message : « Ce plat n'existe pas dans nos données. Vous pouvez l'ajouter en allant sur la page d'ajout » dont nous parlons plus tard dans le Chapitre 6.2.4, est affiché (voir Fig. 7).

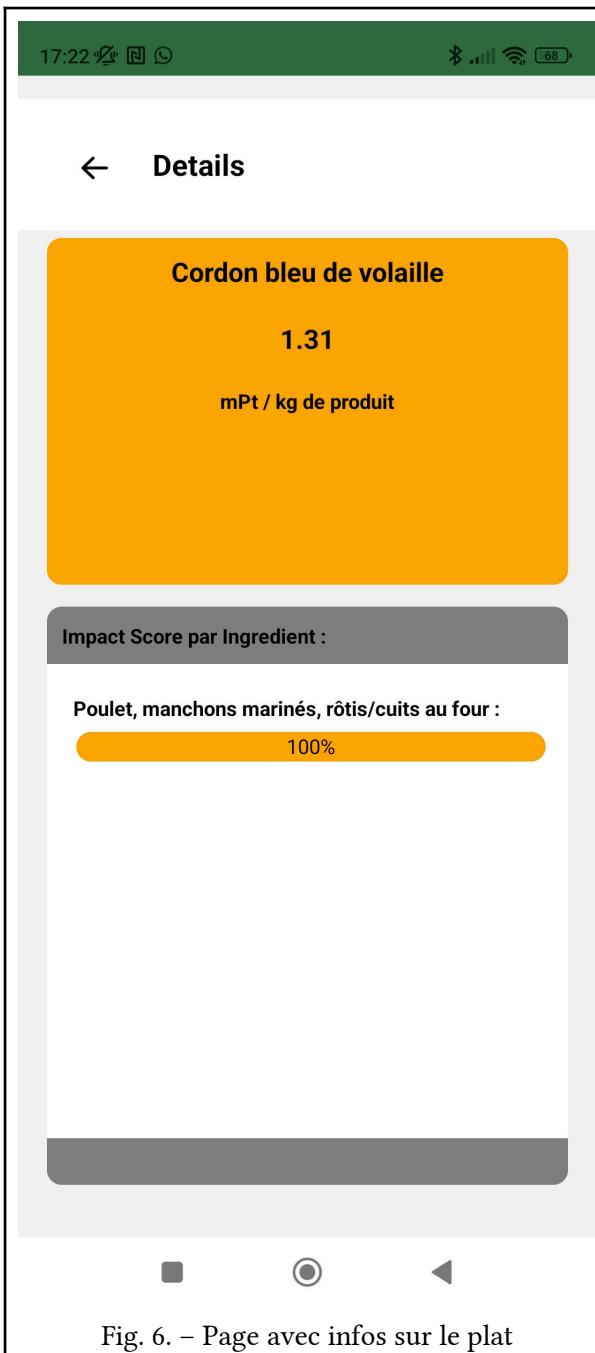


Fig. 6. – Page avec infos sur le plat

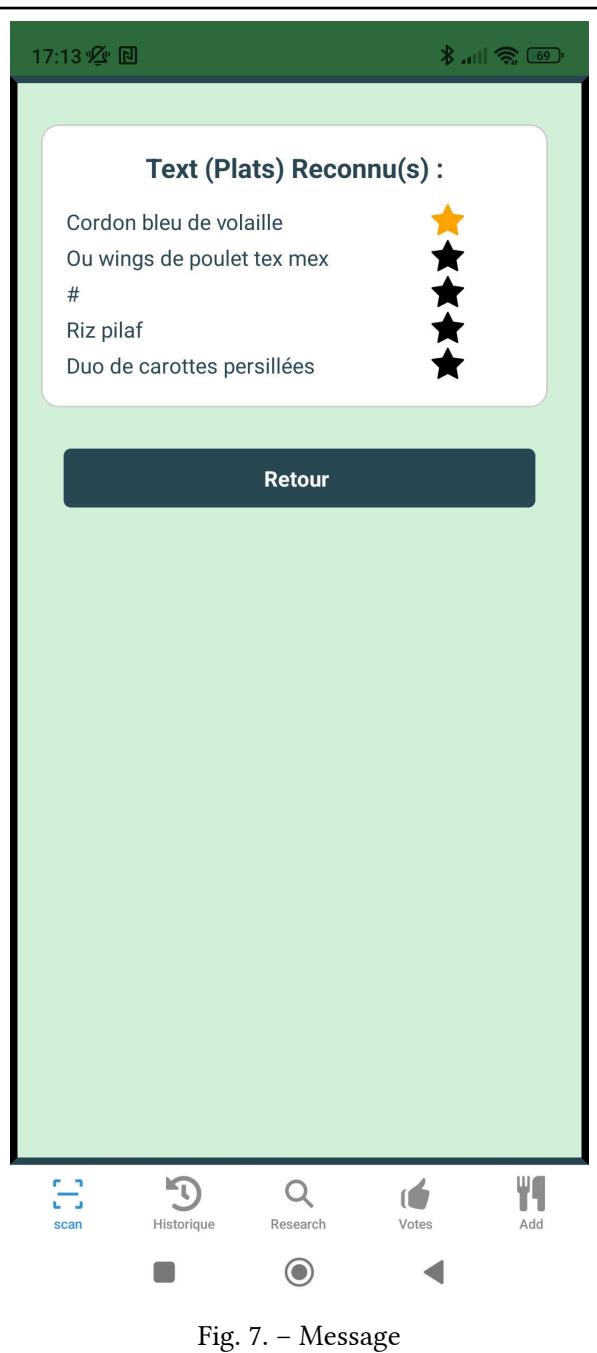


Fig. 7. – Message

Tableau 10. – Page de details d'un plat et alert si le plat n'existe pas dans la base de données

6.2.2 Historique

Sur cette page, on retrouve tous les menus que l'utilisateur a scanné. Pour cela on récupère stockées dans la base de données locale de l'application (voir Fig. 8). On peut aussi les trier par date, du plus anciens au plus récents ou et vice versa (voir Fig. 10 et Fig. 9). En cliquant sur un menu, on est redirigé vers la page de détails du plat (voir Fig. 6).

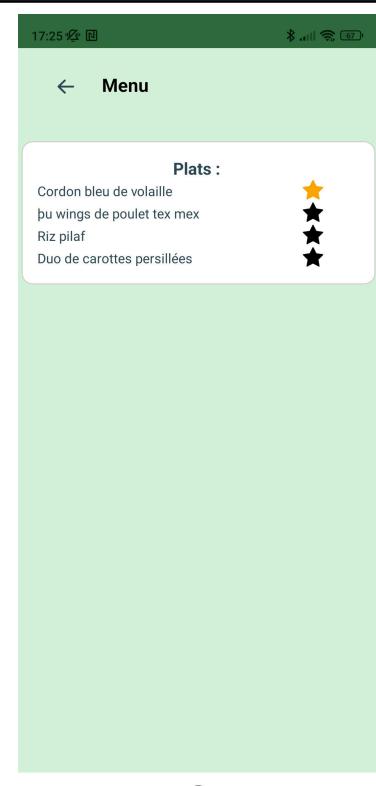
 <p>Fig. 8. – Un menu de la page Historique</p>	 <p>Fig. 9. – Historique trié par date plus récents</p>	 <p>Fig. 10. – Trié par date plus anciens</p>
---	---	--

Tableau 11. – Historique des plats scannés

6.2.3 Recherche

Cette page est constituée de 3 parties.

La première partie est un champ de recherche qui permet de rechercher un plat dans la base de données. Il suffit de commencer par taper le nom du plat et une liste déroulante s'affiche avec les plats existants dans notre base de données et qui correspondent à la recherche (voir Fig. 11). En cliquant sur un plat, on est redirigé vers la page de détails du plat (voir Fig. 6).

La deuxième partie (Favoris) est une liste de plats existant dans notre base de données et qui sont les plus recherchés par les utilisateurs (voir Fig. 12). En cliquant sur un plat, on est redirigé vers la page de détails du plat (voir Fig. 6).

La troisième partie (Suggestions) est une liste de plats existant dans notre base de données et qui sont les moins polluants (voir Fig. 12). En cliquant sur un plat, on est redirigé vers la page de détails du plat (voir Fig. 6).

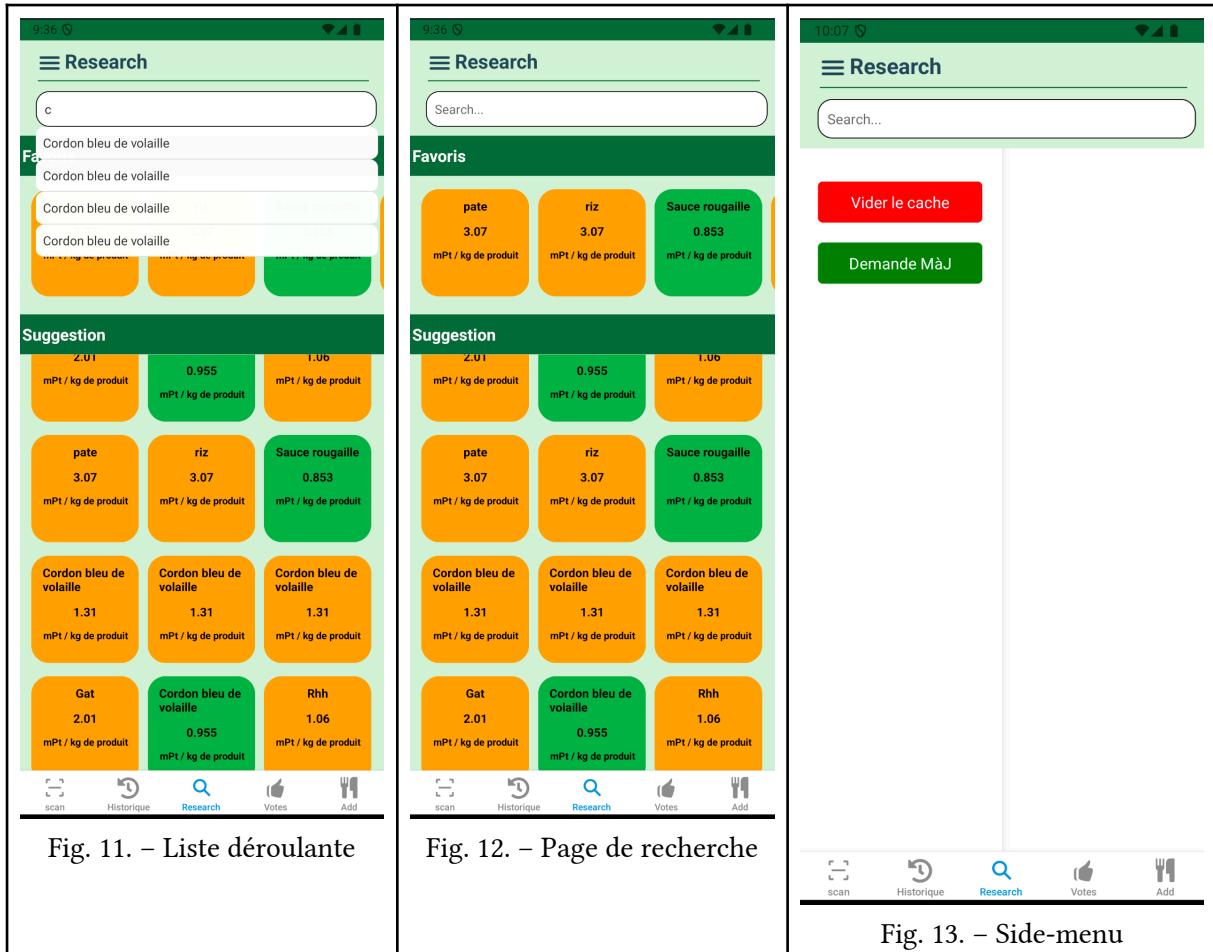


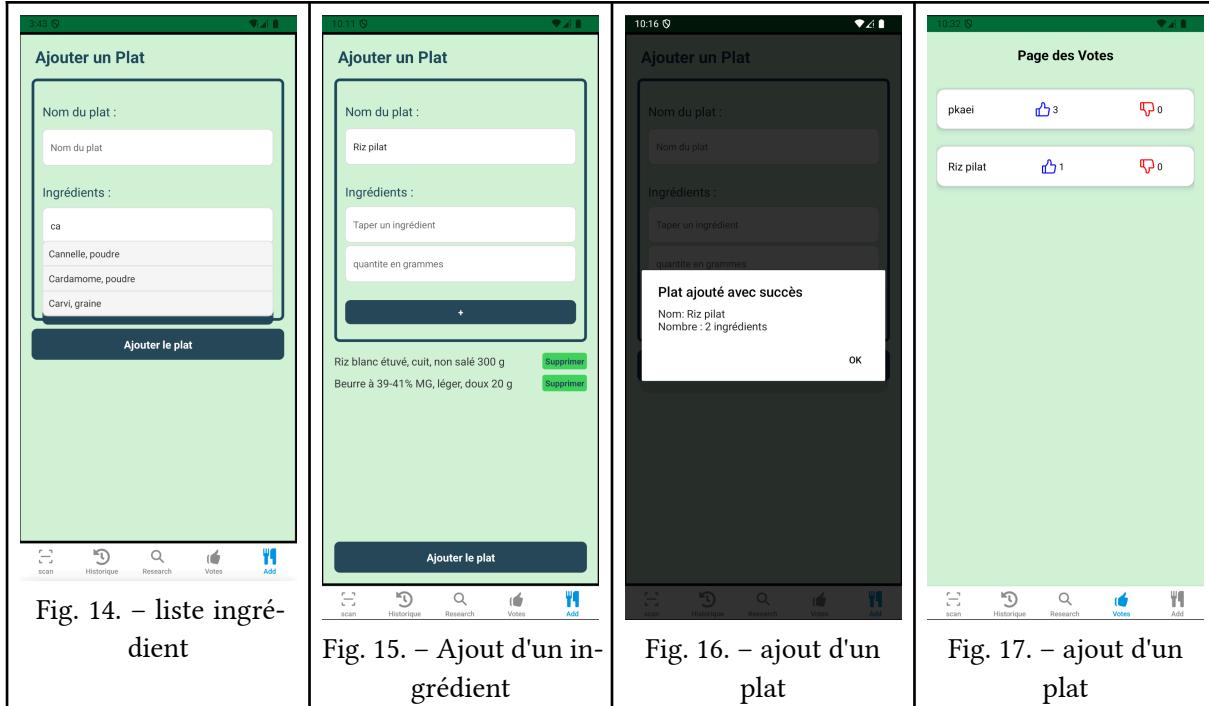
Tableau 12. – Page de recherche et liste déroulante et side-menu

On a aussi tout en haut à gauche de la page, un bouton de menu qui permet d'afficher un side-menu. Ce menu permet d'avoir accès deux fonctionnalités de l'application. La première est la possibilité de vider le cache de l'application grâce au bouton **Vider le cache**. En effet, l'application stocke les données permettant de faire l'affiche de cette page, dans le cache de l'appareil, pour permettre la fluidité de l'affichage et d'éviter de nombreuses requêtes à la base de données. Il est donc possible de vider le cache pour libérer de la place sur l'appareil. La deuxième fonctionnalité est la possibilité de faire une demande de mise à jour des données de l'application manuellement grâce à un bouton **Demande de M&J**, même si la mise à jour se fait automatiquement une fois par jour, lorsque l'utilisateur est connecté à internet. En cliquant sur ce bouton, on envoie une requête au serveur pour lui demander de mettre à jour les données de l'application. Le serveur va alors envoyer une réponse à l'application avec les données qui manquent s'il y en a. Sinon, juste une réponse pour dire que tout est à jour. Cette fonctionnalité est implémentée par l'algorithme Fig. 22, voir dans la partie Annexe (Chapitre 8).

6.2.4 Ajout de plat et vote

Cette page permet à l'utilisateur d'ajouter un plat à la base de données. Pour cela, il doit remplir un formulaire avec le nom du plat, chaque ingrédient et la quantité de chaque ingrédient dans le plat. Pour ajouter un ingrédient, il suffit de commencer par entrer le nom de l'ingrédient et une liste déroulante s'affiche avec des ingrédients de notre base de données, qui correspondent au nom entré (voir Fig. 14) et il choisit. Ensuite, il doit entrer la quantité de l'ingrédient dans le plat. Une fois cela fait, il doit appuyer sur le bouton + pour ajouter l'ingrédient à la liste des ingrédients du plat. Cette opération est répétable pour chaque ingrédient du plat.

Chaque ingrédient ajouté est affiché avec son nom et sa quantité dans le plat, sur la page. On a la possibilité de supprimer un ingrédient en cliquant sur le bouton **Supprimer** à droite de chaque ingrédient de la liste (voir Fig. 15).



Une fois le formulaire rempli, il peut cliquer sur le bouton **Ajouter** pour envoyer les données au serveur. Avant l'envoie des données, on vérifie la connection au serveur en faisant un ping. Et si on est connecté, les données sont envoyées et on affiche un message avec le nom du plat et le nombre d'ingrédient ajouté (voir Fig. 16). Sinon, on affiche un message d'erreur : « Erreur de connexion : Veuillez vérifier votre connexion au serveur et reessayer ».

Les plats ajoutés et envoyés au serveur sont stockés sur le serveur et sont visibles par tous les utilisateurs de l'application en allant sur la page vote. Ils peuvent aussi voter positivement pour le plat en cliquant sur le pouce bleu ou voter négativement en cliquant sur le pouce rouge (voir Fig. 17). Chaque utilisateur peut voter pour un plat une seule fois. Et il peut voir le nombre de vote qu'a eu un plat (il est affiché). Si l'utilisateur a déjà voté pour le plat, il ne peut plus voter. Le vote est stocké dans la base de données du serveur.

En cliquant sur le nom du plat, on est redirigé vers la page de détails du plat comme pour les plats depuis la page research (voir Fig. 6).

Les plats ajoutés par les utilisateurs pourront être ajoutés à la base de données de l'application si le plat est validé par un administrateur depuis le serveur, de la table des plats ajoutés par les utilisateurs vers la tables de plats definitif de l'application. Une fois cette action effectuée, lors de la mise à jour journalière, la base de données de l'application sera mise à jour avec les nouveaux plats.

6.2.5 Design

Afin de réfléchir au fonctionnalité de l'application et de son design général, nous avons commencé le projet en créant des croquis de l'application sur Figma. Cela nous a permis de nous orienter tout au long du projet en se mettant d'accord sur la base de celui-ci. Sur Figma, nous pouvions créer des design de page, ajouter des boutons pour passer d'une page à l'autre.



Tableau 14. – Design établi sur Figma au début du projet

6.3 Backend

Pour le backend, nous avons choisi d'utiliser **Fastify**, un framework Node.js qui permet de créer des applications web et des API rapidement et facilement. Fastify est connu pour sa rapidité et sa simplicité d'utilisation, ce qui en fait un excellent choix pour notre projet. Nous avons mis en place un serveur qui gère les requêtes de l'application mobile. Le serveur est responsable de la gestion des données, de la communication avec la base de données et de l'envoi des réponses à l'application. Il utilise une base de données SQLite pour stocker les informations sur les plats, les utilisateurs et les votes. Les plats ajoutés et votés par les utilisateurs sont visibles sur la page ajout de la backend. Le(s) administrateur(s) peuvent valider ou supprimer les plats ajoutés par les utilisateurs. Les plats validés sont ajoutés à la base de données de l'application mobile lors de la mise à jour journalière. Le serveur de l'application est hébergé, ce qui rend l'utilisation de l'application utilisable par tous le monde.

6.4 Fonctionnalités non implémenté

Bien que l'application soit fonctionnelle, plusieurs fonctionnalités n'ont pas pu être implémentées. Cela principalement par manque de temps, ou par lacune technique. Nous allons présenter les pistes les plus intéressantes sur lesquelles nous pourrions travailler pour une future version de l'application.

Dans la première page du menu, nous avions envisagé que l'utilisateur puisse choisir entre prendre une photo directement dans l'application ou la sélectionner dans la galerie de son appareil. Finalement, nous n'avons gardé que la deuxième solution pour simplifier l'application durant son développement.

L'APK de l'application n'est disponible que sur Android. Nous avions prévu de le rendre disponible sur iOS, mais le processus de création d'un APK pour iOS est plus complexe que pour Android et après plusieurs tentatives de build infructueuses, nous avons décidé de ne le laisser en suspend. Il serait donc intéressant de le rendre disponible sur iOS pour la prochaine version de l'application.

Ensuite, il serait vraiment intéressant que les utilisateurs puissent se créer un compte. Cet ajout serait très important avant la publication de l'application pour pouvoir modérer les ajouts de plat ou pour limiter le nombre de votes par plat par utilisateur. Les autres fonctionnalités seraient toujours accessibles sans se créer de compte.

Enfin, nous avions pensé à géolocaliser les restaurants dont les utilisateurs ont scanné un menu. Avec cela, on pourrait chercher sur une carte les restaurants en fonction de l'impact écologique des plats qu'il propose. Nous avons déjà ajouté les attributs dans les tables pour que cela puisse être mis en place, mais n'avons pas fini cette partie de l'application.

6.5 Statistiques

Pour faire un bilan de notre application, elle se compose de 48 fichiers rédigés par nos soins, répartis de cette manière :

- 34 fichiers pour l'application frontend,
- 10 fichiers pour le serveur backend,
- 4 fichiers pour le code externe (SQL et Python).

En tout, nous avons écrit **5 432 lignes de code** dans ce projet. Cela inclut les fichiers TypeScript, JavaScript, HTML, CSS, Python et SQL. Ce volume reflète l'ampleur du travail accompli pour développer une application complète et fonctionnelle.

7 Conclusion

Pour conclure, l'application Crok'eco répond efficacement aux objectifs fixés en permettant aux utilisateurs de s'informer sur l'impact écologique des plats qu'ils consomment. Grâce à son approche collaborative et à l'utilisation de bases de données fiables comme AGRIBALYSE, elle offre une solution innovante et accessible pour sensibiliser le public à l'empreinte carbone de leur alimentation.

Cependant, certaines fonctionnalités prévues n'ont pas pu être implémentées, comme l'intégration de la caméra ou la gestion des comptes utilisateurs. Ces éléments représentent des pistes d'amélioration pour les futures versions de l'application. Malgré ces limitations, le projet a permis de développer des compétences techniques variées, allant de la gestion de bases de données à l'utilisation de frameworks modernes comme Expo.

Enfin, ce projet nous a permis de mettre en pratique nos connaissances acquises en Licence Informatique mais aussi de développer de nouvelles compétences essentielles pour la suite de notre parcours.

Bibliographie

- [1] INSEE, « Empreinte carbone de l'alimentation ».. <https://www.insee.fr/fr/statistiques/7728883?sommaire=7728903>
- [2] INSEE, « Évolution de la consommation des ménages selon les territoires de 2009 à 2019 ».. <https://www.insee.fr/fr/statistiques/7728873?sommaire=7728903>
- [3] ADEME, « Base Empreinte ».. <https://base-empreinte.ademe.fr/>
- [4] CIQUAL, « Table de composition nutritionnelle des aliments ».. <https://ciqual.anses.fr/>
- [5] Commission, E., « Environmental Footprint Methods ».. <https://ec.europa.eu/environment/eussd/smgp/index.htm>

Tables des figures

Image 1: Emission de kg de CO2 en fonction du type de rapas	
Source: ADEME [3]	5
Image 2: Diagramme de Gantt	7
Image 3: Architecture de l'application	9
Image 4: Diagramme de séquence	10
Image 5: Logo AGRIBALYSE	11
Image 6: Schéma du modèle Entité Association de la base de données final	12
Tableau 7: Analyse menu	14
Tableau 8: Tableau descriptif de l'attribution des couleurs	15
Tableau 9: Update image et resultat de l'analyse	15
Tableau 10: Page de details d'un plat et alert si le plat n'existe pas dans la base de données	16
Tableau 11: Historique des plats scannés	17
Tableau 12: Page de recherche et liste déroulante et side-menu	18
Tableau 13: Liste des ingrédients, ajout d'un ingrédient et d'un plat + vote	19
Tableau 14: Design établi sur Figma au début du projet	20

8 Annexes

```
export default function index() {
    // Charger les données de localisation, Insert les données en local ensuite les envoyer au serveur
    async function LoadLocAndInsertClient_SendDataToServeur(lines: any[]) {
        let Longitude = 0; let Latitude = 0;
        if (!loc) { await getLocation(); }
        if (location) {
            Longitude = location.coords.longitude;
            Latitude = location.coords.latitude;
            const resto = { NomResto: nomResto, Latitude, Longitude, Adresse, };
            // Insertion des données dans la base de données locale
            const idresto = await addRestaurants_Historique(resto);

            const menu = { NomMenu: nomResto, ID_restaurant: idresto };
            const res = await addMenus_Historique(menu);

            const textReconnu = JSON.stringify(lines);
            const recherche = {
                Text_request: textReconnu,
                ID_menu: res,
                Date: new Date().toLocaleDateString("fr-FR"),
            };
            await addRecherches_Historique(recherche);

            setNomResto("");
            const res1 = await DoSomethingWhenServerReady(resto, PostResto);
            const menuPost = { NomMenu: nomResto, ID_restaurant: res1.code };
            const res2 = await DoSomethingWhenServerReady(menuPost, PostMenu);
            const recherchePost = {
                Text_request: textReconnu,
                ID_menu: res2.code,
                Date: new Date().toLocaleDateString("fr-FR"),
            };
            // Envoi des données au serveur si la connexion est ok sinon on attend la reconnexion puis l'env
            await DoSomethingWhenServerReady(recherchePost, PostRecherche);
        }
    }
}
```

Fig. 21. – Algorithme d'envoie de données avec attente si l'utilisateur n'est pas connecté

```

    /**
     * Fonction qui vérifie si une mise à jour est disponible
     * et effectue la mise à jour si nécessaire
     * @returns {Promise<void>}
     */
    async function CheckForUpdates2(): Promise<void> {
        console.log("Demande de maj");
        const ele = await getLastElementPlats();
        if (ele != null) {
            const data = { ID_plat: ele?.ID_plat };
            const laMaj = await PostUpdateRequest(data);
            if (laMaj.code == DO_MAJ_CODE) await DoUpdates(laMaj);
            else {
                console.log("Pas de mise à jour disponible");
                await loadData();
            } else {
                console.log(`Y a pas de plats dans la BD !`);      You, 1 second ago • Uncommitted changes
            }
        }
    }

    /**
     * Fonction qui vérifie si une mise à jour est disponible
     */
    async function CheckForUpdates() {
        await checkForDailyUpdate(CheckForUpdates2);
    }

```

Fig. 22. – Algorithme de demande de la mise à jour

```

    /**
     *
     * @param data : {message: string; code: number | string; last: string;}
     * @description Fonction qui effectue la mise à jour de la base de données
     * en fonction des données reçues
     * @returns {Promise<void>}
     */
    async function DoUpdates(data: {message: string; code: number | string; last: string;}) {
        //console.log("Lance la MAJ");
        if (data.code === DO_MAJ_CODE) {
            const objet = JSON.parse(data.message);
            const plats: any[] = objet.plats;
            const plats_ingredients: any[] = objet.plats_ingredients;
            // Inserer les plats dans la bd de l'appli
            for (const plat of plats) {
                await addPlats(plat);
            }
            // Inserer les associations plat--ingredients
            for (const plat_ingredient of plats_ingredients) {
                await addPlats_Ingredients(plat_ingredient);
            }

            const last_plat = await getLastElementPlats();
            if (last_plat.ID_plat != data.last) {
                console.log("Pb: Maj pas complète");
            }
            await clearAllCache();
            console.log("Mise à jour terminée !!!");
        } else {
            console.log(data.message);
        }
    }

```

Fig. 23. – Algorithme de réalisation de la mise à jour automatique