Documentation technique

Green IA

MBAYE Ahmadou Bamba – PICHARD Quentin - CHARLEMAGNE Clément

Projet : 20240913_green_ia

Sommaire :

Table des figures :

I. Architecture du système et présentation du projet

1. Présentation du projet

Contexte

Green IA est née d'un constat simple : l'accès à l'information concernant l'empreinte carbone des produits alimentaires est difficile. Lorsque d'autre applications permettent d'informer l'utilisateur sur l'impact d'un produit sur sa santé, la nôtre se concentrera sur l'impact environnemental d'un produit alimentaire.

Objectifs

L'objectif de Green IA est de devenir une application indispensable pour les consommateurs souhaitant améliorer leurs habitudes de consommations pour réduire leur emprunte carbone. Green IA les accompagnera dans leur quête en leur donnant accès aux données relatives à l'impact écologique de leurs produits alimentaires. Ses objectifs principaux sont :

- Élever la conscience publique sur l'empreinte des produits alimentaires et rendre accessible les données à tous.
- Fournir des données fiables et transparentes concernant les produits alimentaires et leurs empreintes carbone, consommation d'eau et plus encore.
- Proposer des options plus écologiques aux produits trop polluants.
- Mesurer et suivre la réduction de l'empreinte hebdomadaire ou mensuelle de l'utilisateur.
- Construire une plate-forme non seulement d'information mais aussi de sensibilisation et d'encouragement.

2. Qu'est-ce que l'éco-score ?

L'Eco-score est un indicateur expérimental représentant l'impact environnemental des produits alimentaires. Il classe les produits en 5 catégories (A, B, C, D, E), de l'impact le plus faible, à l'impact le plus élevé. L'impact environnemental tient compte de plusieurs facteurs, dont en voici la liste exhaustive :

- Émissions de gaz à effet de serre (CO2)
- Destruction de la couche d'ozone
- Émissions de particules fines
- Oxydation photochimique
- Acidification
- Radioactivité
- Épuisement des ressources en eau
- Pollution de l'eau douce
- Épuisement des ressources non renouvelables
- Eutrophisation (terrestre, eau douce & marine)
- Utilisation des terres
- Toxicités (eau douce & humaine)
- Perte de biodiversité

Les impacts sont étudiés tout au long du cycle de vie du produit

Figure 1, cycle de vie d'un produit alimentaire

3. Vue d'ensemble

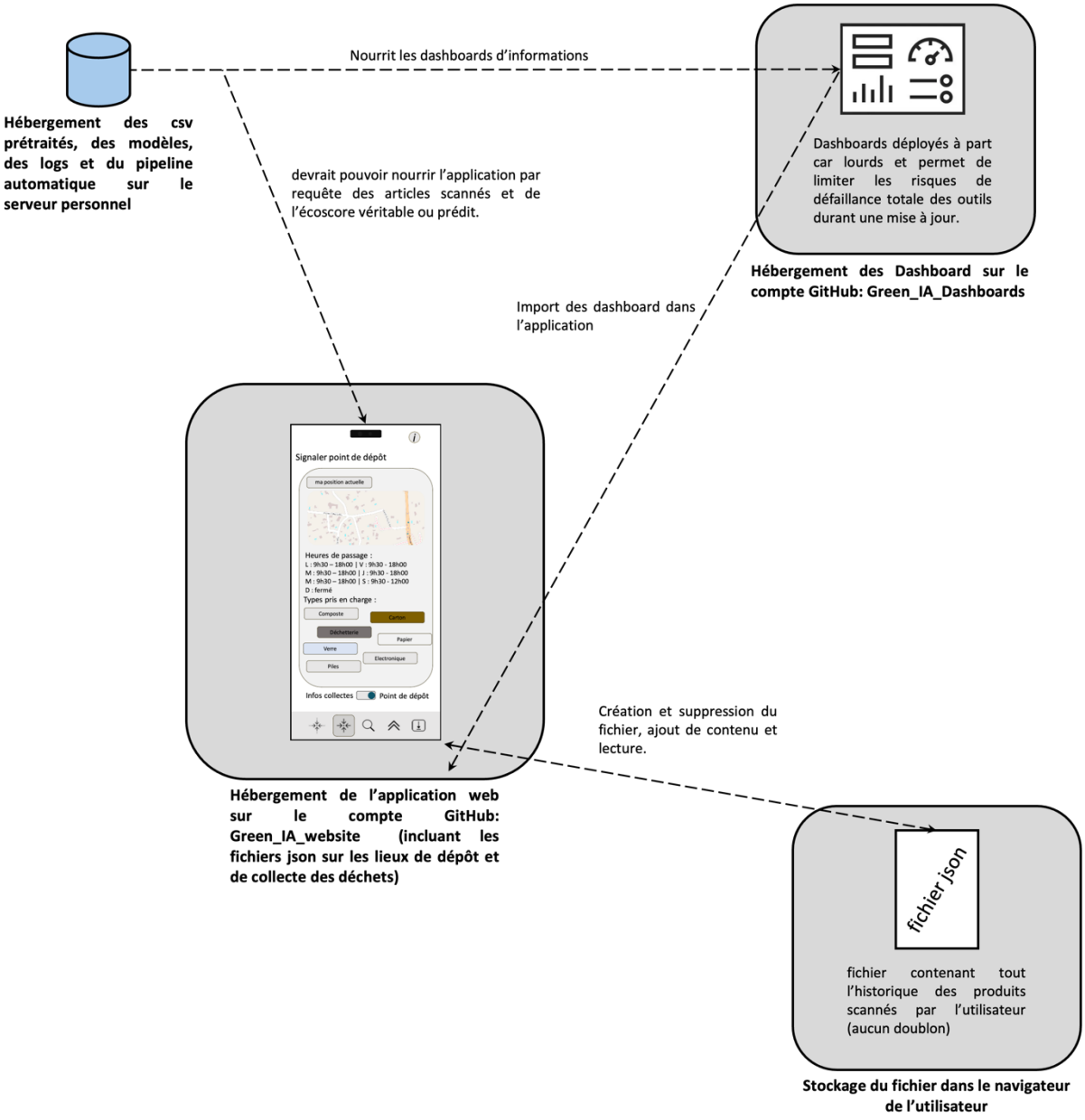

Figure 2, vue d'ensemble du projet

4. Use case des fonctionnalités principales et séquence d'actions

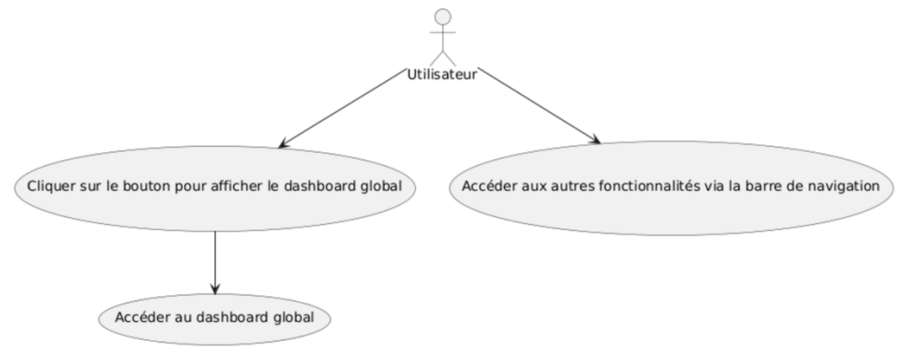

Figure 3, use case page dashboard utilisateur

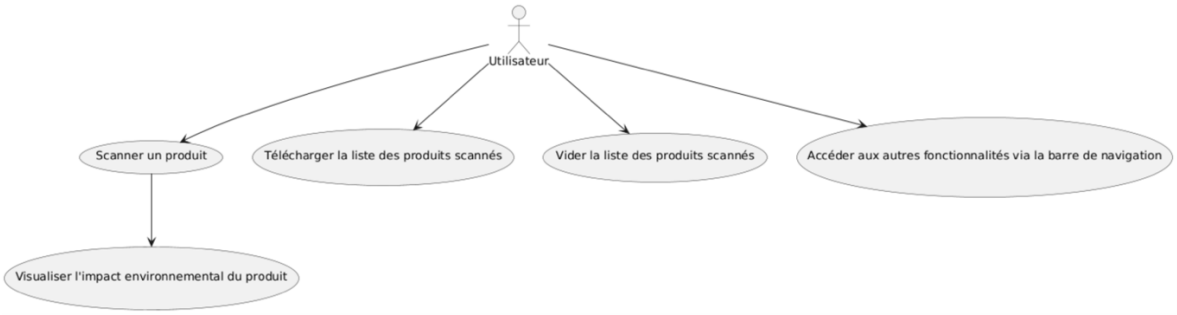

Figure 4, use case page principale de scanne

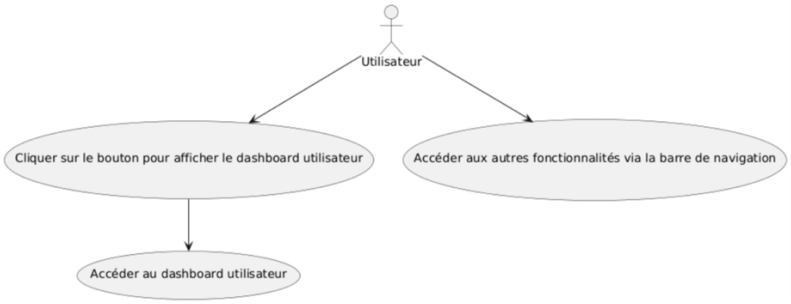

Figure 5, use case page dashboard utilisateur

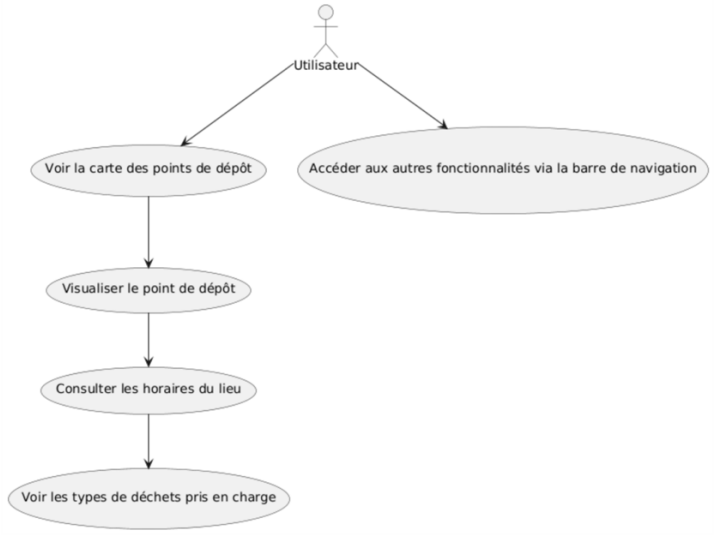

Figure 6, use case page collecte des points de dépot

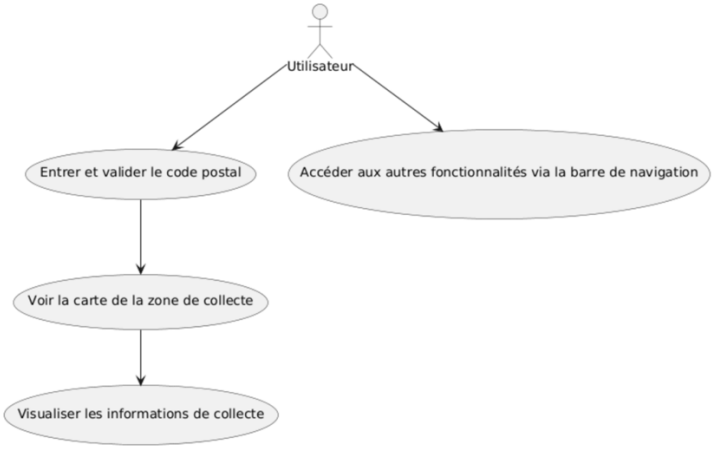

Figure 7, use case page informations de collecte

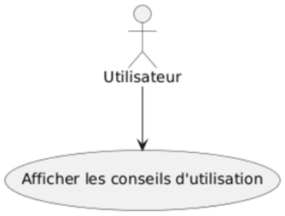

Figure 8, use case page d'informations

II. Développement et technologies utilisées

5. Langages de Programmation et normes de codage

- **JavaScript** : utilisé pour la gestion des messages envoyés à l'utilisateur, les actions des boutons ou encore de la caméra.
- **Python** : utilisé principalement pour les scripts de backend et les modèles d'intelligence artificielle, en raison de ses bibliothèques riches pour le machine learning et la manipulation de données.
- **HTML5 et CSS3** : utilisés pour structurer et styliser les pages web de l'application, assurant une facilité de conception pour nous Data Scientist, sans expérience dans la création d'application mobile ou d'interface web.
- **Json** : pour le stockage de données, tel que le fichier de configuration du pipeline automatique. Facile d'utilisation et léger.
- **CSV** : pour la gestion des données d'Open Food Facts, fichiers faciles à manipuler, stocker et importer comme data frame.

Pour assurer un développement soigné et compréhensible par tous nos pairs, nous avons suivi la convention PEP-8 Python, dont voici un aperçu :

Naming conventions

Never use l, O, or I single letter names as these can be mistaken for 1 and 0, depending on typeface	O = 2 # This may look like you're trying to reassign 2 to zero
Function	function, my_function
Variable	x, var, my_variable
Class	Model, MyClass
Method	class_method, method
Constant	CONSTANT, MY_CONSTANT, MY_LONG_CONSTANT
Module	module.py, my_module.py
Package	package, mypackage

Maximum Line Length and Line Breaking

PEP 8 suggests lines should be limited to 79 characters. This is because it allows you to have multiple files open next to one another, while also avoiding line wrapping.

Python will assume line contin uation if code is contained within parent heses, brackets, or braces:

```
def functi on( arg _one, arg_two,
                        arg _three, arg_four):
        return arg_one
```

If it is impossible to use implied contin uation, then you can use backsl ashes to break lines instead:

```
from mypkg import example1, \
        exa mple2, example3
# Recomm ended
total = (first _va riable
                + second _va riable
                - third_ var iable)
# Not Recomm ended
total = (first _va riable +
                  sec ond _va riable -
                  thi rd_ var iable)
```

Identation

Use 4 consecutive spaces to indicate indentation.

Prefer spaces over tabs.

Comments

Limit the line length of comments and docstrings to 72 characters.	Use complete sentences, starting with a capital letter.	Make sure to update comments if you change your code.

Block Comments

Indent block comments to the same level as the code they describe.	Start each line with a # followed by a single space.	Separate paragraphs by a line containing a single #.

Inline Comments

Use inline comments sparingly.

Write inline comments on the same line as the statement they refer to.

Separate inline comments by two or more spaces from the statement.

Start inline comments with a # and a single space, like block comments.

Don't use them to explain the obvious.

When to Avoid Adding Whitespace

The most important place to avoid adding whitespace is at the end of a line. This is known as trailing whitespace

Immediately inside parentheses, brackets, or braces:

Before a comma, semicolon, or colon:

Before the open parenthesis that starts the argument list of a function call:

Before the open bracket that starts an index or slice:

Between a trailing comma and a closing parenthesis:

To align assignment operators:

immediately inside brackets, as well as before commas and colons.

Programming Recommendations

Don't compare boolean values to True or False using the equivalence operator.	Use the fact that empty sequences are falsy in if statements.
Use is not rather than not ... is in if statements.	Don't use if x: when you mean if x is not None:

Use .startswith() and .endswith() instead of slicing.

Figure 9, extrait PEP-8 1/2

Code Layout

Surround top-level functions and classes with two blank lines

Surround method definitions inside classes with a single blank line.

Use blank lines sparingly inside functions to show clear steps.

Indentation Following Line Breaks

There are two styles of indentation you can use.

The first of these is to align the indented block with the opening delimiter:

An alternative style of indentation following a line break is a hanging indent. This is a typographical term meaning that every line but the first in a paragraph or statement is indented.

Indentation Following Line Breaks 2

```
def function(arg_one, arg_two,
                          arg _three, arg_four):
        return arg_one
x = 5
if (x > 3 and
       x < 10):
        pri nt(x)
x = 5
if (x > 3 and
       x < 10):
       # Both conditions satisfied
        pri nt(x)
x = 5
if (x > 3 and
               x < 10):
        pri nt(x)
# hanging indent
var = function(
        arg _one, arg_two,
        arg _three, arg_four)
```

Where to Put the Closing Brace

PEP 8 provides two options for the position of the closing brace in implied line continuations:

1 - Line up the closing brace with the first non-wh ite space character of the previous line:

```
list_o f_n umbers = [
       1, 2, 3,
       4, 5, 6,
       7, 8, 9
```

Where to Put the Closing Brace (cont)

```
>    ]
```

2 - Line up the closing brace with the first character of the line that starts the construct:

```
list_of_numbers = [
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
]
```

Documentation Strings

Surround docstrings with three double quotes on either side, as in """This is a docstring""".

Write them for all public modules, functions, classes, and methods.

Put the """ that ends a multiline docstring on a line by itself:

For one-line docstrings, keep the """ on the same line:

Whitespace Around Binary Operators

Surround the following binary operators with a single space on either side:

Assignment operators (=, +=, -=, and so forth)	Comparisons (==, !=, >, <, >=, <=) and (is, is not, in, not in)	Booleans (and, not, or)
When = is used to assign a default value to a function argument, do not surround it with spaces.	def function(default_paramet-er=5):	
y = x**2 + 5	z = (x+y) * (x-y)	

```
if x>5 and x%2==0:
# Treat the colon as the operator with lowest priority
list[x+1 : x+2]
```

In an extended slice, both colons must be surrounded by the same amount of whitespace

list[3:4:5]	list[x+1 : x+2 : x+3]
The space is omitted if a slice parameter is omitted	list[x+1 : x+2 :]

Figure 10, extrait PEP-8, 2/2

6. Framework et bibliothèques

Ci-dessous, une liste non exhaustive des librairies et Framework que nous avons utilisé :

absl-py==2.1.0
asttokens==2.4.1
astunparse==1.6.3
attrs==23.2.0
beautifulsoup4==4.12.3

bleach==6.1.0
Bottleneck
bson==0.5.9
cachetools==5.3.3
certifi==2024.7.4
cffi
charset-normalizer==3.3.2
click
comm==0.2.2
contourpy
cryptography
cycler
debugpy==1.8.2
decorator==5.1.1
defusedxml==0.7.1
dnspython
exceptiongroup==1.2.2
executing==2.0.1
fastjsonschema==2.20.0
flatbuffers==24.3.25
fonttools
gast==0.4.0
google-auth==2.32.0
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
grpcio==1.64.1
h5py==3.11.0
idna
ipykernel==6.29.5
ipython==8.26.0
jedi==0.19.1
Jinja2==3.1.4
joblib
jsonschema==4.23.0
jsonschema-specifications==2023.12.1
jupyter_client==8.6.2
jupyter_core==5.7.2
jupyterlab_pygments==0.3.0
keras==2.10.0
Keras-Preprocessing==1.1.2
kiwisolver
libclang==18.1.1
Markdown==3.6
MarkupSafe==2.1.5
matplotlib
matplotlib-inline==0.1.7
mistune==3.0.2
nbclient==0.10.0

nbconvert==7.16.4
nbformat==5.10.4
nest-asyncio==1.6.0
nltk
numexpr
numpy
oauthlib==3.2.2
opt-einsum==3.3.0
packaging
pandas
pandocfilters==1.5.1
parso==0.8.4
patsy
pexpect==4.9.0
pillow
platformdirs==4.2.2
ply==3.11
prompt_toolkit==3.0.47
protobuf==3.19.6
psutil==6.0.0
ptyprocess==0.7.0
pure-eval==0.2.2
pyasn1==0.6.0
pyasn1_modules==0.4.0
pycparser
Pygments==2.18.0
pymongo
pyparsing
PyQt5==5.15.10
PyQt5-sip
python-dateutil
pytz
pyzmq==26.0.3
referencing==0.35.1
regex
requests==2.32.3
requests-oauthlib==2.0.0
rpds-py==0.19.0
rsa==4.9
scikit-learn
scipy
seaborn
sip
six
soupsieve==2.5
stack-data==0.6.3
statsmodels
tensorboard==2.10.1

tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.10.1
tensorflow-estimator==2.10.0
tensorflow-io-gcs-filesystem==0.37.1
termcolor==2.4.0
threadpoolctl
tinycss2==1.3.0
tomli
tornado
tqdm
traitlets==5.14.3
typing_extensions==4.12.2
tzdata
unicodedata2
urllib3==2.2.2
wcwidth==0.2.13
webencodings==0.5.1
Werkzeug==3.0.3
wrapt==1.16.0

7. Systèmes d'exploitation et environnement de travail

Pour ce projet nous utilisons trois systèmes d'exploitation différents, nous permettant de tester le fonctionnement de nos infrastructures sur différentes machines ; une Ubuntu 24, un MacOs Sonoma ainsi que deux Windows 11. L'autre atout de travailler sur différents OS, est la diminution du risque de mises à jour comportant des failles, ce qui rendrait la machine concernée inutilisable quelques temps.

Pour éviter les problèmes de déploiement et les conflits durant l'installation sur une nouvelle machine, nous avons exclusivement travaillé sur un environnement virtuel. Sa configuration aura été scrupuleusement mise à jour au fur et à mesure par chaque membre de l'équipe, grâce au fichier requirements.txt. Cet environnement virtuel est une reproduction à l'identique de notre environnement de déploiement.

8. Déploiement de nos solutions

Actuellement nous suivons une stratégie basée sur trois couleurs, le bleu correspondant à l'environnement de développement, le rouge à celui de test, et enfin le vert pour celui de production. Le vert n'existe en réalité pas encore, notre application n'étant pas disponible à tous, mais seulement à un nombre limité d'utilisateurs.

Sur ce projet qui n'est à ce stade qu'un prototype, aucun pipeline automatique de tests ou de validation n'a été mise en place, nous comptons sur le suivi des bonnes pratiques par chaque membre de l'équipe, listées dans la documentation organisationnelle.

9. Contenu du projet, fichiers et exécution

Partie data science

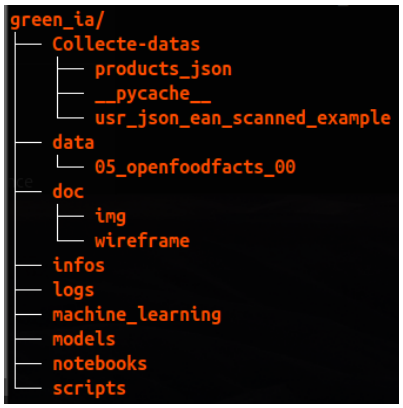

Figure 11, arborescence Green IA Data Science

Le projet s'organise de la manière présentée à gauche, un dossier Collecte-data est destiné au scraping des données des lieux de dépôt des déchets et des informations sur les collectes. Un dossier data contenant tous les fichiers csv et jsonl d'Open Food Facts, donc qui contient la documentation organisationnelle et technique ainsi que les schémas et les wireframes, infos contenant un simulateur de calcul d'éco score au format Excel, logs contenant les logs du pipeline automatique, model contenant tous les modèles sauvegardés, notebooks avec tous nos fichiers d'essais, et enfin : scripts avec tous nos scriptes python finaux, le script d'exécution du pipeline automatique au format bash ainsi que le fichier de configuration associé au format json.

Figure 12, exemple nom fichier de log

Les modèles sont identifiés par un code unique tel que : jour + mois + année + heures + minutes + secondes + millisecondes + _model.ci, étant la date de sauvegarde du modèle. Le principe de nommage est exactement le même pour les fichiers de logs au format txt.

Concernant le dossier contenant tous les csv prétraités issus du fichier jsonl d'Open Food Facts, la première valeur en début de nom représente l'identifiant unique donné par l'utilisateur pour l'identifier.

Figure 13, exemple nom dossier

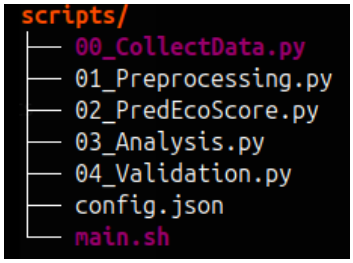

Figure 15, noms fichiers python

Pour les scriptes Python et les notebooks Jupiter, notre logique est que chaque fichier Python doit avoir son alter égo au format ipynb. De plus, la valeur du début de nom représente le numéro de l'étape dans le processus auquel le fichier appartient. Par exemple, l'étape numéro zéro est celle à laquelle nous téléchargeons les données d'Open Food Facts et nous les mettons à disposition de l'étape suivante, le numéro une, qui a pour but de préparer ces données téléchargées à l'entrainement d'un modèle d'intelligence artificielle.

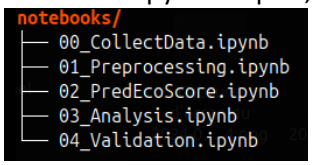

Figure 14, noms des notebooks

Partie web

L'application web se trouve dans un repository Git Hub dissocié de celui orienté Data Science, on y retrouve à la racine du projet, un dossier functions contenant les scripts JavaScript ainsi que le fichier index.html, contenant la page principale, exécutable en l'ouvrant avec un navigateur web. Dans le dossier public se trouvent les autres pages au format html, les polices d'écriture utilisées, les images et icones, le fichier css pour la mise en forme, les csv contenant les informations de dépôt et de collecte des déchets.

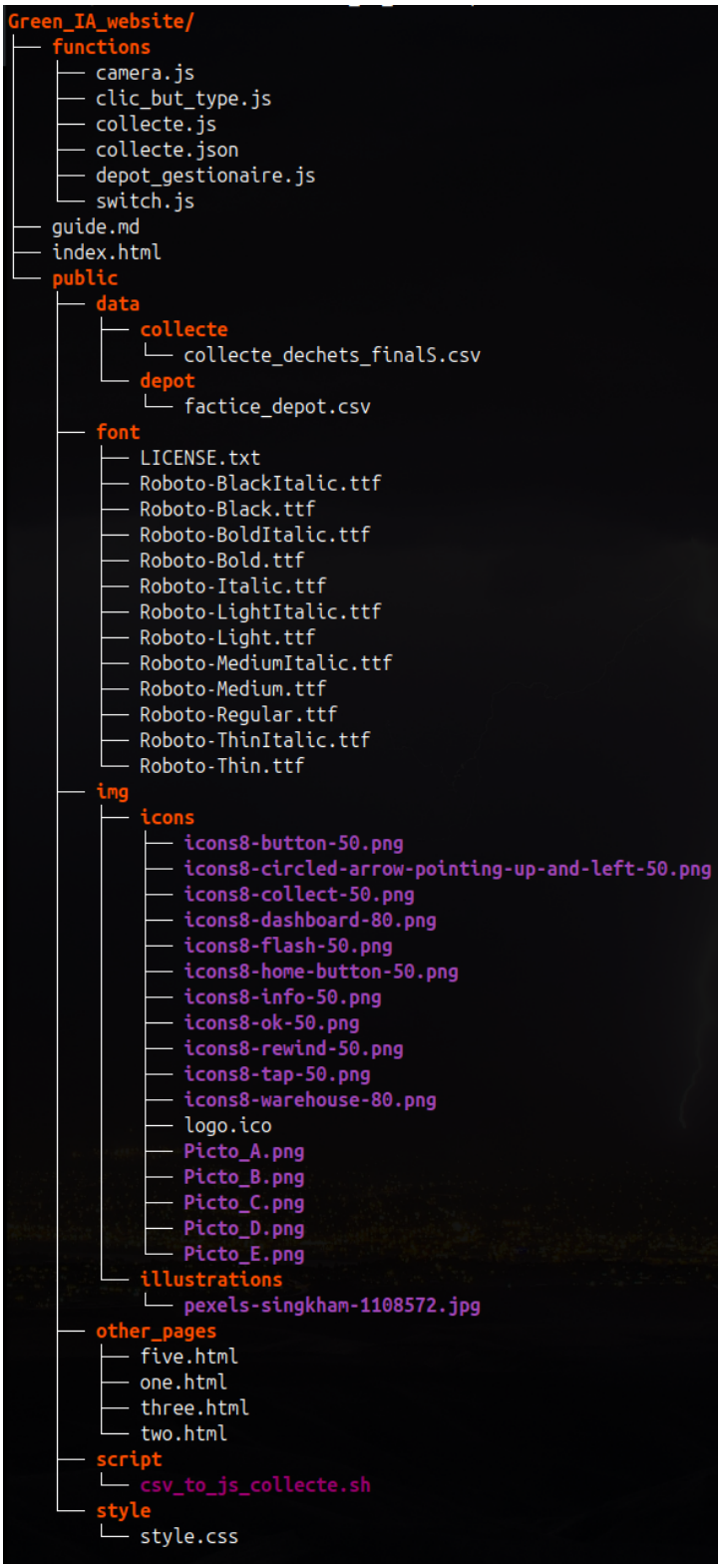

Figure 16, arborescence application web

III. IA et Data

10. Collecte des données

Ici, trois sources de données ont été utilisées ; OpenFoodFacts pour la prédiction de l'éco score, le portail métropolitain, ainsi que M data pour la gestion des points de collecte de déchets dans le département des Bouches du Rhône. Nous avons utilisé des techniques de scraping sur ces sites internet, notamment grâce à la librairie Beautiful Soup. Beautiful Soup permet de parser des documents HTML et XML, facilitant l'extraction de données à partir de pages web.

Les données pour la prédiction de l'éco score auront été plus complexes à traiter, en effet seulement deux fichiers compressés contiennent la totalité des articles d'Open Food Facts, un au format jsonl, l'autre au format Mongo DB Dump. Nous avons développé un script qui permet de télécharger le fichier compressé, avec la capacité de reprendre le téléchargement là où il s'était arrêté en cas de coupure réseau.

11. Préparation des données

Une fois le fichier compressé téléchargé, notre script le décompressera en fichier jsonl, puis le parcourra en écrivant au fur et à mesure chaque ligne dans un fichier au format csv. Chacun de ces fichiers csv contiendra un maximum de 10 000 lignes, afin de les rendre exploitables sur des machines de particuliers. Par la suite, chaque csv sera ouvert et se verra amputer des colonnes jugées inutiles, il passera de 720 colonnes par ligne à une petite quinzaine.

Enfin, toutes les techniques de préparation classiques seront réalisées sur chacun des csv, avec une normalisation des valeurs numériques, une tokenisation du contenu textuel, ainsi que la fusion de plusieurs colonnes.

12. Entraînement et validation des modèles

Régulièrement, un nouveau jeu de données provenant d'Open Food Facts sera téléchargé. Systématiquement un nouveau modèle sera entrainé sur ces nouvelles données, en parallèle du meilleur modèle que nous avons pu entrainer par le passé. Ainsi, un algorithme pourra comparer les résultats sur notre nouveau jeu de données de nos deux modèles, le meilleur sera mis de côté pour la prochaine mise à jour des données. Le csv avec le plus de bonnes réponses sera déployé.

Pour comparer les performances de nos réseaux de neurones, nous avons choisi les cinq métriques suivantes :

- L'erreur quadratique moyenne (MSE)
- L'erreur absolue moyenne (MAE) et médiane
- L'erreur quadratique moyenne racine (RMSE)
- Le coefficient de détermination (R^2)

Ces métriques d'évaluation sont adaptées à notre volonté de prédire une variable continue à partir d'un ensemble de caractéristiques, c'est-à-dire une tache de régression.

13. Stockage et mise à disposition des données

A ce stade du projet, nous recueillons les données d'Open Food Facts sur lesquelles nous faisons des prédictions puis que nous rendons disponible par le biais d'un fichier csv contenant nos prédictions.

Une prochaine étape serait de tout ajouter dans une base de données qui nous appartiennes, ou bien de demander à Open Food Facts d'implémenter nos résultats dans leur propre base de données.

14. Dashboard utilisateur

toto

15. Dashboard global

Titi

16. Pipeline automatique

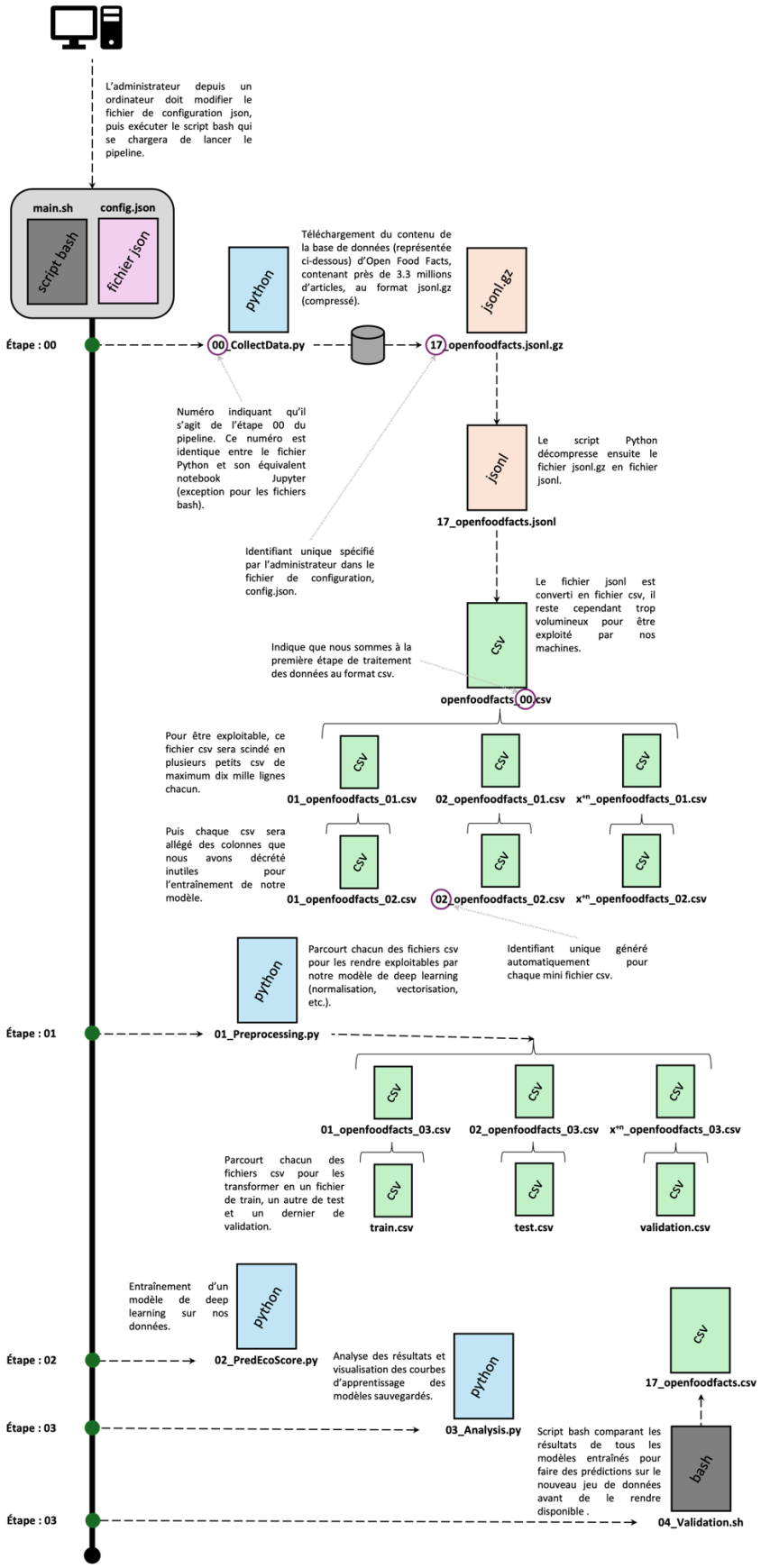

Figure 17, pipeline représentation graphique

IV. Interface utilisateur

17. Sécurité et authentification

A ce stade du projet, aucune authentification n'est nécessaire pour utiliser l'application, il n'est d'ailleurs tout simplement pas possible de se créer un compte. Les données personnelles de l'utilisateur ainsi que son historique d'articles scannés sont sauvegardés dans des fichiers json en local dans son navigateur. Aucune donnée ne nous est transmise, et stockée sur nos serveurs, le navigateur assurant de plus un niveau de sécurité que nous serions capable d'assurer sans un expert du domaine.

18. Intégration avec des services externes

Voulant garder du temps de développement, de nombreuses solutions développées par d'autres entreprises ont été importés dans notre application web. Pour la carte nous avons fait le choix d'utiliser OpenStreetMap. Gratuite, Open Source et facilement implémentable, cette solution nous a semblé être la meilleur pour afficher des cartes à nos utilisateurs, bien loin des prix exorbitants de Google. Pour scanner les articles de l'utilisateur par le biais de sa caméra, nous avons utilisé Quagga dans sa version 0.12.1, une libraire Open Source écrite en Java Script. Ces solutions ne sont utilisables que pour des applications déployées en https, le cas échant, ces librairies sont inaccessibles.

19. Présentation des maquettes

Page principale et page d'informations

Permet à l'utilisateur de scanner des articles alimentaires, télécharger sa liste de produits scannés et la vider, ainsi que visualiser le nombre d'articles, hors doublons comptabilisés par l'application. La page d'information sert de manuel utilisateur ligne, tous les conseils d'utilisation et toutes les informations sur le traitement des données utilisateur y figures.

Input : Images de la caméra de l'utilisateur.

Output : Liste d'historique des produits téléchargés, informations sur l'impact environnemental du produit scanné.

Figure 18, wireframe conseils

Figure 19, wireframe principale

Dashboards utilisateur et global

Un bouton permet à l'utilisateur de passer de son dashboard personnel à un dashboard global, ces dashboards ne sont pas hébergés sur la même machine que l'application web elle-même. Les graphiques s'adaptent à la taille de l'écran.

Input : Position du bouton.

Output : Affiche le dashboard associé à la position du bouton.

Figure 21, wireframe db utilisateur

Figure 20, wireframe db global

Informations dépôts et collectes

La page de gauche affiche sur une carte les heures de passage ainsi que les déchets traités, une fois le code postal de la ville où se trouve l'utilisateur entré. Celle de droite affiche directement l'endroit précis où se trouve l'utilisateur, affichant les lieux recyclant les types de déchets sélectionnés par l'utilisateur.

Input : Code postal de l'utilisateur, activation du bouton de soumission, chaque type de déchet pris en charge par l'application.

Output : Affiche les informations sur les points de dépôts et les horaires de collecte des déchets.

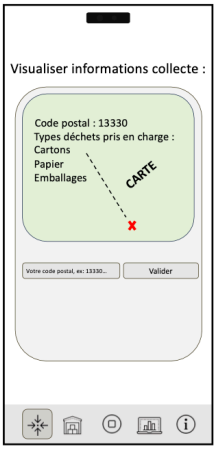

Figure 22, wireframe collecte

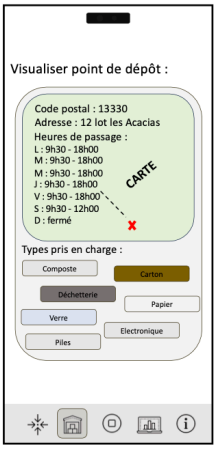

Figure 23, wireframe dépôt

20. Accessibilité

Ce choix des technologies web, nous permet d'éviter le développement d'une application Android et IOS, environnements avec lesquels nous ne sommes absolument pas familiers, impliquant de nombreuses heures de formations et de développement, pour un résultat sûrement médiocre. Avec le web nous avons pu développer une application responsive, aussi bien disponible sur ordinateur, que sur mobile. Il suffit de cliquer se rendre sur le lien suivant : Green Ia. Il n'est pas non plus nécessaire de créer un compte pour utiliser notre application.

L'unique prérequis pour accéder à cette application web est un accès un internet (wifi, 4G, 5G, etc) ainsi qu'un navigateur web. Attention cependant à certains navigateurs trop restrictifs qui pourraient bloquer l'accès à la caméra où aux notifications.

21. Chemins utilisateur

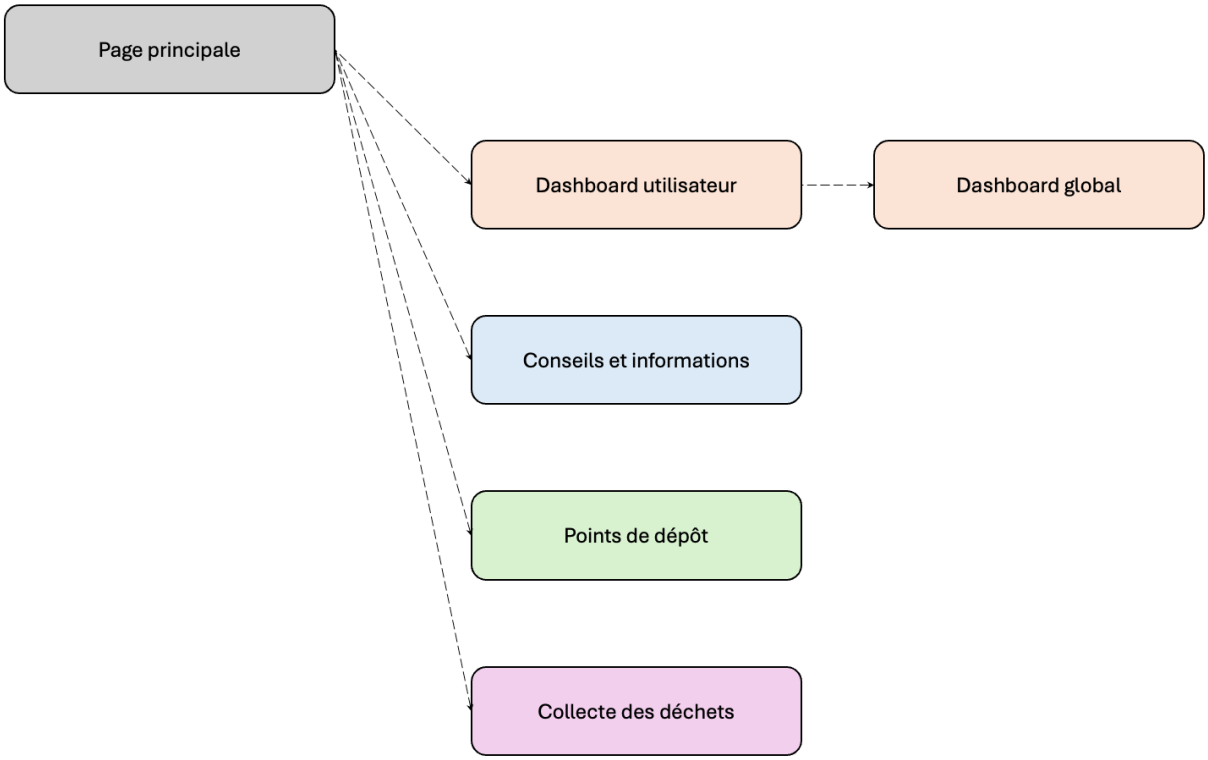

Figure 24, chemin utilisateur

V. Sécurité, maintenance et support

22. Contraintes de sécurité et RGPD

Nous ne traitons pas de données sensibles dans ce projet, aucune donnée personnelle ou médicale. A terme, une base de données sera créée et hébergée chez AWS ou Google, la sécurité des données leur sera ainsi confiée. Nous ferons en revanche auditer par un expert de la sécurité, la robustesse de nos sites web et applications qui afficheront et réaccueillerons ces données, notamment des données utilisateur. Mais à ce stade nos contraintes et risques sont très limités.

Si nous déployons notre deuxième version en Europe, nous devrons répondre aux exigences fixées par les RGPD. Il faut par exemple obtenir le consentement explicite de l'utilisateur pour collecter et traiter ses données personnelles, lui donner un droit d'accès pour visualiser ses données stockées, lui offrir la possibilité de demander la suppression de ses données, ainsi que notifier les autorités de protection des données et les utilisateurs en cas de violation des données. Si l'application est déployée au Royaume-Uni, nous devrons nous plier aux DPA. Celle-ci impose des contrôles de sécurité pour protéger les informations sensibles, une amélioration continue pour suivre et se mettre à jour des nouvelles techniques de piratage (hameçonnage, social engineering, etc.…).

23. Plan de maintenance et MCO

Une journée de maintenance préventive sera à prévoir chaque semaine, les trois premiers mois de déploiement, puis tous les mois durant le reste de la première année, puis à renouveler les années suivantes, les objectifs étants :

- Assurer une disponibilité continue de l'application en s'assurant du bon fonctionnement sur la précédente période (aucune saturation d'espace de stockage, aucune indisponibilité pour une raison particulière, ou avertissement)
- Correction de bugs mineures relevés par les administrateurs ou les utilisateurs
- Lancer le pipeline automatique de données pour entraîner un nouveau modèle sur les nouvelles données d'Open Food Facts
- Mettre à jour chaque librairie pour faire de petites montées de versions plus régulières.

Dans le cas où des défaillances majeures sont détectées, une demande d'intervention devra nous être adressée par courriel ou téléphone, afin de convenir d'un délai et d'une date d'intervention le plus rapidement possible.

24. Surveillance et mises à jour

Dans le pipeline automatique, chaque étape est sauvegardée par des logs. De la collecte des données à la validation des modèles en passant par le traitement des données ; tous les paramètres, les dates d'exécution des scripts et les chemins sont enregistrés dans un fichier json, envoyé par courriel aux administrateurs. Cela nous permet de vérifier le bon fonctionnement de tous nos scriptes ainsi que les performances de nos modèles et l'évolution de la composition de nos jeux de données.

Une version 2.0 de l'application, ouverte à tous pourrait être déployée après six mois de développement à plein temps, une fois une équipe de développeurs mobiles (Android et IOS), data scientists, administrateur systèmes et réseaux ainsi qu'un chef de projet ne soient missionnés. Les fonctionnalités supplémentaires arriveront avec le temps, une fois la version 2.0 pleinement opérationnelle.

VI. Tests, validation et améliorations

25. Critères d'acceptation pour chaque fonctionnalité

C : conforme, EC : en cours, NC : non conforme.

Identifiant fonctionnalité	Résultat attendu	Constatations	Validation
Site web			
01102023SW05	Le menu tous les boutons nécessaires, tous cliquables et redirigeant vers la bonne page	Ok	C
01102023SW06_00	Possibilité de scanner en boucle le code barre de produits alimentaires. Il faut afficher un message d'erreur dans le cas où le produit est inconnu. Pour ne pas empiéter sur la vie privée de nos utilisateurs, il sera également obligatoire de valider l'utilisation de sa caméra à chaque ouverture d'application	~~Impossible de scanner plusieurs articles sans fermer et ouvrir l'application de nouveau~~	C

01102023SW06_01	Le bouton doit permettre le téléchargement de la liste d'historique des produits scannés par l'utilisateur	Ok	C
01102023SW06_02	Bouton permettant à l'utilisateur de vider son historique	Ok	C
01102023SW07_00	L'utilisateur doit avoir un champ pour entrer son adresse postale, ainsi qu'un bouton	Ok	C
01102023SW07_01	Le code postal renseigné doit permettre d'afficher la position de la ville sur la carte, avec les informations de collecte des déchets associés, stockés dans le fichier json	Ok	C
01102023SW08_00	L'utilisateur doit pouvoir filtrer les types de déchets qu'il souhaite jeter	Ok	C
01102023SW08_01	Seuls les points de dépôt appartenant à une catégorie choisie par l'utilisateur et autour de lui, doivent apparaitre sur une carte. De plus, au clic sur l'un de ses points, toutes les informations sur le lieu doivent apparaitre (horaires, types, jours d'ouverture)	~~Différencier la couleur du pin de l'utilisateur de celui des boutiques~~	C
01102023SW11_00	Cette page doit contenir toutes les informations utiles à l'utilisateur pour comprendre le projet, ses objectifs et répondre à toutes ces questions courantes	~~Le bouton du menu associé à la page n'est pas mis en évidence comme les autres lorsqu'il est cliqué~~	C
01102023SW12	Aucun code inutile ne doit être présent, il doit être compréhensible par tous, respecter la PEP-8	Aucun soucis	C
01102023SW13	Le site web doit être déployer en https pour faire fonctionner tous les composants externes, disponible depuis n'importe quel navigateur	Ok	C
Dashboard			
01102023DBrd04	Dashboard utilisateur impactant, pour lui donner ses pistes d'amélioration.	Manque de responsivité. Certains graphiques apparaissent tout petit sur l'application.	EC
01102023DBrd05	Dashboard global doit servir à informer l'utilisateur ainsi que lui permettre de se comparer aux autres utilisateurs de l'application.		C
01102023DBrd06	Les dashboard doivent être déployés en https pour être compatible avec l'application web.		EC
Prédiction éco score			

01102023PES01	Ne produisant pas de données et n'ayant pas la possibilité de payer, nous devons nous tourner vers une ou plusieurs sources gratuites	Ok	C
01102023PES07	Étant donné la complexité de prédire l'éco score sur des produits alimentaires, un objectif de 80% de positifs sur des données de validation serait un bon résultat		EC

26. Stratégie de tests

Notre stratégie est d'avancer de manière itérative et incrémentale tout au long du cycle de développement du projet. D'abord, il y a la planification des tests, qui définit les critères d'acceptation et les scénarios pour chaque fonctionnalité développée (documentation organisationnelle). Ensuite, des tests automatisés sont développés pour les tests unitaires, d'intégration et de performance, en utilisant des scripts automatisés. Ces tests sont ensuite exécutés régulièrement pendant les phases de développement, d'intégration continue et de déploiement. Enfin, une validation globale du système est réalisée avant chaque mise en production pour s'assurer que toutes les fonctionnalités fonctionnent ensemble.

27. Tests unitaires, d'intégration et de performance

Chaque fonction, méthode et module est testé de manière isolée pour s'assurer qu'il se comporte comme prévu, avant d'être push sur le Git du projet. Les tests unitaires sont intégrés dans le pipeline CI/CD pour une exécution automatique à chaque modification du code. Nos tests unitaires se concentres sur les scripts python du pipeline automatique, pas sur le site web, étant un prototype.

Chacun d'entre nous a testé l'application et toutes ses fonctionnalités pour faire remonter tous les bugs avant le déploiement et une fois le déploiement effectué. Par exemple, la fonctionnalité de carte interactive qui localise les points de dépôt et de collecte de déchets est testée pour s'assurer que les données des points de collecte sont correctement récupérées et affichées sur la carte. Différents environnements de test ont été mis en place.

S'agissant d'un prototype, aucun test de charge n'a été effectué sur l'application. Ces tests seront cependant à prévoir dans le cas où le projet soit amené à grossir en ayant ses premiers utilisateurs récurrents, soutenu par une association ou financé par d'autres investisseurs.

28. Améliorations envisagées et versioning

Pour cette version 1.0 qui n'est qu'un prototype, les objectifs nous semble atteints, pour une version 2.0, de nombreux points seront cependant à améliorer, si nous voulons la rendre accessible à tous :

- Une base de données dans le cloud avec redondance dans un autre data center.
- La possibilité de créer un compte utilisateur.
- Changer les technologies de l'application web, pour de l'Angular.
- Améliorer l'UI et l'UX de l'application.
- Discuter avec Open Food Fact pour rendre disponible les prédictions d'éco score directement via leur API.

29. Confidentialité et responsabilités

Dans notre groupe de travail restreint, chacun d'entre nous a les mêmes responsabilités. La perte de fichiers, la fuite de données ou une éventuelle régression des performances sera directement imputée au collaborateur n'ayant pas validé lui-même, puis fait valider son travail par un pair avant de le mettre en production.

Nous ne sommes soumis à aucune réglementation nous interdisant de divulguer des informations sur notre projet. Celui-ci n'ayant aucune vocation lucrative et ne contenant aucune innovation à breveter.

VII. Annexes

30. Glossaire

Abréviation	Signification
IA	Intelligence Artificielle
Framework	Environnement de travail facilitant le développement d'une solution technique.
Responsivité	Possibilité d'adapter la taille du logiciel à la taille de l'écran de l'utilisateur
Front	Développement des aspects visuels du logiciel
Back	Développement de la partie logique du logiciel (caché à l'utilisateur)
API REST	API Representational State Transfer Application Program Interface est un style architectural qui permet aux logiciels de communiquer entre eux sur un réseau ou sur un même appareil. Le plus souvent les développeurs utilisent des API REST pour créer des services web. Souvent appelés services web RESTful, REST utilise des méthodes HTTP pour récupérer et publier des données entre un périphérique client et un serveur.
Open Source	En libre accès
Dashboard	Tableau dynamique et interactif pour visualiser ses données.
UI	Conception de l'interface utilisateur
UX	Conception de l'expérience utilisateur
PCA	Plan de continuité d'activité
PRA	Plan de reprise d'activité

31. Documents applicables

Description	Identification
23-24 Modalités Évaluations Titre EISI N7 Étudiants – AYC Pour les M2	REF [0]
Dépôt Moodle filière informatique M2	REF [1]

32. Diffusion du document

	Statut	Nom	Emis le
Diffusion	Edition	Charlemagne	05/07/2024
	Edition	Bamba	19/07/2024
	Edition	Pichard	23/07/2024

33. Historique des modifications

Version	Auteur	Description de la modification - Auteur	Date
0.30	Charlemagne	Architecture globale du document	05/07/2024
0.35	Bamba	Collecte des données	19/07/2024
0.60	Pichard	Sécurité, présentation du projet, dashboard, roles, glossaire	22/07/2024
1.00	Charlemagne	Tests, validation, schémas, IA, Data, IHM, architecture technique	23/07/2024