Rapport finale d’ESP

Na’el Benaïssa

ESP

2025

Table des matières

[1. Identification du projet 3](#_Toc199757195)

[2. Description du projet 3](#_Toc199757196)

[2.1. Objectifs généraux 3](#_Toc199757197)

[2.2. Problématique et contexte 3](#_Toc199757198)

[2.3. Utilité et public cible 3](#_Toc199757199)

[3. Identification des domaines de l’informatique concernés 3](#_Toc199757200)

[3.1. Développement mobile multiplateforme 4](#_Toc199757201)

[3.2. Intelligence artificielle 4](#_Toc199757202)

[3.3. Cloud computing 4](#_Toc199757203)

[3.4. Conteneurisation et virtualisation 4](#_Toc199757204)

[3.5. Services backend (BaaS) – Supabase 5](#_Toc199757205)

[3.6. Intégration d’API externes 5](#_Toc199757206)

[4. Réalisation du projet 5](#_Toc199757207)

[4.1. Méthodologie adoptée 5](#_Toc199757208)

[4.2. Étapes de réalisation 5](#_Toc199757211)

[4.3. Technologies choisies 6](#_Toc199757212)

[4.4. Nombre d’heures consacrées 6](#_Toc199757213)

[4.5. Ressources mobilisées 6](#_Toc199757214)

[5. Sources d'information et de documentation 7](#_Toc199757215)

[5.1. Ressources techniques 7](#_Toc199757216)

[5.2. Données d'entraînement 7](#_Toc199757217)

[5.3. Ressources graphiques 8](#_Toc199757218)

[5.4. Autres sources 8](#_Toc199757219)

[6. Difficultés rencontrées et solutions apportées 8](#_Toc199757220)

[7. Limites de la réalisation 9](#_Toc199757221)

[8. Compétences du DEC/BTS réinvesties 10](#_Toc199757222)

[8.1. Analyse des besoins et conception 10](#_Toc199757223)

[8.2. Développement d’applications 10](#_Toc199757224)

[8.3. Mise en production et déploiement 11](#_Toc199757225)

[8.4. Exploitation de bases de données et services web 11](#_Toc199757226)

[8.5. Communication technique et documentation 11](#_Toc199757227)

[8.6. Valorisation du projet et communication professionnelle 11](#_Toc199757228)

[9. Bilan des apprentissages réalisés 12](#_Toc199757229)

[9.1. Compétences techniques approfondies 12](#_Toc199757230)

[9.2. Compétences méthodologiques 12](#_Toc199757231)

[9.3. Travail autonome et prise d’initiative 12](#_Toc199757232)

## 1. Identification du projet

**LeafGuard – Assistant intelligent pour le soin des plantes**

LeafGuard est une application mobile développée avec Flutter, intégrant une intelligence artificielle accessible via une API Python conçu et déployée dans un conteneur Docker sur Google Cloud Platform. Elle permet aux utilisateurs de diagnostiquer les maladies des plantes à partir de photos et de recevoir des conseils de soins personnalisés.

## 2. Description du projet

### 2.1. Objectifs généraux

L’objectif principal du projet LeafGuard est de concevoir une application mobile d’assistance dédiée à la santé des plantes. Cette application repose sur une intelligence artificielle également conçu et capable de diagnostiquer des maladies végétales à partir de photographies prises par les utilisateurs. Elle vise un public d’amateurs de jardinage en leur fournissant des conseils personnalisés pour l’entretien et la guérison de leurs plantes, tout en restant simple d’utilisation.

### 2.2. Problématique et contexte

Le soin des plantes, bien que courant, reste un domaine technique où les erreurs d’entretien ou l’ignorance des signes de maladies peuvent rapidement conduire à leur détérioration. Identifier avec précision une maladie végétale nécessite souvent des compétences spécialisées, rarement accessibles aux particuliers. Dans ce contexte, LeafGuard se propose de combler ce fossé en généralisant l’accès à un outil de diagnostic végétal intelligent, directement depuis un smartphone, grâce à l’intelligence artificielle et au développement mobile multiplateforme.

### 2.3. Utilité et public cible

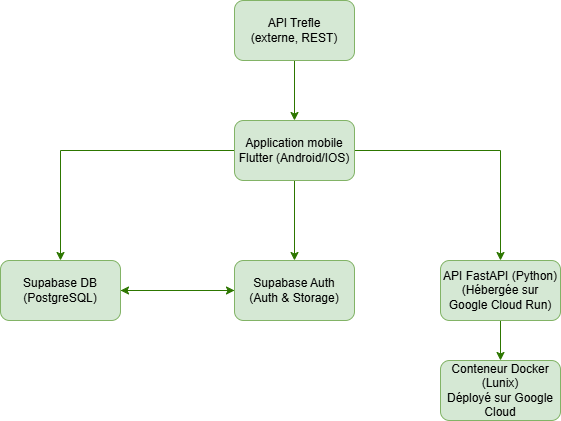
LeafGuard est conçu pour les passionnés de jardinage, les propriétaires de potagers, les jardiniers urbains ou toute personne souhaitant entretenir efficacement ses plantes. L'application apporte une réelle valeur ajoutée en :

* réduisant les incertitudes liées à l'identification des maladies ;
* proposant des soins ciblés et personnalisés ;
* facilitant l’organisation des tâches d’entretien par des rappels automatiques ;
* donnant un accès rapide à des fiches informatives sur les plantes courantes.

Dans sa version académique, le projet se concentre sur trois plantes : pommes de terre, poivrons et tomates. L’extension à d’autres variétés est envisagée comme évolution future.

## 3. Identification des domaines de l’informatique concernés

La réalisation du projet LeafGuard fait intervenir plusieurs domaines majeurs de l’informatique. Chacun de ces domaines a joué un rôle essentiel dans le développement, l’intégration et le déploiement de l’application mobile.



### 3.1. Développement mobile multiplateforme

Le cœur du projet repose sur le framework Flutter, qui permet de créer des applications mobiles compatibles à la fois avec Android et iOS à partir d’une base de code unique. Ce domaine englobe la conception d’interfaces utilisateur, la gestion des interactions, la navigation entre les écrans, et l’intégration des fonctionnalités natives des appareils (appareil photo, notifications, etc.).

### 3.2. Intelligence artificielle

L’intelligence artificielle constitue un volet fondamental du projet. Elle est utilisée pour l’analyse d’images et la reconnaissance de maladies végétales. Un modèle d’IA, entraîné avec des images annotées, est intégré dans une API Python, exposée à l’application mobile. Ce domaine inclut le prétraitement des images, l’inférence, et l’interprétation des résultats de l’algorithme de classification.

### 3.3. Cloud computing

L’intelligence artificielle est encapsulée dans un service accessible via une API REST, déployée sur Google Cloud Platform à l’aide de Google Cloud Run. Cela permet de bénéficier d’une infrastructure évolutive, performante et disponible à la demande, tout en simplifiant la gestion des déploiements. Le cloud est également utilisé pour le stockage distant de certaines ressources.

### 3.4. Conteneurisation et virtualisation

Pour garantir la portabilité et l’isolement du service d’IA, celui-ci est encapsulé dans un conteneur Docker. Cela permet de maîtriser l’environnement d’exécution, de faciliter le déploiement sur le cloud, et de reproduire facilement le comportement du service en local. Le projet s’inscrit ainsi dans les pratiques DevOps modernes.

### 3.5. Services backend (BaaS) – Supabase

LeafGuard s’appuie sur Supabase, un service backend open-source, pour gérer :

* l’authentification des utilisateurs ;
* le stockage des données ;
* la gestion de fichiers multimédias (photos des plantes) ;
* l’interrogation d’une base de données PostgreSQL.

### 3.6. Intégration d’API externes

L’application communique également avec l’API Trefle, une base de données botanique en ligne permettant de compléter les informations sur les plantes. Cette intégration nécessite des compétences en consommation d’API REST, en gestion des appels HTTP et en traitement de données JSON.

## 4. Réalisation du projet

### 4.1. Méthodologie adoptée

### Le projet a été réalisé selon une approche itérative et incrémentale, en suivant une logique agile adaptée à un projet individuel. Chaque fonctionnalité a été développée, testée et validée de manière progressive.

### Un fichier texte simple a été utilisé pour recenser les tâches à faire et celles déjà réalisées, servant ainsi de support de suivi. Par ailleurs, une utilisation rigoureuse de Git a permis de suivre efficacement l’évolution du projet, en documentant clairement chaque ajout de fonctionnalité, chaque modification de code, ainsi que les différentes étapes de correction et d’intégration.

### 4.2. Étapes de réalisation

Le projet s’est structuré autour des étapes suivantes :

|  |  |
| --- | --- |
| Étape | Description |
| 1 | **Étude de faisabilité** : vérification des outils à utiliser (Flutter, Supabase, FastAPI, Docker, Railway (puis Google Cloud Platform)). |
| 2 | **Conception fonctionnelle** : définition des cas d’utilisation, choix de la maquette de l’application, choix des APIs. |
| 3 | **Entraînement et intégration du modèle d’IA** : entraînement du modèle de classification d’images et création de l’API FastAPI. |
| 4 | **Déploiement dans un conteneur Docker** : conteneurisation de l’API et déploiement sur railway, puis pour des raisons financières Google Cloud Run. |
| 5 | **Développement de l’interface mobile** : création de l’application avec Flutter (accueil, diagnostic, rappels, guide). |
| 6 | **Connexion aux services externes** : intégration de Supabase (authentification, base de données) et de l’API Trefle. |
| 7 | **Tests et validation fonctionnelle** : vérification de la compatibilité mobile, test des réponses de l’IA, et vérification des performances. |

### 4.3. Technologies choisies

|  |  |  |
| --- | --- | --- |
| Domaine | Technologie | Rôle |
| Développement mobile | **Flutter** (Dart) | Interface utilisateur multiplateforme |
| IA & API backend | **Python**, **FastAPI** | API REST contenant le modèle d’IA |
| Conteneurisation | **Docker** | Isolation et portabilité de l’API |
| Cloud | **Google Cloud Run** | Hébergement scalable de l’API |
| Base de données & Auth | **Supabase (PostgreSQL)** | Stockage des utilisateurs et des diagnostics |
| API externe | **Trefle API** | Données botaniques enrichies |
| Outils de versioning | **Git + GitHub** | Suivi du code source |

### 4.4. Nombre d’heures consacrées

Le projet a nécessité environ 85 heures de travail réparties comme suit :

|  |  |
| --- | --- |
| Activité | Temps estimé |
| Conception fonctionnelle | 10 h |
| Développement Flutter | 30 h |
| IA & API Python | 20 h |
| Docker & déploiement cloud | 5 h |
| Intégrations externes (Supabase, Trefle) | 10 h |
| Tests, corrections, optimisation | 10 h |

### 4.5. Ressources mobilisées

#### 4.5.1. Ressources matérielles

* Ordinateur portable personnel
* Tablette Android pour les tests

#### 4.5.2. Ressources logicielles

* IDE : Visual Studio Code (Python) & Android Studio (Flutter)
* Docker Desktop
* Outils Google Cloud (Cloud Run, Container Registry)

#### 4.5.3. Ressources financières

* Le projet a été conçu sans frais, en utilisant des outils gratuits ou avec des crédits étudiants.
* Le crédit gratuit de Google Cloud pour les étudiants notamment a permis d’héberger l’API IA sans surcoût.

## 5. Sources d'information et de documentation

Le développement du projet LeafGuard s’est appuyé sur diverses ressources documentaires, outils en ligne et référentiels communautaires. Ces sources ont été essentielles à la conception, à l’implémentation ainsi qu’à l’optimisation des différentes composantes du projet, tant sur le plan visuel que technique.

### 5.1. Ressources techniques

* **Documentation officielle de Flutter** : utilisée pour la mise en œuvre des interfaces, la gestion des états, les animations et l’accès aux fonctionnalités natives des smartphones.  
  → <https://docs.flutter.dev>
* **Documentation de FastAPI** : pour la conception et le déploiement de l’API Python destinée à l’IA.  
  → <https://fastapi.tiangolo.com>
* **Documentation Docker** : pour la création, la configuration et la gestion des conteneurs d’exécution.  
  → <https://docs.docker.com>
* **Google Cloud Run** : documentation et guides de déploiement pour héberger l’API.  
  → <https://cloud.google.com/run>
* **Supabase Documentation** : pour l'intégration de l’authentification, du stockage et de la base de données.  
  → <https://supabase.com/docs>
* **TensorFlow et bibliothèques associées (NumPy, Keras, OpenCV)** : pour l’entraînement et l’inférence du modèle d’IA.  
  → <https://www.tensorflow.org>

### 5.2. Données d'entraînement

Les images utilisées pour entraîner le modèle d’intelligence artificielle proviennent d’un jeu de données librement accessible sur Kaggle :

* **Plant Disease Dataset**  
  → <https://www.kaggle.com/datasets/emmarex/plantdisease>

Ce jeu de données contient plusieurs milliers d’images annotées de plantes atteintes de différentes maladies, réparties selon l’espèce et le type de pathologie. Il a été essentiel pour la phase de classification.

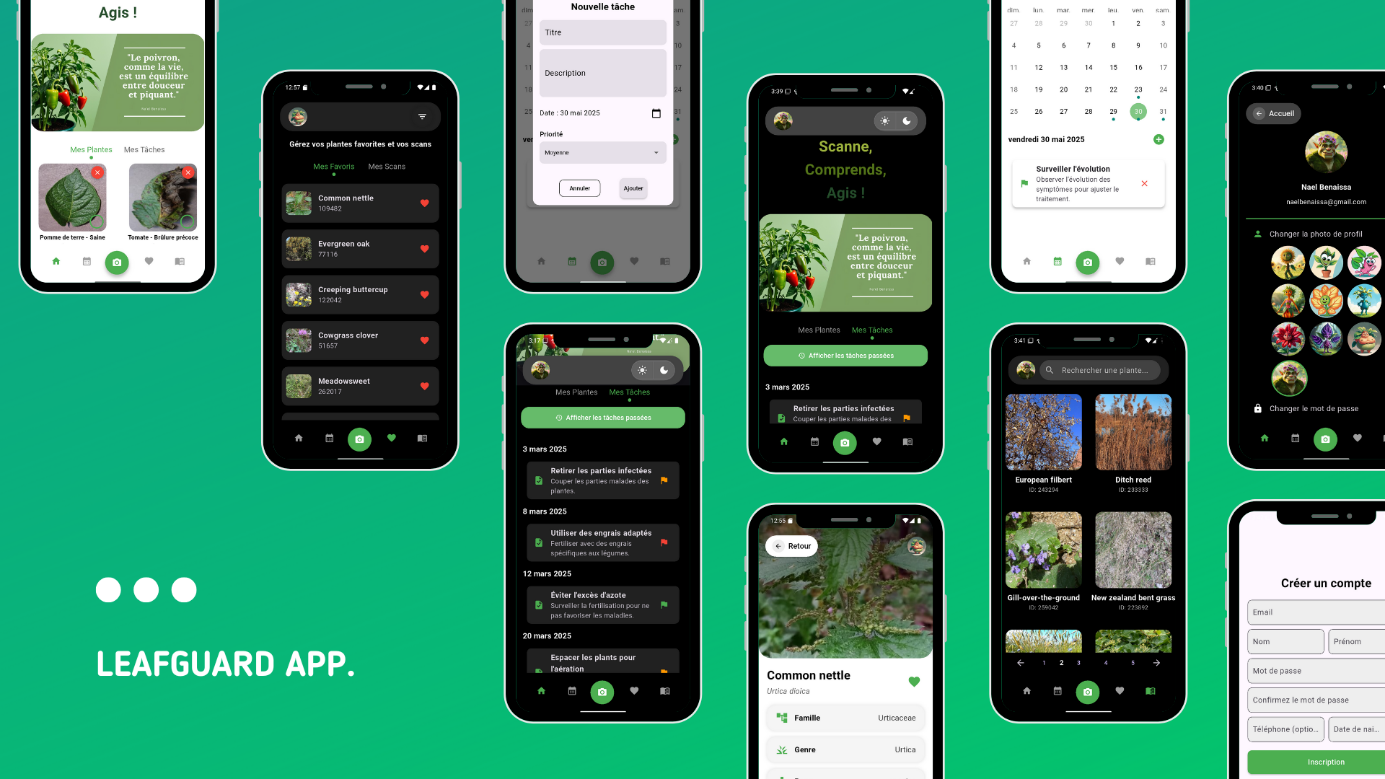
### 5.3. Ressources graphiques

L’interface utilisateur de l’application s’inspire du travail de design proposé sur Dribbble, qui a guidé le choix des couleurs, la disposition des éléments et l’ergonomie générale :

* **Plant Management App**  
  → <https://dribbble.com/shots/25166605-Plant-Management-App>

### 5.4. Autres sources

* Forums communautaires : Stack Overflow, Reddit r/flutterdev
* Tutoriels vidéo YouTube (Flutter, FastAPI)
* Articles Medium sur l’optimisation de modèles d’IA et le déploiement de projets via Docker



## 

## 6. Difficultés rencontrées et solutions apportées

Au cours du développement du projet LeafGuard, plusieurs obstacles techniques et organisationnels ont été rencontrés, nécessitant des ajustements et la mise en place de solutions alternatives.

L’un des problèmes majeurs a concerné la phase d’entraînement du modèle d’intelligence artificielle. Mon ordinateur personnel, bien que adapté au développement logiciel, ne disposait pas de la puissance de calcul nécessaire pour traiter rapidement les milliers d’images contenues dans le dataset. Cette contrainte matérielle a significativement allongé les délais d’entraînement. Afin de limiter ce temps sans compromettre les performances du modèle, j’ai pris la décision d’alléger les paramètres d’apprentissage, notamment en réduisant le nombre d’epochs (itérations complètes sur le jeu de données), ce qui a permis d’obtenir un bon compromis entre qualité de prédiction et temps de traitement.

Un autre point bloquant est survenu lors de la mise en production du backend IA. Dans un premier temps, j’avais choisi de déployer l’API Python via la plateforme Railway, qui proposait un crédit gratuit de 5 USD. Cependant, cette allocation a rapidement été consommée pendant la phase de test, rendant le service inutilisable en milieu de développement. Pour contourner ce problème, je me suis orienté vers Google Cloud Platform, qui offre un crédit étudiant de 350 USD ainsi que deux mois d’utilisation gratuite de Google Cloud Run. Cette solution m’a permis de déployer durablement l’API dans un conteneur Docker, en garantissant sa disponibilité au moins jusqu’au 14 juin, date de fin des avantages étudiants.

Sur le plan de l’interface, une autre difficulté s’est manifestée : n’étant pas designer mais développeur full-stack, la conception d’une interface graphique esthétique et ergonomique a nécessité un effort particulier. Ne maîtrisant pas tous les codes visuels d’une bonne expérience utilisateur, j’ai choisi de faire tester l’application par mon professeur ainsi que par plusieurs camarades. Leurs retours m’ont permis de corriger certaines incohérences visuelles et de détecter quelques bugs fonctionnels pendant la phase de validation.

L’adaptation de l’interface à différents formats d’écrans (smartphones, tablettes) a également représenté un défi technique important. Certaines vues s’affichaient mal ou se comportaient de manière inattendue sur des écrans plus larges. Pour résoudre ce problème, j’ai dû intégrer systématiquement des MediaQuery dans les composants Flutter afin d’ajuster dynamiquement les tailles, marges et espacements en fonction des dimensions de l’écran.

Enfin, la phase d’expérimentation des différentes options d’hébergement a nécessité de nombreuses reconfigurations. À chaque changement de plateforme, il me fallait adapter la méthode de déploiement ainsi que les paramètres de connexion, ce qui a occasionné une perte de temps non négligeable. Pour pallier cette instabilité, j’ai uniformisé l’environnement d’exécution à l’aide de Docker, ce qui m’a permis d’emballer l’API et ses dépendances dans un conteneur unique, reproductible quel que soit le fournisseur d’hébergement choisi.

## 7. Limites de la réalisation

Bien que le projet LeafGuard ait permis de développer une application fonctionnelle intégrant une intelligence artificielle et une interface mobile complète, certaines limitations techniques et fonctionnelles subsistent dans sa version actuelle.

La principale limite concerne la portée de l’intelligence artificielle. Le modèle développé ne prend en charge que la détection de maladies sur trois types de plantes : la pomme de terre, le poivron vert, et la tomate. Ce choix restreint est principalement dû aux contraintes liées aux données d'entraînement disponibles et à la capacité de traitement. Le dataset utilisé (provenant de Kaggle) contient un grand nombre d’images, mais toutes les espèces n’y sont pas représentées de manière équilibrée. Afin de garantir un entraînement de qualité dans un délai raisonnable, seules les classes les plus représentées et les plus pertinentes pour une version de démonstration ont été conservées. Une extension à d'autres variétés serait envisageable dans une future version, avec des moyens informatiques plus puissants ou via des services cloud d'entraînement.

D’un point de vue fonctionnalité, certaines idées initiales n’ont pas pu être mises en œuvre dans le cadre du rendu académique. Par exemple, l’ajout d’une fonctionnalité de géolocalisation pour recommander des soins en fonction du climat local, ou encore la possibilité de suivre l’évolution d’une plante dans le temps via un système de journal d’entretien avec photos. Ces éléments ont été volontairement écartés pour recentrer le projet sur un périmètre réaliste et techniquement abouti dans le temps imparti.

Sur le plan de l’ergonomie, bien que des efforts aient été faits pour rendre l’application utilisable et agréable à l’œil, certaines limites subsistent en raison de mon profil non spécialisé en design. Certains choix graphiques, bien que fonctionnels, pourraient être optimisés pour améliorer l'expérience utilisateur globale (animations, transitions, hiérarchie visuelle). Ces points ont été partiellement compensés par des retours utilisateurs, mais n'ont pas pu être entièrement retravaillés.

Enfin, bien que l’API ait été déployée avec succès sur Google Cloud Run, cette solution reste dépendante de crédits étudiants limités dans le temps. À partir du 14 juin, une solution d’hébergement durable devra être envisagée si le projet devait évoluer en application réelle et distribuée à grande échelle. De plus, l’utilisation de services gratuits comme Supabase impose des limites de quota (requêtes, bande passante) qui pourraient restreindre l’utilisation en cas d’augmentation du nombre d’utilisateurs.

## 8. Compétences du DEC/BTS réinvesties

La réalisation du projet LeafGuard a mobilisé un large éventail de compétences issues des programmes de BTS Services Informatiques aux Organisations (SIO) et du Diplôme d'Études Collégiales (DEC) en informatique. Le caractère complet du projet, allant de la conception à la mise en production, a permis de mettre en pratique des compétences techniques, méthodologiques et professionnelles acquises tout au long de la formation.

### 8.1. Analyse des besoins et conception

J’ai appliqué les compétences d’analyse pour identifier les besoins fonctionnels du projet, construire une architecture logicielle cohérente et modulaire, et définir les principales fonctionnalités de l’application. Cette étape a été réalisée en autonomie, avec une approche centrée utilisateur, intégrant les retours de futurs utilisateurs pour adapter l’interface et les usages.

Compétences DEC/BTS :

* Recueillir et analyser les besoins du client
* Élaborer un cahier des charges ou un schéma fonctionnel
* Maîtrise des diagrammes fonctionnels simples (UML, flux)

### 8.2. Développement d’applications

La majeure partie du projet a consisté en la mise en œuvre de l’application mobile avec Flutter et Dart. J’ai également développé une API Python via FastAPI et intégré des fonctionnalités d’intelligence artificielle. Le code a été versionné avec Git et structuré selon les bonnes pratiques de développement logiciel.

Compétences DEC/BTS :

* Développer des composants logiciels à partir de spécifications
* Implémenter une interface utilisateur (mobiles et responsive)
* Utiliser un gestionnaire de versions (Git/GitHub)
* Concevoir des scripts de traitement (Python pour IA)

### 8.3. Mise en production et déploiement

Le projet m’a permis d’aborder concrètement la conteneurisation avec Docker, le déploiement cloud avec Google Cloud Run, et la gestion des environnements d’exécution. Ces compétences s’inscrivent dans une logique de professionnalisation et d’adaptation aux pratiques modernes (DevOps, cloud computing).

Compétences DEC/BTS :

* Préparer et déployer une solution applicative
* Automatiser le déploiement et la mise à jour
* Utiliser des environnements Linux et maîtriser les outils de conteneurisation

### 8.4. Exploitation de bases de données et services web

L’utilisation de Supabase m’a permis de manipuler une base de données relationnelle (PostgreSQL), d’implémenter l’authentification des utilisateurs, et de sécuriser les échanges entre l’application et le backend.

Compétences DEC/BTS :

* Interagir avec des bases de données (SQL)
* Intégrer des API REST dans une application cliente
* Gérer des droits et des accès (authentification)

### 8.5. Communication technique et documentation

Enfin, le projet a requis la rédaction de documentation technique (commentaires de code, configuration, procédures de test) ainsi que la préparation de ce rapport structuré et des documentations techniques. Ces livrables démontrent la capacité à documenter une solution de manière claire et professionnelle.

Compétences DEC/BTS :

* Documenter une application ou un livrable
* Rédiger un rapport structuré à des fins de soutenance ou de diffusion

### 8.6. Valorisation du projet et communication professionnelle

Le projet LeafGuard m’a également permis de développer des compétences en communication numérique et en valorisation professionnelle, éléments de plus en plus importants dans les métiers de l’informatique. En documentant le projet sur des plateformes comme GitHub, j’ai appris à structurer un dépôt de manière claire (README, captures, arborescence logique), à écrire un code compréhensible pour d’autres développeurs, et à présenter mes travaux dans une logique de publication ouverte.

Par ailleurs, j’ai utilisé ce projet comme support de communication sur LinkedIn, afin de mettre en avant mes compétences et ma capacité à mener un projet complet. J’ai rédigé un post synthétique avec un lien vers le code, des visuels, et une brève explication technique, renforçant ainsi ma présence professionnelle en ligne. Cette démarche contribue à la gestion de mon image numérique en tant que futur développeur, à la fois autonome, polyvalent et tourné vers les bonnes pratiques du métier.

Compétences DEC/BTS :

* Communiquer avec clarté sur des plateformes techniques (GitHub)
* Mettre en valeur ses compétences dans un contexte professionnel (LinkedIn)
* Développer une posture professionnelle à travers sa présence en ligne

## 9. Bilan des apprentissages réalisés

La réalisation du projet LeafGuard a constitué une expérience formatrice et riche, permettant de consolider mes acquis tout en explorant de nouveaux domaines techniques. Ce projet m’a offert l’opportunité de conduire un développement complet, de la conception à la mise en production, dans des conditions proches de celles rencontrées en milieu professionnel.

### 9.1. Compétences techniques approfondies

J’ai renforcé mes compétences en développement mobile avec Flutter, en concevant une interface fluide, ergonomique et responsive, compatible avec différents formats d’écran. J’ai également perfectionné ma maîtrise du langage Python, notamment dans le contexte de l’intelligence artificielle et du traitement d’images. L’intégration de bibliothèques telles que TensorFlow et OpenCV m’a permis de comprendre les enjeux liés à l’entraînement, à l’optimisation et à l’exploitation de modèles de classification.

Le projet m’a également permis de me familiariser avec les techniques de conteneurisation (Docker) et avec le déploiement d’applications cloud, via Google Cloud Run. Ces outils, incontournables dans le monde professionnel, ont été utilisés pour assurer la portabilité, la scalabilité et la disponibilité de l’API d’IA.

### 9.2. Compétences méthodologiques

Sur le plan méthodologique, j’ai appris à structurer efficacement mon travail grâce à une organisation personnelle rigoureuse. La planification des tâches, le suivi de l’avancement via des fichiers de suivi, et l’usage intensif de Git m’ont permis de gérer le projet de manière fluide et cohérente, malgré les contraintes de temps et de ressources. J’ai également mis en œuvre une approche itérative, en testant chaque fonctionnalité de manière isolée avant son intégration finale.

### 9.3. Travail autonome et prise d’initiative

Ce projet a également développé mon sens de l’autonomie et ma capacité à trouver des solutions en situation réelle. J’ai dû surmonter des difficultés techniques, gérer des limites matérielles, chercher des alternatives d’hébergement et m’adapter à des domaines hors de ma spécialité, comme le design d’interface. Les tests utilisateurs et les échanges avec mon encadrant m’ont permis d’améliorer la qualité globale du produit final.

En conclusion, ce projet m’a permis de combiner des compétences issues de ma formation en BTS SIO et de mon DEC en informatique, tout en me préparant concrètement aux exigences du développement logiciel en entreprise. Il représente une synthèse concrète de mon parcours, de mes acquis techniques, et de ma capacité à mener à bien un projet complet de A à Z.