Institut d'Informatique Claude Chappe
Le Mans Université

Cahier des Charges
Application Android ShrooML

Bilal MEZRHAB
Luka COGNARD
Maelig PESANTEZ
Mewen PUREN

Master Informatique Mention IA
Module DevOps

Encadrants :
Frédéric LAGRANGE
Aghilas SINI

Janvier 2026 - Avril 2026

Le Mans Université

Table des matières

1 Introduction **2**
1.1 Objectif du document . . . 2
1.2 Contexte du projet . . . 2
1.3 Définitions, acronymes et abréviations . . . 2

2 Description Générale **3**
2.1 Objectifs de l'application . . . 3
2.2 Périmètre fonctionnel . . . 3

3 Exigences Fonctionnelles **4**

4 Exigences Non-Fonctionnelles **5**

5 Contraintes Techniques **6**
5.1 Application mobile . . . 6
5.2 Backend et intelligence artificielle . . . 6

6 Architecture Système **6**

7 Hypothèses et Dépendances **7**

8 Tableau des Priorités et Roadmap **8**

9 Critères d'Acceptation et Tests **8**
9.1 Tests fonctionnels . . . 8
9.2 Tests unitaires . . . 8
9.3 Tests d'intégration . . . 9
9.4 Tests de performance . . . 9
9.5 Validation globale . . . 9

10 Annexes **10**

1 Introduction

1.1 Objectif du document

Ce document constitue le cahier des charges de l'application Android **ShrooML**. Il a pour objectif de définir de manière formelle le périmètre du projet, les fonctionnalités attendues, les contraintes techniques ainsi que les exigences fonctionnelles et non fonctionnelles. Ce document sert de référence tout au long du cycle de développement, aussi bien pour l'équipe projet que pour les encadrants pédagogiques, et vise à assurer une compréhension commune des objectifs et des choix techniques retenus.

1.2 Contexte du projet

Le projet **ShrooML** est réalisé dans le cadre du module *DevOps* en première année du Master Informatique, mention Intelligence Artificielle, à l'Université du Mans. Il est développé au cours du second semestre par un groupe de quatre étudiants de M1 et encadré par un enseignant-chercheur et un intervenant de l'université.

L'objectif pédagogique du projet est de mettre en pratique les concepts abordés dans le module DevOps, notamment la conteneurisation, l'architecture logicielle, l'intégration continue et le déploiement continu, tout en s'appuyant sur un cas d'usage concret mêlant développement mobile et intelligence artificielle.

Sur le plan applicatif, ShrooML est une application Android dédiée à la reconnaissance de champignons à partir de photographies ou de descriptions de critères morphologiques. Elle repose sur un package python **AutoML** précédemment développé au premier semestre du même master, dans les modules Méthodologie pour l'IA et Méthodes Classiques en IA. L'application intègre également des fonctionnalités interactives et éducatives, telles que les possibles champignons poussant autour de l'utilisateur selon sa position géographique et conditions météorologiques, des quiz, des scores de confiance et des observations soumises par les utilisateurs.

1.3 Définitions, acronymes et abréviations

- **API** : *Application Programming Interface*, interface permettant la communication entre différents composants logiciels.
- **Docker** : Plateforme de conteneurisation permettant le déploiement d'applications de manière isolée et reproductible.
- **AutoML** : Pipeline de *Machine Learning* automatisé permettant l'entraînement et l'évaluation de modèles sans intervention manuelle extensive.
- **CI/CD** : *Continuous Integration / Continuous Deployment*, ensemble de pratiques visant à automatiser les phases de test, d'intégration et de déploiement.

— **MVP** : *Minimum Viable Product*, version minimale du produit intégrant les fonctionnalités essentielles.

2 Description Générale

2.1 Objectifs de l'application

L'application **ShrooML** a pour objectif principal de fournir un outil mobile permettant la reconnaissance de champignons à partir de photographies ou de descriptions de critères morphologiques, en s'appuyant sur des algorithmes de *Machine Learning*. L'application vise à proposer une identification rapide et accessible, tout en affichant des résultats accompagnés d'un score de confiance afin d'informer l'utilisateur sur la fiabilité de la prédiction.

Un objectif central du projet est de proposer une **interface intuitive et ergonomique**, adaptée à une utilisation sur appareil mobile. L'application doit permettre une prise en main simple, aussi bien pour des utilisateurs novices que pour des utilisateurs plus expérimentés, en minimisant le nombre d'actions nécessaires pour effectuer une reconnaissance.

ShrooML poursuit également un objectif **éducatif et de sensibilisation**. À travers des fonctionnalités telles que des quiz, des informations détaillées sur les espèces reconnues et des comparaisons entre espèces proches, l'application vise à renforcer les connaissances des utilisateurs sur la mycologie.

2.2 Périmètre fonctionnel

Fonctionnalités principales :
— Capture ou importation de photos
— Saisie manuelle de critères morphologiques
— Appel à l'API AutoML pour prédiction
— Consultation des résultats avec score de confiance
— Gestion des utilisateurs et sessions (connection / enregistrement manuel, ou via Facebook ou Google)
— Interaction avec API ShroomLoc pour proposition de champignons probables selon zone géographique, biome et conditions météorologiques
— Partage et export de résultats
— Quizz et contenu éducatif
— Système de ranking
— Personnalisation du profil

3 Exigences Fonctionnelles

Chaque exigence fonctionnelle est identifiée par un identifiant unique (Fx) et décrit une fonctionnalité attendue du système.

- **F1** : Capture de photo via l'appareil photo ou importation depuis la galerie de l'utilisateur.
- **F2** : Saisie manuelle de critères morphologiques (forme du chapeau, couleur, lamelles, pied, habitat, etc.).
- **F3** : Envoi des données (image ou critères morphologiques) à l'API AutoML pour analyse et prédiction.
- **F4** : Affichage des résultats de prédiction avec un score de confiance associé.
- **F5** : Consultation d'une fiche détaillée du champignon identifié (images, noms, description, comestibilité).
- **F6** : Interaction avec l'API ShroomLoc afin de proposer une liste de champignons probables en fonction de la zone géographique, du biome et des conditions météorologiques, avec possibilité de rafraîchir les résultats.
- **F7** : Gestion des utilisateurs et des sessions (inscription, connexion manuelle ou via Google/Facebook).
- **F8** : Personnalisation du profil utilisateur (photo, espèce favorite, historique des identifications).
- **F9** : Partage et export des résultats d'identification (image, prédiction, score de confiance).
- **F10** : Accès à des contenus éducatifs et à des quizz pour l'apprentissage de la mycologie.
- **F11** : Mise en place d'un système de classement (ranking) basé sur l'activité et les résultats des quizz.

FIGURE 2 – Extrait de la maquette illustrant certaines exigences fonctionnelles, notamment F1, F2 et F3.

Pour des raisons de lisibilité, la maquette complète n'est pas présentée ici. Le lien vers la maquette interactive sur Canva est disponible dans la section Annexes du cahier des charges.

4 Exigences Non-Fonctionnelles

Cette section décrit les contraintes non-fonctionnelles du système, notamment en termes de performances, sécurité, ergonomie, compatibilité, qualité logicielle, scalabilité et tests.

- **Performance** : L'application devra fournir un temps de réponse inférieur à 3 secondes pour l'affichage des résultats de prédiction, hors latence réseau exceptionnelle. Les appels aux APIs devront être optimisés afin de limiter la consommation de données et de batterie.
- **Sécurité** : Les échanges entre l'application mobile et les APIs devront être sécurisés via HTTPS. L'authentification des utilisateurs reposera sur des mécanismes de jetons (JWT). Les données sensibles (sessions, identifiants) ne devront pas être stockées en clair sur l'appareil.
- **UI / UX** : L'interface utilisateur devra être intuitive, accessible et cohérente avec les standards Android (Material Design). L'utilisateur devra pouvoir comprendre rapidement les résultats d'identification ainsi que leur niveau de fiabilité.
- **Compatibilité** : L'application devra être compatible avec les versions récentes du système Android (minimum Android 10), et s'adapter à différentes tailles d'écrans et résolutions.
- **Qualité du code** : Le code devra être structuré, documenté et respecter les

bonnes pratiques du développement Android et Java (nomenclature, modularité, séparation des responsabilités).
— **Scalabilité** : L'architecture backend devra permettre l'ajout futur de nouveaux utilisateurs, fonctionnalités et modèles d'intelligence artificielle sans remise en cause majeure de l'existant.
— **Tests** : Des tests unitaires et tests fonctionnels devront être réalisés afin de garantir la stabilité de l'application et la fiabilité des prédictions.

5 Contraintes Techniques

Dans le cadre de ce projet, plusieurs contraintes techniques ont été définies afin de guider la conception et l'implémentation de la solution.

5.1 Application mobile

L'application devra être développée sous forme native Android, sans recours aux frameworks multiplateforme ou facilitant le développement (tels que Flutter, React Native, etc.). Elle devra exploiter au maximum les fonctionnalités matérielles du smartphone, notamment :
— Système de localisation (GPS)
— Appareil photo
— Accéléromètre
D'autres capteurs du téléphone pourraient être utilisés pour de futures fonctionnalités.

Le langage de programmation autorisé pour le développement Android est Kotlin et Java. Dans le cadre de ce projet, le choix s'est porté sur Java que nous maîtrisons mieux.

5.2 Backend et intelligence artificielle

Le projet intègre un module d'apprentissage automatique destiné à l'identification des champignons à partir d'images et de critères morphologiques. Ce module repose sur un modèle AutoML entraîné et exposé via une API REST. Les traitements lourds (entraînement, prédiction, évaluation) sont déportés côté serveur afin de limiter la charge côté client mobile.

Le backend devra être conteneurisé (Docker) afin de faciliter le déploiement, la maintenance et l'évolutivité du système.

6 Architecture Système

L'architecture du système repose sur une application mobile Android communiquant avec un ensemble de services backend conteneurisés. Cette organisation vise

à assurer la modularité, la maintenabilité et l'évolutivité de la solution. Cette organisation se construit tel que :

— **App Android ShrooML** : ShrooML est le projet central et est développé selon le design pattern MVC (Model Vue Controller). Dans ce cadre, il contient la vue (le front-end) et les controllers (le back-end interne à l'application, celui qui interagit avec les APIs externes). Ainsi, ShrooML interagit principalement avec trois APIs : AutoML, ShroomLoc et Shroomer. Ces APIs permettent respectivement l'identification de champignons, la récupération de champignons probables selon la position de l'utilisateur et la gestion des utilisateurs.

— **API AutoML** : AutoML est un package Python développé dans le cadre des modules *Méthodologie pour l'IA* et *Méthodes Classiques en IA* au cours du premier semestre. Il a été transformé en API conteneurisée via Docker et déployé sur un serveur dédié. AutoML expose des fonctions de gestion des utilisateurs sécurisée (base de données, jetons JWT) ainsi que des fonctions de machine learning : entraînement d'un modèle (*fit*), prédiction (*predict*) et évaluation (*eval*).

— **API ShroomLoc** : ShroomLoc est une API développée au début du projet. Elle est conteneurisée et déployée sur un serveur dédié. Elle permet d'obtenir une liste de champignons probables autour de l'utilisateur via sa position géographique. Elle fait appel à deux APIs météorologiques (pour gestion du fallback) ainsi qu'à une API OpenStreetMap pour l'identification du biome. Enfin, elle utilise une API externe pour la récupération d'images des champignons. ShroomLoc est sécurisée et intègre une gestion des utilisateurs via jetons JWT.

— **API Shroomer** : Shroomer est une API en cours de développement permettant la gestion des utilisateurs, de leur classement ainsi que des événements quotidiens (synchronisés pour l'ensemble des utilisateurs).

7 Hypothèses et Dépendances

Le projet repose sur plusieurs hypothèses et dépendances externes :

— Qualité suffisante des images fournies par les utilisateurs pour permettre une identification fiable.
— Disponibilité continue des APIs backend (AutoML, ShroomLoc, Shroomer).
— Stabilité et performances du modèle AutoML entraîné.
— Dépendance à des services tiers : APIs météorologiques, OpenStreetMap, API d'images.

— Utilisation de bibliothèques Android standards et de dépendances Java éprouvées.

8 Tableau des Priorités et Roadmap

Les fonctionnalités sont organisées selon leur priorité de développement :

— **MVP (Minimum Viable Product)** :
  — Capture ou importation de photos
  — Prédiction via API AutoML
  — Affichage des résultats avec score de confiance
  — Interaction avec l'API ShroomLoc
  — Gestion basique des utilisateurs
— **Fonctionnalités secondaires (post-MVP)** :
  — Quizz et contenu éducatif
  — Système de ranking
  — Personnalisation avancée du profil
  — Partage et export des résultats

9 Critères d'Acceptation et Tests

Cette section décrit les critères de validation ainsi que les principaux types de tests réalisés afin de garantir le bon fonctionnement et la qualité de l'application.

9.1 Tests fonctionnels

Les tests fonctionnels vérifient que chaque exigence fonctionnelle est correctement implémentée.

— **F1** : L'utilisateur peut capturer ou importer une photo sans erreur.
— **F2** : Les critères morphologiques peuvent être saisis et transmis correctement.
— **F3** : Une prédiction est retournée par l'API AutoML.
— **F4** : Les résultats sont affichés avec un score de confiance.
— **F5** : Les champignons probables sont affichés et rafraîchissables via ShroomLoc.

9.2 Tests unitaires

Des tests unitaires sont réalisés sur les composants critiques de l'application (validation des entrées, contrôleurs, appels réseau).

9.3 Tests d'intégration

Les tests d'intégration vérifient la communication entre l'application Android et les APIs externes ainsi que la gestion de l'authentification.

9.4 Tests de performance

Les tests de performance évaluent le temps de réponse des prédictions et l'impact sur les ressources du téléphone.

Critère d'acceptation : le temps de réponse global ne doit pas dépasser 3 secondes.

9.5 Validation globale

L'application est considérée comme validée lorsque l'ensemble des fonctionnalités principales fonctionne sans erreur bloquante.

10 Annexes

— **Organisation du projet, repositories** :

— **Organisation GitHub ShrooML-Team :** `https://github.com/ShrooML-Team/`
— **Repo Application Android :** `https://github.com/ShrooML-Team/ShrooML`
— **Repo API AutoML :** `https://github.com/ShrooML-Team/AutoML`
— **Repo API ShroomLoc :** `https://github.com/ShrooML-Team/ShroomLoc`
— **Documentation ShrooML :** voir README.md du repo ShrooML.

— **Maquette** :

— **Maquette du projet** : `https://github.com/ShrooML-Team/Documents/blob/master/ShrooML_maquette_v2.pdf`

— **Jeux de données et modèles AutoML** :

— Kaggle, dataset de correspondance critères - commestibilité : `https://www.kaggle.com/datasets/uciml/mushroom-classification`
— Kaggle, dataset de correspondance images - espèce : `https://www.kaggle.com/datasets/thehir0/mushroom-species`