Cahier des Charges - Application Mobile Nutrition & Santé

1. Présentation du Projet

1.1 Vue d'ensemble

Développement d'une application mobile complète de nutrition et santé utilisant l'intelligence artificielle pour offrir une expérience personnalisée aux utilisateurs. L'application combine analyse nutritionnelle, suivi personnalisé, et services de proximité.

1.2 Objectifs principaux

- Fournir un assistant nutritionnel IA personnalisé
- Analyser automatiquement les aliments par image et code-barres
- Proposer des recettes adaptées aux besoins nutritionnels
- Offrir un tableau de bord complet de suivi santé
- Localiser les services de santé de proximité
- Collecter et analyser les données pour améliorer l'expérience utilisateur

1.3 Technologies cibles

- **Frontend** : Flutter (iOS/Android)
- **Backend** : Node.js avec Express.js
- **Infrastructure** : Cloud-based Data Lakehouse
- **IA** : Services de machine learning intégrés

2. Spécifications Fonctionnelles Détaillées

2.1 Module 1 : Chatbot Nutritionniste IA

2.1.1 Fonctionnalités principales

- **Chat textuel intelligent** : Conversation naturelle avec l'IA nutritionniste
- **Communication vocale** : Support audio bidirectionnel (speech-to-text et text-to-speech)
- **Conversation proactive** : L'IA peut initier des conversations basées sur les habitudes utilisateur
- **Personnalisation** : Adaptation du discours selon le profil nutritionnel de l'utilisateur

2.1.2 Fonctionnalités avancées

- Historique des conversations avec recherche
- Suggestions de questions fréquentes
- Mode consultation spécialisée (régimes, allergies, objectifs sportifs)

- Notifications intelligentes pour rappels nutritionnels
- Intégration avec le dashboard pour contextualiser les conseils

2.1.3 Spécifications techniques

- API de traitement du langage naturel (OpenAI GPT ou équivalent)
- Service de reconnaissance vocale (Google Speech-to-Text)
- Service de synthèse vocale (Google Text-to-Speech)
- Système de cache intelligent pour réduire la latence
- Chiffrement end-to-end des conversations

2.2 Module 2 : Analyse d'Image IA

2.2.1 Fonctionnalités principales

- **Reconnaissance automatique** : Identification des aliments dans une image
- **Estimation des portions** : Calcul approximatif des quantités
- **Analyse nutritionnelle** : Calcul automatique des macros et micronutriments
- **Correction manuelle** : Interface permettant d'ajuster les portions et ingrédients
- **Historique photo** : Sauvegarde des analyses précédentes

2.2.2 Fonctionnalités avancées

- Reconnaissance multi-aliments dans une seule image
- Détection de la vaisselle pour améliorer l'estimation des portions
- Comparaison avec des références visuelles
- Mode batch pour analyser plusieurs images simultanément
- Suggestions d'amélioration nutritionnelle basées sur l'analyse

2.2.3 Spécifications techniques

- Modèle de vision par ordinateur (TensorFlow/PyTorch)
- API de reconnaissance d'objets spécialisée nutrition
- Compression et optimisation des images
- Base de données nutritionnelle comprehensive
- Algorithmes d'estimation de volume 3D

2.3 Module 3 : Scanner de Codes-Barres

2.3.1 Fonctionnalités principales

- **Scan instantané** : Reconnaissance rapide des codes-barres/QR codes

- **Base de données produits** : Accès à une base étendue de produits alimentaires
- **Informations nutritionnelles** : Affichage détaillé des valeurs nutritionnelles
- **Évaluation santé** : Score de qualité nutritionnelle (Nutri-Score, etc.)
- **Alertes personnalisées** : Avertissements selon les restrictions alimentaires

2.3.2 Fonctionnalités avancées

- Reconnaissance de codes endommagés ou partiels
- Mode hors-ligne pour les produits courants
- Comparaison avec des alternatives plus saines
- Historique des produits scannés
- Ajout de produits manquants par la communauté

2.3.3 Spécifications techniques

- Bibliothèque de scan optimisée (ZXing ou ML Kit)
- API externe pour base de données produits (Open Food Facts)
- Cache local intelligent
- Système de mise à jour incrémentielle
- API RESTful pour gestion des produits

2.4 Module 4 : Bibliothèque de Recettes

2.4.1 Fonctionnalités principales

- **Catalogue étendu** : Large sélection de recettes nutritionnellement équilibrées
- **Filtres avancés** : Recherche par ingrédients, régime, temps de préparation, etc.
- **Personnalisation** : Suggestions basées sur les préférences et objectifs
- **Instructions détaillées** : Étapes illustrées avec photos/vidéos
- **Calcul nutritionnel** : Valeurs nutritionnelles automatiques par portion

2.4.2 Système de filtres

- **Restrictions alimentaires** : Végétarien, vegan, sans gluten, etc.
- **Objectifs nutritionnels** : Perte de poids, prise de masse, maintien
- **Temps de préparation** : Rapide (<15min), moyen (15-45min), élaboré (>45min)
- **Difficulté** : Débutant, intermédiaire, expert
- **Type de repas** : Petit-déjeuner, déjeuner, dîner, collation
- **Saison** : Ingrédients de saison disponibles

- **Budget** : Économique, moyen, premium

2.4.3 Fonctionnalités sociales

- Sauvegarde des recettes favorites
- Notation et commentaires utilisateurs
- Partage sur réseaux sociaux
- Liste de courses automatique
- Planificateur de repas hebdomadaire

2.5 Module 5 : Géolocalisation des Pharmacies

2.5.1 Fonctionnalités principales

- **Localisation automatique** : Géolocalisation GPS de l'utilisateur
- **Pharmacies de proximité** : Affichage des pharmacies dans un rayon configurable
- **Horaires en temps réel** : Statut ouvert/fermé avec horaires détaillés
- **Services de garde** : Identification des pharmacies de garde/nuit
- **Navigation intégrée** : Intégration avec les applications de cartographie

2.5.2 Fonctionnalités avancées

- Filtres par services spéciaux (orthopédie, homéopathie, etc.)
- Disponibilité de médicaments spécifiques
- Évaluations et avis utilisateurs
- Réservation de médicaments (si possible)
- Alertes de changement d'horaires de garde

2.5.3 Spécifications techniques

- API de géolocalisation (Google Maps/OpenStreetMap)
- Base de données des pharmacies avec mise à jour régulière
- API d'horaires en temps réel
- Cache géographique intelligent
- Optimisation pour consommation batterie réduite

2.6 Module 6 : Dashboard Nutritionnel et Santé

2.6.1 KPIs Principaux

- **Apports nutritionnels quotidiens** : Calories, protéines, glucides, lipides
- **Micronutriments** : Vitamines, minéraux, fibres

- **Hydratation** : Suivi de la consommation d'eau
- **Objectifs personnalisés** : Progression vers les objectifs fixés
- **Tendances temporelles** : Évolution sur 7, 30, 90 jours

2.6.2 Visualisations avancées

- **Graphiques interactifs** : Courbes d'évolution, diagrammes circulaires
- **Cartes de chaleur** : Identification des patterns alimentaires
- **Comparaisons** : Objectifs vs réalisé, recommandations nutritionnelles
- **Prédictions** : Projections basées sur les tendances actuelles
- **Scores de santé** : Indices synthétiques personnalisés

2.6.3 Fonctionnalités d'analyse

- Détection d'anomalies nutritionnelles
- Suggestions d'amélioration personnalisées
- Rapports exportables (PDF, CSV)
- Partage avec professionnels de santé
- Mode comparaison avec moyennes populationnelles

3. Architecture Technique

3.1 Architecture Frontend (Flutter)

3.1.1 Structure modulaire

```
lib/
├── core/
│   ├── constants/
│   ├── utils/
│   ├── errors/
│   └── network/
├── features/
│   ├── chatbot/
│   ├── image_analysis/
│   ├── barcode_scanner/
│   ├── recipes/
│   ├── pharmacy_locator/
│   └── dashboard/
├── shared/
│   ├── widgets/
│   ├── services/
│   └── repositories/
└── main.dart
```

3.1.2 Packages principaux

- **State Management** : Bloc/Cubit ou Riverpod
- **Navigation** : Auto Route ou Go Router
- **Networking** : Dio avec interceptors
- **Local Storage** : Hive ou Sembast
- **Image Processing** : image_picker, image_cropper
- **Charts** : fl_chart pour les visualisations
- **Maps** : google_maps_flutter
- **Audio** : speech_to_text, flutter_tts
- **Camera** : camera, mobile_scanner

3.2 Architecture Backend (Node.js)

3.2.1 Structure microservices

```
services/
├── api-gateway/          # Point d'entrée principal
├── auth-service/         # Authentification et autorisation
├── user-service/         # Gestion des utilisateurs et profils
├── nutrition-service/    # Analyse nutritionnelle et calculs
├── ai-service/           # Services IA (chatbot, analyse d'image)
├── recipe-service/       # Gestion des recettes
├── location-service/     # Géolocalisation et pharmacies
├── analytics-service/    # Traitement des données et analytics
└── notification-service/ # Notifications push et email
```

3.2.2 Technologies backend

- **Framework** : Express.js avec TypeScript
- **Base de données** : MongoDB (documents) + PostgreSQL (relationnel)
- **Cache** : Redis pour les sessions et cache
- **Message Queue** : Bull/Agenda pour les tâches asynchrones
- **IA** : OpenAI API, TensorFlow.js, Hugging Face
- **Stockage** : AWS S3 ou Google Cloud Storage
- **Monitoring** : Winston pour logs, Prometheus pour métriques

3.3 Data Lakehouse Architecture

3.3.1 Couches de données

- **Bronze Layer** : Données brutes (logs, événements, images)
- **Silver Layer** : Données nettoyées et structurées
- **Gold Layer** : Données agrégées pour analytics et ML

3.3.2 Technologies de traitement

- **Stream Processing** : Apache Kafka + Apache Spark
- **Batch Processing** : Apache Airflow pour orchestration
- **Storage** : Delta Lake sur cloud storage
- **Analytics** : Apache Superset pour BI
- **ML Pipeline** : MLflow pour versioning des modèles

4. Modèle de Données

4.1 Entités principales

4.1.1 Utilisateur

json

```
{
  "userId": "uuid",
  "profile": {
    "personalInfo": {
      "age": "number",
      "gender": "string",
      "height": "number",
      "weight": "number",
      "activityLevel": "enum"
    },
    "nutritionalGoals": {
      "targetCalories": "number",
      "macroRatios": "object",
      "restrictions": "array"
    },
    "preferences": {
      "language": "string",
      "units": "metric|imperial",
      "notifications": "object"
    }
  },
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}
```

4.1.2 Aliment/Produit

json

```
{
  "foodId": "uuid",
  "barcode": "string",
  "name": "string",
  "brand": "string",
  "nutritionalInfo": {
    "per100g": {
      "calories": "number",
      "protein": "number",
      "carbs": "number",
      "fat": "number",
      "fiber": "number",
      "sodium": "number",
      "vitamins": "object",
      "minerals": "object"
    }
  },
  "categories": "array",
  "allergens": "array",
  "nutriScore": "string",
  "images": "array"
}
```

4.1.3 Consommation

json

```
{
  "consumptionId": "uuid",
  "userId": "uuid",
  "foodId": "uuid",
  "quantity": "number",
  "unit": "string",
  "meal": "breakfast|lunch|dinner|snack",
  "timestamp": "timestamp",
  "source": "scan|image|recipe|manual",
  "confidence": "number"
}
```

4.2 Base de données nutritionnelle

- Integration avec Open Food Facts API
- Base propriétaire enrichie pour produits locaux
- Système de validation communautaire

- Mise à jour automatique des informations produits

5. Interface Utilisateur et Expérience

5.1 Principes de design

- **Material Design 3** pour Android, **Cupertino** pour iOS
- **Design responsive** s'adaptant aux différentes tailles d'écran
- **Accessibilité** conforme aux standards WCAG 2.1
- **Dark/Light mode** avec détection automatique
- **Animations fluides** pour améliorer l'expérience

5.2 Navigation principale

- **Bottom Navigation Bar** avec 5 onglets principaux
- **Floating Action Button** pour ajout rapide d'aliments
- **Search Bar** accessible depuis tous les écrans
- **Profile/Settings** accessible via menu hamburger

5.3 Écrans critiques

5.3.1 Dashboard (Écran principal)

- Vue d'ensemble des KPIs du jour
- Graphiques de progression
- Recommandations personnalisées
- Accès rapide aux fonctions principales

5.3.2 Chatbot

- Interface de chat moderne
- Bouton vocal facilement accessible
- Suggestions de questions
- Historique recherchable

5.3.3 Analyse d'image

- Caméra plein écran avec overlay informatif
- Prévisualisation avant analyse
- Interface de correction intuitive
- Résultats visuellement attractifs

6. Sécurité et Conformité

6.1 Sécurité des données

- **Chiffrement** : AES-256 pour le stockage, TLS 1.3 pour les transmissions
- **Authentification** : OAuth 2.0 + JWT avec refresh tokens
- **Autorisation** : RBAC (Role-Based Access Control)
- **Audit** : Logs de toutes les actions sensibles
- **Backup** : Sauvegardes chiffrées automatiques

6.2 Conformité réglementaire

- **RGPD** : Consentement explicite, droit à l'oubli, portabilité des données
- **Réglementation santé** : Conformité aux standards de dispositifs médicaux
- **Protection des mineurs** : Validation d'âge et contrôles parentaux
- **Données de santé** : Anonymisation et pseudonymisation

6.3 Gestion des permissions

- **Localisation** : Demande d'autorisation explicite
- **Caméra/Microphone** : Permissions granulaires
- **Notifications** : Opt-in avec personnalisation
- **Contacts** : Uniquement si partage activé

7. Performance et Scalabilité

7.1 Optimisations frontend

- **Lazy loading** des écrans et composants
- **Image optimization** avec mise en cache intelligente
- **State management** optimisé pour réduire les re-rendus
- **Bundle splitting** pour réduire la taille de l'app

7.2 Optimisations backend

- **Mise en cache** multi-niveau (Redis, CDN)
- **Rate limiting** pour prévenir les abus
- **Connection pooling** pour les bases de données
- **Compression** automatique des réponses API

7.3 Scalabilité

- **Auto-scaling** basé sur la charge
- **Load balancing** avec health checks
- **Database sharding** par région utilisateur
- **CDN global** pour les assets statiques

8. Testing et Qualité

8.1 Stratégie de tests

- **Unit tests** : Couverture minimale 80%
- **Integration tests** : Tests API et base de données
- **Widget tests** : Tests Flutter pour l'UI
- **E2E tests** : Parcours utilisateur critiques

8.2 Qualité du code

- **Linting** : ESLint (backend), Dart Analysis (frontend)
- **Code formatting** : Prettier, Dart Format
- **Code review** : Pull requests obligatoires
- **Static analysis** : SonarQube pour la qualité

8.3 Monitoring en production

- **APM** : Application Performance Monitoring
- **Error tracking** : Sentry ou équivalent
- **Analytics** : Firebase Analytics + analytics personnalisées
- **Crash reporting** : Firebase Crashlytics

9. DevOps et Déploiement

9.1 CI/CD Pipeline

```yaml
Développement → Tests → Build → Deploy Staging → Tests E2E → Deploy Production
```

9.2 Environnements

- **Development** : Environnement local avec données de test
- **Staging** : Miroir de production pour tests
- **Production** : Environnement live avec monitoring complet

9.3 Infrastructure as Code

- **Terraform** pour provisioning cloud
- **Docker** pour containerisation
- **Kubernetes** pour orchestration
- **Helm** pour déploiements

10. Analytics et Business Intelligence

10.1 Métriques produit

- **User engagement** : DAU, MAU, session duration
- **Feature adoption** : Utilisation par module
- **Retention** : Taux de rétention par cohorte
- **Health metrics** : Précision des analyses IA

10.2 Métriques business

- **Acquisition** : Sources de trafic, coût d'acquisition
- **Conversion** : Funnel d'onboarding
- **Revenue** : Si modèle freemium/premium
- **Satisfaction** : NPS, ratings app stores

10.3 Data Science

- **Modèles prédictifs** : Prédiction des habitudes alimentaires
- **Recommandation** : Système de recommandation de recettes
- **Segmentation** : Clustering d'utilisateurs pour personnalisation
- **A/B Testing** : Tests de nouvelles fonctionnalités

11. Roadmap et Phases de Développement

Phase 1 : MVP (3 mois)

- Authentification et profil utilisateur
- Scanner de codes-barres basique
- Dashboard simple avec KPIs essentiels
- Base de données nutritionnelle

Phase 2 : IA Core (2 mois)

- Chatbot nutritionniste IA
- Analyse d'image basique
- Amélioration du dashboard

Phase 3 : Features Avancées (2 mois)

- Module recettes complet
- Géolocalisation pharmacies
- Fonctionnalités sociales
- Optimisations performance

Phase 4 : Intelligence & Analytics (1 mois)

- Data lakehouse complet
- Analytics avancées
- Machine learning personnalisé
- Système de recommandations

12. Ressources et Budget

12.1 Équipe technique requise

- **1 Tech Lead** (Full-stack)
- **2 Développeurs Flutter** (Mobile)
- **2 Développeurs Node.js** (Backend)
- **1 Data Engineer** (Analytics/ML)
- **1 DevOps Engineer**
- **1 UI/UX Designer**
- **1 QA Engineer**

12.2 Infrastructure estimée

- **Cloud computing** : $500-2000/mois selon usage
- **Services IA** : $200-1000/mois selon volume
- **Bases de données** : $100-500/mois
- **CDN et stockage** : $50-200/mois
- **Monitoring et tools** : $100-300/mois

12.3 Licences et APIs

- **OpenAI API** : Variable selon usage
- **Google Maps API** : $200/mois pour 100k requêtes
- **Open Food Facts** : Gratuit
- **Services cloud** : Pricing standard AWS/GCP/Azure

13. Risques et Mitigation

13.1 Risques techniques

- **Précision IA** : Tests extensifs, amélioration continue
- **Performance** : Monitoring proactif, optimisations
- **Compatibilité** : Tests sur multiples devices
- **Sécurité** : Audits réguliers, penetration testing

13.2 Risques business

- **Adoption utilisateur** : Beta testing, feedback continu
- **Concurrence** : Différenciation par l'IA et l'UX
- **Réglementation** : Veille juridique, conformité proactive
- **Monétisation** : Modèle freemium équilibré

13.3 Risques opérationnels

- **Retention équipe** : Bonnes pratiques, documentation
- **Scalabilité** : Architecture cloud-native
- **Maintenance** : Code quality, tests automatisés
- **Support client** : Documentation utilisateur, FAQ

14. Conclusion

Cette application mobile de nutrition et santé représente un projet ambitieux combinant intelligence artificielle, analyse de données, et expérience utilisateur premium. Le succès dépendra de l'exécution technique rigoureuse, de la précision des algorithmes IA, et de la capacité à créer une expérience utilisateur engageante et personnalisée.

La roadmap proposée permet un développement itératif avec validation continue du marché, tout en construisant une architecture solide pour supporter la croissance future. L'accent sur la collecte et l'analyse des données positionnera l'application pour des améliorations continues et une personnalisation de plus en plus précise.

Prochaines étapes :

1. Validation du cahier des charges par l'équipe
2. Setup de l'environnement de développement
3. Démarrage Phase 1 avec sprint planning détaillé
4. Mise en place des métriques et monitoring dès le début