## **Cahier des Charges Fonctionnel et Technique**

### **Amélioration de l'Accessibilité PMR en Milieu Urbain**

Date : 19 Juin 2025

Version : 1.0

Auteur : gregory Le Terte (Développeur IA)

[**Cahier des Charges   
Fonctionnel et Technique 1**](#_xupo6vetx8ay)

[Amélioration de l'Accessibilité PMR en Milieu Urbain 1](#_6kyo9okldi90)

[1. Introduction et Contexte du Projet 3](#_yhrpny1iw4d4)

[2. Objectifs du Projet 4](#_bx3x3etmddrv)

[4. Architecture Technique Détaillée 6](#_rid8ox1cf4mu)

[5. Organisation du Projet et Livrables 13](#_szxj64c4bqw8)

[6. Critères de Succès et Validation 14](#_3otb6jv12jjo)

### **1. Introduction et Contexte du Projet**

1.1 Contexte Général :

L'accessibilité pour les Personnes à Mobilité Réduite (PMR) est un enjeu sociétal, réglementaire et économique majeur en France et en Europe. La Loi 2005-102 et l'objectif d'accessibilité des transports d'ici 2025 soulignent ces obligations légales. Malgré cela, des défis de conformité persistent, avec, par exemple, 40% des Établissements Recevant du Public (ERP) non conformes en 2025 et des critiques de l'ONU. Les Jeux Olympiques et Paralympiques de Paris 2024 ont par ailleurs accéléré les initiatives en la matière.

1.2 Problématique Identifiée :

Le manque d'informations précises, dynamiques et personnalisées sur l'accessibilité des points d'intérêt urbains (transports, commerces, loisirs) limite l'autonomie et la capacité de planification des PMR. Les informations actuelles sont souvent statiques ou binaires ("accessible/non accessible") et ne reflètent pas la complexité réelle du terrain.

1.3 Rôle de l'IA :

L'Intelligence Artificielle offre des opportunités uniques pour aller au-delà des solutions actuelles et fournir des informations plus intelligentes et proactives, renforçant l'autonomie des PMR.

### 

### **2. Objectifs du Projet**

2.1 Objectif Général :

Développer une solution d'IA capable de prédire et de visualiser la difficulté d'accessibilité PMR de points d'intérêt urbains pour une meilleure planification et une autonomie accrue des PMR.

**2.2 Objectifs Spécifiques (pour la certification RNCP38616) :**

* **Bloc 03 - Machine Learning :**
  + **Classification d'Accessibilité ML :** Prédire le niveau de difficulté d'accessibilité PMR pour un point d'intérêt donné, selon 3 catégories : "Facilement Accessible", "Modérément Accessible", "Difficilement Accessible".
  + **Sélection et Justification d'Algorithme ML :** Choisir l'algorithme d'apprentissage automatique le plus adapté (ex: Régression Logistique, Random Forest) en comparant les performances et les caractéristiques des différentes familles d'algorithmes
  + **Évaluation de Performance ML :** Évaluer la performance de l'algorithme d'apprentissage automatique et du modèle en utilisant des métriques pertinentes (accuracy, précision, rappel, F1-score) et une matrice de confusion.
  + **Identification des Facteurs Clés :** Déterminer quelles caractéristiques d'un lieu influencent le plus sa difficulté d'accessibilité.
* **Bloc 05 - Deep Learning :**
  + **Classification d'Accessibilité DL :** Prédire le niveau de difficulté d'accessibilité PMR en utilisant un algorithme de Deep Learning.
  + **Préparation de Données Non Structurées :** Préparer des données non structurées (images) en les convertissant en données numériques et sous forme tabulaire pour le modèle DL.
  + **Sélection et Justification d'Algorithme DL :** Choisir l'algorithme d'apprentissage profond le plus adapté en comparant ses performances et caractéristiques, notamment par rapport au modèle ML.
  + **Évaluation de Performance DL :** Évaluer la performance du modèle DL avec les mêmes métriques que le ML, et comparer les deux approches.
  + **Déploiement du Modèle DL :** Démontrer le déploiement efficace du modèle de Deep Learning via une API web, en utilisant des outils et plateformes de production adaptés (MLOps).

**3. Cas d'Usage et Fonctionnalités Clés de l'Application (Maquette fournie)**

3.1 Fonctionnement Global :

L'application web, telle qu'illustrée par les maquettes, servira d'interface pour interagir avec les modèles d'IA. Elle prendra en entrée des informations sur un point d'intérêt urbain et, optionnellement, une image de son entrée, et fournira en sortie une prédiction sur sa difficulté d'accessibilité.

**3.2 Fonctionnalités Détaillées :**

* **1. Prédiction de Difficulté d'Accès (Données Structurées - ML & DL) :**
  + **Saisie et Recherche du Lieu :** Un champ de texte permettra à l'utilisateur de saisir un lieu (ex: "Tour Eiffel") et un bouton "Rechercher sur la carte" simulera la sélection du lieu, affichant son nom et adresse (ex: "Tour Eiffel, Champ de Mars..."). Pour la démonstration, les caractéristiques du lieu seront des données tabulaires pré-définies ou extraites de votre dataset synthétique.
  + **Déclenchement de l'Analyse :** Un bouton "Analyser l'accessibilité" enverra les caractéristiques du lieu à l'API backend.
  + **Affichage des Résultats :** Les résultats comprendront :
    - Le **Niveau de difficulté** prédit (ex: "Modérément Accessible").
    - Les **Probabilités** pour chaque catégorie (Facilement Accessible, Modérément Accessible, Difficilement Accessible).
    - Une **Interprétation textuelle** concise décrivant la difficulté et les raisons sous-jacentes (ex: "Accès au rez-de-chaussée facile, mais certaines zones peuvent présenter des obstacles ponctuels.").
* **2. Analyse d'Image pour l'Accessibilité (Données Non Structurées - DL) :**
  + **Téléversement d'Image :** Une section dédiée "Analyse d'Image pour l'Accessibilité (Optionnel)" avec un bouton "Choisir une photo" permettra à l'utilisateur de téléverser une image de l'entrée d'un lieu (ex: "slider-1-shop-ramp.jpg").
  + **Déclenchement de l'Analyse Visuelle :** Un bouton "Analyser l'image" enverra l'image à l'API backend.
  + **Affichage du Résultat Visuel :** L'application affichera l'image téléversée et un "Résultat de l'analyse visuelle" (ex: "Facilement Accessible", "Aucun obstacle majeur détecté visuellement"). Cet aspect répond à l'objectif de "Préparer des données non structurées" du Bloc 05.

### 

### **4. Architecture Technique Détaillée**

**4.1 Préparation des Données : La Pierre Angulaire du Projet (Accent sur la certification)**

* **4.1.1 Source des Données :**
  + **Origine :** Le dataset sera **synthétique**. Cela permet une maîtrise totale du processus de génération, assurant la diversité des cas et la pertinence des labels, éléments clés pour la démonstration.
  + **Logique de Création du Dataset Structuré :**
    - **Caractéristiques (Features) :** Générer les valeurs pour les attributs suivants, en veillant à la diversité et au réalisme des données :
    - Choisir 2 criteres Maxi parmi les catégories ci dessous :
      * **Descriptives possibles :**
        + largeur\_trottoir\_cm (numérique, float): Exprimée en centimètres (ex: 50.0 à 250.0).
        + pente\_acces\_degres (numérique, float): Exprimée en degrés (ex: 0.0 à 20.0).
        + distance\_transport\_m (numérique, int): Distance au point de transport le plus proche en mètres (ex: 0 à 1000).
        + nombre\_obstacles\_fixes (numérique, int): Nombre d'obstacles permanents (poteaux, poubelles) devant l'entrée (ex: 0 à 5).
      * **Infrastructurelles :**
        + presence\_ascenseur (catégorielle, bool): True/False.
        + etat\_ascenseur (catégorielle, str): "fonctionnel", "en\_panne\_occasionnelle", "hors\_service".
        + type\_revetement\_sol (catégorielle, str): "lisse" (carrelage, asphalte), "pavé", "gravier", "terre".
        + age\_infrastructure\_an (numérique, int): Ancienneté approximative en années (ex: 0 à 100).
      * **Contextuelles :**
        + type\_lieu (catégorielle, str): "commerce", "musée", "gare", "restaurant", "hôpital", "hôtel", "mairie", "école".
        + type\_transport\_proximite (catégorielle, str): "bus", "métro", "tram", "rer", "aucun".
        + zone\_geographique (catégorielle, str): "centre\_historique", "résidentiel", "affaires", "périphérie".
        + affluence\_heure\_pointe (catégorielle, str): "faible", "moyenne", "forte".
    - **Classe Cible (Label) :** difficulté accessibilité (catégorielle, str): "Facilement Accessible", "Modérément Accessible", "Difficilement Accessible".
    - **Règles de Génération des Labels :** Définir des fonctions Python claires et des règles logiques pour assigner un label d'accessibilité à chaque ensemble de caractéristiques. Ces règles simuleront la complexité du monde réel, permettant de créer des cas ambigus ou des exceptions pour rendre le dataset plus riche. Par exemple, une largeur\_trottoir\_cm élevée associée à une pente\_acces\_degres faible pourrait donner "Facilement Accessible", tandis qu'un type\_revetement\_sol "gravier" augmenterait la difficulté.
  + **Volume du Dataset Structuré :** Générer entre 3000 et 6000 exemples au total. Il est recommandé d'introduire un léger déséquilibre (par exemple, 40% Facile, 35% Modéré, 25% Difficile) pour démontrer la capacité à gérer des classes minoritaires (souvent les plus critiques pour la détection).
  + **Données Non Structurées (Images - pour Bloc 05) :** Si l'analyse d'image est incluse (fortement recommandé), collecter ou générer un petit jeu d'images (100-200 images, idéalement 30-50 par classe si le temps le permet) représentant les différentes difficultés d'accès visuelles (ex: présence de marches, de rampes, d'obstacles, de portes automatiques, etc.). Chaque image sera associée à un label d'accessibilité visuelle (ex: "rampe présente", "marches visibles", "porte étroite"). Ces images serviront à entraîner un modèle de Computer Vision (CNN).
* **4.1.2 Nettoyage et Prétraitement des Données (Critique pour la certification - C2 Bloc 03, C1 Bloc 05) :**
  + **Gestion des Valeurs Manquantes :**
    - **Stratégie :** Utilisation de sklearn.impute.SimpleImputer.
    - **Justification :** Pour les colonnes numériques, la stratégie median est préférée à la moyenne car elle est moins sensible aux valeurs aberrantes. Pour les colonnes catégorielles, la stratégie most\_frequent (mode) est appropriée. Démontrez que vous savez identifier les valeurs manquantes et appliquer une stratégie justifiée.
    - **Implémentation :** Code Python clair montrant l'application de l'imputation sur les données brutes.
  + **Encodage des Variables Catégorielles :**
    - **Méthode :** Utilisation de sklearn.preprocessing.OneHotEncoder.
    - **Justification :** Convertit les variables catégorielles en un format numérique compréhensible par les modèles ML/DL sans introduire de relation d'ordre arbitraire.
    - **Implémentation :** Montrer un exemple de transformation (type\_lieu transformé en type\_lieu\_commerce, type\_lieu\_musée, etc.).
  + **Standardisation des Variables Numériques :**
    - **Méthode :** Utilisation de sklearn.preprocessing.StandardScaler.
    - **Justification :** Centre les données autour de zéro et les met à l'échelle pour avoir une variance unitaire. C'est essentiel pour les algorithmes sensibles à l'échelle des *features* (comme les réseaux de neurones, les SVM et la régression logistique) pour garantir une convergence rapide et des performances optimales.
    - **Implémentation :** Appliquer StandardScaler après l'encodage si nécessaire.
  + **Division des Données :**
    - **Méthode :** Utilisation de sklearn.model\_selection.train\_test\_split.
    - **Proportions :** Découper en 3 ensembles : 70% pour l'entraînement, 15% pour la validation (utilisé pour le *tuning* des hyperparamètres et l'EarlyStopping), et 15% pour le test (pour l'évaluation finale et objective).
    - **Stratification :** Utiliser l'argument stratify=y pour s'assurer que la proportion des classes cibles est maintenue de manière égale dans chaque ensemble (entraînement, validation, test). C'est crucial pour les datasets déséquilibrés et pour garantir que le modèle est évalué sur une distribution de classes réaliste.
  + **Prétraitement Spécifique aux Images (si inclus - C1 Bloc 05) :**
    - **Redimensionnement :** Utiliser tf.image.resize pour uniformiser toutes les images à une taille standard (ex: 224x224 pixels, taille d'entrée attendue par MobileNetV2).
    - **Normalisation des pixels :** Appliquer la normalisation spécifique au modèle pré-entraîné (ex: tf.keras.applications.mobilenet\_v2.preprocess\_input pour transformer les valeurs de pixels de [0, 255] à [-1, 1]).
    - **Augmentation de données :** Pour l'entraînement (uniquement sur l'ensemble d'entraînement), appliquer des techniques d'augmentation (tf.keras.layers.RandomFlip, RandomRotation, RandomZoom) pour augmenter artificiellement la taille du dataset et améliorer la généralisation du modèle, réduisant le surapprentissage.

**4.2 Modélisation :**

* **4.2.1 Modèle Machine Learning (Bloc 03) :**
  + **Choix de l'Algorithme :**
    - **Baseline : Régression Logistique :** Justifier sa simplicité, son interprétabilité et sa rapidité d'entraînement pour établir une première performance de référence.
    - **Modèle plus Performant : Random Forest Classifier :** Justifier sa robustesse aux données non linéaires, sa capacité à gérer de nombreuses *features* et son importance des *features* pour l'interprétabilité.
  + **Implémentation :** Utilisation de sklearn.linear\_model.LogisticRegression et sklearn.ensemble.RandomForestClassifier.
  + **Hyperparamètres :** Définir et justifier les valeurs clés utilisées pour chaque modèle (ex: C pour la Régression Logistique ; n\_estimators, max\_depth, min\_samples\_leaf pour le Random Forest).
  + **Entraînement :** Entraîner le modèle sur l'ensemble d'entraînement (model\_ml.fit(X\_train, y\_train)).
* **4.2.2 Modèle Deep Learning (Bloc 05) :**
  + **Architecture du Réseau :**
    - **Pour données tabulaires uniquement :** Réseau de neurones Fully Connected (ANN) avec plusieurs couches Dense. Ex: Input Layer -> Dense (ReLU) -> Dropout(0.2) -> Dense (ReLU) -> Dropout(0.2) -> Dense (Softmax, 3 neurones pour les 3 classes). Justifier le nombre de couches et de neurones par rapport à la complexité du problème.
    - **Pour données images (si inclus) :** Utiliser une architecture basée sur le **Transfer Learning** avec un Convolutional Neural Network (CNN) pré-entraîné comme **MobileNetV2** (ou un autre modèle léger compatible). Le modèle de base (base\_model) sera initialement gelé (base\_model.trainable = False) pour exploiter les *features* génériques d'ImageNet. Des couches de *pooling* (GlobalAveragePooling2D) et des couches Dense avec Dropout seront ajoutées pour la classification des 3 niveaux d'accessibilité.
  + **Implémentation :** Utilisation de TensorFlow/Keras.
  + **Compilation :**
    - **Optimiseur :** tf.keras.optimizers.Adam.
    - **Fonction de perte :** sparse\_categorical\_crossentropy (adaptée aux labels entiers et classification multi-classes).
    - **Métriques :** accuracy, tf.keras.metrics.Precision, tf.keras.metrics.Recall. Le F1-score sera calculé manuellement ou via Scikit-learn après l'inférence.
  + **Entraînement :** model\_dl.fit(ds\_train\_processed, epochs=EPOCHS, validation\_data=ds\_val\_processed).
    - **Callbacks (Importants pour l'optimisation) :**
      * tf.keras.callbacks.EarlyStopping : Surveille la val\_loss ou val\_accuracy avec une patience (ex: 5 époques) pour arrêter l'entraînement lorsque la performance cesse de s'améliorer, évitant le surapprentissage et économisant des ressources.
      * tf.keras.callbacks.ReduceLROnPlateau : Surveille la val\_loss et réduit le learning\_rate (ex: par un factor de 0.2) si la perte de validation stagne pendant une certaine patience (ex: 3 époques).
      * tf.keras.callbacks.ModelCheckpoint : Sauvegarde automatiquement les meilleurs poids du modèle basés sur la val\_accuracy.
    - **Stratégie de Fine-tuning (si CNN pré-entraîné) :** Démontrer clairement deux phases d'entraînement : une première phase avec les couches du modèle de base gelées (transfer learning), puis une seconde phase où les dernières couches du modèle de base sont dégelées et entraînées avec un taux d'apprentissage très faible (fine-tuning).

**4.3 Évaluation et Interprétation :**

* **Métriques d'Évaluation :** Pour les deux modèles (ML et DL), sur l'ensemble de test :  
  + **Accuracy :** Précision globale des prédictions.
  + **Precision (par classe) :** Proportion de vrais positifs parmi toutes les prédictions positives pour cette classe. Utile pour comprendre la qualité des prédictions spécifiques.
  + **Recall (par classe) :** Proportion de vrais positifs détectés parmi tous les positifs réels de cette classe. Crucial pour la classe "Difficilement Accessible".
  + **F1-score (par classe) :** Moyenne harmonique de la précision et du rappel, particulièrement pertinente pour les classes déséquilibrées.
  + **Matrice de Confusion :** Visualisée avec seaborn.heatmap, elle permet de comprendre quelles classes sont confondues par le modèle, fournissant des insights clés sur les erreurs.
* **Visualisations :**
  + **Courbes d'Apprentissage (Loss et Accuracy vs. Époques) :** Pour le modèle DL, ces graphiques permettent d'analyser la convergence, de détecter le surapprentissage (écart entre train et validation), et de visualiser l'impact des callbacks.
  + **Importance des Caractéristiques :** Pour le modèle ML (notamment Random Forest), générer et interpréter un graphique de l'importance des caractéristiques (feature\_importances\_) pour répondre à l'objectif "Identifier les facteurs clés".
* **Comparaison des Modèles :**
  + Analyse quantitative et qualitative des performances du modèle ML et du modèle DL.
  + Justification du modèle final choisi pour le déploiement sur l'API, en se basant sur les métriques (en particulier pour la classe "Difficilement Accessible") et la complexité/vitesse d'inférence.

**4.4 Déploiement et MLOps (MVP - Minimum Viable Product - C4 Bloc 05) :**

* **Sauvegarde du Modèle :** Le modèle de Deep Learning final (ou celui choisi comme "final") sera sauvegardé au format standard de Keras (.h5 ou .keras).
* **Création d'une API (FastAPI) :**
  + **Outil :** Utilisation de **FastAPI** pour construire une API RESTful légère et performante, comme vu en formation.
  + **Architecture :**
    1. Un endpoint POST (/predict\_tabular) qui accepte un objet JSON contenant les caractéristiques d'un lieu (ex: {"largeur\_trottoir\_cm": 120, "pente\_acces\_degres": 7.5, ...}).
    2. Un endpoint POST (/predict\_image) qui accepte le téléversement d'un fichier image (via UploadFile de FastAPI), si l'analyse d'image est incluse.
  + **Implémentation :** L'API Python chargera les modèles sauvegardés en mémoire au démarrage (pour éviter de les recharger à chaque requête). Elle implémentera la logique pour :
    1. Recevoir les données brutes de la requête.
    2. Appliquer le même pipeline de prétraitement (StandardScaler, OneHotEncoder, et/ou le prétraitement d'image) utilisé lors de l'entraînement. Il est crucial d'utiliser les mêmes objets scalers/encoders entraînés.
    3. Effectuer la prédiction à l'aide du modèle chargé.
    4. Formater la réponse en JSON (ex: {"niveau\_difficulte": "Modérément Accessible", "probabilites": {"Facile": 0.3, "Modere": 0.6, "Difficile": 0.1}}).
* **Démonstration d'Intégration (Simulée) :**
  + **Interface Web :** L'interface web simplifiée (fichiers HTML/CSS/JavaScript) sera la vitrine de cette intégration. Le JavaScript enverra des requêtes HTTP (fetch ou axios) aux endpoints de l'API FastAPI et affichera les résultats de manière dynamique.
  + **Objectif MLOps :** Bien que le déploiement ne soit pas sur un serveur cloud pour la certification, la mise en place d'une API démontre une compréhension des principes MLOps. Expliquer que cette API est le moyen standard de rendre un modèle d'IA accessible à d'autres applications. Aborder succinctement les étapes futures pour un déploiement réel : conteneurisation (Docker), orchestration (Kubernetes), surveillance des performances en production, et mises à jour continues du modèle.

### 

### **5. Organisation du Projet et Livrables**

**5.1 Rôles et Responsabilités (votre rôle de Développeur IA - selon la formation) :**

* **Mission :** Construire et affiner la solution IA technique.
* **Responsabilités :**
  + Génération/Collecte et Prétraitement des données (structurées et non structurées).
  + Développement des algorithmes d'apprentissage (Machine Learning et Deep Learning).
  + Entraînement et évaluation des modèles.
  + Optimisation des modèles (hyperparamètres, régularisation, gestion du déséquilibre).
  + Mise en œuvre technique du déploiement via l'API (FastAPI).

**5.2 Livrables pour la Certification :**

* **Cahier des Charges :** Ce document détaillé.
* **Notebooks Python :** Organisés, clairs et richement commentés, démontrant chaque étape :
  + 01\_Data\_Preparation.ipynb : Script de génération du dataset, démonstration complète des étapes de nettoyage, encodage, standardisation, et division des données (train/val/test). Inclure une analyse exploratoire rapide.
  + 02\_ML\_Model\_Training\_Evaluation.ipynb : Implémentation, entraînement, évaluation du modèle ML. Comparaison des algorithmes ML, justification des choix, métriques, matrice de confusion, et analyse de l'importance des *features*.
  + 03\_DL\_Model\_Training\_Evaluation.ipynb : Implémentation, entraînement, évaluation du modèle DL. Inclure le prétraitement des données non structurées (si images), l'architecture du réseau, les callbacks, les métriques, la matrice de confusion et les courbes d'apprentissage.
  + 04\_Model\_Comparison\_Selection.ipynb : Comparaison finale des performances ML vs DL, justification du modèle retenu pour le déploiement.
  + api\_accessibilite.py : Fichier Python contenant le code de l'API FastAPI, avec les endpoints pour les prédictions tabulaires et images. Inclure les imports nécessaires et la logique de chargement du modèle.
  + frontend\_demo/ : Dossier contenant index.html, style.css, script.js pour l'interface web de démonstration.
* **Dossier Technique :** Un document écrit (Word/PDF) structuré selon la trame fournie, qui sera le support principal de votre justification. Il doit détailler :
  + Les choix de la source des données et la logique de génération.
  + Chaque étape du prétraitement (avec justification).
  + Les architectures des modèles ML et DL (justification des couches, activations, etc.).
  + Les processus d'entraînement (optimiseurs, fonctions de perte, callbacks, hyperparamètres).
  + L'analyse détaillée des performances (toutes les métriques, matrices de confusion, courbes d'apprentissage).
  + La comparaison des modèles.
  + L'architecture de l'API FastAPI et la démonstration de son fonctionnement.
  + Les perspectives d'amélioration et la prise en compte des enjeux éthiques.
* **Présentation :** Un support visuel (slides) clair et concis pour la soutenance orale. Il résumera le projet, les choix techniques, les résultats clés, les démonstrations de l'API et de l'interface, et les aspects éthiques.

### **6. Critères de Succès et Validation**

Le jury évaluera votre capacité à :

* **Performance du Modèle :** Le modèle d'IA doit démontrer une capacité fiable à classer la difficulté d'accessibilité. Un F1-score macro-moyenne d'au moins 75% sur l'ensemble de test serait un bon indicateur. La performance sur la classe "Difficilement Accessible" sera particulièrement scrutée, car c'est la plus critique.
* **Pertinence Fonctionnelle :** La solution doit concrètement adresser le besoin d'informations nuancées et proactives pour les PMR, telle qu'illustrée par l'interface de démonstration.
* **Démonstration du Déploiement :** La capacité à lancer l'API FastAPI localement et à interagir avec elle via l'interface web sera validée. La fluidité de cette interaction est clé.
* **Compréhension et Justification :** Vous devez être capable d'expliquer et de justifier oralement chaque choix technique (source de données, prétraitement, algorithme ML/DL, hyperparamètres, métriques, architecture API) et de discuter des résultats obtenus.
* **Prise en Compte Éthique :** Démontrer une reconnaissance des défis éthiques liés à l'IA et aux données PMR (potentiels biais dans le dataset, questions de confidentialité et de sécurité des données, limitations du modèle), et proposer des solutions ou des pistes d'atténuation.

Idees capture Theme appli :





