# Conception et Développement d'un Microservice d'IA pour la Génération Automatisée d'Assets 3D Unity via MCP

## 1. Introduction

Ce rapport propose une feuille de route pour un microservice capable de transformer des données non structurées (emails, PDF, XML) en objets 3D dans Unity.

### 1.1 Objectifs du Microservice

L'objectif principal est de développer une architecture microservice autonome capable de :

1. **Ingérer** des fichiers multi-formats (PDF, Email, CSV, XML, TXT).
2. **Nettoyer et Extraire** les métadonnées spécifiques ("métadonnées Unity") via un LLM.
3. **Générer l'Objet 3D** (Approche neuronale ou procédurale).
4. **Piloter Unity via MCP** : Au lieu d'un simple téléchargement, le microservice agit comme un client MCP qui ordonne à l'éditeur Unity (Serveur MCP) d'instancier, de configurer et de placer l'objet dans la scène active.

## 2. État de l'Art : Ingestion, LLM et Protocole MCP

### 2.1 Le Model Context Protocol (MCP) et Unity

Le **Model Context Protocol (MCP)** est un standard récent (2024-2025) promu par Anthropic pour standardiser la connexion entre les assistants IA et les sources de données ou outils locaux.

* **Concept :** MCP permet de définir une architecture Client-Hôte-Serveur. Dans votre cas, Unity agit comme un **Serveur MCP** exposant des "Outils" (ex: SpawnObject, SetMaterial, LogToConsole), et votre microservice Python agit comme un **Client MCP** (ou Hôte) qui appelle ces outils.
* **Avantage PFE :** Cela remplace les vieilles méthodes de "polling" (boucle infinie qui vérifie si un fichier existe) par une communication bidirectionnelle robuste via WebSockets ou Stdio. Des projets open-source comme unity-mcp démontrent la faisabilité de ce pont.

### 2.2 Traitement des Documents (Unstructured)

L'utilisation de la bibliothèque unstructured reste la norme pour le partitionnement intelligent des PDF et Emails, permettant de séparer le "bruit" (signatures, clauses légales) des données techniques (dimensions, couleurs).1

### 2.3 Génération 3D

* **Génératif :** Modèles comme **TripoSR** (rapide) ou **LGM** pour le Text-to-Mesh.3
* **Procédural :** Génération de scripts C# via des modèles de code comme **Qwen 2.5 Coder**.5

## 3. Architecture Technique avec MCP

### 3.1 Stack Technologique Mise à Jour

| **Composant** | **Technologie** | **Justification** |
| --- | --- | --- |
| **API / Orchestration** | **FastAPI + Celery** | Gestion asynchrone des tâches lourdes (Inférence 3D).6 |
| **Protocole Agent** | **MCP (Python SDK)** | Standard pour connecter le "Cerveau" (Python) à l'outil (Unity). |
| **Pont Unity** | **Unity MCP Bridge (C#)** | Plugin Unity exposant des méthodes C# comme des "Tools" MCP.8 |
| **Ingestion** | **Unstructured** | Parsing sémantique des fichiers hétérogènes. |
| **Inférence LLM** | **vLLM / Llama.cpp** | Exécution locale optimisée des modèles (Llama 3, Mistral). |
| **Broker** | **Redis** | Communication inter-processus. |

### 3.2 Flux de Données "Agentique" (Data Flow)

1. **Réception :** L'utilisateur envoie un Email/PDF au microservice.
2. **Analyse :** Le LLM extrait le JSON de métadonnées (ex: {"type": "chair", "color": "red"}).
3. **Génération :** Le microservice crée le fichier .glb ou le script .cs.
4. **Action MCP :**
   * Le microservice (Client MCP) détecte que Unity est connecté.
   * Il envoie une requête MCP call\_tool("ImportAsset", { "path": "/data/chair.glb" }).
   * Il envoie ensuite call\_tool("SetTransform", { "position": }).
5. **Exécution :** Unity (Serveur MCP) reçoit les commandes et met à jour la scène en temps réel sous les yeux de l'utilisateur.

## 4. Backlog Agile Détaillé (4 Mois / 16 Semaines)

Le planning est dense. La phase MCP est intégrée comme une évolution majeure en fin de parcours pour transformer le projet d'un simple "générateur" en un "assistant intelligent".

### Phase 1 : Infrastructure & Ingestion

*Objectif : Socle technique et nettoyage des données.*

* **US-1.1 [Infra] :** Setup Docker-Compose (FastAPI, Redis, Celery).
* **US-1.2 :** Implémentation du pipeline unstructured pour parser PDF et Emails (.eml).
* **US-1.3 :** Création de filtres Regex pour nettoyer le "bruit" (signatures, headers).
* **US-1.4 [API] :** Endpoint /upload asynchrone qui renvoie un task\_id.

### Phase 2 : Le Cerveau - LLM & Extraction

### *Objectif : Comprendre la demande utilisateur.*

* **US-2.1 [IA] :** Intégration de **Llama-3-8B-Instruct** (via llama-cpp-python).
* **US-2.2 [IA] :** Développement du Prompt Système pour forcer la sortie JSON (Schema Enforcement).
* **US-2.3 [IA] :** Gestion des cas complexes : extraction de tableaux de dimensions depuis un PDF.
* **US-2.4 :** Validation Pydantic des métadonnées extraites.

### Phase 3 : Le Constructeur - Génération 3D

*Objectif : Créer l'asset numérique.*

* **US-3.1 :** Intégration du modèle **TripoSR** pour la génération Text-to-Mesh rapide.
* **US-3.2 [IA Code] :** Pipeline alternatif : génération de scripts C# procéduraux pour les formes géométriques.
* **US-3.3 :** Conversion et optimisation des assets (OBJ vers GLB).
* **US-3.4 :** Gestion du stockage temporaire des fichiers générés.

### Phase 4 : Serveur MCP Unity - L'Intégration Agentique

### \*\*\*\*

*Objectif : Remplacer l'import manuel par un pilotage automatique via MCP.*

* **US-4.1 [MCP Python] :** Implémenter un **Client MCP** dans le microservice Python (utilisant le SDK mcp).
* **US-4.2 [MCP Unity] :** Développer (ou adapter unity-mcp) un **Serveur MCP en C#** dans Unity qui écoute sur un port WebSocket/Stdio.
* **US-4.3 :** Définir les "Outils" Unity exposés à l'IA :
  + SpawnGLB(url, position)
  + CreatePrimitive(type, scale)
  + LogMessage(text)
* **US-4.4 [Orchestration] :** Connecter le pipeline de fin de génération (Celery) à l'appel d'outil MCP. Dès que le fichier est prêt, l'ordre d'import est envoyé.

### Phase 5 : UX & Finalisation

### *Objectif : Expérience utilisateur et livrables.*

* **US-5.1 [Unity] :** Créer une fenêtre "Moniteur IA" dans Unity pour voir les logs de réception MCP.
* **US-5.2 :** Test de bout en bout : Email -> Extraction -> Génération -> Apparition dans Unity sans clic.
* **US-5.3 :** Rédaction du mémoire et documentation technique de l'architecture MCP.
* **US-5.4 [Optim] :** Nettoyage du code et optimisation de la VRAM.

| **Sprint** | **Durée (Semaines)** | **Tâches Incluses (US)** | **Objectif Principal du Sprint** |
| --- | --- | --- | --- |
| **1** | 2 | US-1.1, US-1.2, US-1.3 | Mise en place de l'infrastructure Docker/FastAPI/Celery et le pipeline d'ingestion des documents (unstructured). |
| **2** | 2 | US-1.4, US-2.1, US-2.2 | Endpoint d'upload asynchrone et intégration initiale du LLM (Llama-3) avec la contrainte de sortie JSON. |
| **3** | 2 | US-2.3, US-2.4, US-3.1 | Finalisation de l'extraction de métadonnées complexes (tableaux), validation Pydantic, et intégration du modèle **TripoSR** (Text-to-Mesh). |
| **4** | 2 | US-3.2, US-3.3, US-3.4 | Développement du pipeline alternatif de génération de scripts C# procéduraux, conversion et gestion du stockage des assets (.glb). |
| **5** | 2 | US-4.1, US-4.2, US-4.3 | Implémentation du **Client MCP** (Python SDK) et du **Serveur MCP** (Unity C#) avec la définition des premiers "Outils" Unity. |
| **6** | 2 | US-4.4, US-5.1, US-5.2 | Connexion du pipeline de génération à l'appel d'outil MCP (Orchestration), création du "Moniteur IA" dans Unity, et test de bout en bout complet. |
| **7** | 2 | US-5.3 | **Focus :** Rédaction du mémoire de fin d'études et de la documentation technique de l'architecture MCP. |
| **8** | 2 | US-5.4, Révision, Polissage | Nettoyage final du code, optimisation de la VRAM et révision/polissage général avant la livraison. |

## 

## 5. Ressources Spécifiques pour ce Backlog

### Modèles IA (Hugging Face) (peuvent changer selon les contraintes materielles)

* **Extraction :** meta-llama/Meta-Llama-3-8B-Instruct (Le standard actuel pour suivre des instructions).
* **Code C# :** Qwen/Qwen2.5-Coder-7B-Instruct (Excellent pour le code Unity et le JSON).
* **Génération 3D :** stabilityai/TripoSR (Le plus rapide et stable pour un usage local).

### Bibliothèques MCP & Unity

* **Python SDK :** pip install mcp (Officiel Anthropic).
* **Unity MCP Implementation :** Regarder le dépôt GitHub **CoplayDev/unity-mcp** ou **ModelContextProtocol/csharp-sdk** pour la base du serveur C#.
* **Ingestion :** pip install unstructured[pdf,email].

### Datasets (Kaggle / HF)

* **Objaverse-XL :** Pour comprendre la structure des objets 3D (si besoin de fine-tuning).
* **DeepCAD :** Pour les descriptions techniques (CAO).

#### Sources des citations

1. Quickstart - Unstructured document, consulté le février 3, 2026, <https://docs.unstructured.io/open-source/introduction/quick-start>
2. Partitioning - Unstructured Documentation, consulté le février 3, 2026, <https://docs.unstructured.io/open-source/core-functionality/partitioning>
3. TripoSR: Fast 3D Object Reconstruction from a Single Image - arXiv, consulté le février 3, 2026, <https://arxiv.org/html/2403.02151v1>
4. Best AI 3D Generator? I Tested Every Major Platform (Shocking Results!) - YouTube, consulté le février 3, 2026, <https://www.youtube.com/watch?v=jfk8e4ykp-s>
5. The 11 best open-source LLMs for 2025 - n8n Blog, consulté le février 3, 2026, <https://blog.n8n.io/open-source-llm/>
6. SteliosGian/fastapi-celery-redis-flower - GitHub, consulté le février 3, 2026, <https://github.com/SteliosGian/fastapi-celery-redis-flower>
7. Building Scalable Background Jobs with FastAPI + Celery + Redis | by Shaikh Asif - Medium, consulté le février 3, 2026, <https://medium.com/@shaikhasif03/building-scalable-background-jobs-with-fastapi-celery-redis-e43152829c61>
8. UnityCopilot is an AI-powered chat interface for the Unity Editor. It is designed to assist game developers in building their games in Unity. - GitHub, consulté le février 3, 2026, <https://github.com/Danejw/UnityCopilot>