PROJET STATION TV

Dossier de Conception Général

SUIVI DES MODIFICATIONS DU DOCUMENT			
Version	Date	Validation	Commentaire
1.0	19/11/2025	Frédéric Chauvin	Rédaction initiale

REDACTEUR : Dorian BRISSON	VALIDATEUR : Frédéric CHAUVIN
CLIENT : Polytech Tours	ENCADRANT : Mathieu DELALANDRE

Table des matières

PROJET STATION TV 1

OBJET DU DOCUMENT 3
POSITION DANS LE CYCLE DE DEVELOPPEMENT 4
PERIMETRE DU DOCUMENT 5
PRÉSENTATION GÉNÉRALE 6
ARCHITECTURE LOGICIELLE ET MATERIELLE 7
Architecture matérielle 7
Architecture logicielle 7
COMMUNICATION ET REPARTITION DES RESPONSABILITES 8
Répartition des responsabilités 8
Flux de données 9
OUTILS LOGICIELS ET VERSIONS 9
Langage 9
Bibliothèques principales 9
Outils de développement 10
Outils de planification et documentation 10
FONCTIONNEMENT DU SYSTEME 10
Description fonctionnelle générale 10
Pipeline général de traitement 11
Étapes détaillées 11
COMPORTEMENT ATTENDU ET SYNCHRONISATION 13
Fiable 13
Stable 13
Performant 13
Synchronisé 13
RELATIONS ENTRE PIPELINE GLOBAL ET SOUS-MODULES 14
Rôle fonctionnel de chaque module dans la chaîne 14
INTEGRATION ET PREPARATION AUX TESTS D'INTEGRATION 14
Stratégie générale d'intégration 14
Stratégie d'intégration par module 15
Étape 1 — Intégration du Preprocessing 15
Étape 2 — Intégration du Core (Transcription) 15
Étape 3 — Intégration du Monitoring QoS 15
Étape 4 — Intégration du module Export 16
Étape 5 — Intégration finale du pipeline complet 16
INTERFACES ET I/O DU SYSTEME 16
Entrées du système 16
Sorties du système 17
OBJECTIFS ET SCENARIOS DE TESTS D'INTEGRATION 17
Objectifs principaux 17
Scénarios de tests 17

OBJET DU DOCUMENT

Le présent document constitue le **Dossier de Conception Général (DCG)** du projet **Station TV – Système de transcription audio haute performance**.
Il a pour objectif de présenter l'architecture globale du système, les choix de conception majeurs, ainsi que l'organisation fonctionnelle et technique du pipeline de traitement audio.

Le DCG fournit une vision d'ensemble cohérente et structurée du système avant la phase de conception détaillée. Il sert de référence pour :

- comprendre les rôles et interactions entre les modules (prétraitement, transcription, export, monitoring),
- définir les interfaces techniques essentielles,
- préparer les étapes de développement, d'intégration et de validation,
- garantir que le système répond aux besoins exprimés par le LIFAT dans le cadre du projet Station TV.

Position dans le cycle de développement

Le projet Station TV adopte une **méthodologie Agile**, organisée en sprints de 2 à 3 semaines, comme défini dans le Plan de Développement.
Cette approche permet une progression incrémentale**,** itérative **et** pilotée par les résultats de tests, particulièrement adaptée à un projet individuel impliquant des expérimentations intensives et des ajustements continus.

Dans cette organisation :

- les documents de cadrage et d'analyse (FOP, CDC, SPER, Plan de Développement, Specifications) sont produits et validés lors des premiers sprints ;
- le DCG intervient à un moment charnière du projet, avant le développement détaillé et avant les phases intensives d'optimisation et de tests ;
- il consolide l'architecture cible et le fonctionnement général du système, servant de base aux sprints suivants.

Le DCG occupe ainsi une place centrale dans la démarche Agile :
il fixe le référentiel architectural, tout en laissant la flexibilité nécessaire aux ajustements réalisés au fil des sprints.

Périmètre du document

Le présent document couvre l'ensemble des éléments structurants du système de transcription Station TV, notamment :

- l'**architecture logicielle globale**, composée des modules Preprocessing, Core Transcription, QoS Monitoring, Export et Utilities ;
- le **fonctionnement général du pipeline audio** : conversion, transcription, agrégation des résultats, génération de rapports ;
- les **interactions et dépendances** entre les modules techniques ;
- la **répartition des responsabilités** au sein du système ;
- les **principes de conception** guidant les futurs développements.

Sont exclus de ce DCG :

- la description interne des modules Python (détaillée dans le DCD),
- les algorithmes complets d'optimisation CPU/RAM ou d'équilibrage de charge,
- les spécifications bas niveau des fichiers de configuration,
- les tests unitaires et les résultats détaillés de performance.

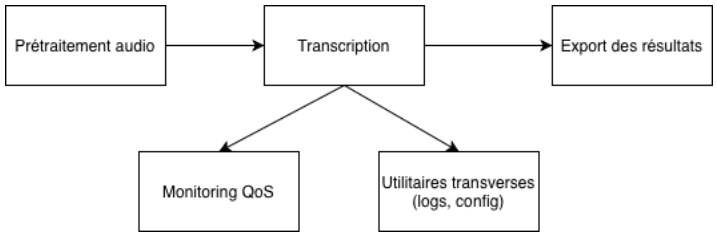
Ces éléments seront traités dans le Dossier de Conception Détaillé et dans les documents de tests.

PRÉSENTATION GÉNÉRALE

Le système développé pour Station TV repose sur une **architecture logicielle modulaire**, pensée pour assurer :

- une transcription audio fiable sur de très grands volumes (588 h et plus),
- une robustesse élevée lors des longues sessions d'exécution,
- une supervision complète de la performance (CPU, RAM, throughput),
- une extensibilité facilitant l'ajout de nouveaux formats, modèles ou métriques.

L'architecture est organisée en cinq modules principaux, articulés autour d'un pipeline de traitement automatisé :



Chaque module est indépendant et communique via des structures de données standardisées (fichiers WAV normalisés, JSON, CSV).

L'objectif est d'obtenir un système **automatisé, stable, reproductible**, capable de transcrire de larges corpus TNT sans intervention manuelle.

Architecture logicielle et matérielle

Architecture matérielle

Le système est exécuté sur une station Dell Precision 5820, dont la configuration est conçue pour répondre aux contraintes élevées de Whisper :

- **CPU** : Intel Xeon W-2295 (18 cœurs / 36 threads)
- **RAM** : 256 Go DDR4 ECC
- **Stockage** : SSD NVMe (pour les I/O rapides)
- **OS** : Windows Server 2022 ou Ubuntu 22.04 LTS
- **GPU** : Non utilisé (choix porté sur CPU-only pour garantir la reproductibilité et éviter la saturation VRAM)

Architecture logicielle

L'architecture logicielle du système Station TV se compose des modules suivants :

1. Preprocessing (Prétraitement audio)

- Conversion MP3 → WAV
- Normalisation (48 kHz, PCM16, mono)
- Segmentation optionnelle
- Suppression des silences

Ce module garantit que tous les fichiers d'entrée sont compatibles et optimisés pour Whisper.

2. Core (Transcription)

Module central du système :

- gestion du chargement des modèles Whisper (Tiny → Large),
- exécution multiprocess avec affinité CPU,
- équilibrage intelligent de charge (approche gloutonne),
- extraction des timestamps et scores de confiance,
- isolation mémoire pour éviter les saturations.

3. QoS (Supervision et monitoring)

Fonctions transversales utilisées pendant toute la durée du traitement :

- mesures périodiques CPU/RAM,
- calcul throughput (x temps réel),
- calcul WER (sur échantillons),
- export CSV + graphiques PNG haute résolution.

4. Export (Formats de sortie)

- Génération TXT, JSON, CSV, SRT,
- Intégration des métadonnées (chaîne, date, modèle utilisé),
- Système de sauvegarde automatique et backups.

5. Utilities (Outils transverses)

- Logging centralisé,
- Gestion des chemins,
- Lecture/écriture de fichiers,
- Chargement de configuration YAML.

Ces modules composent un système structuré, robuste et facilement maintenable, conçu pour des workflows complexes.

Communication et répartition des responsabilités

Le système adopte une communication simple et explicite entre ses modules, basée sur des **formats standardisés** :

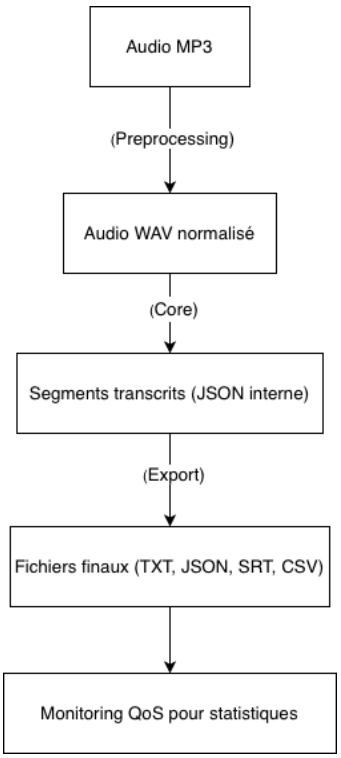
- WAV normalisés (entre Preprocessing et Core)
- JSON/CSV (entre Core et Export)
- CSV/PNG (sorties du module QoS)

Répartition des responsabilités

Module	Responsabilités principales
Preprocessing	Prépare et valide les données audio
Core Transcription	Réalise la transcription et gère le multiprocess
QoS	Mesure et analyse les performances
Export	Produit les formats finaux + intégration Station TV
Utilities	Fournit les services transverses (logs, config)

Flux de données

Le flux général se résume ainsi :



Chaque étape est indépendante, évitant les effets de bord et simplifiant les tests d'intégration.

Outils logiciels et versions

Les outils et technologies utilisés ont été sélectionnés pour leur stabilité, leur maturité et leur compatibilité avec les contraintes du projet.

Langage

- **Python 3.10+** (langage principal du pipeline)

Bibliothèques principales

Outil	Rôle
Whisper (OpenAI)	Moteur STT
PyTorch	Backend DL

Outil	Rôle
FFmpeg	Conversion audio
Librosa	Analyse audio
Multiprocessing	Parallelisation CPU
Psutil	Monitoring CPU/RAM
Pandas	Manipulation data
Matplotlib	Graphiques QoS

Outils de développement

- Visual Studio Code
- GitHub + Git
- Environnements virtuels (venv)

Outils de planification et documentation

- GitHub Projects (Kanban Agile)
- GanttProject
- Teams pour suivi encadrant
- Markdown / Word / PDF pour documentation

Ces outils garantissent un environnement fiable pour le développement incrémental du système.

Fonctionnement du système

Description fonctionnelle générale

Le système Station TV met en œuvre un pipeline de traitement audio entièrement automatisé, conçu pour transformer de larges volumes de fichiers MP3 issus des chaînes TNT en transcriptions exploitables.

Le fonctionnement repose sur quatre grandes étapes successives :

1. **Prétraitement des sources audio**
   Conversion, normalisation et préparation des fichiers d'entrée.
2. **Transcription multiprocess**
   Exécution parallèle des modèles Whisper sur CPU, avec gestion affinée de l'affinité processeur et de la mémoire.
3. **Export et structuration des résultats**
   Génération des sorties finales dans les formats demandés (TXT, JSON, CSV, SRT).

4. **Monitoring et évaluation QoS**
   Collecte et analyse des performances (CPU, RAM, throughput, WER).

Ces modules interagissent via des structures de données standardisées, assurant une indépendance forte entre les étapes et une reproductibilité maximale.

Pipeline général de traitement

Le pipeline est conçu pour fonctionner de manière séquentielle entre modules, mais parallèle à l'intérieur du module de transcription.

Voici la représentation fonctionnelle simplifiée :

1. Scan fichiers audio → Construction CSV métadonnées
2. Conversion MP3 → WAV normalisé
3. Transcription multiprocess Whisper
4. Export formats (TXT, JSON, SRT, CSV)
5. Monitoring QoS (CPU, RAM, throughput, WER)

Étapes détaillées

1. Scan des fichiers audio

Le système parcourt automatiquement un répertoire source, identifie les fichiers MP3 et génère un CSV contenant :

- nom du fichier,
- durée,
- chaîne d'origine,
- horaire d'émission,
- chemin complet.

2. Prétraitement (Preprocessing)

Chaque fichier MP3 est transformé en un fichier WAV conforme aux contraintes Whisper :

- 48 kHz
- PCM16
- Mono

Cela garantit une homogénéité d'entrée pour la transcription.

3. Transcription multiprocess

La transcription est réalisée selon un modèle **CPU-only parallèle** :

- création de N processus (3 à 12 selon configuration),
- attribution de cœurs dédiés via affinité CPU,
- répartition gloutonne des fichiers selon leur durée,
- chargement indépendant du modèle dans chaque processus,
- génération des segments transcrits avec timestamps synchronisés.

4. Export des résultats

À partir des segments bruts, le système génère :

- **TXT** : texte brut pour lecture humaine,
- **JSON** : structure riche avec métadonnées, timestamps et confidence scores,
- **CSV** : table finale pour analyses,
- **SRT** : sous-titres horodatés.

Des mécanismes de sauvegarde et de backups garantissent la sécurité et la persistance des données.

5. Monitoring QoS

Le module QoS collecte et agrège les métriques :

- utilisation CPU (moyenne, min, max),
- utilisation RAM,
- throughput (heures audio / heure réelle),
- vitesse moyenne de transcription,
- WER (sur échantillons de test).

Les résultats sont exportés sous forme :

- de CSV brut
- de graphiques PNG
- de rapports textuels récapitulatifs.

Comportement attendu et synchronisation

Le système doit garantir un comportement :

Fiable

- Aucun fichier n'est perdu : chaque unité de traitement génère sa sortie.
- Les erreurs locales (un fichier corrompu, un crash d'un processus) n'arrêtent pas l'ensemble du pipeline.

Stable

- Isolation mémoire par processus (pas de partage de modèle).
- Redémarrage automatique d'un processus en cas d'erreur.

Performant

- Exploitation optimale du CPU.
- Réduction des conflits de ressources RAM entre processus.

Synchronisé

Les modules fonctionnent selon des règles précises :

- Preprocessing → Core : synchronisation par fichiers WAV normalisés
- Core → Export : synchronisation par fichiers JSON
- Core → QoS : synchronisation par enregistrement périodique des métriques

Cette gestion garantit l'absence d'effet de bord entre modules et facilite les tests à chaque sprint Agile.

Relations entre pipeline global et sous-modules

Rôle fonctionnel de chaque module dans la chaîne

Module	Rôle dans le pipeline	Dépend de
Preprocessing	Prépare les données audio	Entrée MP3
Core	Transcrit et produit les segments	WAV normalisés
Export	Convertit en formats finaux	JSON segments
QoS	Analyse performance du Core	Core (indirect)
Utilities	Support global	Tous

Le pipeline ainsi structuré assure une clarité de conception, une lisibilité des échanges et une facilité de maintenance.

Intégration et préparation aux tests d'intégration

Stratégie générale d'intégration

Le système Station TV est conçu selon une architecture modulaire dont les composants peuvent être développés, testés et intégrés de manière incrémentale.
L'intégration suit la logique Agile du projet, où chaque sprint produit un ensemble fonctionnel testable.

La stratégie adoptée repose sur trois principes :

1. **Intégration progressive par modules**
   Chaque module (Preprocessing, Core, QoS, Export) est développé et validé individuellement avant d'être intégré.
2. **Tests continus au fil des sprints**
   Les tests sont réalisés progressivement, dès que les modules sont fonctionnels, sans attendre la fin du développement.
3. **Automatisation maximale**
   Des scripts dédiés (RunPipeline.py, BasicTestWhisper.py, ComputeQoS.py) permettent de tester rapidement la cohérence du système à chaque itération.

L'objectif est de réduire drastiquement les risques techniques (crash RAM, instabilité Whisper, erreurs d'export), tout en garantissant une montée en puissance progressive.

Stratégie d'intégration par module

Étape 1 — Intégration du Preprocessing

Objectifs :

- Valider la conversion MP3 → WAV,
- Assurer l'homogénéité du format (48 kHz, PCM16, mono),
- Vérifier les métadonnées audio (durée, taille).

Tests associés :

- conversion sur 10 fichiers représentatifs,
- contrôle des durées WAV vs MP3,
- mesure du temps de prétraitement.

Étape 2 — Intégration du Core (Transcription)

Objectifs :

- Vérifier le chargement des modèles Whisper (Small, Medium),
- Contrôler l'exécution multiprocess,
- Valider l'attribution d'affinité CPU,
- Assurer la stabilité RAM sur sessions 30–60 minutes.

Tests associés :

- test unitaire : BasicTestWhisper.py,
- test parallèle sur 3 → 6 processus,
- mesure du throughput,
- détection d'éventuels crashs RAM.

Étape 3 — Intégration du Monitoring QoS

Objectifs :

- Mesurer en temps réel l'utilisation CPU/RAM,
- Calculer throughput, vitesse, WER,
- Exporter les données brutes (CSV) et graphiques (PNG).

Tests associés :

- test long (> 48h) pour validation stabilité métriques,
- comparaison entre runs consécutifs,
- vérification absence de dérive mémoire.

Étape 4 — Intégration du module Export

Objectifs :

- Générer les formats TXT/JSON/CSV/SRT,
- Vérifier la cohérence des timestamps,
- Ajouter les métadonnées Station TV (chaîne, date, modèle utilisé).

Tests associés :

- comparaison texte brut / JSON / SRT,
- validation via un lecteur SRT,
- vérification des champs JSON.

Étape 5 — Intégration finale du pipeline complet

Objectifs :

- Vérifier l'enchaînement complet Preprocessing → Core → Export → QoS,
- Contrôler la gestion des erreurs (try/catch),
- Tester la stabilité sur de longues sessions.

Scripts mobilisés :

- RunPipeline.py,
- RunBatchWhisper.py,
- ComputeQoS.py.

Interfaces et I/O du système

Entrées du système

Le système consomme les éléments suivants :

- **Fichiers audio MP3** (48 kHz, 256 kbps),
- **CSV de métadonnées initial** généré automatiquement,
- **Fichier de configuration YAML** définissant :
  - chemins d'entrée et de sortie,
  - modèle Whisper à utiliser,
  - nombre de processus,
  - options preprocessing.

Sorties du système

Le pipeline produit :

- **Transcriptions TXT**
- **Sous-titres SRT**
- **Segments JSON** avec timestamps + confidence scores
- **Tableaux CSV** consolidés (multi-fichiers)
- **Rapports QoS** (CSV + PNG graphiques)
- **Logs détaillés** (journal horodaté par module)

Ces sorties sont compatibles avec les outils d'analyse du LIFAT et peuvent être directement intégrées dans Station TV.

Objectifs et scénarios de tests d'intégration

Les tests d'intégration ont pour but de valider le fonctionnement cohérent de l'ensemble du système.

Objectifs principaux

1. **Vérifier la stabilité du pipeline long** (> 5 jours)
2. **Garantir l'absence de fuite mémoire ou crash CPU**
3. **Valider l'enchaînement automatique des modules**
4. **Confirmer la cohérence des formats exportés**
5. **Atteindre les objectifs de performance** :
    - < 7 jours pour 588h,
    - throughput ≥ 1.4× (model Medium),
    - WER < 10 % sur audio propre.

Scénarios de tests

Test	Description	Critère de réussite
TI-01	Pipeline simple sur un fichier	Tous les formats générés correctement
TI-02	Pipeline multiprocess sur un lot court	Aucun crash, throughput stable
TI-03	Test longue durée (≥ 48h)	RAM stable, QoS cohérente
TI-04	Export multi-formats	Timestamps synchronisés, JSON valide
TI-05	Mesure QoS complète	CSV + PNG générés, WER calculé
TI-06	Pipeline 588h (simulation ou découpage)	Temps total < 7 jours