Modules à transformer en services

Principes directeurs pour la transformation

Après une analyse détaillée de l'architecture actuelle, voici les principes qui ont guidé l'identification des modules à transformer en services:

1. **Haute cohésion, faible couplage**: Les modules qui ont une responsabilité unique et bien définie
2. **Réutilisabilité inter-composants**: Fonctionnalités utilisées par plusieurs parties du système
3. **Scalabilité requise**: Modules susceptibles de nécessiter une mise à l'échelle indépendante
4. **Criticité du domaine**: Fonctionnalités centrales au métier de l'application
5. **Indépendance technique possible**: Composants pouvant fonctionner de manière autonome

Modules prioritaires à transformer en services

1. Module d'Apprentissage (Learning)

Localisation actuelle: `src/ai/learning/`

Justification:

- Responsabilité claire: gestion de l'apprentissage et des parcours personnalisés
- Déjà partiellement structuré comme un service (voir `src/ai/services/learning/`)
- Utilisé par de multiples composants du système
- Contient des sous-modules qui pourraient être des microservices:
  - Fine-tuning local (`LocalFineTuningSystem.ts`)
  - Adaptation intelligente (`IntelligentAdapter.ts`)
  - CODA virtuel (apprentissage inverse)

Structure de service proposée:

```
services/
├── learning/
│   ├── core/                # Fonctionnalités centrales d'apprentissage
│   ├── adaptation/          # Adaptation des contenus selon contexte
│   ├── fine-tuning/         # Optimisation locale des modèles
│   ├── evaluation/          # Évaluation des progrès et compréhension
│   ├── coda-virtuel/        # Système d'apprentissage inversé
│   ├── gamification/        # Éléments de ludification
│   └── immersive/           # Environnements d'apprentissage immersifs
```

2. Système de Validation Collaborative

Localisation actuelle: `src/ai/validation/`

Justification:

- Composant critique pour la qualité et l'éthique de l'application
- Implique potentiellement des utilisateurs externes (experts LSF)
- Besoin de scalabilité indépendante lors des phases de validation
- Fonctionnalité transversale utilisée par de nombreux modules

Structure de service proposée:

```
services/
├── validation/
│   ├── api/                  # API REST pour l'interaction externe
│   ├── consensus/            # Algorithmes de consensus et validation
│   ├── experts/              # Gestion des profils d'experts
│   ├── clubs/                # Gestion des clubs thématiques
│   └── feedback/             # Système de retour et amélioration continue
```

3. Système d'Expressions et Emotions

Localisation actuelle: `src/ai/systems/expressions/` et sous-répertoires

Justification:

- Calcul intensif (rendu, animations)
- Utilisé par différentes interfaces (avatar sourd, avatar entendant)
- Potentiellement distribué sur différents devices (client/serveur)
- Besoin de mise à l'échelle indépendante

Structure de service proposée:

```
services/
├── expressions/
│   ├── facial/               # Expressions faciales
│   ├── gestual/              # Mouvements et gestes
│   ├── emotional/            # Adaptation émotionnelle
│   ├── cultural/             # Adaptation culturelle
│   └── synchronization/      # Synchronisation multimodale
```

4. Gestion de l'Espace Spatial

Localisation actuelle: `src/ai/spatial/`

Justification:

- Domaine spécialisé avec ses propres règles et complexités
- Utilisé par plusieurs composants (expressions, apprentissage, etc.)
- Contient des fonctionnalités avancées (maintien des points de référence)

Structure de service proposée:

```
services/
├── spatial/
│   ├── reference/            # Gestion des références dans l'espace
│   ├── coherence/            # Validation de la cohérence spatiale
│   ├── mapping/              # Mapping mental des espaces de signation
│   └── analysis/             # Analyse des structures spatiales
```

5. Gestion des Variantes Dialectales

Localisation actuelle: `src/ai/cultural/`

Justification:

- Besoin d'évolution indépendante (ajout de nouvelles variantes)
- Base de connaissances spécifique
- Fonctionnalité transversale utilisée par d'autres services

Structure de service proposée:

```
services/
├── dialectal/
│   ├── cartography/          # Cartographie des variantes régionales
│   ├── detection/            # Détection de l'origine du signant
│   ├── adaptation/           # Adaptation aux variantes
│   └── enrichment/           # Enrichissement continu du lexique
```

6. Orchestrateur Central

Localisation actuelle: `src/ai/coordinators/SystemeOrchestrateurCentral.ts`

Justification:

- Point central de coordination
- Responsable du routage et de la distribution des requêtes
- Besoin de haute disponibilité et résilience

Structure de service proposée:

```
services/
├── orchestrator/
│   ├── router/              # Routage intelligent des requêtes
│   ├── dispatcher/          # Distribution des charges
│   ├── monitoring/          # Surveillance des performances
│   └── cache/               # Gestion avancée du cache
```

7. Génération de Contenu

Localisation actuelle: Dispersée dans différents modules

Justification:

- Fonctionnalité identifiée dans les diagrammes mais peu structurée
- Besoin croissant de contenu pédagogique généré
- Peut bénéficier d'une API dédiée

Structure de service proposée:

```
services/
├── content-generation/
│   ├── educational/         # Contenu pédagogique
│   ├── exercises/           # Exercices interactifs
│   ├── personalization/     # Personnalisation du contenu
│   └── visualization/       # Visualisations et diagrammes
```

8. Système de Feedback et Métriques

Localisation actuelle: `src/ai/feedback/` et divers sous-modules

Justification:

- Utilisé par l'ensemble du système
- Besoin de centralisation des métriques
- Analyse de données potentiellement intensive

Structure de service proposée:

```
services/
├── feedback/
│   ├── collection/          # Collecte des retours utilisateurs
│   ├── analysis/            # Analyse des données
│   ├── metrics/             # Métriques d'utilisation et performance
│   └── optimization/        # Recommandations d'optimisation
```

Modules secondaires à transformer ultérieurement

1. **Système de Sécurité Renforcée** (`src/ai/security/`)
2. **Monitoring Unifié** (`src/ai/monitoring/`)
3. **Système Multi-modal** (`src/ai/multimodal/`)
4. **Système Conversationnel Avancé** (`src/ai/conversational/`)
5. **Système d'Intégration** (`src/ai/integration/`)

Stratégie de transformation progressive

1. **Étape 1**: Transformer les services liés à l'apprentissage (priorité immédiate)
2. **Étape 2**: Migrer les services d'expressions et validation collaborative
3. **Étape 3**: Transformer les services spatiaux et dialectaux
4. **Étape 4**: Finaliser avec l'orchestrateur et la génération de contenu

Pour chaque transformation:

1. Créer l'interface du service (API contracts)
2. Implémenter le service avec ses dépendances
3. Adapter les consommateurs actuels
4. Mettre en place les tests d'intégration
5. Déployer et valider en environnement contrôlé