# Système de Scraping et Résumé Automatique d’Articles de Presse avec Intelligence Artificielle

## 1. Introduction

* **Contexte** : Développement d’une solution innovante de scraping automatique et résumé intelligent d’articles de presse
* **Cadre** : Projet de fin d’études de master à l’ESRMI
* **Objectif Principal** : Automatisation de la veille technologique via l’intelligence artificielle

## 2. Problématique

### Défis Actuels

* Volume informationnel massif dans la presse technologique
* Temps limité pour la veille technologique
* Difficulté de filtrage et de priorisation
* Qualité variable des sources d’information

### Impact

* Surcharge informationnelle pour les professionnels et étudiants
* Besoin de synthèses rapides et fiables
* Nécessité d’une solution automatisée

## 3. Spécifications des Besoins

### Besoins Fonctionnels

1. **Collecte Automatique**
   * Scraping multi-sources (The Verge, CNET, Wired)
   * Minimum 20 articles par source par jour
   * Traitement en temps réel
2. **Résumés IA**
   * Génération automatique via Google Gemini
   * Résumés de 100-200 mots
   * Analyse sémantique du contenu
3. **API REST**
   * CRUD complet
   * Recherche et filtrage par source
   * Temps de réponse < 2 secondes
4. **Automatisation**
   * Planification automatique des tâches
   * Gestion des erreurs
   * Monitoring des processus

### Technologies Utilisées

1. **Backend**
   * AdonisJS v6 + TypeScript
   * Node-cron pour la planification
2. **Intelligence Artificielle**
   * Google Gemini AI
   * Traitement du langage naturel
3. **Web Scraping**
   * Puppeteer
   * Cheerio
4. **Base de Données**
   * Supabase (PostgreSQL)
   * API auto-générée

## 4. Conception de la Solution

### Architecture en Couches

┌─────────────────────────────────────────────┐  
│ API LAYER │  
│ AdonisJS v6 + TypeScript │  
├─────────────────────────────────────────────┤  
│ BUSINESS LAYER │  
│ ScraperService │ AIService │ CronService │  
├─────────────────────────────────────────────┤  
│ INTEGRATION LAYER │  
│ Puppeteer │ Cheerio │ Google Gemini │  
├─────────────────────────────────────────────┤  
│ DATA LAYER │  
│ Supabase (PostgreSQL) │  
└─────────────────────────────────────────────┘

### Services Principaux

* **ScraperService** : Collecte d’articles
* **AIService** : Génération de résumés
* **CronService** : Planification
* **SupabaseService** : Gestion des données

## 5. Méthodologie de Travail

### Approche Agile Adaptée

#### Phases de Développement

1. **Phase 1** : Analyse des besoins et conception
   * Étude de l’existant
   * Choix technologiques
   * Architecture système
2. **Phase 2** : Développement Core
   * Services de base
   * Intégration APIs externes
   * Développement scrapers
3. **Phase 3** : API et Intégration
   * Endpoints REST
   * Système de planification
   * Tests d’intégration
4. **Phase 4** : Finalisation
   * Tests de performance
   * Documentation
   * Préparation démo

### Gestion des Risques

|  |  |  |
| --- | --- | --- |
| Risque | Impact | Mitigation |
| **Changement structure sites** | Élevé | Architecture modulaire par source |
| **Limites API IA** | Moyen | Gestion d’erreurs et fallback |
| **Performance** | Moyen | Tests et optimisation continue |

## 6. Démonstration : Étapes

### 1. Configuration Initiale

* Installation des dépendances
* Configuration des variables d’environnement
* Initialisation de la base de données

### 2. Scraping d’Articles

* Démonstration du scraping en temps réel
* Visualisation des données collectées
* Gestion des erreurs

### 3. Génération de Résumés

* Traitement par l’IA
* Qualité des résumés
* Performances

### 4. API REST

* Endpoints disponibles
* Exemples de requêtes
* Réponses et formats

### 5. Monitoring

* Dashboard de suivi
* Logs et statistiques
* État du système