SEO Scraper Service

Micro-service FastAPI de scraping haute performance avec Crawl4AI.

Installation

```
# Creer l'environnement virtuel et installer
make install

# Ou pour le developpement
make install-dev
```

Lancement

```
make run
```

Le service demarre sur le port 8000 par defaut.

Verification

```
make check
# ou
curl http://localhost:8000/health
```

Tests

PROF

```
make test
```

API Endpoints

`GET /health`

Verifie l'etat du service.

```
curl http://localhost:8000/health
```

`POST /scrape`

Scrape une URL et retourne le contenu en Markdown.

```
curl -X POST http://localhost:8000/scrape \
  -H "Content-Type: application/json" \
  -d '{"url": "https://example.com", "ignore_body_visibility": true}'
```

Request:

```
{
  "url": "https://example.com",
  "ignore_body_visibility": true,
  "timeout": 30000
}
```

Response:

```
{
  "url": "https://example.com",
  "success": true,
  "markdown": "# Example Domain\n\nThis domain is for use in
illustrative examples...",
  "content_length": 1234
}
```

`POST /scrape/batch`

Scrape plusieurs URLs en parallele.

PROF

```
curl -X POST http://localhost:8000/scrape/batch \
  -H "Content-Type: application/json" \
  -d '["https://example.com", "https://example.org"]'
```

Configuration

Variables d'environnement:

Variable	Default	Description
`HOST`	`0.0.0.0`	Adresse d'ecoute
`PORT`	`8000`	Port d'ecoute

Integration avec Python.SEO.Gemini

L'app Flask utilise ce micro-service via le client `scraper.py`:

```
# Dans le .env de Python.SEO.Gemini
SCRAPER_SERVICE_URL=http://localhost:8000
SCRAPER_TIMEOUT=60
```

Documentation API

Swagger UI disponible sur: http://localhost:8000/docs