````
# 法令知識ベース連携モジュール

法令知識ベース連携モジュールは、e-Gov 法令 API を利用して日本法の条文を取得・パース・インデックス化し、対話システム（Gemini）や上流モジュールが利用できるように「条文検索 API」「条文要約 API」「論点抽出 API」を提供します。

## 機能概要

- **法令取得**: e-Gov API から法令データを取得・パース
- **条文検索**: 法令や条文を検索・取得
- **条文要約**: AI（Gemini）を使用した条文要約
- **論点抽出**: 複数条文から法的論点を抽出
- **キャッシュ**: Redis を使用したパフォーマンス最適化
- **同期バッチ**: 定期的な法令データ更新

## システム要件

- Python 3.10+
- PostgreSQL 14+
- Redis 6+
- Google Gemini API Key

## インストール

### 1. 依存パッケージのインストール

```bash
pip install -r requirements.txt
```

### 2. 環境変数の設定

`.env` ファイルを作成し、以下の環境変数を設定してください：

```bash
cp env.example .env
````
````

```
```

`.env` ファイルを編集:

```env
# Google Gemini API
GEMINI_API_KEY=your_gemini_api_key_here
GEMINI_MODEL=gemini-1.5-flash

# e-Gov API
E_GOV_API_KEY=your_egov_api_key_here
E_GOV_BASE_URL=https://elaws.e-gov.go.jp/api/1

# PostgreSQL Database
DATABASE_URL=postgresql://user:password@localhost:5432/law_chat_db

# Redis Cache
REDIS_URL=redis://localhost:6379/0
CACHE_TTL=86400

# ログ設定
LOG_LEVEL=INFO
```

3. データベースの初期化

```bash
# Alembic でマイグレーション(将来実装)
alembic upgrade head
```

使用方法

開発サーバーの起動

```bash
```

```
python -m app.main
```

または

```bash
uvicorn app.main:app --reload --port 8083
```

API エンドポイント

1. 法令リスト取得

```bash
GET /laws/list?page=1&per_page=20&law_type=法律
```

レスポンス:
```json
{
  "laws": [
    {
      "law_id": "CIVIL_LAW_001",
      "title": "民法",
      "law_no": "明治 29 年法律第 89 号",
      "law_type": "法律",
      "enact_date": "1896-04-27T00:00:00"
    }
  ],
  "total": 1,
  "page": 1,
  "per_page": 20
}
```

2. 法令詳細取得

```bash
GET /laws/{law_id}
```

レスポンス:

```json
{
  "law_id": "CIVIL_LAW_001",
  "title": "民法",
  "law_no": "明治 29 年法律第 89 号",
  "enact_date": "1896-04-27T00:00:00",
  "articles": [
    {
      "article_no": "第 1 条",
      "heading": "私権の内容",
      "text": "私権は、公共の福祉に適合しなければならない...",
      "structure": {...}
    }
  ],
  "metadata": {...}
}
```

3. 条文取得

```bash
GET /laws/{law_id}/articles/{article_no}
```

レスポンス:

```json
{
  "article_no": "第 1 条",
  "heading": "私権の内容",
  "text": "私権は、公共の福祉に適合しなければならない。",
```

```
  "structure": {
    "items": [
      {"level": 1, "text": "私権は、公共の福祉に適合しなければならない。"}
    ]
  }
}
```

4. 条文要約

```bash
POST /laws/{law_id}/articles/{article_no}/summarize
Content-Type: application/json

{
  "max_length": 200,
  "style": "plain"
}
```

レスポンス:

```json
{
  "summary_text": "私権は公共の福祉に適合する必要があり...",
  "original_reference": {
    "law_id": "CIVIL_LAW_001",
    "article_no": "第１条"
  },
  "citations": [],
  "style": "plain",
  "word_count": 150
}
```

5. 論点抽出

```bash
POST /laws/extract_topics
Content-Type: application/json

{
  "texts": ["条文 1", "条文 2", "条文 3"],
  "mode": "topic_extraction",
  "max_topics": 5
}
```

レスポンス:
```json
{
  "topics": [
    {
      "id": "1",
      "title": "私権と公共の福祉",
      "description": "私権の行使は公共の福祉に適合する必要がある...",
      "source_refs": ["民法第 1 条", "民法第 2 条"]
    }
  ],
  "relations": [
    {
      "topic_id_1": "1",
      "topic_id_2": "2",
      "relation_type": "補足"
    }
  ]
}
```

6. 法令検索

```bash
POST /laws/search?keyword=民法&page=1&per_page=20
```

```
```

ETL バッチ（同期処理）

フル同期（初回インポート）

```bash
python -m app.scripts.sync_egov --mode full
```

差分更新

```bash
python -m app.scripts.sync_egov --mode update
```

Cron 設定（毎日午前 2 時に実行）

```bash
# crontab -e
0 2 * * * cd /path/to/WP2-3 && python -m app.scripts.sync_egov --mode update
```

テスト

単体テストの実行

```bash
# 全てのテストを実行
pytest

# 特定のテストファイルを実行
pytest tests/test_parser.py

# カバレッジレポート付きで実行
pytest --cov=app tests/
```

```
```

テスト結果例

```
tests/test_parser.py::test_parse_xml PASSED
tests/test_parser.py::test_extract_articles PASSED
tests/test_api.py::test_root_endpoint PASSED
```

プロンプトテンプレート

要約プロンプト(summarize_ja.txt)

法令条文の要約プロンプトテンプレート

論点抽出プロンプト(extract_topics_ja.txt)

法令条文からの論点抽出プロンプトテンプレート

アーキテクチャ

```
app/
├── main.py             # FastAPI アプリケーション
├── config.py           # 設定管理
├── logger.py           # ログ設定
├── schemas.py          # Pydantic スキーマ
├── models/             # SQLAlchemy モデル
│   └── models.py
├── services/           # ビジネスロジック
│   ├── egov_client.py   # e-Gov API クライアント
│   ├── xml_parser.py    # XML パーサー
│   ├── summarizer.py    # 要約サービス
│   ├── topic_extractor.py # 論点抽出サービス
│   └── cache_service.py # キャッシュサービス
```

```
├── clients/            # 外部 API クライアント
│   └── gemini_client.py # Gemini API クライアント
├── api/                # API ルーター
│   └── laws.py         # 法令 API
├── scripts/            # バッチスクリプト
│   └── sync_egov.py    # e-Gov 同期バッチ
└── utils/              # ユーティリティ
    └── error_mapping.py # エラーマッピング

prompt_templates/       # プロンプトテンプレート
├── summarize_ja.txt
└── extract_topics_ja.txt

tests/                  # テスト
├── test_parser.py
└── test_api.py
```

データベーススキーマ

legal_refs(法令マスタ)

- law_id (PK): 法令 ID
- title: 法令名
- law_no: 法令番号
- law_type: 法令種別
- enact_date: 施行年月日
- source_url: e-Gov API の URL
- raw_xml: 元の XML データ
- created_at, updated_at: タイムスタンプ

articles(条文)

- article_id (PK): 条文 ID
- law_id (FK): 法令 ID
- article_no: 条番号

```
- heading: 見出し
- text: 条文本文
- parsed_json: 正規化された構造データ
- created_at, updated_at: タイムスタンプ

### article_embeddings(条文埋め込み)

- embedding_id (PK): 埋め込み ID
- article_id (FK): 条文 ID
- embedding: ベクトルデータ
- model_name: モデル名
- created_at, updated_at: タイムスタンプ

### sync_log(同期ログ)

- sync_id (PK): 同期 ID
- sync_type: 同期種別
- started_at, finished_at: 開始・終了時刻
- result_count: 取得件数
- status: 状態
- error_message: エラーメッセージ

## 開発ガイド

### API キーの取得

1. **Google Gemini API Key**
   - [Google AI Studio](https://makersuite.google.com/app/apikey) で取得

2. **e-Gov API Key**
   - [e-Gov API](https://www.e-gov.go.jp/) で登録(利用規約を確認)

### デバッグ

ログレベルを `DEBUG` に設定:
```

```env
LOG_LEVEL=DEBUG
```

モックモード

Gemini API キーが設定されていない場合、自動的にモックモードで動作します:

- 要約: 条文の先頭部分を返す
- 論点抽出: 簡易的なキーワード抽出を返す

トラブルシューティング

データベース接続エラー

```bash
# PostgreSQL が起動しているか確認
pg_isready

# 接続テスト
psql $DATABASE_URL
```

Redis 接続エラー

```bash
# Redis が起動しているか確認
redis-cli ping

# 接続テスト
redis-cli -u $REDIS_URL ping
```

API レート制限エラー

e-Gov API のレート制限に注意:

- 1 分間に 60 リクエスト
- 日次制限あり（API 仕様を確認）

ライセンス

このプロジェクトは内部使用目的です。

連絡先

問題や質問がある場合は、プロジェクトメンテナーに連絡してください。