

Excellent choix ! Un assistant de recherche juridique pour vos études en droit est un projet parfait :

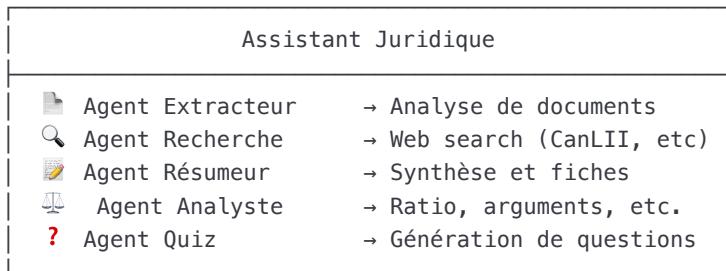
- Vous êtes l'utilisateur cible (motivation quotidienne)
- Le domaine juridique réutilise certains concepts du projet actuel
- Beaucoup de potentiel pour des agents intelligents

Vision : Assistant d'études en droit

Fonctionnalités possibles

Fonctionnalité	Description
Analyse de jurisprudence	Extraire les faits, ratio decidendi, obiter dicta d'un jugement
Résumé de doctrine	Résumer articles et chapitres de manuels
Recherche juridique	Trouver des sources pertinentes (CanLII, Légis Québec)
Fiches de révision	Générer des fiches à partir de vos notes
Quiz/Flashcards	Tester vos connaissances
Plan de dissertation	Aider à structurer une réponse juridique

Agents envisageables



Prochaines étapes proposées

Phase 1 : Réorientation (cette semaine)

1. Renommer/refactorer le projet (ex: legal-study-assistant)
2. Adapter les modèles de données (Dossier → Cours/Matière, Document → Source)
3. Simplifier le workflow existant pour un premier use case

Phase 2 : Premier use case fonctionnel

- Choisir UN cas d'usage prioritaire (ex: résumé de jurisprudence)
- Adapter les agents existants
- Tester avec de vrais documents de vos cours

Phase 3 : Exploration patterns avancés

- RAG avec vos notes de cours
- Agents autonomes avec recherche web
- Multi-agents pour analyse approfondie

Questions pour préciser :

1. Quel serait votre **premier besoin concret** ? (résumer des jugements, créer des fiches, rechercher de la jurisprudence, etc.)
2. Préférez-vous **réutiliser ce repo** (refactor) ou **partir d'un nouveau projet** propre ?

Mon premier besoin concret est : résumer des jugements

Je souhaite partir un nouveau projet, mais je crois que nous pouvons réutiliser une grande partie du code déjà développé. Je souhaite toujours utiliser Agno, SurrealDB et permettre d'utiliser Claude Anthropic, ou MLX, ou Ollama ou HuggingFace.

Excellent ! Créons ce nouveau projet ensemble.

Proposition : Legal Study Assistant

Nom du projet

Quelques suggestions :

- legal-assistant - Simple et clair
- juristudy - Combinaison juridique + étude
- lexis - Référence au droit (lex = loi en latin)
- casebrief - Terme juridique pour résumé de jugement

Architecture prévue

```
legal-assistant/
├── backend/
│   ├── config/
│   │   ├── settings.py      # ← Réutiliser de notary
│   │   └── models.py        # ← Réutiliser (providers LLM)
│   ├── services/
│   │   ├── model_factory.py # ← Réutiliser (Claude/MLX/Ollama/HF)
│   │   ├── surreal_service.py
│   │   └── case_summary_service.py # NOUVEAU
│   ├── workflows/
│   │   └── summarize_judgment.py    # NOUVEAU (basé sur analyse_dossier)
│   ├── models/
│   │   ├── judgment.py       # NOUVEAU
│   │   └── summary.py        # NOUVEAU
│   ├── routes/
│   │   └── judgments.py     # NOUVEAU
│   └── main.py              # ← Adapter de notary
├── frontend/                # ← Adapter de notary (Next.js + shadcn)
└── docker-compose.yml       # ← Réutiliser (SurrealDB)
└── docs/
```

Agents pour résumé de jugement

```
Workflow: SummarizeJudgment
├── Agent Extracteur
│   → Extraire : parties, tribunal, date, numéro
├── Agent Analyseur
│   → Identifier : faits, questions en litige, arguments
├── Agent Synthétiseur
│   → Produire : ratio decidendi, obiter dicta, conclusion
└── Agent Formateur
    → Générer : fiche structurée (format case brief)
```

Format de sortie (Case Brief)

```
## [Nom de l'affaire] (Année)
**Tribunal:** Cour supérieure du Québec
**Référence:** 2024 QCCS 1234
```

Faits

- Point 1
- Point 2

Questions en litige

1. Question principale

Règles de droit applicables

- Article X du Code civil
- Principe de jurisprudence Y

Analyse / Ratio decidendi

...

Conclusion / Dispositif

...

Notes personnelles

(espace pour vos annotations)

Prochaines étapes**Voulez-vous que je :**

1. Crée la structure du nouveau projet dans un dossier adjacent (~/legal-assistant/) ?
2. Copie et adapte les fichiers réutilisables de notary ?
3. Implémente le premier workflow de résumé de jugement ?

Aussi, quel nom préférez-vous ?

- legal-assistant
- juristudy
- lexis
- casebrief
- Autre suggestion ?

Oui à toutes les questions et je préfère le nom "legal-assistant" comme nom de projet

Update Todos

Creating legal-assistant project structure

Copier et adapter les fichiers réutilisables de notary

Créer les modèles de données (Judgment, Summary)

Implémenter le workflow de résumé de jugement

Créer les routes API

Adapter le frontend

Tester le workflow complet

Bash mkdir -p /home/user/legal-

assistant/{backend/{config, services, workflows, models, routes, data/{surreal, uploads}, tests}, frontend, docs}

(Aucun contenu)

Bash cp /home/user/notary/docker-compose.yml /home/user/legal-assistant/ && cp

/home/user/notary/backend/pyproject.toml /home/user/legal-assistant/backend/

(Aucun contenu)

Read /home/user/notary/backend/config/settings.py

155 lignes lues

Read /home/user/notary/backend/services/model_factory.py

352 lignes lues

Read /home/user/notary/backend/config/models.py

444 lignes lues

Update Todos

- Crée la structure du projet legal-assistant
- Copier les fichiers réutilisables de notary
- Créer les modèles de données (Judgment, Summary)
- Implémenter le workflow de résumé de jugement
- Créer les routes API
- Adapter le frontend
- Tester le workflow complet

Write /home/user/legal-assistant/backend/config/settings.py

```
"""
Configuration centralisée de l'application Legal Assistant.

Ce fichier charge et valide toutes les variables d'environnement
en utilisant Pydantic Settings pour une gestion type-safe.
"""

from pathlib import Path
from typing import Literal

from pydantic import Field
from pydantic_settings import BaseSettings, SettingsConfigDict

class Settings(BaseSettings):
    """
    Configuration de l'application.

    Les valeurs sont automatiquement chargées depuis:
    1. Variables d'environnement
    """

    model_config = SettingsConfigDict(env_file=".env", env_file_encoding="utf-8")
```

Show full diff (128 more lines)

```
Bash cp /home/user/notary/backend/config/models.py /home/user/legal-assistant/backend/config/ && cp
    /home/user/notary/backend/services/model_factory.py /home/user/legal-assistant/backend/services/
(Aucun contenu)
```

Write /home/user/legal-assistant/backend/config/__init__.py

```
"""Configuration module for Legal Assistant."""

from .settings import settings

__all__ = ["settings"]
```

Write /home/user/legal-assistant/backend/services/__init__.py

```
"""Services module for Legal Assistant."""

from .model_factory import create_model

__all__ = ["create_model"]
```

Update Todos

- Crée la structure du projet legal-assistant
- Copier et adapter les fichiers réutilisables de notary
- Creating data models
- Implémenter le workflow de résumé de jugement
- Créer les routes API
- Adapter le frontend

Tester le workflow complet

Write /home/user/legal-assistant/backend/models/judgment.py

```
"""
Modeles de donnees pour les jugements juridiques.

Ce module definit les schemas Pydantic pour:
- Judgment: Un jugement/decision de justice
- JurisdictionType: Types de juridictions (Quebec, Canada, etc.)
- CourtLevel: Niveaux de tribunaux
"""

from datetime import date, datetime
from typing import Literal, Optional

from pydantic import BaseModel, Field

# Types de juridictions
JurisdictionType = Literal[
    "quebec",      # Droit civil quebecois
    "canada",      # Droit federal canadien
    "common_law",   # Provinces de common law
]
```

[Show full diff \(120 more lines\)](#)

Write /home/user/legal-assistant/backend/models/summary.py

```
"""
Modeles de donnees pour les resumes de jugements (Case Briefs).

Ce module definit les schemas Pydantic pour:
- CaseBrief: Resume structure d'un jugement
- LegalIssue: Question en litige
- LegalRule: Regle de droit applicable
- AnalysisPoint: Point d'analyse
"""

from datetime import datetime
from typing import Literal, Optional

from pydantic import BaseModel, Field

# Type de resume
SummaryType = Literal[
    "case_brief",      # Resume complet (format case brief)
    "quick_summary",   # Resume rapide (points cles)
]
```

[Show full diff \(157 more lines\)](#)

Write /home/user/legal-assistant/backend/models/__init__.py

```
"""Models module for Legal Assistant."""

from .judgment import (
    CourtLevel,
    Judgment,
    JudgmentBase,
    JudgmentCreate,
    JudgmentList,
    JudgmentStatus,
    JudgmentUpdate,
    JurisdictionType,
    LegalDomain,
)
from .summary import (
```

```
AnalysisPoint,
CaseBrief,
CaseBriefCreate,
CaseBriefUpdate,
LegalIssue,
LegalRule,
```

Show full diff (32 more lines)

Update Todos

- Créer la structure du projet legal assistant
- Copier et adapter les fichiers réutilisables de notary
- Créer les modèles de données (Judgment, Summary)
- Implementing judgment summary workflow
- Créer les routes API
- Adapter le frontend
- Tester le workflow complet

Write /home/user/legal-assistant/backend/workflows/summarize_judgment.py

```
"""
Workflow de resume de jugements juridiques.

Ce workflow utilise 4 agents specialises pour analyser un jugement
et produire un resume structure (Case Brief).

Agents:
1. ExtractorAgent – Extrait les informations de base (parties, tribunal, date)
2. AnalyzerAgent – Identifie les faits, questions en litige, arguments
3. SynthesizerAgent – Extrait le ratio decidendi et la conclusion
4. FormatterAgent – Genere le case brief final structure

Usage:
from workflows.summarize_judgment import create_summarize_workflow
from services.model_factory import create_model

model = create_model("ollama:qwen2.5:7b")
workflow = create_summarize_workflow(model=model)

result = workflow.run(judgment_text="...")

```

Show full diff (426 more lines)

Write /home/user/legal-assistant/backend/workflows/__init__.py

```
"""Workflows module for Legal Assistant."""

from .summarize_judgment import SummarizeJudgmentWorkflow, create_summarize_workflow

__all__ = ["SummarizeJudgmentWorkflow", "create_summarize_workflow"]
```

Write /home/user/legal-assistant/backend/main.py

```
"""
Legal Assistant API - Point d'entree principal.

Application FastAPI pour l'assistant d'études juridiques.

import logging
from contextlib import asynccontextmanager

import uvicorn
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
```

```
from config.settings import settings

# Configuration du logging
logging.basicConfig(
    level=logging.INFO,
    format"%(asctime)s - %(name)s - %(levelname)s - %(message)s"
)
```

Show full diff (109 more lines)

Read /home/user/legal-assistant/backend/pyproject.toml

131 lignes lues

Edit /home/user/legal-assistant/backend/pyproject.toml

```
[project]
name = "notary-backend"
version = "0.1.0"
description = "Assistant IA pour notaires - Backend API"
name = "legal-assistant"
version = "0.1.0"
description = "Assistant d'etudes juridiques - Resume de jugements"
```

Read /home/user/legal-assistant/docker-compose.yml

130 lignes lues

Write /home/user/legal-assistant/docker-compose.yml

```
services:
  # =====
  # SurrealDB - Base de donnees multi-modele
  # =====
  surrealdb:
    image: surrealdb/surrealdb:latest
    container_name: legal-assistant-surrealdb
    restart: unless-stopped

    environment:
      TZ: America/Montreal

    ports:
      - "8001:8000" # HTTP + WebSocket

    volumes:
      # Persister les donnees RocksDB
      - ./backend/data/surrealdb:/data
      # Script d'initialisation (optionnel)
      - ./backend/data/surreal/:init
```

Show full diff (19 more lines)

Write /home/user/legal-assistant/README.md

```
# Legal Assistant

Assistant d'etudes juridiques pour etudiants en droit - Resume automatique de jugements.

## Fonctionnalites

- **Resume de jugements**: Analyse automatique de jugements pour generer des case briefs structures
- **Multi-providers LLM**: Support Ollama, Claude, MLX, HuggingFace
- **Persistance**: SurrealDB pour stocker jugements et resumes
- **API REST**: FastAPI avec documentation Swagger

## Structure du projet

```
```

```

```

legal-assistant/
└── backend/
    ├── config/          # Configuration (settings, models LLM)
    ├── models/           # Modeles Pydantic (Judgment, Summary)
    ├── workflows/        # Workflows Agno (summarize_judgment)
    └── services/         # Services (model_factory)

```

Show full diff (141 more lines)

Write /home/user/legal-assistant/backend/.env.example

```

# =====
# Configuration Legal Assistant
# =====
# Copiez ce fichier vers .env et adaptez les valeurs

# ===== API =====
API_HOST=0.0.0.0
API_PORT=8000
API_RELOAD=true
DEBUG=true

# ===== Securite =====
SECRET_KEY=change-this-in-production-use-a-long-random-string

# ===== SurrealDB =====
SURREAL_URL=ws://localhost:8001/rpc
SURREAL_NAMESPACE=legal
SURREAL_DATABASE=legal_db
SURREAL_USERNAME=root
SURREAL_PASSWORD=root

```

Show full diff (34 more lines)

Update Todos

- Créer la structure du projet legal-assistant
- Copier et adapter les fichiers réutilisables de notary
- Créer les modèles de données (Judgment, Summary)
- Implémenter le workflow de résumé de jugement
- Créer les fichiers de configuration (.env, .gitignore, README)
- Créer les routes API
- Adapter le frontend
- Tester le workflow complet

Write /home/user/legal-assistant/.gitignore

```

# Python
__pycache__/
*.py[cod]
*$py.class
*.so
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
*.egg-info/
.installed.cfg

```

Show full diff (59 more lines)

```
Bash touch /home/user/legal-assistant/backend/data/uploads/.gitkeep && find /home/user/legal-assistant -  
type f | head -30  
/home/user/legal-assistant/README.md  
/home/user/legal-assistant/.gitignore  
/home/user/legal-assistant/docker-compose.yml  
... +14 lignes
```

Projet legal-assistant cree avec succes!

Voici ce qui a ete cree dans /home/user/legal-assistant/ :

Structure du projet

```
legal-assistant/  
|   __init__.py  
|   config/  
|       __init__.py      # Configuration adaptee (namespace: legal)  
|       settings.py  
|       models.py         # Config LLM (copie de notary)  
|   models/  
|       __init__.py  
|       judgment.py      # Modele Judgment (nouveau)  
|       summary.py        # Modele CaseBrief (nouveau)  
|   workflows/  
|       __init__.py  
|       summarize_judgment.py  # Workflow 4 agents (nouveau)  
|   services/  
|       __init__.py  
|       model_factory.py # Factory LLM (copie de notary)  
|   routes/              # (vide - a creer)  
|   tests/               # (vide - a creer)  
|   data/  
|       surreal/  
|       uploads/  
|   main.py              # Point d'entree FastAPI  
|   pyproject.toml        # Dependances Python  
|   .env.example          # Variables d'environnement  
|   frontend/             # (vide - a adapter de notary)  
|   docs/                 # (vide)  
|   docker-compose.yml    # SurrealDB configure  
|   .gitignore  
|   README.md
```

Fichiers clees crees

Fichier	Description
models/judgment.py	Modele pour les jugements (parties, tribunal, date, etc.)
models/summary.py	Modele CaseBrief (faits, ratio, obiter, conclusion)
workflows/summarize_judgment.py	Workflow avec 4 agents specialises
config/settings.py	Configuration adaptee (namespace legal)

Prochaines etapes

Pour tester le projet :



```
# 1. Aller dans le projet
cd /home/user/legal-assistant

# 2. Demarrer SurrealDB
docker-compose up -d

# 3. Installer les dependances
cd backend
uv sync

# 4. Copier le fichier d'environnement
cp .env.example .env

# 5. (Si vous avez Ollama) Telecharger le modele
ollama pull qwen2.5:7b

# 6. Tester le workflow directement
uv run python workflows/summarize_judgment.py

# 7. Lancer l'API
uv run python main.py
```

Reste à faire

1. **Routes API** (routes/judgments.py) - Endpoints pour upload et resume
2. **Frontend** - Adapter de notary ou créer nouveau
3. **Tests** - Ajouter tests unitaires et intégration

Voulez-vous que je continue avec les routes API ou le frontend ?