

**Equipe :** AZZOUZ Chaimae  
AIDOUN Ibtissam  
BOUHNINE Safaa  
BOUSSAID Meryeme

Ce pdf contient le compte rendu de sprint 0 et sprint 1 où on a préparé le workflow de travail de ce projet avant d'entamer l'application et le codage.

# Workflow complet : Chatbot de recommandation de cours universitaires avec ontologie pédagogique

## **Vue d'ensemble du projet :**

Utilisateur → Chatbot (NLP) → Agent IA → Ontologie (Web Sémantique) → Recommandations

## **Architecture complète :**

- ❖ Phase 1 : Construction de l'ontologie (Web Sémantique)
- ❖ Phase 2 : Agent IA avec raisonnement
- ❖ Phase 3 : NLP et compréhension du dialogue
- ❖ Phase 4 : Intégration du chatbot
- ❖ Phase 5 : Interface utilisateur

## **PHASE 1 : Construction de l'ontologie pédagogique :**

### **Étape 1.1 : Modélisation de l'ontologie :**

#### **=> Classes principales à créer :**

- Cours (Course)
- Étudiant (Student)
- Compétence (Skill)
- Domaine (Domain)
- Prérequis (Prerequisite)
- Niveau (Level: Débutant, Intermédiaire, Avancé)

**=> Relations (Object Properties) :**

- aPrerequis (Course → Course)
- enseigneCompetence (Course → Skill)
- appartientADomaine (Course → Domain)
- aNiveau (Course → Level)
- possèdeCompetence (Student → Skill)
- aInteretPour (Student → Domain)
- aSuivi (Student → Course)

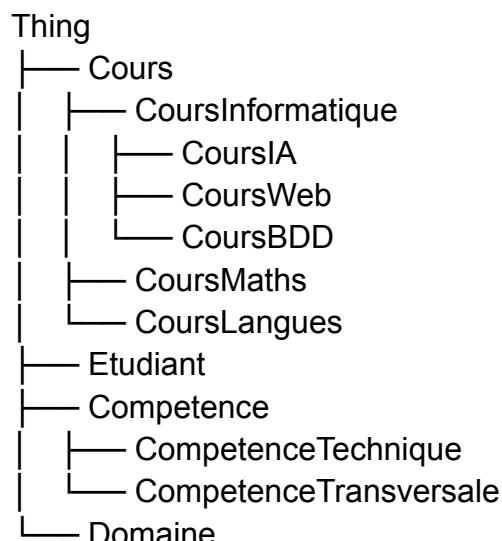
**=> Data Properties :**

- nomCours (string)
- codeCours (string)
- credits (int)
- duree (int)
- difficulte (int 1-5)
- description (string)

**Étape 1.2 : Création de l'ontologie avec Protégé :**

**=> Outil :** Protégé Desktop (gratuit)

1. Télécharger Protégé : <https://protege.stanford.edu/>
2. Créer une nouvelle ontologie OWL
3. Définir les classes hiérarchiquement :



4. Ajouter les propriétés d'objet et de données
5. Créer des règles SWRL (optionnel) pour inférences automatiques

#### **Étape 1.3 : Peuplement de l'ontologie avec des données :**

=> Créez un fichier Python pour peupler l'ontologie.

=> Ou utiliser un fichier CSV pour automatiser.

#### **PHASE 2 : Agent IA avec raisonnement :**

##### **Étape 2.1 : Moteur de raisonnement SPARQL :**

=> Créez un module de requêtes SPARQL.

##### **Étape 2.2 : Logique d'agent décisionnel :**

#### **PHASE 3 : NLP et compréhension du dialogue :**

##### **Étape 3.1 : Extraction d'intentions et entités (sans entraînement)**

Utiliser spaCy + modèles pré-entraînés :

##### **Étape 3.2 : Gestion du contexte conversationnel :**

#### **PHASE 4 : Intégration du Chatbot :**

##### **Étape 4.1 : Orchestration complète.**

#### **PHASE 5 : Interface utilisateur :**

##### **Étape 5.1 : Interface avec Streamlit (simple et rapide)**

#### **PHASE 6 : Structure du projet et dépendances :**

##### **Structure des fichiers :**

university-chatbot/



  |

  |   data/

  |      |   cours.csv

```

competences.csv
university_ontology.ttl

src/
    __init__.py
    sparql_reasoner.py
    recommendation_agent.py
    nlp_processor.py
    conversation_manager.py
    chatbot.py

scripts/
    populate_ontology.py

app.py
requirements.txt
README.md

```

### **Requirements.txt :**

### **PHASE 7 : Exécution et tests :**

#### **Étape 7.1 : Setup initial :**

1. Installer les dépendances

pip install -r requirements.txt

python -m spacy download fr\_core\_news\_md

2. Peupler l'ontologie

python scripts/populate\_ontology.py

3. Lancer l'application

streamlit run app.py

#### **Étape 7.2 : Scénarios de test :**

##### **Test 1 : Recommandation simple**

User: "Je cherche des cours en intelligence artificielle"

→ Le chatbot extrait l'intention et le domaine

→ L'agent interroge l'ontologie via SPARQL

→ Retourne les cours d'IA avec prérequis

## **Test 2 : Vérification de prérequis**

User: "Est-ce que je peux suivre le cours IA-401 ?"

→ NLP extrait le code cours

→ Agent vérifie dans l'ontologie si l'étudiant a les prérequis

→ Répond oui/non avec explications

## **Test 3 : Parcours d'apprentissage**

User: "Je veux devenir expert en Machine Learning, par où commencer ?"

→ Agent calcule le chemin dans le graphe de prérequis

→ Retourne une séquence de cours ordonnée

## **PHASE 8 : Améliorations et extensions :**

### **Extensions possibles :**

#### **1. Améliorer le NLP**

- Ajouter un modèle de similarité sémantique pour mieux matcher les questions
- Utiliser des embeddings pour trouver des cours similaires
- Gérer les synonymes et variations de langage

#### **2. Enrichir l'ontologie**

- Ajouter des données sur les professeurs
- Intégrer les horaires et disponibilités
- Ajouter des avis étudiants

#### **3. Raisonnement avancé**

- Implémenter des règles SWRL plus complexes
- Ajouter un moteur d'inférence (Pellet, HermiT)
- Optimisation de parcours avec contraintes

#### **4. Interface**

- Visualisation du graphe de cours
- Dashboard de progression
- Intégration calendrier