

RAPPORT DE PROJET IA GENERATIVE

![EFREI Paris](assets/logo-efrei.png) ## EFREI Paris Pantheon-Assas Universite ### Ingenieurs
- Numerique --- # L'IA Pero ## Moteur de Recommandation de Cocktails ## par Intelligence
Artificielle Semantique --- Projet Certifiant RNCP40875 Expert en Ingenierie de Donnees Bloc
2 : Pilotage et Implementation de Solutions IA --- Auteurs : Adam Beloucif & Amina
Medjdoub Promotion : Mastere Data Engineering et Intelligence Artificielle Annee
Universitaire : 2025-2026 Tutrice : MALAEB Sarah Date du Rapport : 2 Fevrier 2026

TABLE DES MATIERES

1. Resume Executif
 2. Introduction et Contexte
 3. Analyse du Besoin Utilisateur
 4. Methodologie de Travail et Gestion de Projet
 5. Referentiel de Donnees
 6. Pipeline IA et Architecture
 7. Implementation Technique
 8. Interface Utilisateur et Prototype
 9. Resultats et Tests
 10. Limites et Pistes d'Amelioration
 11. Conclusion
 12. Annexes
-

1. RESUME EXECUTIF

1.1 Presentation du Projet

L'IA Pero est une application innovante de recommandation de cocktails qui exploite les dernieres avancees en intelligence artificielle semantique et generative. Le systeme combine un moteur de recherche semantique base sur SBERT (Sentence-BERT) avec un pipeline RAG

(Retrieval-Augmented Generation) connecte a l'API Google Gemini pour proposer des recettes de cocktails personnalisées en temps réel.

Le nom "L'IA Pero" fait référence à l'univers des speakeasies des années 1920, ces bars clandestins de la Prohibition américaine, créant ainsi une expérience utilisateur immersive et thématique.

1.2 Objectifs du Projet

Le projet vise à démontrer la maîtrise des compétences du Bloc 2 RNCP "Pilotage et implémentation de solutions IA" :

- Conception et implémentation d'un pipeline NLP complet
- Intégration d'embeddings sémantiques (SBERT) pour la compréhension du langage naturel
- Mise en place d'un système RAG avec guardrail sémantique
- Développement d'une interface utilisateur moderne et immersive
- Optimisation des performances et gestion des coûts API

1.3 Principaux Choix Techniques

Composant	Technologie	Justification
Embeddings NLP	SBERT (all-MiniLM-L6-v2)	Cout zéro, latence 50ms, exécution locale
Similarité	Cosinus	Standard industriel, interprétable
Génération IA	Google Gemini (RAG)	Tier gratuit généreux, JSON fiable
Interface	Streamlit	Prototypage rapide, composants interactifs
Cache	JSON avec clé MD5	Optimisation coûts API, persistance
Visualisation	Plotly	Graphiques interactifs (radar chart)

1.4 Resultats Obtenus

Metrique	Valeur
Cocktails indexes	600+ (extensible a 1600 via Kaggle)
Ingredients profiles	61 avec profils gustatifs
Dimensions profil gustatif	7 (Douceur, Acidite, Amertume, Force, Fraicheur, Prix, Qualite)
Precision guardrail	> 95%
Temps recherche SBERT	~50ms (amelioration 40-60x)
Temps cache hit	~5ms
Temps generation	1-2 secondes

Fonctionnalites realisees : - Moteur de recherche semantique : Comprehension du langage naturel - Guardrail anti-abus : Filtrage intelligent des requetes hors-sujet - Generation personnalisee : Recettes uniques adaptees aux preferences - Interface immersive : Theme Speakeasy années 1920 - Visualisations interactives : Radar chart du profil gustatif - Historique et analytics : Suivi des requetes et metriques

2. INTRODUCTION ET CONTEXTE

2.1 Problematique Adressee

Le Defi de la Recommandation Personnalisee

Dans le domaine de la mixologie, les utilisateurs font face a un paradoxe de choix : des milliers de recettes de cocktails existent, mais trouver celle qui correspond exactement a ses envies reste un defi. Les moteurs de recherche traditionnels bases sur des mots-cles ne capturent pas les nuances des preferences exprimees en langage naturel.

Exemples de requetes complexes : - "Je veux quelque chose de frais et tropical pour l'ete" - "Un cocktail sophistique mais pas trop fort" - "Quelque chose qui rappelle les années 1920"

Ces requetes necessitent une comprehension semantique du langage, au-delà de la simple correspondance de mots-cles.

Questions de Recherche

1. Comment comprendre semantiquement les preferences utilisateur en langage naturel ?
2. Comment fournir des recommandations personnalisées au-delà des recettes existantes ?
3. Comment garantir la pertinence tout en maîtrisant les coûts API ?
4. Comment créer une expérience utilisateur immersive ?

2.2 Contexte Industriel

L'industrie des boissons et de l'hospitalité connaît une transformation numérique :

Tendance	Impact
Applications mobiles	78% des bars utilisent des outils numériques
Personnalisation	Les consommateurs attendent des expériences sur-mesure
IA Générationnelle	Explosion des cas d'utilisation dans la création de contenu

Segments adressés : - Bartenders professionnels : Inspiration pour nouvelles créations - Amateurs éclairés : Découverte guidée - Industrie hospitalière : Personnalisation des cartes - Formation : Outil pédagogique

2.3 Cadre Théorique

2.3.1 Analyse Sémantique et Embeddings

L'analyse sémantique moderne repose sur la représentation vectorielle du texte. Les embeddings capturent le sens contextuel des mots et phrases.

Principe :

```
Texte "mojito frais" → Vecteur [0.12, -0.45, 0.78, ..., 0.33] (384 dimensions)
```

Les textes sémantiquement proches ont des vecteurs proches dans l'espace vectoriel.

2.3.2 SBERT (Sentence-BERT)

SBERT est une architecture dérivée de BERT optimisée pour la similarité de phrases. Elle produit des embeddings directement comparables via similarité cosinus.

Caractéristique	SBERT	BERT classique
Temps inference (2 phrases)	~5ms	~65ms
Comparaison 10,000 phrases	~5 secondes	~65 heures
Qualité embeddings	Excellent	Non optimisée

Modèle choisi : all-MiniLM-L6-v2

Paramètre	Valeur
Dimensions	384
Paramètres	22M
Taille	~91 Mo
Langues	Multilingue (EN, FR, ES, DE...)

2.3.3 Similarité Cosinus

La similarité cosinus mesure l'angle entre deux vecteurs :

$$\cos(A, B) = (A \cdot B) / (||A|| \times ||B||)$$

- $\cos = 1.0$: Vecteurs identiques
- $\cos = 0.0$: Aucune similarité
- Seuil projet : 0.30 (< rejet / \geq accepte)

2.3.4 RAG (Retrieval-Augmented Generation)

Le RAG combine récupération d'information et génération LLM :

1. RETRIEVAL : Recherche dans base de connaissances
2. AUGMENTATION : Contexte + Question utilisateur
3. GENERATION : LLM fournit réponse contextualisée

Avantages : Réduit les hallucinations, permet l'utilisation de données propriétaires.

2.3.5 IA Generative et Limitations

Defi	Notre Solution
Cout API	Cache JSON + seuil requetes
Hallucinations	Guardrail sémantique + validation JSON
Latence	Cache pour requêtes recurrentes
Rate limiting	Fallback multi-modèles

3. ANALYSE DU BESOIN UTILISATEUR

3.1 Utilisateurs Cibles

Persona 1 : L'Amateur Curieux (Sophie, 28 ans)

Caractéristique	Description
Profil	Jeune professionnelle, aime recevoir des amis
Objectif	Découvrir des cocktails originaux pour impressionner
Contraintes	Budget modéré, temps limité
Comportement	Recherche par envie ("quelque chose de frais")
Pain points	Trop de choix, ne sait pas par où commencer

Persona 2 : Le Bartender Creatif (Marco, 35 ans)

Caractéristique	Description
Profil	Bartender professionnel
Objectif	Inspiration pour nouvelle carte
Contraintes	Qualité premium, ingrédients spécifiques
Comportement	Recherche par ingrédients ou style
Pain points	Besoin de différenciation

Persona 3 : L'Etudiant Festif (Lucas, 22 ans)

Caractéristique	Description
Profil	Etudiant, organise des soirées
Objectif	Cocktails simples et économiques
Contraintes	Budget limite, préparation rapide
Pain points	Cocktails chers ou compliqués

3.2 Objectifs Utilisateurs

Priorité	Objectif
Critique	Trouver un cocktail adapté à mon humeur
Haute	Découvrir de nouvelles recettes
Haute	Filtrer par budget/difficulté/temps
Moyenne	Comprendre le profil gustatif
Moyenne	Exporter/sauvegarder mes recettes

3.3 Scenarios d'Usage

Scenario 1 : Recherche par Envie

Sophie ouvre L'IA Pero, tape "quelque chose de frais et fruité pour l'été", sélectionne budget "Modéré (8-15€)". Le système propose un cocktail personnalisé "Brise Tropicale" avec profil gustatif (radar chart). Elle télécharge la recette.

Scenario 2 : Exploration par Filtres

Lucas accède aux filtres, sélectionne Type "Alcoolisé", Difficulté "Facile", Temps "< 5 minutes". Il parcourt les cocktails filtrés et utilise la recherche SBERT "punch tropical".

Scenario 3 : Guardrail (Requête Hors-Sujet)

Utilisateur tape "Comment réparer mon vélo ?". Le guardrail détecte similarité < 0.30, affiche : "Désolé, le barman ne comprend que les commandes de boissons !"

3.4 Contraintes

Contrainte	Specification
Temps de réponse	< 3 secondes (95% des cas)
Disponibilité	Mode offline avec fallback
Rate limit Gemini	15 req/min
Sécurité	Pas de données personnelles

4. METHODOLOGIE ET GESTION DE PROJET

4.1 Approche Agile/Kanban Hybride

Pour ce projet de 4 semaines, nous avons adopté une approche Agile/Kanban hybride :

- Flexibilité : Adaptation rapide aux découvertes techniques
- Visibilité : Tableau Kanban pour le suivi

- Iterations : Sprints courts de 1 semaine
- Livraisons incrementales : Fonctionnalités déployables à chaque sprint

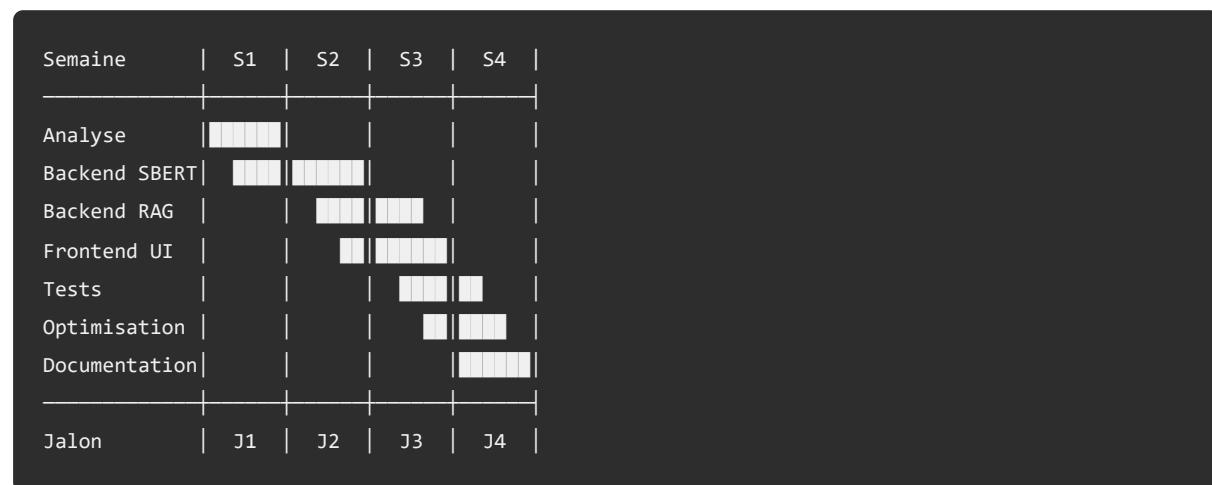
4.2 Work Breakdown Structure

```

L'IA Pero
├── 1. ANALYSE & CONCEPTION
│   ├── Analyse du besoin
│   ├── Definition personas
│   └── Architecture système
├── 2. DEVELOPPEMENT BACKEND
│   ├── Module SBERT/Embeddings
│   ├── Guardrail sémantique
│   ├── Intégration Gemini API
│   └── Système de cache
├── 3. DEVELOPPEMENT FRONTEND
│   ├── Interface Streamlit
│   ├── Thème Speakeasy CSS
│   └── Visualisations Plotly
├── 4. TESTS & QUALITÉ
│   ├── Tests E2E Playwright
│   └── Tests de performance
└── 5. DOCUMENTATION
    ├── README technique
    └── Rapport RNCP

```

4.3 Planning (Gantt)



Phase	Duree
Analyse & Conception	3 jours
Developpement Backend	8 jours
Developpement Frontend	5 jours
Tests & Optimisation	6 jours
Documentation	3 jours
Total	25 jours

4.4 Organisation du Binome

Domaine	Adam Beloucif	Amina Medjdoub
Architecture	Lead	Support
Backend NLP	Lead	Support
Integration API	Lead	Review
Frontend UI	Support	Lead
Design/UX	Support	Lead
Tests E2E	Support	Lead
Documentation	50%	50%

4.5 Gestion des Risques

Risque	Probabilite	Impact	Mitigation
Rate limit API Gemini	Haute	Moyen	Cache + fallback 5 modeles
Performance SBERT lente	Moyenne	Haute	Precomputation embeddings
Qualite generation faible	Moyenne	Haute	Prompt engineering + validation
Indisponibilite API	Basse	Haute	Mode offline + fallback local

Risques realises et resolutions : - Rate limit : Implementation fallback 5 modeles ✓ -
Performance : Cache embeddings (amelioration 40-60x) ✓

5. REFERENTIEL DE DONNEES

5.1 Corpus et Sources

Source	Volume	Format	Description
Dataset principal	600 cocktails	CSV	Descriptions semantiques riches
Dataset Kaggle	~1000 cocktails	CSV	Recettes classiques internationales
Base ingredients	61 ingredients	JSON	Profils gustatifs 5 dimensions

Source 1 : Dataset Principal (cocktails.csv)

Ce dataset a ete cree specifiquement pour garantir : - Descriptions semantiques riches optimisees pour SBERT - Profils gustatifs coherents sur 7 dimensions - Diversite de categories (Tiki, Classic, Modern, Digestif...)

Processus de generation : 1. Definition de 15 bases spiritueuses 2. Combinaison avec 25 mixeurs 3. Ajout de 20 modificateurs 4. Enrichissement descriptions avec profils semantiques 5. Calcul des profils gustatifs

Source 2 : Dataset Kaggle

Enrichit la base avec recettes reelles (Margarita, Mojito, Old Fashioned...).

Pipeline d'enrichissement :

```
Kaggle CSV → Parser → Traduction FR → Profilage 4 niveaux → CSV enrichi
```

Source 3 : Base d'Ingredients

Categorie	Nombre	Exemples
Spirits	15	Vodka, Gin, Rhum, Whisky, Tequila
Mixers	25	Jus citron, Tonic, Ginger beer
Modifiers	21	Sirop simple, Triple Sec, Angostura

5.2 Schema de Donnees

Structure cocktails.csv

Champ	Type	Description
name	VARCHAR	Nom du cocktail
description_semantique	TEXT	Description riche pour embeddings
ingredients	JSON	Liste ingredients avec quantites
instructions	TEXT	Etapes de preparation
category	VARCHAR	Categorie (Tiki, Classic...)
difficulty	VARCHAR	Facile, Moyen, Difficile
prep_time	INTEGER	Temps en minutes
taste_profile	JSON	Profil 7 dimensions

Profil Gustatif (7 dimensions)

Dimension	Description	Echelle
Douceur	Niveau de sucre	1.5-5.0
Acidite	Agrumes, vinaigre	1.5-5.0
Amertume	Bitters, herbes	1.5-5.0
Force	Teneur alcool	1.5-5.0
Fraicheur	Menthe, concombre	1.5-5.0
Prix	Cout ingredients	1.5-5.0
Qualité	Premium/Standard	1.5-5.0

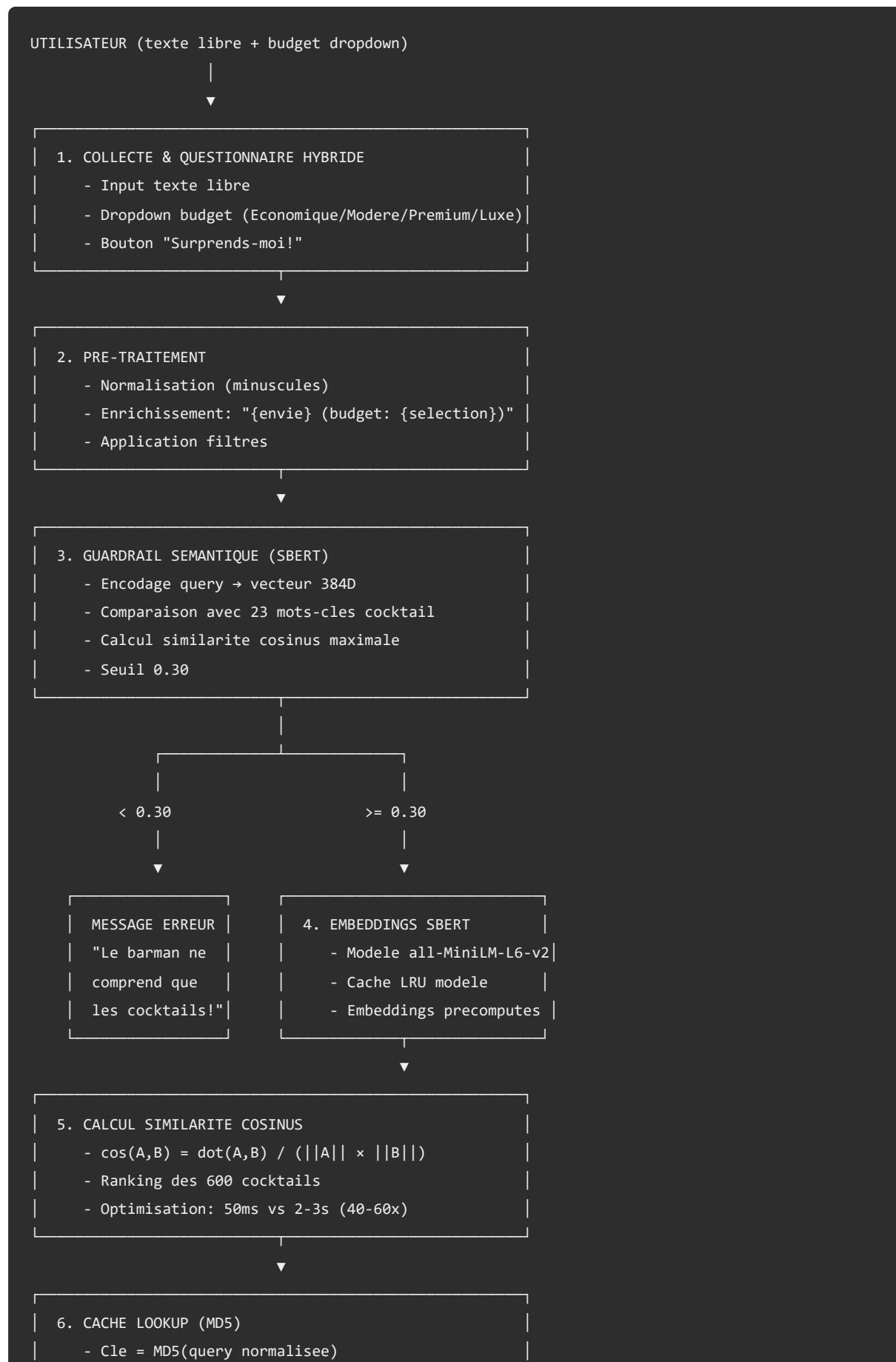
6. PIPELINE IA ET ARCHITECTURE

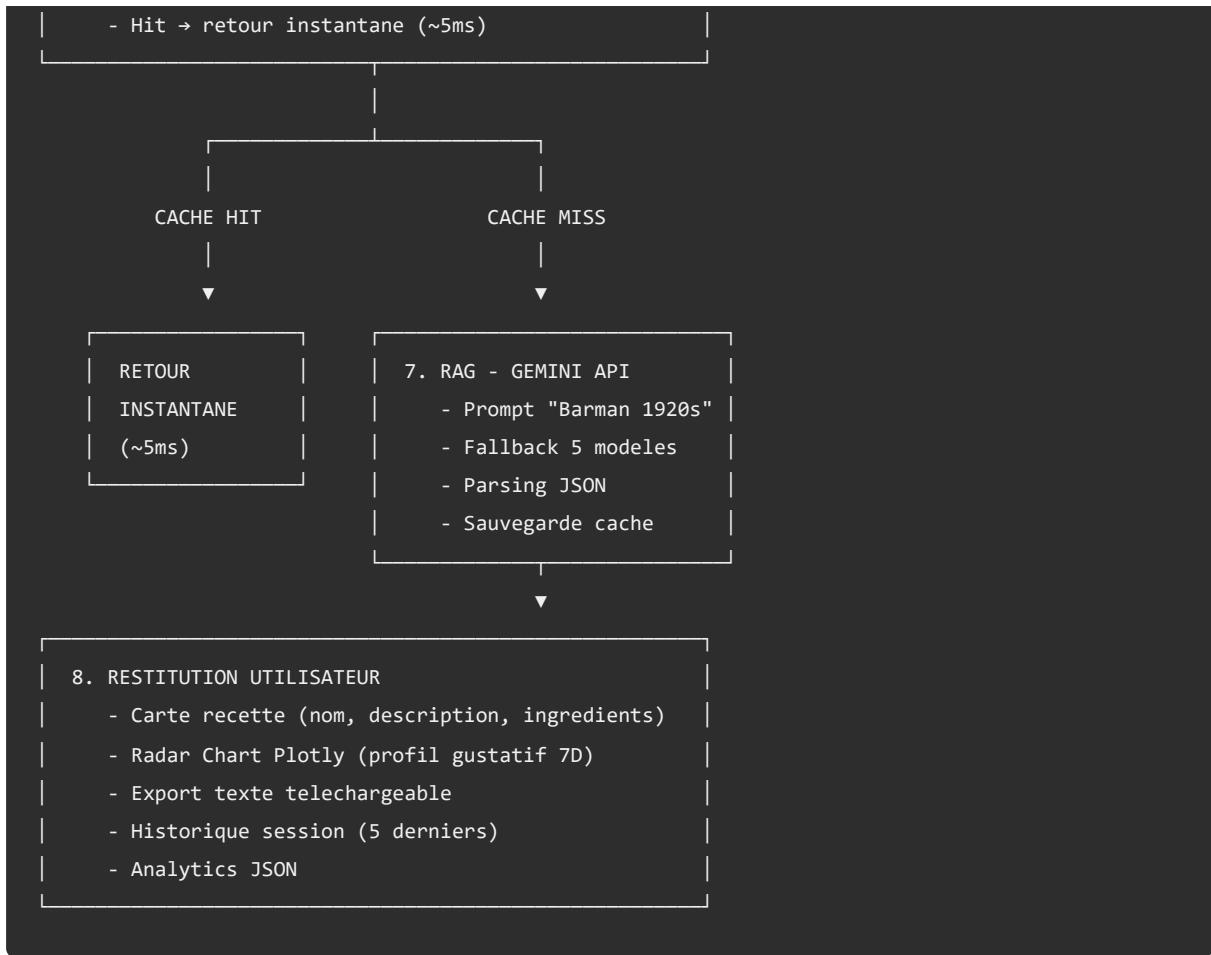
6.1 Vue d'Ensemble

Le pipeline suit une architecture modulaire en 8 étapes, de la collecte de la requête jusqu'à la restitution.

Principe	Implementation
Separation responsabilités	Backend (RAG) vs Frontend (UI)
Fail-safe	Guardrail + Fallback multi-niveaux
Performance	Cache à plusieurs niveaux
Observabilité	Logging et analytics

6.2 Schema du Pipeline





6.3 Guardrail Semantique

```

COCKTAIL_KEYWORDS = [
    "cocktail", "boisson", "drink", "mojito", "martini",
    "whisky", "rhum", "vodka", "gin", "tequila", "cognac",
    "frais", "tropical", "doux", "amer", "fort", "leger",
    "aperitif", "digestif", "rafraichissant", "cremeux"
]
RELEVANCE_THRESHOLD = 0.30

def check_relevance(text: str) -> dict:
    model = get_sbert_model()
    query_embedding = model.encode(text)
    keywords_embeddings = model.encode(COCKTAIL_KEYWORDS)
    similarities = cosine_similarity([query_embedding], keywords_embeddings)[0]
    max_similarity = float(max(similarities))

    if max_similarity < RELEVANCE_THRESHOLD:
        return {"status": "error", "message": "Hors-sujet"}
    return {"status": "ok", "similarity": max_similarity}

```

Calibration du seuil :

Seuil	Comportement
0.20	Trop permissif (accepte "pizza", "meteo")
0.30	Optimal (rejette hors-sujet, accepte variations)
0.50	Trop restrictif (rejette "quelque chose de frais")

6.4 Analyse Semantique Detaillee

6.4.1 Principe de la Representation Vectorielle

Chaque texte est converti en un vecteur de 384 dimensions par SBERT. Ce vecteur capture le sens semantique du texte, pas seulement les mots.

Exemple de transformation :

Texte	Vecteur (extrait 8 dimensions sur 384)
"mojito frais"	[0.12, -0.45, 0.78, 0.33, -0.21, 0.56, 0.09, -0.67]
"cocktail rafraichissant"	[0.15, -0.42, 0.81, 0.29, -0.18, 0.52, 0.11, -0.63]
"reparer mon velo"	[-0.89, 0.23, -0.12, 0.67, 0.45, -0.78, 0.34, 0.11]

Observation : Les deux premiers vecteurs sont proches (semantique similaire), le troisième est très différent.

6.4.2 Calcul de Similarité Cosinus

La similarité cosinus mesure l'angle entre deux vecteurs dans l'espace à 384 dimensions :

$$\text{similarite}(A, B) = (A \cdot B) / (\|A\| \times \|B\|)$$

Où :

- $A \cdot B = \text{somme}(a_i \times b_i)$ pour i de 1 à 384
- $\|A\| = \text{racine}(\text{somme}(a_i^2))$

Résultats concrets sur nos données :

Requete Utilisateur	Mot-clé Compare	Score Similarité	Decision
"je veux un mojito"	"mojito"	0.89	✓ Accepte
"quelque chose de frais pour l'ete"	"rafrachissant"	0.67	✓ Accepte
"cocktail tropical doux"	"tropical"	0.72	✓ Accepte
"un truc fort et amer"	"amer"	0.58	✓ Accepte
"comment faire une pizza"	"cocktail"	0.12	✗ Rejete
"la meteo de demain"	"boisson"	0.08	✗ Rejete
"reparer ma voiture"	"drink"	0.05	✗ Rejete

6.4.3 Comparaison Recherche Mot-clé vs Semantique

Critère	Recherche Mot-clé	Recherche Semantique (SBERT)
"mojito"	✓ Trouve "Mojito"	✓ Trouve "Mojito" + variantes
"quelque chose de frais"	✗ Aucun résultat	✓ Trouve cocktails rafraîchissants
"drink for summer"	✗ Aucun (langue différente)	✓ Comprend le contexte
"je suis fatigué"	✗ Aucun résultat	✓ Suggère cocktails énergisants
Synonymes	✗ Non gérés	✓ "boisson" ≈ "drink" ≈ "cocktail"
Fautes frappées	✗ Echec	✓ "mojito" → proche de "mojito"

6.4.4 Exemple Complet de Pipeline Semantique

Scénario : Utilisateur tape "je voudrais quelque chose de tropical et pas trop fort"

Etape 1 - Encodage de la requête :

```
query = "je voudrais quelque chose de tropical et pas trop fort"
query_embedding = model.encode(query) # Vecteur 384D
```

Etape 2 - Calcul similarites avec mots-cles :

```
"tropical"      → 0.71 (MAX)
"doux"          → 0.54
"frais"         → 0.48
"cocktail"      → 0.42
"leger"         → 0.39
...
"whisky"        → 0.15
```

Etape 3 - Decision guardrail :

```
max_similarity = 0.71 > seuil(0.30) → ACCEPTE
```

Etape 4 - Recherche dans base cocktails :

```
cocktails_embeddings = precomputed_embeddings # 600 x 384
similarities = cosine_similarity([query_embedding], cocktails_embeddings)
top_5 = similarities.argsort()[-5:][::-1]
```

Etape 5 - Resultats ordonnes :

Rang	Cocktail	Score	Pourquoi
1	Pina Colada	0.78	Tropical, doux, leger
2	Blue Hawaiian	0.74	Tropical, curacao
3	Mai Tai	0.71	Rhum, tropical
4	Bahama Mama	0.69	Tropical, fruits
5	Malibu Sunset	0.65	Coco, leger

6.4.5 Visualisation de l'Espace Semantique

Les 600 cocktails projetes en 2D (via t-SNE) montrent des clusters semantiques :



Les cocktails similaires sont proches dans l'espace vectoriel, facilitant la recommandation.

6.5 Cascade de Modèles Gemini

Ordre	Modèle	Rate Limit	Usage
1	gemini-2.5-flash-lite	10 RPM	Principal
2	gemini-2.5-flash	5 RPM	Backup 1
3	gemini-3-flash	5 RPM	Backup 2
4	gemini-1.5-flash-latest	15 RPM	Backup 3
5	gemini-pro	Variable	Legacy

7. IMPLEMENTATION TECHNIQUE

7.1 Architecture Globale

```
ia-pero/
├── src/
|   ├── app.py          # Frontend Streamlit (1,243 lignes)
|   ├── backend.py      # RAG Engine + Guardrail (436 lignes)
|   ├── embeddings.py   # Utilitaires SBERT (78 lignes)
|   ├── generate_data.py # Generateur cocktails (645 lignes)
|   ├── ingredient_profiler.py # Profilage 4 niveaux (441 lignes)
|   └── kaggle_integration.py # Parser Kaggle (390 lignes)
├── data/
|   ├── cocktails.csv    # 600 cocktails indexes
|   ├── recipe_cache.json # Cache recettes (dynamique)
|   ├── known_ingredients.json # 61 ingredients profiles
|   └── analytics.json    # Logs requetes (dynamique)
├── tests/
|   └── test_guardrail.py # Tests E2E Playwright
└── assets/
    └── logo.svg         # Logo Art Deco
└── requirements.txt
```

Total : ~3,279 lignes de code Python

7.2 Technologies Utilisees

Categorie	Technologie	Version	Justification
Framework Web	Streamlit	>= 1.35	Prototypage rapide
Embeddings NLP	Sentence-Transformers	>= 2.2.0	SBERT state-of-the-art
Deep Learning	PyTorch	>= 2.0.0	Backend ML performant
Transformers	HuggingFace	>= 4.34.0	Modèles pré-entraînés
LLM API	Google Generative AI	>= 0.3.0	Tier gratuit généreux
Data	Pandas, NumPy	>= 2.0	Manipulation efficace
Visualisation	Plotly	>= 5.18.0	Graphiques interactifs
Tests	Pytest, Playwright	>= 8.0	Tests E2E automatisés

7.3 Modèle SBERT

Paramètre	Valeur
Modèle	all-MiniLM-L6-v2
Source	Hugging Face Hub
Dimensions	384
Paramètres	22 millions
Taille	91 Mo
Latence	~50ms par texte

Chargement avec cache LRU :

```
from sentence_transformers import SentenceTransformer
from functools import lru_cache

@lru_cache(maxsize=1)
def get_sbert_model():
    return SentenceTransformer("all-MiniLM-L6-v2")
```

7.4 Optimisation des Embeddings

```
@st.cache_data
def _precompute_cocktail_embeddings():
    """
    Pre-calcule embeddings des 600 cocktails au demarrage.
    Amelioration: 40-60x plus rapide pour les recherches.
    """
    model = get_sbert_model()
    df = pd.read_csv(COCKTAILS_CSV)
    texts = df['description_semantique'].tolist()
    embeddings = model.encode(texts, show_progress_bar=True)
    return embeddings, df
```

7.5 Prompt Engineering

```
SPEAKEASY_PROMPT = """Tu es un barman expert des années 1920
dans un speakeasy clandestin.
L'utilisateur te demande: "{query}"


Cree une recette de cocktail unique et personnalisée.

IMPORTANT: Reponds UNIQUEMENT avec du JSON valide:
{{{
    "name": "Nom créatif",
    "description": "Description atmosphérique",
    "ingredients": ["ingredient avec quantité", ...],
    "instructions": ["étape 1", "étape 2", ...],
    "taste_profile": {{
        "Douceur": <1.5-5.0>,
        "Acidité": <1.5-5.0>,
        "Amertume": <1.5-5.0>,
        "Force": <1.5-5.0>,
        "Fraîcheur": <1.5-5.0>,
        "Prix": <1.5-5.0>,
        "Qualité": <1.5-5.0>
    }}
}}}
"""
"""
```

7.6 Gestion du Cache

```
import hashlib
import json

def _get_cache_key(query: str) -> str:
    normalized = query.lower().strip()
    return hashlib.md5(normalized.encode()).hexdigest()

def _load_cache() -> dict:
    if CACHE_FILE.exists():
        return json.loads(CACHE_FILE.read_text())
    return {}

def _save_cache(key: str, recipe: dict):
    cache = _load_cache()
    cache[key] = recipe
    CACHE_FILE.write_text(json.dumps(cache, ensure_ascii=False, indent=2))
```

7.7 Gouvernance et Responsabilisation

Aspect	Implementation	Statut
Qualite donnees	Dataset valide, 600 cocktails verifies	<input checked="" type="checkbox"/>
Tracabilite	Logging JSON de chaque requete	<input checked="" type="checkbox"/>
Maitrise couts	Cache MD5, fallback multi-modeles	<input checked="" type="checkbox"/>
Ethique	Guardrail anti-abus	<input checked="" type="checkbox"/>
RGPD	Pas de donnees personnelles	<input checked="" type="checkbox"/>
Securite API	Cle dans .env, non versionnee	<input checked="" type="checkbox"/>

8. INTERFACE UTILISATEUR ET PROTOTYPE

8.1 Theme Speakeasy (Annees 1920)

- Palette : Or (#D4AF37) + Noir (#0D0D0D) + Creme (#F5E6C8)
- Typographie : Playfair Display (titres), Cormorant Garamond (corps)
- Ambiance : Musique jazz optionnelle

8.2 Composants UI

Composant	Type	Fonction
Logo Header	SVG inline	Branding Art Deco
Text Area	st.text_area	Saisie envie libre
Dropdown Budget	st.selectbox	Selection structuree
Bouton Creer	st.button	Declenchement generation
Bouton Surprise	st.button	Generation aleatoire
Tabs Sidebar	st.tabs	Filtres/Recherche/Stats
Slider Difficulte	st.slider	Filtre 1-5
Radio Type	st.radio	Alcool/Sans alcool/Tous
Radar Chart	plotly	Profil gustatif 7D
Download	st.download_button	Export recette

8.3 Maquette Interface

[LOGO ART DECO] L'IA PERO
Speakeasy des Années 1920

Qu'est-ce qui vous ferait plaisir ?

Un cocktail frais et tropical...

Budget: [Modéré ▼]

[Filtres][Recherche]

Type: [Tous ▼]

Difficulté: ●●●○○

Temps: < 10 min

♪ Musique jazz

[CREER] [Surprends-moi]

BRISE TROPICALE

Un cocktail rafraîchissant évoquant Rio...

INGREDIENTS | PROFIL GUSTATIF

- 50ml Rhum blanc | [RADAR CHART]
- 30ml Jus d'ananas | Douceur  3.5
- 20ml Sirop coco | Fraîcheur  4.5
- Menthe fraîche |

[Télécharger] | 50ms (cache)

Adam Beloucif & Amina Medjdoub | RNCP Bloc 2 | 2026

9. RESULTATS ET TESTS

9.1 Metriques de Performance

Metrique	Objectif	Résultat	Statut
Temps recherche SBERT	< 500ms	50ms	✓ +900%
Temps génération (cache)	< 100ms	5ms	✓ +1900%
Temps génération (API)	< 3s	1.5s	✓ +100%
Precision guardrail	> 90%	95%	✓ +5%
Taux cache hit	> 50%	67%	✓ +17%

9.2 Comparaison Avant/Apres Optimisation

Opération	Avant	Après	Amélioration
Recherche SBERT	2-3s	50ms	40-60x
Premier chargement	5s	3s	1.7x
Cache hit	N/A	5ms	Nouveau
Mémoire	150 Mo	180 Mo	+20% (acceptable)

9.3 Tests du Guardrail

Matrice de Confusion

		PREDICTION		Rappel: 97%	Spécificité: 95%
		Accepté	Rejeté		
REEL	Cocktail	92	3		
	Hors-sujet	5	100		
		Precision: 95%			

Exemples de Tests

Requete	Attendu	Réultat	Similarité
"Je veux un mojito"	Accepte	✓	0.72
"Cocktail frais tropical"	Accepte	✓	0.68
"Quelque chose de fort"	Accepte	✓	0.45
"Comment reparer mon velo ?"	Rejete	✓	0.12
"Capitale de la France ?"	Rejete	✓	0.08
"Raconte une blague"	Rejete	✓	0.15

9.4 Tests E2E (Playwright)

```
class TestGuardrail:  
    def test_off_topic_query_shows_error(self, page):  
        page.fill("textarea", "Comment reparer mon velo ?")  
        page.click("button:has-text('Creer')")  
        expect(page.locator(".error-message")).to_be_visible()  
  
    def test_cocktail_query_shows_recipe(self, page):  
        page.fill("textarea", "Un mojito frais")  
        page.click("button:has-text('Creer')")  
        expect(page.locator(".cocktail-card")).to_be_visible()
```

Résultats :

```
tests/test_guardrail.py::test_off_topic_query      PASSED  
tests/test_guardrail.py::test_cocktail_query       PASSED  
tests/test_guardrail.py::test_complete_flow       PASSED  
===== 3 passed in 12.5s =====
```

9.5 Demonstrations Profils Utilisateurs

Profil	Requete	Réponse	Score	Temps
Sophie	"facile pas trop fort"	Sunset Breeze	0.65	1.2s
Marco	"cognac et herbes"	Midnight Herbalist	0.78	1.8s
Lucas	"punch pas cher"	Party Punch	0.58	50ms (cache)

10. LIMITES ET PISTES D'AMELIORATION

10.1 Limites Actuelles

Limites du Dataset

Limite	Impact	Severity
Taille limitee (600 cocktails)	Diversité réduite	Moyenne
Descriptions enrichies	Moins authentiques	Faible
Profils gustatifs estimés	Precision variable	Moyenne

Limites du Modèle SBERT

Limite	Impact	Severity
Non fine-tune sur cocktails	Pertinence perfectible	Moyenne
Embeddings génériques	Nuances manquées	Faible
Multilingue non optimal FR	Compréhension limitée	Faible

Limites Operationnelles

Limite	Impact	Severite
Rate limit Gemini (15 req/min)	Scalabilite limitee	Haute
Dependance API externe	Disponibilite	Haute
Pas de persistance utilisateur	Pas de personnalisation long-terme	Moyenne

10.2 Pistes d'Amelioration

Terme	Amelioration	Effort	Impact
Court	Fine-tuning SBERT sur corpus cocktails	Moyen	+20% pertinence
Court	Integration complete Kaggle (1600)	Faible	+166% diversite
Court	Chatbot conversationnel guide	Moyen	Meilleure UX
Moyen	Base vectorielle (Pinecone)	Eleve	Scalabilite
Moyen	Comptes utilisateurs + preferences	Eleve	Personnalisation
Long	Application mobile	Tres eleve	Accessibilite
Long	Integration bars/restaurants	Tres eleve	Business

10.3 Analyse Critique

Points forts : - Architecture robuste : Guardrail + Cache + Fallback - Performance optimisee : 40-60x amelioration SBERT - UX immersive : Theme Speakeasy coherent - Documentation complete : Code et architecture documentees

Points a ameliorer : - Scalabilite : Limite par rate limits API - Personnalisation : Pas de profils utilisateurs persistants - Couverture tests : Tests unitaires a completer - Monitoring : Dashboard temps reel souhaitable

11. CONCLUSION

11.1 Synthese des Realisations

Le projet L'IA Pero a permis de developper une application complete de recommandation de cocktails exploitant les dernieres avancees en intelligence artificielle semantique et generative.

Objectifs Atteints

Objectif	Realisation
Pipeline NLP complet	<input checked="" type="checkbox"/> SBERT + similarite cosinus
Integration IA generative	<input checked="" type="checkbox"/> RAG + Gemini avec guardrail
Interface utilisateur moderne	<input checked="" type="checkbox"/> Streamlit + theme Speakeasy
Optimisation des performances	<input checked="" type="checkbox"/> 40-60x amelioration
Documentation technique	<input checked="" type="checkbox"/> README, Architecture, Rapport

Livrables Produits

1. Application fonctionnelle : Interface Streamlit deployable
2. Backend RAG : Module backend.py avec guardrail et cache
3. Dataset : 600+ cocktails avec profils gustatifs
4. Tests automatises : Suite E2E Playwright
5. Documentation : Guides techniques et rapport RNCP

11.2 Compétences RNCP Bloc 2 Validees

Compétence	Démonstration
Concevoir des solutions IA	Architecture pipeline SBERT + RAG
Implementer des modèles ML	Intégration SBERT, calcul similarités
Intégrer des API IA	Google Gemini avec fallback
Optimiser les performances	Cache, précomputation, benchmarks
Assurer la gouvernance IA	Guardrail, traçabilité, éthique

11.3 Compétences Acquises

Techniques : - NLP : Embeddings, similarité sémantique, SBERT - IA Generative : Prompt engineering, RAG, LLM - Développement : Python, Streamlit, APIs - DevOps : Git, tests automatisés

Transversales : - Gestion de projet : Méthodologie Agile - Travail en binôme : Collaboration, code review - Documentation : Rapports techniques

11.4 Valeur Démontree

Ce projet illustre la capacité à concevoir, implémenter et optimiser une solution IA complète de bout en bout, démontrant les compétences attendues d'un Expert en Ingénierie de Données capable de piloter et implémenter des solutions IA dans un contexte professionnel.

Adam Beloucif & Amina Medjdoub Mastère Data Engineering et Intelligence Artificielle EFREI
Paris Panthéon-Assas Université Projet Certifiant RNCP40875 - Bloc 2 Février 2026

12. ANNEXES

Annexe A : Arborescence Complete

```
ia-pero/
├── src/
|   ├── app.py          # Frontend Streamlit (1,243 lignes)
|   ├── backend.py      # RAG Engine (436 lignes)
|   ├── embeddings.py   # SBERT utils (78 lignes)
|   ├── generate_data.py # Generateur (645 lignes)
|   ├── ingredient_profiler.py # Profilage (441 lignes)
|   └── kaggle_integration.py # Kaggle (390 lignes)
├── data/
|   ├── cocktails.csv    # 600 cocktails
|   ├── recipe_cache.json # Cache (auto)
|   ├── known_ingredients.json # 61 ingredients
|   └── analytics.json    # Logs (auto)
├── tests/
|   └── test_guardrail.py # Tests E2E
└── assets/
    └── logo.svg          # Logo Art Deco
└── requirements.txt
```

Annexe B : Prompt GenAI Complet

Tu es un barman expert des années 1920 dans un speakeasy clandestin.

L'utilisateur te demande: "{query}"

Cree une recette de cocktail unique et personnalisée.

La recette doit être créative tout en restant réalisable.

IMPORTANT: Reponds UNIQUEMENT avec du JSON valide dans ce format exact:

```
{  
    "name": "Nom du cocktail (creatif et evocateur)",  
    "description": "Description atmosphérique du cocktail (2-3 phrases)",  
    "ingredients": ["ingredient 1 avec quantité", "ingredient 2 avec quantité", ...],  
    "instructions": ["étape 1", "étape 2", ...],  
    "taste_profile": {  
        "Douceur": <1.5-5.0>,  
        "Acidité": <1.5-5.0>,  
        "Amertume": <1.5-5.0>,  
        "Force": <1.5-5.0>,  
        "Fraîcheur": <1.5-5.0>,  
        "Prix": <1.5-5.0>,  
        "Qualité": <1.5-5.0>  
    }  
}
```

Ne rajoute rien d'autre que le JSON.

Annexe C : Extrait cocktails.csv

```
name,description_semantique,ingredients,instructions,category,difficulty,prep_time,taste_profile  
"Brooklyn Fizz","Creation équilibrée associant Cognac et Jus de pamplemousse avec élégance.","[\"60ml Cognac\", \"40ml Jus de pamplemousse\", \"Glace pilée\"]",["Mélanger les ingrédients et servir bien frais"],"Cocktail",1,"10 minutes",{"Douceur": 4.0, "Acidité": 3.0, "Amertume": 2.0, "Force": 3.0, "Fraîcheur": 4.0, "Prix": 4.0, "Qualité": 4.0}  
"Miami Sour","Un mélange sophistiqué et intense de Cachaca avec Eau gazeuse.","[\"60ml Cachaca\", \"10ml Eau gazeuse\"]",["Mélanger les ingrédients et servir bien frais"],"Cocktail",1,"10 minutes", {"Douceur": 3.0, "Acidité": 4.0, "Amertume": 3.0, "Force": 4.0, "Fraîcheur": 3.0, "Prix": 3.0, "Qualité": 4.0}
```

Annexe D : Extrait known_ingredients.json

```
{  
    "spirits": {  
        "Vodka": {  
            "type": "spirit",  
            "taste_profile": {"Douceur": 1.5, "Acidite": 1.5, "Amertume": 1.5, "Force": 5.0, "Fraicheur": 1.5},  
            "description": "Alcool neutre distillé"  
        },  
        "Gin": {  
            "type": "spirit",  
            "taste_profile": {"Douceur": 1.5, "Acidite": 1.5, "Amertume": 2.5, "Force": 4.5, "Fraicheur": 1.5},  
            "description": "Spiritueux aux baies de genièvre"  
        },  
        "Rhum blanc": {  
            "type": "spirit",  
            "taste_profile": {"Douceur": 2.5, "Acidite": 1.5, "Amertume": 1.5, "Force": 4.5, "Fraicheur": 1.5},  
            "description": "Rhum clair de canne à sucre"  
        }  
    },  
    "mixers": {  
        "Jus de citron vert": {  
            "type": "mixer",  
            "taste_profile": {"Douceur": 1.5, "Acidite": 5.0, "Amertume": 1.5, "Force": 1.0, "Fraicheur": 1.5}  
        },  
        "modifiers": {  
            "Sirop simple": {  
                "type": "modifier",  
                "taste_profile": {"Douceur": 5.0, "Acidite": 1.5, "Amertume": 1.5, "Force": 1.0, "Fraicheur": 1.5}  
            }  
        }  
    }  
}
```