



Mimir - Itération 1

PINCHON - PERROT - HIRTZ - MIJATOVIC



Objectifs

L'objectif de cet itération est de réaliser ces points :

- Application Web fonctionnelle
- Réalisation de toutes les pages front-end
- Mise en place de la base de donnée et liée à l'application Web
- Vérifier la preuve de concept de l'intelligence artificielle



Réalisation de l'équipe

- PoC IA
- Extracteur de PDF vers texte
- Architecture
- Base de données
- Application Morte



Poc IA

Test de différents modèles de LLM (Large Language Model) :

- BERT : Modèle intéressant mais besoin du contexte et d'une question pour avoir une réponse (pas de génération de questions)
- GPT-2 : Modèle le plus récent gratuit d'Open AI. Résultats incohérents et aberrants par rapport au résultat attendu.
- T5 : Possibilité de générer des questions d'une manière intéressante mais les résultats n'étaient pas cohérents non plus



Poc IA

- Mistral : Modèle rapide et fonctionnel mais “fine-tuned” en anglais, donc les résultats en français ont quelques erreurs
- Vigogne : Sous-modèle de Vicuna “fine-tuned” en français. Les résultats sont assez satisfaisants et ne fait pas vraiment d’erreurs de français, excepté si une erreur est dans le contexte et qu’il pose des questions/réponses dessus.



Poc IA

Modèle retenu : Vigogne puisque marche en français, fonctionnel et rapide.

Le retour texte de l'extracteur PDF est envoyé au modèle à l'aide de l'API de Ollama par une requête POST.

Le résultat de ce traitement est ensuite récupéré en json pour pouvoir l'inclure aux decks.

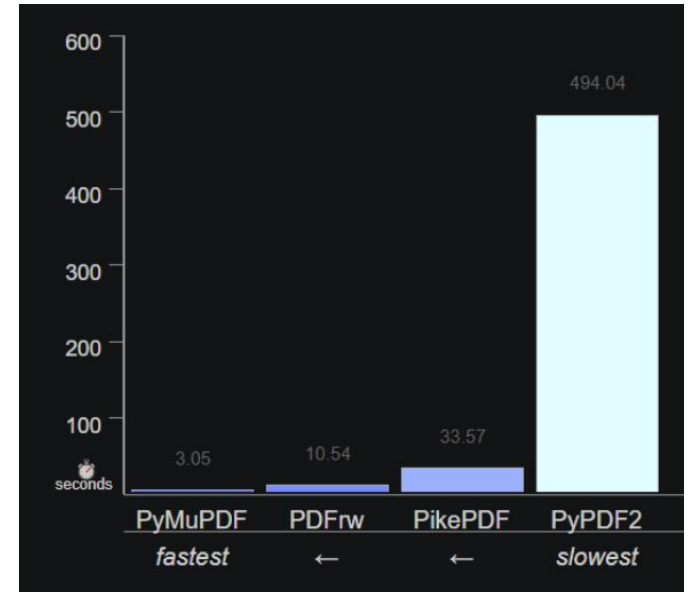
Quelques soucis au niveau génératif du modèle.

Extracteur de PDF vers texte

Utilisation du langage Python pour ses librairies dont :

- PyMuPDF
- pikepdf
- PyPDF2
- pdfrw
- pdfplumber

Choix de PyMuPDF pour sa rapidité et sa simplicité d'utilisation





Extracteur de PDF vers texte

Implémentation de PyMuPDF

Les résultats sont concluants mais quelques bémols :

- Mauvaise reconnaissance de certains caractères ('e au lieu de é)
- Informations inutiles (en-tête, pied de page) pour l'IA



Extracteur de PDF vers texte

Mise en place d'un serveur Web

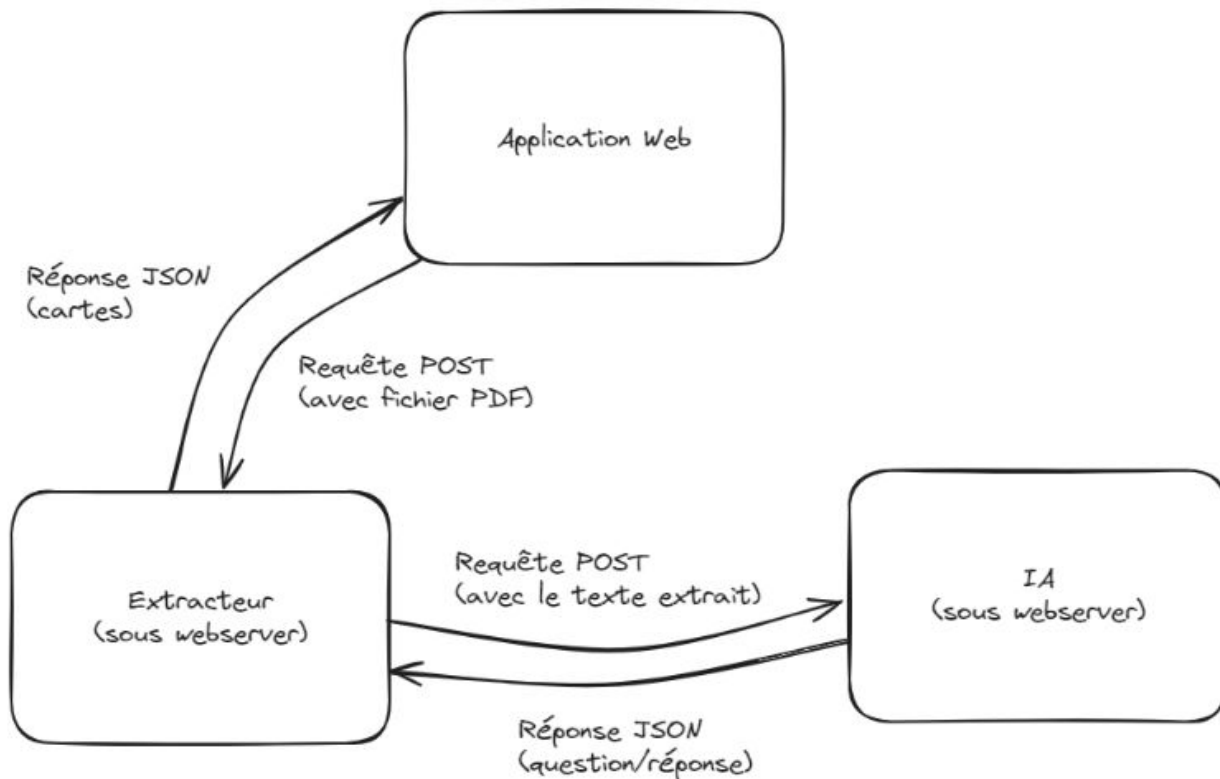
Sous Flask :

- Serveur web léger
- Facile et rapide à mettre en place

Pouvoir effectuer des requêtes depuis l'application Web :

- Extraire le texte
- Générer les questions via Ollama
- Retourner un JSON qui convertit le texte en cartes pour notre deck

Extracteur de PDF vers texte





Extracteur de PDF vers texte

Upload new File

Choisir un fichier

01_optimi...06_05.pdf

Upload



```
{ "question": "Quels sont les algorithmes possibles pour la descente de gradient ?", "réponse": "Les algorithmes possibles pour la descente de gradient comprennent la descente de gradient avec pas d'apprentissage et la descente de gradient avec pas d'apprentissage." }
```



Architecture

Introduction

Le projet a été construit sur Turbo Repo, alimenté par Vercel, pour gérer efficacement les projets monorepo.

Site Web

Next.js a été utilisé pour la création du site web, un framework JavaScript basé sur React, conçu pour la production et offrant une variété de fonctionnalités utiles.



Architecture

Support de librairies externes

Des efforts ont été déployés pour faire fonctionner Tailwind CSS avec Next.js, permettant ainsi de créer des interfaces utilisateur réactives et attrayantes.

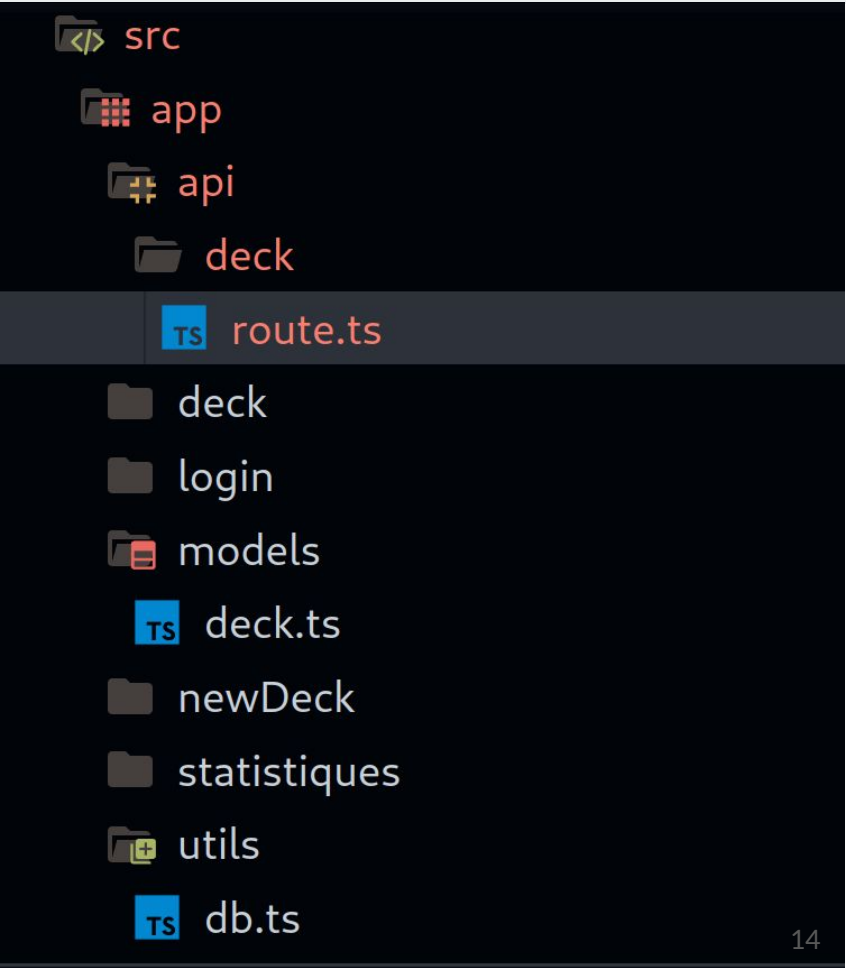


Tailwind CSS

Base de données

Base de données

La base de données est structurée autour de trois fichiers principaux : `db.ts` pour la connexion à MongoDB via Mongoose, `deck.ts` pour définir le modèle de données, et `/api/deck/route.ts` pour définir les points de terminaison API. Ces points de terminaison prennent actuellement en charge les requêtes POST et GET.



```
src
├── app
├── api
│   ├── deck
│   ├── login
│   ├── models
│   ├── deck.ts
│   ├── newDeck
│   ├── statistiques
│   ├── utils
│   └── db.ts
```

Base de données



Visualisation des données

Une fois que les données ont été ajoutées, nous pouvons instantanément les consulter grâce l'interface de mongo atlas (cloud.mongodb.com)

Données de test du système

Postman a été utilisé pour tester l'unique endpoint de l'api créé pour le moment.
Cela permet de s'assurer que notre système est bien fonctionnel.





Application Morte (Web UI)

Réalisé (itération 1) :

Prise en main de l'environnement next + Implémentation de l'application morte + ajout de librairie de composants JS

Retard : Finir pages (Profil + Stats)

A venir (itération 2) :

Implémentation des classes métier + gestion des cartes



Planning pour l'itération 2

Système d'authentification (Jules)

Système de partage des deck via URL (Alexandre)

Gestion des contacts entre utilisateurs (Yann)

Classes métier + Création API avec la gestion des cartes (Théo)



Annexe

Interface postman

Vue du dashboard mongo atlas



GET

http://localhost:3000/api/deck

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

Beautify

```
1 {
2   "id": 2,
3   "title": "Test Title 2",
4   "tags": ["tag1", "tag2"],
5   "isPublic": true,
6   "isEducational": false,
7   "votes": {
8     "up": 10,
9     "down": 2
10  },
11   "deadline": "2024-01-31T00:00:00Z",
12   "owner_id": 1,
13   "cards": [
14     {
15       "id": 1,
16       "question": "Question 1",
17       "answer": "Answer 1",
18       "proficiency": 1
```

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 113 ms Size: 1.08 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "votes": {
4       "up": 10,
5       "down": 2
6     },
7     "_id": "65a0fb1e1563eb5ff15c1ea8",
8     "id": 1,
9     "title": "Test Title",
10    "tags": [
11      "tag1",
12      "tag2"
13    ],
14    "isPublic": true,
15    "isEducational": false,
16    "deadline": "2024-01-31T00:00:00.000Z",
17    "owner_id": 1,
18    "cards": [
```

DATABASES: 1 COLLECTIONS: 2

 VISUALIZE YOUR DATA

 REFRESH

+ Create Database

 Search Namespaces

test

decks

topics

test.decks

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 698B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 20KB

Find

Indexes

Schema Anti-Patterns 

Aggregation

Search Indexes

INSERT DOCUMENT

Filter 

Type a query: { field: 'value' }

Reset

Apply

Options 

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('65a0fb1e1563eb5ff15c1ea8')
id: 1
title: "Test Title"
tags: Array (2)
isPublic: true
isEducational: false
votes: Object
  deadline: 2024-01-31T00:00:00.000+00:00
  owner_id: 1
cards: Array (1)
  createdAt: 2024-01-12T08:41:02.124+00:00
  updatedAt: 2024-01-12T08:41:02.124+00:00
__v: 0
```

```
_id: ObjectId('65a109d2330c5c29e12ef95e')
id: 2
title: "Test Title 2"
tags: Array (2)
isPublic: true
isEducational: false
votes: Object
  deadline: 2024-01-31T00:00:00.000+00:00
  owner_id: 1
```

