# Atelier: coder avec une IA générative

Utilisation de Large Language Models pour développer un logiciel en respectant les standards de l'industrie

Partie 2

Nicolas Debeissat
nicolas.debeissat@mail-formateur.net

# Plan du module

### Installation de l’extension continue.dev

Dans VSCode, avec ollama en local

### Utiliser la configuration de base

Création d’une stack React + FastAPI + Postgresql

### Configuration avancée et codebase

Comment adapter ses réponses

### RAG avec Langchain

API de génération de requêtes ElasticSearch

### Model Context Protocol (mode Agent)

Création d’outils et connexion à son agent

### Evaluation

Quizz et rendu du projet effectué en cours

# Initialiser le projet React

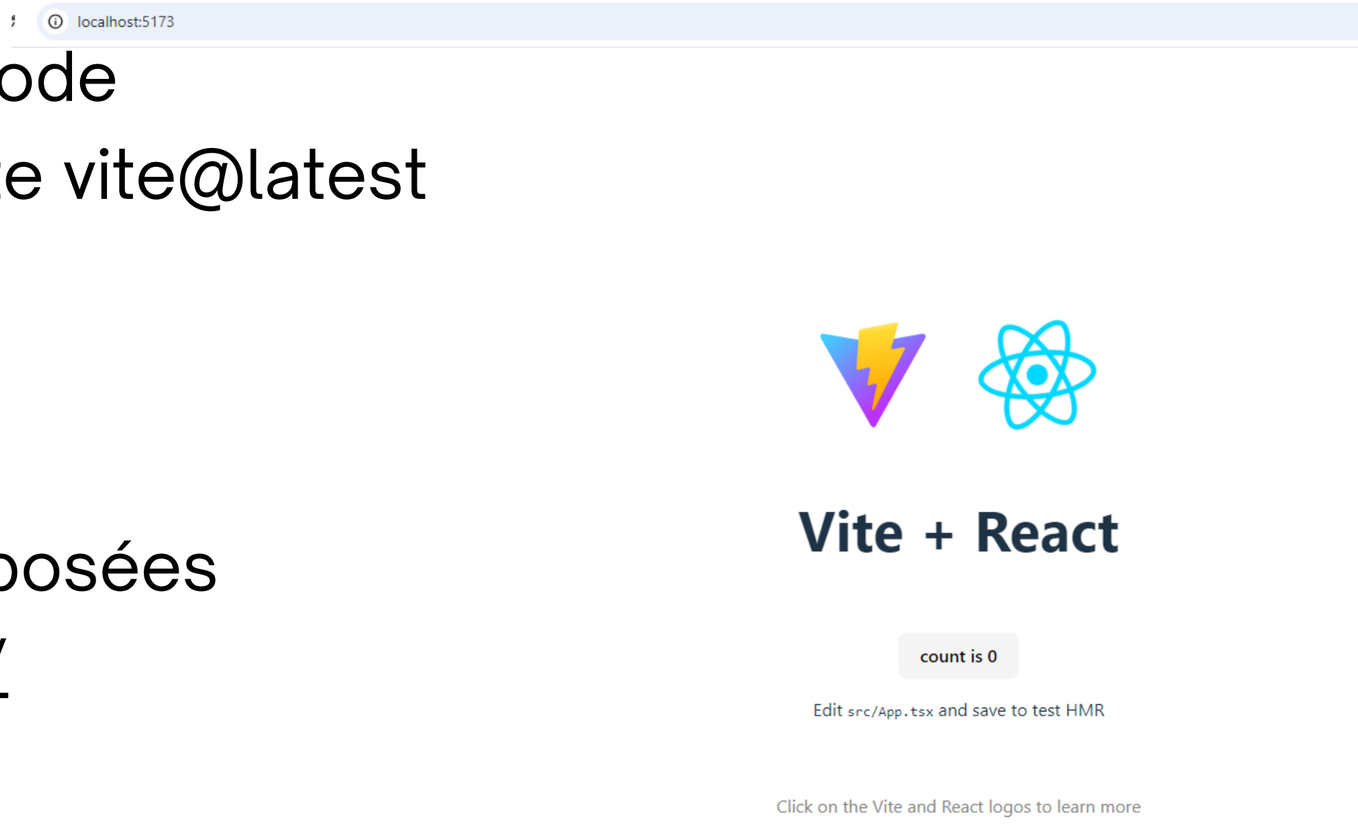
- Créer un répertoire projet_vscode
- dans ce répertoire : npm create vite@latest

√ Project name: …

√ Select a framework: » React

√ Select a variant: » TypeScript

- exécuter les commandes proposées
- aller sur http://localhost:5173/



# Editer le projet React

- Installer l’extension Vite
- Ouvrir le répertoire vite-project dans une nouvelle fenêtre VSCode
- Sélectionner la ligne avec le count, taper sur Ctrl+L puis écrivez le prompt suivant :

*call an API to retrieve the value of count*

- Sélectionner tout le fichier, Ctrl+L, puis

*when clicking the button i want to call a endpoint which increments count*

- Copier le code généré et regarder le résultat dans l’inspecteur Chrome

# Initialiser le projet back FastApi

- Installer les extensions Python, Python Debugger et Pylance
- créer un répertoire back-fastapi
- Dans ce répertoire, dans un terminal git bash :

poetry init
poetry config virtualenvs.in-project true --local
poetry add fastapi
poetry add uvicorn

- Ouvrir ce répertoire back-fastapi dans une nouvelle fenêtre VSCode pour que le venv soit pris en compte automatiquement
- Créer un fichier README.md
- Créer un répertoire app avec deux fichiers :
  - main.py
  - __init__.py

# Initialiser le projet FastApi

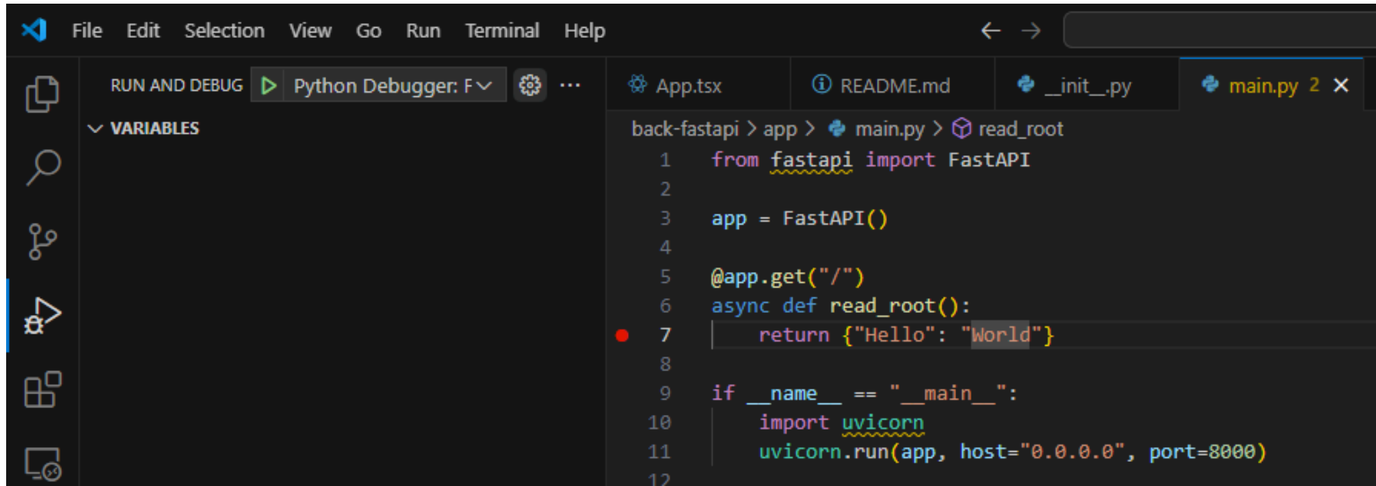
- Dans le fichier main.py, taper Ctrl+I, puis dans le prompt :

*Write fastapi initial code*

- Accepter la modification
- A la fin du fichier main.py, retaper Ctrl+I, sélectionner dans History, le prompt précédent
- Accepter la modification
- Exécuter le mode debug

fastapi

- Ouvrir http://localhost:8000



# Ecrire les Dockerfile

- Dans le projet back_fastapi :
  - créer un fichier Dockerfile
  - “write a Dockerfile for FastAPI with poetry”

- Dans le projet vite-project :
  - créer un fichier Dockerfile
  - “write a dockerfile for a vite project”

# Dockerfile back

```
FROM python:3.12-slim

WORKDIR /app

COPY pyproject.toml poetry.lock ./
RUN pip install poetry
RUN poetry config virtualenvs.create false
RUN poetry install --no-root

COPY . .
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

# Dockerfile front

```
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

EXPOSE 5173

CMD ["npm", "run", "dev", "--", "--host"]
```

# Ecrire le docker-compose

- Ouvrir le projet projet_vscode avec VSCode
- Créer un fichier docker-compose.yml, et demander à l'IA :

*write a docker compose file with a postgresql docker, a fastapi docker in folder back-fastapi and a vite for react docker in folder vite-project*

- Demander à ajouter un network en bridge
- Lancer le service db

# Créer un modèle de données

- Créer un répertoire models sous app/
- Créer un fichier __init__.py
- Créer un fichier count_table.py
- Dans ce fichier, Ctrl+I, et demander à écrire un modèle sqlachemy pour une table avec un champ count_number
- Demander à créer la base et la table, la vider et créer une ligne avec un count à 0

# Ecrire les endpoints

- Dans le fichier app/main.py, demander un endpoint qui permet de récupérer le count avec en contexte, le fichier count_table.py
- Puis demander un endpoint qui va incrémenter ce count

# Limitations naturelles de l’IA

https://blog.cubed.run/100-accurate-ai-claimed-by-acurai-openai-and-anthropic-confirm-acurais-discoveries-98fce1ddeb5b

https://youtu.be/K4Wg6QzPfyI

“LLMs self-organize around Noun Phrases; and that the behavior of LLMs can be controlled through Noun Phrase manipulation.”

Attention aux pronoms, ils peuvent mener à des ambigüités

# Qualité du code

- Attention à ne pas surcommenter, surtout si c’est généré par l’IA

```
const { connect } = require('lancedb'); // Importer le module LanceDB

(async () => {
  try {
    // Chemin vers votre base de données LanceDB
    const dbPath = './path/to/your/lancedb'; // Remplacez par le chemin réel de votre base LanceDB

    // Connexion à la base de données
    const db = await connect(dbPath);

    // Nom de la table à parcourir
    const tableName = 'my_table'; // Remplacez par le nom de votre table

    // Vérifier si la table existe
    const table = await db.table(tableName);
    if (!table) {
      console.error(`La table "${tableName}" n'existe pas dans la base de données.`);
      return;
    }
```

```java
public static class Content {
    private String mimeType;
    private String mimeTypeName;
    private long sizeInBytes;
    private String encoding;

    // Getters and setters
    public String getMimeType() {
        return mimeType;
    }

    public void setMimeType(String mimeType) {
        this.mimeType = mimeType;
    }

    public String getMimeTypeName() {
        return mimeTypeName;
    }

    public void setMimeTypeName(String mimeTypeName) {
        this.mimeTypeName = mimeTypeName;
    }

    public long getSizeInBytes() {
        return sizeInBytes;
    }
}
```

# Qualité du code

✅ **Filebeat config to output only the log message as plain text**

yaml — Copier — Modifier

```yaml
filebeat.inputs:
  - type: filestream
    paths:
      - /var/lib/docker/containers/*/*.log
    parsers:
      - docker: ~

processors:
  - decode_json_fields:
      fields: ["message"]
      target: ""
      overwrite_keys: true
  - drop_fields:
      fields: ["@timestamp", "host", "input", "agent", "ecs", "log", "container", "docker
  - script:
      lang: javascript
      source: >
        function process(event) {
          event.Put("message", event.Get("message"));
        }

output.file:
  path: "/var/log/custom-nginx-logs"
  filename: "nginx.log"
  codec.format:
    string: '%{[message]}'
```

- Souvent du code superflu, avec la répétition des valeurs par défaut ou sans les outils de génération de code type lombok
- Attention aux versions des outils que vous utilisez, les documentations indexées ont généralement plus d’un an.

# Qualité du code

- Sur-qualité : gérer tous les caractères selon toutes les plate-formes est overkill, il est préférable de tout filtrer dès le début

```python
# Détecter l'OS
if sys.platform.startswith("win"):  # Windows
    forbidden_chars = r'[\\/:*?"<>|]'
    reserved_names = {
        "CON", "PRN", "AUX", "NUL",
        "COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "COM8", "COM9",
        "LPT1", "LPT2", "LPT3", "LPT4", "LPT5", "LPT6", "LPT7", "LPT8", "LPT9"
    }
elif sys.platform.startswith("darwin"):  # macOS
    forbidden_chars = r'/'  # `/` est interdit
    reserved_names = set()  # macOS n'a pas de noms réservés stricts
else:  # Linux (ou autres UNIX)
    forbidden_chars = r'/'  # `/` est interdit
    reserved_names = set()  # Pas de noms réservés sur Linux

# Remplace les caractères interdits
filename = re.sub(forbidden_chars, replacement, filename)

# Supprime les espaces et points en fin de fichier (Windows)
filename = filename.rstrip(" .")

# Vérifie si le nom est réservé (Windows uniquement)
if filename.upper() in reserved_names:
    filename += replacement  # Ajoute un caractère pour éviter les conflits

# Tronquer si nécessaire
filename = filename[:max_length] if len(f name) > max_length else filename
```

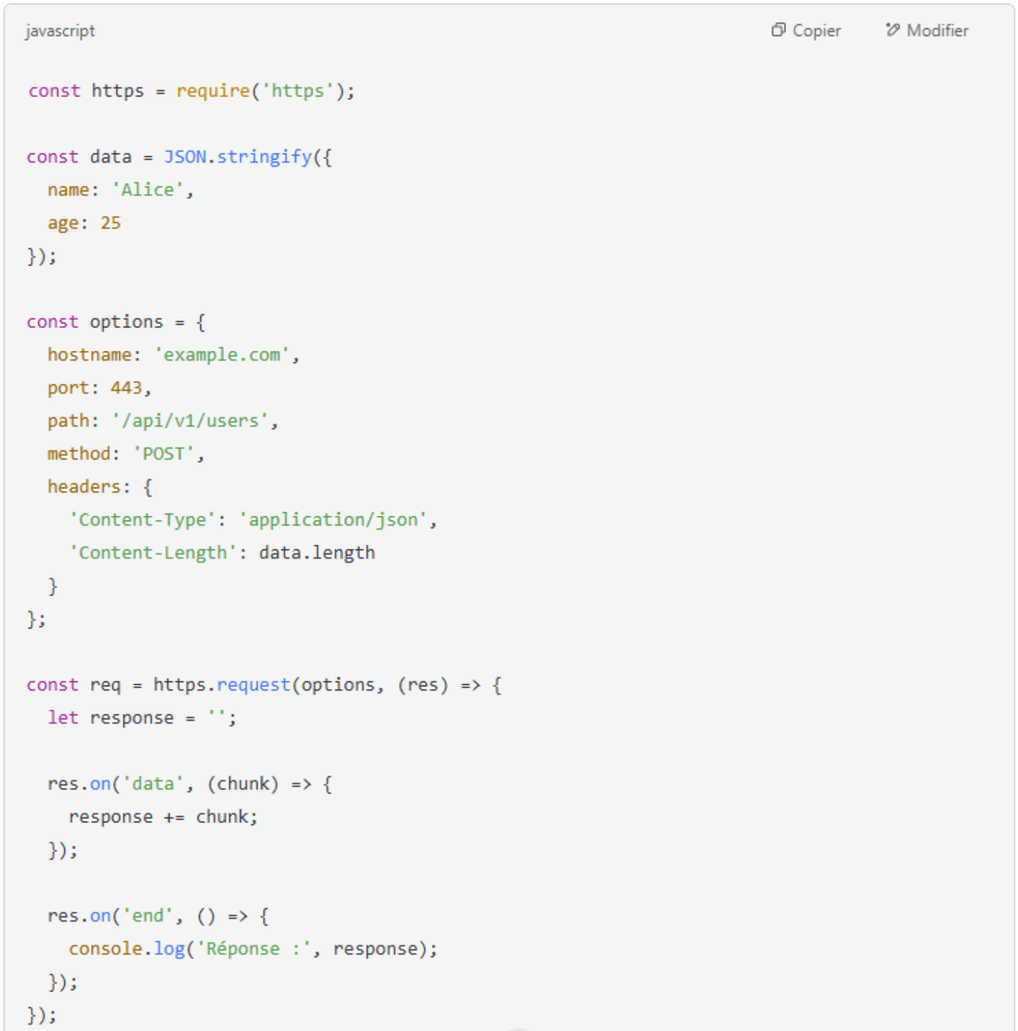
# Duplication du code

- L’IA aura du mal à détecter que du code peut être réutilisé
- Si chaque développeur génère sa fonctionnalité indépendamment, risque d’augmentation importante de la quantité du code du projet
- Certaines fonctionnalités sont mutualisable, en rendant le code plus générique, moins spécialisé, l’IA aura du mal à s’en rendre compte

=> L’IA ne peut pas communiquer avec vos collègues à votre place !

# Erreurs courantes

- Le sens des mots-clés techniques peut être ambigü et est parfois mal compris, ou sur-interprété par les LLMs qui génèrent alors du code faux



# Erreurs courantes

- Le sens des mots-clés techniques peut être ambigü et est parfois mal compris, ou sur-interprété par les LLMs qui génèrent alors du code faux
- ‘Content-Length’ : data.length est faux, ce qui est attendu est la taille en octet, non pas le nombre de caractères

javascript Copier Modifier

```javascript
const https = require('https');

const data = JSON.stringify({
  name: 'Alice',
  age: 25
});

const options = {
  hostname: 'example.com',
  port: 443,
  path: '/api/v1/users',
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Content-Length': data.length
  }
};

const req = https.request(options, (res) => {
  let response = '';

  res.on('data', (chunk) => {
    response += chunk;
  });

  res.on('end', () => {
    console.log('Réponse :', response);
  });
});
```

# Erreurs courantes

- Ils génèrent une bonne base mais vous devrez être expert à sa place

Pour envoyer une requête de recherche Elasticsearch qui compte le nombre d'utilisateurs basés à Montpellier dans un index nommé **users**, vous pouvez utiliser la bibliothèque **elasticsearch** en Python. Voici un exemple de code pour réaliser cela :

Copier

```python
from elasticsearch import Elasticsearch

# Initialiser le client Elasticsearch
es = Elasticsearch([{'host': 'localhost', 'port': 9200}])

# Définir la requête de recherche
query = {
    "query": {
        "match": {
            "location": "Montpellier"
        }
    }
}

# Envoyer la requête de recherche
response = es.search(index="users", body=query)

# Extraire le nombre total d'utilisateurs
total_users = response['hits']['total']['value']

print(f"Nombre total d'utilisateurs basés à Montpellier : {total_users}")
```

# Erreurs courantes

- Ils génèrent une bonne base mais vous devrez être expert à sa place
- "track_total_hits": true doit être spécifié dans la requête pour avoir le nombre exact, si plus de 10000 hits, sinon vous aurez :

{"total":{"value":10000,"relation":"gte"} }

- match est une requête full text, les termes approchant vont remonter aussi

Pour envoyer une requête de recherche Elasticsearch qui compte le nombre d'utilisateurs basés à Montpellier dans un index nommé **users**, vous pouvez utiliser la bibliothèque **elasticsearch** en Python. Voici un exemple de code pour réaliser cela :

Copier

```python
from elasticsearch import Elasticsearch

# Initialiser le client Elasticsearch
es = Elasticsearch([{'host': 'localhost', 'port': 9200}])

# Définir la requête de recherche
query = {
    "query": {
        "match": {
            "location": "Montpellier"
        }
    }
}

# Envoyer la requête de recherche
response = es.search(index="users", body=query)

# Extraire le nombre total d'utilisateurs
total_users = response['hits']['total']['value']

print(f"Nombre total d'utilisateurs basés à Montpellier : {total_users}")
```