Manual técnico

Manual Técnico - Evaluación de Candidatos con FastAPI y Vue.js

1. Introducción

Este sistema permite evaluar candidatos a partir de sus **CVs en formato PDF o DOCX**, determinando su idoneidad para un puesto de trabajo en base a una descripción de la vacante. La API está desarrollada en **FastAPI**, utilizando **PyMuPDF** y **python-docx** para extraer texto, y un modelo de **Ollama AI** para realizar la evaluación. La UI está desarrollada en **Vue.js**.

---

2. Tecnologías Utilizadas

Backend (FastAPI)

- **FastAPI**: Framework para la creación de la API REST.
- **PyMuPDF (fitz)**: Extracción de texto de archivos PDF.
- **python-docx**: Extracción de texto de archivos DOCX.
- **Ollama**: Modelo de IA para evaluar la información de los candidatos.

Frontend (Vue.js)

- **Vue.js**: Framework para la UI.
- **AXIOS**: Para consumir los endpoints de FastAPI.

---

3. Instalación y Configuración

3.1 Requisitos Previos

- Python 3.11+
- Node.js 18+
- FastAPI
- Ollama instalado localmente

3.2 Instalación del Backend

```
# Crear un entorno virtual
python -m venv venv
source venv/bin/activate  # Linux/macOS
venv\Scripts\activate     # Windows

# Instalar dependencias
pip install fastapi uvicorn python-docx pymupdf ollama
```

3.3 Instalación del Frontend (Vue.js)

```
# Moverse al directorio del frontend
cd frontend

# Instalar dependencias
npm install
```

---

4. Estructura del Proyecto

```
project/
├── app/
│   ├── __init__.py       # Inicialización del módulo
│   ├── main.py           # Punto de entrada de FastAPI
│   ├── models.py         # Modelos de datos
│   ├── schemas.py        # Definición de esquemas con Pydantic
│   ├── services.py       # Lógica de procesamiento de archivos y evaluación
│   ├── utils.py          # Funciones auxiliares
├── frontend/             # Aplicación Vue.js
├── requirements.txt      # Dependencias de Python
├── package.json          # Dependencias de Vue.js
```

---

5. Desarrollo del Backend (FastAPI)

5.1 ** `main.py` ** - Configuración de la API

```python
from fastapi import FastAPI, UploadFile, File, Body
from fastapi.middleware.cors import CORSMiddleware
from app.services import process_cv
from app.models import extract_text_from_pdf, extract_text_from_docx
from app.schemas import FeedbackRequest

app = FastAPI()

# Configuración de CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.post("/procesar-cv/")
async def procesar_cv(
    file: UploadFile = File(...),
    job_description: str = Body(...),
):

    return await process_cv(file, job_description)

@app.post("/feedback/")

async def recibir_feedback(feedback: FeedbackRequest):
    return {"mensaje": f"Feedback recibido para {feedback.candidato}:
{feedback.comentario}"}
```

5.2 ** `services.py` ** - Procesamiento y Evaluación de CVs

```python
import ollama

import os

from app.models import extract_text_from_pdf, extract_text_from_docx

from app.database import SessionLocal
```

```python
from app.models import CV



async def save_cv_to_db(texto: str, job_description: str):
    db = SessionLocal()
    nuevo_cv = CV(cv=texto, job_desc=job_description)
    db.add(nuevo_cv)
    db.commit()
    db.refresh(nuevo_cv)
    db.close()
    return nuevo_cv

async def process_cv(file, job_description):

    """Procesa el CV, extrayendo texto y enviando la solicitud a Ollama."""

    file_ext = file.filename.split(".")[-1]
    # Guardamos el archivo temporalmente
    file_path = f"./temp.{file_ext}"
    with open(file_path, "wb") as f:
        f.write(await file.read())

    # Extraemos el texto del archivo

    text = ""
    if file_ext == "pdf":
        text = extract_text_from_pdf(file_path)
    elif file_ext == "docx":
        text = extract_text_from_docx(file_path)
    else:
        os.remove(file_path)  # Limpieza en caso de formato no soportado
        return {"error": "Formato no soportado"}


    # Creamos el prompt para Ollama

    prompt = f"""

    Analiza el CV del candidato y evalúa su adecuación para el puesto basado
en la descripción del trabajo:
    Descripción del trabajo y área requerida:
    {job_description}

    CV del candidato:
    {text}
```

Devuelve solo lo siguiente en español:

- Calificación del candidato de 1 a 10, de acuerdo con las siguientes directrices:

  - **1-3**: El candidato no cumple con los requisitos básicos del puesto. No tiene experiencia clave o habilidades fundamentales necesarias para el puesto.

  - **4-5**: El candidato tiene algunas habilidades relevantes, pero le faltan conocimientos o experiencia en áreas esenciales del puesto.

  - **6-7**: El candidato tiene experiencia en algunas áreas clave, pero le falta habilidad o experiencia en otros aspectos cruciales del puesto.

  - **8-9**: El candidato cumple con la mayoría de los requisitos del puesto, aunque podría necesitar capacitación adicional en áreas específicas.

  - **10**: El candidato cumple completamente con todos los requisitos del puesto y tiene la experiencia adecuada para desempeñarse con éxito.

- **Si el candidato no es adecuado para el puesto y tiene calificación baja**:

  - Menciona **2 áreas clave** que el candidato necesita mejorar para cumplir con los requisitos del puesto. Estas áreas deben estar directamente relacionadas con las habilidades y experiencias clave necesarias para el puesto.

- **Si el candidato es adecuado para el puesto y tiene una calificación decente**:

  - Proporciona **3 razones claras y específicas** por las que el candidato es adecuado para el puesto, basadas en los requisitos detallados en la descripción del trabajo.

Nota: Sé **estricto** en la evaluación. Asegúrate de que la evaluación se base principalmente en la relevancia de las habilidades específicas del puesto. Si el candidato no cumple con los requisitos esenciales, **no

```
menciones las razones de adecuación** y otorga una calificación baja.
    """
    # Interacción con el modelo Ollama
    response = ollama.chat(model="llama3.2:3b", messages=[{"role": "user",
"content": prompt}])
    evaluacion = response["message"]["content"]

    # Guardar los datos en la base de datos
    await save_cv_to_db(text, job_description)
    os.remove(file_path)
    return {"evaluacion": evaluacion}
```

5.3** `models.py` ** - Extraccion de Texto

```
import fitz
import docx
def extract_text_from_pdf(file_path):
    """Extrae el texto de un archivo PDF."""
    doc = fitz.open(file_path)
    text = "\n".join([page.get_text() for page in doc])
    return text

def extract_text_from_docx(file_path):
    """Extrae el texto de un archivo DOCX."""
    doc = docx.Document(file_path)
    text = "\n".join([para.text for para in doc.paragraphs])
    return text
```

5.4** `utils.py` ** - Utilidades

```
import os
def remove_temp_file(file_path):
    """Elimina el archivo temporal después de procesarlo."""
    if os.path.exists(file_path):
        os.remove(file_path)
```

---

database.py`** - Manejo de Base de Datos**

```
from sqlalchemy import create_engine

from sqlalchemy.ext.declarative import declarative_base

from sqlalchemy.orm import sessionmaker

import os



DATABASE_URL = os.getenv("DATABASE_URL",
"postgresql+psycopg2://postgres:1106@localhost:5432/tech-mahindra")



engine = create_engine(DATABASE_URL)



SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)



Base = declarative_base()
```

---

6. Integración con el Frontend (Vue.js)

El frontend en **Vue.js** permite subir archivos y visualizar la evaluación.

6.1 Código para consumir la API en Vue.js

```
async function enviarArchivo(archivo, descripcion) {
    const formData = new FormData();
    formData.append("file", archivo);
    formData.append("job_description", descripcion);

    const response = await fetch("http://localhost:8000/procesar-cv/", {
        method: "POST",
        body: formData
    });
```

```
    return response.json();
}
```

---

7. Ejemplo de Uso

Solicitud (POST a ** `/procesar-cv/` **)

```
curl -X 'POST' 'http://localhost:8000/procesar-cv/' \
  -F 'file=@cv_ejemplo.pdf' \
  -F 'job_description=Desarrollador Backend con Python'
```

Respuesta Esperada

```
{
  "evaluacion": "El candidato tiene experiencia en Python, Django y
PostgreSQL. Puntaje: 8/10."
}
```

---

8. Despliegue y Producción

8.1 Ejecutar Backend

```
uvicorn app.main:app --reload
```

8.2 Ejecutar Frontend

```
cd frontend
npm run dev
```

---

9. Conclusiones

Este sistema permite evaluar automáticamente candidatos a través de IA y FastAPI, optimizando el proceso de selección de talento. La integración con **Vue.js** facilita su uso

mediante una UI moderna y responsiva.