# **APB VISION – Fase 2: Construção e Implementação do Protótipo**

**Nome:** Arthur Paulo de Carvalho  
 **Instituição:** SENAI  
 **Local:** Uberlândia  
 **Data:** 09/07/2025

## **🔷 Objetivo Geral da Fase 2**

Transformar o protótipo visual feito no Figma (Fase 1) em um sistema **funcional** de triagem inteligente de currículos, combinando backend (servidor), frontend (interface), banco de dados e lógica de processamento.

## **🔧 Ferramentas Utilizadas**

* **Python**: linguagem de programação principal
* **Flask**: framework web leve para Python
* **PyPDF2 / python-docx**: extração de texto de currículos em PDF ou DOCX
* **spaCy**: biblioteca de PLN (Processamento de Linguagem Natural)
* **SQLite**: banco de dados local simples
* **HTML / CSS / JavaScript**: criação das páginas e interatividade
* **Figma**: base visual das interfaces

## **📅 Cronograma (Fase 2)**

| **Dias** | **Etapas** |
| --- | --- |
| Dias 6 a 7 | Backend (servidor Flask, extração, processamento e banco de dados) |
| Dias 8 a 10 | Frontend (telas HTML, CSS, comportamento com JavaScript) |
| Dias 11 a 13 | Integração (conexão entre frontend e backend) |
| Dias 14 a 15 | Testes, ajustes visuais e validação final |

## **🔹 Dias 6-7: Criando o Backend (o "Cérebro")**

### **✅ Passo 1: Instalar o Python e Criar o Projeto**

1. Acesse o site<https://www.python.org> e instale o Python.
2. Crie uma pasta chamada apb\_vision no seu computador.

Abra o terminal e navegue até essa pasta:  
  
 bash  
CopiarEditar  
cd caminho/para/apb\_vision

Crie um ambiente virtual:  
  
 bash  
CopiarEditar  
python -m venv venv

1. Ative o ambiente:

**Windows**:  
  
 bash  
CopiarEditar  
.\venv\Scripts\activate

**Mac/Linux**:  
  
 bash  
CopiarEditar  
source venv/bin/activate

### **✅ Passo 2: Instalar as Bibliotecas**

bash

CopiarEditar

pip install Flask PyPDF2 python-docx spacy

python -m spacy download pt\_core\_news\_sm

### **✅ Passo 3: Criar o Servidor Flask**

Crie um arquivo app.py com este código:

python

CopiarEditar

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return "Servidor funcionando"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Execute com:

bash

CopiarEditar

python app.py

Acesse no navegador:<http://localhost:5000>

### **✅ Passo 4: Criar a Rota de Upload de Currículo**

python

CopiarEditar

import os

@app.route('/upload', methods=['POST'])

def upload():

arquivo = request.files['arquivo']

caminho = os.path.join('uploads', arquivo.filename)

arquivo.save(caminho)

return "Arquivo recebido com sucesso!"

### **✅ Passo 5: Extração de Texto (PDF ou DOCX)**

python

CopiarEditar

import PyPDF2, docx

def extrair\_texto(caminho):

if caminho.endswith('.pdf'):

with open(caminho, 'rb') as f:

reader = PyPDF2.PdfReader(f)

return " ".join([p.extract\_text() for p in reader.pages])

elif caminho.endswith('.docx'):

doc = docx.Document(caminho)

return " ".join([p.text for p in doc.paragraphs])

return ""

### **✅ Passo 6: Processamento de Linguagem Natural com spaCy**

python

CopiarEditar

import spacy

nlp = spacy.load('pt\_core\_news\_sm')

texto = extrair\_texto('caminho\_arquivo.pdf')

doc = nlp(texto)

palavras = [token.text for token in doc if not token.is\_stop and not token.is\_punct]

### **✅ Passo 7: Criar Pontuação com Base em Palavras-chave**

python

CopiarEditar

palavras\_chave = ['Python', 'SQL', 'Django']

pontos = sum(1 for palavra in palavras if palavra in palavras\_chave)

### **✅ Passo 8: Criar Banco de Dados SQLite**

python

CopiarEditar

import sqlite3

conn = sqlite3.connect('candidatos.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS candidatos (

id INTEGER PRIMARY KEY AUTOINCREMENT,

nome TEXT,

email TEXT,

pontuacao INTEGER,

habilidades TEXT

)

''')

conn.commit()

Para inserir dados:

python

CopiarEditar

cursor.execute('''

INSERT INTO candidatos (nome, email, pontuacao, habilidades)

VALUES (?, ?, ?, ?)

''', ("João", "joao@email.com", pontos, ", ".join(palavras)))

conn.commit()

## **🔹 Dias 8-10: Criando o Frontend (o "Rosto")**

### **✅ Passo 1: Criar as Páginas HTML**

Crie os seguintes arquivos:

* login.html (com campos de login e botão “Entrar”)
* dashboard.html (painel principal com botão de triagem)
* upload.html (tela de envio de currículos)
* processing.html (animação de carregamento)
* results.html (lista ranqueada de candidatos)
* candidate\_details.html (informações do candidato)

### **✅ Passo 2: Criar o Estilo CSS**

Crie o arquivo style.css e defina:

* Fontes
* Cores preto, branco e tons de cinza
* Margens e espaçamentos
* Responsividade

Ligue o CSS ao HTML com:

html

CopiarEditar

<link rel="stylesheet" href="style.css">

### **✅ Passo 3: Criar a Interatividade com JavaScript**

Crie o arquivo script.js:

javascript

CopiarEditar

// Simular login

document.getElementById("btnLogin").onclick = () => {

window.location.href = "dashboard.html";

};

// Envio de arquivos

document.getElementById("btnUpload").onclick = () => {

const formData = new FormData();

formData.append("arquivo", document.getElementById("arquivo").files[0]);

fetch("http://localhost:5000/upload", {

method: "POST",

body: formData

}).then(() => {

window.location.href = "processing.html";

});

};

## **🔹 Dias 11-13: Integração Backend + Frontend**

### **✅ Passo 1: Fazer o Upload de Arquivos**

A partir do JavaScript (upload.html), envie o currículo para /upload no Flask usando fetch.

### **✅ Passo 2: Exibir Tela de Processamento**

Após o upload, redirecione para processing.html.

### **✅ Passo 3: Receber Resultados e Preencher Lista**

O backend envia uma lista de candidatos e pontuações. No results.html, use JavaScript para preencher a tabela dinamicamente.

### **✅ Passo 4: Ver Detalhes de Candidato**

Na lista de candidatos, cada botão "Ver Detalhes" redireciona para candidate\_details.html, carregando os dados completos (via backend ou simulado).

## **🔹 Dias 14-15: Testes e Polimento**

### **✅ Testar o Sistema Completo**

1. Simule login
2. Envie diferentes tipos de currículos
3. Verifique pontuações
4. Teste a navegação entre as páginas

### **✅ Corrigir Bugs**

Use o console do navegador e o terminal para identificar e corrigir erros. Use print() no Python e console.log() no JS.

### **✅ Ajustes Visuais**

* Alinhamentos
* Tamanhos de botões
* Textos claros e legíveis

### **✅ Preparar para Validação**

* Deixe alguns currículos de teste separados
* Prepare um cenário que demonstre claramente a pontuação automática
* Mostre que o sistema é funcional e compreensível

## **✅ Resultado Final Esperado**

* **Backend** rodando com Flask, extração e banco de dados
* **Frontend** completo e funcional
* **Integração** entre backend e frontend
* **Triagem automática** de currículos
* **Sistema pronto para ser apresentado ou validado**