**app.py**

import streamlit as st

from services.blob\_services import upload\_blob

from services.credit\_card\_services import analize\_credit\_card

def configure\_interface():

st.title("Upload de Arquivo DIO Desafio 1 Azure Fake Docs")

uploaded\_file = st.file\_uploader("Escolha um arquivo", type=["png", "jpg", "jpeg"])

if uploaded\_file is not None:

file\_name = uploaded\_file.name

# Enviar para o Blob Storage

blob\_url = upload\_blob(uploaded\_file, filename) # Adicione a lógica para enviar o arquivo e obter o URL

if blob\_url:

st.write(f"Arquivo '{file\_name}' enviado com sucesso para o Azure Blob Storage")

# Chamar a função de detecção de informações de cartão de crédito

credit\_card\_info = analize\_credit\_card(blob\_url)

else:

st.write(f"Erro ao enviar o arquivo '{file\_name}' para o Azure Blob Storage")

def show\_image\_and\_validation(blob\_url, credit\_card\_info):

# Exibe a imagem enviada a partir do URL do Blob Storage

st.image(blob\_url, caption="Imagem enviada", use\_column\_width=True)

# Exibe o resultado da validação

st.write("Resultado da validação:")

# Verifica se as informações do cartão de crédito estão disponíveis

if credit\_card\_info and credit\_card\_info.get("card\_name"):

# Exibe mensagem de cartão válido

st.markdown("<h1 style='color: green;'>Cartão Válido</h1>", unsafe\_allow\_html=True)

st.write(f"Nome do Titular: {credit\_card\_info['card\_name']}")

st.write(f"Banco Emissor: {credit\_card\_info['bank\_name']}")

st.write(f"Data de Validade: {credit\_card\_info['expiry\_date']}")

else:

# Exibe mensagem de cartão inválido

st.markdown("<h1 style='color: red;'>Cartão Inválido</h1>", unsafe\_allow\_html=True)

st.write("Este não é um cartão de crédito válido.")

if \_\_name\_\_ == "\_\_main\_\_":

configure\_interface()

**Config.py**

import os

from dotenv import load\_dotenv

load\_dotenv()

class Config:

ENDPOINT = os.getenv("ENDPOINT")

KEY = os.getenv("SUBSCRIPTION\_KEY")

AZURE\_STORAGE\_CONNECTION\_STRING = os.getenv("AZURE\_STORAGE\_CONNECTION\_STRING")

CONTAINER\_NAME = os.getenv("CONTAINER\_NAME")

**blob\_service.py**

import os

import streamlit as st

from utils.Config import Config

from azure.storage.blob import BlobServiceClient

def upload\_blob(file, file\_name):

try:

blob\_service\_client = BlobServiceClient.from\_connection\_string(Config.AZURE\_STORAGE\_CONNECTION\_STRING)

blob\_client = blob\_service\_client.get\_blob\_client(container=Config.CONTAINER\_NAME, blob=file\_name)

blob\_client.upload\_blob(file, overwrite=True)

return blob\_client.url

except Exception as ex:

st.error(f"Erro ao enviar o arquivo para o Azure Blob Storage: {ex}")

return None

**requirements.txt**

azure.core

azure-ai-documentintelligence

streamlit

azure-storage-blob

python-dotenv

**arquivo dotEnv**

ENDPOINT=" Informe seu Endpoint "

SUBSCRIPTION\_KEY=" informe sua chave "

AZURE-STORAGE-CONNECTION-STRING=" informe a connection string"

CONTAINER-NAME="informe o container"

**credit\_card\_services.py**

from azure.core.credentials import AzureKeyCredential

from azure.ai.documentintelligence import DocumentIntelligenceClient

from azure.ai.documentintelligence.models import AnalyzeDocumentRequest

from utils.Config import Config

def analize\_credit\_card(card\_url):

credential = AzureKeyCredential(Config.KEY)

document\_client = DocumentIntelligenceClient(Config.ENDPOINT, credential)

card\_info = document\_client.begin\_analyze\_document(

"prebuilt-creditCard", AnalyzeDocumentRequest(url\_source=card\_url)

)

result = card\_info.result()

for document in result.documents:

fields = document.get('fields', {})

return {

"card\_name": fields.get('CardHolderName', {}).get('content'),

"card\_number": fields.get('CardNumber', {}).get('content'),

"expiry\_date": fields.get('ExpirationDate', {}).get('content'),

"bank\_name": fields.get('IssuingBank', {}).get('content'),