# **Prueba Laboral Appu: Configuración y Pasos de la Prueba de Código**

A continuación encontrarás los pasos esenciales y la descripción de la prueba de código. Incluye clonación, configuración de AWS CLI, creación de tabla DynamoDB (CLI y código), uso de boto3 y carga de archivos en S3.

## **1. Clonar el Repositorio**

Abre PowerShell o terminal.

Navega al directorio deseado:

cd C:\ruta\a\tu\carpeta

Clona el repositorio:

git clone <URL\_DEL\_REPOSITORIO>

cd nombre-del-repo

Usa Git para controlar versiones del proyecto. Se recomienda usar Git Flow (ramas main, develop, feature/\*, release/\*, hotfix/\*).

## **2. Configurar AWS CLI**

Inicializa AWS CLI (salida en JSON, región us-east-1):

aws configure

# access\_key\_id YOUR\_ACCESS\_KEY\_ID

# secret\_access\_key YOUR\_SECRET\_ACCESS\_KEY

# output json

# region us-east-1

Verifica credenciales:

aws sts get-caller-identity

## **3. Crear Tabla DynamoDB con AWS CLI**

Utiliza AWS CLI para crear la tabla DynamoDB con el nombre definido en la variable DYNAMODB\_TABLE\_NAME.

La tabla debe tener document\_id como clave primaria (HASH) y un Global Secondary Index sobre file\_name.

No se detalla el comando: cada candidato debe ejecutar la instrucción con la sintaxis de AWS CLI de su elección.

## **4. Variables de Entorno (.env)**

Crea un .env con:

AWS\_ACCESS\_KEY\_ID=YOUR\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY=YOUR\_SECRET\_ACCESS\_KEY

AWS\_DEFAULT\_REGION=us-east-1

BUCKET\_NAME=prueba-appu

DYNAMODB\_TABLE\_NAME=documents-<TU\_NOMBRE>-prueba

AWS\_S3\_CUSTOM\_DOMAIN=prueba-appu.s3.amazonaws.com

## **5. Entorno Virtual e Instalación**

python -m venv venv # crea entorno virtual

venv\Scripts\Activate.ps1 # activa en PowerShell

pip install -r requirements.txt

requirements.txt incluye:

Django==4.2.5

boto3==1.34.2

pandas==1.4.2

requests==2.29.0

python-dotenv==1.0.1

## **6. Descripción de la Prueba de Código**

Implementa una app Django documents con boto3:

### **6.1 Módulo dynamodb.py**

def create\_document(data: dict) -> dict: ...

def get\_document\_by\_id(id: str) -> dict: ...

def get\_documents\_by\_name(name: str) -> List[dict]: ...

def update\_document(id: str, updates: dict) -> dict: ...

def delete\_document(id: str) -> None: ...

### **6.2 Subida a S3 y URL pública**

import boto3

from datetime import datetime

s3 = boto3.client('s3')

file\_name = f'facturas/{image\_file.name}\_{datetime.now().strftime("%Y%m%d%H%M%S")}'

s3.upload\_fileobj(image\_file, settings.AWS\_STORAGE\_BUCKET\_NAME, file\_name)

imagen\_url = f'https://{settings.AWS\_S3\_CUSTOM\_DOMAIN}/{file\_name}'

Guarda download\_url = imagen\_url en el ítem DynamoDB.

### **6.3 Patrón MVC en Django**

Front-end libre: cualquier tecnología o framework.

URLs en urls.py:

POST /api/documents/ → crear documento

GET /api/documents/ → listar documentos

GET /api/documents/<id>/ → detalle de documento

DELETE /api/documents/<id>/ → eliminar documento

Views en views.py usando dynamodb.py y S3.

Models opcional con ORM o dynamodb.py.

Templates opcional con Bootstrap o front externo.

### **6.4 Listar y Buscar**

Listado de todos los documentos (consulta DynamoDB).

Filtro por nombre usando GSI.

### **6.5 Eliminación**

Elimina objeto en S3 y registro en DynamoDB.

## **7. Variables de Entorno (.env) - Resumen**

El archivo .env debe contener al menos las siguientes variables:

AWS\_ACCESS\_KEY\_ID=YOUR\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY=YOUR\_SECRET\_ACCESS\_KEY

AWS\_DEFAULT\_REGION=us-east-1

BUCKET\_NAME=prueba-appu

DYNAMODB\_TABLE\_NAME=documents-<TU\_NOMBRE>-prueba

AWS\_S3\_CUSTOM\_DOMAIN=prueba-appu.s3.amazonaws.com

Y en tu settings.py define:

AWS\_S3\_CUSTOM\_DOMAIN = '%s.s3.amazonaws.com' % AWS\_STORAGE\_BUCKET\_NAME

Tiempo estimado: 2 horas.

Evaluación: uso de boto3, creación de tabla con CLI, estructura en S3, MVC y tests.