# Arquitetura AWS — Sistema de Entrevistas Jurídicas (Case)

Automatiza o fluxo de entrevistas de testemunhas e colaboradores no contexto jurídico, desde o cadastro do processo até o envio e resposta do questionário, garantindo rastreabilidade, resiliência e observabilidade.

## Visão Geral

O sistema é serverless e event-driven, utilizando os principais serviços da AWS para garantir escalabilidade, baixo acoplamento e observabilidade integrada com Datadog.  
  
**Fluxo principal:**1. O usuário interno cadastra ou altera um processo.  
2. Quando o processo está com status concluído, uma Lambda envia a mensagem para a fila SQS.  
3. Uma Lambda consumidora lê o SQS, grava o status no DynamoDB e dispara o Step Function.  
4. O Step Function executa duas Lambdas: uma busca os colaboradores/testemunhas pelo CPF, e outra envia o e-mail com link de entrevista.  
5. Cada etapa grava status no DynamoDB.  
6. Em caso de erro, há até 3 tentativas e, após isso, a mensagem é enviada à DLQ.  
7. O colaborador recebe o link, responde e os dados são gravados no DynamoDB.

## Componentes AWS

|  |  |  |
| --- | --- | --- |
| Componente | Tipo | Função |
| API Gateway | Entrada | Exposição de endpoints REST para cadastro e respostas |
| Lambda - Process Handler | Função | Detecta status 'Concluído' e envia para SQS |
| Amazon SQS | Mensageria | Desacopla e garante resiliência no processamento |
| Lambda - Worker Process | Função | Consome mensagens e grava estado no DynamoDB |
| Step Functions | Orquestrador | Controla a execução das lambdas de busca e envio |
| Lambda - Buscar Colaboradores | Função | Busca testemunhas pelo CPF do ex-colaborador |
| Lambda - Enviar E-mail | Função | Dispara e-mail via SES com link de expiração |
| Lambda - Receber Respostas | Função | Recebe respostas e grava no DynamoDB |
| Amazon DynamoDB | Banco NoSQL | Persiste status e respostas |
| Datadog | Observabilidade | Monitora logs, métricas e alarmes |
| DLQ (Dead Letter Queue) | Resiliência | Armazena mensagens que falharam após 3 tentativas |
| Angular (Web App) | Interface | Front-end usado pelos colaboradores/testemunhas |

## Fluxo de Execução Detalhado

1. Cadastro do processo e envio à SQS.  
2. Consumo pela Lambda e gravação no DynamoDB.  
3. Step Function orquestra execução das Lambdas de busca e envio.  
4. Logs e status registrados em DynamoDB.  
5. DLQ usada após 3 falhas.  
6. Usuário recebe link, responde, e respostas são armazenadas.

## Observabilidade e Monitoramento

- Datadog APM: rastreamento de Lambdas e Step Functions.  
- CloudWatch Logs: registros detalhados de execução.  
- Métricas customizadas: contagem de processamentos, falhas e tempo médio.

## Resiliência e Segurança

- Retry Policy: até 3 tentativas automáticas.  
- DLQ: mensagens retidas por 14 dias.  
- Idempotência: chave única (ProcessId + CPF).  
- TTL no DynamoDB: expiração automática de links.  
- IAM: permissões mínimas necessárias.  
- KMS: criptografia dos dados sensíveis.

## Arquitetura Simplificada (Fluxo):

[Usuário Jurídico / Sistema Origem]  
 │  
 ▼  
┌──────────────────────────────┐  
│ Lambda 1 - Process Handler │  
│ Trigger: API Gateway / Event │  
│ Ação: verifica status │  
│ “Concluído” e envia p/ SQS │  
└──────────────────────────────┘  
 │  
 ▼  
┌──────────────────────────────┐  
│ SQS - Fila Principal │  
│ (mensagens de processos) │  
└──────────────────────────────┘  
 │  
 ▼  
┌──────────────────────────────┐  
│ Lambda 2 - Worker Process │  
│ Trigger: SQS │  
│ Atualiza DynamoDB │  
│ e inicia Step Function │  
└──────────────────────────────┘  
 │  
 ▼  
┌──────────────────────────────┐  
│ Step Function (Orquestrador) │  
│ Sequência de Lambdas │  
└──────────────────────────────┘  
 │ │  
 ▼ ▼  
┌──────────────┐ ┌────────────────┐  
│ Lambda 3 │ │ Lambda 4 │  
│ Buscar Dados │ │ Enviar E-mail │  
│ de Testemunha│ │ (SES / Link TTL)│  
└──────────────┘ └────────────────┘  
 │ │  
 ▼ ▼  
 └────────→ DynamoDB (Status e Logs)  
 │  
 ▼  
 ┌────────────────────────────┐  
 │ Lambda 5 - Receber Resposta│  
 │ Trigger: API Gateway (link)│  
 │ Grava respostas no DynamoDB│  
 └────────────────────────────┘

Arquitetura:

## Conclusão

A arquitetura adota Clean Architecture e event-driven design, promovendo escalabilidade, resiliência, observabilidade e segurança com mínima acoplagem entre componentes.