# Manual Técnico do Projeto MAGIC – Versão 8

## 1. Introdução ao Projeto MAGIC

Este projeto visa gerenciar um hostal no México chamado MAGIC. O sistema permite reservas, pagamentos, visualização de serviços e dashboards administrativos.

## 2. Arquitetura Geral

A arquitetura é baseada em microsserviços utilizando AWS Lambda, API Gateway, RDS, DynamoDB, Glue, Athena, SageMaker, QuickSight e um frontend Wix integrado a um chatbot com ChatGPT.

## 3. Backend (app.py)

Arquivo: app.py. Define as rotas da API para reservas, serviços e interações. Usa Flask/FastAPI.

## 4. Banco de Dados (RDS / DynamoDB)

Infraestrutura com PostgreSQL ou MySQL (RDS) e tabelas adicionais em DynamoDB. Dados de serviços, clientes e reservas.

## 5. Serviços REST com Lambda + API Gateway

Os endpoints são expostos por funções Lambda (lambda\_api\_payload.zip) conectadas via API Gateway.

## 6. Infraestrutura com Terraform

Arquivos: terraform.yml, terraform\_step\_function\_hostal.tf, terraform\_glue\_hostal.tf. Automatizam recursos como Glue e Step Functions.

## 7. Infraestrutura com CloudFormation

Arquivo: cloudformation\_glue\_hostal.yaml. Provisão alternativa com template declarativo.

## 8. Infraestrutura com AWS CDK

Arquivo: hostal\_stack.py. Define Glue, SageMaker e Step Functions programaticamente com Python.

## 9. Step Functions (orquestração)

Arquivo: step\_function\_pipeline\_hostal.json. Define fluxo entre Glue Crawler, Glue Job, e chamada ao SageMaker endpoint.

## 10. AWS Glue (crawler e job)

Provisão feita via terraform\_glue\_hostal.tf e cloudformation\_glue\_hostal.yaml. Automatiza ETL.

## 11. Amazon SageMaker (modelo)

Modelo hospedado via CDK (hostal\_stack.py). Pode ser usado para recomendações e previsão de demanda.

## 12. Amazon Athena (consultas)

Arquivo: Athena\_estrutura\_consulta.sql. Utiliza estrutura de dados no S3 para consultas analíticas.

## 13. Amazon QuickSight (painel)

Arquivo: quicksight\_analysis\_hostal.json. Permite visualização gráfica dos dados de hóspedes e reservas.

## 14. CI/CD com pipeline\_generator.py

Script que automatiza a criação de Step Functions via boto3 como alternativa ao Terraform/CloudFormation.

## 15. Ambientes Staging e Produção

Infraestrutura replicada para dois ambientes com configurações distintas. Separação lógica por variáveis.

## 16. Frontend (Next.js ou integração)

Previsto para consumo das APIs via REST. Interface será construída com foco em reservas e painel de hóspedes.

## 17. Integração com Wix (interface do cliente)

Uso do Wix Velo para conectar interface visual com as APIs do backend, exibindo dados em tempo real.

## 18. Chatbot com ChatGPT via API

Arquivo: chatbot\_handler.js. Permite integração de assistente virtual no site com a API da OpenAI.

## 19. Integração do Chatbot com o Backend

O chatbot consulta o backend em tempo real para listar serviços ou registrar reservas.

## 20. Resumo de Arquivos do Projeto

Inclui app.py, lambda\_api\_payload.zip, terraform, CDK, Glue, QuickSight, chatbot\_handler.js, pipeline\_generator.py, etc.

## 21. Instruções de Deploy e Execução

1) Provisionar infraestrutura (Terraform/CDK); 2) Subir API; 3) Configurar chatbot no Wix; 4) Executar frontend e monitorar no QuickSight.