# TECH CHALLENGE - FASE 04

Gabriel de Alcantara Rodrigues Soares - RM363700

## FIAP

### Arquitetura e Desenvolvimento em Java

# Visão Geral do Projeto

**Observação:** O vídeo de apresentação está no final do arquivo

O **FIAP ADJ8 Feedback Platform** é uma plataforma modular e escalável para coleta, gestão e monitoramento de feedbacks de alunos. Ela engloba três subprojetos principais, cada um com responsabilidades claras, permitindo automação de notificações e geração de relatórios semanais.

O projeto segue **arquitetura hexagonal**, garantindo separação de responsabilidades entre domínio, aplicação e infraestrutura, facilitando manutenção, testes e evolução futura.

## Objetivos Principais

- Centralizar o gerenciamento de feedbacks de alunos
- Notificar administradores sobre feedbacks urgentes
- Gerar relatórios semanais automatizados
- Fornecer APIs REST para consulta e integração
- Garantir segurança via roles e autenticação HTTP Basic

## Tecnologias e Ferramentas

- **Linguagem:** Java 17 / 21
- **Frameworks:** Spring Boot 3, Google Cloud Functions Framework
- **Banco de Dados:** PostgreSQL 16 (Cloud SQL)
- **Serviços GCP:** App Engine, Cloud Functions, Pub/Sub, Cloud Scheduler, Artifact Registry
- **Gerenciamento de Builds:** Maven Multi-módulo
- **Outros:** Shell scripts para gerenciamento de Service Accounts e infraestrutura, Flyway para versionamento de banco

---

# Arquitetura Geral


Arquitetura Principal

A arquitetura apresentada segue um modelo **event-driven** com componentes **serverless** e mensageria **Pub/Sub** para integração entre funções e aplicações. A sequência de operação é a seguinte:

## Cloud Scheduler

- Um job é agendado para execução semanal, especificamente todo domingo às 00:00.
- O Scheduler dispara eventos de forma programática e confiável, sem necessidade de servidores dedicados.

## Pub/Sub: `weekly-feedback-reports`

- O evento disparado pelo Cloud Scheduler é publicado em um tópico Pub/Sub chamado `weekly-feedback-reports`.
- Este tópico atua como um **broker de mensagens**, desacoplando o disparo do evento da execução da função que processa o relatório semanal.

### Weekly Report Function

- Uma **Cloud Function** que consome mensagens do tópico `weekly-feedback-reports`.
- Essa função gera ou agrega relatórios semanais baseados nos dados de feedback recebidos.
- Após o processamento, a função disponibiliza os resultados para consumo do **Feedback App** via **REST API**.

### Feedback App

- Aplicação central que armazena e gerencia todos os feedbacks dos usuários.
- Consome os relatórios gerados pela **Weekly Report Function** via chamadas REST.
- Identifica feedbacks urgentes ou críticos que precisam de atenção imediata.

### Pub/Sub: `feedback-alerts`

- Quando o **Feedback App** detecta feedbacks urgentes, ele publica eventos em um tópico Pub/Sub chamado `feedback-alerts`.
- Esse tópico serve para propagar **alertas críticos** de forma assíncrona e confiável.

### Notify Admin Function

- Função **serverless** que consome mensagens do tópico `feedback-alerts`.
- Responsável por enviar notificações ou alertas imediatos para os administradores, garantindo resposta rápida a feedbacks críticos.

## Características e Benefícios

- **Arquitetura desacoplada:** Uso de Pub/Sub evita acoplamento direto entre produtores e consumidores.
- **Serverless:** Funções e agendamento não exigem infraestrutura dedicada, garantindo escalabilidade automática.
- **Event-driven:** Todo o fluxo é baseado em eventos (Scheduler → Pub/Sub → Functions → App → Alerts).
- **Segurança e permissões:** Cada função e serviço interage via **Service Accounts** específicas, garantindo **least privilege** e rastreabilidade.
- **Automatização de relatórios e alertas:** O sistema permite gerar relatórios periódicos e disparar alertas críticos sem intervenção manual.

---

## Subprojetos

### 1. Feedback App

O **Feedback App** é uma aplicação Spring Boot responsável pelo envio, consulta e gestão de feedbacks. Ele disponibiliza endpoints REST para integração e publicação de mensagens em tópicos Pub/Sub para feedbacks urgentes.

**Principais Funcionalidades:**

- CRUD de feedbacks via REST
- Publicação de mensagens em Pub/Sub para feedbacks urgentes
- Fornecimento de dados para relatórios semanais
- Controle de acesso via roles STUDENT e ADMIN
- Integração com Cloud SQL para persistência de dados

**Fluxo de Interações:**

- Feedbacks marcados como urgentes são enviados para Pub/Sub (`feedback-alerts`) e consumidos pela função Notify Admin.
- A função Weekly Report consulta dados via REST para gerar relatórios semanais.




Feedback App

---

## 2. Notify Admin Function

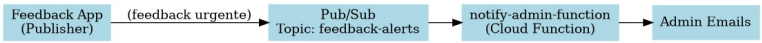
A **Notify Admin Function** é uma função do GCP responsável por enviar notificações de feedbacks urgentes aos administradores.

**Principais Características:**

- Acionada por mensagens Pub/Sub (`feedback-alerts`)
- Decodifica mensagens e converte avaliações em formato adequado
- Envia email aos administradores com notificações imediatas
- Permissões via Service Accounts dedicadas para deploy e runtime

**Fluxo Principal:**

1. Feedback App publica mensagem urgente no Pub/Sub
2. Pub/Sub aciona a função Notify Admin
3. Função envia email para administradores




Notify Admin Function

---

## 3. Weekly Report Function

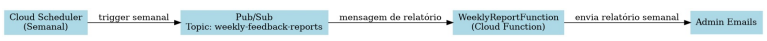
A **Weekly Report Function** gera relatórios semanais de feedbacks para administradores cadastrados.

**Principais Características:**

- Acionada automaticamente pelo **Cloud Scheduler**
- Recebe trigger via Pub/Sub (`weekly-feedback-reports`)
- Consulta feedbacks da semana via REST do Feedback App
- Gera relatório em formato HTML e envia por email
- Permissões via Service Accounts dedicadas

**Fluxo Principal:**

1. Cloud Scheduler dispara evento semanal
2. Mensagem publicada em Pub/Sub (`weekly-feedback-reports`)
3. Função consome mensagem, consulta Feedback App e gera relatório
4. Relatório enviado para administradores




Weekly Report Function

---

## Fluxos de Interações

- **Feedback Urgente:**
  Feedback App → Pub/Sub (`feedback-alerts`) → Notify Admin Function
- **Relatório Semanal:**
  Cloud Scheduler → Pub/Sub (`weekly-feedback-reports`) → Weekly Report Function → Feedback App (REST)

---

## Segurança

- Autenticação via **HTTP Basic**
- Roles:
  - STUDENT: acesso a /feedback/**
  - ADMIN: acesso a /admin/** e monitoramento geral
- Usuários pré-configurados para teste em memória
- Senhas codificadas via BCrypt
- Controle de permissões de funções e deploy via Service Accounts GCP

---

## Estrutura Geral de Projetos

```
fiap-adj8-feedback-platform/
├─ pom.xml # POM pai (Maven multi-módulo)
├─ fiap-adj8-feedback-app/ # Aplicação principal Spring Boot
├─ functions/
│ ├─ fiap-adj8-feedback-notify-admin-function/
│ └─ fiap-adj8-feedback-weekly-report-function/
```

---

## Objetivos Arquiteturais

- Modularidade e separação de responsabilidades
- Escalabilidade horizontal dos serviços
- Automação de notificações e relatórios
- Governança centralizada via projeto pai Maven
- Facilidade para evolução e manutenção futura

---

# Service Accounts, Provisionamento e Deploy no GCP

## Lista de Service Accounts e Papéis

- **sa-infra**
  - Função / Propósito: Provisionamento de infraestrutura (Cloud SQL, App Engine, Artifact Registry, Compute, etc.)
  - Roles principais:
    - cloudsql.admin
    - artifactregistry.admin
    - pubsub.admin

        - cloudscheduler.admin
        - iam.serviceAccountUser
        - resourcemanager.projectIamAdmin
        - compute.networkAdmin
        - iam.securityAdmin
        - editor
    - Observações:
        - Usada para criar recursos e configurar permissões de outros SAs.
- **sa-deploy-feedback-app**
    - Função / Propósito: Deploy da aplicação Feedback App
    - Roles principais:
        - appengine.deployer
        - artifactregistry.reader
        - appengine.serviceAdmin
        - storage.admin
        - logging.viewer / logging.logWriter
        - serviceusage.serviceUsageViewer
        - cloudbuild.builds.editor
        - iam.serviceAccountTokenCreator
        - cloudsql.client
        - secretmanager.secretAccessor
    - Observações:
        - Necessário para build/push de Docker, deploy no App Engine e acesso a segredos.
- **sa-deploy-notify-admin**
    - Função / Propósito: Deploy da Cloud Function `notify-admin`
    - Roles principais:
        - cloudfunctions.developer
        - pubsub.admin
        - logging.viewer
        - storage.admin
    - Observações:
        - Deploy da função que envia alertas por e-mail.
        - Precisa criar/atualizar tópicos Pub/Sub.
- **sa-deploy-weekly-report**
    - Função / Propósito: Deploy da Cloud Function `weekly-report`
    - Roles principais:
        - cloudfunctions.developer
        - pubsub.admin
        - cloudscheduler.admin
        - logging.viewer
        - storage.admin
    - Observações:
        - Deploy da função de geração de relatórios semanais.
        - Precisa criar tópicos Pub/Sub e jobs do Cloud Scheduler.
- **sa-runtime-feedback-app**
    - Função / Propósito: Execução da aplicação Feedback App
    - Roles principais:
        - cloudsql.client
        - pubsub.publisher
        - logging.logWriter
    - Observações:
        - Publica mensagens em tópicos Pub/Sub.
        - Conecta ao Cloud SQL.
- **sa-runtime-notify-admin**
    - Função / Propósito: Execução da função Notify Admin
    - Roles principais:
        - pubsub.subscriber
        - logging.logWriter

    - Observações:
        - Consome tópicos com alertas.
        - Envia e-mails com base nas mensagens.
- **sa-runtime-weekly-report**
    - Função / Propósito: Execução da função Weekly Report
    - Roles principais:
        - pubsub.publisher
        - logging.logWriter
    - Observações:
        - Publica mensagens de relatórios semanais.
        - Recebe eventos do Cloud Scheduler.

## Sessão de Provisionamento e Interação com GCP

- **Autenticação com SA**
    - Propósito: Garantir que scripts e deploys usem credenciais corretas
    - Ferramenta / Comando: `gcloud auth activate-service-account --key-file=KEY.json`
    - Motivação / Visão: Evita usar credenciais de usuário e garante automação segura.
- **Habilitar APIs necessárias**
    - Propósito: Disponibilizar serviços GCP para o projeto
    - Ferramenta / Comando: `gcloud services enable appengine.googleapis.com ...`
    - Motivação / Visão: Sem APIs habilitadas, recursos como Cloud SQL, Cloud Functions e Artifact Registry não funcionam.
- **Criar Cloud SQL**
    - Propósito: Banco de dados PostgreSQL
    - Ferramenta / Comando: `gcloud sql instances create ...`
    - Motivação / Visão: Armazenamento persistente de dados do Feedback App.
- **Criar Artifact Registry**
    - Propósito: Repositório de imagens Docker
    - Ferramenta / Comando: `gcloud artifacts repositories create ...`
    - Motivação / Visão: Necessário para push/pull de imagens Docker usadas pelo App Engine.
- **Criar App Engine**
    - Propósito: Hospedar a aplicação
    - Ferramenta / Comando: `gcloud app create --region ...`
    - Motivação / Visão: Ambiente gerenciado para executar a aplicação e gerenciar escalabilidade automaticamente.
- **Build & Push Docker**
    - Propósito: Preparar imagem da aplicação
    - Ferramenta / Comando: `docker build`, `docker tag`, `docker push`
    - Motivação / Visão: Centraliza a aplicação em um container versionado para deploy no GCP.
- **Deploy App Engine**
    - Propósito: Colocar a aplicação em produção
    - Ferramenta / Comando: `gcloud app deploy`
    - Motivação / Visão: Publicação final da aplicação com variáveis de ambiente configuradas.
- **Criar Tópicos Pub/Sub**
    - Propósito: Comunicação assíncrona
    - Ferramenta / Comando: `gcloud pubsub topics create`
    - Motivação / Visão: Permite desacoplar funções (Notify Admin, Weekly Report) do app principal.
- **Deploy Cloud Functions**

    - Propósito: Funções serverless
    - Ferramenta / Comando: `gcloud functions deploy`
    - Motivação / Visão: Automação de alertas e relatórios; funções podem escalar automaticamente.
- **Criar Cloud Scheduler Jobs**
    - Propósito: Agendamento de tarefas
    - Ferramenta / Comando: `gcloud scheduler jobs create pubsub`
    - Motivação / Visão: Garante execução semanal da função de relatórios.
- **EnvVars / .env**
    - Propósito: Configuração central
    - Ferramenta / Comando: `envsubst` / `env.yaml`
    - Motivação / Visão: Mantém segredos e URLs de serviços separados do código-fonte.
- **Validação final**
    - Propósito: Teste do deploy
    - Ferramenta / Comando: `gcloud pubsub topics publish`, `gcloud functions logs read`
    - Motivação / Visão: Confirma que a comunicação entre serviços e funções está funcionando.

## Motivo geral da arquitetura e automação

1. **Segurança e segregação de responsabilidades:** cada SA tem permissões mínimas necessárias (princípio do least privilege).
2. **Automação e reprodutibilidade:** scripts permitem provisionar infra, criar roles, SA, deploys e funções de forma repetível.
3. **Desacoplamento e escalabilidade:** Pub/Sub + Cloud Functions + Scheduler permite que cada módulo (alertas, relatórios, app principal) funcione de forma independente.
4. **Observabilidade:** roles de logging e roles de execução permitem monitorar ações de cada SA.

---

## Observação

Os detalhes de cada comando, criação de SAs, provisionamento de infra e deploy estão nos arquivos:

- `manage-sa.sh` (root)
- `infra.sh` (root)
- `deploy.sh` em cada projeto correspondente

### PARA DETALHES MAIS TÉCNICOS, visualize o README.md de cada projeto, partindo de: README.md

## Clonagem do Repositório e Atualização com Submodules

Para clonar o projeto `fiap-adj8-feedback-platform`, você pode usar **SSH** ou **HTTPS**, dependendo de como suas chaves/credenciais estão configuradas:

### Clonando via SSH

```
        git clone --recurse-submodules git@github.com:gabriel-dears/fiap-adj8-feedback-platform.git
```

### Clonando via HTTPS

```
        git clone --recurse-submodules https://github.com/gabriel-
dears/fiap-adj8-feedback-platform.git
```

⚠ **O parâmetro –recurse-submodules garante que todos os submodules do projeto também sejam clonados automaticamente.**

Claro! Aqui está o conteúdo formatado em Markdown:

## Atualizando o Repositório com Submodules

Depois de já ter o repositório clonado, para atualizar o código principal e todos os submodules, use:

```
        git pull --recurse-submodules
        git submodule update --remote
```

- **git pull –recurse-submodules:** Atualiza o branch principal e verifica se os submodules têm commits novos.
- **git submodule update –remote:** Sincroniza os submodules com os commits mais recentes dos repositórios remotos.

## Link para o GitHub:

- **https://github.com/gabriel-dears/fiap-adj8-feedback-platform**

## Vídeo de apresentação

- **https://drive.google.com/file/d/1IAia4uFERwWyZAtmCtmC5Ki3TJ0W3r2r/view?usp=sharing**