

1. Contextualização do Desafio

No cenário corporativo atual, a agilidade na comunicação interna é vital. No entanto, departamentos de Recursos Humanos frequentemente enfrentam uma sobrecarga operacional ao redigir comunicados, avisos e e-mails personalizados para diferentes públicos. O desafio consistia em criar uma ferramenta que automatizasse essa redação, garantindo que o texto final fosse profissional, coerente e adaptado a tons de voz específicos, partindo apenas de tópicos brutos fornecidos pelo usuário.

2. Justificativa para uso de IA Generativa na Solução

Diferente de automações baseadas em modelos (templates) estáticos, a IA Generativa permite:

- **Flexibilidade:** Adaptar o mesmo conjunto de informações para públicos distintos (ex: diretoria vs. operacional).
- **Criatividade e Coesão:** Transformar listas de tópicos em textos fluidos e naturais.
- **Personalização em Escala:** Ajustar o tom de voz (empático, formal ou direto) sem a necessidade de intervenção humana em cada mensagem, reduzindo o tempo de produção de minutos para segundos.

3. Explicação sobre o Modelo LLM Utilizado

Para esta solução, optou-se pela utilização do modelo **GPT-4o mini**. Este modelo foi escolhido por ser uma das opções mais eficientes e inteligentes da OpenAI, oferecendo um equilíbrio ideal entre alta performance e baixa latência. O GPT-4o mini é otimizado para tarefas de raciocínio rápido e redação, tornando-o perfeito para uma ferramenta de produtividade que exige respostas quase instantâneas com um processamento de linguagem natural refinado em português.

4. Elaboração do Prompt

O prompt foi elaborado utilizando a técnica de **Persona e Instrução Estruturada** para minimizar alucinações e garantir o alinhamento corporativo. A estrutura enviada à IA segue este padrão:

1. **Atribuição de Papel:** Define a IA como uma "Especialista em Redação de RH".
2. **Variáveis de Contexto:** Injeção dinâmica dos dados capturados pelo formulário (Tipo de Texto, Público, Tom).
3. **Delimitação de Conteúdo:** Instrução clara para basear o texto exclusivamente nos tópicos fornecidos.
4. **Formatação:** Comando específico para entregar apenas o texto final, pronto para revisão e envio, sem metadados ou comentários adicionais da IA.

5. Benefícios Percebidos e Desafios Enfrentados

- **Benefícios:** Padronização da comunicação interna, redução drástica do tempo de resposta do RH e facilidade de uso por colaboradores sem conhecimento técnico.
- **Desafios:** O principal desafio foi a instabilidade inicial de endpoints e versões beta de outros modelos. A migração para o GPT-4o mini através de uma interface estável resolveu problemas de conexão e permitiu um tratamento de erros mais robusto via blocos `try-catch` no Google Apps Script.

6. Limites Éticos e de Segurança

- **LGPD e Vazamento de Dados:** A solução foi configurada para não armazenar dados sensíveis. O formulário instrui expressamente o usuário a não inserir dados pessoais identificáveis (como CPFs ou salários) nos campos de tópicos.
- **Viés da IA:** Como modelos LLM podem reproduzir preconceitos contidos em sua base de treinamento, o fluxo do CorpGenius estabelece que o texto gerado é um **rascunho**. O e-mail automático serve para que o solicitante atue como um revisor humano obrigatório antes de qualquer publicação oficial.
- **Segurança:** As chaves de API foram tratadas como segredos de sistema, sendo removidas do código-fonte antes de qualquer publicação em repositórios públicos como o GitHub, seguindo as melhores práticas de cibersegurança.

Código do AppsScript (.gs):

```
function salvarChave() {
  PropertiesService.getScriptProperties()
    .setProperty("OPENAI_API_KEY", "Sua Chave");
}

function onFormSubmit(e) {
  if (!e || !e.values) return;

  var respostas = e.values;
  var userEmail = respostas[1];
  var tipoTexto = respostas[2];
  var topics = respostas[3];
  var publico = respostas[4];
  var tomVoz = respostas[5];

  var prompt = `Atue como redator de RH.  
Crie um ${tipoTexto} para ${publico},`
```

```
com tom ${tomVoz}.

Conteúdo: ${topicos}
`;

var apiKey = PropertiesService.getScriptProperties()
.getProperty("OPENAI_API_KEY");

var url = "https://api.openai.com/v1/chat/completions";

var payload = {
  model: "gpt-4o-mini",
  messages: [{ role: "user", content: prompt }],
  temperature: 0.7
};

var options = {
  method: "post",
  contentType: "application/json",
  headers: {
    Authorization: "Bearer " + apiKey
  },
  payload: JSON.stringify(payload),
  muteHttpExceptions: true
};

try {
  var response = UrlFetchApp.fetch(url, options);
  var json = JSON.parse(response.getContentText());

  var textoIA = json.choices[0].message.content;

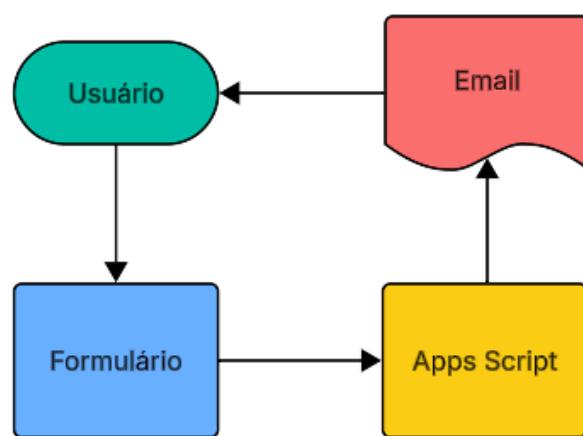
  MailApp.sendEmail(
    userEmail,
    "Seu texto está pronto",
    textoIA
  );

} catch (err) {
  console.error("Erro:", err);
}
```

```
 }  
 }
```

Link do Projeto:

[Github](#)



[Drive com o Vídeo](#)