**Relatório Técnico de Projeto: Sistema Inteligente para Geração de Vale Refeição (VR)**

**Relatório Técnico de Projeto: Sistema Inteligente para Geração de Vale Refeição (VR)**

**Autores:** Ellen Heidi

**Data:** 9 de Setembro de 2025

**Curso Agentes Autônomos com redes Generativas**

**Professor:** Celso Azevedo

**1. Introdução e Objetivo do Projeto**

O presente projeto teve como objetivo o desenvolvimento de uma solução tecnológica avançada para automatizar, consolidar e otimizar o complexo processo de cálculo e geração da folha de pagamento do Vale Refeição (VR). O desafio consistiu em evoluir de um script inicial para uma aplicação web robusta, capaz de interpretar múltiplas fontes de dados e aplicar um conjunto detalhado de regras de negócio, em conformidade com a legislação trabalhista e acordos sindicais.

A solução final é uma ferramenta inteligente e acessível, projetada para o departamento de Recursos Humanos, que utiliza a Inteligência Artificial do Google (Gemini) como um motor de pré-análise, complementado por uma sólida lógica de cálculo em Python.

**2. O Problema: Requisitos de Negócio Detalhados**

O desafio central era substituir um processo manual propenso a erros por um sistema automatizado que atendesse aos seguintes requisitos:

* **Consolidação de Múltiplas Bases:** O sistema precisava unificar dados de diversas folhas de cálculo (ATIVOS, FÉRIAS, DESLIGADOS, ADMISSÃO ABRIL, etc.), garantindo que nenhum funcionário fosse omitido e que as informações mais recentes (como datas de admissão e desligamento) tivessem prioridade.
* **Tratamento de Exclusões:** A aplicação deveria identificar e excluir do pagamento final todos os profissionais inelegíveis, como diretores, estagiários, aprendizes, funcionários em férias, afastados (exceto casos especiais como licença-maternidade, dependente de acordo coletivo) e aqueles que atuam no exterior.
* **Cálculo Automatizado e Proporcional:** O cálculo do benefício não é fixo. Era mandatório que o sistema calculasse a quantidade de dias úteis a pagar de forma dinâmica, considerando:
  + **Admissões no Mês:** Pagamento proporcional a partir da data de admissão.
  + **Desligamentos no Mês:** Pagamento proporcional até à data de desligamento.
* **Regras de Negócio Específicas:** Uma regra crítica de negócio precisava ser implementada: para funcionários desligados, se o desligamento ocorresse **até** ao **dia 15** do mês, o benefício não deveria ser pago. Se ocorresse após o dia 15, o pagamento proporcional seria devido.
* **Variações Sindicais:** O sistema deveria aplicar o valor diário do VR e o número de dias úteis base de cada sindicato de forma individual para cada funcionário.
* **Entrega Final:** A aplicação deveria gerar uma folha de cálculo no formato exato exigido pela operadora, incluindo colunas para matrícula, admissão, sindicato, competência, dias, valores e a divisão de custos (80% empresa, 20% profissional), além de uma aba de validação para auditoria.

**3. A Solução Proposta: Uma Arquitetura Híbrida**

A solução implementada utiliza uma arquitetura híbrida, onde a Inteligência Artificial e a lógica de programação tradicional trabalham em conjunto, cada uma na sua área de especialidade.

**3.1. Lógica de Negócio (Backend)**

A classe SistemaVRComGemini foi o núcleo do desenvolvimento, com os seguintes componentes-chave:

* **Consolidação de Dados Robusta (carregar\_dados):** Esta função foi reescrita para ser mais inteligente. Ela primeiro cria uma base com todos os funcionários da folha "ATIVOS". Em seguida, ela itera sobre as outras folhas de cálculo (DESLIGADOS, ADMISSÃO ABRIL, etc.) e não apenas adiciona novos funcionários, mas **atualiza os registos existentes**. Por exemplo, se uma matrícula já existe, a data de desligamento da folha "DESLIGADOS" substitui a informação anterior, garantindo a precisão dos dados.
* **Motor de Pré-Análise com IA (consultar\_gemini\_elegibilidade):** A função da IA foi redefinida. Em vez de tomar a decisão final, o Gemini agora atua como um **pré-processador inteligente**. O "prompt" solicita que a IA exclua apenas os casos inequivocamente inelegíveis (estagiários, diretores, etc.). Crucialmente, casos complexos como **funcionários desligados** são classificados como **"elegíveis para o processo"**, permitindo que a lógica de cálculo em Python tome a decisão final sobre o pagamento. Esta abordagem híbrida aproveita a capacidade da IA de interpretar texto (cargos, situações) e reserva os cálculos precisos para o código.
* **Motor de Regras de Negócio (processar\_beneficio):** Esta é a função mais crítica do sistema. Após a pré-análise da IA, ela executa as regras de negócio:
  1. **Identifica o Sindicato:** Aplica as regras de dias e valores corretas para cada funcionário.
  2. **Verifica a Situação:** Distingue entre funcionários de mês completo e aqueles com cálculo proporcional.
  3. **Aplica a Regra de Desligamento:** Verifica a data de desligamento. Se for até ao dia 15, o funcionário é marcado como inelegível para pagamento e movido para a folha de validação. Caso contrário, prossegue para o cálculo proporcional.
  4. **Calcula os Dias:** Para funcionários de mês completo, utiliza os "dias base" do sindicato. Para admitidos e desligados, chama a função calcular\_dias\_uteis\_proporcionais para obter o número exato de dias trabalhados.
  5. **Trata a Licença-Maternidade:** Funcionárias nesta situação recebem o valor integral, assumindo que o acordo coletivo garante o benefício.
* **Geração da Folha de Cálculo Final (gerar\_planilha\_final):** O método final gera uma planilha Excel no formato solicitado, com todas as colunas na ordem correta e com os nomes exigidos, pronto para ser enviado à operadora.

**3.2. Interface do Utilizador (Frontend)**

Após dificuldades com outras bibliotecas no ambiente de nuvem, optou-se pelo **Gradio**. Esta escolha demonstrou ser mais estável e permitiu a criação de uma interface limpa e funcional, focada na simplicidade para o utilizador final.

**3.3. Publicação e Acessibilidade (Deployment)**

A aplicação foi publicada na plataforma **Hugging Face Spaces**, utilizando o seu SDK nativo para o Gradio. Esta abordagem eliminou a complexidade do Docker e provou ser a mais fácil implementação. A chave de API do Gemini foi armazenada de forma segura utilizando a funcionalidade "Repository secrets".

**4. Conclusão**

O projeto final é uma demonstração clara de como cursistas, com o auxílio da IA resolveram problemas de negócio complexos. Ao delegar a interpretação de texto à IA e a execução de regras numéricas estritas ao código Python, a aplicação alcança um alto grau de precisão e automação. O sistema cumpre todos os requisitos detalhados, transformando um processo manual, propenso a erros, numa solução automatizada e confiável.