Serviço Público Federal

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DA INTEGRAÇÃO LATINO-AMERICANA
SISTEMA INTEGRADO DE PATRIMÔNIO, ADMINISTRAÇÃO E CONTRATOS

PROCESSO 23422.009795/2026-01

ELETRÔNICO

Cadastrado em 12/05/2026

Processo disponível para recebimento com código de barras/QR Code

Nome(s) do Interessado(s):	E-mail:	Identificador:
AUDITORIA INTERNA	auditoria@unila.edu.br	10010516

Tipo do Processo:
PROPOSTA

Assunto do Processo:
NÃO DEFINIDO

Assunto Detalhado:
PLANEJAMENTO DO DESENVOLVIMENTO DO SOFTWARE AVALRIG

Unidade de Origem:
AUDITORIA INTERNA (10.01.05.16)

Criado Por:
MARIA ELIZA RATUCZNE CARDENAS

Observação:

MOVIMENTAÇÕES ASSOCIADAS

Data	Destino	Data	Destino

Para visualizar este processo, entre no **Portal Público** em https://sig.unila.edu.br/public e acesse a Consulta de Processos.

Visualizar no Portal Público

AvalRIG: Planejamento reverso do desenvolvimento de uma ferramenta de avaliação do Relatório Integrado de Gestão da UNILA

FOZ DO IGUAÇU

2026

Auditoria Interna – AUDIN

Guillermo Javier Díaz Villavicencio

William Mori Junior

André Rodrigues Matsumoto

Davi Camargo de França

Maria Eliza Ratuczne Cardenas

Noelle Mariana Santos Araujo Fritzen

Elaboração:

André Rodrigues Matsumoto - Economista

SUMÁRIO

1. INTRODUÇÃO

O presente documento apresenta o planejamento do desenvolvimento do software AvalRIG, uma aplicação em Python voltada à avaliação automatizada do Relatório Integrado de Gestão (RIG) da UNILA, tendo como objetivo verificar sua conformidade:

- ✅ com os princípios e diretrizes de elaboração do Relatório Integrado de Gestão estabelecidos pela Instrução Normativa TCU nº 84/2020; e
- ✅ com os elementos de conteúdo exigidos e organizados em capítulos, conforme o Anexo da Decisão Normativa TCU nº 198/2022.

O AvalRIG configura-se como uma ferramenta de apoio à auditoria, concebida para simular a atuação de um auditor do Tribunal de Contas da União (TCU), utilizando recursos de IA generativa (Google Gemini) para atribuição de pontuações e elaboração de fundamentações técnicas sobre o grau de aderência do RIG aos normativos aplicáveis.

Embora se apresente como um documento de planejamento, sua elaboração segue uma lógica de planejamento reverso (*planning from the end*). Parte-se do sistema já implementado para, então, reconstruir e explicitar objetivos, requisitos, arquitetura e decisões de projeto, assegurando alinhamento entre:

- o produto final desejado (um avaliador automatizado do RIG aderente à IN TCU nº 84/2020 e à DN TCU nº 198/2022);
- as evidências de desempenho esperadas no processo de avaliação; e
- a organização da solução tecnológica desenvolvida a partir desses referenciais.

2. JUSTIFICATIVA

A elaboração de Relatórios Integrados de Gestão exige:

- aderência a princípios como foco estratégico, conectividade da informação, materialidade, concisão, confiabilidade, comparabilidade, clareza, tempestividade e transparência;
- cumprimento de elementos de conteúdo específicos para cada capítulo (mensagem

do dirigente, visão organizacional, riscos e oportunidades, governança e desempenho, informações orçamentárias, financeiras e contábeis).

Na prática, a avaliação desses relatórios é trabalhosa, subjetiva e demandante de tempo, especialmente quando se busca:

- fundamentação técnica detalhada para cada quesito/elemento;
- consistência entre diferentes avaliadores e ciclos de avaliação;
- geração rápida de relatórios consolidados.

O AvalRIG responde a essa demanda ao:

- permitir carregar o RIG em PDF e extrair seu conteúdo textual;
- estruturar a avaliação em dois módulos (Princípios e Elementos de Conteúdo), conforme as diretrizes normativas;
- utilizar modelos de IA para propor pontuações (0–3) e gerar argumentos técnicos padronizados, em linguagem de parecer do TCU;
- consolidar as avaliações em relatórios textuais com totais, percentuais e conceito geral.

Assim, justifica-se o desenvolvimento do AvalRIG como:

- apoio metodológico à avaliação do RIG da UNILA;
- instrumento de sistematização da análise, com ganho de transparência, rastreabilidade e comparabilidade ao longo do tempo;
- exemplo de aplicação prática de IA generativa na auditoria pública.

3. OBJETIVOS

3.1. Objetivo geral

Desenvolver o software AvalRIG, em Python, com interface gráfica, capaz de avaliar o Relatório Integrado de Gestão da UNILA nos eixos de Princípios e Elementos de Conteúdo, atribuindo pontuação e produzindo fundamentação técnica para subsidiar o processo de auditoria e melhoria contínua da prestação de contas.

3.2. Objetivos específicos

1. Implementar, no AvalRIG, um módulo de avaliação dos Princípios do RIG, contemplando os 10 princípios e seus respectivos quesitos, com escala de 0 a 3:
    - 0 – Insuficiente
    - 1 – Razoável
    - 2 – Bom
    - 3 – Muito Bom
2. Implementar, no AvalRIG, um módulo de avaliação dos Elementos de Conteúdo, contemplando 5 capítulos e 26 elementos, com escala de 0 a 3:
    - 0 – Item faltando no RIG
    - 1 – Consta, mas insatisfatório
    - 2 – Consta, medianamente satisfatório
    - 3 – Consta de modo satisfatório
3. Permitir, no AvalRIG, o carregamento de um relatório em formato PDF, com extração de texto (via PyMuPDF e/ou pdfplumber).
4. Integrar o AvalRIG à API Google Gemini, de modo que:
    - cada quesito/elemento seja avaliado automaticamente pela IA,
    - com retorno em JSON contendo pontuacao (0–3) e argumento técnico.
5. Apresentar ao usuário:
    - interface gráfica amigável (Tkinter);
    - comboboxes para seleção de princípio/quesito e capítulo/elemento;
    - exibição da pontuação e da fundamentação;
    - tabelas de pontuações e relatórios consolidados.
6. Possibilitar exportação dos resultados produzidos pelo AvalRIG:
    - exportar fundamentação de um quesito/elemento em .txt;
    - exportar relatórios consolidados (Princípios e Elementos) em .txt.
7. Implementar no AvalRIG mecanismos de feedback e status (barra de status, progress bar), bem como validações de:
    - API Key configurada;
    - PDF carregado;
    - bibliotecas obrigatórias instaladas.

4. ESCOPO DO SISTEMA

4.1. Escopo funcional

O AvalRIG contempla os seguintes requisitos funcionais (RF):

- RF01 – Carregar PDF:
  Permitir a seleção de um arquivo PDF e extrair seu texto para uso nas avaliações.
- RF02 – Configurar API Gemini:
  Permitir informar e ocultar/mostrar a chave da API (GEMINI_API_KEY) e selecionar o modelo Gemini a utilizar.
- RF03 – Avaliação por Princípios:
  - Selecionar um princípio (1 a 10) e, em seguida, um quesito associado.
  - Enviar, para a IA, o texto do relatório + descrição do princípio + texto do quesito.
  - Receber JSON com pontuacao (0–3) e argumento.
  - Armazenar o resultado em memória (resultados_principios).
  - Exibir pontuação, cor associada e fundamentação.
  - Permitir avaliar todos os quesitos de um princípio em lote.
- RF04 – Avaliação por Elementos de Conteúdo:
  - Selecionar um capítulo (1 a 5) e, em seguida, um elemento.
  - Enviar o contexto do capítulo + texto do elemento + texto do relatório para a IA.
  - Receber e armazenar pontuacao e argumento em resultados_elementos.
  - Exibir resultado e permitir avaliar todos os elementos de um capítulo em lote.
- RF05 – Tabelas de Pontuação:
  - Exibir, em Treeview, listagem de todos os quesitos/elementos avaliados, com:
  - identificação (Princípio/Quesito ou Capítulo/Elemento),
  - pontuação numérica,
  - descrição textual da classificação,
  - cores/estilos visuais por faixa de nota.
- RF06 – Relatórios Consolidados:
  - Gerar, para Princípios:

    - subtotal por princípio (pontos obtidos, máximo, porcentagem),
    - resumo geral (pontuação total, percentual e conceito geral).
    - Gerar, para Elementos:
    - subtotal por capítulo,
    - resumo geral semelhante ao de princípios.
    - Mostrar esses relatórios em campos de texto roláveis.
- RF07 – Exportações:
    - Exportar o resultado de um quesito/elemento em arquivo .txt.
    - Exportar relatório consolidado (Princípios/Elementos) em .txt.
- RF08 – Verificação de Dependências:
  Na inicialização do AvalRIG, verificar se as bibliotecas obrigatórias estão instaladas. Caso não, exibir mensagem com instruções de instalação.

4.2. Escopo não funcional

- RNF01 – Linguagem e Plataforma:
  Implementação do AvalRIG em Python 3, com interface via Tkinter.
- RNF02 – Usabilidade:
    - Interface com abas separando claramente os dois módulos.
    - Uso de cores, ícones e textos explicativos (labels, mensagens de status).
    - Mensagens de status amigáveis e claras.
- RNF03 – Desempenho:
    - Extração de texto otimizada, com uso de threads para não travar a interface.
    - Limitação de caracteres enviados à IA (truncar_texto) para respeitar limites de contexto.
- RNF04 – Segurança:
    - Ocultação da API Key no campo de texto (modo senha).
    - Uso apenas de arquivos de saída explícitos (exportações), sem armazenamento desnecessário de dados sensíveis.
- RNF05 – Manutenibilidade:
    - Organização modular (funções de avaliação, extração, GUI).
    - Centralização das estruturas de dados (PRINCIPIOS, CAPITULOS, escalas de pontuação).

5. ARQUITETURA DO SOFTWARE

5.1. Visão Geral

A arquitetura do AvalRIG segue um modelo simples, em camadas lógicas:

1. Camada de Interface (GUI)
    - Classe principal AvalRIGApp(tk.Tk)
    - Responsável por:
        - janelas, abas, comboboxes, botões, textos;
        - exibição de resultados, relatórios e mensagens.
2. Camada de Negócio / Domínio
    - Estruturas de dados:
        - PRINCIPIOS (dicionário: princípio → descrição → quesitos),
        - CAPITULOS (capítulo → descrição → elementos),
        - escalas de pontuação (SCORES, SCORES_ELEMENTOS).
    - Lógica de avaliação:
        - funções avaliar_quesito e avaliar_elemento, que:
        - constroem prompts,
        - chamam o modelo Gemini,
        - tratam o retorno JSON.
3. Camada de Acesso a Serviços Externos
    - Módulo de integração com:
        - API Google Gemini (google.generativeai),
        - bibliotecas de leitura de PDF (fitz / pdfplumber).
4. Camada de Utilidade
    - Funções auxiliares como:
        - extrair_texto_pdf,
        - truncar_texto,
        - verificar_dependencias,
        - métodos de exportação e salvamento de relatórios.

6. DESCRIÇÃO DOS MÓDULOS

6.1. Módulo 1 – Avaliação dos Princípios

Objetivo do módulo:
No contexto do AvalRIG, avaliar, para cada quesito de cada princípio, o nível de atendimento no RIG, atribuindo nota de 0 a 3 e produzindo fundamentação técnica.

Funcionalidades:

- Seleção de Princípio (combobox populado a partir de PRINCIPIOS).
- Seleção de Quesito (combobox alimentado dinamicamente conforme o princípio).
- Exibição da descrição do princípio.
- Botão “Avaliar Quesito (Gemini)”:
    - chama avaliar_quesito(...).
- Botão “Avaliar Princípio Completo”:
    - percorre todos os quesitos e avalia em sequência, com feedback de progresso.
- Aba “Quesito Atual”:
    - mostra rótulo com princípio/quesito,
    - pontuação com cor,
    - texto da fundamentação técnica.
- Aba “Tabela de Pontuações”:
    - Treeview com todas as avaliações de quesitos.
- Aba “Relatório Consolidado”:
    - relatório textual por princípio, com:
    - listagem dos quesitos avaliados,
    - subtotal por princípio,
    - resumo geral e conceito final.

6.2. Módulo 2 – Avaliação dos Elementos de Conteúdo

Objetivo do módulo:
Dentro do AvalRIG, verificar a presença e a qualidade dos elementos de conteúdo obrigatórios do RIG, por capítulo, com escala de 0 a 3.

Funcionalidades:

- Seleção de Capítulo (combobox via CAPITULOS).
- Seleção de Elemento (combobox dinâmico, com preview abreviado do elemento).
- Exibição de:
    - descrição do capítulo,
    - descrição do elemento selecionado.
- Botão “Avaliar Elemento (Gemini)”:
    - chama avaliar_elemento(...).
- Botão “Avaliar Capítulo Completo”:
    - avalia todos os elementos do capítulo selecionado, em sequência.
- Aba “Elemento Atual”:
    - exibe Capítulo/Elemento,
    - pontuação com cor,
    - fundamentação técnica.
- Aba “Tabela de Pontuações”:
    - Treeview com todas as avaliações por capítulo/elemento.
- Aba “Relatório Consolidado”:
    - relatório por capítulo, com:
    - pontuações por elemento,
    - subtotais por capítulo,
    - pontuação total e conceito geral.

7. INTEGRAÇÕES E TECNOLOGIAS UTILIZADAS

- Linguagem: Python 3
- Interface gráfica (AvalRIGApp): tkinter e ttk
- Leitura de PDF (AvalRIG):
    - PyMuPDF (fitz) – tentativa primária;
    - pdfplumber – fallback, caso fitz não esteja disponível.
- IA Generativa:
    - google.generativeai (API do Google Gemini)
    - Modelos configuráveis:
    - "gemini-3.1-pro", "gemini-2.5-pro", "gemini-2.0-flash", "gemini-2.5-flash",

"gemini-1.5-flash"

- Multithreading:
    - threading.Thread para evitar travamento da interface do AvalRIG durante:
    - extração de PDF,
    - chamadas à API Gemini.

8. RISCOS E PREMISSAS

8.1. Riscos identificados

- Dependência de serviços externos (API Gemini):
    - indisponibilidade da API;
    - limitação de uso (quota);
    - variações de resposta e atualização de modelos.
- Qualidade da extração de texto do PDF:
    - PDFs digitalizados ou com layout complexo podem gerar texto incompleto ou confuso;
    - isso impacta diretamente a qualidade das avaliações da IA no AvalRIG.
- Interpretação da IA:
    - apesar da instrução detalhada (prompt), pode haver:
    - superestimação/subestimação de notas,
    - uso de evidências não desejadas,
    - erros pontuais no JSON retornado (tratados via json.loads e sanificação mínima).
- Mudanças normativas:
    - alterações futuras nas diretrizes de RIG podem exigir atualização de:
    - princípios, quesitos,
    - capítulos, elementos,
    - escalas de avaliação no AvalRIG.

8.2. Premissas

- O usuário do AvalRIG dispõe de:
    - PDF do RIG da UNILA em formato textual (não apenas imagem);

    - chave válida da API Gemini;
    - ambiente Python com as bibliotecas requeridas instaladas.
- O AvalRIG é um apoio à avaliação, não substituto da análise humana:
    - o avaliador pode revisar, discutir e complementar os argumentos técnicos gerados.

9. CRONOGRAMA (PLANEJAMENTO REVERSO)

Aplicando a lógica de planejamento reverso ao AvalRIG:

1. Produto final desejado

   O AvalRIG totalmente funcional, com:
    - dois módulos (Princípios e Elementos),
    - avaliação automática via IA,
    - relatórios consolidados exportáveis.
2. Evidências de conclusão
    - Código Python executável;
    - Interface do AvalRIG funcionando (classe AvalRIGApp);
    - Avaliações realizadas com sucesso em um RIG da UNILA;
    - Arquivos de saída (.txt) de relatórios.
3. Desdobramento em etapas (de trás para frente)

- Etapa 5 – Ajustes e refino
    - Ajuste visual (cores, temas, labels).
    - Ajuste de textos de status e mensagens de erro.
    - Revisão da estrutura dos relatórios gerados.
- Etapa 4 – Geração de relatórios e exportações
    - Implementar _gerar_relatorio_principios e _gerar_relatorio_elementos.
    - Implementar _salvar_relatorio, _exportar_quesito, _exportar_elemento.
- Etapa 3 – Lógica de avaliação com IA
    - Implementar avaliar_quesito e avaliar_elemento com:
    - prompts detalhados,
    - escala de pontuação,
    - retorno em JSON.

    - Integrar essas funções com os botões da interface e tabelas do AvalRIGApp.
- Etapa 2 – Interface gráfica
    - Criar AvalRIGApp, abas para módulos, comboboxes, text areas.
    - Criar barras de status e progress bar.
    - Implementar callbacks de seleção (princípios/quesitos, capítulos/elementos).
- Etapa 1 – Base de dados e serviços
    - Modelar e codificar:
    - PRINCIPIOS e CAPITULOS;
    - escalas SCORES e SCORES_ELEMENTOS.
    - Implementar extrair_texto_pdf com suporte a PyMuPDF e pdfplumber.
    - Implementar verificar_dependencias.

10. CONCLUSÃO

O planejamento aqui descrito reconstrói, de forma reversa, a lógica de concepção do AvalRIG, um software de avaliação do Relatório Integrado de Gestão da UNILA. Demonstra-se como o sistema que permite:

- traduzir os princípios e elementos de conteúdo em estruturas de dados e fluxos de interação;
- integrar extração de texto, IA generativa e interface gráfica;
- oferecer pontuações e argumentos técnicos padronizados, com relatórios consolidados.

Essa documentação confere ao AvalRIG uma base metodológica e técnica clara, permitindo sua apresentação em contextos acadêmicos, institucionais ou de auditoria, reforçando a transparência quanto ao seu propósito, funcionamento e limitações.

PROPOSTAS Nº 1/2026 - AUDIN

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 12/05/2026 10:44)
GUILLERMO JAVIER DIAZ VILLAVICENCIO
CHEFE DA AUDITOR(A)IA INTERNA - TITULAR
AUDIN (10.01.05.16)
Matrícula: ###903#1

*Visualize o documento original em https://sig.unila.edu.br/documentos/ informando seu número: **1**, ano: **2026**, tipo: **PROPOSTAS**, data de emissão: **12/05/2026** e o código de verificação: **a2f93dc740***

TERMO DE CESSÃO DE USO E DISPONIBILIZAÇÃO DE CÓDIGO-FONTE DE SOFTWARE

1. Identificação das partes

Instituição: Universidade Federal da Integração Latino-Americana – UNILA
Unidade/Setor: Auditoria Interna (AUDIN)
Servidor Desenvolvedor: ANDRÉ RODRIGUES MATSUMOTO, Economista, SIAPE 2141111, lotado na Auditoria Interna (AUDIN)
Chefia Imediata: GUILLERMO JAVIER DIAZ VILLAVICENCIO, Chefia da AUDIN, SIAPE 2090381, Auditoria Interna (AUDIN)

---

2. Identificação do software

Nome do software: AvalRIG
Linguagem de programação: Python
Interface: Aplicação desktop com interface gráfica (Tkinter)
Bibliotecas principais:

- `tkinter`, `tkinter.ttk`, `tkinter.scrolledtext` (interface gráfica)
- `google-generativeai` (motor de IA – Google Gemini)
- `pymupdf` e/ou `pdfplumber` (extração de texto de arquivos PDF)

Tipo de aplicação:
Ferramenta de apoio à auditoria interna, voltada à avaliação automatizada do Relatório de Gestão (RIG) com base em critérios alinhados às orientações do TCU, utilizando recursos de inteligência artificial (Google Gemini).

Resumo funcional:
O AvalRIG permite carregar relatórios de gestão em formato PDF, extrair o texto do documento, e realizar a avaliação automatizada preliminar de:

- Princípios e respectivos quesitos;
- Capítulos e respectivos elementos de conteúdo,

atribuindo pontuações sugestivas, classificações qualitativas (por ex.: “INSUFICIENTE”, “RAZOÁVEL”, “BOM”, “MUITO BOM”) e gerando fundamentação textual com apoio de IA.

As avaliações geradas constituem apoio à análise técnica do auditor, não se caracterizando como avaliação final. A avaliação definitiva, inclusive quanto à

pontuação atribuída, é sempre de responsabilidade do(a) servidor(a) que opera o software, que poderá acatar, revisar ou rejeitar as sugestões apresentadas.

A ferramenta organiza os resultados em telas específicas, exibe tabelas-resumo, permite gerar relatórios consolidados em texto (para princípios e elementos) e possibilita a exportação dos resultados em arquivos `.txt`, contribuindo para ganho de produtividade, padronização e rastreabilidade das análises da Auditoria Interna.

---

3. Finalidade do documento

O presente termo tem por finalidade:

a) Formalizar a disponibilização, pelo servidor desenvolvedor, do software AvalRIG para uso oficial no âmbito da Auditoria Interna da UNILA;
b) Registrar a concordância da chefia imediata quanto à implementação e utilização do AvalRIG como ferramenta de apoio às atividades de auditoria interna e avaliação do RIG;
c) Estabelecer as condições de uso, manutenção, evolução e divulgação da autoria do software;
d) Registrar que as avaliações geradas pela ferramenta possuem caráter auxiliar e não vinculante, cabendo ao(à) servidor(a) responsável a avaliação final.

---

4. Declaração do desenvolvedor

Eu, ANDRÉ RODRIGUES MATSUMOTO, Economista, SIAPE 2141111, servidor da Universidade Federal da Integração Latino-Americana – UNILA, declaro para os devidos fins:

1. Que sou autor do software denominado “AvalRIG”, desenvolvido em linguagem Python, com interface gráfica em Tkinter e utilização, entre outras, das bibliotecas `google-generativeai`, `pymupdf` e/ou `pdfplumber`, destinado a apoiar as atividades da Auditoria Interna, especialmente a avaliação do Relatório de Gestão (RIG) à luz de critérios adotados pelo TCU.
2. Que o AvalRIG possibilita, em síntese:
    - Carregar arquivos PDF de Relatório de Gestão e extrair o texto;
    - Realizar pré-validação de requisitos (como presença de chave de API e bibliotecas necessárias);
    - Selecionar Princípios e Quesitos ou Capítulos e Elementos de conteúdo;

    - Enviar o texto do relatório para avaliação preliminar por meio da API Google Gemini, obtendo pontuações e fundamentações sugestivas para cada item avaliado;
    - Consolidar os resultados, exibindo tabelas com pontuações sugeridas, resumos por princípio/capítulo e cálculos percentuais;
    - Gerar relatórios textuais detalhados (princípios e elementos), com subtotais, resumo geral e conceito sugerido;
    - Exportar avaliações específicas (quesitos/elementos) e relatórios completos em arquivos `.txt`.
3. Que as pontuações, conceitos e fundamentações produzidos pelo AvalRIG e pela inteligência artificial não substituem o juízo profissional do(a) auditor(a), tendo caráter estritamente auxiliar ao processo de análise.
4. Que a avaliação final, inclusive com a pontuação que será efetivamente considerada, é de responsabilidade do(a) servidor(a) que conduz o trabalho de auditoria e opera o AvalRIG, cabendo-lhe confirmar, ajustar ou desconsiderar as sugestões geradas pela ferramenta.
5. Que concedo à Universidade Federal da Integração Latino-Americana – UNILA o direito de uso livre, não exclusivo e sem ônus do AvalRIG, no âmbito institucional, para fins administrativos e acadêmicos internos, incluindo sua utilização corrente pela Auditoria Interna e demais unidades que venham a se beneficiar da ferramenta.
6. Que autorizo a disponibilização do código-fonte (script em Python) do AvalRIG, incluindo os trechos referentes à interface gráfica, rotinas de extração de texto de PDF, integração com a API Google Gemini, validação de dependências e geração/exportação de relatórios, para registro no processo administrativo correspondente e/ou em repositórios internos da UNILA, permitindo sua consulta, uso, cópia e adaptação por outros servidores da instituição.
7. Que reconheço que o AvalRIG poderá ser, a critério da instituição, aperfeiçoado, adaptado, integrado a outros sistemas ou mesmo reestruturado tecnicamente, preservada a referência à autoria original e o histórico de evolução.
8. Que assumo a responsabilidade pela exatidão das informações técnicas prestadas sobre o funcionamento do AvalRIG e pela correspondência entre o código-fonte disponibilizado e a versão efetivamente testada e utilizada pela unidade na data deste termo.

---

5. Declaração da chefia imediata

Eu, GUILLERMO JAVIER DIAZ VILLAVICENCIO, Chefia da AUDIN, SIAPE 2090381, da Auditoria Interna da UNILA, declaro:

1. Que concordo com a implementação e utilização oficial do software AvalRIG como ferramenta de apoio às atividades da Auditoria Interna, em especial na análise do Relatório de Gestão (RIG) e em simulações de procedimentos de auditoria alinhados às diretrizes do TCU.
2. Que tenho ciência de que as avaliações, pontuações, conceitos e fundamentações produzidos pelo AvalRIG possuem natureza preliminar e auxiliar, sendo insuscetíveis de substituírem a análise crítica e o juízo profissional do(a) auditor(a) responsável.
3. Que reconheço que a avaliação final, inclusive quanto à pontuação definitiva a ser considerada em relatórios, pareceres ou manifestações da Auditoria Interna, é sempre de responsabilidade do(a) servidor(a) que conduz o trabalho e opera o AvalRIG, podendo este aceitar, ajustar ou rejeitar as sugestões apresentadas pela ferramenta.
4. Que reconheço o potencial de ganho de produtividade, padronização e qualidade nas análises, decorrente da automação proporcionada pelo AvalRIG, inclusive quanto à geração de relatórios textuais padronizados e registros consistentes de pontuações e fundamentações.
5. Que tenho ciência da disponibilização do código-fonte pelo servidor desenvolvedor e da autorização para seu uso institucional, comprometendo-me a zelar pela utilização adequada do AvalRIG no âmbito da unidade e a observar as normas de segurança da informação aplicáveis, especialmente no tratamento de documentos oficiais.
6. Que reconheço e apoio o registro da autoria do servidor ANDRÉ RODRIGUES MATSUMOTO em documentos internos, relatórios técnicos, apresentações, registros em currículo e demais meios em que o AvalRIG seja mencionado, como produto de desenvolvimento interno da Auditoria Interna da UNILA.
7. Que manifesto anuência para que este termo seja juntado a processo administrativo próprio, de forma a formalizar a adoção do AvalRIG e a cessão de uso institucional em âmbito da UNILA.

---

6. Condições de uso e manutenção

1. O AvalRIG será utilizado exclusivamente para fins institucionais, no âmbito das competências da UNILA, observando-se a legislação aplicável, as normas internas, as políticas de segurança da informação e as condições de uso das APIs/bibliotecas externas (como Google Gemini, `pymupdf`, `pdfplumber` etc.).
2. Caberá à unidade usuária:
    - Designar responsáveis pela operação do AvalRIG e pelo gerenciamento de chaves de API eventualmente utilizadas (por

exemplo, chave da API Gemini), garantindo seu uso adequado e sigiloso;
    - Zelar pela guarda dos arquivos, dados e relatórios produzidos;
    - Comunicar ao desenvolvedor eventuais problemas relevantes identificados durante o uso, sempre que possível.
3. Caberá ao desenvolvedor, na medida de suas possibilidades e sem prejuízo de suas atribuições funcionais, prestar esclarecimentos técnicos básicos sobre o funcionamento do AvalRIG e, quando possível, colaborar em ajustes e melhorias, não se constituindo, porém, obrigação de suporte permanente ou exclusivo.
4. A UNILA poderá promover, diretamente ou por intermédio de suas unidades, ajustes, correções, melhorias, atualizações ou integrações do AvalRIG com outros sistemas institucionais, preservando a referência ao desenvolvimento original e à autoria do servidor.

---

7. Propriedade intelectual e autoria

1. A autoria do desenvolvimento do software AvalRIG é atribuída ao servidor ANDRÉ RODRIGUES MATSUMOTO, Economista, SIAPE 2141111, para fins de registro em currículo, avaliações de desempenho, relatórios de gestão, publicações técnicas, bem como em quaisquer documentos internos ou externos em que o sistema seja mencionado.
2. A instituição poderá mencionar, em apresentações, relatórios e demais meios institucionais, que o AvalRIG foi desenvolvido internamente na Auditoria Interna da UNILA, destacando a autoria de ANDRÉ RODRIGUES MATSUMOTO.

---

8. Anexos

Passa a integrar o presente termo, para todos os fins:

- Anexo I – Código-fonte ou indicação de repositório interno;
- Anexo II - Evidências de testes e uso experimental na Auditoria Interna.

---

9. Vigência

O presente termo entra em vigor na data de sua assinatura digital pelas partes e permanecerá válido enquanto o AvalRIG estiver em uso na instituição, podendo ser revisado ou atualizado mediante manifestação da unidade e/ou do desenvolvedor, por meio de novo registro em processo administrativo.

TERMO Nº 2/2026 - AUDIN

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 12/05/2026 12:04)
ANDRE RODRIGUES MATSUMOTO
ECONOMISTA
AUDIN (10.01.05.16)
Matrícula: ###411#1

(Assinado digitalmente em 12/05/2026 10:43)
GUILLERMO JAVIER DIAZ VILLAVICENCIO
CHEFE DA AUDITOR(A)IA INTERNA - TITULAR
AUDIN (10.01.05.16)
Matrícula: ###903#1

*Visualize o documento original em https://sig.unila.edu.br/documentos/ informando seu número: **2**, ano: **2026**, tipo: **TERMO**, data de emissão: **12/05/2026** e o código de verificação: **1ab01b7d33***

Anexo I – Código-fonte ou indicação de repositório interno

```
import sys
import os
import json
import threading
from pathlib import Path


# GUI
import tkinter as tk
from tkinter import ttk, filedialog, messagebox,
scrolledtext


# PDF reading
try:
    import fitz  # PyMuPDF
    PYMUPDF_AVAILABLE = True
except ImportError:
    PYMUPDF_AVAILABLE = False


try:
    import pdfplumber
    PDFPLUMBER_AVAILABLE = True
```

```
except ImportError:
    PDFPLUMBER_AVAILABLE = False

# AI - Google Gemini
try:
    import google.generativeai as genai
    GEMINI_AVAILABLE = True
except ImportError:
    GEMINI_AVAILABLE = False

# ═══════════════════════════════════════════════
# DADOS DOS PRINCÍPIOS E QUESITOS
# ═══════════════════════════════════════════════

PRINCIPIOS = {
    1: {
        "titulo": "1. Foco Estratégico e no Cidadão",
        "descricao": (
            "Além de prestar contas sobre os fatos pretéritos, os responsáveis devem apresentar "
            "a direção estratégica da organização na busca de resultados para a sociedade, "
            "proporcionando uma visão de como a estratégia se relaciona com a capacidade de gerar "
```

```
            "valor público no curto, médio e longo
prazos e demonstrar o uso que a UPC faz dos "

            "recursos, bem como os produtos, os
resultados e os impactos produzidos."

        ),

        "quesitos": {

            "a": "Os objetivos estratégicos estão
claramente apresentados e compreensíveis.",

            "b": "Há contextualização dos objetivos
estratégicos em um quadro de geração de valor a
curto, médio e longo prazos.",

            "c": "Há considerações sobre riscos e
oportunidades claramente associadas à definição dos
objetivos estratégicos e à sustentabilidade dos
capitais.",

            "d": "Há uma demonstração de como os
recursos ou capitais são utilizados e combinados para
produção de bens e serviços (modelo de negócios).",

            "e": "A estratégia considera medidas
atuais e futuras de manutenção, regeneração,
priorização ou aprimoramento dos capitais.",

            "f": "Há clareza na apresentação dos
principais processos produtivos, bem como dos
capitais e dos bens e serviços a serem produzidos.",

            "g": "Os principais processos produtivos
estão associados a objetivos táticos e
operacionais.",

            "h": "Os principais processos produtivos
possuem indicadores e metas associados.",

        },
```

```
    },
    2: {
        "titulo": "2. Conectividade da Informação",
        "descricao": (
            "As informações devem mostrar uma visão integrada da inter-relação entre os resultados "
            "alcançados, a estratégia de alocação dos recursos e os objetivos estratégicos definidos "
            "para o exercício; e da inter-relação e da dependência entre os fatores que afetam a "
            "capacidade de a UPC alcançar os seus objetivos ao longo do tempo."
        ),
        "quesitos": {
            "a": "Os objetivos estratégicos estão justificados em função da missão da organização.",
            "b": "A estratégia do exercício corrente leva em consideração uma avaliação dos resultados de exercícios anteriores.",
            "c": "A organização demonstra como seus objetivos estão associados ao atendimento dos legítimos interesses e necessidades das principais partes interessadas.",
            "d": "Há um desdobramento claro dos objetivos estratégicos em objetivos táticos e operacionais.",
            "e": "Há clara demonstração de bens e serviços efetivamente produzidos, associada aos
```

```
principais processos produtivos, aos objetivos
operacionais e a indicadores e metas.",

            "f": "São demonstrados os custos e
capitais efetivamente utilizados na produção de bens
e serviços dentro dos principais processos
produtivos.",

            "g": "Está demonstrado o valor público
gerado em função de bens e serviços produzidos,
inclusive por meio de metas alcançadas.",

            "h": "Há uma demonstração da relação entre
indicadores financeiros e resultados e valor
gerado.",

        },

    },

    3: {

        "titulo": "3. Relações com Partes
Interessadas",

        "descricao": (

            "As informações devem prover uma visão da
natureza e da qualidade das relações que a UPC "

            "mantém com suas principais partes
interessadas, incluindo como e até que ponto a UPC "

            "entende, leva em conta e responde aos
seus legítimos interesses e necessidades."

        ),

        "quesitos": {

            "a": "Foram apresentadas ações
direcionadas à identificação das partes interessadas
```

```
e de suas necessidades, de acordo com a missão da
UPC.",

            "b": "A apresentação dos objetivos
estratégicos faz menção a mecanismos de identificação
das necessidades das partes interessadas.",

            "c": "Foram informados instrumentos ou
indicadores de avaliação da satisfação das partes
interessadas e de como eles retroalimentam a
estratégia.",

            "d": "Foram apresentadas informações de
como a UPC avalia o valor gerado em relação a sua
missão e ao valor e sustentabilidade da própria
organização.",

        },

    },

    4: {

        "titulo": "4. Materialidade",

        "descricao": (

            "Devem ser divulgadas informações sobre
assuntos que afetam, de maneira significativa, "

            "a capacidade de a UPC alcançar seus
objetivos de geração de valor público no curto, "

            "médio e longo prazos e com conteúdo
relevante para a sociedade."

        ),

        "quesitos": {

            "a": "Há informações acerca dos critérios
de seleção de conteúdo em razão da materialidade.",
```

```
            "b": "As informações apresentadas estão
situadas ou são agregadas numa mesma faixa de
materialidade ou em faixas relevantes com a estrutura
organizacional.",

            "c": "É possível identificar os principais
processos e os principais recursos da UPC que
contribuem para seu resultado geral.",

            "d": "É possível identificar os principais
produtos e de que forma eles contribuem para o valor
gerado e o resultado geral.",

        },

    },

    5: {

        "titulo": "5. Concisão",

        "descricao": (

            "Os textos não devem ser mais extensos do
que o necessário para transmitir a mensagem "

            "e fundamentar as conclusões."

        ),

        "quesitos": {

            "a": "O texto é suficiente para a
compreensão da mensagem.",

            "b": "Existem links para informações
complementares de forma adequada apenas para
complementar a informação.",

            "c": "A disposição do texto facilita a
leitura e os infográficos estão situados em pontos
adequados do conteúdo.",
```

```
            "d": "A quantidade e o volume das
informações apresentadas está em volume adequado e
sem excessos ou itens por demais detalhados.",

        },

    },

    6: {

        "titulo": "6. Confiabilidade e Completude",

        "descricao": (

            "Devem ser abrangidos todos os temas
materiais, positivos e negativos, de maneira "

            "equilibrada e isenta de erros
significativos, de modo a evitar equívocos ou vieses
"

            "no processo decisório dos usuários das
informações."

        ),

        "quesitos": {

            "a": "O relatório contém uma explicação do
todo da gestão que permita a compreensão de suas
partes e de como se conectam.",

            "b": "O relatório possui uma explicação do
todo de seu conteúdo que facilite a compreensão de
suas partes e de como estão encadeadas.",

            "c": "Há informações sobre os processos de
obtenção e produção das informações e dados que são
apresentados.",

            "d": "Há informações de como podem ser
obtidos dados completos e mais detalhados da gestão
```

```
que não estão apresentados especificamente no
relatório.",
        },
    },
    7: {
        "titulo": "7. Coerência e Comparabilidade",
        "descricao": (
            "As informações devem ser apresentadas em
bases coerentes ao longo do tempo, de maneira "
            "a permitir acompanhamento de séries
históricas da UPC e comparação com outras unidades "
            "de natureza similar."
        ),
        "quesitos": {
            "a": "Os objetivos estratégicos e suas
metas possuem alguma contextualização com cenário
externo, com ameaças e oportunidades.",
            "b": "Os resultados alcançados estão
contextualizados no cenário externo e nos resultados
de outras organizações do mesmo setor ou ramo de
negócio.",
            "c": "Os resultados alcançados no
exercício são comparados com os de outros
exercícios.",
            "d": "Os custos e os recursos alocados são
comparados com os de outros exercícios.",
        },
```

```
    },
    8: {
        "titulo": "8. Clareza",
        "descricao": (
            "Deve ser utilizada linguagem simples e
imagens visuais eficazes para transformar "
            "informações complexas em relatórios
facilmente compreensíveis, além de fazer uma "
            "distinção inequívoca entre os problemas
enfrentados e os resultados alcançados pela "
            "UPC no exercício e aqueles previstos para
o futuro."
        ),
        "quesitos": {
            "a": "O relatório possui uma estrutura
clara e objetiva.",
            "b": "As informações são apresentadas de
forma simples e objetiva.",
            "c": "A sequência de informações é
logicamente adequada e alinhada com a estrutura do
RI.",
            "d": "Há instrumentos visuais e links bem
posicionados, claros e de fácil compreensão.",
        },
    },
    9: {
        "titulo": "9. Tempestividade",
```

```
        "descricao": (
            "As informações devem estar disponíveis em tempo hábil para suportar os processos de "
            "transparência, responsabilização e tomada de decisão por parte dos cidadãos e seus "
            "representantes, dos usuários de serviços públicos e dos provedores de recursos."
        ),
        "quesitos": {
            "a": "Os setores envolvidos providenciam as informações em tempo hábil à elaboração do relatório de gestão.",
            "b": "A equipe que elabora o relatório recebe as informações em tempo hábil à elaboração do relatório.",
            "c": "O escopo proposto para o relatório de gestão contribui para a tempestividade em sua elaboração.",
            "d": "A compreensão das informações demandadas para o relatório ocorre de modo fácil e rápido.",
        },
    },
    10: {
        "titulo": "10. Transparência",
        "descricao": (
            "Deve ser realizada a comunicação aberta, voluntária e transparente das atividades e "
```

```
            "dos resultados da organização e a
divulgação de informações de interesse coletivo ou "
            "geral, independente de requerimento."
        ),
        "quesitos": {
            "a": "A informação contida no relatório é
compreensível para o público em geral.",
            "b": "A forma de apresentar as informações
mostra-se adequada ao público em geral.",
            "c": "O relatório indica o porquê de as
informações apresentadas terem sido as escolhidas
para constar no documento.",
            "d": "O relatório é facilmente encontrado
e foi devidamente divulgado interna e externamente.",
        },
    },
}


# ──────────────────────────────────────────────────
# DADOS DOS CAPÍTULOS E ELEMENTOS DE CONTEÚDO
# ──────────────────────────────────────────────────


CAPITULOS = {
    1: {
        "titulo": "1. Mensagem do dirigente máximo da
unidade",
```

```
        "descricao": (
            "A mensagem do dirigente máximo deve
apresentar os principais resultados e reconhecer "
            "a responsabilidade pela integridade do
relatório de gestão."
        ),
        "elementos": {
            1: (
                "Apresentação, em forma de tabelas e
gráficos, dos principais resultados alcançados, "
                "incluindo aqueles que indiquem o grau
de alcance das metas fixadas nos planos da "
                "organização, considerando os
objetivos estratégicos e de curto prazo, bem como as
"
                "prioridades da gestão [UPC em
números], que estão mais bem detalhados no corpo do
relatório."
            ),
            2: (
                "A mensagem do dirigente deve conter o
reconhecimento de sua responsabilidade por "
                "assegurar a integridade
(fidedignidade, precisão e completude) do relatório
de gestão, "
                "ou deve conter as ressalvas quanto a
esse aspecto e as medidas adotadas para a correção "
                "dos problemas."
```

```
            ),
        },
    },
    2: {
        "titulo": "2. Visão geral organizacional e
ambiente externo",
        "descricao": (
            "(1) O que é a organização, o que faz e
quais são as circunstâncias em que atua? "
            "(2) Qual o modelo de negócios da
organização? "
            "(3) Como a organização determina os temas
a serem incluídos no relatório de gestão "
            "e como estes temas são quantificados ou
avaliados?"
        ),
        "elementos": {
            3: "Identificação da UPC e declaração da
sua missão e visão.",
            4: "Principais normas direcionadoras de
sua atuação, com links de acesso respectivos.",
            5: (
                "Organograma da estrutura
organizacional, incluindo as estruturas de governança
"
                "(conselhos ou comitês de governança,
entre outros)."
```

```
            ),
            6: (
                "Apresentação do modelo de negócios da
UPC, abrangendo insumos, atividades, produtos, "
                "impactos, valor gerado e seus
destinatários e diagrama de cadeia de valor, visando
"
                "proporcionar compreensão abrangente
da visão geral organizacional."
            ),
            7: (
                "Se for o caso, a relação de políticas
e programas de governo/ações orçamentárias, "
                "bem como de programas do Plano
Plurianual, de outros planos nacionais, setoriais e "
                "transversais de governo nos quais
atua, com seus respectivos objetivos e metas."
            ),
            8: (
                "Informações sobre contratos de gestão
firmados e de que forma são integrados no valor "
                "gerado pela unidade."
            ),
            9: (
                "Relação com o ambiente externo e com
os destinatários dos bens e serviços produzidos "
                "pela organização."
```

```
            ),
            10: "Capital social e participação em
outras sociedades, se aplicável.",
        },
    },
    3: {
        "titulo": "3. Riscos, oportunidades e
perspectivas",
        "descricao": (
            "(1) Quais são os riscos e oportunidades
específicos que afetam a capacidade de a "
            "organização gerar valor em curto, médio e
longo prazo e como a organização lida com "
            "esses riscos? (2) Quais os desafios e as
incertezas que a organização provavelmente "
            "enfrentará ao buscar executar seu plano
estratégico e as potenciais implicações para "
            "seu modelo de negócio e desempenho
futuro?"
        ),
        "elementos": {
            11: (
                "Principais riscos identificados que
podem afetar a capacidade de a UPC alcançar "
                "seus objetivos e como a UPC lida com
essas questões."
            ),
```

```
            12: (
                "Quais são as principais oportunidades
identificadas que podem aumentar a capacidade "
                "de a UPC atingir seus objetivos e as
respectivas ações para aproveitá-las."
            ),
            13: (
                "As fontes específicas de riscos e
oportunidades, que podem ser internas, externas "
                "ou, normalmente, uma combinação das
duas."
            ),
            14: (
                "Avaliação, pela UPC, da probabilidade
de que o risco ou a oportunidade ocorram e a "
                "magnitude de seu efeito, caso isso
aconteça, levando em consideração, inclusive, as "
                "circunstâncias específicas que
levariam à ocorrência do risco ou da oportunidade."
            ),
        },
    },
    4: {
        "titulo": "4. Governança, estratégia e
desempenho",
        "descricao": (
```

```
            "(1) Para onde a organização deseja ir e
como ela pretende chegar lá? "
            "(2) Como a estrutura de governança da
organização apoia sua capacidade de gerar valor "
            "em curto, médio e longo prazo? "
            "(3) Quais os principais resultados
alcançados e até que ponto a organização alcançou "
            "seus objetivos no exercício?"
        ),
        "elementos": {
            15: (
                "Descrição de como a estrutura de
governança apoia o cumprimento dos objetivos "
                "estratégicos, abordando o
relacionamento com a sociedade e as partes
interessadas "
                "da organização, bem como a
consideração de suas necessidades e expectativas na "
                "definição da estratégia, a gestão de
riscos e a supervisão da gestão."
            ),
            16: (
                "Objetivos estratégicos, responsáveis,
indicadores de desempenho, com as metas "
                "pactuadas para o período e seu
desdobramento anual, bem como sua vinculação ao "
```

```
                "Plano Plurianual, aos planos
nacionais e setoriais do governo e dos órgãos de "
                "governança superior, indicando os
resultados já alcançados, comparando-os com as "
                "metas e os objetivos pactuados."
            ),
            17: (
                "Planos de curto prazo da organização
com a indicação dos objetivos anuais, das "
                "medidas, iniciativas, projetos e
programas necessários ao seu alcance, dos prazos, "
                "dos responsáveis, das metas para o
período a que se refere o relatório de gestão, "
                "e os resultados alcançados
comparando-os com as metas e os objetivos pactuados."
            ),
            18: (
                "Apresentação resumida dos resultados
das principais áreas de atuação e/ou de "
                "operação/atividades da UPC e dos
principais programas, projetos e iniciativas, "
                "abrangendo ainda, conforme o caso, a
contribuição de autarquias e fundações "
                "vinculadas e de empresas controladas,
contratos de gestão e SPEs, conforme a "
                "materialidade da contribuição dos
segmentos na composição do valor gerado pela UPC."
```

```
            ),
            19: (
                "Medidas adotadas em relação aos
indicadores de governança e gestão levantados, a "
                "exemplo dos que foram tratados pelo
TCU nos Acórdãos 588/2018-Plenário e "
                "2.699/2018-Plenário (ambos da
Relatoria do Ministro Bruno Dantas)."
            ),
            20: (
                "Principais ações de supervisão,
controle e de correição adotadas pela UPC para a "
                "garantia da legalidade, legitimidade,
economicidade e transparência na aplicação "
                "dos recursos públicos."
            ),
        },
    },
    5: {
        "titulo": "5. Informações orçamentárias,
financeiras e contábeis",
        "descricao": (
            "Quais são as principais informações
orçamentárias, financeiras e contábeis, inclusive "
            "de custos, que dão suporte às informações
sobre o desempenho da organização no período? "
```

```
            "(A evidenciação deve contemplar as
principais unidades que compõem a UPC, de forma "
            "individual e agrupada, de acordo com a
materialidade e a relevância para os resultados "
            "do conjunto)."
        ),
        "elementos": {
            21: (
                "Resumo da situação financeira
contábil da UPC (saldos das principais contas e/ou "
                "grupos de contas, resultados,
receitas e despesas) e da evolução no exercício de "
                "referência e em comparação com o
último exercício."
            ),
            22: (
                "As contas relativas aos fundos de
financiamento devem apresentar informações sobre "
                "o patrimônio global e os resultados
das operações de crédito realizadas à conta "
                "desses recursos em face dos objetivos
estabelecidos."
            ),
            23: (
                "Principais fatos contábeis, contas ou
grupos de contas, saldos e ocorrências "
```

```
                "relativos à atuação e à situação
financeira da UPC no exercício."
            ),
            24: (
                "Conclusões de auditorias
independentes e/ou dos órgãos de controle público e
as "
                "medidas adotadas em relação a
conclusões ou eventuais apontamentos."
            ),
            25: (
                "Indicações de locais ou endereços
eletrônicos em que demonstrações contábeis e "
                "notas explicativas estão publicadas
e/ou podem ser acessadas em sua íntegra."
            ),
            26: (
                "Esclarecimentos acerca da forma como
foram tratadas as demonstrações contábeis em "
                "caso de a UPC possuir em sua
composição mais de uma entidade contábil,
considerando "
                "que: as UPC que compreenderem apenas
um órgão no Siafi devem considerar os valores "
                "contábeis consolidados nesse órgão;
as UPC que compreenderem mais de um órgão no "
                "Siafi devem apresentar informações
referentes aos principais dados desses órgãos de "
```

```
                "forma individualizada e, ainda, devem
apresentar informações com base na integração "
                "dos principais saldos para efeito de
associação com o resultado geral da UPC; as UPC "
                "que não atuam no Siafi devem adotar
procedimento semelhante."
            ),
        },
    },
}


SCORES = {0: "0 - Insuficiente", 1: "1 - Razoável",
2: "2 - Bom", 3: "3 - Muito Bom"}
SCORE_COLORS = {
    "0 - Insuficiente": "#e74c3c",
    "1 - Razoável":     "#e67e22",
    "2 - Bom":          "#2980b9",
    "3 - Muito Bom":    "#27ae60",
}


# Escala específica para elementos de conteúdo
SCORES_ELEMENTOS = {
    0: "0 - Item faltando no RIG",
    1: "1 - Consta, mas insatisfatório",
    2: "2 - Consta, medianamente satisfatório",
```

```
    3: "3 - Consta de modo satisfatório",
}
SCORE_COLORS_ELEMENTOS = {
    "0 - Item faltando no RIG":            "#e74c3c",
    "1 - Consta, mas insatisfatório":      "#e67e22",
    "2 - Consta, medianamente satisfatório":
"#2980b9",
    "3 - Consta de modo satisfatório":     "#27ae60",
}


MODELOS_GEMINI = [
    "gemini-3.1-pro",
    "gemini-2.5-pro",
    "gemini-2.0-flash",
    "gemini-2.5-flash",
    "gemini-1.5-flash",
]


# ─────────────────────────────────────────────────
# EXTRAÇÃO DE TEXTO DO PDF
# ─────────────────────────────────────────────────


def extrair_texto_pdf(caminho: str) -> str:
```

```
    texto = ""
    if PYMUPDF_AVAILABLE:
        try:
            doc = fitz.open(caminho)
            for page in doc:
                texto += page.get_text()
            doc.close()
            if texto.strip():
                return texto
        except Exception:
            pass
    if PDFPLUMBER_AVAILABLE:
        try:
            with pdfplumber.open(caminho) as pdf:
                for page in pdf.pages:
                    t = page.extract_text()
                    if t:
                        texto += t + "\n"
            if texto.strip():
                return texto
        except Exception:
            pass
    raise RuntimeError(
```

```
        "Não foi possível extrair texto do PDF.\n"
        "Instale PyMuPDF (pip install pymupdf) ou
pdfplumber (pip install pdfplumber)."
    )


def truncar_texto(texto: str, max_chars: int = 28000)
-> str:
    if len(texto) <= max_chars:
        return texto
    return texto[:max_chars] + "\n\n[... texto
truncado por limite de contexto ...]"


# ─────────────────────────────────────────────
# AVALIAÇÃO VIA GEMINI – PRINCÍPIOS
# ─────────────────────────────────────────────


def avaliar_quesito(
    api_key: str,
    texto_relatorio: str,
    num_principio: int,
    letra_quesito: str,
    modelo: str = "gemini-3.1-pro",
```

```
) -> dict:
    principio = PRINCIPIOS[num_principio]
    quesito   = principio["quesitos"][letra_quesito]

    genai.configure(api_key=api_key)

    system_instruction = (
        "Você é um auditor sênior do Tribunal de
Contas da União (TCU), especializado em "
        "avaliação de Relatórios de Gestão de unidades
da administração pública federal. "
        "Sua função é avaliar com rigor técnico,
imparcialidade e embasamento normativo "
        "se o Relatório de Gestão da UNILA atende aos
princípios estabelecidos para "
        "elaboração de Relatórios Integrados de
Gestão.\n\n"
        "Responda SEMPRE em JSON válido com exatamente
duas chaves:\n"
        "  'pontuacao': inteiro de 0 a 3\n"
        "  'argumento': string com fundamentação
técnica detalhada em português\n\n"
        "Escala de pontuação:\n"
        "  0 = Insuficiente: o quesito não está
contemplado ou está ausente no relatório.\n"
        "  1 = Razoável: o quesito é parcialmente
atendido, mas com lacunas relevantes.\n"
```

```
        "  2 = Bom: o quesito é atendido de forma satisfatória, com pequenas omissões.\n"
        "  3 = Muito Bom: o quesito é plenamente atendido com informações claras e completas.\n\n"
        "O argumento técnico deve:\n"
        "- Citar evidências concretas encontradas (ou ausentes) no relatório;\n"
        "- Relacionar a avaliação ao princípio e ao quesito específico;\n"
        "- Ter tom formal e técnico, como um parecer do TCU;\n"
        "- Ter entre 80 e 200 palavras.\n"
        "- NÃO incluir markdown, apenas texto puro dentro do JSON."
    )

    user_prompt = (
        f"PRINCÍPIO AVALIADO: {principio['titulo']}\n"
        f"Descrição do princípio: {principio['descricao']}\n\n"
        f"QUESITO {letra_quesito.upper()}: {quesito}\n\n"
        f"TRECHO DO RELATÓRIO DE GESTÃO DA UNILA:\n"
        f"{truncar_texto(texto_relatorio, 28000)}\n\n"
        "Com base no trecho acima, avalie o quesito informado e retorne JSON com "
```

```
        "'pontuacao' (0-3) e 'argumento'
(fundamentação técnica)."
    )

    model = genai.GenerativeModel(
        model_name=modelo,
        system_instruction=system_instruction,
generation_config=genai.types.GenerationConfig(
            temperature=0.2,
            response_mime_type="application/json",
        ),
    )

    response = model.generate_content(user_prompt)
    raw = response.text.strip()

    if raw.startswith("```"):
        raw = raw.split("```")[1]
        if raw.startswith("json"):
            raw = raw[4:]
    raw = raw.strip()

    data       = json.loads(raw)
```

```
    pontuacao = int(data.get("pontuacao", 0))
    pontuacao = max(0, min(3, pontuacao))
    argumento = data.get("argumento", "Sem argumento
retornado pela IA.")

    return {"pontuacao": pontuacao, "argumento":
argumento}



# ──────────────────────────────────────────────────
# AVALIAÇÃO VIA GEMINI – ELEMENTOS DE CONTEÚDO
# ──────────────────────────────────────────────────

def avaliar_elemento(
    api_key: str,
    texto_relatorio: str,
    num_capitulo: int,
    num_elemento: int,
    modelo: str = "gemini-3.1-pro",
) -> dict:
    capitulo       = CAPITULOS[num_capitulo]
    descricao_elem =
capitulo["elementos"][num_elemento]
```

```
    genai.configure(api_key=api_key)


    system_instruction = (
        "Você é um auditor sênior do Tribunal de
Contas da União (TCU), especializado em "
        "avaliação de Relatórios Integrados de Gestão
de unidades da administração pública federal. "
        "Sua função é verificar se os elementos de
conteúdo obrigatórios estão presentes e "
        "adequadamente tratados no Relatório Integrado
de Gestão (RIG) da UNILA.\n\n"
        "Responda SEMPRE em JSON válido com exatamente
duas chaves:\n"
        "  'pontuacao': inteiro de 0 a 3\n"
        "  'argumento': string com fundamentação
técnica detalhada em português\n\n"
        "Escala de pontuação para elementos de
conteúdo:\n"
        "  0 = Item de informação está FALTANDO no RIG
(ausente ou inexistente).\n"
        "  1 = Item consta no RIG, mas de modo
INSATISFATÓRIO (lacunas graves, superficial).\n"
        "  2 = Item consta no RIG, mas de modo
MEDIANAMENTE SATISFATÓRIO (atende parcialmente).\n"
        "  3 = Item consta no RIG de modo SATISFATÓRIO
(atende plenamente ao esperado).\n\n"
        "O argumento técnico deve:\n"
```

```
        "- Citar evidências concretas encontradas (ou
ausentes) no relatório;\n"
        "- Relacionar a avaliação ao capítulo e ao
elemento específico;\n"
        "- Ter tom formal e técnico, como um parecer
do TCU;\n"
        "- Ter entre 80 e 200 palavras.\n"
        "- NÃO incluir markdown, apenas texto puro
dentro do JSON."
    )

    user_prompt = (
        f"CAPÍTULO AVALIADO: {capitulo['titulo']}\n"
        f"Descrição do capítulo:
{capitulo['descricao']}\n\n"
        f"ELEMENTO {num_elemento}:
{descricao_elem}\n\n"
        f"TRECHO DO RELATÓRIO INTEGRADO DE GESTÃO DA
UNILA:\n"
        f"{truncar_texto(texto_relatorio, 28000)}\n\n"
        "Com base no trecho acima, avalie se o
elemento de conteúdo informado está presente "
        "e adequadamente tratado no RIG. Retorne JSON
com 'pontuacao' (0-3) e 'argumento' "
        "(fundamentação técnica)."
    )
```

```
    model = genai.GenerativeModel(
        model_name=modelo,
        system_instruction=system_instruction,
generation_config=genai.types.GenerationConfig(
            temperature=0.2,
            response_mime_type="application/json",
        ),
    )

    response = model.generate_content(user_prompt)
    raw = response.text.strip()

    if raw.startswith("```"):
        raw = raw.split("```")[1]
        if raw.startswith("json"):
            raw = raw[4:]
    raw = raw.strip()

    data       = json.loads(raw)
    pontuacao = int(data.get("pontuacao", 0))
    pontuacao = max(0, min(3, pontuacao))
    argumento = data.get("argumento", "Sem argumento
retornado pela IA.")
```

```
    return {"pontuacao": pontuacao, "argumento":
argumento}




# ──────────────────────────────────────────────────

# INTERFACE GRÁFICA – APLICAÇÃO PRINCIPAL

# ──────────────────────────────────────────────────


class AuditApp(tk.Tk):

    def __init__(self):

        super().__init__()

        self.title("Avaliação do Relatório Integrado
de Gestão UNILA – Auditor TCU (Gemini IA)")

        self.geometry("1150x860")

        self.minsize(950, 720)

        self.configure(bg="#1e2a38")


        self.pdf_path    = tk.StringVar()

        self.api_key_var =
tk.StringVar(value=os.getenv("GEMINI_API_KEY", ""))

        self.modelo_var  =
tk.StringVar(value=MODELOS_GEMINI[0])

        self.texto_pdf   = ""
```

```
        # Resultados separados por módulo
        self.resultados_principios: dict = {}   #
(num_principio, letra) -> {pontuacao, argumento}
        self.resultados_elementos:  dict = {}   #
(num_capitulo, num_elemento) -> {pontuacao,
argumento}


        self._build_ui()


    # ── Construção da UI principal
─────────────────────────────────


    def _build_ui(self):
        # Cabeçalho
        header = tk.Frame(self, bg="#152232", pady=10)
        header.pack(fill="x")
        tk.Label(
            header,
            text="🏛  Ferramenta de Apoio - Avaliador
RIG UNILA",
            font=("Helvetica", 16, "bold"),
            bg="#152232", fg="#ecf0f1",
        ).pack()
        tk.Label(
            header,
```

```
            text="Simulação de auditoria - Princípios
e Elementos de Conteúdo  |  Powered by Google Gemini
| Version 1.0",

            font=("Helvetica", 9),

            bg="#152232", fg="#95a5a6",

        ).pack()


        # Painel de configurações e PDF
(compartilhado)

        self._build_config_panel()


        # Notebook de módulos

        mod_nb = ttk.Notebook(self)

        mod_nb.pack(fill="both", expand=True, padx=12,
pady=(0, 4))


        self.tab_principios = tk.Frame(mod_nb,
bg="#1e2a38")

        self.tab_elementos  = tk.Frame(mod_nb,
bg="#1e2a38")


        mod_nb.add(self.tab_principios, text="  📐
Módulo 1 - Princípios do RIG  ")

        mod_nb.add(self.tab_elementos,  text="  📋
Módulo 2 - Elementos de Conteúdo  ")
```

```
self._build_modulo_principios(self.tab_principios)

self._build_modulo_elementos(self.tab_elementos)


        # Barra de status

        self.status_var = tk.StringVar(value="Pronto.
Carregue um PDF e configure a Gemini API Key.")

        tk.Label(

            self,

            textvariable=self.status_var,

            bg="#0d1b2a", fg="#bdc3c7",

            anchor="w", padx=10,

            font=("Helvetica", 9),

        ).pack(fill="x", side="bottom")


    def _build_config_panel(self):

        """Painel superior compartilhado: API Key,
Modelo e PDF."""

        panel = tk.Frame(self, bg="#0d1b2a", pady=6)

        panel.pack(fill="x", padx=12, pady=(6, 0))


        # Linha 1: API Key e Modelo

        row1 = tk.Frame(panel, bg="#0d1b2a")

        row1.pack(fill="x", pady=(0, 4))
```

```
        tk.Label(row1, text="Gemini API Key:",
bg="#0d1b2a", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(side="left", padx=(6, 4))


        self.api_entry = tk.Entry(
            row1, textvariable=self.api_key_var,
show="*",
            bg="#2c3e50", fg="#ecf0f1",
insertbackground="white",
            relief="flat", font=("Helvetica", 9),
width=42,
        )
        self.api_entry.pack(side="left", ipady=4)
        tk.Button(
            row1, text="👁", bg="#2c3e50",
fg="#ecf0f1",
            relief="flat", cursor="hand2",
            command=self._toggle_api_key,
        ).pack(side="left", padx=(2, 20))


        tk.Label(row1, text="Modelo Gemini:",
bg="#0d1b2a", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(side="left", padx=(0, 4))
```

```
        ttk.Combobox(
            row1, textvariable=self.modelo_var,
            values=MODELOS_GEMINI, state="readonly",
            font=("Helvetica", 9), width=22,
        ).pack(side="left")

        # Linha 2: PDF
        row2 = tk.Frame(panel, bg="#0d1b2a")
        row2.pack(fill="x")

        tk.Label(row2, text="Documento PDF:",
bg="#0d1b2a", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(side="left", padx=(6, 4))

        self.pdf_entry = tk.Entry(
            row2, textvariable=self.pdf_path,
            bg="#2c3e50", fg="#ecf0f1",
insertbackground="white",
            relief="flat", font=("Helvetica", 9),
state="readonly", width=55,
        )
        self.pdf_entry.pack(side="left", ipady=4)

        tk.Button(
```

```
            row2, text="Carregar PDF",
            bg="#2980b9", fg="white", relief="flat",
            cursor="hand2", font=("Helvetica", 9,
"bold"),
            padx=8, command=self._carregar_pdf,
        ).pack(side="left", padx=(6, 12))

        self.pdf_info_label = tk.Label(
            row2, text="Nenhum documento carregado.",
            bg="#0d1b2a", fg="#7f8c8d",
font=("Helvetica", 8, "italic"),
        )
        self.pdf_info_label.pack(side="left")

        self.progress = ttk.Progressbar(panel,
mode="indeterminate")
        self.progress.pack(fill="x", padx=6, pady=(4,
0))

    # ── MÓDULO 1 - PRINCÍPIOS
──────────────────────────────────

    def _build_modulo_principios(self, parent):
        main = tk.Frame(parent, bg="#1e2a38")
```

```
        main.pack(fill="both", expand=True, padx=8,
pady=8)


        left = tk.Frame(main, bg="#1e2a38", width=360)
        left.pack(side="left", fill="y", padx=(0, 8))
        left.pack_propagate(False)


        right = tk.Frame(main, bg="#1e2a38")
        right.pack(side="left", fill="both",
expand=True)


        self._build_principios_left(left)
        self._build_principios_right(right)


    def _build_principios_left(self, parent):
        self._section_label(parent, "🔎  Seleção para
Avaliação")


        tk.Label(parent, text="Princípio:",
bg="#1e2a38", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(anchor="w", padx=6, pady=(4, 0))


        principio_opts = [f"{k}. {v['titulo'].split('.
', 1)[1]}" for k, v in PRINCIPIOS.items()]
```

```
        self.principio_var = tk.StringVar()

        self.principio_cb  = ttk.Combobox(

            parent, textvariable=self.principio_var,

            values=principio_opts, state="readonly",
font=("Helvetica", 9),

        )

        self.principio_cb.pack(fill="x", padx=6,
pady=(2, 4))

        self.principio_cb.bind("<<ComboboxSelected>>",
self._on_principio_selected)


        tk.Label(parent, text="Quesito:",
bg="#1e2a38", fg="#bdc3c7",

                 font=("Helvetica",
9)).pack(anchor="w", padx=6, pady=(4, 0))

        self.quesito_var = tk.StringVar()

        self.quesito_cb  = ttk.Combobox(

            parent, textvariable=self.quesito_var,

            values=[], state="readonly",
font=("Helvetica", 9),

        )

        self.quesito_cb.pack(fill="x", padx=6,
pady=(2, 8))


        tk.Label(parent, text="Descrição do
Princípio:", bg="#1e2a38", fg="#bdc3c7",
```

```
                font=("Helvetica",
9)).pack(anchor="w", padx=6)

        self.p_desc_text = tk.Text(

            parent, height=5, wrap="word",

            bg="#152232", fg="#bdc3c7",

            relief="flat", font=("Helvetica", 8),

            state="disabled",

        )

        self.p_desc_text.pack(fill="x", padx=6,
pady=(2, 8))


        btn_frame = tk.Frame(parent, bg="#1e2a38")

        btn_frame.pack(fill="x", padx=6, pady=4)


        self.btn_avaliar_p = tk.Button(

            btn_frame,

            text="▶  Avaliar Quesito (Gemini)",

            bg="#27ae60", fg="white", relief="flat",
cursor="hand2",

            font=("Helvetica", 10, "bold"), padx=10,
pady=6,

            command=self._avaliar_quesito,

        )

        self.btn_avaliar_p.pack(fill="x", pady=(0, 4))
```

```
        self.btn_avaliar_tp = tk.Button(
            btn_frame,
            text="🔄  Avaliar Princípio Completo",
            bg="#8e44ad", fg="white", relief="flat",
cursor="hand2",
            font=("Helvetica", 10, "bold"), padx=10,
pady=6,
            command=self._avaliar_principio_completo,
        )
        self.btn_avaliar_tp.pack(fill="x", pady=(0,
4))

        self.btn_relatorio_p = tk.Button(
            btn_frame,
            text="📊  Gerar Relatório Final",
            bg="#c0392b", fg="white", relief="flat",
cursor="hand2",
            font=("Helvetica", 10, "bold"), padx=10,
pady=6,
            command=self._gerar_relatorio_principios,
        )
        self.btn_relatorio_p.pack(fill="x")

    def _build_principios_right(self, parent):
```

```
        self._section_label(parent, "📋  Resultado - Princípios")

        self.nb_principios = ttk.Notebook(parent)
        self.nb_principios.pack(fill="both", expand=True, pady=(4, 0))

        self.p_tab_quesito   = tk.Frame(self.nb_principios, bg="#152232")
        self.p_tab_tabela    = tk.Frame(self.nb_principios, bg="#152232")
        self.p_tab_relatorio = tk.Frame(self.nb_principios, bg="#152232")

        self.nb_principios.add(self.p_tab_quesito, text="  Quesito Atual  ")
        self.nb_principios.add(self.p_tab_tabela, text="  Tabela de Pontuações  ")
        self.nb_principios.add(self.p_tab_relatorio, text="  Relatório Consolidado  ")

        self._build_p_tab_quesito(self.p_tab_quesito)
        self._build_p_tab_tabela(self.p_tab_tabela)
        self._build_p_tab_relatorio(self.p_tab_relatorio)

    def _build_p_tab_quesito(self, parent):
```

```
        top = tk.Frame(parent, bg="#152232")
        top.pack(fill="x", padx=10, pady=8)

        tk.Label(top, text="Princípio / Quesito:",
                 bg="#152232", fg="#95a5a6",
font=("Helvetica", 9)).grid(row=0, column=0,
sticky="w")
        self.p_lbl_pq = tk.Label(top, text="-",
                                 bg="#152232",
fg="#ecf0f1", font=("Helvetica", 9, "bold"))
        self.p_lbl_pq.grid(row=0, column=1,
sticky="w", padx=(6, 0))

        tk.Label(top, text="Pontuação:",
                 bg="#152232", fg="#95a5a6",
font=("Helvetica", 9)).grid(row=1, column=0,
sticky="w", pady=(4, 0))
        self.p_lbl_pontuacao = tk.Label(top, text="-",
                                        bg="#152232",
fg="#ecf0f1", font=("Helvetica", 12, "bold"))
        self.p_lbl_pontuacao.grid(row=1, column=1,
sticky="w", padx=(6, 0))

        tk.Label(parent, text="Fundamentação
Técnica:",
                 bg="#152232", fg="#95a5a6",
font=("Helvetica", 9)).pack(anchor="w", padx=10)
```

```
        self.p_arg_text = scrolledtext.ScrolledText(
            parent, wrap="word",
            bg="#0d1b2a", fg="#ecf0f1",
            relief="flat", font=("Helvetica", 10),
            padx=10, pady=10, state="disabled",
        )
        self.p_arg_text.pack(fill="both", expand=True,
padx=10, pady=(4, 10))


        tk.Button(
            parent, text="💾  Exportar este Resultado
(.txt)",
            bg="#2c3e50", fg="#ecf0f1", relief="flat",
cursor="hand2",
            font=("Helvetica", 9),
command=self._exportar_quesito,
        ).pack(padx=10, pady=(0, 10), anchor="e")


    def _build_p_tab_tabela(self, parent):
        frame = tk.Frame(parent, bg="#152232")
        frame.pack(fill="both", expand=True, padx=6,
pady=6)
```

```
        cols   = ("Princípio", "Quesito", "Pontuação",
"Classificação")

        widths = [280, 60, 90, 160]



        self.p_tree = ttk.Treeview(frame,
columns=cols, show="headings", height=20)

        for col, w in zip(cols, widths):

            self.p_tree.heading(col, text=col)

            self.p_tree.column(col, width=w,
anchor="w" if col == "Princípio" else "center")



        sb = ttk.Scrollbar(frame, orient="vertical",
command=self.p_tree.yview)

        self.p_tree.configure(yscrollcommand=sb.set)

        self.p_tree.pack(side="left", fill="both",
expand=True)

        sb.pack(side="right", fill="y")



        self.p_tree.tag_configure("ins",
foreground="#e74c3c")

        self.p_tree.tag_configure("raz",
foreground="#e67e22")

        self.p_tree.tag_configure("bom",
foreground="#2980b9")

        self.p_tree.tag_configure("mb",
foreground="#27ae60")
```

```
    def _build_p_tab_relatorio(self, parent):
        self.p_rel_text = scrolledtext.ScrolledText(
            parent, wrap="word",
            bg="#0d1b2a", fg="#ecf0f1",
            relief="flat", font=("Courier", 9),
            padx=12, pady=12, state="disabled",
        )
        self.p_rel_text.pack(fill="both", expand=True,
padx=6, pady=6)

        tk.Button(
            parent, text="💾  Salvar Relatório
(.txt)",
            bg="#2c3e50", fg="#ecf0f1", relief="flat",
cursor="hand2",
            font=("Helvetica", 9), command=lambda:
self._salvar_relatorio(self.p_rel_text),
        ).pack(padx=10, pady=(0, 8), anchor="e")

    # ── MÓDULO 2 - ELEMENTOS DE CONTEÚDO
──────────────────────────

    def _build_modulo_elementos(self, parent):
        main = tk.Frame(parent, bg="#1e2a38")
```

```
        main.pack(fill="both", expand=True, padx=8,
pady=8)


        left = tk.Frame(main, bg="#1e2a38", width=360)
        left.pack(side="left", fill="y", padx=(0, 8))
        left.pack_propagate(False)


        right = tk.Frame(main, bg="#1e2a38")
        right.pack(side="left", fill="both",
expand=True)


        self._build_elementos_left(left)
        self._build_elementos_right(right)


    def _build_elementos_left(self, parent):
        self._section_label(parent, "🔎  Seleção para
Avaliação")


        tk.Label(parent, text="Capítulo:",
bg="#1e2a38", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(anchor="w", padx=6, pady=(4, 0))


        cap_opts = [f"{k}. {v['titulo'].split('. ',
1)[1]}" for k, v in CAPITULOS.items()]
```

```
        self.capitulo_var = tk.StringVar()
        self.capitulo_cb  = ttk.Combobox(
            parent, textvariable=self.capitulo_var,
            values=cap_opts, state="readonly",
font=("Helvetica", 9),
        )
        self.capitulo_cb.pack(fill="x", padx=6,
pady=(2, 4))
        self.capitulo_cb.bind("<<ComboboxSelected>>",
self._on_capitulo_selected)

        tk.Label(parent, text="Elemento:",
bg="#1e2a38", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(anchor="w", padx=6, pady=(4, 0))
        self.elemento_var = tk.StringVar()
        self.elemento_cb  = ttk.Combobox(
            parent, textvariable=self.elemento_var,
            values=[], state="readonly",
font=("Helvetica", 9),
        )
        self.elemento_cb.pack(fill="x", padx=6,
pady=(2, 4))
        self.elemento_cb.bind("<<ComboboxSelected>>",
self._on_elemento_selected)
```

```
        tk.Label(parent, text="Descrição do
Capítulo:", bg="#1e2a38", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(anchor="w", padx=6)
        self.e_cap_desc = tk.Text(
            parent, height=4, wrap="word",
            bg="#152232", fg="#bdc3c7",
            relief="flat", font=("Helvetica", 8),
            state="disabled",
        )
        self.e_cap_desc.pack(fill="x", padx=6,
pady=(2, 4))


        tk.Label(parent, text="Descrição do
Elemento:", bg="#1e2a38", fg="#bdc3c7",
                 font=("Helvetica",
9)).pack(anchor="w", padx=6)
        self.e_elem_desc = tk.Text(
            parent, height=5, wrap="word",
            bg="#152232", fg="#95a5a6",
            relief="flat", font=("Helvetica", 8),
            state="disabled",
        )
        self.e_elem_desc.pack(fill="x", padx=6,
pady=(2, 8))
```

```
        btn_frame = tk.Frame(parent, bg="#1e2a38")
        btn_frame.pack(fill="x", padx=6, pady=4)

        self.btn_avaliar_e = tk.Button(
            btn_frame,
            text="▶  Avaliar Elemento (Gemini)",
            bg="#27ae60", fg="white", relief="flat",
cursor="hand2",
            font=("Helvetica", 10, "bold"), padx=10,
pady=6,
            command=self._avaliar_elemento,
        )
        self.btn_avaliar_e.pack(fill="x", pady=(0, 4))

        self.btn_avaliar_tc = tk.Button(
            btn_frame,
            text="🔄  Avaliar Capítulo Completo",
            bg="#8e44ad", fg="white", relief="flat",
cursor="hand2",
            font=("Helvetica", 10, "bold"), padx=10,
pady=6,
            command=self._avaliar_capitulo_completo,
        )
```

```
        self.btn_avaliar_tc.pack(fill="x", pady=(0,
4))


        self.btn_relatorio_e = tk.Button(
            btn_frame,
            text="📊  Gerar Relatório Final",
            bg="#c0392b", fg="white", relief="flat",
cursor="hand2",
            font=("Helvetica", 10, "bold"), padx=10,
pady=6,
            command=self._gerar_relatorio_elementos,
        )
        self.btn_relatorio_e.pack(fill="x")


    def _build_elementos_right(self, parent):
        self._section_label(parent, "📋  Resultado -
Elementos de Conteúdo")


        self.nb_elementos = ttk.Notebook(parent)
        self.nb_elementos.pack(fill="both",
expand=True, pady=(4, 0))


        self.e_tab_elem     =
tk.Frame(self.nb_elementos, bg="#152232")
        self.e_tab_tabela   =
tk.Frame(self.nb_elementos, bg="#152232")
```

```
        self.e_tab_relatorio =
tk.Frame(self.nb_elementos, bg="#152232")


        self.nb_elementos.add(self.e_tab_elem,
text="  Elemento Atual  ")
        self.nb_elementos.add(self.e_tab_tabela,
text="  Tabela de Pontuações  ")
        self.nb_elementos.add(self.e_tab_relatorio,
text="  Relatório Consolidado  ")


        self._build_e_tab_elem(self.e_tab_elem)
        self._build_e_tab_tabela(self.e_tab_tabela)

self._build_e_tab_relatorio(self.e_tab_relatorio)


    def _build_e_tab_elem(self, parent):
        top = tk.Frame(parent, bg="#152232")
        top.pack(fill="x", padx=10, pady=8)


        tk.Label(top, text="Capítulo / Elemento:",
                 bg="#152232", fg="#95a5a6",
font=("Helvetica", 9)).grid(row=0, column=0,
sticky="w")
        self.e_lbl_ce = tk.Label(top, text="-",
                                 bg="#152232",
fg="#ecf0f1", font=("Helvetica", 9, "bold"))
```

```
        self.e_lbl_ce.grid(row=0, column=1,
sticky="w", padx=(6, 0))


        tk.Label(top, text="Pontuação:",

                 bg="#152232", fg="#95a5a6",
font=("Helvetica", 9)).grid(row=1, column=0,
sticky="w", pady=(4, 0))

        self.e_lbl_pontuacao = tk.Label(top, text="-",

                                        bg="#152232",
fg="#ecf0f1", font=("Helvetica", 12, "bold"))

        self.e_lbl_pontuacao.grid(row=1, column=1,
sticky="w", padx=(6, 0))


        tk.Label(parent, text="Fundamentação
Técnica:",

                 bg="#152232", fg="#95a5a6",
font=("Helvetica", 9)).pack(anchor="w", padx=10)


        self.e_arg_text = scrolledtext.ScrolledText(

            parent, wrap="word",

            bg="#0d1b2a", fg="#ecf0f1",

            relief="flat", font=("Helvetica", 10),

            padx=10, pady=10, state="disabled",

        )

        self.e_arg_text.pack(fill="both", expand=True,
padx=10, pady=(4, 10))
```

```
        tk.Button(
            parent, text="💾  Exportar este Resultado (.txt)",
            bg="#2c3e50", fg="#ecf0f1", relief="flat", cursor="hand2",
            font=("Helvetica", 9), command=self._exportar_elemento,
        ).pack(padx=10, pady=(0, 10), anchor="e")


    def _build_e_tab_tabela(self, parent):
        frame = tk.Frame(parent, bg="#152232")
        frame.pack(fill="both", expand=True, padx=6, pady=6)

        cols   = ("Capítulo", "Elem.", "Pontuação", "Classificação")
        widths = [280, 50, 90, 220]

        self.e_tree = ttk.Treeview(frame, columns=cols, show="headings", height=20)
        for col, w in zip(cols, widths):
            self.e_tree.heading(col, text=col)
            self.e_tree.column(col, width=w, anchor="w" if col == "Capítulo" else "center")
```

```
        sb = ttk.Scrollbar(frame, orient="vertical",
command=self.e_tree.yview)

        self.e_tree.configure(yscrollcommand=sb.set)

        self.e_tree.pack(side="left", fill="both",
expand=True)

        sb.pack(side="right", fill="y")



        self.e_tree.tag_configure("ins",
foreground="#e74c3c")

        self.e_tree.tag_configure("raz",
foreground="#e67e22")

        self.e_tree.tag_configure("bom",
foreground="#2980b9")

        self.e_tree.tag_configure("mb",
foreground="#27ae60")


    def _build_e_tab_relatorio(self, parent):

        self.e_rel_text = scroll


    def _build_e_tab_relatorio(self, parent):

        self.e_rel_text = scrolledtext.ScrolledText(

            parent, wrap="word",

            bg="#0d1b2a", fg="#ecf0f1",

            relief="flat", font=("Courier", 9),

            padx=12, pady=12, state="disabled",
```

```
        )
        self.e_rel_text.pack(fill="both", expand=True,
padx=6, pady=6)

        tk.Button(
            parent, text="💾  Salvar Relatório
(.txt)",
            bg="#2c3e50", fg="#ecf0f1", relief="flat",
cursor="hand2",
            font=("Helvetica", 9), command=lambda:
self._salvar_relatorio(self.e_rel_text),
        ).pack(padx=10, pady=(0, 8), anchor="e")

    # ── Helpers gerais
─────────────────────────────────────────

    def _section_label(self, parent, texto):
        f = tk.Frame(parent, bg="#0d1b2a")
        f.pack(fill="x", pady=(6, 2))
        tk.Label(
            f, text=texto,
            bg="#0d1b2a", fg="#3498db",
            font=("Helvetica", 9, "bold"),
            padx=8, pady=4,
        ).pack(anchor="w")
```

```
    def _toggle_api_key(self):
        self.api_entry.config(show="" if
self.api_entry.cget("show") == "*" else "*")

    def _set_status(self, msg: str):
        self.status_var.set(msg)
        self.update_idletasks()

    def _set_buttons_state(self, state: str):
        for btn in (
            self.btn_avaliar_p, self.btn_avaliar_tp,
self.btn_relatorio_p,
            self.btn_avaliar_e, self.btn_avaliar_tc,
self.btn_relatorio_e,
        ):
            btn.config(state=state)

    # ── Carregar PDF
─────────────────────────────────────────────

    def _carregar_pdf(self):
        path = filedialog.askopenfilename(
            title="Selecionar Relatório de Gestão
(PDF)",
```

```
            filetypes=[("PDF files", "*.pdf"), ("All
files", "*.*")],

        )

        if not path:

            return


        self.pdf_path.set(path)

        self._set_status(f"Extraindo texto de:
{Path(path).name} …")

        self._set_buttons_state("disabled")

        self.progress.start(10)


        def _worker():

            try:

                texto = extrair_texto_pdf(path)

                self.texto_pdf = texto

                palavras = len(texto.split())

                chars    = len(texto)

                self.after(0, lambda:
self.pdf_info_label.config(

                    text=f"✓ Carregado: {palavras:,}
palavras | {chars:,} caracteres",

                    fg="#27ae60",

                ))
```

```
                self.after(0, lambda:
self._set_status(
                    f"PDF carregado com sucesso:
{Path(path).name}"
                ))
            except Exception as e:
                self.after(0, lambda:
messagebox.showerror("Erro ao ler PDF", str(e)))
                self.after(0, lambda:
self._set_status("Erro ao carregar PDF."))
                self.after(0, lambda:
self.pdf_info_label.config(
                    text="Erro ao carregar o
arquivo.", fg="#e74c3c",
                ))
            finally:
                self.after(0, self.progress.stop)
                self.after(0, lambda:
self._set_buttons_state("normal"))

        threading.Thread(target=_worker,
daemon=True).start()

    # ── Validações comuns
──────────────────────────────────────────

    def _validar_pre_avaliacao(self) -> bool:
```

```
        if not self.texto_pdf:
            messagebox.showwarning("Atenção", "Nenhum
PDF carregado. Carregue o documento primeiro.")
            return False
        if not self.api_key_var.get().strip():
            messagebox.showwarning("Atenção", "Informe
a Gemini API Key.")
            return False
        if not GEMINI_AVAILABLE:
            messagebox.showerror(
                "Erro",
                "Biblioteca 'google-generativeai' não
instalada.\n"
                "Execute: pip install
google-generativeai",
            )
            return False
        return True

    # ── Seleção - Princípios
──────────────────────────────────────

    def _on_principio_selected(self, event=None):
        sel = self.principio_var.get()
        if not sel:
```

```
            return

        num       = int(sel.split(".")[0])

        principio = PRINCIPIOS[num]


        opts = [f"{k}) {v}" for k, v in
principio["quesitos"].items()]

        self.quesito_cb.config(values=opts)

        self.quesito_var.set("")


        self.p_desc_text.config(state="normal")

        self.p_desc_text.delete("1.0", "end")

        self.p_desc_text.insert("end",
principio["descricao"])

        self.p_desc_text.config(state="disabled")


    def _get_selecao_principio(self):

        p_sel = self.principio_var.get()

        q_sel = self.quesito_var.get()

        if not p_sel or not q_sel:

            messagebox.showwarning("Atenção",
"Selecione um Princípio e um Quesito.")

            return None, None

        num   = int(p_sel.split(".")[0])

        letra = q_sel.split(")")[0].strip()
```

```
        return num, letra

    # ── Seleção - Elementos ──────────────────────────────

    def _on_capitulo_selected(self, event=None):
        sel = self.capitulo_var.get()
        if not sel:
            return
        num     = int(sel.split(".")[0])
        capitulo = CAPITULOS[num]

        opts = [f"{k}) {v[:80]}…" if len(v) > 80 else
f"{k}) {v}"
                for k, v in
capitulo["elementos"].items()]
        self.elemento_cb.config(values=opts)
        self.elemento_var.set("")

        self.e_cap_desc.config(state="normal")
        self.e_cap_desc.delete("1.0", "end")
        self.e_cap_desc.insert("end",
capitulo["descricao"])
        self.e_cap_desc.config(state="disabled")
```

```
        self.e_elem_desc.config(state="normal")
        self.e_elem_desc.delete("1.0", "end")
        self.e_elem_desc.config(state="disabled")

    def _on_elemento_selected(self, event=None):
        cap_sel  = self.capitulo_var.get()
        elem_sel = self.elemento_var.get()
        if not cap_sel or not elem_sel:
            return
        num_cap  = int(cap_sel.split(".")[0])
        num_elem = int(elem_sel.split(")")[0].strip())

        descricao =
CAPITULOS[num_cap]["elementos"][num_elem]
        self.e_elem_desc.config(state="normal")
        self.e_elem_desc.delete("1.0", "end")
        self.e_elem_desc.insert("end", descricao)
        self.e_elem_desc.config(state="disabled")

    def _get_selecao_elemento(self):
        cap_sel  = self.capitulo_var.get()
        elem_sel = self.elemento_var.get()
        if not cap_sel or not elem_sel:
```

```
            messagebox.showwarning("Atenção",
"Selecione um Capítulo e um Elemento.")
            return None, None
        num_cap  = int(cap_sel.split(".")[0])
        num_elem = int(elem_sel.split(")")[0].strip())
        return num_cap, num_elem

    # ── Avaliação - Princípios ──────────────────────────────

    def _avaliar_quesito(self):
        num, letra = self._get_selecao_principio()
        if num is None:
            return
        if not self._validar_pre_avaliacao():
            return

        self._set_buttons_state("disabled")
        self.progress.start(10)
        self._set_status(f"Avaliando Princípio {num},
Quesito {letra.upper()} com Gemini …")

        def _worker():
            try:
```

```
                resultado = avaliar_quesito(

api_key=self.api_key_var.get().strip(),
                    texto_relatorio=self.texto_pdf,
                    num_principio=num,
                    letra_quesito=letra,
                    modelo=self.modelo_var.get(),
                )
                self.resultados_principios[(num,
letra)] = resultado
                self.after(0, lambda:
self._exibir_resultado_principio(num, letra,
resultado))
                self.after(0,
self._atualizar_tabela_principios)
            except Exception as e:
                self.after(0, lambda:
messagebox.showerror("Erro na API Gemini", str(e)))
                self.after(0, lambda:
self._set_status("Erro na avaliação."))
            finally:
                self.after(0, self.progress.stop)
                self.after(0, lambda:
self._set_buttons_state("normal"))

        threading.Thread(target=_worker,
daemon=True).start()
```

```
    def _avaliar_principio_completo(self):
        p_sel = self.principio_var.get()
        if not p_sel:
            messagebox.showwarning("Atenção",
"Selecione um Princípio.")
            return
        if not self._validar_pre_avaliacao():
            return

        num    = int(p_sel.split(".")[0])
        letras =
list(PRINCIPIOS[num]["quesitos"].keys())

        self._set_buttons_state("disabled")
        self._set_status(f"Avaliando todos os quesitos
do Princípio {num} …")

        def _worker():
            for i, letra in enumerate(letras):
                self.after(0, lambda l=letra, i=i:
self._set_status(
                    f"Gemini — Princípio {num},
Quesito {l.upper()} ({i+1}/{len(letras)}) …"
                ))
```

```
                self.after(0, self.progress.start)
                try:
                    resultado = avaliar_quesito(
api_key=self.api_key_var.get().strip(),
texto_relatorio=self.texto_pdf,
                        num_principio=num,
                        letra_quesito=letra,
                        modelo=self.modelo_var.get(),
                    )
                    self.resultados_principios[(num,
letra)] = resultado
                    self.after(0, lambda n=num,
l=letra, r=resultado:
self._exibir_resultado_principio(n, l, r))
                    self.after(0,
self._atualizar_tabela_principios)
                except Exception as e:
                    self.after(0, lambda err=e:
messagebox.showerror("Erro na API Gemini", str(err)))
                    break

            self.after(0, self.progress.stop)
```

```
            self.after(0, lambda:
self._set_buttons_state("normal"))

            self.after(0, lambda: self._set_status(f"✓
Avaliação do Princípio {num} concluída."))

            self.after(0, lambda:
self.nb_principios.select(1))


        threading.Thread(target=_worker,
daemon=True).start()


    def _exibir_resultado_principio(self, num: int,
letra: str, resultado: dict):

        pontuacao     = resultado["pontuacao"]

        argumento     = resultado["argumento"]

        principio     = PRINCIPIOS[num]

        quesito_texto = principio["quesitos"][letra]

        label_score   = SCORES[pontuacao]


        self.p_lbl_pq.config(

            text=f"Princípio {num} — Quesito
{letra.upper()}: {quesito_texto[:60]}…"

        )

        cor = SCORE_COLORS.get(label_score, "#ecf0f1")

        self.p_lbl_pontuacao.config(text=label_score,
fg=cor)
```

```
        self.p_arg_text.config(state="normal")
        self.p_arg_text.delete("1.0", "end")
        self.p_arg_text.insert("end", argumento)
        self.p_arg_text.config(state="disabled")


        self.nb_principios.select(0)
        self._set_status(f"✓ Avaliado: Princípio {num}, Quesito {letra.upper()} → {label_score}")


    def _atualizar_tabela_principios(self):
        for item in self.p_tree.get_children():
            self.p_tree.delete(item)
        for (num, letra), resultado in sorted(self.resultados_principios.items()):
            pontuacao = resultado["pontuacao"]
            tag = {0: "ins", 1: "raz", 2: "bom", 3: "mb"}[pontuacao]
            self.p_tree.insert(
                "", "end",
                values=(PRINCIPIOS[num]["titulo"], letra.upper(), str(pontuacao), SCORES[pontuacao]),
                tags=(tag,),
            )
```

```
    # ── Avaliação - Elementos ──────────────────────────────────────


    def _avaliar_elemento(self):
        num_cap, num_elem = self._get_selecao_elemento()
        if num_cap is None:
            return
        if not self._validar_pre_avaliacao():
            return

        self._set_buttons_state("disabled")
        self.progress.start(10)
        self._set_status(f"Avaliando Capítulo {num_cap}, Elemento {num_elem} com Gemini …")

        def _worker():
            try:
                resultado = avaliar_elemento(
                    api_key=self.api_key_var.get().strip(),
                    texto_relatorio=self.texto_pdf,
                    num_capitulo=num_cap,
                    num_elemento=num_elem,
                    modelo=self.modelo_var.get(),
```

```
                )

                self.resultados_elementos[(num_cap,
num_elem)] = resultado

                self.after(0, lambda:
self._exibir_resultado_elemento(num_cap, num_elem,
resultado))

                self.after(0,
self._atualizar_tabela_elementos)

            except Exception as e:

                self.after(0, lambda:
messagebox.showerror("Erro na API Gemini", str(e)))

                self.after(0, lambda:
self._set_status("Erro na avaliação."))

            finally:

                self.after(0, self.progress.stop)

                self.after(0, lambda:
self._set_buttons_state("normal"))


        threading.Thread(target=_worker,
daemon=True).start()


    def _avaliar_capitulo_completo(self):

        cap_sel = self.capitulo_var.get()

        if not cap_sel:

            messagebox.showwarning("Atenção",
"Selecione um Capítulo.")

            return
```

```
        if not self._validar_pre_avaliacao():
            return

        num_cap  = int(cap_sel.split(".")[0])
        elementos =
list(CAPITULOS[num_cap]["elementos"].keys())

        self._set_buttons_state("disabled")
        self._set_status(f"Avaliando todos os
elementos do Capítulo {num_cap} …")

        def _worker():
            for i, num_elem in enumerate(elementos):
                self.after(0, lambda ne=num_elem, i=i:
self._set_status(
                    f"Gemini — Capítulo {num_cap},
Elemento {ne} ({i+1}/{len(elementos)}) …"
                ))
                self.after(0, self.progress.start)
                try:
                    resultado = avaliar_elemento(

api_key=self.api_key_var.get().strip(),

texto_relatorio=self.texto_pdf,
```

```
                        num_capitulo=num_cap,
                        num_elemento=num_elem,
                        modelo=self.modelo_var.get(),
                    )

self.resultados_elementos[(num_cap, num_elem)] =
resultado
                    self.after(0, lambda nc=num_cap,
ne=num_elem, r=resultado:

self._exibir_resultado_elemento(nc, ne, r))
                    self.after(0,
self._atualizar_tabela_elementos)
                except Exception as e:
                    self.after(0, lambda err=e:
messagebox.showerror("Erro na API Gemini", str(err)))
                    break

            self.after(0, self.progress.stop)
            self.after(0, lambda:
self._set_buttons_state("normal"))
            self.after(0, lambda: self._set_status(f"✓
Avaliação do Capítulo {num_cap} concluída."))
            self.after(0, lambda:
self.nb_elementos.select(1))
```

```
        threading.Thread(target=_worker,
daemon=True).start()


    def _exibir_resultado_elemento(self, num_cap: int,
num_elem: int, resultado: dict):
        pontuacao   = resultado["pontuacao"]
        argumento   = resultado["argumento"]
        label_score = SCORES_ELEMENTOS[pontuacao]


        self.e_lbl_ce.config(
            text=f"Capítulo {num_cap} — Elemento
{num_elem}"
        )
        cor = SCORE_COLORS_ELEMENTOS.get(label_score,
"#ecf0f1")
        self.e_lbl_pontuacao.config(text=label_score,
fg=cor)


        self.e_arg_text.config(state="normal")
        self.e_arg_text.delete("1.0", "end")
        self.e_arg_text.insert("end", argumento)
        self.e_arg_text.config(state="disabled")


        self.nb_elementos.select(0)
```

```
        self._set_status(f"✓ Avaliado: Capítulo
{num_cap}, Elemento {num_elem} → {label_score}")


    def _atualizar_tabela_elementos(self):
        for item in self.e_tree.get_children():
            self.e_tree.delete(item)
        for (num_cap, num_elem), resultado in
sorted(self.resultados_elementos.items()):
            pontuacao = resultado["pontuacao"]
            tag = {0: "ins", 1: "raz", 2: "bom", 3:
"mb"}[pontuacao]
            self.e_tree.insert(
                "", "end",
                values=(
                    CAPITULOS[num_cap]["titulo"],
                    str(num_elem),
                    str(pontuacao),
                    SCORES_ELEMENTOS[pontuacao],
                ),
                tags=(tag,),
            )


    # ── Relatórios ──────────────────────────────────────
```

```
    def _gerar_relatorio_principios(self):
        if not self.resultados_principios:
            messagebox.showwarning("Atenção", "Nenhuma
avaliação de princípios realizada ainda.")
            return

        linhas = []
        linhas.append("=" * 80)
        linhas.append("  RELATÓRIO DE AVALIAÇÃO -
PRINCÍPIOS DO RIG UNILA")
        linhas.append("  Simulação de Auditoria - TCU
|  Motor: Google Gemini")
        linhas.append("=" * 80)
        linhas.append("")

        total_pontos = total_quesitos = 0
        resumo = {}

        for num, dados in PRINCIPIOS.items():
            letras_av = [l for l in
dados["quesitos"].keys()
                         if (num, l) in
self.resultados_principios]
            if not letras_av:
                continue
```

```
            linhas.append("—" * 70)
            linhas.append(f"PRINCÍPIO {num}:
{dados['titulo']}")
            linhas.append("—" * 70)
            soma_p = 0
            for letra in sorted(letras_av):
                res   =
self.resultados_principios[(num, letra)]
                pont  = res["pontuacao"]
                soma_p        += pont
                total_pontos  += pont
                total_quesitos += 1
                linhas.append(f"\n  Quesito
{letra.upper()}: {dados['quesitos'][letra]}")
                linhas.append(f"  Pontuação:
{SCORES[pont]}")
                linhas.append(f"  Fundamentação:
{res['argumento']}")

            max_p = len(letras_av) * 3
            pct   = (soma_p / max_p * 100) if max_p >
0 else 0
            resumo[num] = (soma_p, max_p, pct)
            linhas.append(f"\n  SUBTOTAL Princípio
{num}: {soma_p}/{max_p} ({pct:.1f}%)")
```

```
            linhas.append("")


        max_total = total_quesitos * 3
        pct_total = (total_pontos / max_total * 100)
if max_total > 0 else 0


        linhas.append("=" * 80)
        linhas.append("  RESUMO GERAL - PRINCÍPIOS")
        linhas.append("=" * 80)
        for num, (soma_p, max_p, pct) in
resumo.items():
            linhas.append(
                f"  Princípio {num:2d} -
{PRINCIPIOS[num]['titulo']:<45} "
                f"{soma_p:2d}/{max_p:2d}
({pct:5.1f}%)"
            )


        conceito = (
            "MUITO BOM"    if pct_total >= 80 else
            "BOM"          if pct_total >= 60 else
            "RAZOÁVEL"     if pct_total >= 40 else
            "INSUFICIENTE"
        )
```

```
        linhas.append(f"\n  PONTUAÇÃO TOTAL:
{total_pontos}/{max_total} ({pct_total:.1f}%)")
        linhas.append(f"  CONCEITO GERAL: {conceito}")
        linhas.append("=" * 80)


        relatorio = "\n".join(linhas)
        self.p_rel_text.config(state="normal")
        self.p_rel_text.delete("1.0", "end")
        self.p_rel_text.insert("end", relatorio)
        self.p_rel_text.config(state="disabled")
        self.nb_principios.select(2)
        self._set_status("✓ Relatório de Princípios
gerado.")


    def _gerar_relatorio_elementos(self):
        if not self.resultados_elementos:
            messagebox.showwarning("Atenção", "Nenhuma
avaliação de elementos realizada ainda.")
            return


        linhas = []
        linhas.append("=" * 80)
        linhas.append("  RELATÓRIO DE AVALIAÇÃO -
ELEMENTOS DE CONTEÚDO DO RIG UNILA")
```

```
        linhas.append("  Simulação de Auditoria - TCU
|  Motor: Google Gemini")
        linhas.append("=" * 80)
        linhas.append("")

        total_pontos = total_elementos = 0
        resumo = {}

        for num_cap, dados in CAPITULOS.items():
            elems_av = [e for e in
dados["elementos"].keys()
                        if (num_cap, e) in
self.resultados_elementos]
            if not elems_av:
                continue

            linhas.append("—" * 70)
            linhas.append(f"CAPÍTULO {num_cap}:
{dados['titulo']}")
            linhas.append("—" * 70)
            soma_p = 0
            for num_elem in sorted(elems_av):
                res  =
self.resultados_elementos[(num_cap, num_elem)]
                pont = res["pontuacao"]
```

```
                soma_p          += pont
                total_pontos    += pont
                total_elementos += 1
                linhas.append(f"\n  Elemento
{num_elem}: {dados['elementos'][num_elem][:120]}…"
                              if
len(dados['elementos'][num_elem]) > 120
                              else f"\n  Elemento
{num_elem}: {dados['elementos'][num_elem]}")
                linhas.append(f"  Pontuação:
{SCORES_ELEMENTOS[pont]}")
                linhas.append(f"  Fundamentação:
{res['argumento']}")


            max_p = len(elems_av) * 3
            pct   = (soma_p / max_p * 100) if max_p >
0 else 0
            resumo[num_cap] = (soma_p, max_p, pct)
            linhas.append(f"\n  SUBTOTAL Capítulo
{num_cap}: {soma_p}/{max_p} ({pct:.1f}%)")
            linhas.append("")


        max_total = total_elementos * 3
        pct_total = (total_pontos / max_total * 100)
if max_total > 0 else 0
```

```
        linhas.append("=" * 80)

        linhas.append("  RESUMO GERAL - ELEMENTOS DE
CONTEÚDO")

        linhas.append("=" * 80)

        for num_cap, (soma_p, max_p, pct) in
resumo.items():

            linhas.append(

                f"  Capítulo {num_cap} -
{CAPITULOS[num_cap]['titulo']:<50} "

                f"{soma_p:2d}/{max_p:2d}
({pct:5.1f}%)"

            )


        conceito = (

            "MUITO BOM"    if pct_total >= 80 else

            "BOM"          if pct_total >= 60 else

            "RAZOÁVEL"     if pct_total >= 40 else

            "INSUFICIENTE"

        )

        linhas.append(f"\n  PONTUAÇÃO TOTAL:
{total_pontos}/{max_total} ({pct_total:.1f}%)")

        linhas.append(f"  CONCEITO GERAL: {conceito}")

        linhas.append("=" * 80)


        relatorio = "\n".join(linhas)
```

```
        self.e_rel_text.config(state="normal")
        self.e_rel_text.delete("1.0", "end")
        self.e_rel_text.insert("end", relatorio)
        self.e_rel_text.config(state="disabled")
        self.nb_elementos.select(2)
        self._set_status("✓ Relatório de Elementos de
Conteúdo gerado.")


    # ── Exportações
─────────────────────────────────────────────────────


    def _exportar_quesito(self):
        conteudo = self.p_arg_text.get("1.0",
"end").strip()
        if not conteudo:
            messagebox.showwarning("Atenção", "Nenhum
resultado para exportar.")
            return
        path = filedialog.asksaveasfilename(
            defaultextension=".txt", filetypes=[("Text
files", "*.txt")],
            title="Salvar resultado do quesito",
        )
        if path:
```

```
            with open(path, "w", encoding="utf-8") as
f:

f.write(f"{self.p_lbl_pq.cget('text')}\n")
                f.write(f"Pontuação:
{self.p_lbl_pontuacao.cget('text')}\n\n")
                f.write("FUNDAMENTAÇÃO TÉCNICA:\n")
                f.write(conteudo)
            self._set_status(f"✓ Exportado: {path}")


    def _exportar_elemento(self):
        conteudo = self.e_arg_text.get("1.0",
"end").strip()
        if not conteudo:
            messagebox.showwarning("Atenção", "Nenhum
resultado para exportar.")
            return
        path = filedialog.asksaveasfilename(
            defaultextension=".txt", filetypes=[("Text
files", "*.txt")],
            title="Salvar resultado do elemento",
        )
        if path:
            with open(path, "w", encoding="utf-8") as
f:
```

```
f.write(f"{self.e_lbl_ce.cget('text')}\n")
                f.write(f"Pontuação:
{self.e_lbl_pontuacao.cget('text')}\n\n")
                f.write("FUNDAMENTAÇÃO TÉCNICA:\n")
                f.write(conteudo)
            self._set_status(f"✓ Exportado: {path}")


    def _salvar_relatorio(self, widget:
scrolledtext.ScrolledText):
        conteudo = widget.get("1.0", "end").strip()
        if not conteudo:
            messagebox.showwarning("Atenção", "Nenhum
relatório para salvar.")
            return
        path = filedialog.asksaveasfilename(
            defaultextension=".txt", filetypes=[("Text
files", "*.txt")],
            title="Salvar Relatório Final",
        )
        if path:
            with open(path, "w", encoding="utf-8") as
f:
                f.write(conteudo)
            self._set_status(f"✓ Relatório salvo:
{path}")
```

```
# ──────────────────────────────────────────────────
# VERIFICAÇÃO DE DEPENDÊNCIAS
# ──────────────────────────────────────────────────

def verificar_dependencias():
    faltando = []
    if not GEMINI_AVAILABLE:
        faltando.append("google-generativeai")
    if not PYMUPDF_AVAILABLE and not
PDFPLUMBER_AVAILABLE:
        faltando.append("pymupdf  OU  pdfplumber")
    if faltando:
        root = tk.Tk()
        root.withdraw()
        messagebox.showerror(
            "Dependências ausentes",
            "As seguintes bibliotecas são necessárias
e não estão instaladas:\n\n"
            + "\n".join(f"  • {lib}" for lib in
faltando)
            + "\n\nInstale com:\n  pip install
google-generativeai pymupdf pdfplumber",
        )
```

```
# ──────────────────────────────────────────────────────
# ENTRADA
# ──────────────────────────────────────────────────────

if __name__ == "__main__":
    verificar_dependencias()
    app = AuditApp()

    style = ttk.Style(app)
    try:
        style.theme_use("clam")
    except Exception:
        pass

    style.configure("TCombobox",
                    fieldbackground="#2c3e50",
background="#2c3e50",
                    foreground="#ecf0f1",
selectbackground="#2980b9")
    style.configure("TNotebook", background="#1e2a38",
borderwidth=0)
    style.configure("TNotebook.Tab",
background="#2c3e50", foreground="#bdc3c7",
```

```
                    padding=[10, 4],
font=("Helvetica", 9))

    style.map("TNotebook.Tab",

              background=[("selected", "#152232")],

              foreground=[("selected", "#3498db")])

    style.configure("Treeview",

                    background="#152232",
foreground="#ecf0f1",

                    fieldbackground="#152232",
rowheight=24, font=("Helvetica", 9))

    style.configure("Treeview.Heading",

                    background="#0d1b2a",
foreground="#3498db",

                    font=("Helvetica", 9, "bold"))

    style.map("Treeview", background=[("selected",
"#2980b9")])



    app.mainloop()
```

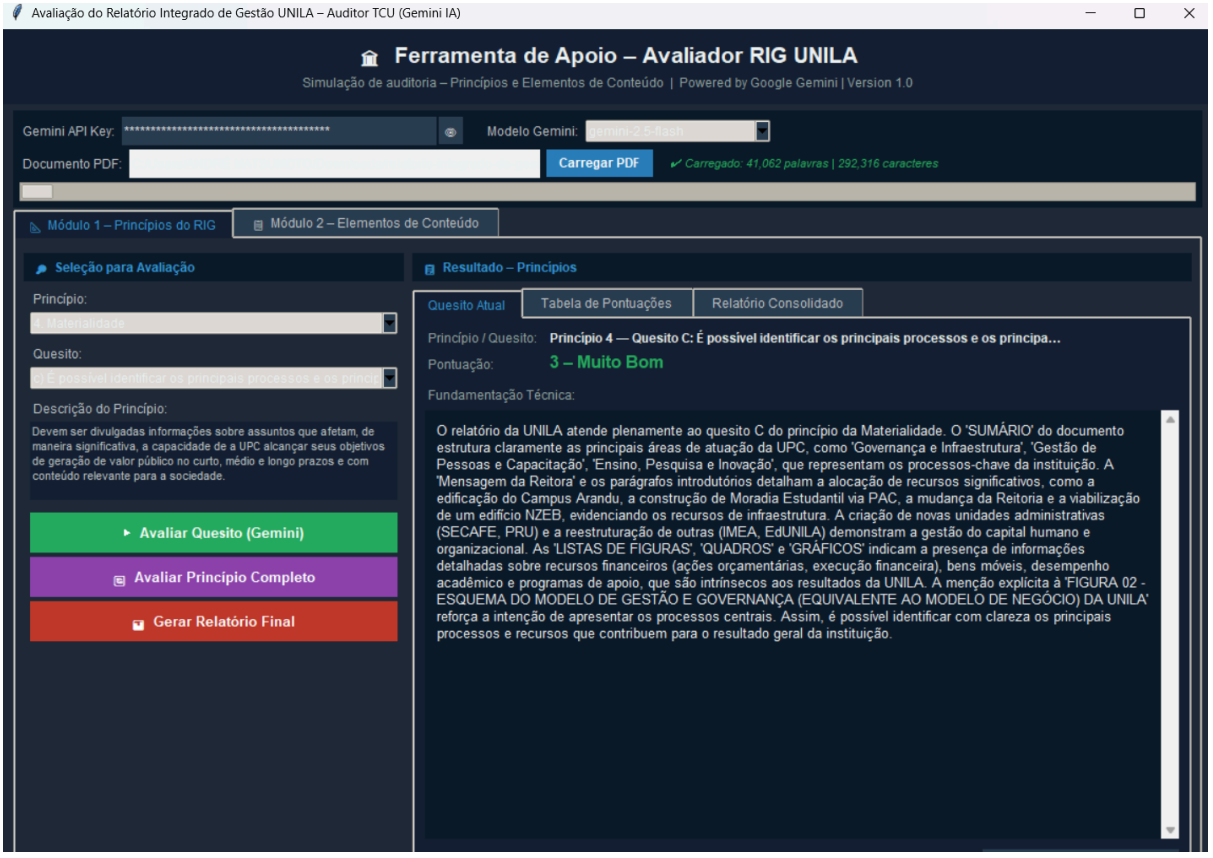
TERMO Nº 3/2026 - AUDIN

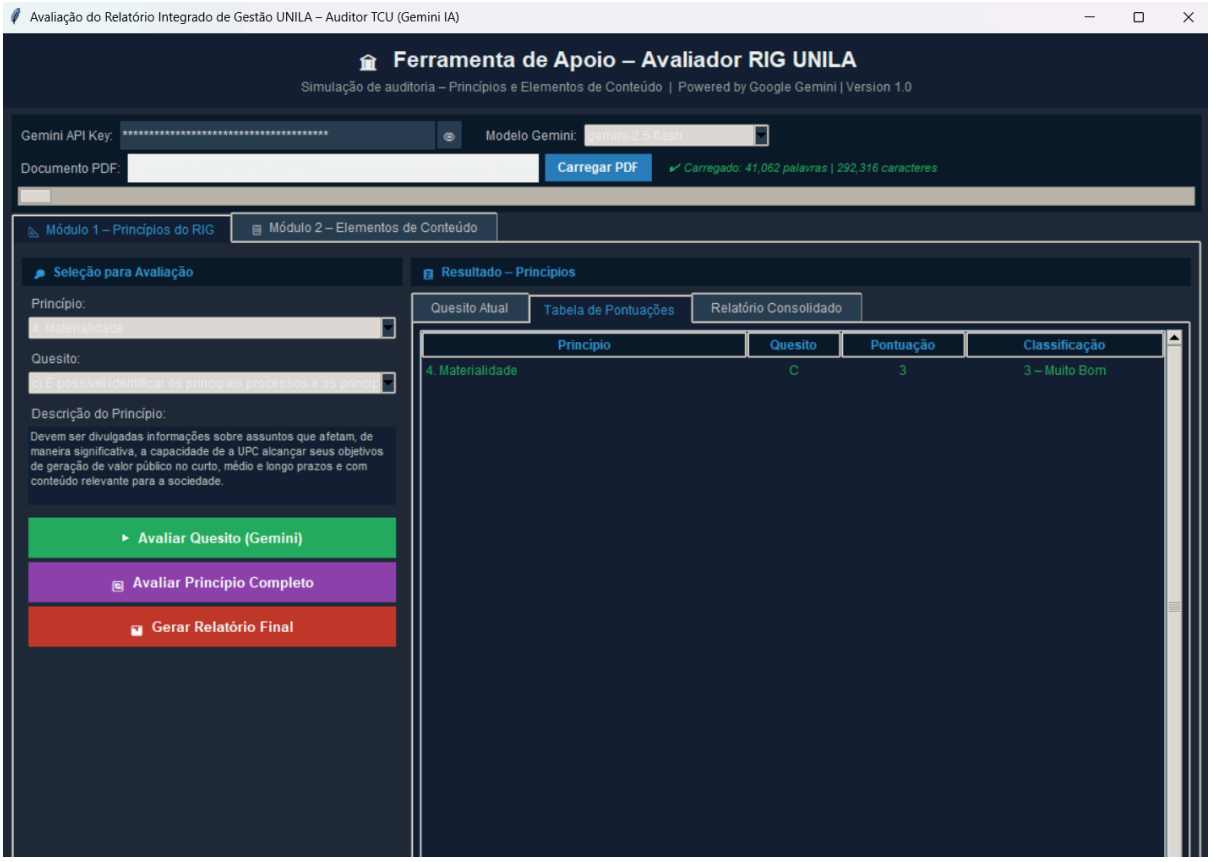
(Nº do Protocolo: NÃO PROTOCOLADO)

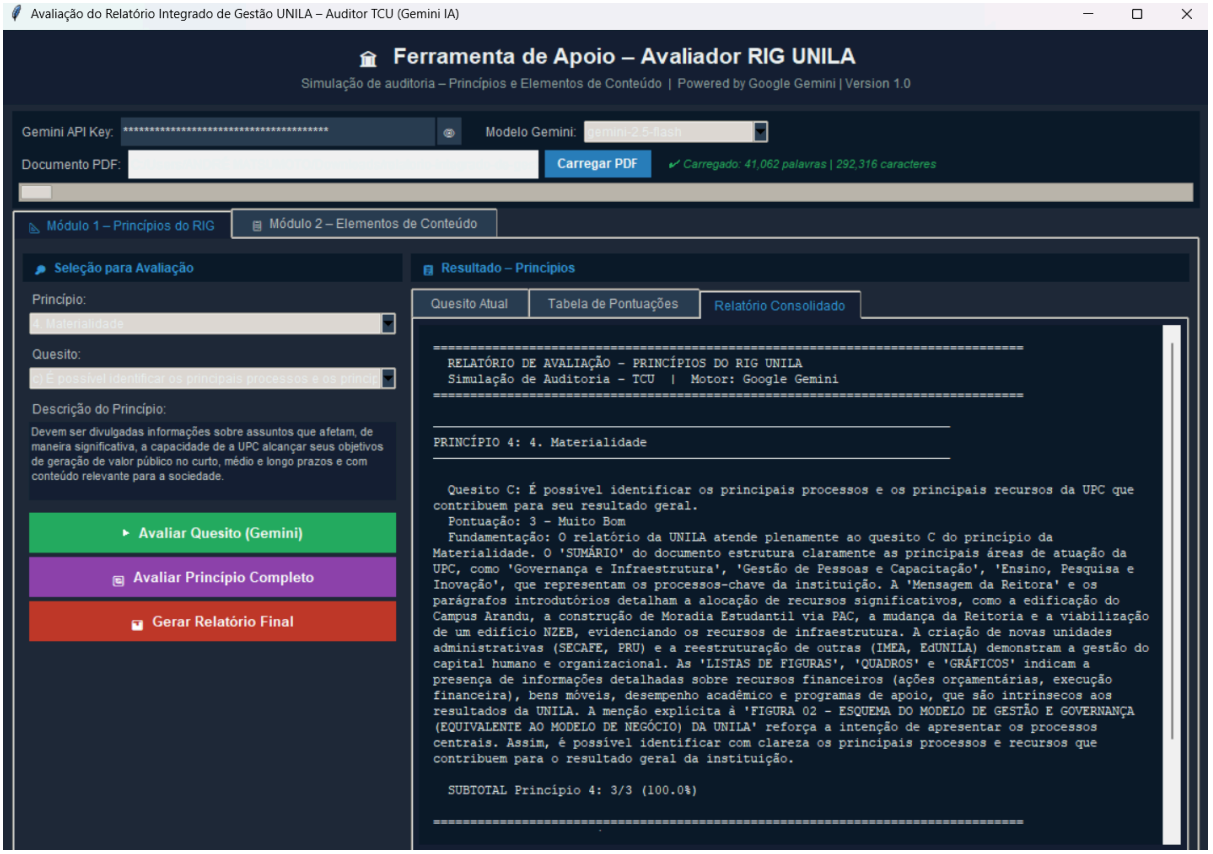
(Assinado digitalmente em 12/05/2026 10:43)
GUILLERMO JAVIER DIAZ VILLAVICENCIO
CHEFE DA AUDITOR(A)IA INTERNA - TITULAR
AUDIN (10.01.05.16)
Matrícula: ###903#1

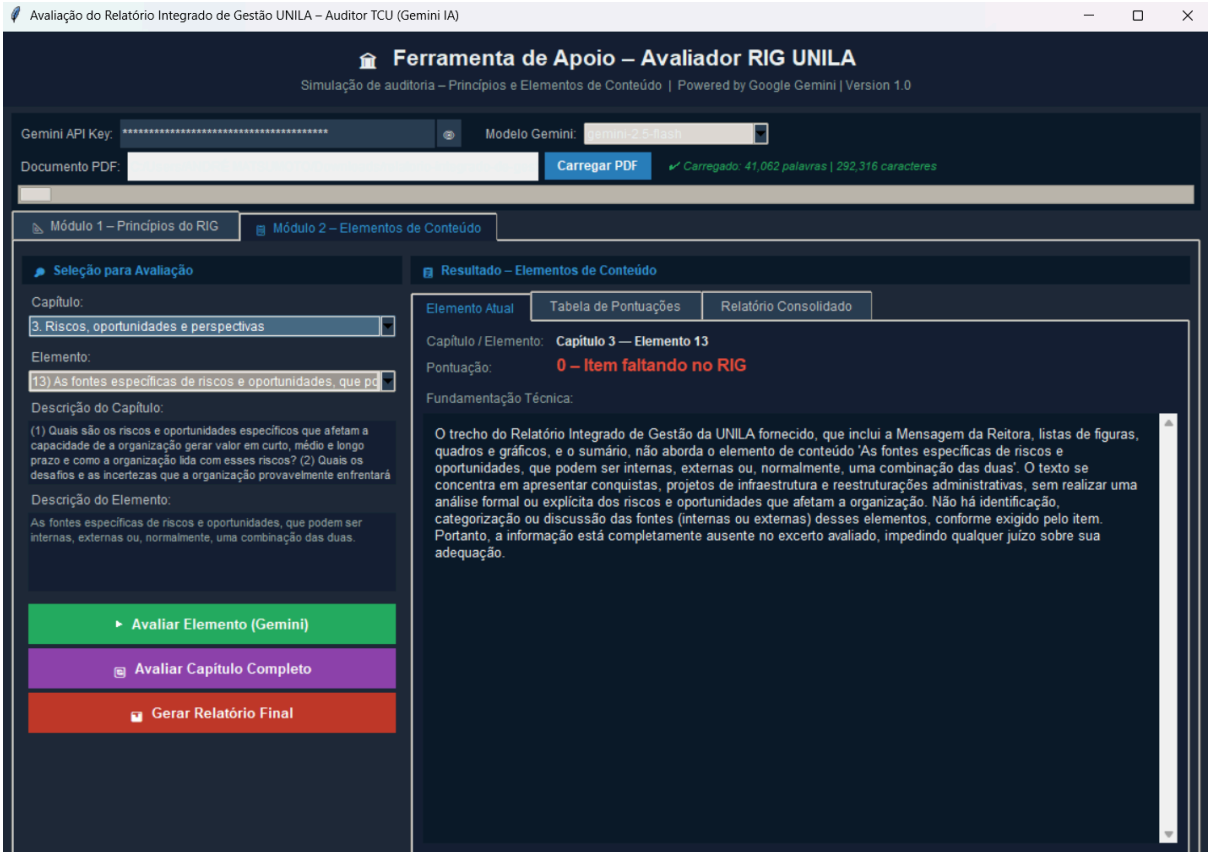
*Visualize o documento original em https://sig.unila.edu.br/documentos/ informando seu número: **3**, ano: **2026**, tipo: **TERMO**, data de emissão: **12/05/2026** e o código de verificação: **45977b7476***

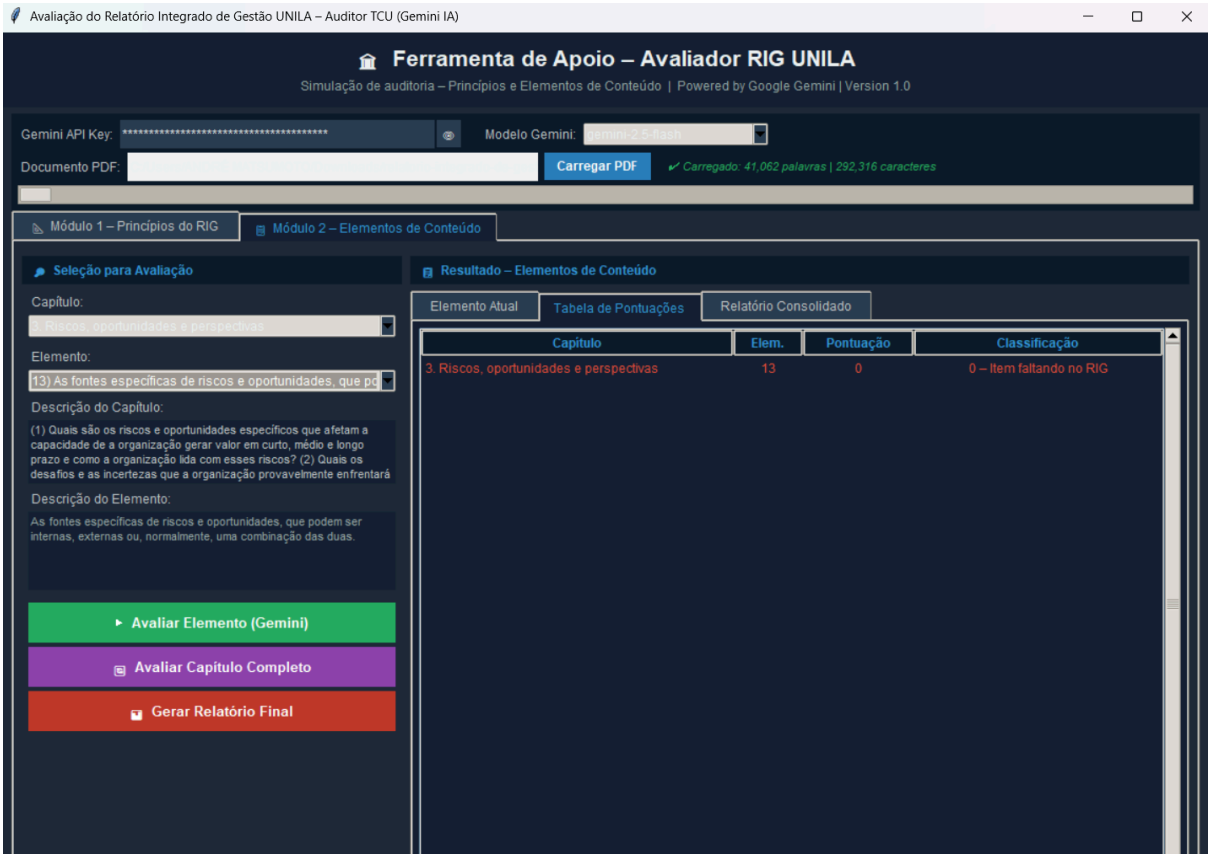
Anexo II - Evidências de testes e uso experimental na Auditoria Interna

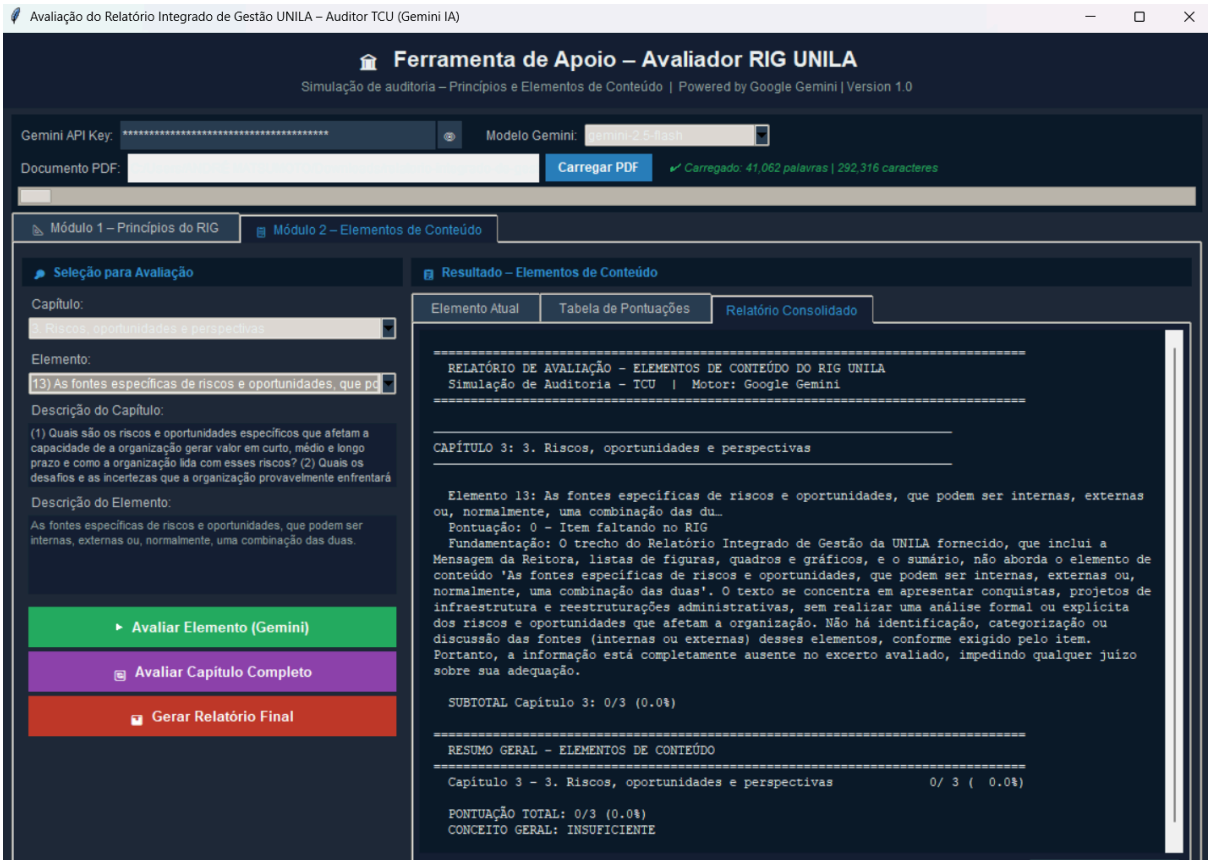





Avaliação do Relatório Integrado de Gestão UNILA – Auditor TCU (Gemini IA)
Ferramenta de Apoio – Avaliador RIG UNILA
Simulação de auditoria – Princípios e Elementos de Conteúdo | Powered by Google Gemini | Version 1.0
Gemini API Key:
Modelo Gemini:
Documento PDF:
Carregar PDF
Carregado: 41,062 palavras | 292,316 caracteres
Módulo 1 – Princípios do RIG
Módulo 2 – Elementos de Conteúdo
Seleção para Avaliação
Princípio:
4. Materialidade
Quesito:
c) É possível identificar os principais processos e os princip
Descrição do Princípio:
Devem ser divulgadas informações sobre assuntos que afetam, de maneira significativa, a capacidade de a UPC alcançar seus objetivos de geração de valor público no curto, médio e longo prazos e com conteúdo relevante para a sociedade.
Avaliar Quesito (Gemini)
Avaliar Princípio Completo
Gerar Relatório Final
Resultado – Princípios
Quesito Atual
Tabela de Pontuações
Relatório Consolidado
RELATÓRIO DE AVALIAÇÃO – PRINCÍPIOS DO RIG UNILA
Simulação de Auditoria – TCU | Motor: Google Gemini
PRINCÍPIO 4: 4. Materialidade
Quesito C: É possível identificar os principais processos e os principais recursos da UPC que contribuem para seu resultado geral.
Pontuação: 3 – Muito Bom
Fundamentação: O relatório da UNILA atende plenamente ao quesito C do princípio da Materialidade. O 'SUMÁRIO' do documento estrutura claramente as principais áreas de atuação da UPC, como 'Governança e Infraestrutura', 'Gestão de Pessoas e Capacitação', 'Ensino, Pesquisa e Inovação', que representam os processos-chave da instituição. A 'Mensagem da Reitora' e os parágrafos introdutórios detalham a alocação de recursos significativos, como a edificação do Campus Arandu, a construção de Moradia Estudantil via PAC, a mudança da Reitoria e a viabilização de um edifício NZEB, evidenciando os recursos de infraestrutura. A criação de novas unidades administrativas (SECAFE, PRU) e a reestruturação de outras (IMEA, EdUNILA) demonstram a gestão do capital humano e organizacional. As 'LISTAS DE FIGURAS', 'QUADROS' e 'GRÁFICOS' indicam a presença de informações detalhadas sobre recursos financeiros (ações orçamentárias, execução financeira), bens móveis, desempenho acadêmico e programas de apoio, que são intrínsecos aos resultados da UNILA. A menção explícita à 'FIGURA 02 – ESQUEMA DO MODELO DE GESTÃO E GOVERNANÇA (EQUIVALENTE AO MODELO DE NEGÓCIO) DA UNILA' reforça a intenção de apresentar os processos centrais. Assim, é possível identificar com clareza os principais processos e recursos que contribuem para o resultado geral da instituição.
SUBTOTAL Princípio 4: 3/3 (100.0%)


Avaliação do Relatório Integrado de Gestão UNILA – Auditor TCU (Gemini IA)
Ferramenta de Apoio – Avaliador RIG UNILA
Simulação de auditoria – Princípios e Elementos de Conteúdo | Powered by Google Gemini | Version 1.0
Gemini API Key:
Modelo Gemini:
Documento PDF:
Carregar PDF
Carregado: 41,062 palavras | 292,316 caracteres
Módulo 1 – Princípios do RIG
Módulo 2 – Elementos de Conteúdo
Seleção para Avaliação
Capítulo:
3. Riscos, oportunidades e perspectivas
Elemento:
13) As fontes específicas de riscos e oportunidades, que pc
Descrição do Capítulo:
(1) Quais são os riscos e oportunidades específicos que afetam a capacidade de a organização gerar valor em curto, médio e longo prazo e como a organização lida com esses riscos? (2) Quais os desafios e as incertezas que a organização provavelmente enfrentará
Descrição do Elemento:
As fontes específicas de riscos e oportunidades, que podem ser internas, externas ou, normalmente, uma combinação das duas.
Avaliar Elemento (Gemini)
Avaliar Capítulo Completo
Gerar Relatório Final
Resultado – Elementos de Conteúdo
Elemento Atual
Tabela de Pontuações
Relatório Consolidado
Capítulo / Elemento: Capítulo 3 — Elemento 13
Pontuação: 0 – Item faltando no RIG
Fundamentação Técnica:
O trecho do Relatório Integrado de Gestão da UNILA fornecido, que inclui a Mensagem da Reitora, listas de figuras, quadros e gráficos, e o sumário, não aborda o elemento de conteúdo 'As fontes específicas de riscos e oportunidades, que podem ser internas, externas ou, normalmente, uma combinação das duas'. O texto se concentra em apresentar conquistas, projetos de infraestrutura e reestruturações administrativas, sem realizar uma análise formal ou explícita dos riscos e oportunidades que afetam a organização. Não há identificação, categorização ou discussão das fontes (internas ou externas) desses elementos, conforme exigido pelo item. Portanto, a informação está completamente ausente no excerto avaliado, impedindo qualquer juízo sobre sua adequação.


Avaliação do Relatório Integrado de Gestão UNILA – Auditor TCU (Gemini IA)
Ferramenta de Apoio – Avaliador RIG UNILA
Simulação de auditoria – Princípios e Elementos de Conteúdo | Powered by Google Gemini | Version 1.0
Gemini API Key:
Modelo Gemini:
Documento PDF:
Carregar PDF
Carregado: 41,062 palavras | 292,316 caracteres
Módulo 1 – Princípios do RIG
Módulo 2 – Elementos de Conteúdo
Seleção para Avaliação
Capítulo:
Elemento:
13) As fontes específicas de riscos e oportunidades, que pc
Descrição do Capítulo:
(1) Quais são os riscos e oportunidades específicos que afetam a capacidade de a organização gerar valor em curto, médio e longo prazo e como a organização lida com esses riscos? (2) Quais os desafios e as incertezas que a organização provavelmente enfrentará
Descrição do Elemento:
As fontes específicas de riscos e oportunidades, que podem ser internas, externas ou, normalmente, uma combinação das duas.
Avaliar Elemento (Gemini)
Avaliar Capítulo Completo
Gerar Relatório Final
Resultado – Elementos de Conteúdo
Elemento Atual
Tabela de Pontuações
Relatório Consolidado
Capítulo
Elem.
Pontuação
Classificação
3. Riscos, oportunidades e perspectivas
13
0
0 – Item faltando no RIG


Avaliação do Relatório Integrado de Gestão UNILA – Auditor TCU (Gemini IA)
Ferramenta de Apoio – Avaliador RIG UNILA
Simulação de auditoria – Princípios e Elementos de Conteúdo | Powered by Google Gemini | Version 1.0
Gemini API Key:
Modelo Gemini:
Documento PDF:
Carregar PDF
Carregado: 41,062 palavras | 292,316 caracteres
Módulo 1 – Princípios do RIG
Módulo 2 – Elementos de Conteúdo
Seleção para Avaliação
Capítulo:
Elemento:
13) As fontes específicas de riscos e oportunidades, que pc
Descrição do Capítulo:
(1) Quais são os riscos e oportunidades específicos que afetam a capacidade de a organização gerar valor em curto, médio e longo prazo e como a organização lida com esses riscos? (2) Quais os desafios e as incertezas que a organização provavelmente enfrentará
Descrição do Elemento:
As fontes específicas de riscos e oportunidades, que podem ser internas, externas ou, normalmente, uma combinação das duas.
Avaliar Elemento (Gemini)
Avaliar Capítulo Completo
Gerar Relatório Final
Resultado – Elementos de Conteúdo
Elemento Atual
Tabela de Pontuações
Relatório Consolidado
RELATÓRIO DE AVALIAÇÃO – ELEMENTOS DE CONTEÚDO DO RIG UNILA
Simulação de Auditoria – TCU | Motor: Google Gemini
CAPÍTULO 3: 3. Riscos, oportunidades e perspectivas
Elemento 13: As fontes específicas de riscos e oportunidades, que podem ser internas, externas ou, normalmente, uma combinação das du…
Pontuação: 0 – Item faltando no RIG
Fundamentação: O trecho do Relatório Integrado de Gestão da UNILA fornecido, que inclui a Mensagem da Reitora, listas de figuras, quadros e gráficos, e o sumário, não aborda o elemento de conteúdo 'As fontes específicas de riscos e oportunidades, que podem ser internas, externas ou, normalmente, uma combinação das duas'. O texto se concentra em apresentar conquistas, projetos de infraestrutura e reestruturações administrativas, sem realizar uma análise formal ou explícita dos riscos e oportunidades que afetam a organização. Não há identificação, categorização ou discussão das fontes (internas ou externas) desses elementos, conforme exigido pelo item. Portanto, a informação está completamente ausente no excerto avaliado, impedindo qualquer juízo sobre sua adequação.
SUBTOTAL Capítulo 3: 0/3 (0.0%)
RESUMO GERAL – ELEMENTOS DE CONTEÚDO
Capítulo 3 – 3. Riscos, oportunidades e perspectivas 0/ 3 (0.0%)
PONTUAÇÃO TOTAL: 0/3 (0.0%)
CONCEITO GERAL: INSUFICIENTE

TERMO Nº 4/2026 - AUDIN

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 12/05/2026 10:43)
GUILLERMO JAVIER DIAZ VILLAVICENCIO
CHEFE DA AUDITOR(A)IA INTERNA - TITULAR
AUDIN (10.01.05.16)
Matrícula: ###903#1

*Visualize o documento original em https://sig.unila.edu.br/documentos/ informando seu número: **4**, ano: **2026**, tipo: **TERMO**, data de emissão: **12/05/2026** e o código de verificação: **73b2a22363***