BPMAPPR - PRODCUT REQUIRMENTS DOCUMENT

1. Sumário Executivo

O BPMappr é uma plataforma SaaS de “Process AI” end-to-end que converte conhecimento organizacional disperso (texto, documentos, imagens) em modelos BPMN 2.0 consistentes, versionados e prontos para automação. O MVP busca construir um produto com ingestão multimodal (documentos e imagens), RAG corporativo, orquestração multi-agente, editor visual inteligente bpmn-js com ferramenta drag-and-drop, versionamento e avaliação objetiva de qualidade.

Resultado-alvo do MVP:

- Gerar um processo BPMN 2.0 “bom o suficiente” a partir de PDF/Doc e/ou imagem de fluxograma e/ou texto imputado.
- Editar esse diagrama por comandos naturais dentro do editor e ferramenta drag-and-drop (copiloto).
- Permitir que o usuário valide e visualize o layout do fluxo montado.
- Permitir que o usuário versione e exporte o modelo (BPMN XML + PNG).

2. Contexto, Problema e Oportunidade

- Problemas típicos nas organizações: conhecimento disperso, baixa adoção de BPMN, alto custo/tempo para mapear processos e barreiras técnicas para usuários de negócio.
- Oportunidade: posicionar-se como a primeira plataforma “Process AI” com pipeline de ingestão → RAG → multiagente → BPMN 2.0 (geração/edição) embutida em um editor web moderno.

3. Objetivos (KPIs do MVP 1.0)

- Precisão estrutural na geração inicial: ≥ 70% de correspondência com um ground truth por GED/RGED (bench interno).
- Taxa de sucesso de edição por comando natural (ex.: “duplique a tarefa X e conecte com um gateway”): ≥ 85% em 30 casos comuns.
- Tempo de ponta-a-ponta (doc → diagrama visual): ≤ 60 s (arquivos até 10 páginas).
- Satisfação do usuário early-adopter (CSAT no app): ≥ 4/5.
- Custo médio por processo (tokens + compute): alvo ≤ US$ 0,12 no perfil “texto+pdf curto”.

4. Personas

- Analista de Processos (principal), pode ser da empresa ou um consultor de uma consultoria que está oferecendo o serviço de modelagem de processo para a empresa: quer rapidez e padrão.
- Gestor de Área: quer entender e revisar sem “falar BPMNês”.

- Arquiteto/CoE de Automação: quer conformidade, versionamento e exportação estável (BPMN XML).

5. Escopo do MVP 1.0 (em epics)

EPIC A — Ingestão Multimodal Essencial

- Upload: PDF/DOCX, TXT/Markdown e Imagem (PNG/JPG).
- Parsing de documentos: Docling ou Unstructured + PyMuPDF (texto, layout, tabelas, imagens incorporadas).
- Normalização multimodal: unificar tudo em "Artefatos" (texto limpo + metadados + referências/citações de origem).

EPIC B — RAG Corporativo (v1)

- Vetorização com embeddings e armazenamento em PostgreSQL + pgvector (evita DB adicional e simplifica deploy).
- Indexação por tipo (documento/texto/imagem) e chunking com metadados (página).
- Consulta: *retrievers* por semântica + filtros (tipo de artefato), com citações exibidas no chat/insights.
- Framework sugerido: LlamaIndex (ou LangChain nativo) para pipeline de dados/avaliabilidade.

EPIC C — Orquestração Multiagente (LangGraph)

- Agente de Ingestão (extrai/limpa, detecta idioma, PII light).
- Agente de Extração de Passos (do corpus RAG, imagem de fluxograma).
- Agente de Síntese BPMN (esboça fluxo em representação interna JSON). Um paper do projeto mostra ganhos de confiabilidade ao editar via JSON e converter para XML depois.
- Agente Linter/Validador (regras BPMN, *dead ends*, gateways consistentes).
- Agente Layout/Visualização (ELK.js — layered, com hierarquia de pools/lanes e message flows).
- Sincronização Editor↔IA: a cada edição no canvas (eventBus/commandStack.changed), salvar XML, atualizar o BPMN_JSON "vivo" e versionar; o agente sempre opera sobre a última versão.
- Supervisor (controle de loop humano-no-meio e resolução de conflitos).
- Framework recomendado: LangGraph (controle de fluxo, memória, streaming).

EPIC D — Editor Visual Inteligente com drag-and-drop

D0. Objetivo
Permitir que, após a geração automática do BPMN pela IA, o usuário modele e refine o diagrama por drag-and-drop em um modeler web com paleta de elementos, contexto de edição, *snapping*, atalhos de teclado, undo/redo, e painel de propriedades — mantendo conformidade com BPMN 2.0. Base tecnológica: bpmn-js (toolkit oficial *bpmn.io*) sobre diagram-js.

D1. Decisões de tecnologia

- Modeler: bpmn-js (viewer + modeler). Paleta, context pad, replace menu, edição de rótulos e atalhos; inclui ferramentas create/append anything (atalhos N e A).
- Base de interação: diagram-js (mover/criar/redimensionar; *snapping*; bendpoints; re-roteamento).
- Painel de Propriedades: bpmn-js-properties-panel (editável e extensível; integra ciclo de edição com undo/redo).
- Linting: bpmnlint com conjunto mínimo de regras (ex.: *start-event-required*, *no-disconnected*, *no-complex-gateway*).
- Extras recomendados: minimap (navegação em diagramas grandes), seleção múltipla e context pad multi-elemento (edição em lote).

D2. Requisitos Funcionais (RF-D Drag-and-Drop)

- RF-D1 Paleta: exibir categorias (Eventos, Gateways, Tarefas, Subprocessos, Artefatos, Pools/Lanes) e permitir arrastar elementos para o canvas.
- RF-D2 Context Pad: ao selecionar um elemento, mostrar ações rápidas (apendar tarefa, trocar tipo/gateway, conectar, excluir). Suportar multi-seleção com *context pad* multi-elemento.
- RF-D3 Criação & Conexão: criar elementos por drag a partir da paleta ou por *create/append anything* (atalhos N/A), com snapping e guias de alinhamento.
- RF-D4 Movimento & Layout manual: mover blocos livremente; auto-reroute de conexões; ajustar bendpoints e segmentos ao arrastar.
- RF-D5 Redimensionar & Distribuir: *handles* visuais de drag/resize; alinhamento e distribuição uniforme de múltiplos elementos.
- RF-D6 Atalhos & Ferramentas: undo/redo, copy/paste, *lasso tool*, *hand tool* (pan), zoom (wheel), space tool.
- RF-D7 Propriedades: edição no properties panel (nome, documentação, tipos de tarefa, *multi-instance*, marcadores, mensagens, etc.).
- RF-D8 Linting em tempo real: avisos não-bloqueantes; bloqueios apenas para erros críticos (ex.: diagrama desconectado). Regras base bpmnlint.
- RF-D9 Import/Export: importar BPMN XML (validar e abrir no modeler) e exportar BPMN XML + SVG/PNG com a posição pós-edição.
- RF-D10 Traçabilidade: ao selecionar um elemento, exibir evidências (trechos de PDF/frames) em painel lateral.
- RF-D11 IA no Canvas: comandos de linguagem natural aplicados ao que está selecionado (ex.: “duplique e conecte por gateway exclusivo”). A IA aplica patches no BPMN_JSON, atualiza o XML e reimporta

D3. Requisitos de Usabilidade & Acessibilidade

- Snapping e guias de alinhamento sempre ativos; grid opcional.
- Teclado: foco navegável; atalhos para criar/apendar (N/A), desfazer/refazer (Ctrl/Cmd+Z/Y).
- A11y: propriedades e ações acessíveis por teclado; contraste AA; *tooltips* descritivas.
- Desempenho: pan/zoom fluido com 200+ elementos; minimizar *reflows*.

D4. Critérios de Aceite (drag-and-drop)

- Criar por arraste: do palette para o canvas → elemento é criado, snap alinhado, e já permite append por context pad/atalho (A).
- Mover e reconectar: ao arrastar uma tarefa, as conexões re-roteiam; ajustar bendpoints é possível via *handles*.
- Multi-seleção: arrastar seleção, context pad multi-elemento visível, distribuir/alinha com um clique.
- Edição guiada: ao selecionar um elemento e pedir “converter em gateway XOR”, a alteração acontece e o lint não gera erro crítico.
- Export/Import: salvar como BPMN XML, reabrir o arquivo e preservar o layout manual.

D5. Limites & Notas

- Pools/Lanes: ELK.js é o padrão para layout lane-aware e message flows; casos extremos podem exigir retoques manuais. Disponibilizar “Fixar seleção” e “Reaplicar layout (ELK)”.

EPIC E — Versionamento, Comparação & Auditoria

- Versionar diagrama + artefatos (commit-like).
- Diff visual (GED aproximado por nós/arestas alterados) e changelog legível.
- Histórico auditável de prompts/comandos e alterações.

EPIC F — Avaliação & Métricas embutidas

- Harness de avaliação com GED/RGED (baseline × gerado), conforme o paper.
- Testes de edição (sucesso/fracasso por tipo de comando).
- Telemetria: duração, custo/token, taxa de retrial/erro.

EPIC G — Entrega & Operação

- BYOK (hospedado: chaves no navegador, nunca salvas no servidor).
- Docker Compose (FastAPI + worker + Postgres + pgvector) e deploy simples (Render/Railway/Fly).
- Observabilidade (logs estruturados, tracing básico, alertas de custo).

6. Fora de Escopo (MVP 1.0)

- Input de áudio/vídeo para o modelo.
- Mineração de processos por logs (event logs) e conformance checking avançado.
- Execução BPMN (motor de processos) e automação RPA.
- Conectores enterprise (SharePoint/Confluence/Drive/Slack) — deixar para 1.1+.
- Otimizações estéticas avançadas de layout (além do suporte padrão do ELK.js) e roteamento ortogonal complexo 100% automático.

7. Fluxos Principais (MVP)

- “Do arquivo ao BPMN”: subir PDF → RAG extrai passos → agente sintetiza → JSON→XML → editor abre com layout (ELK.js).
- “Do desenho à norma”: subir imagem de fluxograma → visão extrai → sintetiza BPMN → avalia/linta → editor.

- “Versionar e comparar”: salvar versão, gerar diff (GED) e exportar .bpmn.

8. Requisitos Funcionais (detalhados)

RF-A. Ingestão
- A1: Upload de {PDF/DOCX/TXT/PNG/JPG} até 30 MB/arquivo.
- A2: Parsing com Docling (preferido) ou Unstructured + PyMuPDF; OCR quando necessário.
- A3: Normalizar e persistir *artefatos* com metadados (página e tipo).

RF-B. RAG
- B1: Indexação semântica no pgvector (HNSW/IVFFlat) com filtros.
- B2: Resultados com citações (arquivo/página).
- B3: Modo *grounded* para o agente de síntese (passos fundamentados).

RF-C. Multiagente
- C1: Orquestração LangGraph; estados persistidos; *human-in-the-loop*.
- C2: Política de re-tentativas e *rollback* por versão interna JSON.

RF-D. Editor Visual (ver seção EPIC D para detalhes do drag-and-drop)
- D1: bpmn-js (v18.x) como modeler; bpmn-moddle p/ I/O BPMN 2.0.
- D2: Comandos do copiloto com *tool-calls* que alteram o XML e reimportam no canvas.
- D3: Layout (ELK.js) como padrão; “Fixar seleção” (pins) para respeitar posições manuais; fallback interno simples (grid) apenas em emergência.
- D4: Exportar `.bpmn` + `.png` + `.json` interno.
- Sincronização Editor↔IA: toda edição manual persiste o XML, atualiza o BPMN_JSON “vivo” e registra versão; os próximos comandos da IA operam sobre esta versão.

RF-E. Versionamento & Auditoria
- E1: Versões com mensagem (tipo git).
- E2: Diff visual (nós/arestas) + métrica GED.
- E3: Trilhas (quem/quando/o que) e histórico de prompts.
- RF-F. Avaliação
- F1: *Runner* de *test cases* (entrada → BPMN) medindo GED/RGED.
- F2: Painel de qualidade (por fonte: PDF, imagem).

9. Requisitos Não Funcionais

- Desempenho: 95º percentil ≤ 60 s (pipeline doc→BPMN ≤10p).
- Segurança: BYOK (no browser) por padrão; TLS; logs sem PII; *rate limiting*.
- Confiabilidade: *Retries* de modelo; *dead letters* no worker; *healthchecks*.
- Compatibilidade: BPMN 2.0 (bpmn-moddle); layout por ELK.js (layered/hierárquico).
- Acessibilidade: teclado/atalhos no editor; contraste AA.

- i18n: PT-BR/EN no mínimo.

10. Arquitetura de Referência (MVP 1.0)

- Front-end: Next.js + TypeScript + Tailwind + shadcn/ui (UI moderna) + bpmn-js.
- Back-end: FastAPI (coerente com o paper/protótipo), Celery/RQ/BullMQ equivalente p/ jobs, PostgreSQL + pgvector, MinIO/S3 p/ mídia.
- Agentes: LangGraph; serviços de LLMs suportados (OpenAI/Anthropic/Gemini/Fireworks).
- Layout: ELK.js (layered; pools/lanes como contêineres e ports para message flows).

Diagrama (texto):
Usuário → Next.js (chat/copiloto + canvas bpmn-js) → FastAPI (REST) →
(1) Ingestão (Docling/Unstructured/PyMuPDF) →
(2) Vetor/Index (pgvector) →
(3) Multiagente (LangGraph: síntese→JSON, lint→XML) →
(4) Layout (ELK.js) →
(5) Retorno XML + preview no bpmn-js →
(6) Persistência: modelos, versões, artefatos, métricas.
(7) Edições no editor disparam /sync: saveXML → parse → atualizar BPMN_JSON “vivo” e versão.

Observações técnicas chaves:
- bpmn-js está ativo e evoluindo (v18.9.0 recente).
- ELK.js adotado para pools/lanes e message flows; considerar custo/latência em diagramas muito grandes; manter pins e cache de layout.
- JSON interno para edição aumenta robustez; converter a BPMN XML só no final (paper).

11. Modelo de Dados (essencial)

- ProcessModel: id, name, owner, created_at.
- ModelVersion: id, process_model_id, bpmn_xml, internal_json, changelog, metrics(GED etc.).
- Artifact: id, type(pdf, img), storage_url, meta(page), text_content.
- EmbeddingChunk: id, artifact_id, chunk, vector, meta.
- AuditEntry: actor, action, payload(redacted), ts.

12. UX / UI

- Dashboard (projetos recentes, botão “Criar do arquivo”).
- Studio (split view - possível aumentar e diminuir o tamanho de cada janela do studio):
  - esquerda: Editor bpmn-js com minimap e propriedades,
  - direita: Copiloto (chat com ações), abas de Artefatos e Citações; botão

- Layout (ELK.js) com tooltip: funcionamento, trade-offs, “Fixar seleção” e “Reaplicar layout (ELK)”.
- Diff View: lista de mudanças + sobreposição no canvas.
- Cores: neutras/tecnológicas, alta legibilidade; componentes shadcn/ui.
- Referências de toolkit: bpmn-js (embed, walkthrough, releases).

13. Critérios de Aceite (amostra)

- Dado um PDF de 5-10 páginas (processo de onboard simples), ao clicar “Gerar”, um diagrama BPMN é aberto no editor em ≤ 60s, com ≥ 70% de correspondência (GED/RGED) × gabarito.
- Dada a instrução “divida a tarefa ‘Validar dados’ em duas com gateway exclusivo”, o copiloto executa com sucesso e o layout permanece legível.
- Upload de imagem de fluxograma simples → processo BPMN com tarefas, eventos e gateways equivalentes (quando inferível).
- Exportações .bpmn e .png funcionam e reimportação do `.bpmn` mantém a estrutura.
- Após uma edição manual (mover/trocar um gateway), o próximo comando do copiloto considera exatamente essa alteração (sincronismo Editor↔IA comprovado).

14. Segurança & Privacidade

- BYOK no front (chaves não saem do navegador).
- Criptografia em trânsito; mascaramento de PII leve; retenção configurável; *rate limiting* nas rotas do copiloto.
- *Flags* para rodar todo processamento on-prem (posterior à prova do MVP).

15. Roadmap sugerido (sequência, sem datas)

1. Fundação: parsing + pgvector + LangGraph esqueleto + JSON↔XML.
2. Geração v1: do corpus RAG + imagem → BPMN; editor bpmn-js integrado; exportações.
3. Edição guiada: copiloto com ações frequentes; *auto-layout* com guardrails.
4. Qualidade & Métricas: harness de GED/RGED; diff/versão; telemetria/custo.
5. Polimento UX: atalhos, estados de erro, hints, exemplos.
6. Pilotos com 3-5 processos típicos (Finanças, Suprimentos, RH).

16. Riscos & Mitigações

- ELK.js: custo/latência e casos extremos (muitos pools/lanes/message flows).
- Alucinação na síntese → *grounding* forte via RAG + validações de grafo (linter).
- Custo de modelos multimodais → *fallbacks* (combinar modelos), *batching* e cortes por confiança.
- Pins, cache por hash do grafo, debounce de execução e opção de layout incremental/por seleção; permitir retoque manual com grid/snap.

17. Anexos (operacionais para vibe coding)

A. rules.md (trecho exemplo) — para o agente/coder:

- “Minimizar difs; mudanças pequenas por PR.”
- “NUNCA logar segredos.”
- “Rate limit em todas as rotas públicas.”
- “Converter para **JSON interno** antes de editar; exportar XML no final.”

B. Prompts base (trechos)

- *System* do Agente de Síntese: “Você recebe artefatos (texto e recortes de imagens) e produz um BPMN_JSON válido seguindo o schema X… Depois converta para BPMN XML.”

C. Esquema JSON interno (resumo)

```
{
 "process": {"id":"Process_1","name":"..."},
 "elements":[
{"id":"Task_1","type":"task","name":"..."},
{"id":"Gateway_1","type":"exclusiveGateway"}
],
 "flows":[{"source":"Task_1","target":"Gateway_1","type":"sequenceFlow"}]
}
```

D. API (rascunho)

- POST /ingest (arquivo) → artifact_id
- POST /generate ({artifact_ids[]}) → model_version_id
- POST /edit ({model_version_id, command}) → nova versão
- GET /export (model_version_id, fmt: xml|png|json)

E. Referências técnicas do editor: **bpmn-js** doc/walkthrough/releases.

18. Base técnica atual confirmada no repositório

- Core features: criação/edição/explicação, drag-and-drop de `.bpmn`, visão (OpenAI), BYOK no front hospedado, subconjunto de elementos suportados.

19. Referências usadas

- Repositório bpmn-assistant (funcionalidades atuais, BYOK, visão/elementos suportados). (https://github.com/jtlicardo/bpmn-assistant/tree/main)
- bpmn-js (toolkit, walkthrough, releases recentes 18.x). (https://bpmn.io/toolkit/bpmn-js/)
- ELK.js / elkjs (Eclipse Layout Kernel para JS) — layout layered com hierarquia/ports.(https://eclipse.dev/elk/)
- BPMN Assistant – paper (arXiv): arquitetura FastAPI+Vue, JSON interno vs. XML, métricas GED/RGED. (https://ar5iv.org/abs/2509.24592)
- RAG: pgvector em Postgres; visão geral LlamaIndex. (https://github.com/pgvector/pgvector)