Documento de Requisitos do Sistema (DRS)

Projeto: Vida RPG — Gerenciador de Rotina Gamificado com IA Versão: 0.1 Autores: Joelson + Assistente Data: 2025-10-11

---

1. Visão Geral Um gerenciador de vida gamificado que transforma tarefas e rotinas em missões narrativas personalizadas. O sistema usa IA para gerar histórias, recompensas e sugestões adaptativas com base no contexto e comportamento do usuário.

One-liner Transforme sua rotina em uma jornada RPG com narrativas e recompensas geradas por IA adaptadas ao seu contexto.

---

2. Objetivos do Sistema Engajar usuários na execução de tarefas por meio de narrativa e progressão. Personalizar missões e recompensas usando modelos de linguagem (ex: Gemini/GPT). Servir como produto de portfólio com documentação completa e qualidade de engenharia.

---

3. Personas 1. Lucas — Estudante (22): precisa de disciplina pra estudar; gosta de estética medieval e premiações visuais. 2. Mariana — Jovem Profissional (28): busca produtividade, prefere recompensas práticas (ex: foco, tempo livre). 3. Coach Rafael — Profissional (35): usa a plataforma com clientes; quer criar capítulos e acompanhar progresso.

---

4. Requisitos 4.1 Requisitos Funcionais (RF) RF01: Usuário pode se registrar e autenticar (email, Google OAuth). RF02: Usuário pode criar, editar e excluir tarefas (type: diária, única, recorrente). RF03: Ao completar tarefa, sistema concede XP e atualiza streaks. RF04: Sistema apresenta Dashboard com XP, nível, streak, progresso do capítulo. RF05: Admin/Editor cria capítulos narrativos e quests base. RF06: Módulo IA — gerar capítulo/narrativa personalizada por usuário. RF07: Módulo IA — sugerir recompensas personalizadas conforme perfil e engajamento. RF08: IA Coach — chat contextual com recomendações e motivação. RF09: Logs de IA (prompt, response, userId, timestamp) guardados para auditoria. RF10: Cache de respostas IA para evitar regeneração frequente.

4.2 Requisitos Não-Funcionais (RNF) RNF01: Latência do dashboard < 500ms (cache onde aplicável). RNF02: Sistema deve suportar 10k usuários iniciais com crescimento escalável. RNF03: Respostas de IA armazenadas com TTL configurável (ex: 7 dias). RNF04: Segurança: senhas hashed, tokens JWT com expiração, proteção CSRF onde aplicável. RNF05: Privacidade: logs de IA devem mascarar dados sensíveis.

---

5. Escopo MVP 1. Autenticação (email + Google). 2. CRUD de tarefas (diárias, únicas, recorrentes). 3. XP e nível básico. 4. Dashboard com progresso e streak. 5. Admin simples para capítulos (texto estático). 6. Endpoint IA simples: gerar capítulo baseado em template e dados do usuário (cacheado). 7. Deploy (Vercel front / Render back) + README + DRS mínimo.

---

6. Arquitetura (visão alta) Frontend: Next.js + TypeScript + Tailwind + React Query Backend: Node.js + Express/NestJS + TypeScript DB: MongoDB (inicial) ou PostgreSQL + Prisma (opcional) AI Layer: Módulos que chamam API externa (Gemini / OpenAI) Deploy: Vercel (front) + Render/Heroku (back) Fluxo: Front → API → AI Module (cache) → DB → Front

---

7. Módulos de IA (definição e contratos) 7.1 Story Engine Entrada: userProfile, userProgress, userPreferences, contextSummary Saída: { chapterTitle, chapterSummary, quests[] } Endpoint: POST /ai/story/generate Observações: Responses cacheadas por par (userId, chapterVersion). Prompts versionados.

7.2 Reward Engine Entrada: userProfile, completionHistory, preferences Saída: [{ rewardId, type, label, effect }] Endpoint: POST /ai/rewards/suggest

7.3 AI Coach (chat) Entrada: chatHistory, userContext Saída: conversation reply + suggested micro-actions Endpoint: POST /ai/coach/chat

---

8. Modelo de Dados (rascunho) User { id, name, email, passwordHash, level, xp, coins, preferences, createdAt } Task { id, userId, title, description, type, xp, status, recurrence, createdAt } Chapter { id, title, description, createdBy, version, quests[] } Quest { id, chapterId, title, xp, prereqQuestId? } AIResponseCache { id, userId, type, inputHash, responseJson, createdAt, ttl } AIInteractionLog { id, userId, module, promptHash, responseHash, timestamp }

---

9. Endpoints APIs (inicial) POST /auth/register POST /auth/login GET /users/me CRUD /tasks POST /tasks/:id/complete GET /dashboard GET /chapters POST /ai/story/generate POST /ai/rewards/suggest POST /ai/coach/chat

---

10. Prompt Base (exemplo rápido) Story Engine - prompt template (protótipo): User profile: {name, preferences, level, xp, recentActivity} Goal: Generate a chapter title, summary and 3 quests aligned with the user's current priorities. Output JSON. Tone: motivational, short, and narrative. Use user's preferred theme: {theme}.

---

11. Segurança e Custos de IA Implementar rate limiting e caching para reduzir custos. Monitoramento de consumo de tokens e orçamentos. Armazenar prompts e respostas essenciais para debugging e regressão.

---

12. Backlog Inicial (épicos e user stories) Épicos EPIC-Auth: Autenticação e perfil EPIC-Tasks: CRUD de tarefas e sistema de XP EPIC-Missions: Chapters & Quests EPIC-AI: Story Engine, Reward Engine, Coach EPIC-Admin: Painel de criação de capítulos

User Stories (exemplos) US-001: Como usuário, quero me registrar com Google para acessar minhas missões. US-002: Como usuário, quero criar tarefas recorrentes para automatizar missões diárias. US-003: Como usuário, ao completar tarefa, quero ganhar XP e ver meu nível atualizado. US-010: Como usuário, quero que a IA gere um capítulo personalizado baseado nos meus objetivos.

---

13. Roadmap de Sprints (sugestão) Sprint 0 (setup): repo, infra, DB, CI básico — 3 dias Sprint 1: Auth + Tasks CRUD + XP system — 7 dias Sprint 2: Dashboard + Chapters static + Admin basic — 7 dias Sprint 3: IA Story Engine integration (prototype) — 10 dias Sprint 4: Reward Engine + Coach prototype — 10 dias Sprint 5: Polimento, testes e deploy final — 7 dias

---

14. Métricas & Telemetria DAU/WAU, retention D7, tasksCompletedPerUser, avgSessionTime, AI calls per user, cost per 1k AI tokens.

---

15. Entregáveis para a sessão de 1h30 DRS mínimo pronto (este documento). Backlog inicial com épicos e ~15 user stories. Modelo de dados rascunho e endpoints. Prompt base protótipo para Story Engine.

---

16. Próximos passos sugeridos 1. Escolher stack final (Mongo vs Postgres) — impacto no modelo de dados. 2. Definir tema/narrativa inicial (influencia prompts). 3. Configurar ambiente dev + secrets para IA. 4. Implementar Sprint 0 e seguir roadmap.

Fim do DRS v0.1