AxonAI - Documento de Iniciação de Projeto

1. Objetivo, Escopo e Filosofia do Produto

- **Problema Resolvido:** Resolve a perda de contexto e a ineficiência do fluxo de trabalho ao usar LLMs para tarefas de desenvolvimento de software.
- **Utilizador Final:** Desenvolvedores de software, tanto individuais quanto em pequenas equipas.
- **Visão do Produto:** Criar um ambiente de execução de projetos que integre nativamente a ideação assistida por IA com a execução estruturada de tarefas, validando que um fluxo de trabalho com contexto focado aumenta significativamente a produtividade.
- **Filosofia de Engenharia:** Este projeto será guiado por princípios de engenharia de software que visam a construção de um sistema robusto, manutenível e seguro.
    - **Arquitetura Limpa e Testável:** O design favorece o baixo acoplamento e a alta coesão, com uma separação rigorosa entre o núcleo de negócio e a infraestrutura, garantindo a testabilidade e a longevidade do sistema.
    - **Segurança por Design:** A segurança não é uma camada adicional, mas um pilar da arquitetura, com padrões robustos implementados em todas as camadas.
    - **Escalabilidade e Resiliência:** A arquitetura é desenhada para ser operada em produção, com componentes que suportam escalabilidade e políticas de resiliência para lidar com falhas.

2. Arquitetura e Plataforma de Execução

- **Padrão Arquitetural:** Monolito Modular aplicando os princípios da Arquitetura Hexagonal (Ports and Adapters).
    - **Justificativa:** Garante uma organização de código e separação de responsabilidades de alto nível. Isola o núcleo de negócio da infraestrutura, permitindo a extração futura para microsserviços, se necessário.

- **Tecnologias Principais:**
  - **Backend:** Java 21 LTS, Spring Boot 3.5.5, Maven.
  - **Frontend:** React 19, Vite.
  - **Base de Dados:** PostgreSQL.
- **Estratégia de Implantação (Deployment):**
  - **Plataforma:** Render (PaaS - Plataforma como Serviço).
  - **Justificativa:** Abstrai a complexidade de infraestrutura, oferecendo "Git push-to-deploy", gestão integrada de bases de dados, gestão de segredos e escalabilidade simplificada.
- **Gestão de Configuração:**
  - **Método:** Utilização de perfis do Spring Boot e variáveis de ambiente para todas as configurações, incluindo segredos e URLs de conexão.
  - **Justificativa:** Padrão nativo do Spring que se integra perfeitamente com a gestão de segredos de plataformas PaaS.
- **Documentação da API:**
  - **Padrão:** OpenAPI 3 com a biblioteca `springdoc-openapi`.
  - **Justificativa:** Gera automaticamente uma especificação OpenAPI a partir dos controllers, garantindo que a documentação esteja sempre atualizada e fornecendo uma UI Swagger para testes interativos.

3. Arquitetura de Software de Referência

A arquitetura hexagonal é implementada com uma separação rigorosa de responsabilidades.

- **Domínio e Persistência:**
  - **Estratégia:** O Modelo de Domínio é puro e isolado, contendo a lógica de negócio. A camada de persistência é um `Adapter` que utiliza Entidades JPA e Mappers para traduzir os objetos de domínio para o formato do banco de dados, implementando os `Repository Ports`.
  - **Justificativa:** Desacopla totalmente a lógica de negócio da tecnologia de persistência, alinhando-se com os princípios da Arquitetura Hexagonal.
- **Camada de Interação com IA:**

    - **Estratégia:** A interação com a IA é gerida por um pipeline de componentes (`AIFacade`, `PolicyEngine`, `ProviderRouter`, `ApiKeyVault`), cada um com responsabilidades específicas.
    - **Justificativa:** Esta abordagem modular e extensível substitui um serviço unificado, permitindo a implementação de políticas de segurança, governança e resiliência de forma mais robusta e manutenível.
- **Controle de Custos de IA:**
    - **Estratégia:** Um `PolicyEngine` irá gerir o uso com base em "unidades de IA", com custos diferentes por tipo de operação e limites associados a planos de subscrição.
    - **Justificativa:** Cria um mecanismo de controle de custos granular, justo e escalável, essencial para a viabilidade do produto.

4. Estratégia de Segurança

A segurança é implementada com padrões de nível de produção.

- **Autenticação:**
    - **Estratégia: Refresh Token Rotation** com `Access Tokens` (JWT RS256) de curta duração e `Refresh Tokens` opacos de longa duração.
    - **Justificativa:** Oferece um mecanismo robusto para gestão de sessão e revogação, mitigando os riscos de roubo de tokens.
- **Segurança das Chaves BYOK:**
    - **Estratégia: Envelope Encryption** com um KMS (Key Management Service) dedicado.
    - **Justificativa:** Garante a proteção das chaves de API dos utilizadores com um padrão de segurança empresarial, separando a gestão de chaves da aplicação.
- **Segurança da Aplicação (Essenciais):**
    - **Hashing de Senha:** Bcrypt é mandatório.
    - **Proteção de Input:** Validação rigorosa de input e proteção contra SSRF são implementadas para mitigar riscos elevados.

5. Arquitetura Frontend

- **Gestão de Estado:**
  - **Estado do Servidor:** TanStack Query para automatizar o fetching, caching e sincronização de dados da API.
  - **Estado Global do Cliente:** Zustand para um estado global minimalista e de alto desempenho.
- **Biblioteca de Componentes:** Shadcn/ui ou Mantine para um desenvolvimento de UI rápido e de alta qualidade.
- **Qualidade e Testes:** A arquitetura inclui estratégias definidas para testes (unitários, integração e E2E), qualidade de código (ESLint, Prettier) e sincronização de tipos com o backend via OpenAPI.

6. Funcionalidades Essenciais e Estado Inicial

- **Funcionalidades Essenciais:**
  1. Autenticação de utilizador (local e Google) com gestão de sessão segura.
  2. Criação e gestão de Workspaces e Projetos.
  3. Mecanismo de "proposta e aprovação" para criação de tarefas via IA.
  4. Visualização do projeto com tarefas e checklists.
  5. "Focus Mode": chat contextual focado num `ChecklistItem`.
- **Estado Inicial do Sistema (Data Seeding):**
  - **Estratégia:** Scripts de migração de schema geridos pelo Flyway e um `CommandLineRunner` do Spring Boot para dados iniciais.
  - **Justificativa:** Garante um processo de inicialização do sistema automatizado, repetível e versionado.

7. Plano de Evolução da Arquitetura

- **Transição do MVP:** Este documento formaliza a superação da "dívida técnica consciente" assumida durante a fase de MVP.
- **Gatilho para Refatoração:** As simplificações da fase de MVP (modelo unificado, segurança pragmática) foram substituídas pelas estratégias

robustas descritas neste documento como parte da transição para um produto de produção.

- **Monitoramento Contínuo:** A arquitetura será continuamente revisada para garantir que continue a atender aos requisitos de escalabilidade, segurança e manutenibilidade do produto.

Fontes