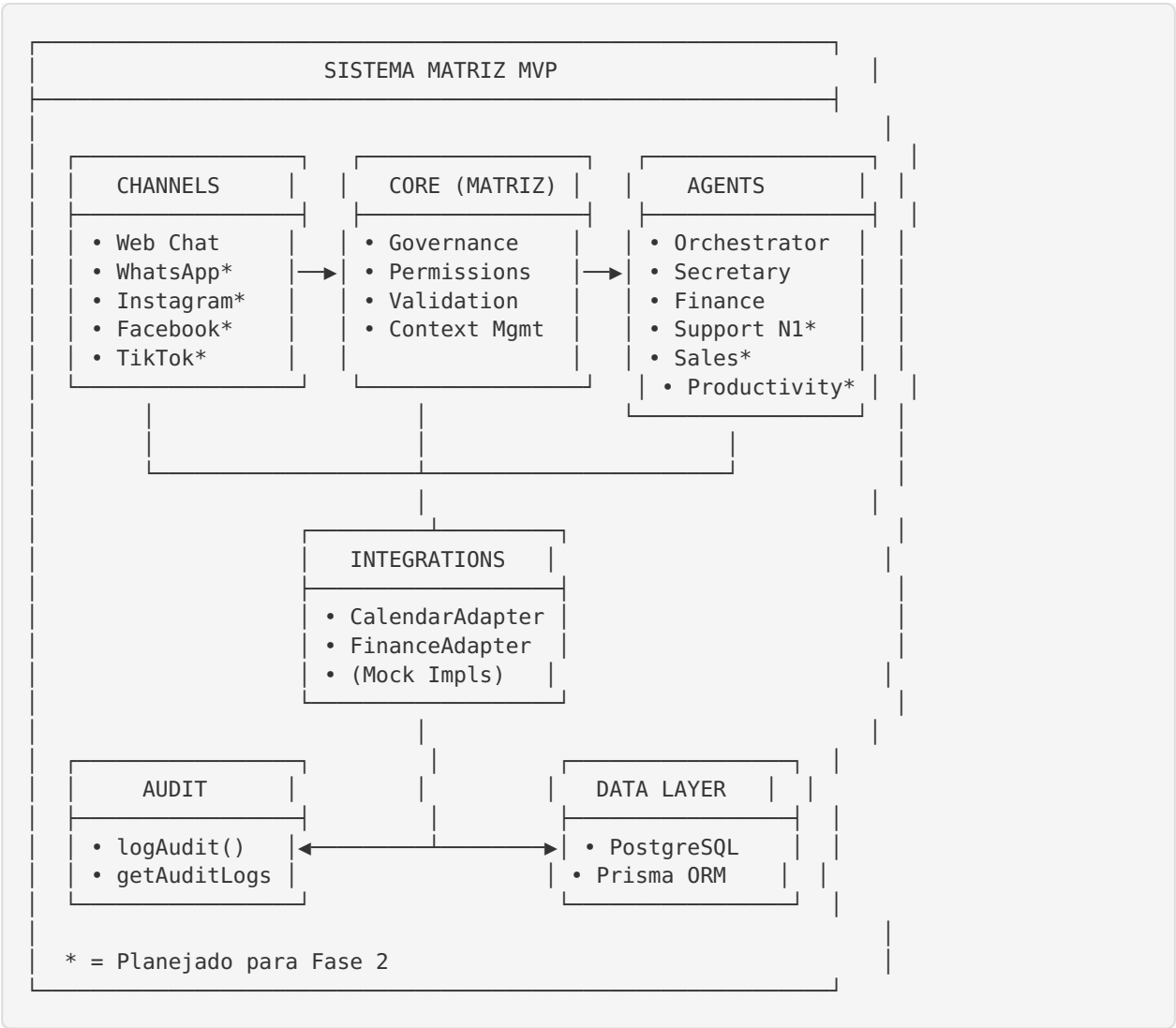


Arquitetura do Sistema Matriz

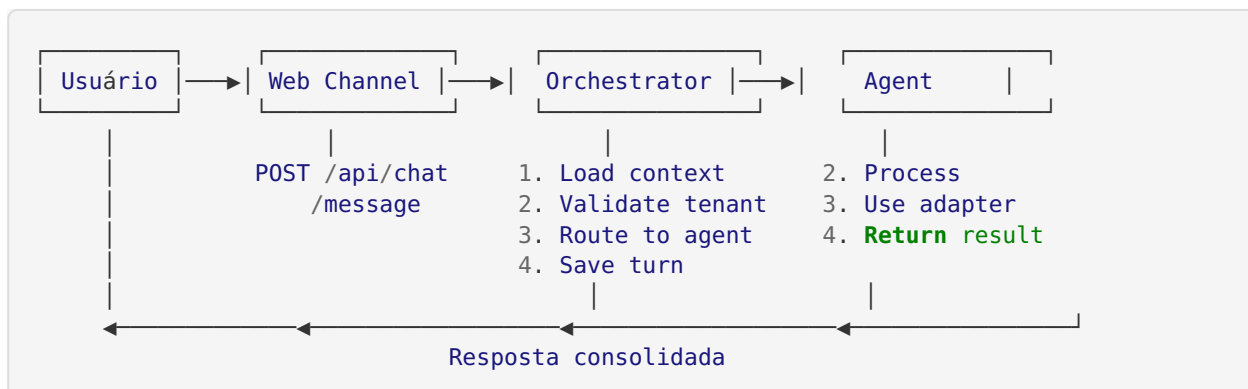
Visão Geral

O Sistema Matriz é uma **plataforma SaaS multi-tenant** que governa um hub universal de agentes de IA. Ele atua como o “source of truth” que controla quais agentes existem, quais integrações estão disponíveis, quais clientes (tenants) podem usar quais recursos, e com quais permissões.

Diagrama de Componentes



Fluxo de uma Mensagem



Passos Detalhados:

1. **Channel Adapter** recebe mensagem do usuário
2. **Orchestrator** carrega contexto do tenant (governança)
3. **Orchestrator** valida permissões do plano
4. **Orchestrator** identifica intenção e roteia para agente
5. **Agente** processa usando **Integration Adapter**
6. **Agente** solicita confirmação se necessário
7. **Audit Service** registra ação crítica
8. **Orchestrator** salva turno e contexto
9. **Channel Adapter** retorna resposta

Separação de Responsabilidades

Core (lib/core)

- **Governança:** Validação de permissões por tenant/plano
- **Contexto:** Gerenciamento de contexto de conversa
- **Repositórios:** Acesso a dados do sistema matriz

Agents (lib/agents)

- **Orchestrator:** Roteamento, handoff, unificação de respostas
- **Secretary:** Lógica específica de agenda
- **Finance:** Lógica específica financeira

Integrations (lib/integrations)

- **Contracts:** Interfaces TypeScript dos adapters
- **Mocks:** Implementações mockadas (salvam no banco)

Audit (lib/audit)

- **Logging:** Registro de ações críticas com before/after

Estratégia Multi-Tenant

Cada tenant possui:

- Configuração de agentes habilitados (`TenantAgentConfig`)
- Configuração de integrações habilitadas (`TenantIntegrationConfig`)

- Plano associado com limites e features
- Dados isolados (todas as tabelas têm `tenant_id`)

Hierarquia de Permissões:

```
Plan (define limites globais)
└─> Tenant (herda do plano)
    └─> TenantAgentConfig (customiza comportamento)
        └─> User (opera dentro das permissões do tenant)
```

Preparação para Microsserviços

A arquitetura modular permite extração futura de:

1. **Serviço de Agentes:** `lib/agents/*` pode se tornar serviço independente
2. **Serviço de Integrações:** `lib/integrations/*` com filas de sincronização
3. **Serviço de Auditoria:** `lib/audit/*` com armazenamento dedicado
4. **API Gateway:** Channel adapters unificados

Tecnologias

- **Framework:** Next.js 14 (App Router)
- **Linguagem:** TypeScript
- **ORM:** Prisma
- **Banco de Dados:** PostgreSQL
- **Autenticação:** NextAuth.js v4
- **Estilização:** Tailwind CSS
- **Animações:** Framer Motion