Documentação de Arquitetura do NeurotrackApp

Introdução

Este documento apresenta a arquitetura completa do NeurotrackApp, uma aplicação projetada para auxiliar pessoas neurodivergentes no gerenciamento de suas atividades diárias, medicações, humor e compromissos. A documentação segue o modelo C4 (Contexto, Contêineres, Componentes e Código) e inclui diagramas complementares que abordam aspectos específicos da arquitetura.

Índice

Visão Geral do Sistema

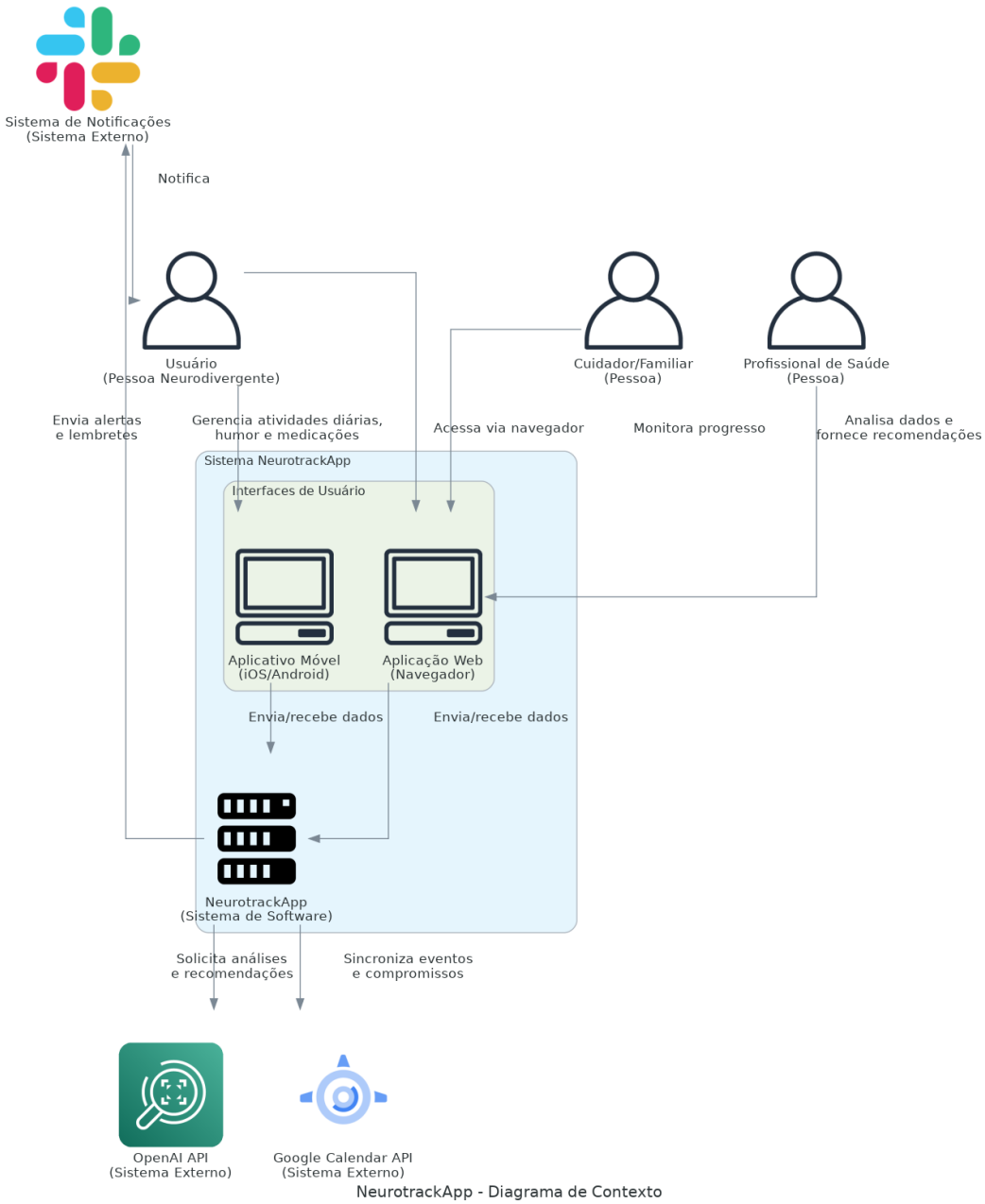
O NeurotrackApp é uma aplicação completa para auxiliar pessoas neurodivergentes no gerenciamento de suas atividades diárias. O sistema é composto por:

- **Aplicativo móvel**: Interface principal para usuários finais (iOS e Android)
- **Aplicação web**: Interface alternativa acessível via navegador
- **Backend API**: Serviços que processam as requisições e gerenciam os dados
- **Bancos de dados**: Armazenamento persistente de informações dos usuários
- **Integrações externas**: Conexões com serviços como Google Calendar e OpenAI

A arquitetura foi projetada seguindo princípios de microserviços, com foco em escalabilidade, segurança e experiência do usuário.

Diagramas de Arquitetura C4

Diagrama de Contexto



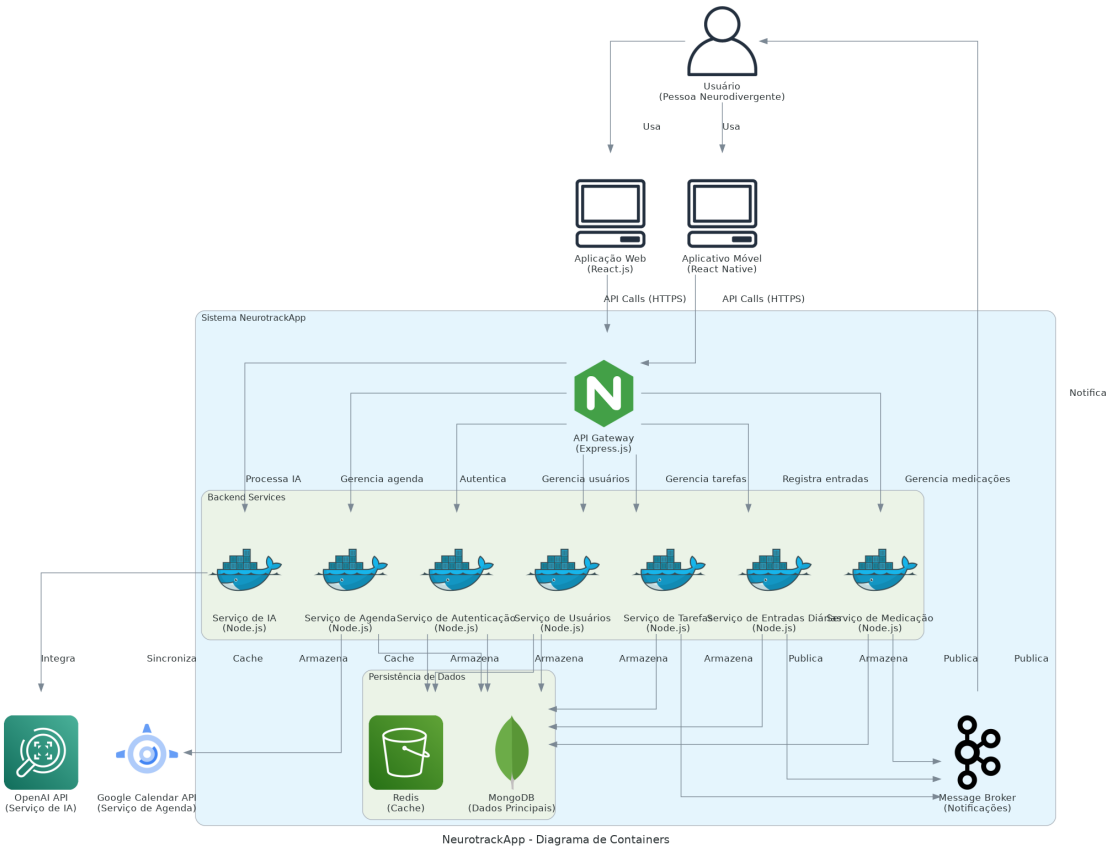
NeurotrackApp - Diagrama de Contexto

O diagrama de contexto mostra a visão de mais alto nível do sistema, identificando:

- **Usuários**: Pessoas neurodivergentes que utilizam o aplicativo
- **NeurotrackApp**: O sistema como um todo
- **Serviços Externos**: Sistemas de terceiros com os quais o NeurotrackApp se integra
- OpenAI API: Fornece capacidades de processamento de linguagem natural e análise
- Google Calendar API: Permite sincronização de eventos e compromissos
- Serviço de Notificações: Envia lembretes e alertas aos usuários
- Serviço de Armazenamento: Armazena dados não estruturados como imagens e anexos

Este diagrama estabelece os limites do sistema e suas interações externas principais.

Diagrama de Contêineres



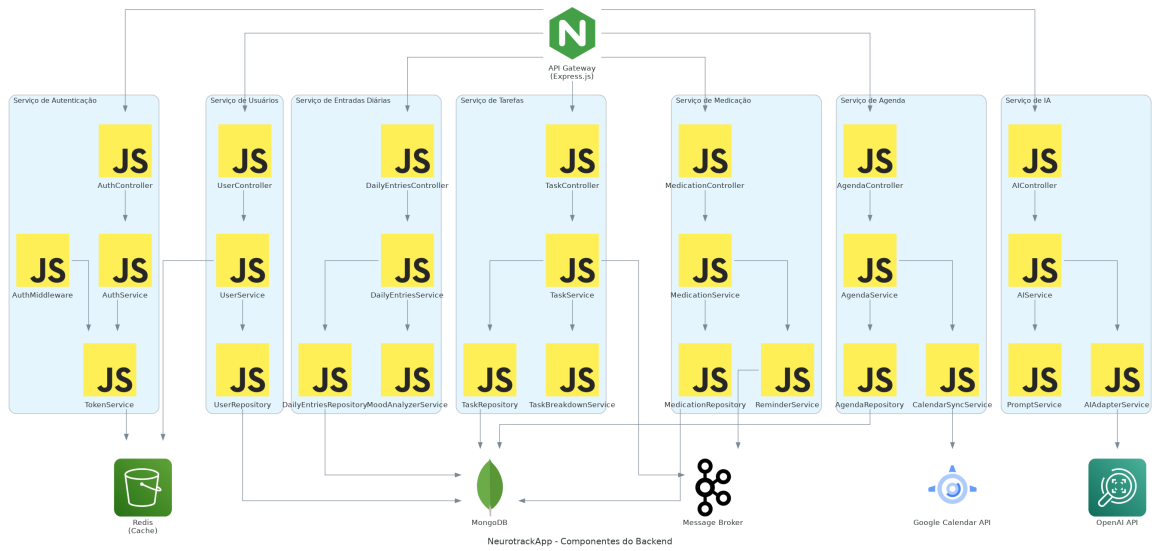
O diagrama de contêineres detalha os principais componentes de alto nível do sistema:

- **Aplicativo Móvel**: Aplicativo nativo para iOS e Android
- **Aplicação Web**: Interface baseada em navegador usando React

- **API Backend**: Serviço RESTful implementado em Node.js
- **Bancos de Dados**:
- MongoDB: Armazenamento principal de dados
- Redis: Cache e gerenciamento de sessões
- **Serviços Externos**: Integrações com sistemas de terceiros

Este nível mostra como os contêineres se comunicam entre si e como os usuários interagem com o sistema.

Diagrama de Componentes do Backend



O diagrama de componentes do backend detalha a estrutura interna da API:

- **Controllers**: Gerenciam as requisições HTTP
- AuthController: Autenticação e autorização
- UserController: Gerenciamento de perfis
- DailyEntriesController: Registro de humor e atividades
- TasksController: Gerenciamento de tarefas
- AIController: Interações com IA
- AgendaController: Gerenciamento de eventos

- MedicationController: Controle de medicações

- **Middlewares**: Processam requisições antes dos controllers

- AuthMiddleware: Validação de tokens
- RateLimiter: Controle de taxa de requisições
- InputValidator: Validação de dados de entrada

- ErrorHandler: Tratamento padronizado de erros

- **Services**: Implementam a lógica de negócio

- UserService: Operações relacionadas a usuários
- TaskService: Operações relacionadas a tarefas
- AIService: Integração com OpenAI
- AgendaService: Gerenciamento de eventos
- GoogleCalendarService: Integração com Google Calendar

- MedicationService: Gerenciamento de medicações

- **Models**: Definem a estrutura dos dados

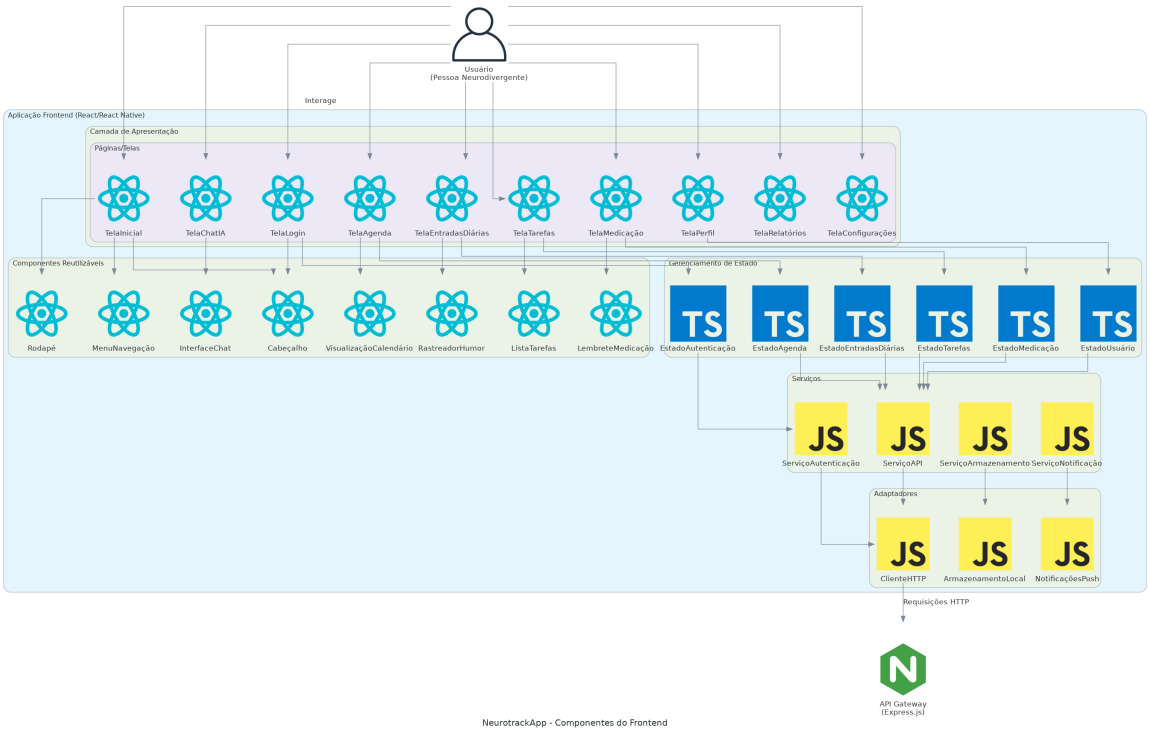
- User: Perfil do usuário
- Task: Tarefas e subtarefas
- DailyEntry: Registros diários
- Event: Eventos de calendário

- Medication: Medicações e lembretes

- **Utils**: Funções utilitárias

- TokenManager: Geração e validação de tokens
- DateUtils: Manipulação de datas
- NotificationService: Envio de notificações
- LoggerService: Registro de logs

Diagrama de Componentes do Frontend

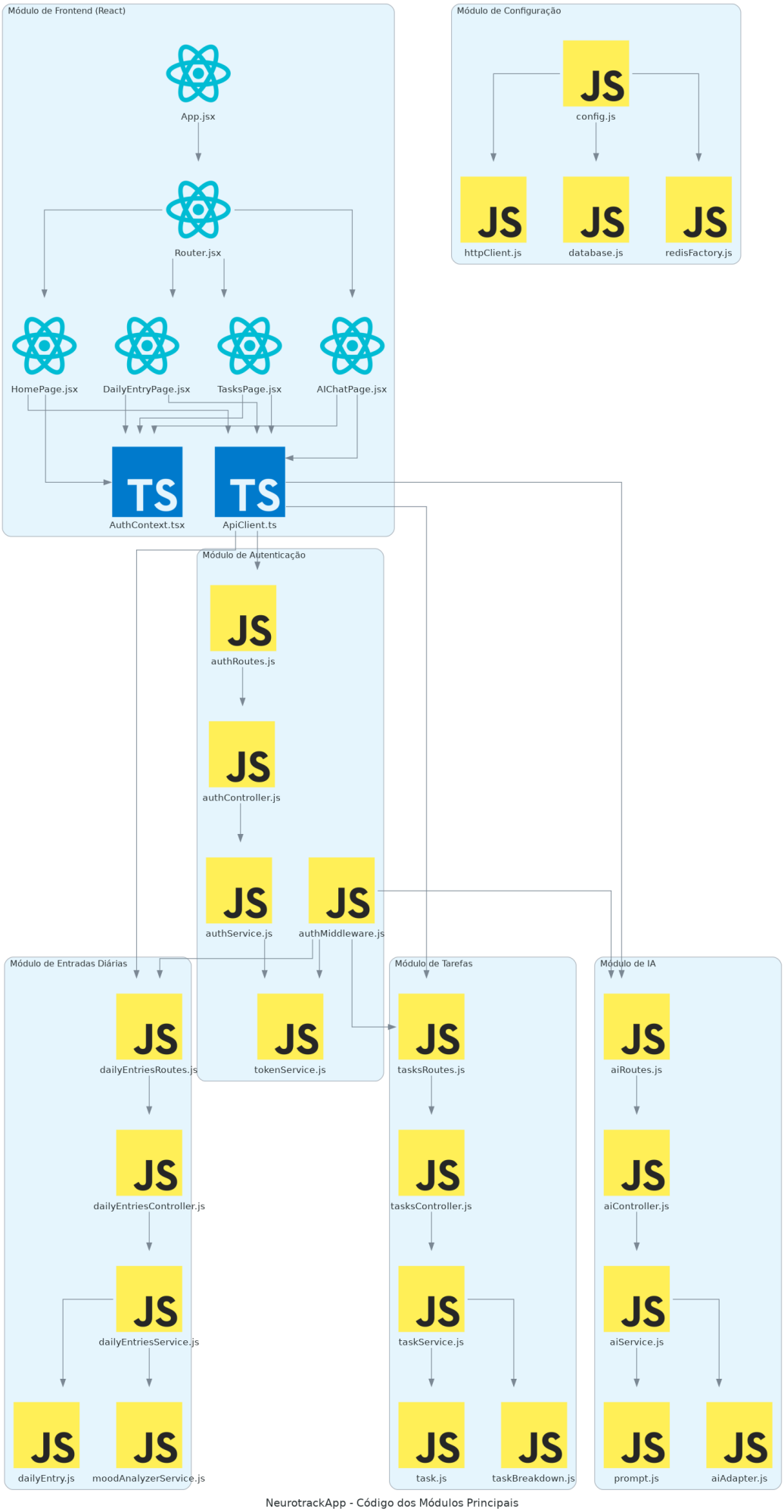


O diagrama de componentes do frontend detalha a estrutura da aplicação móvel e web:

- **Telas/Páginas**: Interfaces de usuário principais
- Login/Registro: Autenticação de usuários
- Dashboard: Visão geral das atividades
- Perfil: Gerenciamento de informações pessoais
- Entradas Diárias: Registro de humor e atividades
- Tarefas: Gerenciamento de tarefas
- Calendário: Visualização e gerenciamento de eventos
- Medicações: Controle de medicações
- Chat IA: Interação com assistente de IA
- **Componentes Compartilhados**: Elementos reutilizáveis
- Header/Footer: Elementos de navegação
- Forms: Componentes de formulário
- Cards: Exibição de informações
- Modals: Janelas de diálogo
- Notifications: Alertas e lembretes

- **Serviços**: Lógica de comunicação com o backend
- ApiService: Cliente HTTP para comunicação com a API
- AuthService: Gerenciamento de autenticação
- StorageService: Armazenamento local
- SyncService: Sincronização de dados offline
- **Estado**: Gerenciamento de estado da aplicação
- Redux/Context: Armazenamento centralizado
- Reducers: Manipuladores de estado
- Actions: Ações que modificam o estado

Diagrama de Código



NeurotrackApp - Código dos Módulos Principais

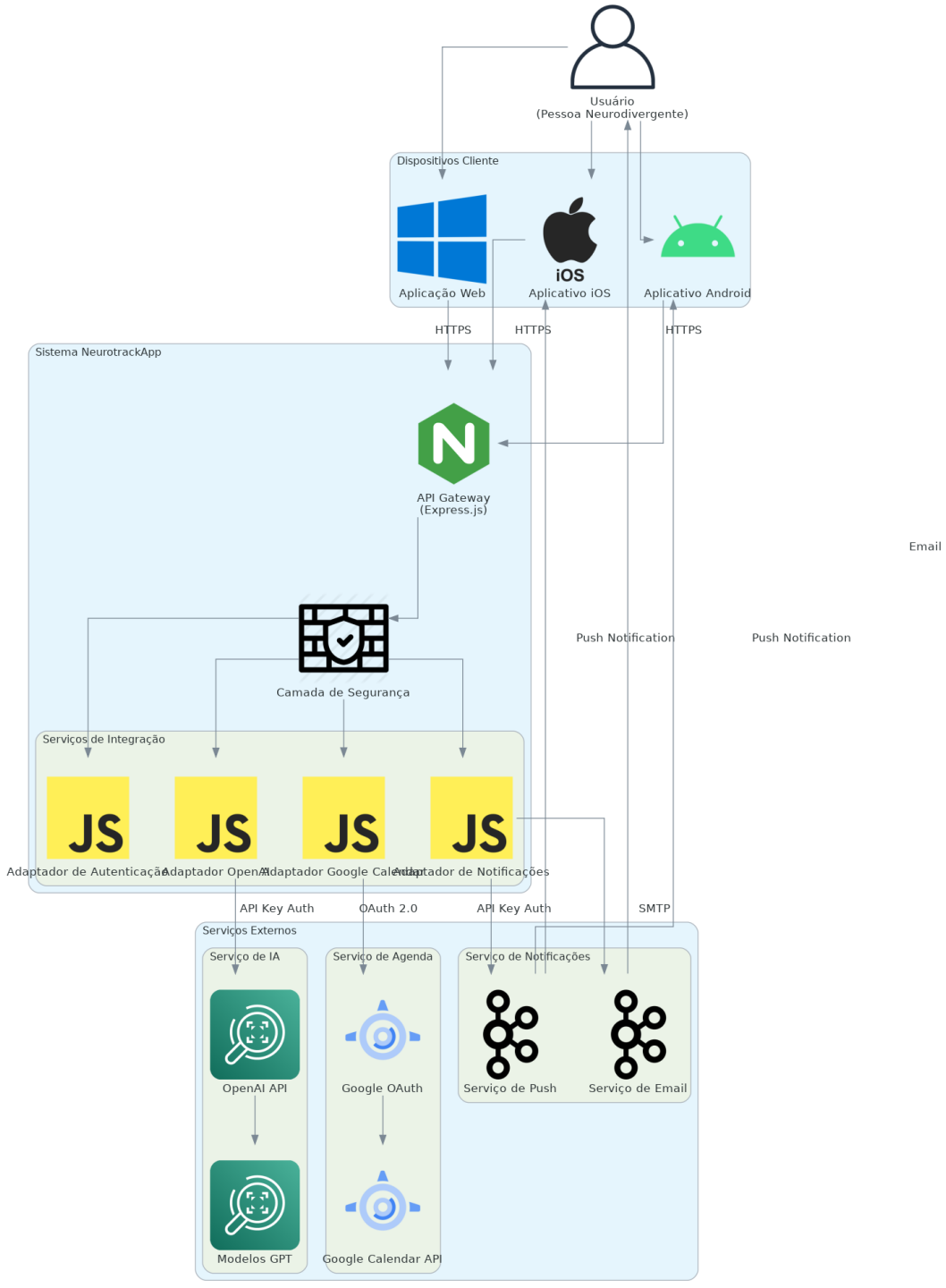
O diagrama de código mostra a estrutura detalhada dos principais módulos do sistema:

- **Estrutura de Diretórios do Backend**:
- `/src/controllers` : Controladores da API
- `/src/middlewares` : Middlewares de processamento
- `/src/services` : Serviços de negócio
- `/src/models` : Modelos de dados
- `/src/utils` : Funções utilitárias
- `/src/routes` : Definições de rotas
- `/src/config` : Configurações do sistema
- **Principais Classes e Funções**:
- `AuthController.js` : Gerencia autenticação
- `TaskService.js` : Implementa lógica de tarefas
- `AIService.js` : Integra com OpenAI
- `User.js` : Define modelo de usuário
- `Task.js` : Define modelo de tarefa
- `authMiddleware.js` : Valida tokens de acesso

Este nível de detalhe é útil para desenvolvedores que precisam entender a implementação específica do código.

Diagramas Complementares

Diagrama de Integração com Serviços Externos

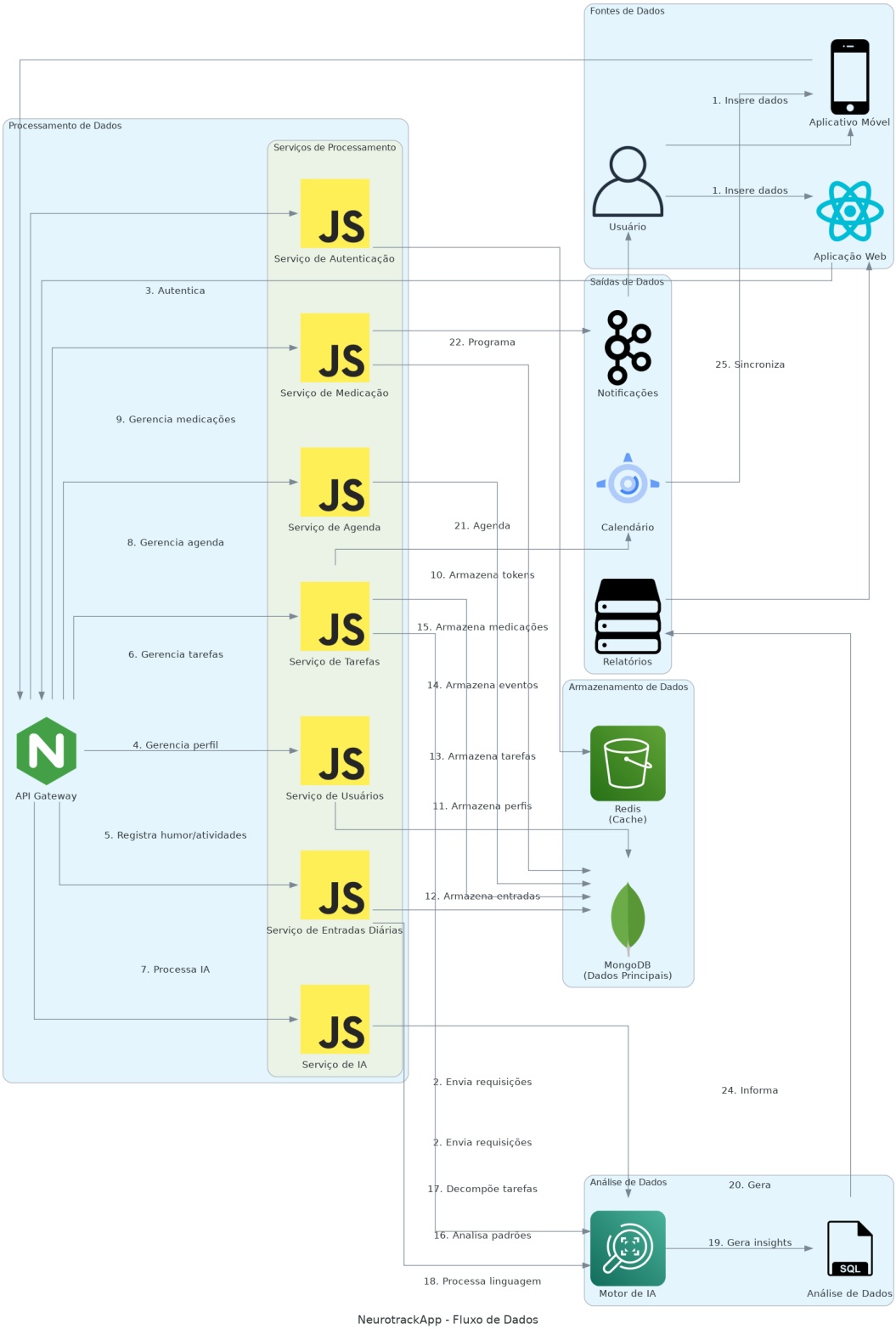


NeurotrackApp - Integração com Serviços Externos

Este diagrama detalha como o NeurotrackApp se integra com serviços externos:

- **Integração com OpenAI API**:
- Análise de entradas diárias para identificar padrões
- Decomposição de tarefas complexas em subtarefas gerenciáveis
- Assistente de IA personalizado para suporte ao usuário
- Geração de insights e recomendações
- **Integração com Google Calendar**:
- Sincronização bidirecional de eventos
- Criação de eventos a partir de tarefas
- Lembretes de compromissos
- Gerenciamento de conflitos de agenda
- **Integração com Serviços de Notificação**:
- Notificações push para lembretes de medicação
- Alertas para tarefas próximas do prazo
- Lembretes de compromissos
- Notificações de bem-estar
- **Integração com Serviços de Armazenamento**:
- Upload de imagens e anexos
- Armazenamento de documentos
- Backup de dados do usuário

Diagrama de Fluxo de Dados



NeurotrackApp - Fluxo de Dados

O diagrama de fluxo de dados mostra como as informações circulam pelo sistema:

1. **Entrada de Dados**:

2. Usuários inserem informações via aplicativo móvel ou web

3. Dados são validados e processados pelos serviços correspondentes

4. **Processamento**:

5. Serviços aplicam lógica de negócio aos dados
6. Middleware garante segurança e consistência

7. Integrações externas enriquecem os dados

8. **Armazenamento**:

9. Dados persistentes são salvos no MongoDB
10. Dados temporários e tokens são armazenados no Redis

11. Anexos são armazenados em serviço de armazenamento

12. **Análise**:

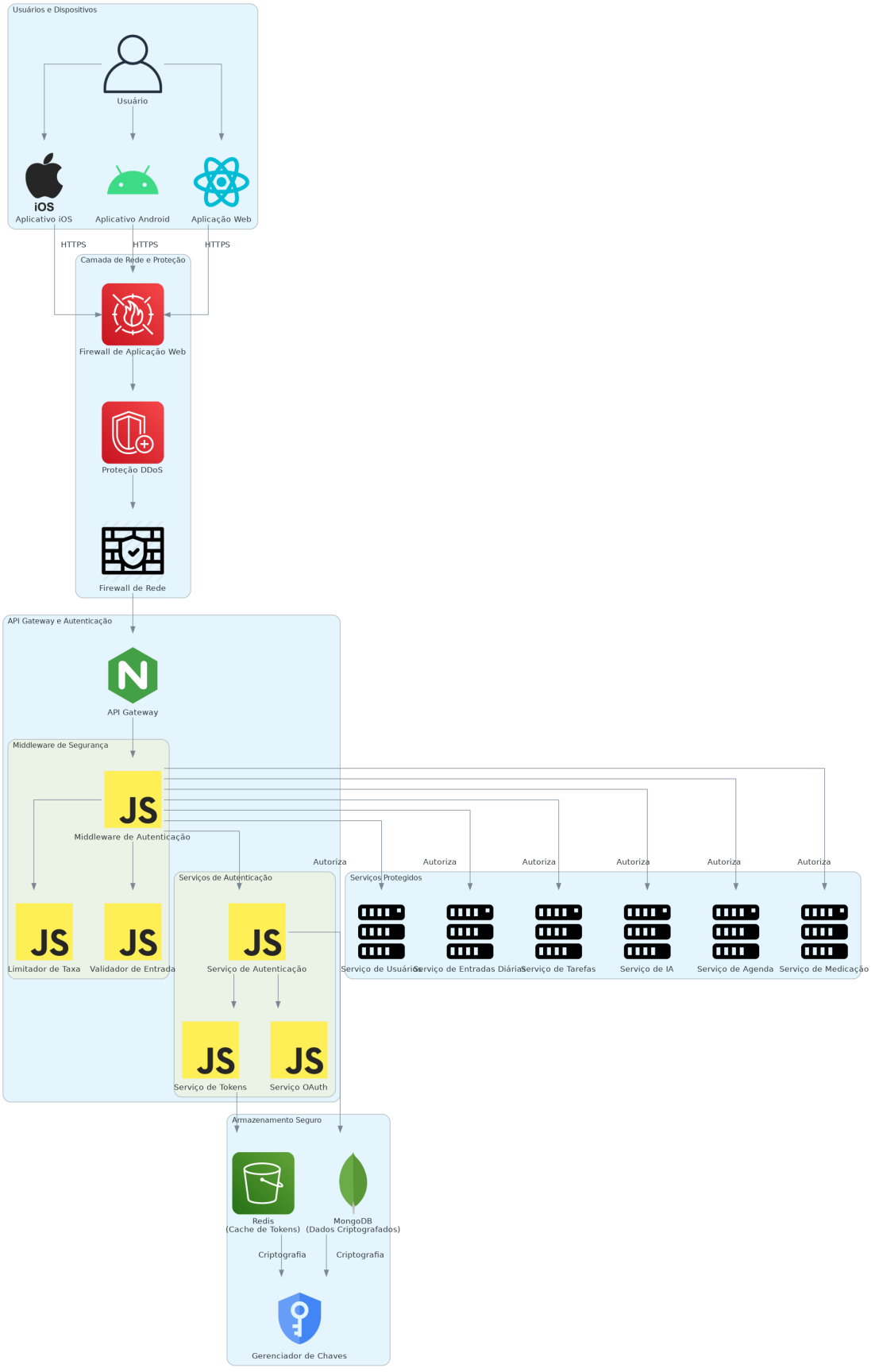
13. Dados de entradas diárias são analisados para identificar padrões
14. IA processa informações para gerar insights

15. Relatórios são gerados a partir dos dados coletados

16. **Saída**:

17. Notificações são enviadas aos usuários
18. Eventos são sincronizados com calendários
19. Relatórios são disponibilizados na interface

Diagrama de Segurança e Autenticação

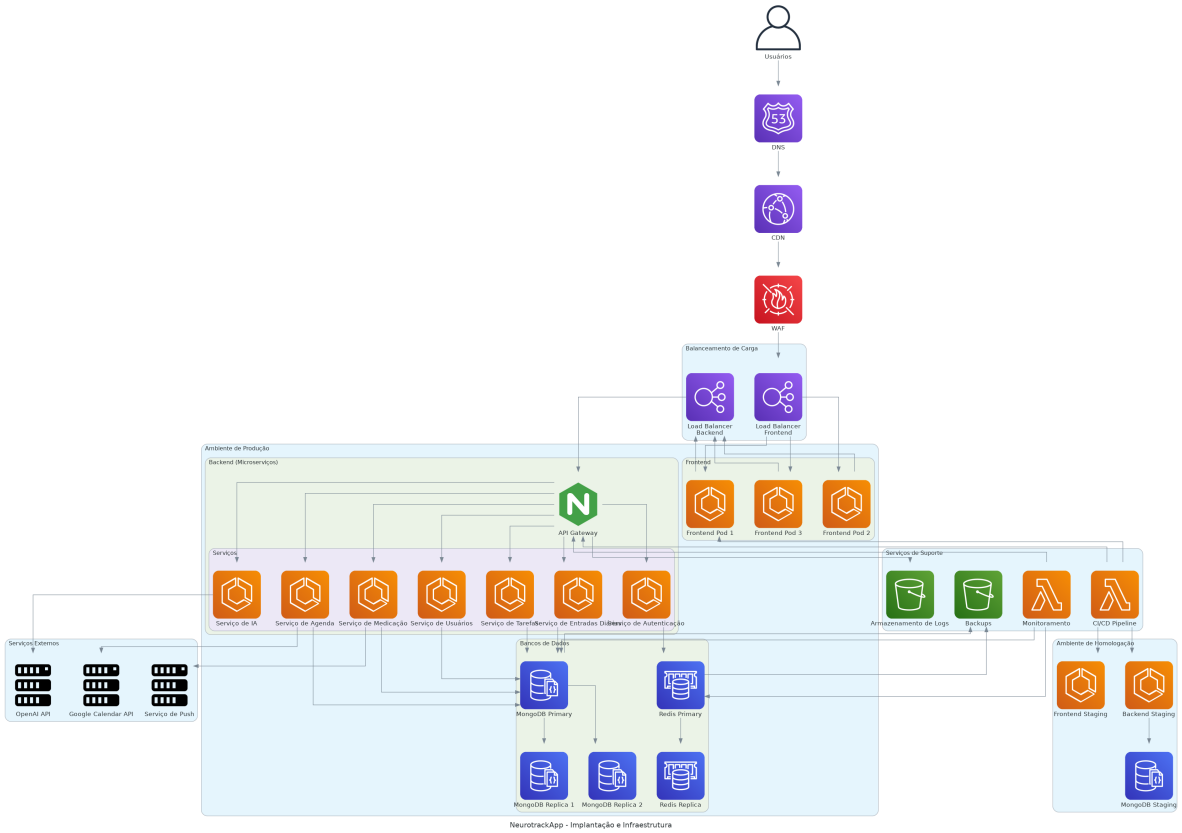


NeurotrackApp - Segurança e Autenticação

O diagrama de segurança e autenticação detalha os mecanismos de proteção do sistema:

- **Autenticação de Usuários**:
- Login com email/senha
- Autenticação OAuth (Google, Apple)
- Autenticação de dois fatores
- Tokens JWT para sessões
- **Proteção de API**:
- Firewall de aplicação web (WAF)
- Proteção contra DDoS
- Rate limiting
- Validação de entrada
- **Segurança de Dados**:
- Criptografia em trânsito (HTTPS)
- Criptografia em repouso
- Gerenciamento seguro de chaves
- Mascaramento de dados sensíveis
- **Controle de Acesso**:
- Autorização baseada em papéis
- Políticas de acesso granulares
- Auditoria de acessos
- Revogação de tokens

Diagrama de Implantação e Infraestrutura



O diagrama de implantação e infraestrutura mostra como o sistema é hospedado e implantado:

- **Ambiente de Produção**:
- Balanceadores de carga para distribuição de tráfego
- Contêineres para frontend e backend
- Clusters de bancos de dados com replicação
- CDN para entrega de conteúdo estático
- **Ambiente de Homologação**:
- Réplica do ambiente de produção para testes
- Bancos de dados isolados
- Integração com serviços externos em modo sandbox
- **Pipeline de CI/CD**:
- Integração contínua para testes automatizados
- Implantação contínua para ambientes de homologação e produção

- Rollbacks automatizados em caso de falha

- **Monitoramento e Logs**:

- Coleta centralizada de logs
- Alertas para problemas de performance
- Métricas de uso e desempenho
- Rastreamento de erros

Conclusões e Recomendações

Pontos Fortes da Arquitetura

1. **Modularidade**: A arquitetura baseada em microserviços permite evolução independente dos componentes.
2. **Escalabilidade**: O uso de contêineres e balanceadores de carga facilita o escalonamento horizontal.
3. **Segurança**: Múltiplas camadas de proteção garantem a segurança dos dados dos usuários.
4. **Integração**: Conexões bem definidas com serviços externos ampliam as funcionalidades do sistema.

Recomendações para Melhorias

1. **Implementar Cache Distribuído**: Adicionar uma camada de cache distribuído para melhorar a performance.
2. **Aprimorar Observabilidade**: Implementar rastreamento distribuído para melhor diagnóstico de problemas.
3. **Expandir Testes Automatizados**: Aumentar a cobertura de testes para garantir maior qualidade.
4. **Implementar Backup Geográfico**: Adicionar redundância geográfica para maior resiliência.

Próximos Passos

1. **Revisão de Segurança**: Conduzir uma análise de segurança completa.
2. **Teste de Carga**: Verificar o comportamento do sistema sob alta demanda.
3. **Documentação de API**: Expandir a documentação da API para facilitar integrações.
4. **Treinamento da Equipe**: Garantir que todos os desenvolvedores compreendam a arquitetura.

Este documento fornece uma visão abrangente da arquitetura do NeurotrackApp, desde o nível mais alto (contexto) até os detalhes de implementação (código), complementada por diagramas específicos para aspectos importantes como segurança, fluxo de dados e implantação. A documentação segue o modelo C4, facilitando a compreensão por diferentes públicos, desde stakeholders até desenvolvedores.