OpenInsurance

MOP|Módulo de Operações OPIN

Março | 2025

OpenInsurance

MOP EVOLUÇÃO – DISCOVERY

- Escopo
- Módulos
- Planejamento
- Princípios de Arquitetura
- Arquitetura MOP
- Definição da Tecnologia

Março de 2025

Escopo

OpenInsurance

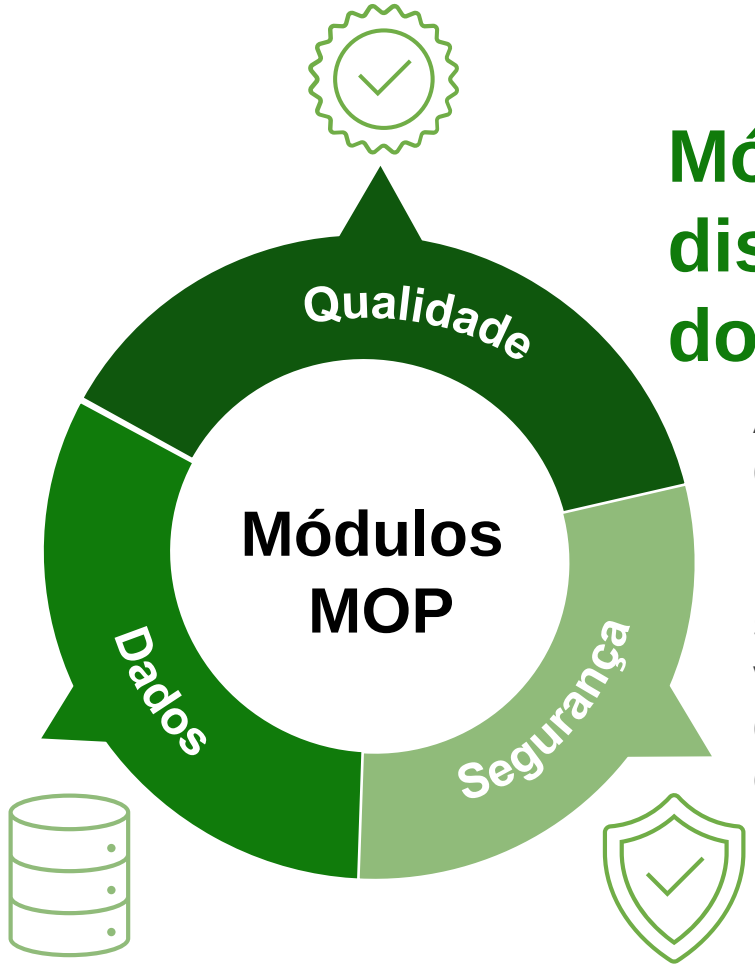
Escopo

Desenhar e planejar em alto nível a etapa de Evolução do MOP

Contexto

A motivação para desenvolvimento do **Módulo de Operações OPIN (MOP)** é a promoção da **transparência** e **confiabilidade** no âmbito do **Open Insurance.**

Facilitando o acesso direto à infraestrutura dos participantes, o desenvolvimento da plataforma pretende **eliminar obstáculos** na comunicação e atuar como um **acelerador** na implementação e análise de qualidade dos dados e prevenção à riscos de cibersegurança e fraudes.



Módulos para disponibilização do container IaC do MOP

A entrega da solução proposta neste documento visa atender a três módulos de container da plataforma MOP **(dados, qualidade e segurança),** para inspecionar e validar os processos de tráfego de dados da transações a fim de apontar e identificar possíveis problemas.

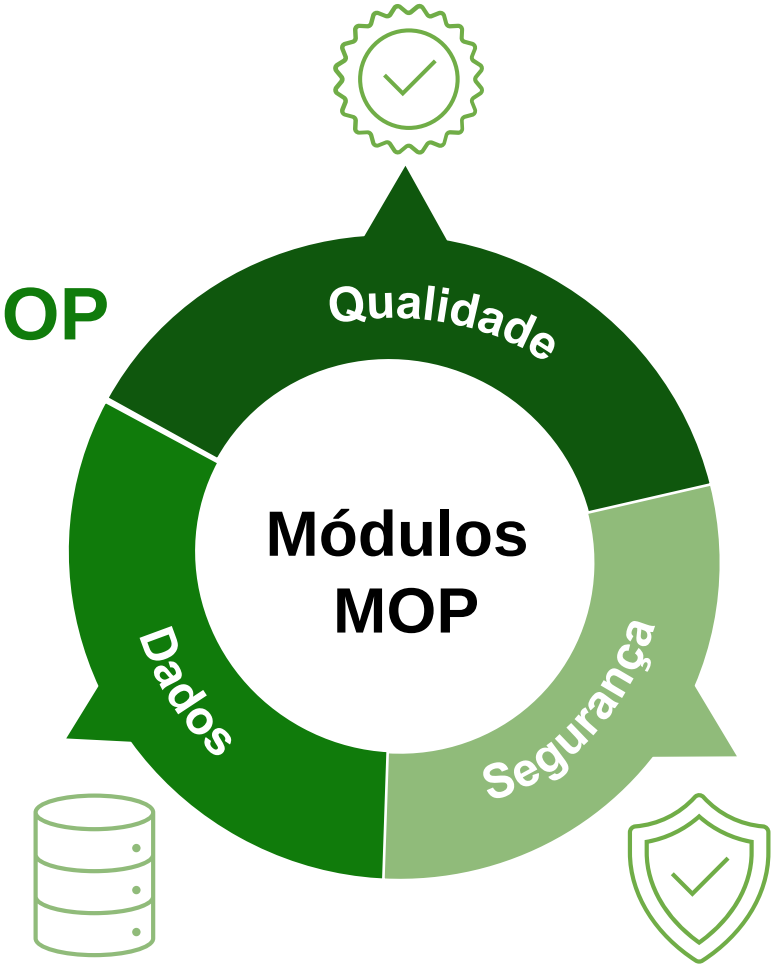
Escopo

Desenhar e planejar em alto nível a etapa de Evolução do MOP

Módulos para disponibilização do container IaC do MOP

A plataforma, em modelo **container IaC**, deverá ser disponibilizada em **código aberto** para as instituições e hospedado nas respectivas infraestruturas.

Este container deverá ter 3 módulos: **dados, qualidade e segurança**, os quais serão detalhados mais adiante.



A plataforma inspecionará a qualidade das requisições e respostas após as transações, isto é, ela não servirá ainda para que as instituições validem dados **durante o processo** para identificar problemas.

De forma assíncrona, ela deverá **enviar os dados coletados** para a Estrutura OPIN.

Escopo

Elementos funcionais e não funcionais

MÓDULOS MOP | ESCOPO DE ENTREGA

Projeto e Desenvolvimento do MOP com estrutura modular para uso no ecossistema Open Insurance

A implantação do Node Admin na nova estrutura criada para ser a base do MOP

Importante: Funcionalidades Adicionais e Aceleradores serão avaliados conforme priorização e *capacity* disponíveis, a ser definido em tempo de projeto

REQUISITOS FUNCIONAIS

Módulo de Dados
Consolidação e métricas do PCM: Métricas Dinâmicas (transacionais e do funil de consentimento) (1)

Módulo de segurança
Dados gerados pelo MISP, tais quais as ocorrências de log e eventos relacionados

Módulo de Qualidade
Geração de métricas sobre os dados das chamadas das APIs (com especificação *swagger* regulatório). Métricas sobre header e body como preenchimento de atributos obrigatórios, seu preenchimento, etc. (1)

REQUISITOS NÃO-FUNCIONAIS

Open Source
Toda a solução deve ser implementada com software de código aberto.

Performance
Deve apresentar performance para atender a demanda de requisições futuras

Segurança
A solução deve ser segura para prevenção de ataques ou vazamento de dados

Tecnologias de mercado
Utilizar tecnologia/ infraestrutura de acordo com disponibilidade e mercado

Multicloud
Nova estrutura para o MOP, considerando uma abordagem multicloud

Flexível
Solução minimamente intrusiva no ambiente das participantes

Resiliência
A solução apresentada deve ser escalável e resiliente pois no futuro deverá ser uma plataforma que intermedia as requisições

Módulos

OpenInsurance

Módulos

Primeira entrega

Módulo de Dados

Realizar o monitoramento focado nos dados trafegados nas APIs integradas ao Open Insurance, é essencial garantir que os dados sejam acessados, utilizados e armazenados de forma eficiente, consistente e segura. O Open Insurance envolve a troca de informações sensíveis e com requisitos regulatórios, então, é fundamental o **controle de qualidade, integridade, conformidade e desempenho dos dados.**

DORES ATACADAS

- Inconsistência nos reports de dados
- Maior amplitude de visão das métricas do ecossistema

Módulo de Qualidade

Monitorar a qualidade das APIs do Open Insurance, tem foco em **garantir que as APIs funcionem corretamente, com desempenho consistente, alta disponibilidade e conformidade com os padrões de qualidade** estabelecidos. A qualidade da API é fundamental para garantir uma boa experiência do utilizador e uma integração eficaz entre as partes envolvidas.

DORES ATACADAS

- Automação da certificação estrutural e validação em produção das APIs

Módulo de Segurança

Realizar o **monitoramento focado na segurança das APIs e Dados trafegados** é essencial **para cobrir diversas áreas críticas, bem como garantir a proteção dos dados e conformidade com regulamentações**. O Open Insurance envolve a troca de informações sensíveis, como dados pessoais e financeiros, então a **segurança sempre é tratada como prioridade**.

DORES ATACADAS

- Maior visibilidade e controle dos itens e riscos de segurança

Planejamento

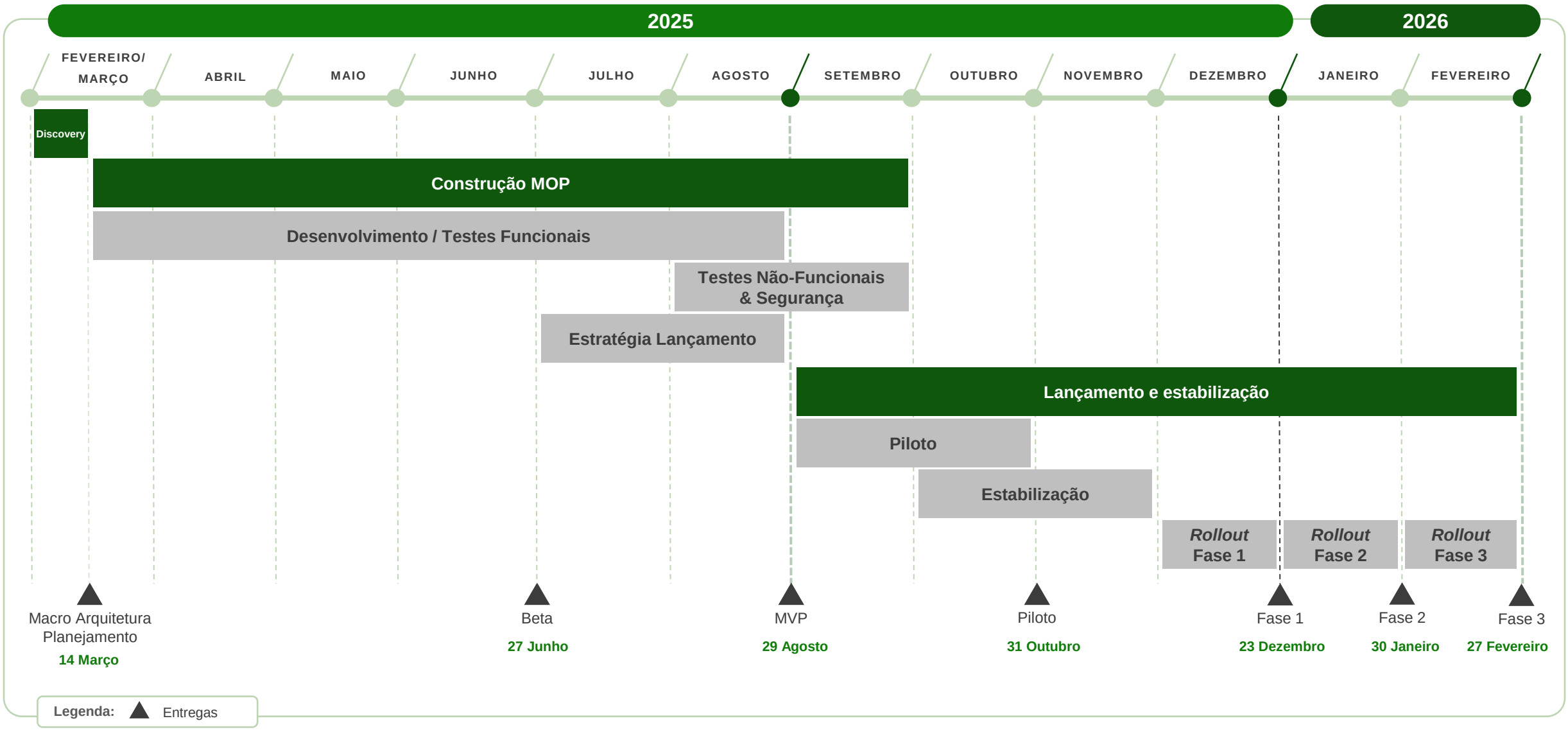
OpenInsurance

Planejamento

Estimativa de implantação e *Go-Live*

OpenInsurance

NÃO EXAUSTIVO



Princípios de Arquitetura

OpenInsurance

Princípios de Arquitetura (1/3)

Definição dos principais princípios que nortearão o design e construção do MOP

1 | Compliance Regulatória e Segurança

- **Aderência à SUSEP e Opin Brasil:** Implementação dos requisitos normativos estabelecidos pelo Open Insurance Brasil.
- **Segurança de Dados (LGPD e OWASP):** Proteção de dados sensíveis dos clientes, garantindo anonimização e criptografia de dados em trânsito e repouso.
- **Autenticação e Autorização:** Seguir todas as definições de segurança do Manual de Segurança do Open Insurance.
- **Auditoria e Monitoramento:** Implementação de logs estruturados, rastreamento distribuído e trilhas de auditoria para compliance e segurança.

2 | Arquitetura Cloud-Native (1)

- **Escalabilidade e Elasticidade:** Uso de contêineres para ajuste dinâmico de capacidade conforme demanda.
- **Resiliência e Alta Disponibilidade:** Implantação em múltiplas zonas de disponibilidade para garantir continuidade do serviço.
- **Confiabilidade:** Sistemas resilientes a falhas, garantindo recuperação rápida e operação consistente por meio de automação, monitoramento e escalabilidade.
- **Otimização de Custo:** Ajuste da capacidade à demanda, evitando desperdícios e escolhendo modelos de precificação adequados.
- **Infraestrutura como Código (IaC):** Automação de provisionamento usando ferramentas de IaC (Infrastructure as Code).

3 | Micro-serviços e API-First

- **Design Baseado em Micro-serviços:** Separação de domínios de negócio em serviços independentes com comunicação via padrões abertos de comunicação.
- **API Gateway e Rate Limiting:** Uso de API Gateway para expor serviços de forma segura e controle de consumo via rate limiting.
- **Observabilidade e Monitoramento:** Uso de ferramentas e técnicas para monitoramento e diagnóstico de serviços.

(1) NOTA: O cliente do MOP poderá ser executado em qualquer plataforma (Cloud, Híbrida ou On-prem)

Princípios de Arquitetura (2/3)

Definição dos principais princípios que nortearão o design e construção do MOP

4 | Interoperabilidade e Padrões de Mercado

- **Adoção de Open APIs:** Implementação de padrões abertos de Open Banking e Open Insurance.
- **Suporte a Mensageria Assíncrona:** Uso de plataformas de fila e streaming de mensagens para comunicação desacoplada entre serviços.
- **Compatibilidade com OpenID e FAPI:** Adoção do padrão para segurança de APIs financeiras.

5 | Performance

- **Caching Estratégico:** Uso de tecnologias de *caching* e distribuição de conteúdo para minimizar latência e aliviar carga nos bancos de dados.
- **Banco de Dados Otimizado:** Indexação eficiente, uso de fragmentação de dados e replicação para melhorar tempos de resposta e escalabilidade.
- **Processamento Assíncrono:** Uso de filas e arquitetura orientada a eventos para evitar bloqueios e melhorar tempo de resposta das APIs.

6 | Gestão de Dados e Persistência

- **Banco de Dados Multi-Modelo:** Uso de bancos relacionais e NoSQL conforme necessidade.
- **Data Lake e Analytics:** Armazenamento estruturado para análise de dados, agregação e relatórios regulatórios.
- **ETL e Streaming de Dados:** Processamento em tempo real de eventos e transações com frameworks para processamento distribuído de dados.

7 | Orquestração e Automação

- **CI/CD e DevOps:** Pipelines automatizados com ferramentas de CI/CD para entregas contínuas.
- **Testes Automatizados e Shift-Left:** Testes de unidade/integração/segurança aplicados no início do ciclo de desenvolvimento.
- **Feature Toggles e Blue-Green Deploys:** Estratégias para lançamentos controlados e *rollback* seguro.

Princípios de Arquitetura (3/3)

OpenInsurance

Definição dos principais princípios que nortearão o design e construção do MOP

8 | Minimização de Lock-in Tecnológico

- **Adoção de Tecnologias Open Source sempre que possível:** Priorização de frameworks, bancos de dados e ferramentas de código aberto para reduzir a dependência de fornecedores proprietários.
- **Arquitetura Multi-Cloud e Cloud-Agnostic:** Utilização de serviços compatíveis com múltiplas nuvens para facilitar migrações futuras, sempre equilibrando custo-benefício.
- **Uso de Contêineres:** Facilitar a portabilidade entre provedores de nuvem e ambientes on-premises, garantindo independência tecnológica relativa.
- **APIs e Padrões Abertos:** Adoção de OpenAPI, REST, gRPC e mensageria desacoplada para evitar dependência de tecnologias proprietárias.

9 | Eficiência no Consumo de Recursos na Nuvem (FinOps)

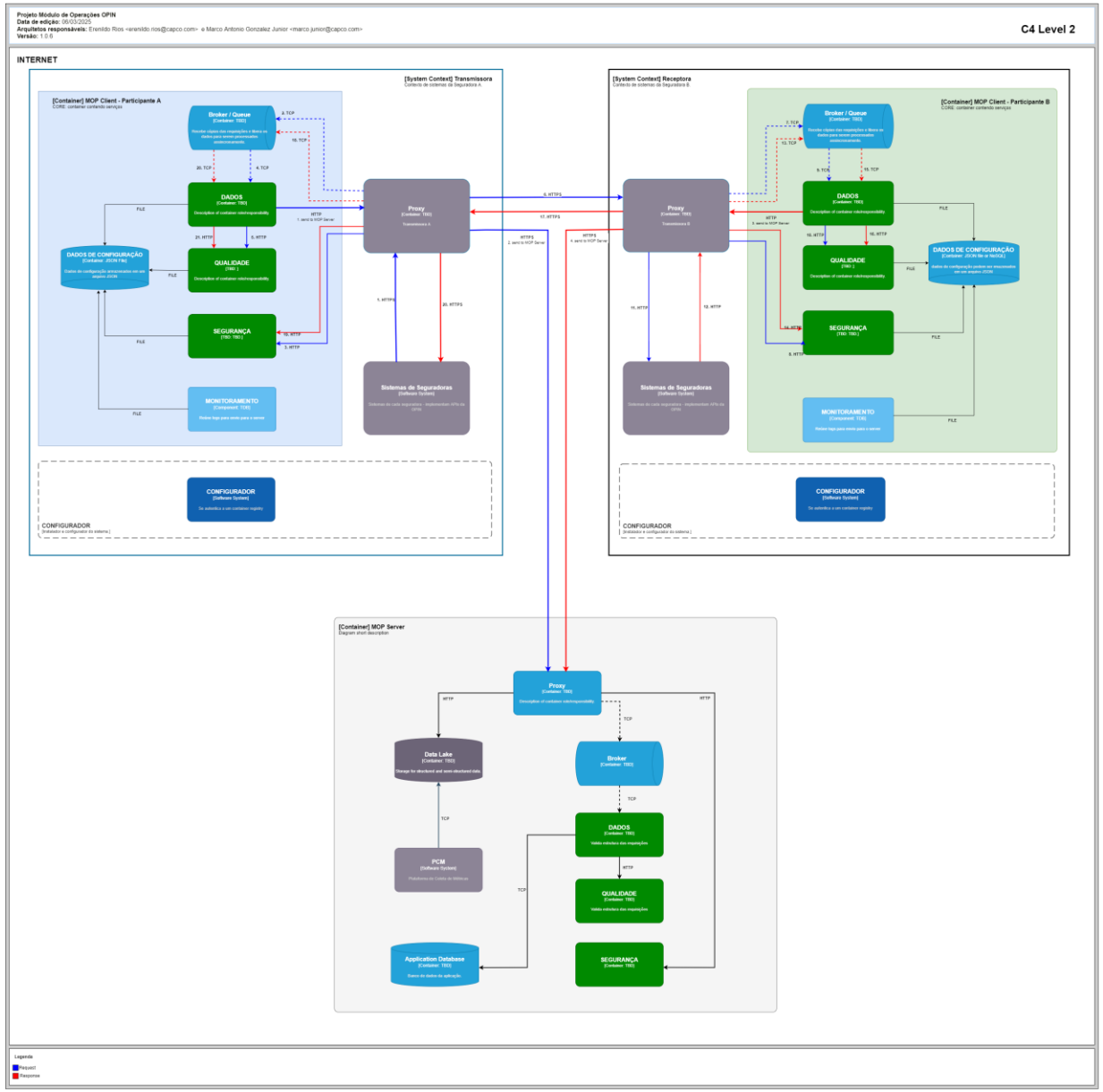
- **Dimensionamento Ótimo de Infraestrutura:** Uso de *autoscaling*, *spot instances* e *right-sizing* para ajustar a capacidade computacional conforme demanda.
- **Monitoramento de Custos e Análises Preditivas:** Implementação de práticas FinOps para otimização contínua do gasto em nuvem, com ferramentas de controle e otimização de custos.
- **Serverless e Event-Driven quando aplicável:** Utilização de computação sob demanda para reduzir custos fixos e pagar apenas pelo uso real de recursos.
- **Governança e Orçamento:** Definição de políticas de uso da nuvem para evitar desperdícios e garantir previsibilidade dos custos.

Diagrama

OpenInsurance

Diagrama Container

C4 L2



NOTA: Diagrama disponível no Portal do Desenvolvedor

Definição da Tecnologia

OpenInsurance

Definição da Tecnologia (1/3)

Landscape

Segurança

Desenvolvimento

DevOps

Cloud Native

Infraestrutura

Definição da Tecnologia (2/3)

OpenInsurance

MOP | Contexto de utilização e benefícios das soluções recomendadas

#	Contexto	Tecnologia	Motivo
1	**Segurança**	TLS 1.2+ , Certificados ICP-Brasil, KeyVault, OAuth, Zero Trust	▪ Recomendação de uso para conformidade com as melhores práticas de mercado.
2	**Desenvolvimento**	Java, Spring Boot, Quarkus Contexto específico Client: TypeScript, NodeJS, Go (Golang), ExpressJS	▪ **Java:** Linguagem de programação robusta, portátil e orientada a objetos, amplamente usada em aplicações empresariais. ▪ **SpringBoot:** Facilita o desenvolvimento Java com configuração automática e suporte eficiente a microsserviços. ▪ **Quarkus:** Otimizado para cloud e Kubernetes, oferecendo inicialização ultrarrápida, baixo consumo de memória ▪ **TypeScript:** Performance, eficiência em utilização de recursos, facilidade no desenvolvimento e manutenção, ecossistema, escalabilidade e concorrência, comunidade e bibliotecas, disponibilidade de desenvolvedores. ▪ **NodeJS:** Melhor para APIs, arquitetura orientada a eventos, microsserviços. ▪ **Go (Golang):** Nativo Linux, binário único, melhor integração com sistemas. ▪ **ExpressJS:** Framework minimalista e rápido para construir APIs e aplicações web em Node.js.
3	**DevOps**	GitHub, GitHub Actions, ArgoCD, Bruno	▪ **GitHub/GitHub Actions:** Otimo para CI e automação do *build* e *deploy*. ▪ **ArgoCD:** Ideal para o *Continuous Delivery* declarativo no Kubernetes via GitOps. É uma ferramenta declarativa de entrega contínua para Kubernetes, garantindo deploys automatizados e rastreáveis. ▪ **Bruno API Client:** Ferramenta de código aberto para testar e gerenciar APIs, oferecendo uma alternativa rápida e leve ao Postman, com suporte a coleções locais em arquivos JSON.

Definição da Tecnologia (3/3)

OpenInsurance

MOP | Contexto de utilização e benefícios das soluções recomendadas

#	Contexto	Tecnologia	Motivo
4	**Cloud Native**	Docker, Kubernetes, <u>Contexto específico Client:</u> Helm	▪ **Docker:** Plataforma de contêinerização que facilita a criação, distribuição e execução de aplicações isoladas e portáveis. ▪ **Kubernets:** Melhor escolha para orquestração de containers por sua escalabilidade, automação e resiliência. ▪ **Helm:** Proxy reverso e balanceador de carga dinâmico para microsserviços, com integração nativa a Kubernetes e configuração automática.
5	**Infraestrutura**	HAProxy <u>Contexto específico Client:</u> Traefik <u>Contexto específico Server:</u> Prometheus, MongoDB, Logstash,Grafana RabbitMQ, PostgreSQL, Azure APIM, Azure Tables, Azure Container Register	▪ **HAProxy:** Oferece balanceamento de carga eficiente, alta disponibilidade e proxy reverso de alto desempenho. ▪ **Traefik:** Proxy reverso e balanceador de carga dinâmico para microsserviços, com integração nativa a Kubernetes e configuração automática. ▪ **MongoDB:** Banco de dados NoSQL orientado a documentos, escalável e flexível para armazenar grandes volumes de dados. ▪ **RabbitMQ:** Enfileiramento de mensagens tradicional, latência, performance, tolerância a falhas (built-in DLQs), não retenção mensagens, *downstream analytics,* manutenção, simplicidade. ▪ **PostgreSQL:** Banco de dados relacional open-source, robusto, extensível e com suporte avançado a SQL e JSON. ▪ **Logstash:** Ferramenta de ingestão de dados que processa, transforma e envia logs para múltiplos destinos, como Elasticsearch. ▪ **Grafana/Prometheus:** Utiliza agentes externos ou sidecars para coletar e processar os dados gerados pelos containers ▪ **Azure:** Retrocompatibilidade com os componentes existente no ambiente do OPIN Brasil.