Projeto Thalamus

Documento de Arquitetura de Software

Versão 1.0

| Histórico de Revisões | | | |
| --- | --- | --- | --- |
| Versão | Data | Descrição | Autor |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sumário

[1. Introdução 4](#_heading=h.gjdgxs)

[2. Identificação do Projeto 4](#_heading=h.1fob9te)

[3. Contexto 4](#_heading=h.4mz158hcsfiq)

[4. Decisões Arquiteturais 4](#_heading=h.7k5crya60buv)

[5. Representação Arquitetural 5](#_heading=h.3znysh7)

[6. Metas e Restrições da Arquitetura 6](#_heading=h.2et92p0)

[7. Considerações de Segurança 6](#_heading=h.j3upa6rwe9n)

[8. Referências 6](#_heading=h.3j2qqm3)

[9. Aprovações 7](#_heading=h.1y810tw)

Documento de Arquitetura de Software

# Introdução

Este documento apresenta a visão geral da arquitetura de software adotada para a evolução das aplicações legadas desenvolvidas com Vue.js e Laravel PHP, que se integrarão ao Projeto Thalamus. Nosso objetivo principal é transformar essas aplicações em sistemas distribuídos, aproveitando as vantagens oferecidas pela tecnologia de contêineres Docker.

# Identificação do Projeto

| Projeto | Projeto Thalamus |
| --- | --- |
| Requisitante | Darley Wilson Dias |
| Gerente de Projetos | Darley Wilson Dias |

# Contexto

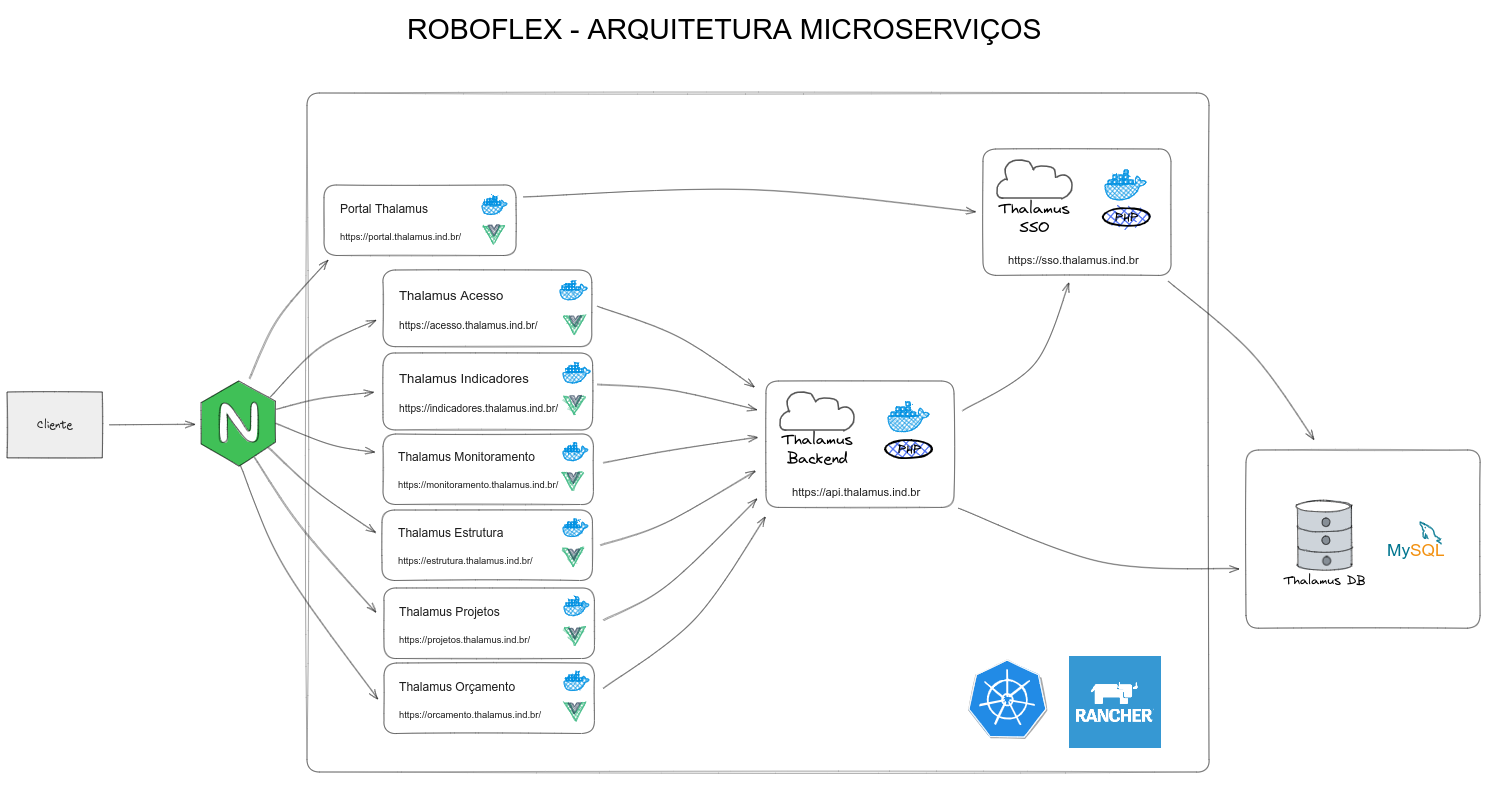
As aplicações legadas foram desenvolvidas utilizando o framework Vue.js para o front-end e Laravel PHP para o back-end. Embora essas tecnologias tenham atendido às necessidades iniciais do projeto, a crescente complexidade e demanda por escalabilidade nos levaram a buscar uma abordagem mais distribuída e flexível.

# Decisões Arquiteturais

* 1. Microserviços Distribuídos: Optamos por adotar uma arquitetura de microserviços, onde cada funcionalidade específica é encapsulada em um serviço independente. Isso permite uma maior modularidade, facilitando o desenvolvimento, teste e implantação.
  2. Contêineres Docker: Escolhemos utilizar contêineres Docker para empacotar e distribuir nossos microserviços. Docker oferece uma abordagem leve e portátil para a virtualização, garantindo consistência entre os ambientes de desenvolvimento, teste e produção.
  3. Orquestração com Kubernetes: Para gerenciar nossos contêineres Docker em escala, decidimos adotar o Kubernetes como nossa plataforma de orquestração. Kubernetes oferece recursos avançados de gerenciamento de contêineres, como balanceamento de carga, auto-escalabilidade e alta disponibilidade.
  4. API Gateway: Implementamos um gateway de API para centralizar o acesso aos nossos microserviços. Isso simplifica a comunicação entre os diferentes componentes do sistema e oferece uma camada de segurança adicional.
  5. Padrões de Comunicação: Adotamos padrões de comunicação RESTful para facilitar a integração entre os microserviços. Isso garante uma comunicação eficiente e flexível, permitindo a interoperabilidade entre diferentes partes do sistema.

# Representação Arquitetural

O diagrama abaixo, representa a arquitetura do módulo WEB em alto nível.



Após uma análise do panorama atual das aplicações web desenvolvidas para atender às demandas da Roboflex, que serão integradas ao Projeto Thalamus, reconhecemos a necessidade de revisão da arquitetura existente. Esta revisão visa não apenas alinhar a arquitetura com as condições atuais, mas também prepará-la para futuras exigências do projeto.

Atualmente, os diversos módulos das aplicações foram desenvolvidos de maneira independente, cada um com seu próprio conjunto de artefatos de front-end, desenvolvidos utilizando o framework Vue.js, conectando-se à API central (backend), desenvolvida em PHP 8.1, utilizando o framework Laravel 11, onde coordena todas as funcionalidades. Diante desse contexto, propomos a adoção de containers Docker para hospedar todos os módulos. Essa abordagem permite uma escalabilidade horizontal e elástica, preparando-nos para um crescimento contínuo da demanda.

# Metas e Restrições da Arquitetura

Esta abordagem de arquitetura, tem como objetivo entregar um produto de software modular e escalável, e preservar os conhecimentos e resultados já conquistados pela equipe de desenvolvimento, considerando pequenas, mas importantes adequações no código-fonte legado, garantindo mais robustez, segurança, reutilização e portabilidade.

# Considerações de Segurança

Todos os endpoints do Backend e validações de authenticação e autorização do Frontend, deverão estar integrados ao Thalamus SSO, onde os frontends deverão registrar as bibliotecas thalamus-sso-lib e thalamus-request-handler eu seu arquivo de implentação package.json.

# Aprovações

| Aprovações | | |
| --- | --- | --- |
| Participante | Assinatura | Data |
|  |  |  |
|  |  |  |