STIC Subsecretaria de Tecnologia da Informação e Comunicação

DOCUMENTO DE ARQUITETURA DE SOFTWARE – DAS

Sistema: [Modelo aplicação em PHP]
Versão:1.1

Ministério da Educação

PÁTRIA AMADA BRASIL GOVERNO FEDERAL

Sumário

DOCUMENTO DE ARQUITETURA DE SOFTWARE – DAS – PHP

Controle de Versões			
Versão	**Data**	**Autor**	**Descrição**
1.0	17/12/2019	BASIS	Informações extraídas do Documento de Arquitetura de Software elaborado e entregue pela fábrica de software (BASIS) em 17/12/2019.
1.1	25/05/2021	G4F	Informações extraídas do Documento de Arquitetura de Software, atualizada conforme ambiente disponível no MEC e entregue ao CGS do MEC.

1 Introdução

Este documento visa descrever a arquitetura lógica e física dos projetos na linguagem de programação PHP a serem desenvolvidos e mantidos no ambiente do Ministério da Educação.

Este documento se aplica somente para os sistemas novos, ou seja, aqueles que ainda serão desenvolvidos e entregues ao MEC.

1.1 Definições, Acrônimos e Abreviações

Abreviação	Descrição
API	Application Programming Interface (Interface de Programação de Aplicativos)
JWT	JSON Web Tokens (Chave de Conexão para Objetos Secretos)
REST	Representational State Transfer (Formato de Transferência de Dados pela Web)
SSO	Single Sign-On (Acesso Unico)
Strangler pattern	Design Pattern para transformar de maneira incremental uma aplicação monolítica em microsserviços, substituindo uma a uma as funcionalidades por novos microsserviços
Helm	Gerenciador de pacotes para Kubernetes (gerenciador de pacotes kubernetes)
Chart	A chart is a collection of files that describe a related set of Kubernetes resources (Pacote de gerenciador de kubernetes para aplicação)
Gitlab	O GitLab é um gerenciador de repositório de software baseado em git
migration	Permite a manipulação de bancos de dados
Container	O contêiner é um ambiente isolado com uma versão de código-fonte

Tabela 1-Tabela de abreviações

2 Objetivo do Documento

O objetivo deste documento é apresentar e padronizar os sistemas que serão desenvolvidos por empresas parceiras do MEC, sejam elas universidades, institutos federais ou empresas regidas pelos contratos administrativos.

3 Visões da Arquitetura

3.1. Visão Geral das Camadas Arquiteturais

Em arquitetura de sistemas, damos o nome de camada a uma estrutura lógica de componentes de um software que interagem entre si para cumprir um objetivo específico, responsabilizando-se pela execução de uma dada tarefa ou um conjunto de tarefas semelhantes. Em alguns casos, esse agrupamento pode não ser apenas lógico, mas sim assumir um caráter físico. Isso ocorre quando uma camada possui a habilidade de ser executada separadamente do restante da aplicação.

A arquitetura baseada em múltiplas camadas oferece um modelo flexível e reutilizável para o desenvolvimento de software. Ao se dividir uma aplicação em camadas, evitamos o forte acoplamento entre os componentes, de forma que a maioria das alterações necessárias possam ser feitas de maneira mais isolada e pontual, reduzindo o esforço necessário à manutenção do sistema.

Na figura abaixo, apresentamos uma visão simplificada das camadas que compõem o sistema e a forma como elas se relacionam. Nas seções a seguir, detalhamos conceitualmente cada camada e enumeramos as ferramentas e plataformas que serão utilizadas para sua construção.

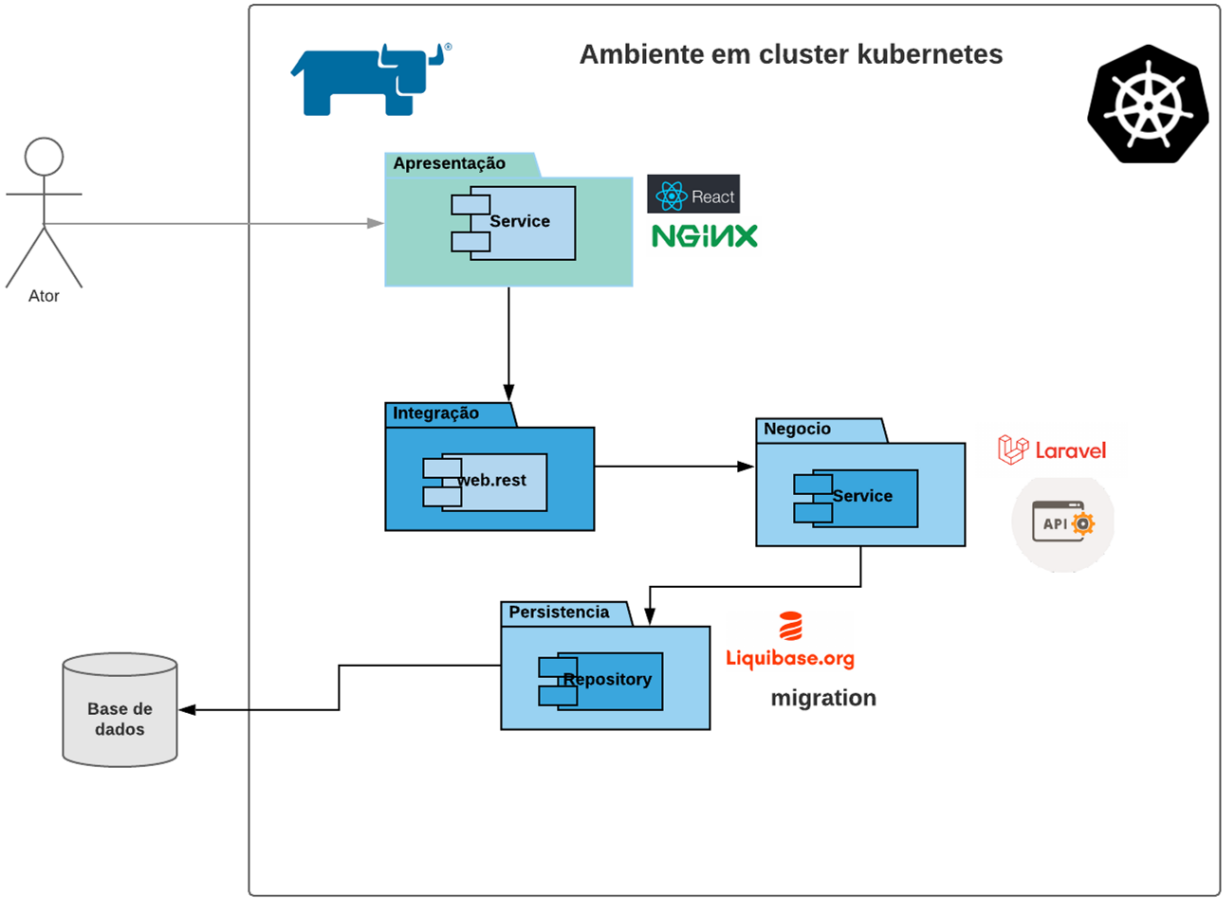

Figure 1. Visão Geral das Camadas Arquiteturais

4 Camada de Apresentação

A camada de apresentação é responsável por fazer a lógica de construção das páginas para serem exibidas pelos usuários, tratar os eventos do browser, como cliques, e gerenciar o fluxo de execução do sistema.

4.1 Camada de Integração

O objetivo da camada de integração é prover um meio de comunicação entre o sistema que vai ser desenvolvido e os demais sistemas que o circundam. Os objetos dessa camada fornecem à camada de negócio uma simplificação para o acesso aos sistemas externos, livrando-a da responsabilidade de tratar detalhes inerentes aos protocolos de comunicação envolvidos, formatações e conversões de tipos de dados e demais particularidades sintáticas e semânticas inerentes aos sistemas externos.

4.2 Camada de Negócio

A camada de negócio é responsável pela implementação lógica da aplicação. Ela expõe os serviços para a camada de apresentação por meio de uma interface bem definida e obtém as informações necessárias para mostrar ao usuário por meio da Camada de Persistência.

4.3 Camada de Persistência

A camada de persistência é responsável pela lógica de acesso ao banco de dados e pelo mapeamento dos dados em entidades representativas. O objetivo em mapear o banco de dados em entidades representativas ao sistema é diminuir a diferença semântica entre o modelo abstrato do banco e o problema do mundo real.

4.4 Camada de Serviços

A camada de serviços encapsula diversos serviços que são providos as outras camadas da aplicação.

5 Ambiente de Desenvolvimento e Ambiente em Contêiner

Todas as aplicações a serem desenvolvidas e entregue ao MEC deverão prever arquitetura híbrida com backend separado do frontend, conforme imagem abaixo.

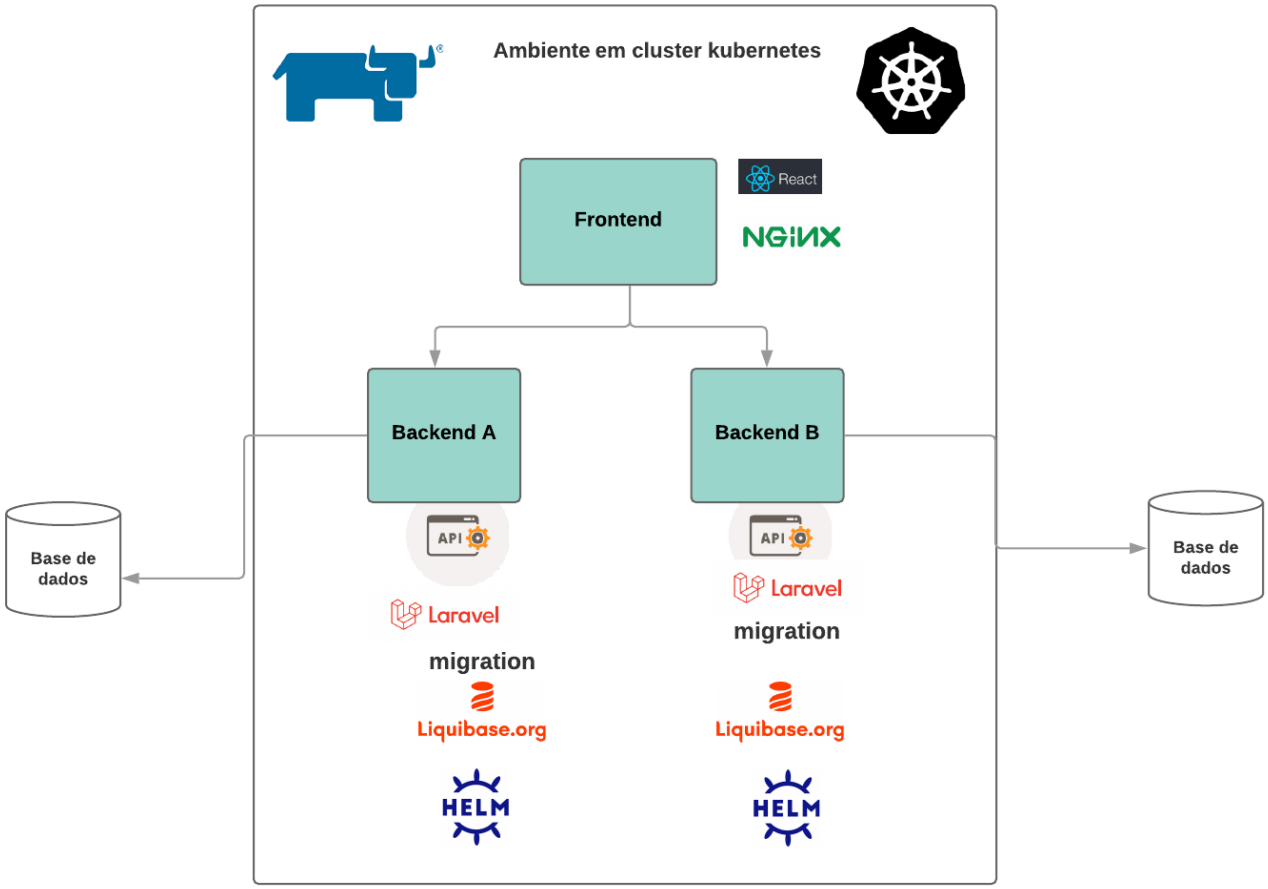

Figure 2. Visão arquitetura hibrida

Arquitetura híbrida é aquela aplicação onde se tem bem definido, uma camada de backend e outra de frontend com distribuição de responsabilidade, o frontend está diretamente exposto a usuários finais. Como resultado, atualmente todas as aplicações deverão ser desenvolvidas nessa estrutura no qual é requisito obrigatório no momento da implantação no ambiente DevOps do MEC.

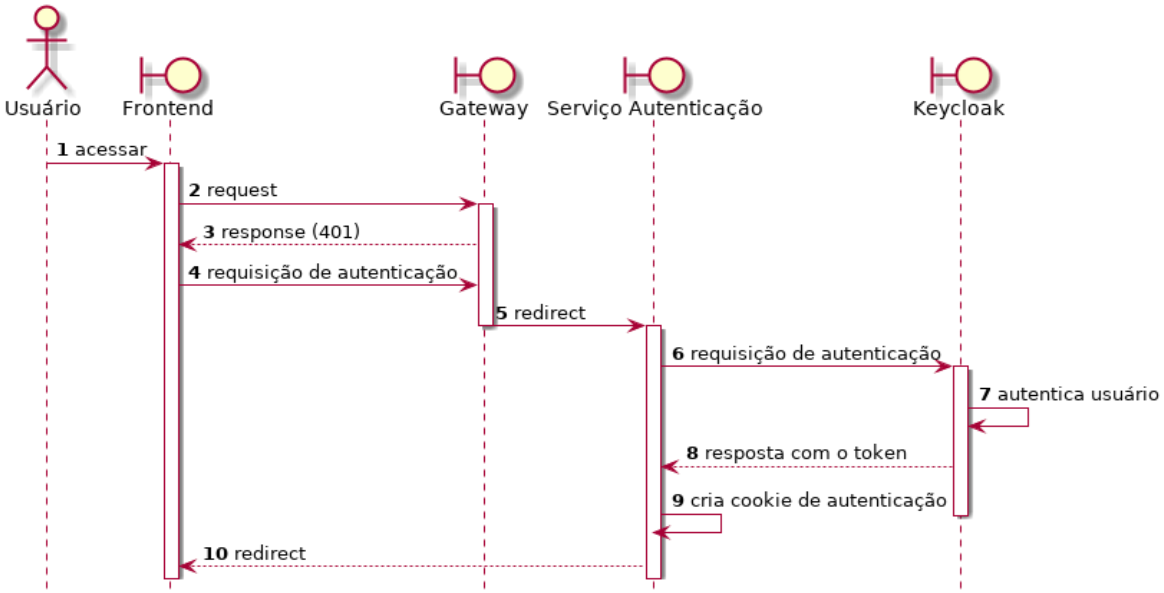

Figura 3. Visões de comunicação

5.1 Módulo Frontend

Módulo que faz interação com o cliente. Este módulo contém as regras básicas de validações de campos sem nenhuma regra de negócio que possa conter informações sensíveis do negócio. O objetivo deste módulo é coletar as informações e/ou as exibir para o usuário de forma amigável a fim de melhorar a experiência do usuário. Este módulo é organizado internamente em Componentes e Serviços.

Os componentes são responsáveis pela coleta e exibição de dados, juntamente com as opções de controles de telas. Os services são responsáveis pela validação dos dados recebidos pelos componentes e além disso fazer a comunicação com a API.

5.2 Módulo Gateway

Módulos responsáveis por encaminhar as requisições para os serviços. O Gateway também adicionará no cabeçalho da requisição o token que será recuperado do cookie de sessão. Esse procedimento possibilita a integração dos novos serviços com os sistemas legados de forma transparente, permitindo assim uma maior flexibilidade na organização da infraestrutura e a implementação dos serviços. A implementação será feita utilizando o Istio Gateway.

5.3 Módulo Serviço Autenticador

Módulos responsáveis por realizar a autenticação dos serviços no SSO e disponibilizar uma interface estável entre os serviços e os clientes, permitindo assim uma customização da lógica de autenticação de forma centralizada.

5.4 Visão de Integração

A visão de integração apresenta as integrações pertinentes ao sistema. Essas integrações podem ser internas ou externas e é de suma importância que os projetistas e desenvolvedores conheçam em detalhes essas integrações, bem como os contratos e protocolos de comunicação entre os sistemas/componentes envolvidos na integração.

A fim de atender os requisitos funcionais e não-funcionais da aplicação, o sistema deve se integrar aos seguintes serviços:

- Outros Serviços/APIS.
- **Istio Gateway** – Descreve um balanceador de carga operando na borda da malha recebendo conexões HTTP / TCP de entrada ou de saída. A especificação descreve um conjunto de portas que devem ser expostas, o tipo de protocolo a ser usado, configuração SNI para o balanceado de carga.
- Serviço Autenticador.
- **Keycloak -** é um produto de software de código aberto que permite logon único com gerenciamento de identidade e acesso voltado para aplicativos e serviços.

5.5 Integração com outros Serviços/APIS

A integração com outros serviços deverá ser feita utilizando o padrão REST (Representational State Transfer). Nesse modelo, a "Aplicação A" deverá realizar uma requisição HTTP a "Aplicação B" a fim de obter algum tipo de informação que é do seu domínio. A "Aplicação B" resolverá a requisição e devolverá uma resposta para a "Aplicação A". A aplicações estarão protegias através da lógica de autenticação por token JWT. Os dados a serem tratados nessa integração devem obedecer ao tipo JSON.

6 Inicialização do Sistema/Projeto

Ao ser iniciado um projeto no ambiente do MEC para as aplicações que serão desenvolvidas na linguagem PHP será disponibilizado pela equipe de arquitetura da CGS um esqueleto denominado modeloapp, disponível na url abaixo, no qual se prevê os arquivos e arquitetura inicial para o início do desenvolvimento.

Link do Gitlab do Backend: http://gitlabbuilder.mec.gov.br/modeloapp/modelophp

Frontend deverá ser desenvolvido em conjunto com o ReactJs e fazendo uso da interface visual do Design System do Governo Federal (DSGOV) disponível em https://dsgov.estaleiro.serpro.gov.br/

Link do Gitlab do Frontend: http://gitlabbuilder.mec.gov.br/modeloapp/modeloreact

Aplicação também deverá contemplar autenticação com **acessoGov** o roteiro de integração está disponível na Url : https://manual-roteiro-integracao-login-unico.servicos.gov.br/pt/stable/iniciarintegracao.html

A integração com o Keycloak será utilizada para autenticação dos usuários do sistema através do protocolo da autenticação Open ID Connect (OIDC).

Todos as aplicações deverão prever a integração com acessoGov, que garante a identificação de cada cidadão que acessa os serviços digitais do governo, os meios para realizar essa integração, tais como cadastro de client ID serão fornecidos pela área gestora da coordenação geral de sistema do MEC.

6.1 Ferramentas recomendadas no ambiente de desenvolvimento

1. Docker
2. Docker-compose
3. Composer (gerenciador de dependências)
4. PHP 7.4
5. Laravel
6. GIT
7. Racher
8. Helm
9. Chart

6.2 Banco de Dados

A solução pode prever acessos às seguintes bases de dados com as respectivas versões;

- Mysql versão (**5.5.61**);
- Microsoft SQL Server versão (**SQL Server 2017 - 14.0.3281.6 – Enterprise Edition (64-bit));**
- Oracle versão **(12.2.0.1);**
- PostgresSql versão **(11.00);**

Em relação aos SGBDS que serão adotados para aplicação a ser construída, deverá ser levado em consideração a necessidade da aplicação e será uma decisão em conjunto, envolvendo equipe técnica do MEC e equipe técnica da contratada.

7 Qualidade de código e auditoria nas entregas

O ambiente de integração contínua prevê testes com uso da ferramenta sonarqube, gitlab (CodeSniffer e Mess Detector) com a finalidade de analisar os padrões de código, auxiliando na qualidade do código produzido. O codeSniffer realizará auditoria no repositório do gitlab da aplicação no qual será verificado as

melhores práticas do mercado para o desenvolvimento de software utilizando, no caso do backend, PSR (PHP Standard Recommendation). A equipe de arquitetura do MEC poderá no momento de a entrega realizar auditório de código-fonte, auditoria realizada, será de forma manual no qual o código-fonte será analisado e ser for necessário solicitar os ajustes a contratada, desse modo a mesma será notificada para realizar os devidos ajustes, com o objetivo de que entrega esteja nos padrões de qualidade exigidos pelo MEC.

8 Ambiente de Homologação

O ambiente de homologação será usado para que o gestor do Aplicação realize os devidos testes de aceitação na aplicação que está sendo desenvolvida ou correção de algum bug, o ambiente de homologação é de responsabilidade da equipe de arquitetura e dos gestores do MEC. Após o aceite formal por parte da área negocial, a aplicação estará apta para ser disponibilizada em produção.

9 Ambiente de Implantação do Rancher

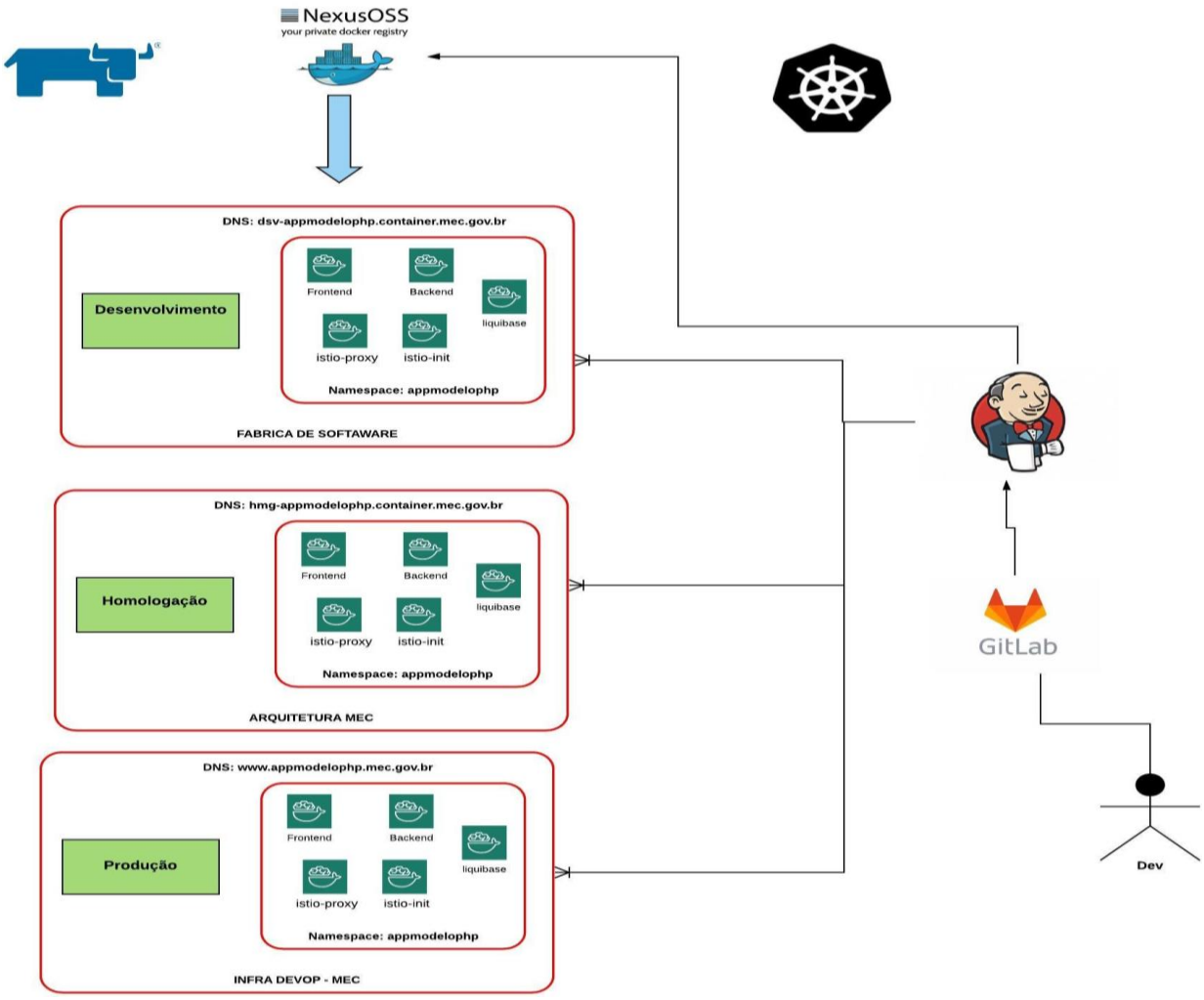

Figure 4. Diagrama de Implantação

Branch Master: Essa branch é de total responsabilidade da equipe de INFRAESTRUTURA do MEC, visto que a integração-continua escuta essa branch e qualquer atualização nesta, a pipeline de deploy é disparada e assim o ambiente de produção é atualizado de forma automática, não havendo intervenção humana.

Desta forma os deploys em produção partirão obrigatoriamente do ambiente de homologação e serão disparados para equipe de infra via chamado (BMC). Caso tenha script de banco de dados, antes da execução do script um chamado deverá ser encaminhado para equipe de administração de dados para validação. O chamado deverá ser criado pelo responsável designado do sistema, e com o apoio das informações repassadas pelo time de arquitetura.

Branch Homologação: Essa branch é de total responsabilidade da equipe de ARQUITETURA do MEC, visto que a integração-continua escuta essa branch e qualquer atualização nesta, a pipeline de deploy é disparada e assim o ambiente de homologação é atualizado de forma automática, não havendo intervenção humana.

Branch Desenvolvimento: Essa branch é de total responsabilidade da equipe da FABRICA DE SOFTWARE do MEC, visto que a integração-continua escuta essa branch e qualquer atualização nesta, a pipeline de deploy é disparada e assim o ambiente de desenvolvimento é atualizado de forma automática, não havendo intervenção humana.

9.1 Estrutura dos ambientes no Gitlab

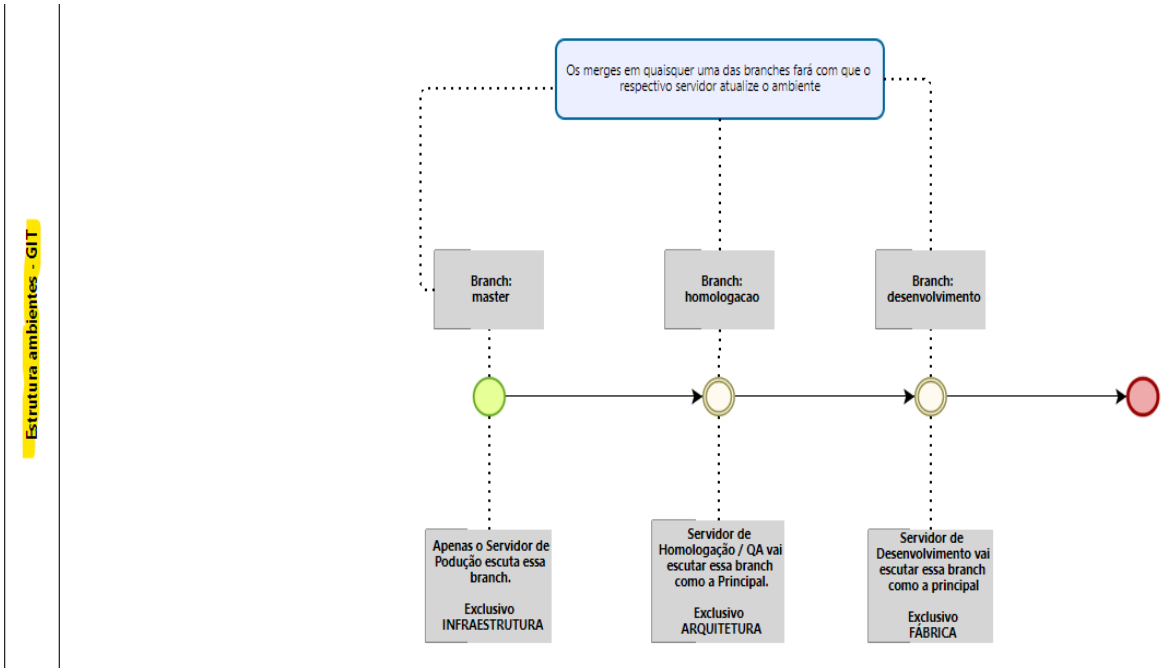

Figure 5. Estruturas de branch do gitlab

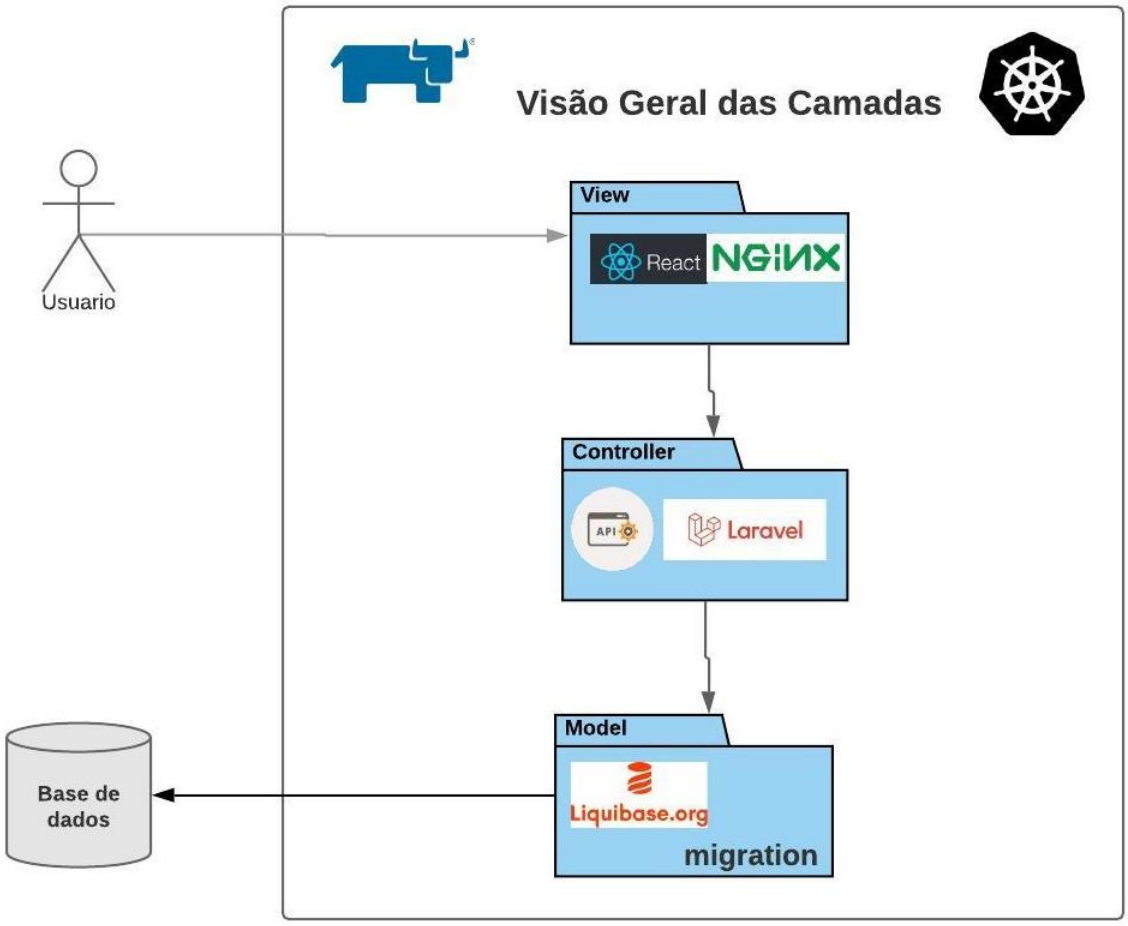

Figure 6. Visão geral das camadas

- **View:** é responsável pela interface que será apresentada, mostrando as informações do Model para o usuário
- **Controller:** É a camada de controle, responsável por ligar o Model e a View, fazendo com que os Models possam ser repassados para as Views e vice-versa
- **Model**: é responsabilidade dos Models é representar o negócio. Também é responsável pelo acesso e manipulação dos dados na sua aplicação

10 Requisitos para Implementação das Camadas

Os itens apresentados a seguir referem-se à lista de tecnologias e requisitos necessários para suportar as camadas de apresentação, negócio e persistência:

- PHP 7.4
- Composer
- Docker
- Laravel
- Symfony
- ReactJs

10.1 Bibliotecas a serem utilizadas pela aplicação:

- Dsgov
- React-dsgov-mec;
- Liquibase (symfony) ou Migration (Laravel)

Controle de Revisão			
Versão	**Data**	**Autor**	**Descrição**
1.1	28/05/2021	WISLEY ALVES DO COUTO	Atualização do documento
1.2	28/05/2021	JOSÉ CARLOS FERNANDES	Atualização do documento