SINTRA
TECNOLOGIAS DIGITAIS
ECONOMIA E SOCIEDADE

Licenciatura em Tecnologias Digitais e Segurança de Informação, 2ºano

Unidades Curriculares de Criptografia Aplicada, Programação para a Internet, Sistemas Distribuídos e Segurança

Docentes: Prof. Carlos Serrão, Prof. Thiago Bessa Pontes, Prof. João Pedro Pavia

Sistemas Para Armazenamento Seguro de Informação na Cloud

Trabalho realizado por:

Eliaquim Alexandre, nº 122051

Gonçalo Liberato Ferreira, nº 120037

Grupo 8

Abril de 2025

Índice

Análise do Modelo de Arquitetura

Introdução

O presente trabalho tem como objetivo apresentar o modelo de arquitetura do sistema para armazenamento seguro de informação em *cloud* do grupo 8. Esta apresentação será feita por meio de diagramas e curtos textos que expliquem em maior detalhe as operações e interações dos componentes envolvidos.

Existem dois principais tipos de interação relevantes de mencionar, que permitem ao sistema desempenhar as suas funções de forma eficiente e segura. Para ilustração dos mesmos por completo, o grupo recorrerá ao desenho de 8 diagramas.

Descrição do Modelo Principal

O cliente que pretenda conectar-se ao sistema de armazenamento em *cloud* interage primeiro com um servidor *NGINX*, que serve de *proxy*, estando este encarregue de grande parte dos mecanismos de segurança do sistema bem como roteamento de tráfego. Será configurado para funcionar como *Control Plane* da rede de *Kubernetes* do grupo, mais especificamente, *Kube-Scheduler* e *Kube-Proxy.*

Isso significa que alocará *Pods* em nós da rede desocupados *(Scheduler)* e encarregará diferentes *Pods* com tarefas de forma equilibrada, respeitando o limite máximo de recursos autorizados para alocamento *(Proxy).*

Cada *Pod* alocará um ou mais dos seguintes servidores:

- *Frontend*
- *Backend Principal*
- *Backend PKI (Public Key Infrastructure)*
- *DataBase*
- *Log Management*
- *HoneyPot*

Elementos como o HoneyPot, Log Management e Backend PKI não podem partilhar Pods com os restantes por questões de afinidade. O grupo procura isolar os mesmos devido ao facto de conterem informações críticas ou possibilitarem Privilege Escalation.

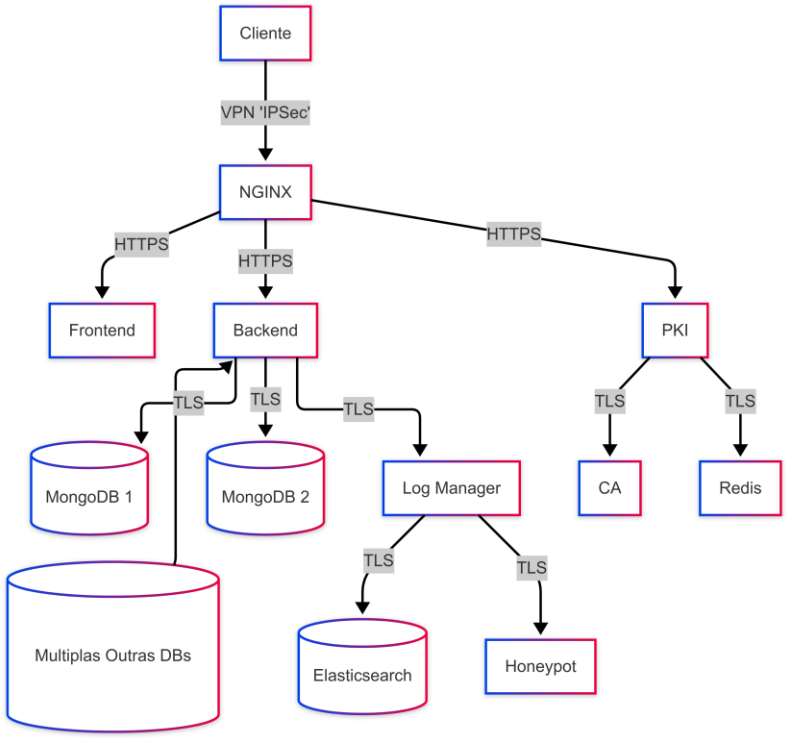

Figura 1- Fluxo Geral do Sistema Distribuído

É importante realçar que o sistema de armazenamento seguro de informação na cloud do grupo se trata, fundamentalmente, de uma rede de Kubernetes. Ou seja, embora o fluxo se faça nestes sentidos, a direção do tráfego nunca será tão linear como a ilustrada na figura 1. Para melhor compreensão, eis alguns dos componentes principais da rede e os seus papeis:

Rede Kubernetes

O fluxo geral do sistema é garantido pelos seguintes componentes:

- *kube-scheduler*
    i. Decide em que *Node* cada *Pod* será executado.

Critérios: Disponibilidade de recursos (*CPU, RAM*), afinidade/anti-afinidade, *taints/tolerations*.

Exemplo: O *Pod VPN* (*StrongSwan*) é colocado no *Node* 1 para garantir baixa latência.

- *kube-controller-manager*
    ii. Garante que o estado desejado (e. G. 3 réplicas do *Backend*) é mantido.
    iii. Gere *ReplicaSets* e recria *Pods* em caso de falha.
- *etcd*
    iv. Armazena o estado do *cluster* (ex: configurações de *Pods*, *Services*).

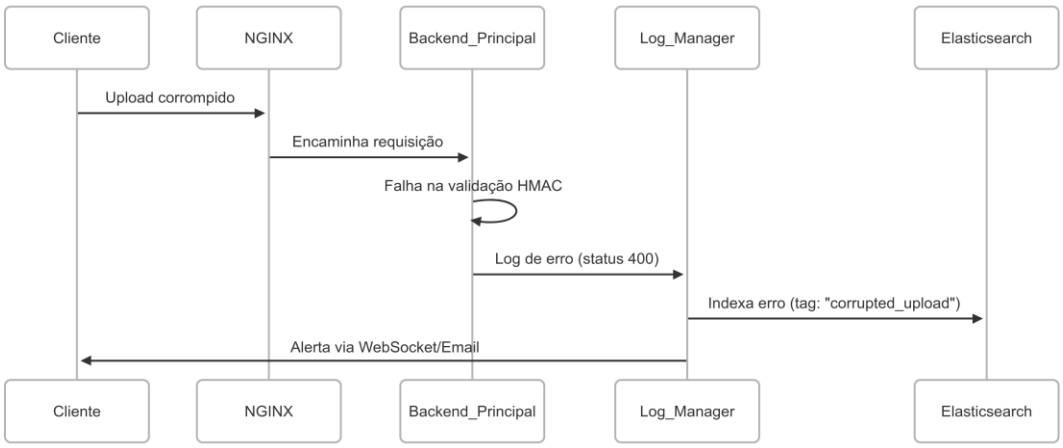

Figura 2- Operações que oferecem tolerância a falhas

Os três principais tipos de interação dentro do sistema:

Comunicação Cliente-Servidor NGINX

1. Pré-Requisitos

- *strongSwan*

O *strongSwan* é um *software* (mais especificamente, um conjunto *de* ferramentas *e* bibliotecas) que permite implementar *VPN*s baseadas no protocolo *IPsec* e/ou *IKEv2*. O facto de ser *open-source* e permitir o uso, alteração e distribuição do seu código fonte são as principais motivações por trás da escolha. Porém, uma vez que tem integração com dispositivos móveis e suporta autenticação via certificados digitais, *Pre-Shared Keys (PSK)* ou *Extensible Authentication Protocol (EAP),* tornou-se uma opção indispensável para o grupo.

- Certificados Digitais Assinados pelo grupo

Um certificado emitido pela *PKI* do sistema (*X.509*) para o cliente, e um certificado emitido pela e para autenticação da própria empresa, com a finalidade de permitir a validação por parte dos clientes.

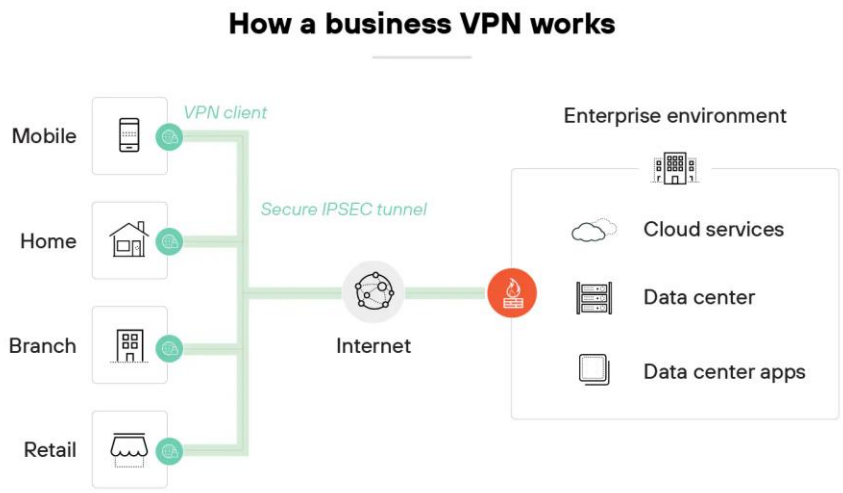

Figura 3- Como funcionará a *VPN*, ícone da FW representa o servidor NGINX

2. Estabelecimento do túnel *IPsec (IKEv2)*

2.1. Fase 1

O cliente inicia a comunicação com o servidor *VPN strongSwan* via *UDP*/500, sendo que a resposta será um conjunto de algoritmos recomendados para estabelecer uma ligação segura. Os algoritmos escolhidos pelo grupo serão:

I. Criptografia: *AES-256-GCM*
II. Integridade: *SHA384*
III. Grupo *Diffie-Hellman*: *MODP_3072*

Estas duas entidades trocarão *nounces* (números de série utilizados uma vez por sessão) para evitar ataques de repetição.

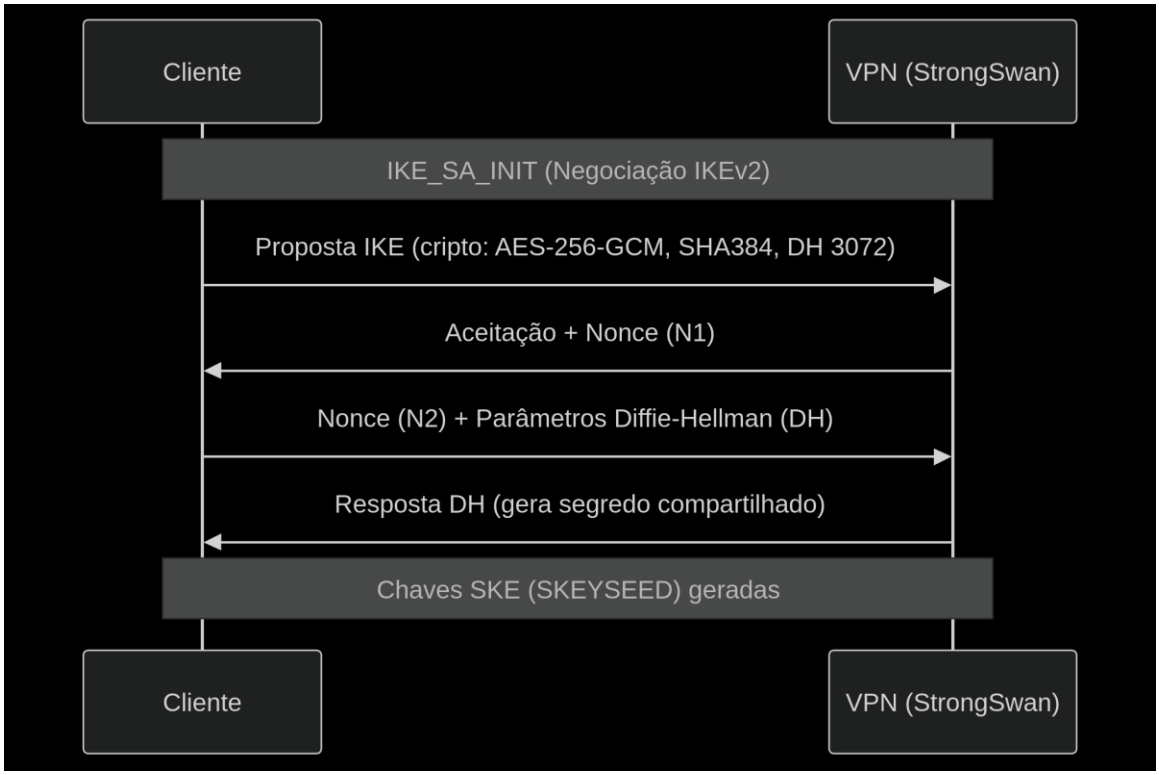

Figura 4- Negociação de algoritmos

2.2. Fase 2

O cliente começa por enviar ao servidor *VPN* o seu certificado digital, assinado pela empresa. Este certificado passa por um processo de validação na *PKI* do sistema, que responde com uma mensagem de validação. O servidor *VPN*, que contém uma cópia do certificado digital assinado pela empresa, envia o mesmo e espera pela validação do cliente.

Uma vez confirmada a identidade de cada uma das entidades, estas começam um processo de definição de parâmetros para o túnel de comunicação seguro, sendo obrigatória

a implementação de pelo menos um dos protocolos: AH (Authentication Header), ESP (*Encapsulating Security Payload)*.

O cliente não deve guardar as chaves trocadas no disco rígido por motivos de segurança, enquanto o servidor o fará em memória *kernel,* no formato *xfrm.*

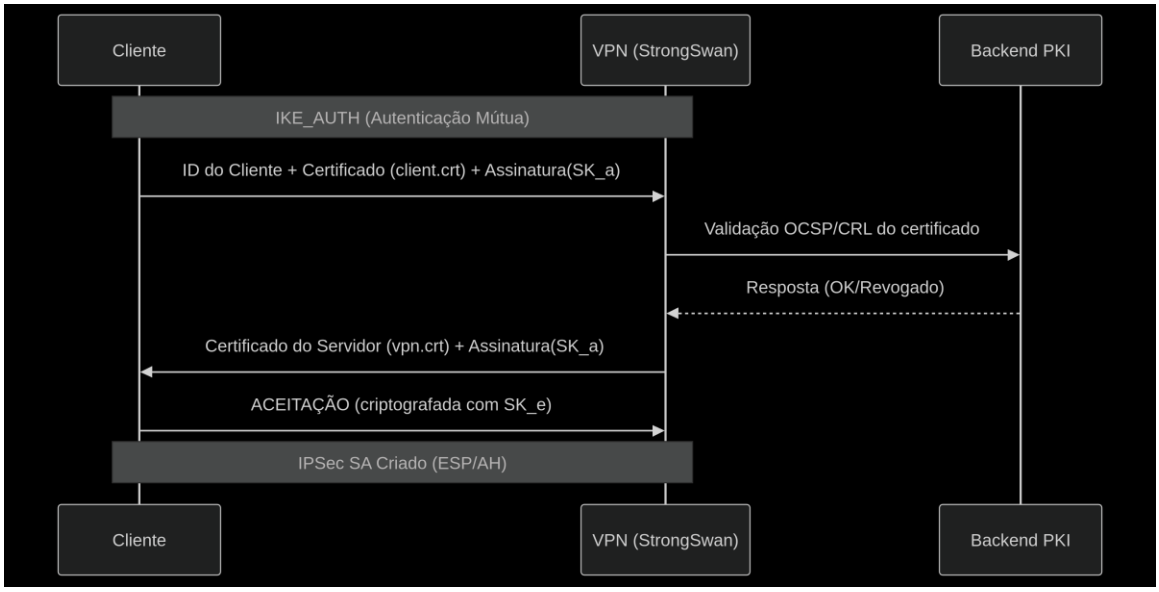

Figura 5- Troca de certificados

Eis como é feita a troca de ficheiros depois de estabelecida a camada de segurança *IPsec-SA* (modo túnel com *ESP* e/ou *Autentication Header*):

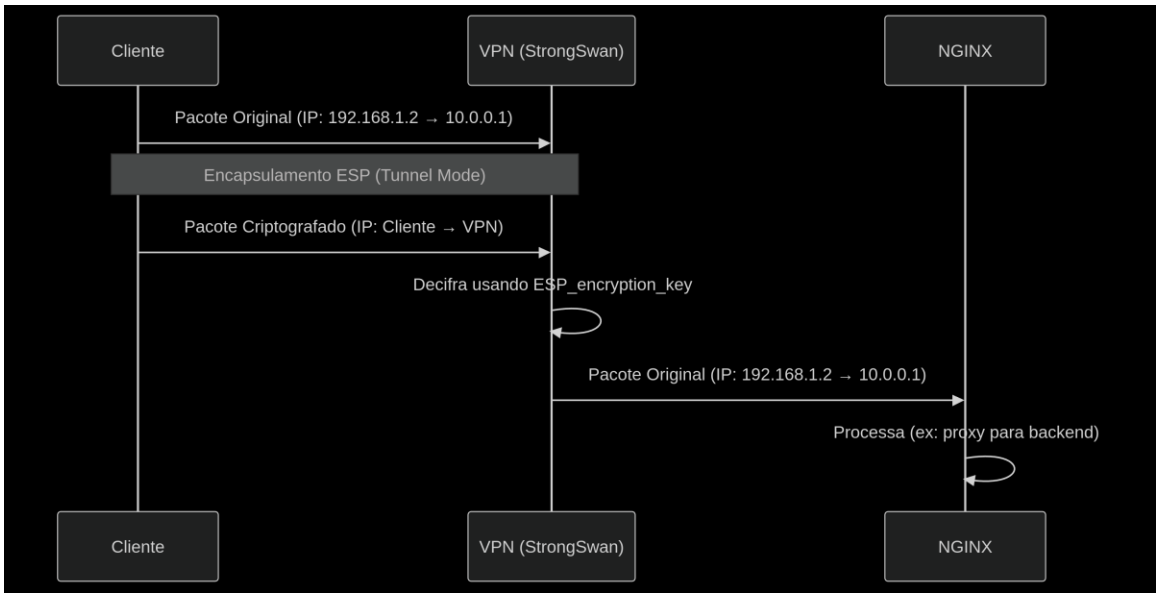

Figura 6- Troca de dados no modo túnel

Comunicação entre *Pods* de Serviço Crítico

A comunicação interna, entre *Pods* de serviço crítico, como *DB* e *PKI*, deve ser ainda assegurada pelo protocolo *TLS*. Uma vez que se corre o risco de elevar demasiado a latência, será configurado o *mTLS* (*TLS* mútuo) para utilização da versão 1.3, que reduz o *handshake* para *1-RTT* (*One Round-Trip Time*) em comparação a versões mais antigas do TLS. O grupo admite ainda não ter testado a ideia, mas tenciona colocar a mesma em prática no caso de ser viável. Eis um exemplo de comunicação entre um servidor *Backend* e uma *DataBase*:

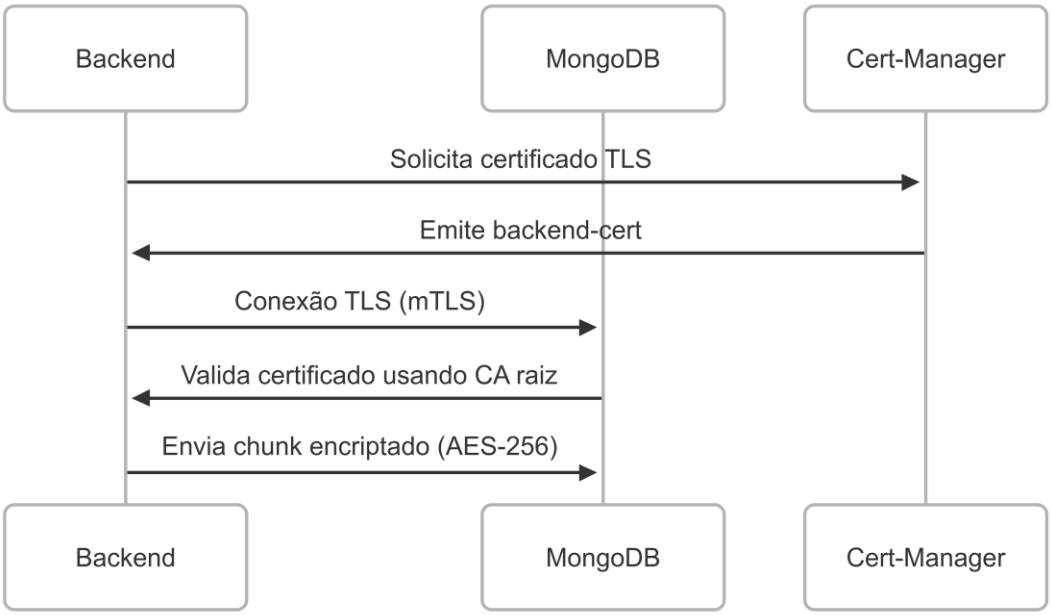

Figura 7- Comunicação iniciada com *handshake TLS* entre *BE* e *DB*

Comunicação entre Pods de Serviços Não-Críticos

As restantes comunicações serão feitas por *HTTP*. Os mecanismos de segurança já mencionados tratarão de proteger a rede, e como tal, o grupo decidiu apenas adicionar uma camada extra de protocolos a alguns dos serviços. Procura-se evitar uma velocidade de transferências baixa, bem como uma enorme latência nas comunicações menos críticas. Segue-se um diagrama do fluxo geral do sistema, sendo que a verde se podem encontrar os servidores cujas comunicações são protegidas por *TLS:*

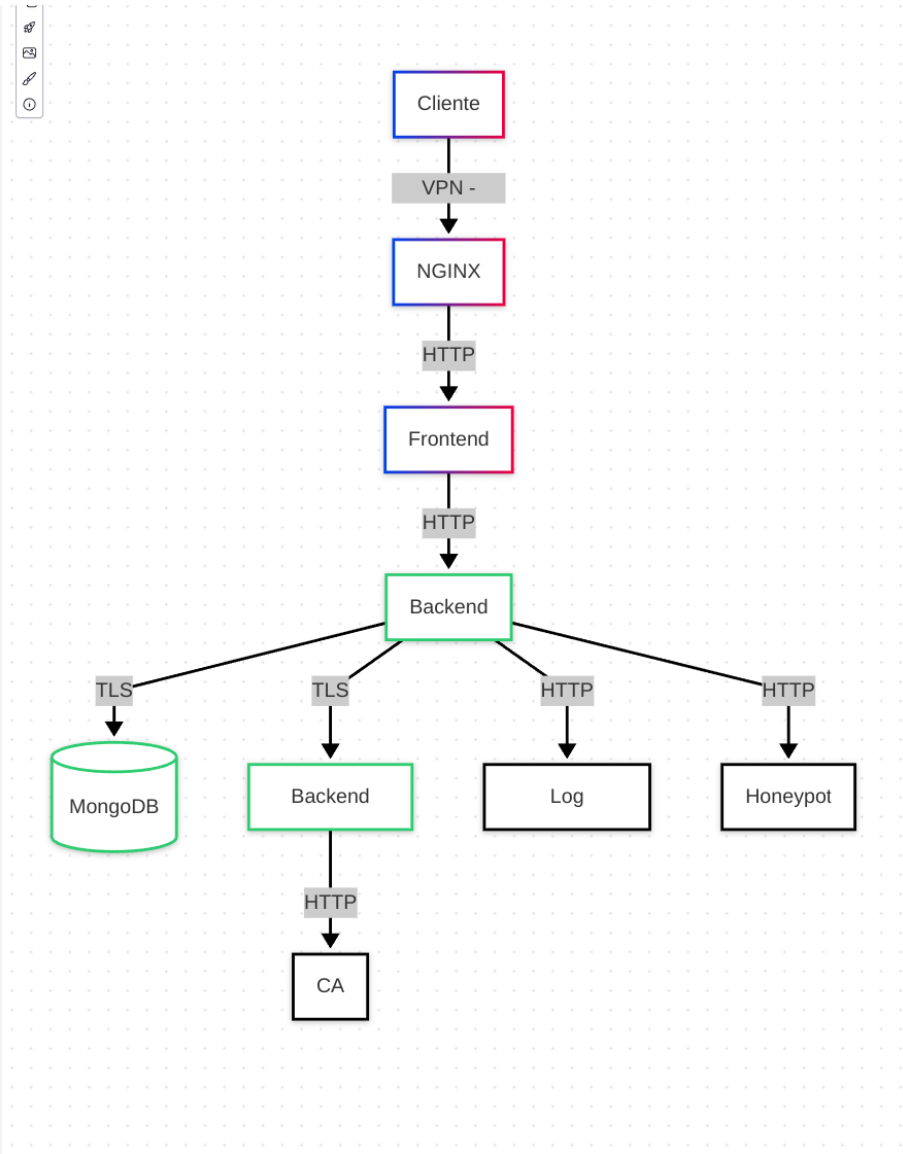

Figura 8- Fluxo geral do sistema

Conclusão

A arquitetura proposta para o sistema de armazenamento em *cloud,* baseado em *Kubernetes, Docker* e criptografia de ponta a ponta, demonstra um equilíbrio eficaz entre segurança, escalabilidade e desempenho. Ao integrar componentes como um *proxy* NGINX, uma rede de *pods* distribuídos para processamento de dados, um *backend PKI* para gestão de certificados digitais, e um *honeypot* para deteção de intrusos, o sistema atende aos requisitos funcionais e não funcionais definidos na tarefa anterior.