FACULDADE DE TECNOLOGIA DE TAUBATÉ
CURSO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL MARQUES DE ALMEIDA

ANÁLISE E DESENVOLVIMENTO DE UM MIDDLEWARE DE SEGURANÇA PARA APIS

TAUBATÉ – SP
2021

FACULDADE DE TECNOLOGIA DE TAUBATÉ
CURSO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL MARQUES DE ALMEIDA

ANÁLISE E DESENVOLVIMENTO DE UM MIDDLEWARE DE SEGURANÇA PARA APIS

Trabalho de Graduação apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Centro Estadual de Educação Tecnológica Paula Souza para a obtenção do diploma de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. M.e Luiz Eduardo Souza Evangelista
Co-orientador: Prof. M.e Fernando Masanori Ashikaga

TAUBATÉ – SP
2021

RAFAEL MARQUES DE ALMEIDA

ANÁLISE E DESENVOLVIMENTO DE UM MIDDLEWARE DE SEGURANÇA PARA APIS

Trabalho de Graduação apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Centro Estadual de Educação Tecnológica Paula Souza para a obtenção do diploma de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. M.e Luiz Eduardo Souza Evangelista

Co-orientador: Prof. M.e Fernando Masanori Ashikaga

Taubaté, _____ de _____ de 2021.

BANCA EXAMINADORA

_____

Prof. M.e Luiz Eduardo Souza Evangelista

Faculdade de Tecnologia de Taubaté

_____

Prof. M.e Fernando Masanori Ashikaga

Faculdade de Tecnologia de São José dos Campos

_____

Prof. Dra. Divani Barbosa Gavinier

Faculdade de Tecnologia de Taubaté

Dedico este trabalho aos meus avós por demonstrarem que uma vida com propósitos, princípios, virtudes e disciplina, é o único caminho que permite conquistar a vida de maneira íntegra e virtuosa.

AGRADECIMENTOS

Aos meus familiares, que apresentaram os reais valores da vida e o impacto gerado por cada ação ou decisão tomada.

A minha tia Cleide por me capacitar no universo musical e em diversas áreas da vida.

Ao meu tio Reginaldo por me proporcionar momentos e ensinamentos inesquecíveis na infância.

A todos os mestres, professores e amigos que participaram da minha formação estudantil e profissional.

Aos meus senseis, mestres, amigos e adversários da modalidade desportiva do Judô, estes que me ensinaram que sozinho não se chega a lugar algum, que para avançar uma etapa deve-se tentar repetidas vezes até compreender todas as variáveis necessárias para superar e entre estes citados, um agradecimento especial ao meu pai, por ser meu sensei no tatami e shihan na vida.

Ao meu atual gestor Luiz Gewers, este quem permitiu a minha atuação e entrega do meu primeiro projeto de desenvolvimento de sistemas a nível profissional, projeto este que viabilizou a criação e conclusão deste trabalho de graduação. Além do projeto, concedeu oportunidades em que pude amadurecer e evoluir profissionalmente.

A todos os amigos e adversários que me incentivaram, acreditaram, duvidaram e me desafiaram a ser quem eu pude me tornar.

A minha companheira Vanessa, esta quem tem muita participação na obtenção deste mérito e venceu diversos desafios em conjunto.

Ao mestre Fernando Masanori, por me instruir, auxiliar e apresentar a tecnologia que hoje me permite obter provisões e autodesenvolvimento.

Ao mestre Evangelista a quem auxiliou e se disponibilizou em diversas etapas na conclusão dessa graduação.

A todos os profissionais e mestres das FATECs de Taubaté, São José dos Campos e ao Centro de Paula Souza pois toda a dedicação e trabalho permitiram em que eu pudesse me aprimorar e me desafiar constantemente.

Ao Criador, quem permitiu que todas essas existissem através de suas promessas e graça.

Nunca se orgulhe por vencer um adversário, ao que vencestes hoje, poderia derrotar-te amanhã. A única vitória que perdura é a que se conquista sobre a própria ignorância.

(J. Kano)

RESUMO

Apresenta-se neste trabalho o desenvolvimento de um sistema responsável por assegurar a integridade do ambiente de produção da empresa Multtv Consultoria e Locacoes de Equipamentos SA. A MultTV é uma empresa Business to Business (B2B) que viabiliza serviços de TV ao vivo por IP (IPTV) para provedores de internet e para tal, realiza-se o transporte dos canais via satélite a partir de um centro de tratamento e distribuição de áudio e vídeo (headend) próprio da empresa até os provedores. Além do transporte, oferta-se um sistema para os provedores, ao qual disponibiliza recursos para gestão operacional dos produtos e clientes de IPTV da operação. Esse sistema de gestão nomeado como Beenius, permite que provedores possam integrar o novo sistema com os sistemas de Gestão de Relacionamento com o Cliente (CRM) ou Sistema integrado de gestão empresarial (ERP) em operação, através de uma Interface de Programação de Aplicação (API). Apesar do recurso tecnológico possuir um alto nível de complexidade, sua estrutura sistêmica considera que a utilização da API seja utilizada de maneira interna e de modo administrativo, diferente do modelo de negócio adotado pela MultTV que o emprega de maneira compartilhada aos clientes e após estudos na documentação desse sistema, identificou-se uma vulnerabilidade em que um serviço principal da API é capaz em apagar todas as operações existentes com uma simples requisição. Além dessa vulnerabilidade, o sistema não produz registros das requisições ocorridas via API e não dispõe de recursos para gerenciar a autenticação e autorização dos usuários, portanto, viabiliza-se um projeto para tratar esses pontos focais. Este trabalho descreverá todas as etapas necessárias entre o levantamento de requisitos até a conclusão e entrega do projeto.

Palavras-chave: middleware de segurança. soap api. beenius. authbee.

ABSTRACT

This work presents the process of a system development responsible to grant the integrity of Multtv Consultoria e Locacoes de Equipamentos SA production environment company. MultTV is a B2B company that provides live TV over IP (IPTV) services for internet providers, so, the channels transportation is made via satellite, originated from a audio and video treatment and distribution center (headend) from the company to providers also, a management system is offered with the channels transport to providers, which provides a lot of resources to the operation's IPTV products and customers management. This management system named Beenius, allows providers to integrate the new system with the existent Customer Relationship Management (CRM) or an Integrated Business Management System (ERP) systems from operator side, through an Application Programming Interface (API). Although the technological resource has a high complexity level, that systemic structure considers that API is must be used like internally and administratively way, different from the business model adopted by MultTV, which API usage is running in a shared way with customers and after studies about the system documentation, a vulnerability has been identified inside a core service from API is able to erase all existing operations with a simple request. In addition to this vulnerability, the system does not produce records of requests made via API and does not have resources to manage the authentication and authorization of users, therefore, a project has been allowed to fix those focal points. This work will describe all necessary steps between requirements gathering, up to completion and delivery the project.

Keywords: security middleware. soap api. beenius. authbee.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicação
B2B	Business to Business
BITS	Dígitos Binários
CDN	Rede de fornecimento de conteúdo
CPU	Unidade Central de Processamento
CRM	Gestão de Relacionamento com o Cliente
CSS	Folha de Estilo em Cascata
CSV	Valores Separados Por Virgula
DNS	Sistema de Nomes de Domínio
DOC	Extensão para Arquivos de Processamento de Textos Proprietário da Microsoft
DOCX	Extensão para Arquivos de Processamento de Textos Proprietário da Microsoft
DSF	Django Software Foundation
ERP	Sistema integrado de gestão empresarial
HTML	Linguagem de Marcação de Hipertexto
HTTP	Protocolo de Transferência de Hipertexto
HTTPS	Protocolo de Transferência de Hipertexto Seguro
IP	Endereço de Protocolo da Internet
IPTV	TV ao Vivo por IP
JS	JavaScript
NoSQL	Banco de dados Não Relacional
PSF	Python Software Foundation
REST	Transferência Representacional de Estado
RPC	Chamada de Procedimento Remoto
SGBD	Sistema de Gerenciamento de Banco de Dados
SO	Sistema Operacional
SOAP	Protocolo Simples de Acesso a Objetos
SQL	Linguagem de Consulta Estruturada

TAG	Palavra-chave em linguagens de marcação
URL	Localizador Uniforme de Recursos
WEB	Rede
WSDL	Linguagem de Descrição de Serviços Web
XLSX	Extensão para Arquivos de Processamento de Planilhas Proprietário da Microsoft
XML	Linguagem de Marcação Extensível
XML-RPC	Protocolo de Chamada de Procedimento Remoto

SUMÁRIO

1 INTRODUÇÃO

O ecossistema tecnológico mescla-se com a realidade física e seus sistemas são indispensáveis na execução de processos ou tomadas de decisão. Para harmonizar e potencializar o progresso deste universo, as interfaces de programação de aplicação ou API de maneira abreviada, permitem que estes sistemas se comuniquem e integrem-se de modos e propósitos distintos.

Uma API é um conjunto de instruções e parâmetros pré-estabelecidos por uma aplicação ou serviço, permitindo o consumo das suas funcionalidades por outras aplicações e serviços que não pretendem implementar ou replicar o mesmo sistema, mas apenas consumirem os seus serviços.

O objetivo principal de uma API é ofertar publicamente suas funções, para que qualquer indivíduo ou sistema seja capaz em consumi-la de alguma forma, seja para pequenas consultas ou realizar integrações entre sistemas de linguagens distintas, de maneira ágil.

A proposta em não ser mais necessário sair de casa para enfrentar filas nos bancos, ir até a uma lotérica para quitar um boleto ou realizar compras, é sustentada pela alta disponibilidade dos serviços web atualmente. Estes serviços são APIs projetadas para comunicarem-se através das redes, sejam estas locais ou globais.

Cada ação ocorrida na interface de uma aplicação, executa diversas chamadas de sistema nos bastidores e a cada requisição, recebe uma resposta de acordo com a ação executada. Quem seria o ator principal nos bastidores? Uma API, de fato.

Quando acessamos um aplicativo pelo celular, um site de e-commerce ou qualquer serviço web, este provavelmente está trabalhando em conjunto a uma ou diversas APIS. É comum identificarmos em um formulário de cadastro, uma função que permite um cadastro de maneira automatizada, utilizando nossas contas existentes na Google, Facebook etc.

Isto só ocorre por haver uma API trabalhando para que o processo de um novo registro de usuário seja mais dinâmico, enquanto os sistemas trocam informações entre si, sem que possamos perceber, diversas chaves são trocadas, entregando e validando dados como nossos nomes, endereços eletrônicos, informações pessoais e outros dados relevantes ao processo de autenticação.

Tendo em vista a crescente demanda em se otimizar a integração e o desenvolvimento de novas aplicações baseadas no reaproveitamento e disponibilidade das APIs, fora identificado que a preocupação nos requisitos de segurança não recebe a devida atenção ou os cenários em que uma aplicação poderia ser exposta, não foram considerados durante os levantamentos realizados. Além dessa visão, fora realizado uma pesquisa em campo sobre este cenário e os sistemas atuantes na área de segurança, ou são escassos ou são sistemas dedicados a projetos específicos.

Considerando essas informações como uma necessidade e um cenário propício de entrada, torna-se proveitoso o desenvolvimento de um serviço estruturado como um middleware de segurança, que irá gerenciar o acesso a APIs, controlar as possíveis requisições e respostas durante as interações com a API e disponibilizar dados como log de registros ou estatísticos.

O tema propõe tratativas para questões de segurança e auditoria durante o ciclo de vida de API utilizada em integrações entre sistemas ou o seu consumo direto.

Uma API é criada quando uma empresa de software tem a intenção em que outros criadores de software desenvolvam produtos associados ao seu serviço e muitas destas APIs possuem funções críticas que poderiam comprometer toda uma base de dados se não receber devidas prevenções, para isto, idealiza-se um middleware que esteja alocado entre o serviço da API e os clientes da MultTV.

Existem três principais tipos de API disponíveis e para esta contextualização, descreveremos a do tipo privado. APIs privadas são utilizadas internamente por um nicho e permitem diversas integrações entre sistemas.

Algumas destas APIs de formatação privada possuem funções críticas que poderiam comprometer toda uma base de dados se não receberem devidas prevenções e se utilizadas de maneira compartilhada.

Para isto, idealiza-se um middleware que esteja alocado entre o serviço da API e as requisições de origem com destino a API, originadas nos clientes da MultTV. Existem diversas formas para se controlar o acesso e garantir a integridade de uma API, autenticar e autorizar o requerente de maneira válida, para tratar estes processos, responsabiliza-se este middleware para tratar as requisições e autorizar o consumo aos serviços desta API, somente para clientes devidamente cadastrados no middleware de segurança. Após considerar-se os diversos cenários, níveis de exposição e possíveis tratativas como resolução, pergunta-se: É possível permitir a utilização de uma API privada, de maneira compartilhada ou garantir bons níveis de segurança em serviços web?

2 REFERENCIAL TEÓRICO

Apresenta-se neste capítulo o estudo e análise de possíveis cenários em que uma API ou um serviço Web, comprometem a integridade e a autenticidade dos dados pelo modo em que estes estão dispostos e são disponibilizados. Tecnologias e meios que garantam a integridade serão explanados e dá-se ênfase em um middleware ao qual fora produzido como solução para gerenciar o acesso a uma API SOAP dedicada, enquanto sua utilização ocorre de maneira compartilhada. Nomeou-se este middleware por Authbee, formado pela junção entre o substantivo autorização traduzido para o inglês e o nome do sistema tratado (Beenius), enfatizando a alusão ao tratamento de segurança recebido por este sistema.

Durante a elaboração deste referencial, constatou-se que seria relevante decorrer sobre as linguagens, base de dados e os requisitos necessários, dos quais permitiram a primeira versão funcional da solução.

2.1 LEVANTAMENTO E ANÁLISE DE REQUISITOS

A maturidade de uma aplicação ou sistema é fortemente ligada ao quanto esta análise de requisitos recebe atenção e é desenvolvida. Para que um projeto tome forma, os atores envolvidos necessitam dominar a compreensão sobre o resultado esperado e como suas funcionalidades serão aplicadas. Estas etapas podem ocorrer de maneira objetiva enquanto atores estiverem dispostos a coletar estes requisitos, entender os possíveis cenários em que essa aplicação irá atuar, quais as possíveis resoluções para cada cenário e após validar o que de fato é relevante, enfatiza-se as atividades que devam ser priorizadas.

2.2 MÁQUINAS VIRTUAIS

Através de processos de virtualização, permite-se em que seja criado um ambiente virtual, isolado da camada física onde consome uma fração dos recursos físicos de hardware disponíveis e essa fração é determinada nas configurações desse ambiente, durante o processo de criação da máquina ou tanto quanto após a existência de uma configuração inicial. Existem sistemas que permitem que o acréscimo ou decréscimo de recursos disponíveis ocorram com o ambiente em execução. Essas máquinas virtuais possuem como propósito alocar sistemas operacionais pretendidos, de modo que um sistema computacional de alta performance tenha melhor aproveitamento tratando-se do consumo dos seus recursos e a virtualização disponibiliza mais recursos de segurança ao sistema como um todo.

2.2.1 VMWARE

Conforme a documentação oficial da VMWare (entre 1998 e 2021), é uma organização que possui produtos focados em virtualização de sistemas computacionais. Para a execução desse projeto, adotou-se o sistema de virtualização VMWare ESXi, pois trata-se da solução atual no ambiente de produção da empresa para os processos de virtualização de sistemas operacionais e este recurso fora considerado para disponibilizar um ambiente virtual e seguro para alocar o Authbee.

2.3 FEDORA SERVER

Fedora Server é um sistema operacional (SO) Linux/Unix Like escolhido para ser o sistema principal dessa máquina virtualizada. Os fatores que influenciaram nessa decisão é que conforme a documentação oficial em Fedora Project (entre 2003 e 2021) este possui uma comunidade ativa em seu aprimoramento, sua criação tem origem na Red Hat e este atualmente é um dos principais investidores neste projeto, sendo essa uma empresa norte-americana que possui como produtos diversos sistemas operacionais e soluções de desenvolvimento, focados em segurança cibernética. Além da qualidade constatada em sua origem, esse SO possui um cockpit de gerenciamento remoto, ou seja, se este servidor possuir um endereço com Sistema de Nomes de

Domínio (DNS) público, é possível acessá-lo, monitorá-lo e gerenciá-lo de qualquer lugar.

2.4 PARADIGMAS DE PROGRAMAÇÃO

Antes que se inicie a codificação de qualquer programa ou aplicação, deve-se relacionar a forma de desenvolvimento ao propósito de sua utilização. A maior preocupação com o paradigma utilizado é como o produto será executado e como se comportará. O paradigma orientado a objetos atende a necessidade de um sistema que necessita ser dividido em diversos componentes interdependentes e se comportem como um organismo, onde estes são capazes em invocar e consumir suas classes e funções. “Alan Kay sugere em seu postulado que um sistema se comporte como um organismo vivo, enquanto cada elemento autônomo atua em conjunto a outros para atingir um objetivo, criando assim o conceito orientado a objetos”.

2.5 LINGUAGENS DE PROGRAMAÇÃO

Os ambientes computacionais são formados por diversos elementos interdependentes, que garantem essa sensação em que tudo ocorre em tempo real, contínuo e que diversas aplicações e sistemas são executados a modo simultâneo.

O objetivo não é citar em como os sistemas operam, mas como as linguagens de programação são as responsáveis por toda a evolução ocorrida ao decorrer dos anos.

Uma linguagem de programação é formada por diversos componentes, enquanto cada uma obedece a uma sintaxe e semântica, estas são utilizadas para transformar instruções humanas em procedimentos computacionais. Para que isto ocorra, é necessário um tradutor que compreenda as instruções recebidas e as transforme em linguagem computacional e estes tradutores se subdividem entre as linguagens compiladas ou interpretadas. Além dos interpretadores, uma instrução humana gerada em um sistema computacional, passará por diversas camadas de abstração no sistema, até que esta instrução ou dado seja convertido em bits e estes possam ser processados pela CPU. Utiliza-se uma linguagem computacional para criar instruções que executem e automatizem processos computacionais.

2.5.1 PYTHON

Segundo a Python Software Foundation (entre 2001 e 2021), Python é uma linguagem de alto nível, interpretada com suporte a diversos paradigmas de programação e é multiplataforma. Estes diversos paradigmas suportados, tornam-na versátil para atingir diversos propósitos que possam surgir em um projeto. “Esta linguagem fora desenvolvida pelo Guido Van Rossum ao final da década de 80”.

2.5.1.1 VERSÃO DO PYTHON

A caráter de segurança, sempre se atualiza as versões das ferramentas utilizadas e até o momento deste documento, a versão utilizada é a 3.10.1.

2.5.1.2 CARACTERÍSTICAS DO PYTHON

O Python é uma linguagem fortemente tipada e possui um alto nível a tratamento de exceções. Seu maior potencial é a praticidade na manutenção e agilidade durante o desenvolvimento de código-fonte.

2.5.1.3 VANTAGENS DO PYTHON

Ele abstrai a parte complexa em tratar listas, dicionários, índices e para quesitos de segurança, é uma das linguagens mais utilizadas para este propósito.

Comparando com outras linguagens de programação, ele possui desenvolvimento muito ágil, pois descarta questões como “inserir ponto e vírgula (;) ao final de toda linha ou chaves ({ }) para indicar o final de um bloco de código.

Opta-se por sua utilização pois é a linguagem de domínio neste curso de graduação, logo, fora a primeira escolha para este projeto.

2.5.1.4 BIBLIOTECA ZEEP

Conforme Tellingen (2016), esta biblioteca disponível para Python, inspeciona o documento WSDL existente em uma SOAP API e gera um código correspondente em que torna um cliente capaz em realizar requisições de serviços web.

Esta biblioteca tornou-se crucial para tratar as requisições da SOAP API, em vista que durante os processos de desenvolvimento, nenhuma outra forma fora identificada para lidar com este tipo de protocolo.

2.5.2 JS

JavaScript (JS) é uma linguagem de programação voltada para desenvolvimento em soluções web, como jogos ou simples páginas institucionais. Junto ao Python, é uma das linguagens de script mais eficientes na atualidade.

2.6 LINGUAGENS DE MARCAÇÃO E ESTILIZAÇÃO

linguagens de marcação e estilização são utilizadas para definir o formato de um arquivo e como este será interpretado. Para que um arquivo receba essa marcação, atribui-se uma extensão ao final de seu nome, segue-se exemplos como “site.html, dicionario.csv, texto.docx, planilha.xlsx”. Através destas estruturas de marcação ou “tags”, diversos sistemas são capazes em interpretar corretamente o conteúdo de qualquer arquivo.

2.6.1 HTML

Um arquivo HTML que possui sua extensão declarada com a sigla do próprio nome (.html), conforme MDN Web Docs (entre 2005 e 2021) é um arquivo estruturado que permite criar páginas web, utilizando identificadores, invocando scripts em linguagens como JS para manipular suas funcionalidades, arquivos CSS para estilizar seu visual. A principal funcionalidade de uma página HTML é ofertar uma interface entre o usuário e o serviço web.

2.6.2 CSS

Um arquivo CSS que possui sua extensão declarada com a sigla do próprio nome (.css), é um arquivo estruturado em dicionários (parâmetros e valores) utilizado para estilizar graficamente, páginas estruturadas em linguagens de marcação como HTML.

2.6.3 XML

Um XML que possui sua extensão declarada com a sigla do próprio nome (.xml), segundo Liam (2016), é um arquivo estruturado de marcação baseado em hierarquia ou herança. Cada nível ou elemento possui parâmetros diacríticos para identificá-los, permitindo sua localização ou chamada de maneira externa ou remota, caso esteja arquivo seja manipulado por um sistema.

2.7 BASE DE DADOS

O código-fonte recebe instruções para se comunicar e autenticar-se com a base de dados. Na base de dados armazena-se dados, como: IP público das operações que possuem acesso a API, logs de registro das funcionalidades consumidas e diversos dados pertinentes ao propósito.

2.8 SGBD

Um sistema de gerenciamento de banco de dados (SGBD) realiza a gerência de uma base de dados. Permite maior usabilidade e eleva os níveis de segurança e gerência da base. O SGBD utilizado para este projeto é de estrutura NoSQL.

2.8.1 MONGODB

Segundo a documentação oficial do MongoDB Inc. (entre 2007 e 2021), o MongoDB é um banco de dados não-relacional, que armazena registros em formatos de dicionário, ou seja, em formato JSON. Estes dicionários são formados por chaves e cada chave recebe um ou diversos valores. Como a API demanda de requisições e respostas em arquivo estruturado, uma base de dados que armazena registros em formato de dicionário soou como a melhor proposta para atender o desenvolvimento deste middleware.

Para facilitar o processo de implementação do middleware, criou-se uma instância local do MongoDB Enterprise Server para servir como base de dados, no mesmo servidor virtual que aloca o middleware de segurança.

2.9 WEB SERVICES

Serviços WEB são sistemas estruturados que disponibilizam acesso a funcionalidades consumíveis através da rede, por requisições. Esta rede pode ser local quanto global. Este serviço por si é estruturado de maneira similar a uma API, porém, uma API não pode ser considerada como um serviço WEB, tendo em vista que um serviço possui interfaces visuais ao usuário.

2.10 APIS

Conforme Negresiolo (2019), APIs são recursos ou serviços de um sistema disponíveis para consulta através de requisições por outros sistemas ou clientes. Existem diversas maneiras em se estruturar uma API e suas funcionalidades mas o principal objetivo em se estruturar uma API ou integrar sistemas via API é que essas funcionalidades podem ser distribuídas sem que os sistemas necessitem serem equivalentes, considerando a compatibilidade entre sistemas, plataformas, sistemas operacionais, linguagens utilizadas na implementação do sistema como um todo, ou seja, através deste tipo de tecnologia é possível sintetizar o consumo desses serviços,

encurtando o tempo necessário para atuar entre possíveis cenários e processos que possam surgir.

2.11 REQUESTS

Requisições são chamadas realizadas a um serviço, através de protocolos e sockets existentes nas camadas computacionais de rede.

2.11.1 HTTP

O HTTP é o protocolo padrão de comunicação de dados na web. Este atua na camada de aplicação de rede e ocorre na comunicação via TCP/IP.

2.11.2 WSDL

Segundo a W3C (entre 1999 e 2019), este recurso estruturado em linguagem XML, define como um serviço web deve ser requisitado por cliente, especificando a composição esperada em uma solicitação aos serviços existentes.

2.11.3 XML-RPC

Conforme Winer (1999), XML-RPC trata-se de um protocolo que permite a solicitação de procedimentos remotos estruturados em linguagem XML e utiliza-se o protocolo HTTP como veículo de transporte das requisições.

2.11.4 SOAP

Segundo a W3C (entre 1999 e 2020), SOAP baseia-se em linguagem XML, define o formato de uma mensagem de solicitação ou resposta feita via requisições web, como em uma SOAP API, onde suas chamadas ocorrem via XML-RPC. Essas mensagens trafegam via protocolo HTTP e suas chamadas são definidas pelo recurso WSDL.

2.12 IP

IP é a abreviação para “Protocolo de Internet”. Ele é um protocolo de rede e é o principal responsável pela comunicação pela internet. Sua finalidade é endereçar e encaminhar pacotes de dados que trafegam pela rede. O Middleware deste projeto rastreia qual é o IP de origem de cada requisição e checa se este IP possui permissão para consumir as funcionalidades disponíveis na API.

2.13 FRAMEWORKS

Conforme Noleto (2020), um framework no âmbito do desenvolvimento computacional, resume-se a uma coleção pré-definida de recursos, bibliotecas e funcionalidades cuja pretensão é automatizar, facilitar processos e encurtar o tempo de desenvolvimento de soluções e rotinas de testes.

2.14 DJANGO

Django é um framework WEB estruturado na linguagem Python. Conforme a Django Software Foundation (entre 2005 e 2021), este framework possui funções construídas que otimizam o processo de desenvolvimento para aplicações e serviços WEB. Por disponibilizar diversas funcionalidades de segurança previamente construídas e apesar em serem desenvolvidas com uma abrangência generalizada, sua estrutura fora identificada como facilitadora no desenvolvimento da solução.

2.15 BOOTSTRAP

Este framework é composto por bibliotecas em JS, CSS e HTML. Conforme SOUZA (2019), sua principal utilização é focada para estilização e habilitação de recursos como responsividade em páginas web. Para que seus recursos tenham efeito, os elementos interagem-se através do elemento nomeado como “class” dentro da estrutura de cada elemento HTML. Existem duas formas para que a página web seja capaz em consumir os recursos deste framework, a primeira ocorre através de uma

CDN e a outra forma seria baixar os arquivos do sítio do fornecedor e incluir o caminho do diretório dentro da página web.

2.16 BEENIUS

Beenius é um fornecedor de uma solução que leva o nome da empresa no produto. Esta solução trata-se de um middleware que permite que um ISP possa gerenciar produtos e assinantes de IPTV. A Beenius é uma empresa de origem eslovena e possui parceria com a MultTV. O Beenius é um middleware baseado em Wildfly (antigo JBoss AS) Server, escrito em java e trabalha sobre Java Servlets, em que estes servlets permitem a comunicação entre dispositivos, servidores, recursos desse sistema e por fim, a SOAP API disponibilizada pelo sistema para fins de integração com ERPs e CRMs.

2.17 MIDDLEWARE

Segundo a equipe de redação da SoftwareONE (2021), um middleware é um sistema que opera entre uma ou várias origens e destinos, mediando a comunicação entre os diversos serviços existentes, sem que haja limitações por haver diferenças entre protocolos, infraestruturas, recursos e funcionalidades desses sistemas.

No núcleo de funcionamento de um middleware, considera-se que sua estrutura é baseada nas definições de uma API, pois este disponibiliza serviços e comunicação através de protocolos de chamadas remotas e requisições.

3 METODOLOGIA

O desenvolvimento do sistema proposto tomará como base a exposição identificada pela empresa MultTV, esta que identificou uma vulnerabilidade crítica no controle atual da API disponibilizada aos clientes. Realizou-se o levantamento de requisitos junto ao gestor da equipe técnica através do método de entrevista, identificando as tratativas necessárias para o início do projeto. O sistema auxiliará no processo de controle em quem acessa e consome as funcionalidades da API, registra todas as atividades ocorridas e impede o acesso indevido ao setor crítico desta API.

Fora necessário estudos mais aprofundados na linguagem de programação Python, protocolos de rede, sockets e para o MongoDB, que atuará como SGBD NoSQL para a base de dados e utilizados durante o desenvolvimento do middleware.

O sistema é composto como um serviço web, baseado no framework Django. Sua atuação trata as requisições recebidas para os endereços atribuídos ao serviço e garante a funcionalidade entre o recebimento e entrega das requisições, de maneira segura e contínua.

Após a análise de requisitos, fora elaborado o modelo lógico do funcionamento do middleware de segurança, disponibilizando um diagrama de caso de uso e um outro de sequência.

Para o desenvolvimento da interface do sistema onde foram alocadas as funcionalidades para concessão e bloqueio de acesso aos módulos disponíveis da API do Beenius, acesso aos registros históricos das requisições recebidas pelas operações autorizadas e outras funcionalidades deste middleware, utilizou-se as linguagens de marcação HTML5, CSS3 e Bootstrap 4.

Efetuou-se testes de verificação e análise para identificar a coerência do sistema e sua usabilidade. Observou-se que o middleware atende todas as necessidades para tratar a segurança da API e garantiu-se também sua integridade.

4 LEVANTAMENTO E ANÁLISE DE REQUISITOS DO SISTEMA

Apresenta-se abaixo os principais requisitos levantados, necessários para que o middleware trate e ofereça as funcionalidades desejadas. Entre os recursos disponíveis, o administrador do Authbee é capaz em gerenciar as autorizações dos ISPs, tratando quais módulos são autorizados para cada cliente, permite consultas aos logs de registro, remoção ou cadastro de uma nova operação no sistema.

4.1 DESCRIÇÃO DOS REQUISITOS

As funcionalidades do middleware de segurança baseiam-se nas premissas que as operações já integradas com a API do Beenius, não necessitem em modificar ou gerar uma nova integração, que não seja necessário a utilização de dados de autenticação do lado do cliente como palavras secretas, tokens ou outros meios de autenticação.

A segunda premissa é que nenhum cliente tenha acesso ao módulo Core da API, pois este disponibiliza serviços que são capazes em retornar dados de todas as operações cadastradas no Beenius e o mais crítico que é o serviço que permite excluir todas as operações com uma simples requisição, sem a necessidade de confirmação ou autorização prévia, portanto, o Authbee possui mecanismos para vetar qualquer requisição ocorrida a este serviço. Para aprimorar ainda mais a segurança, o arquivo XML que contém todos os serviços existentes na API do Beenius, recebeu uma modificação em que o serviço capaz em deletar todas as operações fora removido.

4.1.1 Cadastro de Operações

O Authbee gerencia as requisições baseadas no IP público de origem das requisições recebidas pelo middleware. Após receber uma requisição, este verificará se o IP público existe na base de dados como fator de autenticação e em seguida, consulta-se quais serviços este IP possui autorização. A API do Beenius possui cerca de oito módulos onde os serviços estão devidamente distribuídos e para atender a premissa que resolve a vulnerabilidade apontada pelo módulo Core da API, este é o módulo em que ao cadastrar uma nova operação no middleware, de forma alguma deve-se habilitar o acesso Core a essa nova operação. Para reforçar a gestão desse middleware, replicou-se essa estrutura que trata o módulo Core para os módulos restantes, a fim de concentrar as requisições somente aos módulos indispensáveis para a gestão da operação via integração pela parte do cliente.

4.1.2 Listar Operações Cadastradas

Para que um usuário possa gerenciar e verificar se uma operação já está liberada no Authbee, disponibilizou-se uma feature em que o usuário pode consultar quais operações estão cadastradas, quais módulos este possui acesso, quando esta operação foi cadastrada e por quem foi registrada.

4.1.3 Remoção de Operações Cadastradas

Para elevar o nível de integridade em relação as autorizações, nenhuma funcionalidade para modificar as operações existentes fora disponibilizado, se necessário alterar alguma autorização ou dados da operação, será necessário removê-la e cadastrá-la novamente. A função que permite a remoção de uma operação fora disponibilizada em um botão de ação que é apresentado ao lado de cada operação na feature que lista as operações cadastradas.

4.1.4 Rastreabilidade e Consulta a Requisições

Para suprir o déficit em registros históricos das requisições e ações ocorridas via API, estruturou-se uma base de dados que armazena tanto a requisição, quanto a resposta de cada requisição. Nestes registros constam em texto bruto o identificador da operação que interagiu com a API e o corpo da mensagem das interações com a API.

4.2 REQUISITOS FUNCIONAIS

4.2.1 Cadastrar Nova Operação

RF_CNO_01 – Obrigatório o preenchimento de um IP Público e habilitar os módulos da API ao qual possuirá permissão de acesso ou não.

4.2.2 Consultar Operações Cadastradas

RF_COC_01 – Permite listar todas as operações cadastradas, identificar suas autorizações e quem cadastrou a operação.

4.2.3 Remover Operação

RF_RM_01 - Permite ao usuário do middleware remover operações existentes na feature que lista todas as operações cadastradas.

4.2.4 Acessar Logs de Registro

RF_ALR_01 – Permite que o usuário consulte todos os registros históricos das requisições ocorridas.

4.3 REQUISITOS NÃO-FUNCIONAIS

RNF_01 – Um usuário existente no Authbee deve ser capaz em gerenciar as operações e consultar os registros históricos.

RNF_02 – O sistema deve disponibilizar uma interface web responsiva para o usuário administrar as operações e consultar os registros históricos.

RNF_03 – O acesso ao ambiente de gerenciamento do middleware deve ocorrer através de um e-mail e palavra secreta como meio de autenticação.

RNF_04 – Os dados das operações cadastradas devem ser armazenas em um bando de dados não relacional.

RNF_05 – O sistema deve permitir o consumo da API somente se a operação estiver cadastrada e possuir os níveis pretendidos de autorização.

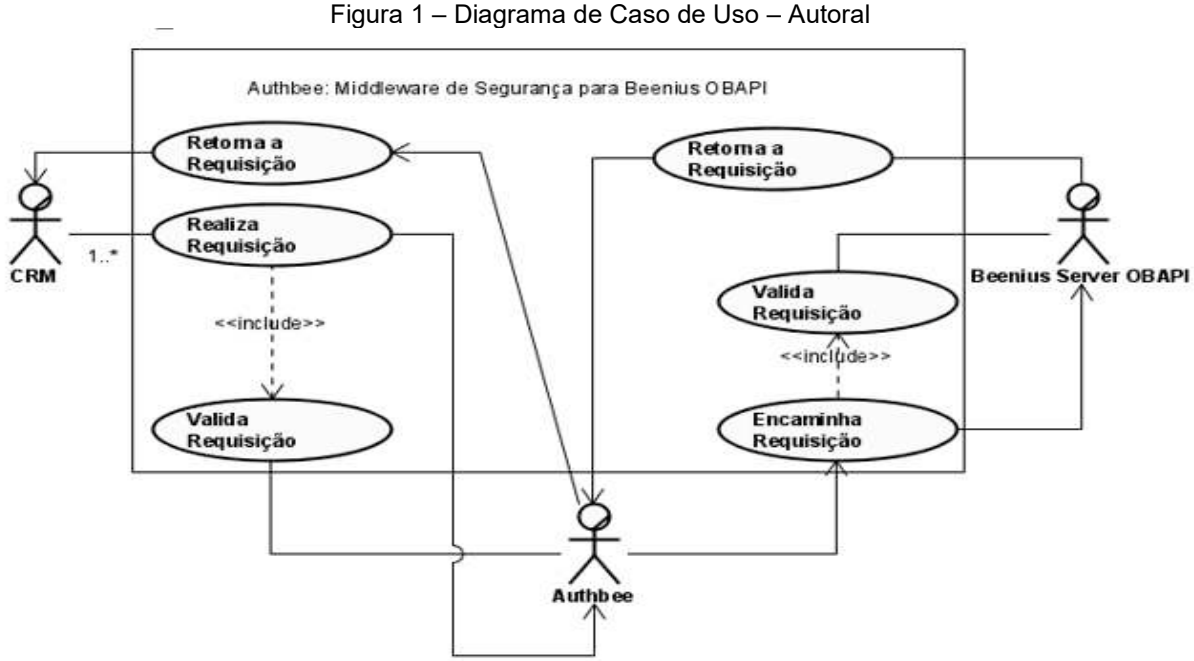
5 DESCRIÇÃO DO MIDDLEWARE DE SEGURANÇA

Descreve-se neste capítulo as funcionalidades disponibilizadas pelo Authbee e a infraestrutura do seu funcionamento para que este atenda de maneira devida, suas premissas e requisitos para garantir a integridade, autenticidade, rastreabilidade e segurança da API do Beenius.

5.1 CASO DE USO

Abaixo está a figura que representa o diagrama que esclarece a as funcionalidades de maneira mais abstrata entre a origem das requisições que partem dos clientes da MultTV, com o objetivo em consumir a API ofertada para a solução de IPTV.

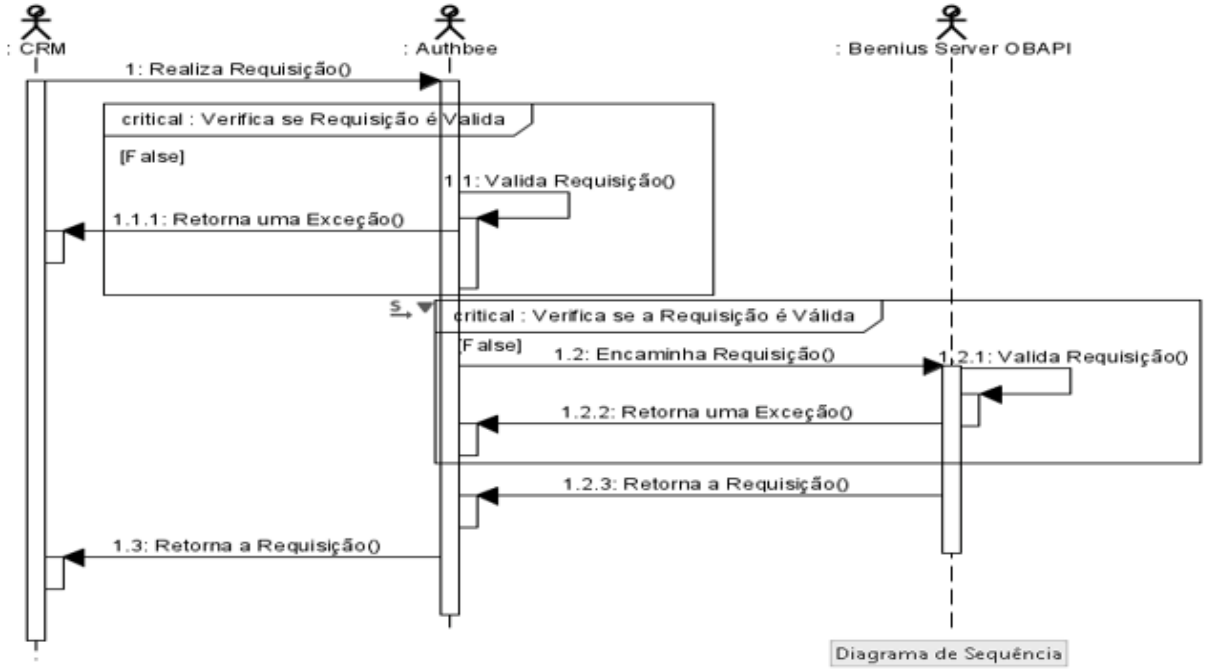
Figura 1 – Diagrama de Caso de Uso – Autoral



5.2 DIAGRAMA DE SEQUÊNCIA

A figura abaixo representa o fluxo lógico de comunicação entre a origem e o destino, onde toda comunicação está passiva de validação antes que seja capaz em atingir o servidor da API.

Figura 2 - Diagrama de Sequência – Autoral



5.3 ACESSO AO SISTEMA

O Authbee fora estruturado como um serviço web e este dispõe de uma interface gráfica para que um operador possa gerenciar os níveis de autorização de cada ISP cadastrado no middleware de segurança. Para acessar o sistema, existe um formulário que requer um e-mail e palavra-chave previamente cadastrado. Neste middleware existe somente um nível de acesso que é um acesso completo a todas as suas funcionalidades.

Outro detalhe é que a máquina virtual que aloca este serviço, está protegida atrás do firewall de produção na rede de gerência da empresa, portanto, para que se

tenha acesso a sua interface, o operador deve possuir acesso a VPN para que seja capaz em acessar o sistema.

5.4 MENU PRINCIPAL

O menu principal da solução, permite que o operador do sistema navegue entre os recursos que permitem cadastrar uma nova operação, listar as operações cadastradas para consultar seus dados e autorizações ou removê-las, acessar os registros de logs gerados pelo middleware e acessar a documentação do Beenius.

5.5 CADASTRAR OPERAÇÃO

O formulário que permite o registro de uma nova operação, possui parâmetros que exigem a quais módulos da API a operação terá autorização para consumir, o nome da operação, o identificador da operação existente no Beenius e o IP público do servidor alocado no ISP, que fará as requisições para a API do Beenius.

Este formulário verifica se o IP inserido é um IPV4 válido e se este já não está cadastrado na base de dados, caso haja algum conflito em relação as duas premissas anteriores, um novo registro não será permitido para essa operação.

Como o objetivo principal do Authbee é vetar o acesso ao módulo Core da API para qualquer ISP, quando é executado o cadastro de uma operação, deve-se deixar desmarcado o módulo que permite acesso ao CORE.

5.6 LISTAR OPERAÇÕES

Este recurso após invocado, realiza uma query na base de dados e retorna todas as operações cadastradas, contendo os seus dados de cadastro, autorizações, a data e hora do seu cadastro, o e-mail do operador que cadastrou a operação e o IP do operador no momento deste cadastro.

5.7 REMOVER OPERAÇÕES

Como uma premissa de segurança, adota-se o conceito que nenhum registro ou operação cadastrada possa ser alterada, portanto, caso uma operação esteja com parâmetros indevidos, é necessário remover esse registro existente e cadastrá-la novamente. Para realizar a remoção de uma operação, deve-se utilizar o botão de Remoção existente na linha de cada operação, na sessão em que lista as operações cadastradas.

5.8 LOGS DE REGISTRO

Para consulta dos registros de logs do middleware, considerou-se a divisão em módulos existentes na estrutura da API tratada, portanto, para acessar os logs deve-se optar entre as sessões do core, content, epg, live, purchase, ta, user e legacy.

6 RESULTADOS OBTIDOS

As premissas definidas pela MultTV, requerente pela solução, fora bem pontual na necessidade em resolver as limitações e vulnerabilidades existentes na API do Beenius, sem que esta resolução gerasse trabalhos extras a seus clientes, já que as integrações na maioria dos casos, fora executada por terceiros, algo que elevaria custos operacionais para adequar os ISPs a qualquer alteração que afetasse as integrações em produção.

O sistema possuirá utilização interna e administrativa e até o momento, somente o desenvolvedor deste sistema é quem gerencia as autorizações a API e o maior motivador deste foco é pela familiaridade a solução desenvolvida e por ser responsável por quem deve ter acesso aos recursos da API.

6.1 BASE DE DADOS

A figura abaixo representa a estrutura de um documento responsável por armazenar dados no MongoDB. Este documento contém os atributos pertencentes a um usuário cadastrado no Authbee, com autorizações de super usuário com acesso a todas as funcionalidades do sistema.

Figura 3 - Parâmetros de Um Administrador do Authbee – Autoral

A próxima figura exibe os atributos de um documento que contém as autorizações de uma Operação. Observa-se que o atributo coreService, é o responsável por impedir que um operador consuma os recursos existentes no módulo principal da API, onde fora identificado a vulnerabilidade principal tratada por este middleware.

Figura 4 - Parâmetros de Uma Operação no Authbee – Autoral

A figura abaixo representa um documento que armazena dados históricos de uma requisição recebida e tratada pelo Authbee, necessária para permitir que os recursos de rastreabilidade e geração de logs seja disponibilizada no middleware.

Figura 5 - Parâmetros de Um Log de Registro no Authbee

6.2 ALOCAÇÃO E ARRANQUE DO AUTHBEE

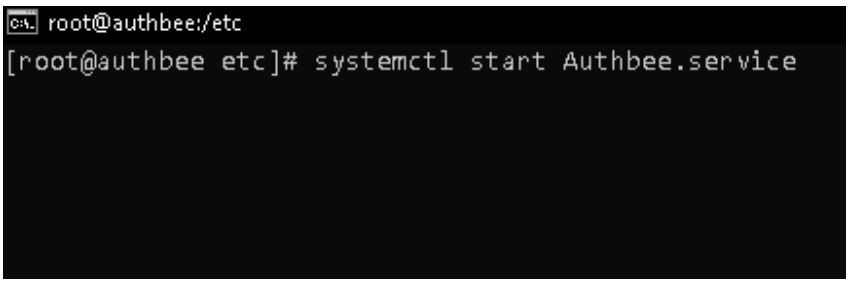
O middleware fora alocado em uma máquina virtual e a distribuição do sistema operacional utilizada fora o Fedora Server. Para facilitar o gerenciamento do middleware, este fora estruturado para que o gerenciador de serviços do sistema operacional pudesse gerenciá-lo.

Figura 6 – Arquivo do Serviço do Authbee - Autoral

```
[Unit]
Description= Authentication Middleware for Beenius API Server
Requires=network.target
After=network.target
[Service]
Type=simple
WorkingDirectory=/home/Authbee
User=root
Restart=always
RestartSec=10
#EnvironmentFile=/home/Authbee/manage.py
ExecStart=/usr/bin/python /home/Authbee/manage.py runserver 0.0.0.0:765
Restart=always
RestartSec=10
[Install]
WantedBy=multi-user.target
```

Abaixo a figura demonstra como o serviço do Authbee é invocado via prompt de comando, utilizando o gerenciador de serviços para gerenciá-lo.

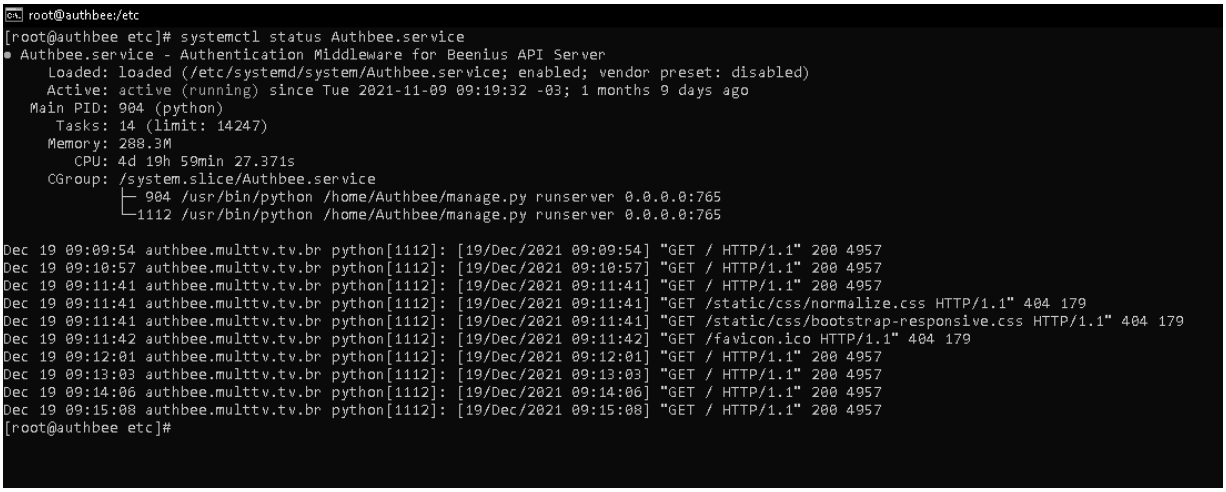
Figura 7 - Invocando o Serviço do Authbee – Autoral



6.3 CONFERINDO O STATUS DO AUTHBEE

Para verificar se o serviço está ativo e disponível, é utilizado recursos do systemctl que é o gerenciador de serviços dessa distribuição, para consultar o status do serviço através do terminal.

Figura 8 - Conferindo o Status do Authbee – Autoral



6.4 ACESSANDO E AUTENTICANDO-SE NO AUTHBEE

Para acessar o recurso web do middleware, utiliza-se o endereço https://authbee.multtv.tv.br:765. A página carregará um formulário de login, obriga-se o preenchimento dos campos com um e-mail e palavra-chave para o processo de autenticação.

Figura 9 - Formulário de Login do Authbee – Autoral

6.5 MENU PRINCIPAL DO AUTHBEE

Para acessar as features que disponibilizam as funcionalidades exigidas na solução, fora necessário a centralização em uma interface principal para facilitar a navegação e acesso aos recursos do usuário do sistema.

Figura 10 - Menu Principal do Authbee – Autoral

6.6 CADASTRANDO UMA OPERAÇÃO

Para acessar o formulário de cadastro, utiliza-se o caminho MENU PRINCIPAL > ADICIONAR OPERAÇÃO. Deve ser informado o IP Público da operação a ser cadastrada e as demais informações solicitadas no formulário.

Figura 11 - Formulário de Cadastro de Operação no Authbee – Autoral

6.7 LISTANDO OPERAÇÕES

Para acessar a relação das operações cadastradas, utiliza-se o caminho MENU PRINCIPAL > LISTAR OPERAÇÔES.

Figura 12 - Listagem e Remoção das Operações Cadastras no Authbee – Autoral

6.8 REMOVER OPERAÇÕES CADASTRADAS

Para acessar a funcionalidade de remoção de uma operação, utiliza-se o caminho MENU PRINCIPAL > LISTAR OPERAÇÕES, na linha de registro de cada operação, existe um botão para remover o registro da base de dados.

6.9 TRECHO DO CÓDIGO QUE TRATA AS REQUISIÇÕES AO CORE

A figura abaixo exibe o código gerado para tratar as requisições que o módulo principal da API recebe. A função verifica as autorizações para o IP público e se o acesso ao módulo requisitado está devidamente autorizado no cadastro da operação.

Para elevar o controle de todos os recursos da API, a estrutura dessa funcionalidade que trata o módulo Core fora replicada paras os diversos módulos da API, elevando a gestão da segurança da solução.

Caso a origem da requisição possua as devidas autorizações, ela será capaz em baixar o arquivo WSDL para iniciar a construção da sua requisição.

Figura 13 - Trecho Código da Função Core: Método GET – Autoral

```
if request.method=='GET':

    form = FormCoreService(request.GET)

    if getOperator != None:

        if getIp == getOperator['operatorIp'] and getOperator['coreService'] == True:

            http = Session()

            try:

                core_Form = open('middleware/templates/core.xml').read()
                return HttpResponse(core_Form, content_type='application/xml',)

            except Exception as Error:

                print(Error)

            else:

                pass

        else:

            return HttpResponseForbidden()

    else:

        return HttpResponseForbidden()
```

No segundo trecho da função que trata as requisições, recebe-se a requisição originada pelo cliente e reencaminhada para o servidor da API, caso a requisição seja válida. O sistema aguardar uma resposta do servidor da API e a retorna após recebê-la para a origem da requisição e para garantir o fluxo, um log de registro é gerado antes de disponibilizar a resposta para a origem.

Figura 14 - Trecho Código da Função Core: Método POST – Autoral

```
if request.method=='POST':

    form = FormCoreService(request.POST)

    request_Size = request.META['CONTENT_LENGTH']
    request_Size = int(request_Size)

    body_post = request.read(int(request_Size))
    body_post_init = time()
    url = 'http://beeapi.multtv.tv.br:765/OB/core/endpoint?wsdl'
    headers = {'Content-Type': 'text/xml; charset=utf-8',}
    response = requests.post(url, data=body_post, headers=headers)
    body_post_end = time()
    time_duration = body_post_end - body_post_init

    request_Log = {
    'bindingService': 'CoreService',
    'ip_Address': getIp,
    'operatorName': operatorName,
    'time': getTime,
    'date': getDate,
    'httpHost': httpHost,
    'method': 'POST',
    'post': body_post.decode('utf-8', ),
    'response': response.text,
    'request_duration': time_duration
    }

    collection_Logs.save(request_Log)

    return HttpResponse(response.text)
```

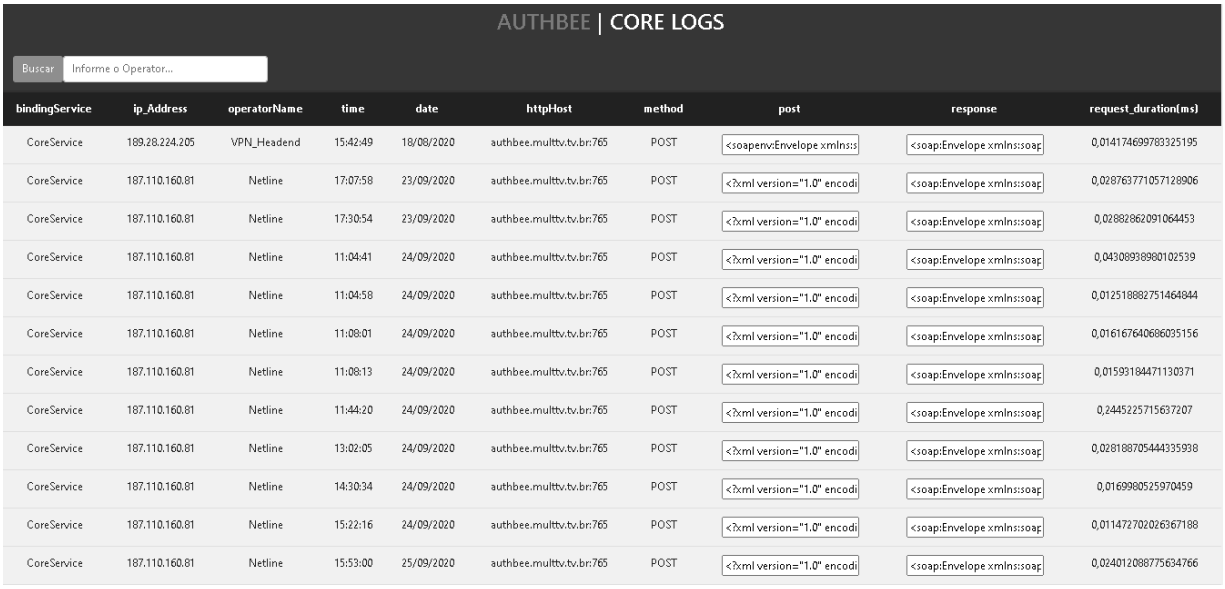
6.10 ACESSAR LOGS

Para acessar os logs disponíveis nos registros do sistema, utiliza-se o caminho MENU PRINCIPAL > ACESSAR LOGS > Optar pelo serviço pretendido.

Figura 15 - Acesso ao Menu de Logs do Authbee - Autoral

Abaixo a figura demonstra o retorno dos logs de registro após um usuário do Authbee acessar a feature de logs do sistema.

Figura 16 - Acesso aos Logs do Core do Authbee - Autoral



bindingService	ip_Address	operatorName	time	date	httpHost	method	post	response	request_duration(ms)
CoreService	189.28.224.205	VPN_Headend	15:42:49	18/08/2020	authbee.multtv.tv.br:765	POST	<soapenv:Envelope xmlns:s	<soap:Envelope xmlns:soap	0,014174699783325195
CoreService	187.110.160.81	Netline	17:07:58	23/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,028763771057128906
CoreService	187.110.160.81	Netline	17:30:54	23/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,02882862091064453
CoreService	187.110.160.81	Netline	11:04:41	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,04308938980102539
CoreService	187.110.160.81	Netline	11:04:58	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,012518882751464844
CoreService	187.110.160.81	Netline	11:08:01	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,016167640686035156
CoreService	187.110.160.81	Netline	11:08:13	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,01593184471130371
CoreService	187.110.160.81	Netline	11:44:20	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,2445225715637207
CoreService	187.110.160.81	Netline	13:02:05	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,028188705444335938
CoreService	187.110.160.81	Netline	14:30:34	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,0169980525970459
CoreService	187.110.160.81	Netline	15:22:16	24/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,011472702026367188
CoreService	187.110.160.81	Netline	15:53:00	25/09/2020	authbee.multtv.tv.br:765	POST	<?xml version="1.0" encodi	<soap:Envelope xmlns:soap	0,024012088775634766

A figura abaixo possui a estrutura de uma requisição realizada para o módulo Core, devidamente autorizado para uma operação. Estes dados são contidos no atributo “post” de um log de registro alocado na base de dados do middleware.

Figura 17 - Log gerado no sistema para o Serviço Core: Post – Autoral

```
<soapenv:Envelope xmlns:bees="http://www.beesmart.tv/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header/>
    <soapenv:Body>
        <bees:getFilteredRegionsV2>
            <regionFilterV2>
                <operatorUid>default</operatorUid>
            </regionFilterV2>
        </bees:getFilteredRegionsV2>
    </soapenv:Body>
</soapenv:Envelope>
```

Abaixo a figura demonstra a resposta obtida pelo Authbee após mediar a requisição recebida de uma operação devidamente autorizada. Estes dados são contidos no atributo “response” de um log de registro alocado na base de dados do middleware.

Figura 18 - Log gerado no sistema para o Serviço Core: Response – Autoral

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <bs:serverTime xmlns:bs="http://www.beenius.tv/OB">2020-08-18T18:42:49.428Z</bs:serverTime>
    </soap:Header>
    <soap:Body>
        <ns2:getFilteredRegionsV2Response xmlns:ns2="http://www.beesmart.tv/">
            <regionResponseList>
                <regionResponse>
                    <regionUid>NIPBR</regionUid>
                    <operatorUid>default</operatorUid>
                    <name>NIPBR</name>
                    <country>BR</country>
                    <defaultLanguage>pt</defaultLanguage>
                    <additionalLanguages>
                        <additionalLanguage>
                            <language>en</language>
                            <active>true</active>
                        </additionalLanguage>
                    </additionalLanguages>
                </regionResponse>
                <regionResponse>
                    <regionUid>Taubate</regionUid>
                    <operatorUid>default</operatorUid>
                    <name>Taubate</name>
                    <description>Taubaté</description>
                    <country>BR</country>
                    <defaultLanguage>pt</defaultLanguage>
                </regionResponse>
            </regionResponseList>
        </ns2:getFilteredRegionsV2Response>
    </soap:Body>
</soap:Envelope>
```

7 CONCLUSÕES

O projeto baseou-se em limitações e vulnerabilidades existentes em uma solução tecnológica em ambientes de produção empresarial no setor de telecomunicações. Essa exposição do serviço principal da API, caracterizou-se como vulnerabilidade pela utilização inadequada à sua estruturação original, pois a API fora planejada para utilização de modo interno pelo detentor do sistema e não de forma compartilhada com os clientes da MultTV.

Fez-se necessário receber treinamento técnico durante um período de um mês para aprender a gerenciar e compreender o funcionamento do Beenius e outros três meses de estudo da documentação da API para compreender e validar todas as tratativas necessárias para solucionar os pontos focais em questão.

Por se tratar de um trabalho decorrido em um sistema proprietário e por ocorrer na primeira experiência como desenvolvedor de sistemas, as formas iniciais adotadas para encontrar a solução ocorreram de modo ineficiente enquanto não houvesse amadurecimento para compreender os conceitos que circundam as tecnologias envolvidas no projeto. Não houve facilidades para realizar pesquisas em campo e encontrar materiais específicos que atendessem os requisitos do projeto, a forma encontrada para suprir os requisitos surgiram após compreender como ocorrem as requisições entre sistemas e serviços web, APIs, sockets, conceitos básicos de redes e por fim, como tratar dados e manipular requisições.

O projeto final demandou aproximados dez meses até a sua conclusão, validação e implementação em ambiente de produção, atendendo com segurança todas as necessidades que o projeto requer para as questões de integridade, autenticidade, auditoria, disponibilidade e privacidade do serviço como um todo.

Durante a execução desse projeto, permitiu-se o amadurecimento em conceitos de segurança cibernética, a aplicação de conceitos de engenharia de software para desenvolvimento de projetos, como ocorrem as comunicações entre sistemas, como sistemas distribuídos operam e são estruturados, formas distintas de automação, tratamento de dados, como trabalhar com diversos frameworks e linguagens distintas de maneira interdependente, caminhos mais precisos em como desenvolver pesquisas e encontrar formas de resolver desafios computacionais.

Desde a entrega e a operacionalização do Authbee da qual ocorrera em junho de 2020, este tornou-se o middleware entre os clientes da MultTV e o servidor da API do Beenius. As integrações realizadas com esta API antes da implementação dessa solução não necessitaram de qualquer alteração, exceto atualizar o apontamento para o novo endereço ao qual recebe as requisições da API, a partir da entrega do projeto.

Para concluir, compreende-se que entender a origem de um problema permite que a metade do desafio já esteja resolvido, basta compreender como e pelo que pesquisar para encontrar meios e definir a melhor solução a ser aplicada para cada processo.

7.1 TRABALHOS FUTUROS

Além do middleware principal da MultTV que é o Beenius, implementou-se um novo fornecedor para cobrir outras frentes do produto de IPTV, sistema este que também dispõe de uma API para realizar integrações, no entanto, para economizar força de trabalho dos clientes da empresa, um novo projeto fora aprovado que consiste em integrar o sistema antigo com o novo nos bastidores, reaproveitando as integrações já executadas pelos clientes. Para atender este propósito, faz-se necessário algumas adequações na estrutura das funções existentes no Authbee, ou seja, se na requisição recebida existir uma TAG pré-definida como gatilho, a função compreenderá que a requisição deverá ser desviada para um novo middleware e que este trate das ações esperadas pelo novo projeto.

O Authbee necessita de melhorias em seu layout, principalmente nos recursos pertinentes a responsividade da interface web disponibilizada, esta implementação ainda não fora iniciada ou planejada por não estar no topo das prioridades da equipe de engenharia neste momento.

Através dessas implementações, a qualidade e o alcance do serviço serão aprimorados e os fluxos de dados e requisições mantêm-se centralizados, como se ao final, existisse somente um único middleware de gestão operacional para os clientes da MultTV, independentemente da quantidade de middlewares distintos em produção.

REFERÊNCIAS

DSF. **Meet Django. Django Software Foundation, entre 2005 e 2021.**
Disponível em: https://www.djangoproject.com/
Acesso em: 1 de nov. de 2019

Equipe de Redação. **Afinal, o que é Middleware? SoftwareONE, 28 de jul. de 2021.**
Disponível em: https://www.softwareone.com/pt-br/blog/artigos/2020/03/02/middleware-o-que-e-e-quais-sao-as-vantagens-de-usar
Acesso em: 21 de ago. de 2021

Fedora Project. **Fedora Server Documentation. Fedora Project, entre 2003 e 2021.**
Disponível em: https://docs.fedoraproject.org/en-US/fedora-server/
Acesso em: 5 de nov. de 2019

Liam. **Extensible Markup Language (XML). W3C, 11 de out. de 2016.**
Disponível em: https://www.w3.org/XML/
Acesso em: 1 de fev. de 2020

SOUZA, Ivan. **Bootstrap: saiba neste guia para iniciantes o que é, por que e como usá-lo. Rockcontent, 12 de dez. 2019.**
Disponível em: https://rockcontent.com/br/blog/bootstrap/
Acesso em: 20 de dez. de 2019

MDN Web Docs. **Uma Visão Geral do HTTP. Mozilla, entre 2005 e 2021.**
Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview
Acesso em: 3 de fev. de 2020

MongoDB Inc. **Getting Started. MongoDB, entre 2007 e 2021.**
Disponível em: https://docs.mongodb.com/manual/tutorial/getting-started/
Acesso em: 3 de fev. de 2020

NEGRESIOLO, Leticia. **O admirável mundo de possibilidades das APIs para desenvolvimento web. Gizmodo Brasil, São Paulo, 18 de jan. de 2019.**
Disponível em: https://gizmodo.uol.com.br/apis-para-desenvolvimento-web
Acesso em: 21 set. de 2020

NOLETO, Cairo. **Framework: o que é, como ele funciona e para que serve? Trybe, 13 de fev. de 2020.**
Disponível em: https://blog.betrybe.com/framework-de-programacao/o-que-e-framework/
Acesso em: 15 fev. de 2020

PSF. **General Python FAQ. Python Software Foundation, entre 2001 e 2021.**
Disponível em: https://docs.python.org/3/faq/general.html#general-information
Acesso em: 10 de dez. de 2019

ROZLOG, Mike. **REST e SOAP: Usar um dos dois ou ambos? Infoq, 2 de out. de 2013.**
Disponível em: https://www.infoq.com/br/articles/rest-soap-when-to-use-each/
Acesso em: 10 de dez. de 2021

TELLINGEN, Michael. **Zeep: Python SOAP Client. PYTHON-ZEEP, 15 de jun. de 2016.**
Disponível em: https://docs.python-zeep.org/en/master/
Acesso em: 10 de dez. de 2019

VMWare. **Quem somos. VMWare, entre 1998 e 2021.**
Disponível em: https://www.vmware.com/br/company.html
Acesso em: 29 de set. de 2019

W3Schools. **XML Soap. W3Schools, entre 1999 e 2020.**
Disponível em: https://www.w3schools.com/xml/xml_soap.asp
Acesso em: 3 de fev. de 2020

W3Schools. **XML WSDL. W3Schools, entre 1999 e 2020.**
Disponível em: https://www.w3schools.com/xml/xml_wsdl.asp
Acesso em: 3 de fev. de 2020

WebConcepts. **REST API Concepts and Examples. Youtube, 14 de jul. de 2014.**
Disponível em: https://www.youtube.com/watch?v=7YcW25PHnAA
Acesso em: 21 de set. de 2020

WINER, Dave. **XML-RPC Specification. XML-RPC, 15 de jun. De 1999.**
Disponível em: http://xmlrpc.com/spec.md#update3
Acesso em: 10 de fev. de 2020