UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Pedro Henrique Bufulin de Almeida

Sistema de vigilância *open source* e customizável

Uberlândia, Brasil

2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Pedro Henrique Bufulin de Almeida

Sistema de vigilância *open source* e customizável

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Pedro Frosi Rosa

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2021

Pedro Henrique Bufulin de Almeida

Sistema de vigilância *open source* e customizável

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 01 de novembro de 2016:

Pedro Frosi Rosa
Orientador

Professor

Professor

Uberlândia, Brasil
2021

Resumo

Os sistemas de vigilância carecem de soluções que sejam customizáveis. Apesar de existirem no mercado câmeras que fazem a gravação e o armazenamento de imagem, elas o fazem com com baixa resolução e limitações de espaço de memória severas. Mesmo os equipamentos de monitoramento mais sofisticados, como os que possuem reconhecimento de imagem, limitam o usuário de aprimorar essas funções por ter o *software* como *closed source*. Além disso, as soluções providas pela segurança pública ou privada, quando existem, podem usar as imagens coletadas para seus próprios benefícios, o que afeta a privacidade do indivíduo. Este trabalho propõe uma solução que seja fácil de implementar, com ferramentas prontas para uso e completamente customizável por ser *open source*. Dadas essas características, surge o chamado *Self-Sovereign Camera System* (SSCS)

Palavras-chave: código aberto, visão computacional, IP câmera, streaming multimídia.

Abstract

Surveillance systems lack customizable solutions. Despite the existence of cameras on the market that handle both recording and storage of images, they do so at low resolution and with severe memory space limitations. Even the most sophisticated monitoring equipment, such as those with image recognition capabilities, restrict the user from enhancing these features due to the software being closed source. Furthermore, solutions provided by public or private security, when available, may use the collected images for their own benefit, affecting individual privacy. This work proposes a solution that is easy to implement, equipped with ready-to-use tools, and fully customizable as it is open source. Given these characteristics, the so-called Self-Sovereign Camera System (SSCS) emerges.

Keywords: open source, computer vision, IP camera, multimedia streaming.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SSCS	Self-Sovereign Camera System
CAGR	Compound annual growth rate, ou taxa de crescimento anual.
API	Application Programming Interface
REST	Representational State Transfer
HSL	HTTP Live Streaming
RTMP	Real Time Messaging Protocol
LBPH	Local Binary Patterns Histogram

Sumário

1 Introdução

Em 2019 foi estimado que existem 200 milhões de câmeras de vigilância na China e na última década, avanços tecnológicos tornaram essas câmeras ainda mais eficientes em monitorar 1.4 bilhões de chineses. Ainda segundo o autor, o reconhecimento de rostos por câmeras começou a ser uma realidade em 2010 quando pesquisadores descobriram algoritmos de deep learning usados para reconhecer imagens e voz. Esses algoritmos podem também inferir em tempo real a quantidade e a densidade de pessoas numa dada imagem (QIANG, 2019).

Esta intensificação do uso da tecnologia de vigilância não se limita à China; ou está se tornando ou já é a realidade em muitos outros países. De acordo com um relatório da Market Research Future (GUPTA, 2018), a América do Norte é atualmente líder em termos de participação de mercado de câmeras de CCTV, detendo 28,5% do mercado global, seguida pela Europa com 18,3%. Este domínio é reflexo de um mercado de vigilância por vídeo que tem mostrado crescimento substancial: avaliado em 45,9 bilhões de dólares em 2021, este mercado tem uma projeção de crescimento para 110,2 bilhões de dólares até 2030, com uma taxa composta anual de crescimento (CAGR) de 11,6% para o período de 2022 a 2030. Esses dados evidenciam não apenas a escala atual da indústria de vigilância, mas também sua trajetória ascendente, realçando a importância de abordar questões de privacidade e segurança em um contexto tão amplo e em rápido crescimento.

À medida que o mercado de vigilância por vídeo se expande, os sistemas associados também se tornam progressivamente mais complexos e de maior alcance. Isso demonstra a necessidade de infraestruturas mais robustas e eficientes. Nesse contexto, é intrigante observar que as complicações técnicas enfrentadas pelos sistemas de vigilância têm muitas semelhanças com aquelas encontradas em plataformas de streaming de vídeo. Ambos os tipos de sistemas têm uma dependência substancial de infraestruturas distribuídas para assegurar a prestação de serviços em tempo real, de maneira eficaz e confiável.

Entretanto, a privacidade dos indivíduos não é uma preocupação primária dos principais provedores desse tipo de serviço. Nesse contexto, tecnologias de código aberto podem oferecer uma solução, pois permitem que qualquer pessoa verifique se o software tratando a segurança e a privacidade de maneira adequada (MARDJAN; JAHAN, 2016).

Considerando-se a necessidade de criar uma solução que traga os benefícios da vigilância e mantendo a privacidade individual, surge neste trabalho a proposta da construção de um software de vigilância.

1.1 Objetivos

O propósito deste trabalho é projetar um sistema de monitoramento integralmente sustentado por software livre, com capacidade de integração com uma ampla variedade de dispositivos de hardware. Este sistema oferecerá visualização em tempo real de imagens de câmeras, incorporando algoritmos de visão computacional para análise destas. Será possível armazenar e buscar vídeos, utilizando critérios identificados pelos algoritmos. Uma das grandes virtudes do projeto é sua adaptabilidade: ele é projetado para permitir a inclusão de novas funcionalidades, como notificações de eventos, e a modificação de características existentes. Por exemplo, o armazenamento pode ser configurado para ser local ou em nuvem, e o processamento para reconhecimento de imagens pode ocorrer tanto em transmissões ao vivo quanto em vídeos arquivados. Para facilitar a implementação e personalização do sistema, será disponibilizada uma documentação abrangente, delineando várias opções de uso para satisfazer as distintas necessidades dos usuários.

1.1.1 Objetivos específicos

Os objetivos específicos envolvem:

- Desenvolver todo o software com código aberto e mantê-lo publicamente disponível no Github.
- Usar a conteinerização como estratégia para conseguir portabilidade entre vários hardwares diferentes.
- Implementar uma funcionalidade de visualização em tempo real de imagens provenientes de câmeras.
- Incorporar algoritmos de visão computacional robustos para análise e processamento das imagens em tempo real.
- Projetar e desenvolver um módulo de armazenamento flexível, permitindo opções tanto locais quanto em nuvem.
- Habilitar a busca de vídeos com critérios baseados nas identificações realizadas pelos algoritmos de visão.
- Possibilitar a customização do sistema, incluindo a adição de novas funcionalidades como notificações de eventos.
- Adaptar o sistema para permitir variações no processamento de reconhecimento de imagens, seja em transmissões ao vivo ou em vídeos arquivados.

- Escrever testes automatizados e *benchmarks* para manter a qualidade do software e das soluções em um nível aceitável.
- Produzir uma documentação detalhada e abrangente, facilitando a implementação, personalização e manutenção do sistema por parte dos usuários.

2 Método de Desenvolvimento

A metodologia selecionada para o avanço deste projeto se baseia, inicialmente, em um exame das soluções de software atualmente presentes no mercado. O objetivo é compreender suas características, funcionalidades e possíveis vulnerabilidades de segurança. Esse estudo, de caráter essencialmente qualitativo, procura identificar tendências, padrões e lacunas nas soluções existentes.

Em Seguida, serão exploradas as tecnologias e técnicas comumente adotadas na indústria. Essa investigação abrange desde protocolos de transporte e arquiteturas predominantes de hardware e software até algoritmos sofisticados de compressão e reconhecimento. O intuito é discernir os padrões de uso e as preferências do setor.

Dentro desse contexto, uma análise focada no SSCS será realizada. Investigaremos se ele propõe uma reinvenção dessas soluções padrão ou se opta por uma integração harmônica com elas. Esta etapa é crucial para entender a proposta de valor e o diferencial que o sistema proposto oferece em relação às alternativas tradicionais.

Posteriormente, adentraremos na esfera econômica, conduzindo uma pesquisa sobre os custos envolvidos. Essa avaliação não se restringirá apenas aos valores de aquisição de software ou hardware, mas também contemplará os gastos operacionais, incluindo a execução de softwares e os custos de aquisição, manutenção e atualização de hardwares.

2.1 Estado da Arte de Câmeras de Vigilância

Diversas soluções de vigilância estão atualmente disponíveis no mercado, cada uma adotando um conjunto único de padrões e tecnologias. Entre as várias opções, as câmeras CCTV (*Closed-circuit television*) emergem como um sistema tradicional e amplamente empregado. Curiosamente, o conceito de CCTV remonta a 1927, quando o físico russo Léon Theremin desenvolveu o primeiro dispositivo desse tipo (GLINSKY, 2000). Essas câmeras funcionam em um sistema que transmite os sinais de vídeo para um conjunto específico e privado de monitores. Ao contrário de soluções mais modernas que utilizam transmissão via IP, as câmeras CCTV são frequentemente associadas a sistemas de transmissão analógica, embora modelos mais recentes possam incorporar capacidades digitais e transmissão pela web. Atualmente, os modelos mais comuns de câmeras de vigilância já possuem integração direta à internet usando o protocolo IP.

No quesito dos métodos de transmissão multimídia por essas câmeras, temos três métodos principais: *streaming* tradicional, *download* progressivo, e *streaming* adaptativo. Cada um deles tem suas vantagens e desvantagens quanto aos critérios de latência, qua-

lidade, requerimentos de processamento e compatibilidade com outros tipos de software. O *Streaming* tradicional requer um protocolo *stateful* que estabelece uma sessão entre o servidor e o cliente. Nesse último método, a mídia é transmitida como uma corrente constante de pacotes por UDP ou TCP. Já no método do streaming por *download* progressivo, usa-se um servidor HTTP para transferir o vídeo entre o cliente e o servidor, de maneira *stateless*. Os clientes fazem requisições de multimídia, que é enviado progressivamente para um buffer local. Assim que ele tem informação o suficiente, o vídeo começa a tocar. Se a taxa de *playback* excede a taxa de transmissão de dados, então o vídeo é interrompido até que o buffer seja preenchido adequadamente. O *streaming* adaptativo por sua vez, consiste em detectar a largura de banda de rede e capacidade de CPU do cliente para ajustar a qualidade do vídeo transmitido buscando a melhor opção para as condições dadas. Isso requer um *encoder* para prover vídeo em múltiplos *bit rates* diferentes, ou múltiplos *encoders* e pode ser usada uma Content Delivery Network (CDN) para aumentar a escalabilidade. É comum em streaming adaptativo utilizar a técnica de *stream switching*. Esse é um método híbrido que usa HTTP como o protocolo de entrega no lugar de definir um novo protocolo. Os dados de multimídia são segmentados em pequenas partes de mesmo tamanho, codificados no formato desejado e armazenadas em um servidor. Os clientes então requisitam os segmentos sequencialmente por download progressivo e eles são tocados em ordem já que são contíguos. O resultado é uma experiência de usuário praticamente livre de gargalos de buffering em condições normais de rede e CPU.

No quesito dos riscos de segurança, as câmeras IP apresentam várias vulnerabilidades que podem expô-las a ameaças. Uma das falhas mais comuns é a utilização de credenciais de login padrão, que são facilmente acessíveis para qualquer pessoa com conhecimento básico em segurança de rede. Além disso, políticas de autenticação fracas frequentemente permitem a seleção de senhas simples e fáceis de adivinhar. A criptografia inadequada de informações confidenciais é outro ponto de preocupação, pois muitas vezes os dados são transmitidos sem qualquer processo de criptografia, tornando-os suscetíveis a interceptações e ataques. Como exemplo dessas falhas, em (ABDALLA; VAROL, 2020) uma câmera smart identifica como Onvif YY HD passou por um teste de penetração. Nesse teste, um ataque do tipo man in the middle (MITM) foi capaz de visualizar em texto plano todos os dados transmitidos pela câmera, inclusive credenciais.

3 Fundamentação Teórica

Neste capítulo estão as tecnologias e os conceitos que serão utilizados no decorrer da construção deste projeto. São conceitos relacionados à arquitetura do sistema que será construído, as abordagens que já existem, *hardware*, linguagens e *frameworks* que serão utilizados.

3.1 Arquitetura de transmissão em tempo real

A figura abaixo mostra um sistema de cliente-servidor para trocar de dados de multimídia. Na origem, tem-se a gravação comprimida, codificada e armazenada no dispositivo de armazenamento, como um disco rígido, por exemplo. Em seguida, por meio de algum software de tratamento de mídia que será produzido, esses arquivos são requisitados por usuários e entregues de acordo. Um protocolo de transferência é utilizado para entregar os dados de multimídia para o cliente (tais como RTMP ou HLS), onde eles são primeiramente armazenados na memória principal e eventualmente decodificados e apresentados ao usuário. (LEE, 2005)

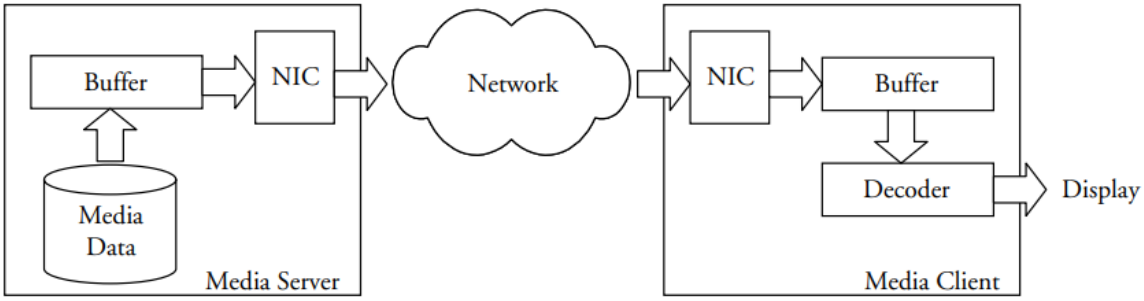

Figura 1 – Arquitetura genérica de transmissão em tempo real

3.2 Sistema distribuído aberto

Esta arquitetura é a que levarei em consideração na implementação do projeto. O servidor que será implementado utilizando um microcomputador contem a câmera e sistema operacional para realizar o processamento e transmissão dos dados de multimídia. Outra alternativa seria o microcomputador apenas enviar os fluxo de vídeo para um terceiro servidor que faria o processamento da imagem caso a capacidade daquele não

seja suficiente, resultando em um sistema distribuído. Essa outra abordagem também respeita a arquitetura citada acima.

3.3 Framework de desenvolvimento front end

Como é de se esperar, um sistema que permite a visualização em tempo real necessitará de um meio para apresentar ao usuário as saídas e o resultado do processamento delas. Como se trata de um sistema de segurança, o mais viavel é que o usuário sempre tenha em mãos o dispositivo que fornece a imagem, ou seja, o *smartphone*. Portanto, faz sentido utilizar um *framework* que permita a criação de um front end mobile. Além disso, a aplicação resultante deverá ser replicável em múltiplos dispositivos, para que o usuário não seja tolhido por não ter o meio adequado. Dadas essas condições, o *framework* que satisfaz elas é o *React Native*

O React Native é um framework de desenvolvimento mobile híbrido (iOS e Android simultâneamente) voltado para a construção de interfaces de usuário criado pelo facebook, utilizando *javascript* com a linguagem de programação e compilando-o de forma que seja executável pela plataforma nativa. Vale lembrar que o React Native não é uma solução *full-stack* que vai lidar com tudo desde o banco de dados à atualizações em tempo real de *web sockets*. Ele é simplesmente a *View layer* da aplicação. (BODUCH, 2017)

3.4 Tecnologias de desenvolvimento back end

O back end é a parte do sistema que realizará o processamento da imagem que consiste de um servidor contendo o banco de dados e uma aplicação que fornece os *endpoints* responsáveis pela comunicação com o front end. Nesses *endpoints*, o usuário consegue fazer o envio, requisição e alteração de dados por meio da abstração fornecida pelo front end.

Neste trabalho, a aplicação do *back end* será feita utilizando o *NodeJS*, que é um ambiente de execução de javascript no servidor baseado em *event-driven architecture* capaz de entrada e saída assíncrona, muito utilizado em aplicações de comunicação em tempo real. (ORSIN, 2013) Além disso, Em Node.js, HTTP é um cidadão de primeira classe, projetado para que tenha um alta taxa de fluxo e baixa latência. Isso torna o Node.js uma ótima escolha para servir como base para uma biblioteca web ou para um framework. (DAHL, 2009). Mais um motivo para a utilização de Node.js, é a grande quantidade de módulos disponíveis no npm. Este projeto particularmente utilizará vários deles, sendo os principais o Express e o node-media-server. O primeiro facilita a criação da aplicação web no servidor, e o segundo servirá para utilizar o protocolo RTMP, que é o ideal para transmissão de vídeo e áudio em alta performance.

Também é necessária a implementação de um banco de dados que se comunicará

com a aplicação do servidor. Para este projeto, foi escolhido o PostgreSQL, sendo ele um banco de dados relacional de código aberto. Este sistema de banco de dados foi escolhido por possuir *Write-ahead Logging*, *point-in-time-recovery*. Essas duas características são importantes para garantir a confiabilidade e a possível recuperação dos dados em caso de perda. Além disso, a comunidade ativa do PostgreSQL é responsável por criar diversas extensões e alguma delas podem ajudar no desenvolvimento do projeto. O último aspecto importante dessa escolha, é que esse sistema de banco de dados também permite *multi-factor authentication*. (GROUP, 1996)

Uma última camada de segurança da aplicação é necessária, de forma que tanto as informações do usuário quanto o servidor em si estejam protegidos de possíveis ataques. Para proteger o servidor de acesso indevido aos *endpoints* vou utilizar o JSON Web Token (JWT). O JSON Web Token é um meio de representar reivindicações à serem transferidas entre duas partes. Os dados são codificados com um JSON que é usado como o *payload* de uma estrutura JSON Web Encryption (JWE), permitindo que essas reivindicações sejam assinadas digitalmente ou a integridade protegida com *Message Authentication Code* ou criptografada. (BRADLEY, 2015)

3.5 Algoritmo de reconhecimento facial

(Separe isso em openCV e Haar Cascade.)

Para a funcionalidade de reconhecimento facial da câmera, usarei uma biblioteca de código aberto de *computer vision* chamada *OpenCV*. Essa biblioteca possui mais de 2500 algoritmos otimizados, incluindo os clássicos e os mais recentes do estado da arte. Eles podem ser utilizados para detectar e reconhecer faces, identificar objetos classificar ações humanas em vídeo e mais. (TEAM,). Ela suporta múltiplas linguagens incluindo Java, C++ e Python. para este trabalho, será usada a biblioteca em Python.

Para realizar o reconhecimento facial, é preciso conseguir antes reconhecer que há um rosto em vídeo. O método que utilizarei, facilitado pelo *OpenCV*, é o chamado *Haar Cascades* (VIOLA; JONES, 2001). Este algoritmo é baseado em *machine learning* onde a função de cascata é treinada por meio de muitas imagens positivas e negativas. No caso, as imagens positivas são várias fotos com rostos e as negativas são fotos sem rosto. O conceito de classificadores em cascata desse algoritmo por sua vez, vem do fato de que no lugar de aplicar as características encontradas numa única etapa, elas são agrupadas em diferentes estágios de classificação uma a uma. Se um desses estágios falhar, então a imagem é descartada, desconsiderando as características remanescentes nela. Assim sendo, a foto que passar por todos os estágios é uma região de um rosto.

A próxima parte do processo envolve usar o algoritmo que reconhece que existe um rosto no vídeo junto com um outro algoritmo que reconhecerá à quem pertence este rosto.

Para o algoritmo classificador reconhecedor que será usado agora, são necessárias varias imagens do rosto a ser reconhecido, que serão convertidas para *grayscale*, dessa forma mantendo o plano de luminância e separando o de crominância de cada imagem o que por sua vez reduz a quantidade de informação desnecessária presente e, por isso, facilita a identificação das características importantes para classificação. Em seguida, será usado o algoritmo Local Binary Patterns Histogram (LBPH) para realizar o treinamento do modelo que reconhecerá os rostos usando as imagens preparadas anteriormente. O LBPH foi escolhido por funcionar bem com as imagens em *grayscale* e ser capaz de desconsiderar a luminosidade como característica importante, como mostra a figura abaixo:

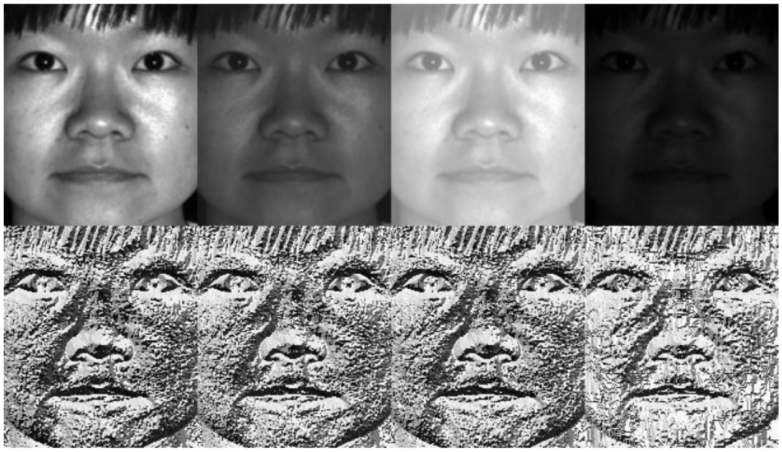

Figura 2 – Imagens resultantes do algoritmo LBP abaixo, geradas à partir da imagem em *grayscale* acima.

3.6 Arquitetura de microsserviços

Explicar aqui o que são microsseviços e como eles funcionam. Fazer talvez uma comparação em relação à uma monolito

3.7 Arquitetura Monolítica

Explicar aqui como funciona uma arquitetura monolítica e quais as desvantagens e vantagens em relação ao microsserviço.

3.8 Protoclo RTSP

Explciar como o protocolo RTSP é usado para pegar o feed das imagens. Existe o RFC to protocolo RTSP aqui:

3.9 Conteinerização

Explicar aqui o porquê de servir o software no docker

3.10 Daemon

Explicar o conceito de Daemon e o porquê de ser últil manter o software rodando como um processo segundo plano

3.11 Proxy

Explicar o porquê de às vezes usar um proxy. Às vezes queremos dar um feed de imagem, mas não tem como fazer uma conexão direta ao sistema que está provendo diretamente a imagem com RTSP. Tanto porque o sistema não consegue lidar com muitas requisições ao mesmo tempo, ou porque ele está numa intranet que precisa de um acesso seguro para prover ao resto da rede.

3.12 Licenças de Software

Explciar o porquê de usar uma licença de software adequada. E como diferentes Licenças de Software podem afetar como esse sistema é construído e mantido. Citar aquela licença (Acho que é a BPD) que pode absolutamente obliterar todo mundo que já estava usando um software específico.

3.13 Encodings de vídeo e som

Tenho que citar alguns encodings usados.

3.13.1 H.264

O que acontece nesse tipo de encoding? Em quais cenários ele é usado? Quais os problemas dele?

Essa informação aqui é importante para o streaming com WebRTC

WebRTC doesn't support H264 streams with B-frames: H264 is a video compression codec. In H264 encoding, three types of frames are used: I-frames (Intra frames), P-frames (Predictive frames), and B-frames (Bidirectional predictive frames). The error message is stating that WebRTC does not support H264 streams that include B-frames.

B-frames depend on both previous and future frames to derive their data, and this can cause issues in real-time streaming scenarios like those handled by WebRTC, where

latency and real-time performance are crucial. Therefore, WebRTC doesn't support H264 streams with B-frames.

3.14 Tecnologia de Hardware

Todos os algoritmos que irão processar a imagem da câmera e aplicação do back end precisam ficar alocados num servidor. Assim sendo, foi escolhido um raspberry pi zero W 1.1 para ser usado como o dispositivo que funcionará como servidor. Além disso, o Camera Module V2 será responsável por captar a imagem. Se fizermos uma comparação com os dispositivos de monitoramento presentes no mercado, esses dispositivos não são muito potentes. A empresa força tarefa de Uberlândia que fornece serviços de segurança e portaria remota, por exemplo, utiliza câmeras IP da Intelbras, muitas delas com sensores de visibilidade noturna, como o modelo VIP 3230 B SL. Por meio de uma conexão de rede, a informação de vídeo dessas câmeras vai para os servidores da empresa que possuem alta capacidade de processamento. Entretanto, mesmo com a capacidade reduzida dos dispositivos que serão usados neste trabalho, os algoritmos de reconhecimento facial, a aplicação do servidor e o front end poderão ser transferidos para outros melhores.

4 Revisão Bibliográfica

Um ou mais capítulos (por exemplo, se há duas linhas de trabalhos relacionados).

5 Desenvolvimento

Inicialmente, é vantajoso estabelecer um meio de simular um fluxo de vídeo para o SCSS processar. Esta abordagem facilita os testes e proporciona um ambiente de desenvolvimento que replica com fidelidade as condições sob as quais o sistema operará quando implementado em um ambiente de produção. É importante esclarecer que, ao referir-se a "sistema em produção", estamos discutindo a aplicação prática do software em ambientes comerciais, residenciais, ou outros contextos reais onde a vigilância é empregada para segurança e monitoramento.

No desenvolvimento do sistema, foi empregado o MediaMTX (ROS, 2023), um servidor de mídia em tempo real de código aberto que facilita a publicação, leitura, gravação e roteamento de streams de vídeo. Este software foi originalmente concebido para funcionar como um roteador de mídia de ponta a ponta, proporcionando uma plataforma robusta a partir da qual o SSCS pode acessar e consumir streams de vídeo. Entretanto, para que o servidor seja operacional, é necessário que a mídia seja publicada nele. Conforme indicado na documentação do MediaMTX, uma das abordagens recomendadas para isso é a utilização do ffmpeg, um framework de multimídia que permite a codificação, decodificação, transcodificação, muxing, demuxing, streaming e reprodução de dados de mídia. Através do ffmpeg (BELLARD, 2023), é possível direcionar streams de vídeo para o servidor MediaMTX, criando assim um ambiente propício para a integração e operação eficiente do SSCS.

Como é muito comum que sistemas de vigilância possuam várias câmeras ao mesmo tempo, fazem sentido as duas arquiteturas nas figuras abaixo. Ambas as arquiteturas lidam exclusivamente com a parte da transmissão para o servidor de mídia SSCS.

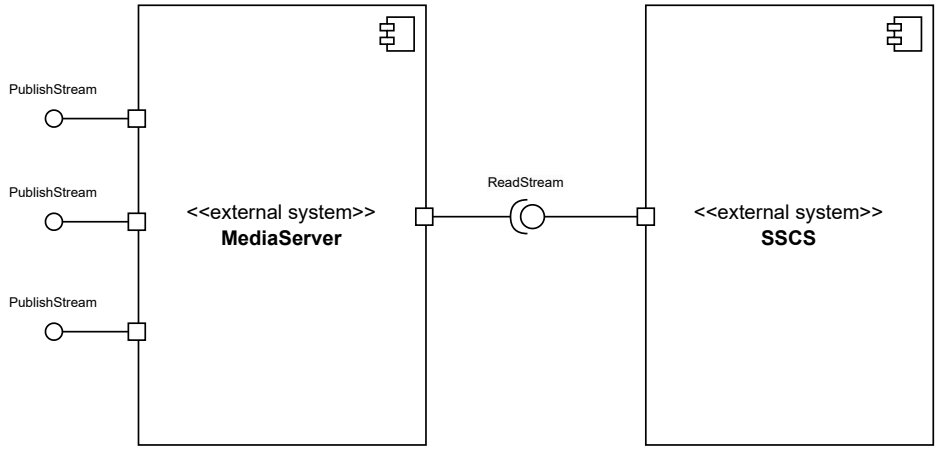

Figura 3 – Arquitetura de transmissão, múltiplas fontes para um servidor de mídia

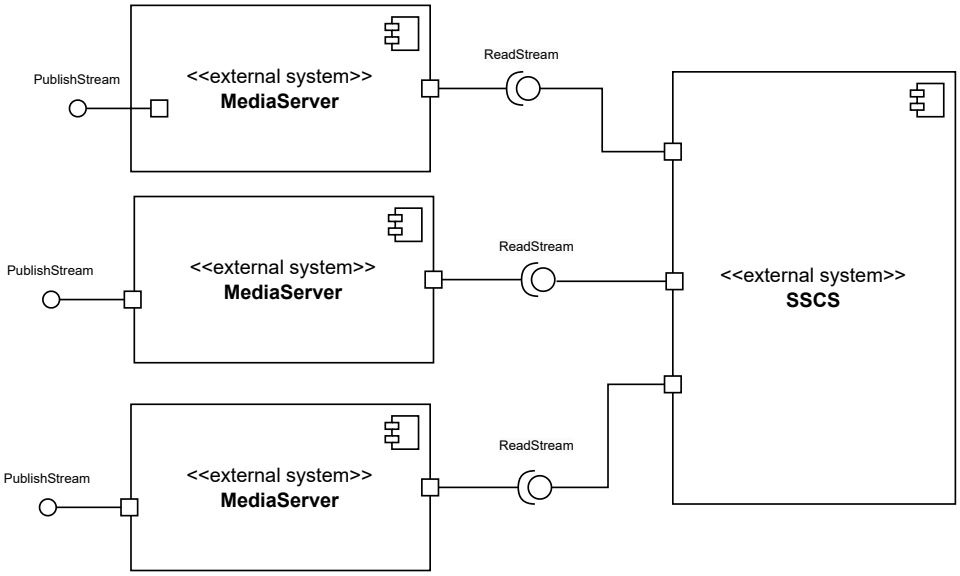

Figura 4 – Arquitetura de transmissão, múltiplos servidores de mídia

A arquitetura na figura 3, demonstra um caso onde as câmeras ou outras fontes de streaming podem enviar streams para um único servidor intermediário. Esse servidor intermediário

6 Conclusão

E daí?

Referências

ABDALLA, P. A.; VAROL, C. Testing iot security: The case study of an ip camera. In: *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*. [S.l.: s.n.], 2020. p. 1–5. Citado na página 14.

BELLARD, F. *FFmpeg*. 2023. GitHub repository. Disponível em: <https://github.com/FFmpeg/FFmpeg>. Citado na página 22.

BODUCH, A. *React and React Native*. [S.l.]: Packt Publishing Ltd, 2017. Citado na página 16.

BRADLEY, N. S. J. *What is PostgreSQL?* 2015. <https://www.postgresql.org/about/>. Citado na página 17.

DAHL, R. *Sobre Node.js®*. 2009. <https://nodejs.org/pt-br/about/>. Citado na página 16.

GLINSKY, A. *Theremin: ether music and espionage*. Urbana: University of Illinois Press, 2000. 46–47 p. ISBN 0252025822. Citado na página 13.

GROUP, T. P. G. D. *What is PostgreSQL?* 1996. <https://www.postgresql.org/about/>. Citado na página 17.

GUPTA, A. *Video Surveillance Market Research Report Information By Component (Hardware, Software, Services), By Hardware (Camera, Storage System And Others), By Application (Residential, Commercial, Defense And Infrastructure), And By Region (North America, Europe, Asia-Pacific, And Rest Of The World) – Market Forecast Till 2030*. 2018. Disponível em: <https://www.marketresearchfuture.com/reports/video-surveillance-market-95q7>. Citado na página 10.

LEE, J. *Scalable continuous media streaming systems: Architecture, design, analysis and implementation*. [S.l.]: John Wiley & Sons, 2005. Citado na página 15.

MARDJAN, M. J.; JAHAN, A. Open reference architecture for security and privacy. CreateSpace Independent Publishing Platform, 2016. Citado na página 10.

ORSIN, L. *What You Need To Know About Node.js*. 2013. <https://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/>. Citado na página 16.

QIANG, X. The road to digital unfreedom: President xi's surveillance state. *Journal of Democracy*, Johns Hopkins University Press, v. 30, n. 1, p. 53–67, 2019. Citado na página 10.

ROS, A. *MediaMTX*. 2023. GitHub repository. Disponível em: <https://github.com/bluenviron/mediamtx>. Citado na página 22.

TEAM, O. *About Open CV*. <https://opencv.org/about/>. Citado na página 17.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001.* [S.l.], 2001. v. 1, p. I–I. Citado na página 17.

Apêndices

APÊNDICE A – Quisque libero justo

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

APÊNDICE B – Coisas que fiz e que achei interessante mas não tanto para entrar no corpo do texto

Nunc velit. Nullam elit sapien, eleifend eu, commodo nec, semper sit amet, elit. Nulla lectus risus, condimentum ut, laoreet eget, viverra nec, odio. Proin lobortis. Curabitur dictum arcu vel wisi. Cras id nulla venenatis tortor congue ultrices. Pellentesque eget pede. Sed eleifend sagittis elit. Nam sed tellus sit amet lectus ullamcorper tristique. Mauris enim sem, tristique eu, accumsan at, scelerisque vulputate, neque. Quisque lacus. Donec et ipsum sit amet elit nonummy aliquet. Sed viverra nisl at sem. Nam diam. Mauris ut dolor. Curabitur ornare tortor cursus velit.

Morbi tincidunt posuere arcu. Cras venenatis est vitae dolor. Vivamus scelerisque semper mi. Donec ipsum arcu, consequat scelerisque, viverra id, dictum at, metus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut pede sem, tempus ut, porttitor bibendum, molestie eu, elit. Suspendisse potenti. Sed id lectus sit amet purus faucibus vehicula. Praesent sed sem non dui pharetra interdum. Nam viverra ultrices magna.

Aenean laoreet aliquam orci. Nunc interdum elementum urna. Quisque erat. Nullam tempor neque. Maecenas velit nibh, scelerisque a, consequat ut, viverra in, enim. Duis magna. Donec odio neque, tristique et, tincidunt eu, rhoncus ac, nunc. Mauris malesuada malesuada elit. Etiam lacus mauris, pretium vel, blandit in, ultricies id, libero. Phasellus bibendum erat ut diam. In congue imperdiet lectus.

Anexos

ANEXO A – Eu sempre quis aprender latim

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANEXO B – Coisas que eu não fiz mas que achei interessante o suficiente para colocar aqui

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

ANEXO C – Fusce facilisis lacinia dui

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.