FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS

FATEC PROFESSOR Jessen Vidal

NATHAN MAIA VILAR NUENS

NOFAPEX - SISTEMA PARA BLOQUEIO DE CONTEÚDO ADULTO E CONTROLE PARENTAL

São José dos Campos

2018

SUMÁRIO

[1 Introdução 3](#_Toc532182753)

[1.1 Definição do problema 3](#_Toc532182754)

[1.2 Objetivo 4](#_Toc532182755)

[2 Desenvolvimento 4](#_Toc532182756)

[2.1 Arquitetura 4](#_Toc532182757)

[2.1.1 Extensão Mozilla Firefox 5](#_Toc532182758)

[2.1.2 Servidor Flask WSGI 14](#_Toc532182759)

[2.1.3 Inteligência Artificial Amazon 19](#_Toc532182760)

[2.1.4 MongoDB 19](#_Toc532182761)

[2.1.5 DigitalOcean 20](#_Toc532182762)

[2.2 Modelo de Dados 20](#_Toc532182763)

[2.3 Detalhes 21](#_Toc532182764)

[3 Resultados e Discussão 26](#_Toc532182765)

# Introdução

O Manual Diagnóstico e Estatístico de Transtornos Mentais (DSM–5) considera jogos online na internet uma substancia nociva a saúde (SUD). Foi adicionado à sessão 3 da sua versão mais recente lançada em 2013. Alguns estudos, no entanto, avaliam o caráter patológico da internet de forma mais ampla (Durkee et al., 2012) (Fu, et al., 2010) e abrem o caminho para pesquisas de outros vícios comportamentais que possam ter relação com a grande rede (Petry, 2013).

Desde o aparecimento da pornografia na internet à acessibilidade, a demanda infinita e o anonimato do consumo de estímulos sexuais visuais aumentaram e atraíram milhões de usuários (Cooper, 2000). Estudos recentes sobre o funcionamento do cérebro humano apontam que o consumo excessivo de pornografia pode levar ao vício (Kühn S, 2014) e ao desenvolvimento de disfunções psicológicas como depressão, ansiedade, stress, diminuição do apetite sexual, entre outras (Voon V et al., 2014).

Desta forma, este trabalho de graduação tem por objetivo desenvolver uma extensão para o navegador Mozila Firefox que sirva como ferramenta de controle para pais que desejam evitar que seus filhos tenham acesso à conteúdos adultos na internet.

## Definição do problema

Estatísticas do instituto americano de proteção das crianças e da família na internet, Enough Is Enough, apontam que o primeiro contato de uma criança com a pornografia ocorre aos 11 anos de idade. Essa exposição pode ocorrer de forma voluntária ou não.

O contato precoce maximiza os efeitos colaterais tendo em vista que o cérebro de crianças e adolescentes por ainda estarem em formação possui uma capacidade de reação aos estímulos de recompensa de dopamina muito maior que os de adultos (Casey et al., 2010). Isso as torna mais suscetíveis ao vício e vulneráveis ao desenvolvimento de doenças mentais (Moghaddam, 2011).

Uma forma mais eficaz e assertiva de se lidar com possíveis problemas de saúde mentais em crianças e adolescentes, é investir na prevenção e tratamento dentro de um modelo integrado (Weisz et al., 2005).

Existem atualmente algumas formas diferentes de auxiliar os pais na missão de manter seus filhos longe de conteúdos impróprios. Uma delas é funcionalidade nativa do mecanismo de busca do Google que consiste em apenas restringir resultados de buscas na plataforma o que é insuficiente quando considerado todas as possibilidades.

## Objetivo

Este trabalho tem por objetivo desenvolver uma extensão para o navegador Mozila Firefox, visando prover uma ferramenta de controle parental capaz de avaliar e bloquear imagens com conteúdo adulto no formato jpg.

# Desenvolvimento

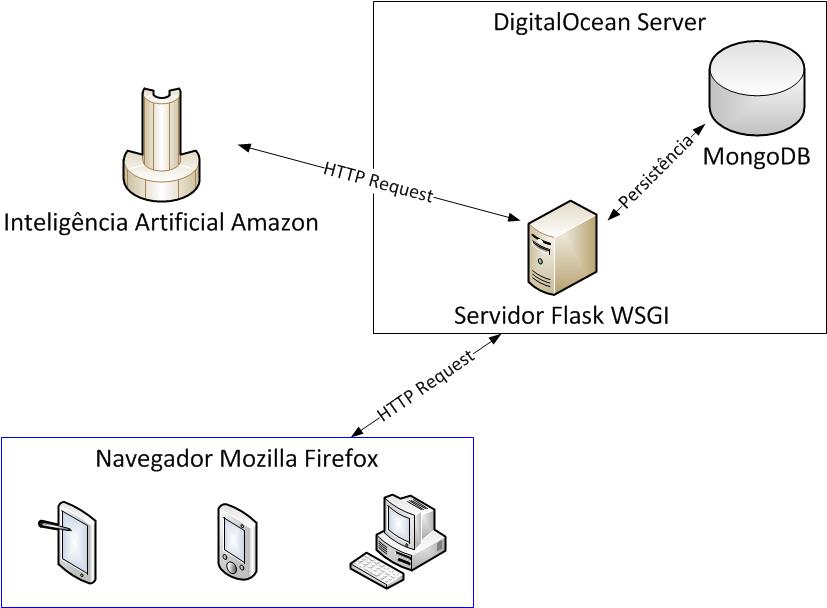
O sistema desenvolvido denominado NoFapEx é uma solução robusta que utiliza diversos recursos tecnológicos como: inteligência artificial, extensão de navegador, *WSGI* *web framework,* banco de dados não relacional e *REST API*.

As tecnologias serão apresentadas em detalhe no próximo tópico vide arquitetura da solução.

## Arquitetura

A Figura 1 ilustra todos os componentes utilizados e como interagem entre si.

**Figura 1 – Arquitetura Global.**



Fonte: Autor (2018).

### Extensão Mozilla Firefox

Extensões são pequenos softwares que permitem ampliar e modificar funcionalidades de um navegador de internet através de WebExtensions API, um padrão de desenvolvimento suportado por diversos navegadores como Google Chrome, Opera, Mozilla Firefox entre outros (Mozzila Web Doc). Neste caso, as funcionalidades operam com o navegador em modo normal e anônimo. Seus componentes são basicamente Javascript, CSS e HTML. Para o desenvolvimento deste trabalho foram utilizadas as bibliotecas Jquery-3.3.1 e Bootstrap

A extensão NoFapEx quando instalada, cria um pequeno ícone na barra de ferramentas padrão conforme mostra a Figura 2.

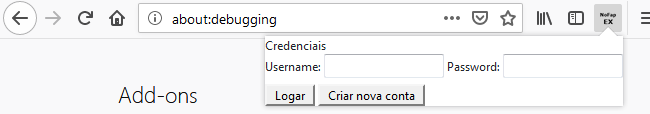
**Figura 2 – Ícone NoFapEx.**



Fonte: Autor (2018).

Ao clicar no ícone os usuários visualizam um modal para inserir as credenciais de acesso da conta, conforme representado na Figura 3.

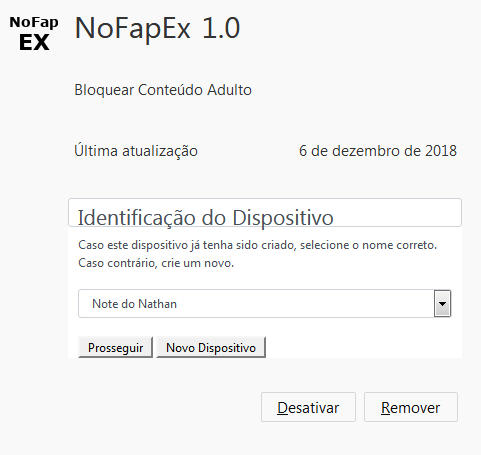
**Figura 3 – Modal de *login*.**



Fonte: Autor (2018).

Caso as informações fornecidas estejam corretas, o usuário será notificado que o do sucesso da operação e redirecionado imediatamente para a tela de identificação do dispositivo, de acordo com a Figura 4.

**Figura 4 – Identificação do Dispositivo.**

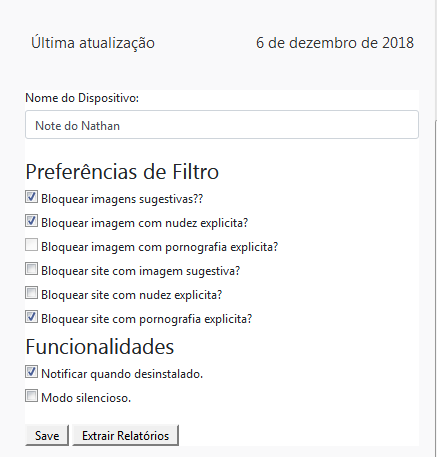


Fonte: Autor (2018).

Sempre que o usuário efetua o *login* após instalação, ele será solicitado a identificar o dispositivo ou criar um novo. Isso ocorre porque toda vez que a extensão é desinstalada, o *local storage* é zerado. Algumas alternativas para isso foram testadas durante o desenvolvimento e serão detalhadas no capítulo de Resultados.

Independente da escolha do usuário por utilizar um dispositivo já existente ou criar um novo, o próximo passo será a configuração das funcionalidades. A única diferença, é que um dispositivo existente já possui alguma configuração pré definida que estava armazenada no servidor, como indica a Figura 5

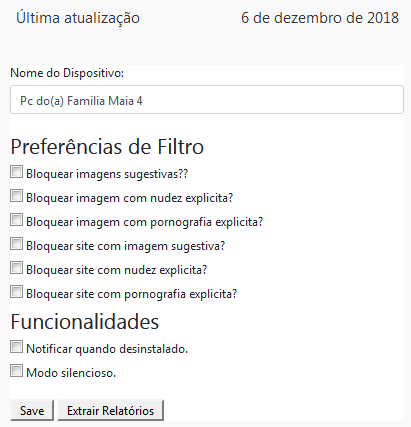
**Figura 5 – Dispositivo já existente.**



Fonte: Autor (2018).

Um dispositivo novo é criado sem nenhuma configuração e com um nome já definido, conforme elucidado na Figura 6.

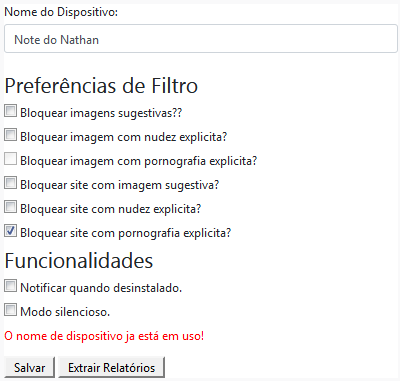
**Figura 6 – Novo dispositivo.**



Fonte: Autor (2018).

Para salvar, existem algumas validações. Uma delas é em relação ao nome do dispositivo que deve ser único dentro da conta. A Figura 7 apresenta o erro mostrado nesse caso.

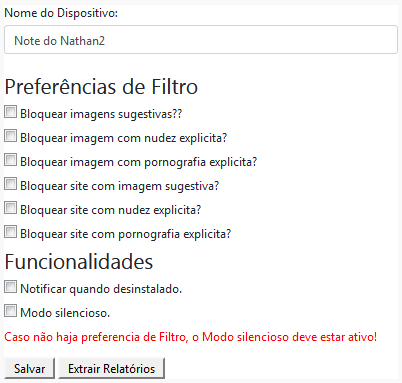
**Figura 7 – Erro de nome duplicado.**



Fonte: Autor (2018).

Outra validação existente diz respeito às opções de filtro conforme indica a Figura 8.

**Figura 8 – Validação preferências de filtro.**

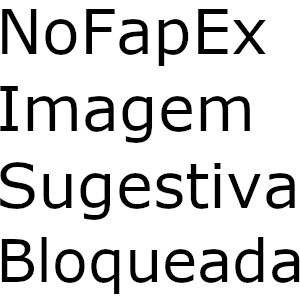


Fonte: Autor (2018).

Uma vez respeitadas todas as regras, as configurações serão salvas, uma mensagem exibida ao usuário e a aba fechada. A partir disto, o sistema NoFapEx já estará operando e avaliando o conteúdo da navegação conforme a Figura 19 do capítulo Inteligência Artificial Amazon.

Para as opções de bloqueio de imagem, a partir do momento em que o conteúdo impróprio é detectado o mesmo é substituído e em seu lugar mostrado imagens que explicam a interrupção da navegação conforme indicado nas Figuras 9, 10 e 11.

**Figura 9 – NoFapEx imagem sugestiva.**



Fonte: Autor (2018).

**Figura 10 – NoFapEx imagem nudez.**



Fonte: Autor (2018).

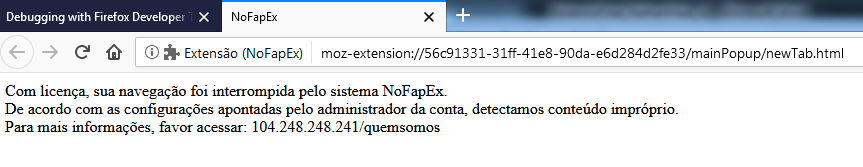
**Figura 11 – NoFapEx imagem pornográfica.**



Fonte: Autor (2018).

No caso das opções para bloquear todo o site, ao encontrar conteúdo impróprio a aba do mesmo é fechada e uma nova se abre com a mensagem mostrada na Figura 12.

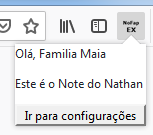
**Figura 12 – Mensagem site bloqueado.**



Fonte: Autor (2018).

Uma vez que a conta já tenha sido sincronizada na extensão, caso o usuário clique novamente no ícone da barra de ferramentas o modal mostrará uma mensagem referenciando o *username* da conta e o nome do dispositivo conforme ilustra a Figura 13.

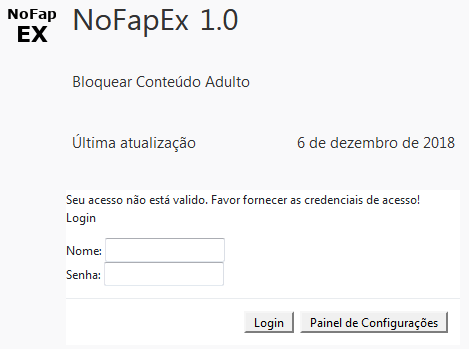
**Figura 13 – Extensão sincronizada.**



Fonte: Autor (2018).

Partindo da premissa que apenas o administrador da conta poderá alterar as configurações, sempre ocorrerá uma verificação antes que a Figura 6 seja disponibilizada. Esta verificação acontece por meio de uma *access token* que fica armazenada na *local storage* do navegador. Ela possui validade de apenas 5 minutos. Quando expirada, um novo *login* deve ser realizado, conforme Figura 14.

**Figura 14 – Token inválida**



Fonte: Autor (2018).

O *local storage* do navegador é onde informações importantes para o funcionamento da extensão ficam armazenadas. Elas são constantemente recuperadas, reescritas e atualizadas. Por padrão estão sempre em formato de JSON. Foram definidos então 3 objetos principais: *User\_Settings, Token\_Settings e Device\_Settings* conforme Figura 15.

**Figura 15 – Local Storage JSON.**



Fonte: Autor (2018).

O objeto *Token\_Settings* guarda as chaves para utilização de rotas restritas no servidor. São geradas e transmitidas depois das operações de login. Exercem papel importante nos mecanismos de segurança conhecido como *JSON Web Token.*

Em resumo, a extensão opera como um interceptador que observa tudo o que é acessado pelos usuários, envia ao servidor, recebe e trata as respostas para tomar a decisão de liberar ou não as imagens.

A Figura 16 apresenta um exemplo de como fica a navegação com alguns bloqueios.

**Figura 16 - Exemplo final.**



Fonte: Autor (2018).

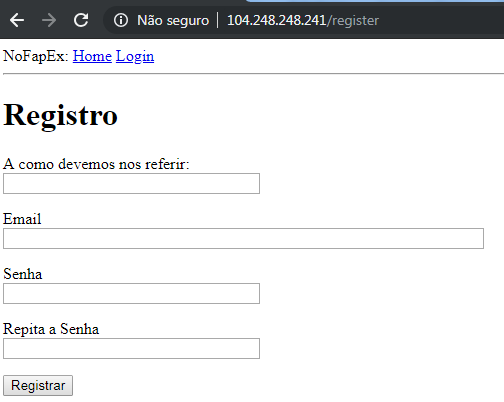
### Servidor Flask WSGI

O ponto central na arquitetura, responsável por intermediar os *insights* provindos da Amazon e a tomada de decisão da extensão no navegador.

No presente trabalho foram utilizadas as seguintes bibliotecas: flask\_login, flask\_jwt\_extended, flask\_mongoengine, flask\_restful, flask\_cors, werkzeug, boto3, json e datetime

A conta a ser utilizada na extensão deve ser criada no endereço http://104.248.248.241/register conforme aponta a Figura 17.

**Figura 17 – Tela de cadastro.**



Fonte: Autor (2018).

A Tabela 1 apresenta todas as rotas para acesso das API’s criadas.

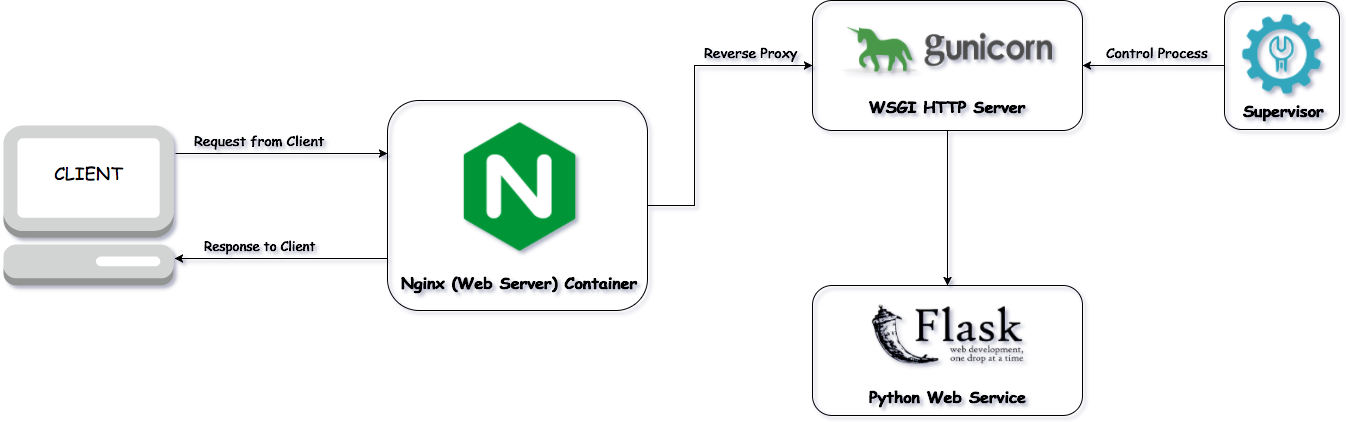
**Tabela 1 – Rotas de serviço.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **URL: HTTP://104.248.248.241/** | **Método HTTP** | **Parâmetro** | **Retorno** | **Código HTTP** |
| login | POST | {“username”:string, “password”:string} | {"access\_token":string,  "refresh\_token":string,  "username" : string,  "email" : string } | 200 |
| {“msg”:Credenciais incorretas”} | 204 |
| checkDeviceName | POST | {"deviceHash":string,  "username" : string,  "deviceAlias": string} | {“msg”:Nome livre para uso nesta conta”} | 200 |
| {“msg”:”Nome existente nessa conta”} | 204 |
| createNewDevice | POST | headers:{“Authorizati  zation”:”Bearer “+token} | {"deviceAlias":string,  "deviceHash":string,  "blockImgSugest":boolean,"blockSiteSugest":boolean,"blockImgPorn":boolean,"blockImgNud" : boolean,"blockSitePorn":boolean,"blockSiteNud":boolean,"notifDesinst":boolean,"modSilencioso":boolean} | 200 |
| {“msg”:”Token has expired”} | 401 |
| getAccountDevices | POST | {“username”: string} | {[“deviceAlias” : string ]} | 200 |
| {“msg”:Não há dispositivos nessa conta”} | 204 |
| getDeviceHash | POST | {“username”:string, “deviceAlias”:string} | {"deviceAlias":string,  “deviceHash":string,  "blockImgSugest":boolean,"blockSiteSugest" :boolean,"blockImgPorn":boolean,"blockImgNud":boolean,“blockSitePorn":boolean,"blockSiteNud":boolean,"notifDesinst":boolean,"modSilencioso":boolean} | 200 |
| {“msg”:”Nenhum dispositivo com este alias”} | 204 |
| rekognition | POST | {“url”: string} | {["confidence":float,"name":string,  "parentName":string ]} | 200 |
| {“msg”: “Não foi possível baixar a imagem”} | 204 |
| saveDeviceSetting | POST | {"deviceAlias":string,  "deviceHash":string,  "blockImgSugest":boolean,"blockSiteSugest":boolean,"blockImgPorn":boolean,"blockImgNud":boolean,"blockSitePorn":boolean,"blockSiteNud":boolean,"notifDesinst":boolean,  modSilencioso: boolean} | {“msg”:”Configuração Salva”} | 200 |
| {“msg”:”Algum parametro faltou”} | 204 |
| updateClientDevice | POST | {“username”:string, “deviceHash”:string} | {"deviceAlias":string,  “deviceHash":string,  "blockImgSugest":boolean,"blockSiteSugest" :boolean,"blockImgPorn":boolean,"blockImgNud":boolean,“blockSitePorn":boolean,"blockSiteNud":boolean,"notifDesinst":boolean,"modSilencioso":boolean} | 200 |
| {“msg”:”Algum parametro faltou”} | 204 |
| verificaTokenValida | GET | Headers: {“Authorization”: “Bearer “ + token} | {“msg”:”Token valida”} | 200 |
| {“msg”:”Token has expired”} | 401 |

Fonte: Autor (2018).

Para que o servidor seja capaz de suportar diversas requisições às rotas REST criadas, foi necessário criar uma solução WSGI robusta para a aplicação, conforme pode-se constatar na Figura 18.

**Figura 18 – Arquitetura Aplicação WSGI.**



Fonte: Site Medium - Rahul (2018).

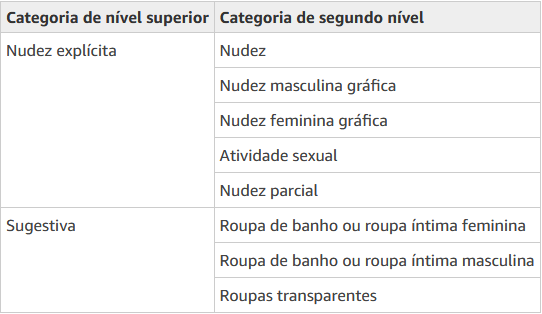
### Inteligência Artificial Amazon

O Amazon Rekognition e um serviço de análises de imagens e vídeos para aplicativos baseado em inteligência artificial da Amazon. Diversos tipos de análises são possíveis como detectar textos de imagens, detectar objetos e atores em imagens, detectar celebridades em imagens ou até analises completas em vídeos.

O sistema desenvolvido neste trabalho utiliza o serviço de detecção de conteúdos não seguros: DetectModerationLabels - através da biblioteca boto3 para a linguagem de progração Python.

A Figura 19 mostra as possibilidades de resposta de acordo com a imagem fornecida.

**Figura 19 – Resposta DetectModerationLabel**



Fonte: AWS Documentation (2018).

### MongoDB

Um banco de dados não relacional baseado em documentos de dados. Foi escolhido para solução devido a sua fácil integração com a linguagem de programação Python.

Está instalado no servidor e atua com o Flask via Mongo Engine. Será detalhado na sessão de Modelo de Dados.

### DigitalOcean

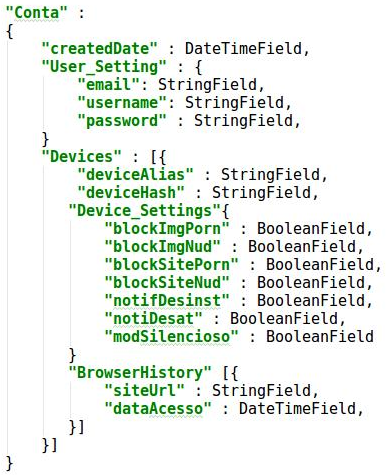
Um provedor de infraestrutura em nuvem que hospeda o servidor Flask, viabilizando a solução NoFapEx que foi criada em cima do sistema operacional Linux Ubuntu 16.04

## Modelo de Dados

No total foram criados apenas 2 *collections* principais para manipular os dicionários de dados na camada do servidor.

A Figura 20 ilustra como as contas estão estruturadas.

**Figura 20 – Documento Conta**

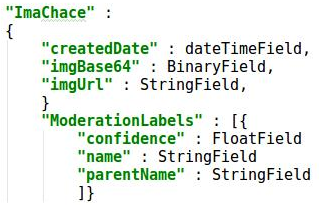
****

Fonte: Autor (2018).

Este é o documento que armazena as informações das contas dos usuários bem como todos os dispositivos contidos na conta. Cada dispositivo terá um conjunto de configurações e um *array* sobre o histórico de navegação do mesmo.

Por fim, a Figura 21 apresenta como as imagens e suas avaliações são persistidas.

**Figura 21 – Documento imgCache.**

****

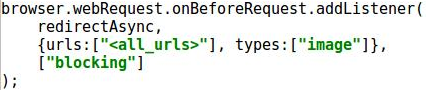
Fonte: Autor (2018).

É neste documento que o servidor vai salvar cada imagem avaliada para otimizar futuras consultas.

## Detalhes

A extensão opera interceptando as requisições HTTP de todas as imagens envolvidas no site a ser acessado pelo usuário. Isso só é possível através do *listener* demonstrado na Figura 22.

**Figura 22 – Código do interceptador de imagens**

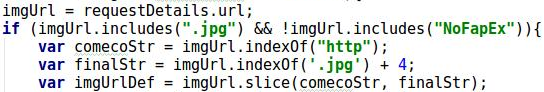


Fonte: Autor (2018).

Fica definido então que todas as url’s que contiverem imagem serão submetidas a função de *callback* denominada redirectAsync com a permissão para bloqueio. As imagens podem ser do tipo jpeg e png.

A primeira coisa a ser feita pela função é tratar as url’s que são obtidas pelo objeto de retorno chamado requestDetail, pois muitas delas possuem informações que poderiam atrapalhar o servidor quando for baixar a imagem conforme se pode notar na Figura 23.

**Figura 23 – Tratamento de url**

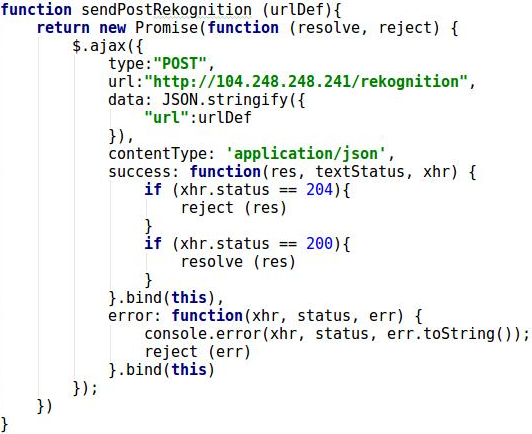


Fonte: Autor (2018).

Para evitar um loop entre as interceptações, foi adicionado uma condição avaliando o nome da imagem que fica armazenada no servidor do NoFapEx.

O próximo passo é solicitar ao servidor as análises sobre o conteúdo da url. Isso é feito através de uma requisição REST utilizando a biblioteca Jquery (AJAX) conforme Figura 24.

**Figura 24 – Função sendPostRekognition**



Fonte: Autor (2018).

Esta requisição fica encapsulada e retorna uma *promise* para que seja possível realizar múltiplas chamadas conforme a necessidade. Ela recebe a ulr definitiva que foi tratada e, em caso de sucesso, retorna a resposta que vem do servidor Flask.

Sendo assim, o retorno da função redirectAsync também será uma *promise* que chama o serviço de reconhecimento rekognition do servidor conforme Figura 25.

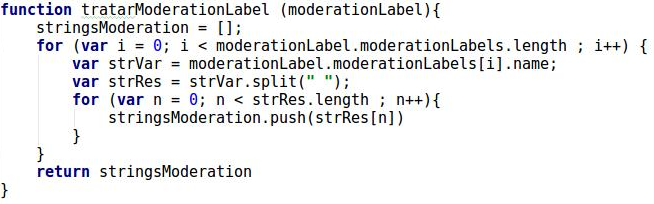
**Figura 25 – Retorno da função redirectAsync**



Fonte: Autor (2018).

Portanto na primeira função da *promise* da Figura 25, que é a função do rosolve, espera-se receber um array de moderationLabels proveniente do servidor e que deve ser tratado pela função tratarModetaionLabel ilustrada na Figura 26.

**Figura 26 – Função tratarModerationLabel**

****

Fonte: Autor (2018).

Esta é a função que trata a resposta recebida do servidor. Ela monta um array de palavras para facilitar as comparações futuras.

Tendo isto pronto, resta consultar as preferências de funcionamento previamente salvas pelo usuário para entender qual a tomada de decisão esperada. Conforme exemplifica a Figura 27.

**Figura 27 – Exemplo da tomada de decisão.**

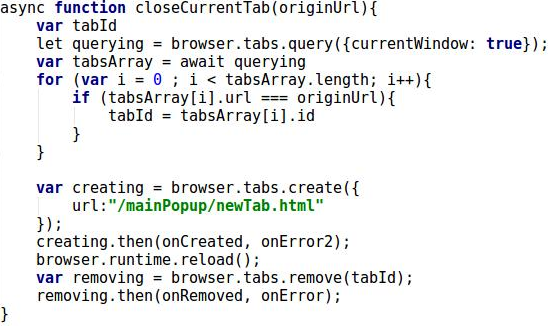


Fonte: Autor (2018).

Então quando o usuário escolheu alguma das opções para bloqueio de imagem, a *promise* principal da função redirectAsync retorna um json com a propridade de redirectUrl que fará com que outra imagem seja carregada no lugar do conteúdo impróprio.

Já quando o usuário espera que o site seja bloqueado, a função da Figura 28 é chamada.

**Figura 28 – Função closeCurrentTab.**

****

Fonte: Autor (2018).

Esta função utiliza alguns métodos para obter o Id da tab do navegador onde se encontra o site a ser bloqueado para então fechá-lo. Na seqüencia, cria uma nova tab que abre uma página contida na extensão do navegador e que mostrará a mensagem contida na Figura 12.

Quando a função redirectAsync cai no reject, ela retorna um simples json: ({cancel: false})) que libera a imagem para ser carregada pelo navegador.

Na parte do servidor Flask, a Figura 29 apresenta a estrutura do principal objeto utilizado para armazenar as informações das contas e dispositivos dos usuários.

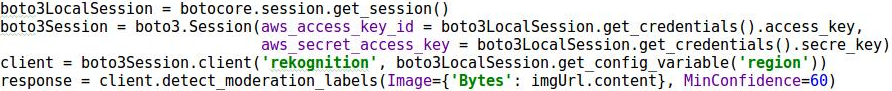
**Figura 29 – Estrutura do objeto conta.**



Fonte: Autor (2018).

Em relação ao serviço Rekognition que é consumido da Amazon via biblioteca boto3, ele é demonstrado na Figura 30.

**Figura 30 – Boto3 Rekognition.**

****

Fonte: Autor (2018).

As credenciais de acesso (Access\_Key e secret\_access\_key) foram previamente configuradas no servidor através da *AWS Command Line Interface* por questões de segurança.

Importante ressaltar que a variável *MinConfidence* determina um mínimo de confiança nas respostas de 60%. Ou seja, qualquer avaliação da inteligência artificial abaixo deste número não será retornada ao servidor.

# Resultados e Discussão

Inicialmente o projeto foi planejado e projetado para que a extensão funcionasse no navegador Google Chrome. Decisão baseada na quantidade majoritária de dispositivos que utilizam este navegador na grande Web, segundo fontes da W3C.

Em determinado momento do desenvolvimento, foi identificado que o principal *listener* apontado na Figura 22 não aceitaria como retorno da função *callback*, uma *promise.* Fato este que inviabilizaria a extensão neste navegador uma vez que, as chamadas assíncronas devem aguardar o retorno da resposta do servidor para cada url.

Este contra ponta está evidenciado no site oficial de bugs do Chromium: https://bugs.chromium.org/p/chromium/issues/detail?id=904365#c5

Em contra partida, o desenvolvimento do projeto fluiu sem maiores problemas na plataforma Firefox que mostrou ter mecanismos mais flexíveis e compatíveis com funções e recursos mais atuais da linguagem Javascript.

Outro problema enfrentado foi em relação à biblioteca em Javascript fingerprint2 que supostamente teria a capacidade de gerar um código *hash* único para cada navegador de um dispositivo. Este seria um recurso importante para melhorar a experiência dos usuários em relação à configuração do dispositivo na extensão conforme foi citado na figura 4 do presente relatório.

Depois de diversos testes chegou-se a conclusão que uma simples alteração nas configurações do navegador alteraria o resultado do código gerado. Isto demandaria uma tratativa maior nos algoritmos de reconhecimento do navegador no servidor Flask. Isso porque partindo do princípio que cada navegador teria um identificador único, ele logicamente seria uma chave que caso trocada, geraria a criação de um novo dispositivo.

A solução foi cancelar o uso da mesma e transferir ao usuário a responsabilidade de identificar se o dispositivo em questão já havia sido criado ou não. A partir disto uma nova chave passou a ser gerada mas no servidor Flask via biblioteca random, método getrandbits(). Este passou a ser o identificador de cada dispositivo juntamente com o seu nome. Todo esse controle é transparente ao usuário.

Em relação ao serviço da Amazon, foi identificada uma incoerência nas respostas de imagens que estão em menores resoluções. Ou seja, uma imagem com resoluções 500x500 é interpretada normalmente mas a mesma imagem em 100x80 não. Suas respostas de moderation label são distintas.

De maneira geral, a extensão mostrou-se eficaz no bloqueio das imagens e dos sites. A comunicação ocorre de maneira rápida, levando em média de 2 a 3 segundos para completar todo processo de avaliação e retornar a resposta ao navegador.

**REFERÊNCIAS**

Durkee, T. , Kaess, M. , Carli, V. , Parzer, P. , Wasserman, C. , Floderus, B. , Apter, A. , Balazs, J. , Barzilay, S. , Bobes, J. , Brunner, R. , Corcoran, P. , Cosman, D. , Cotter, P. , Despalins, R. , Graber, N. , Guillemin, F. , Haring, C. , Kahn, J. , Mandelli, L. , Marusic, D. , Mészáros, G. , Musa, G. J., Postuvan, V. , Resch, F. , Saiz, P. A., Sisask, M. , Varnik, A. , Sarchiapone, M. , Hoven, C. W. and Wasserman, D. (2012), Pathological internet use among adolescents. Addiction, 107: 2210-2222. doi:10.1111/j.1360-0443.2012.03946.x

Fu, K., Chan, W., Wong, P., & Yip, P. (2010). Internet addiction: Prevalence, discriminant validity and correlates among adolescents in Hong Kong. British Journal of Psychiatry, 196(6), 486-492. doi:10.1192/bjp.bp.109.075002

Petry, N. M. and O'Brien, C. P. (2013), Internet gaming disorder and the DSM‐5. Addiction, 108: 1186-1187. doi:10.1111/add.12162

Cooper, Al. Cybersex: The Dark Side of the Force: A Special Issue of the Journal Sexual Addiction & Compulsivity. London, United Kingdom: Brunner-Routledge; 2000.

Kühn S, Gallinat J. Brain Structure and Functional Connectivity Associated With Pornography ConsumptionThe Brain on Porn. JAMA Psychiatry. 2014;71(7):827–834. doi:10.1001/jamapsychiatry.2014.93

Voon V, Mole TB, Banca P, Porter L, Morris L, Mitchell S, et al. (2014) Neural Correlates of Sexual Cue Reactivity in Individuals with and without Compulsive Sexual Behaviours. PLoS ONE 9(7): e102419. https://doi.org/10.1371/journal.pone.0102419

B.J. Casey, Rebecca M. Jones, Neurobiology of the Adolescent Brain and Behavior: Implications for Substance Use Disorders, Journal of the American Academy of Child & Adolescent Psychiatry, Volume 49, Issue 12, 2010, Pages 1189-1201,ISSN 0890-8567,

https://doi.org/10.1016/j.jaac.2010.08.017.

Moghaddam, Bita. Teen brains over-process rewards, suggesting root of risky behavior, mental ills (2011,January 26).

Weisz, J. R., Sandler, I. N., Durlak, J. A., & Anton, B. S. (2005). Promoting and Protecting Youth Mental Health Through Evidence-Based Prevention and Treatment. American Psychologist, 60(6), 628-648.

https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

https://medium.com/ymedialabs-innovation/deploy-flask-app-with-nginx-using-gunicorn-and-supervisor-d7a93aa07c18