SHIELD: SISTEMA PARA PROTEÇÃO DE CÓDIGO PROPRIETÁRIO EM APLICAÇÕES MULTIPLATAFORMA

Pedro Welbert da Silva Pereira[1*], Victor Silva Andeloci[1*], Rodrigo de Oliveira Plotze[1]

[1] Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)
Ribeirão Preto, SP – Brasil

[*] Os autores contribuíram igualmente para o desenvolvimento do trabalho

pedro@valkyriatech.com.br, victor@valkyriatech.com.br,
rodrigo.plotze@fatec.sp.gov.br

Resumo. *O posicionamento da publicidade digital no mercado e sua iminente igualdade com os anúncios tradicionais, resultou em uma disposição pelo mercado online, da qual empresários buscam conquistar espaços através de aplicações, codificadas por desenvolvedores autônomos em troca de compensação financeira, inexistindo vínculo empregatício ou contratual. A plataforma visa proteger o desenvolvedor contra interferências dolosas do contratante e servir de ferramenta para o gerenciamento de acesso a tais aplicações. O Shield foi desenvolvido em diversas tecnologias e é um sistema de gerenciamento que bloqueia total ou parcialmente, de forma acessível, a propriedade intelectual. Disponível em: https://shield.valkyriatech.com.br*

Abstract. *The placement of digital advertising in the market and its imminent equality with traditional ads, resulted in a disposition for the online market, of which entrepreneurs seek to conquer spaces through web, desktop and mobile applications, coded by independent developers in exchange for financial compensation, without employment or contractual relationship. The platform aims to protect the developer against alleged intentional interference by the defaulting contractor, as well as to serve as a tool for managing access to such applications. Shield was developed in several technologies and is a management system that blocks, in whole or in part, in a succinct and accessible way, the intellectual property. Available at: https://shield.valkyriatech.com.br*

1. Introdução

A publicidade digital deve se igualar à tradicional até 2023, segundo a consultoria PwC, na 20° edição da Pesquisa Global de Entretenimento e Mídia (BALAN, 2019). Feito em 53 países, o estudo faz projeções de investimento anual de US $468 bilhões no ramo, sendo uma parte deste montante, de US $3 bilhões ao Brasil em 2019.

Um nicho deste mercado se refere ao posicionamento de pequenas e médias empresas no mercado online. O Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação informa que, em 2018, cerca de 93% dos domicílios possuíam ao menos um telefone celular ou um computador de mesa no Brasil (CETIC, 2018). Números estes que despertam interesse de empresários em buscar profissionais para alcançar o mercado digital, por meio de aplicações web (sites e lojas virtuais), desktop

(programas para computadores de mesa) e mobile (voltados para dispositivos móveis). Os desenvolvedores são responsáveis pela construção de tais sistemas, tendo este como o produto intelectual e proprietário.

A codificação de um sistema é feita por um único ou grupo de profissionais da tecnologia da informação, utilizando uma ou mais tecnologias, sendo este responsável pela implementação do código proprietário, que é ofertado ao cliente final em troca de recompensa financeira, podendo ou não existir vínculo empregatício ou contratual entre as partes. Em contrapartida, um detalhe pode influenciar na firmação deste tipo de relação: O Serviço de Proteção ao Crédito – SPC Brasil, apurou que 61 milhões de pessoas, aproximadamente 29% dos brasileiros, começaram 2020 com alguma conta em atraso e com restrições para obter crédito ou fazer compras parceladas (G1, 2020).

Considerando o exposto, a plataforma visa proteger os dados da aplicação alvo contra supostas interferências fraudulentas por parte do cliente, enquanto trâmite da negociação estiver em andamento, preservando a integridade do produto intelectual (código proprietário) ofertado pelo desenvolvedor, garantindo segurança para o profissional sob o uso de seu produto, mesmo em caso de intenções dolosas do usuário final inadimplente.

2. Referencial teórico

Trabalhos acordados informalmente são comuns em diversas áreas relacionadas ao meio criativo: publicidade, comunicação e desenvolvimento de software. Comumente conhecidos como '*jobs'* ou '*freelas'*, são recorrentes principalmente para profissionais destas áreas que estão iniciando suas carreiras e/ou buscam novos projetos para complementar uma renda necessária. (MIRANDA, 2019).

Estes projetos são desenvolvidos, geralmente, sem qualquer auxílio ou suporte técnico por parte do cliente contratante. O desenvolvedor, designer ou artista, nesta relação denominado prestador de serviço, tem a responsabilidade de arcar com eventuais despesas técnicas durante toda a fase de desenvolvimento de qualquer que seja o projeto, até que o produto seja entregue ao contratante dentro do período anteriormente estipulado.

Essa forma de desenvolvimento, tratando-se de produtos de software ou peças artísticas, por mais comum que seja, não é única: em alguns casos, o prestador de serviços permanece alocado na empresa contratante ou recebe equipamentos e ferramentas para cumprir com sua parte do acordo. (MIRANDA, 2019).

2.1. A segurança do desenvolvedor de software

A vulnerabilidade dos profissionais que desenvolvem software em regime de trabalho intermitente é explícita na falta de um contrato formal. Estes prestadores de serviço acabam vulneráveis neste tipo de acordo informal, entre prestador e contratante, quando se trata da entrega do produto especificado no acordo informal e o recebimento do valor acordado, seja este monetário ou não.

Enquanto um artista, por sua vez, pode utilizar técnicas simples para proteger seus direitos autorais e garantir que o contratante cumprirá com sua parte do acordo, tais como marca d'água ou exemplificação da arte digital final em baixa resolução; o desenvolvedor de software, uma vez que sua parte no acordo consiste na instalação do sistema proprietário em ambientes adquiridos e controlados pelo cliente (servidores ou repositórios), não pode utilizar de tais ou semelhantes técnicas. (HAMAGUCHI, 2013).

Os resultados gerados por esse tipo de transação, quando ocorre inadimplência por parte do cliente contratante ou quando este age de má-fé, são desastrosos ao desenvolvedor de software, infringindo sua propriedade intelectual e a lei de direitos autorais, de acordo com o Art. 1o da Lei nº 9.610, de 19 de fevereiro de 1988 (BRASIL, 1988).

Embora seja uma infração legal, são comuns os casos nos quais o contratante não atende com sua parte do acordo. Tão comuns que até acabam viralizando em sites e redes sociais, como o caso de Rafael Fideli, que originou o movimento "Pague meu Freela" (SAROSINI, 2014).

Prestador de serviços relacionados a software para a internet, Rafael entrou em contato e estabeleceu um contrato informal com uma grande empresa. Mesmo após várias tentativas de cobrança, não obteve a remuneração anteriormente estabelecida. Rafael acabou por produzir um website e movimento digital bem-humorado, em crítica a esse tipo de ocorrido.

2.2. Plataformas para contratação de 'freelas'

No entanto, aos contratantes e prestadores de serviços existe uma opção de se proteger: um intermediador. Plataformas como a Workana, site e aplicativo construídos para realizar essa comunicação entre contratantes e prestadores, são regularmente utilizados.

Após a especificação de todos os detalhes do projeto a ser desenvolvido, o cliente contratante pode selecionar um ou alguns dentre os diversos profissionais previamente cadastrados para realizar a tarefa. A plataforma detém todo valor monetário investido pelo cliente e monitora toda comunicação. Assim que ambas as partes demonstram que estão igualmente satisfeitas com o resultado obtido, o valor anteriormente acordado é entregue ao prestador, assim como o sistema ou código proprietário é entregue à outra parte.

Este tipo de plataforma e seu funcionamento passam confiança a ambas as partes do acordo, porém seu uso não é regra, além do agravante de taxas cobradas pelo intermediador com base no valor final do projeto. Na maioria dos casos, esses acordos são firmados de forma informal, por meio de indicações ou por buscas simples em redes sociais.

2.3. Medidas de segurança

Dentre as principais ideias e medidas de proteção ao prestador do serviço requisitado, algumas se destacam (COSTA, 2018):

- Cobrar um valor de entrada antes de iniciar o desenvolvimento do projeto;
- Observar o comportamento do cliente e verificar um possível perigo a sua integridade;
- Um contrato formalizado.

Um contrato, embora eficaz, não é eficiente, seja este um e-mail, um documento formalizado ou uma conversa por mensagem. Questões judiciais requerem tempo e a necessidade de aparato legal responsável; o que, por sua vez, exige mais tempo e dinheiro investidos.

As dificuldades enfrentadas por profissionais autônomos são diversas. O perigo de não serem remunerados após a realização do serviço acordado é real e, tratando-se de desenvolvedores de software, não possui ferramenta ou medida de segurança que o torne completamente seguro(a).

3. Materiais e Métodos

As etapas para desenvolver a aplicação para o problema proposto foram divididas em subseções, cada uma focada em determinados itens que compõem o sistema geral e seu funcionamento.

3.1. Conceito de funcionamento

O sistema denominado 'Shield' é dividido em três partes, que operam em diferentes ambientes, escritas e desenvolvidas em diferentes tecnologias. A base de seu funcionamento e alguns de seus principais requisitos são a possibilidade de um usuário desativar ou ativar uma aplicação, independentemente de sua plataforma, sem necessidade de acesso ao ambiente de instalação, assim como gerenciar tais aplicações e usuários registrados em seu grupo (ou domínio).

3.1.1. Base e API

A base de dados da plataforma armazena as informações referentes aos usuários, seus domínios e aplicações. A API (do inglês, *Application Programming Interface*), é um sistema de identificação de ações a serem tomadas, respondendo a solicitações de seus sistemas auxiliares plataforma e plugin. A API é responsável por toda troca de informações entre os sistemas auxiliares e o banco de dados, assim como por outras atividades correlacionadas.

É importante lembrar que, por se tratar de um sistema que gerencia o acesso à determinada aplicação, a API não deve prejudicar a experiência do usuário final, causando lentidão ou quaisquer erros de processamento. Por esse motivo, como parte dos resultados já obtidos, conta com verificação do status do servidor antes da verificação da aplicação, assim como tempo de resposta geral inferior a 0,5 segundos.

3.1.2. Plugin

A parte denominada *Plugin* - extensão que adiciona uma funcionalidade ao sistema - deve ser instalada em conjunto com a aplicação cliente, em seu ambiente. Como será abordado em outro tópico, esse sistema auxiliar foi desenvolvido e já está disponível em diversas tecnologias, permitindo que aplicações de diferentes tipos de plataformas utilizem o sistema como um todo.

A requisição que o *plugin* faz à base de dados ocorre antes ou durante o carregamento da aplicação cliente, de forma assíncrona nas tecnologias que permitem tal recurso. Como parte do corpo da mensagem, um atributo é enviado à base: 'app_identifier'. Este atributo será utilizado pela API para identificar o atributo relativo àquela aplicação, mantido no banco de dados sob nome 'status'.

Após enviar a requisição, o *plugin* deve aguardar uma resposta da API, contendo um valor *boolean* (verdadeiro ou falso), informando se deve prosseguir com o carregamento da aplicação cliente ou não.

3.1.3. Plataforma

Esta parte do sistema é a responsável por gerir as aplicações e usuários, assim como seus atributos principais. A aplicação deve permitir que o usuário administrador manipule aplicações cliente e usuários de seu domínio, assim como o manuseio do atributo 'status', responsável pelo bloqueio e por consequência, proteção do código proprietário.

3.2. Tecnologias utilizadas

Dentre as três aplicações (ou partes) distintas que compõem o sistema, todas são escritas e desenvolvidas com o uso de diferentes linguagens e tecnologias. As três aplicações comunicam-se entre si, sendo responsáveis por consultar e gerir uma mesma base de dados. A apresentação de tais tecnologias e linguagens de programação utilizadas é importante para a compreensão de seu funcionamento e responsabilidades das respectivas partes.

3.2.1. MySQL

O banco de dados responsável pelo armazenamento das diferentes informações de usuários, aplicações e domínios registrados na plataforma foi desenvolvido em MySQL. Esta tecnologia é um Banco de Dados relacional (RDBMS – *Relational Database Management Systems*) de código aberto, com um modelo de 'cliente-servidor': software cliente que se conecta ao servidor conforme necessidade ou requisição. (ANDREI, 2019).

A tecnologia trabalha em um modelo de 'solicitação - resposta', na qual uma requisição é enviada à base de dados através da linguagem SQL (*Structured Query Language*) e interpretada pelo software. Dependendo do tipo de requisição, uma resposta será transmitida, contendo informações relevantes quanto às operações contidas no código SQL.

3.2.2. PHP

O PHP (um acrônimo recursivo para PHP: *Hypertext Preprocessor*) é uma linguagem de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web. (PHP, 2021). A linguagem é executada no lado servidor, sendo utilizada no desenvolvimento da API, responsável pelo gerenciamento e comunicação entre os sistemas auxiliares e a base de dados.

3.2.3. Flutter

Flutter é um *framework* - biblioteca com funcionalidades prontas para serem incorporadas ou utilizadas - desenvolvido em DART, que permite o desenvolvimento de aplicações nativas para as plataformas Android, iOS, *desktop* e web.

O *framework* opera através de *widgets* - blocos programados que contém determinados atributos e subsequentemente, outros blocos. Outro importante aspecto é a manipulação de estado em seus componentes, determinando a renderização de elementos gráficos e valores em tempo de execução.

3.2.4. ReactJS

ReactJS é uma biblioteca desenvolvida em JavaScript com foco semelhante ao Flutter: desenvolvimento de aplicações cliente com interfaces de usuário interativas, manipulação de componentes e de seus estados. ReactJS é costumeiramente utilizada em aplicações web, tornando-se uma das tecnologias principais por trás do sistema auxiliar denominado plataforma.

3.3. *Plugins* multiplataforma

O sistema auxiliar denominado *Plugin* não tem uma tecnologia de desenvolvimento única, existindo um foco em sua diversidade. O fato de toda a comunicação entre os sistemas auxiliares e a base de dados ser gerida pela API, permite que as tecnologias

exploradas no desenvolvimento de novos *Plugins* sejam variadas. Isso possibilita abranger mais desenvolvedores e aplicações cliente, de diferentes plataformas e tecnologias.

No momento da publicação deste trabalho, existem *Plugins* desenvolvidos nas seguintes tecnologias e linguagens de programação: PHP, JavaScript, Python e Flutter. Disponível em: https://github.com/ValkyriaTech/shield-plugins.

3.4. Levantamento de requisitos

Os requisitos funcionais e não funcionais das diferentes aplicações que compõem o sistema serão apresentados em forma de tabela. O documento de requisitos está disponível através do link: https://shield.valkyriatech.com.br/docs/documento_requisitos.pdf.

3.5. Modelagem do sistema

É importante desenhar as aplicações que compõem todo o sistema para facilitar e guiar o desenvolvimento, utilizando softwares para ilustrar os diferentes tipos de diagramas através de UML (*Unified Modeling Language*).

Os diagramas de Classe e Atividade do sistema, assim como o Modelo Relacional da base de dados foram desenvolvidos com as ferramentas StarUML e brModelo. Todos os diagramas estão disponíveis em: https://shield.valkyriatech.com.br/docs/.

4. Desenvolvimento

O desenvolvimento da aplicação *back-end* - *software* executado no servidor, responsável pela API e gerenciamento da base de dados - e dos sistemas auxiliares ocorreu de forma paralela. Funcionalidades e processos foram implementados na API conforme necessidade.

4.1. Banco de dados

O banco de dados foi desenvolvido em MySQL e compreende as tabelas relacionadas às informações armazenadas dos usuários, seus domínios e aplicações cliente. As tabelas que compõem sua estrutura são:

- **user** - responsável pelo armazenamento das informações dos usuários da plataforma, através dos atributos: *id, active, username, password, email* e *phone;*
- **domain** - responsável pelo armazenamento das informações dos domínios (grupos de usuários e aplicações cliente), através dos atributos: *id, name, description, icon_url, owner_id, color* e *blocked_message;*
- **application** - responsável pelo armazenamento das informações das aplicações cliente, através dos atributos: *id, name, description, icon_url, app_identifier, app_type, status, redirect_page, domain_id, user_registration_id* e *modification_log;*
- **user_domain** - compreende as relações e privilégios estabelecidos entre user e domain. Seus atributos são: *id, user_id, domain_id* e *privilege.*

O relacionamento entre as entidades está documentado no Modelo Relacional, disponível através do link: https://shield.valkyriatech.com.br/docs/. Os atributos *id* de cada entidade compreendem uma chave primária. Quanto aos atributos *user_id,*

owner_id, domain_id e *user_registration_id* representam as chaves estrangeiras das entidades.

4.2. Base e API

A estrutura da aplicação responsável pelo gerenciamento da base de dados e controle do sistema de arquivos do servidor utiliza arquitetura MVC (*Model-View-Controller*). O padrão MVC divide a aplicação em camadas: uma da interface do usuário denominada *View*, uma para manipulação lógica de dados chamada *Model*, e uma terceira camada de fluxo da aplicação chamada *Control*. (MASSARI, 2017).

A camada *View* representa classes e funcionalidades utilizadas no controle e visualização de recursos da plataforma ao usuário final. A exibição de determinadas interfaces, assim como as respostas JSON (*JavaScript Object Notation*) de requisições dos sistemas auxiliares são manipuladas e gerenciadas por esta camada.

Model é responsável pela manipulação de atributos na base de dados e pelo gerenciamento de arquivos e diretórios no servidor, respondendo às solicitações da camada *Control.*

Por último, a camada *Control* compreende as principais operações do sistema. Além do gerenciamento e validação de todos os dados lógicos, classes auxiliares para *upload* de arquivos, geração de *logs,* criptografia de dados sensíveis e criação de mensagens padronizadas para o usuário, tal camada possui uma das classes mais importantes do sistema, responsável por interpretar as requisições de todos os sistemas auxiliares: *API*.

A classe *API* é responsável pela interpretação, validação de dados e gerenciamento de resposta para as diferentes requisições provenientes dos sistemas auxiliares, em suas respectivas plataformas. Todas suas requisições são identificadas através do parâmetro *action,* sendo os itens a seguir ações válidas:

- ***status -*** verificação do status do servidor e da conexão do software com a base de dados;
- ***uploadImage*** e ***getParticlesJson -*** controle para upload de arquivos de imagem, utilizados na plataforma e JSON de estilização da página inicial;
- ***login, registerUser, getUser, editUser, deleteUserFromDomain, assignUserToDomain, getUsersFromDomain*** e ***confirmInvitation -*** ações relacionadas aos usuários da plataforma, gerindo a manipulação dos mesmos dentro do sistema e dos domínios ao qual pertencem, assim como o controle de acesso às funcionalidades da plataforma;
- ***registerDomain, editDomain, deleteDomain -*** ações relacionadas à manipulação dos atributos e informações dos domínios;
- ***registerApplication, editApplication, deleteApplication, getApplications, getAppCount, setStatus, verifyStatus, checkSite*** e ***inactiveApp -*** ações relacionadas à manipulação dos atributos e informações das aplicações cliente registradas. Controle do atributo *status,* responsável pelo bloqueio das aplicações e interface de erro.

4.3. Plugin

O fluxo da aplicação *plugin* é o mesmo em todas as tecnologias para as quais foi desenvolvida. De forma assíncrona ou antes da execução da aplicação cliente, uma requisição é feita à API através do parâmetro *verifyStatus.* O parâmetro *app_identifier*

também é enviado, contendo informações únicas relacionadas àquela aplicação para identificação na base de dados.

A resposta retornada pela API contém dois atributos relevantes:

- ***status*** **-** valor *boolean,* informando se a aplicação cliente deve prosseguir com sua execução ou não;
- ***redirect_page*** **-** atributo que pode ser manipulado pelo usuário na plataforma. Quando preenchido, informa ao *plugin* que um redirecionamento para uma página customizada deve acontecer.

4.4. Plataforma

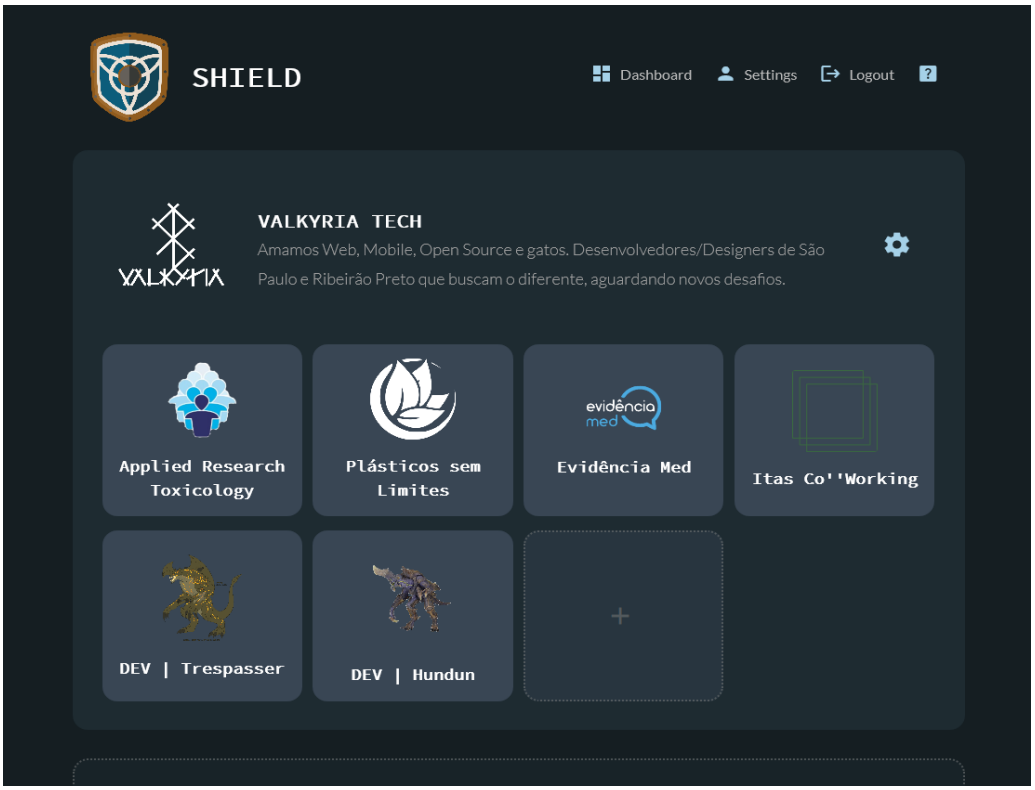

Figura 1. Captura de tela da plataforma, versão *web*

O conjunto de interfaces utilizadas para manipulação de todos os atributos relacionados aos usuários, domínios e aplicações do sistema é denominado 'Plataforma'. Foi desenvolvida com tecnologias distintas, para execução em dois ambientes diferentes: *web* (navegadores de internet) e *mobile* (dispositivos móveis que utilizam os sistemas operacionais Android e iOS).

Em ambas tecnologias e ambientes de execução, as funcionalidades da aplicação são as mesmas. Elas se diferem, no entanto, em sua estrutura e desenvolvimento.

A aplicação *web* foi desenvolvida em ReactJS. Sua estrutura, as funcionalidades existentes e seus processos utilizam a linguagem de programação *JavaScript* para enviar requisições e manipular os dados relacionados, disponibilizados nas respostas da API. Sua estrutura lógica compreende páginas (ou *screens)* em arquivos separados e importados na página inicial da aplicação, sendo o *script* **main.js** responsável por gerir as transições entre páginas. Os elementos visuais, no entanto, utilizam a linguagem de estilização *CSS3* para definições de aparência, cores, animações etc.

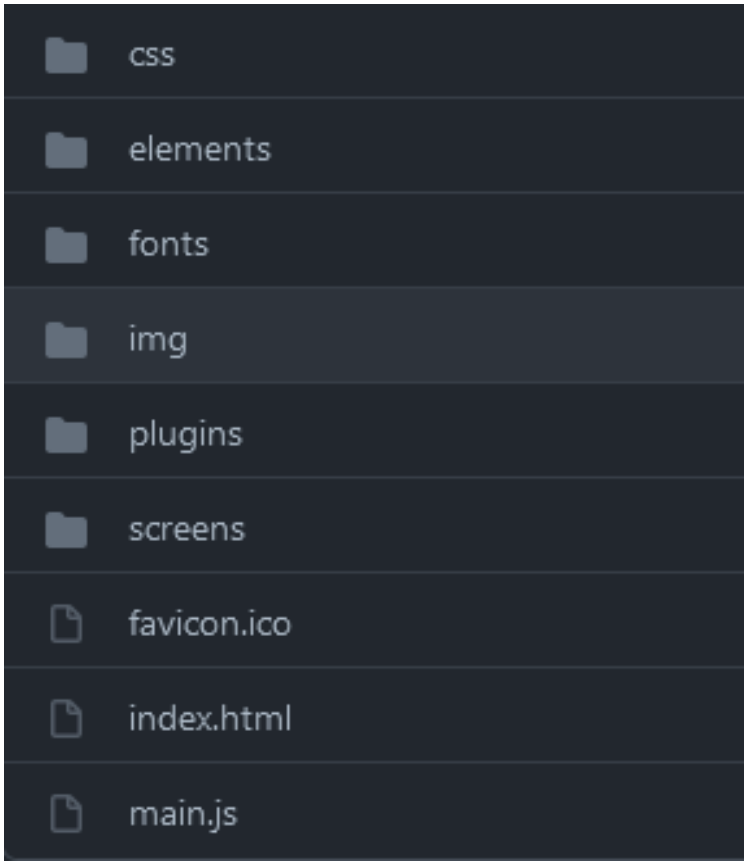

Figura 2. Estrutura da aplicação, versão *web*

A versão *mobile* compreende as mesmas funcionalidades. A linguagem de programação *DART* foi utilizada para requisição à API, manipulação de dados de resposta e controle de telas de visualização. A estrutura de componentes das telas e sua estilização gráfica foram desenvolvidas utilizando o *framework Flutter.*

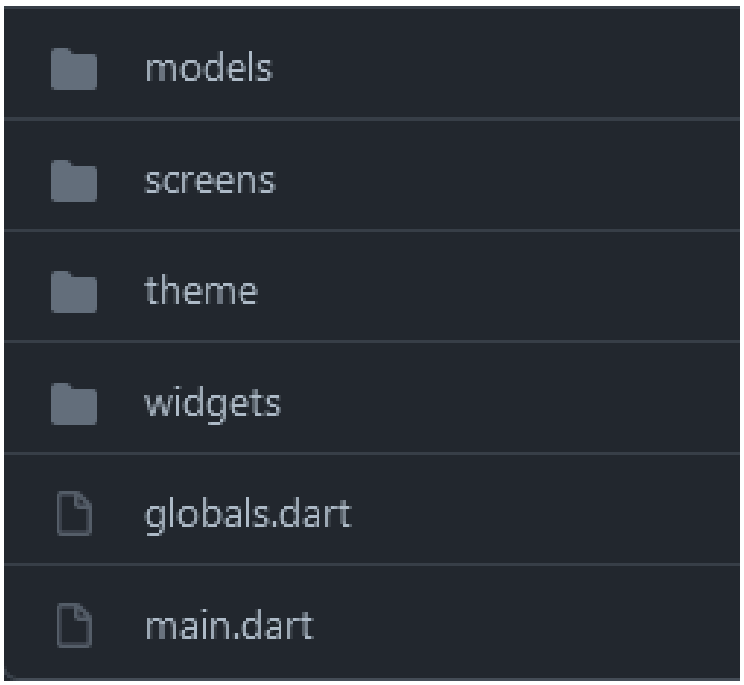

Figura 3. Estrutura da aplicação, versão *mobile*

5. Resultados

Todas as aplicações que compõem o sistema 'Shield' foram modeladas e desenvolvidas com foco em uma possível solução ao problema previamente apresentado. O sistema atende aos requisitos elicitados, gerindo o acesso e a execução de aplicações cliente sem necessidade de acesso ao ambiente final de instalação, independente da tecnologia utilizada no mesmo.

Figura 4. Captura de tela da aplicação alvo bloqueada

No momento da publicação deste trabalho, a versão *mobile* da plataforma, para os sistemas operacionais Android e iOS, encontra-se em desenvolvimento. Entretanto, todos os recursos do sistema já estão disponíveis em sua versão *web*, disponível em: https://shield.valkyriatech.com.br.

6. Referências

ANDREI, L. (2019) O Que é MySQL?, https://www.hostinger.com.br/tutoriais/o-que-e-mysql, Maio.

BALAN, K. (2019) Publicidade Digital Deve se Igualar à Tradicional até 2023, https://www.meioemensagem.com.br/home/midia/2019/10/08/publicidade-digital-deve-se-igualar-a-tradicional-ate-2023.html, Maio.

CETIC. (2018) Acesso à Internet por Banda Larga Volta a Crescer nos Domicílios brasileiros, https://www.cetic.br/noticia/acesso-a-internet-por-banda-larga-volta-a-crescer-nos-domicilios-brasileiros/, Maio.

COSTA, L. (2018) Como Fugir do Calote na Vida de Freela, https://www.vivendodefreela.com.br/como-fugir-do-calote-na-vida-de-freela/, Maio.

G1, EDITORIAL ECONOMIA. (2020) Inadimplência Cresce entre Brasileiros, https://g1.globo.com/economia/noticia/2020/02/14/apos-2-meses-de-queda-inadimplencia-cresce-em-janeiro-e-atinge-613-milhoes-de-brasileiros.ghtml, Maio.

HAMAGUCHI, A. (2013) Direito Autoral nas Redes Sociais: Como Proteger Fotos e

Arte Digital?, https://www.techtudo.com.br/artigos/noticia/2013/01/direito-autoral-nas-redes-sociais-como-proteger-fotos-e-arte-digital.html, Maio.

MASSARI, J. (2017) Padrão MVC | Arquitetura Model-View-Controller, https://www.portalgsti.com.br/2017/08/padrao-mvc-arquitetura-model-view-controller.html, Maio.

MIRANDA, O. O. (2019) O Contrato de Trabalho Intermitente e suas Implicações, https://ambitojuridico.com.br/cadernos/direito-do-trabalho/o-contrato-de-trabalho-intermitente-e-suas-implicacoes/, Maio.

PHP. (2021) O Que é o PHP?, https://www.php.net/manual/pt_BR/intro-whatis.php, Maio.

SANTANA, F. TABLELESS. (2019) Flutter: Porque Você Deveria Apostar Nesta Tecnologia, https://tableless.com.br/flutter-porque-investir-nessa-tecnologia/, Outubro.

SAROSINI, M. EXTRA. (2014) Profissional Leva Calote de Empresa e Lança Campanha 'Pague meu Freela', https://extra.globo.com/noticias/viral/profissional-leva-calote-de-empresa-lanca-campanha-pague-meu-freela-14253057.html, Maio