Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

DANILO COLAVITE DE ANDRADE

ROBÔS DE IMPRESSÃO PARA LOJA VIRTUAL: UMA BUSCA PELA HUMANIZAÇÃO EM PROCESSOS AUTOMATIZADOS.

Assis/SP
2021

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

DANILO COLAVITE DE ANDRADE

ROBÔS DE IMPRESSÃO PARA LOJA VIRTUAL: UMA BUSCA PELA HUMANIZAÇÃO EM PROCESSOS AUTOMATIZADOS.

Projeto de pesquisa apresentado ao curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Danilo Colavite de Andrade
Orientador(a): Esp. Célio Desiró.

Assis/SP
2021

FICHA CATALOGRÁFICA

ANDRADE, Danilo Colavite.
Robôs de impressão para Loja Virtual: Uma busca pela humanização em processos automatizados. Danilo Colavite de Andrade. Fundação Educacional do Município de Assis – FEMA – Assis, 2021.
55 páginas.

Palavras-chave: Robô; Impressão; Automação; Loja virtual.

CDD: 001.6
Biblioteca da FEMA

ROBÔS DE IMPRESSÃO PARA LOJA VIRTUAL: UMA BUSCA PELA HUMANIZAÇÃO EM PROCESSOS AUTOMATIZADOS.

DANILO COLAVITE DE ANDRADE

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Esp. Célio Desiró.

Examinador: Dr. Osmar Aparecido Machado.

Assis/SP

2021

LISTA DE ILUSTRAÇÕES

ÍNDICE

RESUMO

Este trabalho descreve a implementação de um robô virtual para impressões personalizadas que também busca, mesmo que de forma singela, uma maior humanização nos processos automatizados criando impressões artísticas pré definidas. Seu objetivo é, além de automatizar o processo de impressão personalizada, realizar uma maior aproximação, ainda que de forma profissional, entre o cliente e o lojista, já que na geração tecnológica em que estamos vivendo atualmente, torna-se cada vez mais comum a realização de trabalhos utilizando a internet. e, com isso, a comunicação direta entre as pessoas foi ficando cada vez mais escassa e, às vezes, desnecessária, visto que muita coisa pode ser resolvida on-line ou por meio de um robô virtual que faça o trabalho. Os métodos utilizados para sua realização foram artigos e livros científicos sobre automatização, ASP, SQL Server e Javascript, e seus resultados mostraram-se satisfatórios, visto que o robô foi programado com sucesso e conseguiu cumprir os objetivos almejados.

Palavras-chave: Robô; Impressão; Automação; Loja virtual.

ABSTRACT

This work describes the implementation of a virtual robot for personalized prints that also seeks, even if in a simple way, a greater humanization in the automated processes, creating pre-defined artistic prints. Its objective is, in addition to automating the personalized printing process, to bring the customer and the retailer closer, albeit professionally, since in the technological generation we are currently living in, it becomes increasingly common to carrying out work using the internet. and, as a result, direct communication between people became increasingly scarce and, at times, unnecessary, as a lot of things can be resolved online or through a virtual robot that does the work. The methods used for its realization were scientific articles and books on automation, ASP, SQL Server and Javascript, and their results were satisfactory, as the robot was successfully programmed and managed to fulfill the desired objectives.

Keywords: Robot; Print; Automation; Virtual store.

1. INTRODUÇÃO

O sistema ERP (*Enterprise Resource Planning* - Planejamento de Recursos Empresariais, em tradução literal) é um software integrado de gestão empresarial que objetiva reunir em uma única solução as informações gerenciais dos setores de uma empresa. Esses setores podem ser: finanças; contabilidade; patrimônio, dentre outros. O ERP integra as atividades, automatiza os processos e facilita a gestão empresarial, além de descartar a necessidade de haver vários programas e controles de departamentos paralelos.

"Os sistemas ERP (*Enterprise Resource Planning*), também conhecidos como Sistemas Integrados de Gestão, são uma das tecnologias mais utilizadas e discutidas na área de Sistemas de Informação, nos últimos anos." (SACCOL, 2004). A busca por esse software é grande por se tratar de um sistema prático, objetivo, organizado e financeiramente viável que ajuda a facilitar os processos de gestão utilizando apenas um banco de dados.

> "O ERP é um sistema integrado que possibilita um fluxo de informações único, contínuo e consistente por toda a empresa, sob uma única base de dados. É um instrumento para a melhoria de processos como a produção, compras ou distribuição, informações *on-line* e em tempo real. Em suma, o sistema permite visualizar as transações efetuadas pela empresa desenhando um amplo cenário de seus negócios." (PADILHA, 2004, p. 65).

Segundo PADILHA (2004), alguns dos motivos pelos quais várias empresas estão optando pelos pacotes ERP são por conta de frustrações com sistemas incompatíveis, departamentos de tecnologia de informação que não possibilitam a integração entre esses sistemas e outros fatores que influenciam diretamente na obtenção de maior competitividade. Em outras palavras, e como já dito acima, é um sistema de uso prático e objetivo que ajuda a facilitar o trabalho de quem o obtém e mais compreensível também.

A utilização de um robô virtual atuando no ERP torna as possibilidades de controle e monitoramento ainda mais ilimitadas, isso porque a margem de erro do robô é muito menor, praticamente inexistente, em comparação com a mesma operação

quando realizada por um humano. Trata-se da operação de automatização.

Utilizar essa tecnologia é um aspecto interessante e viável, em termos de qualidade, para as empresas. Uma empresa que agrega grande quantidade de clientes muitas vezes não tem a oportunidade de dar uma atenção mais qualificada a cada um deles por vários motivos, sendo um deles a falta de tempo. A tecnologia, que é muito utilizada atualmente e vem evoluindo cada vez mais, pode ser utilizada também em prol da humanização. Um algoritmo criado para automatização pode demonstrar, mesmo que em algo simples, o valor e importância, não somente financeira, de cada cliente à empresa, promovendo à relação do lojista e seus clientes a humanização.

1.1 OBJETIVOS

1.1.1 Objetivos gerais

O objetivo geral desta pesquisa acadêmica é o desenvolvimento de uma aplicação para automatizar o processo de impressões personalizadas com integração com a loja virtual Tray, visando a humanização da relação entre lojista e cliente.

1.1.2 Objetivos específicos

Este trabalho tem por objetivo realizar um estudo sobre as ferramentas ASP, SQL Server, Activex e Javascript. Além disso, este trabalho também pretende realizar um estudo sobre o processo de automatização de tarefas. Por fim, pretende implementar uma estratégia de impressão personalizada para criar o nome e sobrenome de cada cliente em, por exemplo, um desenho de um coração, para ser enviado aos respectivos clientes em seus aniversários ou datas comemorativas.

Com isso, o presente trabalho visa também a humanização de tais processos automatizados, fazendo com que os clientes se sintam importantes recebendo algo feito especialmente para eles, podendo transformar, dessa forma, a relação entre os clientes e o lojista, tornando-se uma relação mais humanizada ainda que utilizando robôs para isso.

1.2 MOTIVAÇÃO

Com o avanço da tecnologia, principalmente em relação às inteligências artificiais, os seres humanos estão cada vez mais se afastando uns dos outros e dando espaço aos serviços tecnológicos. Com isso fica mais evidente a falta de humanização em certos serviços, visto que os computadores e suas tecnologias fazem tudo com mais rapidez e automatização. Abaixo vê-se um exemplo do afastamento das pessoas por conta da tecnologia:

> "[...] há aproximadamente 8 anos faltou acesso à *internet* na Companhia de Planejamento do Distrito Federal, o problema ocorreu por aproximadamente um dia, foi notado que a falta do aplicativo MSN – *Messager* dentre outros demais aplicativos de comunicação fez em alguns setores, pessoas que trabalhavam a muito tempo no local e nunca tinham se conhecido pessoalmente, foi um relato incrível, pois ao mesmo tempo que a tecnologia aproxima ela também afasta as pessoas. Os trabalhadores da empresa se encontraram nos corredores e finalmente puderam ver com quem trabalhavam." (POLI, 2017).

A primeira motivação para a realização deste trabalho é a de implementar esse produto na empresa na qual trabalho, Web Managers, de forma que a automatização seja usada para ajudar no relacionamento entre os seres humanos e não para afastá-los como vem acontecendo ao longo dos anos à medida em que a tecnologia se expande.

Por esse motivo então, este trabalho visa aproximar as pessoas através da demonstração de atenção aos clientes para que se sintam importantes, para que reflitam que, atrás do trabalhador que são, vive um ser humano, e que merecem ser tratados como tal, não excluindo a parte tecnológica do processo, mas dando um toque humanizado no mesmo.

Há também a questão de tornar visível o lojista como um ser humano, como alguém que se importa com seus clientes e que busca atendê-los da melhor maneira possível, não os tratando apenas como um número, como variáveis que dão lucro à empresa, mas sim como os seres subjetivos que são.

1.3 METODOLOGIA DE PESQUISA

Será utilizado para a realização deste trabalho artigos e livros científicos sobre automatização, ASP, SQL Server, ActiveX e Javascript extraídos de sites de pesquisas acadêmicas e bibliotecas virtuais e, também, de livros disponíveis na biblioteca da universidade, juntamente com a utilização do computador para descrever a pesquisa.

Também será feito o reaproveitamento de código da parte de impressão do ERP WM10, para manter o padrão exigido pela Web Managers. Será utilizado o ASP para fazer a parte de manipulação de *string* (texto), a qual será responsável pelo desenho do texto no padrão escolhido no sistema.

O ActiveX será utilizado para fazer a conexão do navegador com o sistema operacional para que este faça comunicação com a impressora instalada no sistema, possibilitando a automatização do processo de impressão.

Já o Javascript e o HTML5 serão utilizados para moldar o *front-end* da aplicação, possibilitando, por exemplo, que a página seja carregada com o intervalo de tempo determinado pelo usuário, para que o robô fique sempre de prontidão e possa responder às requisições do usuário.

O banco de dados utilizado para guardar as informações será o SQL Server da Microsoft. Nele guardaremos as informações necessárias para realizar o processo de automatização da impressão, inclusive os padrões utilizados por ela.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma: o capítulo 1 compreende a introdução; o capítulo 2 irá descrever as ferramentas que serão utilizadas na implementação do projeto; o capítulo 3 descreverá a modelagem da ferramenta a ser implementada; o capítulo 4 irá descrever os detalhes da implementação da ferramenta, bem como sua avaliação; o capítulo 5 apresentará as conclusões.

2. PRINCIPAIS TECNOLOGIAS UTILIZADAS NO PROJETO

2.1 ASP

A linguagem de programação ASP (*Active Server Pages*) é uma linguagem que gera páginas HTML de forma dinâmica, *server-side scripts* e, também, *client- side scripts*. É possível rodar códigos que geram páginas HTML dinamicamente e mandá-las para o Browser a partir de um Windows NT Server com IIS3 ou superior (*Internet Information Server*) instalado, por exemplo. Com isso, quando uma página ASP for requisitada pelo Browser, ela será executada pelo servidor e, então, gerará uma página HTML a ser enviada ao browser que a requisitou. (FAETERJ-RIO - Faculdade de Educação Tecnológica do Rio de Janeiro, 2015, p. 1)

A linguagem ASP, além de consultar o banco de dados, serve também para enviar e receber correio eletrônico por meio de páginas HTML; para criação de rotinas de propaganda rotativa; para leitura de arquivos de texto; para identificação e autenticação de usuários, dentre muitas outras aplicações. Trata-se de uma linguagem de programação criada para complementar a linguagem HTML, trazendo à mesma todo o poder do acesso ao banco de dados; do acesso a arquivos de texto; do uso de variáveis; da captação de informações de formulário, dentre outros. (FAETERJ-RIO - Faculdade de Educação Tecnológica do Rio de Janeiro, 2015, p. 2).

ASP é uma junção de programação em VBScript e Objetos ActiveX, sendo a VBScript uma linguagem criada à partir do Visual Basic, porém com algumas limitações por motivos de segurança; e Objetos ActiveX são funções prontas já existentes no servidor, as quais captam os parâmetros de entrada dos dados, os manipula de acordo com a sua função e os envia para a saída. Em outras palavras, o ActiveX são bibliotecas que contém vários objetos reunidos, especialmente objetos que necessitam de instâncias para funcionar, o que significa que cada acesso a um banco de dados deve ocupar uma instância daquele objeto, por exemplo. (FAETERJ-RIO - Faculdade de Educação Tecnológica do Rio de Janeiro, 2015, p. 1).

Quando a linguagem ASP é usada somente em alguns “pedaços” de páginas HTML, existe um símbolo que diferencia o que é programação em HTML e, também, em ASP: **<% código asp %>**. Segue abaixo alguns exemplos de utilização da

linguagem ASP:

<head>

<title>página em HTML e ASP</title>

</head>

<html>

** Olá Mundo!!! **

<%

' (aspas simples significa observação dentro do código) ' a linha abaixo tem o mesmo efeito da linha acima response.write " Olá Mundo "

%>

</html>

I. **<html>**

<head><title>Como funciona o ASP?</title></head>

<body bgcolor="#FFFFFF">

** Ola Mundo!!! **

<%

response.write " Ola Mundo!!! "

%>

** <%="Ola Mundo!!!"%> **

</body></html>

Os códigos acima são exemplos tirados de uma apostila da Faculdade de Educação

Tecnológica do Rio de Janeiro – FAETERJ-RIO (2015), códigos esses que exibem a frase “Olá Mundo!!!” de três modos diferentes:

- O primeiro é feito utilizando-se apenas da programação HTML;
- O segundo, usando apenas comandos ASP; e
- O terceiro trata-se de uma mistura de ambos os códigos.

Como visto acima, não se faz necessário constituir uma página inteira utilizando apenas códigos ASP, mas é possível criar páginas onde se mesclem as duas programações. Na verdade, a única regra é que, a partir do momento em que determinado código da página possua ao menos uma linha em ASP, a terminação do nome da página deve ser feita na mesma linguagem, ou seja, .asp, pois somente dessa maneira o servidor poderá distinguir quais páginas devem ser executadas antes de enviar ao Browser.

Ainda utilizando exemplos da apostila citada acima, segue abaixo a simbologia e as convenções de uso da linguagem ASP:

<% -> início do trecho de código ASP

%> -> final do trecho de código ASP

' (aspas simples) -> usada antes de comentários dentro do código

= -> é usado no modo de programação por mesclagem de códigos HTML e ASP.

Além de trazer o poder do VBScript para a página HTML, a linguagem de programação ASP traz também o poder dos componentes ActiveX e dos Objetos. Os chamados Objetos tratam-se de add-ons que dão capacidade de captura e transmissão de variáveis entre as páginas; capacidade de captura de informações sobre o Browser do usuário sobre o servidor; de criação e manipulação de cookies; de consulta, alteração e adição de dados no banco de dados; de *sessions* e publicidade rotativa nos sites; de envio e recebimento de e-mails via página Web;

dentre muitas outras funções. A ASP possui uma estrutura bem elaborada já que ela pode ser implementada com os Objetos (FAETERJ-RIO - Faculdade de Educação Tecnológica do Rio de Janeiro, 2015, p. 2).

2.2 SQL Server

SQL Server é um SGBD (sistema gerenciador de Banco de Dados) criado pela Microsoft. Trata-se de um Banco de Dados vigoroso que é utilizado por sistemas corporativos de diversos portes. Foi lançado originalmente em 1988 em uma parceria com a empresa Sybase. Naquela época ele funcionava exclusivamente em equipamentos com o sistema operacional OS/2, característica essa que foi modificada em 1994 quando foi disponibilizada uma versão do mesmo para Windows. Isso ajudou a popularizar e a sedimentar esse SGBD no mercado. (IMPACTA, 2017).

Nos primórdios da criação dos bancos de dados, grandes empresas de tecnologia começaram a desenvolver diversos tipos de sistemas especializados nesse processo, sendo o SQL Server um dos gerenciadores mais robustos e seguros dessa modalidade. Ele é indicado também para profissionais que estão iniciando sua carreira em tecnologia da informação e que necessitam conhecer os recursos mais comuns da linguagem de programação SQL (*Structured Query Language*), que é a principal ferramenta para realizar consultas e operações em bancos de dados. Mas, de forma geral, o SQL Server é indicado para qualquer usuário ou organização que tenha a necessidade de guardar informações tanto para consulta quanto para seu uso posterior.

Existem algumas edições do SQL Server disponíveis, as quais variam para atender a diferentes públicos e plataformas. Elas variam especialmente em relação à carga de trabalho que suportam. Em outras palavras, desde pequenas aplicações locais à possibilidade de acesso a milhões de pessoas por meio da internet. Ele pode ser encontrado tanto em versões gratuitas quanto pagas, sendo, no último caso, mais barato do que outras versões similares de SGBD's conhecidos. Constam abaixo cinco edições da versão de 2017 do SQL Server:

I. Express: a edição mais básica do banco de dados, é gratuita e serve

principalmente para quem deseja ter seu primeiro contato com o produto. Indicada para pequenas empresas e desenvolvedores independentes;

II. Desenvolvedor: essa edição traz alguns recursos a mais que a anterior, incluindo as funcionalidades da versão Enterprise (a mais completa), mas é licenciado somente para o uso em ambiente de testes e desenvolvimento, sendo, por esse motivo, recomendado para empresas e profissionais que criam e testam aplicações mais robustas;

III. Web: é uma edição de baixo custo e se faz ideal para a hospedagem e o gerenciamento de dados em diversos sites;

IV. Standart: essa versão oferece básicos recursos de gerenciamento, ferramentas para uso local e na nuvem, e traz também funcionalidades referentes a *Business Intelligence* (BI);

V. Enterprise: se trata da versão mais completa do produto. Traz abrangentes recursos relacionados a gerenciamento do banco de dados e otimização do desempenho, sendo ideal para grandes corporações que necessitam gerenciar bancos de dados com milhões de registros e constantes acessos.

É preciso considerar a possibilidade da realização de um *upgrade* para uma edição superior sempre que necessário. Em relação à segurança, esse SGBD atua com sistemas de criptografia integrada, o que garante que tais dados sejam vistos e alterados somente por usuários que possuem autorização para isso. Por meio do uso de controle sobre os dados, o SQL Server impede que inconsistências que inviabilizem a utilização precisa de informações sejam geradas.

A partir da versão de 2008 do SQL Server foram adicionadas novas funcionalidades de auditoria de dados, os quais simplificaram ainda mais o trabalho dos DBA's (administrador de banco de dados), sendo elas a *Change Tracking* (CT), *Change Data Capture* (CDC) e SQL *Audit*, as quais listarei abaixo de acordo com a definição do site oficial da Microsoft.

I. Change Tracking (controle de alterações), ou CT, fornece um mecanismo de controle de alterações eficiente para aplicativos. Os desenvolvedores de aplicativos precisavam implementar mecanismos personalizados de controle de alteração para permitir que esses aplicativos consultassem as alterações nos dados

de um banco de dados e que acessassem as informações relacionadas às alterações. Trata-se de uma solução leve;

II. Change Data Capture (captura de dados alterados), ou CDC, contém registros que inserem, atualizam e excluem atividades aplicadas a uma tabela do SQL Server e disponibiliza também os detalhes das mudanças em um formato relacional que é facilmente utilizável. Tanto as informações de coluna quanto os metadados exigidos para a aplicação de alterações em um ambiente de destino são capturados para as linhas modificadoras e então, armazenados nas tabelas de alteração, tabelas essas que espelham a estrutura da coluna das tabelas originais rastreadas;

III. SQL Audit (auditoria do SQL) permite criar auditorias de servidor, as quais podem conter especificações de auditoria de servidor para eventos no nível de servidor e também especificações de auditoria de banco de dados para eventos no nível banco de dados. Os eventos auditados podem ser gravados tanto nos *logs* de eventos quanto nos arquivos de auditoria.

Esclarecendo um ponto importante, o processo de auditoria em banco de dados respalda-se em "verificar se tudo está de acordo com as regras estabelecidas na empresa, não se limitando a registrar o que está sendo manipulado ou processado, mas sim apresentar e comprovar o que foi feito, quando, como foi e quem fez" (SANTANA, 2015). A auditoria de uma instancia do Mecanismo de Banco de Dados do SQL Server ou de outro banco individual envolve principalmente o controle e o registro em *log* dos eventos que ocorrem no Mecanismo de Banco de Dados.

> "Com a não aplicação de uma auditoria nas bases de dados, muitas empresas ou setores estão sujeitos a erros de processo, fraudes, perca de dados, alteração de registros importantes sem a identificação do responsável pela alteração, entre outros. Todo processo que não estiver funcionando em seu fluxo correto tende a causar grandes prejuízos financeiros para a empresa" (SANTANA, 2015, p. 1).

2.3 JAVASCRIPT

A JavaScript é uma das linguagens mais antigas utilizadas na web, sendo uma linguagem de Script criada pela Netscape em 1995. Ela foi lançada junto com o navegador *Netscape Navigator* 2.0. Após o seu lançamento, as páginas na internet começaram a implementar um mínimo de dinamicidade por conta do modo com o

qual a linguagem acessa e manipula os componentes do seu hospedeiro, no caso o *Netscape Navigator*. Além disso, a JavaScript pode também ser embutida em diversos ambientes, não se limitando apenas aos navegadores de internet. (FRANK, 2004, p. 1).

> "Apesar da semelhança entre os nomes JavaScript e Java, a primeira linguagem não foi derivada da segunda, apesar de possuírem alguns pontos comuns. JavaScript, por ser uma linguagem interpretada, é mais flexível que o Java, dando a liberdade ao programador de não declarar uma variável antes de utilizá-la ou de adicionar novas propriedades e novos métodos a um objeto a qualquer momento. O JavaScript foi criado para que se pudesse inserir em um site certos efeitos típicos do Java, sem que fosse preciso se preocupar com programação propriamente dito e que se tornasse mais leve do que a aplicação de um arquivo Java. Existe a possibilidade de que as duas linguagens se comuniquem por meio da tecnologia". (FRANK, 2004, p. 1).

Existem dois tipos de linguagem de programação para quando trabalhamos com aplicações Web, as quais são: *Client Side* e *Server Side*. A *Client Side* é executada no lado do usuário, enquanto a *Server Side* é executada no lado do servidor. A linguagem *JavaScript* é uma linguagem do tipo *Client Side*, pois quem interpreta e gera os resultados é o computador do usuário. Essa linguagem consegue trabalhar com variáveis, alterar a aparência de elementos, gerar resultados, consegue interagir com praticamente todos os elementos de uma página HTML e tudo isso sem a necessidade do recarregamento contínuo da página. (SCHIMID, 2012).

É possível identificar a presença da *JavaScript* no código de uma página pela presença das *tags*: <SCRIPT> e </SCRIPT>. Os *scripts Client Side* da linguagem são incluídos nas páginas HTML das três formas listadas abaixo, todas sendo exemplos tirados do artigo de Frank (2004):

I. Colocando as instruções entre as *tags* <SCRIPT> e </SCRIPT>, como por exemplo:

```
<HTML>
 <HEAD>
 <TITLE>Título da Janela</TITLE>
 </HEAD>
 <BODY>
```

Linha em HTML
<SCRIPT>
document.write(“Linha em JavaScript”);
</SCRIPT>
Linha em HTML
</BODY>
</HTML>

A *tag* <SCRIPT> pode ser colocada dentro da *tag* <HEAD> e da *tag* <BODY>, ou entre ambas. De acordo com a regra de uso geral, coloca-se dentro da *tag*

<HEAD> as partes do *script* que devem ser inicializadas antes de serem acessadas, como funções, criação de variáveis, declaração de objetos, dentre outros; e dentro da *tag* <BODY> coloca-se as instruções do *script* que serão executadas no mesmo momento ou que farão chamadas aos elementos cuja inicialização foi dada previamente.

II. Inserindo manipuladores de eventos dentro de *tags* HTML específicas, como por exemplo:

<BODY onLoad=”umaFuncão();”>

Inserida na *tag* <BODY>, a propriedade *onLoad* é uma manipuladora de evento. No caso do exemplo acima, a mesma propriedade será ativada no momento em que o conteúdo inserido entre as *tags* <BODY> e </BODY> tiver sido carregado, e quando isso ocorrer, as instruções JavaScript declaradas entre as aspas serão executadas e a função **umaFuncão** será chamada.

III. Inserindo o código JavaScript dentro de um arquivo com extensão .JS e colocando seu nome com valor do atributo **SRC** da *tag* <SCRIPT>, como por exemplo:

<SCRIPT SRC=”scriptexterno.js;”></SCRIPT>

2.4 CSS

Conforme se deu o desenvolvimento e evolução da Web, o HTML (*Hypertext Markup Language)* foi muito utilizado de todas as formas possíveis, porém não demorou muito para que as pessoas percebessem que essa linguagem não funcionaria no futuro por ser muito limitada. “Em vez de tentar disponibilizar um documento monolítico aos navegadores web, fazia muito mais sentido dar aos navegadores os blocos de construção do conteúdo em si e, então, deixar que o navegador cuidasse de juntar tudo.” (LEWIS, 2017). Esse princípio citado pelo autor é conhecido como separação de interesses (*separation of concerns*), e é neste ponto que entra a CSS.

A linguagem de programação CSS (*Cascading Style Sheet*), assim como o XML e o HTML, é uma recomendação do W3C. Essa linguagem permite a separação do conteúdo dos documentos de sua apresentação. Juntamente com *scripts* aos elementos, é possível alterar o formato do documento, bem como sua interface com o usuário. A CSS projeta *layouts* suportados por todos os navegadores, mas sem o desperdício de tempo ao atualizar formatações de conteúdos, tornando a manutenção das páginas muito mais rápidas. (DATAMEC S.A., 2001, p. 5).

> “As CSS têm por finalidade devolver à marcação HTML/XML o propósito inicial da linguagem. A HTML foi criada para ser uma linguagem exclusivamente de marcação e estruturação de conteúdos. Isso significa que, segundo seus idealizadores, não cabe à HTML fornecer informações ao agente do usuário sobre a apresentação dos elementos. Por exemplo: cores de fontes, tamanhos de textos, posicionamentos e todo o aspecto visual de um documento não devem ser funções da HTML. Cabem às CSS todas as funções de apresentação de um documento, e essa é a sua finalidade maior.” (SILVA, 2008, p. 25).

É possível criar uma CSS em um arquivo separado e então, posteriormente, aplicar a mesma em todas as páginas do site. Dada a adequada utilização deste recurso, é possível centralizar praticamente toda a apresentação de um site em um ou mais arquivos. É aconselhável a utilização da extensão **.css**, mas não é obrigatório e pode funcionar com outra extensão (DATAMEC S.A., 2001, p. 6). Para utilizar uma CSS armazenada em um arquivo de texto separado deve-se usar o elemento <LINK>, como citado no exemplo abaixo extraído do artigo “CSS - Cascading Style

Sheets” da Datamec S.A. (2001):

```
<LINK REL=STYLESHEET
HREF=”http://www.myserver.com/mysheet.css” TYPE=”text/css”>
```

Nem sempre é necessário armazenar a linguagem em um arquivo separado. Os códigos podem ser colocados dentro de cada documento HTML, porém caso seja feito dessa forma, tais códigos afetarão somente o documento em que foram colocados. O elemento utilizado para adicionar a CSS a um documento é o <STYLE>. Eis um exemplo abaixo, no qual o atributo TYPE especifica o tipo MIME:

```
<HTML> <HEAD> </HEAD>
    <STYLE
    TYPE=”text/css”>
    Definições de estilo
    </STYLE>
    <BODY> </BODY> </HTML>
```

Além disso, é possível também importar a linguagem de outros arquivos. Contudo, para isso, deve utilizar o comando **@import** passando como parâmetro o caminho e o nome do arquivo. Alguns navegadores ainda não suportam este comando. É extremamente necessário o uso da *tag* <STYLE>, como no exemplo abaixo:

```
<STYLE TYPE=”text/css”>
        @import url (http://www.myserver.com/style.css) ;
    </STYLE>
```

Faz-se fundamental a utilização da CSS na construção de grandes sites. Sua adequada utilização pode contribuir em aspectos como velocidade no carregamento das páginas; personalização de sites; reutilização de código; existência de um padrão de cores e navegação para todo o site; administração

centralizada; dentre outros. Porém, mesmo com as inúmeras vantagens, a construção de um site através da CSS é uma tarefa difícil e requer uma administração centralizada. (DATAMEC S.A., 2001, p.9).

2.5 HTML5

Desde o início de seu desenvolvimento, a linguagem de programação HTML5 traz uma proposta moldada na interoperabilidade, ou seja, trata-se de uma linguagem independente de navegadores, plataformas e outros meios de acesso. Isso significa menor custo e torna possível criar apenas um código HTML (*Hypertext Markup Language*) que pode ser lido por vários meios, ao invés de diferentes versões para distintos dispositivos. (MENEZES, 2013, p. 2).

Para essa nova versão do HTML foi utilizada uma metodologia diferente que passa a se modular, e tanto o HTML quanto o CSS são divididos em módulos, ou seja, os navegadores não precisam esperar que todo padrão seja desenvolvido e testado para poder começar a implementar alguma funcionalidade. Trata-se de um imenso avanço em comparação às versões anteriores do HTML, nas quais era preciso que toda a especificação da versão tivesse sido concluída e testada até que algum novo recurso fosse disponibilizado (HTML ou CSS) e somente após isso era utilizada pelos navegadores e desenvolvedores, implicando assim uma grande perda de tempo até que novas funcionalidades chegassem ao usuário. Inclusive, muitas vezes quando essa nova funcionalidade era lançada, ela já estava desatualizada. (MENEZES, 2013, p. 2).

A linguagem HTML5 traz dois novos elementos, sendo esses vídeo e áudio, que permitem a reprodução de conteúdos multimídia sem que se faça necessário a utilização de *plugins*. A inserção desses elementos se dá da seguinte forma:

Áudio: <audio src=teste1.mp3”></audio>

Vídeo: **<video scr=teste2.mp4”></video>**

“O HTML5 introduz também o acesso a um novo elemento designado *Canvas*, que permite a renderização avançada de gráficos numa página HTML, sem que para isso seja necessário a utilização de *plugins”* (MENEZES, 2013, p. 2). Essa

característica define o *Canvas* como um *bitmap* de modo imediato (*immediate mode*) que fornece uma superfície dinâmica cuja manipulação seja feita pelo *JavaScript*. A *tag* de marcação deste elemento é: <canvas></canvas> e ele possui dois atributos denominados *width* e *height* que definem as dimensões do *Canvas* no HTML. Se caso o navegador não suportar a *tag*, nada será mostrado, nem mesmo a mensagem de erro, porém é possível colocar um texto para passar o *feedback* (*tag* essa que poderá ser manipulada pela JavaScript caso o navegador consiga suportá-la), como mostrado no exemplo abaixo extraído do artigo Menezes (2013):

```
<canvas width=”320” heigh=”240” id=”canvas”>
  <p>O seu navegador não suporta o canvas!</p>
  </canvas>
```

Contudo, o HTML5 não consegue implementar ações web sozinho por se tratar de uma linguagem que define conteúdo. Por esse motivo ela necessita do JavaScript e do CSS para complementar as aplicações quanto à interação e apresentação.

> “O HTML5 é uma tecnologia aberta, ou seja, não necessita de softwares para ser implementada, basta somente ter um browser. Junto com as tecnologias CSS e Javascript, o HTML5 é capaz de disponibilizar recursos para a implementação de aplicações mais dinâmicas e com melhor interatividade” (MENEZES, 2013, p. 5).

2.6 BOOTSTRAP

O Bootstrap é, atualmente, o principal *framework* CSS para a construção do *front-end* de aplicações Web. Desenvolvido pelos engenheiros do *twitter* Mark Otto e Jacob Thornton, o Bootstrap vinha com o intuito de aperfeiçoar o desenvolvimento de sua plataforma por meio da adoção de uma estrutura única, o que reduziria inconsistências entre as variadas formas de se codificar, as quais variam de um profissional para outro. Com respostas muito positivas dos resultados, os desenvolvedores resolveram lançá-lo no *GitHub* (site mais conhecido de hospedagem de projetos **git.**) como um *software* livre. Após seu lançamento, recebeu a contribuição de inúmeros desenvolvedores de vários países, tornando-se assim o *software* livre mais ativo do mundo. (MIGUEL et al, 2015).

O Bootstrap oferece uma grande variedade de *plugins* e possui integração com qualquer linguagem de programação. Um de seus recursos de mais destaque é o sistema de *grid* responsivo, o qual permite desenvolvermos páginas que se adaptam aos diferentes tamanhos de tela com facilidade. (TECHIO, 2016). Um resumo bem elaborado dele consta em seu próprio site oficial: “Bootstrap torna o desenvolvimento *front-end* mais rápido e fácil. Ele é feito para pessoas de todos os níveis de habilidade, dispositivos de todos os formatos e projetos de qualquer tamanho.”

Com o Bootstrap não é necessário perder tempo digitando uma linha completa de CSS novamente, pois ele possui vários *plugins* em JavaScript (*jQuery*) que facilitam o desenvolvimento. Sua aplicação na prática é a criação de sites responsivos, ou seja, *mobile*. Essa ferramenta ajuda o desenvolvedor a implementar recursos como o *modal*; *carousel*; *slideshow*; *menu dropdown*; dentre outros, os quais são aplicados com mais facilidade. Assim como consta no seu site oficial, a ferramenta “usa CSS tradicional, mas seu código fonte utiliza os dois pré-processadores CSS mais populares, *Less* e *Sass*”.

> “Hoje o bootstrap não é apenas um *framework* com *design* responsivo eficaz, mas oferece todos os tipos de opções de funcionalidade e estilo. Seus arquivos CSS e JavaScript podem ser incluídos em um projeto para ajudar na criação de elementos como por exemplo *pop ups*, menus e slide shows.” (MIGUEL et al, 2015).

O Bootstrap recebe contribuições diárias, além de ser construído para funcionar na versão mais atualizada dos desktops e navegadores portáteis. Nas versões antigas, alguns elementos podem ser renderizados de diferentes formas. Com as amplas características próprias dos dispositivos móveis atuais, o Bootstrap é projetado para adaptar-se às telas em tais dispositivos, fornecendo assim tanto uma vasta biblioteca de componentes que permitem aplicações e desenvolvimento nas linguagens, como também nas técnicas de Web design disponíveis. (MIGUEL et al, 2015).

2.7 COMMERCE TRAY

A Commerce Tray é uma plataforma de E-commerce que auxilia lojistas a criarem

sua própria loja virtual. Como consta em seu próprio site oficial, a Tray proporciona uma plataforma de loja virtual intuitiva e adaptável para que seus clientes tenham uma visão completa de seu negócio, podendo, com isso, tomar decisões mais rápidas e focar no que realmente importa. A Tray cuida da infraestrutura da loja e ajuda o lojista a crescer profissionalmente. (TRAY - Site oficial, 2020).

> "A plataforma de e-commerce é uma das principais peças de uma loja virtual. Ele é o sistema responsável pelo gerenciamento – *back end*, e visualização da loja, o *front end*. É o sistema que nos possibilita criar a loja virtual e também gerenciá-la, incluindo produtos, gerenciando estoques, preços e estoques, além de outras funções que fazem parte do dia a dia de um comércio eletrônico." (GUIA DE E-COMMERCE. 2019).

A Tray foi criada há dezessete anos e conta com mais de duzentos recursos para seus clientes. Ela possui vários planos a serem contratados, os quais são: Start; Light; Basic; Plus; Pro e Exclusive, indo do mais básico ao mais *Premium*. Seus valores vão de R$49,00 a R$749,00 reais mensais, onde cada plano se diferencia pelos recursos disponibilizados. (TRAY - Site oficial, 2020).

A plataforma conta também com um suporte especializado via web e telefone e com o *Opencode*, o qual possui uma loja de temas exclusiva com várias opções de *layout* onde o cliente possui total acesso à gestão do *front end* da página para deixar a loja como bem entender, seja como desenvolvedor próprio ou em parceria com agências. A Tray também possui o *Easy Checkout* 2.0 que evita que o cliente precise passar por várias etapas para efetuar sua compra. Este último recurso citado é totalmente grátis para todos os lojistas da Tray, é 100% responsivo em computadores, celulares e tablets e aumenta em até 46% nas conversões da loja virtual. (TRAY - Site oficial, 2020).

Por meio da Commerce Tray é possível também que o lojista gerencie seu negócio pelo celular com o aplicativo e, além disso, é possível que ele integre qualquer recurso à plataforma através da APIs (*Application Programming Interface*) Públicas, fortalecendo ainda mais sua loja virtual. De acordo com o site oficial da Tray, os principais *marketplaces* do Brasil são integrados com a mesma.

2.8 WEBHOOK

Webhook é um padrão de HTTP (*Hypertext Transfer Protocol*) que fornece um

modelo pub/sub simples para ligar APIs Web e serviços SaaS (*Software as a service*). Recebeu esse nome por funcionar como um gancho (*hook*) de *software* na web. (MICROSOFT - Site oficial, 2012).

Ele permite que programas simples e independentes sejam criados com o objetivo específico de funcionar em cadeia conforme outros comandos são executados. Em outras palavras, é um recurso que envia informações em tempo real, tornando, desta maneira, o desenvolvimento de sistemas mais fácil. Em muitos casos, os Webhooks são também uma maneira mais simplificada de receber um alerta quando algo acontece em outra ferramenta, bem como são usados para a comunicação entre sistemas. (NARDE, 2018).

O que os difere das APIs comuns é que os Webhook entregam dados a outros aplicativos na velocidade em que eles acontecem, ou seja, imediatamente, enquanto nas APIs comuns é necessário pesquisar informações frequentemente para se manter atualizado. As requisições geradas a partir dos Webhooks são efetuadas com o método POST, onde o conteúdo no *body* deve estar no formato JSON, como no exemplo abaixo, de Narde (2018):

Content-Type: application/json; charset=UTF-8 User-Agent: Vindi-Hookshot/1.0

A plataforma espera que sua aplicação seja correspondente ao código HTTP 2XX (200, 201, etc) em, no máximo, vinte segundos. Como explica Narde (2018) “códigos de redirecionamento (3XX) não serão seguidos e serão considerados como falha”. O Webhook é um padrão que varia da maneira como é usado dependendo do serviço, mas sua ideia de base permanece a mesma. Como é descrito no site oficial da *Microsoft*, normalmente a solicitação HTTP POST contém um objeto JSON ou dados de formulário HTML, os quais são determinados pelo remetente do Webhook e inclui, ainda, informações sobre o evento causador de o Webhook ser disparado.

Utilizando o exemplo também do site da *Microsoft*, um corpo de solicitação de postagem de Webhook do *GitHub*, tem a aparência demonstrada abaixo como resultado de uma nova questão a ser aberta em um repositório específico:

```
{
  "action": "opened",
  "issue": {
     "url": "https://api.github.com/repos/octocat/Hello-World/issues/1347",
    "number": 1347,
     ...
```

```
 },
 "repository": {
    "id": 1296269,
   "full_name": "octocat/Hello-World",
   "owner": {
      "login": "octocat",
      "id": 1
       ...
   },
   ...
},
"sender": {
   "login": "octocat",
   "id": 1,
   ...
}
}
```


A solicitação POST é protegida de alguma forma e verificada logo em seguida pelo receptor, isso para garantir que o Webhook seja realmente do remetente pretendido.

2.9 JQUERY

A jQuery é uma biblioteca JavaScript criada por John Resig, um desenvolvedor americano que teve como objetivo simplificar a escrita da JavaScript. De acordo com sua própria definição, o foco da jQuery é a simplicidade para que não haja a necessidade de desenvolvedores escreverem longos e complexos códigos visando criar simples efeitos. Essa nova criação permite fazer com menos linhas um código que seria mais complexo com a JavaScript pura.

Entre os principais recursos da jQuery encontra-se a manipulação do DOM (*Document Object Model*), que é simplificado por suas funções e seletores. Ela

pode ser resumida a um arquivo JavaScript gravado com a extensão **.js** que detém a função de simplificar a sintaxe JavaScript. Segue abaixo alguns exemplos de simplificação tirados do artigo de Silva (2020):

I. Sintaxe JavaScript: **document.getElementsByTagName("p")**

Sintaxe jQuery: **$("p")**

II. Sintaxe JavaScript: **document.getElementById("um").setAttribute("class", "cor")**

Sintaxe jQuery: **$("#um").attr("class", "cor")**

III. Sintaxe JavaScript: **document.getElementById("um").setAttribute("class", "cor")**

Sintaxe jQuery: **$("#um").attr("class", "cor")**

A biblioteca da jQuery não requer instalação por se tratar de um arquivo JavaScript formal e, atualmente, encontra-se na versão 1.26, sendo fornecida em três formatos: **jquery-1.2.6.js**, relativo a um arquivo JavaScript comentado que contém espaçamento entre as linhas do código, criado com a finalidade de poder ser facilmente lido e entendido por qualquer pessoa objetivando estudá-lo por qualquer motivo; **jquery-1.2.6.min.js**, trata-se do mesmo arquivo descrito acima, porém uma versão onde retiraram-se todos os espaçamentos do código. Uma versão compactada em relação à anterior, recomendada para o uso em desenvolvimento e hospedagem para o site no ar; **jquery-1.2.6.pack.js**, relativo a um arquivo do JavaScript compactado com uma ferramenta de compressão de código JavaScript que mesmo sendo menor que o da versão mini, seu tempo de carregamento é praticamente igual, pois deve-se computar o tempo de descompressão quando o usuário recebe a página. Ela possui algumas desvantagens em relação às versões anteriores, sendo uma delas que essa versão não vai para o cache, portanto necessita ser carregada sempre que o usuário volta ao site. Outra desvantagem é que seu processo de descompressão pode acabar sendo imperfeito, introduzindo, com isso, bugs inexistentes. (SILVA, 2020).

Dentre as principais vantagens do uso da jQuery ao invés da JavaScript tradicional encontram-se: manipulação de conteúdos, sem limitações, com poucas

linhas de código; possibilidade de inserir uma grande variedade de efeitos de animação com uma linha de código simples; simplificação na criação de scripts; acesso direto a qualquer componente do DOM; emprego *cross-browser*; uso simplificado com AJAX e linguagens de programação como PHP e ASP; suporte para toda a gama de eventos de interação com o usuário sem limitações impostas pelos navegadores; dentre outros. (SILVA, 2020).

2.10 AJAX

Ajax (*Asynchronous JavaScript and XML*) é uma tecnologia utilizada para acessar páginas *web services* por meio da JavaScript. Ela torna os aplicativos muito mais dinâmicos e com maior capacidade de resposta, utilizando a JavaScript, XML e HTML dinamicamente. A Ajax não é uma nova linguagem de programação, mas sim uma nova forma de utilização das linguagens já existentes. (MEDEIROS, 2019?).

> “No Ajax o Javascript faz uma solicitação ao servidores, no servidor nada muda com a utilização de Ajax, ele continua respondendo cada solicitação exatamente como fazia antigamente quando não se usava Ajax. Porém a resposta do servidor agora retornará apenas os dados que a página precisa sem qualquer marcação ou apresentação. Uma característica que será notada pelo usuário que está acessando a página que está utilizando Ajax é que grande parte da página não será alterada, mas sim apenas partes que necessitarão de atualização. Antigamente a página inteira era carrega, porém com Ajax esse paradigma muda.” (MEDEIROS, 2019?)

Para iniciar a implementação do Ajax deve-se criar, primeiramente, uma função JavaScript que cria um objeto para fazer solicitações ao servidor. Segue abaixo um exemplo de uma criação da função a ser incorporada na página HTML de onde partirá a solicitação, retirada do artigo de Medeiros (2019?):

```
<head>

  <script language="javascript"
```

```
type="text/javascript"> var request = null;
function
  createRequest() { try {
    request = new XMLHttpRequest();
  } catch
    (trymicrosoft) { try
    {
      request = new ActiveXObject("Msxml2.XMLHTTP");
    } catch
      (othermicrosoft) { try
      {
        request = new ActiveXObject("Microsoft.XMLHTTP");
      } catch
        (failed) {
        request =
        null;
      }
    }

  }


  if (request == null)
    alert("Error creating request object!");
}
</script>
</head>
```

No início é declarada a variável *request* como nula e sua responsabilidade será a de armazenar o objeto de solicitação. Na atribuição ao objeto *request* no comando **request = new XMLHttpRequest()** há a tentativa da criação do objeto de solicitação. O tipo do objeto solicitação é **XMLHttpRequest**. É observável tanto no comando **request = new ActiveXObject("Msxml2.XMLHTTP")** quanto no

comando **request = new ActiveXObject("Microsoft.XMLHTTP")** que há também a tentativa de criar o objeto solicitação, porém de forma que funcione no Internet Explorer também. Se ocorrer um erro, o comando **request = null** dentro do **catch (failed)**, será executado, e isso garantirá que a variável da solicitação continue nula, que, se caso ocorrer, chegará uma mensagem no *browser* do usuário por meio do comando **alert**.

Tendo o objeto de solicitação, é possível seguir para a próxima etapa, a qual se trata da função JavaScript responsável por solicitar um dado em especial ao servidor, como exemplificado abaixo:

```
<head>
  <script language="javascript"
   type="text/javascript"> function
   getInformacaoQualquer() {
     createRequest();
     var url =
     "getInformacaoQualquerDoServidor-ajax.php";
     request.open("GET", url, true);
     request.onreadystatechange = atualizaPagina;
     request.send(null);
  }
  </script>
</head>
```

Primeiramente chama-se a função **createRequest()** para criar um novo objeto de solicitação. Após isso, tem-se a configuração da URL com o comando **var url = "getInformacaoQualquerDoServidor-ajax.php**, que será necessário para se conectar e obter os dados necessários. O comando **request.open("GET", url, true);** possui a inicialização da conexão e, também, a informação ao objeto de solicitação de como se conectar no servidor. No comando **request.onreadystatechange = atualizaPagina;** vem a configuração dos valores na página e, por último, **request.send(null);** apenas expõe que nenhum dado está sendo enviado na solicitação. Neste caso, o *script* apenas retornará o

resultado esperado sem precisar de nenhum dado. Após isso, chama-se a função criada a partir do HTML, que pode ser feito, por exemplo, utilizando um botão HTML, como consta no exemplo abaixo:

```
<form method="GET">
  <input value="Mostre a informação"
        type="button"
        onClick="getInformacaoQualquer();" />
 </form>
```

Um manipulador de eventos *onClick* do botão *button* foi utilizado para chamar a função JavaScript anteriormente, a qual foi criada para fazer a solicitação ao servidor. Feito isso, a última etapa seria atualizar a página web com as novas informações vindas do servidor. Por esse motivo a função **atualizaPagina** deverá ser configurada para realizar o procedimento. Como explica Medeiros (2019?), o procedimento **request.onreadystatechange = atualizaPagina;** indica que assim que o navegador receber a resposta do servidor, chamará a função **atualizaPagina**, e isso acontece porque a comunicação com o servidor em aplicativos Ajax é assíncrona, portanto não se sabe quando a resposta chegará. Com isso, cabe então ao navegador se responsabilizar por isso já que é ele quem sabe quando a resposta chegará do servidor, e então o desenvolvedor deve manipular essa resposta que veio do servidor, como mostra-se no exemplo abaixo:

```
function atualizaPagina() {
   if (request.readyState == 4) {
     var respostaDoServidor = request.responseText;
   }
 }
```

Essa função exemplifica o uso da propriedade **responseText** do objeto de solicitação, a qual contém a resposta que o servidor retornou. Com o comando **var respostaDoServidor = request.responseText**, captura-se a resposta do servidor e a coloca em uma variável qualquer. Detendo a resposta do servidor, basta apenas atribuir a mesma para algum lugar da página.

Dentre as vantagens de se utilizar a Ajax encontra-se o aumento da velocidade de

um site; flexibilidade para escolha da linguagem de programação do lado do servidor; melhor uso por parte do usuário; dentre outros. Também há as desvantagens, dentre elas dificuldade na manutenção do site, pois ao utilizar muitos Ajax, o código pode acabar ficando poluído com diversas solicitações e diferentes tratamentos; botões de avançar e retornar do navegador que não voltará ao conteúdo anterior; dentre outros. (MEDEIROS, 2019?).

2.11 JSON

JSON (*JavaScript Object Notation*) trata-se de um modelo de armazenamento e transmissão de informações no formato texto. É um modelo simples, porém muito utilizado por aplicações web por conter a capacidade de estruturar informações de forma mais compacta do que a XML consegue, o que torna o *parsing* dessas informações mais rápido. Empresas como Yahoo e Google adotaram o JSON, visto que suas aplicações precisam transmitir grandes volumes de dados. (GONÇALVES, 2012).

A ideia utilizada pelo JSON para representar informações é a seguinte: atribui- se, para cada valor representado, um nome/rótulo que descreve o seu significado. Trata-se de uma sintaxe derivada da forma utilizada pelo JavaScript de representar informações. Utilizando exemplos do artigo de Gonçalves (2012), para representar o ano de 2012 usa-se a seguinte sintaxe:

“ano”: 2012

Um par de nome e valor deve ser representado pelo nome entre aspas duplas seguido de dois pontos e com o valor logo em seguida. Os valores podem possuir apenas três tipos básicos, sendo eles o numérico (inteiro ou real), *booleano* e *string*. Eis abaixo quatro exemplos destes:

I. Representando um número real: **“altura”: 1.72**

II. Representando uma *string*: **“site”: “www.devmedia.com.br”**

III. Representando um número negativo: **“temperatura”: -2**

IV. Representando um valor booleano: **“casado”: true**

A partir dos básicos torna-se possível construir os mais complexos como *array* e objeto, sendo os *aray* delimitados por colchetes e com seus elementos separados entre as vírgulas, como nos exemplos abaixo:

I. Array de Strings: **[“RJ”, “SP”, “MG”, “ES”]**

II. Matriz de Inteiros:

```
[
    [1,5],
    [-1,9],
    [1000,0]
]
```

A palavra-chave *null* deve ser utilizada para representar valores nulos, como por exemplo:

“site”:null

Pode-se dizer que o JSON é visto como um concorrente da XML no quesito “troca de informações”. Algumas de suas principais semelhanças são que ambos representam informações no formato texto; são capazes de representar informação complexa, difícil de representar no formato tabular; podem ser usados para transportar informações em aplicações AJAX; são independentes de linguagens; possuem natureza auto descritiva; dentre outros. Mas também há suas diferenças, dentre as quais se encontram que o JSON não permite a execução de instruções de processamento, algo possível em XML; JSON, como dito anteriormente, representa informações de forma mais compacta; JSON é destinado para a troca de informações, enquanto o XML possui mais aplicações; dentre outros. (GONÇALVES, 2012).

2.12 API REST

A API Rest (*Representational State Transfer*) é um modelo a ser utilizado para projetar arquiteturas de *software* distribuído e baseia-se em comunicação via rede. Foi descrito por um dos principais criadores do protocolo HTTP, Roy Fielding, em

sua tese de doutorado, e vários desenvolvedores perceberam que esse modelo poderia ser usado para a implementação de web *services*, objetivando integrar aplicações pela web, além de utilizá-lo como uma alternativa ao SOAP (*Simple Object Access Protocol*). (FERREIRA, 2017).

“REST na verdade pode ser considerado como um conjunto de princípios, que quando aplicados de maneira correta em uma aplicação, a beneficia com a arquitetura e padrões da própria Web.” (FERREIRA, 2017). Ele foi adotado como modelo a ser utilizado na evolução da arquitetura do protocolo HTTP e consiste em alguns princípios e regras que permitem a criação de um projeto com interfaces bem definidas quando seguidas corretamente.

> “REST é um conceito arquitetural muito complexo, mas que no fim visa tirar vantagem de todas as características do protocolo HTTP, que é um protocolo de transporte. O JSON é somente uma forma de representar informações que precisam ser transportadas de um lado para outro. Sendo assim, podemos utilizar o protocolo HTTP para fazer o transporte de dados entre um cliente e um servidor.” (CAMPOMORI, 2017)

Para utilizar o protocolo HTTP da forma correta, pode-se adotar uma arquitetura baseada em Rest, e para fazer uma representação das informações que necessitam de transporte por meio do protocolo HTTP em uma arquitetura Rest, é possível utilizar o Json. (CAMPOMORI, 2017).

3. MODELAGEM DA FERRAMENTA

O processo de automação se iniciará com a instalação do aplicativo na loja virtual Tray feita pelo cliente. Após instalar o aplicativo e dada as devidas permissões, a Tray redirecionará o cliente para a *landpage* da aplicação onde será apresentada a ferramenta a ele e, então, o encaminhará para a página da aplicação, onde ele deverá realizar um cadastro. Feito o cadastro, o processo de automação se iniciará e o cliente já estará habilitado a usar o sistema de impressão.

O processo de automação será realizado graças a implementação do Webhook, que ficará sempre esperando um pedido ser efetuado na loja virtual tray, apontando para uma página em ASP que receberá as notificações do Webhook da Tray. Depois que a página receber a notificação do Webhook, executará a *procedure* no banco de dados, o qual cuidará do processo de listagem dos pedidos da loja virtual, pedidos esses que serão capturados através da API Rest da Tray, recebendo um JSON com as informações desejadas para que sejam preparadas da maneira almejada para o processo de impressão.

A vantagem de se utilizar o Webhook é que não será necessário ficar realizando requisições na loja virtual para saber se há novos pedidos constantemente, pois o Webhook cuidará dessa parte do processo, avisando a aplicação somente quando for realizado algum cadastro, atualização dos pedidos ou clientes na loja virtual Tray. Haverá alguns modelos padrões de impressão que o cliente poderá escolher, possibilitando a automação do processo, porém haverá também a possibilidade de criar uma impressão personalizada. A impressão será feita automaticamente pela impressora instalada no sistema operacional do computador do cliente.

Para esse projeto será desenvolvida uma aplicação na loja virtual Tray onde o cliente poderá instalar tal aplicação na sua própria loja virtual. Depois de instalado e dado as devidas permissões, o cliente será redirecionado para uma *landpage* que será desenvolvida para a aplicação. A *landpage* conterá informações sobre o produto a ser oferecido, neste caso o robô de impressão, e redirecionará para a página da aplicação onde o usuário criará uma conta para poder utilizar a

aplicação.

Durante a confirmação do cadastro feita por meio do e-mail utilizado, o Webhook já estará em funcionamento, pois a permissão de acesso aos dados foi concedida pelo usuário. O Webhook responderá a cadastros de pedidos da loja virtual do cliente. Assim que o cadastro de um novo pedido for efetuado, o Webhook notificará a aplicação através de uma página desenvolvida para atendê-lo, página essa onde será feita a chamada da *procedure* do banco de dados que cuidará de todo o processo de conexão com a loja virtual, a qual será feita por intermédio da API da Tray. Após a conexão por meio da API, uma variável guardará o retorno que contém um JSON com todos os detalhes do pedido, o qual será salvo em uma tabela do banco de dados.

Ainda na *procedure* será feita a chamada da impressão com o modelo escolhido pelo cliente na aplicação. A impressão será realizada na impressora de bobina padrão do sistema. Para que o robô de impressão funcione corretamente o usuário poderá escolher modelos de impressão montados previamente no site da aplicação ou poderá criar, também, modelos personalizados de impressão selecionando variáveis presentes na aplicação, modelos esses que serão utilizados pelo processo de automação. Ainda na *procedure* será salvo em uma tabela do banco de dados os dados do cliente que efetuou o pedido para que o usuário tenha acesso a dados como: nome; data de nascimento; e-mail; dentre outros, para que ele possa mandar impressões personalizadas exclusivamente para esse cliente, deixando menos fria e mecânica a interação entre cliente e lojista.

Encontra-se abaixo a forma como será feita a modelagem:

Figura 1: Webhooks
Fonte: Blog Vindi

Figura 2: Commerce Tray
Fonte: Edm2 Marketing

Figura 3: Landing Page
Fonte: Ideal Marketing

Figura 4: Banco de Dados
Fonte: Icon Ninja

MODELAGEM

Figura 5: Seta

Fonte: Flaticon

Figura 6: Impressora de bobina

Fonte: Anúncios na Bahia/OLX

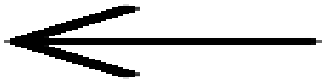

Figura 7: Aplicação

Fonte: Città Telecom

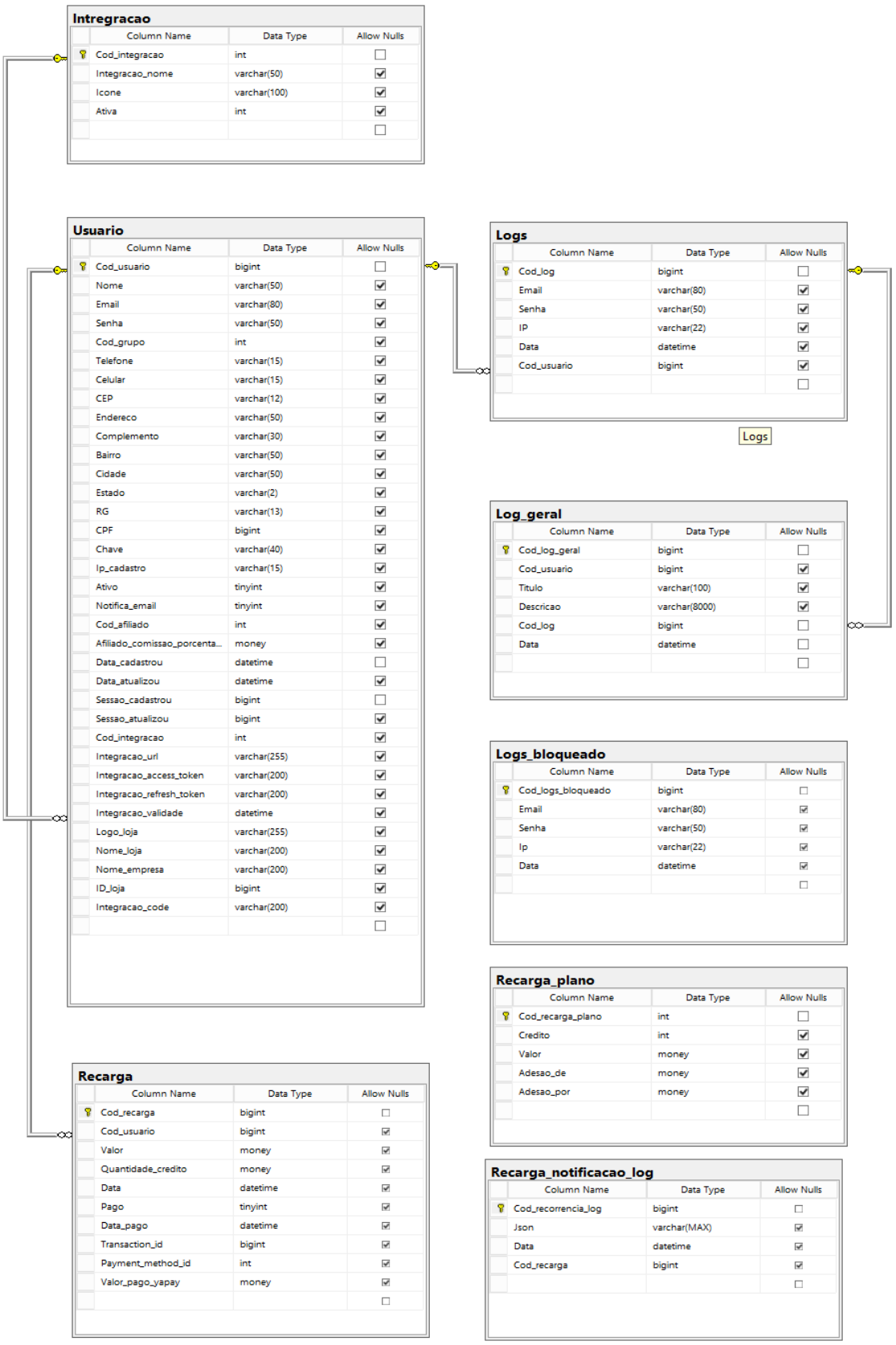
Intregracao
Column Name | Data Type | Allow Nulls
Cod_integracao | int
Integracao_nome | varchar(50)
Icone | varchar(100)
Ativa | int
Usuario
Column Name | Data Type | Allow Nulls
Cod_usuario | bigint
Nome | varchar(50)
Email | varchar(80)
Senha | varchar(50)
Cod_grupo | int
Telefone | varchar(15)
Celular | varchar(15)
CEP | varchar(12)
Endereco | varchar(50)
Complemento | varchar(30)
Bairro | varchar(50)
Cidade | varchar(50)
Estado | varchar(2)
RG | varchar(13)
CPF | bigint
Chave | varchar(40)
Ip_cadastro | varchar(15)
Ativo | tinyint
Notifica_email | tinyint
Cod_afiliado | int
Afiliado_comissao_porcenta... | money
Data_cadastrou | datetime
Data_atualizou | datetime
Sessao_cadastrou | bigint
Sessao_atualizou | bigint
Cod_integracao | int
Integracao_url | varchar(255)
Integracao_access_token | varchar(200)
Integracao_refresh_token | varchar(200)
Integracao_validade | datetime
Logo_loja | varchar(255)
Nome_loja | varchar(200)
Nome_empresa | varchar(200)
ID_loja | bigint
Integracao_code | varchar(200)
Logs
Column Name | Data Type | Allow Nulls
Cod_log | bigint
Email | varchar(80)
Senha | varchar(50)
IP | varchar(22)
Data | datetime
Cod_usuario | bigint
Logs
Log_geral
Column Name | Data Type | Allow Nulls
Cod_log_geral | bigint
Cod_usuario | bigint
Titulo | varchar(100)
Descricao | varchar(8000)
Cod_log | bigint
Data | datetime
Logs_bloqueado
Column Name | Data Type | Allow Nulls
Cod_logs_bloqueado | bigint
Email | varchar(80)
Senha | varchar(50)
Ip | varchar(22)
Data | datetime
Recarga_plano
Column Name | Data Type | Allow Nulls
Cod_recarga_plano | int
Credito | int
Valor | money
Adesao_de | money
Adesao_por | money
Recarga
Column Name | Data Type | Allow Nulls
Cod_recarga | bigint
Cod_usuario | bigint
Valor | money
Quantidade_credito | money
Data | datetime
Pago | tinyint
Data_pago | datetime
Transaction_id | bigint
Payment_method_id | int
Valor_pago_yapay | money
Recarga_notificacao_log
Column Name | Data Type | Allow Nulls
Cod_recorrencia_log | bigint
Json | varchar(MAX)
Data | datetime
Cod_recarga | bigint

Figura 8: MER

Figura 9: Diagrama de caso de uso

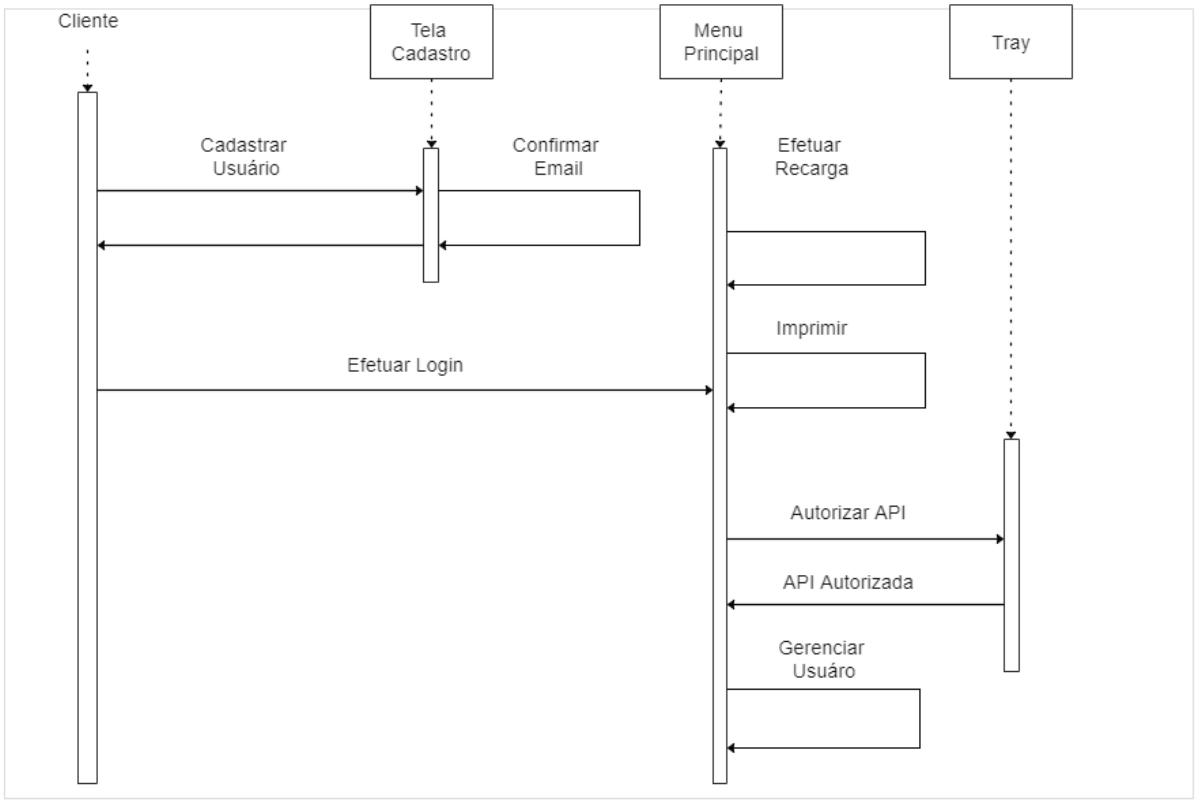

Figura 10: Diagrama de sequência

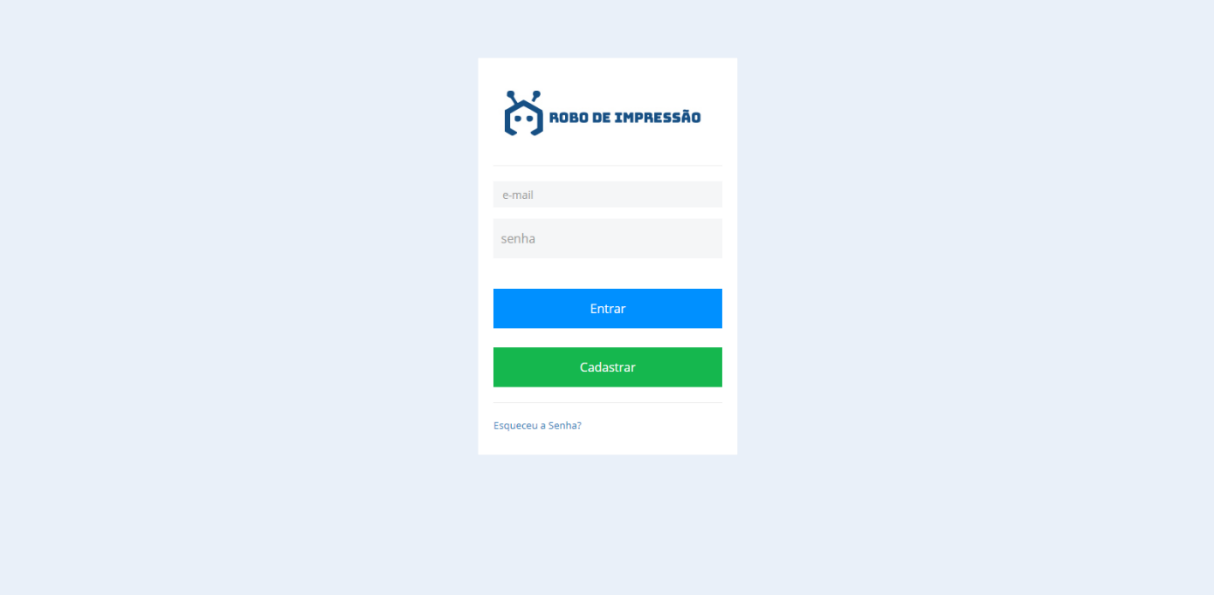

Figura 11: Tela de *login* e cadastro

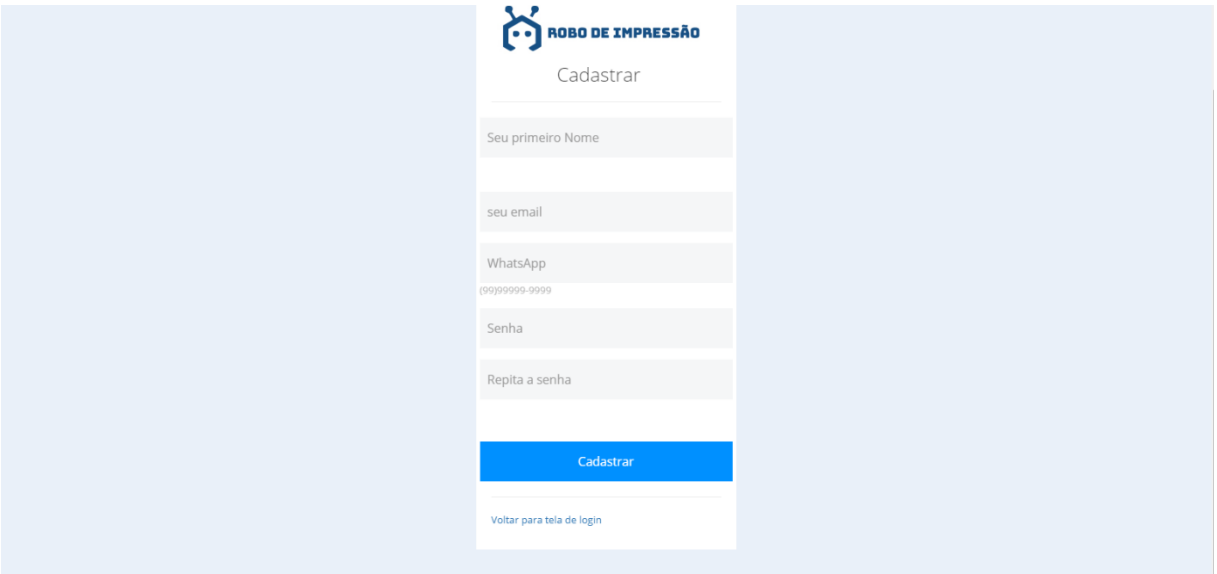

Figura 12: Tela de cadastro

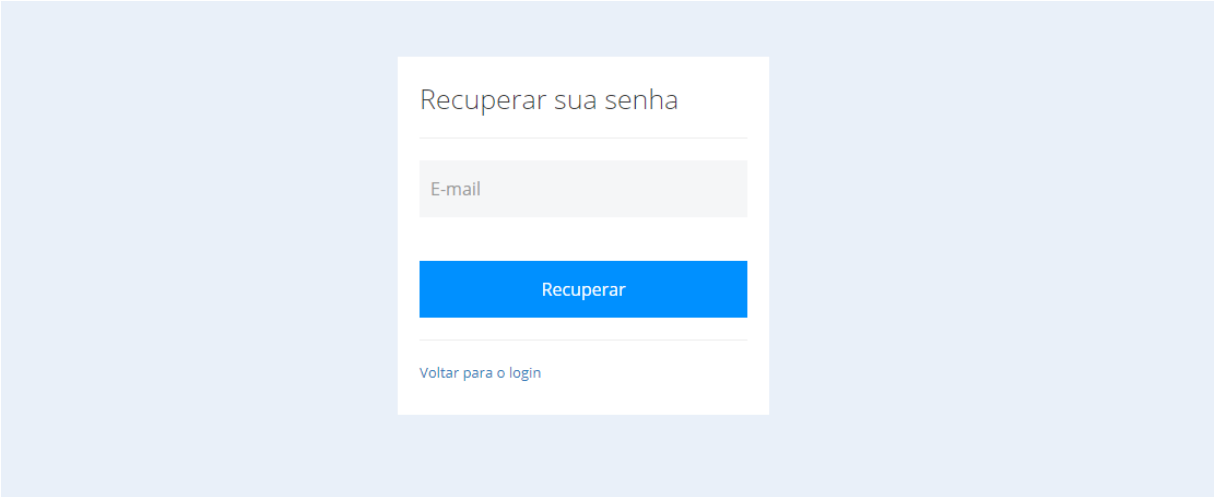

Figura 13: Tela de recuperação de senha

Figura 14: Tela de listagem de pedidos

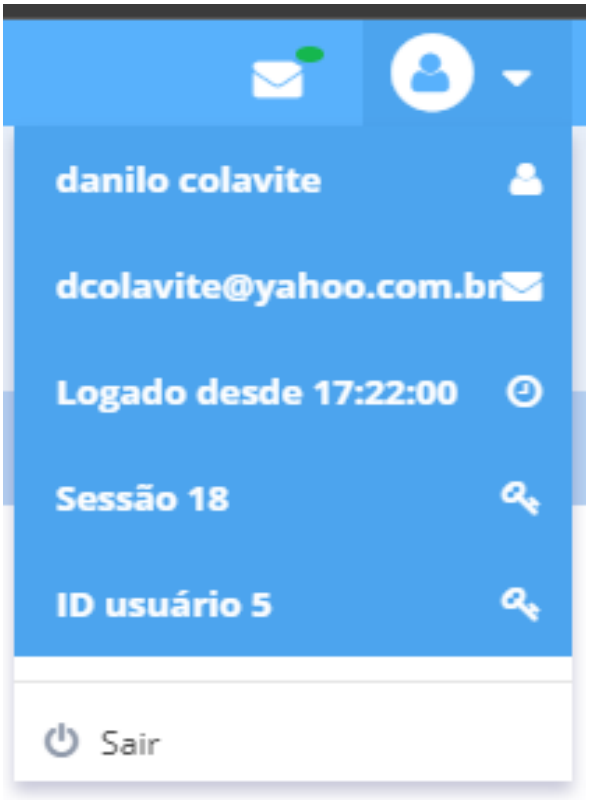

Figura 15: Painel de informações da sessão

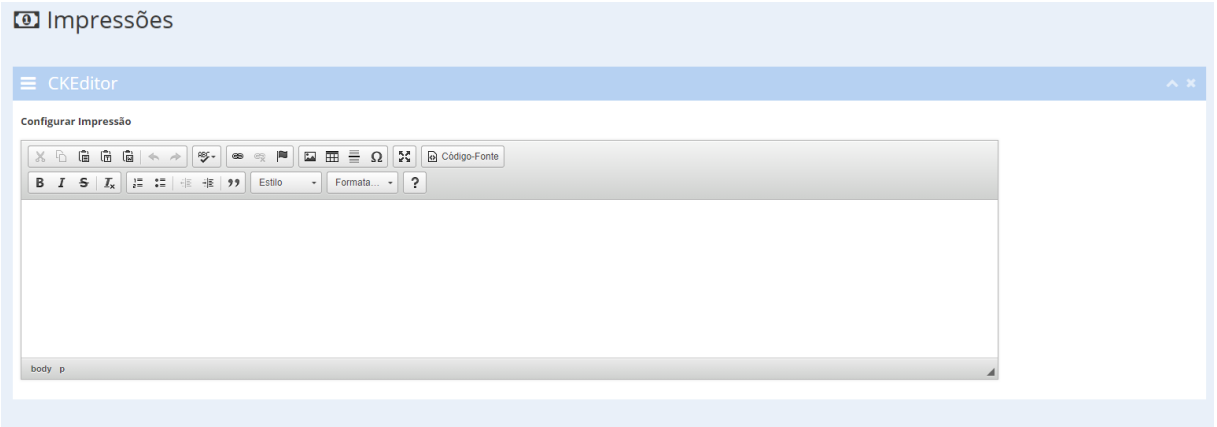

Figura 16: Exemplo do componente para criação de impressões personalizadas

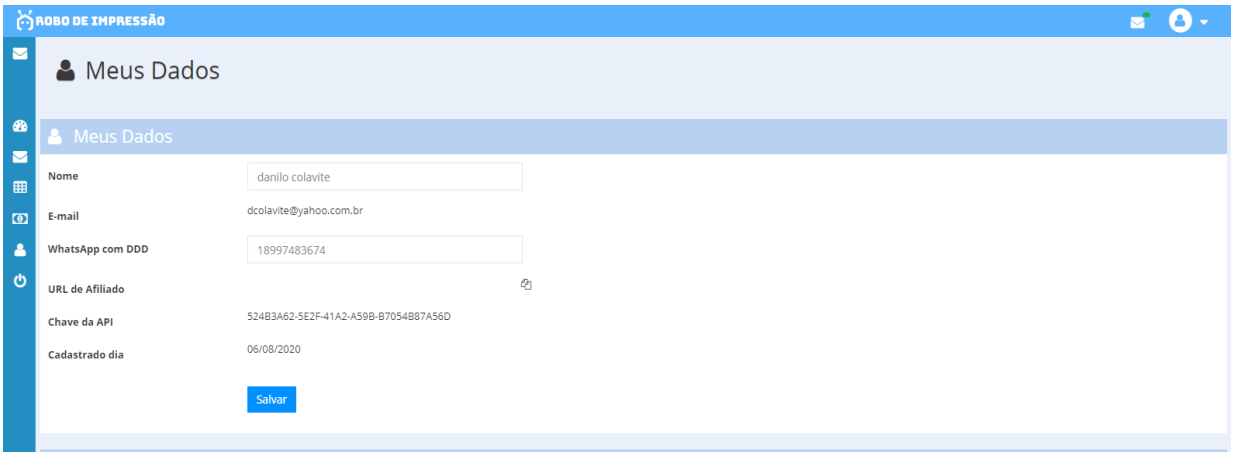

Figura 17: Tela de dados pessoais do cliente

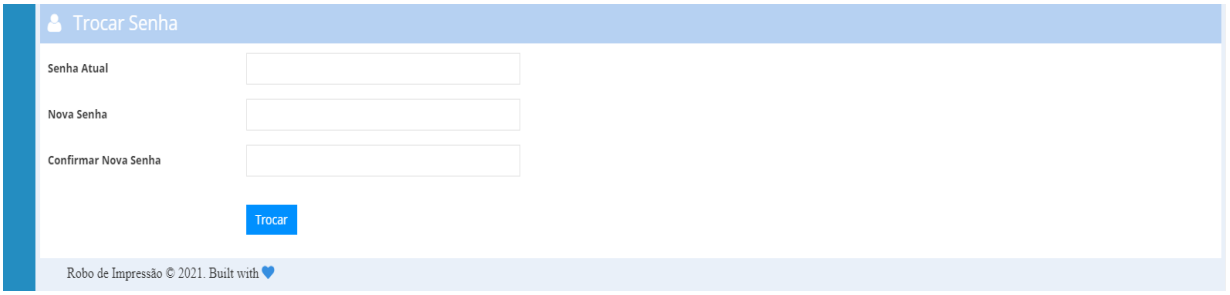

Figura 18: Tela de troca de senha

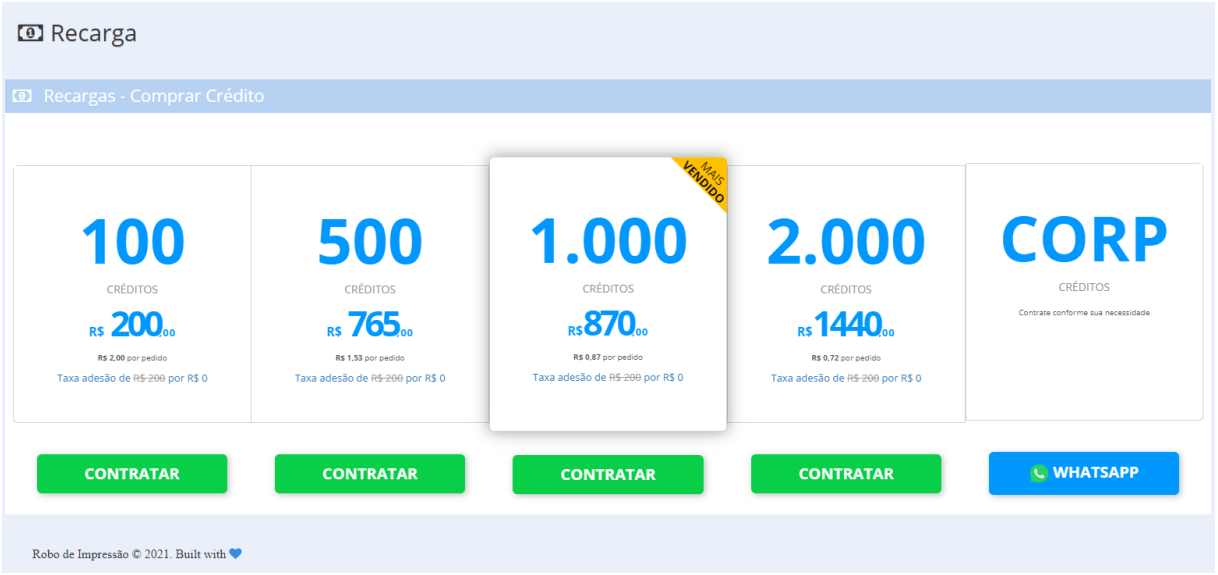

Figura 19: Tela de recarga

3.1. DETALHES DA IMPLEMENTAÇÃO DA FERRAMENTA

Banco de Dados: O Banco de Dados criado no SQL Server será utilizado apenas para guardar e processar informações do usuário, impressões, *login* e utilitários para personalização das impressões. Optou-se por listar os pedidos da loja virtual Tray por meio de sua API, pois as informações dos pedidos da loja virtual não serão salvas no Banco de Dados, economizando, assim, recursos e agilizando o processo.

Integração da API com a Loja virtual: Foi desenvolvida toda a parte de integração via API com a loja virtual Tray, e com o Webhook disponibilizado por ela. A integração foi toda desenvolvida em ASP, armazenando no banco de dados apenas as chaves e códigos necessários para a autenticação da API Tray, os quais precisam ser atualizados sempre que a data de expiração for atingida.

Integração da API com paypal: Será desenvolvida ainda, utilizando a linguagem ASP, a integração com a empresa de pagamentos online PayPal, onde ela será utilizada para realizar uma cobrança recorrente para que o cliente possa continuar utilizando o robô de impressão. A empresa Web Managers ainda não estipulou planos e preços para o robô de impressão, mas assim que o fizerem, divulgarão antes do lançamento do produto.

Impressão: O robô de impressão trabalhará com impressoras térmicas, pois além de serem mais rápidas, estas geram um custo mais baixo ao cliente por ser necessário apenas trocar as suas bobinas. Será desenvolvida uma checagem de status dos pedidos da loja virtual, onde os mesmos serão salvos no Banco de Dados, e com isso o cliente poderá escolher em qual status será realizada a impressão. O Webhook da Tray notificará o robô quando o status do pedido for mudado na loja virtual, como por exemplo: de “aguardando pagamento” para “pagamento aprovado”, assim os pedidos serão impressos apenas nos status desejado pelo cliente.

Será criada a possibilidade de o cliente selecionar o intervalo de tempo em que o robô deverá realizar as impressões automatizadas, portanto sempre que o robô atingir o tempo selecionado, ele buscará na loja virtual os pedidos que possuem o status selecionado pelo cliente, para então realizar as impressões.

Também serão desenvolvidos modelos padrões de impressão que o cliente poderá utilizar na aplicação, com a opção de criar modelos personalizados de impressão nos quais poderá selecionar as variáveis desejadas, como por exemplo: o preço do produto; nome do cliente; endereço; etc.

Será criada, ainda, uma tabela no banco de dados com o nome de “Variável” onde serão cadastradas as variáveis que os clientes poderão utilizar nas impressões. Esse cadastro será realizado de acordo com o Json de retorno da listagem de pedidos da loja virtual.

Pedido WM10: 251683
Cod. Pedido LV:

Data / Hora: 07/09/2020 22:31:00

Local do Pedido :

CLIENTE

Nome: CONSUMIDOR
CPF:
Email:
Endereço: , ,

Local da Venda:
Telefone:
CEP:
Cidade/UF:

PRODUTOS

Cod Prod	Produto	Referencia	Qtd	Unitário	Subtotal	Localizacao
955	Kit Shimano Xtr Câmbios / Trocadores 11v	001134	3	R$ 35,00	R$105,00	SemLocalização
12315	PNEU (DELLI TIRE) 29X2.10 MISTO SA-258	1055	2	R$ 31,19	R$62,38	SemLocalização

Total: R$ 167,38

ENVIO

Forma de envio escolhida : Valor : R$ 0,00 Frete:
Peso do Pedido : 0

OBS

Figura 20: Exemplo de impressão personalizada

Entre as impressões padrões serão desenvolvidas as impressões artísticas a serem utilizadas para enviar aos clientes em datas comemorativas, fazendo com que o cliente se sinta importante.

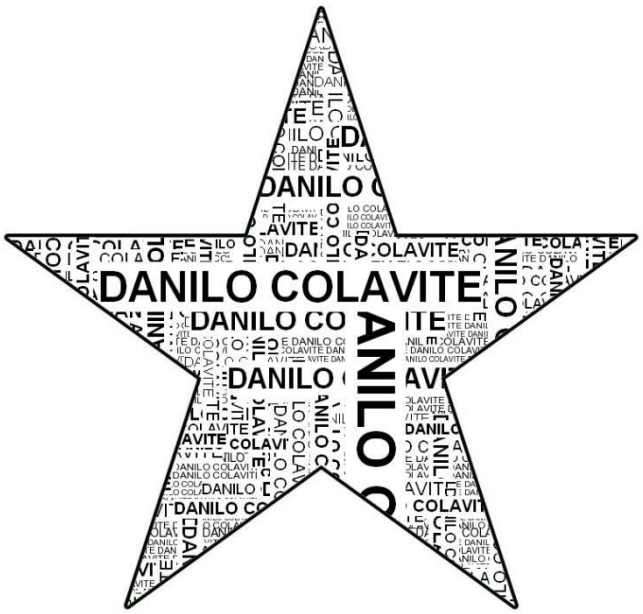

Figura 21: Exemplo de impressão artística

Todo processo de integração com a Tray será feito na linguagem ASP, pois se realizado via Banco de Dados, poderá haver atrasos na comunicação entre o robô de impressão e a Tray. Essa arquitetura adotada, inclusive, economiza recursos do Banco de Dados.

4. RESULTADOS

Os resultados mostraram a eficácia do robô em concluir a automatização das impressões personalizadas, já sendo utilizado por dezenas de clientes.

É possível iniciar e configurar o robô logo no seu modal inicial como mostra a imagem abaixo:

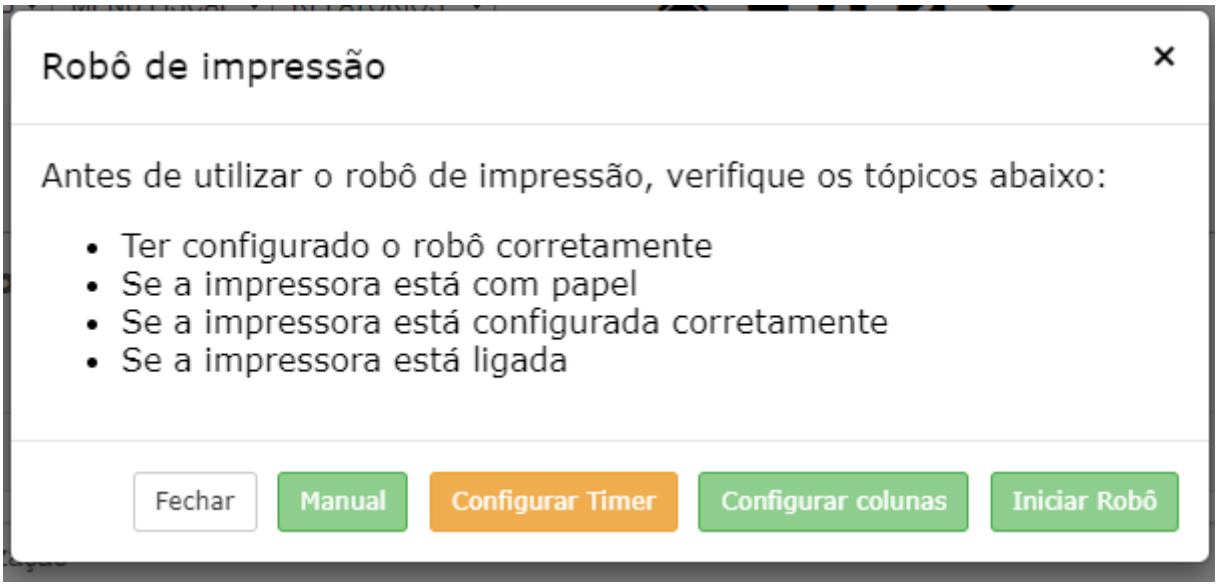

Figura 22: Modal Principal

Sendo possível configurar as colunas pré definidas que serão impressas desta forma:

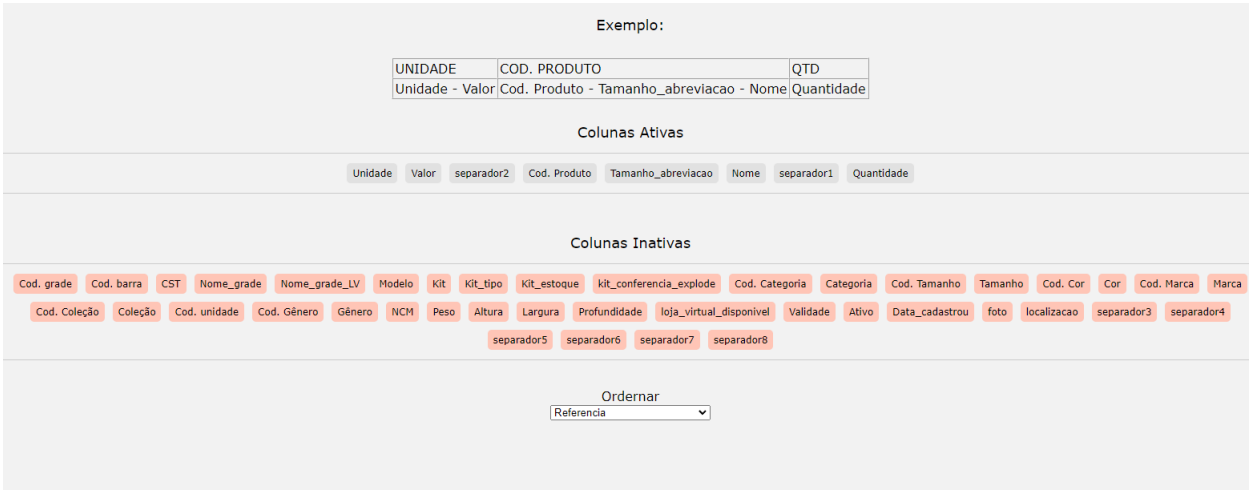

Figura 23: Colunas pré definidas

O usuário também poderá configurar o temporizador do robô:

Figura 24: Configuração do temporizador do robô

Na parte de personalização de impressões, é possível através do componente e de variáveis pré definidas pelo robô, personalizar as impressões como o usuário desejar. Também estão disponíveis impressões pré definidas com foco na humanização, onde o usuário poderá utilizar da forma que desejar.

Segue abaixo uma imagem do componente de personalização com uma impressão com foco na humanização pré definida:

Figura 25: Exemplo de impressão pré definida com foco em humanização

A mesma impressão já sendo visualizada para imprimir, com o nome do cliente da venda:

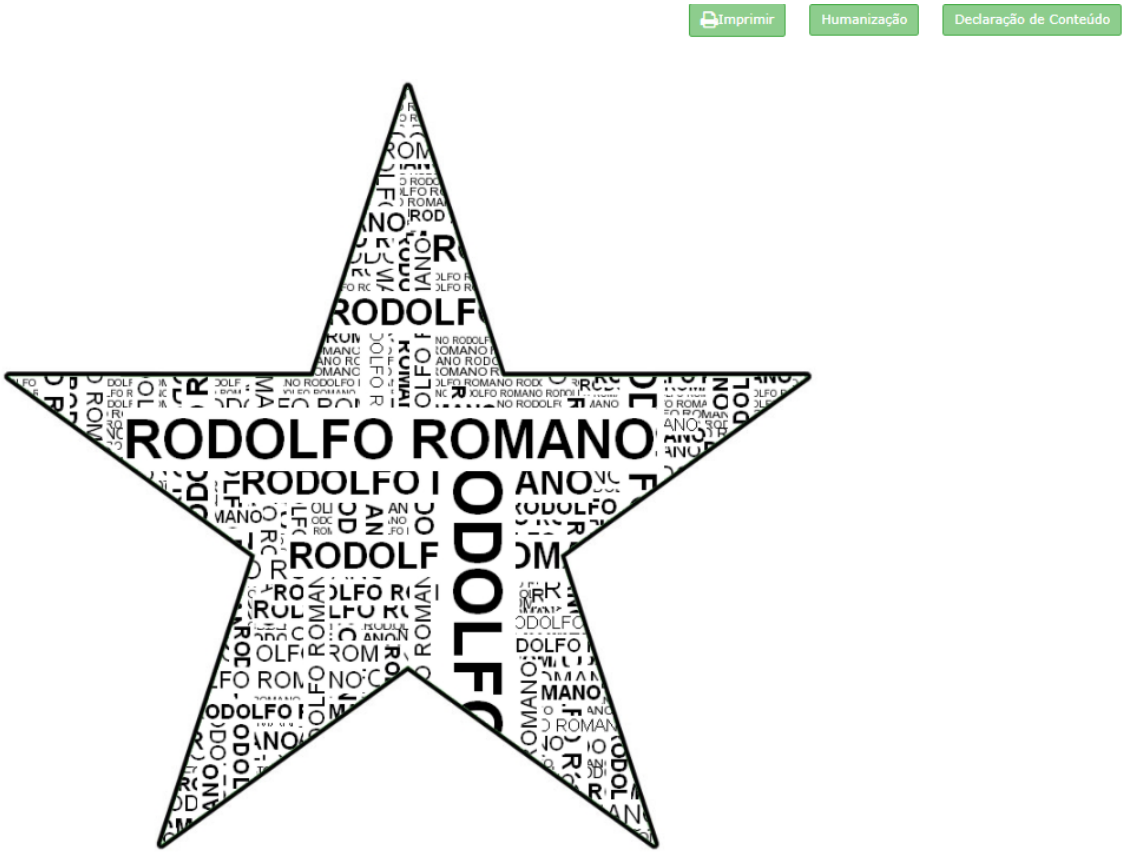

Figura 26: Exemplo com o nome do cliente já preparado para impressão

Para facilitar as impressões personalizadas, foram criadas as variáveis pré definidas abaixo, onde o cliente poderá utilizá-las na impressão:

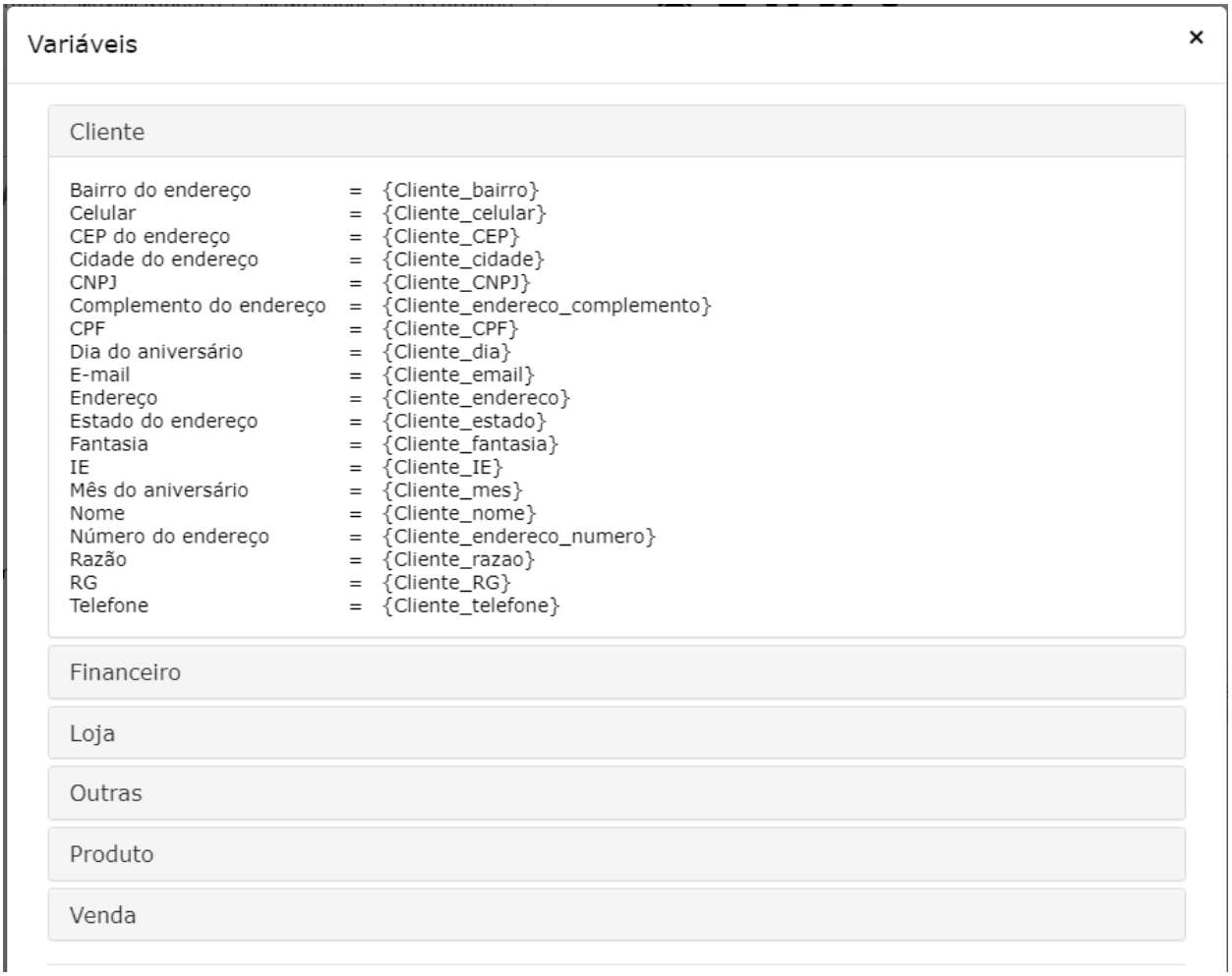

Figura 27: Variáveis pré definidas

Com essas opções de personalização o usuário já consegue criar uma boa variedade de tipos de impressões, desde uma etiqueta para envio até uma carta com um foco na humanização de processos automatizados.

Os resultados da pesquisa mostraram-se, então, satisfatórios, pois o robô funcionou da maneira almejada.

5. CONCLUSÃO

O robô de impressão desenvolvido obteve sucesso na automatização para impressão de pedidos da loja virtual, dando mais agilidade, customização e redução de custos aos clientes. Notou-se que a agilidade obtida nesse processo foi crucial para os clientes, pois sem utilizar o robô de impressão, eles perdiam muito tempo realizando o processo manualmente.

Os *templates* padrões do robô conseguiram criar impressões específicas para cada cliente de forma dinâmica de acordo com o pedido vindo da loja virtual, como por exemplo: uma impressão de aniversário com nome e sobrenome do cliente em formato de coração, fazendo com que o objetivo da humanização do processo seja atingido.

Com isso, pode-se concluir que tudo o que foi planejado para o produto, foi implementado sem maiores problemas. Mesmo com toda a tecnologia atual, foi possível utilizar um robô web para tornar mais humano, ainda que de maneira simples, um processo automatizado. Portanto os objetivos deste trabalho foram alcançados com sucesso.

Além do que foi proposto no trabalho, nas últimas semanas do projeto a empresa Web Managers solicitou também que o robô de impressão fosse integrado ao ERP WM10. A integração no ERP foi efetuada com sucesso e já se encontra em produção sendo utilizada pelos clientes do ERP WM10.

Futuramente, espera-se realizar novas integrações com demais lojas virtuais e não somente com a Tray, abrindo um leque maior de clientes que poderão utilizar o produto robô de impressão. Pretende-se também inserir novos modelos padrões de impressão periodicamente, para facilitar cada vez mais a operação dos clientes.

6. REFERÊNCIAS

APOSTILA de ASP. **FAETERJ-RIO – Faculdade de educação tecnológica do Rio de Janeiro**, 2015. Disponível em: https://www.faeterj- rio.edu.br/downloads/bbv/0033.pdf . Acesso em: 07 Marc.2020.

AUDITORIA do SQL Server (Mecanismo de Banco de Dados). **Microsoft**, 2016. Disponível em: https://docs.microsoft.com/pt-br/sql/relational-databases/security/auditing/sql-server-audit-database-engine?view=sql-server-ver15

. Acesso em: 07 Marc.2020.

CAMPOMORI, Cleber. REST não é simplesmente retornar JSON: indo além com APIs REST. **TreinawebBlog**, 2017. Disponível em: https://www.treinaweb.com.br/blog/rest-nao-e-simplesmente-retornar-json-indo-alem-com-apis-rest/ . Acesso em: 16 de Marc.2020.

CSS: *Cascading Style Sheets*. **Datamec S.A.**, Ago. 2001. Disponível em: https://iconectado.com.br/download-apostila-de-css . Acesso em: 08 Marc.2020.

ERP Webby Managers. **WM10**, 2021. Disponível em: https://www.wm10.com.br/ . Acesso em: 05 Out.2021.

FERREIRA, Rodrigo. REST: Princípios e boas práticas. **Caelum**, 2017. Disponível em: https://blog.caelum.com.br/rest-principios-e-boas-praticas/ . Acesso em: 16 Marc.2020.

FRANK, Diego R., SEIBT, Leonardo. JavaScript. **LSeibt**, 2004. Disponível em: https://fit.faccat.br/~leonardoseibt/ArtigoJavaScript.pdf . Acesso em: 08 Marc.2020.

GONÇALVES, Eduardo Corrêa. JSON Tutorial. **DevMedia**, 2012. Disponível em: https://www.devmedia.com.br/json-tutorial/25275 . Acesso em: 15 Marc.2020.

LEWIS, Joseph R.; MOSCOVITZ, Meitar. CSS Avançado. **Novatec**, 2017. Disponível em: https://s3.novatec.com.br/capitulos/capitulo-9788575222201.pdf . Acesso em: 08 Marc.2020.

MEDEIROS, Higor. Introdução ao Ajax. **Linhadecódigo**, (2019?). Disponível em: http://www.linhadecodigo.com.br/artigo/3585/ajax-basico-introducao.aspx . Acesso em: 15 Marc.2020.

MENEZES, Hanna França. Uma reflexão sobre o HTML5: Como essa tecnologia tem possibilitado a criação de páginas web mais interativas. **Encontros Universitários (Plataforma Encontros)**, 2013. Disponível em: https://conferencias.ufca.edu.br/index.php/encontros-universitarios/eu-2013/paper/view/2514. Acesso em: 12 Marc.2020.

MIGUEL, Flavia de Azevedo Marques; COSTA, Josélia Leite. Desenvolvimento de Sites Responsivos Utilizando o Framework Bootstrap com Aplicação de User Experience. **SlideShares**, 2015. Disponível em: https://pt.slideshare.net/joselialcosta/desenvolvimento-de-sites-responsivos-utilizando-o-framework-bootstrap-com-aplicao-de-user-experience . Acesso em: 13 Marc.2020.

NARDE, Wagner. O que são e como funcionam os Webhooks?. **Vindi**, 2018. Disponível em: https://atendimento.vindi.com.br/hc/pt-br/articles/203305800-Webhooks . Acesso em: 14 Marc.2020.

O QUE é uma plataforma de E-commerce e qual sua importância. **Guia de E-commerce**, 2019. Disponível em: https://www.guiadeecommerce.com.br/o-que-e-plataforma-de-ecommerce/ . Acesso em: 14 Marc.2020.

PADILHA, Thais Cássia Cabral et al . Tempo de implantação de sistemas ERP: análise da influência de fatores e aplicação de técnicas de gerenciamento de projetos. **Gest. Prod.**, São Carlos, v. 11, n. 1, p. 65-74, Apr. 2004. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2004000100006&lang=pt . Acesso em: 05 Nov.2019.

POLI, Márcio Schuster. A Influência da Tecnologia da Informação no Comportamento Humano. **Revista Científica Multidisciplinar Núcleo do Conhecimento**. Edição 02, Ano 02, Vol. 01. pp 101-113, Maio de 2017. ISSN:2448-0959. Disponível em: https://www.nucleodoconhecimento.com.br/tecnologia/influencia-comportamento-humano . Acesso em: 04 Set.2020.

REDAÇÃO Impacta. O que é Banco de Dados SQL Server? **Blog Impacta**, 2017. Disponível em: https://www.impacta.com.br/blog/entenda-de-uma-vez-por-todas-o-banco-de-dados-sql-server/ . Acesso em: 07 Marc.2020.

SACCOL, Amarolinda Zanela et al . Avaliação do impacto dos sistemas ERP sobre variáveis estratégicas de grandes empresas no Brasil. **Rev. adm. contemp.** Curitiba, v. 8, n. 1, p. 9-34, Mar. 2004. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1415-65552004000100002&lang=pt . Acesso em: 05 Nov.2019.

SANTANA, Luiz Antônio F. T.; PEREIRA, Flávius Anderson Félix. Novas Funcionalidades de Auditoria do SQL Server. **ISSUU**, 2015. Disponível em: https://issuu.com/luizfsantana/docs/artigo_luiz_antonio_-_pos_bd/1?ff . Acesso em: 07 Marc.2020.

SCHIMID, Jaison. Introdução ao JavaScript. **DevMedia**, 2012. Disponível em: https://www.devmedia.com.br/introducao-ao-javascript/25548 . Acesso em: 08 Marc.2020.

SILVA, Maurício Samy. Construindo Sites com CSS e (X)HTML: Sites controlados por folhas de estilo em cascata. **Novatec**, 2008. Disponível em: http://www.martinsfontespaulista.com.br/anexos/produtos/capitulos/674122.pdf . Acesso em: 08 Marc.2020.

SILVA, Maurício Samy. Introdução à jQuery. **Linhadecódigo**, 2020. Disponível em: http://www.linhadecodigo.com.br/artigo/2068/introducao-a-jquery.aspx . Acesso em: 14 Marc.2020.

SOBRE o change data capture (SQL Server). **Microsoft**, 2019. Disponível em: https://docs.microsoft.com/pt-br/sql/relational-databases/track-changes/about-change-data-capture-sql-server?view=sql-server-ver15 . Acesso em: 07 Marc.2020.

SOBRE o controle de alterações (SQL Server). **Microsoft**, 2016. Disponível em: https://docs.microsoft.com/pt-br/sql/relational-databases/track-changes/about-change-tracking-sql-server?view=sql-server-ver15 . Acesso em: 07 Marc.2020.

TECHIO, Gabriel Bressan; CHICON, Patricia Mariotto Mozzaquatro. Implementação dos frameworks bootstrap e Foundation aplicados na construção de um objeto de aprendizagem para o ensino da Engenharia de Software. **Anais do EATI – Encontro anual de Tecnologia da Informação**, 2016. Disponível em: https://docplayer.com.br/81126514-Implementacao-dos-frameworks-bootstrap-e-foundation-aplicados-na-construcao-de-um-objeto-de-aprendizagem-para-o-ensino-da-engenharia-de-software.html . Acesso em: 13 Marc.2020.

VISÃO geral de Webhooks do ASP.NET. **Microsoft**. 2012. Disponível em: https://docs.microsoft.com/pt-br/aspnet/webhooks/ . Acesso em: 14 Marc.2020.