

UNIVERSIDADE FEDERAL DA BAHIA  
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA  
BACHARELADO EM SISTEMAS DE INFORMAÇÃO  
TRABALHO DE CONCLUSÃO DE CURSO

GUIDE AUTOMATOR VIDEO JS: UMA PROPOSTA DE  
GERAÇÃO AUTOMÁTICA DE DOCUMENTAÇÃO COM IMAGENS  
E VÍDEO EM AMBIENTES COM INTEGRAÇÃO CONTÍNUA

ÍCARO ERASMO SOUZA BARREIRO

Salvador - Bahia

JUNHO DE 2021

# GUIDE AUTOMATOR VIDEO JS

ÍCARO ERASMO SOUZA BARREIRO

Trabalho de Conclusão de curso apresentado como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

**Orientador:** Prof. Dr. Rodrigo Rocha Gomes e Souza.

Salvador - Bahia

Junho de 2021

# GUIDE AUTOMATOR VIDEO JS

ÍCARO ERASMO SOUZA BARREIRO

Trabalho de Conclusão de curso apresentado  
como requisito parcial para obtenção do título  
de Bacharel em Sistemas de Informação.

## **Banca Examinadora:**

---

Prof. Dr. Rodrigo Rocha Gomes e Souza (Orientador)  
UFBA

---

Prof. Dr. Ivan do Carmo Machado  
UFBA

---

Prof. Dr. Mario Jorge Pereira  
IFBA, UCSAL

*A Maria Augusta, minha mãe*

# Agradecimentos

Agradeço a todos os que me apoiaram durante a jornada da graduação, especialmente Rafaela Almeida que esteve comigo desde o início compartilhando as refeições no Restaurante Universitário, o medo de sermos assaltados no percurso para casa e os cochilos no ônibus após as 22 horas. Todo o nosso esforço não foi em vão.

*“A man provided with paper, pencil,  
and eraser, and subject to strict  
discipline, is in effect a  
universal machine”.*

*Alan Turing*

# Lista de abreviações e acrônimos

API — Application Program Interface

CSS — Cascading Style Sheets

DOM — Document Object Model

HTML — HyperText Markup Language

MD — Markdown

MP4 — Moving Picture Expert Group 4

PDF — Adobe Portable Document Format

# Resumo

Criar documentação para projetos de desenvolvimento é uma tarefa contínua. A cada alteração no escopo e identidade visual da aplicação é necessário também realizar a alteração da documentação, principalmente se forem alterações estéticas e hajam imagens das telas em quaisquer dos documentos. Com a ascensão das metodologias ágeis, o retrabalho, que antes era abominado, passa a ser aceito como um aspecto positivo da evolução do software. Conseqüentemente, a frequência das alterações das documentações, ainda que o volume de documentos seja menor no contexto ágil, aumenta. A integração contínua consegue propor solução factível para identificar falhas de integração em ambiente de produção e permite diminuir bastante o tempo de disponibilização das correções para o público o qual o sistema é direcionado. Já é possível obter uma série de documentos que descrevem entradas e saídas de interfaces e modelagem dos sistemas de forma automática. O seguimento de ferramentas de documentação carece de opções de ferramentas disponíveis que permitam gerar documentação das funcionalidades em formato de vídeo e texto em formato PDF automaticamente, ainda que a ação de gerar a documentação parta de intervenção humana.

O objetivo deste trabalho é desenvolver e avaliar a ferramenta Guide Automator Video JS, que foi proposta a fim de sanar os problemas de geração de documentação descritiva de telas e funcionalidades de um sistema web qualquer navegando por suas páginas de forma automática, tendo feita a configuração de como acontecerá essa navegação apenas uma vez através de um script. O Guide Automator Video JS é membro de uma série de outras aplicações implementadas sob orientação do professor Rodrigo Rocha Gomes e Souza e que possuem, além do nome Guide Automator, o objetivo de gerar documentação a partir de sistemas web de forma automatizada.

Para avaliar o desempenho do Guide Automator Video JS em um contexto real de Integração Contínua, a aplicação foi configurada para executar junto ao sistema Micro Livraria [1], um sistema web simples. O repositório do sistema Micro Livraria foi ramificado a partir do repositório original e no novo repositório foram configuradas as ações a serem executadas quando o repositório recebe um comando push. Quando o repositório percebe a execução do comando push, um gatilho é acionado e inicia a execução do Guide



Automator Video JS, e ao final da execução os arquivos de documentação em PDF e MP4 são disponibilizados na seção “*releases*” do repositório.

Todo o processo de disponibilização da documentação como artefatos da release acontece como esperado embora o tempo para finalização da execução seja relativamente alto e a biblioteca utilizada para a gravação de vídeo não emita taxa constante de frames para a obtenção de um vídeo fluido. A não obtenção de um vídeo fluido ocasiona também a dessincronização do áudio e legendas presentes no vídeo.

# Abstract

To create documentation for development projects is a continuous task. Each change in scope and visual identity of the application causes also changes in the documentation, mainly if the changes have happened in the system look and feel and there are system screen's images in any of the documents. With agile methodologies growth, rework, which was once abominated, becomes accepted as a positive aspect of software evolution. Consequently, the frequency of changes in documents, even though documents volume is lower in agile context, grows. Continuous Integration can propose tangible solution to identify integration errors in production environment and allows to shorten the time of delivery of those fix to the public those artifacts were meant to. It is already possible to get several kinds of documents which describe inputs and outputs of interfaces and model of systems in an automated manner. There is a lack of options in the segment of documentation tools capable of generating features text documentation in PDF format and video automatically from the application being built, even if the action of generate documentation comes from human intervention.

This work aims to develop and evaluate Guide Automator Video JS, which was meant as a solution to create describing documentation of screens and features of any web system browsing its pages automatically, having configured the way of how this browsing is going to happen only once through a script. Guide Automator Video JS is part of a series of other applications implemented under the guidance of Professor Rodrigo Rocha Gomes e Souza which, besides of having the name Guide Automator, has also the goal of generate documentation from web systems in an automated manner.

To evaluate the performance of Guide Automator Video JS in a real Continuous Integration Environment, The application was configured to run along with the system Micro Livraria [1], a simple web system. The Micro Livraria repository was forked from the original repository and in the newer one were configured the actions that will be run whenever the repository receives a push command. When the repository realizes the push command execution, a trigger is fired and Guide Automator Video JS execution starts, and in the end of execution PDF and MP4 files are put in the releases section of repository.

All process of artifacts release happens as meant although deployment time takes

too long and video record library does not emit a constant frame rate to get a smooth video. Freezing video causes also audio and subtitles synchronization problems.

# Sumário

<b>Lista de abreviações e acrônimos</b>	<b>1</b>
<b>1 Introdução</b>	<b>8</b>
<b>2 Fundamentação Teórica</b>	<b>10</b>
2.1 Práticas Contínuas . . . . .	10
2.2 Ferramentas de automação de navegadores . . . . .	11
2.3 Headless Recorder . . . . .	13
<b>3 Trabalhos relacionados</b>	<b>14</b>
3.1 GuideAutomator . . . . .	14
3.2 GuideAutomator Mobile . . . . .	15
3.3 Guide Automator Video . . . . .	16
3.4 Ari . . . . .	16
3.5 Quadro comparativo entre as ferramentas . . . . .	17
<b>4 Guide Automator Video JS</b>	<b>19</b>
4.1 Execução . . . . .	19
4.1.1 Parâmetros de execução . . . . .	20
4.1.2 Comandos . . . . .	20
4.2 Guide Automator Recorder . . . . .	21
4.3 Implementação . . . . .	22
4.3.1 Ferramentas e bibliotecas utilizadas . . . . .	24
<b>5 Avaliação</b>	<b>29</b>
5.1 Descrição da aplicação utilizada nos testes . . . . .	29
5.2 Execução do Guide Automator Video JS . . . . .	30
5.3 Resultados obtidos . . . . .	31
5.4 Limitações . . . . .	32
<b>6 Conclusão</b>	<b>34</b>

<b>Anexos</b>	<b>35</b>
1 cover.html . . . . .	35
2 livraria.md . . . . .	35
3 guide-automator-video-publish.yml . . . . .	36
4 Dockerfile . . . . .	38
5 github-package-publish.yml . . . . .	40
6 output.pdf . . . . .	42
<b>Referências Bibliográficas</b>	<b>45</b>

# Capítulo 1

## Introdução

Integração contínua, Implantação contínua e Entrega contínua são processos cada vez mais frequentes entre as empresas que coordenam e executam o desenvolvimento de software. O principal motivo é que estes agilizam processos de construção dos artefatos e também agilizam o processo de teste, bem como a disponibilização de novas versões em produção.

O intuito da utilização de ferramentas que executem estes processos é automatizar tarefas antes executadas manualmente e que eram suscetíveis a erros humanos ou ainda, suscetíveis a ações imprudentes de desenvolvedores ou mantenedores que poderiam evitar etapas como a etapa de testes a fim de agilizar a execução da tarefa e reduzir o trabalho executado por eles.

Quando há um fluxo pré-estabelecido para a execução das etapas, fluxo este definido na ferramenta utilizada e que somente pessoas com privilégio para tal têm permissão para alterar, a possibilidade de que erros como os mencionados aconteçam é diminuída.

Martin Fowler [2] define que “Integração contínua é uma prática de desenvolvimento de software onde membros de um time integram seus trabalhos frequentemente, usualmente cada pessoa integra pelo menos uma vez por dia - levando a múltiplas integrações por dia”. Complementando a definição de Fowler; Shahin, Babar e Zhu [3] descrevem as práticas contínuas da seguinte maneira: “*Práticas contínuas, i.e., integração contínua, entrega contínua e implantação contínua, são práticas da indústria de desenvolvimento de software que permitem a organizações frequentemente e confiavelmente disponibilizar novas funcionalidades e produtos*”. Através da utilização dos processos de integração contínua é possível a geração automática de documentos de modelagem e descrição de interfaces de APIs. Porém, os esforços para a produção de documentação automática em vídeo e em formato de manual ainda continuam baixos. Ainda há poucas opções de ferramentas que forneçam documentação contendo imagens das telas de um sistema produzidas automaticamente e este trabalho ainda é feito de forma manual. Gravar um

vídeo e criar um documento contendo telas do sistema ainda demanda tempo sempre que for necessário adaptá-los conforme a evolução das telas dos sistemas.

O Guide Automator Video JS, assim como o Guide Automator Video, Guide automator Mobile e Guide Automator, versões anteriores a este, tem o objetivo de automatizar a criação destes artefatos de forma que o usuário final possa ter um documento descritivo e também um vídeo das telas do sistema em sua versão mais recente sem que seja necessário atualizar vídeo e documento escrito de forma manual. Isso é feito gerando vídeo-tutorial descrevendo a utilização da aplicação com áudio e legendas e manual escrito em formato PDF. Como entrada é inserido um roteiro contendo textos e comandos que simulam a interação do usuário com um navegador web intercalados. A primeira versão, o Guide Automator Video, apesar de gerar documentação como previsto, necessita forçar a espera do carregamento das páginas após cada requisição efetuada pois não dispõe de mecanismo que identifique se os elementos foram carregados como deveriam. O objetivo é que o Guide Automator JS seja de fácil execução e que possa lidar com intercorrências que possam variar o tempo de carregamento das páginas. A ferramenta necessita ser adaptada ao contexto de Integração Contínua e definir manualmente o tempo de carregamento das páginas atrapalharia o fluxo em uma aplicação real, podendo ocasionar erros de execução da implantação caso uma determinada ação demore mais que o previsto, por exemplo, ou gerar longos espaços de tempo com a tela parada caso o tempo configurado fosse maior que o necessário.

Neste trabalho serão abordados aspectos relevantes a respeito da implementação e avaliação da solução proposta para o desenvolvimento do Guide Automator Video JS, bem como as ferramentas utilizadas no processo, principalmente a ferramenta de automação de navegador Puppeteer [4], na qual baseia-se majoritariamente a implantação do Guide Automator Video JS. A avaliação consiste na implantação do Guide Automator Video JS em um ambiente real de Integração contínua a fim de observar o comportamento deste e provar que é uma solução viável para aplicações web reais.

O restante desta monografia está organizado como descrito a seguir: o capítulo 2 traz a fundamentação teórica, descrevendo as práticas contínuas e ferramentas de automação de navegadores; o capítulo 3 descreve os trabalhos relacionados, como bibliotecas e aplicações; o capítulo 4 descreve o Guide Automator Video JS e o Guide Automator Recorder de forma geral e o trabalho encerra com os Capítulos 5 e 6 que tratam respectivamente da avaliação da aplicação e da conclusão.

# Capítulo 2

## Fundamentação Teórica

Nesta seção serão abordados os fundamentos de práticas contínuas e ferramentas de automação de navegadores que são aspectos importantes para o entendimento geral deste trabalho.

### 2.1 Práticas Contínuas

Segurança, agilidade nas entregas e código livre de defeitos são requisitos fundamentais para a satisfação do usuário de um sistema de computação. E para que esses requisitos sejam contemplados, o esforço para o crescimento das práticas de automatização de processos de teste e construção de artefatos tem crescido. O DevOps surgiu como o paradigma que propunha soluções para as barreiras encontradas durante a busca por realização destes objetivos. Zhao et al. [5] define o DevOps como “*uma cultura que enfatiza automação dos processos de construção, testes e implantação de software*”. O paradigma traz consigo três definições importantes a respeito dos processos de automação:

- CI: *Continuous Integration* ou Integração contínua em português;
- CD: *Continuous Deployment* ou Implantação contínua em português;
- CDE: *Continuous Delivery* ou Entrega contínua em português.

A integração contínua consiste em constantemente integrar código novo à branch principal. Há uma confusão a respeito desses conceitos pois estes são muito parecidos em definição e por muitas vezes são utilizados em conjunto num projeto. Observe que se não houver adição de código novo com frequência à branch principal, o fluxo não está sendo respeitado e portanto, o processo não pode ser chamado de integração contínua. Implantação contínua diz respeito à construção dos artefatos e também à implantação destes



em ambiente a ser acessado pelo usuário da aplicação de forma inteiramente automática. Sejam eles de desenvolvimento, homologação, produção ou quaisquer outros.

O aspecto que difere a implantação contínua da entrega contínua é que na entrega contínua pode ser necessária a intervenção humana em ambiente de produção. Portanto, aplicar o conceito de implantação contínua implica em também aplicar a entrega contínua. O contrário não é verdade uma vez que nenhum aspecto da implantação contínua deve ser manual.

Um ambiente automatizado em sua completude atualmente vale-se de todos esses conceitos descritos anteriormente. Quando houver, por exemplo, a implementação de uma nova funcionalidade, os artefatos serão construídos, os testes serão executados, se não houver erro o código será integrado diretamente no código principal e se houver, o desenvolvedor será notificado a respeito para que possa realizar os reparos necessários para que a aplicação permaneça estável.

## 2.2 Ferramentas de automação de navegadores

Estas ferramentas foram criadas inicialmente com o objetivo de automatizar a navegação através de páginas web com o objetivo de realizar testes. Porém logo encontrou-se outro propósito para a utilização da tecnologia: automatizar tarefas manuais como preenchimento de formulários e cliques em componentes específicos como botões, tabelas e menus. O site InfoQ [6] cita Selenium [7] e Puppeteer [4] em seu artigo “JavaScript and Web Development InfoQ Trends Report 2020” entre as principais tendências no ramo de testes de aplicações e automação web no ano de 2020.

Há versões de bibliotecas do Selenium [7] para as linguagens C#, JavaScript, Java, Python e Ruby. O Selenium [7] é atualmente a ferramenta de automação mais popular de seu gênero tendo sido utilizada tanto para o contexto de testes de interface como também para implementação de ferramentas de automação de tarefas para sistemas específicos e que são popularmente conhecidas como “Robôs”. Este traz opções para as mais diversas necessidades de projeto se tornando então uma solução muito mais completa do que fora prevista inicialmente e inclusive está adaptada para a execução de testes de interface em ambientes de Integração Contínua. A ferramenta conta com produtos diferentes para cada propósito e necessidades diferentes de projetos. Estes produtos são:

- Selenium IDE: Ferramenta integrada para criação de scripts de testes automatizados e que pode ser utilizada para gravar as ações de um usuário ao interagir com o navegador;
- Selenium WebDriver: Utiliza o próprio navegador como ferramenta de automação

através da API específica de cada navegador, o que torna a navegação mais próxima de uma navegação realizada por um usuário humano;

- Selenium Grid: Permite a execução do script de automação em diversos navegadores diferentes e permite a execução paralela dos scripts em cluster.

O Selenium [8] pode ser configurado com Chrome, Firefox, Edge, Internet Explorer, Opera e Safari.

Já o Puppeteer [4] é executado exclusivamente em JavaScript (node.js [9]) e também tem utilização mais simples embora cumpra objetivos semelhantes ao Selenium [8]. Uma desvantagem do Puppeteer [4] é que este pode ser executado apenas utilizando o navegador Chromium e sua versão proprietária, o Chrome. Há também uma versão do Puppeteer [4] em desenvolvimento que será executada utilizando o Firefox. O Puppeteer [4] tem curva de aprendizado fácil e intuitiva e cumpre bem o papel de automatizar a navegação, permitindo a execução de tarefas como preenchimento automatizado de formulários, navegação automatizada, geração de PDFs e a ação de coleta de dados de páginas HTML da internet, conhecida como “Web Scraping”.

Vaidya [10] descreve em seu artigo um quadro comparativo entre as duas ferramentas da seguinte maneira:

<b>Puppeteer</b>	<b>Selenium</b>
Foi desenvolvido pelo Google e executa o script no Chromium (Chrome).	É uma biblioteca para o node.js que é utilizada para automatizar o Chrome. É de código-aberto e fornece uma API de alto nível para controlar o Chrome.
É uma biblioteca para o node.js.	É um framework web idealizado para o teste de aplicações web.
Funciona apenas com o Chrome ou Chromium e não possui suporte a outros browsers.	Suporta múltiplos browsers como Chrome, InternetExplorer, Firefox, Safari, etc. Suporte entre plataformas é fornecido através de todos os browsers suportados.
Lançado em 2017.	Lançado em 2004.
Suporta apenas o node.js	Suporta múltiplas linguagens como Python, Ruby, Javascript, Java, etc.
Suporta apenas automação web.	Suporta automação web e móvel.
Screenshots podem ser extraídos a partir de arquivos PDF e imagens.	Screenshots podem ser extraídos a partir de arquivos PDF e imagens apenas a partir da versão 4.

Embora o Selenium [8] seja uma ferramenta com maior maturidade e mais rica em recursos, tem um sistema complexo para identificar se uma página foi carregada após uma ação que resulta em navegação entre páginas. O Puppeteer [4], além de ter sido construído para executar nativamente em node.js [9], implementa a função “waitForNavigation” que, de acordo com o valor passado por parâmetro, monitora as movimentações em rede e na própria página web, observando eventos disparados para dar o carregamento da página como finalizado.

## 2.3 Headless Recorder

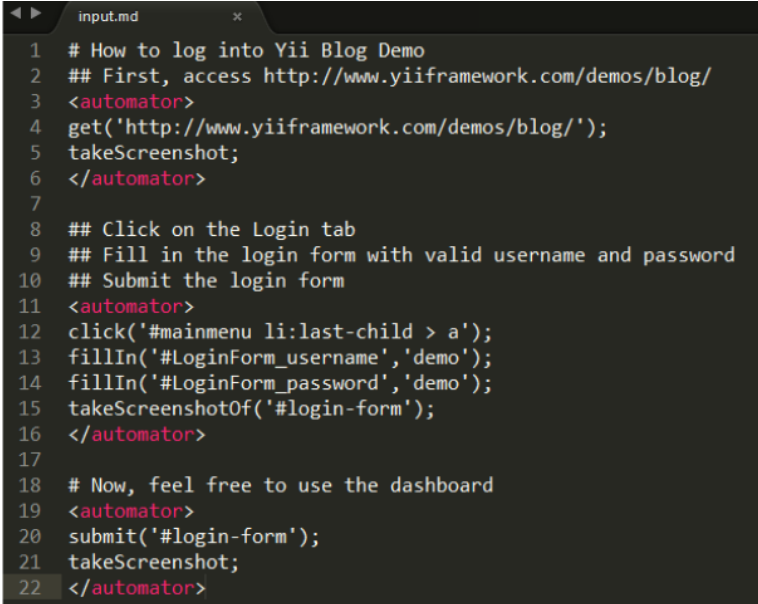
O Headless Recorder [11] é um plugin criado para o Chrome que tem por objetivo capturar ações realizadas no navegador e gerar a partir destas um script escrito em linguagem Javascript para ser utilizado com o Puppeteer [4]. O plugin foi feito na versão 8 do node.js [9] e antes chamava-se “Puppeteer Recorder”. A mudança aconteceu pois agora, além do Puppeteer [4], o Headless Recorder [11] também suporta a ferramenta de automação “Playwright”.

# Capítulo 3

## Trabalhos relacionados

Nesta seção serão descritas as versões anteriores do GuideAutomator [12] e também da biblioteca Ari [13], cujas funcionalidades são a criação de um vídeo narrado a partir da leitura de um arquivo escrito utilizando a linguagem de marcação Markdown [14].

### 3.1 GuideAutomator



```
1 # How to log into Yii Blog Demo
2 ## First, access http://www.yiiframework.com/demos/blog/
3 <automator>
4 get('http://www.yiiframework.com/demos/blog/');
5 takeScreenshot;
6 </automator>
7
8 ## Click on the Login tab
9 ## Fill in the login form with valid username and password
10 ## Submit the login form
11 <automator>
12 click('#mainmenu li:last-child > a');
13 fillIn('#LoginForm_username','demo');
14 fillIn('#LoginForm_password','demo');
15 takeScreenshotOf('#login-form');
16 </automator>
17
18 # Now, feel free to use the dashboard
19 <automator>
20 submit('#login-form');
21 takeScreenshot;
22 </automator>
```

Figura 3.1: Exemplo de sintaxe de arquivo de entrada do GuideAutomator

O GuideAutomator [12] (Figura 3.1) foi proposto em 2016 como um trabalho de conclusão de curso. O seu objetivo era a criação automatizada de um manual de uso de uma determinada aplicação utilizando um arquivo escrito em Markdown [14] como entrada. Este foi o precursor e principal inspiração para todas as evoluções do Guide Automator. Assim como o Guide Automator Video JS, este também foi feito utilizando

o Node.js [9]. Diferentemente do Guide Automator Video JS, o GuideAutomator utiliza o Selenium [8] como ferramenta principal para automatizar as ações no navegador e gera somente manual escrito.

## 3.2 GuideAutomator Mobile



Figura 3.2: Exemplo de entrada na interface do Jupyter utilizada pelo GuideAutomator Mobile

O GuideAutomator Mobile [15] gera manuais automaticamente a partir de aplicativos mobile. Ele utiliza a aplicação Jupyter Notebook como interface para a escrita de um script em linguagem de marcação a ser interpretado a fim de realizar a captura de imagens da aplicação mobile em execução e anexá-las ao manual. É necessário estar com um smartphone com a aplicação alvo instalada conectado ao computador que executará o GuideAutomator Mobile. Este utiliza a mesma sintaxe proposta pelo Guide Automator em seu script e a diferença mais marcante entre este e o GuideAutomator é a interface do Jupyter que foi utilizada a fim de prover ao usuário uma interface mais intuitiva e atrativa, além de da possibilidade de utilização tanto no Android como no iOS. A Figura 3.2 demonstra a interface do Jupyter utilizada na execução do GuideAutomator Mobile. A porção da imagem representada pelo número “1” na imagem corresponde ao bloco de texto

do Jupyter que recebe texto escrito em linguagem de marcação Markdown; A parte representada pelo número “2” é área responsável pela entrada de código na linguagem aceita pelo GuideAutomator Mobile; E a parte “3” exibe a saída processada a partir do código interpretado na seção “2”.

### 3.3 Guide Automator Video

O GuideAutomator Video [16] foi a primeira versão após a criação do GuideAutomator original a propor a criação de vídeo-tutoriais. Este algumas vezes necessitava da adição de um atraso em milissegundos nas suas ações a fim de que esperasse a execução de uma ação anterior. Já o Guide Automator Video JS desde o começo foi proposto com o objetivo de ser mais simples para que usuários com pouco conhecimento técnico fossem capazes de operá-lo sem qualquer dificuldade. A sintaxe de seu script é mais simples se comparado à sintaxe do script do GuideAutomator Video. Outras funcionalidades como a geração de documentação PDF e narração de texto são funcionalidades presentes em ambos. O Guide Automator Video JS foi proposto como solução às deficiências que o Guide Automator Video possui ainda que seja fortemente baseado em suas funcionalidades.

### 3.4 Ari

```

1 speech = c(
2   "I will now perform part of Mercutio's speech from Shakespeare's Romeo
3     and Juliet.",
4   "O, then, I see Queen Mab hath been with you.
5     She is the fairies' midwife, and she comes
6     In shape no bigger than an agate-stone
7     On the fore-finger of an alderman,
8     Drawn with a team of little atomies
9     Athwart men's noses as they lies asleep;")
9 mercutio_file = "death_of_mercutio.png"
10 mercutio_file2 = "mercutio_actor.png"
11
12 shakespeare_result = ari_spin(
13   c(mercutio_file, mercutio_file2),
14   speech, output = "romeo.mp4", voice = "Joanna")
15 isTRUE(shakespeare_result)

```

Figura 3.3: Exemplo de código em linguagem R utilizando a biblioteca Ari

Ari [13] é uma biblioteca feita para a utilização na linguagem R. Esta possui uma abordagem diferente do Guide Automator Video JS uma vez que a captura das imagens e vídeos não é realizada a partir de um navegador web e sim, o próprio usuário que configura sua execução e é quem define quais imagens e arquivos html serão adicionadas para a criação de um vídeo com áudio como em um clipe musical. Este implementa algumas funcionalidades também implementadas pelo Guide Automator Video JS como narração através de síntese de voz e a adição de legendas. O grande diferencial proposto pelo Guide Automator JS é realizar essas configurações de forma transparente a partir da leitura do script escrito pela linguagem aceita pelo Guide Automator JS. A biblioteca Ari configura esses recursos de forma manual. A Figura 3.3 demonstra um trecho de código de exemplo de utilização da biblioteca retirado do artigo “Ari: The Automated R Instructor” [17], que produz um vídeo simples com duas imagens, cujos nomes são “death\_of\_mercutio.png” e “mercutio\_action.png”, e uma voz sintetizada que reproduz o texto representado nas linhas 2 a 9. Na linha 14 é possível observar o nome do arquivo de saída, “romeo.mp4”, e o modelo de voz utilizado cujo nome é “Joanna”.

### 3.5 Quadro comparativo entre as ferramentas

A seguir é demonstrado um quadro comparativo com as funcionalidades presentes em cada um das ferramentas descritas neste capítulo. Cada uma das colunas têm as respectivas descrições:

- Manual: Gera manual em formato PDF como saída;
- Vídeo: Gera vídeo em formato MP4 como saída;
- *Web*: Tem como alvo aplicações *Web*;
- *Mobile*: Tem como alvo aplicações *Mobile*;
- Interface *Web*: Possui interface *Web* para a criação dos scripts;
- *Plugin*: Possui *plugin* que grava ações do navegador geradas a partir da interação do usuário.

	Manual	Vídeo	Web	Mobile	Interface Web	Plugin
Ari		X	X			
GuideAutomator	X		X			
GuideAutomator Mobile	X			X	X	
GuideAutomator Video	X	X	X			
Guide Automator Video JS	X	X	X			X



# Capítulo 4

## Guide Automator Video JS

### 4.1 Execução

A execução do Guide Automator Video acontece da seguinte forma: é lido um arquivo no formato Markdown (.md) [14], como o exibido na Figura 4.1, que interpreta os blocos de código contidos que se refletem em ações executadas na geração da documentação em texto ou da documentação em vídeo. Os blocos de código no Markdown aceitam somente um comando por linha; cada comando é executado individualmente e somente é executado após a finalização do comando anterior caso exista. Conseqüentemente, cada bloco também só inicia a sua execução caso o anterior tenha finalizado. Ao final da execução haverá duas saídas: um arquivo texto com imagens em formato PDF e um arquivo de vídeo em formato MP4. O vídeo resultante da execução contém legendas e narração criada através de síntese de voz.



Figura 4.1: Exemplo de sintaxe aceita pelo Guide Automator Video JS (esquerda) e respectiva saída (direita)

### 4.1.1 Parâmetros de execução

Os parâmetros de execução são invocados ao executar o Guide Automator Video JS da seguinte forma:

```
node main.js \
  [ -i valor | -f valor | -o valor | -r valor | -cv valor | -v | -d | -id ]
```

- Parâmetro -i: Parâmetro obrigatório. Designa o arquivo md de entrada.
- Parâmetro -f: Opcional. Designa o prefixo dos nomes dos arquivos de saída. A parte do nome do arquivo sem a extensão.
- Parâmetro -o: Opcional. Define o caminho da pasta de saída.
- Parâmetro -r: Opcional. Define o caminho da pasta onde estarão os arquivos que serão utilizados na geração do arquivo PDF, como arquivos CSS, por exemplo.
- Parâmetro -cv: Parâmetro obrigatório. Define o caminho do arquivo HTML que servirá como capa do documento PDF.
- Parâmetro -v: Opcional. Define a execução como verbosa. Ou seja: imprime no console o resultado da execução de cada comando no console.
- Parâmetro -d: Opcional. Ativa o modo debug que exibe mensagens internas do Guide Automator Video JS.
- Parâmetro -id: Opcional. Ativa o modo “integration debug” que exibe saída da execução de ferramentas externas utilizadas pela aplicação como o FFMPEG.

### 4.1.2 Comandos

Esta seção descreve os comandos aceitos pelo script .md interpretado pelo Guide Automator Video JS.

- viewport(width, height): Ajusta as dimensões da janela do navegador para as dimensões passadas como entrada em pixels. Criado para manter a compatibilidade com a resolução original a qual o script foi gerado.
- goToPage(url): Navega até o endereço passado como parâmetro.
- screenshot(legenda) / screenshot(seletor, legenda): Retira uma foto da tela e a insere no arquivo PDF com a legenda especificada. Há também a possibilidade de passar um parâmetro contendo o seletor CSS da área que se deseja fotografar. A Figura 4.2 demonstra um exemplo de saída do comando “screenshot”.



Figura 4.2: Exemplo de imagem extraída através do comando “screenshot” no arquivo PDF de saída

- `fillField(seletor, texto)`: Insere o texto passado como parâmetro ao campo de texto identificado pelo seletor CSS conforme Figura 4.3.
- `submitForm(seletor)`: Aciona o envio de um formulário identificado pelo seletor CSS.
- `click(seletor)`: Clica num elemento DOM identificado através do seletor CSS inserido como parâmetro conforme Figura 4.4.
- `select(seletor, valor)`: Seleciona um valor passado como parâmetro presente num combo identificado através do seletor CSS conforme Figura 4.5.
- `speak(texto)`: Adiciona uma legenda seguida de um áudio sintetizando uma voz correspondente ao texto inserido por parâmetro conforme Figura 4.6.

## 4.2 Guide Automator Recorder

O Guide Automator Recorder é uma extensão criada a partir do Headless Recorder [11]. O Headless Recorder [11] é um plugin criado para o navegador Chrome que grava as ações do usuário no navegador com o objetivo de gerar um script escrito em linguagem Javascript para ser usado no Puppeteer [4]. Através da utilização do Headless Recorder [11] usuários leigos são capazes de criar um script para o Puppeteer sem ter conhecimento avançado de programação. A partir de alterações realizadas no código fonte do Headless Recorder [11] foi possível fazê-lo gerar scripts com os comandos aceitos

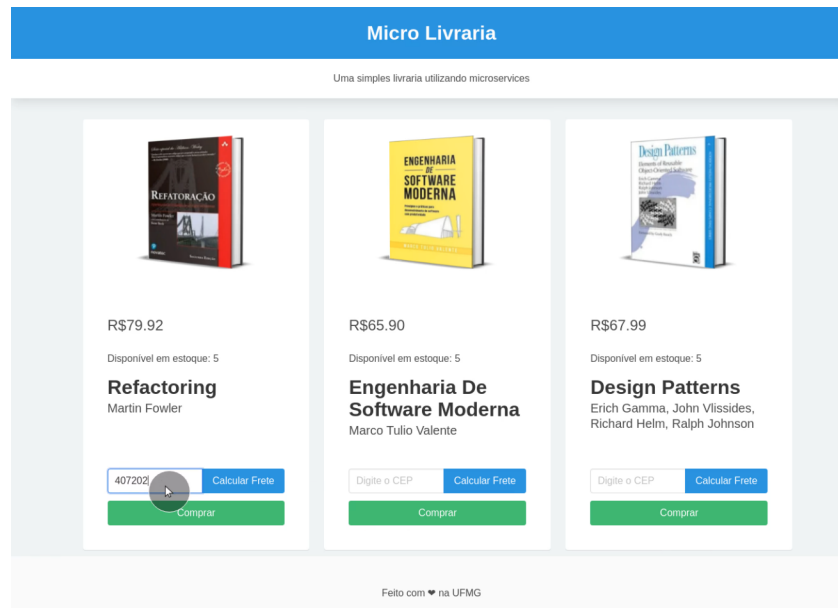


Figura 4.3: Exemplo de texto inserido no campo de texto através do comando “fillField”

pelo Guide Automator Video JS. Esta versão modificada do Headless Recorder [11] foi batizada de Guide Automator Recorder.

Para iniciar a gravação das ações de interação com o navegador deve-se clicar no botão “Record” (Figura 4.7).

Na Figura 4.8 é possível observar o Guide Automator Recorder gravando as ações de clique, e navegação entre as telas. É possível pausar a gravação clicando em “Pause” ou finalizá-la clicando em “Stop”.

É possível adicionar o comando para extrair imagens ao script a partir do menu de contexto clicando com o botão direito do mouse na tela (Figura 4.9). Para capturar a tela inteira, deve-se clicar na opção “Take a screenshot”. Para capturar a imagem de um elemento da tela que deve ser clicado após selecionada a opção, deve-se clicar em “Take a screenshot selector”.

Após o final da gravação, conforme Figura 4.10, clica-se em “Copy to clipboard” para copiar o código gerado e em seguida “Restart” se o usuário optar por iniciar uma nova gravação.

### 4.3 Implementação

O Guide Automator Video JS é uma aplicação escrita em linguagem Javascript, sendo executado sobre ambiente de execução node.js [9] e utiliza o Puppeteer [4] como ferramenta de automação. A escolha pelo Puppeteer [4] se deu por sua implementação ser

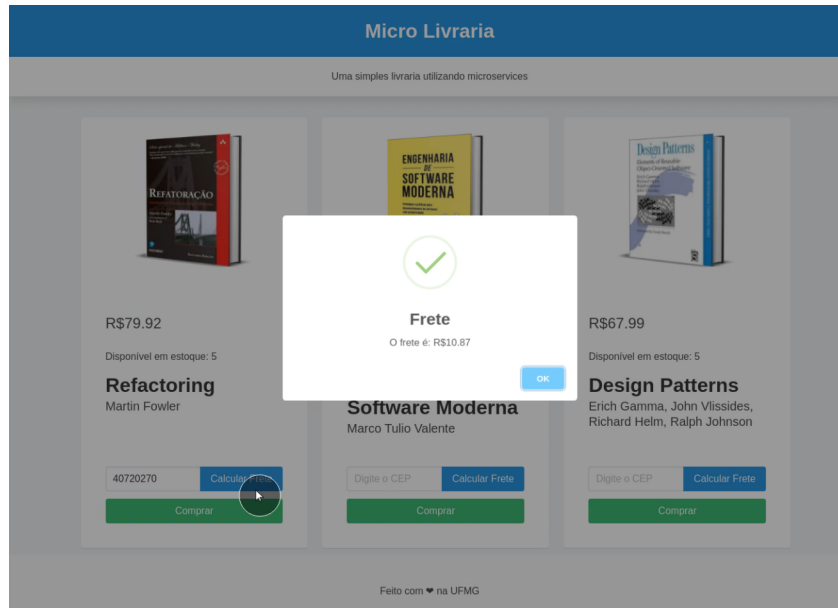


Figura 4.4: Modal aparece após ação do comando “click” no botão “Calcular Frete”

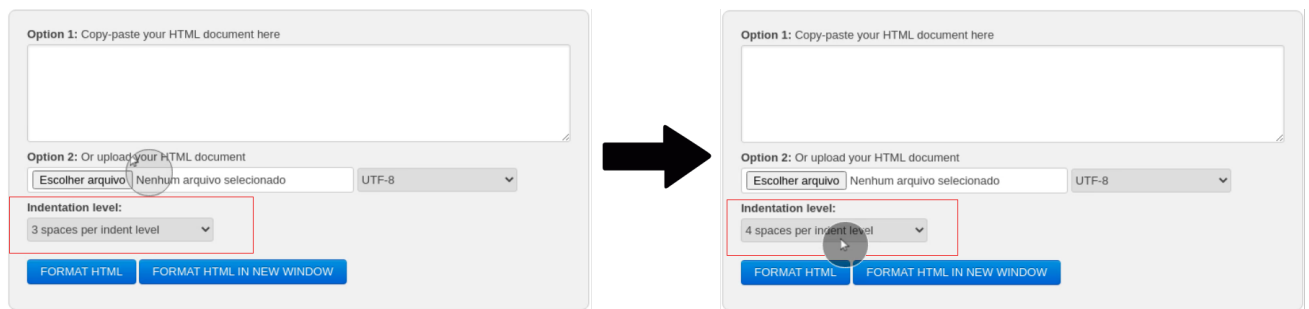


Figura 4.5: Alteração do valor do campo “indentation level” após ação do comando “select”

totalmente compatível e nativa do Javascript. Usar o Selenium [8] não era uma opção por causa dos problemas encontrados anteriormente no desenvolvimento do Guide Automator Video, como por exemplo, dada a complexidade da implementação que força a espera do carregamento de uma página após um clique em um link ou botão. O Puppeteer [4] é capaz de tratar esse tipo de adversidade utilizando o parâmetro “waitUntil” na chamada “waitForNavigation” após invocar eventos como “submit” e “click” ou utilizando o mesmo parâmetro ao executar o comando “goto”. Cada valor possível de atribuir ao parâmetro corresponde respectivamente a uma heurística que identifica se ainda há conteúdo a ser carregado na página assim eliminando a necessidade de cronometrar o tempo de carregamento da página para que todos elementos tenham sido carregados antes de iniciar uma nova ação. Os valores possíveis são:

- load: Considera o carregamento da página finalizado quando o evento “load” é acionado;

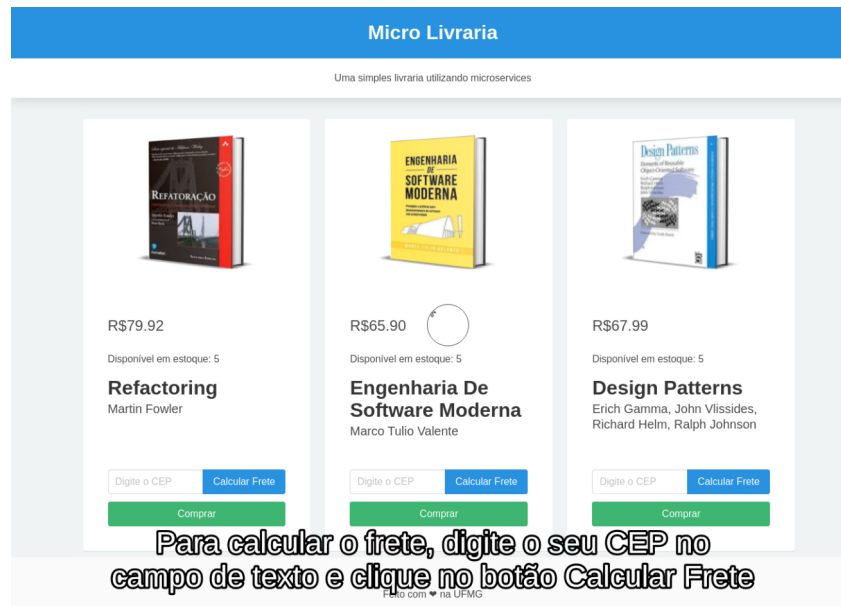


Figura 4.6: Exemplo de legenda gerada pelo comando “speak”

- `domcontentloaded`: Considera o carregamento da página finalizado quando o evento “`domcontentloaded`” é acionado;
- `networkidle0`: Considera o carregamento da página finalizado quando não há nenhuma requisição dentro do intervalo de 500 milissegundos;
- `networkidle2`: Considera o carregamento da página finalizado quando não há mais que duas requisições dentro do intervalo de 500 milissegundos.

O valor para o parâmetro que melhor se encaixou no contexto de execução do Guide Automator Video JS foi o “`networkidle2`” pois algumas aplicações enviam requisições assíncronas após o clique em determinados componentes. Aguardar até que duas requisições não sejam enviadas dentro do intervalo de 500 milissegundos é a opção que melhor abrange os diversos sistemas web disponíveis atualmente.

### 4.3.1 Ferramentas e bibliotecas utilizadas

O Guide Automator Video JS foi criado utilizando as seguintes bibliotecas e ferramentas:

- `node.js v12.15.0 LTS`: Lenon [18] define o `node.js` [9] como um “*ambiente de execução Javascript server-side*”. Isso significa que o `node.js` é um ambiente que executa algoritmos escritos utilizando a linguagem Javascript do lado do servidor. A versão `v12.15.0 LTS` foi utilizada por se tratar de uma versão LTS (Long-term Support).

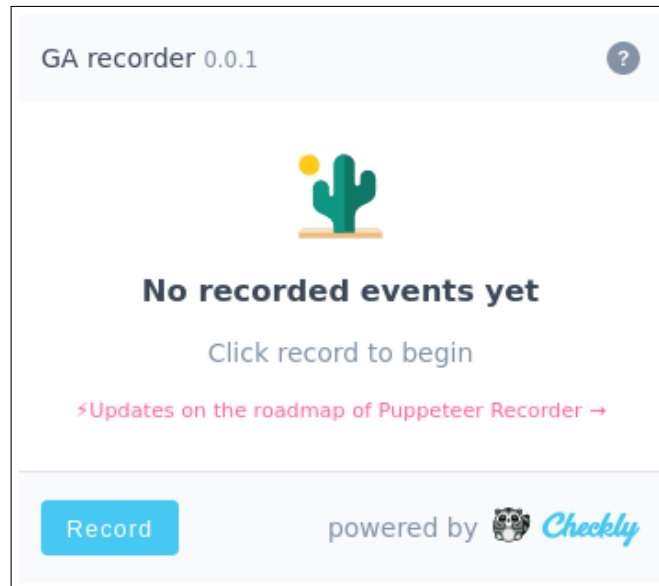


Figura 4.7: Tela inicial do Guide Automator Recorder

Portanto é uma versão estável e que terá maior suporte por parte de seus desenvolvedores. A maior parte da aplicação foi escrita em Javascript.

- Puppeteer: O Puppeteer [4] é uma biblioteca criada para ser executada no node.js [9] e provê uma API para a manipulação dos navegadores Chrome ou Chromium. Foi utilizado não só para a navegação e interação com as páginas através do script executado pelo Guide Automator Video JS, mas também para a gravação do vídeo.
- Image-to-base64: ImageToBase64 [19] é uma biblioteca que facilita a codificação de arquivos de imagem para base64. É utilizada para adição de imagens ao documento PDF de saída.
- Markdown-it: O Markdown-it [20] converte o conteúdo escrito em formato markdown para HTML para que depois seja transformado em PDF com o Wkhtmltopdf.
- Node-apng: O Node-apng é responsável por concatenar todas imagens capturadas em vídeo no formato APNG.
- Wkhtmltopdf: O Wkhtmltopdf [21] transforma o HTML resultante do processamento do Markdown-it em PDF.
- FFMPEG: O FFMPEG [22] é uma ferramenta de código aberto que condifica e decodifica mídias de áudio e vídeo nos mais diferentes formatos disponíveis. No Guide Automator Video JS é responsável pela conversão do vídeo APNG para MP4 e geração e concatenação dos áudios.

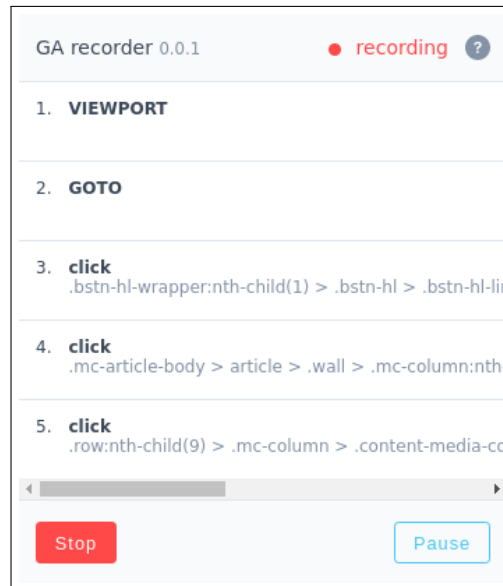


Figura 4.8: Tela exibindo gravação iniciada

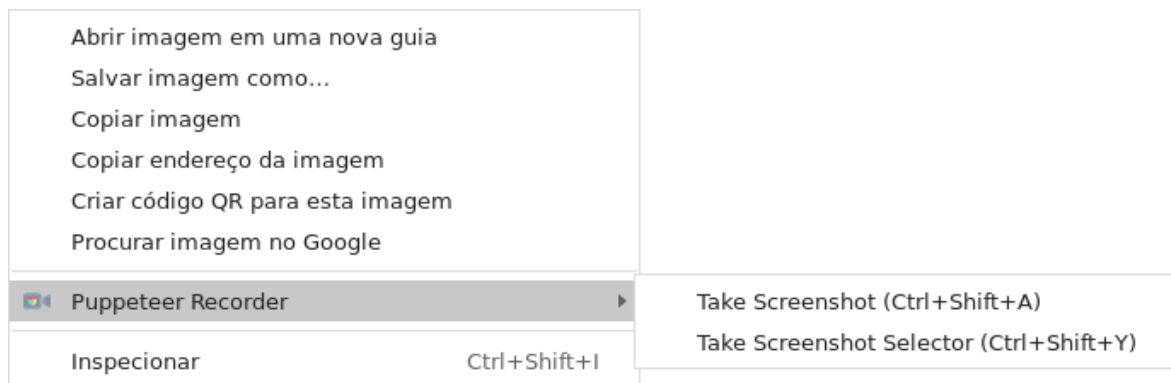


Figura 4.9: Menus de contexto para inserção comandos de screenshot no script

- **Espeak:** O Espeak [23] é um utilitário open-source cuja função é transformar os textos passados para a função “speak” do Guide Automator Video JS em voz através da tecnologia de síntese de voz.

A Figura 4.11 demonstra como acontece a integração entre as ferramentas que o Guide Automator Video JS utiliza. O funcionamento ocorre da seguinte forma:

- Carrega o arquivo Markdown em memória e extrai os blocos de código a partir do arquivo;
- Inicia a extração dos frames;
- Interpreta cada bloco de código;
  - Se houver o comando “screenshot” contido no bloco, a imagem, codificada em base64 utilizando a biblioteca “image-to-base64”, é substituída pelo bloco no





Figura 4.10: Tela exibindo gravação finalizada

arquivo Markdown carregado em memória;

- O comando “Speak”, quando invocado, executa a ferramenta “Espeak” responsável pela síntese de voz;
- Ao final da execução do script, o arquivo Markdown processado é convertido para HTML utilizando a biblioteca “Markdown-it” e a extração dos frames é finalizada;
- O arquivo PDF é gerado utilizando a biblioteca “Wkhtmltopdf”;
- Os frames são unidos e transformados em vídeo no formato APNG;
- As legendas no formato SRT, geradas durante a execução dos blocos de código, vídeo e áudio gerados são unidos utilizando o “FFMPEG”, e é gerada a saída em formato MP4.

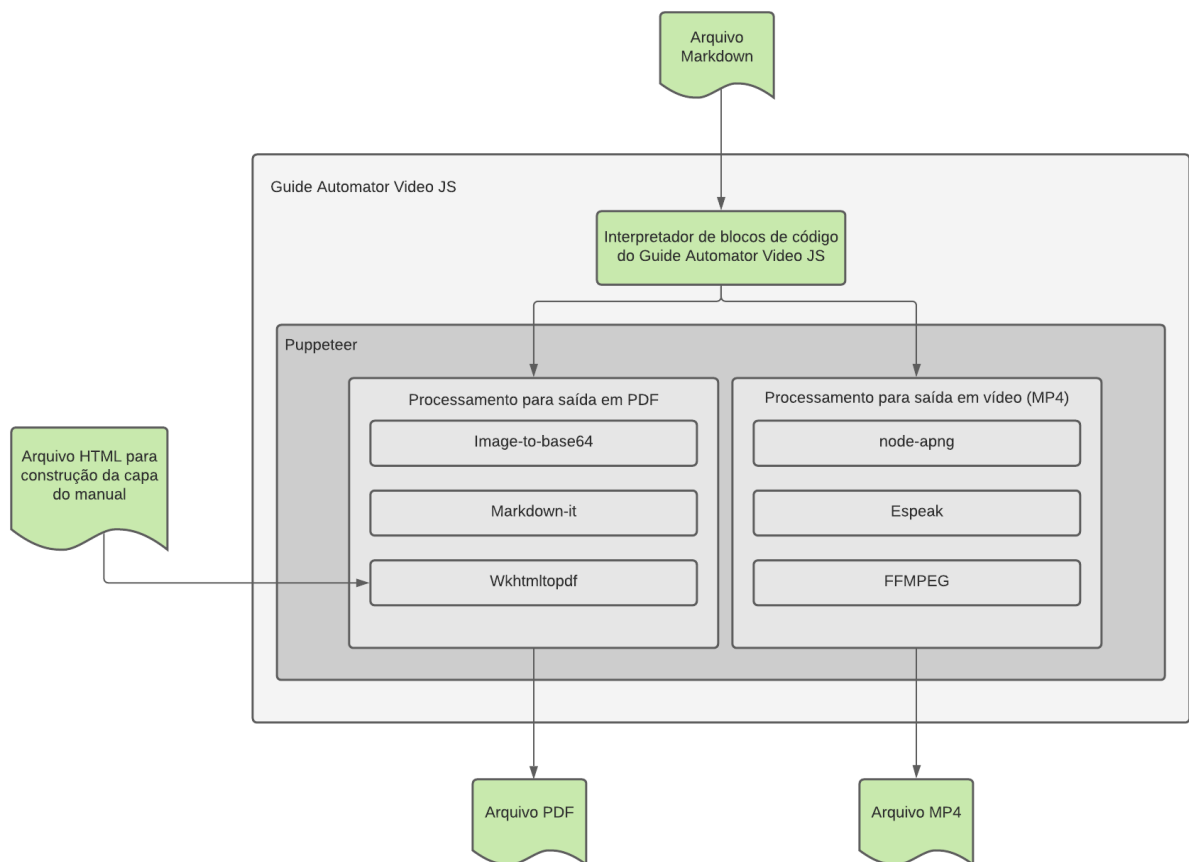


Figura 4.11: Arquitetura do Guide Automator Video JS

# Capítulo 5

## Avaliação

Neste capítulo serão abordadas as etapas percorridas para a implementação do Guide Automator Video JS em uma arquitetura real de Sistema Web que se vale da integração contínua para implantação dos artefatos nos ambientes de desenvolvimento, homologação ou produção.

### 5.1 Descrição da aplicação utilizada nos testes

Para avaliar a utilidade do Guide Automator Video JS foi utilizada para os testes a aplicação Micro Livraria [1] (Figura 5.2). A Micro Livraria foi desenvolvida pelo professor Dr. Marco Túlio Valente do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, com o propósito do ensino da disciplina de Engenharia de Software ministrada por ele. Sua estrutura é bem simples e conta apenas com uma tela para a ação de cálculo de frete e também compra de um livro. Uma vez que a Micro Livraria foi puramente criada para o ensino, as ações de cálculo de frete e compra são simulações e não realizam quaisquer verificações ou venda de fato. Para que fosse viável os testes, os critérios para escolha da aplicação foram ter código-fonte disponível, possuir documentação sobre como implantar e executar, e ser de rápida implantação. Por ser simples, a aplicação Micro Livraria cumpre totalmente com as exigências dessa prova de conceito.

Há um serviço para a consulta dos livros e outro para a consulta do valor de frete. Estes dois são serviços gRPC — serviços executados através de chamada remota de procedimento que utilizam o framework gRPC — e são consumidos pela aplicação backend que recebe requisições da aplicação frontend. Os dois serviços são executados em portas diferentes assim como o backend e o frontend. O esquema de arquitetura pode ser verificado na Figura 5.1.

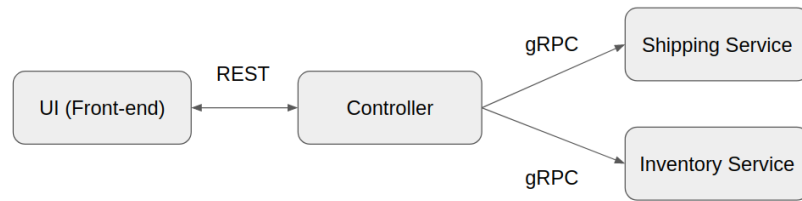


Figura 5.1: Arquitetura da aplicação Micro Livraria

## 5.2 Execução do Guide Automator Video JS

Foi criada uma imagem Docker, que possibilita a execução do Guide Automator Video JS no serviço de integração contínua do GitHub, o “GitHub Actions”, cuja configuração está disponível no arquivo “Dockerfile” (Anexo 4) dentro do repositório oficial do Guide Automator Video JS [24] no GitHub. Essa imagem foi disponibilizada tanto no repositório de pacotes do GitHub (GitHub Packages) como também no repositório oficial de pacotes do Docker, o DockerHub. É através dessa imagem que todas as ações necessárias são executadas para a geração dos arquivos PDF e MP4 que são as saídas do Guide Automator Video JS. A imagem é gerada todas as vezes que a branch principal do repositório do Guide Automator Video JS é atualizada através da ação do GitHub Actions configurada na pasta “.github/workflows” presente no repositório, no arquivo “github-package-publish.yml” (Anexo 5). Há ainda no DockerHub há uma ação configurada para que seja gerada e também feito o upload da imagem no DockerHub.

Para que fosse possível a execução do Guide Automator Video JS através do GitHub Actions foi criada uma pasta chamada “.guide-automator” dentro do repositório da Micro Livraria [25] que foi duplicado a partir do repositório original no GitHub, onde foram colocados todos os arquivos necessários:

- cover.html (Anexo 1): Arquivo em HTML que define como será a capa da documentação em formato PDF.
- livraria.md (Anexo 2): Arquivo em formato Markdown que define a estrutura geral do arquivo PDF e também é responsável pelas ações executadas pelo Guide Automator Video JS.

Dentro da pasta “.github/workflows” foi configurado o arquivo “guide-automator-video-publish.yml” (Anexo 3) cuja responsabilidade é de executar o Guide Automator Video JS e realizar a ação de upload no GitHub Releases através do GitHub Actions.

Toda a execução acontece da forma explicitada na Figura 5.3: É realizada a ação de push na branch principal do repositório que é identificada pelo GitHub Actions e este por sua vez inicia a execução do passo-a-passo de execução contido no

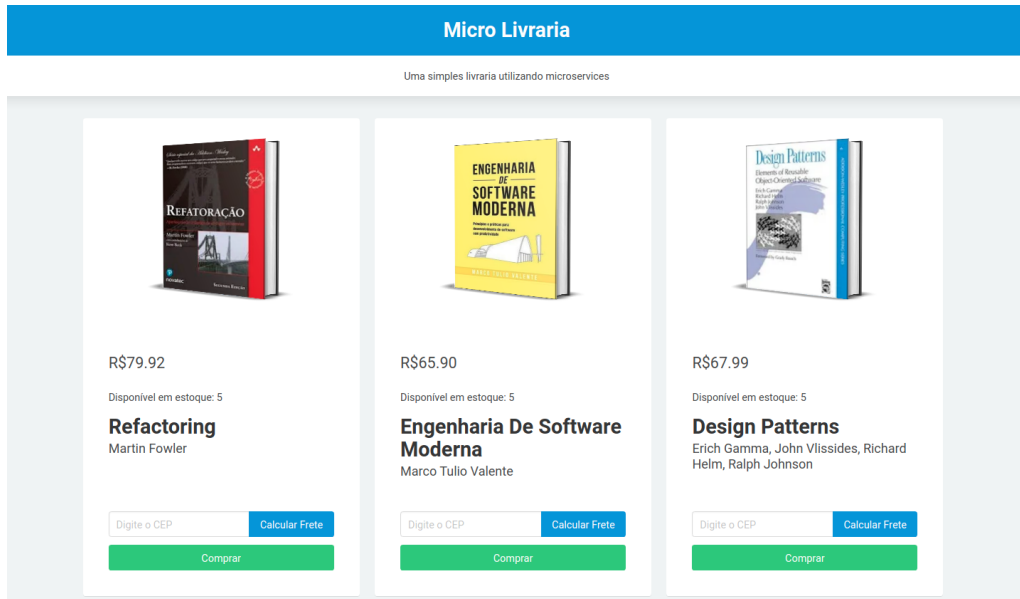


Figura 5.2: Tela principal da aplicação Micro Livraria

arquivo “guide-automator-video-publish.yml”. Primeiro o IP do host é capturado e é substituído no arquivo de execução do Guide Automator Video JS, o arquivo “.guide-automator/livraria.md”. Em seguida a imagem Docker da aplicação Micro Livraria é construída e executada expondo as portas 5000 e 3000 para que seja possível a chamada aos serviços por parte da aplicação frontend. Os últimos passos são a execução da Imagem Docker do Guide Automator Video JS hospedada no Dockerhub que executará todas as ações configuradas de interação com a página web e ao final da execução será realizado o upload dos arquivos PDF (Anexo 6) e MP4 gerados.

### 5.3 Resultados obtidos

Os arquivos foram disponibilizados na seção “releases” do próprio repositório da aplicação Micro Livraria como um artefato da *release*. O que prova que a solução de integração contínua para geração de vídeo e documentação é satisfatória. Um ponto que pode ser melhorado é o tempo de preparação da imagem que dura 3 minutos. A execução total da preparação da imagem e execução do Guide Automator Video JS dura 5 minutos e 58 segundos. Durante os testes, o tempo de execução diminuía conforme aconteciam execuções consecutivas. Obviamente todo o processo de geração de vídeo influi bastante para o tempo total de execução. Porém, ao acompanhar a execução das etapas é evidente que os tempos de download e construção das imagens do Docker são as etapas mais demoradas. Uma vez que o tamanho total da imagem é de 931.14 Megabytes, tamanho relativamente grande para uma imagem do Docker, há de se avaliar o conteúdo

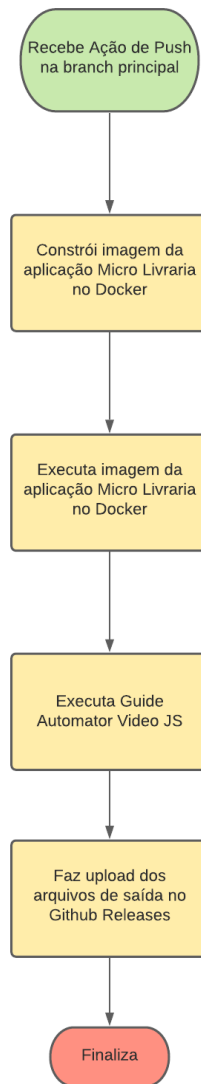


Figura 5.3: Fluxograma da solução proposta

do script que a gera a fim de encontrar possíveis soluções para a diminuição de tamanho e consequente diminuição do tempo de preparação da imagem já que boa parte do tempo desta é atribuído ao download da imagem.

## 5.4 Limitações

Dada as limitações da ferramenta de captura de vídeo do Puppeteer [4], que por sua vez utiliza a API de captura de frames Screencast do Chromium, o vídeo gerado sofre com congelamentos que variam em intensidade de acordo com a movimentação de elementos na tela e velocidade de conexão com a internet. Quanto menos elementos se movem na página configurada no script e também quanto pior for a qualidade de conexão, mais problemas de fluidez aparecem na exibição do vídeo. Estes travamentos geram outros problemas

como sincronização de áudio e legendas do vídeo já que o FFMPEG tenta construir o vídeo a taxa constante de frames por segundo enquanto os frames são disponibilizados a taxa variável conforme os fatores que influenciam na qualidade descritos.

# Capítulo 6

## Conclusão

O Guide Automator Video JS foi idealizado com o objetivo de gerar automaticamente documentação a partir de aplicações web, sanando as deficiências de sua versão anterior e também gerando vídeo com legenda e síntese de voz. Este trabalho deixa como contribuição opções para implementação e relata os desafios envolvidos na criação de ferramentas para geração automática de documentação, principalmente no contexto do paradigma de integração contínua. Para uma futura evolução é desejável melhorias na sincronização de áudio e legendas e também a adição de mais efeitos para que o vídeo seja mais atrativo para o usuário final. É desejável também a busca por uma alternativa de gravação de vídeo já que a taxa de frames por segundo disponibilizada pela API do Chromium é sensível à velocidade de conexão e também à frequência com que os elementos se movem na tela já que o mecanismo de gatilho para a disponibilização de um novo frame baseia-se nas mudanças que acontecem na exibição da página. A baixa taxa de frames acarreta em um vídeo menos fluido. Embora as limitações da biblioteca de captura de vídeo tenham dificultado a captura de vídeo e sincronização de áudio, o resultado final é satisfatório pois prova-se plausível a evolução de ferramentas do tipo pois foi possível a geração de vídeo e documento descritivo conforme relatado no capítulo de avaliação.



# Anexos

## 1 cover.html

Arquivo HTML responsável por gerar a capa da documentação gerada pelo Guide Automator Video JS.

```
1 <html>
2   <head>
3   </head>
4   <body>
5     <div style="text-align:center;">
6       <h1>DEMONSTRAÇÃO GUIDE AUTOMATOR</h1>
7       <h3>Ícaro Erasmo Souza Barreiro</h3>
8       <h3>Federal University of Bahia</h3>
9     </div>
10  </body>
11 </html>
```

## 2 livraria.md

Arquivo Markdown de entrada do Guide Automator Video JS.

```
1 # Micro Livraria UFMG
2
3 ## Introdução
4
5 Este documento demonstra a navegação através da Micro Livraria a fim de
6 exemplificar o funcionamento do Guide Automator.
7
8 ## Screenshots
9
10 viewport 1365 982
11 go-to-page http://localhost:5000
12 speak 'Olá! Este vídeo tem o objetivo de demonstrar o funcionamento da
    aplicação "Micro Livraria".'
```

```

13 screenshot "Página principal"
14   ***
15 Esta é a página principal.
16
17   ***
18 speak 'Para calcular o frete, digite o seu CEP no campo de texto e clique
19   no botão "Calcular Frete"'
20 fill-field 'div.column:nth-child(1) > div:nth-child(1) > div:nth-child(2) >
21   div:nth-child(1) > div:nth-child(2) > div:nth-child(1) > input:nth-
22   child(1)' '40720270';
23 click 'div.column:nth-child(1) > div:nth-child(1) > div:nth-child(2) > div:
24   nth-child(1) > div:nth-child(2) > div:nth-child(2) > a:nth-child(1)'
25 screenshot "Cálculo do frete"
26   ***
27 Esta é a tela de confirmação do valor do frete.
28
29   ***
30 speak 'Em seguida, clique no botão "Comprar"'
31 click '.swal-button'
32 click 'div.column:nth-child(1) > div:nth-child(1) > div:nth-child(2) > div:
33   nth-child(1) > button:nth-child(3)'
34 speak 'Compra finalizada'
35 screenshot "Compra finalizada"
36   ***
37 Esta é a tela de confirmação da Compra.

```

### 3 guide-automator-video-publish.yml

Arquivo de configuração do GitHub Actions. Responsável por executar o Guide Automator Video JS no GitHub Actions.

```

1 name: Docker
2
3 on:
4   push:
5     # Publish `master` as Docker `latest` image.
6     branches:
7       - main
8
9     # Publish `v1.2.3` tags as releases.
10    tags:
11      - v*
12
13    # Run tests for any PRs.

```

```

14 pull_request:
15
16 env:
17 # TODO: Change variable to your image's name.
18 IMAGE_NAME: guide-automator-puppeteer-image
19
20 jobs:
21   push:
22
23     runs-on: ubuntu-latest
24     if: github.event_name == 'push'
25
26     steps:
27       - uses: actions/checkout@v2
28
29       - name: run guide automator
30         run: |
31           # Captures host ip
32           hostip=$(ip a s eth0 | egrep -o 'inet
33 [0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' | cut -d' ' -f2)
34
35           # Replaces host ip into guide automator script
36           sed -i 's/http:\\\\localhost/"http:\\\\$hostip"/g .guide-
37 automator/livraria.md
38
39           # Builds micro livraria image
40           docker build -t micro-livraria .
41
42           # Runs micro livraria in background
43           docker run -d --rm -p 5000:5000 -p 3000:3000 micro-livraria
44
45           # Runs guide automator
46           docker run --rm -v $(pwd)/output:/usr/src/output -v $(pwd)/.guide
47 -automator:/usr/src/guide-automator icaroerasmo/guide-automator-
48 puppeteer -d -i /usr/src/guide-automator/livraria.md -cv /usr/src/guide-
49 automator/cover.html -o /usr/src/output
50
51       - name: Upload video
52         uses: svenstaro/upload-release-action@v2
53         with:
54           repo_token: ${ secrets.GITHUB_TOKEN }
55           file: output/video.mp4
56           asset_name: video.mp4
57           tag: ${ github.ref }
58           overwrite: true

```

```

54     body: "Micro livraria video"
55
56   - name: Upload pdf
57     uses: svenstaro/upload-release-action@v2
58     with:
59       repo_token: ${{ secrets.GITHUB_TOKEN }}
60       file: output/output.pdf
61       asset_name: output.pdf
62       tag: ${{ github.ref }}
63       overwrite: true
64       body: "Micro livraria video"

```

## 4 Dockerfile

Arquivo de configuração da imagem do Docker que executa o Guide Automator Video JS.

```

1 FROM node:12.15.0-buster
2
3 WORKDIR /usr/src/app
4
5 RUN sed -i 's/deb http:\\\\deb.debian.org\\/debian buster main/deb http:\\\\
  deb.debian.org\\/debian buster main contrib non-free/g' /etc/apt/sources.
  list
6 RUN apt update
7 RUN apt upgrade -y
8 RUN apt install -y \
9   espeak \
10  mbrola-br* \
11  ffmpeg \
12  xfonts-75dpi \
13  xfonts-base \
14  ca-certificates \
15  fonts-liberation \
16  libappindicator3-1 \
17  libasound2 \
18  libatk-bridge2.0-0 \
19  libatk1.0-0 \
20  libc6 \
21  libcairo2 \
22  libcups2 \
23  libdbus-1-3 \
24  libexpat1 \
25  libfontconfig1 \

```

```

26  libgbm1 \
27  libgcc1 \
28  libglib2.0-0 \
29  libgtk-3-0 \
30  libnspr4 \
31  libnss3 \
32  libpango-1.0-0 \
33  libpangocairo-1.0-0 \
34  libstdc++6 \
35  libx11-6 \
36  libx11-xcb1 \
37  libxcb1 \
38  libxcomposite1 \
39  libxcursor1 \
40  libxdamage1 \
41  libxext6 \
42  libxfixed3 \
43  libxi6 \
44  libxrandr2 \
45  libxrender1 \
46  libxss1 \
47  libxtst6 \
48  lsb-release \
49  wget \
50  xdg-utils
51
52  RUN export WKHTML_LATEST_VERSION=$( \
53  wget -O - "https://api.github.com/repos/wkhtmltopdf/wkhtmltopdf/
54  releases/latest" | \
55  grep '"tag_name":' | \
56  sed -E 's/.*"([~"]+)"*/\1/' && \
57  wget "https://github.com/wkhtmltopdf/packaging/releases/download/${
58  WKHTML_LATEST_VERSION}-1/wkhtmltox_${WKHTML_LATEST_VERSION}-1.
59  buster_amd64.deb" \
60  && dpkg -i wkhtmltox_${WKHTML_LATEST_VERSION}-1.buster_amd64.deb \
61  && rm wkhtmltox_${WKHTML_LATEST_VERSION}-1.buster_amd64.deb
62
63  COPY . .
64
65  RUN npm install
66
67  ENTRYPOINT ["node", "main.js"]

```

## 5 github-package-publish.yml

Arquivo de configuração do GitHub Actions responsável pelo upload da Imagem Docker no GitHub Packages.

```
1 FROM node:12.15.0-buster
2
3 WORKDIR /usr/src/app
4
5 RUN sed -i 's/deb http:\\\\deb.debian.org\\/debian buster main/deb http:\\\\deb.debian.org\\/debian buster main contrib non-free/g' /etc/apt/sources.list
6
7 RUN apt update
8 RUN apt upgrade -y
9 RUN apt install -y \
10     espeak \
11     mbrola-br* \
12     ffmpeg \
13     xfonts-75dpi \
14     xfonts-base \
15     ca-certificates \
16     fonts-liberation \
17     libappindicator3-1 \
18     libasound2 \
19     libatk-bridge2.0-0 \
20     libatk1.0-0 \
21     libc6 \
22     libcairo2 \
23     libcups2 \
24     libdbus-1-3 \
25     libexpat1 \
26     libfontconfig1 \
27     libgbm1 \
28     libgcc1 \
29     libglib2.0-0 \
30     libgtk-3-0 \
31     libnspr4 \
32     libnss3 \
33     libpango-1.0-0 \
34     libpangocairo-1.0-0 \
35     libstdc++6 \
36     libx11-6 \
37     libx11-xcb1 \
38     libxcb1 \
39     libxcomposite1 \
```

```
39 libxcursor1 \  
40 libxdamage1 \  
41 libxext6 \  
42 libxfixed3 \  
43 libxi6 \  
44 libxrandr2 \  
45 libxrender1 \  
46 libxss1 \  
47 libxtst6 \  
48 lsb-release \  
49 wget \  
50 xdg-utils  
51  
52 RUN export WKHTML_LATEST_VERSION=$(\  
53 wget -O - "https://api.github.com/repos/wkhtmltopdf/wkhtmltopdf/  
54 releases/latest" | \  
55 grep '"tag_name":' | \  
56 sed -E 's/.*"([^\"]+)"\.*/\1/' ) && \  
57 wget "https://github.com/wkhtmltopdf/packaging/releases/download/${  
58 WKHTML_LATEST_VERSION}-1/wkhtmltox_${WKHTML_LATEST_VERSION}-1.  
59 buster_amd64.deb" \  
60 && dpkg -i wkhtmltox_${WKHTML_LATEST_VERSION}-1.buster_amd64.deb \  
61 && rm wkhtmltox_${WKHTML_LATEST_VERSION}-1.buster_amd64.deb  
62  
63 COPY . .  
64  
65 RUN npm install  
66  
67 ENTRYPOINT ["node", "main.js"]
```

## 6 output.pdf

### **DEMONSTRAÇÃO GUIDE AUTOMATOR**

**Ícaro Erasmo Souza Barreiro**

**Federal University of Bahia**



**Índice**

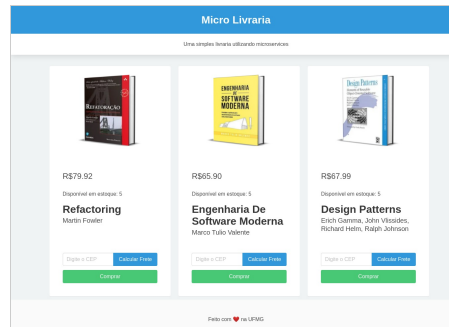
Índice	2
Micro Livraria UFMG	3
Introdução	3
Screenshots	3

# Micro Livraria UFMG

## Introdução

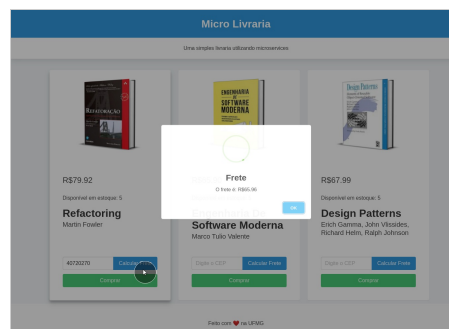
Este documento demonstra a navegação através da Micro Livraria a fim de exemplificar o funcionamento do Guide Automator.

## Screenshots



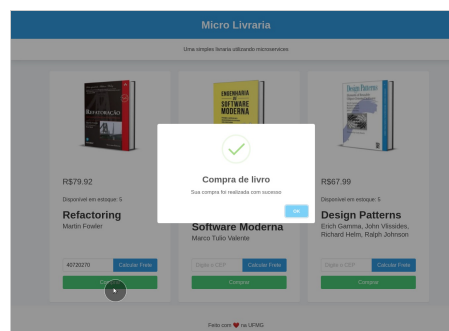
Página principal

Esta é a página principal.



Cálculo do frete

Esta é a tela de confirmação do valor do frete.



Compra finalizada

Esta é a tela de confirmação da Compra.

# Referências Bibliográficas

- [1] VALENTE, M. T. *Micro Livraria*. Acessado em 18 de Abril de 2021. Disponível em: <<https://github.com/aserg-ufmg/micro-livraria>>.
- [2] FOWLER, M. *Continuous integration*. 2006.
- [3] SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, v. 5, p. 3909–3943, 2017.
- [4] PÁGINA no Github do Puppeteer. [S.l.], Acessado em 11 de Novembro de 2020. Disponível em: <<https://github.com/puppeteer/puppeteer>>.
- [5] ZHAO, Y. et al. The impact of continuous integration on other software development practices: A large-scale empirical study. In: *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. [S.l.: s.n.], 2017. p. 60–71.
- [6] SCHIEMANN, D.; COURIOL, B. *JavaScript and Web Development InfoQ Trends Report 2020*. Acessado em 12 de Junho de 2021. Disponível em: <<https://www.infoq.com/articles/javascript-web-development-trends-2020/>>.
- [7] PÁGINA do Selenium. [S.l.], Acessado em 27 de Abril de 2021. Disponível em: <<https://www.selenium.dev/>>.
- [8] DOCUMENTAÇÃO oficial do Selenium - Navegadores Suportados. Acessado em 8 de Maio de 2021. Disponível em: <[https://www.selenium.dev/documentation/en/getting\\_started\\_with\\_webdriver/browsers/](https://www.selenium.dev/documentation/en/getting_started_with_webdriver/browsers/)>.
- [9] PÁGINA do Node JS. Acessado em 16 de Junho de 2021. Disponível em: <<https://nodejs.org/en/>>.
- [10] VAIDYA, N. *Puppeteer vs Selenium: Core Differences*. Acessado em 24 de Maio de 2021. Disponível em: <<https://www.browserstack.com/guide/puppeteer-vs-selenium>>.

- [11] PÁGINA do Headless Recorder. Acessado em 16 de Junho de 2021. Disponível em: <<https://github.com/checkly/headless-recorder>>.
- [12] OLIVEIRA, A. dos S. *Guide Automator: Automated User Manual Generation with Markdown*. 2016. Monografia (Bacharel em Ciência da Computação), UFBA (Universidade Federal da Bahia), Salvador, Brazil.
- [13] PÁGINA da biblioteca Ari. [S.l.], Acessado em 18 de Abril de 2021. Disponível em: <<https://cran.r-project.org/web/packages/ari/index.html>>.
- [14] MARKDOWN Guide - Getting Started. Acessado em 8 de Maio de 2021. Disponível em: <<https://www.markdownguide.org/getting-started/>>.
- [15] SILVA, A. A. da. *Desenvolvimento e Avaliação do Guide Automator Mobile*. 2018. Monografia (Bacharel em Sistemas de Informação), UFBA (Universidade Federal da Bahia), Salvador, Brazil.
- [16] SOUZA, I. S. A. *Guide Automator Video: Gerador de vídeo tutoriais para aplicações Web*. 2019. Monografia (Bacharel em Sistemas de Informação), UFBA (Universidade Federal da Bahia), Salvador, Brazil.
- [17] KROSS, S.; LEEK, J. T.; MUSCHELLI, J. *Ari: The Automated R Instructor*. Acessado em 13 de Junho de 2021. Disponível em: <[https://johnmuschelli.com/ari\\_paper/](https://johnmuschelli.com/ari_paper/)>.
- [18] LENON. *Node.js – O que é, como funciona e quais as vantagens*. [S.l.], 2018. Disponível em: <<https://www.opus-software.com.br/node-js/>>.
- [19] BASTOS, R. *Página no Github do Image-to-base64*. [S.l.], Acessado em 11 de Novembro de 2020. Disponível em: <<https://github.com/renanbastos93/image-to-base64>>.
- [20] IT, M. *Página no Github do Markdown-it*. [S.l.], Acessado em 11 de Novembro de 2020. Disponível em: <<https://github.com/markdown-it/markdown-it>>.
- [21] PÁGINA do Wkhtmltopdf. [S.l.], Acessado em 11 de Novembro de 2020. Disponível em: <<https://wkhtmltopdf.org/>>.
- [22] PÁGINA do Ffmpeg. [S.l.], Acessado em 11 de Novembro de 2020. Disponível em: <<https://ffmpeg.org/>>.
- [23] PÁGINA do Espeak. Acessado em 10 de Maio de 2021. Disponível em: <<http://espeak.sourceforge.net/>>.
- [24] REPOSITÓRIO do Guide Automator Video JS. Acessado em 19 de Maio de 2021. Disponível em: <<https://github.com/icaroerasmo/guide-automator-puppeteer>>.

- [25] FORK do repositório do Guide Automator Video JS. Acessado em 19 de Maio de 2021. Disponível em: <<https://github.com/icaroerasmo/micro-livraria>>.