SoftwareX Article Template

Web Generator: an open-source software for synthetic web-based user interface dataset generation.

Andrés Soto[1,2], Dagoberto Mayorca[1,2,3], Héctor Mora[1,2], Jaime A. Riascos[1,2,3]
[1]Corporación Universitaria Autónoma de Nariño, Pasto, Colombia.
[2]SDAS Research Group, Pasto, Colombia.
[3]Universidad Mariana, Pasto, Colombia.
agsoto@protonmail.com

Abstract.
Recently, Machine Learning algorithms are employed to automate several processes, including software development. However, this action demands large datasets for training these algorithms. To our knowledge, there is no tool for generating synthetic datasets that contain HTML objects (interfaces, codes, wireframe.) Thus, we present the Web Generator, a software designed to mainly provide web pages, designs, and content based on the Bootstrap frontend framework. The software delivers markup code, screenshots, and labels for web elements. With this software, we aim to generate enough material for training and exploring the Machine Learning approach for automatically web design and development.

Keywords:
Synthetic dataset; Dataset generation; HTML; Bootstrap;

Required Metadata

Current code version

Table 1 – Code metadata (mandatory)

Nr	Code metadata description	***Please fill in this column***
C1	Current code version	*0.1*
C2	Permanent link to code/repository used of this code version	*https://github.com/agsoto/webgenerator*
C3	Code Ocean compute capsule	*None*
C4	Legal Code License	*GPL v3*
C5	Code versioning system used	*git*
C6	Software code languages, tools, and services used	*Python, Javascript, CSS.*
C7	Compilation requirements, operating environments & dependencies	*Requirements:* - *Chromium / Chrome Browser > 80.0* - *Selenium Web Driver for Chromium = Browser version* - *Python >= 3.7* - *Pip >= 20.0.2*

		Dependencies: - *selenium = 3.141.0* - *colorharmonies = 1.0.5* - *dominate = 2.4.0* - *utils = 1.0.1* - *python_lorem = 1.1.2* - *palettable = 3.3.0* - *webdriver_manager = 2.3.0* - *libsass = 0.20.1* - *Pillow = 7.2.0* - *Selenium-Screenshot*
C8	If available Link to developer documentation/manual	
C9	Support email for questions	*agsoto@protonmail.com*

1. Motivation and Significance

One of the critical aspects of the software and web development process is the design and implementation of Graphical User Interfaces (GUIs). These interfaces act like bridges between users and the computer actions, integrating the content consumed and analyzed by the user [1]. Given the visual nature of GUIs, the design should consider objects distribution, color theory, legibility, among others. Additionally, GUIs require an understanding of human psychology to determine the best possible interaction modes, which concerns the field of Human-Computer Interaction [2]. Developers usually code interfaces in mark-up languages such as HTML or XML, which requires a deep understanding of the instructions that can produce a visible result. In short, the user interface design and implementation are costly tasks in software development [3].

In the latest years, there have been different strategies to assist developers in the GUI design and implementation to minimize the production costs. Indeed, ongoing research uses algorithms to automate or facilitate the codification of interfaces [4,5,6]. One of the most innovative approaches is to use Deep Learning, evidencing promising results related to fully automated code generation based on the interface's mockup or sketch. However, one limitation is the dataset availability because these algorithms usually require significant amounts of data to produce good results [7]. Thus, this software aims to help web interface generation research by providing parametric algorithms that can generate on-demand web pages with features and labels ready to use as datasets. Despite the vast number of websites available, the particular frameworks, structure, technology, and general characteristics make it challenging to discretize particular coded files and design features. Therefore, using publicly available web pages as datasets requires a comprehensive effort mining and preprocessing the data.

As we mentioned before, it is necessary to use datasets to train and evaluate interface generation methods. Thus, we reviewed some datasets, especially publicly available, for research in web content and interface generation to confirm a synthetic data generation tool's relevance. Firstly, the RICO dataset [8] is one of the most consistent datasets. It is composed of authentic mobile interfaces screenshots with a component's hierarchy, interaction traces, and metadata. This dataset was generated

using data mining tools in several mobile apps from the Google Playstore to produce Data-Driven Design Applications. Nevertheless, it is not suitable for web interface experimentation. Also, Pix2Code [7] provides a dataset that contains different interface elements, but its nature requires a Domain Specific Language (DSL), limiting the number of elements and dispositions in his content severely.

Similarly, in the REDRAW paper [9], the authors used a dataset generated through data mining and stated the public release of it, but after two years of the paper's publication, it is not available yet. Finally, Sketched-Webpages-Generator [10] is the more similar software to ours. It focuses on generating sketches rather than HTML code. For this reason, it makes use of a very different approach, aiming to provide the variations presented in sketching and not in the distribution of elements or styling. The named software could be used in a complementary way, merging functionalities at the code level.

Consequently, we developed a tool to generate datasets at will. We keep code level execution focus for the first version to maximize flexibility until future features are developed. The software is provided with a working example of execution ready to tweak the parameters, showing that the user should only provide an instance of the WebLayoutProbabilities with corresponding values and an instance of WebGenerator with those probabilities as parameters. Finally, calling the Generate method to produce a single result.

2. Software Description

The primary generation focus is the mark-up code, not the styling itself. For that reason, we used the popular front-end framework Bootstrap [11] to adopt the ready-to-use styling classes. We tried to keep custom Cascade Styling Sheets (CSS) to a minimum, using only the essential rules to rely almost entirely on the framework. The Dataset generation feature of Web Generator takes only choices and probabilities as input, which will produce HTML components (the most significant parts of the web page) and the elements (the secondary parts, usually with the actual content). The Fig 1 shows graphically the generation process, which is described in the following phases:

1) Generation choices: The algorithms choose a specific item of the choices given, including the layout, its components, and the content. The variety of available components depends on the defined classes in the corresponding modules. Specially layout probabilities can be set when creating a new layout. The remaining probabilities for content are uniformly distributed for every element in most cases. The choice is made using mainly the choices method of python's native random module. This way choices are made using cumulative weights provided for each choice.

2) HTML Batch Generation: After getting specific generation choices, the constructor of layouts generates a new HTML document with those specific choices for the layout components (header, navbar, sidebar, and footer). The content generator then picks a random number of different HTML elements to be appended to the layout's content container. The textual content is generated using lorem ipsum text for every list, paragraph, and titles. Finally, the HTML nodes are rendered into a text file in the drive.

3) Loop HTML files: The software creates a new instance of a web browser that will be managed by Selenium. Every file is looped and opened with the browser instance to get a screenshot of

the web page with the defined options. The HTML elements are tagged, injecting a Javascript with JQuery[12] snippet into the browser with the current file, detecting a previously determined attribute, and the element's corresponding bounding box. Once the loop is over, the element's coordinates and tags are saved to a JavaScript Object Notation (JSON) file.

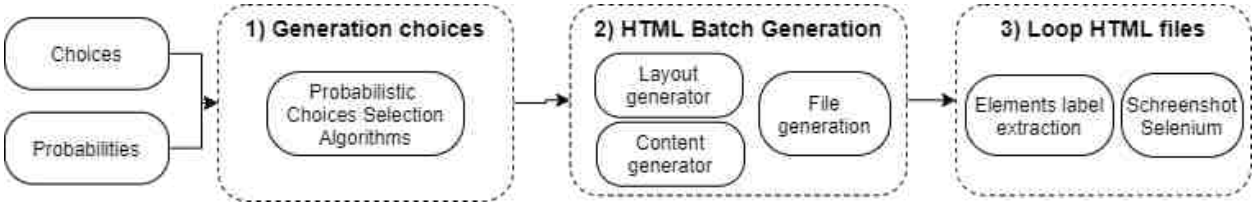

Figure 1. Logical schema overview about the generation process.

Layouts: A layout represents a specific distribution of the main web page components as well as the needed HTML document tags, including head, body, scripts, among others. There are five types of layouts defined by the relationship of the sidebar with the header and footer. Figure 2 presents the layout types, where H represents the header element and the navbar, S the sidebar, and F the footer.

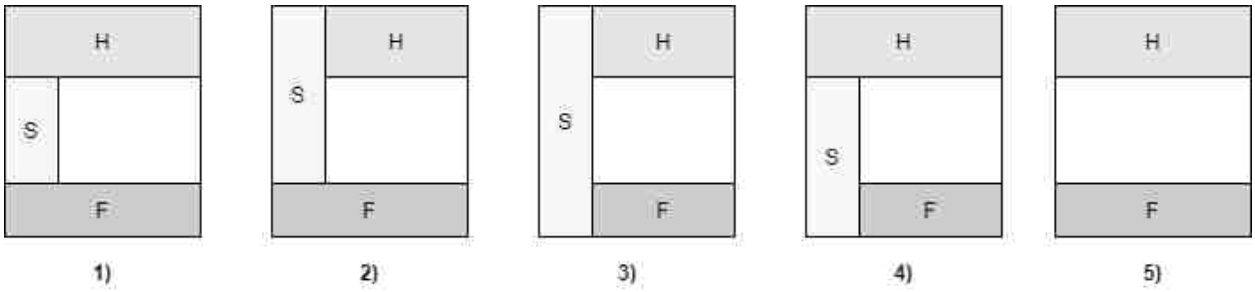

Figure 2. Possible types of layouts. Each number in the picture corresponds to a specific type and its respective elements to be rendered.

Elements: The software includes the principal components of Bootstrap's framework. These composed elements can be constructed as programmable objects to be rendered later. Whether component or element, every class has ultimately associated a mark-up tag, which can be set attributes and tag nodes such as paragraphs, lists, divs, and any HTML tag. These elements constitute the page's content and forms and are picked based on probabilities evenly among the elements. When the automatic pick of elements happens, vertical and horizontal symmetry rules are applied to maintain the design aesthetic.

2.1. Software Architecture

We try to keep interaction among modules as straightforward as possible, helping to the extensibility and maintenance. Indeed, it is especially important because the software could also be applied with different goals other than dataset generation, for instance, standalone color palette or web page generation. We use the package Dominate [13] for the HTML generation using object instances; that is why it is the most important package. The main

architectural pattern used is Blackboard, since initially, the majority of the components are modules using others. The Fig. 3 shows the packages diagram. The most significant parts of the software are divided into modules with logic relations and similar functionality. Base modules are **Randomization**, **Layout**, **Elements**, and **StyleManager**.

Randomization: It integrates the probabilities values, choices, and functions to generate random content and HTML elements. It uses random and lorem-python as dependencies.
Layout: The main element in this module is a class with inheritance from the dominant document class. It contains the main components of the webpage (header, navbar, sidebar, and footer), and the method allows to render the determined disposition of these components.
Elements: This module includes many classes that also use inheritance, in this case, from the tags of Dominate package. Thus, each Bootstrap component has an equivalent class object inside Elements.
StyleManager: This module is responsible for the CSS operations linked to the HTML generated files. The current version generates a random color palette and uses it in Bootstrap variables, generating a framework's custom distribution.

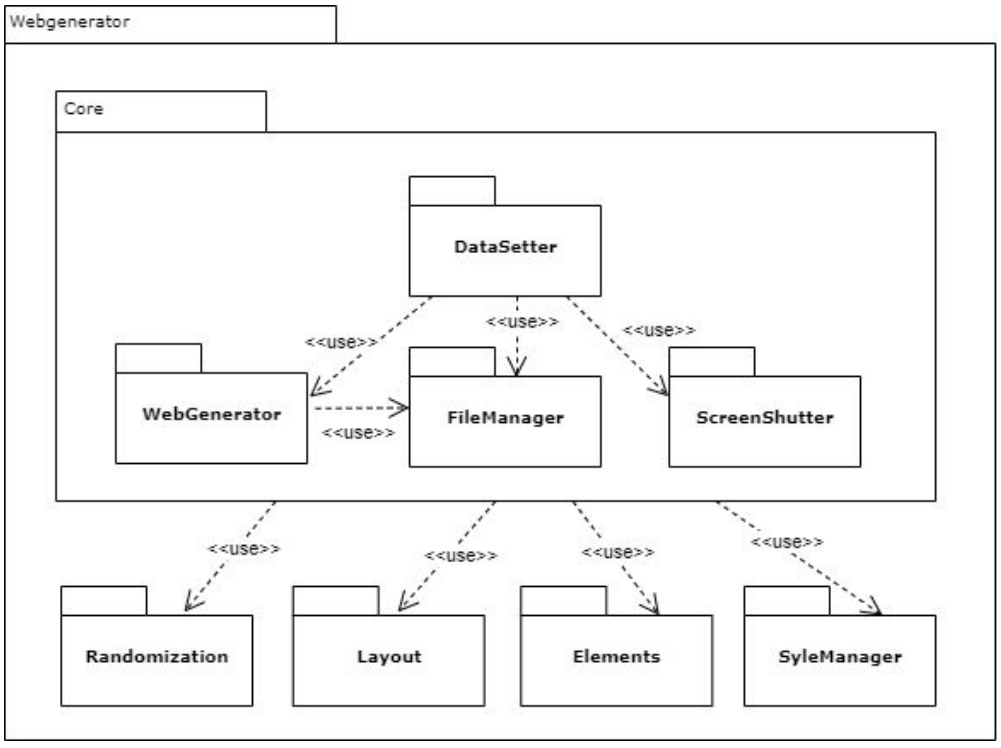

Figure 3. Packages diagram of the software

The core package integrates the base modules previously mentioned into more concrete ones to make them easier to manage. Below we detail each module of the core package:
Webgenerator: The module with the same name of the project allows us to create a new disposition of components of the webpage, internal content, color variation, among others. This module integrates mainly the randomization, layout, and element modules.

FileManager: This module takes care of preparing the required folder structure and other physical operations in the drive.

ScreenShutter: This module can iterate HTML files to generate screenshots and annotations of the tagged elements through an instance of the Chrome / Chromium browser.

DataSetter: Finally, the module with the higher abstraction provides the batch generation of HTML files and the management of the ScreenShutter capabilities.

2.2. Software Functionalities

- **Dataset generation**: Generate probabilistic HTML websites using user-defined probabilities for the layout components and disposition. The software ultimately generates HTML files, images in PNG format, and JSON annotation files with the tags for each web element contained in the image. The color features allow us to generate color palettes and apply different color classes to layout components, which are later compiled in the Bootstrap's CSS file. For the screenshot feature, the user can set whether it should generate full-screen or a custom screen's size for the capture.
- **Html Generation**: Layout and Elements modules lie on top of the Dominate package. Since it follows the same logic as its main dependency, developers can generate Bootstrap's code components to create instances of the corresponding objects.

3. Illustrative Examples

The generation of a dataset produces an output folder with CSS, js, HTML files, and image folders. The root of the output folder contains the JSON files generated with the tags of the HTML elements present in each screenshot. Fig. 4 shows a snippet of the resulting JSON. Inside the CSS folder, the Bootstrap distribution file with the web page's color palette and another file with the necessary CSS rules for the sidebar and extra required styling. The js folder contains the required JQuery and Bootstraps Javascript files. Finally, HTML contains the markup files and images of the screenshots generated. Fig 5. shows examples of the variations of color, size, and distribution of the web page structure and content.


```
{
    "random_webpage.png35430": [{
        "region_attributes": {
            "type": "header"
        },
        "shape_attributes": {
            "height": 234,
            "name": "rect",
            "width": 679,
            "x": 52,
            "y": 0
        }
    }]
}
```


Figure 4. Example of the JSON file containing the tags of the HTML Elements.

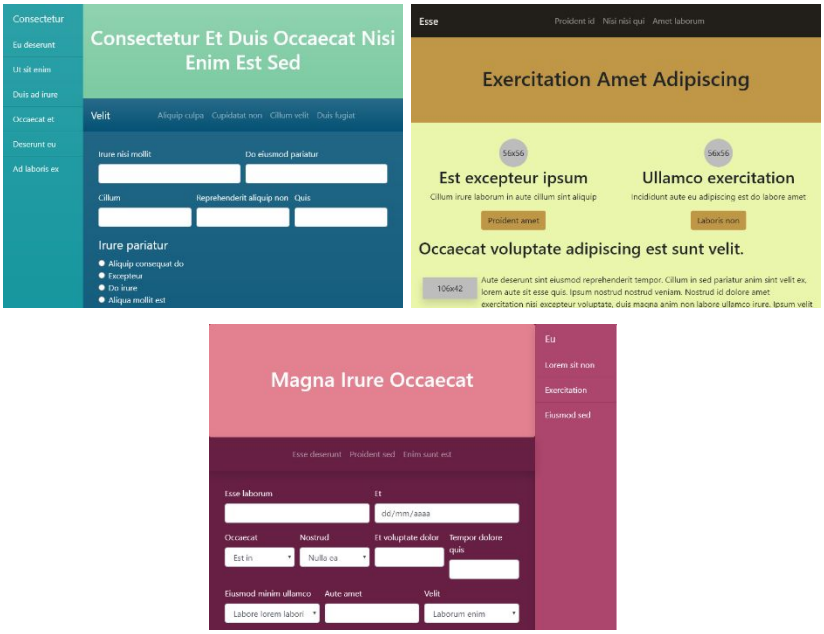

Figure 5. Examples of the screenshots generated by web generator

4. *Impact*

The use of this tool allows researchers to customize their web interface datasets with minimal effort. Thus, it facilitates the exploration of machine learning methods to generate new designs and code based on mockups. The software features help reduce the time employed for dataset preparation and ensure normalized batches in which diversity can also be increased to suit many related research problems. In addition to being a research tool for web interface-related problems, users can employ the individual modules with other aims. For instance, we can create educational software where the Web Generator modules are the building blocks. Thus, the student could set the parameters for generating web elements to and review the equivalent code generated.

5. Conclusions

We presented WebGenerator, a tool that allows researchers to generate synthetic web interfaces and content quickly to consolidate datasets that can be used to train/test different algorithms and support designers and developers in the web development process. The software structure allows the extensibility and customization of web development activities. The software comes with different generation options, such as probabilities of layouts and sections, size of screenshots, and color schemes. Thus, we make possible the generation of datasets that suit the needs of the research. Given the potential usage of this tool, we plan future iterations to develop new features to make a more robust software.

Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

This work is supported by the Smart Data Analysis Systems Group - SDAS Research Group (http://sdas-group.com).

References

[1] P. Bourque and R. E. Fairley,SWEBOK: guide to the software engineering body of knowledge. IEEE Computer Society, 2014. OCLC: 880350861.

[2] W. O. Galitz,The essential guide to user interface design: an introduction to GUI design principles and techniques. John Wiley & Sons, 2007.

[3] A. Schramm, A. Preußner, M. Heinrich, and L. Vogel, "Rapid UI Development for Enter-prise Applications: Combining Manual and Model-Driven Techniques," inModel DrivenEngineering Languages and Systems(D. Hutchison, T. Kanade, J. Kittler, J. M. Klein-berg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen,M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, D. C. Petriu, N. Rou-quette, and . Haugen, eds.), vol. 6394, pp. 271–285, Berlin, Heidelberg: Springer BerlinHeidelberg, 2010.

[4] T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI (T)," in2015 30th IEEE/ACM International Conference on AutomatedSoftware Engineering (ASE), (Lincoln, NE, USA), pp. 248–259, IEEE, Nov. 2015.

[5] Y. Han, J. He, and Q. Dong, "CSSSketch2code: An Automatic Method to Generate WebPages with CSS Style," inProceedings of the 2nd International Conference on Advances in Artificial Intelligence - ICAAI 2018, (Barcelona, Spain), pp. 29–35, ACM Press, 2018.

[6] P. F. F. Pereira, "Gerador de código HTML a partir de maquetes,"IEEE, p. 105, 2018.

[7] T. Beltramelli, "pix2code: Generating Code from a Graphical User Interface Screenshot,"arXiv:1705.07962 [cs], May 2017. arXiv: 1705.07962.

[8] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Ku-mar, "Rico: A mobile app dataset for building data-driven design applications," inPro-ceedings of the 30th Annual ACM Symposium on User Interface Software and Technology,pp. 845–854, 2017.

[9] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machi-ne Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps,"ar-Xiv:1802.02312 [cs], Feb. 2018. arXiv: 1802.02312.

[10] "Sketched-Webpages-Generator". https://github.com/Dev-Tarek/sketched-webpages-generator. Accessed: 17/09/2020.

[11] "Bootstrap Framework". https://getbootstrap.com/. Accessed: 17/09/2020.

[12] "JQuery". https://jquery.com/. Accessed: 17/09/2020.

[12] "Dominate Python Library". https://pypi.org/project/dominate/. Accessed: 17/09/2020.

Current executable software version

Table 2 – Software metadata (optional)

Nr	(Executable) software metadata description	***Please fill in this column***
S1	Current software version	*0.1*
S2	Permanent link to executables of this version	*https://github.com/agsoto/webgenerator*
S3	Legal Software License	*GPL v3*
S4	Computing platforms/Operating Systems	*Linux, OS X and Microsoft Windows*
S5	Installation requirements & dependencies	*Requirements:* - *Chromium / Chrome Browser > 80.0* - *Selenium Web Driver for Chromium = Browser version* - *Python >= 3.7* - *Pip >= 20.0.2* *Dependencies:* - *selenium = 3.141.0* - *colorharmonies = 1.0.5* - *dominate = 2.4.0* - *utils = 1.0.1* - *python_lorem = 1.1.2* - *palettable = 3.3.0* - *webdriver_manager = 2.3.0* - *libsass = 0.20.1* - *Pillow = 7.2.0* - *Selenium-Screenshot*
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	
S7	Support email for questions	*agsoto@protonmail.com*