THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2020, VOL. 43, SPECIAL ISSUE, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS
Article DOI: https://doi.org/10.35219/eeaci.2020.2.01

A WEB PLATFORM FOR GENERATING GRAPHICAL USER INTERFACES

Andreea Elena Țîru

"Dunarea de Jos" University of Galati, Faculty of Automation, Computer Sciences, Electronics, and Electrical Engineering
e-mail: et156@student.ugal.ro

Abstract: This paper is a general description of a Python-based web platform developed by the author that generates Graphical User Interfaces (GUIs) for software programs. It presents both technical and theoretical aspects of programming, Python libraries, chatbot development and auto-generating graphical interfaces. The article proposes an overview of a web system, explained with the help of diagrams.

Keywords: graphical user interface, python, web platform, auto-generating, front-end.

1. INTRODUCTION

A Graphical User Interface (GUI) involves a dynamic and interactive system of visual components for software programs (Meier, 2017). It is well-known as a user-friendly tool easy to manipulate on any device. Its evolution in software applications has increased programming productivity and code complexity. As direct-manipulation interfaces become more comfortable to use, they become harder to create, implement, debug and modify.

This work proposes a Python-based platform for generating GUIs. Well-known for its features as an object-oriented programming language, Python provides along with cross-platform support an extensive standard library. The result achieved using its features was a web platform, committed to providing graphical solutions to Python developers, for small web applications and sequences of code.

When writing any programs, consistency helps us keep the code readable and maintainable. That is why, while developing a module, an extension, or a core patch, developers should strive to follow some guidelines.

Conceived so that third-party extensions could easily have a user interface integrated, the web platform technically bases on a 3-tier architecture (computer data/object storage - digital access controlled through files in the "classes" folder, data access - user-provided content related to storing and retrieving files in the "root" folder and design - the files of the generated interface located in the "/theme folder").

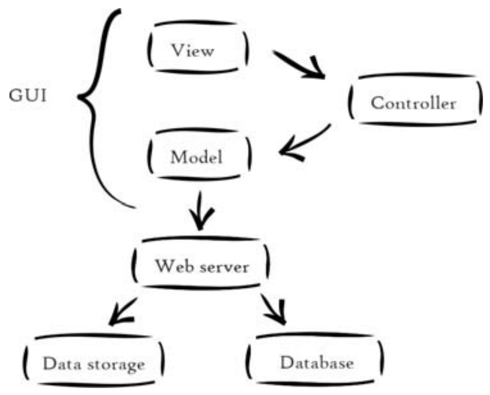


Fig.1. MVC interfacing with 3-tier[1] architecture

1.1. Core Development

The driving idea behind a custom architecture was the need to have a more robust, modular, and fully testable code. Using a proven and popular open-

[1] a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed as independent modules on separate platforms. This is the same principle as a Model View Controller (MVC) architecture, only in a simpler and more accessible way (Fig.1).

THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2020, VOL. 43, SPECIAL ISSUE, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS
Article DOI: https://doi.org/10.35219/eeaci.2020.2.01

source programming language came with the necessity of adopting some coding standards. General conventions as using UTF-8 (Unicode Transformation Format – 8-bit) without BOM (Byte Order Mark) and using the Unix LF (Line Feed) ending with a single blank line are absolute requirements. Regarding Python code conventions, files must follow the PEP-4 and PEP-8 standards.[2] Yoda conditions[3] are also suggested, but not enforced. Naming conventions strictly apply for Controllers & actions, Templates, Routes and Paths.

1.2. Domain-Driven Architecture

The platform's architecture is progressively evolving into a new web generation, inspired by Eric Evan's Domain-driven design - DDD (Haywood, 2009). It makes the architecture more natural to understand and maintain. One of the main principles of DDD is the CQRS - Command Query Responsibility Segregation (Fig.2).

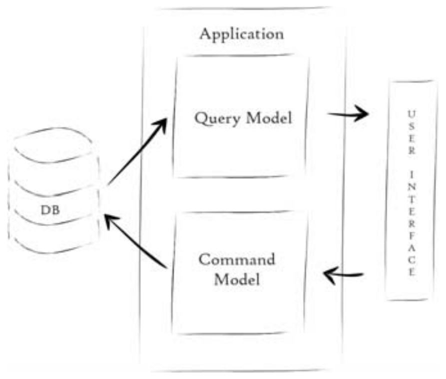


Fig.2. CQRS model.

CQRS represents the implementation of two different models of interaction with the UI (User-Interface) and the DB (Database). This separation occurs based upon whether the methods represent a CM (Command Model) or a QM (Query Model).

Beyond this introduction, the paper is structured as follows: Section 2 exemplifies the architecture of a chatbot; Section 3 illustrates the functionalities of the developed web platform and the steps required to achieve a graphical user interface. The thinking behind the expected result and its possible customization is explained in Section 4.

[2] coding conventions for the Python code comprising the standard library in the main Python distribution.

[3] a programming style where the two parts of an expression are reversed from the typical order in a conditional statement. A Yoda condition places the constant portion of the expression on the left side of the conditional statement.

2. FOLLOWING AN APPLICATION EXAMPLE

A cross-platform application's extensibility revolves around small programs that make use of different functionalities—having a graphical interface generated means automatically having included template (.tpl) files and assets (images, icons, JavaScript, CSS – Cascading Style Sheets) to the source code, to display the interface on different cross-platform devices and software.

Developing and discovering software products is the ideal way to expose solutions to users' needs. Standard components used in web programs display a variety of content, perform many tasks and interact with other tools. The more intuitive the design of an application is, the easier it will be to use.

A type of application that includes all the above is a chatbot.

2.1. Developing a Chatbot

A chatbot is an automated software program used in dynamic dialog systems that simulates human conversations (Raj, 2019). A chatbot requires NLP (natural language processing) implementation and conducts a conversation via auditory or textual methods (Fig.3).

NLTK[4] is a leading library for building Python programs to work with human language data. Common Python chatbots make use of this library and its components, such as classification, tokenization, stemming, tagging, parsing and semantic reasoning.

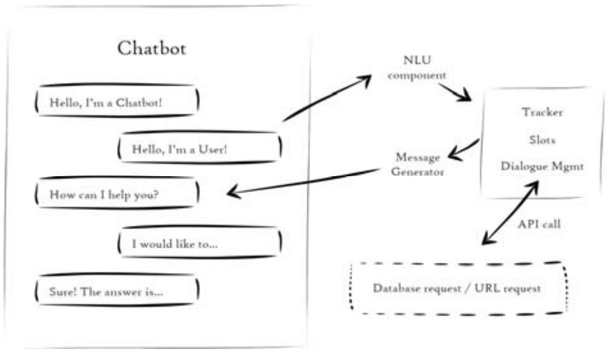


Fig.3. Example of a conversational AI chatbot architecture.

Assuming that a chatbot's functionalities were already developed, and the source code follows the platform's core development guidelines, the

[4] Natural Language Toolkit, a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.

following sections will use it as an example to understand how the interface generator works.

3. GENERATING THE GRAPHICAL USER INTERFACE

In short terms, generating means bringing into existence. Generating a graphical user interface means automatically creating and displaying a design to a specific program. The current platform adapts to software products needs using an intelligent system, and outputs unique front-end displays tailoring the code. The steps for uploading a project are intuitive and include advanced development functionalities (Fig.4).

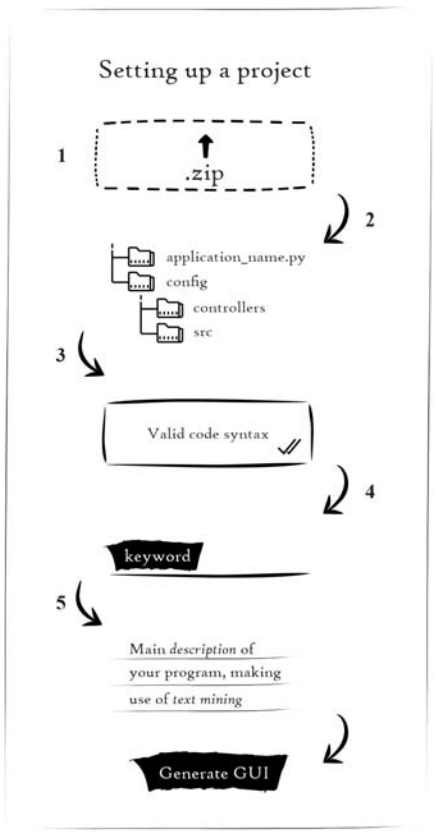


Fig.4. An illustration of the main steps used for generating an interface.

The first step of setting up a project is to upload the Python code of the application on the web platform. It should consist of all required folders and files, compressed within any archive.

The second step the platform does is to verify the uploaded content. A more simplified approach used in this platform's development is to generate a hash of the copied file and compare it to the hash of the original file. In other words, the system browses through folders and files for systematic corruption. Besides this, it also checks if the folder and files are structured as required and if they follow the core development references.

The third step requires a Python syntax validation. An online complex debugger was implemented using the PDB[5] module. Any unexpected error is output and, to simplify the modifying process, a live code editor is available. The user will not have to pause / to break the generating process and upload the project again.

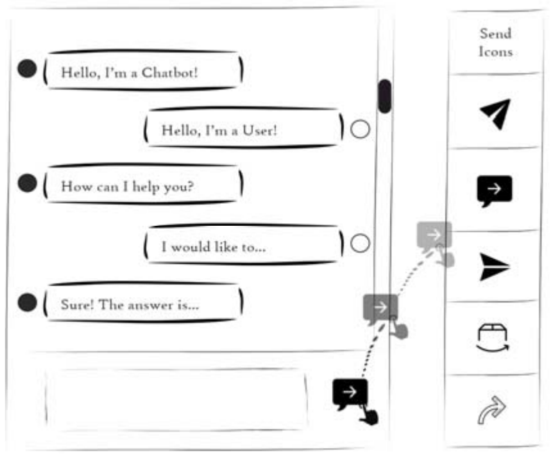
The fourth step involves user interaction. The developer must type in the keywords that best describe the program's category and features. The implemented system searches through an extensive database of web terms and notions, sending the values to the generator's analyser.

The fifth step is not mandatory but recommended if there is a design direction the intelligent generator should follow. Design expectations and ideas should be typed within a natural language, describing as many details as possible, so that the final design will fit exactly the expectations. If the input is not completed, the generated interface will mix components and output editable content. At the base of this functionality stands text mining and natural language processing.

The last step is generating the GUI. The expected result consists of HTML, CSS and Bootstrap code, all wrapped within a modern graphic design.

4. FINAL RESULT

Before downloading the final project, the user can manually edit the interface's appearance by dragging and dropping different template components and then customizing them (Fig. 5).



[5] Python Debugger Module. It defines an interactive source code debugger for Python programs and supports conditional breakpoints at the source line level, an inspection of stack frames, source code listing, and evaluation of arbitrary Python code in the context of any stack frame.

Fig. 5 Illustration of drag and drop system, for an auto-generated chatbot interface.

A template is a basic pre-uploaded design model that includes layers and objects, used to perform a high-quality design output. Based on image recognition and web scraping, its goal is either to generate a close representation of the description typed in by the user, or a unique assembly of components never output before.

The final code is then available to download. It includes the uploaded code and the generated interface code, located in the “/theme folder”. After this, the application is ready to be implemented as intended.

5. CONCLUSIONS

The advantages of having an interface automatically generated and implemented can be classified into two main groups.

First, the quality of the resulting user interface might be higher, for the following reasons: design can be rapidly prototyped and implemented, the reliability of the user interface will be higher because the code is created automatically from a higher-level specification and more effort can be expended on the functionalities’ development of the uploaded application.

Second, once generated, the UI (User Interface) code might be more comfortable and more economical to maintain. This is because there will be better modularization, due to the separation of the UI component from the application and there will also be less front-end code to write because the project can be uploaded on the modification’s webpage.

In conclusion, as tomorrow’s user interfaces will provide speech recognition, vision from cameras, 3-D, intelligent agents and integrated multimedia, a solution to simplify developers current work for UIs based graphics is to provide intelligent tools that can minimize the development time and increase the code quality of the applications.

6. REFERENCES

Haywood, D. (2009). *Domain-Driven Design: Using Naked Objects,* pp.223. The Pragmatic Programmers LLC, Raleigh.

Meier, B.A. (2017). *Python GUI Programming Cookbook: Develop beautiful and powerful GUIs using the Python programming language,* pp.126. Packt Publishing Ltd, Birmingham.

Raj, S. (2019). *Building Chatbots with Python: Using Natural Language Processing and Machine Learning,* pp.205. Apress Media, California.