Chapter 1
Dynamic Generation of Documentation, Code, and Tests for a Digital Marketing Platform’s API

Ricardo Santos  
E-goi, Portugal

Ivo Pereira  
E-goi, Portugal

Isabel Azevedo  
Polytechnic Institute of Porto, Portugal

ABSTRACT

Detailed documentation and software tests are key factors for the success of a web application programming interface (API). When designing an API, especially in a design first approach, it is relevant to define a formal contract, known as API specification. This document must contain all necessary information regarding the API behavior. Thereby, the specification can be used to dynamically generate API components like documentation, client and server code, and software tests, reducing development and maintenance costs. This chapter presents a study of OpenAPI specification and its application on designing a new RESTful API for E-goi. It also presents a set of solutions for generating documentation, client code libraries, and test cases.

DOI: 10.4018/978-1-5225-7455-2.ch001
INTRODUCTION

Nowadays, there is a great need to automate processes to achieve increased productivity. This necessity leads many companies to publish part of their internal services on the web, making them available to their clients through Application Programming Interfaces (APIs) (Abelló Gamazo, et al., 2017). Thus, automation is essential for many businesses, like digital marketing, making APIs a core component, since clients often need to automate and customize marketing processes.

However, web API are often black-boxes to its users, since its code is not publicly accessible (Schwichtenberg, 2017). Developers working on their applications rapidly integrate third-party software to build distributed software systems but learning how to use an API properly is not a trivial task (Meng, 2017).

Thereby, one of the most important characteristics of an Application Programming Interface is its technical documentation, which exists to facilitate communication between users and the API. To make an API easy to use it is essential to assure that its documentation is always correct and updated, and it provides clear examples of its usage (Uddin, 2015; Sohan, Anslow, & Maurer, 2015; De, 2017). In fact, the documentation can be very important to understand how to use libraries and frameworks developed by other teams and whose functionalities are easily accessible through APIs (Robillard, 2011).

Still, API documentation is frequently created manually and needs to be updated whenever the API evolves, otherwise it will become obsolete (Abelló Gamazo, et al., 2017), which is a common problem. This often brings maintenance costs to the developers every time a service is created or updated. Therefore, rather than focusing on automating what’s easy, focus on automating the boring parts (unlike you, computers love repetition), the difficult parts (reduce error-prone steps), and the parts that need to happen when you would rather be asleep. As a human, you are better than computers at improvisation and being flexible, exercising judgment, and coping with variations (Limoncelli, 2018).

Such as for any software, API testing is essential to improve its quality and reduce the costs of developing an application (Tassey, 2002). Yet, sometimes, developing software tests can be even more expensive than building the product itself (Tassey, 2002).

Thus, to maintain an API, including not only its code but also its documentation and software tests, can be a very costly and painful process. Fortunately, some of these maintenance costs can be reduced using an API specification. An API specification establishes a contract and describes the API through a document readable by both humans and machines (Open API Initiative, 2018), allowing the generation of documentation dynamically through that contract. The API is forced to follow the
Related Content

Teaching Model-Driven Engineering in a Master's Program: Three Editions on a PBL-Based Experience
www.igi-global.com/chapter/teaching-model-driven-engineering-in-a-masters-program/219980?camid=4v1a

Operator Overloading as a DSL Parsing Mechanism
www.igi-global.com/chapter/operator-overloading-as-a-dsl-parsing-mechanism/219983?camid=4v1a

Evaluating the Refactoring Index Using Entropy Approach
www.igi-global.com/chapter/evaluating-the-refactoring-index-using-entropy-approach/219982?camid=4v1a
Structural Coverage Analysis Methods
www.igi-global.com/chapter/structural-coverage-analysis-methods/219975?camid=4v1a