Sni'per: a Code Snippet RESTful API*

Ricardo Queirós[1] and Alberto Simões[2]

1 **ESEIG/IPP & INESC-TEC, Porto, Portugal**
`ricardoqueiros@eseig.ipp.pt`
2 **Centro de Estudos Humanísticos & Centro Algoritmi, Universidade do Minho, Braga, Portugal**
`ambs@ilch.uminho.pt`

Abstract

Today we use the Web for almost everything, even to program. There are several specialized code editors gravitating on the Web and emulating most of the features inherited from traditional IDEs, such as, syntax highlight, code folding, autocompletion and even code refactorization. One of the techniques to speed the code development is the use of snippets as predefined code blocks that can be automatically included in the code. Although several Web editors support this functionality, they come with a limited set of snippets, not allowing the contribution of new blocks of code. Even if that would be possible, they would be available only to the code's owner or to the editors' users through a private cloud repository. This paper describes the design and implementation of Sni'per, a RESTful API that allows public access for multi-language programming code-blocks ordered by popularity. Besides being able to access code snippets from other users and score them, we can also contribute with our own snippets creating a global network of shared code. In order to make coding against this API easier, we create a client library that reduces the amount of code required to write and make the code more robust.

1998 ACM Subject Classification D.3 Programming Languages; D.3.3 Language Constructs and Features

Keywords and phrases Programming languages, interoperability, web services, code snippets

Digital Object Identifier 10.4230/OASIcs.SLATE.2016.13

1 Introduction

With the evolution of the Internet, the act of programming, so far exclusive to standalone IDEs, moved to the Web taking advantage of its full potential regarding ubiquity and speed. This transition fostered code sharing and collaboration in real time, whether the goal is pair programming or just get some help from a friend.

In this context, several Web programming editors appeared, in the last decade, such as ICEcoder, CodeAnywhere, CodeMirror, ACEditor, and many others. These editors are often used by web applications for different purposes: 1) computer programming learning, 2) code development and testing, and 3) programming contests.

Regardless of their purposes, they include a set of features, inherited from traditional IDEs, that aims to facilitate and speed up the coding process [5, 2], such as syntax highlight and checking, tab support, indentation, bracket matching, code folding, keyword shortcuts, spell checking, code generation, refatorization, etc.

* This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

One of the most promising features is the code snippets. Code snippets is a well-known IDE feature that increases productivity by reducing the amount of time spent by programmers typing repetitive code or searching for samples [4]. In fact, snippets might be used for standard file skeletons, class and function definitions, HTML tables, and much more.

Despite their importance, most editors do not benefit from the potential that the Web provides in order to leverage code snippets implementation. For instance, there are few editors that support code snippets. Those who support, do not allow the contribution of new snippets and provide only a small range of code snippets based on a private user account or on a specific editor's code repository.

This paper presents the design and implementation of an API – called Sni'per – that allows code snippets management. The RESTful API supports the access to multi-language code blocks and the submission of new snippets to authenticated users. These same users can also score the snippets affecting the results of subsequent requests.

The remainder of this paper is organized as follows. Section 2 analyses several of existing code snippets formats highlighting both their differences and their similar features. In the same section, various snippets API are also covered and a comparative survey is presented based on several criteria. The following section provides details on the design and implementation of Sni'per, including its architecture, the service API, the data model and a client library. The final section summarizes the main contributions of this research and plans futures developments of this service.

2 Related work

The standardization of content and communication is the key for the interoperability on the Web. In this section, we survey the approaches used to formally represent and share snippets over the Web.

2.1 Snippets languages

Code snippets are small blocks of reusable code that can be inserted into a source file to speed up coding. Usually, the representation of a snippet resource is encoded in XML or JSON formats and include the text of the snippet and some metadata for easy discovery. The content of the snippet can be categorized in two groups:

Static snippets: plain text inserted by the user into the source code. The user is not able to specify anything else.

Dynamic snippets: plain text combined with dynamic elements. The user may specify both the content of the dynamic elements, as well as their position relative to the plain text. Good examples are variables such as the current system date or the input from the user that is supplied via a GUI.

Dynamic snippets use *placeholders* to define dynamic elements. These elements are supplied by the user (through graphical user interface and modal dialog boxes) or other external process. Placeholders have special markup syntax that allows the editor to identify the boundaries of placeholders relative to the plain text. There are two typical actions made with placeholders: duplication and transformation. The former allows the user to indicate that the value supplied for one placeholder should be replicated in multiple places, relative to the entire text of the programmable snippet. The later, allows the user to indicate that one or more values supplied for a placeholder should be replicated and transformed in other places within the text of the programmable snippet. For instance, the user may supply a

Listing 1 HelloWorld VS code snippet.

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets
  xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>Hello World VB</Title>
      <Shortcut>HelloWorld</Shortcut>
      <Description>Inserts code</Description>
      <Author>MSIT</Author>
      <SnippetTypes>
        <SnippetType>Expansion</SnippetType>
        <SnippetType>SurroundsWith</SnippetType>
      </SnippetTypes>
    </Header>
    <Snippet>
      <Declarations>
        <Literal>
          <ID>expression</ID>
          <ToolTip>Expression to switch on.</ToolTip>
          <Default>World</Default>
        </Literal>
      </Declarations>
      <Code Language="VB">
        <![CDATA[Console.WriteLine("Hello, $expression$!")]]>
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

document title in one part of the snippet, and specify that the document title should be repeated in other places, with the first instance being all uppercase and every other instance being lowercase.

The previous categorization can include also *scriptable snippets* that consists of runnable segments of code in a scripting language. Although its flexibility, it depends on the programming languages supported by the text editor.

There are several languages to represent code snippets. We detail two of the most popular: VS Code Snippet and TextMate snippet syntax.

2.1.1 VS Code Snippet

VS Code Snippets[1] are predefined pieces of code (e.g. Visual Basic, Visual C#, or XML) that are ready to be inserted into source code within Visual Studio. The VS Code Snippet language uses an XML Schema to formalize a code snippet. Listing 1 shows a simple XML instance that reproduces a "Hello World" snippet:

The root element `CodeSnippets` aggregates two elements: Header and Snippet. The former includes metadata about the snippet, more precisely, its title, author, date, description,

[1] Code Snippets Schema Reference: `https://msdn.microsoft.com/en-us/library/ms171418.aspx`

Listing 2 For Loop snippet for JavaScript using TextMate language snippet.

```
"For Loop": {
  "prefix": "for",
  "body": [
   "for(var ${index} = 0; ${index} < ${array}.length; ${index}++) {",
     "\tvar ${element} = ${array}[${index}];",
     "\t$0",
   "}"
  ],
  "description": "For Loop"
},
```

shortcut and type of the snippet. The type, represented by a set of `SnippetType` elements, can be one of the following values:

- SurroundsWith: allows the code snippet to be placed around a selected piece of code;
- Expansion: allows the code snippet to be inserted at the cursor;
- Refactoring: specifies that the code snippet is used during Visual C# refactoring.

The Snippet element contains the Code element that defines the placeholders and the code snippet. To define a placeholder you must include the Literal element that defines the literals of the code snippet that you can edit and add three sub-elements: ID – specifies a unique identifier for the literal; ToolTip – describes the expected value and usage of the literal; and Default – specifies the literals' default value when you insert the code snippet. Later in the element Code you should signal the placeholder with $. The element has an attribute Language where the language of the code snippet is defined.

2.1.2 Textmate Code Snippet

TextMate snippets[2] are pieces of text which can be inserted into the source code at the current location via a context-sensitive key stroke or tab completion. The text can be, in the simplest case, plain text that you do not want to type systematically (either because you type it a lot, or because the actual text to insert is hard to remember) or dynamic text with interpolated shell code, placeholders and transformations.

The Textmate language uses JSON as the format for the snippet syntax. Each snippet is defined under a snippet name and has a prefix, body and description. The prefix is what is used to trigger the snippet and the body will be expanded and inserted. Possible variables are: $1, $2 for tab stops and $id and $id:label and $1:label for variables variables. Listing 2 shows a For Loop snippet for JavaScript.

Variables with the same id are connected. In the previous example, the $index variable (after set) will reflect the same value for all variables with the same name.

2.2 Code snippets API

With the evolution of Computer Science, several front-end development tools and code snippets repositories appeared having in mind code sharing and collaboration.

There are several tools focused on front-end development, usually coined as live pastebin apps, such as jsbin, jsfiddle or codepen. In this context, Github Gists assumes a leading role.

[2] `https://manual.macromates.com/en/snippets`

Listing 3 Snippet creation with Bitbucket API.

```
$ curl -u {username}
       -X POST https://api.bitbucket.org/2.0/snippets/ \
       -F file=@mySnippet.txt
```

GitHub Gists are a type of pastebin apps that hosts code snippets adding version control, easy forking, and SSL encryption for private pastes.

Regarding code snippets repositories, the most prominent are GistBox, CSnipp, SnippetSource.net, Snipplicious, Bootsnipp, Snip2code and Tagmycode. Most of these repositories offer a GUI to create and manage public or private snippets. If they are public some of the repositories allow the possibility to comment and grade snippets. All of them allow the snippets search based on tags and the "copy & paste" action for the users code editor.

Nevertheless, these are generic tools and do not allow their easy use as a snippet API service. An API for code snippets is not a novel approach. There are a few solutions on the Web that provide access to pieces of code through an API. Most solutions use REST as the software architectural style in order to benefit from its main features (e.g. caching, self-descriptiveness and hypermedia) and JSON as the data format used for the asynchronous browser/server communication. The next subsections details four code snippets API: Bitbucket, Glot, SnippetStash and TagMyCode.

2.2.1 Bitbucket Snippet API

Bitbucket is a Web-based hosting service, owned by the Atlassian group, for projects using revision control systems (e.g. Mercurial, Git). It is similar to GitHub, which primarily uses Git. The Bitbucket cloud hosts several REST API to build third party applications. The snippets API[3] allows you to create, retrieve and delete snippets in the Bitbucket Cloud as well information about them. Snippets can be either public (visible to anyone on Bitbucket Cloud, as well as anonymous users), or private (visible only to the owner, creator and members of the team in case the snippet is owned by a team). Beyond the possibility of managing snippets (e.g. add, update, get and delete), the API allows users to comment and watch for existing snippets. For instance, creating a snippet from a local file is just a single curl command as shown in Listing 3: The API uses JSON as the standard format to exchange data between the server and the browser.

2.2.2 Glot Snippet API

The Glot Snippets API[4] provides an HTTP API for storing and managing snippets. This API enables users to create, update, get, list and delete snippets. Snippets can be saved as either public or secret. Public snippets appear in `/snippets` the endpoint and can be found by search engines. Secret snippets can only be accessed by those who know the URL. In order to create a private snippet an API token is required. Listing 4 shows how to create an anonymous snippet:

This Snippets API uses CouchDB as the datastore engine and JSON for exchanging data.

[3] https://confluence.atlassian.com/bitbucket/snippets-endpoint-719095086.html
[4] https://github.com/prasmussen/glot-snippets/tree/master/api_docs

Listing 4 Snippet creation with Glot API.

```
$ curl --request POST \
       --header 'Content-type: application/json' \
       --data '{
        "language": "python",
        "title": "test",
        "public": true,
        "files": [{"name": "main.py", "content": "print(42)"}]}' \
       --url 'https://snippets.glot.io/snippets'
```

Listing 5 Snippet creation with Glot API.

```
$ curl -u my@email.com:d53a1fb463c7e3180f3ac0f1479ec7daffa2b9b7 \
  http://www.snippetstash.com/snippets.xml
```

2.2.3 SnippetStash API

The SnippetStash API[5] is a REST-based API which allows users to integrate snippets in an easy and uniform way. The SnippetStash has a *public and a private API* hosted in the main endpoint `http://www.snippetstash.com`. The *public API* has three endpoints:

- `/snippets/latest.xml` – obtains the latest twenty snippets in XML format;
- `/tags.xml` – retrieves all existent tags;
- `/tags/[tag].xml` – retrieves all the snippets associated with [tag].

The responses to these requests use an ad-hoc XML representation.

The *private API* allows users to access and manage personal snippets. Since SnippetStash supports authentication via OpenID and basic password authentication, it is necessary to use an API key to access the API functions. After registration, the API key is sent by e-mail to the registered user. To access the API, you simply provide the email address and the API key on every request. For instance, to get your snippets returned as XML use the code included in Listing 5:

Other actions supported by this API are creating, editing and sharing snippets.

2.2.4 TagMyCode Snippet API

The TagMyCode API[6] is a RESTful service that enables users to access to snippets and tags. Firstly, you need to register a new application to get a consumer id and secret. Most of the TagMyCode API calls needs authentication. OAuth 2 is the only way to authenticate your requests. After the authentication, you can execute several actions based on the main endpoint `https://api.tagmycode.com/`, such as:

- `/account?access_token=YOUR_TOKEN` – get logged user information;
- `/languages` – list languages;
- `/snippets` – list snippets;
- `/snippets/:id` – get single snippets;

[5] `http://www.snippetstash.com/api`
[6] `https://tagmycode.com/`

Table 1 Snippet API comparision.

Features	Bitbucket	Glot	SnippetStash	TagMyCode
Web Service	REST	REST	REST	REST
Authentication	HTTP Basic	HTTP Basic	OpenID & Basic	OAuth
Storage	–	CouchDB	–	–
Response format	JSON	JSON	XML	JSON
CRUD actions	YES	YES	YES	YES
Social	Comment	NO	Share & Unshare	NO
GUI	NO	NO	YES	YES
# public snippets	45	87	199	1350
# languages	6	17	19	59

- `/snippets` – create a new snippet; this is a HTTP POST request with the following parameters: the id of the language, the title of the snippet, the description of the snippet (optional), the tags of the snippet (comma or space separated) and the optional boolean private that specifies if the snippet is private;
- `/snippets/:id` – edit a specific snippet; this is a PUT request with the same parameters as the previous endpoint;
- `/snippets/:id` – removes a specific snippet; this is a DELETE request with the identification of the snippet to remove;
- `/search` – search for snippets based on a search keyword.

2.2.5 Snippet API comparision

In this section, we present a comparative table that surveys the four snippets API previously presented. Table 1 compare the four snippets API based on several criteria.

All the snippets API use REST as the software architectural style because of its simplicity.

Regarding authentication, basic access authentication is the preferred method providing a username and password when making a request.

JSON is the elected exchange format. In fact, JSON has a much smaller grammar when compared with XML and maps more directly onto the data structures used in modern programming languages such as JavaScript.

All the API support common actions over snippets, such as, the creation of snippets through the HTTP POST, their edition using the HTTP PUT method, getting a specific snippet or all snippets through the HTTP GET and remove a specific snippet using the HTTP DELETE method. Moreover, snippets can be found by using filters based on tags. Regarding social features, Bitbucket supports comments on snippets and SnippetStash allow users to share snippets with others.

Beyond the documentation of the API, some API (SnippetStash and TagMyCode) have a graphical interface where users can perform all the CRUD actions and others such as embed the snippet in a HTML page or share it on social networks (e.g. Facebook, Twitter).

Lastly, to assess the use and diversity of content API, we obtained data on the number of public snippets and programming languages represented. Based on the numbers, it can be concluded that the SnippetStash and TagMyCode are the API with more activity.

3 Sni'per

This section describes the details on Sni'per server implementation.

Table 2 Snippet API.

Function	REST syntax
List languages	`GET /langs`
List language snippets	`GET /`*lang-id*
List snippets per language/type	`GET /`*lang-id*`/`*keyword*
Retrieve snippet	`GET /snippet/`*snippet-id*
Register snippet	`POST /`*lang-id*`/`*keyword*`/`*username* < *auth-hash, snippet*
Retrieve user info and snippets	`GET /user/`*user-name*
Delete snippet	`DELETE /`*lang-id*`/`*keyword*`/`*username*`/`*auth-hash*
Register	`POST /user/`*user-name* < *email, password*
Authenticate	`POST /user/`*user-name* < *password*
Vote on snippet	`POST /snippet/`*snippet-id*`/`*voting-user* < *auth-hash*

3.1 Architecture

Sni'per is a complete JSON REST service, implemented in Perl, using Dancer 2 Web Framework. This server interacts with a MongoDB NoSQL server, using the Perl Moo OO module. Thus, requests are intercepted by the web server, that translate the request route in a query for the object model. The object model queries the database, and retrieves an answer that is later serialised to JSON.

As Sni'per uses a RESTful approach [1], where HTTP verbs are used to indicate the type of each operation, and without storing any information between each request. This means that each client needs to authenticate itself for each operation that requires it. In order to allow clients not to store the user password, and send it over and over for each request, Sni'per uses a token-based authentication. The client performs its authentication sending the user name and password. It receives a token that identifies that user in his/her consequent requests.

3.2 Service API

Table 2 lists the most important routes available in Sni'per API. Note that for simplicity we decided not to include the full URI, but just the route. Routes requiring extra explanation are:

- The registration and authentication that use the same route but behave differently accordingly with the supplied arguments.
- Users can have only one snippet per language/keyword pair. Note that this is the usual behaviour for most text editors. If the user wants to have different snippets for the same language keyword, they name it differently (for example, `for` and `fori` where the first cycles over the elements of a collection, and the second creates a typical index-based cycle).
- The voting route is used both to vote or remove a vote. Unlike other websites, like *Stack Overflow*, we decided to have a vote-only approach (or the starring mechanic). If the user likes a snippet it can vote or star it. If the user tries to like it again, the vote is removed.
- Although there is nothing against the definition of a body to a `DELETE` request, some web servers and proxies drop that information. Therefore, our `DELETE` routes include, explicitly, all required parameters for authentication.

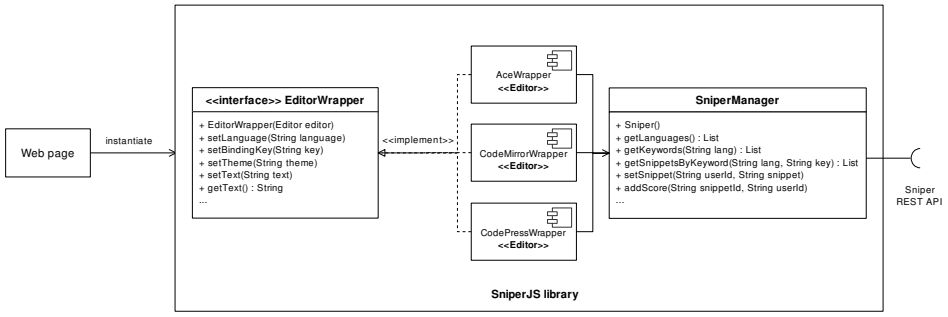

Figure 1 Components diagram of the SniperJS library.

As pointed before, the database uses a NoSQL approach, storing information in a MongoDB database [6]. At the moment, the database consists of a set of collections, for users, languages, snippets and votes. This requires the system logic to be implemented in the OO model representation. In the future we expect to use more features from MongoDB, reducing the complexity of the system logic.

3.3 Client library

In order to foster the use of the Sni'per API on Web environments, we implement a JavaScript library called SniperJS. Figure 1 shows the component diagram of the library. The cornerstone of the library is the interface EditorWrapper that defines the actions that a specific editor must follow. Thus, to support a specific editor, it is necessary to implement this interface through the creation of a wrapper. The interface has the following functions:

- EditorWrapper([Editor] editor) – instantiates a new wrapper;
- setLanguage(String language) – defines the language of the editor;
- setBindingKey(String key) – defines the key combination that will trigger the Sni'per management modal box;
- setTheme(String theme) – defines the theme of the editor;
- setText(String text) – includes new text on the editor;
- getText() : String – obtains text from the editor.

In the HTML page, we instantiate a EditorWrapper passing a reference for the specific editor. Listing 6 shows the initialization of the Ace editor. In turn, the wrapper is responsible for communicating directly with the Snippet API through the SniperManager class that provides a set of functions to facilitate the process of interaction with the Snippet API.

This architecture promotes and facilitates the use of Sni'per API. If a programmer does not find the desired wrapper, he/she only has to create a new one, that implements the EditorWrapper interface and, then, use the SniperManager class to interact with the API without the need to handle HTTP requests or even parsing JSON reponses.

The GUI for the Sni'per library had two design requirements: be simple and responsive. Based on these requirements we opt for using Bootstrap for the creation of the graphical interface of the Sni'per JS library. Bootstrap is an open source frontend Web framework. Since version 2.0 the framework supports responsive web design. This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone). Figure 2 shows the modal box that allows users to select the desired snippet.

Listing 6 Instantiation of the Sni'per library.

```
<body>
...
<div id="editor"></div>
...
<script>
    // Ace editor initialization
    var aceSniper = new EditorWrapper(ace.edit("editor"));
    aceSniper.setLanguage("c_cpp");
    aceSniper.setTheme("ace/theme/cobalt");
    aceSniper.setBindingKey("Shift-Return");
</script>
...
</body>
```

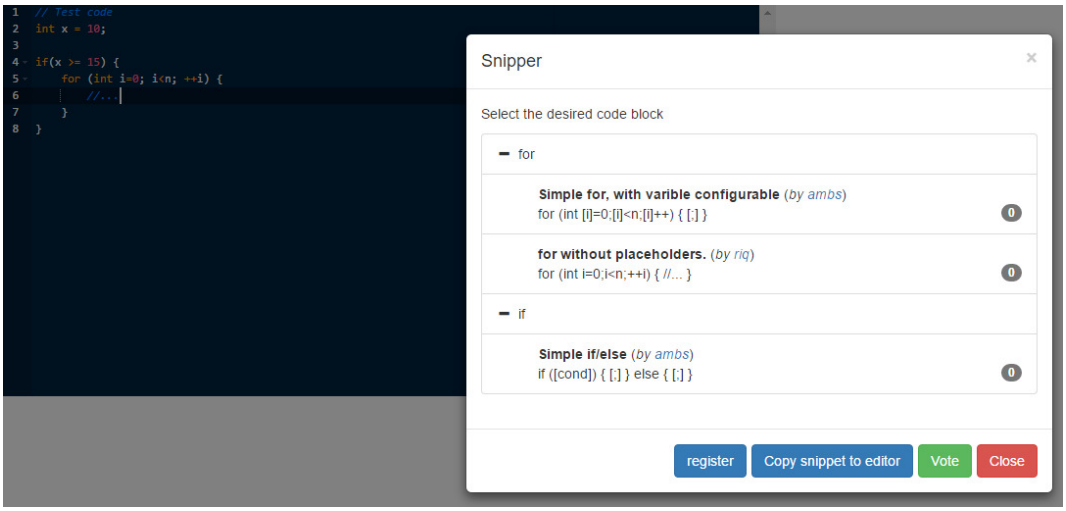

Figure 2 Sni'per library graphical interface.

4 Conclusions

The goal of the research presented in this paper is to promote the interoperability among Web code editors through the use of a standard API for code snippets management. The proposed approach is a service to manage code snippets on-the-fly. The contribution of this research is twofold, the service definition and a client library implementation that will use the service. The service definition comprehends the modular design of the snippet service based on a RESTful API with a set of functions for snippets management such as getting snippets based on a keyword, submit a new snippet of a specific language or vote on a single snippet. The client of the Sni'per service is a library implemented in JavaScript with a Bootstrap-based GUI. Currently, Sni'per supports only a few languages with few snippets each. The idea was initially to have a functional proof of concept. The API can be accessed in the following link `sniper.zbr.pt`[7]. As future work the authors will:

[7] To be available soon.

- increase the number of snippets on the cloud snippets repository
- extend the Sni'per compliance to other snippet formats based on the implementation of injectors of other snippets DSLs
- use source code context to enhance snippet retrieval and parameterization [8], [3]
- improve dynamic snippets using placeholders to define dynamic elements with duplication and transformation actions [7]

References

1 Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. In *Proceedings of the 22Nd International Conference on Software Engineering*, ICSE '00, pages 407–416, New York, NY, USA, 2000. ACM. doi:10.1145/337180.337228.

2 Joel Galenson, Philip Reames, Rastislav Bodik, Björn Hartmann, and Koushik Sen. Codehint: Dynamic and interactive synthesis of code snippets. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 653–663, New York, NY, USA, 2014. ACM. doi:10.1145/2568225.2568250.

3 Tihomir Gvero, Viktor Kuncak, and Ruzica Piskac. Interactive synthesis of code snippets. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 418–423. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-22110-1_33.

4 Miryung Kim, Lawrence Bergman, Tessa Lau, and David Notkin. An ethnographic study of copy and paste programming practices in oopl. In *Empirical Software Engineering, International Symposium on*, pages 83–92, Aug 2004. doi:10.1109/ISESE.2004.1334896.

5 Torben Lorenzen, Lee Mondshein, Abdul Sattar, and Seikyung Jung. A code snippet library for cs1. *ACM Inroads*, 3(1):41–45, March 2012. doi:10.1145/2077808.2077822.

6 Zachary Parker, Scott Poe, and Susan V. Vrbsky. Comparing nosql mongodb to an sql db. In *Proceedings of the 51st ACM Southeast Conference*, ACMSE '13, pages 5:1–5:6, New York, NY, USA, 2013. ACM. doi:10.1145/2498328.2500047.

7 Naiyana Sahavechaphan and Kajal Claypool. Xsnippet: Mining for sample code. *SIGPLAN Not.*, 41(10):413–430, October 2006. doi:10.1145/1167515.1167508.

8 Doug Wightman, Zi Ye, Joel Brandt, and Roel Vertegaal. Snipmatch: Using source code context to enhance snippet retrieval and parameterization. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 219–228, New York, NY, USA, 2012. ACM. doi:10.1145/2380116.2380145.