

A SYSTEM, METHOD AND COMPUTER READABLE MEDIUM FOR SIMULATING APPLICATION PROGRAMMING INTERFACE

TECHNICAL FIELD

[0001] The present disclosure relates generally to a system, method and computer readable medium for simulating application programming interface (API) using a mock-response concept. Embodiments of the disclosure are related to a computer simulator that provides customizable response header, response delay, and response HyperText Transfer Protocol (HTTP) code, using a mock-response concept.

BACKGROUND

[0002] Application Programming Interfaces (APIs) enable the exposure of content or services to internal or external audiences by developers. APIs are certain predetermined functions, protocols or routines enabling application programs and developers to access a set of routines based on a hardware or software without requiring a need to access source code or understanding the details of internal workings. For example, to access data and services of a backend server, a user device makes requests using an API call and the desired data or services are provided in response.

[0003] APIs allow developers to create applications that rely on other applications and services without having to constantly update the application when the internals of the dependent applications or services change. As long as the API itself remains stable, the internal implementation is configurable.

[0004] A common difficulty faced by developers in the process of API deployment is with respect to testing the design of the API before the API is fully implemented. At present, developers are unable to manually deploy the API to a live server for

testing without first designing the functions and protocols of the API and then coding its implementation. If the API design is found to be incorrect after testing, the developer has to go back and not only modify the API definition, but also modify the implementation code. Thus, there is a need for a more efficient API development and testing environment.

[0005] Further, developers commonly use a JavaScript Object Notation (JSON) server which is a node module used to create demonstration Representational state transfer (REST) JSON web services. A key concept of JSON server is to maintain a dataset in a single JSON file. The dataset must be a full set of data following RestfulAPI standard. Thus, there is no way to call an API of a certain resource without creating the parent resource in the dataset in advance. During the development testing period, any standard HTTP methods such as PUT, DELETE, PATCH, or POST API calls change the data stored in that dataset. Hence, the developer is unable to receive the same response when calling the same API after a time interval, which is very inconvenient for debugging. Moreover, the developer needs to reset the dataset after a round of testing. Additionally, the developer is unable to customize the response header, response delay, and response HTTP code on a JSON server.

SUMMARY

[0006] The present disclosure relates generally to a system, method and computer readable medium for simulating API using a mock-response concept. Embodiments of the disclosure are related to a system, method and computer readable medium that provide customizable response header, response delay, and response HyperText Transfer Protocol (HTTP) code, using a mock-response concept.

[0007] The methods, systems and computer readable media disclosed in the present application are advantageous in several ways, including but not limited to,

providing a simulator which is easy and efficient, reduces the overall time spent on simulating API responses with flexible configuration, frees the developer from maintaining a full set of RESTful standard datasets, enables the developer to receive the same response when calling the same API after a time interval thereby leading to efficient debugging, allowing the developer to design more response components such as method, URL, delay time, response HTTP code or response header, which is currently not possible in many servers, accepts regular expression for matching of routes which allows simulation of multiple API calls whose URL can be expressed by the same regular expression and also allows reuse of the same response data.

[0008] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium that is developed based on an existing server, for example a JSON server. The simulator uses a mock-response concept and has two different ports: for simulation and for development of projects.

[0009] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium that enables developers to receive the same response when calling the same API after a time interval thereby leading to efficient debugging and enables the same response data to be optionally reused by other APIs.

[0010] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium in which the simulator only reads and never writes back to the response data file and makes the response data of the simulator independent of each other. Such design frees the developer from maintaining a full set of RESTful standard datasets.

[0011] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium in which the developer can design more response components, not limited to, method, URL, delay time, response HTTP

code or response header. Embodiments of the present invention disclose a simulator wherein the matching of routes can accept regular expression, which allows developers to use only one row in a routes file to simulate multiple API calls whose URL can be expressed by the same regular expression and could reuse the same response data. The simulator as disclosed in the present invention is easy to use and can reduce the overall time spent on simulating API responses with flexible configuration.

[0012] In a first aspect of the present disclosure, a system for simulation of Application Programming Interface (API) using a server is disclosed. The system comprises at least one port configured for a simulator to perform simulation, such that, the simulator comprises: at least one configuration file comprising information about at least one port and at least one router; and at least one simulator file to listen to at least one API call. The system further comprises at least one port configured for a project development, such that, the project development comprises: at least one mock response folder comprising at least one response data file; and at least one routes file comprising a list of API information.

[0013] According to an embodiment in conjunction to the first aspect of the present disclosure, the server is a JSON server.

[0014] According to an embodiment in conjunction to the first aspect of the present disclosure, the simulator file is used to prepare and send a response to the project development.

[0015] According to an embodiment in conjunction to the first aspect of the present disclosure, the list of API information comprises one or more of API method, Universal Resource Locator (URL), response, delay, HTTP code or HTTP headers.

[0016] According to an embodiment in conjunction to the first aspect of the present disclosure, each response data file comprises a response data for the API.

[0017] According to an embodiment in conjunction to the first aspect of the present disclosure, the response data of one API can be used by one or more APIs.

[0018] According to an embodiment in conjunction to the first aspect of the present disclosure, the router information is a list of absolute paths of one or more routes files of one or more projects.

[0019] In a second aspect of the present disclosure, a method for simulation of Application Programming Interface (API) using a server is disclosed. The method comprises starting up a simulator; receiving, by the simulator, a HTTP request from a project development; finding, by the simulator, a matched route to a routes file from a configuration file; obtaining, by the simulator, at least one API information from the routes file; preparing, by the simulator, a response based on the API information; and sending, by the simulator, the response to the project development.

[0020] According to an embodiment in conjunction to the second aspect of the present disclosure, the server is a JSON server.

[0021] According to an embodiment in conjunction to the second aspect of the present disclosure, starting up the simulator comprises: adding, to the configuration file of the simulator, an absolute path of the routes file of the project development; preparing, in the routes file, the API information; storing, in a mock response folder, at least one response data file; and sending, by the project development, the HTTP request to a port of the simulator.

[0022] According to an embodiment in conjunction to the second aspect of the present disclosure, the matched route to the routes file from the configuration file

is obtained by: reading, by the simulator, the routes file in the configuration file; finding, by the simulator, the routes file to match the API information; forming, by the simulator, an absolute path of the response data file using the absolute path of the routes file; and assigning, by the simulator, the absolute path of the response data file as the matched route.

[0023] According to an embodiment in conjunction to the second aspect of the present disclosure, the method may further comprise storing a content of the routes file as variable “routes” which are string or array variables.

[0024] According to an embodiment in conjunction to the second aspect of the present disclosure, if the matched route is found in the configuration file, a corresponding value of the matched route is set to a response data, a delay time and a HTTP code.

[0025] According to an embodiment in conjunction to the second aspect of the present disclosure, if more than one matched route is found in the configuration file, the corresponding value of a first matched route is set to the response data, the delay time and the HTTP code.

[0026] According to an embodiment in conjunction to the second aspect of the present disclosure, if the matched route is not found in the configuration file, a URL expression is used to find the matched route and the corresponding value is set to response data, the delay time and the HTTP code.

[0027] According to an embodiment in conjunction to the second aspect of the present disclosure, if the matched route is not found in the configuration file, a default value is set to the response data, the delay time and the HTTP code.

[0028] According to an embodiment in conjunction to the second aspect of the present disclosure, the URL expression supports regular expression.

[0029] In a third aspect of the present disclosure, a computer readable medium storing a program comprising instructions that, when executed by a system, cause the system to perform the method for simulation of Application Programming Interface (API) using a server is disclosed. The instructions, when executed by the system comprise, starting up a simulator; receiving, by the simulator, a HTTP request from a project development; finding, by the simulator, a matched route to a routes file from a configuration file; obtaining, by the simulator, at least one API information from the routes file; preparing, by the simulator, a response based on the API information; and sending, by the simulator, the response to the project development.

[0030] According to an embodiment in conjunction to the third aspect of the present disclosure, the server is a JSON server.

[0031] According to an embodiment in conjunction to the third aspect of the present disclosure, starting up the simulator comprises: adding, to the configuration file of the simulator, an absolute path of the routes file of the project development; preparing, in the routes file, the API information; storing, in a mock response folder, at least one response data file; and sending, by the project development, the HTTP request to a port of the simulator.

[0032] According to an embodiment in conjunction to the third aspect of the present disclosure, the matched route to the routes file from the configuration file is obtained by: reading, by the simulator, the routes file in the configuration file; finding, by the simulator, the routes file to match the API information; forming, by the simulator, an absolute path of the response data file using the absolute path of the routes file; and assigning, by the simulator, the absolute path of the response data file as the matched route.

[0033] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further comprise, storing a content of the routes file as variable “routes” which are string or array variables.

[0034] According to an embodiment in conjunction to the third aspect of the present disclosure, if the matched route is found in the configuration file, a corresponding value of the matched route is set to a response data, a delay time and a HTTP code.

[0035] According to an embodiment in conjunction to the third aspect of the present disclosure, if more than one matched route is found in the configuration file, the corresponding value of a first matched route is set to the response data, the delay time and the HTTP code.

[0036] According to an embodiment in conjunction to the third aspect of the present disclosure, if the matched route is not found in the configuration file, a URL expression is used to find the matched route and the corresponding value is set to response data, the delay time and the HTTP code.

[0037] According to an embodiment in conjunction to the third aspect of the present disclosure, if the matched route is not found in the configuration file, a default value is set to the response data, the delay time and the HTTP code.

[0038] According to an embodiment in conjunction to the third aspect of the present disclosure, the URL expression supports regular expression.

[0039] The following disclosure is provided to facilitate an understanding of some of the innovative features unique to the disclosed embodiment and is not intended to be a full description. A full appreciation of the various aspects of the embodiments disclosed herein can be gained by taking into consideration the entire specification, claims, drawings, and abstract as a whole.

BRIEF DESCRIPTION OF THE DRAWINGS

[0040] **FIG. 1** is a block diagram depicting an example system **100** for simulating API, in accordance with the disclosed embodiments;

[0041] **FIG. 2** is a diagrammatic representation of an example data structure of a routes file **300** containing a list of API information, in accordance with the disclosed embodiments;

[0042] **FIG. 3** is a diagrammatic representation of an example configuration file **520**, in accordance with the disclosed embodiments;

[0043] **FIG. 4** is a flow diagram that illustrates an example process **600** for initiating a method for simulation using the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments;

[0044] **FIG. 5** is a flow diagram that illustrates an example process **700** for performing a method for simulation using the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments;

[0045] **FIG. 6** is a flow diagram that illustrates an example process **800** for finding a matched route from a configuration file in a method for simulation using the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments; and

[0046] **FIG. 7** is a flow diagram that illustrates an example process **900** for preparing and sending a response in a method for simulation by the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments.

[0047] The summary above, as well as the following detailed description of illustrative embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the present disclosure, exemplary constructions of the disclosure are shown in the drawings. However, the disclosure is not limited to specific methods and instrumentalities disclosed herein. Moreover, those in the art will understand that the drawings are not to scale. Wherever possible, like elements have been indicated by identical numbers.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0048] The particular configurations discussed in the following description are non-limiting examples that can be varied and are cited merely to illustrate at least one embodiment and are not intended to limit the scope thereof.

[0049] The present invention provides a system, method and computer readable medium for simulating APIs that is developed based on a server. Embodiments of the disclosure are related to a system, method and computer readable medium that provide customizable response header, response delay, and response HyperText Transfer Protocol (HTTP) code, using a mock-response concept. Embodiments of the disclosure also relate to a simulator that runs at a different port than the port where the project is being developed.

[0050] **FIG. 1** is a block diagram depicting an example system **100** for simulating API, in accordance with the disclosed embodiments. The system **100** comprises two separate ports, such as a simulator port **102** for simulation used by simulator **106** and project development port **104** for project developments **108**. Project development **108** comprises mock response folders **114** and routes files **116**. The routes file **116** contains a list of API information **122**, including but not limited to, URL, Representational State Transfer (RES), HyperText Transfer protocol (HTTP) code, RES headers, delay time, API method. The mock response folder **114** contains desired or expected response data for a simulator request that is stored

in a response data file **120**. For example, if the system is based on a JSON server, the mock response folder **114** may contain one or more JSON files wherein the desired or expected response data for a simulator request is stored. The simulator **106** comprises configuration files **112** and simulator files **110**. The configuration file **112** contains ports and routers information **118**, to define simulator port number and a list of absolute paths of routes files **116** of different project developments **108**. The main logic of the simulator file **110** is to listen to API calls, prepare and send responses. The present invention supports various simulations required by front-end developers. Using the embodiments of the present invention, and the system **100** disclosed in **FIG. 1**, a front-end developer can just draft the response data into a single file, add the file name into the routes file **116** with desired API method, URL, delay time, response HTTP code, response header, whichever required, and then add the path of the routes file **116** into the simulator's configuration file **112**.

[0051] It should be noted that in accordance with the disclosed embodiments, the system **100** can be a computing system, for example a computer with two different ports, one used for simulation and the other used for project development. The system **100** uses a server, for example a JSON server. The server is imported to enable the simulator to receive requests and send responses.

[0052] **FIG. 2** is a diagrammatic representation of an example data structure elements of a routes file **300** containing a list of API information, in accordance with the disclosed embodiments. As disclosed in **FIG. 2**, the example routes file **300** contains a list of API information, such as API method, Universal Resource Locator (URL), response, delay, HTTP code or HTTP headers. Developers can define all the required API calls here with desired delay time, returned http code, and response header and other configurations not limited to the above-mentioned, which are not supported by a server, for example a JSON server. The “delay” function is a sleep time before sending out a response. The “URL” function in the routes file **300** supports regular expression and hence, it is very flexible for

developers to use a single configuration to fulfill multiple simulations. For example, the configuration “/books?.*” can be used for “/books?name=Sky” as well as “/books?author=Sam”. The API “res” as disclosed in **FIG. 2** is configured with the mock response data in a response data file **120** as disclosed in **FIG. 1**.

[0053] Since the routes file **300** is written by the developers, the developers can themselves determine the configurations required, for example, whether a response header is required, and write said configurations in the routes file **300**. Developers can update the content of routes file **300** by using the example file as disclosed in **FIG. 2**. Any addition or deletion or updation of files in mock response folder **114** of **FIG. 1** can be performed by the developers based on the requirement.

[0054] Each project development comprises a mock response folder at the same hierarchy as the routes file. The mock response folder contains expected response data stored in response data files, for example, JSON files. Each file represents an expected response of an API.

[0055] Developers can put the desired response data in a response data file, for example a JSON file, each for an API. The response data in the response data files are constant and are not modified during any API call. Different APIs may return different response data. Developers can reuse the same response data for other APIs, if applicable. Front-end developers can put any data in the response data file based on the simulation needs. An advantage of the present invention is that if the back-end developer is not ready to develop the API, the front-end developer can mock the response data in response data file.

[0056] It should be noted that, different API can return different response data. The frontend developer can put any data in the mock response file based on their simulation needs. Developer will put the desired response data in a json file, each for an API. These data are constant data which will not be modified during any API call. And same response data can be reused by other APIs if applicable. For

example, considering, API /books will return a response data, in case when backend developer is not ready to develop the API, frontend developer need to mock this response data. For example, “all-books.json” is just the file that stores such mock response data. In **FIG. 2** line 5, the routes.js configured that content in “all-books.json” will be the mock response of API /books.

[0057] **FIG. 3** is a diagrammatic representation of an example configuration file **520**, in accordance with the disclosed embodiments. As disclosed in **FIG. 1**, in the simulator **106**, there is a configuration file **112** which contains ports and routers information **118**, to define simulator port number and a list providing absolute paths for routes files **116** of different project developments **108**.

[0058] The example configuration file disclosed in **FIG. 3** contains two parts, a port **502** and routers **504**. The port **502** is the port number in which the simulator is running. The routers **504** is a list of absolute paths of the routes file of different development projects. Developers can reuse a simulator by just adding a path into routers **504**.

[0059] As shown in **FIG. 3**, the example configuration file **520** has two development projects that are using the simulator containing the configuration file **520** as indicated by rows 4 and 5 in **FIG. 3**. The example configuration file **520** of **FIG. 3** discloses a list of absolute paths of routes.js files running at the port number 9000. The port number can be the one that is not occupied by any of the computer processes. Developers can add a path into the routers file by typing inside the brackets as indicated by ““routers”: [” in row 3 and “]” in row 6.

[0060] **FIG. 4** is a flow diagram that illustrates an example process **600** for initiating a method for simulation using the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 4**, the process **600** can be implemented, for example, by adding an absolute path of a routes file of a project

to the configuration file of simulator at step **602**. Then, at step **604**, the process **600** can be implemented, for example, by preparing the routes file with API information, such as API method, Universal Resource Locator (URL), response, delay, HTTP code or HTTP headers. Further, at step **606**, storing the expected response data in response data files, for example JSON files, in mock responses folder of the project. As disclosed in **FIG. 4**, the process **600** can be implemented, for example, by sending a HTTP request to a running port of the simulator at step **608**. Thereafter, at step **610**, the process includes starting, by the simulator, the method for simulation.

[0061] **FIG. 5** is a flow diagram that illustrates an example process **700** for performing a method for simulation using the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 5**, the process **700** can be implemented, for example by, starting up the simulator at step **701** and receiving by the simulator, an API call simulated by a development project to a simulator port at step **702**. Then, if the URL contains a timestamp at step **704**, said timestamp is removed from the URL parameters and the clean URL is obtained **706**. For example, if URL in request contains timestamp in pattern “&time=\d*” or “time=\d*”, such parameters are removed. If the URL does not contain a timestamp at step **704**, the example process **700** skips step **706**. Thereafter, at step **708**, finding, by the simulator, a matched route from a routes file defined in a configuration file. Thereafter, the process **700** can be implemented, for example, by obtaining the API information from the relevant routes file, at step **710**. Then, at step **712**, the system prepares a response based on the relevant routes’ configuration and at step **714**, sends back the response via a server. In some implementations, optionally, a JSON server is used by the simulator to perform the receiving and sending functions as exemplified at steps **702** and **714**, respectively.

[0062] It should be noted that the “matchedRoute” refers to the term “matched route”, “regex” refers to the term “regular expression”, “responseData” refers to the term “response data”, “delayTime” refers to the term “delay time”, “httpCode” refers

to the term “HTTP code”, “matchedRoute.delay” refers to the term “delay in matched route”, “matchedRoute.httpCode” refers to the term “ HTTP code in matched route”, “delayTime” refers to the term “delay in time”, “resHeaders” refers to the term “response headers” and “route” refers to the term route.

[0063] **FIG. 6** is a flow diagram that illustrates an example process **800** for finding a matched route from a configuration file in a method for simulation using the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 6**, the process **800** can be implemented, for example, by reading one or more routes files configured in the configuration file of the system at step **802**. Thereafter, the process includes storing the content of each routes file as variable “routes” which are string or array variables at step **804**. Any common or preferable practice can be employed by the developers for reading the content of the above-mentioned routes files and converting the “routes” to string or array variables. An example of a read file method is, “let content=fs.readFileSync(filePath);”. Where ‘fs’ is an open-sourced module that provides a method called “readFileSync”. Developers need to provide the file path to this method and the content of the file will be read and converted to variable ‘content’. A lot of similar methods/ functions that can read files and convert to variables can also be used for the above-mentioned steps without limitation. The process further includes at step **806**, finding elements in the “routes” variables to match the API information including but not limited to API method and URL.

[0064] If an element found in the “routes” variables exactly matches the API information at step **808**, the process as disclosed in **FIG. 6** includes assigning the element as “matchedRoute” at step **810**. Then, at step **812**, the process includes forming an absolute path of the response data file using the routes file path and the matched route response data file, matchesRoute.res. Thereafter, at step **814**, the process includes assigning the absolute path of the response data file as the matched route response data file, matchesRoute.res.

[0065] If elements found in the “routes” variables do not exactly match the API information at step **808**, the process includes finding the elements in the “routes” variables based on regular expression (regex) defined in each element, as disclosed in step **816**. If a regex match is found, the process includes assigning the regex matched element as “matchedRoute” as disclosed in step **810**. If more than one regex matches are found at step **816**, the first matched element can be assigned as “matchedRoute” as disclosed in step **810**. Thereafter, the process includes, for example, forming an absolute path of the response data file using the routes file path and the matched route response data file, matchesRoute.res as disclosed in step **812** and assigning the absolute path of the response data file as the matched route response data file, matchesRoute.res as disclosed in step **814**.

[0066] If as disclosed at step **818**, a regex match is not found, the process includes assigning “null” as the “matchedRoute” at step **820**.

[0067] **FIG. 7** is a flow diagram that illustrates an example process **900** for preparing and sending a response in a method for simulation by the example system **100** of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 7**, the process **900** can be implemented, for example, at step **902** by forming or preparing a responseData, delayTime and httpCode based on a matched route from a configuration file in a method for simulation as disclosed in **FIG. 7**. The default value of the variables may be set as “responseData”=null, “delayTime”=0 and “httpCode”=200. If an exact match or a regex match is found and assigned a matched route “matchedRoute”, as disclosed in **FIG. 7**, a response data file in the matched route response data file, “matchesRoute.res” is read by the system and the content is assigned to the “responseData”. If “matchedRoute.delay” and “matchedRoute.httpCode” exist, the same is assigned to “delayTime” and “httpCode”, respectively.

[0068] Then, as disclosed in **FIG. 7**, the process **900** can be implemented, for example, by setting response headers at step **904**. If `matchedRoute.resHeaders` exist, the `resHeaders` are read and headers are set to the response header.

[0069] Thereafter, at step **906**, the process **900** can be implemented by sending, by the system, the response with the defined “`httpCode`” and “`responseData`” at a delay of “`delayTime`”.

[0070] It should be noted that the system **100** can have one or more computing devices connected to perform the method for simulation of Application Programming Interface (API) using a server. A computer readable medium storing a program comprising instructions that, when executed by the system **100**, cause the system **100** to perform the method for simulation of Application Programming Interface (API) using a server of the present invention is also disclosed. The computing device can have such computer readable medium for performing the invention method.

[0071] The present invention discloses a system, method and computer readable medium for simulating application programming interface (API) using a mock-response concept. The present invention discloses two separate ports, one for a developing project and the other for the simulator. The present invention allows customizable response headers, response delay, and response HTTP code. Further, the response data sent by the simulator remains constant and unaffected and unmodified during any API call.

[0072] It will be appreciated that variations of the above disclosed and other features and functions, or alternatives thereof, may be desirably combined into many other different systems or applications. Also, various presently unforeseen or unanticipated alternatives, modifications, variations or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

[0073] Although embodiments of the current disclosure have been described comprehensively in considerable detail to cover the possible aspects, those skilled in the art would recognize that other versions of the disclosure are also possible.

CLAIMS

What is claimed is:

1. A system for simulation of Application Programming Interface (API) using a server comprising:
 - at least one port configured for a simulator to perform simulation;
 - such that, the simulator comprises:
 - at least one configuration file comprising information about at least one port and at least one router; and
 - at least one simulator file to listen to at least one API call;
 - at least one port configured for a project development;
 - such that, the project development comprises:
 - at least one mock response folder comprising at least one response data file; and
 - at least one routes file comprising a list of API information.
2. The system of claim 1, wherein the server is a JSON server.
3. The system of claim 1, wherein the simulator file is used to prepare and send a response to the project development.
4. The system of claim 1, wherein the list of API information comprises one or more of API method, Universal Resource Locator (URL), response, delay, HTTP code or HTTP headers.
5. The system of claim 1, wherein each response data file comprises a response data for the API.
6. The system of claim 1, wherein the response data of one API can be used by one or more APIs.

7. The system of claim 1, wherein the router information is a list of absolute paths of one or more routes files of one or more projects.
8. A method for simulation of Application Programming Interface (API) using a server comprising:
 - starting up a simulator;
 - receiving, by the simulator, a HTTP request from a project development;
 - finding, by the simulator, a matched route to a routes file from a configuration file;
 - obtaining, by the simulator, at least one API information from the routes file;
 - preparing, by the simulator, a response based on the API information; and
 - sending, by the simulator, the response to the project development.
9. The method of claim 8, wherein the server is a JSON server.
10. The method of claims 8 and 9, wherein starting up the simulator comprises:
 - adding, to the configuration file of the simulator, an absolute path of the routes file of the project development;
 - preparing, in the routes file, the API information;
 - storing, in a mock response folder, at least one response data file; and
 - sending, by the project development, the HTTP request to a port of the simulator.
11. The method of claims 8 to 10, wherein the matched route to the routes file from the configuration file is obtained by:
 - reading, by the simulator, the routes file in the configuration file;
 - finding, by the simulator, the routes file to match the API information;
 - forming, by the simulator, an absolute path of the response data file using the absolute path of the routes file; and
 - assigning, by the simulator, the absolute path of the response data file as the matched route.

12. The method of claim 11 further comprising, storing a content of the routes file as variable "routes" which are string or array variables.
13. The method of claims 8 to 12, wherein if the matched route is found in the configuration file, a corresponding value of the matched route is set to a response data, a delay time and a HTTP code.
14. The method of claims 13, wherein if more than one matched route is found in the configuration file, the corresponding value of a first matched route is set to the response data, the delay time and the HTTP code.
15. The method of claims 8 to 12, wherein if the matched route is not found in the configuration file, a URL expression is used to find the matched route and the corresponding value is set to response data, the delay time and the HTTP code.
16. The method of claim 15, wherein if the matched route is not found in the configuration file, a default value is set to the response data, the delay time and the HTTP code.
17. The method of claims 15 and 16, wherein the URL expression supports regular expression.
18. A computer readable medium storing a program comprising instructions that, when executed by a system, cause the system to perform the method for simulation of Application Programming Interface (API) using a server of any of claim 8 to 17.
19. A computing device comprising the computer readable medium of claim 18.