УДК 004.4’2

ENSURING OF WEB SERVICES SCALABILITY FOR “API FIRST” ARCHITECTURE

YEGOSHYNA G., VORONOY S., PALII O.

O. S. Popov Odesa National Academy of Telecommunications
1 Kuznechna St., Odesa, 65029, Ukraine
yegoshyna@onat.edu.ua

ЗАБЕЗПЕЧЕННЯ МАСШТАБОВАНОСТІ ВЕБ-СЕРВІСІВ ДЛЯ АРХІТЕКТУРИ «API FIRST»

ЄГОШИНА Г.А., ВОРОНОЙ С.М., О.Г. ПАЛІЙ

Одеська національна академія зв'язку ім.. О.С. Попова,
1, вул. Кузнечна, м. Одеса, 65029, Україна
yegoshyna@onat.edu.ua

Abstract. *The article considers a possibility of solving the problem of improving scalability index of web applications developed in accordance with the API-first strategy. It is shown that the task of increasing the indicator of software portability is the most relevant for the development of mobile applications. It analyzes the current trend of transferring the main business logic of applications from user devices to the network, which reduces the time spent on developing the client part of the application and greatly simplifies the process of its implementation. These processes affect the development of web-based applications also. The features of using the development strategies “desktop-first” and “mobile-first” are considered. The advantages of using the API-first concept and architecture features of the API-first web service architecture are considered. All requests for web services are carried out through a single and locally standardized web API. The resource-intensive web pages formation can be made in a separate structure hosted on the same machine or in a separate web server. In conditions of complete independence between the implementation of data representations of the server part and the user interface of the client part, it turns out to be impossible to provide instant access to the entire user audience in the updated list of application functions. The proposed solution is based on the using of an intermediate system for dynamic generation of the user interface for the building of the web interface. At the same time, this module should be general both for the further user interface building in the form of a web page, and for its presentation using a dynamic interface assembled from graphic components provided by the application operating system.*

Keywords: *web-service, API-first, module, interface, web-page, application, API-client, API-server, shared user interface.*

Анотація. *У статті розглянуто можливість вирішення проблеми поліпшення показника масштабованості веб-додатків, які розробляються у відповідності зі стратегією «API-first». Показано, що задача підвищення показника портативності програмного забезпечення є найбільш актуальною для сфери розробки мобільних додатків. Аналізується сучасна тенденція переносу основної бізнес-логіки додатків з користувальницьких пристроїв у мережу, що дозволяє скоротити часові витрати на розробку клієнтської частини додатку та значно спрощує процес її реалізації. Встановлено, що зазначені процеси в тій же мірі впливають і на розвиток веб-орієнтованих додатків, поступово стираючи грань між веб-сервісами та прикладними програмами для десктопних або мобільних систем. Розглянуто особливості використання стратегій розробки «desktop-first» та «mobile-first». Показані переваги використання концепції «API-first» та розглянуто особливості архітектури «API-first» веб-сервісу, в якій всі запити на виконання послуг веб-сервісу здійснюються через єдиний, локально стандартизований веб-API інтерфейс, а ресурсоємне формування веб-сторінок виноситься в окрему структуру на тому ж хості або на окремий веб-сервер. Встановлено, що в умовах повної незалежності між реалізацією представленнями даних (API) серверної частини та призначеного для користувача інтерфейсу клієнтської частини програми, виявляється неможливим гарантувати миттєвий доступ всій користувальницької аудиторії до оновленого списку функцій додатку. Запропоноване рішення базується на використанні проміжної системи динамічної генерації користувальницького інтерфейсу для формування веб-інтерфейсу сторінок. При цьому даний модуль повинний бути загальним і для подальшого формування інтерфейсу користувача у вигляді веб-сторінки, і для її представлення за допомогою динамічного інтерфейсу, зібраного з графічних компонентів, що надаються операційною системою додатка.*

Ключові слова *веб-сервіс, API-first, модуль, інтерфейс, веб-сторінка, додаток, API-клієнт, API-сервер, shared user interface*

INTRODUCTION

A widespread trend in recent years is the desire to increase the portability of software - a qualitative indicator responsible for the ability to use the same application in different application runtimes. This process is most revealing in the field of mobile applications. If at the beginning of the century the majority of application programs were intended for desktop computers, then, with the development of mobile devices, the convergence of applications for desktop computers and portable devices is already progressing at the level of the environment.

Web services architecture is currently one of the most topical issues in software development since became actually the standard for interaction various applications operating on heterogeneous platforms [1–3]. A concomitant process is the tendency to transfer the main “logic” of applications from the user device to the network. If earlier most of the application software products were completely independent applications, now you can observe a tendency to shift the main part of applications to cloud platform servers. This approach can significantly reduce the development time of the remaining client part applications for each of the target platforms. Instead of a complete list of application functionality for each specific platform we have to adapt only the basic functions of the formation and management of the user interface, as well as their interaction with the server part of the application through information networks.

However, it is much more important that these processes affect the development of initially web-based applications to the same extent, gradually erasing the line between web services and application programs for desktop or mobile systems.

The scope of this work is to investigate a possibility of the extensibility index improving for web applications developed in accordance with the API-first strategy by providing instant availability of the functionality introduced by the developer of the server application in any client applications that support the proposed dynamic user interface formation scheme.

“DESKTOP-FIRST” VS “API-FIRST” WEB SERVICE ARCHITECTURE

As classic approaches to designing web services development strategies such as “desktop-first” and “mobile –first” can be distinguished. In the “desktop-first” case the application is developed for clients of stationary systems and compatibility with mobile devices is provided as necessary, but only after the main part of the project is completed. “Mobile –first” case the same behavior, but with a priority for mobile platforms. Later the strategy called “API-first” was developed, which is a kind of synergistic combination of the first two strategies and is built around the concept of “web application programming interface”.

Consider an example of a classic web application built on the basis of the MVC design pattern (see Figure 1) [4].

Within this interaction model client applications can be divided into three categories: web client (web browser); installed client application for desktop computers; installed client application for mobile devices.

As a response to a request from the user arriving at the controller the requested data is selected from the database of the server application, its processing and structuring of information in the model. Then, depending on whether the request came from the web client or the application being installed, the information is transmitted to the web presentation module (where the corresponding HTML page is formed and sent to the client side browser), or to the API presentation module.

As a rule, all structures of the server side (with the exception of the database and file resources of the web view) are on the same server. Thus, the load from processing the web and API views on one device is added up.

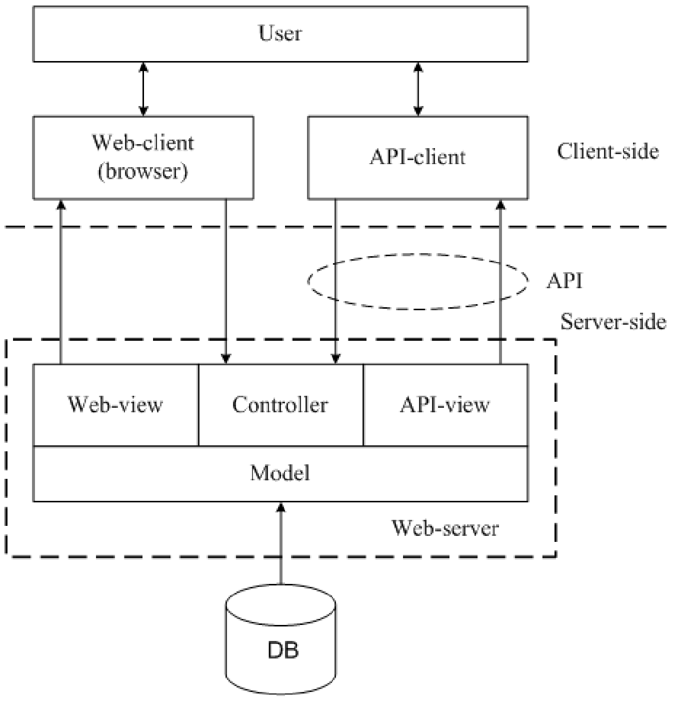

Figure 1 – Example “desktop-first” web service architecture

This fact can negatively affect the speed of forming the API server responses (as a rule, interaction via API consumes fewer computing and network resources), causing unnecessary delays in the operation of the web service for mobile devices.

In turn, an application built in accordance with the concept of "API-first" could be represented by the following architecture (see Figure 2).

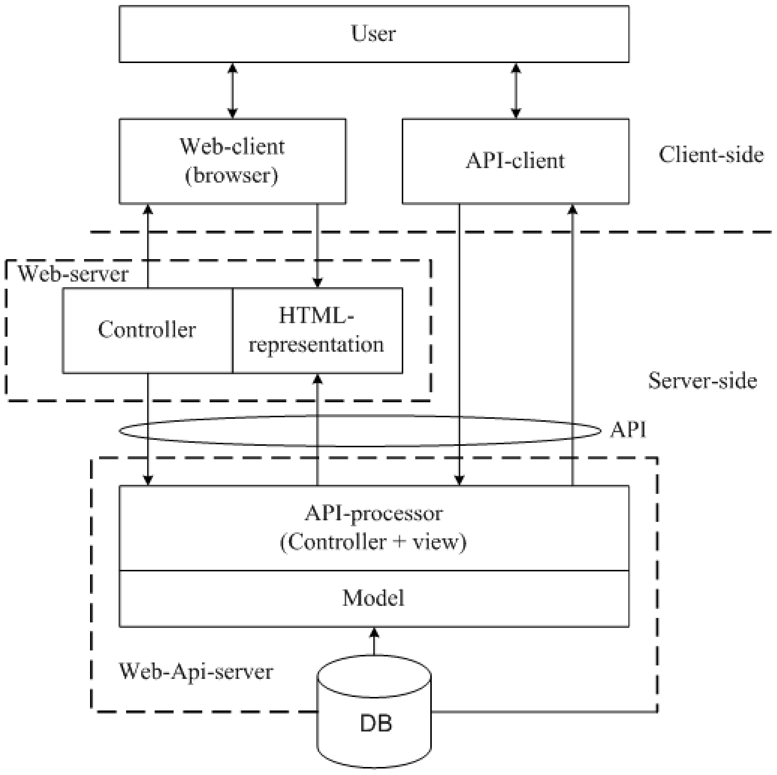

Figure 2 – Example “API-first” web service architecture

In the case of the “API-first” an implementation of an API node is possible [6]. It implements the entire business logic of the application. All requests for web services are carried out through a single and locally standardized web API. The resource-intensive web pages formation can be made in a separate structure hosted on the same machine or in a separate web server. This allows not only to reduce the load on the main server, but also to delegate the development of not only installed but also web clients of the web service to a third-party development team.

Also it is quite simply to implement it on a platform different from the main server (for example, the main server application can be implemented as a compiled persistent Java application hosted on Debian, while a web-view is compiled on a non-persistent application run by PHP on another operating system).

The disadvantage of both architectures is that with the increase in mobility and portability of each of their subsystems, their extensibility is significantly reduced. In conditions of complete independence between the implementation of the data representation (API) of the server part and the user interface presentation of the application client part, it is impossible to guarantee instant access to the entire user audience of the application to the updated list of its functions.

So connecting a new functional module to the server side of the application starts the process of updating client applications of the web service, which can be quite lengthy, especially if the client application is exclusively in the area of responsibility of third-party developers. In the case of the "classical" application structure, this only affects the list of client applications being installed (since the application architecture itself provides for the dynamic generation of the user interface based on the existing functionality for web clients). In the case of “API-first” architectures, the delay extends to the development of the web presentation of the program also.

API FIRST WEB SERVICE ARCHITECTURE WITH SUPPORT FOR SUI

The main goal is to formulate an approach of designing such applications that would provide the web service developer of the server side with the opportunity to determine not only the structure and format of the data provided by the client part, but also their presentation. However, this approach should not limit the development of client applications in the possibilities of creating their own user interface and a set of web service functions in their projects [5].

The proposed solution is creating of an intermediate system for dynamic generation of the user interface similar to "classical" approaches [6] for generating the web presentation of the page interface using HTML markup language tools (see Figure 3).

The difference is that the developed system should be general for further presentation of the user interface as a web page, and for its presentation through the compiled application interface, that was dynamically assembled from the provided graphic components of the operating system.

The inability to use HTML for these purposes based on the fact that HTML was and remains a means for the final presentation of data, while the formation of a user interface with dynamic content requires an approach that is much closer to the concept of page templates creating.

The advantage of this presentation method is that, in contrast to the “classical” architecture of the page’s description distributed by the web-view, the proposed SUI-view (“shared user interface”) allows the user to control the dynamic generation of the user application interface with server side.

Also this presentation method has a much smaller additional load on the application API server. Thus, SUI is not a form of data presentation, which means that the presentation will not change with every change in the transmitted content and could be transmitted once - at the beginning of the client-server interaction session, or with the planned update of the SUI representation from the client.

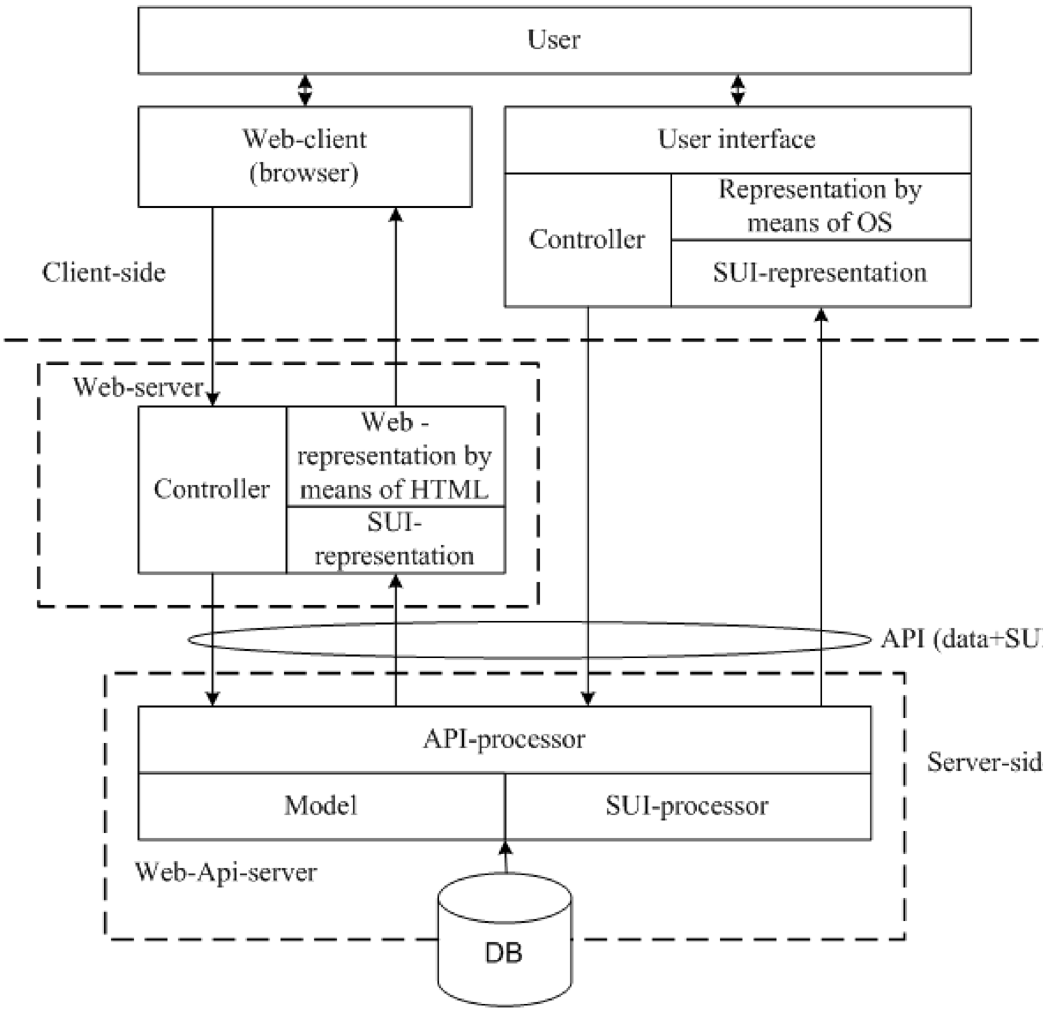

Figure 3 – Example API First Web Service Architecture with support for SUI

Thus, process of updating client systems to expand the functionality of the API server with new functions, based on the SUI system, will be performed in the following order:

– to perform a connecting of a new functional module in the web service server application;

– the new web APIs are defined in the corresponding table of the API server (automatically or manually); the table version field is incremented by one;

– the API client (whether it is a web server or an installed client application) requests the status of the API server;

– if the version of the table of APIs is higher than that used (the data was found in the response), the client requests updates;

– if the developer has implemented support for the SUI system in the client, the request also contains a note about the need to update SUI representations that use the updated functions;

– the server generates a response from the list of web interfaces of the new system functions, a compressed package of SUI representations of the user interfaces using them and sends them to the client;

–the client application updates the saved interface template in accordance with the received SUI descriptions and is ready to immediately provide the user with access to the updated functions of the web service (without any additional coordination from its developer).

Depending on the client part developer tasks, support for the SUI-system could be limited to a specific list of application functional modules, or disabled at all (for example, if a highly specialized application using a strictly limited list of web service functions is being developed, or it implements independent and unique user interface).

On the other hand, the developer gets the opportunity to focus on specific ways of representing the described SUI user interface in a specific application runtime leaving the task of logically linking system functions and specific interface elements of the application form to the SUI module.

CONCLUSION

There is considered a way to increase the extensibility of web services by reducing the delay between the implementation of new functionality modules on the server and client side of the application, which becomes achievable using the dynamic user interface generation system proposed in the work.

The proposed system should:

– to serve as a link between the user interface of the application and the web service functions defined by the APIs in accordance with the API First Web Service SUI Architecture;

– to be suitable for creating descriptions of graphical user interfaces of an application without being tied to any particular runtime, programming language, or type of application being developed;

– to expand the capabilities of developers in order to create portable and extensible web services with the ability to control the user interface structure of client applications from the server.

REFERENCES

1 Fujita S. Dynamic Collaboration of Businesses Using Web-services // NEC Journal for advanced Technology. – Vol. 1. – № 1. – P. 36–42.

2 Kaâniche M., Kanoun K., Martinello M. A User Perceived Availability Evaluation of a Web Based Travel Agency // Proc. of IEEE Computer Society 2003 International Conf. on Dependable Systems and Networks (DSN 2003), 22-25 June 2003. – San Francisco, CA, USA, – 2003. – P. 125–134

3 Web Services Activity Statement // W3C Process Document. <https: //www.w3.org/2002/ws/Activity > accessed Oct. 22 2019.

4 Leonard Richardson. RESTful Web APIs 1st Edition. – O'Relly Media, 2013. – 406 pages.

5 Yegoshyna G.A., Voronoy S.M. Intellectualization of project management web services based on integration with natural language processing modules // *Наукові праці ОНАТ ім. О.С. Попова*. – 2019. С. – 94-101.

6 Fielding R. T. 'REST APIs must be hypertext-driven <http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven> accessed Oct 22 2019.

REFERENCES

1 Fujita, S. "Dynamic Collaboration of Businesses Using Web-services." *NEC Journal for advanced Technology,* Vol. 1, No. 1, pp. 36–42.

2 Kaâniche, M., K. Kanoun, and M. Martinello. "A User Perceived Availability Evaluation of a Web Based Travel Agency" *Proc. of IEEE Computer Society 2003 International Conf. on Dependable Systems and Networks (DSN 2003).* San Francisco, CA, USA, 22-25 June 2003, pp. 125–134.

3 Web Services Activity Statement: W3C Process Document. https://www.w3.org/2002/ws/Activity accessed Oct. 22 2019.

4 Richardson, Leonard. *RESTful Web APIs.* 1st Edition, O'Relly Media, 2013.

5 Yegoshyna, G.A., and S.M. Voronoy. "Intellectualization of project management web services based on integration with natural language processing modules" *Naukovi pratsi ONAZ im. O.S. Popova,* 2019, p 94-101.

6 Fielding, R. T. "REST APIs must be hypertext-driven." http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven. Accessed 22 Oct 2019.