CONFERENCE PROCEEDINGS

МАТЕРІАЛИ КОНФЕРЕНЦІЇ

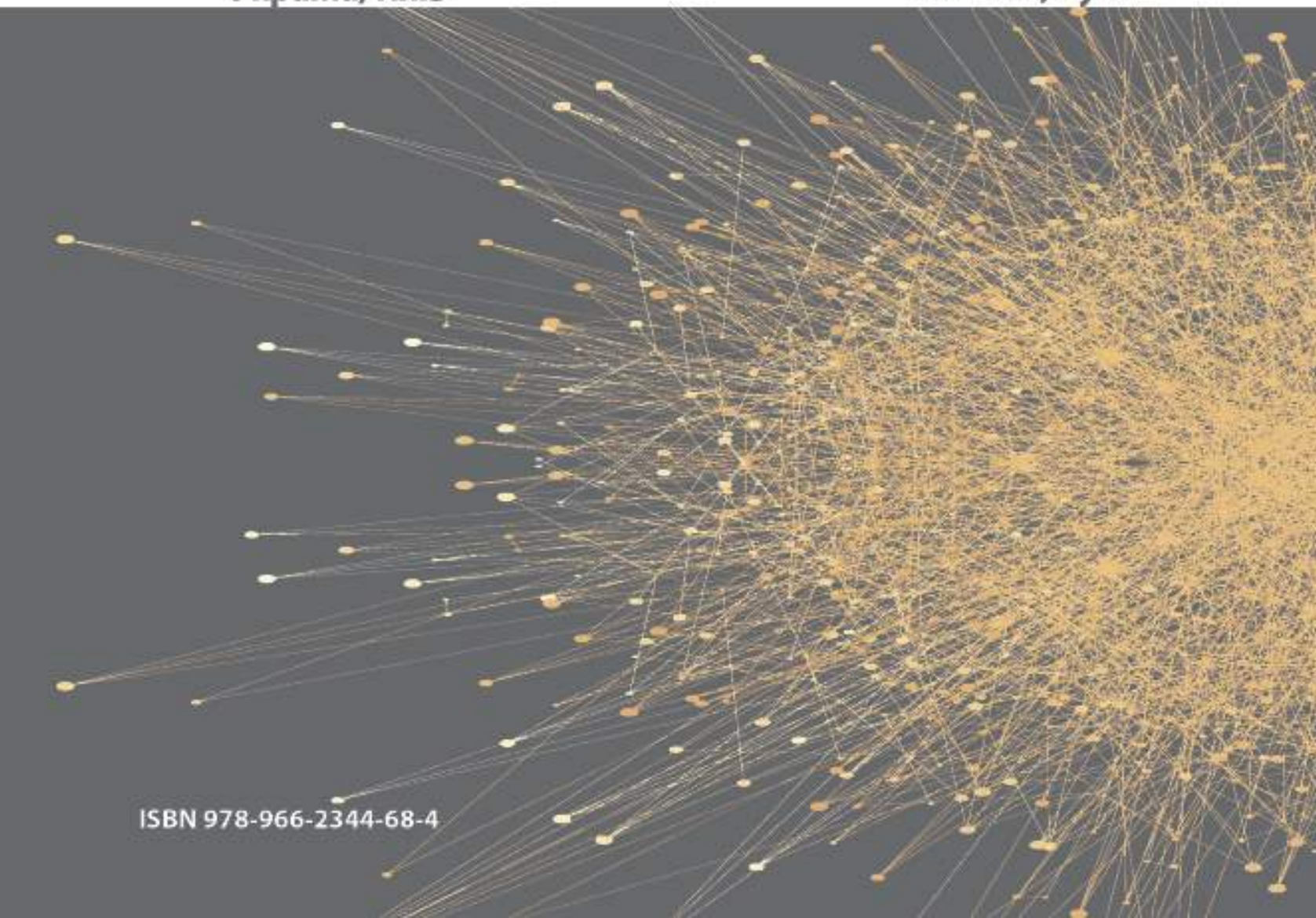
IX МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
З ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Summer InfoCom Advanced Solutions 2020

9th INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE
ON INFORMATION SYSTEMS AND TECHNOLOGIES

20-21 травня 2020 року
Україна, Київ

May 20-21, 2020
Ukraine, Kyiv



ISBN 978-966-2344-68-4

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНСТИТУТ МОДЕРНІЗАЦІЇ ЗМІСТУ ОСВІТИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

SUMMER INFOCOM ADVANCED SOLUTIONS 2020

МАТЕРІАЛИ

IX МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

З ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

CONFERENCE PROCEEDINGS

9th SCIENTIFIC AND PRACTICAL CONFERENCE

КИЇВ, УКРАЇНА

20-21 травня 2020 року

УДК 004

Summer InfoCom 2020: Матеріали IX Міжнародної науково-практичної конференції з інформаційних систем та технологій, м. Київ, 20-21 травня 2020 р. – К.: Вид-во ТОВ "Інжиніринг", 2020. – 46с. – Мови укр., рос., англ.

Конференція входить до Переліку міжнародних та всеукраїнських науково-практичних конференцій здобувачів вищої освіти та молодих учених у 2020 році.

Редакційна колегія дотримується прийнятих міжнародною спільнотою принципів публікаційної етики, відображених, зокрема, в рекомендаціях Комітету з етики наукових публікацій (Committee on Publication Ethics, COPE), а також враховує досвід авторитетних міжнародних видавництв. Щоб уникнути недобросовісної практики в публікаційній діяльності (плагіат, виклад недостовірних відомостей та ін.), з метою забезпечення високої якості наукових публікацій, визнання громадськістю отриманих автором наукових результатів, кожен член редакційної колегії, автор, рецензент, видавець, а також установи, які беруть участь в видавничому процесі, зобов'язані дотримуватися етичних стандартів, норм і правил та вживати всіх можливих заходів для запобігання їх порушень. Дотримання правил етики наукових публікацій усіма учасниками цього процесу сприяє забезпеченню прав авторів на інтелектуальну власність, підвищенню якості видання і виключення можливості неправомірного використання авторських матеріалів в інтересах окремих осіб.

ISBN 978-966-2344-68-4

ПРОГРАМА КОНФЕРЕНЦІЇ

ПРОГРАМА/PROGRAM

20 травня

Інформаційні системи та технології

Тюрін В. **Савчук О.**	Автоматична система генерації запитів до бази даних на основі асинхронних операцій
Bodak B. **Doroshenko A.**	The implementation of RESTful API for social media events platform
Niechkina V.	Criterion for signal smoothing and jump prediction using extrapolation in real-time systems
Теленик А.	Виправлення помилок системних ресурсів при віддаленій інсталяції

Оброблення інформації у складних системах

Syrotkina O. **Aleksieiev M.** **Udovyk I.**	Methods of Optimizing the Diagnostic Data Processing for Information Management Systems

21 травня

Технології програмування

Драбинко В. **Дорошенко А.**	Методика ефективної реалізації односторінкових веб-додатків
Богомол Р.	Централізоване керування музичними сервісами
Коваленко І. **Вовчок Є.**	Система тестування на основі відносного оцінювання по статистичним вибіркам
Telenyk S. **Zharikov E.** **Vovk Y.** **Nowakowski G.** **Tokmenko O.**	Bot creating technology for wide use based on a logical approach

ЗМІСТ/CONTENTS

ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

INFORMATION SYSTEMS AND TECHNOLOGIES

Автоматична система генерації запитів до бази даних на основі асинхронних операцій

Тюрін Валерій
КПІ ім. Ігоря Сікорського
Київ, Україна
tiurinvalery@gmail.com

Савчук Олена
КПІ ім. Ігоря Сікорського
Київ, Україна
savchuk_l1@ukr.net

Анотація. **Робота присвячена розробці автоматичної системи генерації запитів до бази даних на основі асинхронних операцій. Розроблена бібліотека відкритих джерел в нереляційній базі даних та інтегрований застосунок для її тестування.**

Ключові слова: автоматична система, база даних, асинхронні операції.

Вступ

Хмарне обчислення дає можливість широкомасштабного спільного доступу до послуг, що дозволяє користувачам отримувати доступ до технологічних сервісів без знання, досвіду та контролю над технологічною інфраструктурою, яка їх підтримує. Через зростання загальної кількості інформації, виникає необхідність у більш оптимальних системах, що можуть обробляти таку кількість інформації. Системи, що засновані на неблокуючих операціях – є основними в цій гаулзі.

При розробці архітектурної схеми бібліотеки була обрана вірна архітекутра, яка базувалася на шаблонах проєктування для асинхрониих систем. Були обрані раціональні технології, які відповідають основним крітеріям – це безкоштовність, підтримуваність, сучасність, швидкодія, надійність. Завдяки слідуванню сучасним стандартам, набір технологій є звичним та зрозумілим фахівцям, які працюють з хмарними та Java орієнтованими застосунками [1-3]. Усі елементи, за виключенням DynamoDB [4], є безкоштовними, тож заміняючи хмарну DynamoDB локальною версією для розробників, було можливо створити проєкт без додаткових витрат на оплату інших компонентів та розмістити його як Open Source.

Завдяки використанню Spring Boot Framework, бібліотека є платформо-незалежною та легкою для розгортання завдяки вбудованому контейнеру [5-7]. Правильний вибір технологій дозволив створити потужну систему модульного, інтеграційного та навантажувального тестування, що дозволяє підтримувати належну якість застосунку.

Опис процесу генерації запитів застосунку

Процес генерації запитів складається з наступних етапів:

1) впровадження клієнтським застосунком об’єкта, що відповідальний за зв'язок з БД (створювати цей бін робота саме клієнта, бо це прямо впливає на безпеку PII даних); 2) пошук кандидатів (на основі власної анотації над класами бібліотека розуміє, для яких класів потрібно буде генерувати запити); 3) створення метадати для всіх знайдених кандидатів; 4) побудова запитів на основі створеної метадати.

Вибір різних та оптимальних мов програмування для різних задач підвищує якість проєкту, а їх гармонічне поєднання (Scala є дуже схожею на Java) не створює додаткових складнощів для проєкту [8]. Орієнтованість на AWS застосунки, як на цільову аудиторію користувачів, є влучним вибором, бо AWS є найпопулярнішим хмарним провайдером сьогодення, що забезпечує широке коло потенційних користувачів.

Наступним кроком клієнтський застосунок має додати у контекст DynamoDbAsyncClient бін, який має всередені налаштування параметрів підключення до бази даних, такі як accessKey, secretKey та region. Створити цей бін - задача саме клієнтського застосунку для того, щоб бібліотека не залежала від параметрів підключення і не потребувала знань приватних даних – це забезпечує гнучкість системи.

Наступним кроком необхідно вказати значення пакету або пакетів, які програма має просканувати для пошуку кандидатів. Spring дозволяє динамічно змінювати значення полів біна за допомогою додаткових конфігураційних ресурсів.

Для користувача робота з БД перетворюється на модель виклика методу й отримання результату, усі маніпуляції над створенням запиту бере на себе створена бібліотека.

Серед прикладів інтерфейсів найбільш цікавий будівельник – це структурний шаблон, мета якого відокремити логіку створення об’єкту від логіки представлення об’єкта.

Реалізація бібліотеки

Розроблювалася бібліотека для автоматичної генерації найчастіше вживаних запитів. Першим кроком була створена технічна діаграма, яка враховувала такі процеси: ініціалізація бібліотеки, її інтеграція з користувальницьким ПЗ, робочий режим, обробка помилок, можливість автоматизованого тестування.

Наступним кроком була розроблена низькорівнева архітектура застосунку, яка базувалася на шаблонах проєктування, рекомендованих для Spring застосунків.

Були влучно перевикористані можливості Spring Framework для досягання цілей проєкту, а саме для

роботи з завантажуваними класами, інверсією контроля тощо.

Був розроблений простий інтерфейс взаємодії з бібліотекою для користувачів, який дає можливість розробникам одразу використовувати бібліотеку, а не вивчати її внутрішню будову, а також закладена можливість додавання нових інтерфейсів взаємодії для подальшого розширення функціоналу.

Були створені функціональні тести бібліотеки, і автоматизована система зборки та розгортання проєкту.

Тестовий додаток бібліотеки

Для того, щоб провести end-to-end навантажувальне тестування, був розроблений тестовий проєкт, в якому як залежність використовується розроблена бібліотека. За допомогою фреймворку Gatling й додаткової програми на Scala, створюються віртуальні користувачі, які імітують велике одночасне навантаження на систему. Після цього фреймворк будує графіки по основним отриманим результатам – це кількість оброблених запитів, середній час обробки одного запиту, розподіл навантаження по квантилям.

З іншого боку, тестовий додаток слугує інструкцією по інтеграції та використанню бібліотеки, тобто користувачі бібліотеки мають впровадити бібліотеку так само, як в тестовому застосунку, й це гарантовано буде працювати.

Розглянемо розподіли навантаження в синхронному (рис. 3.1) та асинхронному (рис. 3.2) клієнтах.

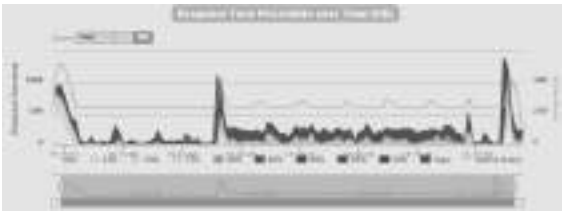


Рис. 3.1. Розподіл навантаження в синхронному клієнті

Важливим показником є те, що синхронний клієнт не виконав 1397 запитів з 36 000. Це означає, що за налаштований максимальний час (1200 мс)

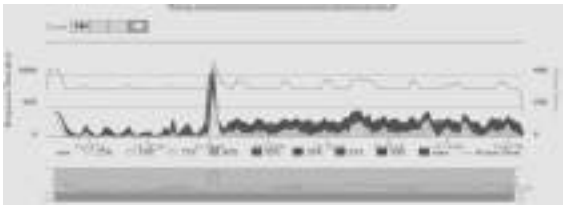


Рис. 3.1. Розподіл навантаження в асинхронному клієнті

віртуальний користувач не зміг отримати ніякої відповіді від системи взагалі. Це дуже показово, що зі збільшенням складності системи, у користувачів синхронного клієнта не залишається механізмів покращення швидкодії – лише горизонтальне масштабування й відповідно більші економічні витрати, або збільшення часу очікування для користувачів, тобто погіршення якості системи.

Для асинхронного клієнту кількість невиконаних запитів є на третину меншою й, що саме головне, природа цих помилок інша, й вони такі, що їх можна вирішити шляхом оптимізації, а не масштабуванням чи збільшенням часу очікування. За результатами тестування було доведено неефективність використання асинхронного підходу для однієї операції та для дуже простих операцій, вищу ефективність та надійність асинхронності при роботі з декількома важкими операціями та необхідність використання правильних підходів при масштабуванні асинхронних додатків.

Висновки

Ця стаття представляє рішення, що задовольняє вимогам ринку з використанням сучасних технологій та орієнтацією на хмарні рішення через зростання їх популярності. Були обрані раціональні технології, які відповідають основним критеріям: актуальності, безкоштовності, підтримуваності, швидкодії, надійності.

За результатами тестування було доведено: вищу ефективність та надійність асинхронності при роботі з декількома важкими операціями; необхідність використання правильних підходів при масштабуванні асинхронних додатків. Було запропоноване рішення для розглянутих проблем: блокування по CPU; блокуючої передачі даних входу/виходу; витоку пам’яті.

Література

[1] Effective Java 3rd edition / Best practices for the Java platform / Joshua Bloch / Addison Wesley Professional 2017.

[2] https://vertex-academy.com/tutorials/ru/java-8-completablefuture/ (дата звернення 01.03.2020).

[3] https://vertex-academy.com/tutorials/ru/java-8-completablefuture-part-2/(дата звернення 01.03.2020).

[4] https://medium.com/swlh/building-dynamodb-brick-by-brick-237e0008b698/Building DynamoDB brick by brick [Електронний ресурс] (дата звернення 10.03.2020).

[5] https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference/ Spring Data JPA [Електронний ресурс] (дата звернення 01.03.2020).

[6] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWork.html (дата звернення 02.03.2020).

[7] https://dzone.com/articles/partitioning-behavior-of-dynamodb(дата звернення 02.03.2020).

[8] https://www.scala-lang.org/The Scala programing language [Електронний ресурс] (дата звернення 05.04.2020).

Проектування архітектури системи дослідження тексту

Bodak Bohdan
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
bohdan.bodak@outlook.com

Doroshenko Anatoliy
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
a-y-doroshenko@ukr.net

***Abstract*. Nowadays, there are many great applications for local search and recommendations, for example Foursquare, Swarm, and Snapchat to some extent. Swarm is a mobile application, which allows users to share their locations with friends and create a record of their experiences in their personal logbook. In addition, a spin-off from and companion app to the older Foursquare, Swarm lets users to check-in to a particular location and interact with people nearby. Besides, these check-ins are listed chronologically in order to create the person's logbook, which in fact represents a digital library. However, many people agree that any social media platform requires a well-designed, scalable, and fast API. In this research, we used RESTful API principles to design a heavily loaded server for the new application. This paper mostly focuses on applying REST and OpenAPI standards using a social media platform as an example. The new specifications and methodologies outlined in this article may be used in the design phase of a heavily loaded backend service**

Keywords: Social media, local search, OpenAPI, REST, RESTful, scalable API, software architecture.

INTRODUCTION

Considering the popularity of various social media applications, our goal was to come up with a successful design and implementation of a backend service. This solution provides users a unified platform for managing, creating, and participating in events. Users will be able to retrieve information about existing events, create custom points of interest, share location and other assets, and invite friends.

The REST [1] used by clients can be thought of as the cornerstone of the World Wide Web. With the rise of cloud platforms and services, REST API's are a perfect choice to allow users to manage and interact with these web services in a platform-independent environment. Moreover, even key companies in the IT industry, such as Google, Amazon, and Microsoft are using it. This kind of API adopts existing HTTP methodologies defined by the RFC 2616 protocol [2]. With REST, all calls should be stateless, which is useful in cloud applications because components can easily scale and accommodate load changes.

RESTful API design was defined by Dr. Roy Fielding [3] and it states that a web service must be in compliance with the next six architectural principles:

- Use of a uniform interface (UI)
- Client-server based
- Stateless operations
- RESTful resource caching
- Layered system
- Code on demand

SYSTEM REQUIREMENTS

The developed social media platform provides a unified database of existing events with detailed information, location, and cost. In addition, it allows a user to create his or her own event in a few steps with automated generation of invitations for friends. The storage of live stories and videos from an event is supported as well.

Throughout the beginning of the system design process, we were able to outline the key features of the platform, which are provided in a list below:

– Token-based authentication. The API has to use JWT or any other type of token to grant permissions to a user.
– Integration with other social media platforms. The API has to support at least two providers: Google and Facebook.
– Retrieval of event's list. The API must return a list of available events to user in form of JSON. Each event contains detailed information, such as location, image, hashtags, status, etc.
– Event creation. The API must have an option for user to create event (private or public), specify details and invite friends. The creator also has an ability to modify the event or delete it.
– Private events. The API provides an interface to create a closed event for a particular group of invited users only.
– Event invitations. The API gives an ability to create and send invitations to user's friends by phone numbers or emails.
– Resources storage. The API provides methods to upload additional information about event, such as images or video into a cloud.

API IMPLEMENTATION

As a backend technology for this API, we decided to select ASP.NET Core, since it is stable and widely supported framework for building REST applications. This suite provides developers with various support lifecycle options to meet the needs of the application. First of all, the database model for our service was created (Fig. 1).

After that, the data layer was generated with a help of Entity Framework, which represented a set of CRUD operations for each entity in the database. Microsoft Azure provides an MSSQL server as a service, which we

used to deploy the database schema. Since this service supports Entity Framework migrations, there is no issue in updating or rolling back changes in a database.

The RESTful API for our social media events platform includes five endpoints documented with

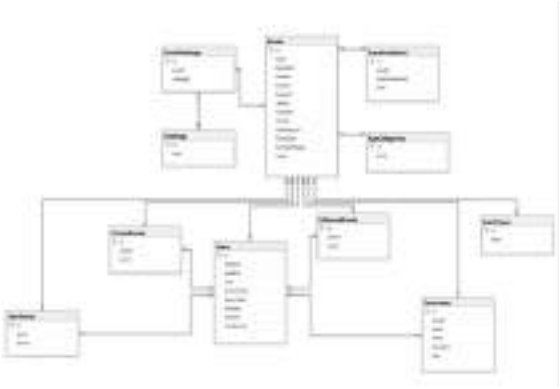


Fig. 1. Database model

OpenAPI v3.0 standard [4]:

- Auth
- Category
- Comments
- Event
- Storage

Each endpoint is also documented according to the standard and provides sample request/response parameters as well as expected status codes with description (Fig. 2).

Fig. 2. Documented API methods

DEPLOYMENT PROCESS

In order to maintain the access to the newest version of the server, it is important to set up an environment and deployment. In our case, it was crucial to deploy the REST API online, so we can test the client application with it. Microsoft Azure is a perfect choice for deploying web applications and services, since it has all pre-build pipelines for building .NET Core backend and publishing it to the virtual machine Azure Pipelines [5] combines continuous integration (CI) and continuous delivery (CD) to constantly and consistently test and build your code and ship it to any target. Continuous integration was used to automate tests and builds in our project. Every time the new code gets pushed to master, Azure automatically runs a build, tests, and saves a result. CD uses these builds to drive automated deployments.

To avoid the problem of breaking the operation of the server, the pipeline features at Azure was used (Fig. 3):

Fig. 3. Azure pipelines

After setting up, every push request at project's Git will trigger a build at Azure, and it will pull from GitHub, and build automatically. This result can be verified at Dashboard of Azure builds. Usually, the code will be published to the staging server for testing purpose, and if the submitted source code was stable and it was decided to publish, it was promoted to the production stage manually.

CONCLUSION

The system described in the article represents a RESTful API for a social media platform. This API was designed according to new methods and standards for a heavily loaded service in order to ensure high performance, scalability, and ease of use. In addition, we demonstrated the continuous integration and delivery process using Microsoft Azure pipelines, which plays a significant role in the testability and overall quality of the product. As a result, developed methodologies may be utilized in future for designing a heavily loaded service.

REFERENCES

[1] What is a RESTful API [Electronic Resource] / Tech Target Network Search App Architecture – Access mode: https://searchapparchitecture.techtarget.com/definition/RESTful-API.

[2] RFC 2616 transfer protocol [Electronic Resource] / W3C – Access Mode: https://www.w3.org/Protocols/rfc2616/rfc2616.html.

[3] Architectural Styles and the Design of Network-based Software Architectures [Electronic Resource] / Roy Thomas Fielding, University of California, Irvine – Access mode: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[4] About Swagger specification [Electronic Resource] / Swagger.io – Acess mode: https://swagger.io/docs/specification/about/.

[5] Build, test, and deploy .NET Core apps [Electronic Resource] / Microsoft – Access mode: https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/dotnet-core?view=azure-devops.

Criterion for signal smoothing and jump prediction using extrapolation in real-time systems

Niechkina Veronika
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
ni2403kalos@gmail.com

Abstract. **The criterion for smoothing random signal changes using a buffer in real-time systems is developed. The problem of the occurrence of rapid spasmodic changes in temperature during the operation of the system for fixing changes in the liquid level in the tank is considered. In the process of solving the problem, a buffering criterion was introduced. Signal changes were written to the buffer.**

Keywords: control system, buffering, delay, extrapolation.

INTRODUCTION

There are systems that analyze signal changes. The purpose of such systems is to analyze the value of the measured signal and control processes and objects.

During the operation of such systems, short-term processes may occur, accompanied by a sudden change in the measured signal. A sharp change in the measured signal is a local phenomenon and does not characterize the measured process. But the system responds to such changes. The system processes the signal value, which is not a characteristic of the current process [1 - 2].

Such sudden changes in the signal are quite short-lived. The signal will return to the previous value. Information about the signal will get into the system. The system will respond to the jump and process the signal data during the jump.

The goal of this work is to eliminate jumps and smooth out the signal. Also, the control system that analyzes the signal changes should stop responding to short-term signal changes.

Buffering will delay data processing. Jumps will be processed in the buffer.

CRITERIA FOR DETERMINING THE TIME OF BUFFERING

The introduction of an additional delay in turn increases the response time of the system to changes in the fluid level. Let the system receive data at intervals 1 second, buffer size – n, basic system response delay – τ_{sys}, maximum allowable delay – $\tau_{\max}$. Buffering time:

$$\tau_{buff} = n \cdot 1c.$$

Then, with the introduction of buffering, the new delay will be:

$$\tau_{\Sigma} = \tau_{sys} + \tau_{buff}.$$

The values $\tau_{\max} - \tau_{sys}$ is small for system, the introduction of a buffer the size of which is sufficient for the detection and processing of jumps will cause the delay to be exceeded $\tau_{\max}$, which is inadmissible. Since the jumps are different from the processes served only in duration, the programmatic jump recognition is not possible without the introduction of buffering [3 – 4].

An additional problem: minimize the delay that is introduced to the system when solving the task.

When buffering is introduced, a delay occurs. It can be minimized if buffering is not continuous. Provided that the signal has changed. If at the end of the accumulation of signal values in the buffer to process them not one by one, but all together in one iteration of the processing cycle, then this approach allows to reduce the total system delay. If the size of the buffer – n, basic system response delay – τ_{sys}, maximum allowable delay – $\tau_{\max}$, then:

$$\tau_{\Sigma} = \max(\tau_{buff}, \tau_{sys}) < \tau_{sys} + \tau_{buff}.$$

If possible, pick one n, at which $\tau_{buff} = n \cdot 1c \le \tau_{sys}$, then:

$$\tau_{\Sigma} = \max(\tau_{buff}, \tau_{sys}) = \tau_{sys}.$$

There will be no further delay in applying the proposed method.

It is necessary to formulate a criterion that will determine the possibility of a jump. If for the processes occurring during the operation of the system, it is possible to choose such a method of extrapolation of the next value of the signal that it will give the maximum deviation from the real value obtained at the places of change of the monotone of the signal, then the difference between the real value of the signal and the extrapolated value at each time can use as a buffer criterion [5].

EXPERIMENTAL RESULTS

Consider the process of fixing the change in the liquid level in the tank [6]. The locking system works in real time and has a sensor immersed in some liquid. If the liquid level decreases, the sensor enters the air, its temperature increases due to changes in heat transfer. One of the algorithms of the system responds to this temperature rise and informs the operator about the liquid level decrease. When the liquid level rises again and covers the sensor, its temperature decreases and the alarm goes off. In some cases, rapid abrupt changes in temperature may occur, which can be recognized by the algorithm as a

decrease/return of the fluid level and lead to an erroneous alarm.

The general view of the signal generated by the system is shown in Fig. 1. If you divide the graph into separate sections, the signal is first constant, then monotonically increasing, then again has a constant plot, then monotonically decreases and becomes constant again, you can approximate the value of the signal by the least squares method, and take the next point on that line, then the value will deviate as much as possible from the actual value obtained at the places of change of monotony.

In the course of the work numerous checks were carried out on real signals. The graph of the signal and the extrapolated values is shown in Fig. 2. It shows that the maximum deviation of the extrapolated value deviates from the real signal at the inflection points.

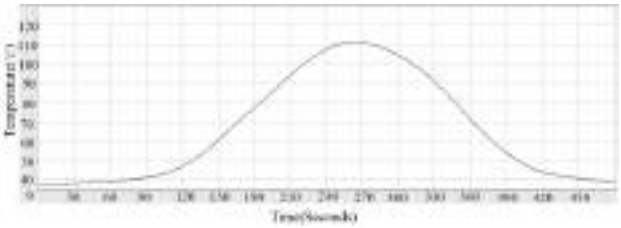


Fig. 1. Graph of the temperature of the liquid level sensor

Fig. 2 present numerous signal values, extrapolated values and deviations of extrapolation from the real signal value. The data confirm that extrapolation can be used as a buffer criterion. Also, by analyzing the deviation, it can be seen that the value at the places of sharp change of the signal is much higher than the deviation at the places of the smooth change of the signal due to the decrease of the liquid level. Therefore, it is possible to set the maximum tolerance value to avoid buffering when the signal changes smoothly.

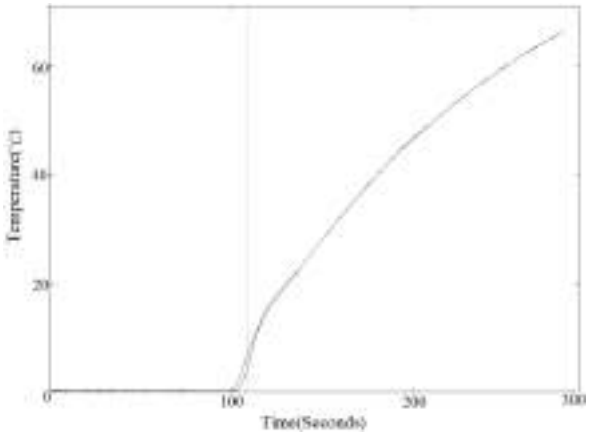


Fig. 2. Graph of the real and extrapolated values of the signal as it grows

The deviation at the monotone sections of the module does not exceed 1. Therefore, for the presented example of [7-8] the input signal, you can use the selected buffer criterion.

The following Fig. 3 illustrate the operation of a system of recording fluid level changes in a tank with a time buffer whose duration is determined by the proposed criterion. The actual color value of the signal is displayed in red, and the values transmitted to the handler are shown in blue.

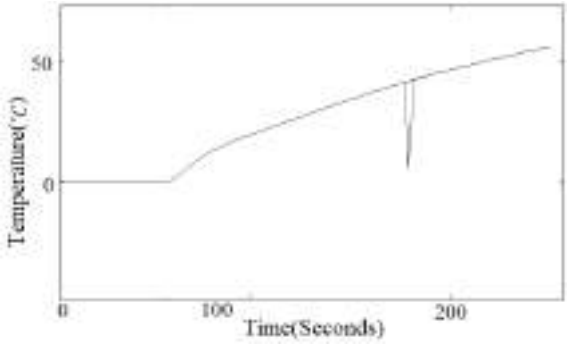


Fig. 3. Smoothing the jump as the signal grows

CONCLUSIONS

Processes that occur during the test of control systems are accompanied by a sudden change in the measurement signal. Such phenomena force the system to react, which is harmful. It is possible to counteract the impact of the jumps by introducing additional sensors or adding buffering to the communication channel between the sensor and the control system. The introduction of buffering will delay the processing of data. Jumping will be processed in the buffer.

The problem of the influence of stubby changes in the measuring signal was solved using the buffer criterion. Using the difference between the actual value of the signal and the extrapolated value at each time point, we recorded the change in the derivative signal during buffering. Processing the buffer after detecting a sudden change in the measured signal is performed by combining the values before and after buffering with one straight line. It was this that made it possible to accurately record the temperature without rapid jumps when changing the liquid level in the tank. The results obtained can be used in measurement and control systems for various objects and processes.

REFERENCES

[1] Marcelo Godoy Simes; Felix A. Farret, “Designing power electronic control systems” in Modeling Power Electronics and Interfacing Energy Conversion Systems, IEEE, 2017, pp.83-116

[2] N.B. Repnikova, “Automatic Control Theory: Classics and Modernity”. K .: NTUU “KPI”, 2011.

[3] Lizhe Tan, Jean Jiang, in Digital Signal Processing (Third Edition), 2019, pp.59-89, pp.173-228

[4] Asif Sabanovic; Kouhei Ohnishi, “Control System Design” in Motion Control Systems, IEEE, 2011, pp.29-59

[5] A. Gaddam, S. C. Mukhopadhyay and G. S. Gupta, “Intelligent bed sensor system: Design, expermentation and results” 2010 IEEE Sensors Applications Symposium (SAS), Limerick, 2010, pp. 220-225.

[6] J. Rivera-Mejía, E. Arzabala-Contreras and Á. G. León-Rubio, “Approach to the validation function of intelligent sensors based on error's predictors” 2010 IEEE Instrumentation & Measurement Technology Conference Proceedings, Austin, TX, 2010, pp. 1121-1125.

[7] V. I. Sivetskyy, O. M. Khalimovskyy, O. L. Sokolskyy and I. I. Ivitskyy, “Automation of intelligent sensor injection inlet in polymer moldings by using vector controlled electric drive” 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), Kiev, 2017, pp. 534-537.

[8] Petr I. Kravets, Tatyana I. Lukina, Valeriy A. Zherebko, Vladimir N. Shimkovich, “Methods of Hardware and Software Realization of Adaptive Neural Network PID Controller on FPGA-Chip” Journal of Automation and Information Sciences. New York, USA, 2011, Vol. 43, Issue 4, pp. 70-77.

Виправлення помилок системних ресурсів при віддаленій інсталяції

Теленик Андрій
КПІ ім. Ігоря Сікорського
Київ, Україна
andrzej.telenik@gmail.com

Анотація. **Мета статті - переглянути наявні способи вирішення проблеми видалення помилок ресурсів віддаленої інсталяції програмного забезпечення, уточнити аспекти їх розгортання та описати бачення автора для вирішення проблеми. Найкращим рішенням проблеми буде впровадження засобів виправлення помилок у системі встановлення віддаленого програмного забезпечення проектора на основі алгоритму Дейкстри, зміненого відповідно до завдання, що дозволить децентралізовано встановити програмне забезпечення в комп'ютерній мережі.**

Ключові слова: системні ресурси; помилка інсталяції; сумісність версій; інсталяційні пакети; алгоритм Дейкстри.

У минулій статті [1] автором була визначена проблема синхронізації та централізації управління ПЗ у комп'ютерних мережах, що полягає у необхідності забезпечення вчасного встановлення уніфікованого програмного забезпечення на комп'ютери підприємства, проаналізовані наявні методи її вирішення, описані їх недоліки та переваги. З урахуванням цієї інформації автором були сформовані вимоги до оптимальної системи автоматичної інсталяції програмного забезпечення: така система має мати клієнт-серверну структуру зі зв'язком між елементами системи, незалежним від топології (працювати як в мережах з P2P-зв'язком, так і в мережах, побудованих за принципом «зірки») та підтримувати «тиху інсталяцію» без конфлікту з основними потоками роботи операційної системи абоненту мережі.

Використання такої системи вимагає наявності у розпорядженні компаній таких важливих ресурсів як операційна пам'ять та пам'ять для запису щоб забезпечити функціонування встановленого корпоративного програмного забезпечення (ПЗ). На відміну від централізації інсталяції ПЗ, централізація менеджменту пам'яті на комп'ютерах корпоративної мережі практично неможлива (і непотрібна) через розподілення функцій комп'ютерів та різницю у факторах їх використання, як-то їх потужність, час роботи, графік оператора та ін. Системні вимоги до програми, що встановлюється за допомогою автоматичної системи віддаленої інсталяції (або цілого пакету ПЗ) можуть виявитися поза можливостей наявної машини з різних причин, як-то фізичне або моральне застаріння обладнання або завеликий розмір пакету інсталяції.

Через це виникає проблема пристосування роботи системи автоматичної дистанційної інсталяції до особливостей конкретної машини, з якою працює система, а саме до наявних ресурсів пам'яті та потужності комп'ютера. Ігнорувати її не можна, тому що нестача ресурсів може призвести до неможливості виконання інсталяції та зриву виробничого завдання. До того ж централізоване виправлення цих проблем є неможливим через високий степінь автоматичності процесу.

Огляд існуючих рішень проблеми менеджменту ресурсів показав, що ні одна з наявних програм автоматичної віддаленої інсталяції – ні Total Software Deployment, ні Maestro AutoInstaller – не мають вбудованого арсеналу інструментів для виправлення можливих помилок ресурсів. Більш того, вони не мають вбудованих сигналізаторів помилки, усі повідомлення про нестачу ресурсів виводяться у вигляді малозрозумілого для користувача системного коду стандартними повідомленнями Windows про помилку.

На погляд автора, спосіб вирішення цієї проблеми криється у самій ідеї інсталяційних пакетів. Вона полягає у тому, що програмне забезпечення, яким диспонує сервер, не встановлюється безпосередньо на комп'ютери-абоненти – з програм формуються інсталяційні пакети ПЗ, які потім запаковуються та встановлюються на комп'ютері клієнта. Завдяки цьому можна вирішити проблему можливої нестачі ресурсів, змінюючи склад та компоновку інсталяційних пакетів. Підкріплюється ця можливість також різністю версій програм, що зберігаються на сервері – різні версії однієї і тієї ж програми можуть мати різні програмні вимоги, як правило, більш ранні версії є менш вимогливими до ресурсів.

Отже, для вирішення проблеми нестачі ресурсів потрібно:

1) автоматично сповістити адміністратора мережі, який проводить інсталяцію відповідно до об'єму наявних на машині ресурсів;
2) Прийняти рішення про повторну інсталяцію, корекцію інсталяційних параметрів або припинення інсталяції.

Максимізувати ефективність вирішення другого пункту можна шляхом забезпечення адміністратора точною інформацією про стан ресурсів на клієнтській

машині та використанням алгоритмів розрахунку оптимізації менеджменту наявних ресурсів. У якості такого алгоритму можна ефективно використати алгоритми пошуку найкоротшого шляху у графах, якщо представити кожну версію певної програми як дискретну вершину графа, а різницю у системних вимогах виразити як вагу ребер між цими вершинами. У такому разі дуже добре підходить модифікований під цю задачу алгоритм Дейкстри.

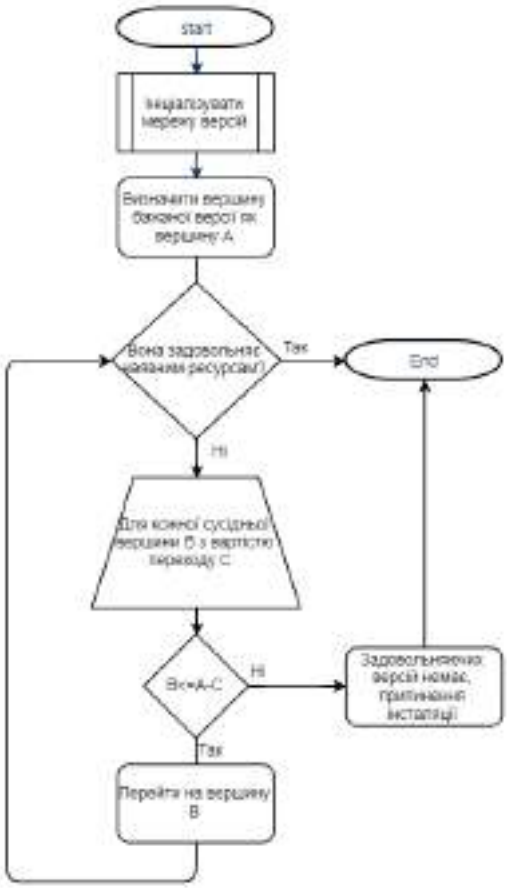


Рис. 1. Блок-схема запропонованого алгоритму

Цей алгоритм має проводити пошук маючих найменшу вагу ребер та, переходячи у результаті до версій, що мінімально поступаються бажаній за системними вимогами (це робиться для того, щоб мінімізувати можливі негативні наслідки від розбіжності у версіях з машинами, інсталяція програми на котрі вдалася) порівнювати вимоги, що виносяться до них з вимогами бажаної версії. Якщо обрана версія «вкладається» у задані рамки, то робота алгоритму припиняється та версія запаковується для інсталяції, якщо ні – пошук продовжується вже з нової «вершини» графу версій програми.

Для графу з вершинами версій V, ребрами e з вагами e.len:

```
for v∈V
        d[v] = ∞∞
        used[v] = false
        d[s] = 0
for i∈V
        v = null
        for j∈V
                if !used[j] and (v == null or d[j] < d[v])
                  v = j
        if d[v] == ∞∞
                break
        used[v] = true
        for e: ребра, що виходять з v
                if d[v] + e.len < d[e.to]
                  d[e.to] = d[v] + e.len
```

Також запропонований алгоритм має бути забезпечений додатковими засобами автоматизації менеджменту ресурсів для збільшення його ефективності. Програмні засоби реалізації алгоритму мають забезпечувати автоматичне вимкнення системного фаєрволу для ліквідації ризику блокування інсталяції плюс автоматичні пошук та видалення файлів невдалої інсталяції та/або минулих версій програми з клієнтської машини для звільнення місця від непотрібної інформації.

Таким чином, з огляду на відсутність реалізації вирішення проблеми менеджменту ресурсів у наявних програмах автоматичної віддаленої інсталяції, найкращим варіантом буде імплементація у проектовану систему віддаленої інсталяції засобів автоматичної корекції інсталяційного пакету та інформування адміністратора, який виконує встановлення про наявні на машині клієнта ресурси та системні вимоги до софту.

Висновки

У результаті проведених досліджень була визначена проблема менеджменту системних вимог та виправлення помилок у системі автоматичної віддаленої інсталяції програмного забезпечення. Проведений аналіз існуючих рішень, запропонований власний шлях вирішення даної проблеми та побудований його алгоритм. З урахуванням цього автором були намічені додаткові вимоги для розробки автоматичної системи дистанційної інсталяції.

References

[1] Теленик А., Автоматизована система віддаленої інсталяції програмного забезпечення. В матеріалах VIII міжнародної науково-практичної конференції з інформаційних систем та технологій Winter InfoCom Advanced Solutions 2019, 2-3 грудня 2019 року, Україна, Київ, стор.23-24.

[2] Dijkstra E. W. A note on two problems in connexion with graphs // Numer. Math — Springer Science+Business Media, 1959. — Vol. 1, Iss. 1. — P. 269–271. — ISSN 0029-599X; 0945-3245 — doi:10.1007/BF01386390

[3] http://maestro-kit.ucoz.ru/ (дата звернення 01.11.2019).

[4] http://computerologia.ru/obzor-programmy-total-software-deployment/ (дата звернення 01.11.2019).

[5] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm(дата звернення 31.12.2019).

Рецензент: д.т.н., проф. каф. АУТС, КПІ ім. Ігоря Сікорського Сергій Теленик

ОБРОБЛЕННЯ ІНФОРМАЦІЇ У СКЛАДНИХ СИСТЕМАХ

INFORMATION PROCESSING IN COMPLEX SYSTEMS

Methods of Optimizing the Diagnostic Data Processing for Information Management Systems

Olena Syrotkina
Dnipro University of Technology
Dnipro, Ukraine

Mykhailo Aleksieiev
Dnipro University of Technology
Dnipro, Ukraine

Iryna Udovyk
Dnipro University of Technology
Dnipro, Ukraine

Abstract. **This paper addresses the issue of creating and applying mathematical methods for optimizing time and computing resources when processing multiple data streams circulating in a distributed information management system. In order to solve this problem, we suggest methods of reducing the space of analyzed states using the data organization structure "m-tuples based on ordered sets of arbitrary cardinality". Using this data structure allows us to minimize the time and computing resources involved.**

Keywords: Big Data, Big Data reduction methods, data organization structure, ordered set of arbitrary cardinality, minimization of time and computing resources.

INTRODUCTION

At the present stage in the creation and application of information technologies in various industries and energy, the problems of diagnosing the operability of complex hardware/software systems of an industrial enterprise remain relevant [1, 2]. In the event of a system failure or an abnormal situation being detected, large volumes of poorly structured low-level primary diagnostic information are generated within seconds, which requires further processing and analysis to restore the system [3, 4]. The shortage of time, computing and information resources in the processing of data, methods used for data analysis and the need for prompt decision-making in real time affect the quality of diagnostics and the recovery time of the system [5].

Therefore, the most significant problem is the creation and application of mathematical methods for optimizing time and computing resources when processing multiple streams of diagnostic data circulating in the information management systems of an industrial enterprise.

ANALYSIS OF APPROACHES TO OPTIMIZE DATA PROCESSING

We propose mathematical methods for working with the data structure "*m*-tuples based on ordered sets of arbitrary cardinality (OSAC)" to solve the problems of analysis and processing of various combinations of several basic parameter sets for information management systems [6, 7].

This data structure allows us to describe a Boolean template in a general way. This template is an ordered set of all subsets of an ordered basis set of arbitrary cardinality for any data type. In this case, there is a triple ordering of data:

a) Ordering by basis set.

Basis set ordering allows an arbitrary algorithm to be created for ordering the data of the basis set for a particular data type and for a specific task of working with data when instantiating a template class based on the given data structure.

b) Ordering by a Boolean of the basis set.

This approach divides a Boolean, elements of which are tuples of element combinations of the basis set into subsets by the criterion for the length of a tuple. This allows us to order the subsets of a Boolean according to the length of the tuple.

c) Ordering by a Boolean subset.

This approach implies that we have a Boolean subset with tuples of the same length. The tuple ordering is consistent with the ordering of the basis set elements.

Triple ordering of data allows us to determine the characteristic properties of the data structure elements and the functional relationships between *m*-tuples based on the analysis of their location in the data structure.

The application of mathematical methods for working with the given data structure based on the properties obtained and the functional dependencies thus derived is one of the ways to solve the problem.

MAIN TERMS AND DEFINITIONS OF THE GIVEN DATA STRUCTURE

The list of designations of the main elements of the data structure "*m*-tuples based on OSAC" is shown in Table 1.

A more detailed description of the basic terms and definitions, properties and patterns, as well as mathematical methods for working with the data structure "*m*-tuples based on OSAC", is given in [6, 7].

TABLE 1

THE LIST OF BASIC ELEMENT DESIGNATIONS FOR THE GIVEN DATA STRUCTURE "M-TUPLES BASED ON OSAC"

Designation	Definition
X	Ordered basis template set
n	Cardinality of set X
x_i	i^{th} element of set X
i	Element index of set X
2^X	Boolean of set X
Y_m^n	Non-empty subset of Boolean 2^X, elements of which are tuples of the same length m consisting of elements of set X with cardinality n
k_m^n	Cardinality of subset Y_m^n
$y_{m,j}^n$	m-tuple which is the j^{th} element of subset Y_m^n

Designation	Definition
m	Length of tuple $y^n_{m,j}$, i.e. the number of elements of basis set X in the tuple
j	Index of tuple $y^n_{m,j}$ in subset Y^n_m
$K^n = \{k^n_m \mid k^n_m = \binom{n}{m} = \frac{n!}{m!(n-m)!}, 1 \le m \le n\}$	Set of cardinalities k^n_m of subsets Y^n_m for Boolean 2^X

MATERIALS AND METHODS

In the data structure being considered, each *m*-tuple is unique and its location is determined univocally. The location of the *m*-tuple in the two-dimensional structure is defined by a pair of indexes (*j*, *m*).

As a result of research of the properties and functional dependencies between the data structure elements "*m*-tuples based on OSAC", certain properties of this data structure, partially described below, were determined and theoretically proven.

A) Certain properties of the ordered set of cardinalities K^n of subsets Y^n_m for Boolean 2^X

Property A.1. The value of the first element of set K^n is always equal to the value of cardinality *n* of basis set *X*.

Property A.2. The value of the last element of set K^n is always equals to 1.

Property A.3. The value of the m^{th} elements corresponding to the first and last elements of set K^n are always equal.

B) Certain properties of maximum elements of an ordered set with cardinalities K^n

Property B.1. If basis set *X* contains an odd number of elements *n*, then the set of cardinalities K^n of subsets Y^n_m for Boolean 2^X has two elements with a maximum value.

Property B.2. If basis set *X* contains an even number of elements *n*, then the set of cardinalities K^n of subsets Y^n_m for Boolean 2^X has one element with the maximum value:

Set $K^n_{\max}$ of maximal elements of set K^n in general is defined as follows:

$$K^n_{\max} := (n\%2)\,?(\{k^n_{\frac{n-1}{2}}, k^n_{\frac{n+1}{2}}\}) :)(\{k^n_{\frac{n}{2}}\}),$$

where % is the operation of obtaining the remainder of the division.

The graphs for evaluating the execution time of some methods for working with the given data structure "*m*-tuples based on OSAC" are given in [6]

CONCLUSIONS

The methods applied for working with the data structure "*m*-tuples based on OSAC" allow optimizing the data processing, since:

- in this data structure only the basis set *X* is initialized, other elements of the structure are generated as required on the basis of a certain set of formal rules for obtaining data structure elements;
- the size of the memory used to store the basis set in comparison with the entire data structure is reduced by the amount $(2^n - n) \cdot sizeof(T)$, where *T* is a type of element of the basis set;
- for a certain class of operations on data structure elements containing a large array with a complex data type structure as a basis set *X*, the mathematical methods used can show *m*-tuples with a complex data type as sets of integers that are indexes of their location in the structure. This allows time and computing resources to be minimized for data processing;
- for a certain class of operations on data structure elements, the determination of functional dependencies between these elements by their position in the structure is defined by a pair of indexes (*j*, *m*). It allows us to calculate the result based on these functional dependencies without performing a cumbersome data processing algorithm for a large array and complex data type.

REFERENCES

[1] K.S. Manoj, "Industrial Automation with SCADA: Concepts, Communications and Security", Chennai: Notion Press, 2019, 242 p.

[2] Technical Manual, "Supervisory Control and Data Acquisition (SCADA) Systems for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities", Washington: Department of the Army, 2006, 94 p.

[3] S. Windmann and O. Niggemann, "Efficient Fault Detection for Industrial Automation Processes with Observable Process Variables", proceedings IEEE International Conference on Industrial Informatics (INDIN) Cambridge: 2015. pp. 121-126.

[4] J. MacGregor and A. Cinar, "Monitoring, Fault Diagnosis, Fault-Tolerant Control and Optimization: Data Driven Methods", Computers & Chemical Engineering, 2012, vol. 47, pp. 111–120.

[5] M. Chen, S. Mao, Y. Zhang and V. Leung, "Big Data. Related Technologies, Challenges, and Future Prospects", Spinger, 2014, 100 p.

[6] O. Syrotkina, M. Alekseyev and O. Aleksieiev "Evaluation to determine the efficiency for the diagnosis search formation method of failures in automated systems", Eastern-European Journal of Enterprise Technologies, 2017, vol. 4, issue 9 (88), pp. 59–68.

[7] O. Syrotkina, M. Alekseyev, V. Asotskyi and I. Udovyk, "Analysis of how the properties of structured data can influence the way these data are processed", Naukovyi Visnyk NHU, Dnipro, 2019, vol. 3 (171), pp. 119-129

ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

PROGRAMMING TECHNOLOGIES

Методика ефективної реалізації односторінкових веб-додатків

Драбинко Віталій
КПІ ім. Ігоря Сікорського
Київ, Україна
v.drabynko@gmail.com

Дорошенко Анатолій
КПІ ім. Ігоря Сікорського
Київ, Україна
a-y-doroshenko@ukr.net

Анотація. **В роботі демонструється результат дослідження ефективності фреймворків та бібліотек JavaScript при реалізації односторінкових веб-додатків. Були розглянуті такі інструменти як React, Vue.js та Angular . Було розроблено веб-додатки з використанням даних інструментів і проведено оцінку їх ефективності.**

Ключові слова: односторінковий веб-додаток, Angular, React, Vue.js, фреймворк.

Вступ

Хмарові веб-додатки набирають популярності. Багато IT-компаній помічають цю тенденцію, і все більше програмних продуктів розвивається на основі віддаленого доступу. Є багато аналогів настільних програм, які пропонують онлайн-версію за невелику щомісячну плату. Вони надають більшої гнучкості та мобільності. Наприклад, ви можете легко вводити дані в CRM або ERP-системи в хмарі через свій мобільний телефон, і це може відбуватися в любому вигідному для вас місці. Сьогодні веб-сайти все частіше розглядаються як користувальницький додаток, а не як статичні сторінки, які були у нас близько 10 років тому. З одного боку, причинами цього є спроби користувачів отримувати і створювати інформацію на основі їх особистих характеристик і вимог. З іншого боку, власники сайтів хочуть надати користувачам більш зручні інтерфейси для роботи з інформацією.

Односторінкові веб-додатки

SPA або Single Page Application – це односторінковий веб-додаток, який завантажується на одну сторінку HTML. Завдяки динамічному оновленню за допомогою JavaScript, немає необхідності перезавантажувати додаткові сторінки під час використання. На практиці це означає, що користувач бачить увесь основний контент у браузері, а при переході на інші сторінки необхідні елементи просто завантажуються замість всього перезавантаження [1].

Односторінкові веб-додатки дозволяють імітувати роботу настільних додатків. Архітектура призначена для оновлення лише частини вмісту при переході на нову сторінку. Таким чином, вам не доведеться перезавантажувати одні і ті ж елементи. Це дуже зручно для розробників та користувачів. Одна з найпопулярніших мов програмування – JavaScript – використовується для розробки SPA. Односторінкові сайти запускаються в браузері і не потребують перезавантаження сторінок або додаткових сторінок під час використання. Такі додатки щодня використовують мільйони користувачів, навіть не помічаючи цього. Найпопулярніші приклади - GitHub, Gmail, Google Maps і навіть Facebook.

Крім того, робота з платформою односторінкових додатків вимагає набагато більше досвіду в галузі архітектури та безпеки. Вони характеризуються більш високою частотою оновлень та появою нових платформ порівняно з традиційними веб-додатками. Крім того, робота з односторінковими додатками може ускладнити налаштування автоматизованих процесів збирання та розгортання та використання параметрів розгортання, таких як контейнери, порівняно з традиційними веб-додатками.

У процесі роботи користувачу може здатися, що він використовує не веб-сайт, а настільний додаток, оскільки він негайно реагує на всі свої дії, не затримуючись або «зависаючи». SPA і PWA – веб-сайти, які поступово витісняють класичні MPA. Це пов’язано з тим, що вони легше розвиваються, працюють швидше і подобаються користувачам. Однак у них є слабкий момент – SEO оптимізація. Не всі браузери все ще можуть нормально працювати з ними, тому для того, щоб такі програми були SEO-дружніми, потрібно вдатися до низки хитрощів. MPA-сайти в цьому плані простіші та надійніші. Для створення складних проектів рекомендується використовувати фреймворки, такі як React, Angular або Vue. Їх архітектура дозволяє створювати гнучкі веб-додатки, а також вони мають великий набір готових рішень, що значно спрощує процес розробки. Крім того, ви можете створювати повні мобільні додатки на основі цих структур.

Фреймворк Angular

AngularJS – це найсучасніша система Google, яка зазвичай використовується для роботи з односторінковими програмами (SPA). AngularJS є відкритим кодом з ліцензією MIT, і ви можете слідкувати за процесом розробки AngularJS на GitHub.

Важлива відмінність між Angular і іншими фреймами полягає в тому, що він використовує TypeScript.TypeScript – строго типізована мова програмування, яка транспілюється на JavaScript. Тобто, надає для JavaScript розробників нові можливості.

Транспіляція – це процес перетворення коду однієї мови програмування в код іншої (або тієї ж мови, але різних версій). Наприклад, перетворення сучасного JavaScript (EcmaScript 2015 - 2018) в більш старий, що

підтримується більшістю браузерів EcmaScript 5. Це дуже корисна можливість даного фреймворка, яка надає багато можливостей.

Подібно перекладу ES2015-2018 у EcmaScript 5, TypeScript також перекладається в JavaScript будь-якої версії, починаючи з EcmaScript 3. Таким чином, підтримка також надається в старих браузерах [2-5].

Фреймворк React

React – це бібліотека, яка була створена для надання можливості розробникам створювати інтерфейс користувача. Вона допомагає вирішувати проблеми не повного, а часткового оновлення сторінки. Це дуже важливо при розробці односторінкових веб-додатків. Дана бібліотека розробляється Facebook, Instagram та просто спільнотою, адже вона відкрита.

React надає можливість програмістам створювати не лише великі веб-додатки, а й односторінкові, які використовують дані, що можуть змінюватися у часі, при цьому, перезавантаження сторінки не потрібне. Ціль React – бути простим та швидким і він з цим справляється.

Так як React відповідає видові у моделі MVC (model-view-controller), то він відповідає лише за обробку інтрефейсу користувача у додатках. Ця його здатність може бути використана з іншими фреймворками або бібліотеками на базі JavaScript.

Завантажуючи сторінку, скрипт приймає під контроль елемент DOM, який ви передали на нього. Тепер у цьому елементі React панує верховно. Він може стежити за діями користувача та змінювати дерево елементів на будь-якому рівні. Для цього він використовує швидкий віртуальний DOM. Цей компонент може подати будь-який компонент (або дерево компонентів), залежно від поточного стану програми. Деякі дрібниці або навіть весь вміст можуть змінитися [6-7].

У React не відома модель MVC, оскільки компонент включає в себе і перегляд (візуалізацію), і логіку. Це означає, що в компоненті, такому як селектор, ви вказуєте як його зовнішній вигляд, так і поведінку. Для зручності використовується спеціальний синтаксис JSX (JavaScript XML). Це схоже на HTML, але насправді це чистий JavaScript.

Фреймворк Vue.js

Vue – новий фреймворк для створення інтерфейсів користувача. На відміну від інших фреймворків, Vue розроблений з нуля, щоб поступово інтегруватися в середу пронрамування. Основна бібліотека орієнтована лише на шар перегляду, і її легко підбирати та інтегрувати з іншими бібліотеками або існуючими проектами. З іншого боку, Vue також чудово здатний створювати складні односторінкові веб-додатки, коли використовується в поєднанні з сучасними інструментами та бібліотеками підтримки [8].

Одна з головних переваг Vue – реактивність. Так як моделі – JavaScript об’єкти, то керувати станами дуже просто. Кожен компонент автоматично слідкує за собою і розуміє коли необхідно ре-рендеритися.

Оцінка ефективності методів

Для порівняння ефективності даних підходів до реалізації односторінкових веб-додатків було створено додаток для планування – список задач. Його можна побачити на Рис. 1.

Всі підходи показали гарні результати в порівнянні з написанням додатка з використанням лише JavaScript. При підрахунках ефективності було взято кількість рядків коду на JS за 1.

Маємо такі показники ефективності:

1. Vue.js – 0,321.
2. Angular – 0,356.
3. React – 0,413.

Тобто, при використанні фремворків, ми зменшуємо кількість написаного коду в 3 рази. Це значно економить час та затрати на створення односторінкового додатка.

Рис. 1. Веб-додаток – список задач

Висновки

У списку бібліографічних посилань наводяться тільки джерела, на які автор посилається у тексті.

В результаті виконання даної роботи були розглянуті методи створення односторінкових веб-додатків і виявлено їх ефективність.

При створенні веб-додатку для списка задач найефективнішим виявився фреймворк Vue.js, трішки менш ефективним Angular та найменш ефективним React. Але у кожного з них є свої плюси і вони ефективніші, ніж простий JS.

Література

[1] Майкл Миковски, Джош Пауэлл. Разработка одностраничных веб-приложений = Single Page Web Applications: JavaScript End-to-end. — ДМК Пресс, 2014. — 512 с.

[2] Фримен А. Angular для профессионалов. — СПб.: «Питер», 2018. — С. 800.

[3] Дилеман П. Изучаем Angular 2. — СПб.: «ДМК Пресс», 2017. — С. 354.

[4] Brad Green, Shyam Seshadri. AngularJS. — O'Reilly Media, 2013. — 196 p.

[5] Холмс С. Стек MEAN. Mongo, Express, Angular, Node. — СПб.: «Питер», 2017. — С. 496.

[6] Мардан Азат. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. — СПб.: «Питер», 2019. — С. 560.

[7] Бэнкс Алекс, Порселло Ева. React и Redux: функциональная веб-разработка. — СПб.: «Питер», 2018. — С. 336.

[8] Callum Macrae. Vue.js: Up and Running. — O’Reilly, 2017. — 219 с.

Централізоване керування музичними сервісами

Богомол Роман
КПІ ім. Ігоря Сікорського
Київ, Україна

***Анотація*. Мета статті – переглянути наявні способи вирішення завдання, виділити їхні переваги і недоліки і змоделю-вати архітектурне рішення, яке об'єднує всі переваги, а також не буде містити недоліків оглянутих аналогів. Найкращим рішенням для реалізації даного завдання являється програмний продукт створнеий на основі клієнт – серверної архітектури серверна частина якої реалізована за допомогою мікросервісного підходу.**

Ключові слова: музичні сервіси, клієнт – серверна архітектура, мікросервіси, управління музичним контентом.

Прослуховування аудіо-композицій з кожним роком стає все важливішою частиною сучасного життя, а способи доступу до аудіо контенту з кожним роком стають простішими. В межах еволюції передачі аудіо-даних можна виділити декілька стадій розвитку: етап превалювання фізичних носіїв для запису файлів електронного формату на кінцевій машині користувача (компакт-дисків) та етап розповсюдження онлайн-доступу.

Недоліки дисків досить численні та полягають у необхідності зберігання даних, нестійкості до пошкоджень, незручності використання, а також можливості втрати даних. До того ж, протягом певного часу тенденцією було зберігання композицій на машині користувача, що було обумовлено лімітованими ресурсами інтернету. В результаті це призвело до появи низки недоліків, які були пов'язані із неможливістю вчасного отримання новинок музики в момент їх виходу та появи на ринку і пошуку існуючої композиції. Дані недоліки породили сервіси з прослуховування музичних композицій онлайн. Даний підхід є зручним для користувача, але в ході розвитку ресурсів такого типу та збільшенням їх кількості виникла проблема забезпечення централізованого доступу до сервісів.

Для визначення шляхів вирішення цих недоліків розглянемо існуючі сервіси на прикладі Spotify, Apple Music, YouTube Music. Музика може бути переглянута або відсортована (знайдена) завдяки можливості пошуку з використанням різних параметрів, таких як виконавець, альбом, жанр, список відтворення. Для користувача надається можливість створення і редагування списків програвання. Проте, наявність платних підписок, необхідність переключатись між різними сервісами породжує завдання централізованого управління в межах міжсервісної взаємодії та агрегації аудіо контенту користувача. Для побудови програмного рішення необхідно проаналізувати існуючі сервіси, а саме формат взаємодії з ними. В якості джерела контенту для подальшого централізованого управління та агрегації. Аналізуючи документацію роботи API (Application Programming Interface) [1] вищезгаданих сервісів можна зробити висновок, що дані сервіси мають схожий тип авторизації за допомогою відкритого протоколу OAuth 2.0 [2], який дозволяє третій стороні надавати права обмеженого доступу до ресурсів користувача без необхідної передачі логіна і пароля.

Для агрегації і централізованого управління музичних сервісів було виділено ряд загальних ознак і функцій, які притаманні кожному із вищезгаданих сервісів таких як: YouTube Music, Deezer Spotify, що відіграють основну роль по взаємодії з музичним контентом. Під час огляду існуючих рішень було знайдено наступні додатки: Soundiiz і Musconv. Додаток Soundiiz, реалізований за допомогою веб-орієнтованої архітектури, об'єднує близько тридцяти музичних сервісів і реалізовує алгоритми передачі музичних даних між потоковими платформами. До основних можливостей даного додатку можна віднести синхронізацію списків відтворення за допомогою якої можна централізовано керувати створенням, видаленням, редагуванням музичного контенту всіх сервісів. Також виконана можливість імпорту та експорту даних за допомогою файлів з розширенням M3U, XSPF, TXT, CSV, URL посилання та ін. MusConv – платформа яка об'єднує в собі більш ніж 30 музичних сервісів, має багато спільних можливостей із Soundiiz, а саме синхронізацію музичних списків, експорт і імпорт даних. Додаток має і суттєву перевагу над Soundiiz, тому що має своє графічне вікно користувача (десктоп) і є доступним для більшості сучасних пристроїв.

Серед переваг даних рішень можна виділити зручність створення нових списків відтворення, адже додається можливість шаблонного завантаження файлів е вказаному форматі і їх програмна обробка (парсинг). Ці файли створюються за допомогою популярних музичних плеєрів, що дозволяє користувачеві відмовитись від ручного вводу інформації, а також надає можливість парсингу даних за допомогою URL-посилання. Серед недоліків можна виділити платну підписку для кожного із продуктів, відсутність віконного додатку під ОС Windows, Linux у Soundiiz, який міг би спростити доступ користувачів до сервісу. Також при великому навантажені на дані ресурси можливе «зависання» програми на деякий час, для того аби не призвести даних незручностей для користувача накладаються обмеження по кількості створених треків в одному із списку відтворень. Відповідно до

проведеного огляду було встановлено переваги та недоліки існуючих систем. В відповідності до наявних недоліків постає завдання по реалізації аналогу існуючих рішень, що забезпечить поєднання переваг та мінімізує кількість недоліків. Виходячи з огляду аналогів завдання на реалізацію програмного засобу складається з написання системи, що міститиме в якості представлення для користувача графічний віконний додаток. Дана система повинна поєднувати всі переваги сервісів, а саме: можливість автоматичного завантаження музичних композицій на основі завантаження спеціалізованого формату файлів, які створюються за допомогою аудіо-програвачів з різним розширенням, можливість отримання списку відтворення за допомогою URL-посилання, надати змогу користувачеві для вільного і та можливість пошуку композиції в наслідок вводу тексту, який буде зчитано і знайдено відповідний трек у кожному із сервісів. Також потрібно уникнути недоліків, які будуть спровоковані за умови обробки великого обсягу даних: наявність суттєвого та помітного зниження швидкості відклику. Важливим аспектом є наявність платної підписки, яка накладає багато суттєвих обмежень по зручності користування.

Для того щоб створити комплексне програмне рішення в межах описаної вище задачі, необхідно визначити загальний тип міжкомпонентної взаємодії в межах побудови архітектурного рішення, виходячи з критерію про наявність декількох пристроїв в одного користувача та значної кількості кінцевих користувачів сервісу, залишаються наступні архітектурні рішення: клієнт-серверна архітектура [3], сервіс-орієнтована архітектура (SOA, англ. service-oriented architecture) [4].

Клієнт-серверну архітектуру можна сприймати як концепцію інформаційної мережі, де головна частина ресурсів знаходиться на серверах, які обслуговують своїх клієнтів. Дана архітектура складається із наступних компонентів:

1) Набір серверів, які обробляють запити клієнтів.

2) Набір клієнтів, що звертаються до серверної частини системи.

3) Рівень комунікації, що базується на мережевих протоколах взаємодії та забезпечує обмін інформацією між компонентами системи, а саме клієнтами та серверами.

Сервіс-орієнтована архітектура(SOA) – модульний підхід для розробки програмного забезпечення, в якому компоненти розподілені в різних вузлах мережі, які є слабоповя 'заними між собою, а також мають стандартизовані інтерфейси для взаємодії за стандартизованими протоколами рис. 1.

Програмні комплекси, які розроблені на основі сервіс-орієнтованої архітектури, реалізуються як набір веб-служб, взаємодіючих зачасту про протоколу SOAP [5].

Один із варіантів сервіс-орієнтованої архітектури – мікросервісна архітектура.

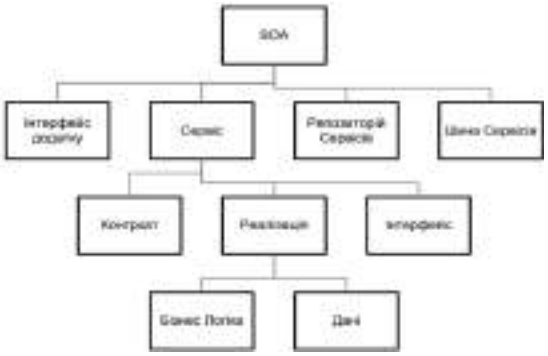


Рис. 1. Структура SOA

Архітектурний стиль мікросервісів – це підхід, коли єдиний додаток реалізовується як сукупність невеликих, самодостатніх, незалежних, не тісно зв'язаних сервісів, що спілкуються між собою за допомогою набору наступних протоколів: HTTP, gRPC, AMQP. Кожен сервіс побудований навколо певних бізнес-потреб і є відповідальним за якийсь конкретний бізнес-процес та розгортається незалежно один від одного за допомогою повністю автоматизованого середовища [6]. Для реалізації кожного сервісу в межах мікросервісної архітектури може бути використана одна із орієнтованих під дану задачу технологій, що надає можливість при створенні в межах окремо взятого сервісу абстрагуватись від технологій розробки та відповідних мов програмування за допомогою яких були реалізовані інші сервіси. Кожен сервіс як окремий компонент мікросервісної архітектури може містити взаємодію з різними типами джерел даних, без привязки до конкретного типу в межах всієї архітектури. Даний підхід має наступний список переваг:

1) Спрощення доповнення існуючого, та модернізації функціоналу за допомогою додавання окремого компоненту в архітектуру чи зміни існуючого відповідно.

2) Можливість абстрагування від існуючих компонентів архітектури, що дозволяє незалежно змінювати логіку поведінки окремо взятої архітектурної одиниці.

3) Відмовостійкість, що забезпечується можливістю розгортання нового екземпляру архітектурної одиниці(сервісу) у випадку його відмови.

У відповідності до специфікації технічного завдання на розробку та специфіки і обмежень, переваг та недоліків кожного з видів архітектурних підходів, було прийняте рішення про реалізацію взаємодії клієнтської сторони додатку з сервером в межах клієнт серверної архітетури та реалізацію серверної частини на основі мікросервісної архітектури. Загальна схема компонентів системи спроектована та продемонстрована на рис. 2.

Розглянемо серверний компонент архітектури. Виходячи з критерію завантаженості класу подібних

Рис. 2. Схема роботи системи

підсистем серверне рішення по агрегації контенту та його отриманню, було спроектоване на основі мікросервісної архітектури, що забезпечує можливість горизонтального та вертикального масштабування [7] кінцевого рішення в відповідності до навантаження в поточний момент часу. Взаємодія в межах мікросервісного рішення побудована з використанням шини даних, що забезпечує централізоване управління міжсервісним трафіком та гарантує його отримання екземпляром конкретного мікросервісу. Шина даних є вхідною точкою в механізм між сервісної комунікації та відповідає за побудову маршрутів для мережевих пакетів. Для отримання обмеженого захищеного доступу до даних користувача в кожному із музичних сервісів на клієнській частині додатку необхідно реалізувати авторизацію на основі протоколу OAuth2.0, який підтримує кожен із сервісів, за допомогою якого буде надано маркер доступу (access token), який в свою чергу надаватиме можливість виконувати дії по обробці даних без безпосередньої участі користувача у даному процесі.

Технічне рішення

Для реалізації спроектованих архітектуриних рішень було проаналізовано наявні технології, на основі яких можна розробити програмний продукт в межах обумовленої архітектури. Було розглянуто перелік технологій, що надають інструментарій по реалізації міжклієнтської взаємодії в межах клієнт – серверної архітектури програмного рішення такі як JAVA, .NET. На основі порівняння технологій [8] WPF і Swing, а саме на основі характеристик ресурсозатратності по швидкодії, оперативній пам’яті, та складності реалізації. Результати порівняння продемонстровано на рис. 3.

Було прийняте рішення по реалізації застосунку на основі платформи .NET, як оптимальної по співвідношенню вище перерахованих критеріїв. Дана платформа включає в себе наступні переваги:

1) Забезпечення узгодженого об'єктно-орієнтованого середовища програмування для локального збереження і виконання об'єктного коду, для локального виконання коду, розподіленого в інтернеті, або для віддаленого виконання.
2) Забезпечення середовища виконання коду, що мінімізує конфлікти при розгортанні програмного забезпечення та управлінні версіями.
3) Забезпечення єдиних принципів розробки для різних типів додатків, таких як додатки Windows і веб-додатки.

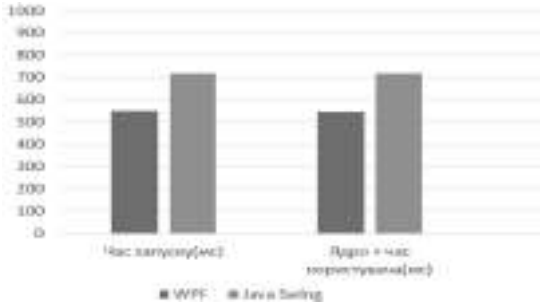


Рис. 3. Порівняння швидкодії WPF та Swing

Станом на сьогодні в межах платформи .NET для реалізації графічної клієнтської частини виділяють технології WPF і WinForms. WinForms – технологія для створення графічного інтерфейсу користувача. Доступ до елементів інтерфейсу відбувається за допомогою обгортки Win32 API в керованому коді [9]. Технологія WPF (Windows Presentation Foundation) є частиною платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

При створенні додатків на основі WinForms за відображення елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI+ [10], то додатки WPF забезпечують побудову графіки застосунку за допомогою технології DirectX. У цьому полягає ключова особливість побудови графічного відображення (рендеринга) в WPF. Використовуючи WPF, значна частина роботи по відображенні графіки, виконується графічним процесором на відеокарті, що також дозволяє скористатися апаратним прискоренням графіки.

Аналізуючи описані вище переваги та наведене порівняння даної технології з аналогами, було прийнято рішення обрати технологію WPF для розробки графічного інтерфейсу користувача. Для реалізації серверної частини додатку в межах архітектурного рішення, на даний момент часу виділяють наступний набір технологій Spring (Java), ASP.NET Core (C#) , GGI (C++). Розглянемо дані технології більш детально. Технологія Java Server Pages (JSP) [11] є компонентом єдиної технології створення застосувань що базуються на J2EE, з використанням web інтерфейсу. Дана технологія аналогічна до ASP Web Forms, на разі її застосування обмежене підтримкою вже існуючих проектів. На зміну даній технології прийшло рішення в вигляді фреймворку Spring, що містить розмежування логіки та представлення. Причому даний фреймворк побудований з урахуванням принципів S.O.L.I.D., що надає гнучкості коду в межах архітектури додатку. Відповідно до порівняльної характеристики фреймворків наведена на рис. 4., фреймворк spring швидший, а також забезпечує архітектурно коректний підхід до реалізації серверного застосування.

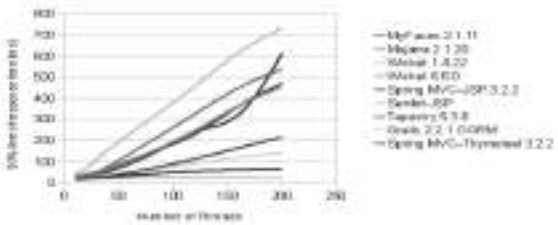


Рис. 4 Порівняння фреймворків

Також наявна широко використовувана технологія CGI (загальний інтерфейс шлюзу), яка спеціально застосовується для створення динамічних веб-сторінок. Відповідно до технології CGI, HTTP запит, який містить посилання на динамічну сторінку, в момент опрацювання веб-сервером, генерує новий процес і запускає потрібну прикладну програму. Технологія CGI для розробки веб-додатків можна використовувати скрипти наприклад Python, Perl, Tcl. Відповідним недоліком такого підходу є відсутність можливості побудови складного рішення в межах архітектури поточного серверного застосування.

Однією із найновіших технологій для розробки веб-додатків – ASP.NET Core. Дана технологія є альтернативою розвитку платформи ASP.NET. ASP.NET Core може працювати поверх кросс-платформної середовища .NET Core, яке може бути розгорнуте на основних популярних операційних системах: Windows, Mac OS, Linux, що забезпечує можливість створення кросс-платформених рішень. Порівнюючи технології CGI та ASP.NET Core [12] можна дійти до висновку, що CGI є складна в реалізації та містить високий поріг для входу спеціалістів, а також не надає можливості побудови коректного архітектурного рішення, що є ключовим фактором для реалізації підсистем даного класу. Порівняння швидкості опрацювання одного і того самого запиту серверним рішенням реалізованим на GGI та ASP.Net зображено на рис 5.

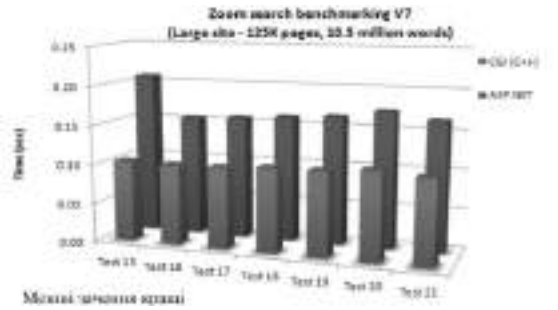


Рис. 5 порівняння швидкодії реалізацій серверних рішень CGI та ASP.NET CORE

В межах огляду було зроблено порівняння техгологій Spring і ASP.NET Core [13] по критеріям швидкості перетворення структури даних у послідовність бітів (серіалізація) результати якого наведено на рис. 6. На основі наведених порівнянь технологій по набору вказаних критеріїв, таких як швидкість обробки запиту, ресурсоємність в межах оперативної пам’яті та процесорного навантаження, а також складності реалізації поставленого завдання

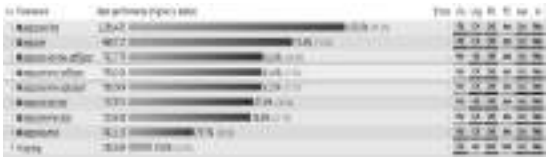


Рис. 6 Порівняння швидкодії серіалізації структур даних на базі ASP.NET Core та системи на базі Spring

за допомогою технології було прийняте рішення про оптимальність використання технології .net core для побудови серверного рішення в межах мікросервісної архітектури для обумовленого технічного завдання на розробку.

Висновки

Під час написання статті було оглянуто завдання централізованого управління музичними сервісами і проаналізовано існуючі аналоги для вирішення поставленої задачі. Під час огляду рішень було виявлено ряд недоліків, які були пов’язані з платною підпискою і накладанням обмежень на користувача, а також можливе навантаження на систему під час обробки великої кількості даних. Для вирішення завдання було змодельоване комплексне архітектурне рішення, складовими частинами якого є клієнт – серверна архітектура в межах взаємодії клієнтського додатку з сервером системи та рішення на основі мікро-сервісної архітектури для серверної частини застосунку в відповідності до вказаних вимог та критеріїв. Під час огляду технічного рішення для реалізації поставленої задачі було проаналізовано і порівняно технології, мови програмування за критеріями швидкодії системи і часових затрат для написання даного продукту. В результаті порівняння було зроблено висновок, про вибір платформи .net core та технологій ASP.NET Core для реалізації кожного мікросервісу в межах серверу та технології WPF для реалізації клієнтського застосунку.

Література

[9] https://www.mulesoft.com/resources/api/what-is-an-api (дата звернення 05.05.2020).
[10] https://oauth.net/2/ (дата звернення 06.05.2020).
[11] https://www.w3schools.in/what-is-client-server-architecture/ (дата звернення 10.05.2020).
[12] https://martinfowler.com/articles/microservices.html (дата звернення 10.05.2020).
[13] https://searchapparchitecture.techtarget.com/definition/SOAP-Simple-Object-Access-Protocol (дата звернення 11.05.2020).
[14] https://www.safetydetectives.com/blog/what-is-rogue-security-software-and-how-to-protect-against-it/(дата звернення 11.05.2020).
[15] http://dspace.nbuv.gov.ua/handle/123456789/161488(дата звернення 11.05.2020).
[16] https://www.codeproject.com/Articles/92812/Benchmark-start-up-and-system-performance-for-Net(дата звернення 14.05.2020).
[17] https://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/API/Win32(дата звернення 14.05.2020).
[18] https://docs.microsoft.com/en-us/windows/win32/gdiplus/-gdiplus-overview-of-gdi--about(дата звернення 14.05.2020).
[19] https://www.tutorialspoint.com/jsp/jsp_overview.htm (дата звернення 15.05.2020).
[20] https://ela.kpi.ua/bitstream/123456789/27860/1/Vovk_magistr.pdf (дата звернення 15.05.2020).
[21] https://www.techempower.com/benchmarks/ (дата звернення 20.05.2020).

Система тестування на основі відносного оцінювання по статистичним вибіркам

Коваленко Ігор
КПІ ім. Ігоря Сікорського
Київ, Україна
meelistenso@gmail.com

Вовчок Євген
КПІ ім. Ігоря Сікорського
Київ, Україна
vovchok.zhenya@gmail.com

Анотація. **Мета публікації – переглянути наявні способи вирішення завдання по проведенню оцінювання якості знання способом тестування і змоделювати архітектурне рішення, яке поєднує всі переваги, а також не буде містити недоліків оглянутих аналогів. Оптимальним рішенням в межах обумовлених в статті критеріїв є програмний продукт створений на основі клієнт – серверної архітектури в якій серверний компонент реалізовано у вигляді єдиної архітектурної одиниці, для забезпечення простоти розгортання та зниження вартості підтримки системи**

Ключові слова: системи тестування; встановлення рівня знань; статистична вибірка; клієнт–серверна архітектура.

У сучасних реаліях ні одна сфера життя не обходиться без оцінки якості рівня знань та підготовки спеціалістів по тим чи іншим напрямам. У результаті постає завдання якісної та коректної оцінки знань, з можливістю їх подальшого вдосконалення в межах процесу навчання, який обумовлено процесом виявлення відсутності знань в межах певного представленого графу знання, по тій чи іншій спеціальності. Дане завдання в сучасному світі не може бути не вирішеним, проте розглядаючи існуючі рішення та наявні життєві приклади можна говорити про недосконаліть, неточність та нешаблонність даних рішень.

Після огляду популярних систем тестування, представлених на веб-порталах, зокрема, мережевої академії Cisco, Canvas, Blackboard Learn, Kahoot!, та на основі досвіду проходження тестів під час навчання, було сформовано перелік основних типів тестів. Для повної реалізації функціональних вимог і конкурентоздатності з переліку було обрано наступні види тестів, які повинні міститися у системі:

1) Правильно/неправильно – користувач повинен визначити чи правдиве твердження в питанні. Це найпростіший тип питання.
2) Вибір однієї відповіді – користувачу потрібно вибрати одну правильну відповідь з запропонованих варіантів.
3) Вибір кількох відповідей – користувач обирає правильні варіанти зі списку. Завдання такого типу складніші чим вибір однієї відповіді, так як кількість правильних відповідей заздалегідь невідома. Правильна відповідь методом випадкового підбору малоймовірна.
4) Коротка відповідь – користувач повинен ввести правильну відповідь в текстове поле. Для правильної відповіді необхідно добре розбиратись у тематиці поточного питання.
5) Послідовність – користувач розміщує елементи у правильній послідовності.
6) Числова відповідь – користувач вводить число у поле для відповіді. Вгадати правильну відповідь малоймовірно.
7) Відповідність – користувач повинен з'єднати пари слів, фраз, або зображень. Додаткові «зайві» варіанти відповідності можуть ускладнити питання.

При створенні тесту є необхідність виставлення прохідного балу. Крім того, існує проблема встановлення еквівалентності ваги правильних відповідей, оскільки різні питання можуть посилатися на різні об'єми знань, можуть мати неточності, бути складними для розуміння. Універсальний рецепт для фіксованого значення прохідного балу відсутній, тому необхідно зробити його відносним. Як рішення для забезпечення об'єктивності тестування обрано динамічне визначення ваги правильних відповідей відносно загальної статистики відповідей користувачів на питання тесту у межах системи.

Для того, щоб охопити якнайбільшу область знань, але при цьому не виснажувати користувача, понижуючи його продуктивність було вирішено надати можливість встановлення довільної кількості запитань адмінінстратором системи та встановлено рекомендований обсяг 25-30 питань. При цьому загальний банк питань повинен бути не менше ніж у 3-4 рази більшим, ніж кількість питань у згенерованому адміністратором тесті з метою забезпечення унікальності тестів у різних користувачів і виключення можливості плагіату та розповсюдження набору готових питань-відповідей на тест. Також для виключення можливості користування додатковими матеріалами користувачем необхідно, щоб тест передбачав обмеження по часу, яке встановлюється адміністратором.

На сьогоднішній день існує безліч систем онлайн-тестування. Скориставшись пошуковою системою Google, та переглянувши знайдені веб-

ресурси і рейтинги, можна знайти ряд схожих за функціоналом та призначенням систем.

Відповідно до призначення тести поділяються на:

1) Навчальні – ті, що допомагають закріпити вивчений матеріал. Зазвичай такі тести можна знайти після кожної глави у курсі в якості невеликої практики. Відсутні обмеження по часу, штрафи за неправильні відповіді. На вирішення задач надається кілька спроб. Після кожної помилки відображаються пояснення. Як приклад, тести що стають доступні після завершення вивчення глав у системі мережевої академії Cisco.

2) Атестаційні – допомагають оцінити знання користувача. В таких тестах присутні обмеження по часу, дається одна спроба на відповідь, відсутні пояснення помилок. Як приклад, Exam.net.

За спеціалізацією, тестові системи поділяються на:

1) Спеціальні – ті, що призначені для застосування лише у певних сферах. Як приклад, нині популярні системи тестування персоналу SHL, TalentQ, Kenexa, або система тестування інтегрована у платформу Udemy, призначена для оцінки засвоєних навичок після проходження курсу.

2) Загальні – ті, що мають широке призначення. Різноманітні конструктори тестів та більш загальні системи як Google Forms.

Більшість наявних систем не об’єднують тестування та навчання шляхом встановлення незнання та його ліквідацію за допомогою опису і вказання правильних відповідей шляхом пояснень. Крім того ні одна з них не вирішує проблему еквівалентності ваги правильних відповідей. Також присутня проблема різноманітності форми тестових завдань і сучасності UX [1] частини систем. Як приклад Moodle, мережева академія Cisco мають застарілий дизайн та вузький перелік форм тестових питань.

1) У результаті огляду прийнято наступні вимоги до проектованого рішення:

2) Можливість додавання пояснень та посилань до відповідних питанням тем та їх пов’язування.

3) Вирішення проблеми еквівалентності ваги правильних відповідей шляхом впровадження динамічного показника відносного загальній статистиці відповідей на питання тесту.

4) Впровадження різноманітних форм тестових питань з можливістю конфігурації.

5) Розробка сучасного дизайну.

6) Широка спеціалізація системи, шляхом налаштувань формату тестів.

Відповідно до сформованих вимог про наявність багатьох користувачів та необхідність централізованого збереження інформації про результати тестів, їх оцінювання, для даного класу підсистем доцільне використання наступних архітектур міжкомпонентної взаємодії: SOA, client-server [2].

Клієнт-серверний підхід

Даний шаблон складається з двох частин: сервера і безлічі клієнтів, рис 1. Серверний компонент надає сервіс клієнтським компонентам. Клієнти запитують сервіс у сервера, а він, у свою чергу, надає їх клієнтам [3].

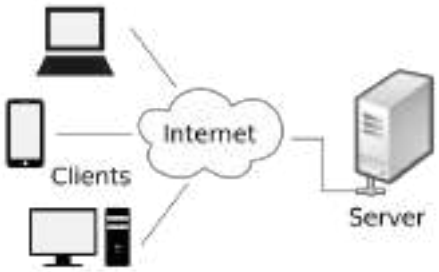


Рис. 1. Схема комп'ютерної мережі клієнтів, що спілкуються з сервером через мережу Інтернет

Зазвичай використовується для реалізації онлайн додатків (електронна пошта, спільний доступ до документів, банківські послуги, тощо).

Переваги:

1) Підходить для моделювання набору сервісів, які можуть запитувати клієнти.

Недоліки підходу:

1) Запити зазвичай виконуються в окремих потоках на сервері.

2) Взаємодія між процесами підвищує ресурсовитратність, тому що різні клієнти мають різне представлення.

Підхід SOA

Сервісно-орієнтована архітектура (SOA) – спосіб побудови міжкомпонентної взаємодії програмного забезпечення, за якого послуги надаються іншим компонентам компонентами додатків, через протокол зв'язку по мережі [3]. Сервіс SOA - це дискретна одиниця функціональності, до якої можна отримати доступ дистанційно, діяти та оновлюватись незалежно, наприклад, отримання виписки з кредитної картки в Інтернеті.

В межах SOA архітектури виділяють наступні елементи, які відображені на рис. 2.

Рис. 2. Елементи SOA

Сервіс відповідно до визначення SOA, має наступні властивості:

1) Представлення поведінкової логіки із заданим результатом.
2) Обмежена відповідальність в межах одного бізнес-завдання.
3) Це чорна скринька для споживачів, тобто споживач не повинен знати про внутрішню роботу служби.
4) Можливість об'єднання набору інших сервісів.

Різні сервіси можуть бути використані спільно, для забезпечення функціональності складної логіки поведінки по комплексній обробці користувацького запиту, що потребує поєднання окремо взятої логічної поведінки в межах єдиної відповідальності кожного сервісу для видачі результату. Проте даний підхід має ряд недоліків:

1) Неоднорідність і складність рішення.
2) Величезний набір тестових комбінацій завдяки інтеграції автономних сервісів.
3) Включення послуг від різних конкуруючих постачальників.
4) Платформа постійно змінюється через наявність нових функцій і послуг.

Технічне рішення

Під час розробки технічного рішення для серверної частини було розглянуто найпопулярніші технології для побудови веб застосунків а саме CGI (C++), ASP.NET CORE (C#) та Spring (Java) [1].

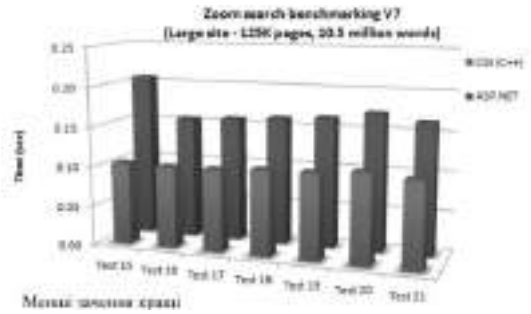


Рис. 3. Порівняння швидкодії реалізацій серверних рішень CGI та ASP.NET CORE

В першу чергу було розглянуто порівняльну характеристику CGI та ASP.NET CORE, а саме швидкість пошуку та видачі даних на порівнюваних платформах. Виходячи з даних графіку представленого на рис. 3, рішення на базі CGI має більшу швидкодію. Проте, на відносно невеликих об'ємах даних розрив у швидкодії скорочується. Крім цього розробка на базі мови C++ є значно витратнішою та займає більше часу, через технічні особливості мови та наявність безпосереднього управління оперативною пам'яттю в межах створюваного процесу [7].

В межах подальшого огляду було проведено порівняння ряду конфігурацій систем ASP.NET CORE та системи на базі Spring [8], результати якого представлено на рис. 4.

Рис. 4. Порівняння швидкодії різних конфігурацій систем на базі ASP.NET CORE та системи на базі Spring

Виходячи з результатів тестів та порівнянь, були зроблені висновки, що для розробки об'ємного проекту найкраще підходить технологія ASP.NET CORE. Протягом вирішення завдання по збереженню та взаємодії з даними, до порівняння було взято відомі СУБД, а саме: PostgreSQL, MS SQL і IBM db2.

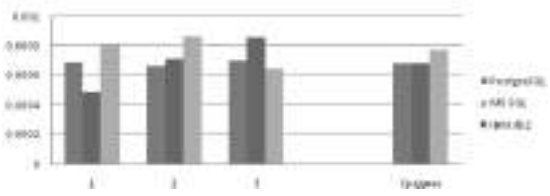


Рис.5. Порівняння швидкодії реалізацій SQL серверних запитів

На основі даних, представлених на рис. 5, було виявлено перевагу MS SQL у швидкості реалізаціїї серверних запитів порівняно з рештою розглянутих СУБД. Виходячи з обраних технологій, постало питання прийняття рішення по вибору способу взаємодії між сервером і базою даних. Оптимальним вибором для прийнятого стеку технологій є Entity Framework [5] на основі наступних критеріїв:

1) Наявність підтримки способів генерації бази даних CodeFirst, DataBaseFirst.
2) Наявність можливості написання запитів засобами мови LINQ та їх подальше виконання всередині бази даних SQL.
3) Містить генерацію бази даних з відповідно заданої об'єктно-орієнтованої моделі.
4) Наявність механізму кешування даних.

Під час розробки технічного рішення для клієнтської частини було розглянуто найпопулярніші фреймворки односторінкових веб-додатків, а саме Vue.js, Angular та React.

Розглянувши результати порівняння швидкодії, представлені на рис. 6, було визначено, що всі перелічені фреймворки мають схожу продуктивність під час виконання програми. Виділяється лише швидкість першого завантаження сторінки з використанням Angular. Це пояснюється тим, що даний фреймворк включає в собі ряд бібліотек, призначених для організації коду, взаємодії з зовнішніми джерелами даних, маршрутизаціїї всередині додатку, організації потоків даних. У випадку розробки веб-орієнтованого додатку зі складним інтерфейсом це є значною перевагою, оскільки решта фреймворків потребує власної імплементації даного функціоналу, що у подальшому значно впливає на

швидкодію продукту та ускладнює розробку системи.

Виходячи з цього для розробки клієнтської частини було обрано фреймворк Angular.

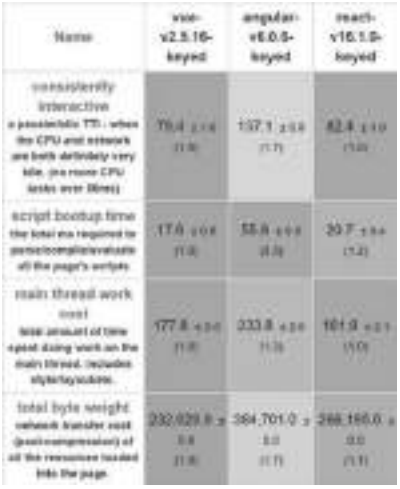


Рис. 6. Порівняння швидкодії фреймворків Angular, React та Vue.js

Власна реалізація

Під час програмної обробки в межах механізму оцінювання тесту робота якого обмежена специфікою роботи Web API та специфікацією HTTP протоколу виникають наступні технічні проблеми:

1) Перерахунок ваги кожного питання.
2) Оновлення спільних даних конкурентними потоками.

Був розроблений механізм перерозподілу ваги відповідей на питання, на основі побудови відношення оцінки кожного окремого питання в тесті, що враховує кількість правильних та неправильних відповідей на нього користувачами. Відповідно, по завершенню проходження окремо взятого тесту кожним з користувачів, серверна частина застосування в фоновому потоці виконує перерозподіл додаткового вагового балу кожного питання, базуючись на оновленні статистичної вибірки, а саме зміні значень кількості правильних та не правильних відповідей на окремо взяте питання рештою користувачів, що пройшли даний тест. Для встановлення ваги окремо взятого питання, в відповідності до описаного алгоритму, маємо наступну формулу:

$$w = \frac{100 Q_m}{\sum_{0}^{Q} Q_m},$$

де w – це вага за питання, Qm – це кількість не вірних відповідей для одного запитання, Q – це кількість питань в тесті. При встановленні значення ваги запитання попередньою формулою є ризик того, що вага питання може прямувати до нуля, що є критичним в нашій системі. Для рішення цього завдання було зроблено обмеження, яке корегується наступною формулою:

$$w_{\min} = \frac{100}{\sqrt{Q^3}},$$

де $w_{\min}$ – це мінімальна вага за питання, Q – це кількість питань в одному тесті.

Під час моделювання системи було виявлено наступну технічну проблему, обумовлену можливістю появи конкуруючих потоків, що в свою чергу будуть оновлювати спільні набори даних в один і той самий момент. З метою вирішення технічної проблеми було розроблено наступний механізм, що на основі планувальнику задач реалізує централізоване управління операціями зміни значень з точкою синхронізації, що створюється в межах додаткового джерела даних, на основі записів поточних та запланованих задач. В якості планувальника задач було використано Hangfire [8], як одне з найпопулярніших рішень в межах технології що використовується. Такий підхід забезпечує рішення технічної проблеми конкурентих потоків в межах всієї системи, що в свою чергу надає можливість зміни та збереження статистичних даних консистентно.

Висновки

В межах наявного огляду виділено клас підсистем направлених на встановлення оцінки якості рівня знань та підготовки спеціалістів по тим чи іншим напрямам. Визначено їх переваги та недоліки, на основі яких встановлено функціональні вимоги до даної системи. Виходячи з функціональних вимог було обрано, як архітектурне рішення клієнт-серверний додаток. Для реалізації архітектурного рішення та функціональних вимог було проведено огляд технічних засобів, що застосовують для створення даного класу систем і сформовано технічне рішення, яке включає побудову односторінкового додатку на основі фреймворку Angular на боці клієнтської сторони системи та додатку на основі платформи .NET Core та СУБД MS SQL Server з боку серверної частини. У даній системі реалізовано логіку динамічного визначення ваги правильних відповідей, яка забезпечує об’єктивність результатів тестів.

Література

[1] https://ela.kpi.ua/bitstream/123456789/27860/1/Vovk_magistr.pdf (дата звернення 12.05.2020).
[2] Леон Шкляр, Річ Розен., книга «Архітектура веб - додатків», рік видання: 2011 року
[3] https://echo.lviv.ua/dev/6455 (дата звернення 11.05.2020).
[4] https://www.ibm.com/developerworks/ru/library/ws-soa-term1/index.html (дата звернення 11.05.2020).
[5] https://habr.com/en/post/262461/ (дата звернення 12.05.2020).
[6] https://www.techempower.com/benchmarks (дата звернення 12.05.2020).
[7] https://docs.microsoft.com/en-us/ef/ef6/ (дата звернення 12.05.2020).
[8] https://www.hangfire.io/ (дата звернення 12.05.2020).

Bot creating technology for wide use based on a logical approach

Telenyk Sergii
Igor Sikorsky Kyiv Politechnic Instiitute
Kyiv, Ukraine
Cracow University of Technology
Cracow, Poland

Zharikov Eduard
Igor Sikorsky Kyiv Politechnic Instiitute
Kyiv, Ukraine

Vovk Yevhenii
Igor Sikorsky Kyiv Politechnic Instiitute
Kyiv, Ukraine

Nowakowski Grzegorz
Cracow University of Technology
Cracow, Poland

Tokmenko Olena
Taras Shevchenko National University
Kyiv, Ukraine

Abstract. **The report proposes the information technology for computer-aided design and implementation of bots based on a logical approach. The general approach to creating bots with the use of a complex of logical and linguistic models has been described. The general scheme of work of the web-based system with the use of a chatbot has been developed. The model of logic of actions, and the inference method have been described, which can be used for the formal description of the bot creating process. An example of using a logical model to create a bot has been described**

Keywords: information technology, messangers, bot, automated bot creating, logic of actions.

INTRODUCTION

Human activity, which can be work, study or leisure, is regulated by template-algorithmic processes. For institutions, enterprises, and companies, it means waiting in queues, writing applications, moving from one office to another. In the sphere of rest, it is the search for a company of people who are ready to spend their free time in a certain way. In educational institutions, it is, on the one hand, notifying students and their parents about changes, events happening or will happen in the institution, and on the other hand, providing answers to questions from students and their parents about such changes, events, issuing various certificates. Such processes regulate the work, but introduce delays.

There is a need for technologies for managing virtual human interaction in solving a wide range of problems. The tools of interaction are electronic document management systems, emails, services of educational institutions for student (pupil) and teacher interaction, specialized services for finding people by interests, location, etc., various messengers, social networks.

However, the use of such tools of the information society has disadvantages that have negative consequences for system owners and their users. The system often pays a relatively small but fixed rent during the period of use of the system within the license agreement with the developer, or one-time but significant costs for developing such a system. In this case, regardless of the method of acquisition, the system owner is not deprived of the costs for maintaining this system in working order by adding the necessary content, scaling in case of increasing the number of users, etc.

In the case of the user, the disadvantages are related to the use of the system. Because modern users have access to online management systems for almost every component of their lives, and each of these systems has its interface that does not always correspond to the usual UX user, as well as its authorization method, login, and password (when using a single password on all systems the user violates the rules of safe use of web systems), there is a need for regular decentralized review of information on each area of life within the system or a set of systems, remembering a large number of passwords and features of each system. At least, this leads to extra time, and in general, we can talk about a set of disadvantages with negative consequences for the user.

In general, the use of decentralized management systems for each individual component of the social component of the user is time-consuming for them, and resource-intensive for the system owner. Therefore, there is a problem which the centralized management of all components of user activity in social networks.

The development of modern technologies, the experience of using solutions based on them have shown that the problem described above can be partially solved by means of rapid automated creation and use of wide-class chatbots. It would be good to supplement such solution with a system which integrates the user's work with all messengers, social networks and information systems of the information society in terms of content, graphics, and owner's features, but it should be the subject of another article by the authors.

PROBLEM STATEMENT FOR DEVELOPING THE BOT CREATING TECHNOLOGY

It is necessary to develop a general approach, mathematical models and methods as the basis of information technology to create wide-class chatbots. Essentially, it means solving traditional problems of coordination in such schemes 'employee-to-employee', 'customer-to-dispatcher', 'student-to-dean's office specialist', 'student-to-teacher' with the help of a new solution aimed at creating special programs – bots, which by using the capacities of social network messengers automate human functions related to informing, ordering and fulfilling orders, implementing multi-step processes, separate stages of which depend on various circumstances, events and participants' features. The general approach, such models and

methods should be aimed at the rapid creation of programs that implement the selection and implementation of the interaction (service) scheme from many possible options for their layout with subschemas depending on the type and the user's individual features, while being deprived of algorithmic approaches associated with the need to reprogram the bot when user's new features appear, implemented subschemas change, or new subschemas occur. This technology must meet a high level of requirements, especially its following features: flexibility, scalability, accessibility to the end-user without special knowledge and skills, reliability, convenience and ease of use, distribution, use of remote resources, load adaptation, fast and simple integration with information systems.

ANALYSIS OF PUBLICATIONS

Instant messaging systems are widely used by individual users and human communities around the world. In fact, today they have evolved from a means of communication between people into a means of obtaining information, and into an incredibly powerful marketing tool. The key part, when using them, is the ability to communicate with bots. Therefore, this report discusses the basic principles of creating bots based on logical and linguistic models within an approach that captures, accumulates, generalizes, and uses the user's positive experience to ensure maximum efficiency of their social network activity. It is important to review developments in this area, analyze scientific publications to select necessary tools for bot development, the basis to integrate into a single system, and examples to demonstrate the efficiency of the proposed solution.

In the last few years, the popularity of instant messaging systems (messengers) has been increasing. Bots have played a significant role in evolving a means of communication. The de facto standard definition of a chatbot has become that it is a computer program responsible for user interaction based on auditory or textual methods. There are many attempts to classify bots, from which it is advisable to choose the classification of chatbots by: the type of messages (commands) – text and voice; the level of interactivity – bots with the response to the call of the specified command and the message analysis, and the team (teams) selection; the type of interaction with the chat server – Long polling and Listeners.

Most available messengers provide the ability to select the interface of the communication level of the chatbot and the messenger server. The next choice is offered at the level of interaction type, where there are two modes of interaction – Long polling, which regulates the messenger server polling via API once in an interval of N milliseconds, where the specification of the number of milliseconds is regulated by a specific API, event subscription mode, while the mentioned API Endpoint will be called if there is a change. With regard to the advantages and disadvantages of regimes in view of the problem of the study, the following well-grounded conclusions can be drawn. The bot performance using the first mode creates an additional load in the messenger server, but does not require an SSL certificate for an external address, and is used mainly in mobile and desktop applications. The bot performance using the second mode is designed for interaction of the 'server to server' type, and reduces the load on the messenger server. Therefore, when implementing the solution in accordance with the principles of web-based architecture, it seems more appropriate to use the second mode.

This useful tool is interesting for business. Bots have been actively used abroad for several years, and are gaining popularity in Ukraine. Thus, a promising niche for IT developers has been created. An important aspect for potential customers is the bot command perception, good understanding of the user's query, and the relevance of its answers.

The report proposes a holistic approach to creating bots for the messenger, describes the architecture of information technology for creating bots based on the models of mathematical logic, artificial intelligence, and linguistics to implement bot functions and its communication with users.

GENERAL DESCRIPTION OF THE AUTOMATED BOT CREATING APPROACH

The proposed solution does not cover the whole set of tasks related to the centralized management of all components of user activity in social networks by creating web-based technologies. However, it can simplify user interaction with the system, gain experience, and create conditions for more efficient solutions.

The implementation of such a solution is to create user interaction through a chat, within a specific messenger, based on a targeted query processing platform that implements a complete algorithm of actions in all cases provided by a particular chat. Fig. 1 shows the general performance of the server architecture of the bot performance level.

Fig. 1. A fragment of the server architecture of the bot performance level

Let's analyze the fragment components. The Message provider component performs the functions of wrapping the message source (messenger). Handlers process messages from the Message provider, delegate tasks to services, update the State storage, and generate responses to the user. The State storage stores the parameters of state while processing user queries. Services implement additional logic for processing user queries.

A chatbot can be used, for example, for registration, ordering a taxi, planning the performance of such order, and monitoring the implementation of the plan in the system of the international provider of this service. A typical task for chatbots is to order a car. The need to

integrate the mechanism of receiving user queries via a chatbot is due to the advantages of user interaction with messengers described above. One of the possible options for specifying the actions, states, and requirements for the input data model for the taxi ordering process takes the form of the following sequence:

1. Country selection.
2. City selection.
3. Car class selection.
4. Determining the customer's mobile phone number.
5. Clarification of the pickup time.
6. Clarification of the pickup address (street, building).

States and transitions are shown in Fig. 2. For each state, there is a relevant list of features.

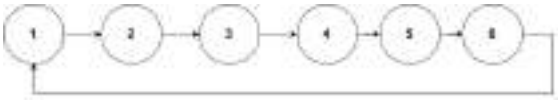


Fig. 2. A fragment of the server architecture of the bot performance level

In fact, we have a traditional model of states and transitions for the control theory, which has turned out to be a convenient form of setting the problem of creating a bot. There are many approaches to implement this model – from a purely algorithmic approach to approaches based on combining problem-solving in the system capacities to solve subproblems.

The above example can be easily implemented based on an algorithmic approach to the sequence of commands created in the chatbot. This easy-to-implement approach is not very convenient for the user and the system owner, as its shortcomings, i.e. lack of interactivity and possible changes in the sequence of actions when implementing the order, lead to significant losses. Thus, adding Action No. 7 to clarify the terms of payment will not change the data, but the software module should be rewritten. Therefore, such a mechanism is simple, but not flexible.

Approaches that respond to state (situation) changes by implementing certain actions, and focus on responding to queries about the possibility of achieving certain states (situations) that learn how to solve problems have proven to be more flexible and efficient. Here, the answer to the query is to generally create an action plan that leads to the target state, and which is then being implemented. To do this, each action of the plan included in the system must be described in terms of objects. Achieving the aim in such systems is associated with the derivation of an action plan that provides the solution to the problem.

However, the implementation of this approach requires the development of a formal logical model for the formation of task conditions, and solutions, and appropriate information technology which is capable to use the experience and knowledge within messengers, bots, the information system to automatically create bots based on logical models and inference methods of appropriate action plans.

ARCHITECTURE OF BOT CREATING TECHNOLOGY

The web-based architecture of information technology for creating bots based on a logical model is shown in Fig. 3.

Such scheme supplements and expands the system functionality, and makes it user-friendly. It will be demonstrated that it also provides the system owner with the benefits of changing the order processing sequence, and adding/removing structural elements to the previously implemented sequence of actions. Let's apply the approach proposed in [4] to the use of templates in creating web-applications for creating templates in determining the sequence of actions. Let's modify the requirements for the input data model for the taxi ordering process:

1. Country selection.
2. City selection.
3. Car class selection.
4. Clarification of the customer's mobile phone number (if absent, then the number available in the messenger).

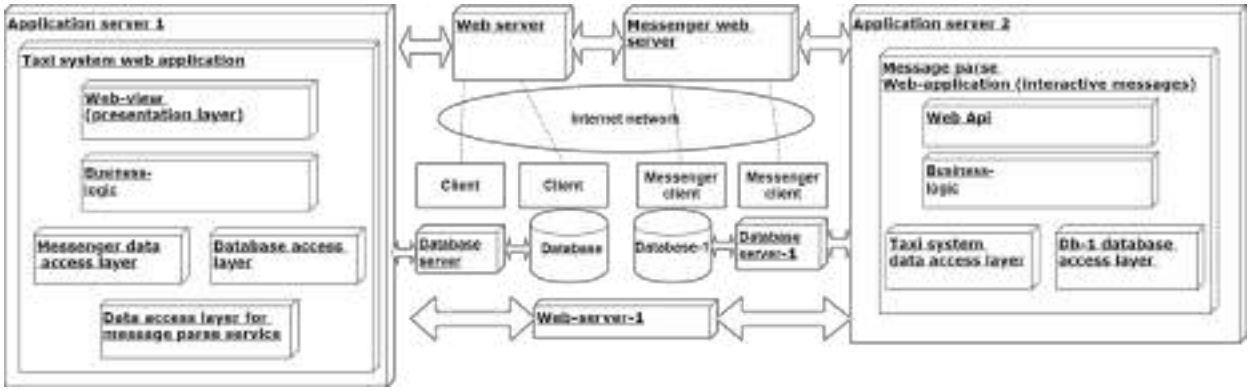


Fig 3. The general scheme of a web-based system performance using a chatbot

5. Clarification of the pickup time. 6. Exact address (street, building).

7. Payment method.

8. Payment (in case of non-cash payment).

9. Use one of the previous routes.

10. Getting an answer about the order status.

The logic of states and transitions in accordance with Fig. 4 has been presented in a graph of a special kind, the problem solving based on it is to find ways using known algorithms.

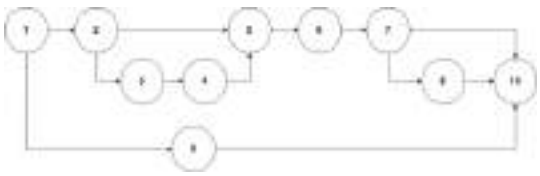


Fig. 4. Modified state storage

The implementation of such behavior is performed by available messenger capabilities. However, if you modify the requirements, and add the possibility of renting a car, which will require checking the driver's license, user's state of health, etc., the algorithm cannot be implemented on the basis of the existing state mechanism, and implemented in modern messengers.

Templating the logic of states for chatbots to form the result based in preconditions and postconditions of each state by constructing a tree of transitions (calls, methods) that correctly process situations from the initial to final precondition and postcondition defined by the user.

Let's analyze a specific scheme for a clearly defined situation. It is necessary to order a car clarifying country, city, car class or make, as a car rental or as a taxi.

The precondition is as follows: the user is authorized, has a license to drive a vehicle. The postcondition is that the user wants to rent a car for one day.

The system has a set of methods that implement sending the agreement, checking the driver's license, city and country, checking the ability of a user to drive the vehicle, work with the payment service. The number of methods is not limited, each of them is characterized by the precondition (determining the need of their call), and the postcondition (determining the results of its fulfillment).

Thus, the system now operates with a set of states defined by the peaks of the graph, in which the transitions are made on the basis of preconditions and postconditions, and for each individual user's query, the precondition (input peak), and the desired result (postcondition) have been specified. Under this approach, to implement the technology of automated creation of the application using the chatbot module, you need:

1) Template the logic of state construction based on the user's input data (construct a graph, bind to the condition peaks).

2) Create an API Endpoint to receive messages from the messenger.

3) Specify a set of commands and the API Endpoint in the messenger configuration at the moment of creating a bot to receive messenger queries.

4) Using the appropriate template, add a Message provider for a particular messenger at the level of business logic.

5) Receive messages by calling the Message provider from the API Endpoint.

6) Describe the behavioral logic of the Handler for each command in the format of the result of its fulfillment.

7) Define the State storage, and template its filing.

8) Connect a third-party server for text message recognition (with further development of your own), and get a set of commands from it.

9) Generate at the level of business logic a class of interaction with a third-party API (when fulfilling No. 8) by available methods, and supplement the behavior of Handlers.

10) Supplement calls of Handlers with further calls of the State storage by a call of the third-party service from No. 8, if necessary.

FORMAL LOGICAL SYSTEM AND INFERENCE METHOD

The formalism proposed in [3] will be used, on the basis of which a software system will be created, which will provide the solution to the problem. To create bots with useful behavior for practice, we will divide the stages of the action and implementation plans. This allows at the planning stage to communicate with the user to quickly and accurately select the next actions, and at the stage of plan implementation – control the conditions of previous actions, the appropriate reaction of the environment, redistribution of actions between participants, and other adjustments.

The second fundamental step is the division of the planning stage into stages of construction of detailed plans and scheme composition. We make plans at the first stage. In the second stage, a complex action plan is combined with simple (template) actions accumulated by the system.

Formalizations in the logic of action planning defined in terms of states and transitions considering the allocation of stages of construction of detailed plans and scheme composition are, on the one hand, objects (TAgent. TRealObject, TConception, TRelation, TValue, TProblem), situations (states) (TSituation), events (TEvent) and actions (TAction) (as well as relevant operations and relationships), on the other hand, plans (as well as relevant operations and relationships).

The state (situation) characterizes the state (situation) of the controlled object through its parameters. The implementation is a sequence of states and transitions that fulfills the conditions of initialization, the possibility of transitions, truth, and completion.

The formalism is designed to respond to queries that specify the initial and target situation (state), and involve the search or construction of a specific action plan, the implementation of which will ensure the transition of the controlled object from the initial situation (state) to the target one.

The main element of the plan is the sequence of actions. To the definition of the plan we enter the constructions of basic actions – the ml empty action, exit from the plan, conditional transition, cycle. Considering the role of the composition, the plan is defined as follows:

1) any action is a plan;

2) if $\pi 1,\pi 2$ – plans, then $x1,x2$ – plan;

3) if $\delta 1$ – action, $\sigma 1,\sigma n$ –situations, $\pi 1,..,xn$ – plans, then case $\delta 1,(\sigma 1,\pi 1),..,(\sigma n,..\pi n)$ – plan;

4) if $x1$ – plan, and $\sigma 1$ – situation, then while $(\sigma 1,x1)$ – plan.

To construct detailed plans, we will use the AP-system, and for the composition of plans – the PC-system, as proposed in [3]. Let's analyze the axiomatization features which are important for creating bots.

In the AP-system within the framework of a clause logic, based on the definitions of symbols, terms and formulas, as proposed in [3], we define:

Type abstracts:

1) formula for individuals – type abstract,

2) if A – formula for individuals, α – type abstract, then $\{\alpha A\}$ – type abstract,

3) no other type abstracts;

Definitors as constructions t def η, where t – individual term, η – term for types;

Specifiers: objects, situations, events t spec A, where t – individual term for objects, situations, events, or term for types, A – type abstract; actions t: $\langle\langle\beta 1\rangle,\langle\beta 2\rangle\rangle$, where t – individual term for action, or term for types, $\beta 1$ and $\beta 2$ – precondition and postcondition as situation specifiers; problems t: $\langle\langle\beta 1\rangle,\langle\beta 2\rangle\rangle$, where t – individual term for TProblem object, or term for types, $\beta 1$ and $\beta 2$ –precondition and postcondition as situation specifiers;

Clauses: expressions A → K, where A – a sequence of atomic formulas of the language of the AP-system, K – one atomic formula.

The knowledge base of the AP-system consists of:

1) A set of specifiers of the description of the initial situation and types;

2) A set of clauses specifying the features of operations and predicates for individual objects and types;

3) A set of definitors and clauses defining the type features;

4) A set of clauses specifying the preconditions and postconditions of actions;

5) A set of clauses specifying the event component of preconditions.

In the definition of the AP-system within the framework of the clause logic of the first order, based on the definitions of symbols, terms and formulas, as proposed in [3], we define:

Action programs:

1) if t – individual term of the action type, then t – action program;

2) if $y1,y2$ – action programs, then $y1,y2$ – action program;

3) if $l1$ – individual term of the action type, $\varepsilon 1,..\varepsilon n$ – formulas of the situation type $y1,..yn$ – action programs, then case $(l1,(\varepsilon 1,y1),..,(\varepsilon n,yn))$ – action program;

4) if $y1$ – action program, and c1 – formula of the situation type, then while $(y1,\varepsilon 1)$ – action program;

5) no other action programs;

Action program specifiers:

1) if $y1$ – scheme term, $y2$ – action program, then $y1,y2$ – action program specifier;

2) no other action program specifiers;

Clauses: expressions A→K, where A – a sequence of atomic formulas of the language of the PC-system, K – one atomic formula.

The knowledge base of the PC-system consists of:

1) A set of action program specifiers;

2) A set of clauses defining problem-solving schemes;

3) A set of clauses defining scheme features.

The inference procedure will be shown on the example of a natural case where queries take the form $\langle\langle\lambda 1\rangle,\langle\lambda 2\rangle\rangle$, where $\lambda 1,\lambda 2$ – abstracts of the initial and target situations. The inference procedure includes the following steps:

1) Establishing an individual situation or a situation type in the AP-system (if the query defines individual terms of the initial and target situations);

2) Establishing an action program in the PC-system (if the query defines abstracts of the initial and target situations);

3) Establishing an action program in the PC-system (if the query defines types of the initial and target situations);

4) If necessary, redefying an action program in the PC-system;

5) If necessary, constructing an action program in the PC-system;

6) If necessary, establishing the necessary type transformations.

To reduce the search, programs for popular queries are saved, and known techniques are used, first of all, the standardization of statements, the method of analogies. To substantiate the accuracy of the obtained output results, the theorems on effective achievability can be used, as proved in [3].

CONCLUSION

Based on the analysis of messengers, chatbots, problems of their creation, use, and development, an approach to the automated creation of bots based on logical and linguistic models has been proposed. The problems of formalization of processes creating bots have been analyzed. One of the key features of this solution is the automation of an action chain, the implementation of which leads to the fulfillment of the user's query.

The logical formalism has been described as the basis of this solution, the formal definition of the aim achievement and conditions, to which the action plans providing the aim achievement should correspond, has been offered. The inference procedure has been proposed, which allows us to form action plans to solve problems formulated as pairs of initial and final states.

The implementation of the proposed solution will allow us to rapidly create bots for a wide range of problems, formulated in terms of pairs of initial and final states, based on a combination of functions of the appropriate classes of objects, also given by pair of initial and final states, to form the appropriate action plan. Thus, the bots can be rapidly created to obtain the necessary information, place orders, provide access to educational resources, etc.

Further research should be conducted to create and study bots in order to solve certain classes of problems, create technologies for efficient natural communication of a bot with the user, create the information technology to integrate the user with all messengers, social networks and information systems.

REFERENCES

[1] Radziwill N. Evaluating Quality of Chatbots and Intelligent Conversational Agents [Electronic Source] / N. Radziwill, M. Benton. – 2017. – Access: https://arxiv.org/ftp/arxiv/papers/1704/1704.04579.pdf

[2] Most popular global mobile messenger apps [Electronic Source]. – 2018. – Access: https://www.statista.com/statistics/258749/most-popular-globalmobile-messenger-apps/

[3] The approach to applications integration for World Data Center interdisciplinary scientific investigations / Grzegorz Nowakowski, Sergii Telenyk, Kostiantyn Yefremov, Volodymyr Khmeliuk // W: Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, September 1–4, 2019, Leipzig, Germany [online] / eds. Maria Ganzha, Leszek Maciaszek, Marcin Paprzycki. – Warszawa : Polskie Towarzystwo Informatyczne, 2019. – (Annals of Computer Science and Information Systems, ISSN 2300-5963 ; 18). – S. 539-545. – doi: 10.15439/2019F71. – ISBN 978-83-952357-8-8 (Web)

[4] Sergii Telenyk, Nowakowski Grzegorz, Zharikov Eduard, Vovk Yewhenii. Conceptual foundations of the use of formal models and methods for the rapid creation of web-applications // The 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. 18-21 September, 2019, Metz, France

ABSTRACTS

INFORMATION SYSTEMS AND TECHNOLOGIES

Page 9

Automatic database query generation system based on asynchronous operations

Valery Tiurin
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
tiurinvalery@gmail.com

Olena Savchuk
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
savchuk_l1@ukr.net

Abstract. **The work is devoted to the development of an automatic database query generation system based on asynchronous operations. An open source library in a non-relational database and an integrated application for its testing had been developed.**

Keywords**:** ***automatic system, database, asynchronous operations.***

Page 11

The implementation of RESTful API for social media events platform

Bodak Bohdan
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
bohdan.bodak@outlook.com

Doroshenko Anatoliy
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
a-y-doroshenko@urk.net

Abstract. **Nowadays, there are many great applications for local search and recommendations, for example Foursquare, Swarm, and Snapchat to some extent. Swarm is a mobile application, which allows users to share their locations with friends and create a record of their experiences in their personal logbook. In addition, a spin-off from and companion app to the older Foursquare, Swarm lets users to check-in to a particular location and interact with people nearby. Besides, these check-ins are listed chronologically in order to create the person's logbook, which in fact represents a digital library. However, many people agree that any social media platform requires a well-designed, scalable, and fast API. In this research, we used RESTful API principles to design a heavily loaded server for the new application. This paper mostly focuses on applying REST and OpenAPI standards using a social media platform as an example. The new specifications and methodologies outlined in this article may be used in the design phase of a heavily loaded backend service.**

Keywords: Social media, local search, OpenAPI, REST, RESTful, scalable API, software architecture.

On reability simulating and evaluating in cloud services system

Veronika Niechkina
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
ni2403kalos@gmail.com

Abstract. **The criterion for smoothing random signal changes using a buffer in real-time systems is developed. The problem of the occurrence of rapid spasmodic changes in temperature during the operation of the system for fixing changes in the liquid level in the tank is considered. In the process of solving the problem, a buffering criterion was introduced. Signal changes were written to the buffer.**

***Keywords: control system, buffering, delay, extrapolation*.**

Remote installation error management

Andrew Telenyk
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine

Abstract. **The purpose of the article is to review available means of solving the problem of centralized installation of software in computer networks, to clarify the aspects of their deployment, and to describe author's vision for solving the problem. The best solution to the problem will be the implementation of error-fixing utilities in the projectant remote software installation system based on Dijkstra's algorythm, changed accordingly to task, which will allow easy installation of software within the computer network.**

***Keywords: system resources; installation error; version compatibility; installation packages; Dijkstra's algorythm*.**

INFORMATION PROCESSING IN COMPLEX SYSTEMS

Methods of Optimizing the Diagnostic Data Processing for Information Management Systems

Olena Syrotkina
Dnipro University of Technology
Dnipro, Ukraine

Mykhailo Aleksieiev
Dnipro University of Technology
Dnipro, Ukraine

Iryna Udovyk
Dnipro University of Technology
Dnipro, Ukraine

Abstract. **This paper addresses the issue of creating and applying mathematical methods for optimizing time and computing resources when processing multiple data streams circulating in a distributed information management system. In order to solve this problem, we suggest methods of reducing the space of analyzed states using the data organization structure "m-tuples based on ordered sets of arbitrary cardinality". Using this data structure allows us to minimize the time and computing resources involved.**

Keywords: Big Data, Big Data reduction methods, data organization structure, ordered set of arbitrary cardinality, minimization of time and computing resources.

PROGRAMMING TECHNOLOGIES

Page 23

Methodology of efficiently realizing single-page application

Vitalii Drabynko
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
v.drabynko@gmail.com

Anatoliy Doroshenko
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
a-y-doroshenko@ukr.net

Abstract. **The paper demonstrates the result of the study of the effectiveness of frameworks and JavaScript libraries in the implementation of single-page web applications. Tools such as React, Vue.js and Angular were considered. Web applications were developed using these tools and their effectiveness was evaluated.**

Keywords: single-page web application, Angular, React, Vue.js, framework.

Page 25

Centralized management of music services

Bohomol Roman
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine

Abstract. **The purpose of the article is to review the available ways to solve the problem, highlight their advantages and disadvantages and model an architectural solution that combines all the advantages, and will not contain the disadvantages of the reviewed analogues. The best solution for the implementation of this task is a software product created on the basis of the client - the server architecture of the server part, which is implemented using a microservice approach.**

Keywords: music services, client - server architecture, microservices, music content management.

Page 29

Testing system based on relative evaluation of statistical samples

Kovalenko Ihor
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine

Vovchok Eugene
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine

Abstract. **The purpose of the article is to review the available ways to solve the problem of assessing the quality of knowledge by testing and to model an architectural solution that combines all the advantages and will not contain the disadvantages of the examined analogues. The optimal solution within the criteria specified in the article is a software product created on the basis of client - server architecture in which the server component is implemented as a single architectural unit, to ensure ease of deployment and reduce the cost of system support.**

Keywords: testing systems; establishing the level of knowledge; statistical sample; client-server architecture.

Page 33

Bot creating technology for wide use based on a logical approach

Telenyk Sergii
Igor Sikorsky Kyiv Politechnic Instiitute
Kyiv, Ukraine
Cracow University of Technology
Cracow, Poland

Zharikov Eduard
Igor Sikorsky Kyiv Politechnic Instiitute
Kyiv, Ukraine

Vovk Yevhenii
Igor Sikorsky Kyiv Politechnic Instiitute
Kyiv, Ukraine

Nowakowski Grzegorz
Cracow University of Technology
Cracow, Poland

Tokmenko Olena
Taras Shevchenko National University
Kyiv, Ukraine

Abstract. **The report proposes the information technology for computer-aided design and implementation of bots based on a logical approach. The general approach to creating bots with the use of a complex of logical and linguistic models has been described. The general scheme of work of the web-based system with the use of a chatbot has been developed. The model of logic of actions, and the inference method have been described, which can be used for the formal description of the bot creating process. An example of using a logical model to create a bot has been described.**

***Keywords: information technology, messangers, bot, automated bot creating, logic of actions*.**