ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 6, Num. 2, 2021

INVESTIGATION OF SERVERLESS ARCHITECTURE

Vladyslav Lakhai, Ruslan Bachynskyy

Lviv Polytechnic National University, 12, Bandera Str, Lviv, 79013, Ukraine.
Authors' e-mail: *vlad.luckhi@gmail.com, bac_ruslan@ukr.net*

https://doi.org/10.23939/acps2021.__.___
Submitted on 01.05.2021

***Abstract*: Serverless computing is a new and still evolving type of cloud computing, which brings a new approach to the development of information systems. The main idea of serverless is to give an approach of doing computing without dealing with a server to a user. Such approach allows to reduce the cost of the system building and system support. It allows small companies to concentrate on their own system designing instead of thinking about infrastructure building and supporting. Also, a big problem of providing the system security on high level is on cloud's provider engineering support service. Serverless approach allows to start business quickly without huge initial investment. There is an attempt to completely analyze features, benefits and drawbacks of serverless approach, its use cases and main patterns of Serverless architecture. What is more, different providers have been analyzed.**

***Index Terms*: serverless, cloud computing, architecture patterns, information systems development, AWS**

I. INTRODUCTION

High level of information technologies distribution and stable interest in their use led to increasing difficulty for individuals and organizations to keep their computing in-house (on their own servers). That is the main reason of cloud computing rapid growth.

Cloud computing refers to delivering on-demand computing services, originally storage, and now more recently processing power and apps, over the internet, with companies using this on a pay-as-you-go basis. It relies on sharing of resources to achieve coherence and economies of scale. Main advantages include cost savings, increasing productivity, speed, efficiency, performance and security.

Cloud computing is not a single piece of technology. There are four traditional types (models) [1]:

IaaS (Infrastructure as a Service)
PaaS (Platform as a Service)
Serverless
SaaS (Software as a Service).

IaaS includes all basic infrastructure components for information systems development. It gives direct access to network resources and virtual computers. This model has the highest level of flexibility. PaaS does not require administration of basic infrastructure. In most cases, it represents a platform for creation of auto-scalable applications. IaaS and PaaS allowed not to think about any hardware, but there were still a lot of things which clients were administrating themselves.

Serverless is a cloud computing model in which client can operate only with code and data. Cloud vendor is providing and administrating all needed hardware and software. SaaS is a model in which client can operate only with data. Serverless and SaaS allow clients to use exactly what they need without thinking about underlying hardware and software.

Serverless architecture is an approach to design and develop information systems [2] using components of serverless and SaaS cloud computing models [3].

II. THE RATIONALE OF THE NEED FOR SERVERLESS ARCHITECTURE

Five years ago, at the start of serverless era, most of technology adopters were startups who were seeking for a possibility to scale up and lower the finance entrance barrier.

Therefore, serverless architecture is extremely good in rapid prototyping. However, are there any benefits for long-run development? Yes, but not for every individual or organization.

Nowadays, even big enterprises start using serverless architecture. It is suitable to run stateless applications, such as event-driven functionality, batch jobs or data transfer. So, the main serverless architecture use cases are:

- High-traffic information systems. With serverless, you can make your system high available and scalable. As a plus, it is often much cheaper and easier in comparison to traditional architecture.
- Storing huge amounts of data. If there is a need to store huge amount of data and work with it in non-blocking way – serverless is one of the best solutions. For example, Amazon DynamoDB can handle more than 10 trillion requests per day or 20 million per second.
- Internet of Things (IoT). The real-time response nature of the serverless approach works great for IoT use cases. IoT devices generate a lot of data from

their environments through sensors and there is a necessity to process this data in scalable way.

- Prototypes. Serverless is the best approach for making proof-of-concepts in most of fields.

III. FORMULATION OF THE PROBLEM

As serverless is relatively fresh and rapidly evolving approach with many interesting and useful features, it is a popular area for investigations [4]. There are a lot of new methods and instruments. In addition, main cloud providers positively affect development of serverless architecture.

There are many papers in this field, separately describing core concepts, main services, architecture patterns or providers' comparison [5 ,6]. However, there is a lack of complete analysis of this approach.

IV. ADVANTAGES AND DRAWBACKS OF SERVERLESS ARCHITECTURE

Like any other technology, serverless computing has its advantages and drawbacks. Some of them were inherited from event-driven architecture (EDA), which is a basis for serverless architecture.

Main advantages of serverless are:

- Reduced time-to-market. Developers can focus their attention on product development. The vendor handles components like network configuration or the physical security of your servers. As a result, development process was simplified which led to reducing time-to-market.
- Lower costs. Serverless approach saves time and resources in two ways. First, serverless is usually about pay-as-you-go pay model. That means that you are charging for resources, which were really used. Idle time is not billed. Second, you are outsourcing the responsibilities of managing servers, databases, and some logic. Besides the actual cost, serverless takes less computing power and human resources.
- Increased flexibility of scaling. With serverless, you break down applications into smaller and smaller pieces, known as decomposition. In addition, you are using EDA, which means that parts of your system are loose coupled and as a result independent. So, this gives an ability to scale them automatically and endlessly.

Main drawbacks:

- Vendor lock-in. Serverless architecture requires you to be reliant on your provider. You do not have full control, and changes may affect you without notice. In addition, it is hard to change your provider. There are many differences in services with similar functionality from two cloud providers.
- Increased security risks. As serverless is about decomposition and multiple independent parts of system, it leads to a larger attack plain.
- Learning curve. Working with serverless architecture requires some additional knowledge and skills.

V. COMPARATIVE ANALYSIS OF CLOUD PROVIDERS

Today, there are many cloud providers. The main are the following: Amazon Web Services (AWS), Google Cloud Platform (GCP) and Microsoft Azure (Azure).

Comparative analysis of cloud providers includes analysis of Gartner (global research and advisory firm) cloud providers' investigation report and comparison of relative search volume.

Gartner is making investigation of cloud providers market on regular basis. One of the main features of this investigation is forming of "magic quadrant" - graphic comparison of cloud providers by two criteria: completeness of vision and ability to execute.

There are four sections in this quadrant:

- Leaders. They execute well against their current vision and are well positioned for tomorrow.
- Visionaries. They understand where the market is going, but do not execute well now.
- Niche Players. They are focused on a small segment and have there some success or unfocused and do not outperform others.
- Challengers. They execute well against their current vision or successfully focused on a large segment, but are bad positioned for tomorrow.

Gartner cloud providers magic quadrant is shown in Fig. 1.

Fig. 1. Gartner cloud provider's magic quadrant

Using Gartner cloud providers "magic quadrant" from research by 2020 (Fig. 1.), we can make next conclusions:

- Amazon Web Services is a leader in both criteria.
- Microsoft Azure takes second place.
- Google Cloud Platform takes third place.
- There are no visionaries or challengers.
- All other cloud providers are in niche players section which means that they are successfully

focused on a small segment or unfocused and do not outperform leaders.

Google Trends is a service by Google for search analysis. It gives an ability to compare and analyze the popularity of different search queries in Google Search. Google Trends comparison is shown in Fig. 2.

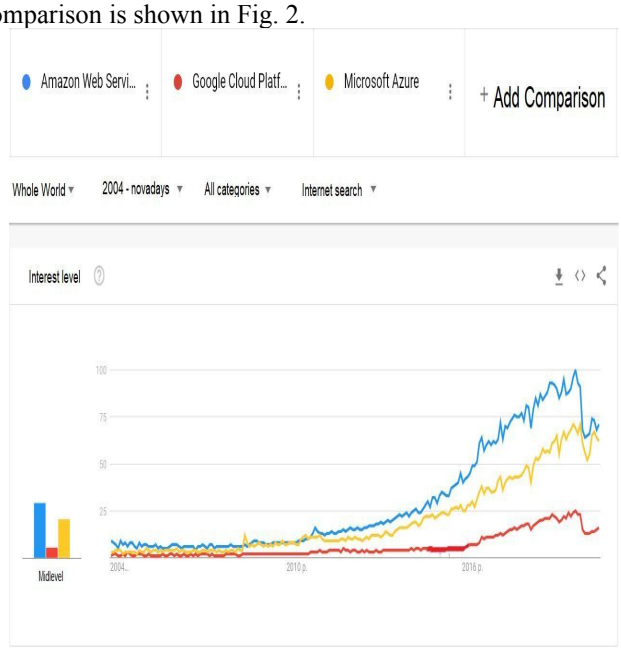

Fig. 2. Google Trends comparison

Using Google Trends comparison of cloud providers search queries popularity, we can make next conclusions:

- Amazon Web Services is a leader in search frequency.
- Microsoft Azure takes second place.
- Google Cloud Platform takes third place and has a big lag from AWS and Azure.

As a pioneer in field of cloud computing, AWS had enough time to form a complete vision on evolution of cloud technologies. Amazon had more than enough power and resources to implement this vision. For now, it takes first place in most of cloud providers' comparisons and provide the widest number of available services.

VI. COMPONENTS OF SERVERLESS ARCHITECTURE

In this paper, serverless architecture will be investigated in conjunction with Amazon Web Services.

AWS divides its serverless services into three categories [4]:

- Compute
- Application integration
- Data store

Compute category represents services that provide computing resources. AWS refers Amazon Fargate to serverless computing, but this service will not be overviewed in paper because of its CaaS (Container as a Service) nature.

Main model of serverless computing for many years is FaaS (Function as a Service) and in AWS there is implementation of this model – AWS Lambda.

AWS Lambda is a serverless computing service that allows clients to run code with zero administration (without provisioning or managing infrastructure). Lambda scales automatically to each event and natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code. AWS Glue architecture icon is shown in Fig. 3.

AWS Glue is a serverless data integration tool for creating, running and monitoring ETL (Extract, transform, load) workflows for data engineering, analytics and machine learning. It provides both visual and code-base interfaces. With code, you can run Python, Spark or PySpark environments. AWS Glue automates much of the effort required for data engineering and supports flexible scaling. AWS Glue architecture icon is shown in Fig. 3.

Main integration services are Amazon API Gateway, Amazon SQS, Amazon SNS, Amazon Cognito and Amazon CloudFront.

Amazon CloudFront is a fast content delivery network service for delivering data, videos, applications and APIs to customers. CloudFront provides low latency, high level of secure and transfer speeds. It has deep integration with AWS and more than 225 points of presence all over the world for ultra-low latency. Amazon Cognito architecture icon is shown in Fig. 3.

Fig. 3. AWS architecture icons (Lambda, Glue, Cognito)

Amazon API Gateway – is a service for creating, publishing, maintaining, monitoring and securing APIs, including RESTful. API Gateway provides these at any scale and with low latency. 6. Amazon API Gateway architecture icon is shown in Fig. 4.

Amazon SQS (Simple Queue Service) is a message queuing service for publishing, storing and receiving messages at any volume. It helps to decouple and scale serverless applications, microservices and distributed systems. Amazon SQS architecture icon is shown in Fig. 4.

Amazon SNS (Simple Notification Service) is a messaging push-based many-to-many service for both A2A (Application to application) and A2P (Application to person) communication. Key units are topic, publisher and subscriber. Possible subscribers: SQS, Lambda, HTTPS endpoint, Kinesis, email, SMS, mobile push and many others. Amazon SNS architecture icon is shown in Fig. 4.

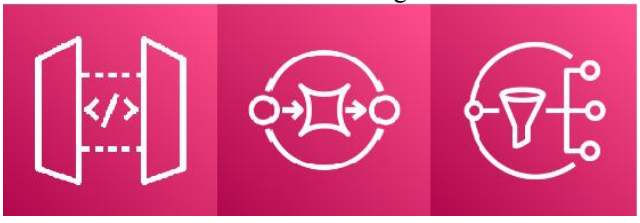

Fig. 4. AWS architecture icons (API Gateway, SQS, SNS)

Amazon Cognito is a service for users' sign-up, sign-in and access control to AWS resources. Service scales to millions of users and supports sign-in with social identity providers. Amazon Cognito architecture icon is shown in Fig. 5.

Amazon S3 (Simple Storage Service) is a service for storing and protecting any amount of data (objects). It provides industry-leading security, performance and durability level. Amazon S3 architecture icon is shown in Fig. 5.

Amazon DynamoDB – "key-value" and document database. It provides extremely high performance, durability and security. In addition, database can handle more than 10 trillion requests per day. DynamoDB is automatically scalable and serverless. Amazon DynamoDB architecture icon is shown in Fig. 5.

Amazon Aurora Serverless – auto-scaling configuration for Amazon Aurora that enables to run database in the cloud without managing any database capacity. Aurora is a MySQL and PostgreSQL-compatible relational database. It is five times faster than standard MySQL and three than PostgreSQL. Amazon Aurora architecture icon is shown in Fig. 5.

Fig 5. AWS architecture icons (Cognito, S3, DynamoDB, Aurora)

VII. SERVERLESS ARCHITECTURE PATTERNS

Architecture pattern is a solution, which can be reusable for solving widespread architectural problems.

Web-application is the most popular use-case of serverless architecture and it is the reason to overview serverless architecture patterns for web-applications development.

Interface is the key component of every app. Interface objects (html, css, js files and other multimedia) can be stored and accessed using Amazon S3. For providing low-latency it is recommended to use Amazon CloudFront. So, simple web-application consists of those two components. Architecture schema of simple web-application is shown in Fig. 6.

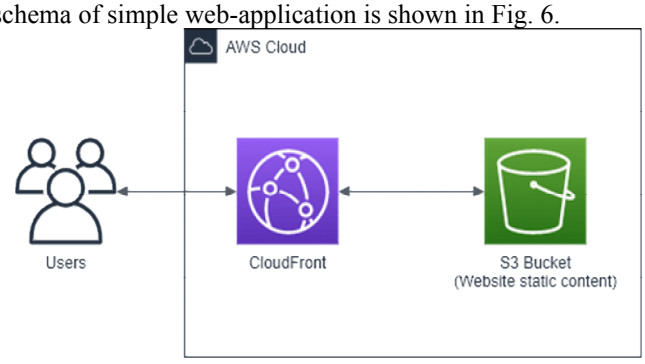

Fig. 6. Architecture schema of simple web-application

For most of modern apps it is not enough to provide only interface – they need communication between the client and application business-logic. Usually, this communication is provided by API requests and Amazon API Gateway can deal with it. As business-logic runner, it's better to use AWS Lambda. We should remember that our code is stateless and we should provide access to stored data. As database in examples, we will use Amazon DynamoDB. So, combining these five components, we can get nearly standard web-application. Architecture schema of standard web-application is shown in Fig. 7.

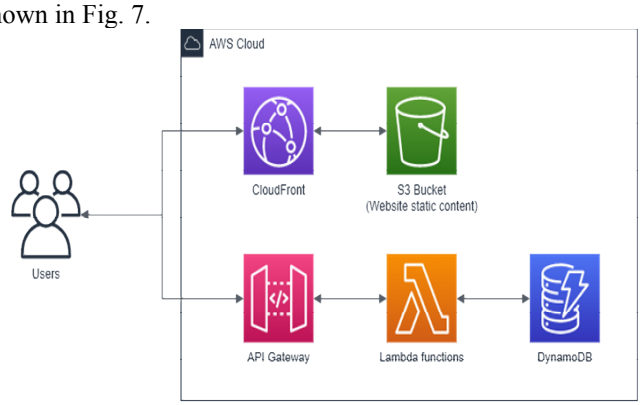

Fig. 7. Architecture schema of standard web-application

The last necessary for most web-application thing is access control. Amazon Cognito can provide it. This is a service for users' sign-in, sign-up and control access. It supports sign-in with social identity providers. Architecture schema of web-application Architecture schema of web-application is shown in Fig. 8.

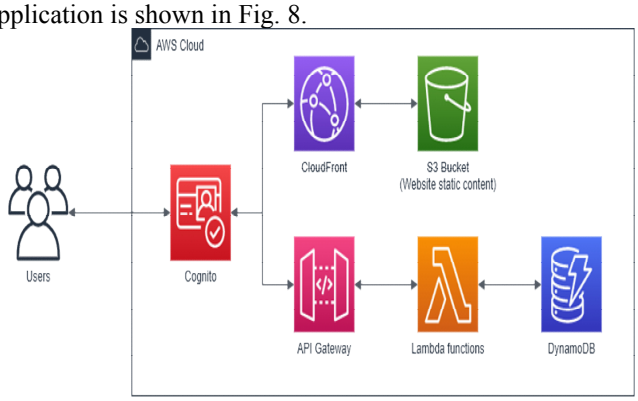

Fig. 8. Architecture schema of web-application

VIII. SERVERLESS ARCHITECTURE DESIGN OF DEMONSTRATION SYSTEM

Serverless architecture is an excellent choice for startups. They are seeking for a possibility to scale up and lower the finance entrance barrier and serverless approach can provide all of it.

Therefore, it will be justified to choose startup-like system as a demonstration system to design. It is a geolocation system-service for family groups codenamed "Luckhi family" [7]. Service main goal is to make people confident that everything is ok with their relatives. It is a bit similar to "Weasley Clock" from Harry Potter. Customers are able to see information about their relatives' geolocation in real time to keep them on track.

First step of designing architecture is to perform functional requirements analysis of service prototype (or

MVP, most viable product). To do this, we need to define basic functional.

This service should be able to handle next actions:

- Registration of user
- Authorization of user
- Creation of family groups
- Joining already created group
- Editing family zones
- Sending geolocation from Owntracks app on users' mobile device
- Getting geolocation of all family group members in appropriate form

After defining basic functional, we can form functional requirements to designed system:

- Low development cost. In most cases, start-ups are limited with money amount they can spend on system prototype (MVP or most valued product) development.
- High scalability. For start-ups, it is important to have ability to scale as fast as their customers' amount grow.
- High response speed. Complex customer requests also should be handled fast. There is no need for customers to know how complex some operations are, but they definitely want to get result fast.
- Ability to integrate with other service. For example, service is positioning itself as a platform that can connect with any third-party geolocation provider. In this demonstration system, we will use Owntracks application as a geolocation provider mainly because of its economic battery consumption.

Next step is creation of general architecture. In case of cloud-native services, it consists of choosing cloud provider and main services.

For demonstration system Amazon Web Services cloud provider will be used. AWS is a big player at cloud providers market and it provides the widest range of services that support serverless approach.

According to functional requirements analysis we can choose main services. As a compute service for running business-logic a good variant is AWS Lambda. As a database service – Amazon DynamoDB, automatically scalable NoSQL database. As a service for communication by REST API between client and "server" – Amazon API Gateway.

As there are some third-party services to integrate with, it is not the best solution to use Amazon Cognito. As an alternative, web-clients authentication can be handled with JWT tokens [8]. For correct identification of requests provided by third-party services, a good practice is to use separate user-manageable secret token.

In previous parts of this paper, it was mentioned that using Amazon Cloud Front and Amazon S3 for frontend content storing and accessing is considered a best practice. Therefore, it is justified to include this solution in our architecture. Architecture schema of demonstration system is shown in the Fig. 9.

All this components-services support serverless approach complies with previously defined functional requirements. Shema of designed general architecture:

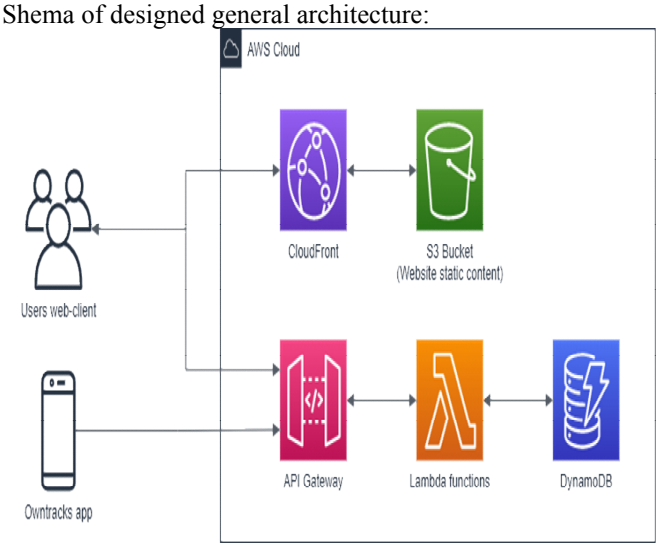

Fig. 9. Architecture schema of demonstration system

Another possible part of architecture design is database design. It is not popular at all, but we have high response speed requirement. Without correct database design, there are some difficulties to fully satisfy this requirement. We are using NoSQL database and it is important to remember that there is no optimal JOIN operation support in such databases. It means that stored data should be normalized so minimally as it is possible. According to all this conditions and service functionality, it seems justified to have three tables:

- Users. This table is for storing full data about user, including data required for authentication.
- Families. This table should not only store general information about families, but also main information of its members (name), family zones information and updated in real time location information (location and its update timestamp). Main idea is to make complex in most cases getting operation as fast as possible to raise response speed.
- Locations. Separate table for storing only location information (user, location, timestamp). In contrast to table Families, which have only "present" location data, table Locations stores it in a historical way. It will be useful for future functionality like data analytics, processing or even applying machine learning algorithms [9 ,10].

Shema of database design is shown in Fig. 10.

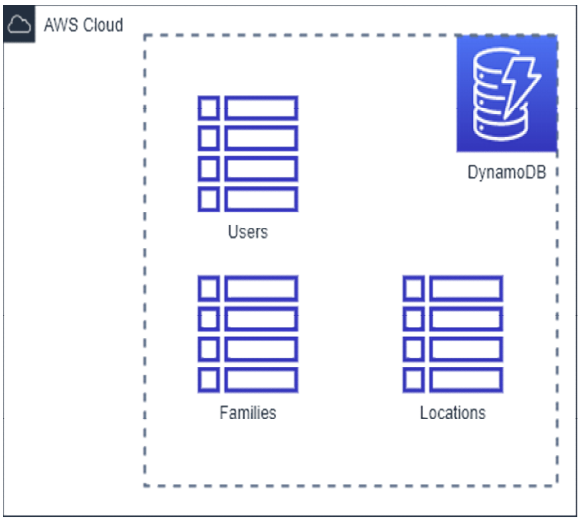

Fig. 10. Schema of database design

IX. CONCLUSIONS

Provided analysis showed that serverless approach is changing all we know about information systems architecture. For now, in most cases there is no need to provide and maintain infrastructure by ourselves. We can fully outsource it to cloud providers and focus on important and valuable things like developing business-logic. Serverless architecture resembles Lego constructor – to get a result you should just combine components-services, your business-logic and data.

To be honest, there are some limitations of this approach. Some of them, for example as inability of serverless services to perform well at long compute-intensive tasks, were successfully overcome with approach evolution (AWS Glue serverless ETL service).

It is a relatively fresh field of cloud computing and there is definitely some space for improvements. This approach is widely used both in startups and enterprises. It helps to save costs, simplify development process and forget about problems with scalability. For sure, it is not a silver bullet and it is unjustified to use this approach literally in all cases, but it will definitely be a part of our future.

References

[1] https://aws.amazon.com/types-of-cloud-computing/

[2] https://aws.amazon.com/serverless/

[3] Li, L., Ge, R., Zhou, S. and Valerdi, R. (2012). Guest Editorial Integrated Healthcare Information Systems. In *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 4, pp. 515-517. DOI: 10.1109/TITB.2012.2198317.

[4] Shen, J., Zhou, T., He, D., Zhang, Y., Sun, X. and Xiang, Y. (2019). Block Design-Based Key Agreement for Group Data Sharing. In Cloud Computing in *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 996-1010. DOI: 10.1109/TDSC.2017.2725953.

[5] Ramasubramaniam, K. S., Annamalai, G. and Krishna, A. (2015). System architecture patterns a domain-based proposition. *International Symposium on ConsumerElectronics (ISCE)*, pp. 1-2. Available at: https://ur.booksc.eu.

[6] Wang, Q., Ma, H., Ke, Q., Wang, C. and Wang, X. (2009). Spatial Analysis of Regional Sustainable Development Based on Geographic Information System and Relative Carrying Capacity of Resources. *International Conference on Environmental Science and Information Application Technology*, pp. 437-440. Available at: https://doi.org/10.3390/su12093923.

[7] Sudharsan, D., Adinarayana, J. and Tripathy, A. K. (2009). Geo-information Services to Rural Extension Community for Rural Development Planning – A Framewor. *International Conference on Advanced Geographic Information Systems & Web Services*, pp. 54-59. DOI 10.1109/GEOWS.2009.9.

[8] Wang, Q. Z. and Liu, J. (2006). Project Uncertainty, Management Practice and Project Performance: An Empirical Analysis on Customized Information Systems Development Projects. *International Engineering Management Conference*, pp. 341-345. DOI: 10.1109/IEMC.2006.4279882.

[9] Satyanarayana, G., Bhuvana, J. and Balamurugan, M. (2020). Sentimental Analysis on voice using AWS Comprehend, *International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1-4. DOI: 10.1109/ICCCI48352.2020.9104105

[10] Massa, D. and Evans, N. R. (2008). The angular separation of the components of the Cepheid AW Per. In *Monthly Notices of the Royal Astronomical Society*, vol. 383, no. 1, pp. 139-149. Available at: https://doi.org/10.1111/j.1365-2966.2007.12520.x

Ruslan Bachynskyy obtained his Ph.D. degree in Computer systems and components at Lviv Polytechnic National University in 2008. He is interested in Embedded systems, Digital Signal Processing, FPGA-based designing.

Vladyslav Lakhai is a fourth-year computer engineering student of Lviv Polytechnic National University. He has production experience in the following fields: Clouds, Serverless and Data Engineering. Interested in IoT and Data Sciences.