Quest Journals
Journal of Software Engineering and Simulation
Volume 8 ~ Issue 12 (2022) pp: 05-17
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

Measuring the performance of the electronic exam registration system on the server and server less architecture

M.Murić [1] , V.Mladenović[2]
[1] *Academy of applied sciences Western Serbia, 31000 Užice, Serbia*
[2] *Faculty of Technical Sciences, 32000 Čačak,Serbia*

Abstract: The paper analyzes the performance of the system for electronic registration of exams that will be used at the Academy of applied sciences Western Serbiain Užice in 2022. Such systems are used increasingly, and it is important to ensure their efficient and safe operation. System performance measurement was performed on the server architecture, which is located on the premises of the Academy, and on the serverless architecture, using the AWS system. The purpose of the work is to show the advantages and disadvantages of both architectures through the obtained results, which would enable a proper choice of hosting for systems of this type while achieving maximum efficiency and minimum costs.

Keywords: *server; serverless;e-learning; AWS; FaaS; RESTApi;*

R*eceived 04 Dec., 2022;* ***R****evised 14 Dec., 2022;* ***A****ccepted 16 Dec., 2022 © The author(s) 2022.*
P*ublished with open access at* ***www.questjournals.org***

I. INTRODUCTION

E-learning is a term that has existed for quite a long time (in England since the middle of the 19th century, and in the USA since the end of the 19th century at the university level) and one of the definitions that can be used to explain this term is "that it is learning with the help of a computer and the Internet with the use of information and communication technologies". This type of learning has its advantages and disadvantages, which became evident as this method of education was used more and more. This type of education does not have to be exclusively online, but blending learning methods are also used, which are partly applied online and partly in a traditional way. [1][2].

Along with the development and increasing application of this type of education, the idea of electronic applications for exams also appeared [3][4][5].

Electronic exam registration systems brought several challenges that had to be overcome for such systems to be reliable and efficient. Today, there are a variety of different solutions, both in terms of design and options, including ways to protect data and authenticate users. [6][7][8].

Following the trends in the development of higher education, the Academy of Vocational Studies in Western Serbia decided to include the electronic application of exams in its offer. This idea existed for a long time, but its realization was accelerated by the coronavirus pandemic and the obvious advantages of this way of applying for the exam. The basic question at the very beginning of the implementation of the system was whether to use ready-made solutions and rely on others or implement such a system relying exclusively on its resources. The decision to rely on our staff prevailed, and the electronic application system will be tested at the Academy of applied sciences Western Serbia, Užice branch, and will be used from January 2022. This system is completely authentic and we can say that it fully satisfied the needs of both students and employees at the Academy.

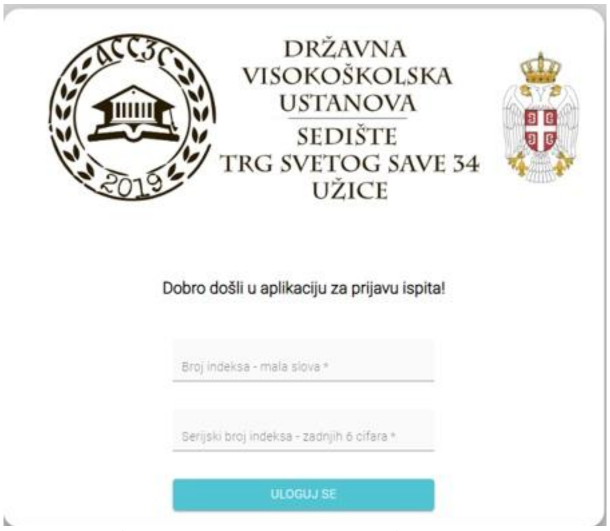

DRŽAVNA
VISOKOŠKOLSKA
USTANOVA
SEDIŠTE
TRG SVETOG SAVE 34
UŽICE
Dobro došli u aplikaciju za prijavu ispita!
Broj indeksa - mala slova *
Serijski broj indeksa - zadnjih 6 cifara *
ULOGUJ SE

Figure 1 - Home page of the system for electronic application for exams at the Academy of applied sciences Western Serbia, Užice department

II. CHARACTERISTICS OF THE SERVER ARCHITECTURE OF THE ACADEMY OF WESTERN SERBIA

The website of the Business and Technical College in Užice, later the College of Vocational Studies, and now the Academy of applied sciences Western Serbia is located on a server located in the premises of the Academy and has been in existence since 2011. The reasons for such a solution are numerous, and above all, they relate to relying on one's resources in the development and implementation of the necessary solutions, as well as to the fulfillment of the requirements in terms of security and data protection. The characteristics of the server are as follows:

- Processor INTEL XEON XP Proliant DL120G7
- 4 GB of RAM
- 2 HDDs of 250 GB each
- Gigabit network

Following the development of higher education and the Academy, which has existed since 2019, the site has had manydesigns and functionality revisions. The current appearance of the websites of the Užice department and the Academy is shown in the following images.

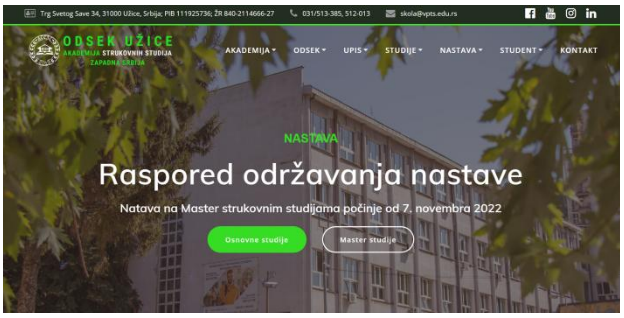

ODSEK UŽICE
AKADEMIJA STRUKOVNIH STUDIJA ZAPADNA SRBIJA
AKADEMIJA
ODSEK
UPIS
STUDIJE
NASTAVA
STUDENT
KONTAKT
NASTAVA
Raspored održavanja nastave
Natava na Master strukovnim studijama počinje od 7. novembra 2022
Osnovne studije
Master studije

Figure 2 - Website of the Academy of Western Serbia, Užice department [9]

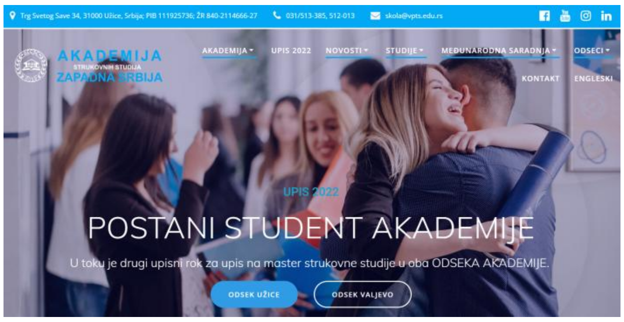
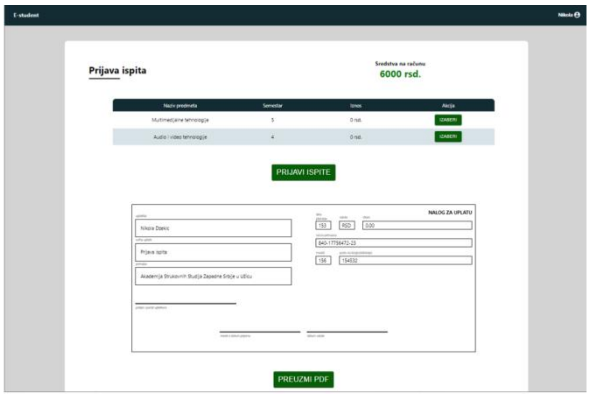


Figure 3 – Website of the Academy of Western Serbia [10]

In 2020, the implementation of the system for electronic registration of exams began in January 2022. The system was implemented using the following technologies: HTML, CSS, React.js, Node.js, Prisma, PostgresSQL, and Heroku. The project is divided into two parts: an interface for users and an interface for administrators. The app is easy to use, functional, has an attractive school-appropriate design, and supports scalability to allow its users to use different devices. The selected software solution is upgradeable.

The CRUD application has an interface for the user (student) and an interface for the admin (student service worker). All paths are protected by checking "cookies" (only a logged-in user can access the user interface related to the exam registration) as well as paths to which only the administrator has access. The web application has a responsive design for the different dimensions of the devices on which it can be executed. The following application functionalities are provided:

1. the student logs into the application uniquely,
2. the logged-in user can select exams from the appropriate list of exams according to his current status (year of study, unpassed exams),
3. the user completes the exam application if he has the funds for it, for which he receives a confirmation,
4. the administrator can add "credits" to users, or register the exam instead of the user.



E-student Nikola

Prijava ispita

Sredstva na računu
6000 rsd.

Naziv predmeta	Semestar	Iznos	Akcija
Multimedijalne tehnologije	5	0 rsd.	IZABERI
Audio / video tehnologije	4	0 rsd.	IZABERI

PRIJAVI ISPITE

NALOG ZA UPLATU

Nikola Dzekic
Prijava ispita
Akademija Strukovnih Studija Zapadne Srbije u Užicu
153 RSD 0.00
840-17756472-23
156 154532

PREUZMI PDF

Figure 4 - Example of the home page when the user successfully logs in

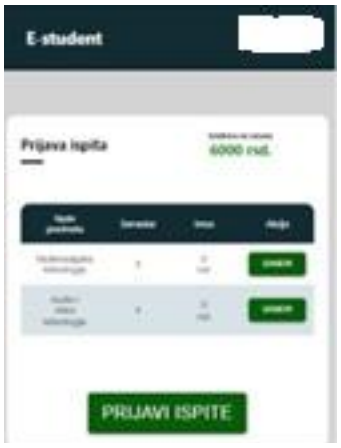
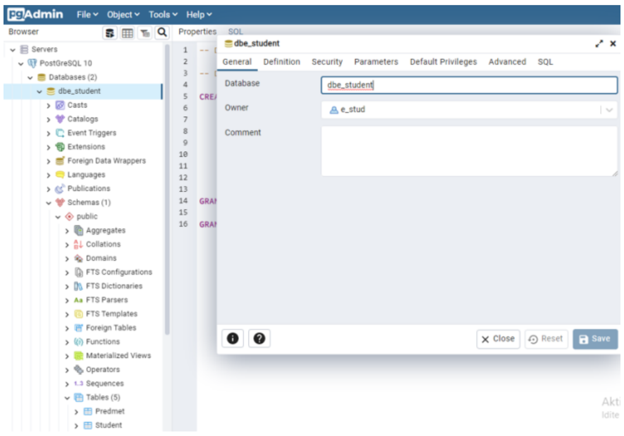


Figure 5 - Appearance of the application on a mobile device

Figure 6 - Exam registration from the admin menu

Figure 7 - Payment of funds



dbe_student

Database: dbe_student
Owner: e_stud
Comment:

Figure 8 - Database layout in PgAdmin

As the situation with the pandemic showed us that it is necessary to have an organized information system, the introduction of this application made it easier for students who are unable to physically come to school to register for the exam in this way. By regular maintenance and monitoring of the functionality of the application, additional refinement of the functions and scalability of the application can be carried out.
This application also does not have to be strictly related to the application of the exam, but can also be extended with additional options to the student portal, as well as to more complex applications, because the components and the base are very modular.

III. CHARACTERISTICS OF SERVERLESS ARCHITECTURE

Serverless architecture has gained a lot of popularity in recent years because it has allowed developers to easily and quickly test pieces of code without having to worry about the actual servers and the requirements they have to meet, and they also no longer have to worry about logging, monitoring or scaling the code. Serverless computing gained the most popularity with the appearance of AWS Lambda in 2014.
There is still no precise definition of what serverless is, but we can refer to some obvious characteristics of serverless architecture:

1. No management of server systems or server applications;
2. Horizontal scaling is automatic, elastic, and managed by the provider;
3. Costs are determined according to precise usage records (pay-as-go);
4. Performance capabilities are defined in terms other than the size or number of available instances;
5. Implicit high availability.

Serverless can be divided into two overlapping areas: Function as a Service and Backend as a Service (managed services). FaaS, also known as serverless computing, is a service that allows application logic to run on compute containers that are fully managed by a third party. Serverless computing is a type of Function as a Service, which is part of event-driven computing.

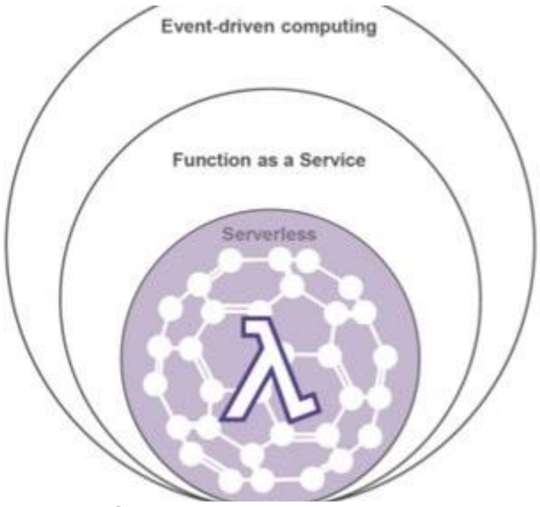


Figure 9 - Relationship between Serverless computing, Function as a Service, and Event-driven computing

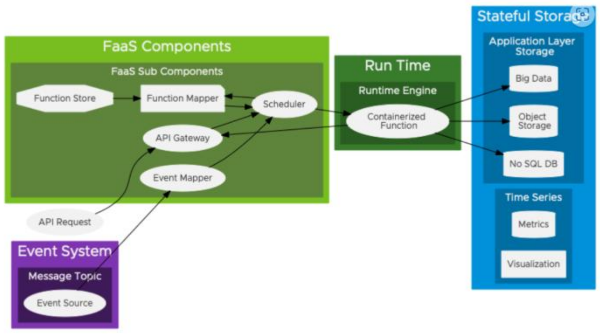


Figure 10 - Faas conceptual design

AWS offers a wide range of managed services for building applications or processing data without owning a server. Using them instead of building a custom solution can greatly reduce development costs and time. A good familiarity with the services offered by AWS is necessary before starting the development of a new solution.
The core of AWS serverless computing is AWS Lambda. This service enables code execution without provisioning a server. Lambda can be invoked by the event from other services and manually using the AWS SDK or the AWS console. The price depends on the execution time and available RAM.

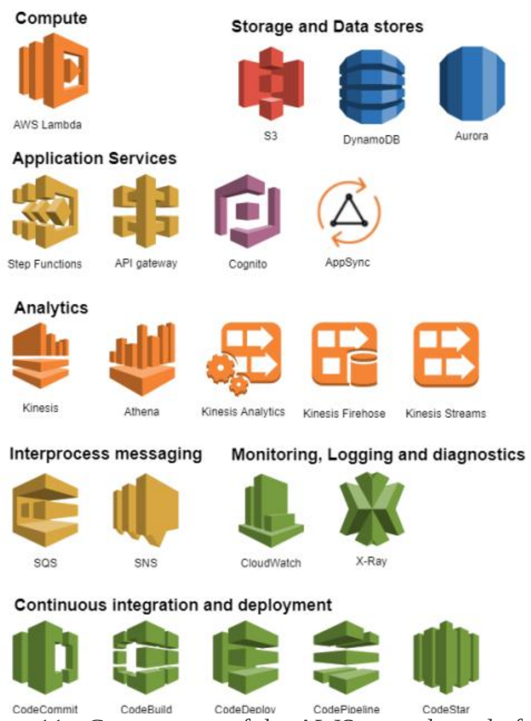


Figure 11 - Components of the AWS serverless platform

AWS Lambda runs code on high-availability computing infrastructure and performs all administration of computing resources, including server and operating system maintenance, capacity provisioning, and automatic scaling and logging. With Lambda, we can run code for virtually any type of application or back-end service.
Creating, invoking, or managing Lambda functions can be done using any of the following interfaces:
- AWS Management Console
- AWS Command Line Interface (AWS CLI)
- AWS SDKs
- AWS CloudFormation
- AWS Serverless Application Model (AWS SAM)

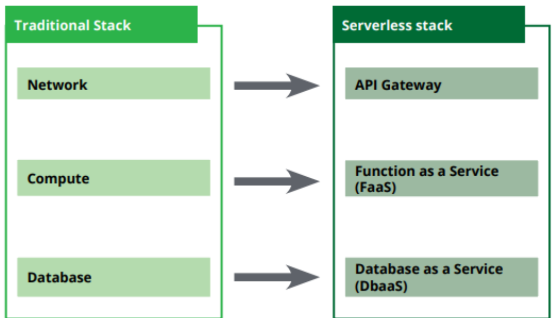


Figure 12 - Comparison of traditional and serverless architecture

The three key technology components of a serverless computing stack include the following:
- API Gateway
- Functions or Function as a Service (FaaS)
- Backend as a Service (BaaS)

The API gateway acts as a communication layer between the frontend and the FaaS layer. It maps REST API endpoints to corresponding functions that drive business logic. Since there are no servers, there is no need to set up and manage a load balancer in this model.
Functions or Function as a Service (FaaS): This is a layer that executes specific business logic (or code) in the cloud that provides a level of abstraction in terms of executing business logic.
Backend as a Service (BaaS): This is essentially a cloud-based distributed NoSQL database that essentially removes the administration costs of traditional databases.
The advantages of serverless computing are:

1. It is a fully managed service provided by cloud service providers and developers do not need to worry about the underlying infrastructure, operating system, code execution time, it's management, and dependencies;
2. Event-based approach is supported: functions are triggered based on events. Various cloud services and existing applications that support the trigger mechanism can initiate and run the function;
3. Provides infinite scalability and built-in high availability: Depending on user traffic, functions scale horizontally in a fully automatic and elastic manner managed by the cloud service provider;
4. Fewer Operations: Serverless does not necessarily mean NoOps for service users; however, it can mean "fewer operations" because operational tasks such as debugging, testing, troubleshooting, etc., remain while infrastructure management is entirely left to the cloud service provider;
5. Payment according to execution time: In the serverless computing model, consumers pay only according to the duration of execution of functions and according to the number of executed functions. When the function is not executed, no fee is charged – eliminating any idle time. This is a significant advantage over cloud computing where users are billed hourly to run virtual machines [11][12].

IV. OBJECTIVE OF THE RESEARCH

The research aims to compare the performance of the electronic exam registration system used at the Academy of Vocational Studies in Western Serbia on the server and serverless architecture.

The system for electronic registration of exams is located on the server in the Academy's premises, as already mentioned. Such a solution has both advantages and disadvantages, but we wanted to compare the behavior of the electronic exam registration system in a serverless environment to have an insight into which architecture, that is, which solution would bring greater benefits.

V. RESEARCH METHODOLOGY

The first step in the research was to check the responsiveness of the system to various queries. For this purpose, we used Postman, a program that enables API testing. An API (Application Programming Interface) defines how an application can request services from libraries, and control access to hardware devices or software functions for which the application may not have permission. Simply put, an API is a way that an application, library, or operating system provides access to other applications and programs. It is a set of procedures, functions, and everything that determines the way to access the application. The API takes the request, sends it to the application for processing, and returns the response. APIs are all around us today, in every application or service, and therefore they must be well-tested. One of the tools that make this possible is Postman.

POSTMAN is an API client used for APIdevelopment, testing, sharing, and documentation. It is used for backend testing where we enter the endpoint URL, send a request, and receive a response. Postman helps to identify API endpoints by quickly executing requests according to the API specification and enables analysis of various parameters such as status code, header, or response body.

The following images show the results of various queries generated in Postman. Postman has a very good option that allows you to create a collection of queries and this was used to analyze the responsiveness of the electronic exam submission system.

If as a result of the query we receive the answer Status: 200 OK, we are sure that the API is working in the right way. Any other answer means that either the address is not good or some kind of authentication is required. The obtained results can be viewed in different formats (JSON, XML, HTML, Text, Auto)[13].

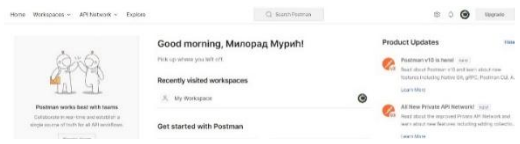
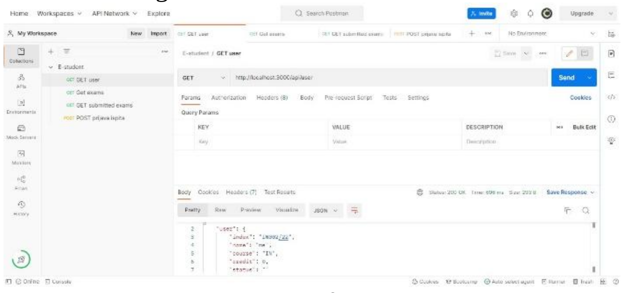


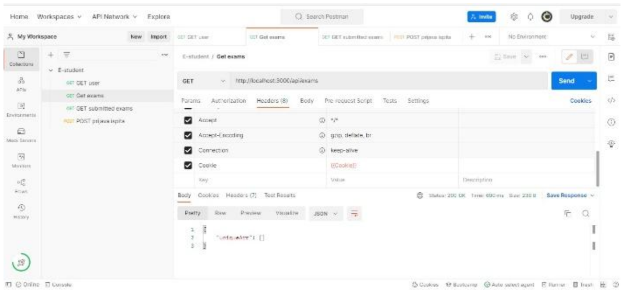
Figure 13 - Postman home screen



http://localhost:3000/api/user

```
"user": {
    "index": "IN982/22",
    "name": "me",
    "course": "IN",
    "credit": 0,
    "status": ""
```

Figure 14 - Result of GET user query



E-student / Get exams

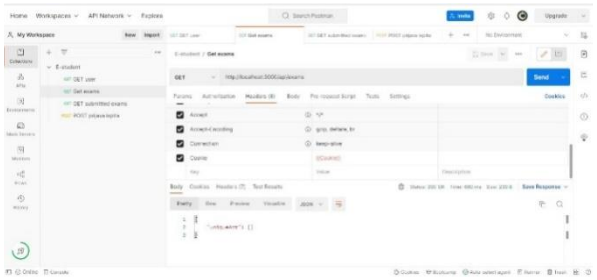
GET http://localhost:3000/api/exams

	Key	Value	Description
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Cookie	{{Cookie}}	

Status: 200 OK Time: 690 ms Size: 230 B

```
{
    "uniqueArr": []
}
```

Figure 15 - Result of GET exams query



E-student / Get exams

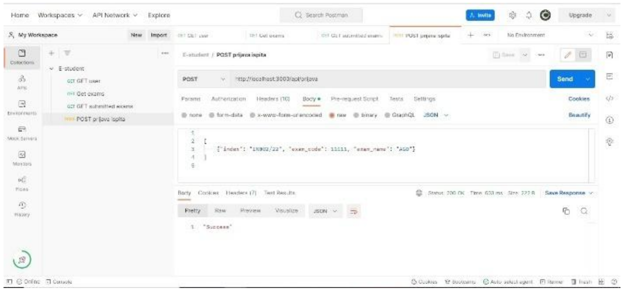
GET http://localhost:3000/api/exams

	Key	Value	Description
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Cookie	{{Cookie}}	

Status: 200 OK Time: 690 ms Size: 230 B

```
{
    "uniqueArr": []
}
```

Figure 16 - Result of the GET submitted exams query



E-student / POST prijava ispita

POST http://localhost:3000/api/prijava

```
[
    {"index": "IN902/22", "exam_code": 11111, "exam_name": "AGD"}
]
```

Status: 200 OK Time: 633 ms Size: 222 B

```
"Success"
```

Figure 17 - Result of the POST exam application query

As can be seen from the previous pictures, all queries were successfully implemented with a good system response and in a very short time.

VI. RESEARCH RESULTS

6.1. CHARACTERISTICS OF APPLIED SERVERLESS ARCHITECTURE

The research aims to compare the traditional and serverless architecture of the system for the electronic application of exams at the Academy of Vocational Studies of Western Serbia, Užice department, to obtain relevant parameters that show which architecture has a better and more economical response.

The research was conducted in the October exam period of 2022, on a sample of about 200 exam applications. The serverless solution implemented for analysis and research includes AWS Lambda, API Gateway, and EC2 services.

The main problem that needs to be solved is how to manage traffic whose intensity is unknown and which can affect the performance of the system. If we know the potential pattern for the query intensity, it is relatively easy to provide automatic scaling or adaptation of the system and properly respond to the query intensity at specific, known intervals. When these intervals have passed, the instances can be deactivated as they are no longer necessary and this is fully supported by the cloud provider.

However, in cases where we do not know the intensity of the queries as well as the intervals of their access, system slowdowns, weaker responses, and time limits occur most often. In such situations, auto-scaling does not respond properly, and additional instances may introduce an even greater time delay. All this requires very careful planning and monitoring of system operation.

If we decided to activate additional instances, it would only lead to increased costs and insufficient utilization of instances in conditions of reduced traffic.

To arrive at an optimal solution, a good knowledge of cloud infrastructure is required, how to use the advantages offered by cloud providing in the best way, to optimize the code in the right way to give the best results,…

When planning an AWS architecture, attention must be paid to several key items that ensure the best performance. It introduces the AWS Well-Architected framework and describes key concepts, design principles, and architectural best practices for designing and running cloud workloads [14].
The framework is presented through the following key pillars:

1. Pillar of operational efficiency
2. Pillar of security
3. Pillar of reliability
4. Performance efficiency pillar
5. Cost optimization pillar

The research is focused on the pillars of reliability and efficiency, but the other elements of the framework should not be neglected because they are all necessary for a successful architectural design.

An important parameter for choosing a cloud provider is the service price model and cost projection for expected system workloads. Considering that prices change, it is necessary to be well-informed and use cost calculators to predict them in the best way, following the usage pattern and the required performance [15].

Serverless architecture is a type of computing that is extremely useful for high-performance, low-cost, and on-demand needs [16][17]. It scales incredibly well and uses small blocks of code and persistent storage (usually). It requires no administration, provisioning, management of servers, environments, cluster logic, etc. However, not all workloads are adequate for serverless operation large workloads are best left to server instances and clusters. This also depends on the use case, in some cases it is simply more economical to stick with classic computing instances than to go for a serverless solution [18].

The cloud provider is responsible for cloud security, while the customer is responsible for cloud security.

Reliability is a major concern that needs to be addressed before the entire system goes into production, meaning that there must not be a single weak link in the overall architecture. Cloud providers know this is critical for their customers, so almost all services available in the cloud are highly redundant, resilient, and built to span an entire region with multiple access zones. For computing services, virtual private clouds, database servers, etc. there are recommendations for creating and deploying these services, and this documentation and best practices are usually available in the cloud provider's documentation. Some cloud providers, such as AWS, even go so far as to create templates with pre-built dev/test/production configurations.
The following AWS cloud services were used in the research:

1. API Gateway
2. Lambda (Serverless compute)
3. EC2 (Elastic Compute instances)

EC2 must be configured to be highly available and is best configured within a VPC (Virtual Private Cloud).
For some tasks, such as event-driven ones, serverless will always win because of its performance, ease of deployment, cost, and scalability. However, for continuous processing, it will be cheaper and easier to maintain a traditional (server) architecture.
There are also problems with debugging serverless applications, various tools are available, however, it is still a big challenge for developers. The final decision depends on the use case and required performance/scaling [19][20].

6.2. TESTING AND RESEARCH RESULTS

Before displaying the test results, it is important to understand Lambda's cold start and warm start. When a Lambda function is run for the first time in a while, there is an entire lifecycle of requests that AWS must perform. AWS takes some time to download the Lambda code, start a new container, deploy the code, calculate the runtime, and run it. All these operations delay the total execution time. Therefore, it is in the programmer's best interest to keep their initial program and code optimized to reduce this time as much as possible (Figure 18.)

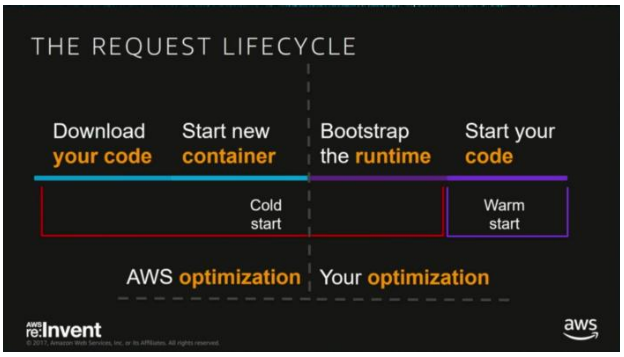


Figure 18 - AWS Lambda Lifecycle [21]

This is considered a cold start and this is the worst performance scenario. Once a lambda is activated, it is stored in memory for a while, and this period varies and depends on the lambda configuration. For example, lambdas deployed within a VPC tend to stay "warm" for a longer time, also, lambdas that require more memory than the default setting (128 MB RAM) tend to stay "warm" longer. There are ways around this by using a lambda "heater", which is a function that periodically pings your lambda to keep it "warm", however, not all lambdas need to be "warm", as most of them have fast activation times (Python and NodeJS) because there is no reason to "warm" them.

Lambda is written in NodeJS and processes the request body, maps the JSON to variables, modifies the request by adding a unique identifier to it, and writes the request data into RDS. There are libraries available for NodeJS to complete all the actions described, most of them are already integrated into AWS so it's just a matter of importing the library, such as the AWS SDK, however, the libraries to communicate with the database need to be manually added to bootstrap, causing Lambda size and startup time to increase, which significantly hurts performance. The tested lambda is located in Frankfurt, Germany, which is the eu-central-1 zone within the AVS. The latency to eu-central-1 from the test site is 63ms. The results are shown in Table 1.

Cold start	Hot start
1256 ms	54 ms

Table 1. Lambda response in milliseconds

The number of active threads is limited based on the size of your EC2 instance. For example, an EC2 instance with 8 v-cpu cores can handle approximately 20-25 threads, this depends on the messages and processing that needs to be done.

Scaling a traditional (server) architecture requires additional time, with the creation of a Linux instance varying between 60 and 90 seconds, plus the deployment time of the application it needs to run. Overprovisioning an EC2 instance is not recommended and is not considered cost-effective. Underprovisioning or even properly sizing your instance can cause requests to time out in fast traffic, depending on how fast the traffic increases, most likely the autoscaling pool won't respond in time.

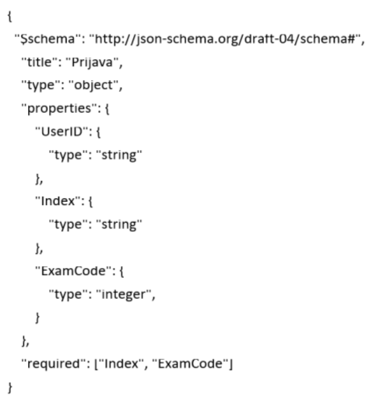
Note that serverless solutions also have their quotas and limits (Table 2), however, these can easily be increased by making requests for more throughput.

AWS Service	Quota/Limit	Can it be increased?
API Gateway	10 000 requests per second + throttling options	Yes
Lambda	1000 executions in parallel	Yes, to tens of thousands

Table 2. Quotas and limits of selected AWS services

Limits for most AWS services may be increased or are already being developed to handle more intensive traffic.

Starting with the API Gateway, the service is highly available, however, it is crucial to properly configure the gateway. To filter and validate requests, rather than triggering a lambda to validate them, as this increases the cost, models on the gateway must be coded to validate these requests before they reach the Lambda function [22]. An example model is shown below:



```
{
 "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Prijava",
  "type": "object",
  "properties": {
    "UserID": {
      "type": "string"
    },
    "Index": {
      "type": "string"
    },
    "ExamCode": {
      "type": "integer",
    }
  },
  "required": ["Index", "ExamCode"]
}
```

Figure 19 - Example of a model

This schema validates an application-type object with properties in the model. If the request does not match the schema, API Gateway will reject the request with an "invalid format" message attached to the response body, making sure that invalid requests never trigger a lambda function and incur additional overhead. Additionally, API Gateway can send custom-coded responses, create and validate tokens (custom or Cognito authorizers), modify requests, validate request headers, etc. so a lot of checking and authorization can be handed over to API Gateway processing with no overhead involved. It is preferable to do this to offload these tasks and take advantage of the impressive optimization and speed with which API Gateway handles it. Another pre-configured security feature is AWS Firewall and Shield to help protect API Gateway from malicious attacks and requests.

Architecture testing was performed with 200 consecutive requests and shows the limits of the server-based architecture (Figure 20) versus the serverless architecture (Figure 21).

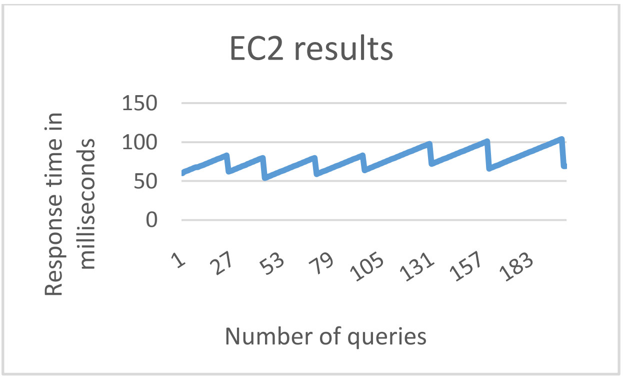


Figure 20 - Response time based on server architecture

As the load increases, on a server-based architecture, autoscaling kicks in and starts spinning up another instance, tuning the application and adding the instance to the load balancer. This can be seen during the spikes shown in Figure 3. The solution is not ideal, due to the time it takes for the instance to become available, if the number of requests was higher, the instance would not become available fast enough and there would be a significant number of outstanding requests.

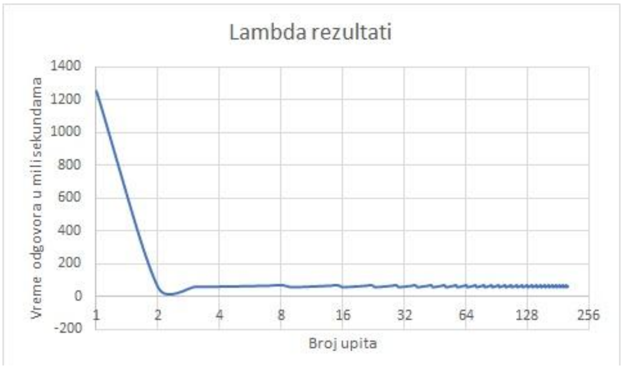


Figure 21 - Serverless architecture response time

When using the serverless architecture (Figure 21), lambda has an initial (cold start) delay, however, successive requests are served without any problems and with only minor differences in response time. This overcomes the autoscaling problems shown in Figure 20.
The average response time is shown in table 3.

Architecture	**Average response time in milliseconds**
Server-based architecture	104 ms
Serverless	66 ms

Table 3. Server and serverless architecture - average response time

VII. CONCLUSION

For architecture to be properly scaled, it must be made to measure. Modifying an existing architecture, and ensuring that it is optimized and easy to scale is even more difficult, as it requires additional people and planning, which is expensive and time-consuming.

Handling intense bursts of traffic is best done with a loosely coupled architecture, deploying services that are already highly available and easily scalable wherever possible. Where it is impossible to use serverless features, it is important to reduce requests and distribute queues, this will ensure that the architecture has enough time to scale, as long as it serves incoming requests (albeit more slowly) and makes sure that no data or incoming requests are lost or timed out.

The system for electronic registration of exams was tested in a relatively short time interval, which does not give completely authoritative results. To obtain relevant parameters, it is necessary to test the system over a longer period, with a significantly larger number of queries, which depends on the specific exam period, i.e. on the number of registered exams.

Based on the results of the previous research, we can conclude that the serverless architecture gives better results in response speed and expected exploitation costs. On the other hand, if it is taken into account that the server architecture is hosted on the premises of the Academy, the server solution is currently more economically profitable.

Of course, it should be borne in mind that the system for electronic registration of exams must meet all the requirements with the number of students at the Academy and with the number of registered examinations, which forces the conclusion that a serverless solution is an optimal choice in the coming period.
This approach requires knowledge and experience in using specific cloud services, depending on the provider, however, in the long run, it is the most efficient and cost-effective approach.

REFERENCES

[1] EP_Skripta1.pdf (link-onlineservice.com)(October 2022)
[2] Microsoft Word - DIPLOMSKI (vladofilipovic.github.io)(October 2022)
[3] Analysis and challenges of robust E-exams performance under COVID-19 (sciencedirectassets.com)(October 2022)
[4] https://www.ajol.info/index.php/gjmas/article/view/111502/101277(October 2022)
[5] Online learning should change the way that exams work (universityworldnews.com)(October 2022)
[6] EJ1145214.pdf (ed.gov)(October 2022)
[7] Факултет техничких наука у Чачку - СТУДЕНТСКИ ПОРТАЛ (kg.ac.rs)(October 2022))
[8] Машински факултет - Електронска пријава испита (bg.ac.rs)(October 2022)
[9] Odsek Užice :: Dobro došli (vpts.edu.rs)(October 2022)
[10] Front Page - AKADEMIJA STRUKOVNIH STUDIJA ZAPADNA SRBIJA (akademijazs.edu.rs)(October 2022)

[11] AWS Lambda - Developer Guide (amazon.com)(October 2022)
[12] FULLTEXT01.pdf (diva-portal.org)(October 2022)
[13] Introduction | Postman Learning Center(October 2022)
[14] AWS Well-Architected Framework. Available at: https://aws.amazon.com/architecture/well-architected/(October 2022)
[15] C. Kotas, T. Naughton, and N. Imam, "A comparison of Amazon Web Services and Microsoft Azure cloud platforms for high-performance computing," 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2018, pp. 1-4, DOI: 10.1109/ICCE.2018.8326349.
[16] G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, 2017, pp. 405-410, DOI: 10.1109/ICDCSW.2017.36.
[17] T. Lynn, P. Rosati, A. Lejeune, and V. Emeakaroha, "A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, 2017, pp. 162-169, DOI: 10.1109/CloudCom.2017.15.
[18] A. Eivy and J. Weinman, "Be Wary of the Economics of "Serverless" Cloud Computing," in IEEE Cloud Computing, vol. 4, no. 2, pp. 6-12, March-April 2017, DOI: 10.1109/MCC.2017.32.
[19] Baldini I. et al. (2017) Serverless Computing: Current Trends and Open Problems. In: Chaudhary S., Somani G., Buyya R. (eds) Research Advances in Cloud Computing. Springer, Singapore. https://doi.org/10.1007/978-981-10-5026-8_1
[20] G. Adzic and R. Chatley, “Serverless computing: economic and architectural impact”, Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017). Association for Computing Machinery, New York, NY, USA, 884–889. DOI:https://doi.org/10.1145/3106237.3117767
[21] AWS Re:Invent 2017. Available at: https://www.youtube.com/watch?t=8m5s&v=oQFORsso2go&feature=youtu.be [09.01.2021]
[22] API Gateway Developer Guide. Available at: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html(October 2022)