DEVELOPMENT AND IMPLEMENTATION OF A COGNITIVE CLOUD ASSISTANT FOR OPTIMAL CLOUD SERVICE PROVIDER SELECTION

Dávid VAĽKO*, Miroslav KAPA**, Norbert ÁDÁM*, Heidar KHORSHIDIYEH*
*Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, Tel. +421 55 602 2839
E-mails: david.valko@tuke.sk, norbert.adam@tuke.sk, heidar.khorshidiyeh@tuke.sk
**IBM Slovakia, Aupark Tower, Protifašistických bojovníkov 11, 040 01 Košice, Slovak Republic
E-mail: miroslav.kapa@ibm.com

ABSTRACT

This paper explores the design and implementation of a cognitive assistant powered by cloud computing. We begin by reviewing existing cloud service providers and assistant solutions. Building on this foundation, we define a set of requirements for our cognitive assistant and utilize them to guide the development process. To validate the functionality of our creation, we compare its outputs with established reference architectures.

Keywords: *AWS, cloud, cloud assistant, Microsoft Azure, reference architectures*

1. INTRODUCTION

In today's information-driven world, selecting the right cloud service provider can be challenging. This research paper aims to develop a cognitive cloud assistant that helps users choose the best provider using cognitive computing approaches, such as natural language processing and machine learning. The assistant will understand user's requirements and preferences, providing relevant results.

The first task is to gather information about cloud service providers, including alternatives, costs, and their application to individual requirements. We will be focusing on AWS and Microsoft Azure. A web application will be created, allowing users to select the best cloud service provider for their project. The results will be displayed in various formats, including PDFs and other visual forms. Afterwards, a filter will be implemented to allow users to filter results based on price and individual preferences. The final step is the verification of the solution through an experiment on a targeted group of users, allowing adjustments and corrections to the encountered problems. The web application will also include a user and system manual for further development.

2. ANALYSIS

Cloud services are increasing in popularity and demand, leading to the emergence of new providers offering a series of products for different purposes. It's difficult to keep track of all these providers, so one option is to find a product that fits a wider range of requirements and develop expert knowledge of it. However, this approach has a downside: if a product is cancelled, architects must choose another product, requiring further expert knowledge. AWS and Microsoft Azure are two top cloud service providers offering a wide range of services for businesses.

The AWS provides scalable infrastructure and flexible deployment options, while Azure offers a similar range of services. Both companies integrate with Microsoft's software ecosystem, making them attractive for organizations investing in Microsoft technologies. AWS and Azure offer global reach, allowing for closer deployment of applications. Reference architectures, a standardized approach for designing and implementing systems, are provided by AWS and Microsoft Azure, guiding architects, developers, and engineers in making informed decisions.

Cloud computing can be divided into four [1] main types:

- Public
- Private
- Hybrid
- Multicloud

Private cloud computing is for a single customer and is usually hosted on-premise. Public cloud computing allows everyone to use its services, with providers like Amazon Web Services, IBM Cloud, Microsoft Azure, and Google Cloud offering their infrastructure and services. Public cloud computing is often hosted on-premise, and compliance with security regulations is mandatory [2].

Hybrid cloud computing is a combination of public and private cloud computing, running on-site for larger projects. It offers better storage management and is particularly useful in critical sectors.

Multicloud computing is a combination of at least two public clouds, offering a price/performance ratio. However, it has drawbacks such as compatibility issues, security and data privacy concerns, and human resource constraints.

Cloud computing types include IaaS (infrastructure as a service), PaaS (platform as a service), and SaaS (software as a service). As they evolve, they shape new ways of working, defining user control over the cloud and its components.

2.1. On-site cloud computing

On-site cloud computing involves the user owning the entire cloud infrastructure and components, not using cloud providers' services. This is advantageous for smaller infrastructures, offering security and flexibility. However, complex infrastructures may reduce product quality and cost, as one expert cannot handle every component. The

user also has to purchase and maintain the entire cloud infrastructure, potentially leading to high costs. In most cases, a large proportion of the infrastructure remains unused [3].

2.2. IaaS (Infrastructure as a Service)

IaaS presents a ”pay as you go” service that allows users to manage virtualization, servers, storage, and networking without the need for on-site infrastructure [4]. This saves money in building and maintaining cloud infrastructure. Users can increase or decrease cloud storage size using a web portal, which is reflected in the provider’s product price. However, security is a concern, but choosing a provider with a good reputation and satisfied clients can partially eliminate this issue.

2.3. PaaS (Platform as a Service)

Platform as a Service (PaaS) presents a ”pay as you go” service that provides a platform for web application development, including building, deploying, managing, updating, and testing [5]. It includes virtualization, servers, storage, and networking, but also handles operating system, middleware, and program operation.

2.4. SaaS (Software as a Service)

Cloud services have evolved from complete freedom in choice and responsibility in on-site to a carefree, controlled service called SaaS (Software as a Service) [6]. This type of service is used when teams lack resources to develop applications, like Salesforce, ServiceNow, or Gmail. However, SaaS risks include migration to another provider or failure, as the entire application is at the provider’s risk of unauthorized access or unexpected shut down.

2.5. Comparing existing solutions

Comparecloud.in is an open-source project that provides an overview of cloud services offered by various providers, including AWS, Microsoft Azure, Google Cloud, IBM Cloud, Oracle Cloud, Alibaba Cloud, and Huawei Cloud. The page categorizes services and products, thus allowing users to click on hyperlinks to their official pages. The project encourages contributions from all users, aiming to improve it overall. However, it does not simplify the selection process for users, as they must study products themselves. Additionally, the application is not automatic, making the data outdated and unusable for developers.

Cloudorado is a web application that compares different cloud service providers in four categories:

- Cloud server
- Cloud hosting
- Cloud storage
- Table with a rough comparison of features

The application allows users to select parameters and features for their projects, and results are presented for individual providers. However, this application only partially meets expectations due to the lack of up-to-date data and the redirection to the provider’s main page after clicking the ”Go to provider” button. The Versus section allows users to compare two specific cloud providers without having to look elsewhere. The authors of this research paper evaluates the application positively, as it allows users to select input parameters for price calculations. However, caution is advised as the application does not include the time and date of data updates.

CloudCmp is a comprehensive programming framework [7] designed to assist customers in selecting the right cloud service provider by accurately estimating price and performance requirements for specific applications. The framework focuses on selected providers like Microsoft Azure, AWS, and GoGrid, addressing the challenges of identifying providers that excel in all categories, such as pricing models, service types, and operating systems. The framework also includes methods for measuring price and performance.

This paper outlines a four-part design for a solution that receives input data through a user interface. The first part involves obtaining user input data, while the second part involves analyzing the data using an AI layer. The data is stored in an output JSON/XML file, processed by a JSON/XML parser. The third part extracts the data and transforms it into a structured format for the UI interface. The fourth part hosts backend services that process the data and provide it to the UI interface. The final part, the UI interface, will display the user’s input data, recommendations for cloud provider services, and pricing information. The frontend technologies of HTML, CSS, and JavaScript will be used for visualisation.

3. WEB INTERFACE AND INPUT DATA HANDLING

The Cognitive Cloud Assistant application uses a web page as the user interface (UI), allowing users to interact with the application on any device with an internet connection. Processes like AI evaluation, price calculation, and data parsing occur in the background, allowing users to interact without interference. The interface includes options for easy application use, ensuring users can work with the application without downloading additional files. This approach ensures seamless user experience and accessibility for all users.

The UI must be designed in terms of both appearance and componentry before any work on it can begin. The user interface must allow the user to carry out the following actions in order for it to be deemed full and functional:

1) Type the entry parameters into the text field
2) See the results of the search
3) Evaluate outcomes from various providers
4) Work with the outcomes

The web page displays a text field for user input requirements, which can be enlarged if needed. A SEND button is located next to the text field, sending the user’s text for processing.

After performing a search, the web page shows an area for users to view search results and perform sorting. The system currently compares only two cloud providers, with the results displayed side by side. If support for other providers is added in the future, the results will be displayed below each other.

Fig. 1, users can sort the results by price or complexity. The system calculates the number of service providers needed to build the desired cloud architecture, with the less complex architecture appearing on the left. Sorting by price or complexity would sort the results from most advantageous to least favorable.

The most important part of the results is the overview of the architecture at the cloud provider. Each result contains elements such as the cloud provider's logo, the list of services contained in the architecture, for example the EC2 instance. The EC2 instance consists of four parts: the image of the service, its name, the price obtained directly from the processed data, and the button (arrow) to show or hide the detailed overview of the service [8].

Second example is Azure and the Virtual Machines service, which contains the same information as the EC2 service from AWS but with the status change. By opening the service, users can view more detailed information and consumption data, represented by properties objects

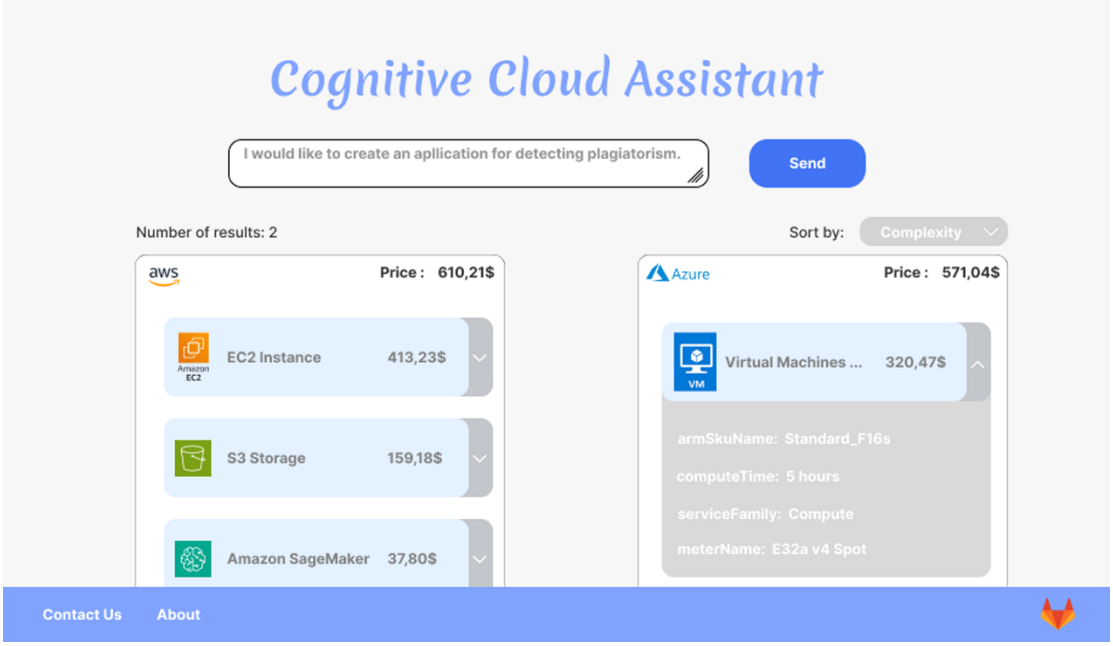


Fig. 1 Design of the Web Interface of the application

3.1. Design and implementation of user interface

The user environment for a task was designed with a simple text field for input, which was limited in terms of precise requirements. To improve accuracy and display more true data, IBM's WatsonX cloud service was used to evaluate input data. The text field was replaced with an embedded Watson Assistant, which the user can interact with directly on the web page. Watson Assistant provides specific data and a visual representation of the assistant.

A custom chat was created to allow the user to interact with Watson Assistant and send messages. The assistant also offers options for the user to choose from, which are displayed using buttons (Fig. 2).

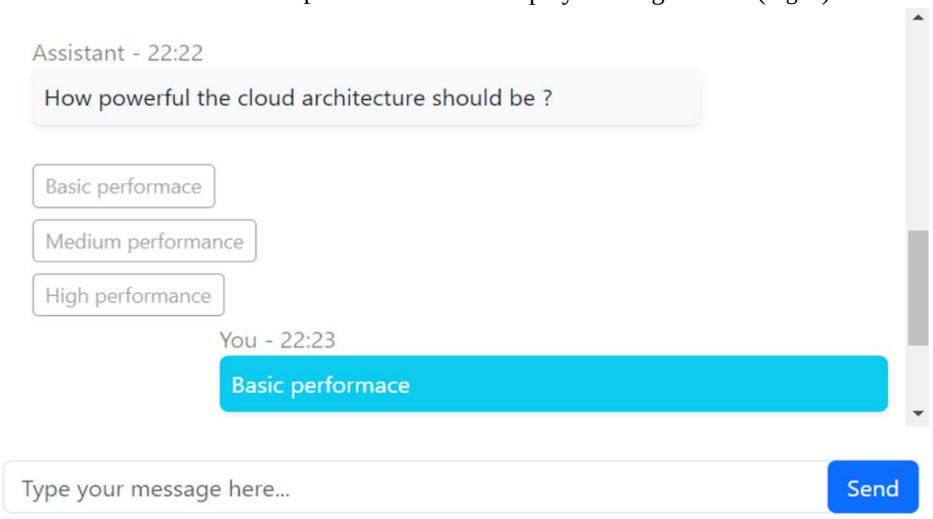


Fig. 2 Implementation of Chat window

The chatbot is configured to greet the user and ask them to type in their requirements for the system. If an eligible architecture is found, the assistant generates JSON and displays it on the web page. If no architecture is found, the assistant informs the user in a dialog box. If multiple eligible architectures are found, the assistant asks additional questions to determine the best suited architecture.

3.2. Backend of the application

This paper focuses on the architecture of web services, starting with the user environment and web page. The user enters text, which is sent to the main endpoint, which is responsible for processing the data. If an error occurs, the user is presented with a dialog box explaining the issue. The endpoint generates JSON containing individual cloud services using AI, and if an error does not occur, the data is sent to the validation service. The validation service determines if the JSON has a predefined structure. If the data is valid, it is sent to the calculation service, which calculates the data and returns it to the web page. The service passes through individual cloud provider services and performs costing based on the provided parameters. To make the solution available online, the backend services and JSON parser are executed on the server. The Render3 platform is chosen for this experiment, which offers a free plan for web services. However, there are limitations, such as performance, availability, and the time the web service can be used for free [9].

3.3. The difference in the design and implementation of backend services

The backend services have undergone changes in the implementation, including a shift from text-based architecture requirements to user interaction with the Watson assistant. The AI layer now returns a list of service types that best fits the user's specification, eliminating the need for the JSON/XML parser. The "validate-data" endpoint has been replaced with a simple JSON with a single text attribute, containing a list of service types needed to create a given architecture. The application contains a list of reference architectures for AWS and Azure, which are compared and assigned a percentage match. If the percentage match is 0, the architecture is not included. If the architecture contains multiple services of the same service type, the type is taken only once, avoiding complex reference architectures having better percentage matches. The output is a list of reference architectures that most closely match the input data.

4. EVALUATION OF PROPOSED CONCEPT

The solution evaluation will involve a two-part experiment, focusing on finding *MatchingService* architectures and cost calculation for the array of architectures. The experiment will involve calls to services, verification using tables and comparison with cloud service provider's price catalogues.

4.1. Verifying the accuracy

The service's accuracy will be verified using 14 cloud architectures, with a table reflecting the results in JSON format. The table, along with the descriptions of each architecture, forms a mapping to determine the percentage match between the architecture and the input. The evaluation procedure involves counting unique service categories, comparing the results with the table, and determining the correctness of the match.

4.2. Evaluation of *CostCalculationService*

This service will be evaluated using a distinct methodology from the one used to assess the prior service. The *CostCalculationService* uses the AWS SDK for Java service in AWS and the Azure service in Azure to determine Retail Prices, as previously mentioned in the categories. Since both of these services operate on the basis of filtering, it is possible to compare the cost on the official catalogue pages of both cloud service providers and the price that is obtained by calculating.

CostCalculationService supports two providers in the implementation of the solution - AWS and Microsoft Azure. Although the calculated pricing of the service acquisition for the two cloud service providers is slightly different, a similar procedure was used to evaluate this service for both of them:

1) Selecting the service whose price will be used to evaluate
2) Evaluating costs using *CostCalculationService*
3) Evaluation of the calculation using the official catalogue/calculator of the cloud provider
4) Compare results from *CostCalculationService*

The Azure Key Vault service from Microsoft Azure was chosen as a specific example of evaluating this service. The Azure Key Vault service was calculated using *CostCalculationService*, and the outcome is displayed on Fig. 3 The filtered price is displayed in pink, while the parameters that were used to determine the price are displayed in yellow.

```
{
  "name": "Azure Key Vault",
  "type": "security_and_identity_and_compliance",
  "id": "6",
  "zone": "EU North",
  "region": "northeurope",
  "vpc": "none",
  "sku": "DZH318Z0BQG0/003D",
  "unitUsage": 750.0,
  "estimatedUsagePrice": "22,5$",
  "pricing": [],
  "unitPrice": [
    "0.03"
  ],
  "measurementType": "10K",
  "properties": {
    "productName": "Key Vault",
    "meterName ": "Operations",
    "skuName": "Standard"
  }
},
```


Fig. 3 Azure Key Vault sample after calculation using *CostCalculationService* in JSON format

On the Fig.4 we can see the outcome of utilizing the pricing table on the service's website to calculate the Azure Key Vault service. The parameters that were used to filter the price are displayed in yellow, and the final price is displayed in pink, much like with the result obtained using *CostCalculationService*. It can be observed that the computation's results match if we use the pricing table on the website for the specified service, where $0.03 is the unit price and 750 is the number of units.

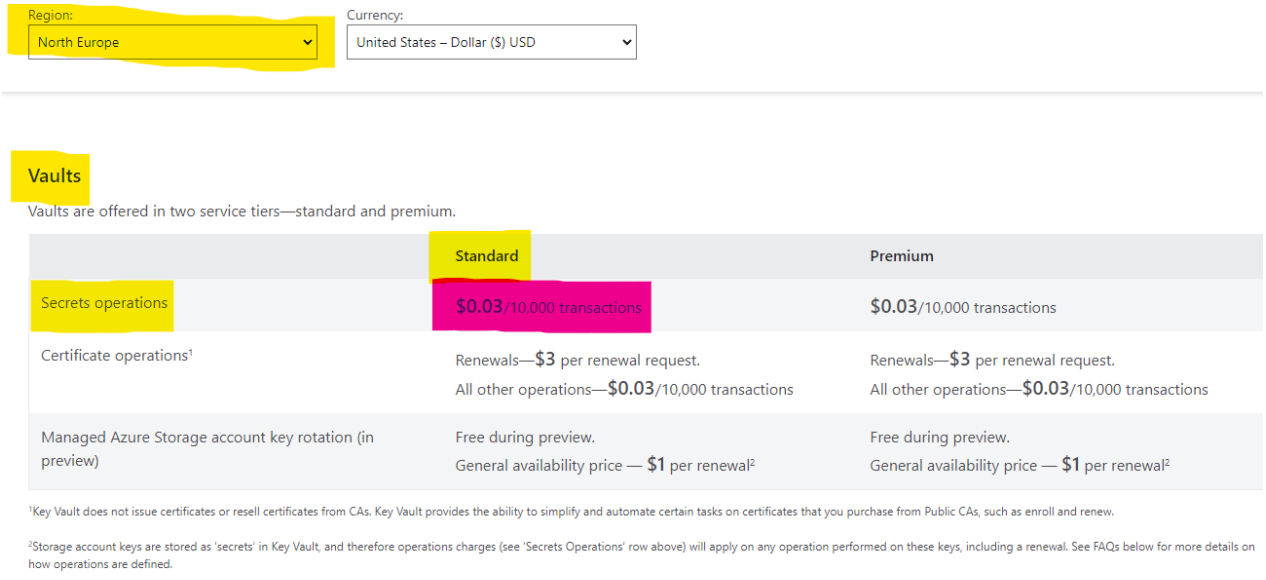


Fig. 4 Sample Azure Key Vault pricing using the Pricing table at Azure Key Vault service page

When evaluating the *CostCalculationService*, it should be taken into account that the more accurate the service description it contains, the more accurate the results will be. This service is also not able to work with marginal prices, so in case of computational retrieval of the service price, multiple values are returned in the result, and by averaging them we could receive inaccurate values.

5. DISCUSSION

The cognitive cloud assistant simplifies cloud service provider selection, enhancing decision-making and reducing the expertise required for evaluation. Its intuitive NLP interface makes it accessible to non-technical users, while detailed cost comparisons help businesses optimize

their budgets. However, the tool's effectiveness relies on accurate and up-to-date data. Security and privacy concerns must be addressed, and integration with existing systems can be challenging. Despite these limitations, the assistant offers significant practical benefits, improving efficiency, cost savings, and user experience in optimizing cloud infrastructure solutions. According to our opinion in general, cloud services will be widely used in the future.

6. CONCLUSION

This research paper aims to create a cognitive cloud assistant that allows users to communicate in natural language and provide recommendations for software architecture design based on input data and internal data models for each cloud provider. The assistant should compare results based on pricing, complexity, and compliance. The analytical part of the research paper discussed cloud computing, cloud services, and comparison criteria. The author also analyzed existing cloud assistant solutions and evaluated their strengths and weaknesses. The main contribution of this research paper is the creation of an application that serves as a Cognitive Cloud Assistant, allowing users to interact with the AI layer to help them select the right architecture.

Possible improvements include a new system for searching suitable architectures that match input data from the AI layer, archiving search results, and exporting results for sharing with other architectures.

REFERENCES

[1] FURHT, B.: “Cloud computing fundamentals,” Handbook of cloud computing, pp. 3–19, 2010.

[2] JANSEN, W. – GRANCE, T., et al.: “Guidelines on security and privacy in public cloud computing,” 2011.

[3] DAVIDOVIC, V. – ILIJEVIC, D. – LUK, V. – POGARCIC, I.: “Private cloud computing and delegation of control,” Procedia Engineering, vol. 100, pp. 196–205, 2015, 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705815003859

[4] RedHat. (2022) Iaas vs paas vs saas. [Online]. Available: https://www.redhat.com/en/topics/cloud-computing/ iaas-vs-paas-vs-saas

[5] AZURE, M.: What is paas. [Online]. Available: https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/ what-is-paas/

[6] IBM. Saas- software-as-a-service. [Online]. Available: https://www.ibm.com/topics/saas

[7] LI, A. – YANG, X. – KANDULA, S. – ZHANG, M.: “Cloudcmp: Comparing public cloud providers,” in 2010 Internet Measurement Conference. Association for Computing Machinery, Inc., November 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/cloudcmp-comparing-public-cloud-providers/

[8] "EC2 Image Builder - Amazon Elastic Compute Cloud." Amazon Web Services, Available: https://docs.aws.amazon.com/imagebuilder/latest/userguide/what-is-image-builder.html. [Online]

[9] "Deploy for Free." Render Docs, Render, Available: https://docs.render.com/free#:~:text=Monitoring%0A%0A%E3%80%9068%E2%80%A0.css. [Online].

Received June 24, 2024, accepted July 8, 2024

BIOGRAPHIES

Dávid Vaľko (Ing. Bc. PhD. student) was born Košice, Slovakia. He completed his master’s degree in Computer science in 2021 at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Since 2021, he has been a PhD. student at the Technical university in Košice. His scientific research focuses on the use of computer science in medicine, particularly the classification and processing of medical signals such as ECG. He also received a bachelor's degree in Urgent medical care in 2023.

Miroslav Kapa (Ing.) was born in Svidník, Slovakia. He completed his master's degree in 2024 at the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. Since he completed his bachelor's degree, he has been working as an automated tester at IBM Slovakia in Hybrid Cloud Solutions department. His professional interests include cloud services, software testing automation, the development of efficient testing frameworks and tools. His interest in cloud services led to his scientific research from the development of Cognitive cloud assistant.

Norbert Ádám (Assoc. Prof.) was born in 1980. In 1998 he graduated (MSc.) summa cum laude at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. He defended his PhD. in the field of Computers and computer systems in 2007; his thesis title was "Contribution to simulation of feed-forward neural networks on parallel computer architectures". Since 2018 he is working as an associate professor at the Department of Computers and Informatics. Since 2008 he is the head of the Computer Architectures and Security Laboratory at the Department of Computers and Informatics. His scientific research is focusing on the parallel computer architectures.

Heidar Khorshidiyeh (Ing., PhD. Student) was born in Al Ain, UAE. He holds Iranian nationality. In 2019, he graduated with a Bachelor's degree (Bc.) from the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. In 2021, he earned his Master's degree (Ing.) from the same department and faculty. Since 2021, he is PhD student at the Technical University of Košice. His scientific research is centered on computer graphics, specifically on the representation of 3D scenes using hierarchical data structures.