Microservice architecture in E-learning

1st Ana Milovanović
Department for e-business
University of Belgrade, Faculty of organizational sciences
Belgrade, Serbia
anamilovanovic994@gmail.com
https://orcid.org/0000-0001-5282-881

***Abstract*—Microservice architecture has found its application in the design and implementation of systems used in e-learning. Multiple cases of applying microservice architecture in e-education were considered and it is determined that there is a place to further investigate integration of the custom e-learning microservices with some existing LMS (Learning Management System) such as Moodle LMS. The aim of this paper is to present a possible solution of a custom microservice architecture that is based on integration with some existing LMS. The approach that was used to identify microservices is Domain Driven Design. Integration of the Cloud Data Platform into this architecture is also considered.**

Keywords—microservices, microservice architecture, e-learning, Cloud Data Platform, LMS, Domain Driven Design

I. INTRODUCTION

There are different definitions of microservices in literature. One of them is given by Sam Newman [1], who defines microservices as autonomous services that work together [1]. Thus, the microservice architecture can be viewed as an approach in the development of applications that are organized as a set of independent and small services that communicate using a mechanism, mainly the HTTP protocol [2]. The deployment of such services is performed independently of the others, and the centralized management of these services also stands out as a separate service [2].

As microservice architecture is used to implement different solutions due to its nature to provide more flexibility and reusability of existing parts (microservices) of the system, the purpose of this paper is to further investigate possible scenarios of integrating microservice architecture in e-learning environments.

Multiple cases of applying microservice architecture in e-education were considered. Some solutions investigate the migration of custom Learning Management Systems (LMS) to microservice architecture [3]. The others consider improvement of the existing microservice architecture of the e-learning platform [4] or VLE (Virtual Learning Environment) [5] or developing their own educational platform for remote access using microservices [6].

Based on the considered existing cases, it is determined that the integration of the custom microservices with some existing LMS is lacking. The goal of this paper is to provide a possible solution of the custom microservice architecture that will be part of the e-learning flow and that is based on integration with some existing LMS (e.g. Moodle). Also, possible design of Cloud Data Platform and its place in microservice architecture is presented.

In order to identify microservices Domain Driven Design should be considered [5]. This approach is based on Bounded Contexts. Those represent consepts that facilitate breaking systems that are complex into smaller parts [5]. Concepts such as The API Gateway and Message Broker, that are commonly used in the implementation of this type of architecture, are illustrated. The solution gives presentation of microservice architecture which consists of identified e-learning microservices, Cloud Data Platform and their communication and integration with Moodle LMS.

II. LITERATURE REVIEW

When comparing microservice architecture and monolithic architecture the first difference that is easily noticeable is in the deploy process. Monolithic architecture deploys entire application and all components of the system at once [2]. This implies that the whole code base and all parts of the application are in one place. It is hard to maintain this approach when working with distributed teams. On the other hand, in microservice architecture deploy of independent services takes place [2].

In order to achieve best practice of deploying microservices independetly [1], changes in one service can be found in production after the release, without being conditioned by changes in other services, i.e. there is no orchestration of deploys of different services [1]. Also, it is important to isolate failures [1] and to know when it is necessary to sacrifice availability, and when consistency in order for the whole system to work [1].

Using microservice architecture approach, new technologies can be adopted much faster and changes affect only one service or part of the system so the risk is lower [1]. By changing technology in the application that is built by monolithic architecture approach the whole system will be affected.

Scalability should also be considered. In the monolithic approach scaling can be done only in one dimension. Multiple copies of the application can be placed behind the load balancer and this causes increase in the number of client requests that can be simultaneously served [2].

In this case, scaling cannot be performed for each component that may have different requirements (e.g. CPU

or memory usage) [2]. This type of scaling is available in microservice architecture.

Service-oriented architecture (SOA) should also be mentioned in the context of comparison with microservice architecture. Compared to the traditional SOA architecture, the microservice architecture is more decoupled, i.e. the components of this architecture are more independent [7].

In their work [7], a group of researchers singled out the most commonly used architectural patterns in microservice architecture. These patterns were divided into three different groups according to their purpose [7]:

- Orchestration and coordination patterns - deal with logical communication and coordination of microservices.
- Deployment patterns - hosting is done via containers or virtual machines.
- Data Management Patterns – especially Data Storage options.

The first group includes the three most commonly used approaches, namely [7]: the API Gateway pattern, the Service Discovery pattern, and the Hybrid pattern.

The API Gateway routes the requests that consumers send to microservices and thus calls microservices and aggregates the results [7]. In [8], the use of this pattern in microservice architecture is compared with the Facade object-oriented design pattern. The most common application of this pattern is reflected in communication with various frontend clients [7]. Each client is provided with the custom API, so that each one of them has access to specific resources only [7]. In some cases, the API Gateway also serves as a load balancer because the locations and addresses of all services are known [7].

Message Broker works on the principle of publish / subscribe message system which is a form of asynchronous communication between services. In this model, every message sent to a particular topic will be received by all services that have subscribed to that topic [9].

There are numerous discussions on the development of cloud platforms, and they concern the choice between cloud-vendor-specific services as opposed to creating a platform that would be cloud independent [10]. The experiences of the authors [10] are such that from the point of view of the support price, the PaaS (Platform as a Service) solution pays off more, even though there is a possibility of vendor lock-in. The universal architecture of the Cloud Data Platform consists of four layers [10]: Ingest, Storage, Processing and Serving.

The data collection layer (Ingest) is responsible for connecting to the data sources, whether the data is in stream form or in batch form [10]. The data storage layer has the main purpose of reliable storage for long-term use. There are two main types of this storage: the “fast” and the “slow” storage [10]. Slow storage stores data for a long time and can be accessed at any time, it is used to quickly read different volumes of data.

When it comes to streams, it is necessary to apply another type of storage, the so-called „fast“ storage. Apache Kafka as well as Amazon Kinesis are some examples of this type of storage [10].

The processing layer is the most important in the architecture and is the central part of the platform. Business logic and all transformations are implemented here, as well as any validations of data. This layer can perform various manipulations on the data, and later store them in the storage [10]. The service layer delivers the results of different analytical processing to different consumers.

In their work [3], researchers from the University of Tampere in Finland investigate the migration of existing custom Learning Management Systems (LMS) to microservice architecture. In order to share LMS resources between several universities (Smart Learning Environments project) and the impossibility for all universities to use the same LMS, the use of microservice architecture is needed [3].

The LMS should provide standardized services, some of those standardized solutions are [3]: SCORM (Sharable Content Object Reference Model) as a set of specifications for creating and sharing e-learning content, xAPI for the metadata on learning analytics and LTI (Learning Tools Interoperability) protocol for authentication and assessment. As basic microservices, the researchers pointed out [3]: authentication, management of user and course-related information, assessment, report generation and analysis.

The scientific paper [4], which is the work of a group of researchers from the University of Applied Sciences Emden / Leer from Germany, deals with the improvement of the existing microservice architecture of the e-learning platform STIMEY.

Several domains have been identified [8]: User profile, Activity stream (dashboard), Community, Social messaging, Online-courses. Based on this division, REST API services were identified [4]: Dashboard, Auth, Chat, Course, Robot and Communities Discussion.

Lihonga VLE (Virtual Learning Environment) from [5] is based on microservices, consists of loosely coupled components. Offline access to data stored on the client is enabled, the system can be hosted on the cloud and due to the use of open API this system can be expanded and adapted to specific environments [5].

The architecture consists of eight components, which are divided into four groups: Gateway, Message Broker (Kafka - stream processing platform), Services and Clients (Android and iOS applications). The core of the architecture is the API Gateway. It generates necessary events, and then sends the response (HTTP Response) to a client application. Events implicitly trigger services through the Message Broker [5].

In [6], the API Gateway component serves as a link between the client and the microservices. It receives client calls, and then communicates with the learning system by calling the appropriate microservice. The LMS component consists of three microservices: RemoteLaboratory, ExperimentGuides and Evaluation System. Each of these microservices further communicates with the data storage [6].

III. METHODOLOGY

In order to create custom microservice architecture and to identify the microservices that make up the system, Domain Driven Design approach and the Bounded Contexts are used.

In this case, it is necessary to identify e-learning microservices whose integration is done with the Moodle platform. Identified e-learning contexts are (TABLE I): authentication, resource search, learning through play, communication, analytics and reporting. A microservice corresponds to each identified context (Fig. 1):

- Authentication service,
- Teaching materials search service – searches for teaching materials by certain criteria
- Education games service – implementation of an educational quiz,
- Communication service – implements chat functionality,
- Analytics and reporting service – generating reports on student activities.

Those microservices communicate with Moodle LMS and with one another using asynchronous publish / subscribe patern in form of Message Broker. So messages sent to specific topic are going to be received by e-learning microservices that are subscribed to that topic [9].

TABLE I. Indentified Contexts

No.	Microservice architecture Contexts	
	E-learning Contexts	***Cloud Data Platform layers***
1.	Authentication	Ingest
2.	Resource search	Storage
3.	Learning through play	Processing
4.	Communication	Serving
5.	Analytics and reporting	

The second part of the architecture is the Cloud Data Platform with application in e-learning. As this is a data management platform, context identification is facilitated and relies on the layers identified in the universal architecture of the Cloud Data Platform [10]. Contexts i.e. the layers are as follows (TABLE I): Ingest, Storage, Processing and Serving. Each of these layers corresponds to a microservice, respectively (Fig. 1): Data reading and validation microservice, Database provider microservice, Data processing microservice and microservice LTI provider.

The GitHub version control platform is used as event emitter. This platform is commonly used for various Social coding [11] activities. If some of that activities occurs, for which webhook is created, Github webhook [12] event is generated.

After the GitHub webhook event has happened, the call is forwarded to the API Gateway which further determines which event needs to be generated and sends it to the Message Broker. The Message Broker forwards this event to the appropriate service. The Data reading and validation service was launched by this event.

It processes the data and validates it. This service further sends an event triggering the Database provider service through the Message Broker.

This service stores data obtained through the events in a NoSQL database, and is also in charge of generating an endpoint for reading and modifying data in this database.

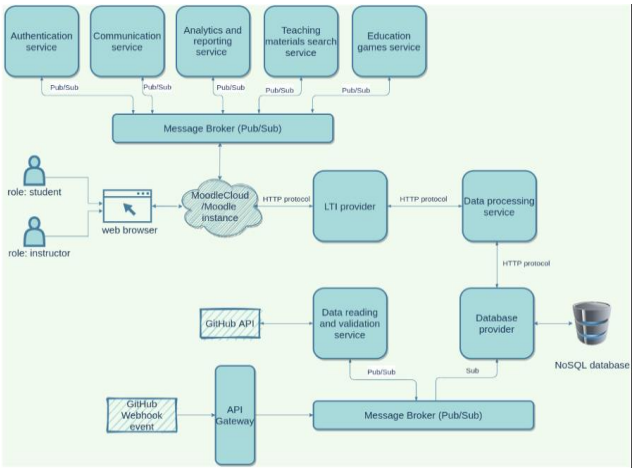

Fig. 1. Microservice architecture diagram

The Data processing service communicates with the previous service via the HTTP protocol. In this way, it obtains data from the database and transforms it. Finally, the service that uses the data generated in this way is the LTI provider that requests it via the HTTP protocol, as shown in the figure (Fig. 1).

In order for the LTI (Learning Tools Interoperability) provider service to successfully connect to the Moodle platform, it is necessary to use the LTI protocol [13]. Three components stand out in this part of the architecture [13]: web browser, provider (microservice LTI provider) and Moodle LMS.

The web browser is in charge of displaying the module that the Moodle LMS (e.g. an instance of the external Moodle website) requests from the provider. At the time of the request, the Moodle sends data (payload) to the provider address by HTTP POST method [13], and in response receives data from the LTI provider in HTML format and thus displays the given module on the page where it is referenced in the settings.

Since the end user can have different roles (student or instructor) and since the role data can be obtained from Moodle [13], if the call comes from a user that has a role of student, the display of the module may differ from the one that is displayed to the user having the role of instructor.

IV. Results and Implications

Exploring literature that dealt with implementing microservice architecture in e-learning environments led to a conclusion that integrating this type of arhitecture with existing LMS solution (e.g. Moodle LMS) is lacking in practice.

In search for possible ways of integration, a proposal of microservice architecture with application in e-education is given. In addition to the general presentation of microservices and their communication, the patterns used in the implementation of this type of architecture are illustrated.

Asynchronous communication is performed between certain services according to the Pub / Sub pattern principle, and thus better performance is achieved. Message Broker is used for communication between microservices. As in paper [5], Kafka can be used for implementation of this pattern.

E-learning microservices whose integration is done with the Moodle platform were identified. They are an extension of the LMS solution. Also, they use data available to Moodle, which is externally available through the Moodle API. Then, a possible design of Cloud Data Platform is considered to be integrated with this type of arhitecture. GitHub is identified as event source. The webhook concept is used to invoke appropriate services in Cloud Data Platform. Microservice patterns the API Gateway and Message Broker are used in this part of architecture.

In conclusion, it is important to state that, as given microservice solution suggests, there is a possible way to implement microservice architecture which is based on integration with existing LMS system. This solution also includes possible scenario of integrating custom Cloud Data Platform into the e-learning flow.

The given solution can be used as a starting point for further consideration of this idea. The main goal of implementing this kind of architecture is to achieve flexibility and reusability of existing parts of the system, that being the basic characteristic of microservices.

Furthermore, adding new e-learning microservices is facilitated since existing microservices are independent, concerning technology (programming language) and functionality that is provided in each one of them.

REFERENCES

[1] S. Newman, *Building microservices : designing fine-grained systems*, First Edition. O'Reilly Media, 2015.

[2] D. Namiot and M. Sneps-Sneppe, "On Micro-services Architecture," *Int. J. Open Inf. Technol.*, vol. 2, no. 9, pp. 24–27, 2014.

[3] P. Niemelä and H. Hyyrö, "Migrating Learning Management Systems Towards Microservice Architecture," 2019. [Online]. Available: https://www.tuni.fi/en.

[4] D. A. Bauer, D. Alessandro Bauer, B. Penz, J. Mäkiö, and M. Assaad, *Improvement of an Existing Microservices Architecture for an E-learning Platform in STEM Education*. 2018.

[5] S. Kapembe and J. Quenum, "Lihonga-a microservice-based virtual learning environment," in *Proceedings - IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018*, Aug. 2018, pp. 98–100, doi: 10.1109/ICALT.2018.00030.

[6] D. A. Segura-Torres and E. F. Forero-Garcia, "Architecture for the management of a remote practical learning platform for engineering education," in *IOP Conference Series: Materials Science and Engineering*, 2019, vol. 519, no. 1, doi: 10.1088/1757-899X/519/1/012020.

[7] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: A systematic mapping study," in *CLOSER 2018 - Proceedings of the 8th International Conference on Cloud Computing and Services Science*, 2018, vol. 2018-January, pp. 221–232, doi: 10.5220/0006798302210232.

[8] S. Zhelev and A. Rozeva, "Using microservices and event driven architecture for big data stream processing," in *AIP Conference Proceedings*, Nov. 2019, vol. 2172, doi: 10.1063/1.5133587.

[9] AWS, "Publish/subscribe messaging," *https://aws.amazon.com/pub-sub-messaging*, May 14, 2021.

[10] D. Zburivsky and L. Partner, *Designing Cloud Data Platforms*, ISBN 9781617296444. Manning Publications, 2020.

[11] AlMarzouq, M., AlZaidan, A. and AlDallal, J. (2020), "Mining GitHub for research and education: challenges and opportunities", International Journal of Web Information Systems, Vol. 16 No. 4, pp. 451-473. https://doi.org/10.1108/IJWIS-03-2020-0016.

[12] GitHub docs, "Webhooks and events," *https://docs.github.com/en/developers/webhooks-and-events/about-webhooks*, May 14, 2021.

[13] IBM tutorials, "Implement learning tool interoperability," *https://developer.ibm.com/tutorials/implement-learning-tool-interoperability*, May 14, 2021.