Improvement of an Existing Microservices Architecture for an E-learning Platform in STEM Education

David Alessandro Bauer, Benjamin Penz, Juho Mäkiö, Manal Assaad
Department of Informatics and Electronics
University of Applied Sciences Emden/Leer
Emden, Germany
Email: david.bauer@hs-emden-leer.de, benjamin.penz@hs-emden-leer.de, juho.maekioe@hs-emden-leer.de, manal.assaad@hs-emden-leer.de

***Abstract*— This paper demonstrates and evaluates the technical improvement of an existing prototype of the STIMEY e-learning platform based on a microservices architectural pattern. The first approach is using our page fragments technology that allows to integrate contents of other microservices in a superordinate context but lead to difficulties regarding maintenance. The second approach holds all page fragments in one microservice, and the specific data is provided separately by domain-specific microservices which makes it easier to work with them, in case of the STIMEY platform, because domain-specific designers can now be assigned to just one respective microservice. Additionally, a conception to migrate the platform to Amazon Web Services (AWS) in the future is shown. A novel three-dimensional architecture model is introduced to visualize the used microservices' architectural pattern. Three patterns are shown for the data access of the individual microservices and for their interconnectedness. At the end, it is discussed how the database design can be implemented.**

Keywords-STEM Education; E-learning; Web platform; Microservices; Architecture; Page fragments; Data Access; Database Design.

I. INTRODUCTION

As modern IT-technologies, like cloud computing and mobile computing, get more and more complex, difficulties with development and maintenance increase as well, thus driving innovation and research pressure to cope with these problems. This led to the construction of concepts like Domain-Driven Design, Continuous Integration, scalable systems and Software as a Service, which form the base of the idea of microservices. Microservices are applications that consist of a set of small, independent services in contrast to the large, monolithic kind of software systems [1]. Each microservice follows its own task but works together with other microservices to fulfill a more general purpose. This approach has several advantages such as [1]:

- the use of different technologies. Because every microservice is isolated, they can run on different platforms, and can even be implemented in different programming languages.
- that it is highly scalable. In contrast to monolithic architectures, the different parts of an architecture based on microservices can be scaled independently.
- a better exchangeability. Microservices can be exchanged independently, instead of the need to exchange the whole system, as it is the case with monolithic architectures
- a more convenient deployment. Each microservice can be deployed separately. With the monolithic approach, the whole system needs to be deployed in one, even if only small amounts of code have been changed.

Although the importance of microservices continually increases, research about this topic is mainly limited on the theoretical concept of microservices itself. [1] Therefore, this paper focuses on the applied architecture of a practical use-case. The explained architecture in this article has been developed for a project to research and create a Science, Technology, Engineering and Mathematics (STEM) related e-learning platform called Science, Technology, Engineering and Mathematics for the Young (STIMEY). The STIMEY [2] project is funded by the European Union and started in September 2016. The purpose of the platform is to interest young people to STEM and to bring teachers, students and their parents together to support STEM related learning and to increase the continent's international competitiveness in that regard.

This paper aims at:

- a concrete use-case of an microservice architecture, the STIMEY platform,
- a conception for migrating to Amazon Web Services (PaaS) is shown,
- a novel visualization of the microservice architecture is illustrated,
- a novel page fragments technology will be introduced, and
- an overview of database implementation in context of microservices is given.

The paper is divided in five sections, starting with a brief description of the STIMEY platform in Section 2. The following section proffers a literature review on microservices is provided in related works. The fourth section presents and discusses the architecture of the STIMEY platform that uses microservices as a basic building block. The section covers mainly the above listed

 ISBN: 978-1-61208-650-7

contributions. The paper then concludes with a conclusion in the last section.

II. THE STIMEY PLATFORM

This section first motivates and then gives a brief overview of the STIMEY platform.

The international competitiveness of Europe strongly depends on the availability of good educated engineers [3]. To get good educated engineers, the STEM education must be more relatable to European youths to raise their interests and involvement in STEM careers. This interest needs to be awoken already at the school. This is the aim of the STIMEY project. To reach this overall goal, STIMEY project proposes a multichannel hybrid e-learning platform for STEM-Education. The platform provides e-learning components that are designed and developed on the base of a well-researched pedagogical framework [4] that is to be developed in the STIMEY project. By doing so, STIMEY aims to lower the barrier of young people to consider a STEM career as an attractive career alternative.

The participation of multiple parties is needed, unified in the STIMEY platform to reach the overall goal of the STIMEY project. Consequently, within the STIMEY platform not only students come together with their teachers. Additionally, other stakeholders, like universities, schools, parents, business and media partners need to join together to reach the common goal and to make STEM a natural part of the daily life of youths. By doing so we hope to give students the feeling that what they do is important and valuable, and we open all stakeholders an ability to demonstrate their engagement for the STEM education [5].

One of the central ideas of the STIMEY project is to get close to the interests and social identity of the students. For this, the STIMEY is constructed to be a socially motivational platform that supports and motivates students towards STEM subjects.

To do so, the STIMEY contains components that the students are familiar with, like social media and games to stimulate the emotional and educational engagement. There also exists a gamification-oriented [6] reward system that enables students to earn badges. These components are there to help with the intrinsic motivation of the students towards STEM subjects.

For example, the integrated social media tools enable the students to communicate about the STEM topics or to help each other in making exercises. The integrated games in turn are constructed such that the students can learn during gaming. The STIMEY-robot is a socially-assistive learning buddy that supports students and gives them feedback. The idea is to effectively use robot fellow artefacts for pupils' emotional engagement, community bonding, efficient learning and motivation. Additionally, within the platform it will be possible to program the STIMEY-robot using a simple command set that is integrated in the STIMEY platform. The STIMEY-radio allows students to get on demand broadcasting programs about STEM subjects [7]. The integrated STIMEY e-portfolio provides the possibility for the students individually to collect information about courses and activities the student has participated in. Additionally, the STIMEY platform will contain entrepreneurial tools for engaging schoolchildren in innovations and stimulating their creative thinking. In order to integrate a wide array of existing educational tools, we have created in cooperation with SCIENTIX a mechanism to integrate existing tools such as solutions provided from SCIENTIX (Community for science education in Europe) in the STIMEY-platform.

The platform is constructed gender-neutral, so that boys and girls alike can identify with the components contained therein. The challenge of STIMEY is to reach even those students who are not enthusiastic about STEM. Especially girls are underrepresented in the STEM-sector. To be gender-inclusive means for STIMEY means that it supports a wide range of user behavior to provide low-threshold access to the platform.

The second central idea of the STIMEY project is to provide teachers the necessary modern tools to teach STEM in an attractive and engaging manner in-class or remotely, while also following up on students' progress. For this, the platform allows teachers to create their own courses and curse materials, to reuse the materials when creating new courses. Whether the content is more theoretical, or more practical and interactive, is determined by the teacher. The teachers are also able to reuse their created materials when creating new courses. Additionally, teachers may share their courses and course materials with other teachers. This supports the community building among teachers for example to share experiences in using the STIMEY in classes.

Other stakeholders, like parents and companies, are supported in suitable way. For example, parents are able to get information about the current state of the studies of their children or to communicate with their teachers and companies are able to add educational material into the platform.

For security reasons, it is considered to store personal data on different servers, to prevent developers and administrators who work on the platform, and possible attackers, to link data which is stored on the platform to personal information. Every user can determine who has access to his/her shared data.

The STIMEY components are based on a thorough research. The fundamental research focuses among others the following questions:

- Why STEM education is an issue today?
- Why STEM careers are not popular among young people?
- How to motivate and encourage young people to go for STEM careers? and
- What makes STEM unattractive or uninteresting and how positive aspects can be emphasized more?

To answer these questions, we are currently creating and implementing a novel pedagogical framework that exploits the full potentials of Social Media for STEM subjects in formal and informal contexts. Therefore, the major target groups and stakeholders are integrated into the development process. Additionally, to track the students' performance

 ISBN: 978-1-61208-650-7

progress, suitable measurements methods, algorithms and tool are under development

All components integrated in the STIMEY-platform serve the goal to raise the attractiveness of STEM subjects among European youth. From the European perspective, it is important that the STIMEY-platform harmonizes with the STEM education in European schools. This is important to reach a critical mass of end users who will be motivated for a STEM career.

III. RELATED WORK

Microservices is a young architectural pattern, which follows the idea of service-oriented computing (SOA). SOA started as an attempt to overcome the drawbacks of monolithic software architectures. It provides loosely coupled services, each usually focused on one specific business process. The services are reusable and could be put together dynamically to address issues in continually changing business environments [8]. Although software that follows SOA architecture can consist of several internal services, the whole product has to be deployed as one unit, leaving it still monolithic, as [9] point out. To maintain scalability, copies of the whole application have to be used, which is not efficient regarding memory consumption. While such architecture makes it easy to develop and deploy large software, it also makes it costly to modify and maintain these applications, because they are difficult to understand. [9]

These problems lead to the development of alternative architectures that aim to conform to modern needs by breaking down the application into smaller independent services [9]. In 2012, a group of software architects chose the term microservices as the most fitting one for the common architectures they had been examining in recent time [1]. Before that, the concept of microservices has been known under different names, e.g. Adrian Cockcroft at Netflix called it “Fine grained SOA” [11]. Dragoni et al. say that there is no common definition of microservices, and therefore the concept is still in its infancy [12]. However, principal features of microservices have been defined by M. Fowler and J. Lewis [11]. For instance, microservices based architectures consist of small independently upgradable, replaceable and reusable components that can be deployed separately. If the code of one microservice is altered, only the microservice itself is affected and must be redeployed, instead of needing the whole application to be deployed again. Another feature of microservices is decentralized governance. The used technology for implementing a specific microservice can be chosen independently. Therefore, different Microservices can use different kinds of databases. The code of microservices can also be written in different programming languages, as long as a common interface is used to handle the communication between the microservices [1]. Other authors like S. Newman, E. Wolff and A. Gupta built on the work of M. Fowler and J. Lewis to make concrete suggestions for conceptions and designs of architectures based on microservices [14][15][16][17]. Villamizar et al. [10] show a slight cost reduction by using microservices instead of a monolithic architecture. The average response time does not differ so much for the microservice architecture (although there is an overhead, through additional server instances). [10] Microservices were introduced in a lot of large companies, to overcome their growing needs of more scalability. For example, Google, Amazon, Microsoft, Netflix and Zalando have successfully introduced microservices for their business [10][13].

A systematic literature research has been conducted by Pahl and Jamshidi to collect and review the existing research on microservices. They discovered that a great amount of research about microservices is only theoretical and just a minor part of studies deals with actual technological solutions [18], leaving a research gap in that department.

One possible use case of microservices is web-based e-learning platforms like the STIMEY platform. D. Chandran and S. Kempegowda [17] state that irrespective of free or commercial products, the implementation of the hardware and software infrastructure are highly cost intensive. Although free e-learning software like Moodle is free to use, the high cost of installation, maintenance, as well as the high learning curve, could make them even more expensive then commercial products. In addition, existing e-learning solutions are often unable to collaborate with other educational facilities or dynamically scale the application. Also, the integration with other systems is expensive due to the proprietary nature of the existing e-learning solutions. Another issue is that in many cases the available hardware is not suitable for web 2.0 applications. D. Chandran and S. Kempegowda therefore propose a cloud-based solution for three defined scenarios to overcome these disadvantages [19]. With cloud-computing free resources can be allocated efficiently, thus providing reduced energy consumption and less costs.

Fazakas et al. [20] propose a technical solution for a collaborative e-learning platform based on microservices. The platform aims at making teaching and learning objectives easier by implementing loosely-coupled software modules. These modules provide synchronous learning functionalities like video communication or text-based chats, and asynchronous functionalities like multimedia authoring.

Martin et al. [21] describe and compare the two different approaches virtual machines and containers for the realization of cloud-based computing. Virtual machines are therefore fully functional operating systems that are running on top of a virtual hardware layer. The advantages of virtual machines are that they can be installed very quickly and booted within a few seconds. They also can be cloned and stacked with centralized tools. Disadvantages are that the additional layers for emulated hardware and operating system come with a significant overhead in performance. In contrast, containers come with performance near that of single-tenant physical servers. In addition, with containers, multiple instances of applications can be run on one single machine. Containers are also very lightweight, because of the absence of system libraries, which makes the boot-process very fast, and makes it possible to create and move containers almost instantly. This enables high scalability [21][22]. Docker is the most used container technology. Based on that it exists also a Kubernetes [23] (orchestrator of containers) solution, former provided by Google (see also

 ISBN: 978-1-61208-650-7

Borg [24], predecessor to Kubernetes). An alternative to Kubernetes is Docker Swarm.

Another paradigm that supports large-scale cloud computing applications is serverless computing or function as a service (FaaS) that is an event-driven approach, utilizing lightweight processes that react to an event. The first service that follows serverless computing is AWS Lambda by Amazon Web Services [25]. For example, Google and Microsoft provide serverless computing services. [26][27]

An evaluation of different cloud computing services has been made by McGrath et al., showing the potential of the underlying technology on two use cases. For instance, systems using serverless computing are more scalable and flexible than previous technologies. [28] Using of AWS Lambda can reduce costs significantly, by 50-60% instead of using microservices [29].

IV. ARCHITECTURE OF THE STIMEY PLATFORM

It was decided to use microservices for the entire platform, which allows for development to take place in small separate teams. Microservices are autonomous, isolated services that are loosely coupled with other services. In general, every microservice has its own data storage (i.e., database, so redundancies are accepted). However, sharing the database may be appropriate for simplification. The data models must be kept separate from each other. Microservices can, for example, communicate via REST or WebSockets but they can also have a web interface. Furthermore, microservices are easily scalable, if they are stateless (no use of sessions). All required data must be loaded from the cache or the database directly.

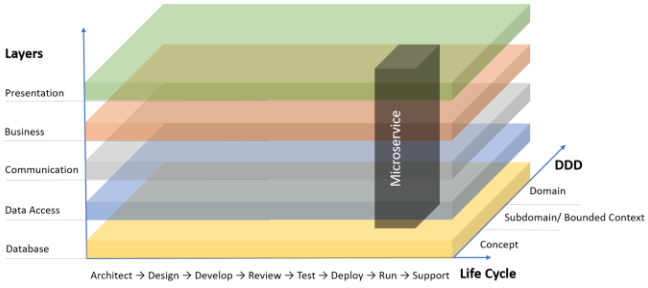

Figure 1. Microservice Architecture Model.

In Figure 1 a novel visualization of the microservice architecture model is presented. In this model microservices may implement multiple layers. Normally, a microservice includes the business logic layer (inclusively offering of services to the outside) and the data access logic to the database (persistence). The communication layer as described above can be implemented using web protocols. In addition, a presentation layer may be included (HTML, JSON and XML). DDD stands for Domain Driven Design by Eric Evans ([30][31]). A concept is the smallest unit within a subdomain (categorization of related concepts). A bounded context is a subset that can be spanned in multiple subdomains. A domain can contain several functional subdomains. The life cycle consists of steps that a microservice goes through: the beginning design, development, testing and production. Maintenance and support are the last stage, or the microservice will be replaced by a better solution and the life cycle starts again. The Reference Architecture Model Industry 4.0 (RAMI4.0) [32] had significant influence in creating Figure 1, which depicts the Microservice architecture model.

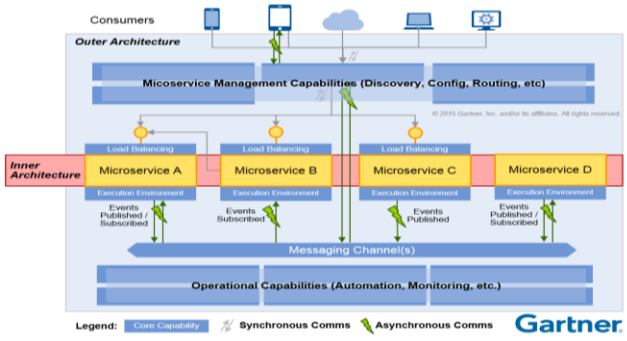

Figure 2. Inner and outer architecture of microservices. [33]

According to Olliffe [33], the platform's internal and external architecture is described in connection with microservices (see Figure 2). The external architecture includes, among other things, tools for configuration, routing, discovery of other microservices, monitoring, and automation (deployment). The microservices themselves are described in the internal architecture and are developed independently from each other. The communication among them is best done asynchronously, while there may be multiple instances of one microservice. A load balancer can be connected to all individual microservices to improve the distribution of workloads, for which the round-robin strategy is commonly used. State changes or notifications can be exchanged by microservices via the so-called messaging channels.

Every microservice on STIMEY platform belongs to one domain, whereas a domain consists of related topics. Each microservice is assigned to one responsible team. All microservices make use of the same technology, for better maintenance. For one, it is easier for the small core team to cope with only one technology. For another, there is a high fluctuance of additional team members (internships, project work, bachelor and master thesis), who can be trained faster that way. One advantage of microservices is that they are fine granular, so they can be easily scaled. There is also no single point of failure, because there are redundant instances for each microservice, and all microservices are not deployed together, like it is the case with a monolithic architecture. That way, older versions of microservices can be replaced during runtime.

A. *Use of Microservices in the STIMEY Platform*

The STIMEY platform in the actual version is divided into several independently developed domains (here: middle-tier, see Figure 3) that are researched, identified and detailed in previous research [4], such as:

- User profile: a visual display of the personal data associated with a specific user in the platform [34].

 ISBN: 978-1-61208-650-7

- Activity stream: typically, a dashboard, where tracked activities or events relevant to a user, community, topic or anything in the platform it's built in [35].
- Community: an online space where individuals can feel part of a group and interact on a common topic or interest by creating posts, commenting, reading in such interest- and niche-specific forums [36].
- Social messaging: refers to the exchange of text messages through a chat tool in real-time [37].
- Online-Course: a supervised learning option for web browsers or mobile applications, that consists of a series of lessons [38].

The communication between microservices works by either calling each other's REST-API, or by messaging (Publish-subscribe), whereas subscribers will be notified by the occurrence of new messages belonging to the subscribed topic. For instance, the gamification microservice (topic: "gamificationstream") must be informed by rewardable activities from other microservices to improve the motivation and experience of the users. Also, for the activity stream, it is necessary to interact with other microservices to access their data which is necessary to calculate the ranking, so that activities or events relevant to the user can be shown in descending order corresponding to the given rank, starting by the highest rank for an activity or event.

The backend includes the microservices for the data access layer (stored files, cache, messaging system, and database for persisting data). The main microservice is part of the frontend, especially for login and sign-up and as a relaying layer (implemented as a proxy) to the underlying middle-tier microservices. It also contains the superordinate web pages of the STIMEY platform (using the technology of page fragments) where the individual microservices are embedded. These should be scalable for further development. Within the microservices, no classic server-based session data is held (data access is via the backend). It should also be noted that a REST-API exists for the platform/robot communication. Furthermore, the platform-to-robot (P2R) communication can be established by an Message Queue Telemetry Transport (MQTT) - Adapter (robot has subscribed on specific topic), which is an integrated part of the messaging service.

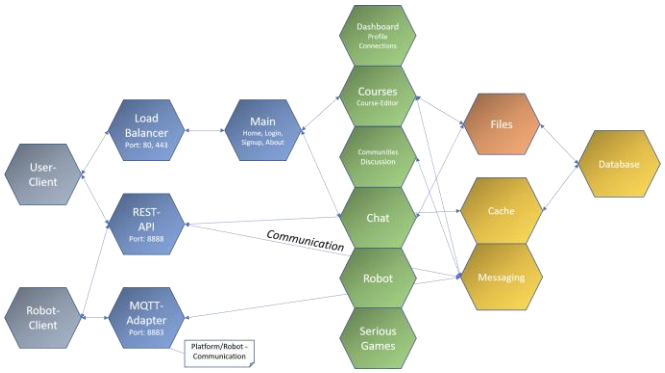

Figure 3. Actual STIMEY platform architecture.

During the login process in the main microservice, an access token is stored as a cookie on the user's browser, which is used for subsequent authentication. Furthermore, a language cookie is also stored for the purpose of multi-language support.

The microservices are deployed in Docker containers, running on CoreOS (Container Linux). This ensures an increased failure safety, as the containers can be restarted accordingly in the event of a failure. The entropy of the system, and hence the error rate, is reduced since the same conditions are always present for the system (in the form of containers, operating system virtualization). There exists also a Kubernetes solution, called Tectonic, which comes from the CoreOS team as well.

This microservice architecture was chosen at first, to follow a different approach for research, instead of the classical approach (mainly over REST-API's). The main microservice uses the page fragments technology, which is explained more in detail in the Section B. Different from the classical approach the microservices provide also their own web pages in the presentation layer. This avoids the maintenance of all web pages in a centralistic way. The main microservice aggregates them in a superordinate context and sends the final web page to the client. In general, the page fragments technology works as intended, but it has to be extended, or further research is necessary, to find a way how it works more smoothly. In the current situation, the whole site will be newly rendered when the page is aggregated, including header, footer and side-menu. A possible solution could be to update only the changing fragment. This has to be further considered.

At the moment, we are planning to migrate to our new architecture model (Figure 4), that is similar to the classical approach of a microservice architecture. The development team recognized that it is more efficient to have the web pages on the same place, because there is an imbalance of developers to designers (much more developers). It is considered as ideal to have interdisciplinary teams working on microservices [15]. One is responsible for the frontend, one for the backend and one for the overall design. We currently do only server-side rendering (using Spring MVC, Model-View-Controller pattern) and plan to switch to client-side rendering (using Vue.js and React.js). This could at least reduce server load. Of course, this also has the disadvantage for the client that it has the costs of processing. One important aspect of the future of the STIMEY platform is also, to make it long-term production ready, as it is only a prototype as of now. This enables the possibility to migrate the STIMEY platform to a PaaS in the future. This reduces the risks involved with self-realizing a microservice architecture. PaaS from established vendors are much more optimized in scalability, resilience, managing, efficiency and uptime. The long-term costs can be reduced, because not so much manpower is needed, and vendors can organize the distribution of their service more efficiently over the world.

The main difference between the old and new architecture (Figure 4) is that with the new architecture only REST-API microservices are used for the communication of the corresponding domains, without providing additional

 ISBN: 978-1-61208-650-7

web pages for each microservice. The REST-Service can be called by the REST-API-Gateway that can orchestrate other services or proxies them to the right target. Instead of a main microservice that assembles page fragments coming from different microservices, there is now a storage microservice that holds all the web pages. All teams are working on this microservice and the corresponding REST-API microservice, for which they are responsible. Additionally, there can be used a FaaS service [39] optionally, which is using Vert.x (a Java alternative to Node.js) in combination with actor4j [40] (an actor-oriented framework). The advantage, in comparison to REST-Services, is that they are finer granulated and light weighted. That way, thousands of stateless functions (or actors) can be deployed easily, in contrast to REST-API microservices, which are heavy weighted. Normally, functions are isolated in containers [41], for security and resilience reasons. In an own maintaining FaaS service this is not mandatory, because the underlying functions can be trusted. It would be otherwise, if different contributors were involved. This results in further performance advances. The serious games microservice is using the Apache web server and PHP and is developed from our partner from Belarus. The REST-API-Gateway can be monitored for generating statistics and further analysis of collected data.

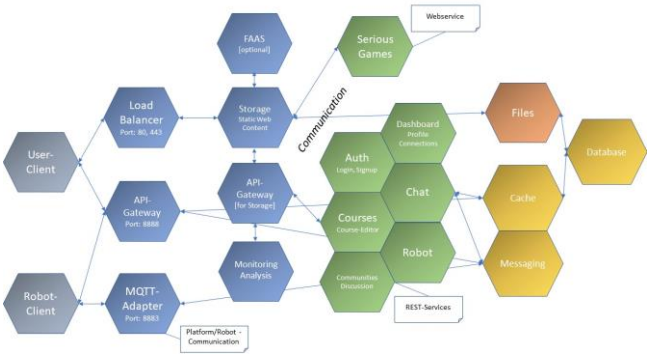

Figure 4. Planned STIMEY platform architecture.

The above described architecture, without page fragments technology, is very suitable to a possible migration to Amazon Web Services (Figure 5). The storage in Figure 4 corresponds to Amazon S3 in Figure 5. In both architectures, there is an API-Gateway for handling according requests sent from the client. Amazon Web Services has an authentification service called Amazon Cognito. This was implemented in the architecture shown in Figure 4 as an independet REST-Service called Auth. The API-Gateway can be monitored in both concepts (in Amazon called CloudWatch). Underlying services can use caching systems, also combined with data storage. It is also possible to use a publish-subscribe system, e.g. for the intercommunication of the services. In Amazon, there are also the Lambda Functions that are more cost-effective than API-calls shown by Villamizar [29]. Our REST services, as seen in Figure 4, are deployed in Docker Containers, so they can be easily migrated to Amazon ECS (Elastic Container Service) later. FaaS has its advantages, but it still has to be proven in practice, whether it is worthwhile to use it with accompanying greater complexity (thousands of light weighted functions) [42].

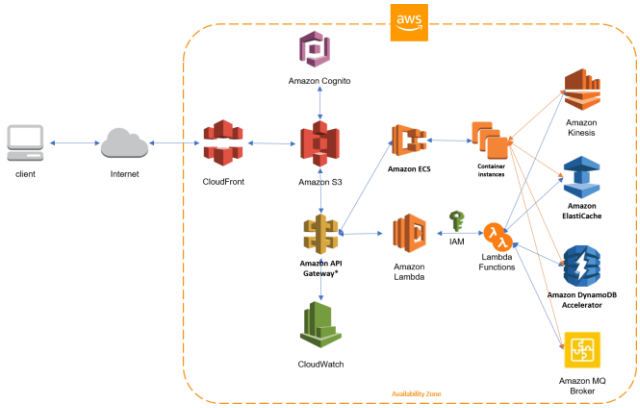

Figure 5. Conception of STIMEY platform architecture with Amazon AWS.

From the beginning, it was not planned to use Amazon Web Services or other PaaS, such as Microsoft Azure and Google Cloud Platform. For some companies, for privacy reasons and protection of intellectual property, an on-premise solution is more preferable. One of the requirements is to use only open-source software to keep the costs low. We decided to use Java as a core-language, because it is widely used and taught on universities. As mentioned before, it is difficult to realize and to maintain an own solution of a microservice architecture, especially in the long-term perspective. In contrast to that, there are open-source solutions like Serverless Framework [43], which makes it easy to switch between different cloud-computing providers (Azure Functions, AWS Lambda, Google Cloud Functions). There are also different providers for container solutions, like Google Kubernetes Engine or, like mentioned above, Amazon ECS and Azure Container Service (also over Kubernetes). "Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications." [23] At the current stage of prototyping, an on-premise approach is for us the best solution to fulfill the above mentioned requirements and initial goals.

B. Use of Page Fragments Technology in the STIMEY Platform

A microservice that uses the page fragments technology can integrate outputs of microservices in a superordinate defined web page.

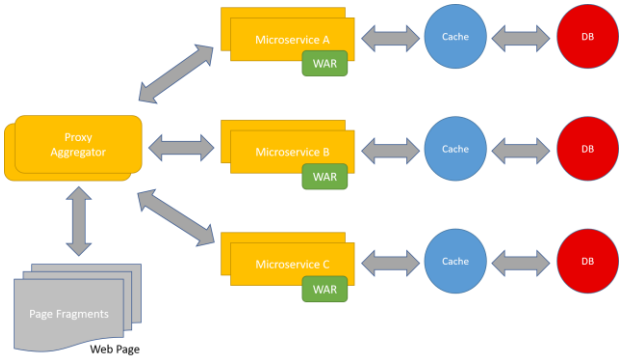

Figure 6. Proxy-Aggregator Design Pattern for page fragments technology, adapted from Gupta. (adapted to [44]and [16])

Here, a proxy server is used that aggregates the individual page fragments (Proxy-Aggregator Design Pattern, see Figure 6), that are references to a microservice that is embedded in the superordinate web page. According to Gupta [16][44], this pattern can be seen as a mixture of the Proxy Microservice Design Pattern and the Aggregator Microservice Design Pattern. It differs only in that way that the microservices are not only REST endpoints. Instead, they are microservices which have also their own web pages (the so-called page fragments) in the presentation layer.

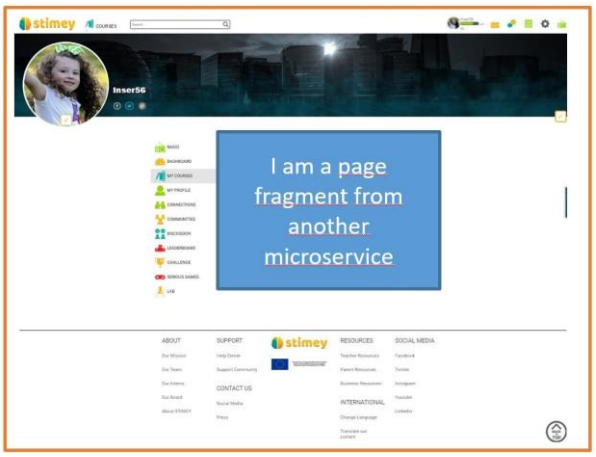

Figure 7. Illustrated example to our platform, showing an embedded microservice.

Figure 7 displays an example based on our future platform, whereas the content of a microservice is embedded in the superordinate web page.

The implementation [45] of the page fragments technolgy is a developed derivative of Smiley's HTTP-Proxy-Servlet (maintained by the MITRE Corporation). Zalando has a similar concept [46] and [47], where fragments (parts of a microservice) where developed by diverse teams (see Figure 8). The "layout service assembles the fragments and streams them to the client" [46]; see also BigPipe [48] by Facebook mentioned in [47].

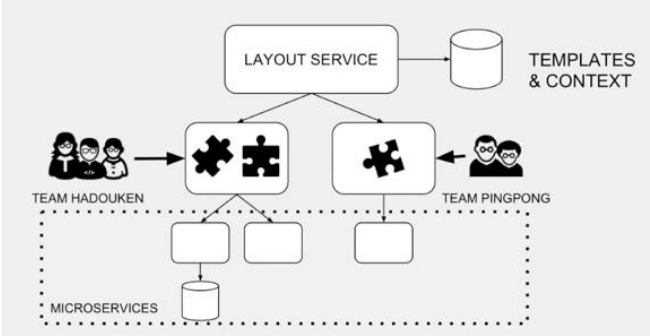

Figure 8. Layout service for front-end microservices [46][47]

C. *Use of Caching/Volatile caching and messaging in the STIMEY platform*

In the previous architecture, for caching/volatile caching and messaging the library was divided into two parts. Either an actor-oriented approach (Actor4j [40] comparable to Akka), or a classic implementation could be used (Redis as cache, NoSQL for persistence, RabbitMQ for messaging). The patterns: "non-volatile caching", "volatile caching", and "messaging", offer the possibility to use messaging channels (state changes and notifications) via appropriate interfaces. The messaging pattern is a special pattern for chatting and message-based content. Only the caching and messaging patterns allow persistence of content.

D. *Design of Database Structure*

As shown in Figure 9, the model is split into different domains. The new database model is a separate model tailored to the needs of individual microservices (largely independent database models). A document-oriented NoSQL database is used as a database. This ensures efficient data processing even with large amounts of data. Known document-oriented NoSQL databases are, for example, MongoDB and CouchDB. A major advantage is the simple serialization / deserialization of the data binding objects into a database document and vice versa (document is stored in the form of a readable JSON string). For the documentation, it is helpful to not only display persistent data structures, but also data structures in the cache, because the data structure of the database is not necessarily the same as the data structure of the cache. Furthermore, the data flow between client and server should be shown in the superordinate context, because the data depends on external triggers. Because of the complexity of the microservice architecture, an extended view is necessary that includes all data structures and the corresponding processes. The whole system is developed in an agile environment so the data structures (and their underlying processes) are subject to evolutionary development processes.

 ISBN: 978-1-61208-650-7

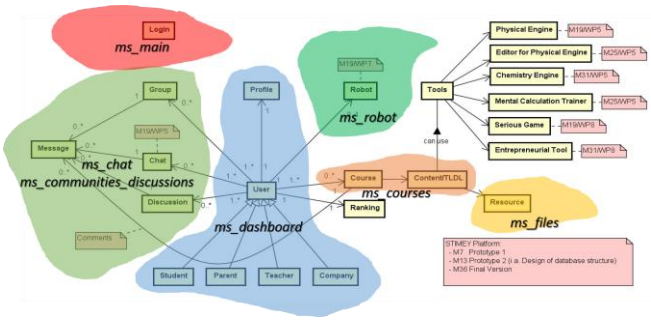

Figure 9. General database structure illustrating microservices affiliation.

V. CONCLUSION

This paper aims at contributing an example of how microservice architecture can be used to build a web platform in practice. This goal has been achieved by introducing the STIMEY platform as a use-case for microservice architecture, and how the architecture has been structured to fulfill the requirements. In addition, the first conception of a future architecture is shown, that builds on the lessons the development team has learned from the work on the current architecture.

The STIMEY platform is divided in several domains, each focused on one specific task, such as user profile, activity streams, and communities. With the microservices approach, each domain is realized by its own microservice. The current architecture of the STIMEY platform has a main microservice as the central hub for server-client communication.

In the current architecture, the main microservice uses the page fragments technology to assemble the outputs of other microservices to a resulting web page and streams them to the client. However, it turned out to be more efficient to have the web pages on one place. For one, the designers can work on only one microservice. For another, the concept of separation of concerns is followed more strictly that way, which makes it easier for the software developers to divide their work. Now they can work in pairs on a domain. One developer is responsible for the REST-API, and the other developer is responsible for the implementation of the front-end with client-side-rendering.

No session data is stored on the server-side, but an access token is stored as a cookie in the client-side browser. To increase failure-safety, docker containers are used for deployment of the microservices. The current system does server-side rendering, using Spring MVC. To reduce server-side cost, it is planned to use client-side rendering with the utilization of Vue.js and React.js in the future. The current architecture uses REST-API for the communication between microservices.

For future-proof scalability, resilience, and efficiency, it is also planned to use FaaS. This is going to make the microservices more lightweight, and to increase the performance significantly. The shift to the new architecture is also suitable for the usage of cloud services like Amazon Web Services that could be used in the future as an option.

The STIMEY platform supports volatile caching, non-volatile caching, and messaging, whereas non-volatile caching and messaging enable persistent data storage.

To fulfill the needs of particular microservices, largely independent database models are used, each modeled specifically to meet the requirements of their corresponding microservice. To allow efficient processing of data, a document-oriented NoSQL database is used.

It is planned to develop testing scenarios in the future to cater with multiple test-cases. Also, a case study that is focused on the research questions introduced in section II is going to be conducted.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement Nº 709515 — STIMEY.

REFERENCES

[1] D. Namiot and M. Sneps-Sneppe, "On Micro-services Architecture," International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no. 9, 2014.

[2] STIMEY, "Grant Agreement," no. 709515, 2015.

[3] European Union, "EU STEM Coalition: STEM Skills for a Future-Proof Europe," April 2016.

[4] M. Assaad and T. Mäkelä, "Integrating Social Media Concepts as Tools in a Pedagogical Approach for a Technology-enhanced Learning Environment," in K. Daimi and S. Semenov (Eds.), *AICT 2017: The Thirteenth Advanced International Conference on Telecommunications* (pp. 67-73), IARIA, 2017, [Online]. Available from: http://www.thinkmind.org/download.php?articleid=aict_2017_4_30_10061 [retrieved: June, 2018]

[5] M. Assaad, J. Mäkiö, T. Mäkelä, M. Kankaanranta, N. Fachantidis, V. Dagdilelis, A. Reid, C. Rioja del Rio, E. V. Pavlysh, and S. V. Piashkun, "Attracting European Youths to STEM Education and Careers: A Pedagogical Approach to a Hybrid Learning Environment," World Academy of Science, Engineering and Technology International Journal of Educational and Pedagogical Sciences, vol. 11, no. 10, 2017

[6] M. Assaad and S. Shi, "Using the Thematic Approach in Integration with Social Media and Gamification for Concept Design in a Hybrid STEM Learning Environment," 1st International Conference on Educational Technology, 2017.

[7] A. Reid, J. Mäkiö, R. Serrano, and C. Rioja Del Rio, "Conventional radio, the latest motivation to learn science among European youth," International Conference on Education and New Learning Technologies, March 2017

[8] L. Ismail, D. Hagimont, and J. Mossi'ere, "Evaluation of the mobile agents technology: Comparison with the client/server paradigm," Information Science and Technology (IST), vol. 19, 2000.

[9] D. Namiot and M. Sneps-Sneppe, "On micro-services architecture," International Journal of Open Information Technologies, vol. 2, no. 9, 2014.

[10] M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," 2015 10th Computing Colombian Conference (10CCC), Bogota, pp. 583-590, 2015.

[11] M. Fowler and J. Lewis, "Microservices," 2014, [Online]. Available from: http://martinfowler.com/articles/microservices.html [retrieved: June, 2018]

[12] N. et al. Dragoni, “Microservices: Yesterday, Today, and Tomorrow,” in M. Mazzara, B. Meyer (eds) Present and Ulterior Software Engineering, Springer, Cham, 2017.

[13] T. Mauro, “Adopting microservices at netflix: Lessons for team and process design,” 2015, [Online]. Available from: http://nginx.com/blog/adopting-microservices-at-netflix-lessons-for-team-and-process-design/ [retrieved: June, 2018]

[14] S. Newman, “Building Microservices,” O’Reilly Media, Inc, 2015.

[15] E. Wolff, “Microservices,” dpunkt.verlag GmbH, 2016.

[16] A. Gupta. "Getting Started with Microservices," DZone refcardz, Refcard #215, Former version.

[17] V. Reynolds and A. Gupta. "Getting Started with Microservices. Design Patterns for Decomposing the Monolith," DZone refcardz, Refcard #215, [Online]. Available from: https://dzone.com/refcardz/getting-started-with-microservices [retrieved: June, 2018]

[18] C. Pahl and P. Jamshidi, “Microservices: A Systematic Mapping Study,” 2016, [Online]. Available from: https://www.researchgate.net/publication/302973857_Microservices_A_Systematic_Mapping_Study [retrieved: June, 2018]

[19] D. Chandran and S. Kempegowda, "Hybrid E-learning platform based on cloud architecture model: A proposal," 2010 International Conference on Signal and Image Processing, Chennai, pp. 534-537, 2010.

[20] B. P. Fazakas, O. C. Iuonas, C. Porumb, and B. Iancu, "Collaborative learning tools for formal and informal engineering education," 2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet), Targu Mures, pp. 1-6, 2017.

[21] A. Martin, S. Raponi, T. Combe, and R. Di Pietro, “Docker ecosystem – Vulnerability Analysis,” Computer Communications, vol. 122, pp. 30-43, 2018.

[22] V. Singh and S. K. Peddoju, "Container-based microservice architecture for cloud applications," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, pp. 847-852, 2017

[23] Kubernetes, [Online]. Available from: https://kubernetes.io/ [retrieved: June, 2018].

[24] Kubernetes, Kubernetes Blog, “Borg: The Predecessor to Kubernetes,” 2015, [Online]. Available from: https://kubernetes.io/blog/2015/04/borg-predecessor-to-kubernetes [retrieved: June, 2018]

[25] Amazon, “Amazon Lambda (AWS Lambda),” [Online]. Available from: https://aws.amazon.com/lambda/ [retrieved: June, 2018]

[26] Google, “Google Cloud Functions,” [Online]. Available from: https://cloud.google.com/functions/ [retrieved: June, 2018].

[27] Microsoft, “Microsoft Azure Functions,” [Online]. Available from: https://azure.microsoft.com/en-in/services/functions/ [retrieved, June, 2018].

[28] G. Mcgrath, J. Short, S. Ennis, and B. Judson, P. Brenner, “Cloud event programming paradigms: Applications and analysis,” in: 2016 IEEE 9th International Conference on Cloud Computing, CLOUD, pp. 400–406, 2016.

[29] M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, pp. 179-182, 2016.

[30] Eric J. Evans, “Domain-Driven Design. Tackling Complexity in the Heart of Software,” Addison Wesley, 2003.

[31] InfoQ 2006, “Domain Driven Design Quickly. C4Media,“ [Online]. Available from: https://www.infoq.com/minibooks/domain-driven-design-quickly [retrieved: June, 2018]

[32] DIN, "DIN SPEC 91345. Reference Architecture Model Industrie 4.0 (RAMI4.0), " 2016.

[33] G. Olliffe, “Microservices: Building Services with the Guts on the Outside,” 2015, [Online]. Available from: http://blogs.gartner.com/gary-olliffe/2015/01/30/microservices-guts-on-the-outside [retrieved: June, 2018]

[34] Wikimedia Foundation Inc, “User profile,” [Online]. Available from: https://en.wikipedia.org/wiki/User_profile [retrieved: June, 2018]

[35] Gartner IT Glossary, “Gartner IT Glossary: Activity Stream,” 2012, [Online]. Available from: http://www.gartner.com/it-glossary/activity-stream/ [retrieved: June, 2018]

[36] H. Baxter, “An Introduction to Online Communities,” [Online]. Available from: http://www.providersedge.com/docs/km_articles/An_Introduction_to_Online_Communities.pdf [retrieved: June, 2018]

[37] M. Rouse, “instant messaging (IM or IM-ing or AIM)” SearchUnifiedCommunications, [Online]. Available from: http://searchunifiedcommunications.techtarget.com/definition/instant-messaging [retrieved: June, 2018]

[38] M. Kerres, "Mediendidaktik. Konzeption und Entwicklung mediengestützter Lernangebote," Oldenburg Verlag, 2016.

[39] D. A. Bauer, "Template for using Vert.x and Actor4j (inclusiveley as FaaS)," 2017, [Online]. Available from: https://github.com/relvaner/relvaner-vertx-template [retrived, June, 2018]

[40] D. A. Bauer, “Actor4j an actor implementation,” 2017, [Online]. Available from: https://github.com/relvaner/actor4j-core [retrieved: June, 2018]

[41] 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, pp. 405-410, 2017.

[42] M. Asay. "Why AWS Lambda and serverless computing won't kill Docker in the enterprise. AWS Lambda and Docker containers may be at odds, but both have a place in a modern enterprise," in TechRepublic, 2017. [Online]. Available from: https://www.techrepublic.com/article/why-aws-lambda-and-serverless-computing-wont-kill-docker-in-the-enterprise/ [retrieved: June, 2018]

[43] Serverless Framework, [Online]. Available from: https://github.com/serverless/serverless [retrieved: June, 2018]

[44] A. Gupta, “Microservice Design Patterns,” 2015, [Online]. Available from: http://blog.arungupta.me/microservice-design-patterns/ [retrieved: June, 2018]

[45] D. A. Bauer, "Derivative of Smiley's HTTP-Proxy-Servlet," 2017, [Online]. Available from: https://github.com/relvaner/HTTP-Proxy-Servlet [retrived: June, 2018]

[46] R. Schaefer, "From Monolith to Microservices at Zalando", in GOTO Conferences, 2016. [Online]. Available from: https://www.youtube.com/watch?v=gEeHZwjwehs [retrieved: June, 2018]

[47] Zalando, "A streaming layout service for front-end microservices," [Online]. Available from: https://github.com/zalando/tailor [retrieved: June, 2018]

[48] C. Jiang, "BigPipe: Pipelining web pages for high performance," in Facebook, Facebook Engineering, 2010, [Online]. Available from: https://www.facebook.com/notes/facebook-engineering/bigpipe-pipelining-web-pages-for-high-performance/389414033919/ [retrieved: June, 2018]

 ISBN: 978-1-61208-650-7