applied sciences

Article

Action Research Approach to Analysis of Teaching of Blockchain Web 3.0 Application Based on MACH Architecture

Gokmen Katipoglu [1], Semih Utku [2], Ivan Mijailović [3], Edis Mekić [4,*], Dženan Avdić [4] and Petar Milić [5]

1 Institute of Science, Dokuz Eylul University, Tinaztepe Kampusu, Buca, Izmir 35390, Türkiye; 2022900626@ogr.deu.edu.tr
2 Department of Computer Engineering, Faculty of Engineering, Dokuz Eylul University, Tinaztepe Kampusu, Buca, Izmir 35390, Türkiye; semih@cs.deu.edu.tr
3 Faculty of Occupational Safety, University of Niš, Čarnojevića 10a, 18000 Niš, Serbia; ivan.mijailovic@znrfak.ni.ac.rs
4 Department of Technical Sciences, State University of Novi Pazar, Vuka Karadzica 9, 36300 Novi Pazar, Serbia; dzavdic@np.ac.rs
5 Faculty of Technical Sciences, University of Priština, Kosovska Mitrovica, Kneza Miloša 7, 38227 Kosovska Mitrovica, Serbia; petar.milic@pr.ac.rs
* Correspondence: emekic@np.ac.rs; Tel.: +381-63-264-095

Abstract: This study investigates the integration of agile methodologies, particularly Scrum, into the teaching of microservices, API-first, cloud-native, and headless (MACH) architecture within a university setting. Using an action research framework, we see the impact of agile approaches on student learning and the practical application of MACH principles. The findings reveal that agile methodologies not only enhance students' technical proficiency, but also foster collaborative and iterative learning environments that simulate real-world software development. Over a 15-week course, students applied agile techniques to design, build, and deploy microservices-based systems, benefiting from structured yet adaptable sprints that broke complex tasks into manageable stages. Quantitative assessments showed substantial improvements in knowledge and confidence, while qualitative feedback emphasized the hands-on, project-based learning's alignment with industry practices. This research underscores the potential of agile frameworks to enhance education in emerging software architectures, offering insights into how higher education can align more closely with the dynamic needs of the software industry.

Keywords: agile methodologies; scrum; MACH architecture; microservices; API-first; cloud-native

Citation: Katipoglu, G.; Utku, S.; Mijailović, I.; Mekić, E.; Avdić, D.; Milić, P. Action Research Approach to Analysis of Teaching of Blockchain Web 3.0 Application Based on MACH Architecture. *Appl. Sci.* **2024**, *14*, 11158. https://doi.org/10.3390/app142311158

Academic Editor: Gianluca Lax

Received: 1 November 2024
Revised: 24 November 2024
Accepted: 26 November 2024
Published: 29 November 2024

1. Introduction

The rapid evolution of digital technology led to a profound shift in how software architectures have been designed, developed, and deployed. Starting from the monolithic architecture, where all components of the software were unified in a single application, the increasing demands and complexity of software required new solutions [1]. This approach was sufficient for standalone, self-sustaining computer systems, but the development of large-scale software systems required software capable of meeting new demands. Thus, this new requirement became the cornerstone for the development of service-oriented architecture (SOA). SOA emphasized services as independent units of functionality that communicate over a network [2,3]. While previously mentioned design SOA was succeeded with other architectural approaches, it again remerged with the development of mobile applications [4]. This paved the way for the development of microservice architecture, where software applications were composed of small, independent services. Each developed service solved a specific demand [5].

Another important point of interest is APIs, or application programming interfaces, which have been around almost as long as modern computing. They emerged decades ago to enable communication between disparate software applications. Based on them, software engineers developed the API-first approach, which prioritizes APIs at the beginning of the software development process, positioning APIs as the building blocks of software. API-first organizations develop APIs before writing other code instead of treating them as afterthoughts. This allows teams to construct applications with internal and external services that are delivered through APIs [6,7].

Cloud native is the software approach of building, deploying, and managing modern applications in cloud computing environments. Modern companies want to build highly scalable, flexible, and resilient applications that they can update quickly to meet customer demands [8].

To do so, they use modern tools and techniques that inherently support application development on cloud infrastructure. These cloud-native technologies support fast and frequent changes to applications without impacting service delivery, providing adopters with an innovative, competitive advantage [9].

The final piece of modern architectural approaches is headless software. The main reason for the development of this software was to decouple the user interface (UI) from the back-end services. As such, headless software runs without a graphical user interface (GUI). The GUI is the part of the software that the user interacts with, so a headless application may not always include a way for the user to directly input data or view results [10].

All this led to one of the most transformative developments in recent years, the microservices, API-first, cloud-native, and headless (MACH) architecture, which combines all previous architecture approaches for highly flexible, scalable, and efficient software systems. By decoupling the frontend from backend services and enabling microservices to operate independently, MACH architecture facilitates the development of modular systems that can rapidly adapt to changing business needs.

In parallel, agile frameworks emerged as a dominant methodology for managing software development. An agile iterative and flexible approach is well-suited to addressing the complexities inherent in modern software architecture, particularly in educational settings [11].

The integration of agile frameworks in academic curricula has shown the enhancement of student engagement, fosters hands-on learning, and better prepares students for the dynamic requirements of the software industry [12,13]. By breaking down complex tasks into manageable sprints, students can iteratively develop and refine their skills in real-world problem-solving environments.

This research aimed to explore the intersection of MACH architecture and agile methodologies in an educational context. Specifically, it investigated how agile frameworks could be applied to teach MACH architecture effectively to university students.

The significance of this research lies in its potential to inform curriculum design in software engineering education, aligning it with industry demands for graduates who are proficient in both agile methods and cutting-edge software architectures such as MACH.

Moreover, it contributes to the existing literature on educational methodologies by providing insights into how agile frameworks can be effectively integrated into higher education to enhance the teaching of emerging architectures such as MACH.

The paper is organized as follows: The first section describes the research design, followed by an explanation of the research methodology and its implementation. Afterward, the results and discussion are presented. Finally, the conclusion and directions for future work are provided at the end of the paper.

2. Research Design

Action research [14,15], a participative and iterative method that is appropriate for problem-solving research, was used for the research presented in this paper. This approach was chosen as its primary objective was to enhance the capability and practice of the researcher; in our case, teachers and students will fulfill the role of researchers, rather than generate theoretical knowledge [16,17]. Action research is inherently different from traditional research. Traditional approaches assume that knowledge is objective if the world can be known, and one can observe cause–effect relationships in phenomena exposed by implementation of empirical and logical methods. It includes randomized controlled trials, quantitative data from random samples, and statistical analysis to ensure findings are valid and reliable. Traditional research is conducted by people who are not part of the phenomenon under study, whereas action research is conducted by those who are part of the phenomenon [18].

While action research is a well-established research methodology, the primary reason for applying it in this study is its connection and compatibility with Industry 4.0. Action research is commonly used as a tool to understand processes in industry [19], to validate different collaboration methods in the software engineering industry [20], and to analyze and propose new solutions for Industry 4.0 policies [21]. Consequently, by utilizing this approach, the aim was to build an educational system that replicates the real application of knowledge in a domain such as the working space [22]. Therefore, in the final instance, it creates additional levels of knowledge for teachers and students alike.

A mixed-method research design was employed to combine quantitative and qualitative methods for the validation of research results. The approach offers a more comprehensive understanding of the implementation and its outcomes. The quantitative data for statistical analysis of performance metrics were used as well as qualitative data to understand student experiences and perspectives.

Finally, we will present a case study of a successfully developed application by students and the successful implementation of all aspects of MACH architectural design.

3. Research Methodology

Action research is a participatory and iterative method that focuses on solving practical problems through planning, acting, observing, and reflecting cycles [23].

The first of the action phases involves identifying a practical problem within the context and collaboratively defining its scope. This phase requires the engagement of all participants (researchers and stakeholders) to ensure mutual understanding and agreement. For instance, the problem is not only identified, but also deeply analyzed to design an actionable plan [24]. The action research framework emphasizes inclusive involvement in defining the goals and gathering baseline data to assess the current state [25]. Tools such as focus groups or semi-structured interviews can be employed during this phase to collect diverse perspectives.

During the second phase, the planned interventions or actions are executed. This phase is characterized by active participation and collaboration among all stakeholders. This phase involves structured activities with clearly defined roles for participants and researchers. Real-time monitoring is crucial to document changes, and adjustments to the action plan may be made as necessary to address emerging issues. The iterative nature of this phase ensures that the interventions remain dynamic and responsive to the needs identified during planning [26].

The final phase focuses on observing and evaluating the outcomes of the implemented actions. Participants engage in reflective practices to assess the effectiveness of the interventions and their impact on the problem. The Action research methodology, as noted by incorporates both communal reflection and self-evaluation to ensure the research remains valid and inclusive. Reflection often leads to the identification of new questions or areas for further inquiry, thus initiating another cycle of action research [27].

One needs to adapt planned curricula, aligning them with action research principles. Here is a step-by-step guide for implementing action research in the context of teaching MACH architecture at the university level using agile methodologies: The first phase is to define objectives of the learning and the specific questions that are to be considered during the course implementation. The next phase is to define the clarity of the learning course objectives, the third phase is to develop a course of study and design interventions, and the fourth one is the preparation of the learning material. Those four phases of the proposed methodology are in line with the first planning phase of the action research approach.

The second planning phase of the action research is acting by conducting the course in two distinctive phases. The first phase represents introduction to agile concepts and the second one is the implementation of the course through agile sprints.

The action phase is introducing the agile and agile sprints.

The observing phase is in line with collecting qualitative and quantitative data during the implementation of the course. A methodological approach was adopted so that the observing phase could overreach all phases of the action research.

The final reflecting phase was performed by interpreting the results in the context of the research questions and hypotheses, correlating the findings with existing literature, and discussing the implications of the findings for educational practices. Since the goal of action research is more interpretive than explanatory, i.e., the teacher does not demonstrate that what is learned is true in all cases but only in the presented case [28], one will support those with the case study. An overall overview of the modified approach developed to seamlessly immerse action research in the learning cycle is given in Figure 1.

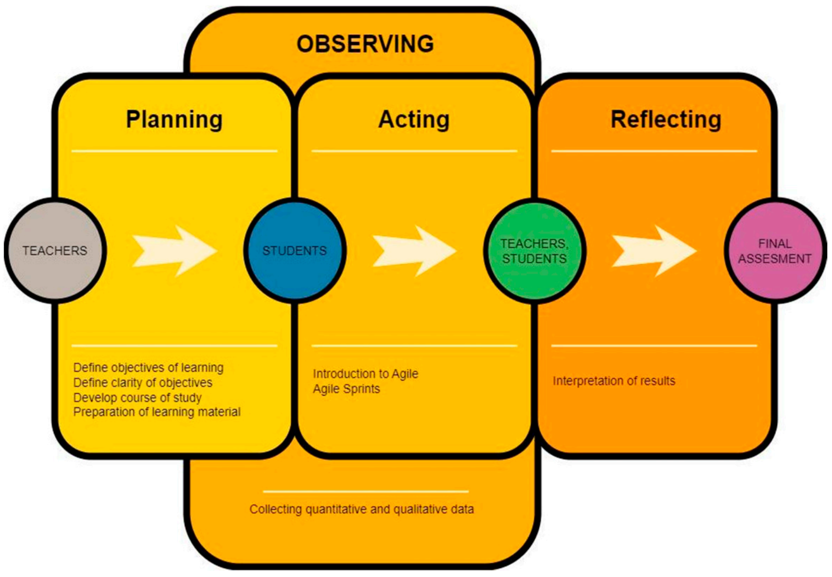

OBSERVING
Planning
Acting
Reflecting
TEACHERS
STUDENTS
TEACHERS, STUDENTS
FINAL ASSESMENT
Define objectives of learning
Define clarity of objectives
Develop course of study
Preparation of learning material
Introduction to Agile
Agile Sprints
Interpretation of results
Collecting quantitative and qualitative data

Figure 1. The action research phases.

The research utilized a range of metrics and tracking tools to monitor the impact of interventions effectively. Pre- and post-course knowledge assessments were conducted using self-assessment questionnaires and conceptual quizzes, providing quantitative data on students' understanding and confidence in MACH principles. Grading components, including assignments (30%), midterm exams (30%), and final projects (40%), offered structured, quantitative evaluations of performance across key milestones. Statistical tools such as paired *t*-tests and Cohen's d were employed to confirm significant knowledge gains, with results showing robust evidence of the course's effectiveness. Additionally, GitHub repositories and Trello boards were used to track student deliverables and team progress, ensuring alignment with agile principles during the sprints.

The research also included external validation through industry feedback, assessing the real-world applicability of MACH principles. Surveys and statistical tests, such as Z-tests, highlighted the relevance of the adopted methodologies, linking them to industry trends. These tools provided a comprehensive approach to measuring the outcomes of the intervention, from academic performance to real-world readiness. Together, these metrics form a solid framework for assessing learning outcomes and ensuring the interventions' effectiveness in both educational and professional contexts.

4. Phases to Implement Proposed Research Methodology

4.1. The Planning Phase of Action Research

4.1.1. Define the Objectives of the Learning and the Specific Questions to Be Considered During the Course Implementation

In the first phases of the planning phase of action research, the research objectives and questions to be considered were defined. Therefore, the main objective was to improve the teaching and learning of MACH architecture using agile methodologies in a university-level course. Furthermore, to accomplish this objective, the following questions were defined and addressed:

1. What are the students' previous knowledge and confidence levels for implementation of MACH architecture?
2. How could the agile methodologies be effectively integrated into the course to enhance learning outcomes?
3. What are the students' perceptions and experiences of learning MACH architecture through agile sprints?
4. How does the use of agile methodologies facilitate the learning process?
5. Is action research compatible with constructive educational methodology?
6. Can action research be modified and integrated as a curriculum-enhancing tool?

Based on this research questions, we devised two hypotheses to prove in or work:

Hypothesis 1: *Integrating agile methodologies, such as Scrum, into the curriculum enhances students' learning outcomes, engagement, and understanding of MACH architecture by fostering iterative, collaborative, and real-world learning experiences.*

Hypothesis 2: *Action research provides a flexible and iterative framework compatible with educational methodologies, allowing continuous curriculum enhancement through stakeholder feedback, iterative refinement, and alignment with industry standards.*

4.1.2. Define the Clarity of the Learning Course Objectives

The intention was to ensure that participants comprehend the course's learning objectives. The course objective is that by the end of it, students should be able to implement microservices, use APIs, and understand cloud-native development.

This choice was based on several significant needs, all of which were articulated and carried out by software engineering firms. The plan was to create a course that would match the skills required by employers. Two major challenges emerged: the first one was the application of MACH architecture, and the second one was the application of agile development in the process of creating novel solutions, particularly in the period of onboarding when recent graduates must quickly adjust to the place of employment. In order to achieve this, five course objectives in line with four MACH principles were set:

1. The first is the implementation of microservices, where each microservice was developed independently, with their own set of functionalities.
2. The second principle is that every developed microservice had a defined application programming interface (API). This principle is called API-first.
3. The third principle is cloud-native development. The design and development of the project were completed in a cloud environment.
4. The final, fourth principle is headless. The user experience was decoupled from the back-end services. This allowed the solution to be applicable across a variety of devices in a multi-channel environment in a cost-effective and efficient manner [29].

4.1.3. Develop a Course Syllabus

Before defining the new syllabus for the course, it was planned to implement this approach in a course that covers some issues raised in MACH architectural design. An advanced course based on students' knowledge of architectural design was chosen, which required knowledge of cloud-native and web-based programming. For accomplishing this task, a web programming course, lasting for 15 weeks, was selected. This course carries six European credits, as defined by the European Credit Transfer System (ECTS), involving 25 h of student work throughout a 15-week semester. The syllabus was developed and implemented at the State University of Novi Pazar during the summer semester of the 2023/2024 school year. This was a modification of the syllabus, which was accredited for the second time for the following seven-year accreditation cycle of the study program in software engineering, bachelor-level studies. In order to achieve the already defined course learning objectives, the following topics for each of the 15 weeks were selected: Introduction to MACH Architecture, Microservices: Principles and Best Practices, API-first Development: REST vs. GraphQL, Cloud-native Architectures: Containers and Kubernetes, Serverless and Event-driven Architectures, Headless CMS and Decoupling Frontends, Building Scalable APIs and Services, Observability and Monitoring in Microservices, Security and Authentication for APIs and Microservices, Cloud-native Databases and Storage Solutions, Deploying and Managing Cloud-native Applications, Building Headless E-commerce Applications, Advanced Topics: MACH in Practice, Project Work: Building a MACH-based Application, Final Presentations, and Course Wrap-up Grading.

To monitor student success, the following grading system was created: during the semester, students received three assignments, and the results achieved on those assignments constituted 30% of the grade. These assignments were as follows: Assignment 1: Microservices Architecture Design (10%)—Assignment out 1st week, due 4th week; Assignment 2: API-first Development (10%)—Assignment out 5th week, due 9th week; and Assignment 3: Cloud-native Deployment (10%)—Assignment out 10th week, due 13th week.

Another 30% of the grade was based on the midterm exam, which focused on concepts learned from weeks 1 to 7 and was held in the 8th week.

The final project accounted for 40% of the grade and was presented as a case study later in the paper, no later than the exam period. The final project was graded based on the project proposal (10%), development and deployment performance (30%), and final presentation and code review (30%).

4.1.4. Design Intervention

The next phase in the action research is to plan specific interventions (e.g., hands-on projects, collaborative activities, and iterative feedback) to address identified challenges. Since our goal was to plan specific interventions and help students to have a better understanding the MACH architecture to facilitate easier onboarding after their studies, short interviews with companies to identify common challenges during the development cycle were conducted. Additionally, three assignments that students should complete during the semester were defined, along with a final project. For all proposed assignments, the objectives, tasks, deliverables, and evaluation criteria were developed. At the final project, students need to complete full scale application based on the MACH architecture.

4.1.5. Prepare Material

The next phase in the action research was to develop course materials, including lectures, reading materials, and assignments. As the first phase in this stage, it was necessity to develop detailed assignment tasks, deliverables, and evaluation criteria. This detailed plan provided teachers with an overview of the basic literature that students would use to achieve the planned learning outcomes. Additionally, this plan gave an overview of the software and hardware needed to implement action research in the learning cycle. In this phase, all the tools and platforms needed for course implementation were set up.

The related books covering the issues in the course were selected, and they were recommended to the students. The first set of books covered the development and proper implementation of microservice architecture [30]. The second one covered design patterns in API development. Then, the literature about cloud-native development using Kubernetes was presented. Finally, it was decided to use WordPress as a native application for headless design, so we selected the following books that cover advanced aspects of PHP programming and the architectural components of WordPress [15,31,32].

Adequate lectures were prepared and delivered to the students using Git 2.39.0.windows.2 as the version control system, with GitHub serving as the file repository. In line with the active research role, each lesson included step-by-step code examples developed and presented in the version control system, in this case, in the Git repository.

Every student had its own Git repository to which each of them uploaded completed tasks and assignments. Each assignment had a different branch in the designated Git repository.

Finally, it was decided to use WordPress 6.1.5 as the application for the headless CMS and front-end development. With the use of PHP, the students can manage the back-end application part and seamlessly integrate it with the front-end part.

4.2. Conduct the Course

The second phase of the action research was acting, where the course is to be conducted in two distinct stages. The first stage will introduce agile concepts, while the second one will implement the course through agile sprints.

4.2.1. Introduce Agile Methodologies

Since the main guiding point for action research is the development of the skills and capacities of all parties involved in the research, it was decided to implement good industrial practices in the course implementation. Thus, it will be aligned with other important educational goals that emerged due to new industrial needs. At the same time, the new educational paradigm expects teachers to use the modern digital infrastructure and technologies with the aim to transform the teaching process. This approach, known as Education 4.0, is compatible with digital transformation, which is at the core of Industry 4.0 [33,34].

Keeping in mind the previously stated, it was decided to deliver the course through the implementation of agile methodologies, as they were the most common approach used in the information technology industry at the time of the research. In this way, all important aspects of the digital transformation process [35] will be covered.

The main reason for the development of the agile approach was that the software engineering process has not been compatible with traditional business approaches. Software development used iterative and interactive interactions, so classical approaches created overheads in development and did not provide the necessary adaptability of the process [36].

A set of four values and 12 principles, the core of agile, were presented by an alliance of 17 engineers and consultants in the "Agile Software Development Manifesto" [37]. These values and principles provide the basis to guide the software development process [38] and distinguish any method with the agility feature.

Agile methodologies, developed over time, encompass a range of processes that align with the core principles and values of the agile philosophy. Each methodology comprises a unique set of practices, which dictate the daily activities of software developers. These practices and terminologies vary across different agile methods, distinguishing one from the other. However, they all share a common foundation in developing software through iterative and incremental processes [39,40].

Throughout the research, the Scrum approach was implemented since most of the companies in implementation have been using this agile methodology. This will ease the process of the student onboarding into jobs after the completion of their studies. Scrum is the implementation of the framework for project management, adapted for the iterative software development process. Since this approach is relatively simple and focused on software management rather than technical software development, it can be used as the backbone for delivering the assignments during the course [41].

Scrum, as a methodology, has three phases: the initial phase, the development one, and the closure one. In the initial phase, the general objectives of the planned assignment are outlined. It also defines the members of project teams and the required tools and resources. The result of this phase is called an iteration. In this phase, the product backlog for documenting teacher assignments as user stories was defined. User stories are based on a simple pseudocode:

<User> wants to <Have> so that <Benefit>.

The requirements are then analyzed and prioritized by the product owner; in this case, this role was taken on by teaching assistants. The product backlog is a live document that can be updated regularly.

The second phase of Scrum consists of sprints. This phase involves a series of cycles, each 4 weeks long, with each sprint producing one product based on assignment as a result. Each sprint includes all the software development phases, from requirements analysis, through design, to the final delivery phase after review.

A sprint starts with an initial meeting between the teacher (product owner) and the students (team). In this meeting, user stories are defined, and then the sprint begins. During the sprint, students have 15 min Scrum meetings daily. After the sprint, there is a sprint review meeting and a sprint retrospective [42].

Once the requirements are met and the software is delivered in line with the product owner's needs, the latest version of the product is ready for the teacher's review.

The overall cycle of the Scrum methodology is given in Figure 2.

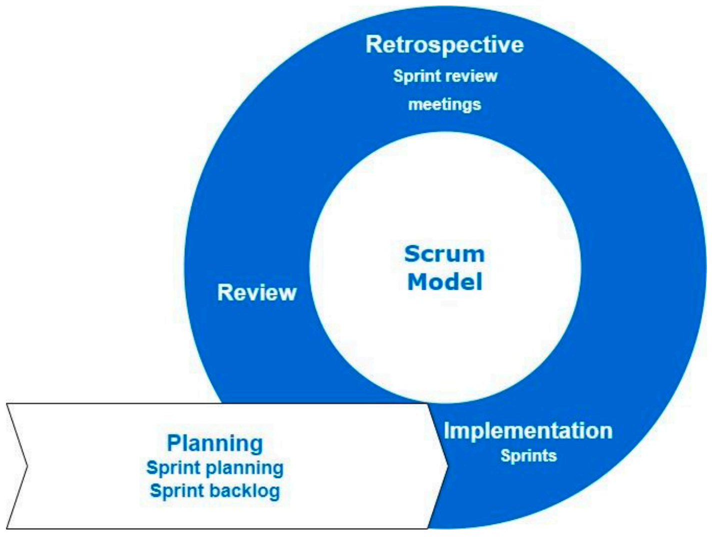


Figure 2. The action research phases.

4.2.2. Execute Sprints

The final phase of the action research before the reflection phase was the implementation of the sprints. Three sprints for the students were prepared, every one of them focused on specific MACH architecture components.

The three main topics for the sprints were Microservices Architecture Design, API-first Design, Security and Authentication in Microservices, and Cloud-Native Development. A detailed list of objectives, tasks, and expected deliverables is given in Table 1.

Table 1. A detailed list of objectives, tasks, and expected deliverables.

Sprint 1		
Title: Microservices Architecture Design		
Objective: students will create a headless e-commerce website using a headless CMS and decoupled front-end.		
Tasks for First Sprint		
Tasks	**Deliverable**	**Evaluation criteria**
Conduct an analysis of the e-commerce platform's requirements to identify distinct microservices needed (e.g., user management, product catalog, and order processing).	Product backlog	Completeness of the architecture Clarity of the API documentation Functionality of the implemented service
For each identified microservice, define the RESTful APIs, including endpoints, request/response formats, and authentication mechanisms.	Detailed documentation of each API, including endpoints, methods, request/response formats, error handling, and authentication mechanisms.	
Create diagrams that illustrate the relationships and interactions between the different microservices. Ensure to include data flow, communication patterns, and dependencies.	A visual representation of the microservices and their relationships, including communication patterns and dependencies.	
Implement the identified microservices and deploy them using Docker containers. Ensure the services are accessible and functioning as expected.	Fully functional code for each microservice, along with Docker files and deployment scripts.	
Sprint 2		
API-first Design		
Objective: design, implement, and document a RESTful API for a specific application using OpenAPI (Swagger) standards.		
Tasks for second Sprint		
Tasks	**Deliverable**	**Evaluation criteria**
Define the overall API design following RESTful principles, using OpenAPI standards. This includes specifying the API structure, paths, operations, and response formats.	A complete specification file defining the API structure, paths, operations, and response formats.	Compliance with OpenAPI standards Functionality of the implemented endpoints Quality of API documentation Successful deployment
Develop endpoints for basic user management functions such as creating, reading, updating, and deleting users (CRUD operations).	Functional CRUD endpoints for user management, supporting versioning and pagination.	
Integrate versioning mechanisms to allow the API to evolve without breaking existing clients. Implement pagination to manage large sets of data efficiently in user management endpoints.	Test results	
Implement at least three key endpoints from the designed API and deploy the API to a cloud platform (e.g., AWS, Azure, or Heroku). Ensure the API is accessible and functioning.	A live, accessible API hosted on a cloud platform with at least three implemented endpoints.	
Create comprehensive API documentation using Swagger UI or Postman. This documentation should cover all endpoints, request/response formats, and examples of how to use the API.	Detailed documentation available through Swagger UI or a Postman collection, covering all implemented features and providing usage examples.	

Table 1. *Cont.*

Sprint 3		
Security and Cloud-Native Development		
Objective: deploy a cloud-native microservice using Kubernetes by containerizing the application		
Tasks for first Sprint		
Tasks	**Deliverable**	**Evaluation criteria**
Create a Dockerfile to containerize the application, ensuring that all necessary dependencies and configurations are included.	A complete Dockerfile that successfully containerizes the application, including all necessary dependencies.	Successful deployment Scalability Automation of the CI/CD pipeline Documentation
Write a Kubernetes deployment file that specifies the desired state of the application, including replica counts, image details, and resource requests. Deploy the application to a Kubernetes cluster.	A YAML file that defines the deployment, including replicas, image configuration, resource requests/limits, and any environment variables.	
Develop a CI/CD pipeline using GitHub actions that automates the build, test, and deployment process for the application to the Kubernetes cluster.	GitHub actions workflow configuration file that automates the building, testing, and deployment process.	
Configure the Kubernetes deployment to support horizontal pod autoscaling (HPA) based on CPU/memory utilization and implement rolling updates to ensure zero downtime during deployments.	A URL or IP address to access the deployed application on the Kubernetes cluster, demonstrating the working deployment.	

5. Research Results

As it was already stated in the previous sections, the observation phase of the research was integrated and implemented throughout the entire action research process. During this implementation, various qualitative and quantitative data were collected.

5.1. Survey for Defining Clarity of Objectives

A set of interviews and surveys was created and delivered to 20 software development companies that already hired alumni from University. The goal was to analyze whether there was an increasing need for software based on MACH architecture by using a Z-test for proportion to check if there was any significant difference for this architectural approach. Additionally, a set of qualitative questions to further clarify the challenges of implementing MACH architecture was delivered. After that, a survey on the adoption of architectural patterns, the success of projects based on selected patterns, the time allocated to architectural design, and the tools and technologies used for MACH was conducted. The results of the survey are given in Table 2.

Table 2. The survey results.

Adoption of Architectural Patterns		
Architectural Pattern	**Number of Projects**	**Percentage (%)**
MACH	90	36%
Microservices	60	24%
Monolithic	30	12%
Layered architecture	40	16%
Event-driven	20	8%
Service-oriented architecture	10	4%
Total	250	100%
Success of Architectural Design by Pattern		

Table 2. *Cont.*

Architectural Pattern	**Projects Rating Success as 4 or 5 (out of 5)**	**Percentage (%)**
MACH	81	90%
Microservices	48	80%
Monolithic	15	50%
Layered architecture	24	60%
Event-driven	14	70%
Service-oriented architecture	7	70%
Time Allocated to Architectural Design (More than 30% of Project Time)		
MACH	40	44.4%
Microservices	15	25%
Monolithic	6	20%
Layered architecture	10	25%
Event-driven	9	45%
Service-oriented architecture	4	40%
Tools and Technologies Used by MACH Projects		
Tool/Technology	**Number of MACH Projects Using It**	**Percentage (%)**
Cloud-native tools	63	70%
API management platforms	77	85%
Headless CMS	50	56%
Microservices frameworks	81	90%

To demonstrate a significant rise in MACH architecture adoption, a Z-test for proportions was performed to show this statistically significant difference. The null hypothesis was defined as the adoption of MACH architecture being equal to or less than the proportion of projects adopting the next most popular architecture, which in the research was microservices architecture. The alternative hypothesis was that the proportion of projects adopting MACH is greater than the proportion of projects following the next most popular architecture. The proportion of adoption for MACH was $p1 = 0.36$, while the proportion of adoption for the next most popular architecture was $p2 = 0.24$. The pooled proportion for these values was $p = 0.3$, and the standard error for the difference between the two proportions was SE = 0.041. Next, a Z-score of 2.93 was calculated, and the p-value for this score was 0.0017. This p-value indicates the rejection of the null hypothesis providing statistically significant evidence that the proportion of projects adopting MACH architecture is higher than that of the next most popular architecture.

Since a set of qualitative answers was required, it was decided to define questions addressing some important engineering challenges. A focus was on the following issues: Challenges in Architectural Design, Decision-Making Process, and Innovation in Design, Stakeholder Influence, Lessons Learned, Adaptability of Design, Documentation Practices, and Future Trends. A detailed list of questions for each topic and examples of answers is provided in Table 3.

Qualitative analysis showed that some challenges in architectural design revolve around integrating with existing systems, maintaining security and coordinating existing processes. In the existing decision-making process, the main reason for adopting MACH architecture was easy and straightforward implementation of customer demands with a high level of cost-effectiveness. The implementation of this architecture showed that successful adoption requires gradual implementation and planning and, most importantly, continuous learning and training. MACH architecture is highly adaptable, allowing companies to respond quickly to changes in business needs, technology, and regulatory environments.

All of this supports and shows that the defined learning objectives are up to date with state-of-the-art architectural designs.

Table 3. A detailed list of questions for each topic and examples of answers.

Topic	Question	Key Themes	Example Answers
Challenges in Architectural Design	What are the most significant challenges you face during the architectural design phase of your projects?	Integration and migration	"Integrating MACH with existing ERP systems was a major challenge due to differences in data formats".
		Performance and scalability	"Handling real-time data processing in a distributed environment was particularly challenging".
		Security and compliance	"Managing data privacy and security during integration with third-party services was challenging".
		Communication and coordination	"Communication across teams working on different microservices was challenging, leading to delays".
Decision-Making Process	Can you describe the decision-making process for selecting the architectural pattern for your current project? What factors were most influential?	Flexibility and scalability	"We chose MACH to support our international expansion, as it provided the flexibility we needed".
		Customer and business demands	"The decision was influenced by the need to deliver consistent experiences across multiple channels".
		Cost-effectiveness	"After evaluating the long-term maintenance costs, MACH emerged as the most cost-effective solution".
		Adaptability and future-proofing	"The decision to move to MACH was largely driven by the need for real-time data processing capabilities".
Innovation in Design	Have you implemented any innovative approaches in your architectural design? If so, please describe them and their impact	Technological innovation	"We implemented a micro-frontend approach, enabling independent deployment of UI components".
		Efficiency and performance	"Our edge computing setup with MACH enhanced global content delivery speed".
		Security and transparency	"Our use of blockchain for transaction management within MACH enhanced security and transparency".
Stakeholder Influence	How do the different stakeholders influence the architectural design? Can you provide examples of how their input shaped the architecture?	Cross-functional influence	"Feedback from our developers about the complexity of maintaining a monolithic architecture led us to MACH".
		Customer and business needs	"Business stakeholders pushed for features that required a highly flexible and scalable solution, leading us to MACH".
		Security and compliance	"The compliance team's requirements shaped our data management strategy within MACH".

Table 3. *Cont.*

Topic	Question	Key Themes	Example Answers
Lessons Learned	What lessons have you learned from previous projects regarding architectural design that you applied to your current project?	Gradual adoption and planning	"Starting small with MACH components and gradually migrating other services helped manage the transition smoothly".
		Training and communication	"Ensuring the team is well-trained on cloud-native tools before starting the project minimized delays".
		Security and monitoring	"We found that setting up a robust monitoring system early in the project was essential".
Adaptability of Design	How adaptable is your architecture to changes in project scope, technology, or requirements? Can you provide examples where you had to adapt your architecture?	Modularity and flexibility	"We could easily add new microservices without affecting existing ones, thanks to the architecture's modularity".
		Scalability	"We could quickly scale our architecture to support a global user base, thanks to its modular design".
		Security and compliance	"When security requirements tightened, we could quickly adapt by implementing new protocols without major overhauls".
Documentation Practices	How do you manage architectural documentation, and what role does it play in your project? How do you ensure that it stays relevant and useful?	Automation and integration	"We use automated testing to ensure our documentation is accurate and up-to-date".
		Centralization and accessibility	"We ensure that all stakeholders, including non-technical ones, can access and understand the documentation".
		Version control and consistency	"We use version-controlled documentation to track changes and maintain consistency across teams".
Future Trends	What trends do you foresee in software architecture that might influence your future projects?	Serverless and edge computing	"I foresee increased adoption of serverless architecture combined with MACH for greater scalability".
		AI and automation	"The increasing complexity of IT environments will drive the need for more automated, AI-driven management tools".
		Security and privacy	"Decentralized architectures, like blockchain, might start integrating with MACH to enhance security and data integrity".

5.2. Delivery of Learning Material and Pre-Course Surveys to Measure Changes in Knowledge and Skills

In line with the planned lecture delivery, the teacher and assistants installed the latest version of Git (2.46.0), the 64-bit version for Windows. The main Git repository was installed on the university server machine. The free Git GUI SourceTree 3.4.15 also installed to enable ease of use. Different tasks were delivered to students through the Trello platform, and the teacher prepared a set of example codes and solutions for problems. GitHub was used as the platform that allowed students to create, store, manage, and share their code.

Students were divided into seven teams, with five students in each team. Each team also created a local Git repository and a remote GitHub repository for joint access. The teachers created seven Trello workspaces, which were used for managing the sprints for each student team [43,44]. Before starting the implementation of the sprint, another set of qualitative and quantitative questionnaires was delivered. The aim of these questionnaires was to measure changes in skills related to the understanding and implementation of MACH principles. The questions were designed to provide insight into theoretical practices and the application of each of the four MACH principles. Each question had two parts: the first was a knowledge assessment where students choose one of four offered answers (one correct answer), and the second was a self-assessment of their skills, where they rated their skills on a scale from 1 to 5, with 1 representing a low skill level and 5 representing a high skill level. The questionnaire is provided in Table 4.

Table 4. Pre-course survey on MACH architecture.

Pre-Course Survey on MACH Architecture
Microservices Implementation
1. Knowledge Assessment:
Which of the following best describes the benefit of using microservices in a MACH architecture?
- (A) Improved database management
- (B) Enhanced system flexibility and scalability
- (C) Easier code debugging
- (D) Increased UI responsiveness
2. Skills Self-Assessment:
How confident are you in developing independent microservices using different programming languages and frameworks? (Rate from 1 to 5)
API-First Principle
3. Knowledge Assessment:
What is the primary role of an API in a microservice architecture?
- (A) To secure the application
- (B) To establish seamless communication between different services
- (C) To improve user interface design
- (D) To store data
4. Skills Self-Assessment:
How proficient are you in designing and implementing APIs that facilitate interaction between front-end and back-end systems? (Rate from 1 to 5)
Cloud-Native Development
5. Knowledge Assessment:
Which of the following is a key characteristic of cloud-native development?
- (A) Monolithic code structure
- (B) In-house data storage
- (C) Development and deployment in a cloud environment
- (D) Exclusive use of on-premise servers
6. Skills Self-Assessment:
How familiar are you with developing and deploying applications in a cloud-native environment? (Rate from 1 to 5)
Headless Architecture
7. Knowledge Assessment:
What is the main advantage of a headless architecture in a MACH environment?
- (A) Enhanced security features
- (B) Separation of user experience from back-end services, enabling multi-channel deployment
- (C) Simplified content management
- (D) Better performance on mobile devices

Table 4. *Cont.*

Pre-Course Survey on MACH Architecture
8. Skills Self-Assessment:
How confident are you in implementing a headless architecture that allows content to be served across multiple devices and platforms? (Rate from 1 to 5)
Overall MACH Principles Application
9. Skills Self-Assessment:
How confident are you in applying the MACH principles (Microservices, API-first, Cloud-native, Headless) to design and develop scalable and flexible applications? (Rate from 1 to 5)
10. Expectations and Goals:
What specific challenges do you expect to encounter when applying MACH principles in your projects? (Open-ended)

The pre-course results on understanding and confidence are showed in Table 5.

Table 5. Pre-course survey on student understanding and confidence on implementing MACH architecture.

Topic	Correct (%)	Common Misconceptions	Average Confidence
Microservices	43	Improved database management,	N/A
API-First Principle	34	Security, UI design	N/A
Cloud-Native Development	29	Monolithic structures, on-premises servers	N/A
Headless Architecture	37	Performance, security aspects	N/A
Overall Confidence in Microservices	N/A	N/A	2.6
Cloud-Native Development Skills	N/A	N/A	2.4
Confidence in Headless Architecture	N/A	N/A	2.3
Ability to Apply MACH Principles	N/A	N/A	2.4

The pre-course results show that only 15 out of 35 students (43%) correctly identified the benefit of microservices as enhancing system flexibility and scalability, while common incorrect answers included confusing microservices with improved database management or UI responsiveness. Regarding the API-first principle, 12 out of 35 students (34%) correctly identified the role of APIs in establishing seamless communication between different services, while most students initially selected options related to security or UI design, reflecting a misunderstanding of the API-first principle.

Only 10 out of 35 students (29%) correctly understood that cloud-native development involves development and deployment in a cloud environment, while the most common misconceptions included associating cloud-native with monolithic structures or on-premise servers. Finally, 13 out of 35 students (37%) correctly identified the advantage of headless architecture in separating the user experience from back-end services for multi-channel deployment, while incorrect answers often focused on performance or security aspects.

Another set of questions regarding student confidence gave the following results: The average confidence level was 2.6 out of 5, with many students rating themselves low due to unfamiliarity with independently developing microservices. Students rated their cloud-native development skills at an average of 2.4 out of 5, with many unfamiliar with cloud environments. Confidence in implementing headless architecture was rated at an average of 2.3 out of 5, reflecting a general uncertainty about its implementation. Overall, students rated their ability to apply MACH principles at an average of 2.4 out of 5.

With all the data ready, it was possible to develop the solutions through a set of sprints and implementation of special case studies, which were the core of the action research.

5.3. Case Study: Development of the Blockchain-Based NFT Trading Web 3.0 Application

The initial step involved the teacher, playing the role of the customer, in defining the product backlog. Since the teachers assumed this role, here is how they defined the product backlog: "I want to create a next-generation Web 3.0 platform that allows users to buy, sell, and trade NFTs (Non-Fungible Tokens) securely. The platform should leverage blockchain technology to ensure transactions are transparent and decentralized, with each NFT purchase being executed through a smart contract. The goal was to make user's being able to browse various NFTs, create their own listings, and securely handle payments using cryptocurrency. The platform should be scalable, reliable, and easy to use, especially for people who may not be familiar with blockchain or Web 3.0 technology. The platform to support a large user base was also needed, and any updates to the system should happen seamlessly without downtime".

Following this, teaching assistants and students analyzed the product backlog. They first divided it and made connections between the demands and the sprints, aligning them with the overall objectives of each sprint.

The non-technical demand for the first sprint was defined as: follows "I want a marketplace where users can list NFTs for sale, browse listings, and make purchases". The non-technical demand for the second sprint was as follows: "Users need to manage their profiles and interact with NFTs, and transactions must be automated through smart contracts". The final demand was as follows: "The platform must be scalable, support thousands of users, and handle updates without downtime".

The product backlog was then divided into a set of user stories presented in Table 6. Each user story had tasks, acceptance criteria, priority, and several story points (a measure of the effort required to deploy and implement the demand).

Table 6. Sprints and user stories.

Sprint No.	User Story	Priority	Story Points	Acceptance Criteria
1	As a developer, I want to identify microservices required for the NFT e-commerce platform so I can structure the system properly.	High	5	A list of identified microservices with their functions clearly described.
	As a developer, I want to define RESTful APIs for each microservice so the services can communicate with each other efficiently.	High	8	API documentation for each service with end-points, request/response formats, and authentication mechanisms.
	As a developer, I want to create an architecture diagram showing the relationships between services so I can visualize service interactions.	Medium	5	A clear and complete architecture diagram illustrating the interaction between services.
	As a developer, I want to implement and containerize the NFT Listing Service using Docker so it can be deployed.	High	8	A fully functional and containerized NFT Listing Service.
	As a user, I want to access the API documentation in a user-friendly interface (Swagger/Postman) so I can understand and interact with the API.	Medium	3	API documentation available through Swagger UI or Postman.

Table 6. *Cont.*

Sprint No.	User Story	Priority	Story Points	Acceptance Criteria
2	As a developer, I want to design the API using OpenAPI (Swagger) standards so the API follows RESTful principles.	High	5	OpenAPI (Swagger) specification file with all API definitions.
	As a developer, I want to implement CRUD endpoints for user management so I can manage user accounts efficiently.	High	8	Working API endpoints for create, read, update, and delete operations.
	As a developer, I want to implement versioning in the API so I can handle future updates without breaking compatibility.	Medium	5	Versioning implemented in the API, with support for v1, v2, etc.
	As a developer, I want to add pagination to the API so large datasets can be managed efficiently.	Medium	5	Pagination functionality for user management endpoints.
	As a DevOps engineer, I want to host the API on a cloud platform so it can be accessed by users.	High	8	API hosted and accessible on a cloud platform.
	As a user, I want to have comprehensive API documentation with examples so I can interact with the API.	Medium	3	Detailed API documentation using Swagger/Postman, including examples for all endpoints.
3	As a DevOps engineer, I want to containerize the application using Docker so it can be easily deployed and scaled.	High	5	A Dockerfile and Docker image for the microservice.
	As a DevOps engineer, I want to create a Kubernetes deployment file so the microservice can be deployed to a Kubernetes cluster.	High	8	A functional Kubernetes deployment YAML file.
	As a DevOps engineer, I want to deploy the microservice to a Kubernetes cluster so the service is accessible and functional.	High	8	Successfully deployed microservice on the Kubernetes cluster.
	As a DevOps engineer, I want to automate the deployment process using a CI/CD pipeline (GitHub actions) so the deployment is fast and reliable.	High	8	Fully automated CI/CD pipeline using GitHub actions for building and deploying the microservice.
	As a DevOps engineer, I want to enable automatic scaling and rolling updates in the Kubernetes deployment so the service remains scalable and resilient.	Medium	5	Kubernetes deployment supports Horizontal Pod Autoscaling (HPA) and rolling updates.
	As a stakeholder, I want documentation of the deployment process so I can understand how the service was deployed and scaled.	Medium	5	Clear documentation of the Docker containerization, Kubernetes deployment, CI/CD setup, and scaling.

In the first sprint, students identified the key microservices that would be the core of the NFT marketplace platform. Each microservice was designed to be independent, scalable, and aligned with a decentralized architecture [45]. Students defined three main microservices to be developed: the User Management Service, NFT Listing Service, and Marketplace Service.

The first microservice handled user accounts within the NFT marketplace platform. This microservice was responsible for user-related operations and account-related create, read, update, and delete (CRUD) operations. The second microservice was responsible for managing NFTs listed for sale on the platform. Its aim was to create, update, and remove listings, closely interacting with the blockchain registry of NFTs. The third microservice handled interactions related to buying, selling, and bidding, serving as the transactional layer of the platform [46,47].

For each microservice, the following deliverables were presented: an API overview, RESTful design principles were highlighted, security mechanisms were elaborated, examples of endpoints were provided, and snippets of Swagger documentation were delivered.

As an example, the deliverables for the Marketplace Service API were presented, which handled the purchase, bidding, and sale of NFTs. This microservice integrated with blockchain-based smart contracts to facilitate secure, decentralized transactions between buyers and sellers.

RESTful Design Principles Highlighted:

Resource Identification: marketplace transactions are treated as resources accessible via /marketplace/transactions.

Separation of Concerns: this API strictly manages the transactional aspects of the marketplace, separate from the NFT listing and user management services.

Statelessness: each transaction request includes the necessary information, and there is no session state maintained on the server.

Security Mechanisms:

Smart Contract Integration: the marketplace API interacts directly with smart contracts on the blockchain to handle ownership transfer and payment verification.

JWT Authentication: buyers and sellers must authenticate using JWT tokens before initiating any transactions.

Example Endpoints:

POST/marketplace/transactions

Description: initiates the purchase of an NFT.

Request Body:

{“nftId”: “xyz789”, “buyerId”: “12345”, “price”: “2 ETH”, “contractAddress”: “0xContractAddress”}

Response: 201 created with the transaction details.

{“transactionId”: “tx67890”, “status”: “completed”, “buyer”: “johndoe”, “nftId”: “xyz789”}

GET/marketplace/transactions/{transactionId}

Description: retrieves details of a specific NFT transaction.

Response: 200 OK

{“transactionId”: “tx67890”, “status”: “completed”, “buyer”: “johndoe”, “nftId”: “xyz789”, “amount”: “2 ETH”}

The final deliverable for this sprint is the architecture diagram shown in Figure 3. This diagram illustrates the relationships and interactions between microservices, which are the basic building blocks of the proposed NFT marketplace. Each microservice is responsible for specific roles, and every single microservice interacts with other microservices through APIs.

Here is an example of the one developed microservices and an explanation of their relationships given in diagram:

NFT Listing Service: Manages the creation, update, and deletion of NFT listings and interacts with blockchain smart contracts to verify NFT ownership. Communicates with Smart Contract (Blockchain) to verify and record NFT ownership when NFTs are listed, updated, or sold. Marketplace Service: to provide NFT metadata and listings for purchase and bidding operations. API Interaction: exposes NFT-related endpoints (e.g., /nfts, /nfts/{nftId}) to manage listings and retrieve NFT details. Blockchain Interaction: interacts with the deployed smart contract on the blockchain to ensure decentralized ownership of NFTs.

For the second sprint, students decided to use WordPress and PHP as the server-side scripting language. In this way, they took advantage of the WordPress REST API infrastructure. They implemented the user management API by leveraging the WordPress REST API and custom PHP code for handling CRUD operations, pagination, and versioning. All the logic for user management was implemented as a custom WordPress plugin. The WordPress REST API supports pagination and versioning, and pagination was implemented by passing the page and per_page query parameters to the API.

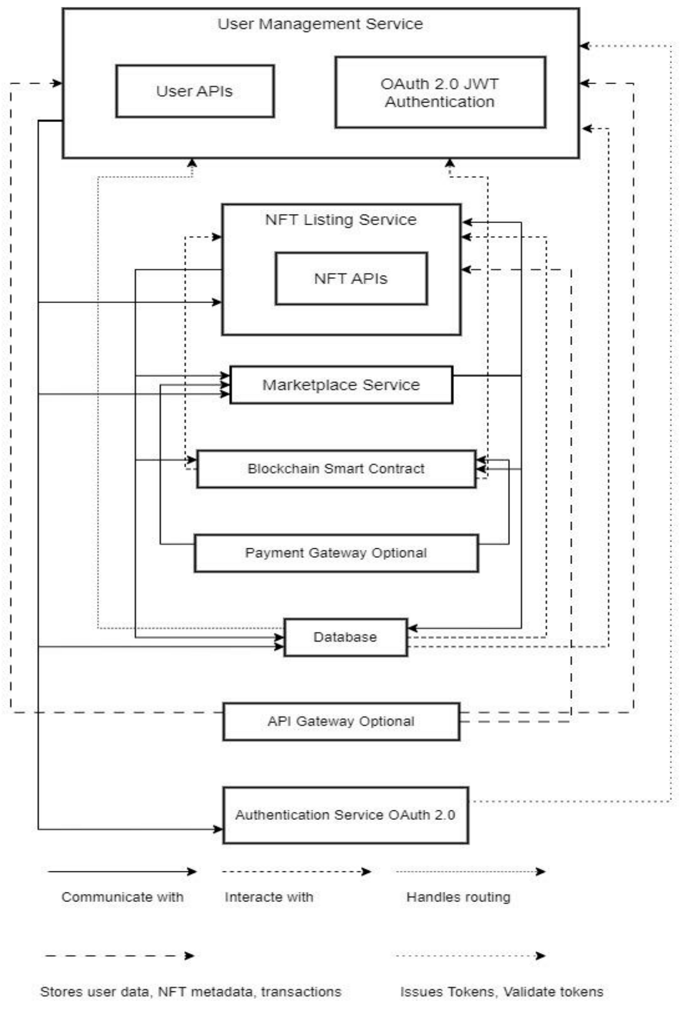


Figure 3. The relationships and interactions between microservices.

The next step was hosting on the cloud using AWS, and the installation was deployed on a LAMP stack. This is given in the following infrastructure diagram.

[Client Requests]–[API Gateway (Optional)]–[Web Server (WordPress on LAMP Stack)]–[MySQL Database]

In the third sprint, the developed microservices were deployed in a cloud-native environment. This was accomplished using Docker, Kubernetes, and a CI/CD pipeline on GitHub. Microservices were containerized using Docker, which ensured that all dependencies were consistent. The containerized microservices were deployed in a Kubernetes cluster for automatic scaling. Here is a snippet of the Kubernetes deployment YAML file used to deploy the User Management Service.

```
apiVersion: apps/v1
kind: Deployment metadata:
name: user-management-service
labels: app: nft-marketplace service: user-management
spec: replicas: 2
selector: matchLabels:
app: nft-marketplace service: user-management
```

A CI/CD pipeline was set up using GitHub actions to automate the build, test, and deployment process for each microservice. This reduced manual intervention and ensured that updates were deployed quickly and consistently across environments.

To measure the effects of the case study development, a set of quantitative metrics was used to generate measurable results and demonstrate the benefits of this approach (Table 7).

Table 7. Metrics and results.

Metric	**Sprint 1**	**Sprint 2**	**Sprint 3**
	Result		
Test cases executed	50	40	45
Test cases passed	48 (96%)	39 (97.5%)	44 (97.7%)
Max concurrent users supported	950	1250	2500
Average response time (100 users)	200 ms (UM)	180 ms (CRUD)	150 ms (CRUD)
Average response time (500 users)	350 ms (UM), 550 ms	350 ms	350 ms
Error rate at 1000 users	5% (MS)	4% (some timeouts)	4% (some timeouts)
Downtime during testing	3 min (99.8%)	N/A	N/A
Max ccaling instances	N/A	6 instances	8 pods
Uptime during rolling updates	N/A	99.98%	99.99%
Scaling trigger	N/A	N/A	80% CPU utilization
Pipeline build time	N/A	N/A	5 min
Scaling efficiency	25%	33%	N/A
JWT authentication success rate	100%	N/A	N/A
Vulnerabilities detected	0 critical, 1 medium, 5 low	2 total (0 critical)	2 total (0 critical)

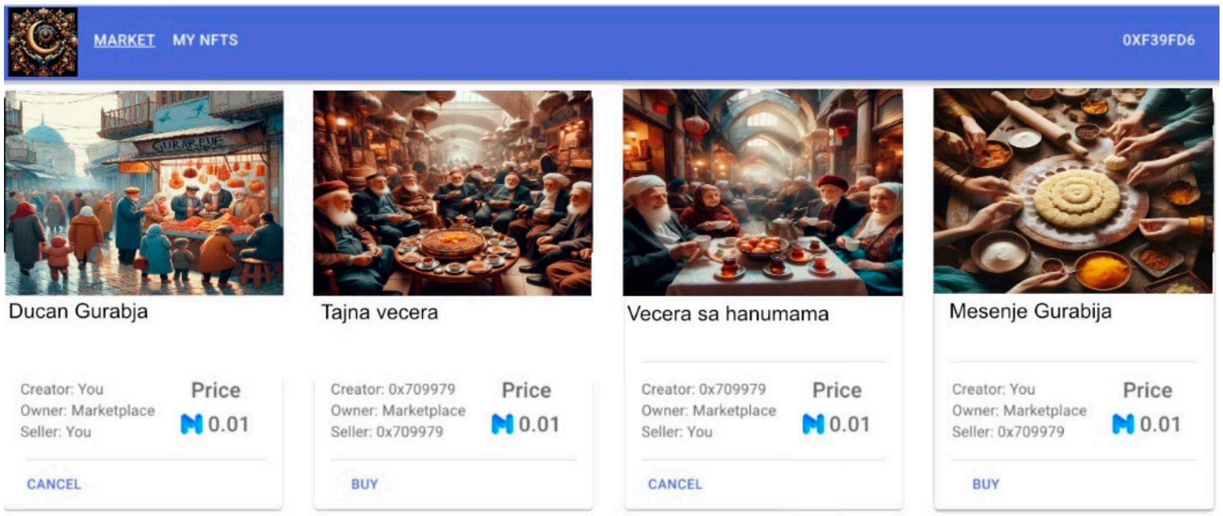
Several key values were tested, including functional correctness, performance, scalability, uptime, and security. Functional correctness was essential to verify that the API endpoints, user operations, and microservices performed as expected, while performance tests measured response times and throughput under varying traffic loads to ensure smooth operation at scale. Scalability was tested to validate the effectiveness of Kubernetes' Horizontal Pod Autoscaler (HPA) in handling increased demand, while uptime tests ensured continuous availability, even during rolling updates. Finally, security tests were conducted to identify and mitigate vulnerabilities. Each of these values is critical to delivering a robust, user-friendly, and scalable NFT marketplace.

The results show that the microservices architecture implemented in Sprint 1 proved highly effective in terms of functional correctness (96% success rate), scalability, and security. The Marketplace Service was the most affected during high traffic (1000 users), though scaling improved performance by 25%. During the tests, the system maintained 99.8% uptime, with effective self-healing capabilities through Kubernetes. Security testing showed robust authentication and protection mechanisms, with no critical vulnerabilities detected.

The API design and cloud deployment in Sprint 2 demonstrated strong performance across all presented metrics, including scalability, uptime, and CI/CD automation. The system handled up to 1250 concurrent users before significant performance degradation, with auto-scaling ensuring a 33% of performance improvement under high load. The rolling updates ensured high availability, with 99.98% of uptime, and the CI/CD pipeline enabled efficient and frequent deployments. Security testing revealed no critical vulnerabilities.

Finally, the cloud-native deployment using Kubernetes in Sprint 3 demonstrated the platform's ability to scale effectively using Horizontal Pod Autoscaler (HPA), with the system handling up to 2500 concurrent users before significant performance degradation. Rolling updates ensured 99.99% of uptime, and the CI/CD pipeline was efficient, with each build and deployment taking approximately 5 min. The platform showed a 36% of performance improvement when scaling up from 2 to 8 pods under high traffic, and security testing revealed no critical vulnerabilities.

The final release was a fully integrated, scalable, and secure application, tested thoroughly for performance, scalability, and security, being capable of handling high traffic while maintaining the core decentralized functionality of the NFT marketplace. Image of the final release is given in Figure 4.



Ducan Gurabja
Creator: You
Owner: Marketplace
Seller: You
Price 0.01

Tajna vecera
Creator: 0x709979
Owner: Marketplace
Seller: 0x709979
Price 0.01

Vecera sa hanumama
Creator: 0x709979
Owner: Marketplace
Seller: You
Price 0.01

Mesenje Gurabija
Creator: You
Owner: Marketplace
Seller: 0x709979
Price 0.01

Figure 4. NFT marketplace.

5.4. Evaluation of the Learning and Action Research Success Rate Results by After-Course Surveys

For this purpose, the same questions which were designed to provide insight into theoretical practices and the application of each of the four MACH principles were used. As it was described thoroughly in Section 5.1, the questions had two parts: the first was a knowledge assessment where students choose one of four offered answers (one was correct). After completing the course, 32 out of 35 students (91%) identified the benefit of microservices correctly. All students correctly identified the primary functions of the API-first principle. On the topic of cloud-native development, 30 out of 35 students (86%) understood the concepts correctly and overcame misunderstandings about monolithic structures and on-premise servers. Finally, 33 out of 35 students (94%) understood the principles of headless architecture correctly.

The second set of questions about confidence showed the following: Average confidence in microservice implementation raised to 4.6 out of 5. Confidence in the API-first principle increased to 4.4 out of 5, confidence in cloud-native development rose to 4.3 out of 5, and overall confidence in applying MACH principles increased to 4.5 out of 5.

A summary of the trends showed significant knowledge gains across all MACH principles, with students demonstrating marked improvements in their understanding. Correct answers increased by an average of 57–66% post-course. In the API-first principle, most dramatic improvement was detected, with a 100% correct response rate post-course.

Students' confidence in applying MACH principles improved by an average of 2.1 points on a 5-point scale. This trend indicates not only a better theoretical understanding, but also increased practical skills and readiness to implement these principles in real-world scenarios.

These trends reflect the effectiveness of the course in addressing key learning objectives. The structured approach to teaching MACH principles, combined with practical exercises, likely contributed to these positive outcomes.

To validate these trends, the paired *t*-tests, *p*-values, and Cohen's d were used. The results are provided in the Table 8.

Table 8. The statistics results for MACH principles.

	t_stat	**p_val**	**Cohen_d**
Microservices	−22.821500596886533	$3.3923419407808097 \times 10^{-22}$	6.196195862237015
API-first	−25.647374299243868	$7.923772762986916 \times 10^{-24}$	5.864323101733134
Cloud-native	−27.19972309531346	$1.1760537569650051 \times 10^{-24}$	5.6955540074295765
Headless	−27.856967342296667	$5.402460105801546 \times 10^{-25}$	5.95071545738165

The t-statistic values for all MACH principles were extremely high, indicating a strong difference between the pre- and post-course confidence levels.

All *p*-values were extremely low (close to 0), far below the typical significance threshold of 0.05. This suggests that the improvements in confidence levels were statistically significant.

The Cohen's d values for all principles were very large, indicating a very strong effect size. This means the course had a substantial impact on improving students' confidence in applying MACH architecture principles.

Regarding qualitative experience, we did not repeat questionnaires, and we decided to use reflective journal to extract details in with initial pre core qualitative research questions. An overview of the most important and most interesting questions is given in Table 9.

Table 9. Qualitative analysis of the reflective journals.

Topic	**Question**	**Key Themes**	**Example Answers**
Challenges in Architectural Design	What are the most significant challenges you face during the architectural design phase of your projects?	Integration and migration	"Integrating MACH principles with our initial monolithic understanding was difficult, especially mapping dependencies between microservices".
		Performance and scalability	"Ensuring the API was scalable and responsive under high load required multiple iterations, and understanding cloud-native tools took time".
		Security and compliance	"Developing secure authentication mechanisms for APIs and ensuring compliance with data protection was overwhelming during the second sprint".
		Communication and coordination	"Dividing responsibilities among team members was difficult at first, as not everyone was equally familiar with the concepts of MACH architecture".
Decision-Making Process	Can you describe the decision-making process for selecting the architectural pattern for your project?	Flexibility and scalability	"We chose a microservices approach because it allowed us to build components independently, which made debugging and scaling easier".
		Customer and business demands	"The MACH architecture aligned well with the client's demand for faster responses and multi-platform availability".
		Cost-effectiveness	"Deploying on a cloud-native environment seemed expensive initially but turned out to be cost-efficient in scaling and managing resources".
		Adaptability and future-proofing	"MACH principles offered a more future-proof solution, especially with the need for modularity and real-time updates in our project".

Table 9. *Cont.*

Topic	Question	Key Themes	Example Answers
Innovation in Design	Have you implemented any innovative approaches in your architectural design?	Technological innovation	"We implemented a custom API gateway to streamline communications between services and improve response time for complex queries".
		Efficiency and performance	"Using Docker containers allowed us to deploy microservices faster, and the system was more efficient during high-traffic testing phases".
		Security and transparency	"We integrated token-based authentication for enhanced security, which was a significant step for us in understanding industry practices".
Stakeholder Influence	How do the different stakeholders influence the architectural design?	Cross-functional influence	"Feedback from teaching assistants helped us simplify our API design, which made it easier to test and deploy".
		Customer and business needs	"Hypothetical client demands shaped how we implemented headless architecture to ensure multi-channel usability".
		Security and compliance	"Suggestions from security experts on best practices for data handling greatly influenced our compliance strategies".
Lessons Learned	What lessons have you learned from previous projects regarding architectural design?	Gradual adoption and planning	"Starting with smaller components helped us better integrate the microservices and avoid overwhelming complexity early on".
		Training and communication	"Discussing coding standards in the daily stand-ups reduced miscommunication and improved our overall code quality".
		Security and monitoring	"Setting up monitoring tools early in the project helped us quickly identify and fix issues, ensuring a smoother deployment process".
Adaptability of Design	How adaptable is your architecture to changes in scope, technology, or requirements?	Modularity and flexibility	"Adding new features was easier because of the modularity of our microservices, and they did not impact existing functionalities".
		Scalability	"We could scale up the system during testing phases by adjusting Kubernetes settings, which highlighted the importance of a cloud-native approach".
		Security and compliance	"When new security requirements were introduced, we could adapt quickly by updating our authentication mechanisms without affecting other services".
Documentation Practices	How do you manage architectural documentation, and what role does it play in your project?	Automation and integration	"We used Swagger for API documentation, which ensured our endpoints were well-documented and accessible to all team members".
		Centralization and accessibility	"Having a shared repository for documentation helped everyone stay on the same page and reduced confusion during sprints".
		Version control and consistency	"Using GitHub for version-controlled documentation allowed us to track changes and maintain consistency across iterations".

Table 9. *Cont.*

Topic	Question	Key Themes	Example Answers
Future Trends	What trends do you foresee in software architecture that might influence your future projects?	Serverless and edge computing	"I think serverless architecture will become more important, especially for handling event-driven workloads in scalable systems".
		AI and automation	"Automation of testing and deployment pipelines will continue to grow, making agile development more efficient".
		Security and privacy	"Blockchain integration could improve data security in future MACH-based applications, especially for decentralized environments".

6. Discussion

The action research approach employed in this study provided a structured yet flexible methodology for examining and refining the teaching of MACH architecture. The iterative cycles of planning, acting, observing, and reflecting created continuous opportunities for improvement and innovation in both the curriculum and instructional approaches.

The study showed a significant improvement in students' comprehension of MACH architecture concepts. This was achieved through the integration of agile methodologies, specifically through Scrum sprints, which facilitated a more dynamic learning environment. Agile's iterative nature aligned well with action research, as both emphasized constant reflection and adjustment.

The action research methodology was particularly effective in preparing students for real-world challenges. By focusing on industry-relevant practices, the course not only imparted technical skills, but also bridged the gap between academic learning and professional application.

Pre and post course surveys demonstrated an increase in students' knowledge, particularly in areas such as understanding microservices and cloud-native environments. This suggests that the agile sprint structure allowed students to build on their knowledge incrementally. Feedback from students and reflective journals indicated that the hands-on, project-based learning was highly effective. Students noted that breaking down large tasks into manageable sprint goals made the learning process less overwhelming and more structured.

This case study, which involved the development of a blockchain-based NFT trading platform, provided a tangible project that students could relate to industry trends. The iterative cycles of building microservices, integrating APIs, and deploying applications on cloud platforms mimicked real-world software development processes, making the learning experience more authentic.

The final project evaluations reflected that students could effectively apply their knowledge to design, build, and deploy a functioning system. The incorporation of continuous integration/continuous delivery (CI/CD) pipelines and Kubernetes for cloud-native deployment also mirrored industry standards.

The reflective phase of the action research cycles allowed for ongoing adjustments to the course content and teaching strategies. Surveys and interviews provided critical data on students' understanding, and the agile approach made it easy to incorporate feedback and refine the course as it progressed.

One notable outcome was the increased student confidence in implementing MACH principles. Post-course assessments showed significant gains in confidence levels, from an average of 2.4 to 4.5 on a 5-point scale. This indicates that the action research approach, combined with agile methodologies, fostered not only knowledge acquisition, but also self-assurance in applying that knowledge.

Instructor observations also revealed that students were more engaged and proactive during the sprints, likely due to the collaborative and iterative nature of both the action research and agile frameworks. The cycle of reflection and adaptation created a feedback loop that continually improved both the teaching and learning processes.

The reflective journals reveal a well-rounded student experience, characterized by both challenges and achievements. Students demonstrated significant growth in their technical knowledge, particularly in applying MACH principles and agile methodologies. However, they also highlighted areas for improvement, such as better preparatory resources and clearer initial guidance on project expectations.

The responses indicate that the course successfully simulated real-world software development environments, fostering critical skills such as collaboration, adaptability, and innovative problem-solving. These qualitative findings align with the course's quantitative results, reinforcing the overall effectiveness of the agile, hands-on approach.

The integration of agile methodologies into educational contexts has been increasingly recognized for its potential to improve learning outcomes and align academic practices with industry demands. A comprehensive review demonstrating that agile frameworks, such as Scrum, facilitate active learning by promoting iterative processes and team-based problem-solving [48]. Our study reinforces these findings, showcasing how the use of Scrum sprints in teaching MACH principles enhances student engagement and prepares them for real-world scenarios.

The application of agile methodologies in educational settings, emphasizing their adaptability to diverse learning environments, was proven earlier [49]. This aligns with our research, where the iterative cycles of agile sprints allowed students to navigate the complexities of MACH architecture through structured yet flexible learning activities. Their emphasis on adaptability mirrors the agility we observed in student teams as they tackled increasingly complex tasks across the sprints.

Further, researchers highlighted how agile methodologies can foster competencies related to sustainability and innovation. This perspective aligns with our findings, as students demonstrated not only technical proficiency, but also an enhanced ability to collaborate, iterate, and respond to evolving project requirements. The development of a blockchain-based NFT platform during our course exemplifies this intersection of innovative thinking and agile application [50].

Finally, effectiveness of agile learning in online collaborative environments, focusing on strategies to improve team regulation and project management, was also studied. While our study was conducted in a traditional classroom setting, the collaborative dynamics observed among student teams echo their findings. Agile principles, such as daily stand-ups and sprint retrospectives, cultivated an environment of accountability and iterative improvement, critical for managing the complexities of MACH-based projects.

These studies collectively underscore the transformative potential of agile methodologies in education. They highlight how iterative frameworks such as Scrum can bridge the gap between theoretical learning and practical application, equipping students with the skills and mindset needed to thrive in a rapidly evolving technological landscape. By integrating agile methodologies with MACH principles, our research contributes to this growing body of evidence, demonstrating the effectiveness of agile frameworks in fostering technical and collaborative competencies in higher education.

A key limitation of this study lies in the absence of a comprehensive final evaluation framework to measure the overall impact of the course and its methodologies on student outcomes. As the course was implemented for the first time, the focus was understandably placed on refining the delivery and ensuring that students effectively engaged with the new content. While iterative assessments and feedback during the course were valuable for real-time adjustments, the lack of a summative evaluation limits the ability to fully assess students' mastery of MACH principles and their readiness to apply these in professional contexts. This gap reduces the potential for a robust understanding of the long-term effectiveness of the methodologies employed.

To address this limitation, future iterations of the course could incorporate structured final evaluations, such as capstone projects, exams, or formalized reviews of student deliverables. These assessments would provide a holistic view of student competencies. Additionally, implementing longitudinal tracking, such as post-course surveys or follow-ups with industry stakeholders, could offer insights into the real-world applicability of students' skills. By comparing outcomes to industry standards and utilizing control groups, future research can provide more rigorous evidence of the course's effectiveness. Such measures would enhance the alignment of the curriculum with both academic goals and industry demands, ensuring the continual refinement of the educational approach.

7. Conclusions

This research explored the integration of agile methodologies into the teaching of microservices, API-first, cloud-native, and headless (MACH) architecture in a university environment. By employing an action research framework, the aim was to evaluate the effectiveness of agile frameworks, specifically Scrum, in enhancing student understanding and application of MACH principles. The results demonstrate that agile methodologies were not only effective in improving students' technical competencies, but also in fostering collaborative, iterative learning environments that mirror the real-world software development processes.

Through a 15-week MACH architecture course, students gained practical experience in designing, building, and deploying microservices-based systems using API-first principles, cloud-native development, and headless architecture. The agile approach allowed for a structured yet flexible learning process, with incremental development sprints enabling students to break down complex tasks into manageable stages. The iterative cycles of reflection and adaptation inherent to both agile and action research methodologies contributed to significant improvements in students' knowledge and confidence in applying MACH principles.

Quantitative assessments indicated substantial gains in student performance, with post-course evaluations showing a marked increase in both knowledge and confidence across all MACH components. Qualitative feedback from students further highlighted the value of the hands-on, project-based approach, noting that the agile sprints facilitated deeper engagement with the subject matter and better preparation for industry demands. The case study demonstrated the potential of agile frameworks to enhance educational outcomes in the rapidly evolving field of software architecture.

Furthermore, this study provides evidence that agile methodologies can be successfully integrated into higher education to teach emerging software architectures such as MACH. By aligning educational practices with industry needs, this approach equips students better with the skills necessary to thrive in a dynamic technological landscape. Future research could further investigate the scalability of this approach across different educational contexts and its applicability to other advanced technical subjects.

Author Contributions: Conceptualization, G.K. and S.U.; methodology, I.M.; software, P.M.; validation, E.M. and D.A.; formal analysis, E.M.; investigation, D.A. and E.M.; resources, E.M.; data curation, G.K.; writing—original draft preparation, P.M.; writing—review and editing, I.M.; visualization, I.M.; supervision, S.U. All authors have read and agreed to the published version of the manuscript.

Funding: Ministry of Science, Technological Development and Innovation of the Republic of Serbia contract no. 451-03-65/2024-03/200155, realized by the Faculty of Technical Sciences in Kosovska Mitrovica, University of Pristina.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available on request on e-mail emekic@np.ac.rs.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kapikul, A.; Savić, D.; Milić, M.; Antović, I. Application Development from Monolithic to Microservice Architecture. In Proceedings of the 28th International Conference on Information Technology (IT), Zabljak, Montenegro, 21–24 February 2024; pp. 1–4. [CrossRef]
2. Kumar, N. Fundamental Principles for Service-Oriented Computing Paradigm. In *Soft Computing Principles and Integration for Real-Time Service-Oriented Computing*; Auerbach Publications: Boca Raton, FL, USA, 2024; pp. 101–116.
3. Luthria, H.; Rabhi, F. Service Oriented Computing in Practice—An Agenda for Research into the Factors Influencing the Organizational Adoption of Service-Oriented Architectures. *J. Theor. Appl. Electron. Commer. Res.* **2009**, *4*, 39–56. [CrossRef]
4. Zein, A. Implementation of service-oriented architecture in mobile applications to improve system flexibility, interoperability, and scalability. *J. Inf. Syst. Technol. Eng.* **2024**, *2*, 171–174.
5. Cerny, T.; Donahoo, M.J.; Trnka, M. Contextual Understanding of Microservice Architecture: Current and Future Directions. *ACM SIGAPP Appl. Comput. Rev.* **2018**, *17*, 29–45. [CrossRef]
6. Beaulieu, N.; Dascalu, S.M.; Hand, E. API-First Design: A Survey of the State of Academia and Industry. In *ITNG 2022: 19th International Conference on Information Technology–New Generations*; Springer International Publishing: Cham, Switzerland, 2022; pp. 73–79. [CrossRef]
7. Laszewski, T.; Arora, K.; Farr, E.; Zonooz, P. *Cloud Native Architectures: Design High-Availability and Cost-Effective Applications for the Cloud*; Packt Publishing Ltd.: Birmingham, UK, 2018.
8. Gannon, D.; Barga, R.; Sundaresan, N. Cloud-native applications. *IEEE Cloud Comput.* **2017**, *4*, 16–21. [CrossRef]
9. Alonso, J.; Orue-Echevarria, L.; Casola, V.; Torre, A.I.; Huarte, M.; Osaba, E.; Lobo, J.L. Understanding the challenges and novel architectural models of multi-cloud native applications–a systematic literature review. *J. Cloud Comput.* **2023**, *12*, 6. [CrossRef]
10. Cao, X.A. *Headless CMS and Qwik Framework and Their Practicalities in the Future of Application Development*; Vassan Ammattiko-rkeakolou University of Applied Sciences: Vaasa, Finland, 2023.
11. Purkovic, S.; Fetahovic, I.; Mekic, E.; Katipoglu, G.; Utku, S. Innovative Approach to Teaching Distributed Systems in Education 4.0. *Int. J. Eng. Educ.* **2024**, *40*, 1–16.
12. Sherstobitova, A.A.; Glukhova, L.V.; Khozova, E.V.; Krayneva, R.K. Integration of Agile Methodology and PMBOK Standards for Educational Activities at Higher School. In *Smart Education and e-Learning 2020*; Springer: Singapore, 2020; pp. 339–349. [CrossRef]
13. Rocha, F.G.; Misra, S.; Soares, M.S. Guidelines for Future Agile Methodologies and Architecture Reconciliation for Software-Intensive Systems. *Electronics* **2023**, *12*, 1582. [CrossRef]
14. Elliot, J. *Action Research for Educational Change*; McGraw-Hill Education: Maidenhead, UK, 1991.
15. Feldman, A.; Minstrell, J. Action Research as a Research Methodology for the Study of the Teaching and Learning of Science. In *Handbook of Research Design in Mathematics and Science Education*; Routledge: London, UK, 2012; pp. 429–455.
16. Cain, T.; Milovic, S. Action Research as a Tool of Professional Development of Advisers and Teachers in Croatia. *Eur. J. Teach. Educ.* **2010**, *33*, 19–30. [CrossRef]
17. El Akhdar, A.; Baidada, C.; Kartit, A.; Hanine, M.; García, C.O.; Lara, R.G.; Ashraf, I. Exploring the Potential of Microservices in Internet of Things: A Systematic Review of Security and Prospects. *Sensors* **2024**, *24*, 6771. [CrossRef]
18. Larrea, M.; Estensoro, M.; Sisti, E. The contribution of action research to industry 4.0 policies: Bringing empowerment and democracy to the economic efficiency arena. *IJAR–Int. J. Action Res.* **2019**, *14*, 15–16.
19. Pontarolli, R.P.; Bigheti, J.A.; de Sá, L.B.R.; Godoy, E.P. Microservice-Oriented Architecture for Industry 4.0. *Eng* **2023**, *4*, 1179–1197. [CrossRef]
20. Bhat, V.S.; Bhat, S.; Gijo, E.V. Simulation-based lean six sigma for Industry 4.0: An action research in the process industry. *Int. J. Qual. Reliab. Manag.* **2021**, *38*, 1215–1245. [CrossRef]
21. Petersen, K.; Gencel, C.; Asghari, N.; Baca, D.; Betz, S. Action research as a model for industry-academia collaboration in the software engineering context. In Proceedings of the 2014 International Workshop on Long-Term Industrial Collaboration on Software Engineering, Vasteras, Sweden, 16 September 2014; pp. 55–62. [CrossRef]
22. Romero, C.A.T.; Ortiz, J.H.; Khalaf, O.I.; Ortega, W.M. Software architecture for planning educational scenarios by applying an agile methodology. *Int. J. Emerg. Technol. Learn.* **2021**, *16*, 132. [CrossRef]
23. Ng, L.-K.; Lo, C.-K. Enhancing Online Instructional Approaches for Sustainable Business Education in the Current and Post-Pandemic Era: An Action Research Study of Student Engagement. *Educ. Sci.* **2023**, *13*, 42. [CrossRef]
24. Tripp, D. Action research: A methodological introduction. *Educ. Pesqui.* **2005**, *31*, 443–466. [CrossRef]
25. Mackenzie, J.; Tan, P.L.; Hoverman, S.; Baldwin, C. The value and limitations of participatory action research methodology. *J. Hydrol.* **2012**, *474*, 11–21. [CrossRef]
26. MacDonald, C. Understanding participatory action research: A qualitative research methodology option. *Can. J. Action Res.* **2012**, *13*, 34–50. [CrossRef]
27. Baskerville, R.L. Investigating information systems with action research. *Commun. Assoc. Inf. Syst.* **1999**, *2*, 19. [CrossRef]
28. Kemmis, S. Action Research as a Practice-Based Practice. *Educ. Action Res.* **2009**, *17*, 463–474. [CrossRef]
29. Kratzke, N. Cloud-Native Applications and Services. *Future Internet* **2022**, *14*, 346. [CrossRef]
30. Bixio, L.; Delzanno, G.; Rebora, S.; Rulli, M. A Flexible IoT Stream Processing Architecture Based on Microservices. *Information* **2020**, *11*, 565. [CrossRef]

31. MACH Alliance. Enterprise MACHified. Available online: https://machalliance.org/ (accessed on 10 November 2024).
32. Shivakumar, S.K. Modern Web Integration Patterns. In *Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 327–357.
33. Baškarada, S.; Nguyen, V.; Koronios, A. Architecting Microservices: Practical Opportunities and Challenges. *J. Comput. Inf. Syst.* **2020**, *60*, 428–436. [CrossRef]
34. Mekić, E.S.; Jovanović, M.N.; Kuk, K.V.; Prlinčević, B.P. Enhancing Educational Efficiency: Generative AI Chatbots and DevOps in Education 4.0. *Comput. Appl. Eng. Educ.* **2024**, *32*, e22804. [CrossRef]
35. Kosińska, J.; Baliś, B.; Konieczny, M.; Malawski, M.; Zieliński, S. Toward the Observability of Cloud-Native Applications: The Overview of the State-of-the-Art. *IEEE Access* **2023**, *11*, 73036–73052. [CrossRef]
36. Sehring, H.W. On the Generation of External Representations of Semantically Rich Content for API-Driven Document Delivery in the Headless Approach. In Proceedings of the Fifteenth International Conference on Creative Content Technologies (CONTENT), Nice, France, 26–30 June 2023; pp. 17–22.
37. Newman, S. *Building Microservices*; O'Reilly Media Inc.: Sebastopol, CA, USA, 2021.
38. Zimmermann, O.; Stocker, M.; Lubke, D.; Zdun, U.; Pautasso, C. *Patterns for API Design: Simplifying Integration with Loosely Coupled Message Exchanges*; Addison-Wesley Professional: Boston, MA, USA, 2022.
39. Jovanović, M.; Mesquida, A.L.; Radaković, N.; Mas, A. Agile Retrospective Games for Different Team Development Phases. *J. Univers. Comput. Sci.* **2016**, *22*, 1489–1508.
40. Losada, B.; Urretavizcaya, M.; Gil, J.M.L.; Fernández-Castro, I. Applying Usability Engineering in InterMod Agile Development Methodology. A Case Study in a Mobile Application. *J. Univers. Comput. Sci.* **2013**, *19*, 1046–1065.
41. Vega, F.; Rodríguez, G.; Rocha, F.; dos Santos, R.P. Scrum Watch: A Tool for Monitoring the Performance of Scrum-Based Work Teams. *J. Univers. Comput. Sci.* **2022**, *28*, 98. [CrossRef]
42. Gucciardi, E.; Mach, C.; Mo, S. Student-faculty team teaching–A collaborative learning approach. *Mentor. Tutoring Partnersh. Learn.* **2016**, *24*, 441–455. [CrossRef]
43. García-Machado, J.J.; Martínez Ávila, M.; Dospinescu, N.; Dospinescu, O. How the support that students receive during online learning influences their academic performance. *Educ. Inf. Technol.* **2024**, *29*, 20005–20029. [CrossRef]
44. Aggarwal, P.K.; Sharma, R.; Khare, R.; Singh, S. E-commerce Application Using PHP and Web Development: A Review. In Proceedings of the 2023 International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 11–12 May 2023; pp. 755–758.
45. Messenlehner, B.; Coleman, J. *Building Web Apps with WordPress: WordPress as an Application Framework*; O'Reilly Media: Sebastopol, CA, USA, 2019.
46. Halimi, K.; Seridi-Bouchelaghem, H. A Web 3.0-Based Intelligent Learning System Supporting Education in the 21st Century. *J. Univers. Comput. Sci.* **2019**, *10*, 1373–1394.
47. Levy, M.; Hadar, I.; Aviv, I. Agile-Based Education for Teaching an Agile Requirements Engineering Methodology for Knowledge Management. *Sustainability* **2021**, *13*, 2853. [CrossRef]
48. Otero, T.F.; Barwaldt, R.; Topin, L.O.; Menezes, S.V.; Torres, M.J.R.; de Castro Freitas, A.L. Agile methodologies at an educational context: A systematic review. In Proceedings of the 2020 IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 21–24 October 2020; pp. 1–5.
49. López-Alcarria, A.; Olivares-Vicente, A.; Poza-Vilches, F. A systematic review of the use of agile methodologies in education to foster sustainability competencies. *Sustainability.* **2019**, *11*, 2915. [CrossRef]
50. Noguera, I.; Guerrero-Roldán, A.E.; Masó, R. Collaborative agile learning in online environments: Strategies for improving team regulation and project management. *Comput. Educ.* **2018**, *116*, 110–129. [CrossRef]