SYLLABUS

Service Oriented Architecture

University year 2025

1. Information regarding the programme

1.1. Higher education institution	**Babeş Bolyai University**
1.2. Faculty	**Faculty of Mathematics and Computer Science**
1.3. Department	**Department of Computer Science**
1.4. Field of study	**Computer Science**
1.5. Study cycle	**Master**
1.6. Study programme/Qualification	**Software Engineering**
1.7. Form of education	

2. Information regarding the discipline

2.1. Name of the discipline	**Service Oriented Architecture**				Discipline code	**MME8027**	
2.2. Course coordinator					**Lect. dr. Ioan Lazar**		
2.3. Seminar coordinator					**Lect. dr. Ioan Lazar**		
2.4. Year of study	2	2.5. Semester	3	2.6. Type of evaluation	E	2.7. Discipline regime	Mandatory

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**3**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**1**
3.4. Total hours in the curriculum	42	of which: 3.5 course	28	3.6 seminar/laboratory/project	**14**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					28
Additional documentation (in libraries, on electronic platforms, field documentation)					14
Preparation for seminars/labs, homework, papers, portfolios and essays					14
Tutorship					7
Evaluations					7
Other activities:					14

3.7. Total individual study hours	**36**
3.8. Total hours per semester	**120**
3.9. Number of ECTS credits	**8**

4. Prerequisites (if necessary)

4.1. curriculum	• Programming Fundamentals
4.2. competencies	• Good programming skills in at least one of the languages Java, C#

5. Conditions (if necessary)

5.1. for the course	
5.2. for the seminar /lab activities	

6.1. Specific competencies acquired [1]

Professional/essential competencies	● C 4.3 Identify models and methods adequate to real life problem solving ● C 2.1 Identify adequate software systems development methodologies ● C 1.1 Proper description of programming paradigms and language specific mechanisms, and identification of semantical an syntactical differences
Transversal competencies	● CT1 Apply organized and efficient work rules and responsible attitude towards didactical and research field, in order to creatively use work potential; respect professional ethical principles ● CT3 Use efficient methods and techniques for: learning, information search, research and development of capacities to adapt to the requirements of a dynamic society and to communicate in an international language

6.2. Learning outcomes

Knowledge	The student knows: development cycle of systems based on services
Skills	The student is able to develop a system based on services
Responsibility and autonomy:	The student has the ability to work independently to build SOA systems

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	● Enhance the students understanding of service oriented concepts through a practical and pragmatic approach ● Provide the students with an environment in which they can explore the usage and usefulness of service oriented concepts in various business scenarios ● Induce a realistic and industry driven view of software design concepts such as design patterns and their inherent benefits

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

7.2 Specific objective of the discipline	● Give students the ability to explore various object oriented programming languages ● Improve the students abilities to tackle business requirements ● Enhance the students understanding of business needs and business value ● Provide students with insights into the way of working towards achieving high quality software through skilled trainers from the IT industry

8. Content

8.1 Course	Teaching methods	Remarks
1. Servers exposing REST services Distributed service design - Stateful versus stateless protocols and services - CRUD operations - Search operations References - FHIR specification, https://www.hl7.org/fhir/http.html - KOA framework, http://koajs.com/ 2. Server-side notifications Distributed service design - Reactive (IO-triggered) and multithreaded designs - ReactiveX, http://reactivex.io/rxjs/ Distributed message sending - Web Sockets - Web sockets API, https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API 3. Securing client-server applications Web security model - Browser security model including same-origin policy - Client-server trust boundaries - JSON Web Tokens, https://jwt.io/ - OAuth, https://oauth.net/2/ Client-side security - Web tokens - Web user tracking 4. Microservices Cloud services - Software as a service - Security - Seneca framework, http://senecajs.org/ 5. Containers Virtualization - Multiple virtual cloud servers - Deploy servicess on multiple servers - Migration of processes Docker	Exposure: description, explanation, examples, discussion of case studies	

- https://www.docker.com/ Explain the advantages and disadvantages of using virtualized infrastructure. 6. Command query responsibility segregation - Separating the update and read operations - CQRS, https://martinfowler.com/bliki/CQRS.html 7. Application architecture based on events - Domain event, event collaboration, event sourcing, aggreemment dispatcher, parallel model - Further patterns of EAA, https://martinfowler.com/eaaDev/ 8. Integration patterns - Messaging systems - Messaging channels - Enterprise integration patterns, http://www.enterpriseintegrationpatterns.com/ 9. Integration patterns - Message construction - Message routing 10. Advanced message queuing protocol - Routing, topics, work queue, publish/subscribe, RPC - RabbitMQ, https://www.rabbitmq.com/getstarted.html 11. Serverless architectures - Backend as a service - Function as a service - https://martinfowler.com/articles/serverless.html 12. IoT applications and services - IoT devices, platforms, services 13. Documenting systems using c4 models The C4 model for visualising software architecture, https://c4model.com/ 14. Documenting systems using UML models		
Bibliography		
8.2 Seminar / laboratory	Teaching methods	Remarks
1. Modern web apps 1.1 PD/Distributed Systems, Usage Implement a simple server: - exposing rest services (CRUD, search) - sending notifications 1.2 PL/Event-Driven and Reactive Programming, Usage [1h] Implement a client app: - using reactive handlers 2. Modern web apps	Dialogue, debate, case studies, examples, proofs	

Use client-side security capabilities in an application. 3. Creating a system based on microservices Describe the scalability challenges associated with a service growing to accommodate many clients. Explain strategies to synchronize a common view of shared data across a collection of devices. Deploy an application that uses cloud infrastructure for computing and/or data resources. 4. Synchronizing servers Use integration patterns to synchronize servers 5. Services implemented using AMQP Use AMQP messaging brokers to implement services 6. Systems based on serverless architectures Provide and consume services defined according to BaaS and FaaS 7. Documenting software systems using c4 & UML models		
Bibliography		

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course respects the IEEE and ACM Curriculla Recommendations for Computer Science studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course is considered by the software companies as important for average programming skills.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	- know the basic principle of the domain; - apply the course concepts	Written exam	30%

	- problem solving		
10.5 Seminar/laboratory	Implement a system with REST services, server side notifications, and data synchronization	Project grading	70%
10.6 Minimum standard of performance			
• A minimum passing grade is defined by attaining at least 50% (5/10) points for the final project and each of the three lab assignments respectively. • No more than 3 absences are allowed for the seminar/lab activities			

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date:	Signature of course coordinator	Signature of seminar coordinator
.30.04.2025	Lect. dr. Ioan Lazar	Lect. dr. Ioan Lazar

Digitally signed by
IOAN LAZAR
Date: 2025.05.04 17:59:44
+03'00'

Date of approval:	Signature of the head of department
...	Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable*.".