BO·TT·EGA
IT minds

Training program:

Modular Monolith Architecture .NET Core

Info:

Name:	**Modular Monolith Architecture .NET Core**
Code:	**NET-arch-monolith**
Category:	.NET Architecture
Target audience:	developers architects
Duration:	3 days
Format:	30% lecture / 70% workshop

The Modular Monolith training is dedicated to people and teams who are looking for techniques and patterns supporting the logical division of a monolithic solution into independent modules, ensuring a compromise between the advantages of developing a single application and the modularity and autonomy of independent services in the microservices approach.

During the training, participants will learn the techniques and challenges related to the division of a monolith into independent parts (so-called vertical slice) and will focus on an example module implementation in a dedicated project.

It's all about the content.

- Pragmatic modularisation
- Microservices-ready arch
- Healthy modules boundaries

BO·TT·EGA
IT minds

Training program

1. Theory

1.1. Division of systems in terms of modularity and dispersion

1.2. "Classic" vs. modular monolith

1.3. Advantages of implementing a modular monolith

2. Application architecture

2.1. Clean architecture

2.2. Basic building blocks from Domain-Driven Design

2.3. CQS/CQRS

3. Implementation

3.1. Application overview, structure of files / directories / projects in a walkthrough.

3.2. Implementation of the new module as a vertical slice.

4. Communication between modules

4.1. Ways of communication between modules

4.2. Local contracts vs. shared contracts

4.3. Implementation of a modular communication mechanism

5. Integration between modules

5.1. Synchronous integration (write transactional)

5.2. Asynchronous integration between modules. Discussion of advantages and disadvantages (fire and forget vs async / await)

5.3. Implementation of the modular integration mechanism

5.4. Shared infrastructure and error handling

6. Distributed business processes

6.1. Overview of existing solutions

6.2. A practical application of the Saga pattern

BO·TT·EGA
IT minds

7. Testing

7.1. Unit tests

7.2. Integration tests

7.3. End-to-end tests

7.4. Contract tests

7.5. Performance tests

8. Deployment strategies

8.1. Overview of mechanisms provided by the framework

8.2. Dynamic implementation of modules

9. Optional 4th day Day - Extraction of a microservice from the module

9.1. The challenges of microservices architecture

9.2. Refactoring modular communication to HTTP communication

9.3. Replacing modular integration with communication using a message broker

9.4. Adaptation of the monolith's infrastructure to support two types of communication, i. e. inside and outside the process