Training program:

Node.js Application Architecture

Info:

Name:	**Node.js Application Architecture**
Code:	**modern-Node**
Category:	Node.js
Target audience:	developers architects
Duration:	2-3 days
Format:	20% lecture / 80% workshop

Node.js provides more freedom compared to traditional enterprise ecosystems. Although some individuals are attempting to transform Node.js into a framework-heavy enterprise platform, there is an alternative approach.

In this course we will learn how to build highly **composable, type-safe** and **testable** systems **without a typical enterprise framework magic.** If you have an aversion to magic or wish to unlearn years of practices such as using:

- Dependency Injection containers
- Class-based programming
- ORMs
- Technology-driven architectures
- Magical conventions
- @Transactional dandruff
- new Date() spread all over the place
- Handwritten DTOs
- Import mocking frameworks
- Reflection and metaprogramming magic

This course is for you. The alternative to heavy enterprise frameworks is not Wild Wild West or building a custom framework. The real alternative is to **embrace the true nature of JS/TS** , make full use of **the good parts** and never get prisoned in another heavy framework again.

What will I learn?

- Compose your building blocks with multiple composition roots and manual dependency injection
- Make full use of extremly composable functions instead of less composable classes
- Write type-safe DB access code that stays in sync with DB schemas and prevents you from mistakes
- Parse don't validate your input and output data to be liberal in what you accept and strict in what you produce
- Organise your code around features, not technology layers
- Handle DB transactions without magic
- Program in a language, not in a framework
- Test with high ROI with smart testing strategies
- Make code testable and don't let more than one invocation of new Date() sneak into your code
- Start your app in full in-memory mode by default
- Separate write model and read model

- Master transferrable skills that will outlive your enterprise framework du jour

BOTTEGA IT minds

Training program

1. Engineering techniques and principles

1.1. Minimal dependencies

1.2. Curried functions over classes

1.3. Parse don’t validate

1.4. Robustness principle

1.5. Full in-memory mode by default

1.6. Feature-driven architecture

1.7. Composition over convention

1.8. End-to-end type safety

1.9. Programming in a language, not in a framework

1.10. Feature-flag driven development

2. Architectural building blocks and concepts

2.1. Pipes and filters (express.js middleware and handlers)

2.2. Ports and adapters

2.3. Routers/controllers (express.js)

2.4. Application services

2.5. Repositories

2.6. Error handlers

2.7. Domain types and tiny types

2.8. Input and output parsers

2.9. DTOs for free

2.10. Composition roots

2.11. Write model vs read model

2.12. External read model (view model) vs internal read model

2.13. Application scope and request scope

3. Testing strategies

3.1. Unit of behavior tests (mocha)

3.2. Integration tests (mocha)

3.3. Component tests (supertest)

3.4. Testing without mocks

3.5. Testing difficult dependencies (time, id generation)

3.6. Data cleanup strategy

4. Integration with a SQL database

4.1. DB migrations

4.2. Type generation from DB schema

4.3. Verifying repository contract

4.4. SQL query builder

4.5. Handling DB transactions

5. Type-safety

5.1. Type-safe config (zod)

5.2. Type-safe DB queries (kysely)

5.3. Type-safe input and output (zod)

5.4. Type-safe dependency injection

5.5. Type-safe API paths (static-path)