Comparison of Architecture Patterns by implementing a simple application

Git Repo: https://github.com/daniel-riffi/ma-saap-ex03.git

Architecture Patterns

We compare two architecture patterns by implementing the same simple application using both approaches. We therefore implement a layered (N-Tier) architecture and a one-layer (kind of MVC) architecture.

Layered (N-Tier) Architecture

We strictly devide the application into distinct layers:

- Controller Layer
- Service Layer
- Business Layer
- Repository Layer

Controllers handle HTTP requests and responses, services validate and preprocess data, business layer contains core business logic, and repositories interact with the database.

Advantages:

- Clear separation of concerns.
- Easier to maintain and test individual layers.
- Scalability as layers can be modified independently.

Disadvantages:

- Increased complexity due to multiple layers.
- Potential performance overhead due to layer interactions.
- Requires careful design to avoid tight coupling between layers.

One-Layer (Kind of MVC) Architecture

In this architecture, we separate the application into three main components:

- View: Responsible for rendering the user interface.
- Controller: Manages user input and handles the business logic.

All logic including business logic and data access is handled within the controller component. This leads to a bloating of the controller, since it combines multiple responsibilities and layers from the layered architecture. The view was not implemented in our example, since we only focus on the backend part of the application.

Advantages:

- Simplicity in design and implementation.
- Easier to understand for small applications.
- Reduced number of components --> better overview.

Disadvantages:

- Poor separation of concerns.
- Difficult to maintain and scale as the application grows --> bloated models.
- Testing can be more challenging due to tightly coupled components.

Application Description

The application is implemented as a small Spring Boot project that manages user accounts. For our test scenario, we just implement the register endpoint that allows users to create an account. This includes an E-Mail notification upon successful registration.

The core application is implemented twice, once using the layered architecture and once using the strict MVC architecture. Both implementations provide the same functionality.