

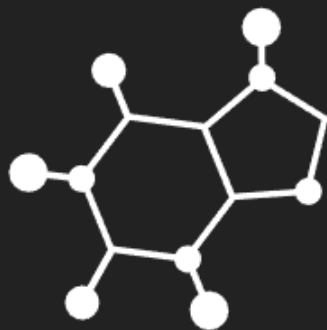


Arquitectura Moderna de Servicios en Java

Círculo Siete Capacitación

Clase 7 de 30
15 Octubre 2024

<https://xmolecules.org/>



xMolecules

Architectural abstractions in code

No all code is born equal. While some elements of a code base are of supportive nature, others represent significant architectural concepts and design patterns. Traditionally, developers have tried to express those concepts through naming conventions.

xMolecules is an umbrella project providing means to express architectural concepts in programming language native metadata and – if applicable – right within the type system.

Read more about the project in the “*Architecturally Evident Applications – How to Bridge the Model Code Gap?*” [whitepaper](#).

[jMolecules](#) – for Java

| [nMolecules](#) – for .NET

| [phpMolecules](#) – for PHP

Motivación

- Desafío en software empresarial de larga duración: Evolución constante debido a cambios en los requisitos.
- Importancia de la arquitectura y el diseño: Facilitar la comprensión del sistema y su modificación.
- Brecha entre diseño arquitectónico y código: La implementación dispersa de conceptos arquitectónicos puede llevar a un aumento de la complejidad y errores.

Solución Propuesta

- **jMolecules**: Un enfoque que permite a los desarrolladores expresar explícitamente patrones arquitectónicos en el código a través de anotaciones e interfaces en Java.
- Beneficios:
 - Comprensibilidad: El código refleja mejor los conceptos arquitectónicos.
 - Documentación precisa: Generada automáticamente desde el código.
 - Verificación de reglas: Asegura conformidad con los conceptos de diseño.
 - Reducción de código boilerplate: Automatiza detalles técnicos (p. ej., mapeo de persistencia).

Conceptos Arquitectónicos


- Bloques de construcción de DDD:
 - Agregados
 - Entidades
 - Repositorios
- Estilos arquitectónicos soportados:
 - Arquitectura en capas
 - Arquitectura onion
 - CQRS (Command Query Responsibility Segregation)
 - Hexagonal
- Anotaciones y tipos en jMolecules: Conectar código y arquitectura.

Arquitectura Limpia y jMolecules

- jMolecules se alinea con Clean Architecture y DDD, entre otras...
- Facilita la separación clara de capas y responsabilidades
- Ofrece componentes clave: Agregados, Entidades, Value Objects, Repositorios

Implementación de un Pedido (Order)

- **Desventaja tradicional:** Código lleno de detalles técnicos (JPA, JSON, etc.).
- **Ventaja jMolecules:** Elimina el código técnico, dejando solo la lógica de dominio.



```
@AggregateRoot
class Order {
    @Entity
    class LineItem { ... }
    @AggregateRoot
    class Customer { ... }
}
```

Herramientas de Verificación

- ArchUnit y jQAssistant:
 - Verifican que la implementación respete las reglas arquitectónicas del modelo.
 - Evitan errores en la proyección del diseño en el código.

Documentación Automática

- Generación de diagramas UML y Canvas del Módulo:
 - Basados en metadatos arquitectónicos presentes en el código.
 - Correctos por definición, ya que se derivan directamente del código.

Reducción de Boilerplate

- Implementación tradicional:
 - Anotaciones JPA y otras dependencias técnicas.
- Con jMolecules:
 - El código es más limpio y directo, manteniendo la semántica arquitectónica explícita.

Objetivos

- Separar claramente las capas de dominio, aplicación e infraestructura
- Evitar la sobreingeniería: mantener el código simple y legible
- Revisar regularmente la correspondencia entre el diseño del dominio y el código

jQAAssistant

Your Software. Your Structures. Your Rules.



scan

+



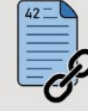
document

=



validate

INTEGRATION



SCAN & ANALYSE

DOKUMENTATION

jQAssistant

- jQAssistant es una herramienta para el análisis y verificación de arquitecturas de software.
- Permite escanear artefactos de software, definir reglas arquitectónicas y generar reportes sobre violaciones.
- Utiliza la base de datos de grafos Neo4j para modelar y consultar la estructura del software.

Características

- 1 Escaneo de artefactos: Clases Java, XML, estructuras de bases de datos.
- 2 Uso de reglas personalizables en Cypher para validar dependencias y arquitecturas.
- 3 Informes automáticos de violaciones arquitectónicas.
- 4 Integración con herramientas como Maven.
- 5 Basado en Neo4j: Aprovecha el modelado gráfico para consultas complejas.

Reglas

- Las reglas se expresan en el lenguaje Cypher de Neo4j, y permiten:
- Definir conceptos arquitectónicos (módulos, capas, controladores, etc.).
- Validar restricciones y dependencias entre clases y módulos.
- Generar reportes automáticos en cada proceso de construcción.

Beneficios

- Mantiene la integridad arquitectónica del software.
- Soporta refactorización hacia arquitecturas limpias o hexagonales.
- Facilita la detección de violaciones arquitectónicas en etapas tempranas.
- Se integra fácilmente con el flujo de desarrollo continuo.