

Validando una Arquitectura con ArchUnit

Con Víctor Madrid



Artículos Referenciados

Basado en la serie de artículos “**Validando una Arquitectura con ArchUnit**” publicados en enmilocalfunciona.io

- **Validando una Arquitectura con ArchUnit (Parte 1)**: Artículo de introducción a Arquitecturas y a los problemas típicos que trataremos de solucionar
- **Validando una Arquitectura con ArchUnit (Parte 2)**: Artículo donde se enseña a utilizar el framework / herramienta [ArchUnit](#)
- **Validando una Arquitectura con ArchUnit (Parte 3)**: Artículo de aplicación de lo anterior sobre un ejemplo “real” y avanzado basado en Spring / Spring Boot

Repositorio de Código

Publicados los ejemplos de forma conjunta en github.com/vjmadrid/enmilocalfunciona-archunit



02

"Existen tantas formas de hacer las cosas como formas de beber agua"



Photo by [Gio Almonte](#) on [Unsplash](#)

03

Definir una Arquitectura = Tomar Decisiones



Photo by Cenk Batuhan Ozaltun on [Unsplash](#)

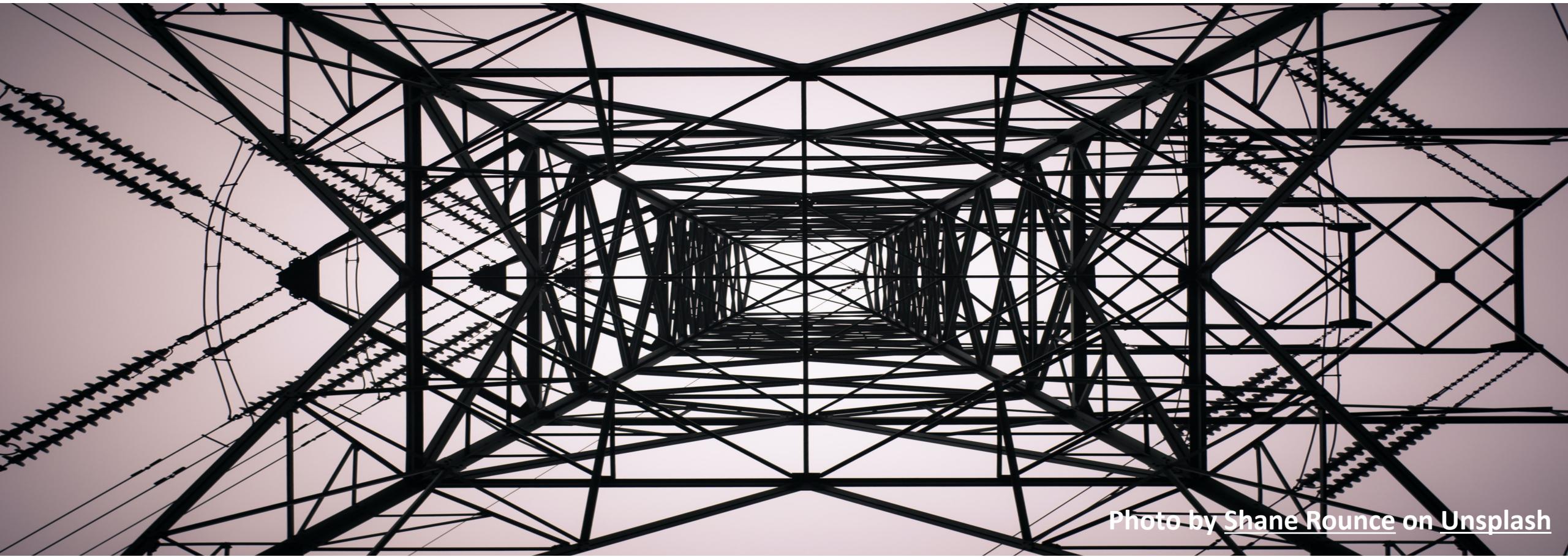


Photo by [Shane Rounce](#) on [Unsplash](#)

**Cada enfoque de estilo arquitectónico resuelve un problema técnico concreto
NO existe un estilo arquitectónico que lo resuelva “todo”
Cada uno de los estilos arquitectónicos tiene pros y contras**



Las “Señales” ...

Organización del código
 (clases, compilados, paquetes, test, configuración)

Nomenclatura (proyectos, paquetes, clases, atributos, etc.)

Selección de Tecnologías

Convenciones de código

Patrones bien diseñados

Cumplimiento de Requisitos (Funcionales y NO Funcionales)

Evitar acoplamiento

...



Cuando las señales...

Aparecen

NO aparecen

Sensación de Confianza

"Olores" de Mala Arquitectura

Mantenibilidad

Inmantenibilidad

Código repetido, complejidad innecesaria, rigidez, fragilidad, viscosidad, etc.





Photo by [James Sutton](#) on [Unsplash](#)



¿Qué cosas hay que “conocer” para hacer una auditoría de código “bien”?

- *Tecnologías*
- *Formas de desarrollo*
- *Contextos corporativos o de proyecto del porqué se hicieron las cosas de una determinada manera*
- *Las decisiones tomadas por los arquitectos y desarrolladores en el momento del desarrollo*
- *Procedimientos de desarrollo de ese momento*
- *Cosas que estaban de moda*
- ...

Reglas Arquitectónicas

Contexto

Validar una **arquitectura** incluye **verificar/probar** ciertos **aspectos** muy **variados**

Normalmente se realiza sobre una aplicación o componente (en concreto sobre su implementación de código) y se trata de **verificar si cumple una serie de reglas/patrones de arquitectura definidas** -> Reglas de Arquitectura o Reglas Arquitectónicas

Aspectos a cubrir

Convenciones de nomenclatura: paquetes, clases, métodos, etc.

Convenciones de codificación: atributos, métodos, privados, públicos, constructores, inicializadores, etc.

Características (Implementación, etc.)

Contenido : paquetes, clases, etc.

Dependencias / Relaciones : entre capas, entre paquetes, entre clases, etc.

Herencia

Uso de anotaciones según ámbito : persistencia, transaccionalidad, seguridad, custom, etc.

Existencia de ciclos de uso

Restricciones

...



Representación en código

Mantener



Evolucionar

Descartar



Herramienta / Tecnología JAVA

Ayuda para validar una arquitectura existente o nueva

Identificación de Reglas Arquitectónicas

Uso mayormente desatendido

Sencilla integración con el código

Uso durante el proceso de construcción

Detección temprana de "problemas"



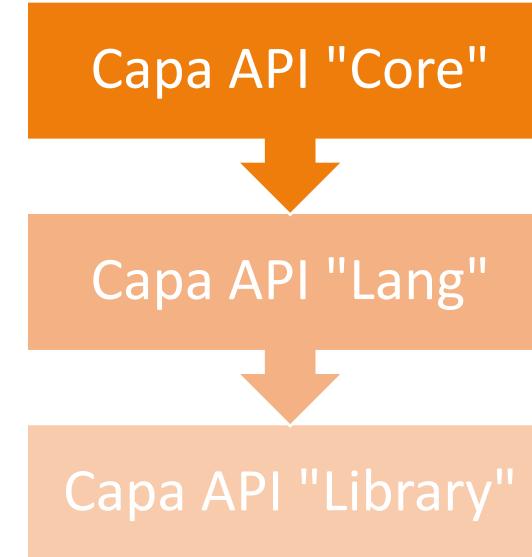
**Verificación Automatizada
de la Arquitectura**



Librería Open Source para **testing de Java** que tiene por objetivo **validar/probar** que una aplicación (en concreto su implementación de código) cumple una serie de **reglas/patrones de arquitectura definidas** -> Reglas Arquitectónicas

Características :

- **Uso mediante el lenguaje Java** -> Cualquier lenguaje que traducen al código de bytes de Java (Kotlin, etc.)
- **NO supone una gran carga sobre los proyectos** -> Espacio ocupado, requerimientos de HW, etc.
- **NO requiere ninguna infraestructura "especial"**
- **Uso con cualquier tipología y tamaño de proyecto**
- Internamente hace uso del **API de Reflection** y el **análisis de bytecodes** (basado en [ASM](#))
- **Integración** en proyectos como **testing unitario automatizado** -> fácil de integra con código viejo y nuevo
- Permite **definir** una **suite** de reglas arquitectónicas -> colección de tests de arquitectura
- Proporciona un **API** completo **con implementaciones** propias y/o personalizadas (customizadas) para el cumplimiento de las reglas arquitectónicas definidas
- Proporciona la **posibilidad** de **ampliar** el **API** con implementaciones específicas de las reglas arquitectónicas mediante el **uso de "condiciones"** -> personalización y extensión
- Facilita la **evolución** de las **reglas de arquitectura** a la vez que evoluciona la arquitectura y/o la aplicación
- Proporciona un **feedback directo**
- **Facilita la documentación** de la representación de una **arquitectura**
- **Facilita el traspaso de conocimiento** a otros desarrolladores o a otros equipos
- **Integración** con los procesos de **construcción automática, CI/CD**, etc.
- Facilita una buena [documentación](#) para su uso y [ejemplos oficiales](#) asociados Existe la opción de usar un plugin directo para Maven ...



Explicación de la DEMO



DIGITAL
EXPERIENCE
SCHOOL



DIGITAL
WOLVES

atsistemas



Recordatorio de los Objetivos Generales

Herramienta / Tecnología JAVA

Ayuda para validar una arquitectura existente o nueva

Identificación de Reglas Arquitectónicas

Uso mayormente desatendido

Sencilla integración con el código

Uso durante el proceso de construcción

Detección temprana de "problemas"

Objetivos de la DEMO

Disponer de un conjunto de artefactos Java que ayuden a validar la arquitectura de los proyectos

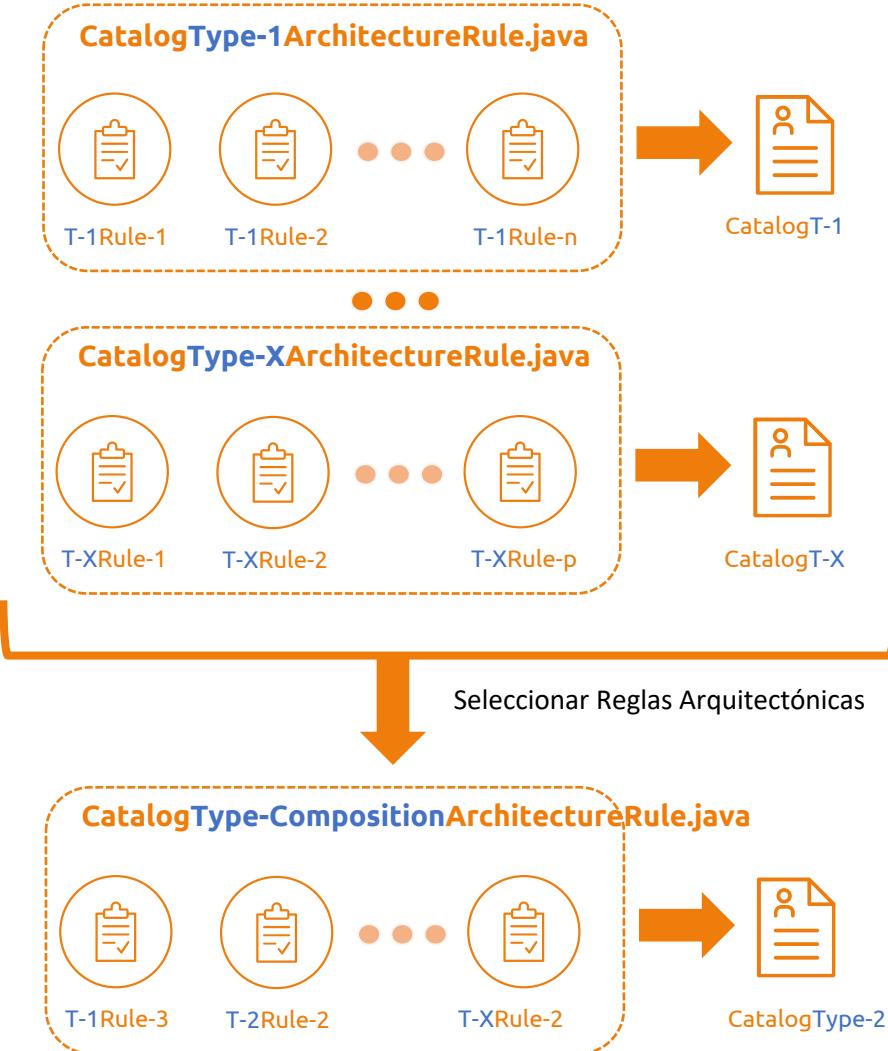
Proporcionar un catálogo inicial de Reglas Arquitectónicas que se encuentre operativo desde un inicio

Proporcionar un soporte de validación ante el uso de diferentes Stacks Tecnológicos (Por ejemplo : Spring, etc.) y diferentes tipos de clases

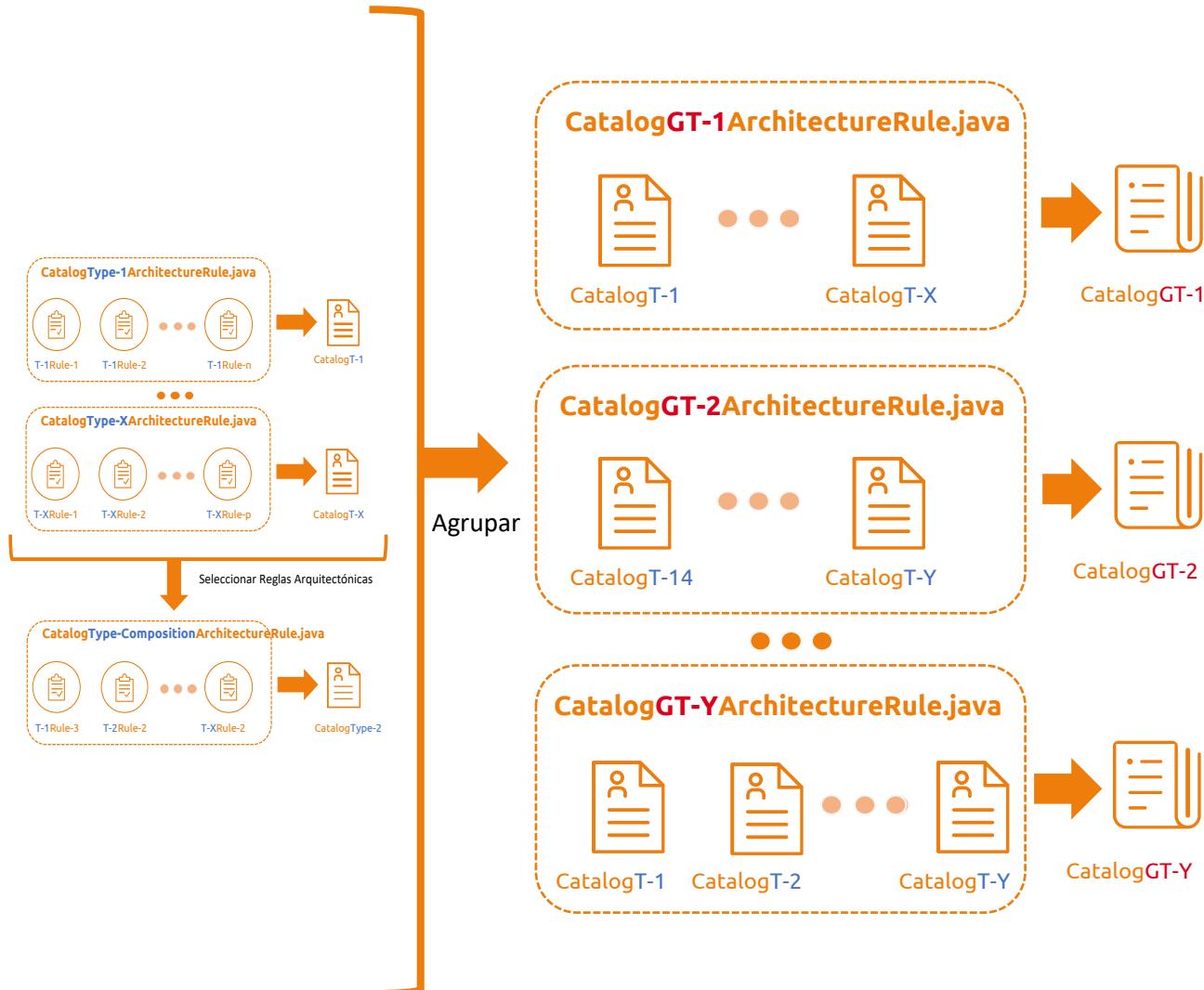
Proporcionar uno o varios procedimientos de uso en los proyectos sencillo, configurable, customizable y poco intrusivo

Demostración de uso con un caso lo más "real" posible

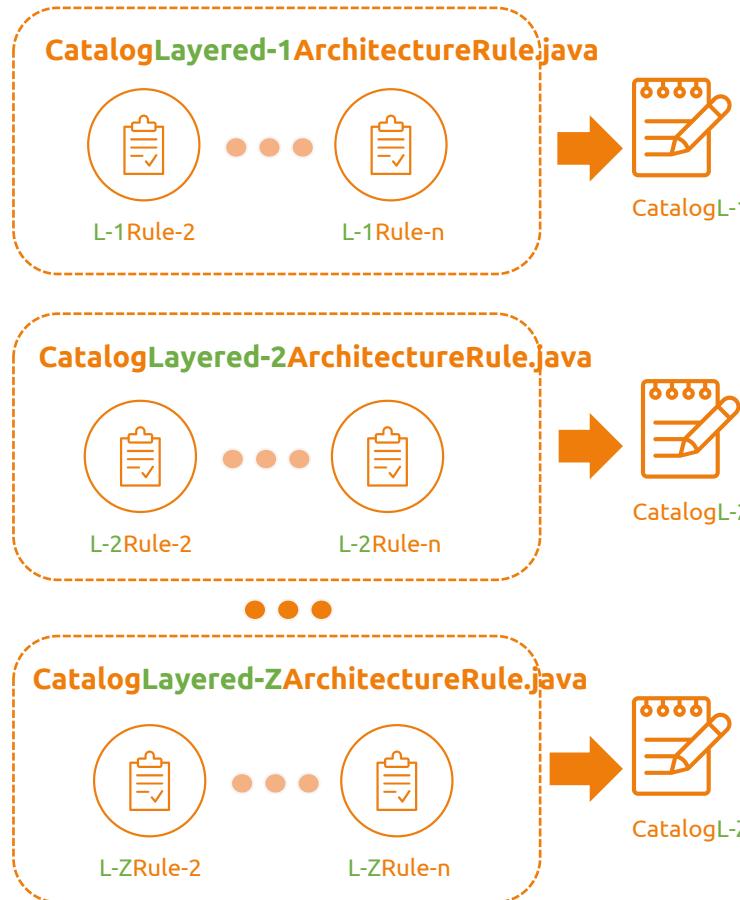




```
1 package com.acme.architecture.testing.spring.archunit.rule.catalog;
2
3 import static com.tngtech.archunit.lang.syntax.ArchRuleDefinition.classes;
4
5 public class CatalogSpringRepositoryArchitectureRule {
6
7     // Common
8
9     @ArchTest
10    public static final ArchRule spring_repository_classes_should_be_in_spring_repository_package = CatalogRepositoryArchitectureRule.repository_interface_classes_should_be_in_repository_package();
11
12    @ArchTest
13    public static final ArchRule spring_repository_classes_should_have_names_ending_with_the_word_repository = CatalogRepositoryArchitectureRule.repository_interface_should_have_names_ending_with_the_word_repository();
14
15    @ArchTest
16    public static final ArchRule repository_interface_classes_should_be_in_repository_package = CatalogRepositoryArchitectureRule.repository_interface_classes_should_be_in_repository_package();
17
18    @ArchTest
19    public static final ArchRule repository_interface_should_have_names_ending_with_the_word_repository = CatalogRepositoryArchitectureRule.repository_interface_should_have_names_ending_with_the_word_repository();
20
21    @ArchTest
22    public static final ArchRule repository_interface_should_be_in_repository_package = CatalogRepositoryArchitectureRule.repository_interface_should_be_in_repository_package();
23
24    @ArchTest
25    public static final ArchRule repository_interface_should_have_names_ending_with_the_word_repository = CatalogRepositoryArchitectureRule.repository_interface_should_have_names_ending_with_the_word_repository();
26
27    @ArchTest
28    public static final ArchRule repository_interface_should_be_in_repository_package = CatalogRepositoryArchitectureRule.repository_interface_should_be_in_repository_package();
29
30    @ArchTest
31    public static final ArchRule repository_interface_should_have_names_ending_with_the_word_repository = CatalogRepositoryArchitectureRule.repository_interface_should_have_names_ending_with_the_word_repository();
32
33    @ArchTest
34    public static final ArchRule repository_interface_should_be_in_repository_package = CatalogRepositoryArchitectureRule.repository_interface_should_be_in_repository_package();
35
36    @ArchTest
37    public static final ArchRule repository_interface_should_have_names_ending_with_the_word_repository = CatalogRepositoryArchitectureRule.repository_interface_should_have_names_ending_with_the_word_repository();
38
39    // Specific
40
41    @ArchTest
42    public static final ArchRule noClasses() = CatalogRepositoryArchitectureRule.noClasses();
43
44    .that().resideInAnyPackage()
45    .and().haveSimpleNameEndingWith(ArchUnitNameConstant.SUFFIX_NAME_REPOSITORY_CLASS)
46    .should().resideInAPackage(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_CLASS);
47
48    @ArchTest
49    public static final ArchRule repository_interface_should_have_names_ending_with_the_word_repository = CatalogRepositoryArchitectureRule.repository_interface_should_have_names_ending_with_the_word_repository();
50
51    .that().resideInAnyPackage()
52    .and().haveSimpleNameEndingWith(ArchUnitNameConstant.SUFFIX_NAME_REPOSITORY_CLASS);
53
54    @ArchTest
55    public static final ArchRule repository_implementations_should_be_in_repository_implementations_package = CatalogRepositoryArchitectureRule.repository_implementations_should_be_in_repository_implementations_package();
56
57    .that().resideInAnyPackage()
58    .and().haveSimpleNameEndingWith(ArchUnitNameConstant.SUFFIX_NAME_REPOSITORY_IMPL_CLASS);
59
60    @ArchTest
61    public static final ArchRule repository_implementations_should_be_in_repository_implementations_package = CatalogRepositoryArchitectureRule.repository_implementations_should_be_in_repository_implementations_package();
62
63    @ArchTest
64    public static final ArchRule repository_implementations_should_have_names_ending_with_the_word_repository_implement = CatalogRepositoryArchitectureRule.repository_implementations_should_have_names_ending_with_the_word_repository_implement();
65
66    .that().resideInAnyPackage(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_IMPL_CLASS)
67    .should().haveSimpleNameEndingWith(ArchUnitNameConstant.SUFFIX_NAME_REPOSITORY_IMPL_CLASS);
68
69    @ArchTest
70    public static final ArchRule repository_implementations_should_be_public = CatalogRepositoryArchitectureRule.repository_implementations_should_be_public();
71
72    .that().resideInAPackage(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_IMPL_CLASS)
73    .should().bePublic();
74
75    @ArchTest
76    public static final ArchRule repository_implementations_should_implement_repository = CatalogRepositoryArchitectureRule.repository_implementations_should_implement_repository();
77
78    .that().resideInAPackage(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_IMPL_CLASS)
79    .should(new ImplementInterfaceWithSameNameArchitectureCondition());
80
81    @ArchTest
82    public static final ArchRule repository_implementations_should_no_be_interface = CatalogRepositoryArchitectureRule.repository_implementations_should_no_be_interface();
83
84    .that().resideInAPackage(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_IMPL_CLASS)
85    .should().notBeInterfaces();
86
87    @ArchTest
88    public static final ArchRule repository_implementations_should_be_in_repository_implementations_package = CatalogRepositoryArchitectureRule.repository_implementations_should_be_in_repository_implementations_package();
89
90    .that().resideInAPackage(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_IMPL_CLASS)
91    .should().notBeInterfaces();
92
93 }
```



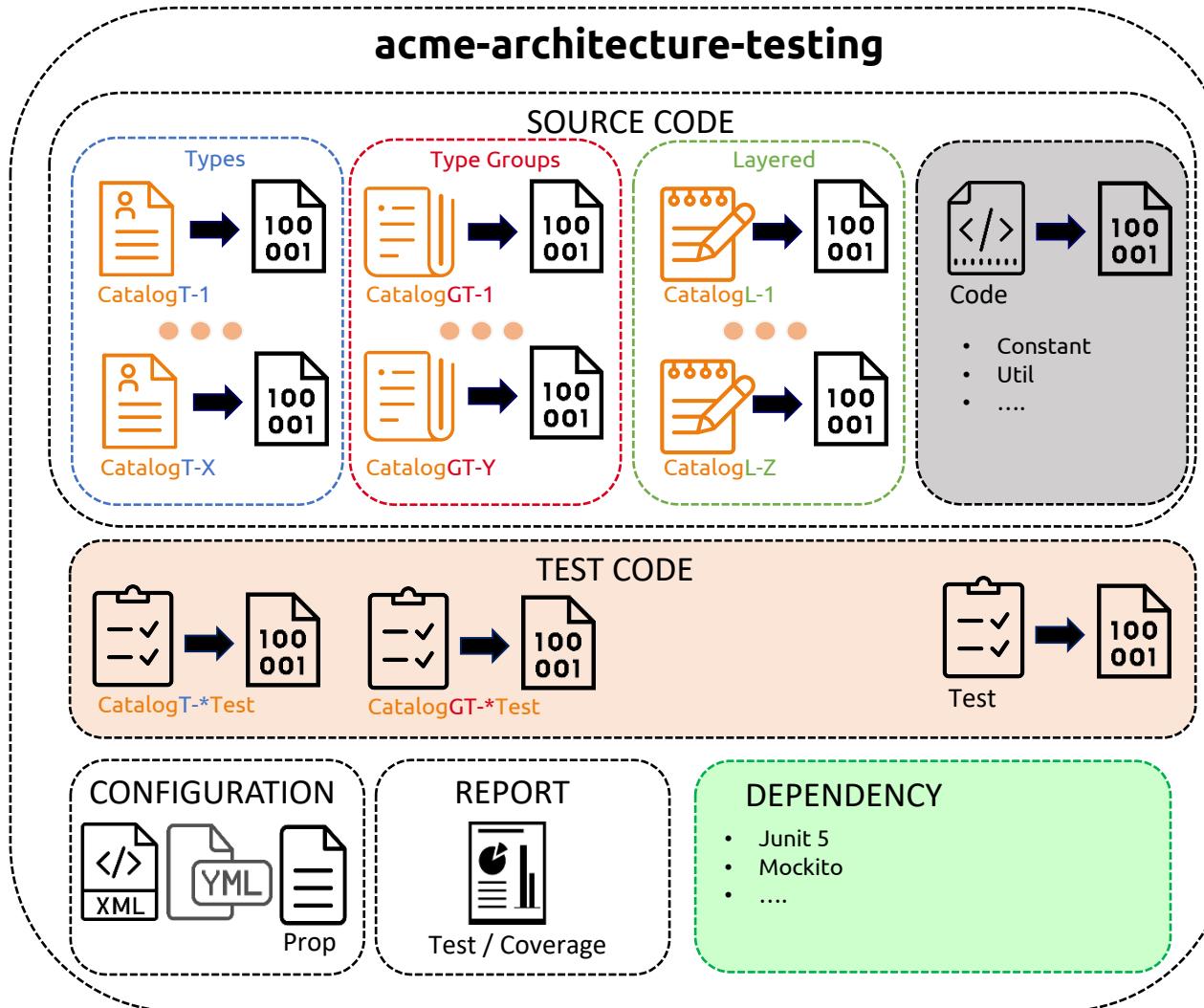
```
1 package com.acme.architecture.testing.archunit.rule.core.group;
2
3 import com.acme.architecture.testing.archunit.rule.core.catalog.CatalogConstantArchitectureRule;
4
5 public class CatalogCoreGlobalArchitectureRule {
6
7     @ArchTest
8     public static final ArchRules base_constant_architecture = ArchRules.in(CatalogConstantArchitectureRule.class);
9
10    @ArchTest
11    public static final ArchRules base_annotation_architecture = ArchRules.in(CatalogCustomAnnotationArchitectureRule.class);
12
13    @ArchTest
14    public static final ArchRules base_data_factory_architecture = ArchRules.in(CatalogDataFactoryArchitectureRule.class);
15
16    @ArchTest
17    public static final ArchRules base_dummy_architecture = ArchRules.in(CatalogDummyArchitectureRule.class);
18
19    @ArchTest
20    public static final ArchRules base_dummy_data_factory_architecture = ArchRules.in(CatalogDummyDataFactoryArchitectureRule.class);
21
22    @ArchTest
23    public static final ArchRules base_entity_architecture = ArchRules.in(CatalogEntityArchitectureRule.class);
24
25    @ArchTest
26    public static final ArchRules base_enumeration_architecture = ArchRules.in(CatalogEnumerationArchitectureRule.class);
27
28    @ArchTest
29    public static final ArchRules base_exception_architecture = ArchRules.in(CatalogExceptionArchitectureRule.class);
30
31    @ArchTest
32    public static final ArchRules base_mapper_architecture = ArchRules.in(CatalogMapperArchitectureRule.class);
33
34    @ArchTest
35    public static final ArchRules base_spring_global_architecture = ArchRules.in(CatalogSpringGlobalArchitectureRule.class);
36
37    @ArchTest
38    public static final ArchRules base_spring_configuration_architecture = ArchRules.in(CatalogSpringConfigurationArchitectureRule.class);
39
40    @ArchTest
41    public static final ArchRules base_spring_repository_architecture = ArchRules.in(CatalogSpringRepositoryArchitectureRule.class);
42
43    @ArchTest
44    public static final ArchRules base_spring_service_architecture = ArchRules.in(CatalogSpringServiceArchitectureRule.class);
45
46    @ArchTest
47    public static final ArchRules base_spring_service_impl_architecture = ArchRules.in(CatalogSpringServiceImplArchitectureRule.class);
48
49    @ArchTest
50    public static final ArchRules base_spring_rest_controller_architecture = ArchRules.in(CatalogSpringRestControllerArchitectureRule.class);
51
52    }
53}
```



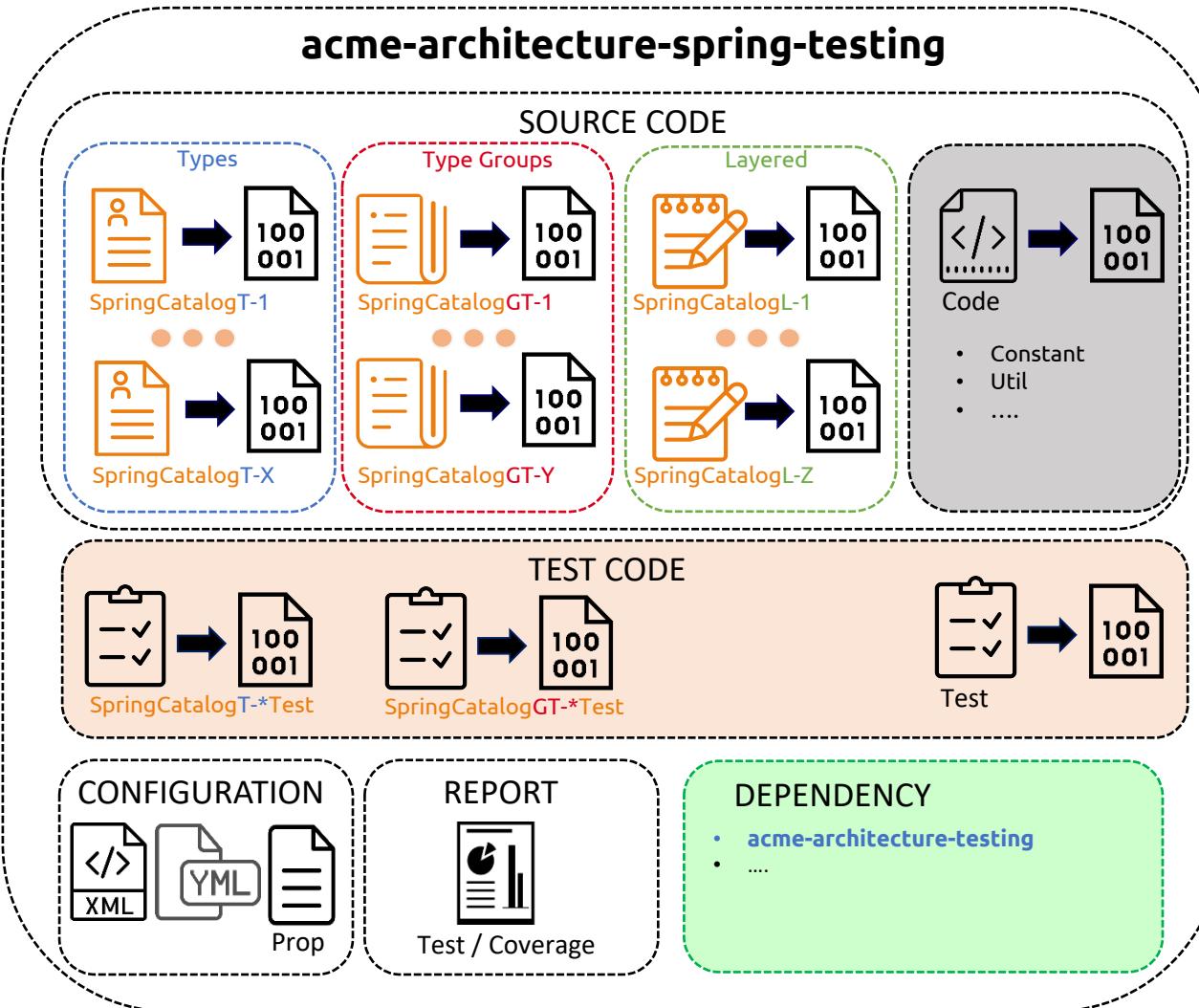
```

1 import static com.tngtech.archunit.library.Architectures.layeredArchitecture;
2
3 public class CatalogApiLayeredArchitectureRule {
4     @ArchTest
5     public static final ArchRule api_layered_architecture_should_have_a_default_definition =
6         layeredArchitecture()
7
8     // *****
9     // *** Layers ***
10    // *****
11
12    // Constants
13    .layer(ArchUnitLayeredArchitectureConstant.CONSTANT_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_CONSTANT_CLASS)
14
15    // Entity
16    .layer(ArchUnitLayeredArchitectureConstant.ENTITY_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_ENTITY_CLASS)
17
18    // DTO
19    .layer(ArchUnitLayeredArchitectureConstant.DTO_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_DTO_CLASS)
20
21    // Repository
22    .layer(ArchUnitLayeredArchitectureConstant.REPOSITORY_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_REPOSITORY_CLASS)
23
24    // Service
25    .layer(ArchUnitLayeredArchitectureConstant.SERVICE_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_SERVICE_CLASS)
26
27    // Controller
28    .layer(ArchUnitLayeredArchitectureConstant.CONTROLLER_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_CONTROLLER_CLASS)
29
30    // Others
31    .layer(ArchUnitLayeredArchitectureConstant.OTHERS_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_OTHERS_CLASS)
32
33    // Conditions
34    .layer(ArchUnitLayeredArchitectureConstant.CONDITIONS_LAYER).definedBy(ArchUnitPackageConstant.RESIDE_FINAL_PACKAGE_CONDITIONS_CLASS)
35
36    // Constant layer
37    .layer("Constant layer").definedBy("...constant")
38
39    // Entity layer
40    .layer("Entity layer").definedBy("...entity")
41
42    // Request DTO layer
43    .layer("Request DTO layer").definedBy("...request")
44
45    // Response DTO layer
46    .layer("Response DTO layer").definedBy("...response")
47
48    // Repository layer
49    .layer("Repository layer").definedBy("...repository")
50
51    // Service layer
52    .layer("Service layer").definedBy("..service..")
53
54    // Controller layer
55    .layer("Controller layer").definedBy("..controller")
56
57    // Configuration layer
58    .layer("Config layer").definedBy("..config")
59
60    // Factory layer
61    .layer("Factory layer").definedBy("..factory")
62
63    // Dummy layer
64    .layer("Dummy layer").definedBy("..dummy")
65
66    // Util layer
67    .layer("Util layer").definedBy("..util..")
68
69    // Mapper layer
70    .layer("Mapper layer").definedBy("..mapper..")
71
72    // *****
73    // *** Layers ***
74    // *****
75}

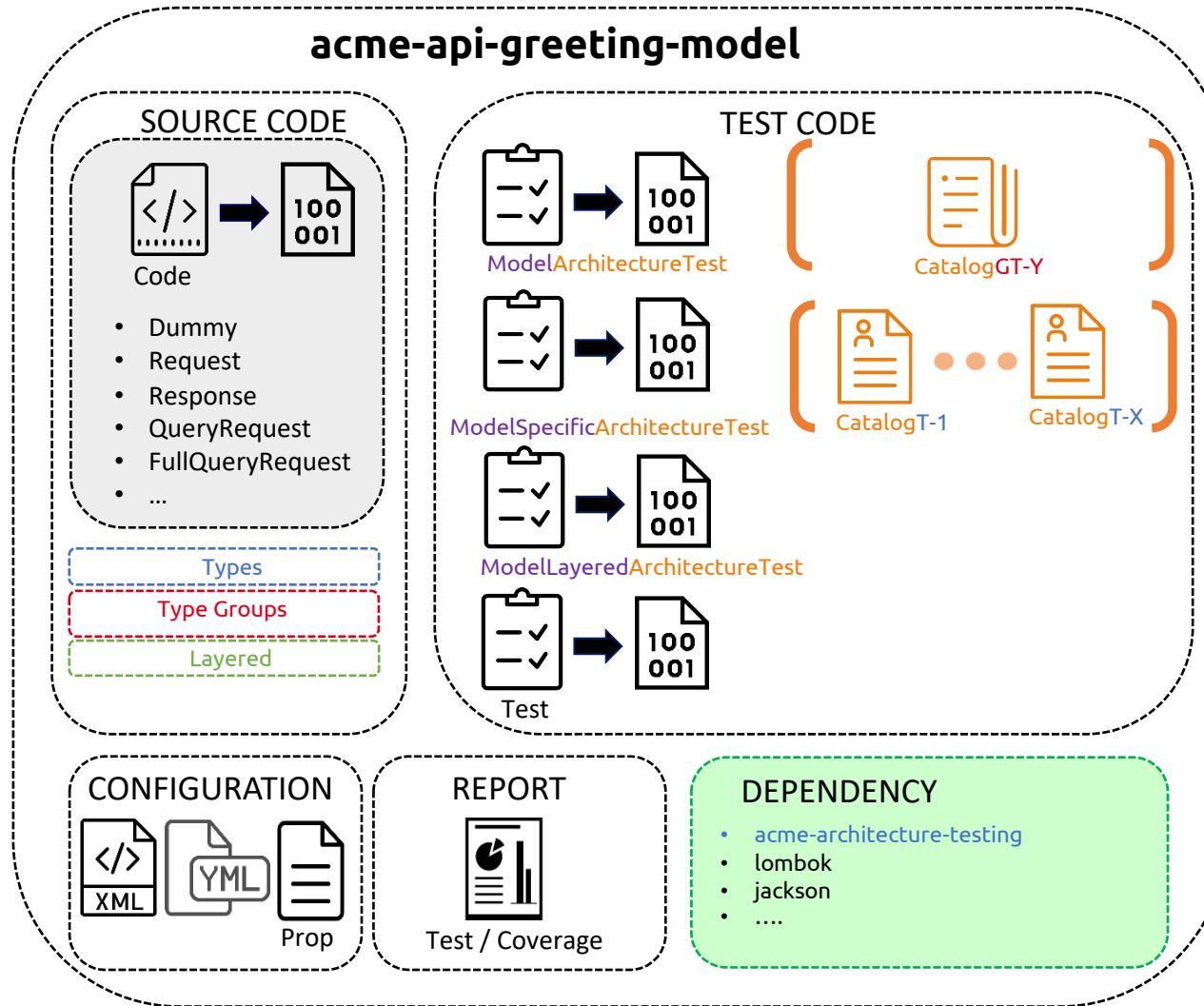
```



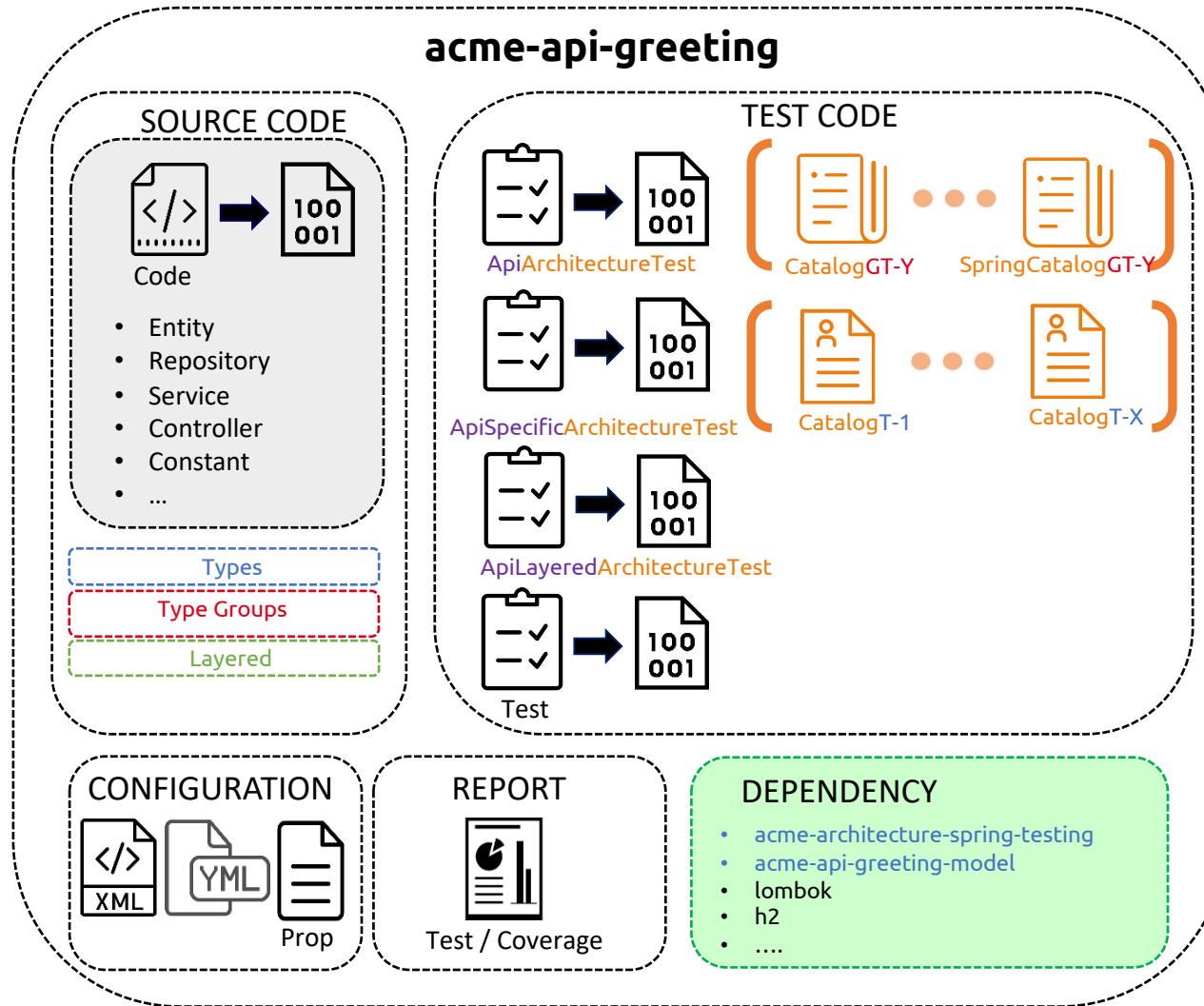
- Librería de Arquitectura personalizada Java
- Objetivos :
 - Soporte al testing unitario / integración general -> Junit5, Mockito, AssertJ,... -> Uso de **Starter Test de Spring**
 - Incorporar ArchUnit
 - Soporte de Reglas Arquitectónicas de propósito general (Entidades, DTOs, Servicios, Utilidades, ...) -> **Types**
 - Soporte a algunas tecnologías utilizadas en desarrollos : lombok, mapstruct, etc. -> **Types**
 - Soporte de Grupos de Reglas Arquitectónicas de propósito general -> **Type Groups**
 - Soporte a Arquitecturas de N-Capas de propósito general -> **Layered**
- Incluye todo tipo de componentes que pueden ser de utilidad en otros proyectos relacionado con testing y soporte de arquitectura (Constantes, Utilidades, etc.)
- Importación en los proyectos como dependencia
- Todas las reglas han sido probadas en el ámbito de test
- Se ha probado consigo mismo 😊



- Librería de Arquitectura personalizada Java
- Objetivos :
 - Soporte de Reglas Arquitectónicas de propósito general para **Spring** (Entidades, Repositorios, Servicios, Configuración, ...) -> **Types**
 - Soporte a algunas tecnologías utilizadas en desarrollos : JPA, etc. -> **Types**
 - Soporte de Grupos de Reglas Arquitectónicas de **Spring** propósito general -> **Type Groups**
 - Soporte a Arquitecturas de N-Capas de **Spring** propósito general -> **Layered**
- Incluye todo tipo de componentes que pueden ser de utilidad en otros proyectos relacionado con todo esto (Constantes, Utilidades, etc.)
- Amplía la funcionalidad de la librería : **acme-architecture-testing**
- Importación en los proyectos como dependencia
- Todas las reglas han sido probadas en el ámbito de test
- Se ha probado consigo mismo ☺



- Librería de modelo de negocio del servicio “**Greeting**”
- Objetivos :
 - *Servir de representación de proyectos del tipo modelo*
 - *Centralización de la forma de comunicarse con el API*
- Incluye diferentes Request / Response según las necesidades
- Incluye todo tipo de componentes que pueda ser de utilidad en otros proyectos relacionado con el modelo (Constantes, Utilidades, etc.)
- Importación en los proyectos como dependencia
- Siempre se puede ampliar el soporte de Reglas Arquitectónicas en el propio proyecto
- Validación de la arquitectura con componentes de la librería : [acme-architecture-testing](#)



- Aplicación que representa un “microservicio” sobre el modelo de negocio “Greeting” implementado con la tecnología Spring - Spring Boot
- Objetivos :
 - *Servir de representación de proyectos del tipo API*
 - *Proporcionar un API REST para el clásico “Hello World” pero de un modo más avanzado y similar al que podemos encontrar en el mundo laboral*
- Incorpora al resto de librerías como dependencias
- Importación en los proyectos como dependencia
- Siempre se puede ampliar el soporte de Reglas Arquitectónicas en el propio proyecto
- Validación de la arquitectura con componentes de la librería : acme-architecture-spring-testing
- Utiliza el modelo de la librería : acme-api-greeting-model

Ejemplo de Uso “acme-api-greeting”



```
* com.acme.greeting.api.model.greeting.constant.GreetingQueryJsonFieldConstant
* com.acme.greeting.api.model.greeting.constant.PatternConstant
* com.acme.greeting.api.constant.GreetingApiRestfulConfigConstant
* com.acme.greeting.api.model.greeting.dummy.DummyGreetingFullQueryRequest
* com.acme.greeting.api.factory.GreetingDataFactory
* com.acme.greeting.api.mapper.struct.GreetingMapperStruct
* com.acme.greeting.api.constant.GreetingRestApiConstant
* com.acme.greeting.api.controller.GreetingRestController
* com.acme.greeting.api.config.GreetingApiRestfulConfig
* com.acme.greeting.api.config.swagger.SwaggerConfig
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time
[INFO] Running ApiSpecificArchitectureTest
[INFO] Tests run: 74, Failures: 0, Errors: 0, Skipped: 0, Time
[INFO] Running ApiSpecificLayeredArchitectureTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time
[INFO] Running ApiArchitectureTest
[INFO] Tests run: 121, Failures: 0, Errors: 0, Skipped: 0, Time
[INFO] Running ApiLayeredArchitectureTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time
[INFO] Results:
[INFO] Tests run: 230, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ acme-api
[INFO] Building jar: /Users/vjmadriv/Workspace/personal/enmilocal
[INFO] --- spring-boot-maven-plugin:2.3.4.RELEASE:repackage (repackage)
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-failsafe-plugin:2.22.2:integration-test (default)
[INFO] T E S T S
[INFO] -----
[INFO] Running com.acme.greeting.api.controller.GreetingRestControllerTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] --- maven-failsafe-plugin:2.22.2:verify (default) @ acme-api
[INFO] --- maven-install-plugin:2.5.2:install (default-install)
[INFO] Installing /Users/vjmadriv/Workspace/personal/enmilocal/jmadriv/.m2/repository/com/acme/greeting/api/acme-api-greeting/0.0.1-SNAPSHOT/acme-api-greeting-0.0.1-SNAPSHOT.jar
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 44.801 s
[INFO] Finished at: 2021-01-16T18:53:52+01:00
[INFO]
[INFO] -----
[INFO] -/workspace/personal/enmilocalfunciona-arquunit/acme-api-green

```

mvn clean install

java -jar target/acme-api-greeting.jar

Swagger UI

localhost:8091/swagger-ui.html

Acme Message API Title ^{1.0}

[Base URL: localhost:8091 /]
<http://localhost:8091/v2/api-docs>

Acme Message API Description

[Terms of service](#)
[Victor Madrid - Website](#)
[Send email to Victor Madrid](#)
[Apache 2.0](#)

basic-error-controller Basic Error Controller

greeting Greeting Rest Controller

operation-handler Operation Handler

web-mvc-links-handler Web Mvc Links Handler

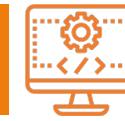
Models



<http://localhost:8091/swagger-ui.html>

10

DEMO



¡Gracias!

A Coruña · Barcelona · Cádiz · Huelva · Las Rozas · Lisboa ·
Madrid · Milán · Palma de Mallorca · Santiago de
Compostela · Sevilla · Zaragoza

✉ marketing@atsistemas.com
☎ 634 888 000
🌐 www.atsistemas.com

