# ARCHUNIT: GUARDING THE LOGIC OF YOUR APPLICATION

JOKER

ANDRES SACCO

saccoandres

andres-sacco

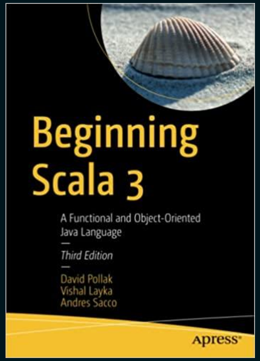
sacco.andres@gmail.com

# ABOUT ME

I’ve worked as developer since 2007 using different languages like Java, Kotlin, Scala, PHP, and NodeJS.

I've participated as a Technical Reviewer of Packt, Apress, and Manning books.

I published books, courses, and different theoretical-practical projects.



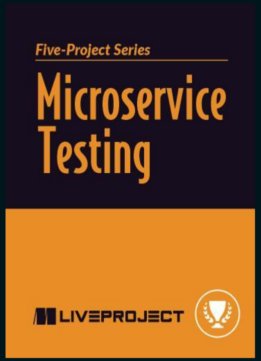
Beginning Scala 3

A Functional and Object-Oriented Java Language

Third Edition

David Pollak
Vishal Layka
Andres Sacco

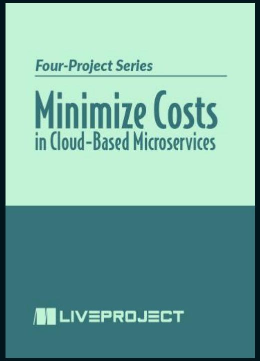
Apress



Five-Project Series

Microservice Testing

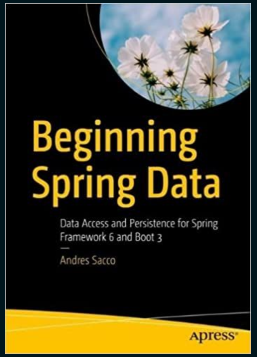
LIVEPROJECT



Four-Project Series

Minimize Costs in Cloud-Based Microservices

LIVEPROJECT



Beginning Spring Data

Data Access and Persistence for Spring Framework 6 and Boot 3

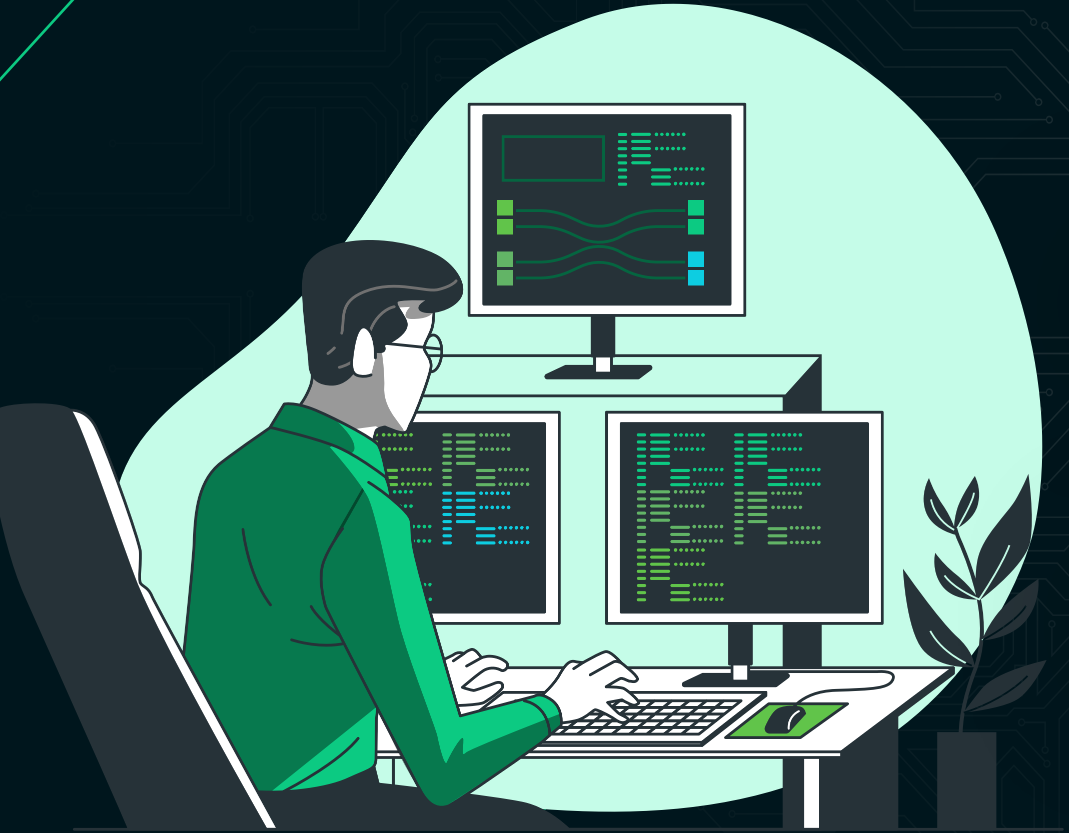
Andres Sacco

Apress

I participated as speaker on different conferences.

# > CONTEXT OF THE SITUATION

You have a large number of microservices which are developed by different teams



EACH DEVELOPER OR TEAM COULD DEFINE THEIR STANDARD

NOT ALL THE APPLICATIONS USE THE SAME PACKAGES NAMES

IT'S DIFFICULT TO FIND THE COMPONENTS TO DEVELOP SOMETHING

EACH LAYER COULD COMMUNICATE WITH OTHER WITHOUT RESTRICTION

# >LET’S SEE SOME EXAMPLES

To see in a simple way, let’s **compare** how different **microservices** do the same

| PROBLEMS | MICROSERVICE A | MICROSERVICE B |
|---|---|---|
| PACKAGE | com.api.flights.repository | com.api.hotels.persistence |
| LOGGERS | static final Logger LOGGER; | Logger logger; |
| NAMING | CatalogController | RerservationHandleHTTP |
| DTO | Without business logic | With tons of business logic |

# > WHICH ALTERNATIVES EXIST?

There are many ways to solve this problem

### INFORMAL

There is a definition for the names and packages, but it's informal. There is not a document.

### DOCUMENT

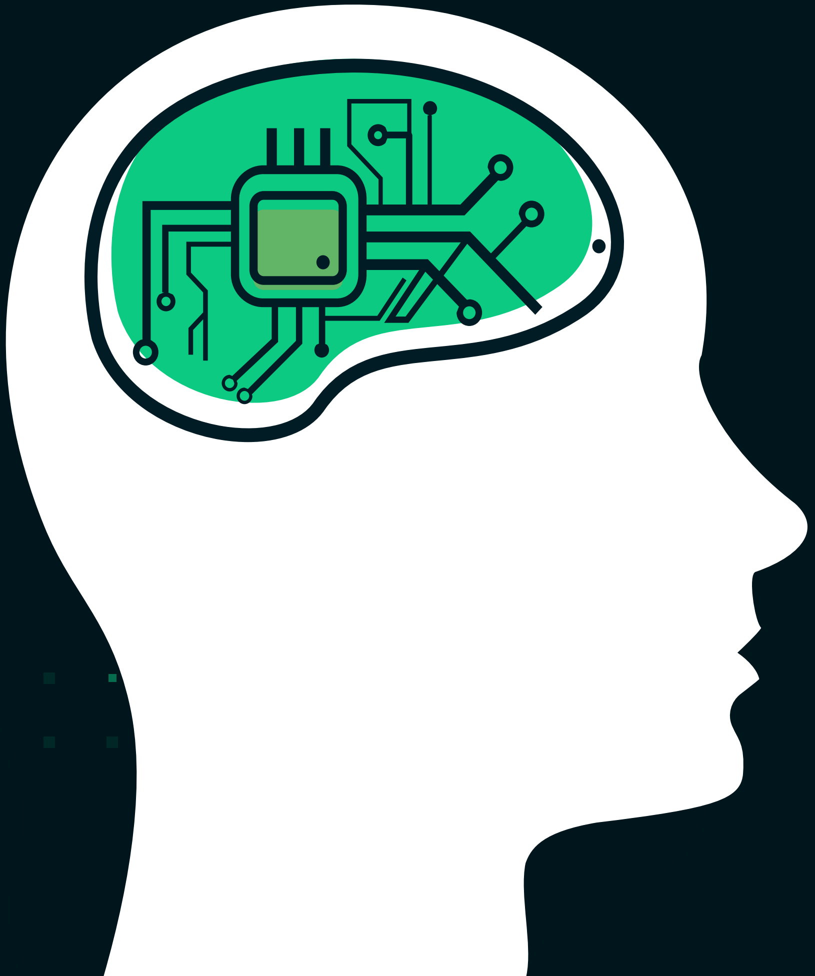
A document contains all the definitions that the teams must follow.

### AUTOMATICALLY

A tool or library contains all the definitions and validates them.

# >WHY INFORMAL APPROACHES FAILED?

There are tons of possible solutions but in most of the cases fails because



- NOT ALL THE DEVELOPERS COULD HAVE THE SAME LEVEL OF EXPERIENCE
- SOMEONE COULD FORGOT TO CHECK THE NAMES OF THE THINGS
- IMPLIES TOO MUCH TIME IN SOME CASES
- THE RULES COULD CHANGE WITHOUT NOTIFY TO ALL THE DEVELOPERS

# > WHICH ALTERNATIVES EXIST TO SOLVE IT?

There are lot of alternatives but the most relevant are

# >A BRIEF COMPARATIVE OF THE ALTERNATIVES

| | MAIN DIFFERENCES | PMD | CHECK STYLE | FINDBUGS | ARCHUNIT |
|---|---|---|---|---|---|
| 01 | WORKS AS UNIT TESTS NOT A STATIC ANALYZER | ✗ | ✗ | ✗ | ✓ |
| 02 | CHECK THE CODE TO DETECT PROBLEMS | ✓ | ✓ | ✓ | ✓ |
| 03 | IT'S POSSIBLE TO CUSTOMIZE THE VALIDATIONS RULES | ✓ | ✓ | ✓ | ✓ |
| 04 | CREATE RULES TO VALIDATE THE PACKAGES NAME'S | ✗ | ✗ | ✗ | ✓ |
| 05 | IT'S POSSIBLE TO DETECT A BAD USE OF THE ARCHITECTURE | ✗ | ✗ | ✗ | ✓ |
| 06 | IT'S POSSIBLE TO DEFINE RULES TO RESTRICT THE FIELD'S NAMES | ✗ | ✗ | ✗ | ✓ |
| 07 | HAVE A GREAT COMMUNITY OF USERS AND DOCUMENTATION | ✓ | ✓ | ✓ | ✓ |

# > WHICH LIBRARY SOLVE ALL THESE PROBLEMS?

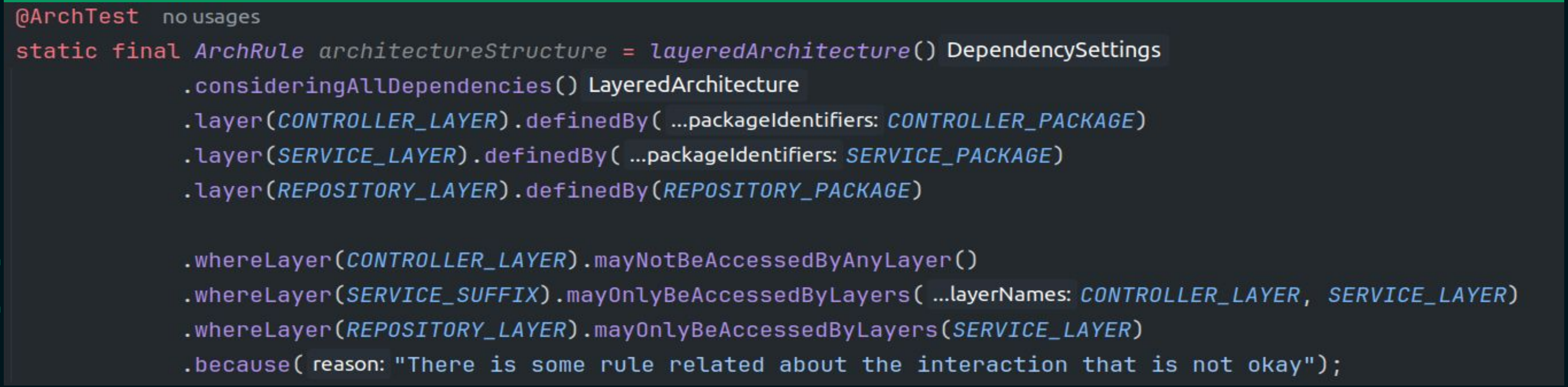
There is a library called Archuit which have the following features

- YOU CAN DEFINE THE RULES TO VALIDATE NAMES OR HIERARCHY
- PROVIDE YOU THE POSSIBILITY TO WRITE AS A UNIT TESTS
- YOU CAN RUN AS A SEPARATE STEP USING PROFILES
- YOU CAN CREATE YOUR OWN RULES TO VALIDATE SOMETHING
- CREATE YOUR LIBRARY WITH ALL THE RULES TO DISTRIBUTE IT
- IT'S POSSIBLE TO USE ON OTHER LANGUAGES LIKE KOTLIN OR SCALA

# >HOW IS THE SYNTAX OF ARCHUNIT?

Let’s show you a simple example of the syntax

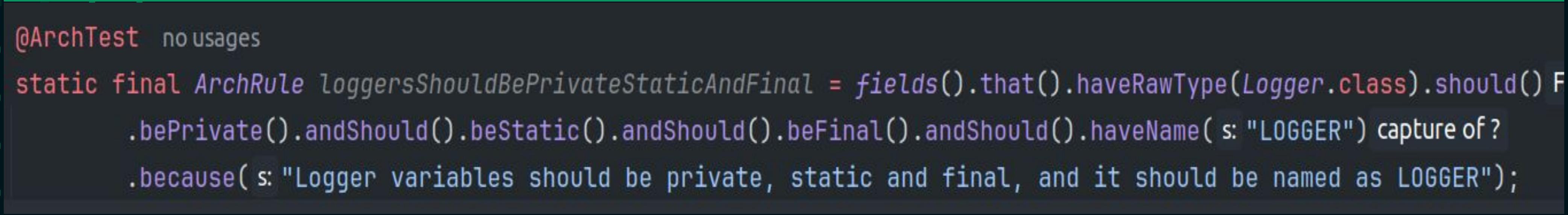
## LAYERS MUST FOLLOW A STRUCTURE



```
@ArchTest  no usages
static final ArchRule architectureStructure = layeredArchitecture() DependencySettings
                .consideringAllDependencies() LayeredArchitecture
                .layer(CONTROLLER_LAYER).definedBy( ...packageIdentifiers: CONTROLLER_PACKAGE)
                .layer(SERVICE_LAYER).definedBy( ...packageIdentifiers: SERVICE_PACKAGE)
                .layer(REPOSITORY_LAYER).definedBy(REPOSITORY_PACKAGE)

                .whereLayer(CONTROLLER_LAYER).mayNotBeAccessedByAnyLayer()
                .whereLayer(SERVICE_SUFFIX).mayOnlyBeAccessedByLayers( ...layerNames: CONTROLLER_LAYER, SERVICE_LAYER)
                .whereLayer(REPOSITORY_LAYER).mayOnlyBeAccessedByLayers(SERVICE_LAYER)
                .because( reason: "There is some rule related about the interaction that is not okay");
```

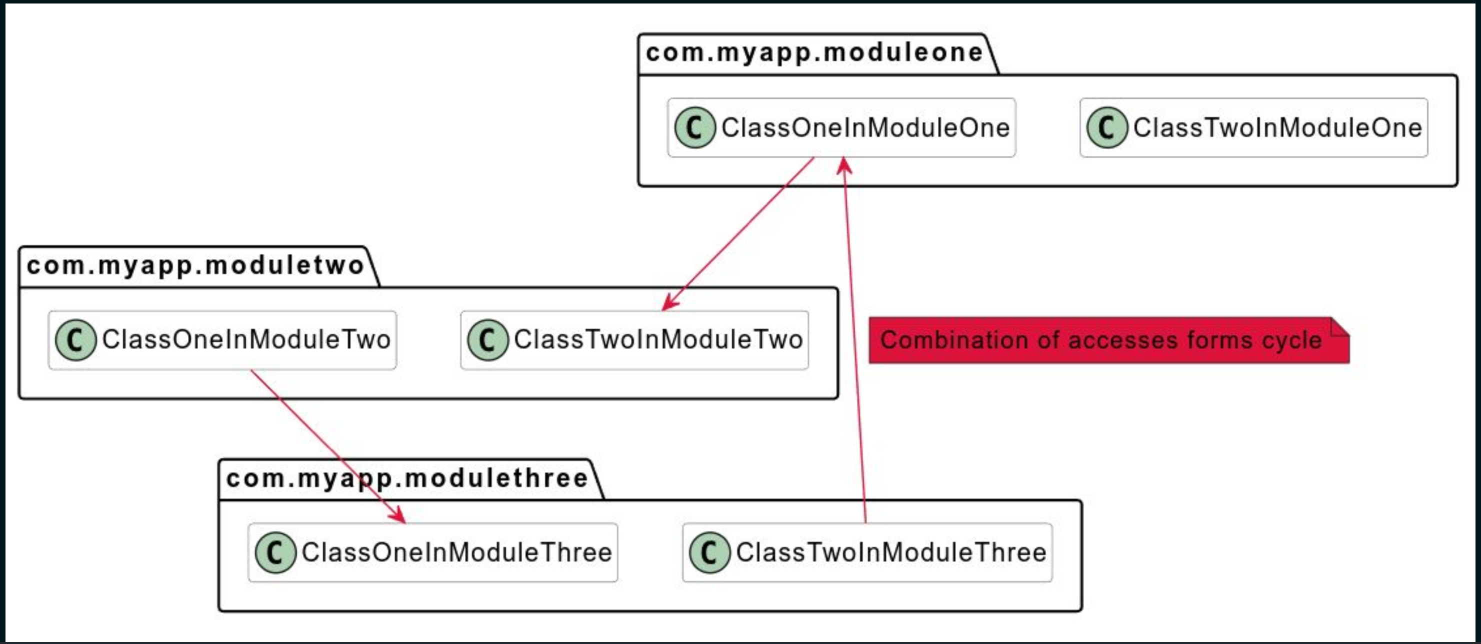
## LOGGERS MUST BE PRIVATE, STATIC, AND FINAL



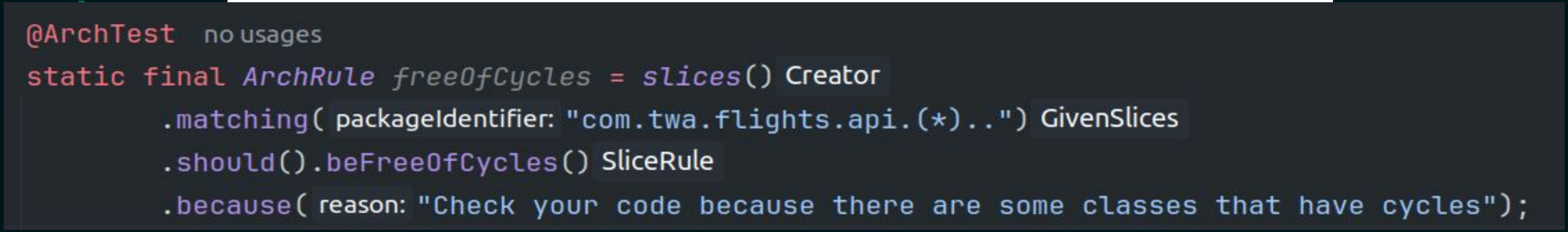
```
@ArchTest  no usages
static final ArchRule loggersShouldBePrivateStaticAndFinal = fields().that().haveRawType(Logger.class).should() F
        .bePrivate().andShould().beStatic().andShould().beFinal().andShould().haveName( s: "LOGGER") capture of ?
        .because( s: "Logger variables should be private, static and final, and it should be named as LOGGER");
```

# >LET'S SEE ANOTHER EXAMPLE

Let's show you another common problem on the applications



Combination of accesses forms cycle



```
@ArchTest  no usages
static final ArchRule freeOfCycles = slices() Creator
        .matching( packageIdentifier: "com.twa.flights.api.(*)..") GivenSlices
        .should().beFreeOfCycles() SliceRule
        .because( reason: "Check your code because there are some classes that have cycles");
```

# > WHAT IMPLIES THE USE OF ARCHUNIT?

Not everything is great with the use of this tool because you need to consider

- SOMEONE NEED TO DEFINE ALL THE RULES
- CREATE A LIBRARY TO EXTERNALIZE THE RULES
- EXPLAIN THE USE OR BENEFITS TO ALL THE COMPANY
- NEED TO HAVE A RESPONSIBLE TO MAINTAIN AND UPDATE THE RULES

# > EXAMPLE

Let’s see an example

https://github.com/andres-sacco/where-is-the-logic

# >BEST PRACTICES

### DEFINE

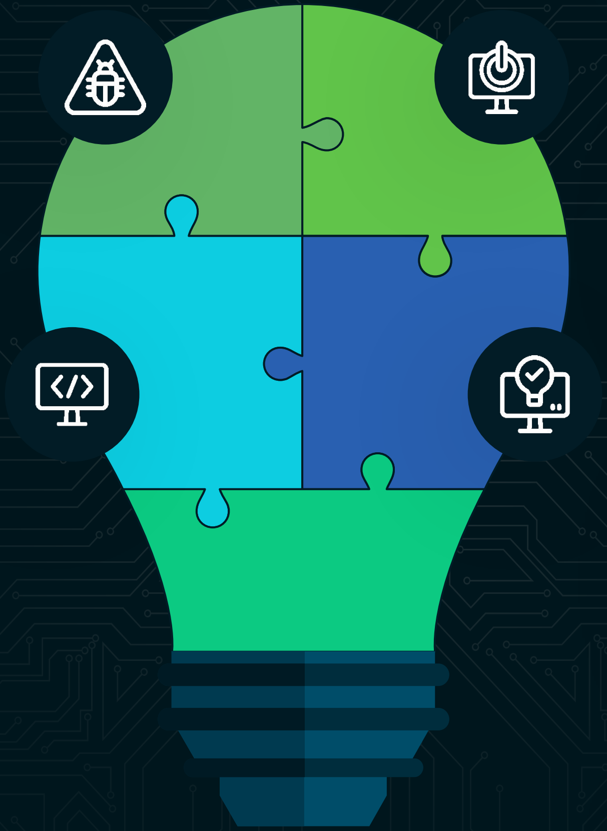
CREATE A DOCUMENT WITH ALL THE RULES

### PREFIX

DEFINE A PREFIX FOR EACH LAYER

### INTERACTION

DEFINE THE INTERACTION BETWEEN THE LAYERS



### CUSTOM

CREATE CUSTOM RULES WHEN IT'S NECESSARY

### EXTERNALIZE

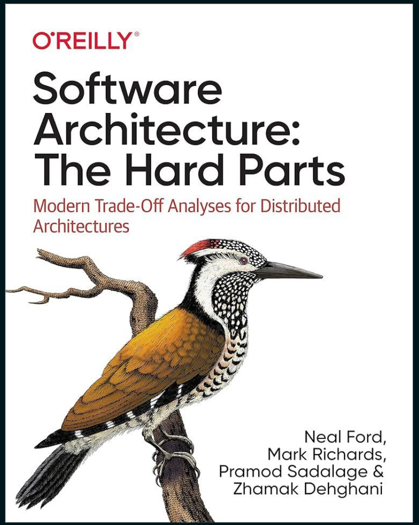
CREATE A LIBRARY WITH ALL THE RULES

### INCREMENTAL

ENABLE THE RULES BY STEPS, NOT ALL TOGETHER

# > ADDITIONAL RESOURCES

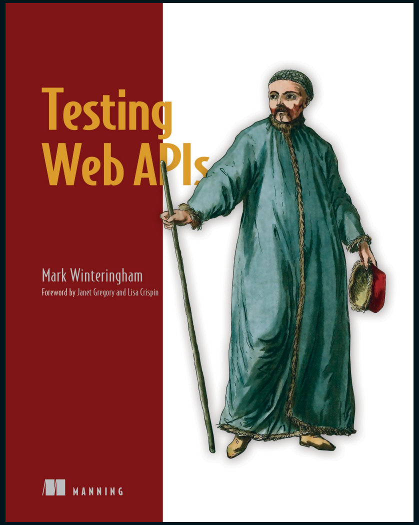
## BOOKS


O'REILLY

Software Architecture: The Hard Parts

Modern Trade-Off Analyses for Distributed Architectures

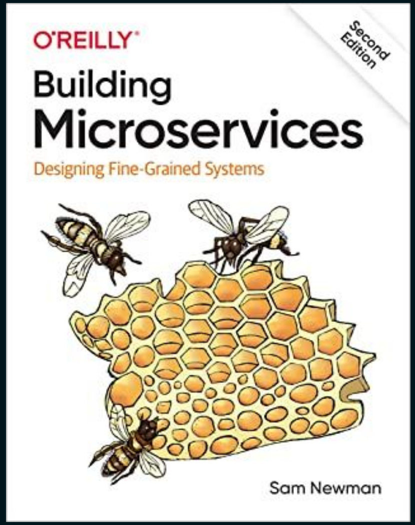
Neal Ford, Mark Richards, Pramod Sadalage & Zhamak Dehghani


Testing Web APIs

Mark Winteringham

Foreword by Janet Gregory and Lisa Crispin

MANNING

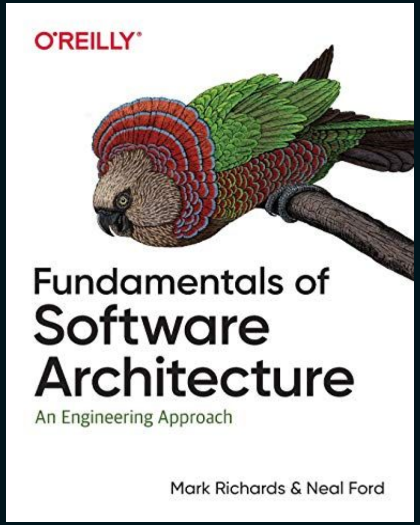

O'REILLY

Second Edition

Building Microservices

Designing Fine-Grained Systems

Sam Newman


O'REILLY

Fundamentals of Software Architecture

An Engineering Approach

Mark Richards & Neal Ford

## BLOGS

https://www.ministryoftesting.com/

https://martinfowler.com/

## REPOSITORY

https://github.com/andres-sacco/

# THANKS

JOKER