



Programación de Aplicaciones Telemáticas

TEMA 8: TESTING EN UNA APLICACIÓN

AGENDA

SESIÓN 1

AGENDA

SESIÓN 2

- Test de Integración
- Wiremock
- TestContainers

AGENDA

SESIÓN 3

- Java Microbenchmark Harness
- JMeter

SESIÓN 1

INTRODUCCIÓN

¿POR QUE NECESITAS TESTS?

Es necesario verificar que el software cumple con las expectativas / requerimientos.

INTRODUCCIÓN

¿POR QUE NECESITAS TESTS?

TEST FUNCIONALES

Una prueba funcional es una prueba de tipo caja negra basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.

TEST FUNCIONALES

- Configuración del Build System
- Unit Tests
- Integration Tests
- Code Coverage
- Mutation Tests Coverage
- Reglas de Arquitectura
- Documentación

TEST NO FUNCIONALES

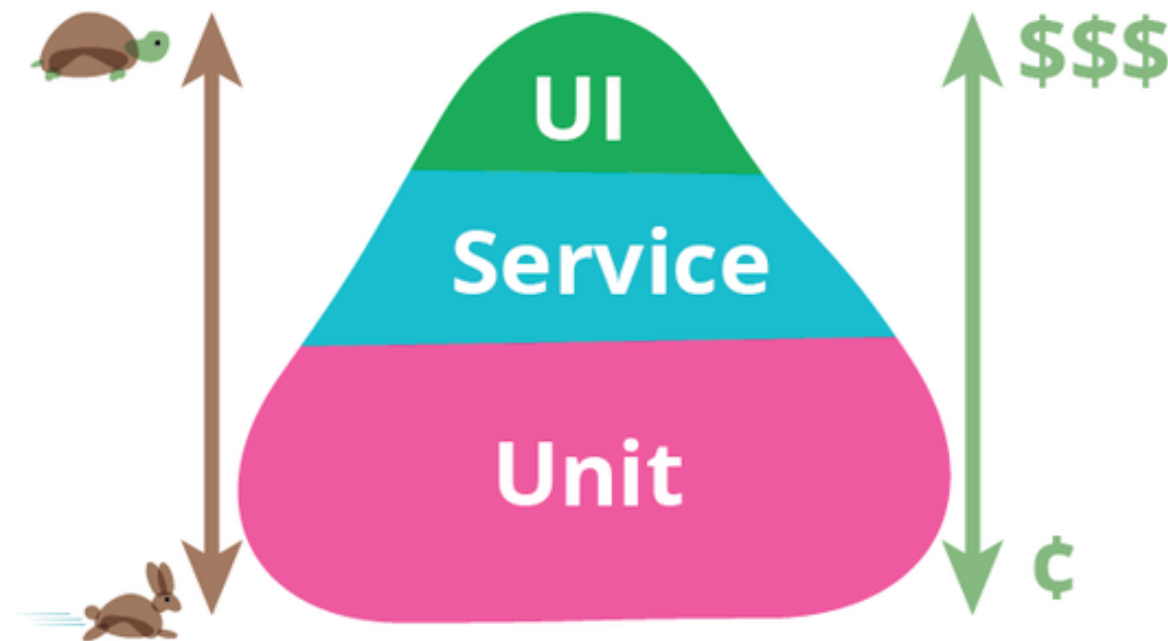
Una prueba funcional es una prueba de tipo caja negra basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.

TEST NO FUNCIONALES

- Performance Testing
- Monitoring Testing
- Profiling Testing
- Security Testing

PIRÁMIDE DE TESTING

The test pyramid is a way of thinking about how different kinds of automated tests should be used to create a balanced portfolio.

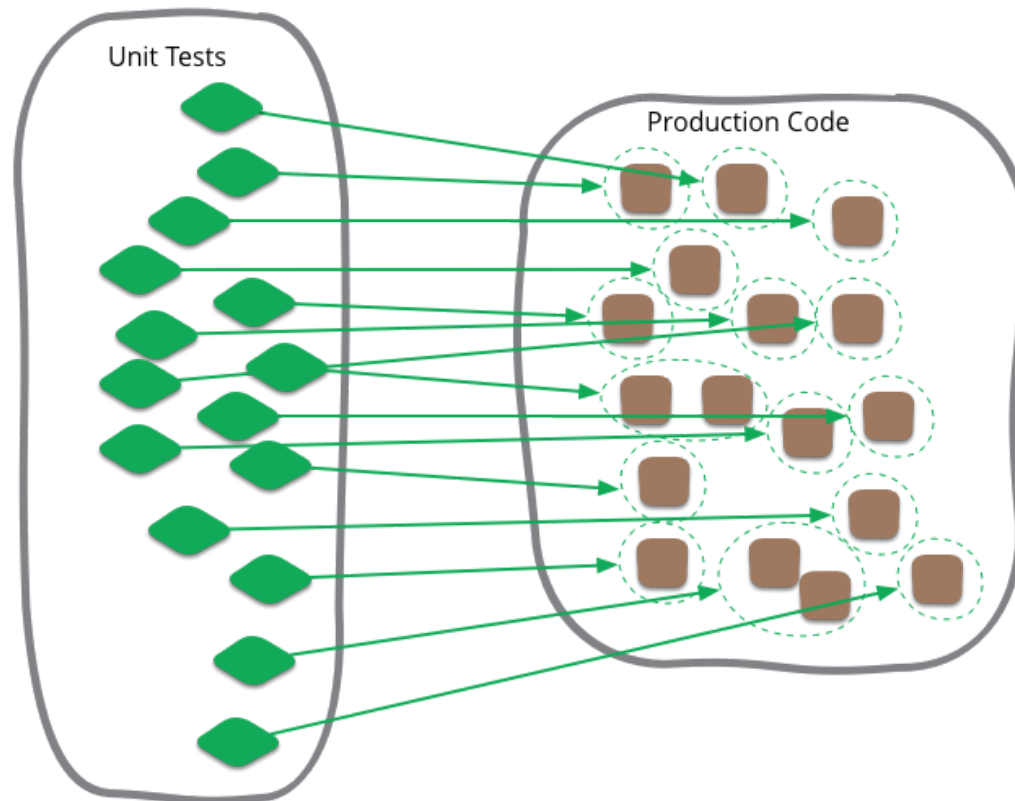


PIRÂMIDE DE TESTING

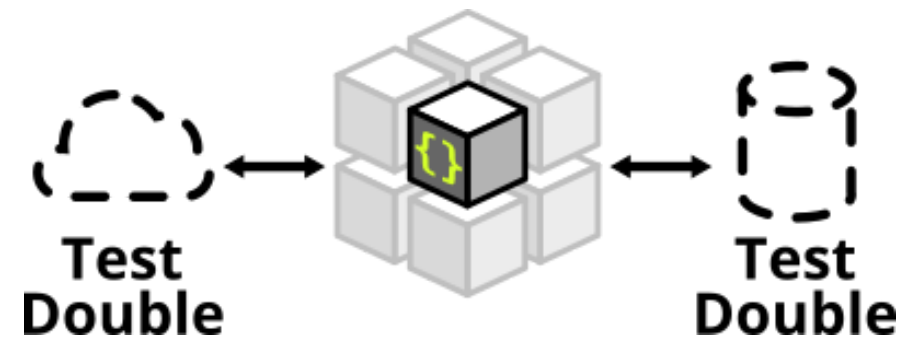
- Unit Tests
- Integration Tests
- E2E Tests

TESTS UNITARIOS

Your unit tests make sure that a certain unit (your subject under test) of your codebase works as intended.



TESTS UNITARIOS



TEST DOUBLES

The term Test Double as the generic term for any kind of pretend object used in place of a real object for testing purposes.

TEST DOUBLES

- Mocks are what we are talking about here: objects pre-programmed with expectations which form a specification of the calls they are expected to receive.
- Stubs provide canned answers to calls made during the test, usually not responding at all to anything outside what's programmed in for the test.
- Spies are stubs that also record some information based on how they were called.
- Fake objects actually have working implementations, but usually take some shortcut.
(an in memory database is a good example)

CONSIDERACION DE DISEÑO DE TESTS

PRINCIPIOS FIRST

- Fast
- Independent
- Repeatable
- Self-validating
- Timely

CONSIDERACION DE DISEÑO DE TESTS

THE RIGHT BICEP

- Right – are the results right?
- B – are all the boundary conditions CORRECT?
- I – can you check inverse relationships?
- C – can you cross-check results using other means?
- E – can you force error conditions to happen?
- P – are performance characteristics within bounds?

CONSIDERACION DE DISEÑO DE TESTS

THE RIGHT BICEP

TESTS FOR BOUNDARY CONDITIONS

JUNIT



JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit

JUNIT

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little, test a little, code a little."

JUNIT

FEATURES

- Provides annotations to identify test methods
- Provides assertions for testing expected results
- Provides test runners for running tests
- Allow coding faster, which increases quality
- Elegantly simple
- Can be run automatically and they check their own results and provide immediate feedback
- Tests can be organized into test suites

JUNIT EXAMPLE

```
public class MyUnit {  
    public String concatenate(String one, String two) {  
        return one + two;  
    }  
}
```

JUNIT

EXAMPLE

```
public class MyUnitTest {  
  
    @Test  
    public void given_MyUnit_when_concatenate_then_Ok() {  
        MyUnit myUnit = new MyUnit();  
  
        String result = myUnit.concatenate("one", "two");  
  
        assertEquals("onetwo", result);  
    }  
}
```

MOCKITO



Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

MOCKITO

FEATURES

- Mockito allows to write test methods compatible with "arrange/act/assert" approach.
- Mockito can be used to write Behavior Driven Development (BDD)-style.
- Mockito provides a nice, easily readable syntax.
- It is easy to read Mockito's error messages.

MOCKITO

EXAMPLE

```
public class WeatherForecast {  
    private WeatherService globalWeather; [1]  
    private WeatherService localService; [1]  
  
    public WeatherForecast(WeatherService globalWeather, WeatherService localService) {  
        this.localService = localService;  
        this.globalWeather = globalWeather;  
    }  
  
    public Weather getForecast(String city) { [2]  
        if (localService.hasForecastFor(city)) {  
            return localService.getWeather(city);  
        }  
        return globalWeather.getWeather(city);  
    }  
}
```

MOCKITO

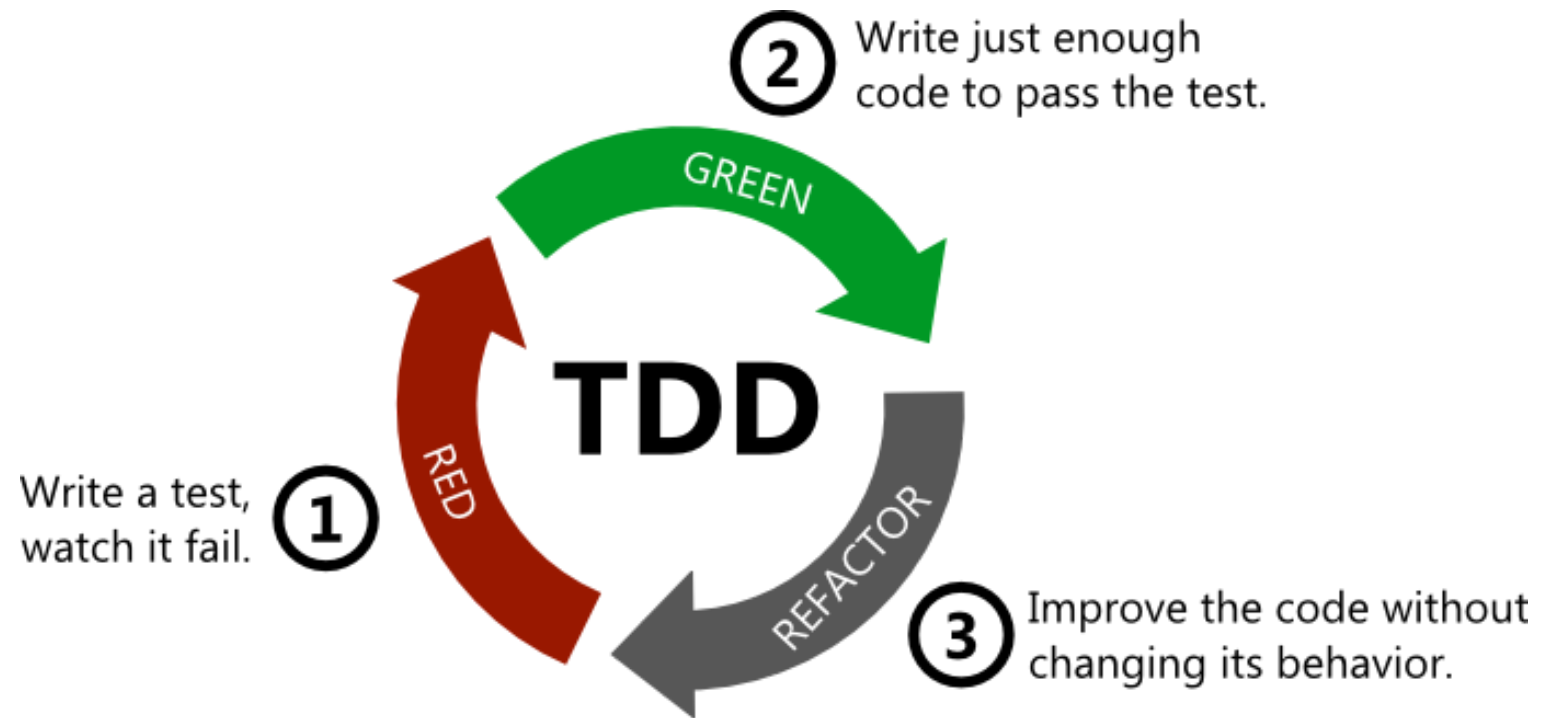
EXAMPLE

```
@Test
public void shouldFetchWeatherForecastFromGlobalServiceIfNotAv
    //Given
    WeatherService localWeatherService = Mockito.mock(Weat
    WeatherService globalWeatherService = Mockito.mock(Wea

    //When
    WeatherForecast forecast = new WeatherForecast(globalW
    Mockito.when(localWeatherService.hasForecastFor(anyStr
        .thenReturn(false);
    Weather forecastedWeather = new Weather();
    Mockito.when(globalWeatherService.getWeather(anyString
        .thenReturn(forecastedWeather);

    //Then
```

TDD



REFERENCIAS

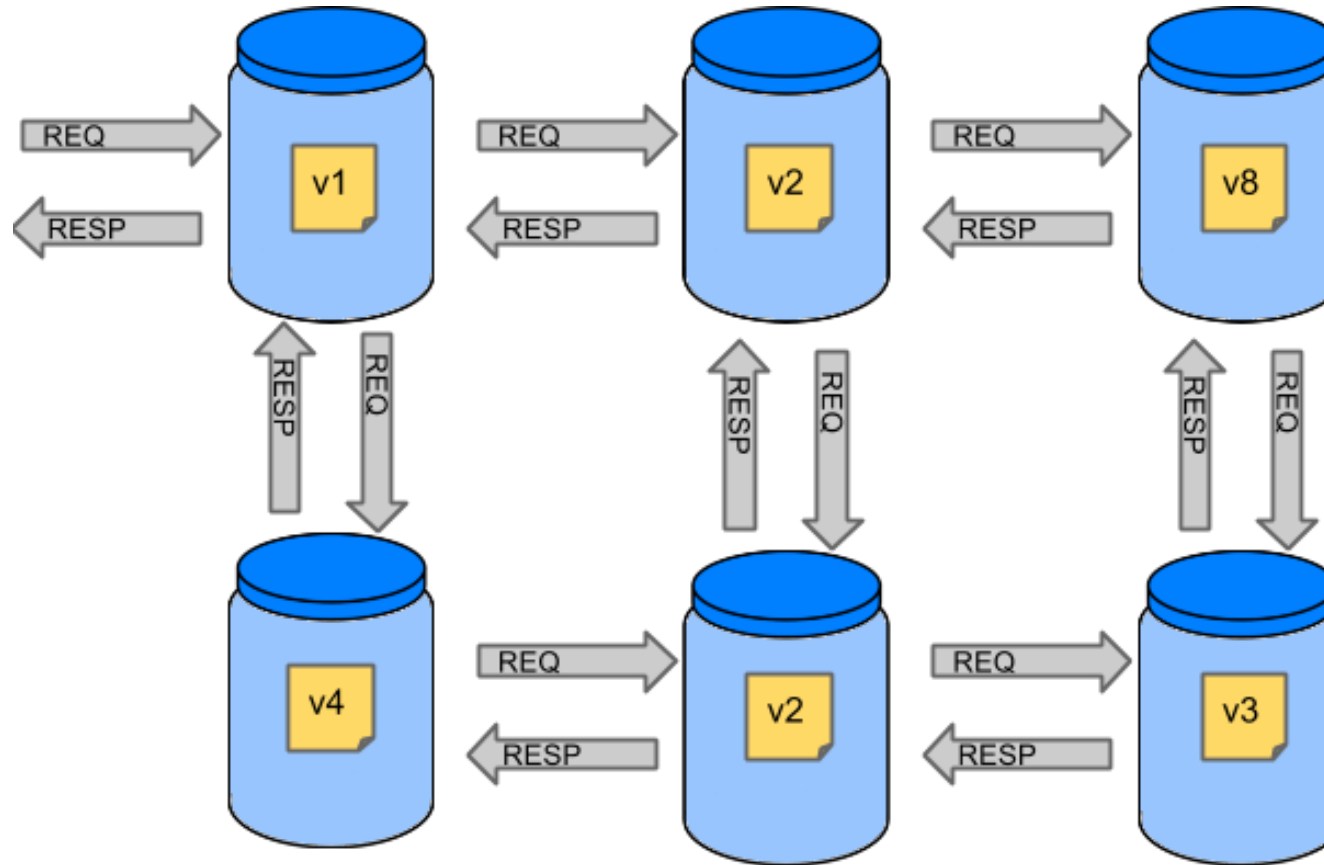
- https://es.wikipedia.org/wiki/Pruebas_de_software
- https://es.wikipedia.org/wiki/Pruebas_funcionales
- <https://martinfowler.com/bliki/TestPyramid.html>
- <https://martinfowler.com/bliki/UnitTest.html>
- <https://martinfowler.com/testing/>
- <https://martinfowler.com/articles/practical-test-pyramid.html>
- <https://xp123.com/articles/3a-arrange-act-assert/>
- <https://martinfowler.com/bliki/GivenWhenThen.html>
- <https://martinfowler.com/bliki/TestDouble.html>
- <https://site.mockito.org/>

SESIÓN 2

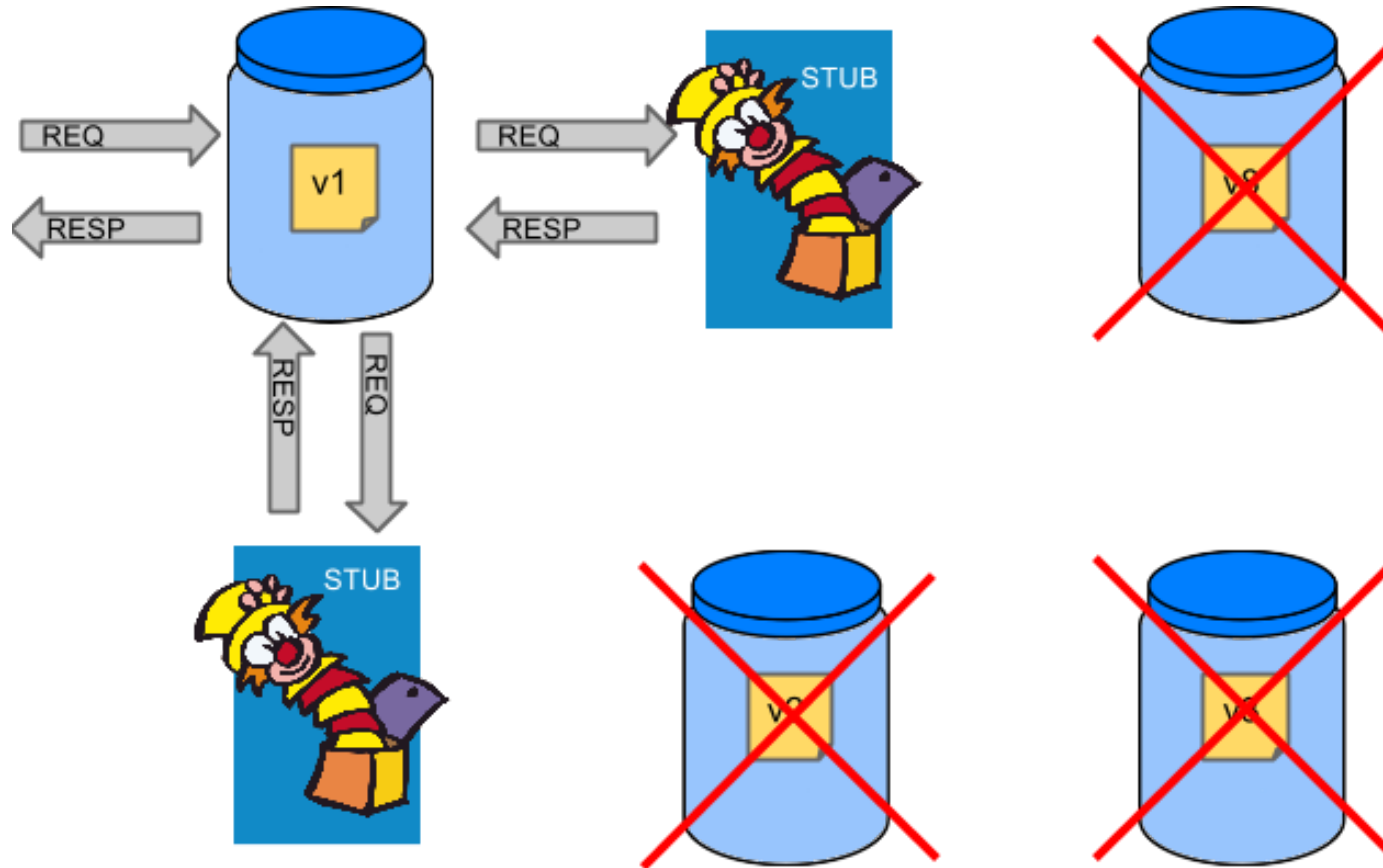
TEST DE INTEGRACIÓN

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group.

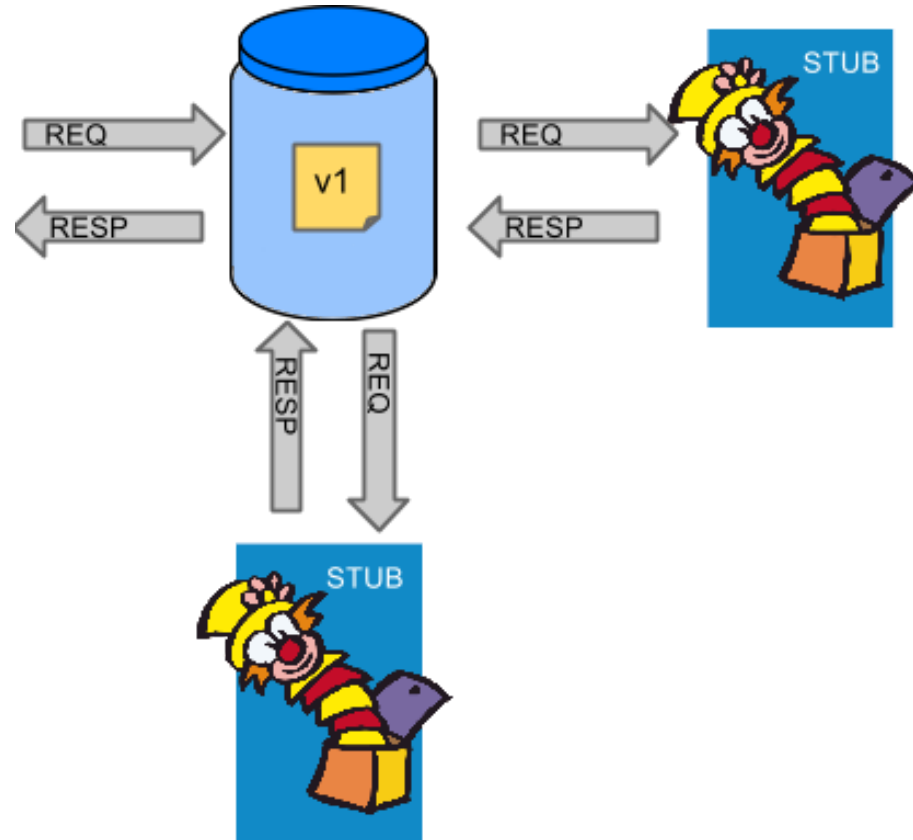
TEST DE INTEGRACIÓN



TEST DE INTEGRACIÓN



TEST DE INTEGRACIÓN



WIREFMOCK

WireMock is a simulator for HTTP-based APIs. Some might consider it a service virtualization tool or a mock server.

WIREMOCK

EXAMPLE

```
@Test
public void exampleTest() {
    stubFor(get(urlEqualTo("/my/resource"))
        .withHeader("Accept", equalTo("text/xml"))
        .willReturn(aResponse()
            .withStatus(200)
            .withHeader("Content-Type", "text/xml")
            .withBody("<response>Some content</response>"))

    Result result = myHttpServiceCallingObject.doSomething();

    assertTrue(result.isSuccessful());

    verify(postRequestedFor(urlMatching("/my/resource/[a-z0-9]
        .withRequestBody(matching(".*<message>1234</message>"))
```

TESTCONTAINERS

Testcontainers is a Java library that supports JUnit tests, providing lightweight, throwaway instances of that can run in a Docker container.

TESTCONTAINERS

EXAMPLE

```
@Testcontainers
class MixedLifecycleTests {
    @Container
    private static final MySQLContainer MY_SQL_CONTAINER =
        @Container
    private PostgreSQLContainer postgresqlContainer = new
        .withDatabaseName("foo")
        .withUsername("foo")
        .withPassword("secret");

    @Test
    void test() {
        assertTrue(MY_SQL_CONTAINER.isRunning());
        assertTrue(postgresqlContainer.isRunning());
    }
}
```

REFERENCIAS

- <https://martinfowler.com/bliki/IntegrationTest.html>
- <https://martinfowler.com/articles/mocksArentStubs.html>
- <https://martinfowler.com/bliki/TestDouble.html>
- <http://wiremock.org/>
- <https://www.testcontainers.org/>

SESIÓN 3

JAVA MICROBENCHMARK HARNESS

JMH is a Java harness for building, running, and analysing nano/micro/milli/macro benchmarks written in Java and other languages targeting the JVM.

JAVA MICROBENCHMARK HARNESS EXAMPLE

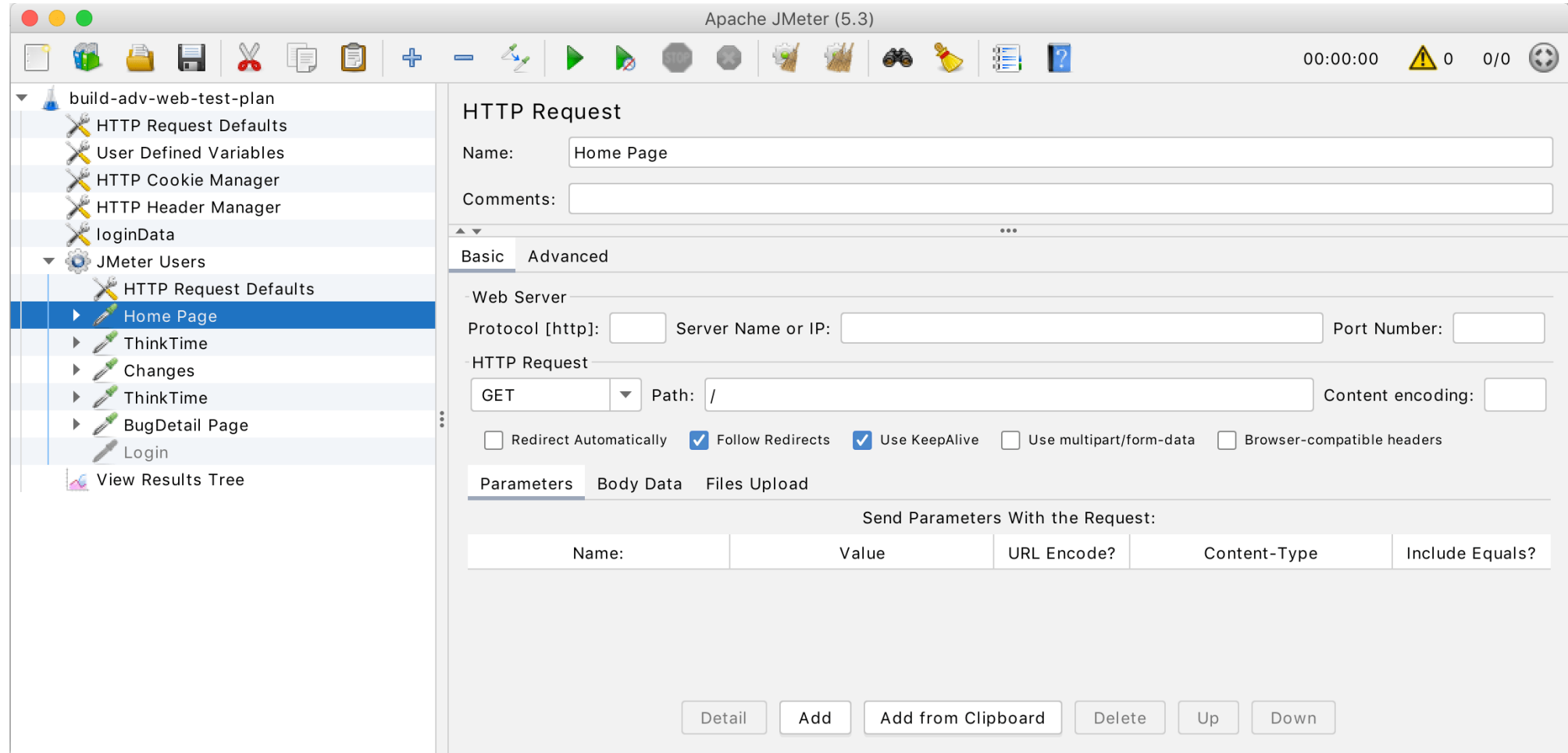
```
public class JMHSample_01_HelloWorld {  
    @Benchmark  
    public void wellHelloThere() {  
        // this method was intentionally left blank.  
    }  
    public static void main(String[] args) throws RunnerExcept  
        Options opt = new OptionsBuilder()  
            .include(JMHSample_01_HelloWorld.class.getSimp  
            .forks(1)  
            .build();  
  
        new Runner(opt).run();  
    }  
}
```

JMETER



The Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.

JMETER



REFERENCIAS

- <https://openjdk.java.net/projects/code-tools/jmh/>
- <https://github.com/openjdk/jmh>
- <https://github.com/openjdk/jmh/tree/master/jmh-samples/src/main/java/org/openjdk/jmh/samples>
- <https://jmh.morethan.io/>
- <https://jmeter.apache.org/>