UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

Doble grado en Ingeniería Informática y Matemáticas

MW Store: facilitador genérico para proveedores de servicios con multi-inquilinato en una infraestructura Cloud

---

Autor

Rubén Morales Pérez

Director

Manuel Isidoro Capel Tuñón

Departamento de Lenguajes y Sistemas Informáticos

Contents

1 Software infraestructure

1.1 Functional overview

1.2 Architecture

1.2.1 Technological stack

Front end: angular (testing with karma + jasmine) Back end: java 17 (testing with junit) Database: PostgreSQL Continous integration and delivery: Jenkins Orchestration: Docker + Kubernetes End to end tests: Selenium

1.2.2 Dependencies

1.3 Continuous integration and delivery

1.3.1 Deployments

1.3.2 Data migrations

1.4 Quality

Feature development will be slower as the program grows. Development in youth programs is faster than development in mature programs due to code complexity and possible regressions. Software must be carefully designed. Higher quality standards will prevent slowdowns in the development of new features, and regressions will be caught earlier (citation needed).

1.4.1 Code quality

SonarQube

1.4.2 Automatic tests

Automatic tests will catch bugs before production release. There are several testing strategies that can be used together:

- End to end tests: test entire workflows, involving front end, back end and database(s).
- Integration tests: specific use cases involving several components.
- Unit tests: test cases involving only one component, mocking other components when needed. In the back end, they don't involve the database, making them much faster.

The desired test amount proportion is described in the test pyramid:

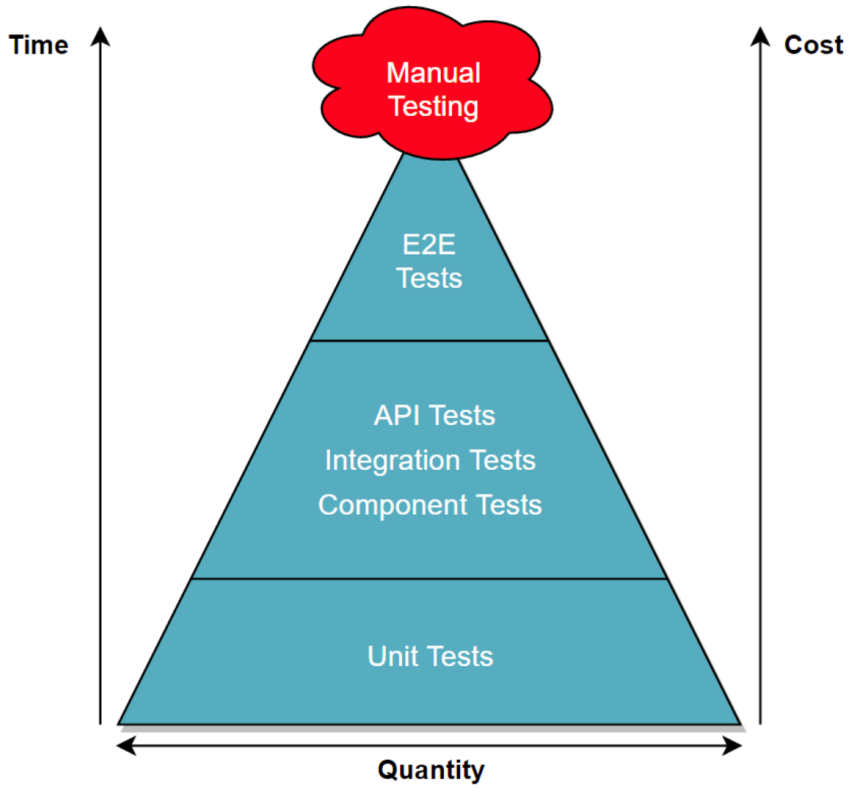

Figure 1: Test pyramid [1]

The inverted testing pyramid (fewer unit and integration tests, with an emphasis on automated and manual functional testing) is considered an anti-pattern, reducing the responsiveness, maintainability and reliability of the test setup [1]. Unit tests help identify faults in earlier phases [1] and must be the backbone of any software product.

2 Mathematical modelling

2.1 Petri nets

2.1.1 Reachability trees

2.2 Performance

2.2.1 Cache

2.3 Scalability

References

[1] Ville Hartikainen. "Defining suitable testing levels, methods and practices for an agile web application project". In: (2020), pp. 18–19.