Pruebas

Puesta en Producción Segura

Ciberseguridad

IES Zaidín-Vergeles

Junta de Andalucía

Práctica de desarrollo, cobertura y pruebas

Tabla de Contenido

Esta actividad nos muestra un ciclo completo de software: entrevista con el "cliente" (profesor), especificación en forma de manual de usuario con mockups, implementación, pruebas unitarias y generación de informe de cobertura.

Objetivos

- Practicar la toma de requerimientos y comunicación con el cliente.
- Traducir requisitos a un manual de usuario con mockups (PlantUML o Mermaid).
- Implementar una aplicación sencilla en el lenguaje elegido.
- Diseñar y ejecutar pruebas unitarias.
- Generar y analizar un informe de cobertura de código.

Cuando se haga referencia a un fichero del tipo `.*` significa que puede tener cualquier extensión (por ejemplo, `manual-usuario.adoc` o `manual-usuario.md`). En general se recomienda usar un formato de texto y no Word o LibreOffice ya que complica la gestión con Git.

Organización

- Grupo: 2 alumnos (o individual si se prefiere).
- Duración recomendada: 2–3 semanas.
- Entregables:
  - repositorio con código,
  - carpeta `docs/` con el manual de usuario y mockups,
  - carpeta `tests/`, informe de cobertura HTML y
  - un `README.md` con instrucciones para ejecutar el proyecto.

Instrucciones

Entrevista con el cliente (profesor)

- El profesor actúa como *stakeholder* y explica a grandes rasgos la aplicación (objetivo, público, funciones clave).
- Los alumnos toman apuntes y realizan preguntas. El profesor puede responder de forma deliberadamente ambigua o incompleta para simular requisitos inciertos. Se advierte que los requisitos pueden cambiar posteriormente.

Manual de usuario y mockups (requisitos)

- A partir de la entrevista, redactar un manual de usuario que describa las características y los flujos (qué debe ocurrir, no cómo se implementa). Incluir mockups de las pantallas o diagramas de requisitos usando PlantUML o Mermaid (Requirement diagrams).
- Entregable:
  - `docs/manual-usuario.*` y
  - `docs/mockups/` con los ficheros PlantUML/Mermaid.

Codificar la aplicación

- Implementar la funcionalidad mínima que soporte los requisitos numerados. Lenguaje a elección del grupo (Python, Node.js, Java, etc.). Mantener el proyecto sencillo: foco en la calidad del código y cumplimiento de requisitos.
- Entregable:
  - directorio `src/` con el código y
  - `README.md` con instrucciones de ejecución.

Pruebas unitarias

- Escribir pruebas unitarias que cubran casos normales, casos límite y errores esperados (input inválido, excepciones).
- Organizar los tests en `tests/` y asegurar que son automáticos y reproducibles.
- Entregable: `tests/` con comandos para ejecutar (explicados en `README.md`).

Pruebas de sistema

- Realizar pruebas de sistema que validen el comportamiento del conjunto en un entorno que simule producción (flujo completo de la aplicación, dependencias externas simuladas o reales según disponibilidad).
- Para aplicaciones web, se recomienda emplear herramientas de pruebas de interfaz como Selenium, Playwright o Cypress para automatizar la interacción con la interfaz gráfica y validar el comportamiento de usuario.
- Entregable: `tests/system/` y un informe breve (`docs/system-test-report.*`) con resultados y observaciones.

Informe de cobertura

- Usar una herramienta de cobertura adecuada al lenguaje para generar un informe (HTML preferido). Incluir el informe dentro de `coverage/` o publicar como artifact si se dispone de CI.
- Analizar el informe: listar las líneas/funciones críticas no cubiertas y justificar si falta test, refactor o es código muerto.
- Entregable: `coverage/` con el informe HTML y `docs/coverage-analisis.*`.

Recomendaciones y herramientas (opcional)

- Para mockups y requisitos: PlantUML (requirement diagram) o Mermaid (requirementDiagram). Mostrar los ficheros fuente (`.puml`, `.mmd`).
- Para pruebas unitarias: pytest (Python), Jest/Mocha (Node), JUnit (Java), etc.
- Para pruebas de interfaz web: Selenium, Playwright, Cypress (automatización de interacción y validación de UI).
- Para cobertura: coverage.py (Python), nyc/istanbul (Node), JaCoCo (Java), gcov/lcov (C/C++).

Evaluación

La evaluación combina producto (entregables) y proceso (colaboración, comunicación). Peso total 100%:

- Manual de usuario + mockups + numeración de requisitos (R001): 30% — completitud, claridad y verificabilidad de los requisitos.

- Implementación (R002): 30% — cumplimiento de requisitos, calidad de código, estructura y documentación (`README`).
- Pruebas unitarias (R003): 15% — cobertura de casos normales, límite y errores; organización de tests.
- Informe de cobertura y análisis (R004): 15% — generación del informe, nivel de cobertura alcanzado (orientativo >=70% líneas) y análisis de las zonas no cubiertas.
- Presentación / defensa (R005): 10% — explicación del proyecto, decisiones tomadas, y respuesta a preguntas del profesor/compañeros.

Cada criterio se calificará con una escala 0–10 y se ponderará según los porcentajes anteriores para obtener la nota final.

Rubrica (breve)

- Excelente (90–100): cumple totalmente con los requisitos, tests completos, cobertura alta y análisis crítico de las carencias.
- Notable (70–80): pequeños faltantes en test o documentación; buena implementación.
- Aprobado (50–60): requisitos básicos implementados, tests mínimos, cobertura limitada pero justificada.
- Suspenso (<50): requisitos incumplidos, ausencia de tests o entrega incompleta.

Entrega

Subir un repositorio (Git) con la estructura mínima:

- README.md — cómo ejecutar y comprobar tests/coverage.
- docs/ — `manual-usuario.adoc`, `entrevista-notas.adoc`, `coverage-analisis.adoc`, `system-test-report.adoc` y mockups.
- src/ — código fuente.
- tests/ — pruebas unitarias.
- tests/system/ — pruebas de sistema y scripts/notes para reproducirlas.
- coverage/ — informe HTML (o indicar cómo generar en README).

Referencias

- https://mkoertgen.github.io/hello.nplant/2-salt-ui/
- https://crashedmind.github.io/PlantUMLHitchhikersGuide/
- https://mermaid.js.org/syntax/requirementDiagram.html
- https://www.selenium.dev/
- https://playwright.dev/
- https://www.cypress.io/