

# Guía para la Presentación Final del Proyecto

Taller de Proyecto de la Especialidad — Técnico en Programación y Análisis de Sistemas

Este documento es para orientarlos por última vez para la preparación de su **presentación final**, la cual debe **sintetizar y demostrar** el trabajo que hicieron a lo largo de sus proyectos.

La presentación **no es un resumen del informe**, sino una **demonstración estructurada** del sistema y del proceso seguido.

**A modo de cierre personal:** agradezco sinceramente el compromiso y la paciencia de cada uno de ustedes durante el desarrollo de esta asignatura y las anteriores. Las exigencias, el nivel de detalle y la forma de trabajo planteada tuvieron siempre un único objetivo: **prepararlos mejor para el ejercicio profesional real**. Esta guía representa el último apoyo que les entrego, y al igual que todas las guías anteriores, siempre las hice con la convicción de que lo aprendido será una base simple pero sólida para su futuro como profesionales responsables, críticos y capaces.

## Duración, fecha y enfoque general

**Formato de tiempo por estudiante (ideal):**

- **15 minutos** de presentación (explicación concisa).
- **15 minutos** para software + testing guiado + preguntas (rueda de preguntas).

**Enfoque:** demostrar, no solo explicar.

**Evidencia clave:** prototipo + pruebas + repositorio GitHub.

**Regla práctica:** La presentación debe ser **concisa** para dejar tiempo a la validación con el software. Idealmente, todo el proceso debe mantenerse **bajo 30 minutos**.

## ¿Qué debe demostrar la presentación?

La presentación final debe dejar claro que el estudiante:

- Entendió el problema y el contexto.
- Diseñó una solución coherente (requerimientos + diseño).
- Implementó un prototipo funcional.
- Probó su sistema.
- Versionó y documentó correctamente.
- Agregó valor más allá del CRUD<sup>a</sup>, apoyando decisiones (MIS básico).
- Explicó lo innovador del proyecto (acordado desde el Avance 1).

---

<sup>a</sup>**CRUD:** crear, leer, actualizar y eliminar datos. En el proyecto es necesario, pero debe acompañarse de un aporte a la toma de decisiones (reportes, alertas, rankings).

## Estructura base de la presentación (15 min)

**La presentación se evalúa por coherencia:** lo diseñado debe verse implementado y probado, con evidencia en GitHub.

### 1) Identificación del proyecto

Debe mostrar:

- Nombre del proyecto.
- Nombre del estudiante.
- Carrera y asignatura.
- Enlace al repositorio GitHub.
- Versión final del sistema (Release<sup>a</sup>).

**Propósito:** establecer formalidad y trazabilidad.

---

<sup>a</sup>**Release:** publicación formal en GitHub que agrupa una versión estable (normalmente asociada a un tag), e incluye notas y, si corresponde, archivos adjuntos.

### 2) Problema y contexto

Debe explicar de manera directa:

- Qué problema real aborda el sistema.
- En qué contexto se aplica (organización, área, proceso).
- Quiénes son los usuarios principales (2–3 roles).

### 3) Objetivos, valor e innovación

Debe incluir:

- Objetivos SMART.
- Valor del sistema (beneficio concreto).
- Qué tiene de innovador (una idea concreta, no marketing).
- Cómo ayuda a tomar decisiones (aunque sea MIS básico: reporte, ranking, alerta).

### 4) Alcance del proyecto

Debe quedar claro:

- Qué funcionalidades **incluye**.
- Qué funcionalidades **NO incluye** (control de alcance).

### 5) Diseño del sistema

Debe mostrar evidencias visuales (sin teoría extra):

- Modelo Entidad-Relación (MER).
- Diagrama de casos de uso.
- Wireframes principales (login + pantalla central).

**Clave:** se evalúa coherencia entre diseño y sistema, no memorizar UML.

### 6) Metodología y organización del trabajo

Debe explicar brevemente:

- Método de trabajo individual (Kanban personal, XP adaptado, Lean).
- Cómo se organizó el trabajo semanal.
- Uso del tablero Kanban en GitHub Projects.

### 7) Demostración del prototipo

Aquí se debe mostrar el sistema funcionando.

Debe incluir un flujo completo:

- Acceso al sistema.
- Ejecución de una funcionalidad principal (CRUD).
- Validación de errores (con ejemplos).
- Consulta o módulo que apoye una decisión: reporte / ranking / alerta (Si es que está implementado).

En la presentación se usan imágenes del software para explicar el flujo. La demo confirma que funciona, pero el flujo debe entenderse aunque la demo falle.

## 8) Pruebas y calidad

Debe evidenciar:

- Plan de pruebas ejecutado.
- Al menos 2 o 3 casos de prueba reales.
- Evidencias (capturas, logs).
- Al menos un error detectado y corregido (mejora real).

**Testing guiado<sup>a</sup> (obligatorio):**

La prueba es guiada y el estudiante debe elegir cómo hacerla y dónde aplicarla. **Debe quedar detallada en el informe** con imágenes que la respalden. En la presentación, se muestra un resumen y dónde está la evidencia en el repositorio.

---

<sup>a</sup>**Testing guiado:** prueba planificada y ejecutada paso a paso (manual o semi-manual), con casos definidos, resultados esperados, resultados obtenidos y evidencias (capturas/logs).

## 9) Evidencia GitHub y versionado

Debe mostrar:

- Repositorio organizado (carpetas y README).
- Commits frecuentes y descriptivos.
- Uso de issues.
- Tag de versión<sup>a</sup>.
- Release final publicado.

**GitHub es parte de la evaluación (no opcional).**

---

<sup>a</sup>**Tag:** etiqueta que marca un punto estable del historial del repositorio (por ejemplo v3.0).

## 10) Cierre y reflexión final

Debe cerrar con:

- Principales logros del proyecto.
- Qué quedó pendiente o qué mejoraría.
- Aprendizajes técnicos y de gestión.
- Proyección profesional del trabajo (portafolio).

**Reflexión personal (obligatoria):** breve, honesta y técnica. Ejemplos: qué decisión fue difícil, qué cambiaría en el diseño, qué aprendió sobre pruebas o sobre control de versiones.

## Errores comunes a evitar

- Leer diapositivas.
- Explicar teoría en exceso (la teoría ya está en los apuntes).
- No mostrar el sistema funcionando.
- No mostrar GitHub (repo, tags, release, evidencias).
- Pasarse del tiempo (no deja espacio a la demo y preguntas).

## Criterio clave (lo que se espera ver)

**Criterio clave:** la presentación debe demostrar coherencia entre lo diseñado, lo implementado y lo documentado. Si algo se prometió en los avances, debe verse reflejado en el prototipo, en las pruebas y en la evidencia del repositorio.