TUP – 2do Año
Gestión de Desarrollo de Software
Prof: Lic. Adrian Aroca

Gestión de Desarrollo del Software
TRABAJO PRÁCTICO FINAL

---

1. INTRODUCCIÓN

El presente trabajo práctico tiene como objetivo aplicar los conceptos de gestión de proyectos de software vistos durante el cuatrimestre. Los equipos realizarán la **gestión integral** del proyecto que desarrollarán en la materia Programación IV.
Importante: Este trabajo consiste en **gestionar** el proyecto de Programación IV, no en desarrollar software adicional. Aplicaremos técnicas de estimación, planificación, seguimiento y control sobre el proyecto real que están construyendo.

Requisito Institucional
Este informe de gestión será solicitado por la coordinación de la carrera como parte de la documentación requerida para la presentación del "Trabajo Final Integrador" (Ex PPS). Por lo tanto, es fundamental que la documentación sea completa, profesional y refleje fielmente el proceso de gestión llevado a cabo durante el proyecto.
La calidad y completitud de este documento no solo impacta en la evaluación de esta materia, sino que también forma parte del expediente académico necesario para la acreditación del Trabajo Final Integrador.

---

2. CONFORMACIÓN DE GRUPOS

- Los grupos deben estar compuestos por **3 a 4 integrantes**.
- Todos los integrantes deben participar activamente en las tareas de gestión y documentación.
- Los grupos serán los mismos que en Programación IV.

---

3. CONTEXTO DEL PROYECTO A GESTIONAR

El presente Trabajo Práctico se realizará sobre el proyecto desarrollado en Programación IV. Dependiendo de la situación académica de cada equipo, existen dos escenarios posibles:

<u>Opción 1:</u> Proyecto Full-Stack (Frontend + Backend)
- Para equipos que cuenten con un backend desarrollado (de Programación III o nuevo)
- Alcance de Gestión: Se gestiona el proyecto completo (frontend + backend), aplicando las técnicas de estimación y planificación a ambos componentes.

<u>Opción 2:</u> Proyecto Frontend con Persistencia Simulada
- Para equipos que NO dispongan de backend real.
- Se utilizará JSON Server u otra herramienta similar para simular la persistencia.
- Alcance de Gestión: Se gestiona el frontend y la capa de simulación como un proyecto integrado.

Nota importante: Las técnicas de gestión (PFA, COCOMO, arquitectura C4, SCRUM) se aplican de igual manera en ambas opciones, adaptando los ejemplos al contexto específico de cada equipo.

4. CRONOGRAMA GENERAL

Semana 1: Presentación y Definición

- Presentación del TP Final
- Conformación definitiva de grupos
- Elección de opción (1 o 2)
- Definición inicial de requisitos funcionales

Semanas 2-3: Estimación y Planificación

- Aplicación de técnicas de medición (PFA)
- Estimación con COCOMO y técnica adicional obligatoria
- Diseño de arquitectura (C4)
- Identificación de riesgos

---

5. ENTREGABLES OBLIGATORIOS

Los equipos deberán presentar un **documento único en formato PDF** que contenga los siguientes apartados:

5.1 Portada y Descripción del Proyecto

Contenido:

- Nombre del proyecto (Carrera, Materia, Año de cursada, Profesor, Ciclo lectivo)
- Integrantes del equipo
- Misión y alcance del proyecto
- Opción elegida (1 o 2)
- Descripción general del sistema

5.2 Especificación de Requisitos Funcionales

Objetivo: Documentar de manera completa todos los requisitos funcionales del proyecto desarrollado en Programación IV, estableciendo un registro oficial que servirá como base para futuros trabajos (Trabajo Final Integrador - TFI).

IMPORTANTE - Requisito Institucional para el TFI:

Este registro completo de requisitos funcionales tiene un propósito crítico: **servirá como constancia oficial de qué funcionalidades fueron efectivamente implementadas en el proyecto de Programación IV**.

Cuando presenten su Trabajo Final Integrador (Tesis), se les exigirá agregar 4 requisitos funcionales nuevos al proyecto ya desarrollado. **Esta tabla de alcance final será el documento de referencia que determinará qué requisitos ya fueron implementados y cuáles son verdaderamente nuevos para el TFI.**

ESTRUCTURA DE ENTREGA - DOS COMPONENTES OBLIGATORIOS:

COMPONENTE 1: Tabla de Alcance Final (TODOS los Requisitos Funcionales)

Deben listar **TODOS** los requisitos funcionales definidos y/o aprobados por el docente de Programación IV para el proyecto, sin excepción.

Formato de la Tabla:

Código	Nombre	Prioridad	Implementado
RF-01	Autenticación de Usuarios	ALTA	SI
RF-02	Gestión de Permisos	ALTA	SI
RF-03	Registro de Actividad	MEDIA	SI
RF-04	Reportes de Gestión	BAJA	NO
...	...	...	...

Columnas obligatorias:

- **Código:** RF-01, RF-02, RF-03, etc. (numeración consecutiva)
- **Nombre descriptivo:** Título claro del requisito funcional
- **Prioridad:** ALTA / MEDIA / BAJA (según criterio acordado con docente de PRG IV)
- **Implementado:** SI / NO (indicar claramente si fue implementado en PRG IV o no)

Nota crítica sobre la columna "Implementado":

- Marquen **SI** únicamente para requisitos que fueron **efectivamente implementados y funcionan** en el proyecto de Programación IV.
- Marquen **NO** para requisitos que fueron planificados, pero no implementados, o que quedaron incompletos.
- Esta columna será el **punto de partida oficial** para la definición de los requisitos adicionales del Trabajo Final Integrador (TFI). Los 4 requisitos extra para el TFI **deben ser funcionales marcados como NO** en esta tabla, o bien requisitos completamente nuevos no listados aquí.

COMPONENTE 2: Documentación Detallada de Requisitos Seleccionados (5-8 Requisitos)

De la tabla completa anterior, seleccionen **entre 5 y 8 requisitos funcionales representativos** para documentarlos en profundidad. La selección debe incluir requisitos de diferentes prioridades y complejidades para dar una visión representativa del proyecto.

Para estos requisitos seleccionados, pueden elegir uno de los dos formatos de documentación detallada que se describen a continuación. Ambas opciones son válidas, pero la **Opción B (Estructura Detallada)** es altamente recomendada por los beneficios que aporta en las siguientes etapas del proyecto (PFA y estimaciones).

OPCIÓN A: Descripción General

Pueden documentar los requisitos utilizando el formato con el que ya están familiarizados, centrado en una descripción narrativa y completa.

Para cada requisito incluir:

- **Código:** RF-01, RF-02, etc.
- **Nombre descriptivo:** Ej. "Gestión de usuarios"
- **Descripción General:** Una descripción narrativa que explique QUÉ hace el sistema y CÓMO lo hace de principio a fin.
- **Prioridad:** Alta / Media / Baja

Ejemplo de la Opción A:

RF-01: Autenticación de Usuarios

Descripción General: El sistema debe permitir que un usuario ingrese su email y contraseña en un formulario. El sistema validará que el email tenga un formato válido y que la contraseña tenga al menos 8 caracteres. Luego, consultará el sistema de persistencia para verificar si las credenciales son correctas. Si lo son, generará un token de sesión JWT (válido por 24 horas) y redirigirá al usuario a su dashboard. Si las credenciales son incorrectas, se mostrará un mensaje de error. Tras 3 intentos fallidos, la cuenta se bloqueará temporalmente por 15 minutos. Las contraseñas se almacenan encriptadas en el sistema de persistencia.

Prioridad: ALTA

OPCIÓN B (Recomendada): Estructura Detallada

Para aquellos equipos que deseen profundizar y obtener una ventaja en las secciones de medición y estimación, se recomienda este formato. Pensar en los siguientes puntos los ayudará a comprender mejor el requisito.

Nota importante sobre requisitos compuestos: Algunos RF pueden agrupar varias operaciones relacionadas (ej: un ABM - Alta, Baja, Modificación). En estos casos, se debe detallar cada operación por separado dentro de los campos correspondientes (Entradas, Salidas, Proceso), utilizando listas para mayor claridad, como se muestra en el nuevo ejemplo.

Para cada requisito incluir:

- **Código:** RF-01, RF-02, etc.
- **Nombre descriptivo:** Ej. "Gestión de usuarios"
- **Descripción detallada:**
    - **Contexto:** Relación con otros requisitos. ¿Es prerrequisito de otro? ¿Qué funcionalidades desbloquea?
    - **Objetivo:** La meta final de la funcionalidad. ¿Qué necesidad del usuario satisface?

    - **Entradas:** Los datos específicos que debe proporcionar el usuario o el sistema.
    - **Salidas:** Los resultados, datos o mensajes que el sistema devuelve.
    - **Proceso:** Los pasos lógicos específicos que sigue el sistema para transformar las entradas en salidas.
    - **Reglas de negocio:** Las condiciones, restricciones o fórmulas que debe cumplir la funcionalidad.
- **Prioridad:** Alta / Media / Baja
- **Notas adicionales:** Cualquier detalle relevante no cubierto anteriormente.

Utilizar la Opción B facilita enormemente el trabajo en las secciones 5.3 (PFA) y 5.4 (Estimaciones), ya que la información requerida para esas etapas ya estará identificada y organizada.

Ejemplo de la Opción B (RF Simple - Autenticación):

RF-01: Autenticación de Usuarios

Descripción: El sistema debe permitir que los usuarios se autentiquen mediante email y contraseña para acceder a las funcionalidades según su rol.

Contexto: Este requisito es base para RF-02 (Gestión de permisos) y RF-03 (Registro de actividad).

Objetivo: Garantizar acceso seguro y diferenciado según perfil de usuario.

Entradas:

- Email (string, formato válido)
- Contraseña (string, mínimo 8 caracteres)

Salidas:

- Token de sesión (JWT)
- Datos del usuario autenticado
- Mensaje de error en caso de credenciales inválidas

Proceso:

1. Usuario ingresa credenciales en formulario de login
2. Sistema valida formato de email y longitud de contraseña
3. Sistema consulta el backend/persistencia para verificar credenciales
4. Si son válidas: genera token y redirige al dashboard
5. Si son inválidas: muestra mensaje de error

Reglas de negocio:

- Máximo 3 intentos fallidos antes de bloqueo temporal (15 minutos)
- Las contraseñas se almacenan encriptadas (bcrypt)
- El token tiene vigencia de 24 horas

Prioridad: ALTA

Ejemplo de la Opción B (RF Compuesto - ABM):

RF-02: Gestión de Usuarios (ABM)

Descripción: Permitir a un administrador crear, visualizar, modificar y desactivar usuarios del sistema.

Contexto: Requiere de RF-01 (Autenticación) para identificar al administrador. Es prerrequisito para RF-04 (Asignación de Roles).

Objetivo: Mantener actualizada y segura la base de usuarios del sistema.

Entradas:

- **Para ALTA:** Datos del nuevo usuario (nombre, email, rol), contraseña temporal.
- **Para CONSULTA/LISTADO:** Filtros opcionales (por nombre, email, rol).
- **Para MODIFICACIÓN:** ID del usuario, campos a modificar.
- **Para BAJA/DESACTIVACIÓN:** ID del usuario a desactivar.

Salidas:

- **Para ALTA:** Confirmación de creación, ID asignado, email de bienvenida (automático).
- **Para CONSULTA/LISTADO:** Lista de usuarios, detalle completo de un usuario.
- **Para MODIFICACIÓN:** Confirmación de cambios, datos actualizados.
- **Para BAJA/DESACTIVACIÓN:** Confirmación de desactivación.

Proceso:

- **Proceso de ALTA:**
    1. Admin completa formulario de alta.
    2. Sistema valida unicidad del email.
    3. Sistema encripta contraseña temporal.
    4. Sistema guarda usuario y asigna ID.
    5. Sistema envía email de bienvenida (asincrónico).
- **Proceso de CONSULTA:**
    1. Admin visualiza listado, puede aplicar filtros.
    2. Al seleccionar un usuario, se muestran sus detalles.
- **Proceso de MODIFICACIÓN:**
    1. Admin edita los datos deseados de un usuario existente.
    2. Sistema valida cambios (ej: nuevo email no exista).
    3. Sistema actualiza la información.
- **Proceso de BAJA/DESACTIVACIÓN:**
    1. Admin selecciona "Desactivar" sobre un usuario.
    2. Sistema solicita confirmación.
    3. Sistema actualiza el estado del usuario a "inactivo" (borrado lógico).

Reglas de negocio:

- No se puede eliminar un usuario físicamente del sistema de persistencia, solo desactivar (borrado lógico).
- El email debe ser único en el sistema.
- Un usuario desactivado no puede iniciar sesión.

- Solo los Superadmins pueden modificar o desactivar a otros administradores.

Prioridad: ALTA

Sugerencia: Independientemente de la opción que elijan para la mayoría de los requisitos, se recomienda intentar desarrollar **al menos los dos primeros (ej: RF-01 y RF-02) con la Opción B**. Esta práctica les permitirá experimentar directamente sus beneficios y decidir con mayor conocimiento si adoptarla para el resto.

RESUMEN DE LO QUE DEBEN ENTREGAR EN LA SECCIÓN 5.2:

Tabla de Alcance Final con TODOS los RF del proyecto (Código, Nombre, Prioridad, Implementado SI/NO)

Documentación detallada de 5-8 RF seleccionados usando Opción A o B.

5.3 Medición Funcional - Análisis de Puntos de Función (PFA)

Objetivo: Aplicar la técnica de Análisis de Puntos de Función sobre los requisitos definidos.

Nota para equipos con Frontend + JSON Server:
Si su proyecto no incluye un backend real:
- Los "ILFs" (Archivos Lógicos Internos) se refieren a las estructuras de datos gestionadas por JSON Server.
- Documenten el JSON Server como un componente del sistema en la arquitectura C4.
- En el PFA, traten las operaciones CRUD sobre JSON Server como si fueran sobre una BD real.
- Ejemplo: El archivo db.json con la colección "usuarios" se cuenta como un ILF llamado "USUARIOS". Si tienen múltiples colecciones (usuarios, productos, pedidos), cada colección es un ILF separado.

Aclaración de Alcance Fundamental:

- Mientras que en la sección 5.2 se documentaron de manera detallada entre 5 y 8 RF a modo de ejemplo, **el análisis de Puntos de Función en esta sección 5.3 debe realizarse sobre la totalidad de los Requisitos Funcionales definidos para el proyecto en Programación IV**.
- El objetivo de la medición es obtener el tamaño funcional **completo** del sistema**.**

5.3.1 Base para el Conteo: La EDT (Estructura de Desglose del Trabajo)
Es obligatorio que el conteo de Puntos de Función se realice sobre una EDT previamente definida. Esto asegura que el análisis sea **sistemático, completo y evite omisiones o duplicaciones**.

- Sin una EDT, el conteo de PFs tiende a ser inconsistente y subjetivo. La EDT actúa como un "mapa" que garantiza que cada componente funcional sea identificado y medido.

5.3.2 Clasificación de la Complejidad
La complejidad (Baja, Media, Alta) de cada función identificada **idealmente no debe ser subjetiva**. El método estándar determina la complejidad utilizando las **tablas oficiales del método PFA**, que evalúan la cantidad de **Tipos de Datos (DETs)** y **Archivos Referenciados (FTRs)**. Se proveerá la tabla de referencia correspondiente en clase.

Alternativa Práctica:
Si la aplicación estricta de las tablas de DETs y FTRs representa una barrera inicial demasiado alta para el equipo, se permite **asignar una complejidad "Media" por defecto a todas las funciones identificadas** para poder continuar con el ejercicio de estimación.

- Esta simplificación **reduce la precisión** del resultado final de los Puntos de Función, ya que no refleja las variaciones reales de complejidad entre las distintas funcionalidades. Lo aconsejado (y valorado), es la utilización de DET's y RET'Ss.

5.3.3 Principio Fundamental: No Re-contar ALIs
Un **Archivo Lógico Interno (ALI o ILF)** representa un grupo de datos lógicamente relacionados mantenido por la aplicación. **Una vez identificado y contado un ALI, no se vuelve a contar** aunque sea utilizado por múltiples funcionalidades.

- **Ejemplo:** Si se identifica el ALI "USUARIOS" para el RF de "Gestión de Usuarios", cualquier otro requisito (ej: "Login", "Auditoría") que consulte o use este mismo archivo **no** generará un nuevo ALI. En su lugar, se registrará como un Archivo de Interfaz Externa (AIE o EIF) si es de solo lectura desde el frontend, o se considerará que las funcionalidades que lo modifican son Entradas Externas (EE o EI) que actúan sobre el ALI ya contado.

Incluir:

1. **Tabla de conteo con las siguientes columnas:**
   - **Tipo de función (EI, EO, EQ, ILF, EIF)**
   - **Descripción:**
     - **Para EI/EO/EQ: Describir la operación (ej: "ALTA de usuario").**
     - **Para ILF/EIF: Identificar el archivo de datos (ej: "ALI: USUARIOS", "AIE: CATÁLOGO_PROD").**
   - **Complejidad (Baja/Media/Alta) - Justificada idealmente con DETs y FTRs.**
   - **Puntos asignados**
   - **Justificación**
2. **Cálculo de Puntos de Función no ajustados**
3. **Factor de Ajuste**:
   - Considerar las 14 características generales del sistema (GSC)
   - Asignar valores de 0 a 5 según influencia
   - Calcular Valor del Factor de Ajuste (VFA)
4. **Puntos de Función Ajustados**

Ejemplo de Aplicación PFA para el RF-02 "Gestión de Usuarios"

Supongamos el RF-02 documentado en la sección 5.2. Su desglose en componentes para el PFA sería:

Tipo de Función	Descripción	Complejidad	Puntos	Justificación (DETs / FTRs)
ILF (ALI)	**USUARIOS** (Archivo que almacena los datos de los usuarios)	Baja	7	~5 DETs (ID, nombre, email, rol, estado). 1 FTR (sí mismo).
EI (EE)	**ALTA de usuario** (Proceso que crea un nuevo registro en USUARIOS)	Baja	3	~4 DETs de entrada. 1 FTR (USUARIOS).
EI (EE)	**MODIFICACIÓN de usuario** (Proceso que actualiza USUARIOS)	Baja	3	~4 DETs de entrada. 1 FTR (USUARIOS).
EI (EE)	**DESACTIVACIÓN de usuario** (Proceso que cambia el estado en USUARIOS)	Baja	3	~1 DET (estado). 1 FTR (USUARIOS).
EQ (CE)	**CONSULTA/LISTADO de usuarios** (Recupera datos de USUARIOS)	Baja	3	~4 DETs de salida. 1 FTR (USUARIOS).

Notas clave del ejemplo:

- El **ILF "USUARIOS"** se cuenta **una sola vez**, aunque varias Entradas (EI) y Consultas (EQ) trabajen sobre él.
- Cada operación (Alta, Modificación, etc.) es una **Entrada Externa (EI)** distinta porque cada una es un proceso transactional único.
- La complejidad se asignó como "Baja" basándose en un conteo estimado de DETs y FTRs según la tabla oficial.

5.4 Estimaciones de Esfuerzo, Tiempo y Costo

Objetivo: Estimar el proyecto utilizando **dos técnicas de estimación obligatorias**:

Para equipos con JSON Server: Al calcular KLOC para COCOMO, deben tratar JSON Server como un "backend simplificado":

- Asignen los Puntos de Función correspondientes a las operaciones de persistencia (los ILFs y transacciones que trabajen con el archivo db.json).
- Utilicen un factor de conversión reducido de 30 LDC/PF para este componente (en lugar de 50 LDC/PF que usarían para Spring Boot).

- Explicación: JSON Server no requiere programar lógica de negocio compleja, validaciones del lado del servidor, ni capas de servicio. Por lo tanto, el esfuerzo en "líneas de código equivalentes" es mucho menor.

Ejemplo (Proyecto con JSON Server):

Componente	Tecnología	PF	Factor (LDC/PF)	KLOC
Backend simplificado	JSON Server	30	30	0.9
Frontend	Angular	20	40	0.8
TOTAL		**50**		**1.7**

Técnica Obligatoria 1: COCOMO
Incluir:

1. Conversión de PF a KLOC (Líneas de código) - Dos Enfoques Posibles

OPCIÓN A (Recomendada - Mayor Precisión): Enfoque Desglosado por Tecnología

- Utilizar la EDT como base para desglosar el proyecto en sus **componentes tecnológicos principales** (ej: Backend Spring Boot, Frontend Angular).
- Para cada componente tecnológico, aplicar su factor de conversión específico.
- **Indicar para cada componente:**
    - Tecnología
    - Factor de conversión utilizado (ej: 50 LDC/PF para Java, 40 LDC/PF para Angular)
    - Puntos de Función asignados
    - KLOC calculadas
- **Sumar las KLOC de todos los componentes** para obtener las KLOC totales.

Ejemplo - Opción A (Proyecto Full-Stack con Spring Boot):

Componente	Tecnología	PF	Factor (LDC/PF)	KLOC
Backend	Spring Boot	30	50	1.50
Frontend	Angular	20	40	0.80
TOTAL		**50**		**2.30**

OPCIÓN B (Simplificada - Para Avance Rápido): Enfoque con Tecnología Representativa

- Seleccionar una **tecnología representativa** para todo el proyecto (ej: la tecnología predominante como Java/Spring Boot o TypeScript/Angular).
- Utilizar un **solo factor de conversión promedio** para convertir el total de Puntos de Función directamente a KLOC.
- **Justificar la elección** de la tecnología y el factor representativo.

Ejemplo - Opción B:

- Proyecto con **50 PF**.
- Tecnología representativa: **Spring Boot (Backend)**.
- Factor de conversión utilizado: **50 LDC/PF**.
- **KLOC totales = 50 PF * 50 LDC/PF = 2.50 KLOC**.

Nota: La Opción A es más precisa y refleja mejor la realidad del proyecto mixto (frontend/backend). La Opción B es aceptable para el Checkpoint 1, pero se sugiere migrar a la Opción A para la entrega final.

2. Aplicación del modelo COCOMO:

- Indicar el tipo de proyecto seleccionado (Orgánico/Semi-acoplado/Empotrado).
- Aplicar las fórmulas del COCOMO Básico utilizando las **KLOC totales** obtenidas.
- Calcular:
    - Esfuerzo (personas-mes)
    - Tiempo de desarrollo (meses)
    - Personas promedio necesarias

3. Justificación:

- Explicar por qué eligieron ese tipo de proyecto.
- Listar los supuestos utilizados (incluyendo la elección del enfoque A o B para KLOC y los factores de conversión).

Técnica Obligatoria 2 (elegir una)

Opción A - Estimación por Analogía:
- Comparar con proyectos similares previos
- Ajustar según diferencias identificadas

Opción B - Estimación por Tres Puntos:
- Escenario optimista
- Escenario más probable
- Escenario pesimista
- Calcular promedio ponderado: (O + 4M + P) / 6

Opción C - Planning Poker:
- Documentar sesión de estimación con el equipo
- Story points asignados a cada requisito
- Conversión a horas/días

4. **Estimación de Costos**:
    - Definir costo por hora/persona (puede ser estimado/teórico)
    - Calcular costo total del proyecto
    - Desglosar por etapas si es posible

5.5 Arquitectura del Sistema - Modelo C4

Objetivo: Documentar la arquitectura del sistema utilizando el modelo C4 (Context, Containers).

Diagrama de Contenedores - Adaptación según escenario:
Opción 1 - Proyecto Full-Stack:

Mostrar tres contenedores principales:
- Frontend (especificar tecnología: Angular/React/Vue)
- Backend (especificar tecnología: Spring Boot/Node.js/etc.)
- Base de Datos (especificar: MySQL/PostgreSQL/etc.)

Opción 2 - Proyecto Frontend + JSON Server:
Mostrar dos contenedores principales:
- Frontend (especificar tecnología: Angular/React/Vue)
- JSON Server (como componente de persistencia simulada)

Importante: En ambos casos, documentar claramente:
- Las tecnologías utilizadas en cada contenedor
- Los protocolos de comunicación entre componentes (HTTP/REST, WebSockets, etc.)
- Las responsabilidades específicas de cada contenedor

Nivel 1: Diagrama de Contexto
Incluir:
- El sistema como caja central
- Actores/usuarios que interactúan con el sistema
- Sistemas externos con los que se integra
- Relaciones y flujos principales

Nivel 2: Diagrama de Contenedores
Incluir:
- Componentes principales del sistema (frontend, backend, base de datos, etc.)
- Otros contenedores relevantes según el proyecto
- Tecnologías utilizadas en cada contenedor
- Protocolos de comunicación entre componentes

Acompañar cada diagrama con:
- Breve descripción textual
- Tecnologías específicas
- Responsabilidades de cada componente

Herramientas sugeridas: Draw.io, Lucidchart, diagrams.net, PlantUML

5.6 Gestión de Riesgos

Objetivo: Identificar y analizar los principales riesgos del proyecto.
Formato: Risk Register con las siguientes columnas:

ID	Riesgo	Categoría	Probabilidad	Impacto	Exposición	Estrategia	Responsable
R-01	Descripción	Técnico/Alcance/Equipo	Alta/Media/Baja	Alto/Medio/Bajo	P×I	Mitigar/Evitar/Transferir/Aceptar	Nombre

Ejemplo:

ID	Riesgo	Categoría	Probabilidad	Impacto	Exposición	Estrategia	Responsable
R-01	Problemas de integración entre frontend Angular y backend Spring Boot	Técnico	Media	Alto	Media × Alto = Alto	**Mitigar**: Realizar pruebas de integración tempranas. Definir contratos de API claros desde el inicio.	Juan Pérez

Incluir mínimo 5-8 riesgos que contemplen:
- **Riesgos técnicos**: Integración entre componentes, rendimiento, seguridad
- **Riesgos de alcance**: Cambios en requisitos, funcionalidades complejas
- **Riesgos de tiempo**: Retrasos, estimaciones incorrectas
- **Riesgos de equipo**: Disponibilidad de miembros, conocimiento técnico

Para cada riesgo definir:

- Estrategia de mitigación específica
- Acciones preventivas concretas
- Responsable de monitorear el riesgo

5.7 Implementación de SCRUM

Objetivo: Documentar la aplicación de la metodología ágil SCRUM durante el desarrollo.

5.7.1 Definición de Roles
Especificar claramente:

- **Scrum Master**: Nombre del integrante y responsabilidades asumidas
- **Product Owner**: Nombre del integrante y responsabilidades asumidas
- **Development Team**: Resto de los integrantes

5.7.2 Product Backlog
Incluir:

- Lista priorizada de requisitos/historias de usuario
- Estimación de cada ítem (horas, story points, o similar)
- Criterios de aceptación resumidos

5.7.3 Sprint 1 (Duración: 2 semanas aproximadamente)
Documentar:
Sprint Planning:

- Objetivo del sprint
- Historias/tareas seleccionadas del backlog
- Estimación de capacidad del equipo
- Distribución de tareas entre los miembros

Sprint Execution:

- Resumen de tareas completadas
- Captura de pantalla del tablero JIRA/similar al finalizar el sprint

Sprint Review:

- Funcionalidades completadas y demostradas
- Funcionalidades no completadas (si las hay) y razones

Sprint Retrospective:

- ¿Qué funcionó bien?
- ¿Qué se puede mejorar?
- Acciones de mejora para el próximo sprint

5.7.4 Sprint 2 (Duración: 1-2 semanas aproximadamente)
Documentar los mismos puntos que Sprint 1

5.7.5 Uso de JIRA/similar
Incluir:

- Capturas de pantalla que evidencien:
  - Product Backlog configurado
  - Tableros de cada sprint (To Do, In Progress, Done)
  - Burndown chart (si está disponible)
  - Asignación de tareas a miembros del equipo

5.8 Gestión de Cambios

Objetivo: Documentar **al menos 1 (una) Solicitud de Cambio** que haya surgido durante la ejecución del proyecto.

Formato de Solicitud de Cambio:

SOLICITUD DE CAMBIO #001

Fecha de solicitud: DD/MM/AAAA
Solicitante: [Nombre o rol]

CAMBIO PROPUESTO:
[Descripción clara y detallada del cambio solicitado]

JUSTIFICACIÓN:
[Razón por la cual se solicita este cambio]

IMPACTO EN EL PROYECTO:

1. Alcance:
  - Requisitos afectados: [listar RF-XX]
  - Nuevas funcionalidades requeridas: [si aplica]

2. Tiempo:
  - Estimación original: [X horas/días]
  - Nueva estimación: [Y horas/días]
  - Impacto: +/- [diferencia]

3. Costo:
  - Costo original: [$XXX]
  - Nuevo costo estimado: [$YYY]
  - Impacto: +/- [diferencia]

4. Recursos:
  - Recursos adicionales necesarios: [si aplica]

5. Riesgos:
  - Nuevos riesgos introducidos: [si aplica]

DECISIÓN:
☐ APROBADO ☐ RECHAZADO ☐ DIFERIDO

JUSTIFICACIÓN DE LA DECISIÓN:
[Explicar por qué se aprobó o rechazó el cambio]

Fecha de decisión: DD/MM/AAAA
Responsable de la decisión: [Nombre - Rol]

5.9 Cierre del Proyecto

Objetivo: Realizar un análisis comparativo entre lo estimado inicialmente y lo ejecutado realmente.

5.9.1 Comparación Estimaciones vs. Real
Tabla comparativa:

Métrica	Estimación Inicial	Valor Real	Desviación	% Desviación
Puntos de Función	[X PF]	[Y PF]	+/- [diferencia]	[%]
Esfuerzo (personas-mes)	[X]	[Y]	+/- [diferencia]	[%]
Tiempo (semanas)	[X]	[Y]	+/- [diferencia]	[%]
Costo total	[$X]	[$Y]	+/- [diferencia]	[%]
Requisitos planificados	[X]	[Y]	+/- [diferencia]	[%]
Requisitos completados	-	[Y]	-	[%]

5.9.2 Análisis de Desviaciones
Para cada desviación significativa (>20%), explicar:

- Causa raíz de la desviación
- ¿Se podría haber previsto?
- ¿Qué acciones se tomaron al detectarla?
- ¿Cómo impactó en el resultado final?

Ejemplo:
El esfuerzo real fue 40% superior al estimado debido a:
1. Subestimación de la complejidad de integración entre componentes
2. Problemas técnicos no contemplados en la estimación inicial
3. Necesidad de refactorizar módulos para adaptarlos a nuevos requisitos

Estas desviaciones se detectaron en el Sprint 1 y se ajustaron las estimaciones del Sprint 2. Se redujo el alcance eliminando RF-07 y RF-08 para cumplir con los plazos.

5.10 Registro de Decisiones Arquitectónicas - ADRs (OPCIONAL PERO VALORADO)

Esta sección es opcional, pero se considerará como un plus en la evaluación final.
Documenta las decisiones técnicas más importantes tomadas durante el desarrollo.

ADR-001: [Título de la primera decisión importante]
[Usar la plantilla del PDF adjunto]

ADR-002: [Título de la segunda decisión]
[Usar la plantilla del PDF adjunto]

6. CHECKPOINTS DE CONTROL

Durante el desarrollo del TP se realizarán controles de avance en las siguientes instancias:

Checkpoint 1 - Fin de Semana 3
Entregables a revisar:

- Requisitos funcionales definidos y priorizados
- PFA completo
- Estimaciones (COCOMO + técnica adicional)
- Arquitectura C4 (niveles 1 y 2)
- Risk Register inicial

Modalidad: Revisión en clase o entrega parcial en plataforma (a definir)

Checkpoint 2 - Fin de Semana 5
Entregables a revisar:

- JIRA/similar actualizado con ambos sprints
- Sprint 1 completamente documentado
- Evidencia de avance en Sprint 2
- Solicitud de cambio documentada

Modalidad: Revisión de JIRA/similar en clase

7. ENTREGA FINAL

Fecha de Entrega
Semana 6 - Fecha exacta a confirmar en clase.

Formato de Entrega
Documento PDF único que contenga todos los apartados solicitados en la sección 5.

Nomenclatura del archivo:
TPFinal_Gestion_Grupo[Número]_[Apellidos].pdf
Ejemplo: TPFinal_Gestion_Grupo3_Fernandez_Lopez_Martinez.pdf

Medio de entrega: Plataforma educativa

Presentación Oral (Opcional/A confirmar)
Si se requiere presentación oral, tendrá las siguientes características:

- Duración: 15-20 minutos por grupo
- Contenido: Resumen ejecutivo del proyecto gestionado
- Foco en: Estimaciones, desviaciones, aprendizaje
- Todos los integrantes deben participar

8. CRITERIOS DE EVALUACIÓN

La evaluación contemplará los siguientes aspectos:
Documentación Técnica

- **PFA**: Correcta aplicación de la técnica, conteo detallado (10%)
- **Estimaciones**: COCOMO correctamente aplicado + técnica adicional bien fundamentada (10%)
- **Arquitectura C4 + Risk Register**: Diagramas claros, identificación completa de riesgos (15%)

Gestión con SCRUM

- **Definición de roles y planificación**: Roles claros, backlog priorizado (10%)
- **Ejecución y seguimiento en JIRA/similar**: Tableros actualizados, evidencia de trabajo continuo (15%)
- **Gestión de cambios**: Cambio bien documentado y analizado (10%)

Análisis y Cierre

- **Calidad del documento final**: Redacción, formato, completitud (10%)
- **Análisis de desviaciones**: Comparación estimado vs real, justificaciones sólidas (10%)

Presentación

- **Claridad expositiva**: Comunicación efectiva de decisiones de gestión
- **Capacidad de justificar**: Argumentación de elecciones realizadas

Nota: Los porcentajes son orientativos y pueden ajustarse según criterio docente.

9. CONSIDERACIONES IMPORTANTES

Coordinación con Programación IV

- Este TP de Gestión se basa en el proyecto de Programación IV, pero **son materias independientes**.
- La evaluación de Gestión se centra en **cómo gestionaron** el proyecto, no en el código final.
- Todas las consultas **técnicas** (Angular, Spring Boot, JSON Server, bases de datos) deben dirigirse al docente de Programación III – IV - BDD.

Responsabilidades de cada materia

Programación IV es responsable de:

- Definir requisitos funcionales del sistema
- Enseñar las tecnologías del proyecto
- Evaluar la calidad del código y funcionalidad

Gestión de Desarrollo de Software es responsable de:

- Verificar la correcta aplicación de técnicas de gestión
- Evaluar la documentación de gestión del proyecto
- Realizar seguimiento del uso de SCRUM y JIRA/similar

Sobre el uso de herramientas

- **JIRA/similar**: Se asume que los equipos ya tienen conocimiento de la herramienta o lo adquirirán de forma autónoma.
- **Metodología SCRUM**: Se asume conocimiento previo.
- **PFA y COCOMO**: Estas técnicas serán vistas y practicadas en clase.

Trabajo en equipo

- Se espera **participación equitativa** de todos los integrantes.
- Cada integrante debe estar capacitado para explicar **cualquier aspecto** de la documentación.
- La distribución de tareas debe ser documentada en JIRA/similar.

Originalidad y fuentes

- La documentación debe ser **producción propia** del equipo.
- Se pueden consultar fuentes externas (libros, artículos, videos) pero deben citarse apropiadamente.

10. RECOMENDACIONES PARA EL TP

Gestión del tiempo

- **No dejar todo para el final**: La documentación de gestión requiere tiempo y reflexionar sobre la misma.
- **Actualizar JIRA/similar semanalmente**: Es más fácil mantenerlo al día que actualizarlo todo al final.
- **Documentar mientras se trabaja**: Es más fácil recordar decisiones y cambios si se documentan en el momento.

Estimaciones realistas

- **Sean conservadores**: Es mejor estimar de más que quedarse cortos.
- **Incluyan buffer**: Contemplen tiempo para imprevistos (20-30% adicional).
- **Ajusten si es necesario**: Si detectan desviaciones temprano, re-estimen.

Comunicación con el docente

- **Consulten dudas temprano**: No esperen a la última semana para resolver incertidumbres.
- **Participen en los checkpoints**: Son oportunidades de feedback para corregir el rumbo.
- **Aprovechen las revisiones**: Las devoluciones de los checkpoints son para mejorar la entrega final.

Calidad de la documentación

- **Redacción clara y profesional**: Revisen ortografía y gramática.
- **Diagramas legibles**: Usen herramientas profesionales, no fotos de dibujos a mano.
- **Formato consistente**: Mantengan el mismo estilo en todo el documento.
- **Referencias cruzadas**: Vinculen secciones relacionadas (ej: "ver RF-03 en sección 5.2").

---

11. RECURSOS DE APOYO

Bibliografía recomendada

- **Software Engineering: A Practitioner's Approach** - Roger Pressman
- **Estimating Software Costs** - Capers Jones
- **The C4 Model for Software Architecture** - Simon Brown (https://c4model.com)
- **Scrum Guide** - Schwaber & Sutherland (https://scrumguides.org)

Herramientas sugeridas

- **JIRA/similar**: Gestión de proyectos ágiles (solicitar licencia educativa gratuita)
- **Draw.io / diagrams.net**: Diagramas C4 y arquitectura
- **Excel / Google Sheets**: Cálculos de PFA y COCOMO
- **Git / GitHub**: Control de versiones (deseable)

Plantillas disponibles

El docente proporcionará (o podrá consultar):

- Plantilla de especificación de requisitos
- Plantilla de Risk Register
- Plantilla de Solicitud de Cambio
- Ejemplo de cálculo PFA
- Ejemplo de aplicación COCOMO

---

12. PREGUNTAS FRECUENTES

¿Podemos cambiar de opción (1 o 2) después de la Semana 1? Solo con justificación válida y aprobación del docente. No se recomienda.

¿Qué pasa si no completamos todos los requisitos en Programación IV? La evaluación de Gestión se centra en **cómo gestionaron**, no en cuánto completaron. Analizar por qué no se completó es parte del cierre.

¿Podemos usar otra herramienta en lugar de JIRA/similar? No. JIRA/similar es obligatorio para garantizar uniformidad en las revisiones.

Si estimamos 30 PF pero al final fueron 50 PF, ¿eso está mal? No necesariamente. Lo importante es **analizar por qué hubo esa desviación** en el cierre del proyecto.

¿Hay penalización por entregar fuera de término? Sí. Consultar con el docente la política de entregas tardías (típicamente 1 punto menos por día de retraso).

¿Podemos reutilizar documentación de años anteriores? No. Cada proyecto es único y debe tener documentación original.

¿Los checkpoints son obligatorios? Sí. No presentarse a un checkpoint sin aviso previo puede impactar en la evaluación.

¿Si elegimos la Opción 2 (JSON Server), tendremos desventaja en la evaluación frente a quienes hagan Full-Stack?. No. La evaluación se centra en la GESTIÓN del proyecto, no en la complejidad técnica de la solución.

ANEXO: CHECKLIST DE ENTREGA

Antes de entregar, verificar que el documento incluya:

- [ ] Portada con nombres de todos los integrantes
- [ ] Índice o tabla de contenidos
- [ ] 5.1 - Misión y alcance del proyecto
- [ ] 5.2 - Especificación de 5-8 requisitos funcionales
- [ ] 5.3 - PFA completo con todas las tablas
- [ ] 5.4 - COCOMO + técnica adicional de estimación
- [ ] 5.5 - Diagramas C4 nivel 1 y 2
- [ ] 5.6 - Risk Register con mínimo 5 riesgos
- [ ] 5.7 - Documentación completa de SCRUM (roles, 2 sprints, JIRA/similar)
- [ ] 5.8 - Al menos 1 solicitud de cambio documentada
- [ ] 5.9 - Análisis de desviaciones y lecciones aprendidas
- [ ] Capturas de pantalla de JIRA/similar incluidas
- [ ] Diagramas claramente visibles y legibles
- [ ] Documento revisado ortográfica y gramaticalmente (no lo descuiden)
- [ ] Formato PDF, nomenclatura correcta del archivo
- [ ] Todas las secciones numeradas y referenciadas correctamente
- [ ] Prolijidad en todo momento!

Estimados alumnos:

Recuerden que este TP es una oportunidad para aplicar conocimientos de gestión en un contexto real. Aprovechen la experiencia para aprender no solo técnicas, sino también a trabajar en equipo y gestionar proyectos bajo presión y con recursos limitados!.