CORPORACIÓN UNIVERSITARIA DEL HUILA – CORHUILA

Programa de Ingeniería de Software

TALLER GUÍADO

Alistamiento del Proyecto – Sprint 2

Semana 2 – Sesión de trabajo en clase

Tiempo estimado: 50 minutos

Asignatura:	Sistemas Distribudos
Docente:	Jesús Ariel González Bonilla
Fecha:	11/02/2026

1. Objetivo del Taller

Preparar a los equipos de trabajo para iniciar formalmente el Sprint 2 del proyecto, estableciendo la estructura de repositorios, definiendo el alcance mediante las técnicas de Design Thinking y MoSCoW, y generando un planning de trabajo que cubra desde la Semana 2 hasta la Semana 3.

1.1. Resultados esperados al finalizar la sesión

- Repositorios creados y organizados en GitLab/GitHub.
- Alcance del proyecto definido y priorizado con MoSCoW.
- Comprensión práctica de Design Thinking aplicado al proyecto.
- Planning semanal elaborado para llegar con el Sprint 2 listo.

2. Contexto del Proyecto

Elemento	Detalle
Semana actual	Semana 2
Sprint anterior	Sprint 1 (Semana 1): Definición de la estrategia de trabajo. Sin entregables formales.
Clases	Martes (weekly) y Miércoles (proyección semanal)
Meta	Tener Sprint 2 listo para la Semana 3

3. Organización de Repositorios

Cada equipo debe crear y organizar sus repositorios siguiendo la estructura que mejor se adapte a la complejidad del proyecto. A continuación se presentan dos estrategias:

3.1. Estrategia Básica (Opcional)

Para proyectos de baja complejidad o equipos pequeños, se puede trabajar con un repositorio principal y sub-carpetas:

Repositorio	Contenido	Ejemplo de nombre
Principal (mono-repo)	Todo el proyecto unificado	proyecto-nombre
Backend (opcional)	API, lógica de negocio	proyecto-backend
Frontend (opcional)	Interfaz de usuario	proyecto-frontend
Base de datos (opcional)	Scripts SQL, migraciones	proyecto-database

3.2. Estrategia Ideal (Recomendada)

Para proyectos con arquitectura de microservicios, se recomienda crear repositorios independientes por cada microservicio:

Microservicio	Backend	Frontend	Base de datos
ms-autenticacion	ms-auth-backend	ms-auth-frontend	ms-auth-database
ms-productos	ms-prod-backend	ms-prod-frontend	ms-prod-database
ms-reportes	ms-rep-backend	ms-rep-frontend	ms-rep-database

3.3. Repositorios adicionales de soporte

Independientemente de la estrategia elegida, cada equipo debe tener repositorios para contenido académico:

Repositorio	Propósito
proyecto-contenidos	Documentación del proyecto: actas, informes, diagramas, presentaciones.
proyecto-material-apoyo	Material de referencia: guías, tutoriales, recursos externos utilizados.

⚠ *IMPORTANTE: Cada repositorio debe tener un archivo README.md con la descripción del contenido, integrantes del equipo y convenciones de commits (Conventional Commits).*

3.4. Convención de Commits

Todos los repositorios deben seguir el estándar de Conventional Commits:

Tipo	Uso	Ejemplo
feat	Nueva funcionalidad	feat: agregar login de usuario
fix	Corrección de errores	fix: corregir validación de email
docs	Documentación	docs: actualizar README
chore	Tareas de mantenimiento	chore: configurar CI/CD
refactor	Refactorización de código	refactor: extraer servicio auth

4. Design Thinking – Material de Apoyo

Design Thinking es una metodología centrada en el usuario que permite resolver problemas complejos de forma creativa e iterativa. Se compone de 5 fases que los equipos deben ejecutar durante la sesión de clase.

4.1. Las 5 Fases de Design Thinking

Fase	Nombre	Descripción	Acción en clase
1	**Empatizar**	Comprender al usuario final, sus necesidades, frustraciones y contexto.	Definir: ¿Quién usará el sistema? ¿Qué problemas tiene? Crear un mapa de empatía.
2	**Definir**	Sintetizar la información y formular el problema concreto a resolver.	Redactar el Problem Statement: "El usuario [X] necesita [Y] porque [Z]".
3	**Idear**	Generar la mayor cantidad de ideas y soluciones posibles sin juzgar.	Brainstorming: cada miembro propone mínimo 3 funcionalidades. Listar todas.
4	**Prototipar**	Construir una representación rápida de la solución para hacerla tangible.	Dibujar wireframes básicos de las pantallas principales del sistema.
5	**Testear**	Validar la solución con usuarios o pares para obtener retroalimentación.	Presentar los wireframes a otro equipo y recoger feedback.

4.2. Ejemplo Aplicado – Sistema de Gestión de Pedidos para Restaurante

Fase 1: Empatizar

Contexto: Un restaurante local recibe pedidos por teléfono y WhatsApp. El dueño olvida pedidos, se confunde con los montos y los domiciliarios no saben a qué dirección ir.

Mapa de empatía:

- Dice: "Pierdo pedidos porque anoto en papelitos".
- Piensa: "Necesito algo simple, no tengo tiempo para aprender sistemas complicados".
- Hace: Anota en cuaderno, llama a confirmar, se equivoca con direcciones.
- Siente: Frustración, estrés, miedo de perder clientes.

Fase 2: Definir

Problem Statement: "El dueño de restaurante necesita un sistema centralizado de pedidos porque actualmente pierde órdenes y comete errores que afectan la satisfacción del cliente".

Fase 3: Idear

Funcionalidades propuestas por el equipo:

1. Registro de pedidos con fecha, hora y estado.
2. Catálogo de productos con precios.
3. Asignación automática de domiciliario.
4. Notificaciones al cliente sobre el estado del pedido.
5. Reporte diario de ventas.
6. Integración con WhatsApp para recibir pedidos.

Fase 4: Prototipar

El equipo dibuja 3 pantallas principales: pantalla de nuevo pedido, lista de pedidos activos y dashboard de ventas del día.

Fase 5: Testear

Se presenta el prototipo al docente u otro equipo. Feedback recibido: "Falta una opción para pedidos recurrentes" → Se agrega a la lista de ideas.

5. MoSCoW – Material de Apoyo

MoSCoW es una técnica de priorización que clasifica los requerimientos según su importancia para el éxito del proyecto. Permite al equipo enfocarse en lo esencial y gestionar expectativas.

5.1. Categorías MoSCoW

Categoría	Significado	Criterio
Must Have	Debe tener. Sin esto, el proyecto no funciona.	Sin esta funcionalidad el producto no se puede entregar.
Should Have	Debería tener. Importante pero no crítico.	Agrega valor significativo. Se incluye si hay tiempo.
Could Have	Podría tener. Deseable, mejora la experiencia.	Nice to have. Se posterga sin impacto crítico.
Won't Have	No se hará (por ahora). Descartado para este ciclo.	Fuera del alcance actual. Se documenta para futuro.

5.2. Ejemplo Aplicado – Sistema de Gestión de Pedidos

Usando las funcionalidades identificadas en Design Thinking, el equipo las clasifica:

Prioridad	Funcionalidad	Justificación
Must Have	Registro de pedidos con fecha, hora y estado	Core del negocio. Sin esto no hay sistema.
Must Have	Catálogo de productos con precios	Necesario para crear pedidos.
Should Have	Asignación automática de domiciliario	Optimiza operaciones pero se puede hacer manual.
Should Have	Reporte diario de ventas	Valor alto para el dueño. Se puede calcular manual.
Could Have	Notificaciones al cliente sobre estado del pedido	Mejora UX pero no es indispensable.
Won't Have	Integración con WhatsApp	Compleja técnicamente. Se deja para fase futura.

⚠ *REGLA PRÁCTICA: Los Must Have no deben superar el 60% del total de funcionalidades. Si hay demasiados, se está sobreestimando la capacidad del equipo.*

6. Planning: Semana 2 – Semana 3

El siguiente planning define las actividades que cada equipo debe completar para llegar con el Sprint 2 listo. Las tareas están distribuidas entre las sesiones de clase y el trabajo autónomo.

6.1. Semana 2 – Martes (Weekly + Taller)

Duración: 50 minutos en clase

#	Actividad	Tiempo	Entregable
1	Revisión del material: Design Thinking y MoSCoW	10 min	Comprensión validada
2	Ejecutar Design Thinking (Fases 1-3: Empatizar, Definir, Idear)	20 min	Lista de funcionalidades
3	Aplicar MoSCoW sobre las funcionalidades identificadas	10 min	Tabla MoSCoW del proyecto
4	Crear repositorios y estructura básica con README.md	10 min	Repos creados con README

6.2. Semana 2 – Miércoles (Proyección)

Duración: Sesión completa

#	Actividad	Tiempo	Entregable
1	Completar Design Thinking (Fases 4-5: Prototipar y Testear)	20 min	Wireframes básicos
2	Definir alcance formal del proyecto (documento escrito)	15 min	Documento de alcance
3	Proyectar tareas de trabajo autónomo para la semana	15 min	Backlog de tareas asignadas

6.3. Trabajo Autónomo (entre Semana 2 y Semana 3)

#	Tarea	Responsable	Fecha límite
1	Configurar estructura de carpetas en cada repositorio	Líder técnico	Antes de Sem. 3

2	Subir documento de alcance y tabla MoSCoW al repo de contenidos	Analista	Antes de Sem. 3
3	Crear el modelo de datos inicial (diagrama ER) para Must Have	DBA / Backend	Antes de Sem. 3
4	Definir endpoints principales de la API (contrato)	Backend	Antes de Sem. 3
5	Crear wireframes detallados de las pantallas Must Have	Frontend / UX	Antes de Sem. 3

6.4. Semana 3 – Martes (Weekly)

Objetivo: Verificar que el Sprint 2 está completo.

1. Revisión del avance por equipo (repositorios, documentos, wireframes).
2. Validación del alcance y priorización MoSCoW.
3. Retroalimentación del docente.
4. Ajustes finales al Sprint 2.

7. Guía para Definir el Alcance del Proyecto

El documento de alcance es la base para todo el desarrollo. Debe responder las siguientes preguntas clave:

Pregunta	Respuesta esperada
¿Qué problema resuelve?	Descripción clara del problema identificado en Design Thinking.
¿Quién lo usará?	Usuarios finales identificados en la fase de Empatizar.
¿Qué funcionalidades tendrá?	Lista priorizada con MoSCoW (solo Must Have y Should Have para Sprint 2).
¿Qué NO incluye?	Won't Have explícitos: funcionalidades descartadas y razón.
¿Qué tecnologías se usarán?	Stack técnico: lenguajes, frameworks, base de datos, despliegue.
¿Cuál es la arquitectura?	Monolítica vs Microservicios. Justificar la elección.

8. Actividad en Clase (50 minutos)

Cada equipo debe completar las siguientes tareas durante la sesión. El docente estará disponible para resolver dudas y validar avances.

Bloque A – Revisión de material (10 min)

1. Leer las secciones 4 y 5 de este documento (Design Thinking y MoSCoW).
2. Resolver dudas con el docente.

Bloque B – Design Thinking aplicado (20 min)

1. Ejecutar las fases 1-3 de Design Thinking para su proyecto.
2. Documentar el mapa de empatía, el Problem Statement y la lista de funcionalidades.
3. Registrar todo en un documento dentro del repositorio de contenidos.

Bloque C – Priorización MoSCoW (10 min)

- Tomar la lista de funcionalidades generada en el bloque B.
- Clasificar cada funcionalidad en Must / Should / Could / Won't.
- Crear la tabla MoSCoW del proyecto.

Bloque D – Repositorios (10 min)

- Crear los repositorios necesarios según la sección 3.
- Agregar README.md con nombre del proyecto, integrantes y convención de commits.
- Subir la tabla MoSCoW y el Problem Statement como primer commit.

> ⚠ *ENTREGA: Al finalizar la clase, cada equipo debe tener los repositorios creados con al menos un commit que incluya el README.md y la documentación inicial del proyecto.*