

Programación distribuida del lado servidor

Modelo Pedagógico a través de Taller Invertido

- **Actividad pre-clase (estudiante):** Antes de cada sesión, el estudiante deberá revisar de forma autónoma los recursos teóricos correspondientes (lecturas del libro de texto, videos, tutoriales, documentación oficial). Se espera que lleguen al taller con la base conceptual necesaria.
- **Actividad durante la clase (docente y estudiantes):** Las 4 horas de sesión se dedicarán casi exclusivamente a la implementación, resolución de problemas y trabajo en el proyecto. El rol docente será el de un facilitador experto, guiando la práctica, resolviendo dudas complejas y realizando revisiones de código.

Herramientas sugeridas para el taller

- **Control de Versiones:** Git (fundamental). Cada hito del proyecto debe ser una entrega en un repositorio.
- **Lenguajes y Frameworks:**
 - **Principal:** C# con [ASP.NET](#) Core para la creación de APIs y servicios.
 - **Alternativo:** PHP con un framework como Laravel o Symfony si algún equipo lo prefiere.
 - **Secundario:** Python para crear scripts de prueba, workers o tareas de automatización.
- **Pruebas de API:** Postman o Insomnia.
- **Contenerización:** Docker y Docker Compose.

Plan de trabajo (15 Semanas)

Fase 1: Del monolito a los servicios básicos (hito 1 del proyecto)

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
1	Unidad I: Fundamentos de	Leer sobre diferencias entre sistemas centralizados y distribuidos. Repasar	Taller 1: Implementación de una aplicación monolítica simple (ej. un sistema	

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
	Sistemas Distribuidos	conceptos de Sockets, HTTP y APIs REST.	de TODOs o un blog básico) con una API REST en C#/ASP.NET Core. Definición del alcance del Proyecto Integrador.	
2	Unidad I: Comunicación y Tolerancia a Fallos	Estudiar patrones de comunicación síncrona vs. asíncrona. Investigar sobre timeouts, reintentos y el patrón Circuit Breaker.	Taller 2: Separar la lógica de la base de datos del monolito en un servicio independiente. Implementar comunicación directa (HTTP síncrona) entre el servicio principal y el de datos.	Avance del Proyecto.
3	Unidad II: Procesamiento Asíncrono (Parte 1)	Investigar qué son las colas de mensajes (Message Queues) y el patrón Productor/Consumidor. Ver tutoriales sobre RabbitMQ o Azure Service Bus.	Taller 3: Introducir una cola de mensajes para manejar tareas asíncronas (ej. envío de notificaciones). Refactorizar la comunicación para que sea asíncrona usando la cola.	
4	Unidad II: Arquitecturas Orientadas a Servicios (Parte 2)	Estudiar el patrón Publicador/Suscriptor (Pub/Sub) y las arquitecturas basadas en eventos.	Taller 4: Implementar un sistema de eventos. Un servicio publica un evento (ej. "UsuarioCreado") y otro servicio se suscribe para reaccionar a él.	Entrega Hito 1 - Sistema Distribuido Básico (15%).
5	Evaluación y Retroalimentación	Preparación para el examen práctico. Revisión de los conceptos de las Unidades I y II.	Examen Práctico 1 - Comunicación Distribuida (12%). Sesión de resolución de problemas y	Calificación del Examen 1.

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
			retroalimentación del Hito 1.	

Fase 2: Escalabilidad, seguridad y resiliencia (hito 2 del proyecto)

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
6	Unidad III: Escalabilidad Horizontal (Parte 1)	Leer sobre balanceo de carga (Load Balancing) y sus algoritmos (Round Robin, Least Connections).	Taller 5: Introducir un balanceador de carga (ej. Nginx) para distribuir el tráfico entre múltiples instancias de un servicio.	
7	Unidad III: Distribución de Datos (Parte 2)	Estudiar conceptos de replicación de bases de datos (primario-secundario) y particionamiento (sharding).	Taller 6: Simular un escenario de replicación de datos. Discutir y diseñar una estrategia de particionamiento para el proyecto.	Avance del Proyecto.
8	Unidad IV: Seguridad en Sistemas Distribuidos (Parte 1)	Estudiar los fundamentos de JWT (JSON Web Tokens) para autenticación y autorización en APIs.	Taller 7: Implementar autenticación basada en JWT. Asegurar los endpoints de los servicios para que solo acepten peticiones con un token válido.	
9	Unidad IV: Comunicación Segura (Parte 2)	Investigar sobre gestión de secretos (variables de entorno, vaults) y comunicación cifrada (HTTPS/TLS).	Taller 8: Configurar la comunicación entre servicios para que use HTTPS. Externalizar secretos y credenciales de la base de código.	Entrega Hito 2 - Servicios con Seguridad (15%).
10	Evaluación y Retroalimentación	Preparación para el examen práctico sobre	Examen Práctico 2 - Arquitectura Escalable	Calificación del Examen 2.

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
		escalabilidad y seguridad.	(12%). Sesión de retroalimentación sobre el Hito 2 y planificación para la fase final.	

Fase 3: Operaciones, monitoreo y modernización (hito 3 del proyecto)

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
11	Unidad V: Contenedorización (Parte 1)	Realizar tutoriales básicos de Docker: crear Dockerfiles y usar Docker Compose para orquestar múltiples contenedores.	Taller 9: “Dockerizar” cada uno de los microservicios del proyecto. Crear un archivo docker-compose.yml para levantar todo el ecosistema con un solo comando.	
12	Unidad V: Monitoreo y Observabilidad (Parte 2)	Investigar sobre las tres “columnas” de la observabilidad: logs, métricas y trazas. Ver herramientas como Prometheus, Grafana y Jaeger.	Taller 10: Implementar logging centralizado (ej. usando el stack ELK/EFK o una solución cloud). Añadir métricas básicas de rendimiento a los servicios (ej. tiempo de respuesta, tasa de errores).	Avance del Proyecto.
13	Unidad VI: Resiliencia y Pruebas	Estudiar conceptos de Chaos Engineering. Leer sobre cómo realizar pruebas de resiliencia y de carga.	Taller 11: Realizar pruebas de resiliencia: simular la caída de un servicio y verificar que el sistema se recupera o degrada controladamente.	Entrega Hito 3 - Sistema Completo (10%).

Semana	Unidad y Temas	Actividades Pre-clase (Estudio Independiente)	Actividades en Taller (4 horas)	Entregables y Evaluación
14	Evaluación y Documentación	Preparación para el examen final. Trabajar en la documentación técnica del proyecto.	Examen Práctico 3 - Sistemas Operables (11%). Sesión de trabajo final en la documentación del proyecto y la presentación.	Calificación del Examen 3 y Entrega Documentación y Código (10%).
15	Presentación Final del Proyecto	Preparar una presentación técnica que justifique las decisiones arquitectónicas tomadas, muestre el sistema funcionando y discuta los aprendizajes.	Presentaciones Finales: Cada equipo presenta su proyecto integrador. Sesión de Q&A, evaluación final y cierre del curso.	Evaluación del Proyecto Integrador (total 40%).

Sobre los módulos optativos de especialización

Dado el enfoque práctico y el tiempo limitado, los temas de los módulos optativos se presentan como “píldoras de conocimiento avanzado” o “próximos pasos”. Se dedican los últimos 15-20 minutos de talleres específicos para introducir un tema y motivar a los estudiantes a explorarlo:

- Al hablar de tolerancia a fallos (Semana 2), mencionar los algoritmos de consenso como **Raft** (Módulo A).
- Al implementar Pub/Sub (Semana 4), introducir **Event Sourcing y CQRS** (Módulo B).
- Al hablar de monitoreo (Semana 12), hablar de **SRE y Chaos Engineering** (Módulo E).