



Práctica en Clases #2

Arquitectura del Dominio con Métodos Asíncronos y Patrones de Programación

Carrera Software	Docente John Cevallos	Nivel Quinto
Período Lectivo 2025-2026(2)	Asignatura Aplicaciones para el Servidor Web	Práctica # 2 (2 horas)

OBJETIVO DE LA PRÁCTICA COMPLEMENTARIA

Implementar una arquitectura robusta y escalable para el dominio del emprendimiento, aplicando patrones de diseño, principios de código limpio y diferentes paradigmas de programación asíncrona en JavaScript/TypeScript para operaciones CRUD.

Los estudiantes lograrán:

- Diseñar e implementar interfaces y clases que representen fielmente el dominio del emprendimiento
- Crear una arquitectura en capas que separe responsabilidades y facilite el mantenimiento
- Aplicar diferentes patrones asíncronos: Callbacks, Promises y Async/Await en operaciones CRUD
- Implementar repositorios de datos en memoria con información de prueba optimizada
- Estructurar el proyecto siguiendo principios de código limpio y arquitectura hexagonal
- Desarrollar un sistema de pruebas funcional desde un archivo main centralizado
- Aplicar principios SOLID y patrones de diseño en el desarrollo del backend

MATERIALES Y TECNOLOGÍAS

Software Requerido:

- Node.js: Versión 18 o superior para soporte completo de ES modules
- TypeScript: Para tipado estático y mejor experiencia de desarrollo
- Editor de Código: Visual Studio Code con extensiones de TypeScript
- Git: Para control de versiones y trabajo colaborativo
- Terminal/CLI: Para ejecutar comandos de desarrollo

Stack Tecnológico:

- TypeScript: Lenguaje principal con tipado estático
- UUID: Generación de identificadores únicos
- ts-node: Ejecución directa de archivos TypeScript
- Chalk (opcional): Colorización de consola para mejor UX

Patrones y Conceptos Aplicados:

- Arquitectura en Capas: Domain, Infrastructure, Application, Presentation
- Patrón Repository: Abstracción de acceso a datos
- Principios SOLID: Especialmente Single Responsibility y Dependency Inversion
- Inyección de Dependencias: Para desacoplamiento de componentes
- Domain-Driven Design: Enfoque en el dominio del negocio

PARADIGMAS ASÍNCRONOS REQUERIDOS

 Distribución Obligatoria por Operación CRUD:

CALLBACKS

Operación: INSERCIÓN (CREATE)

- Función callback con patrón (error, resultado)
- Manejo de errores en el primer parámetro
- Simulación de latencia de red
- Validación de datos antes de insertar

PROMISES

Operación: MODIFICACIÓN (UPDATE)

- Retorno de Promise con resolve/reject
- Encadenamiento con .then() y .catch()
- Validación de existencia del registro
- Actualización parcial de campos

ASYNC/AWAIT

Operación: CONSULTA (READ)

- Funciones async que retornan Promise
- Uso de await para operaciones asíncronas
- Manejo de errores con try/catch
- Consultas individuales y listados completos

ASYNC/AWAIT

Operación: ELIMINACIÓN (DELETE)

- Validación de existencia antes de eliminar
- Retorno de boolean indicando éxito/fallo
- Eliminación lógica o física según el caso
- Manejo elegante de errores

INSTRUCCIONES GENERALES

 **Modalidad de Trabajo:** Cada integrante implementa el dominio completo de una entidad con todos los paradigmas asíncronos requeridos.

 **Distribución de Responsabilidades:** Cada estudiante desarrolla interfaces, modelos, repositorios, servicios y pruebas de su entidad.

 **Entrega:** Cada estudiante sube el proyecto completo integrado en su repositorio personal de prácticas.

PROCEDIMIENTO DETALLADO

 **Paso 1: Configuración del Proyecto Base**
Objetivo: Establecer estructura y entorno de desarrollo.

 **Paso 2: Diseño del Dominio**
Objetivo: Crear interfaces, repositorios, entidades y objetos de valor.

Paso 3: Implementación de Repositorios

Objetivo: CRUD en memoria con paradigmas asíncronos.

Paso 4: Servicios de Aplicación

Objetivo: Orquestar operaciones del dominio.

Paso 5: Archivo Main de Pruebas

Objetivo: Probar consultas, inserciones, modificaciones y eliminaciones.

Paso 6: Integración y Pruebas Grupales

Objetivo: Consolidar el proyecto y documentarlo.

PARÁMETROS DE CUMPLIMIENTO

Criterio	Descripción	Puntos
Arquitectura y Estructura	Organización en capas, separación de responsabilidades, aplicación de SOLID.	20 pts
Implementación del Dominio	Interfaces bien definidas, entidades con lógica de negocio y validaciones.	20 pts
Paradigmas Asíncronos	Uso correcto de callbacks, promises y async/await según lo requerido.	20 pts
Datos de Prueba y Repositorios	Calidad de datos de prueba y repositorios en memoria correctamente implementados.	15 pts
Archivo Main y Pruebas	Sistema de pruebas funcional, logs claros y manejo adecuado de errores.	10 pts
Trabajo Colaborativo y Comunicación	Distribución equitativa, commits descriptivos, documentación clara.	15 pts
TOTAL		100 pts

Requisitos Mínimos Obligatorios:

- Mínimo 4 capas (Domain, Infrastructure, Application, Presentation)
- Al menos 1 entidad completa por integrante
- Implementación correcta de callbacks, promises y async/await
- Mínimo 10 registros realistas por entidad
- Manejo de errores y validaciones implementados
- Pruebas funcionales de todas las operaciones CRUD
- Uso correcto de TypeScript

- README.md completo con instrucciones y evidencias

ENTREGABLE

 No se entregará PDF. La práctica se califica directamente desde el repositorio personal.

El archivo README.md debe incluir:

- Título del Proyecto
- Integrantes y Contribuciones
- Arquitectura del Sistema
- Instrucciones de Instalación
- Instrucciones de Ejecución
- Documentación de APIs
- Paradigmas Implementados
- Evidencias de Funcionamiento
- Conclusiones Individuales

CONCLUSIÓN

Esta práctica introduce a los estudiantes en conceptos fundamentales de arquitectura de software para aplicaciones backend. Se aplican paradigmas asíncronos y separación en capas para bases sólidas.

El uso de callbacks, promises y async/await prepara a los estudiantes para APIs reales, bases de datos y servicios web. La arquitectura será base para futuras iteraciones con persistencia, autenticación y REST APIs.