Información del proyecto

Datos Académicos

Universidad: TecMilenio

Materia: Estructuras de Datos

Profesora: Blanca Aracely Aranda Machorro

Ubicación: Monterrey, Nuevo León

TechSolutions S.A. de C.V



# **Caso de Estudio: TechSolutions S.A. de C.V.**

La empresa, ubicada en Monterrey, tiene varios departamentos (Desarrollo, Ventas, Soporte, Administración).  
**Problemas destacados**

* Los empleados anotaban tareas en hojas de cálculo dispersas.
* Las urgentes quedaban sin atender a tiempo.
* No había control claro de dependencias.
* La administración tardaba en ver el avance real de los equipos.

**Efecto:**  
 Desorganización, pérdida de productividad y retrasos en entregas a clientes.

Problemática identificada

TechSolutions S.A. de C.V. enfrenta dificultades al gestionar tareas en varios departamentos:

* Falta de priorización clara entre tareas urgentes y programadas.
* Dependencias poco visibles entre tareas
* Retrasos por búsquedas lentas de tareas en listas generales.
* Escasa trazabilidad de empleados y su carga de trabajo.

Solución

Hicimos un sistema de gestión empresarial el cual es una aplicación de consola en Java que combina la gestión de tareas departamentales con funcionalidades empresariales avanzadas como gestión de clientes, inventario, ventas, reportes y usuarios con roles.

Tecnologías Seleccionadas

Java 17+ → Lenguaje principal de desarrollo.

Maven → Gestión de dependencias y compilación.

Swing → Interfaz gráfica

MongoDB → Base de datos para persistencia de tareas y empleados.

Estructuras de Datos → Stack, LinkedList, PriorityQueue, HashMap

Patrones de Diseño → MVC, DAO, Singleton.

Alcance del Sistema

* Registro y gestión de tareas (ID, descripción, departamento, urgencia, fecha límite).
* Clasificación de tareas con estructuras de datos:

Pila: tareas urgentes.

Cola: tareas programadas

Lista: tareas por departamento.

Cola de prioridad: por prioridad y fecha.

* Persistencia en MongoDB (cargar, guardar, actualizar y eliminar datos).
* Reportes básicos: conteos, tiempos estimados y estadísticas simples.

Roles del Sistema

1. Empleado

Permisos:

* Registrar y consultar sus propias tareas.
* Visualizar tareas de su departamento.
* Restricciones:
* No puede modificar tareas de otros.
* No gestiona empleados ni dependencias.

2. Líder de Departamento

Permisos:

* Consultar todas las tareas de su área.
* Asignar prioridades dentro de su departamento.
* Generar reportes de productividad del equipo.
* Restricciones:
* No accede a tareas de otros departamentos.
* No crea ni elimina empleados.

3. Administrador

Permisos:

* CRUD completo de tareas y empleados.
* Gestionar dependencias
* Acceso a reportes globales y estadísticas.
* Validar y configurar la conexión a la base de datos.
* Restricciones:
* Requiere autenticación especial (contraseña).

4. Cliente VIP

Permisos:

* Acceder a sus pedidos o tareas asociadas con prioridad especial.
* Tener seguimiento diferenciado en reportes.
* Restricciones:
* No gestiona el sistema directamente, solo recibe trato prioritario en las operaciones relacionadas.

# **Beneficios**

* Priorización justa y automática de tareas.
* Vista clara de dependencias y jerarquía de empleados.
* Rapidez en búsqueda y actualización.
* Persistencia de datos con MongoDB.
* Mejora en productividad y trazabilidad de proyectos.

src/

├─ main/java/

│ ├─ controller/

│ │ └─ SistemaGestionTareas.java # Controlador principal

│ │

│ ├─ dao/

│ │ ├─ MongoConnection.java # Conexión a MongoDB (Singleton)

│ │ ├─ TareaDAO.java # Acceso a datos de tareas

│ │ └─ EmpleadoDAO.java # Acceso a datos de empleados

│ │

│ ├─ model/

│ │ ├─ Tarea.java # Modelo de tarea

│ │ ├─ TareaPrioridad.java # Modelo de tarea con prioridad

│ │ ├─ Empleado.java # Modelo de empleado

│ │ ├─ NodoEmpleado.java # Nodo del árbol binario

│ │ └─ DependenciasGrafo.java # Representación de dependencias

│ │

│ ├─ estructuras/

│ │ ├─ ArbolEmpleados.java # Lógica de árbol binario

│ │ └─ EstructurasAuxiliares.java # Stack, Cola, Lista

│ │

│ ├─ gui/

│ │ ├─ MainFrame.java # Ventana principal (JFrame + JTabbedPane)

│ │ ├─ TabUrgentes.java # Pestaña: pila de urgentes

│ │ ├─ TabProgramadas.java # Pestaña: cola de programadas

│ │ ├─ TabDepartamento.java # Pestaña: lista por departamento

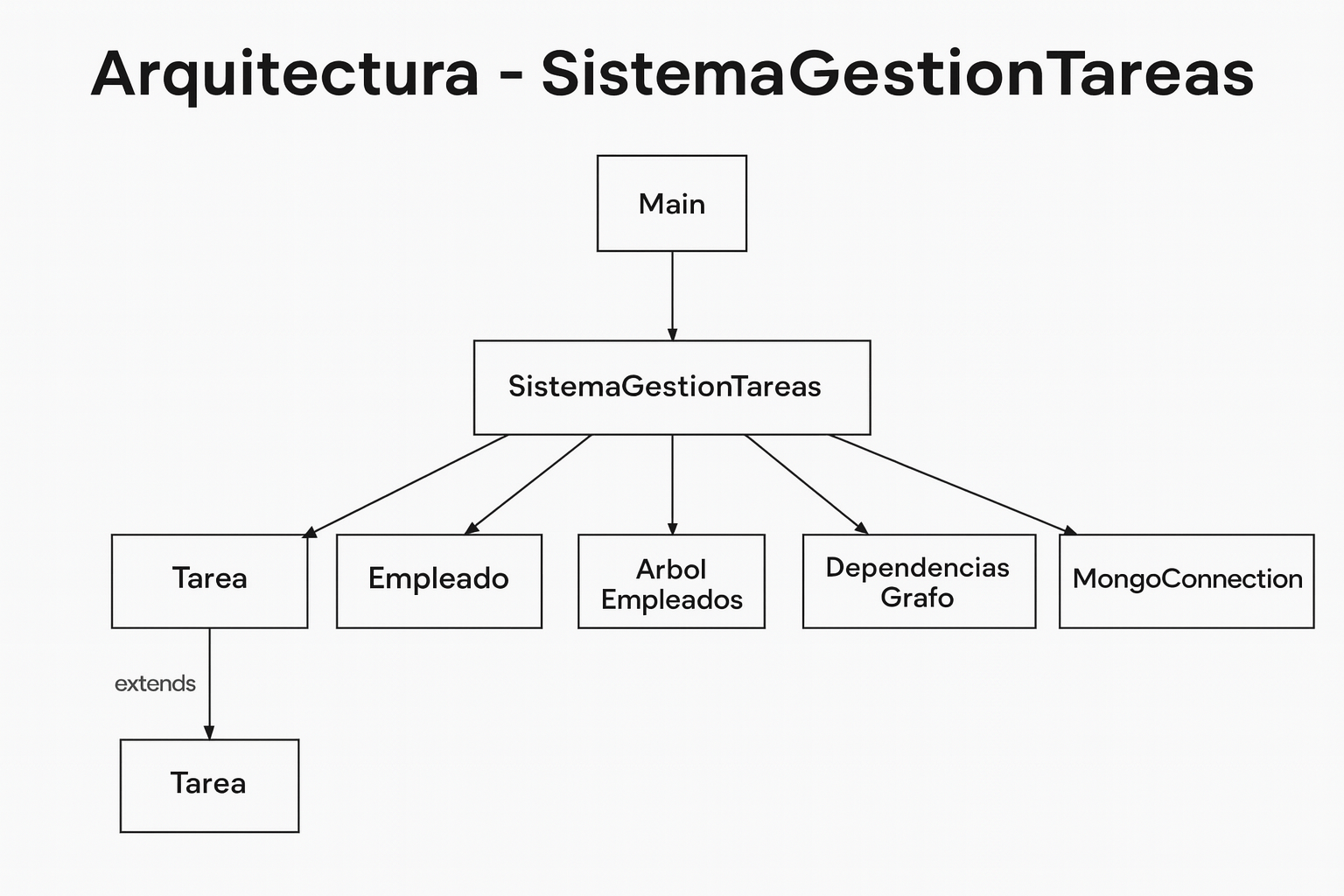
│ │ ├─ TabPrioridad.java # Pestaña: cola de prioridad

│ │ └─ TabEmpleados.java # Pestaña: empleados (árbol)

│ │

│ └─ Main.java # Punto de entrada

Diagrama UML



Estructuras de Datos Aplicadas

Cola de Prioridad

* Propósito: Ordenar tareas con prioridad alta, media o baja considerando la fecha de entrega.
* Complejidad: Insert O(log n), Poll (siguiente) O(log n), Peek O(1).

Lista

* Propósito: Mostrar todas las tareas por departamento (Ventas, Soporte, Desarrollo, etc.).
* Complejidad: Recorrido O(n), acceso por índice O(1).

HashMap

* Propósito: Búsqueda rápida de tareas por su ID único.
* Complejidad: O(1) promedio en búsqueda e inserción.

Grafo de Dependencias

* Propósito: Gestionar dependencias entre tareas (una depende de otra).
* Complejidad: Recorrido O(V+E), detección de ciclos con DFS.

# Requisitos Funcionales

* **RF01:** Registrar tareas con ID, descripción, prioridad, departamento y fecha límite.
* **RF02:** Gestionar tareas urgentes con pila.
* **RF03:** Administrar tareas programadas con cola.
* **RF04:** Visualizar tareas por departamento (lista).
* **RF05:** Mantener cola de prioridad con urgencia y fecha.
* **RF06:** Buscar tarea por ID con HashMap.
* **RF07:** Administrar empleados en un árbol binario.
* **RF08:** Registrar dependencias entre tareas con grafo.
* **RF09:** Persistir datos en MongoDB

Objetivos de Aprendizaje Alcanzados

Entorno listo y compilación OK: Instalé JDK, puse Apache Maven en el PATH y compilé el proyecto con BUILD SUCCESS.

Persistencia con MongoDB: Conecté a mongodb://localhost:27017, creé la BD techsolutions y realicé CRUD desde Java (colecciones tareas y empleados).

POO + Estructuras y algoritmos: Modelé Tarea, Empleado y la herencia TareaPrioridad; utilicé Stack, Queue, PriorityQueue, HashMap, Árbol para empleados y Grafo para dependencias (orden topológico / detección de ciclos).

Patrón DAO bien aplicado: Separé la lógica de datos en TareaDAO y EmpleadoDAO usando MongoConnection, desacoplando UI y persistencia.

UI funcional con Swing: Implementé formularios/tablas para alta, consulta y eliminación; los cambios se reflejan en la base de datos.

Diseño y verificación: Elaboré diagrama UML de clases y validé el flujo completo UI → BD → UI con evidencias en Compass

Glosario

Pila (Stack): Estructura de datos que organiza elementos bajo la política LIFO (Last In, First Out).

Cola (Queue): Estructura de datos basada en FIFO (First In, First Out), donde el primero en entrar es el primero en salir.

Cola de Prioridad (PriorityQueue): Variante de cola en la que los elementos se procesan en orden de prioridad, no necesariamente por orden de llegada.

Lista (List): Colección ordenada de elementos que permite acceso por índice.

HashMap: Estructura de datos que almacena pares clave-valor y permite búsquedas rápidas en O(1) promedio.

Árbol Binario: Estructura jerárquica de nodos donde cada nodo tiene como máximo dos hijos (izquierdo y derecho).

Grafo: Conjunto de nodos y aristas que representan relaciones entre elementos.

DAO (Data Access Object): Patrón de diseño que separa la lógica de acceso a datos de la lógica de negocio.

MVC (Modelo-Vista-Controlador): Patrón arquitectónico que divide un sistema en tres capas para facilitar mantenimiento y escalabilidad.

MongoDB: Base de datos NoSQL orientada a documentos, utilizada en este sistema para almacenar tareas y empleados.

Swing: Librería de Java para el desarrollo de interfaces gráficas de usuario.

CRUD (Create, Read, Update, Delete): Operaciones básicas para manipulación de datos en cualquier sistema.

Bibliografía

Oracle. (2023). Java Platform, Standard Edition Documentation. Recuperado de https://docs.oracle.com/javase/

MongoDB Inc. (2023). MongoDB Manual. Recuperado de https://www.mongodb.com/docs/

Weiss, M. A. (2014). Data Structures and Algorithm Analysis in Java (3rd ed.). Pearson.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.

Deitel, P. & Deitel, H. (2017). Java: How to Program (Early Objects) (11th ed.). Pearson.

Knuth, D. E. (1997). The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley.

# Autores

Jesus David Marroquin Peña AL07101520



Mauricio sanchez Figueroa AL07098275



Juan Porfirio Torres Rojas AL07099471



Ian Carlos Martinez Diaz AL03003400



Conclusión

Logramos construir un sistema de gestión de tareas funcional que integra una UI de Java Swing con persistencia en MongoDB mediante el patrón DAO, gestionado con Apache Maven. El proyecto demuestra dominio de POO, uso de estructuras de datos (Stack, Queue, PriorityQueue, HashMap, árbol binario y grafo de dependencias) y operaciones CRUD reales sobre la base techsolutions (tareas, empleados). La arquitectura modular y los diagramas UML garantizan claridad, y la verificación en MongoDB Compass confirma la correcta interacción UI ↔ BD. En síntesis, cumplí los objetivos clave del curso: modelar, persistir, ejecutar y documentar un sistema completo y reproducible.

Límites y siguientes pasos (breve): fortalecer validaciones y manejo de errores, añadir autenticación/roles, crear pruebas unitarias e integración (JUnit/Testcontainers), definir índices únicos en Mongo, y preparar despliegue con Docker o MongoDB Atlas; opcionalmente exponer un API REST (Spring Boot) y automatizar con CI/CD.