

DOCENTE:

RUBÉN DARÍO GONZÁLEZ BARRERA

ASIGNATURA:

ESTRUCTURA DE DATOS

ESTUDIANTES:

DEIBID BENAVIDES YAYA

MATEO RAMIREZ ORTIZ

MATEO ALEJANDRO VARGAS VALERO

NRC: 80986

INSTITUCIÓN: UNIVERSIDAD MINUTO DE DIOS

FECHA DE ENTREGA: /11/15/2025

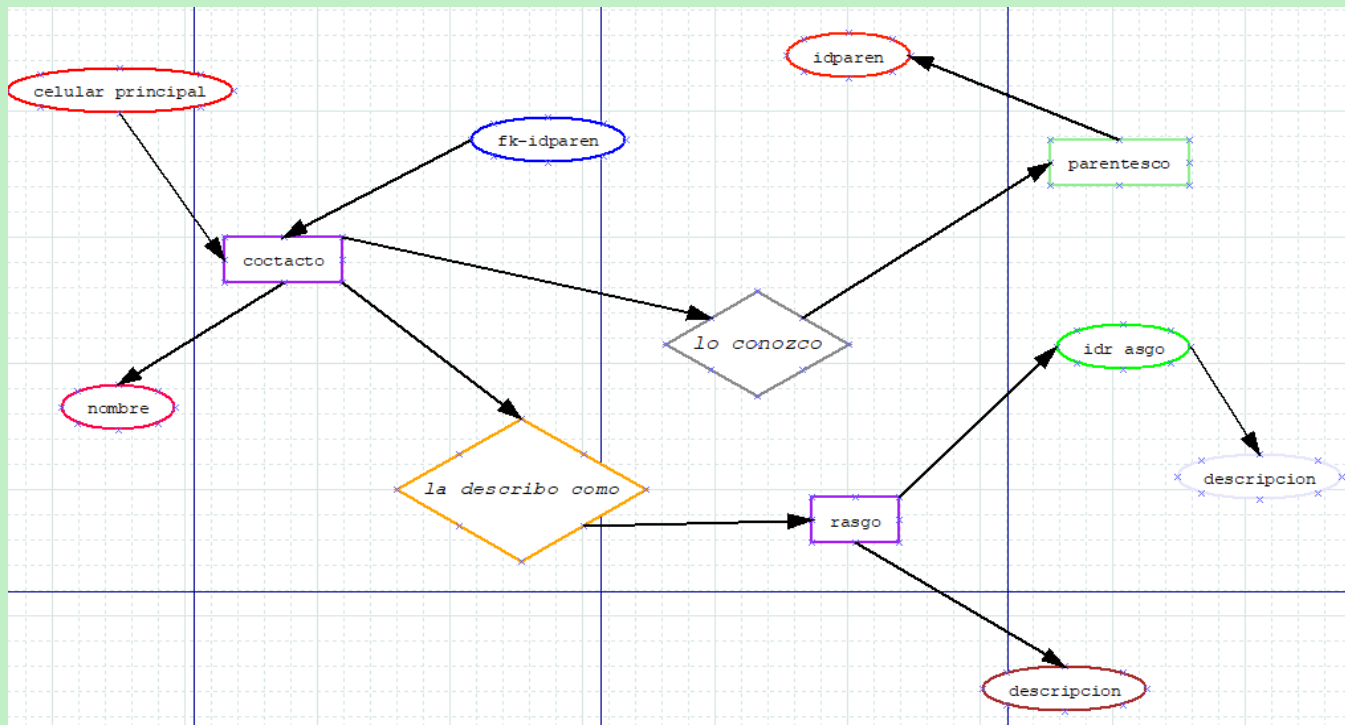
1.Crear el repositorio del proyecto (GitHub o GitLab).

<https://github.com/varagasmateo0-lgtm/mateommm.git>

2. Dividir tareas entre los integrantes del grupo.

NOMBRE DE LA ENTIDAD DONDE RELIZARA EL PROYECTO			
c	ACTIVIDADES	RESPONSABLE	PRODUCTO
ANALISIS	Definición del Alcance y Objetivos	Líder de Proyecto	Documento , Alcance
	Análisis de Contenido Curricular	Expertos Temáticos	Requisitos Funcionales
	Especificación de Requisitos Técnico	Arquitecto del el proyecto	Documento de Especificación
	Diseño Conceptual de la Interfaz (Mockups	Diseñador	Prototipo de Interfaz
	Aprobación del Documento de Análisis	diseñador del ciodigo	Documento Firmado
PLANEACION	Creación del Plan de Trabajo Detallado	Líder de Proyecto	Diagrama de Gantt
	Asignación de Recursos y Roles	Líder de Proyecto	Matriz de
	Diseño de la agenda de contactos	Diseñador Arquitecto	Estructura Base
	Definición de Estándares de Codificación	Diseñador de código	Documento
	Plan de Pruebas del codigo	Tester	Documento de Plan de
EJECUCION	Desarrollo del Módulo de Listas Enlazadas	Programador Senior	Código del Módulo
	Desarrollo del Módulo de programa y avances	Programador	Código del Módulo
	Integración de la Interfaz de Usuario	ProgramadorDiseñad	Interfaz Funcional
	Integración de la Interfaz de Usuario	Programador / Tester	Versión Beta del
	Lanzamiento en Entorno de Pruebas (UAT)	Tester / Líder de	Aprobación
		Tester Líder Proyecto	

3. Programar los módulos principales paso a paso.



4. Aplicar pruebas unitarias y depurar errores.

agregar Contacto (nombre, teléfono, email): Debe verificar que el contacto se añade correctamente a la lista.

eliminar Contacto(id): Debe verificar que el contacto con el ID especificado se retira de la lista y que se manejan los casos donde el ID no existe.

buscar Contacto(criterio): Debe verificar que se devuelve la lista correcta de contactos que coinciden con el criterio de búsqueda.

actualizar Contacto (id, nuevos Datos): Debe verificar que los datos del contacto se modifican correctamente.

obtener Todos (): Debe verificar que devuelve la lista completa de contactos.

Clase Contacto / Modelo de Datos: Debe verificar que los objetos de contacto se crean con los tipos de datos correctos (ej: el teléfono es un número, el email es válido).

Funcionalidad	Caso de Prueba	Resultado Esperado
agregar Contacto	Agregar un contacto válido.	El tamaño de la lista de contactos aumenta en 1, y el contacto existe en la lista.
agregar Contacto	Intentar agregar un contacto sin nombre (o campo requerido).	Se lanza un error (o la operación es rechazada).
eliminar Contacto	Eliminar un contacto existente por su ID.	El tamaño de la lista disminuye en 1, y el contacto ya no está en la lista.
eliminar Contacto	Intentar eliminar un contacto con un ID que no existe.	La lista permanece inalterada y se devuelve una indicación de fallo.
buscar Contacto	Buscar por una parte del nombre (ej: "Juan").	Se devuelve una lista que contiene solo a los contactos con "Juan" en su nombre.
actualizar Contacto	Actualizar el teléfono de un contacto.	

5. Documentar el código y avances.

```
1  const fs = require('fs');
2  const readline = require('readline');
3  const path = require('path');
4
5  const inputFile = path.join(__dirname, 'sales.csv');
6  const outputFile = path.join(__dirname, 'totals.json');
7
8  const salesTotals = {};
9
10 const fileStream = fs.createReadStream(inputFile);
11
12
13 fileStream.on('error', (err) => {
14   console.error('Error al leer el archivo:', err.message);
15   process.exit(1);
16 });
17
18
19 const rl = readline.createInterface({
20   input: fileStream,
21   crlfDelay: Infinity
22 });
23
24 console.log(`Procesando archivo: ${inputFile}`);
25
26
27 rl.on('line', (line) => {
28
29   if (!line.trim()) return;
30
31   // Parsear CSV: fecha, producto, cantidad, precio
32   const parts = line.split(',');
33
34   if (parts.length !== 4) {
35     console.warn('⚠ Línea ignorada por formato inválido: ${line}`);
36     return
37   }
38
39   const [fecha, producto, cantidadStr, precioStr] = parts;
40
41   try {
42     const cantidad = parseFloat(cantidadStr);
43     const precio = parseFloat(precioStr);
44
45     if (isNaN(cantidad) || isNaN(precio) || cantidad <= 0 || precio <= 0) {
46       console.warn('Valores numéricos inválidos ignorados en línea: ${line}`);
47       return;
48     }
49
50     // Actualizar totales
51     if (!salesTotals[producto]) {
52       salesTotals[producto] = {
53         fecha: fecha,
54         cantidad: cantidad,
55         precio: precio
56       };
57     } else {
58       salesTotals[producto].cantidad += cantidad;
59       salesTotals[producto].precio += precio;
60     }
61   }
62
63   // Leer la siguiente línea
64   rl.resume();
65   rl.next();
66 });
67
68 // Cerrar la interfaz de lectura
69 rl.close();
70
71 // Guardar los totales en un archivo JSON
72 const json = JSON.stringify(salesTotals, null, 2);
73 fs.writeFileSync(outputFile, json);
74
75 console.log(`Totales guardados en: ${outputFile}`);
76
77 // Fin del programa
78 process.exit(0);
```

```

console.log(` Procesando archivo: ${inputFile}`);

rl.on('line', (line) => {

    if (!line.trim()) return;

    // Parsear CSV: fecha,producto,cantidad,precio
    const parts = line.split(',');

    if (parts.length !== 4) {
        console.warn(` ⚠ Línea ignorada por formato inválido: ${line}`);
        return;
    }

    const [fecha, producto, cantidadStr, precioStr] = parts;

    try {
        const cantidad = parseFloat(cantidadStr);
        const precio = parseFloat(precioStr);

        if (isNaN(cantidad) || isNaN(precio) || cantidad <= 0 || precio <= 0) {
            console.warn(` Valores numéricos inválidos ignorados en línea: ${line}`);
            return;
        }

        const ingreso = cantidad * precio;

        salesTotals[producto] = (salesTotals[producto] || 0) + ingreso;

    } catch (error) {
        console.error(`Error al parsear o calcular línea: ${line}`, error.message);
    }
});

rl.on('close', () => {
    console.log('---');
    console.log(' Lectura de CSV finalizada. Totales calculados. ');
    console.log('Totales:', salesTotals);

    try {
        const jsonOutput = JSON.stringify(salesTotals, null, 2);

        fs.writeFile(outputFile, jsonOutput, 'utf8', (err) => {
            if (err) {
                console.error('Error al escribir archivo:', err);
            }
        });
    } catch (error) {
        console.error('Error al serializar o escribir archivo:', error.message);
    }
});

```

Java: Ready

```

283 child.on('close', (code) => {
284     console.log(`\nProceso hijo finalizado con código: ${code}`);
285
286     if (code !== 0) {
287         console.error(' El proceso hijo terminó con un código de error. ');
288         return;
289     }
290
291
292     try {
293         const result = JSON.parse(childOutput.trim());
294
295         if (result.status === 'success') {
296             console.log(` Factorial de ${result.number} es: `);
297
298             const displayResult = result.result.length > 80
299                 ? `${result.result.slice(0, 40)}...${result.result.slice(-40)}`
300                 : result.result;
301             console.log(` --> ${displayResult}`);
302         } else if (result.status === 'error') {
303             console.error(' El proceso hijo reportó un error: ${result.message} ');
304         } else {
305             console.error(' Respuesta del hijo con formato inválido. ');
306         }
307     } catch (e) {
308         console.error(' Error al parsear la salida JSON del hijo: ${e.message} ');
309         console.log('Salida cruda:', childOutput);
310     }
311 });
312
313
314 child.stdin.write(String(numberToCalculate) + '\n');
315 child.stdin.end();
316
317 class Contacto:
318     """Clase que define la estructura de un contacto."""
319     def __init__(self, id, nombre, telefono, email):
320         self.id = id
321         self.nombre = nombre
322         self.telefono = telefono
323         self.email = email
324
325     def __repr__(self):
326         # Representación legible para la depuración
327         return f"ID: {self.id}, Nombre: {self.nombre}, Teléfono: {self.telefono}, Email: {self.email}"
328
329

```

Fase de Costos:

C	D	E	F	
recursos	cantidad	Costo Unitario (COP)	Costo Total	
Energía eléctrica	1 mes	40.000	40.000	
Internet	2 mes	50.000	50.000	
Mano de obra (horas de desarrollo)	60 h	11.000	500.000	
Licencias / software	1	30.000	30.000	
Equipos / mantenimiento	2	300.000	300.000	
Total estimado:			*920.000*	