



CS-3.1	Enunciado de Prueba	Año:	2025
Especialidad:	590-107 Informática		
Prueba	Prueba 1A / Primera parte del Ejercicio en caso de Acc 3, 4 y 5	Acceso:	Todos

- OPCIÓN A -

EJERCICIO 1:

Desarrolla un programa orientado a objetos (Java o C++) para gestionar los productos de una tienda. Cada producto tendrá los siguientes atributos:

- **Identificador** (id): Número único.
- **Nombre**: Texto con el nombre del producto.
- **Precio**: Número con decimales.

El programa debe implementar:

- Una clase **Producto** que represente cada producto individual.
- Una clase **GestorProductos** que gestione los objetos **Producto** mediante una estructura de datos en memoria:
 - En caso de un desarrollo en Java utilizarás una `ArrayList<Producto>`.
 - En caso de un desarrollo en C++ utilizarás un `std::vector<Producto>`.

La clase **GestorProductos** deberá permitir realizar las siguientes operaciones:

1. **Añadir un nuevo producto a la estructura en memoria pidiendo datos por consola.**
2. **Guardar todos los productos de la estructura en memoria en un archivo de texto** llamado `productos.txt`. Cada producto debe guardarse en una línea, separando los atributos con comas (`id;nombre;precio`).
3. **Leer los productos desde el archivo `productos.txt`** y cargarlos en la estructura en memoria al iniciar el programa.
4. **Mostrar en consola todos los productos almacenados en la estructura en memoria**, indicando claramente `id`, `nombre` y `precio`.
5. **Mostrar en consola el precio medio de todos los productos almacenados en memoria.**



Ejemplo de menú:

- 1. *Añadir producto*
- 2. *Mostrar productos*
- 3. *Mostrar suma total de precios*
- 4. *Salir*

Seleccione una opción: _

El flujo del programa será:

- Al iniciar, se cargan todos los productos desde el archivo `productos.txt` a la estructura en memoria.
- Durante la ejecución, se permite:
 - Añadir nuevos productos a la estructura en memoria.
 - Mostrar los productos existentes en memoria.
 - Mostrar la suma total de precios de productos en memoria.
 - Salir del programa.
- Antes de finalizar el programa, todos los productos almacenados en memoria se guardan automáticamente en el archivo `productos.txt`.

Ejemplo del fichero `productos.txt`:

1;Manzana;0.50

2;Pan;1.20

3;Leche;0.99

Distribución de la puntuación (máximo 10 puntos):

Funcionalidad evaluada	Descripción	Puntos Máximos
1. Diseño orientado a objetos	Implementación correcta de las clases solicitadas: Producto y GestorProductos.	1
2. Gestión de archivo	Lectura y escritura correcta del fichero <code>productos.txt</code> .	3,5
3. Menú interactivo	Menú claramente presentado y funcional.	1
4. Añadir productos	Permite introducir productos nuevos en memoria correctamente.	1.25
5. Mostrar productos	Lista claramente todos los productos en memoria con atributos.	1.5
6. Mostrar suma total	Calcula correctamente y muestra el precio medio.	1,5
7. Salida del programa	Finaliza el programa correctamente y almacena automáticamente los productos en archivo.	0.25



EJERCICIO 2:

Se dispone de una base de datos MySQL para una tienda donde:

- Se registran departamentos, personas (empleados y clientes) y empleados adscritos a su departamento.
- Los productos, clasificados por categoría, pueden asignarse a un empleado responsable.
- Se definen períodos fiscales y se guarda qué cliente compró qué producto en cada período.

Descripción de las tablas y sus relaciones:

departamento

- `id` INT UNSIGNED AUTO_INCREMENT: clave primaria.
- `nombre` VARCHAR(50) NOT NULL: nombre del departamento.

persona

- `id` INT UNSIGNED AUTO_INCREMENT: clave primaria.
- `nif` VARCHAR(9) UNIQUE: número de identificación fiscal.
- `nombre` VARCHAR(25) NOT NULL: nombre de la persona.
- `apellido1` VARCHAR(50) NOT NULL: primer apellido.
- `apellido2` VARCHAR(50): segundo apellido (opcional).
- `ciudad` VARCHAR(25) NOT NULL: ciudad de residencia.
- `direccion` VARCHAR(50) NOT NULL: dirección postal.
- `telefono` VARCHAR(9): número de teléfono (opcional).
- `fecha_nacimiento` DATE NOT NULL: fecha de nacimiento.
- `sexo` ENUM('H', 'M') NOT NULL: sexo de la persona.
- `tipo` ENUM('empleado', 'cliente') NOT NULL: distingue si la persona es un empleado o un cliente.

categoria

- `id` INT UNSIGNED AUTO_INCREMENT: clave primaria.
- `nombre` VARCHAR(100) NOT NULL: nombre de la categoría de productos.



empleado

- `id_empleado` INT UNSIGNED: clave primaria y foránea que referencia a `persona(id)`.
- `id_departamento` INT UNSIGNED NOT NULL: clave foránea que referencia a `departamento(id)`.

producto

- `id` INT UNSIGNED AUTO_INCREMENT: clave primaria.
- `nombre` VARCHAR(100) NOT NULL: nombre del producto.
- `precio` DECIMAL(10,2) UNSIGNED NOT NULL: precio unitario.
- `tipo` ENUM('físico', 'digital', 'servicio') NOT NULL: tipo de producto.
- `stock` INT UNSIGNED NOT NULL: unidades disponibles.
- `id_categoria` INT UNSIGNED NOT NULL: clave foránea a `categoria(id)`.
- `id_empleado` INT UNSIGNED: clave foránea a `empleado(id_empleado)`, indica el responsable que lo gestiona.

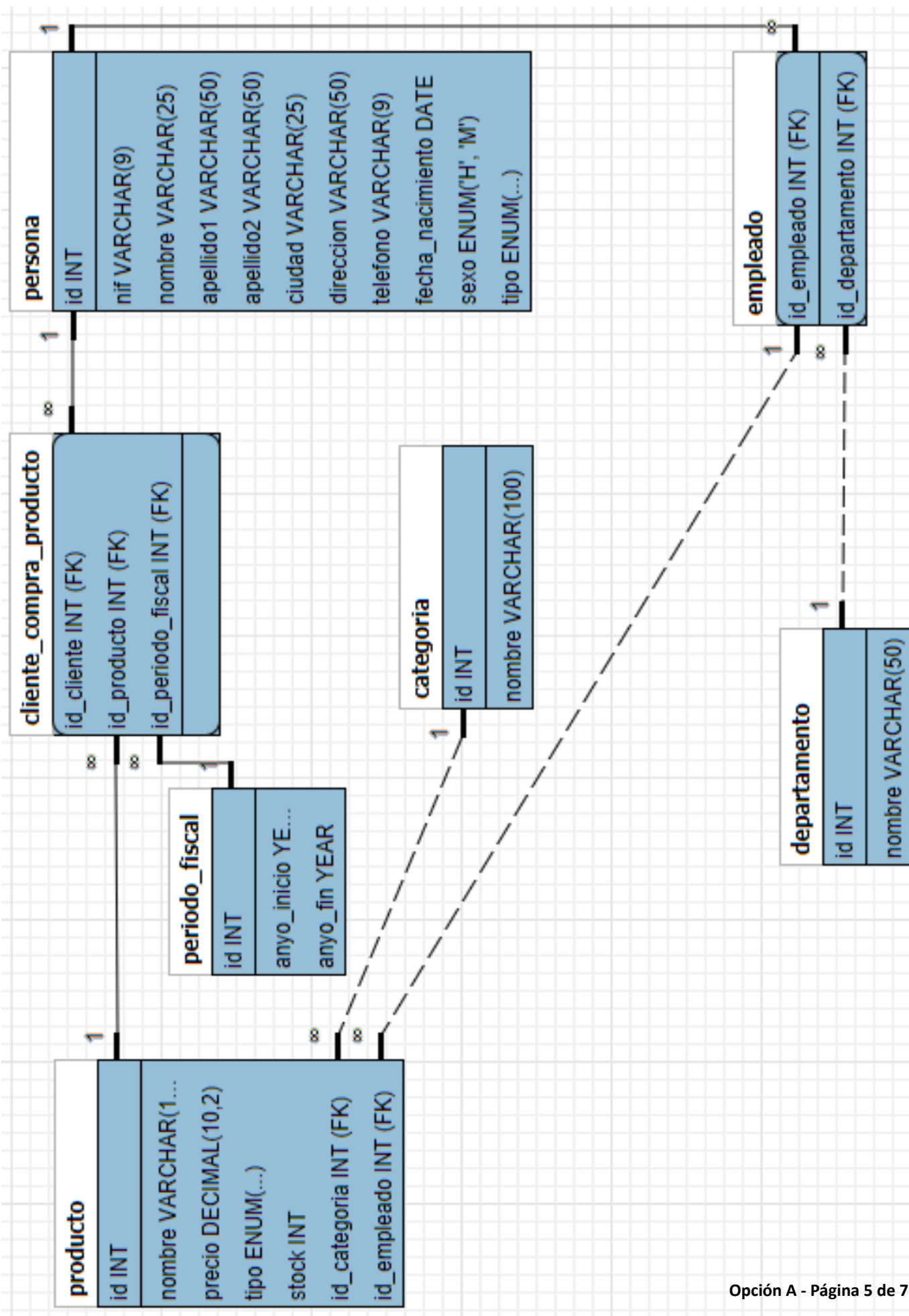
periodo_fiscal

- `id` INT UNSIGNED AUTO_INCREMENT: clave primaria.
- `anyo_inicio` YEAR NOT NULL: año de inicio del período fiscal.
- `anyo_fin` YEAR NOT NULL: año de fin del período fiscal.

cliente_compra_producto

- `id_cliente` INT UNSIGNED NOT NULL: clave foránea a `persona(id)`.
- `id_producto` INT UNSIGNED NOT NULL: clave foránea a `producto(id)`.
- `id_periodo_fiscal` INT UNSIGNED NOT NULL: clave foránea a `periodo_fiscal(id)`.
- **Clave primaria compuesta** (`id_cliente`, `id_producto`, `id_periodo_fiscal`) para registrar qué cliente compró qué producto en cada período fiscal.

Este sería su Diagrama EER:





Se deben escribir las sentencias SQL necesarias para realizar las operaciones indicadas:

1. Mostrar el identificador del empleado, su nombre y apellidos, junto al número de productos que gestiona, pero sólo de aquellos empleados que gestionan al menos 5 productos.
2. Para cada periodo fiscal, mostrar su identificador, años de inicio y fin, y el total de ingresos (suma de precios de los productos vendidos), ordenado por año de inicio.
3. Para cada categoría de producto, mostrar su identificador, nombre, el número total de unidades vendidas y el importe total ingresado en ventas de esa categoría —incluyendo también las categorías que aún no tengan ninguna venta— y ordenar las categorías de mayor a menor importe.
4. Mostrar el nombre y apellidos de los clientes que hayan comprado al menos un producto cuyo precio sea superior al precio medio de su categoría.
5. Creación de un TRIGGER (Disparador) el cual, antes de insertar una nueva compra en cliente_compra_producto, compruebe que el producto tiene stock > 0. Si no hay stock abortará la inserción con el siguiente error de usuario: SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No hay stock disponible para este producto'; si hay stock, restará 1 unidad al stock en producto;

Se valorará la eficiencia, la optimización y el rendimiento en su implementación.

Distribución de puntuación (máximo 10 puntos):

1. **Consulta 1:** Máximo 1,5 puntos.
2. **Consulta 2:** Máximo 1,5 puntos.
3. **Consulta 3:** Máximo 2 puntos.
4. **Consulta 4:** Máximo 2,5 puntos.
5. **Disparador:** Máximo 2,5 puntos.



EJERCICIO 3

ELIGE UNO DE ESTOS 2 EJERCICIOS: el 3A o el 3B

3A - JUSTIFICACIÓN DIDÁCTICA:

Deberás demostrar que sabes **desarrollar y justificar la aplicación práctica de las técnicas necesarias para el ejercicio docente** en el contexto del Ejercicio 2 de esta prueba, ejercicio relacionado con la creación y explotación de una base de datos MySQL para una tienda.

Para ello, se solicita:

1. (1 punto) Que el aspirante **enmarque la justificación didáctica** del citado Ejercicio 2 de esta prueba (el de base de datos) en la **normativa vigente**, incluyendo referencias al marco legal estatal y autonómico aplicable en la Comunidad Autónoma de Extremadura.

2. (1 punto) Que el aspirante **plantee actividades o actuaciones concretas** para llevar a cabo dicho ejercicio en el aula, y **explique el enfoque metodológico** que considera más adecuado, justificando su elección en función del contexto educativo y profesional.

3B - RESOLUCIÓN DE UNA SITUACIÓN CONCRETA EN EL AULA:

*Eres docente del módulo de **Programación de 1º de Desarrollo de Aplicaciones Multiplataforma (DAM)**. A estas alturas del curso, en el mes de marzo, observas que aproximadamente la mitad del alumnado aún no domina los conceptos básicos del lenguaje Java (como la declaración de variables, uso de arrays, clases y métodos). Además, sospechas que esta situación podría estar relacionada, al menos en parte, con el uso inadecuado de herramientas como **ChatGPT**, que algunos alumnos podrían estar utilizando para entregar ejercicios sin comprenderlos.*

Se solicita al aspirante que desarrolle una respuesta que demuestre que **sabe aplicar y justificar las técnicas necesarias para el ejercicio docente** en este contexto de aula. Para ello, deberá:

1. (1 punto) Enmarcar la resolución de esta situación concreta dentro de la **normativa vigente** en el ámbito de la Formación Profesional y del módulo de Programación en la Comunidad Autónoma de Extremadura.

2. (1 punto) Plantear **actuaciones y actividades realistas y viables** que contribuyan a solucionar el problema detectado, **sin perjudicar el desarrollo del resto del alumnado**, y justificar su **enfoque metodológico**.

CS-3.1	Enunciado de Prueba	Año:	2025
Especialidad:	590-107 Informática		
Prueba	Prueba 1A / Primera parte del Ejercicio en caso de Acc 3, 4 y 5	Acceso:	Todos

- OPCIÓN B -

EJERCICIO 1:

Diseña una aplicación que permita a los docentes de un instituto reservar aulas para sus clases, consultar disponibilidad y cancelar sus propias reservas. Este ejercicio puede resolverse en Java o en C++, a elección del aspirante.

Importante: No está permitido utilizar clases predefinidas o bibliotecas avanzadas para gestionar estructuras de datos. El código debe seguir un estilo de programación estructurada, utilizando funciones (en C++) o métodos estáticos (en Java) cuando corresponda.

Datos del sistema:

- Hay **5 aulas**, numeradas del **1 al 5**.
- La semana lectiva tiene **5 días**, del **lunes al viernes**, numerados del **1 al 5**.
- Cada día tiene **6 franjas horarias**, numeradas del **1 al 6**.

Requisitos funcionales:

1. Menú principal:

La aplicación debe presentar un menú interactivo que presenta las opciones y permite repetir tras cada acción, con las siguientes opciones:

```
[1] Reservar aula
[2] Ver disponibilidad de aulas
[3] Cancelar una reserva
[4] Salir del programa
```

Seleccione una opción: _



2. Funcionalidades del menú:

- **[1] Reservar aula:** El usuario deberá introducir el número de aula, día, hora y su nombre.
 - Se debe validar que los datos estén dentro de los rangos permitidos.
 - Si el **aula ya está reservada** en ese día y hora, se debe **mostrar el nombre del docente que hizo la reserva y denegar la nueva reserva**.
 - Si el espacio está **libre**, se debe almacenar el **nombre del docente** en esa posición
- **[2] Ver disponibilidad:** Mostrar en pantalla **todas las aulas, días y franjas horarias**. Para cada una, indicar si está **LIBRE** o **RESERVADA**, y en este último caso, **por quién**.
- **[3] Cancelar una reserva:** El usuario introducirá aula, día, hora y su nombre. La reserva **sólo podrá cancelarse si el nombre coincide con el del docente que la hizo**. En caso contrario, se mostrará un mensaje de error.
- **[4] Salir del programa:** Termina la ejecución del programa.

Requisitos técnicos:

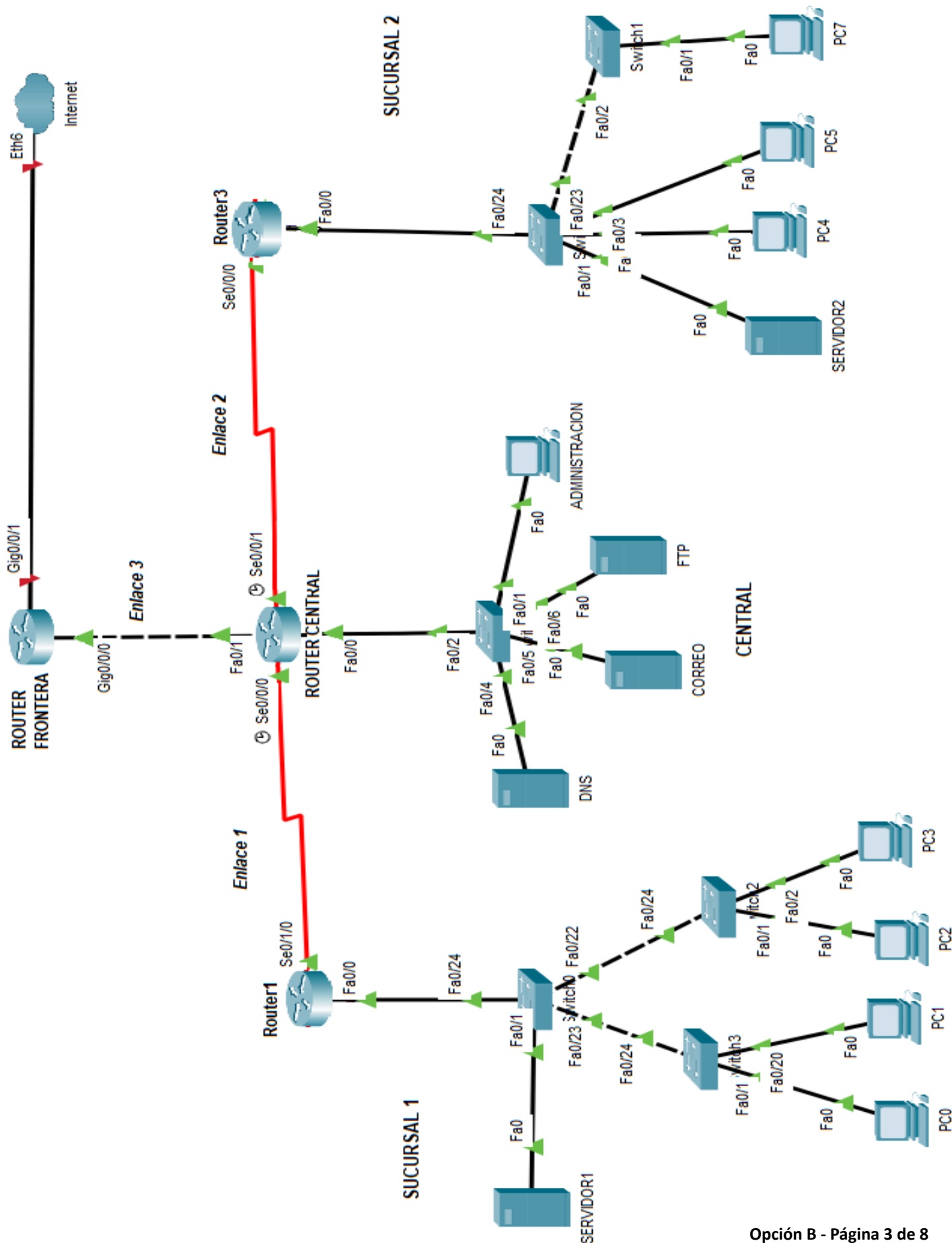
- **Todas las entradas del usuario deben ser validadas**, asegurando que los números de aula, día y hora estén dentro de los rangos establecidos.
- Se debe aplicar una estructura clara y organizada, **modularizar el código mediante funciones**.
- No se debe usar programación orientada a objetos (clases/objetos), solo programación estructurada.
- El uso de **estructuras como arrays bidimensionales o tridimensionales** para gestionar las reservas es adecuado y recomendado.

Distribución de la puntuación (máximo 10 puntos):

Funcionalidad evaluada	Descripción	Puntos Máximos
1. Mostrar menú interactivo	Presenta las opciones y permite repetir tras cada acción. Opción de salir y finalizar correctamente.	1
2. Realizar reserva	Permite reservar un aula específica en un día y hora concretos para un docente.	3
3. Mostrar disponibilidad de aulas	Presenta en pantalla el estado actual (libre o reservado) de todas las aulas, días y franjas horarias.	2
4. Cancelar reserva	Permite al docente cancelar una reserva existente si coincide con su nombre.	2.5
5. Validación de entradas	Garantiza que todas las entradas introducidas sean válidas en cuanto a tipo y rango.	1

EJERCICIO 2:

El esquema de la red de la empresa **EXINF** es el siguiente:





1. Subnetting con VLSM:

Se parte de la dirección de red **172.31.0.0/18**. Se debe subdividir en subredes según los siguientes requisitos:

- **Central:** máximo de 20 equipos
- **Sucursal 1:** máximo de 150 equipos
- **Sucursal 2:** máximo de 70 equipos
- **3 enlaces punto a punto** para interconectar los routers según el diseño anterior.

Dibuja una tabla similar a la siguiente con los datos de cada subred utilizando VLSM para optimizar el uso de direcciones IP.

Nombre	Dirección de subred	Máscara de subred CIDR	Máscara de subred (formato decimal)	Dirección de broadcast
Central	####	####	####	####
Sucursal 1	####	####	####	####
Sucursal 2	####	####	####	####
Enlace 1	####	####	####	####
Enlace 2	####	####	####	####
Enlace 3	####	####	####	####

2. División de SUCURSAL 2 en VLANs

La subred de **SUCURSAL 2** se subdivide en **3 VLANs**, cada una con el **mismo número de equipos**. Dibuja una tabla similar a la siguiente y calcula las direcciones de subred para cada VLAN:

Nombre	Dirección de subred	Máscara de subred (formato CIDR)	Máscara de subred (formato decimal)
VLAN 10	####	####	####
VLAN 20	####	####	####
VLAN 30	####	####	####



3. Asignación de IPs a las interfaces de cada router

Asigna IP a cada una de las interfaces de cada router, sabiendo que debe ser **obligatoriamente** la **ÚLTIMA o PENÚLTIMA DIRECCIÓN IP ÚTIL** de cada subred, según se indica junto al nombre de cada interfaz. Dibuja las tablas similares a las siguientes y complétalas con los datos solicitados:

Router 1		
Interfaz	Dirección IP	Máscara de subred CIDR
Fa0/0 (Última dirección útil de la subred)	####	####
Se0/1/0 (Última dirección útil de la subred)	####	####

Router Central		
Interfaz	Dirección IP	Máscara de subred CIDR
Fa0/0 (Última dirección útil de la subred)	####	####
Fa0/1 (Última dirección útil de la subred)	####	####
Se0/0/0 (Penúltima dirección útil de la subred)	####	####
Se0/0/1 (Penúltima dirección útil de la subred)	####	####

Router 3		
Interfaz	Dirección IP	Máscara de subred CIDR
Se0/0/0 (Última dirección útil de la subred)	####	####
Fa0/0.10 (Última dirección útil de la VLAN 10)	####	####
Fa0/0.20 (Última dirección útil de la VLAN 20)	####	####
Fa0/0.30 (Última dirección útil de la VLAN 30)	####	####



4. Tabla de enrutamiento – Router Central

Sabiendo que en la red se utiliza **enrutamiento estático y que** para acceder a Internet los paquetes se redirigen al **Router Frontera**, rellena la tabla de enrutamiento del **Router Central**, aplicando la **sumarización de redes cuando sea posible**.

Nota: Dibuja una tabla similar a la siguiente y utiliza tantas filas como sean necesarias.

Dirección de subred	Máscara de subred CIDR	Interface	IP siguiente salto
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####
####	####	####	####



5. ACL – Acceso al servidor FTP

5.1. Asigna una IP fija al servidor FTP. Dibuja una tabla similar a la siguiente con sus datos:

Dirección IP del servidor FTP	Máscara de subred CIDR
####	####

5.2. Se desea crear una ACL con los siguientes requisitos:

- **Permitir** que equipos **externos** a la red de nuestra empresa hagan *ping* al servidor FTP.
- **Permitir** que equipos **externos** a la red de nuestra empresa accedan a los servicios **FTP** (puertos 20 y 21).
- **Bloquear** cualquier otro tráfico entrante hacia la red interna de la empresa.

Define la ACL, incluyendo el/los puertos donde corresponda. Para ello, dibuja una tabla similar a la siguiente y rellénala:

Acción	Protocolo	Origen	Destino	Descripción
####	####	####	####	####
####	####	####	####	####
####	####	####	####	####
####	####	####	####	####

5.3. Indica dónde se aplicaría:

- Indica en qué **router** y en qué **interfaz** aplicarías esta ACL extendida que acabas de crear, y si es de entrada o de salida. Para ello, dibuja una tabla similar a la siguiente y rellénala.

Nombre del router	Interfaz del router	Dirección (entrada / salida)
####	####	####

Distribución de la puntuación (máximo 10 puntos)

Tarea evaluada	Puntos Máximos
Subnetting con VLSM	3
División en VLANs con asignación correcta	1.5
Asignación de IPs a las interfaces de los routers	1.5
Tabla de enrutamiento del Router Central	2.5
Definición de la ACL y aplicación correcta	1.5



EJERCICIO 3

ELIGE UNO DE ESTOS 2 EJERCICIOS: el 3A o el 3B

3A - JUSTIFICACIÓN DIDÁCTICA:

Deberás demostrar que sabes **desarrollar y justificar la aplicación práctica de las técnicas necesarias para el ejercicio docente** en el contexto del Ejercicio 2 de esta prueba, ejercicio relacionado con la planificación, diseño y gestión de redes con subnetting (VLSM), VLANs, enrutamiento estático y ACL.

Para ello, se solicita:

1. (1 punto) Que el aspirante **enmarque la justificación didáctica** del citado Ejercicio 2 de esta prueba (el de redes) en la **normativa vigente**, incluyendo referencias al marco legal estatal y autonómico aplicable en la Comunidad Autónoma de Extremadura.

2. (1 punto) Que el aspirante **plantee actividades o actuaciones concretas** para llevar a cabo dicho ejercicio en el aula, y **explique el enfoque metodológico** que considera más adecuado, justificando su elección en función del contexto educativo y profesional.

3B - RESOLUCIÓN DE UNA SITUACIÓN CONCRETA EN EL AULA:

*Eres docente del módulo de **Programación de 1º de Desarrollo de Aplicaciones Multiplataforma (DAM)**. A estas alturas del curso, en el mes de marzo, observas que aproximadamente la mitad del alumnado aún no domina los conceptos básicos del lenguaje Java (como la declaración de variables, uso de arrays, clases y métodos). Además, sospechas que esta situación podría estar relacionada, al menos en parte, con el uso inadecuado de herramientas como **ChatGPT**, que algunos alumnos podrían estar utilizando para entregar ejercicios sin comprenderlos.*

Se solicita al aspirante que desarrolle una respuesta que demuestre que **sabe aplicar y justificar las técnicas necesarias para el ejercicio docente** en este contexto de aula. Para ello, deberá:

1. (1 punto) Enmarcar la resolución de esta situación concreta dentro de la **normativa vigente** en el ámbito de la Formación Profesional y del módulo de Programación en la Comunidad Autónoma de Extremadura.

2. (1 punto) Plantear **actuaciones y actividades realistas y viables** que contribuyan a solucionar el problema detectado, **sin perjudicar el desarrollo del resto del alumnado**, y justificar su **enfoque metodológico**.