Ajuntament de Calvià
Mallorca

DOCUMENT ELECTRÒNIC

Versió NTI:	http://administracionelectronica.gob.es/ENI/XSD/v1.0/documento-e
Identificador:	1377816
Òrgans:	Ajuntament de Calvià
Data Captura:	2025-11-26 13:25:41
Origen:	Administració
Estat elaboració:	EE01
Tipus documental:	Còpia autèntica en paper de document electrònic
Tipus firmes:	Xades Internally Detached

Firmant	Perfil	Data	Estat firma
MARIA JOSE FLORIT TORRES		26/11/2025	Vàlida
BARTOLOME FULLANA BARCELO		26/11/2025	Vàlida

Ajuntament de Calvià Mallorca | Oficina Municipal de Tributos

ACTA Nº3 REALIZACION DEL PRIMER EJERCICIO, TEÓRICO PRÁCTICO, RELACIONADO CON EL TEMARIO ESPECÍFICO PARA LA SELECCIÓN DE UNA PLAZA DE TÉCNICO/A DE DESARROLLO, GRUPO A, SUBGRUPO A2 DE LA OFICINA MUNICIPAL DE TRIBUTOS DE CALVIÀ (BOIB núm. 77 DE 19/06/2025, Y RECTIFICACIÓN BOIB núm. 80 DE 24/06/2025)

Acta de la reunión

Identificación de la reunión:

Fecha: 25.11.2025
Hora: 9:00h
Lugar: Sala de Formación de Policía. Edificio de la Policía Local

Asistentes
Presidente : Bartolomé Fullana Barceló
Vocal 1: Pedro Tous Durán
Vocal 3: Mª José Florit Torres – Secretaria del Tribunal
Vocal 4: María Pomar Forteza

Orden del día

Celebración del **PRIMER EJERCICIO TEÓRICO PRÁCTICO,** relacionado con el temario específico para la selección de una plaza de Técnico de desarrollo, grupo A, subgrupo A2 de la Oficina Municipal de Tributos.

Acuerdos

1. En Calvià a las 9:05, en el Aula de Formación del Edificio de Policía, se da inicio al primer ejercicio de la convocatoria citada anteriormente.

2. De las 3 personas aspirantes se presentan sólo dos.

3. El Tribunal ha decidido dejar 5 minutos de cortesía, y a las 9:05 empieza la prueba informando a los aspirantes de todos los aspectos importantes a tener en cuenta como la duración de la prueba, el uso de hojas de contestación, el código identificador que tienen que indicarse en las hojas de examen así como el material que puede utilizarse durante el desarrollo de la prueba.

4. De los dos modelos de prueba, se ha realizado el sorteo delante los aspirantes saliendo el **modelo A**, el cual se adjunta a la presente acta

5. Durante el desarrollo de la prueba uno de los aspirantes solicita aclaración sobre los “Diagramas de clases UML”, preguntando si es necesario identificar las clases y sus atributos cuando éstos aparecen también en el diagrama UML del apartado 2. Por el Sr. Pedro Tous se informa a los dos asistentes que lo que pueden hacer es no incluirlos, porque los atributos en el diagrama de UML ya que se ha explicado en el primer apartado.

6.- Una vez finalizados los ejercicios, los aspirantes proceden a su entrega al tribunal, procediendo a la separación de las solapas con los datos de identificación de las hojas de ejercicios quedando las solapas en un sobre cerrado y firmado por el Presidente. Igualmente, una vez realizadas

fotocopias de los ejercicios para que puedan ser evaluados por los miembros del tribunal, se depositan los originales en sobre cerrado y firmado por el Presidente, quedando ambos sobres bajo su custodia.

El tribunal acordará próximamente la fecha de reunión para la corrección de los exámenes.

7.- Se informa a los aspirantes que con la publicación de las notas se publicará la fecha de realización del segundo ejercicio que en cualquier caso será con posterioridad al día 6 de enero de 2026.

El presidente levanta la sesión siendo las 13'45 horas y la Secretaria redacta esta acta con el acuerdo de publicarse en el apartado correspondiente del proceso selectivo al cual hace referencia.

La Secretaria

VºBº
El Presidente.

Opción A

Ejercicio 1: DataWarehouse y ETL

Un ayuntamiento de tamaño medio necesita modernizar su sistema de análisis de la recaudación del **Impuesto de Bienes Inmuebles (IBI)**.
Actualmente, el ayuntamiento maneja tres sistemas de información que no están conectados entre sí:

- **Catastro (Datos de Inmuebles):** Un fichero anual que proporciona el **Catastro** (organismo nacional). Contiene la información física y el valor de todos los inmuebles del municipio.
  Campos clave: Referencia_Catastral, Uso (Residencial, Comercial, Industrial), Superficie_m2, Valor_Catastral, Distrito_Postal.
- **Padrón Municipal (Datos de Ciudadanos):** La base de datos de habitantes del municipio.
  Campos clave: DNI, Nombre_Completo, Direccion_Residencia, Fecha_Nacimiento.
- **Sistema de Recaudación (Datos de Pagos):** La base de datos de contabilidad del ayuntamiento donde se generan los recibos y se anota quién ha pagado.
  Campos clave: ID_Recibo, Referencia_Catastral (del inmueble), DNI_Contribuyente (de la persona), Importe, Estado (Pagado, Pendiente, En Fraccionamiento, Embargado), fecha_de_pago.

El objetivo: El equipo de gobierno quiere un Data Warehouse (DW) para responder preguntas clave de forma unificada del estilo siguiente:

- "¿Cuál es el porcentaje de impagos en el distrito postal X?"
- "¿Cuánto recaudamos de media por los inmuebles de 'Uso Comercial' vs 'Residencial'?"
- "¿Cuántos propietarios de inmuebles no residen en el municipio?"
- "¿Cuál fue el ingreso total recaudado el segundo trimestre por este impuesto?"

Se debería implementar un sistema que permita resolver de forma eficiente preguntas acerca de recaudaciones o impagos, por meses, trimestres o años, en diferentes usos y distritos postales, para residentes o no residentes. De forma conjunta o por separado.

Apartado 1: (6 puntos)
Diseña un **Modelo en Estrella (Star Schema)** simple para este problema. Define la Tabla de Hechos central y las Dimensiones principales que la rodean.

Apartado 2: (4 puntos)
Basándote en el modelo DW que diseñaste, describe las actividades clave que realizarías en cada fase del proceso **ETL (Extract, Transform, Load)**.

Ejercicio 2: Diagramas de clases UML

Se desea modelar el sistema de gestión de una tienda. El sistema debe ser capaz de manejar Clientes y Proveedores.

- Tanto Clientes como Proveedores son considerados "Terceros" por la empresa y comparten atributos comunes: nombre, direccion, telefono y email.
- Un Cliente tiene un atributo específico: fecha nacimiento.
- Un Proveedor tiene un atributo específico: Razón Social
- El sistema debe gestionar Documentos Comerciales. Existen dos tipos de documentos: Pedidos y Facturas.
- Todos los documentos (Pedido y Factura) tienen un numero identificativo, una fecha y una lista de lineas de detalle (que contienen producto, cantidad y precio unitario). Las facturas podrán incluir un porcentaje de descuento.
- De los productos necesitamos saber nombre, el precio de venta y el stock
- Un mismo producto puede ser suministrado por diferentes proveedores que nos lo pueden ofrecer a diferentes precios.
- Un Cliente puede realizar múltiples Pedidos. Un Pedido pertenece a un solo Cliente.
- Un Pedido puede generar una (y solo una) Factura.
- Tanto las facturas como los pedidos deben tener un método calcular total:
  - En un Pedido, el total es simplemente la suma de (cantidad * precioUnitario) de todas sus líneas.
  - En una Factura, el total se calcula igual que en el pedido, pero además se le aplica el porcentaje de descuento y un 21% de IVA.

Apartado 1: (6 puntos)

Identifica las clases con sus atributos

Identifica las relaciones, su tipo y explica su multiplicidad cuando la haya.

Explica como se podría aplicar el polimorfismo para calcular los totales de los documentos comercials.

Apartado 2: (4 puntos)

Realiza el diagrama de clases UML que represente el sistema del enunciado

Ejercicio 3: HTML5 y CSS3.

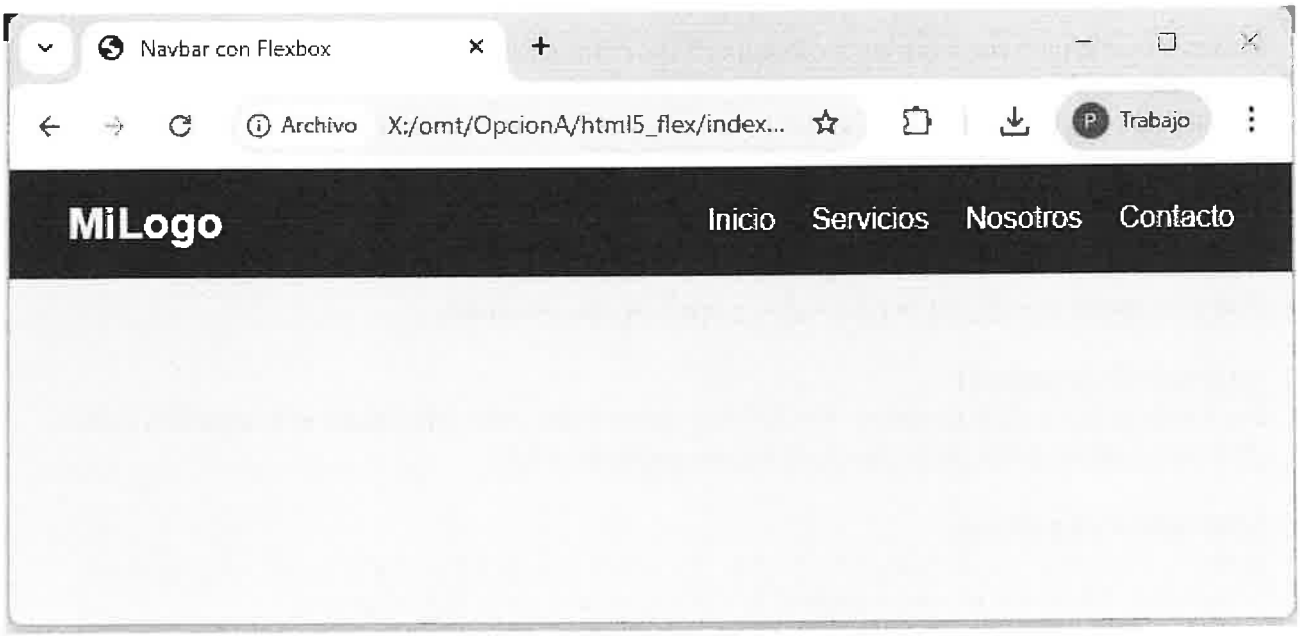
El objetivo de este ejercicio es crear una barra de navegación horizontal y moderna.

Requisitos:

1. Estructura HTML5: Utiliza etiquetas semánticas y listas para los enlaces de navegación.
2. Contenido: La barra debe tener:
   - Un "Logo" (puede ser solo texto).
   - Una lista de enlaces (ej. Inicio, Servicios, Nosotros, Contacto).
3. Estilo CSS3:
   - El Logo y la lista de enlaces deben estar en extremos opuestos: el logo a la izquierda y los enlaces a la derecha. Para ello debes usar un contenedor flex.
   - El Logo y los enlaces deben estar centrados verticalmente el uno con el otro.
   - La lista de enlaces también debe ser un contenedor flex para que cada uno de ellos se ponga en horizontal.
   - Quita los estilos por defecto de la lista (viñetas/bullets) y de los enlaces (subrayado).
   - Añade un efecto :hover para que cambie el color a color: #007bff; cuando el ratón pase por encima de los enlaces. Haz que la transición tenga un efecto suave.
   - Puedes usar los siguiente códigos de colores:
     Color de fondo de body: #f0f0f0
     Color de fondo de la barra de navegación: #333
     Color de los elementos de la barra: white

Objetivo Visual:

Buscas este diseño clásico: Logo a la izquierda, y los enlaces agrupados a la derecha, todo en una sola línea. Los espacios adicionales (padding) de relleno y los márgenes (margin) pueden ser aproximados.



Apartado 1: (6 puntos)

Escribe el código HTML5 para poder representar correctamente la barra de navegación.

Apartado 2: (4 puntos)

Escribe el código CSS3 para poder representar correctamente la barra de navegación.

Ejercicio 4: SQL

Se te proporciona la siguiente estructura de base de datos para gestionar una pequeña universidad.

1. Estudiantes

ID_Estudiante (INT, PK): ID único del estudiante.
Nombre (VARCHAR 40): Nombre del estudiante.
Apellido (VARCHAR 40): Apellido del estudiante.
Fecha_Nacimiento (DATE): Fecha de nacimiento.

2. Profesores

ID_Profesor (INT, PK): ID único del profesor.
Nombre_Profesor (VARCHAR 40): Nombre y apellido del profesor.
Departamento (VARCHAR 40): (ej. "Informática", "Matemáticas", "Historia").

3. Cursos (Cada curso lo imparte un único profesor)

ID_Curso (INT, PK): ID único del curso.
Nombre_Curso (VARCHAR 40): Nombre del curso (ej. "Bases de Datos", "Álgebra Lineal").
ID_Profesor (INT, FK): Profesor que imparte el curso (enlaza con Profesores).
Créditos (INT): Número de créditos del curso.

4. Inscripciones (Los estudiantes se pueden inscribir a N cursos y tienen una nota final)

ID_Inscripcion (INT, PK): ID único de la inscripción.
ID_Estudiante (INT, FK): Estudiante inscrito (enlaza con Estudiantes).
ID_Curso (INT, FK): Curso en el que está inscrito (enlaza con Cursos).
Nota (DECIMAL): Nota final del estudiante en ese curso (puede ser NULL si aún no está calificado).

Deberás usarla para resolver las 5 consultas SQL planteadas:

Apartado 1: (2 puntos)

Obtener el Nombre y Apellido de todos los estudiantes nacidos después del 1 de enero de 1999.

Apartado 2: (2 puntos)

Listar el Nombre del estudiante y el Nombre_Curso en el que están inscritos de aquellos cursos impartidos por profesores del departamento de "Informática" y ordenar los resultados alfabéticamente por el nombre del curso y apellido del estudiante.

Apartado 3: (2 puntos)

Encontrar el promedio de notas - AVG(Nota) - para cada curso mostrando solo aquellos cursos (Nombre_Curso) cuyo promedio de notas sea superior a 8.0.

Apartado 4: (2 puntos)

Identificar a todos los profesores (Nombre_Profesor) que no están impartiendo ningún curso actualmente (es decir, no están asignados a ningún curso en la tabla Cursos).

Apartado 5: (2 puntos)

Generar un listado único de todas las personas activas en la universidad, combinando estudiantes y profesores. El listado debe mostrar dos columnas: NombreCompleto y Rol.

- Para los estudiantes, NombreCompleto será la concatenación (debes usar la función CONCAT) de Nombre y Apellido (ej. "Juan Pérez"), y el Rol será 'Estudiante'.
- Para los profesores, NombreCompleto será Nombre_Profesor, y el Rol será 'Profesor'..

Ejercicio 5: Java y API REST

"Lista de Tareas" (En Memoria)

Implementación en java de las clases necesarias para gestionar una "Lista de Tareas" (To-Do List). La información **no** debe persistir en una base de datos. Todos los datos (la lista de tareas) deben almacenarse en la **memoria del servidor.**

Apartado 1: (1 puntos)
Implementa una clase Tarea con los siguientes atributos:

- `id` (Long): Identificador único (lo generará el servidor).
- `titulo` (String): El texto de la tarea.
- `completada` (boolean): `true` si la tarea está terminada, `false` por defecto.

Apartado 2: (5 puntos)
Implementa la clase que implementa la lógica de la aplicación que permita realizar las siguientes operaciones:

1. Crear Tarea
2. Obtener todas las tareas
3. Obtener una tarea por ID
4. Actualizar los datos de una tarea
5. Eliminar una Tarea

Apartado3: (1 punto)
Describe brevemente como debemos modificar la clase anterior para que funcione como un servicio con el Framework Spring

Apartado 4: (3 puntos)
Inyecta y usa el servicio anterior en una nueva clase controlador Spring (@RestController) que permita realizar las operaciones:

1. Crear una Tarea:
    - **Método:** `POST - @PostMapping`
    - **Ruta:** `/api/tareas`
    - **Body (JSON):** Un objeto `Tarea` (solo con `titulo`, ya que el `id` y `completada` se gestionan en el servidor).
    - **Respuesta:** El objeto `Tarea` recién creado (con `id`, `titulo` y `completada=false`) y un código **201 Created**.

2.Actualizar una Tarea:
    - **Método:** `PUT` (o `PATCH`) - @PutMapping
    - **Ruta:** `/api/tareas/{id}`
    - **Body (JSON):** Un objeto `Tarea` con el estado actualizado (ej. `{"titulo": "Comprar leche", "completada": true}`).
    - **Respuesta (Éxito):** El objeto `Tarea` actualizado y un código **200 OK**.
    - **Respuesta (Error):** Un mensaje de error y un código **404 Not Found** si el `id` no existe.