EXAMEN — DISEÑO DE INTERFACES (2º DAM)

Parte teórica (tipo test) — Diseño de Interfaces

Nombre y apellidos	Curso/Grupo	Fecha	Calificación
_____Iris Pérez Aparicio_____	_____2º DAM_____	06 / 02 / 2026	_____

Instrucciones: rellenar la tabla final con las respuestas que consideres correctas

1. ¿Qué efecto tiene `grid-template-areas`?

```
body{
    display: grid;
    grid-template-rows: 100px 300px auto 150px;
    grid-template-columns: 100%;
    grid-template-areas:
    "nav"
    "header"
    "main"
    "footer";
}
```

A) Define el tamaño de las filas
B) Asigna nombres a zonas del grid para posicionar elementos por `grid-area`
C) Hace que el grid sea responsive automáticamente
D) Convierte `nav` en un elemento fijo

2. ¿Qué significa?

```
grid-template-columns: 100%;
```

A) El elemento tendra 100 columnas
B) Que hay una sola columna que ocupa todo el ancho del elemento
C) Que el grid tiene columnas automáticas según el contenido
D) Que el grid ocupa el 100% del alto del elemento

3. En un contenedor con:

```
display: flex;
align-items: center;
```

¿qué efecto produce `align-items: center` por defecto?

A) Centra los elementos horizontalmente (eje principal)
B) Centra los elementos verticalmente (eje transversal)
C) Centra tanto horizontal como vertical
D) No hace nada si no se usa `justify-content`

4. En `nav ul` se usa : `justify-content: space-around;`. ¿Qué hace `space-around`?

```
nav{
display: flex;
justify-content: space-around;
align-items: center;
}
```

A) Centra los elementos y los pega entre sí
B) Reparte el espacio con márgenes iguales solo en los extremos
C) Reparte el espacio alrededor de los elementos, dejando espacio también en los extremos
D) Hace que los elementos se vayan a una nueva línea

5. En `.contenedor` se usa (flex-wrap: wrap;)

```
.contenedor{
display: flex;
flex-wrap: wrap;
gap: 20px;
}
```

¿Qué provoca?

A) Que las cards se estiren al 100% de ancho
B) Que los elementos salten a la siguiente línea si no caben
C) Que el `gap` solo sea vertical
D) Que el contenedor se convierta en grid

6. En `.card` se usa

```
.card{
transition: 0.7s all ease;
}
```

¿Qué significa exactamente?

A) Transiciona solo el color
B) Transiciona todas las propiedades animables en 0.7s con curva `ease`
C) Transiciona solo el hover
D) Transiciona solo el `transform`

7. ¿cuál afirmación es correcta?

```
li:hover{ transform: scale(1.1); }
a:hover{ transform: scale(1.03); }
```

A) `transform` no funciona con `hover`
B) Se aplica el escalado a todos los elemtos 'li' y 'a' del documento
C) El escalado solo afecta a los elementos 'a'
D) `scale` solo funciona en imágenes

8. En el menú:

```
nav ul a{
  color: whitesmoke;
  transition: 1s all ease;
}
a:hover{ color: aqua;
```

¿Qué ocurrirá al pasar el ratón por un enlace del menú?

A) Cambia el color de golpe (sin animación)
B) Cambia el color a aqua con una transición de 1s
C) Cambia el fondo a aqua
D) Se subraya automáticamente

9. En el `body` se define:

```
display:grid;
grid-template-rows: 100px 300px auto 150px;
grid-template-columns: 100%;
grid-template-areas:
  "nav"
  "header"
  "main"
  "footer";
```

¿Cuántas filas, columnas y áreas se definen en este grid?

A) 4 filas, 1 columna, 4 áreas
B) 4 filas, 4 columnas, 4 áreas
C) 1 fila, 4 columnas, 4 áreas
D) 4 filas, 1 columna, 1 área

10. ¿qué pasa con las imágenes dentro del footer?

```
img{ width: 100%; }
footer img{ width: 150px; height: 80px; }
```

A) Se aplica `img { width: 100% }` y se ignora `footer img`
B) Se aplica `footer img` por ser más específico y sobrescribe el width para esas imágenes
C) Se mezclan: width 100% y height 80px
D) Ninguna regla se aplica por conflicto

Pregunta	Respuesta
1	B
2	B
3	B
4	C
5	B
6	B
7	B
8	B
9	A
10	B

PRACTICA

Para la parte practica se tendra que desarrollar una aplicación de cliente que consuma la api que desarrollamoscon tencnologia .net aqui dejare el program.cs de la api functional.

Puntos a elaborar:

1. Estructurar el document usando display grid (al body se le dara esta estructura).

   -nav

   -header

   -main

   -footer

2. Crearas un div dentro del main a la cual le daras una clase y un id llamdo “contenedor” Para con JS crear las card donde mostraremos todos los atributos de los elemetos que nos da la api (nombre, apellidos, dni, lugarNacimineto, etc). Se tendra en cuenta el diseño de estas card y de la distribucion de este contenedor. Para el layout del contenedor se debera usar display Flex

3. Crearas un formulario que con el evento correspondiente para dar de alta a una nueva persona y enviarlo a la api con todos los atributos completos. Se tendra en cuenta el diseño y funcionamiento del formulario

CODIGO A USAR:

EJEMPLO DE PROGRAM.CS (API):

```
//paso 1
//iniciar los constructores intanciarlo y arrancar la app

var constructor= WebApplication.CreateBuilder(args);


constructor.Services.AddControllers();

 constructor.Services.AddCors( opciones =>
 {
     opciones.AddDefaultPolicy( politica =>
         politica.WithOrigins("http://127.0.0.1:5500", "http://localhost:5500") //agg
las ip que podran hacer peticiones
         .AllowAnyHeader()
         .AllowAnyMethod()
     );
 });

var app = constructor.Build();


app.UseCors();


app.MapControllers();

//lista
var listaPersona = new List<Persona>
{
    new(1, "Andrés", "Salazar Gómez", 29, "V12345678", "Caracas", "Venezuela", "Av.
Libertador, Edif. Sol", "Universitario"),
    new(2, "Mariana", "Pinto Rodríguez", 34, "E23456789", "Madrid", "España", "Calle
Gran Vía 120", "Postgrado"),
    new(3, "José", "Márquez León", 41, "M34567890", "Ciudad de México", "México",
"Col. Roma Norte", "Universitario"),
    new(4, "Valentina", "Ríos Herrera", 22, "C45678901", "Bogotá", "Colombia",
"Chapinero Alto", "Bachiller"),
```

```
    new(5, "Carlos", "Medina Torres", 37, "A56789012", "Buenos Aires", "Argentina",
"Av. Corrientes 540", "Universitario"),
    new(6, "Paula", "Navarro Díaz", 26, "V44556677", "Mérida", "Venezuela", "Av. Las
Américas, Apt. 3B", "Universitario"),
    new(7, "Ricardo", "Figueroa Silva", 45, "P67890123", "Lima", "Perú", "Miraflores,
Calle 9", "Técnico Superior"),
    new(8, "Camila", "Suárez Pacheco", 31, "C78901234", "Medellín", "Colombia", "El
Poblado", "Universitario"),
    new(9, "Daniel", "Peña Castillo", 28, "U89012345", "Montevideo", "Uruguay",
"Barrio Pocitos", "Universitario"),
    new(10, "Lucía", "Ortega Morales", 39, "E90123456", "Barcelona", "España", "Calle
Aragón 88", "Postgrado"),
    new(11, "Miguel", "Ramos Contreras", 24, "V10111213", "Guatire", "Venezuela",
"Sector El Ingenio", "Universitario"),
    new(12, "Natalia", "Vera Montoya", 33, "C14151617", "Cali", "Colombia", "Barrio
Granada", "Universitario"),
    new(13, "Óscar", "Luna Cabrera", 48, "B18192021", "La Paz", "Bolivia", "Zona
Sopocachi", "Técnico Superior"),
    new(14, "Gabriela", "Mendoza Ruiz", 27, "H22232425", "Tegucigalpa", "Honduras",
"Col. Morazán", "Universitario"),
    new(15, "Héctor", "Bautista Flores", 52, "C26272829", "Santiago", "Chile",
"Providencia", "Bachiller")
};



//get todos
app.MapGet("/personas", ()=> Results.Ok(listaPersona));

//get uno
app.MapGet("/personas/{dni}", (string dni) =>
{
    var persona= listaPersona.FirstOrDefault(elemtos => elemtos.Dni == dni);

    if (persona != null)
    {
        return Results.Ok(persona);
    }
    else
    {
       return Results.NotFound("NO ENCONTRADO");
    }
});


//crea uno

app.MapPost("/personas", (Persona nuevaPersona)=>
```

```
{
    if (nuevaPersona.Nombre == null || nuevaPersona.Nombre == "")
    {
        return Results.BadRequest("FALTO EL NOMBRE");
    }

    if (nuevaPersona.Edad == null || nuevaPersona.Edad <= 18 )
    {
        return Results.BadRequest("no hay edad o es menor a 18");
    }


    //creamos el id que continua en la lista
    var nuevoId= listaPersona.Max(elemeto => elemeto.Id) +1;

    //creamos la instancia de la persona
    var persona = new Persona(
        nuevoId,
        nuevaPersona.Nombre,
        nuevaPersona.Apellidos,
        nuevaPersona.Edad,
        nuevaPersona.Dni,
        nuevaPersona.LugarNacimineto,
        nuevaPersona.PaisNacimineto,
        nuevaPersona.Direccion,
        nuevaPersona.UltimoEstudio
    );

    //agg la persona a la lista

    listaPersona.Add(persona);

    return Results.Created("CREADO", persona);


});

//actualiza uno

app.MapPut("/personas", (Persona perosnaActualiza)=>
{
    //buscamos la persona
    var persona = listaPersona.FirstOrDefault(elemento => elemento.Dni ==
perosnaActualiza.Dni);
    //verificamos que existe
    if(persona == null)
    {
        //retornamos un 404 en el caso que no la encuentre
```

```
        return Results.NotFound("persona no encontrada");
    }
    else
    {
        //obtenemos de indice en lista de la persona que recibimos y vamos a
actualizar
        var index = listaPersona.FindIndex(elementos => elementos.Id ==
perosnaActualiza.Id);

        //actualizamos la persona
        listaPersona[index] = perosnaActualiza;

        //retornamos la perosna actualizada que nos llego en el cuerpo
        return Results.Ok(perosnaActualiza);


    }
});

//borra uno

app.MapDelete("/personas/{id:int}", (int id) =>
{
    var persona = listaPersona.FirstOrDefault(elemento => elemento.Id == id);

    // si existe ña persona sera diferente de null
    if (persona != null)
    {
        listaPersona.Remove(persona);

        return Results.Ok("eliminada");

    }
    else
    {
        return Results.NotFound("no encotrada");
    }



});




app.Run(); //arranca el servicio
```

```
//clase record
public record Persona(int Id, string Nombre, string Apellidos, int Edad, string Dni,
string LugarNacimineto, string PaisNacimineto, string Direccion, string
UltimoEstudio);
```

funciones fetch:

dameTodos:

```
let contenedor = document.getElementById("contenedor");
fetch("http://localhost:5085/personas")

    .then(res => {

        if (!res.ok) {
            return;
        } else {

            return res.json();
        }
    })
    .then(todos => {
        if (todos) {

            console.log(todos)

            todos.forEach(persona => {

                contenedor.innerHTML += `
            <div class="card">

                <div class="titulo">

                    <h2> ${persona.id} ${persona.nombre}</h2>
                </div>

                <div class="imagen">
                  <p>${persona.apellidos}</p>
                </div>
```

```
                <div class="precio">
                  <p >edad: ${persona.edad} </p>
                  <p >dni: ${persona.dni} </p>

                  <p >lugar Nacimineto: ${persona.lugarNacimineto} </p>

                </div>

                <div class="botones">
                    <button style="background:green;"
onClick="editarProducto(${persona.id})">EDITAR</button>
                </div>

                <div class="botones">
                    <button
onClick="eliminarProducto(${persona.id})">ELIMINAR</button>
                </div>

            </div>
        `;

            });



        } else {
            return;
        }
    })
```

Crear Nueva Persona:

```
    if(document.getElementById('formulario')){
        let formulario=document.getElementById('formulario');


        formulario.addEventListener('submit', (e)=>{

                e.preventDefault();

                fetch("http://localhost:5246/personas", {

                    method:"POST",
                    headers:{ "Content-Type": "application/json" },
                    body: JSON.stringify({
```

```
                    nombre: formulario.nombre.value,
                    apellidos: formulario.apellidos.value,
                    edad: formulario.edad.value,
                    dni: formulario.dni.value,
                    lugarNacimineto: formulario.lugarNacimineto.value
                })
            })
            .then((res)=>{
                if(res.status == 201){
                    alert("Persona creado")

                    location.href="index.html"
                }
            })
    })
    }
```