CICLO: DAW/DAM

LENGUAJE DE MARCAS

Manipulación de documentos Web

Alberto Gálvez Aguado

14273541V

ÍNDICE

1. Objetivo de la tarea

En esta tarea se pretende diseñar y programar un formulario web completo que permita introducir varios datos (email, contraseña, nombre de usuario y nombre y apellidos) y validarlos utilizando exclusivamente JavaScript, sin recurrir a las validaciones propias de HTML5.

El objetivo es practicar la manipulación del DOM para cumplir los resultados de aprendizaje RA03_c, RA03_d, RA03_e y RA03_f: acceder a los elementos del documento mediante sus identificadores, modificar su aspecto cuando hay errores, limpiar los campos y mostrar mensajes informativos al usuario. Para ello he creado un formulario similar al del enunciado, una hoja de estilos propia y un conjunto de funciones en JavaScript que se encargan de los controles individuales (“Validar”) y del control total del formulario (“Control total + envío” y “Limpiar los campos”).

2. Estructura del formulario (HTML)

Para esta tarea he creado un documento HTML que contiene toda la estructura del formulario y la referencia a la hoja de estilos y al código de JavaScript. A continuación se describe la estructura completa del archivo HTML explicando cada parte:

2.1 Estructura general del documento

TAREA 3 > TAREA3.html > html > head > link

```
1   <!DOCTYPE html>
2   <html lang="es">
3
4   <head>
5       <meta charset="UTF-8">
6       <title>Formulario de validación - Alberto</title>
7       <link rel="stylesheet" href="estilos_formulario.css">
8   </head>
```

Explicación:

- <!DOCTYPE html> indica que el documento usa HTML5.
- <html lang="es"> define el elemento raíz y especifica que el contenido está en español (lang="es"), algo importante para accesibilidad y SEO.
- Dentro de <head>:
  - <meta charset="UTF-8"> establece la codificación de caracteres a UTF-8, evitando problemas con tildes y caracteres especiales.
  - <title>Formulario de validación - Alberto</title> define el título que aparecerá en la pestaña del navegador.
  - <link rel="stylesheet" href="estilos_formulario.css"> enlaza la hoja de estilos externa donde he definido el diseño del formulario.

2.2 Elementos principales dentro de <body>

El cuerpo del documento contiene tres bloques fundamentales:

1. Un <canvas> para el fondo animado.
2. El contenedor central con el formulario.
3. El <footer> con mi nombre

```
<body>
    <canvas id="bgCanvas"></canvas>

    <div class="page-wrapper">
        <main class="form-card">
            <h1>FORMULARIO</h1>

            <form id="formulario" autocomplete="off" action="#">
                <div class="field-row">
                    <label class="field-label" for="email">Email</label>
                    <div class="field-input">
                        <input type="text" id="email" name="email" class="campo" placeholder="Ej: pepe@foc.es">
                    </div>
                    <button type="button" class="btn-small" onclick="validarEmail()">Validar</button>
                </div>

                <div class="field-row">
                    <label class="field-label" for="contrasena">Contraseña</label>
                    <div class="field-input">
                        <input type="password" id="contrasena" name="contrasena" class="campo"
                            placeholder="Ej: Foc12345">
                    </div>
                    <button type="button" class="btn-small" onclick="validarContrasena()">Validar</button>
                </div>

                <div class="field-row">
                    <label class="field-label" for="usuario">Nombre de usuario</label>
                    <div class="field-input">
                        <input type="text" id="usuario" name="usuario" class="campo" placeholder="Ej: ALB">
                    </div>
                    <button type="button" class="btn-small" onclick="validarUsuario()">Validar</button>
                </div>

                <div class="field-row">
                    <label class="field-label" for="nombre">Nombre y apellidos</label>
                    <div class="field-input">
                        <input type="text" id="nombre" name="nombre" class="campo" placeholder="Ej: Alberto Gálvez">
                    </div>
                    <button type="button" class="btn-small" onclick="validarNombre()">Validar</button>
                </div>

                <div class="field-row field-row-buttons">
                    <button type="button" class="btn-primary" onclick="controlTotal()">
                        Control total + envío
                    </button>
                    <button type="button" class="btn-secondary" onclick="limpiarCampos()">
                        Limpiar los campos
                    </button>
                </div>
            </form>
        </main>
    </div>

    <footer>
        ALBERTO GALVEZ AGUADO 2025
    </footer>
```

View 1 edited file

- **<canvas id="bgCanvas"></canvas>:**
  Es un lienzo HTML donde, mediante JavaScript, se dibuja la animación de partículas que se mueven con el ratón. Forma parte del diseño visual de la página, pero no del formulario en sí. El id="bgCanvas" sirve para acceder a este canvas desde JavaScript

- ** :**
  Contenedor principal que se utiliza para centrar el formulario vertical y horizontalmente en la pantalla. La clase page-wrapper se trabaja en el CSS
- **<main class="form-card">:**
  Elemento semántico que indica el contenido principal de la página. Lleva la clase form-card, que define la “tarjeta” blanca con sombra y bordes redondeados donde se sitúa el formulario
- **<h1>FORMULARIO</h1>:**
  Título principal del formulario, visible para el usuario en la parte superior de la tarjeta.
- <form id="formulario" autocomplete="off" action="#">:
  - id=”formulario” permite identificar el formulario si fuera necesario desde JavaScript
  - autocomplete=”off” desactiva el autocompletado del navegador en los campos, para que durante las pruebas se vea la validación desde cero
  - action=”#” indica que el formulario no se envía a niguna página del servidor; todo el procesamiento se realiza en el lado del cliente con JavaScript
- **<footer>ALBERTO GALVEZ AGUADO 2025</footer>:**
  Pie de página fijo donde se muestra mi nombre
- **<script>…</script>**
  Bloque de código donde se incluye todo el JS que controla las validaciones del formulario y el fondo animado. Se sitúa al final del <body> para que el HTML esté cargado cuando se ejecuten los scripts.

2.3 Animación de las partículas

Además de los requisitos mínimos del enunciado, he añadido un fondo animado con partículas que se mueven en función de la posición del ratón. Este efecto visual no forma parte obligatoria de la tarea, pero lo he incluido como mejora estética para que la página se parezca más a una web profesional actual.

La idea de la animación de partículas está inspirada en la página web del IDE Antigravity, que es el entorno que he utilizado para desarrollar y probar esta práctica. En la página principal de Antigravity aparece un fondo interactivo con puntos y líneas que reaccionan al movimiento del ratón, y he intentado recrear un efecto similar adaptado a mi proyecto.

Para conseguirlo, he utilizado un elemento <canvas> con id="bgCanvas" colocado detrás del formulario, y sobre él dibujo las partículas mediante JavaScript. Cada partícula tiene una posición, una velocidad y un tamaño base, y en cada fotograma se actualiza su posición.

Además, añado un “efecto de repulsión” respecto a la posición del ratón, de modo que cuando el usuario mueve el cursor, los puntos se desplazan y cambian ligeramente de tamaño y color, creando una sensación de movimiento fluido.

Aunque el fondo animado no interviene en la validación de los campos del formulario, sí demuestra el uso combinado de HTML, CSS y JavaScript para mejorar la experiencia visual del usuario. También refleja que he aprovechado el propio entorno Antigravity no solo como IDE para escribir el código, sino también como referencia estética para el diseño final de la página.

2.4 Estructura de las filas del formulario

Dentro del <form> he organizado cada campo en una fila con tres elementos:

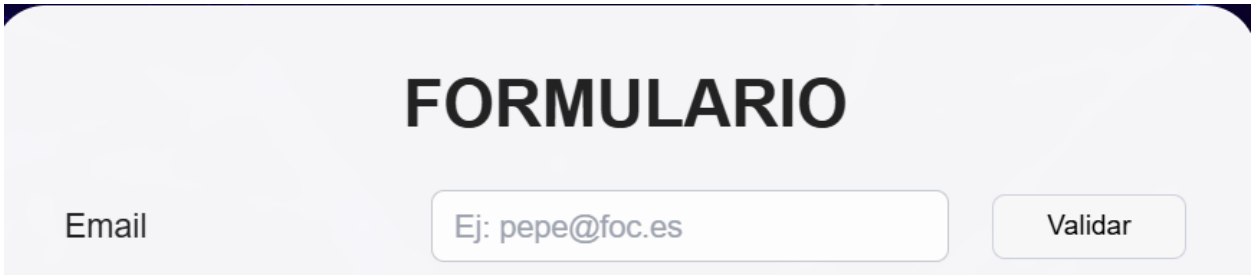
- Etiqueta (<label>)
- Campo de entrada (<input>)
- Botón de validación (<button>)

Cada fila completa está envuelta en un .

```
<canvas id="bgCanvas"></canvas>

<div class="page-wrapper">
    <main class="form-card">
        <h1>FORMULARIO</h1>

        <form id="formulario" autocomplete="off" action="#">
            <div class="field-row">
                <label class="field-label" for="email">Email</label>
                <div class="field-input">
                    <input type="text" id="email" name="email" class="campo" placeholder="Ej: pepe@foc.es">
                </div>
                <button type="button" class="btn-small" onclick="validarEmail()">Validar</button>
            </div>

            <div class="field-row">
                <label class="field-label" for="contrasena">Contraseña</label>
                <div class="field-input">
                    <input type="password" id="contrasena" name="contrasena" class="campo"
                        placeholder="Ej: Foc12345">
                </div>
                <button type="button" class="btn-small" onclick="validarContrasena()">Validar</button>
            </div>

            <div class="field-row">
                <label class="field-label" for="usuario">Nombre de usuario</label>
                <div class="field-input">
                    <input type="text" id="usuario" name="usuario" class="campo" placeholder="Ej: ALB">
                </div>
                <button type="button" class="btn-small" onclick="validarUsuario()">Validar</button>
            </div>

            <div class="field-row">
                <label class="field-label" for="nombre">Nombre y apellidos</label>
                <div class="field-input">
                    <input type="text" id="nombre" name="nombre" class="campo" placeholder="Ej: Alberto Gálvez">
                </div>
                <button type="button" class="btn-small" onclick="validarNombre()">Validar</button>
            </div>

            <div class="field-row field-row-buttons">
                <button type="button" class="btn-primary" onclick="controlTotal()">
                    Control total + envío
                </button>
                <button type="button" class="btn-secondary" onclick="limpiarCampos()">
                    Limpiar los campos
                </button>
            </div>
        </form>
    </main>
</div>

<footer>
    ALBERTO GALVEZ AGUADO 2025
</footer>
```

2.5 Campo Email

```
<form id="formulario" autocomplete="off" action="#">
    <div class="field-row">
        <label class="field-label" for="email">Email</label>
        <div class="field-input">
            <input type="text" id="email" name="email" class="campo" placeholder="Ej: pepe@foc.es">
        </div>
        <button type="button" class="btn-small" onclick="validarEmail()">Validar</button>
    </div>
```



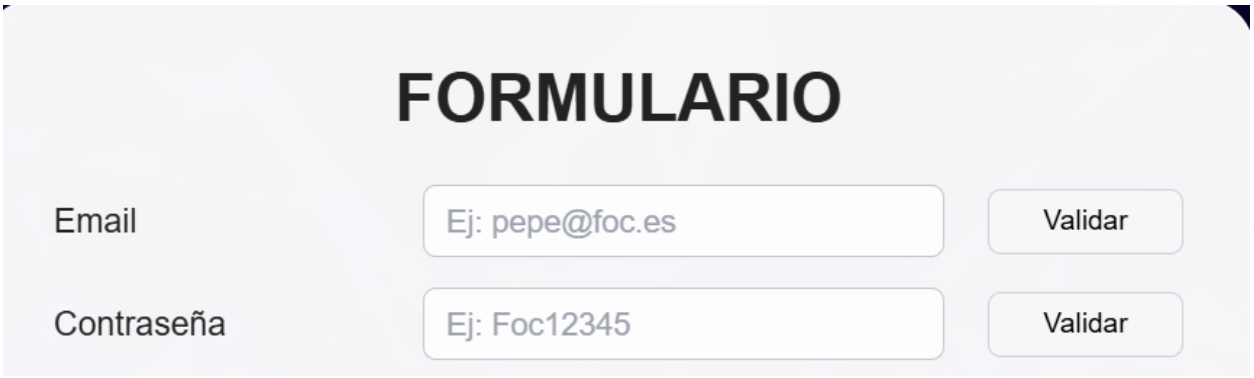
Explicación:

- :
  Indica una fila de formulario. En el CSS, .field-row se convierte en un grid de tres columnas: etiqueta, campo y botón.
- <label class="field-label" for="email">Email</label>:
  - El atributo for="email" vincula la etiqueta con el input cuyo id es email.
  - La clase field-label se usa en el CSS para dar formato uniforme a todas las etiquetas.
- :
  Contenedor del input. Sirve para que el campo se adapte bien al espacio de su columna en el grid.
- <input type="text" id="email" name="email" class="campo" placeholder="Ej: pepe@foc.es">:
  - type="text" indica que es un campo de texto genérico (no utilizo type="email" a propósito, para que la validación se haga solo con JavaScript, como indica la tarea).
  - id="email" permite acceder al campo desde JavaScript con document.getElementById("email").
  - name="email" identifica el campo a nivel de formulario.
  - class="campo" aplica los estilos comunes de los inputs (bordes, relleno, etc.).
  - placeholder="Ej: pepe@foc.es" muestra un ejemplo dentro del campo, para guiar al usuario.
- <button type="button" class="btn-small" onclick="validarEmail()">Validar</button>:
  - type="button" hace que el botón no intente enviar el formulario, solo ejecuta código JavaScript.
  - class="btn-small" aplica el estilo de botón pequeño de validación.
  - onclick="validarEmail()" hace que, al pulsar el botón, se llame a la función JavaScript validarEmail(), que comprueba el email y muestra el mensaje

correspondiente.

2.6 Campo Nombre de usuario

```
<div class="field-row">
    <label class="field-label" for="contrasena">Contraseña</label>
    <div class="field-input">
        <input type="password" id="contrasena" name="contrasena" class="campo"
            placeholder="Ej: Foc12345">
    </div>
    <button type="button" class="btn-small" onclick="validarContrasena()">Validar</button>
</div>
```

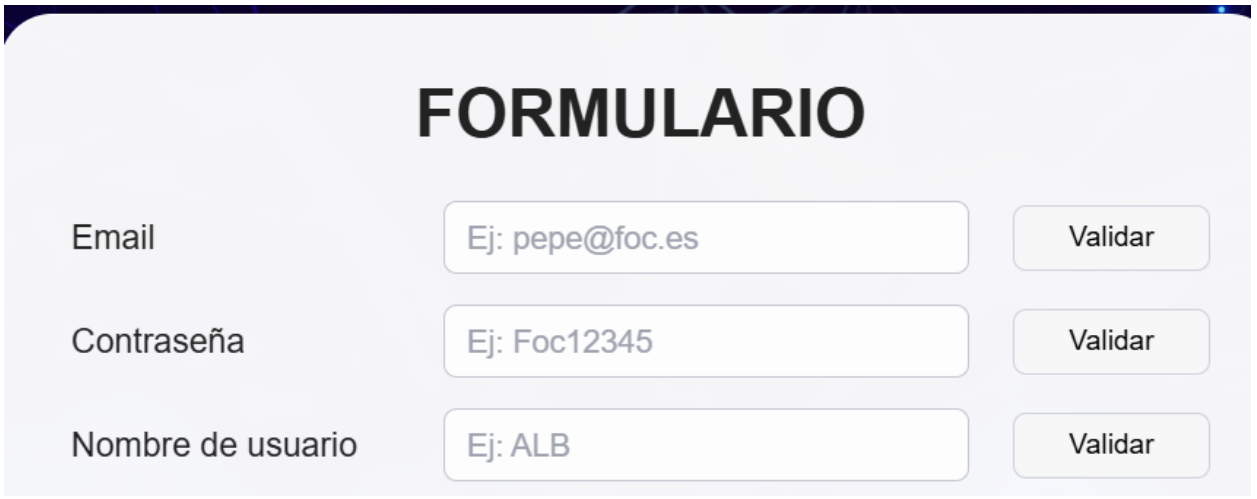


Explicación:

- Se mantiene la misma estructura de fila: field-row con label, input y botón.
- El input usa type="password" para que los caracteres no se vean en claro.
- id="contrasena" y name="contrasena" permiten identificar este campo.
- El placeholder sugiere una contraseña con mayúsculas, minúsculas y números, con un ejemplo
- El botón ejecuta validarContrasena() cuando se pulsa, llamando a la función JavaScript de validación correspondiente.

2.7 Campo nombre de usuario

```html
<div class="field-row">
    <label class="field-label" for="usuario">Nombre de usuario</label>
    <div class="field-input">
        <input type="text" id="usuario" name="usuario" class="campo" placeholder="Ej: ALB">
    </div>
    <button type="button" class="btn-small" onclick="validarUsuario()">Validar</button>
</div>
```



Explicación:

- Sigue la misma estructura que las filas anteriores.
- id="usuario" y name="usuario" identifican este campo.
- El placeholder="Ej: ALB" muestra un ejemplo corto, ya que en la validación solo exijo un mínimo de 3 caracteres.
- El botón lanza la función validarUsuario().

2.8 Campo Nombre y apellidos

```
Ctrl+I for Command, Ctrl+L for Agent
        <div class="field-row">
            <label class="field-label" for="nombre">Nombre y apellidos</label>
            <div class="field-input">
                <input type="text" id="nombre" name="nombre" class="campo" placeholder="Ej: Alberto Gálvez">
            </div>
            <button type="button" class="btn-small" onclick="validarNombre()">Validar</button>
        </div>
```

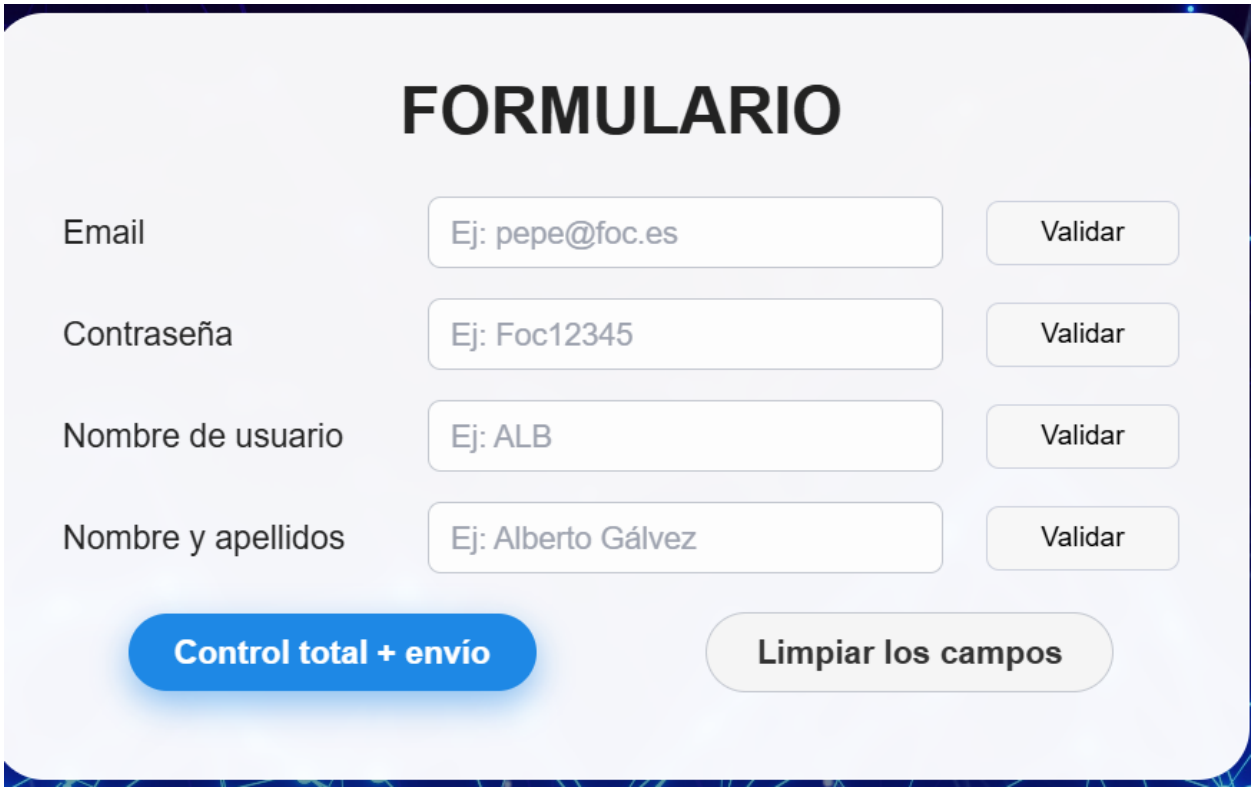
Explicación:

- Misma estructura de fila.
- id="nombre" y name="nombre" se utilizan para acceder al valor y aplicar estilos de error.
- El placeholder muestra un ejemplo realista con nombre y apellido.
- El botón lanza validarNombre(), que comprueba que el campo no esté vacío y que contenga al menos un espacio (nombre + apellido).

2.9 Fila de botones principales: Control total + Limpiar campos

Al final del formulario he añadido una fila para los botones que actúan sobre todos los campos:

```
<div class="field-row field-row-buttons">
    <button type="button" class="btn-primary" onclick="controlTotal()">
        Control total + envío
    </button>
    <button type="button" class="btn-secondary" onclick="limpiarCampos()">
        Limpiar los campos
    </button>
</div>
```



Explicación:

- :
  - Mantiene la clase field-row para seguir usando el grid.
  - Añade la clase field-row-buttons para cambiar el comportamiento del grid en el CSS (en lugar de 3 columnas, aquí usamos 2 columnas grandes para los dos botones).
- <button type="button" class="btn-primary" onclick="controlTotal()">:
  - Botón principal que ejecuta la función controlTotal().
  - Esta función llama internamente a todas las funciones de comprobación (email, contraseña, usuario, nombre) y muestra un mensaje con todos los errores o un mensaje de éxito si todo está correcto.
  - class="btn-primary" le da un estilo más destacado (color azul, sombra, etc.).
- <button type="button" class="btn-secondary" onclick="limpiarCampos()">:

- Botón secundario que llama a limpiarCampos().
- Esta función borra los valores de los inputs y elimina la clase errores para devolver los campos a su aspecto normal.
- class="btn-secondary" aplica otro estilo, más neutro.

3. Hoja de estilos (CSS)

Para dar al formulario un aspecto más profesional y que se parezca a una página web real, he creado una hoja de estilos externa llamada estilos_formularios_css. En esta hoja defino tanto el fondo de la página como la maquetación del formulario, los campos de texto, los botones y el estilo especial para los errores.
A continuación, detallo las partes más importantes de la hoja de estilos.

3.1 Estilos generales del body y del fondo

En primer lugar, he definido el estilo general del body, que establece la fuente, el alto mínimo de la página y el fondo con una imagen a pantalla completa;

Explicación:

- margin: 0; elimina los márgenes por defecto del navegador.
- font-family: Arial, Helvetica, sans-serif; define una fuente sans-serif legible y estándar para todo el documento.
- min-height: 100vh; hace que el cuerpo ocupe como mínimo el alto completo de la ventana del navegador.
- display: flex; flex-direction: column; permite organizar el contenido del body en una columna y centrar fácilmente el formulario mediante otro contenedor.
- background: #000 url("imagenfondo.jpg") no-repeat center center fixed;:
  - Establece un color de fondo negro por defecto (#000) y, encima, una imagen llamada imagenfondo.jpg.
  - no-repeat evita que la imagen se repita.
  - center center centra la imagen.
  - fixed hace que el fondo quede fijo mientras el contenido se desplaza.
- background-size: cover; hace que la imagen de fondo cubra toda la pantalla, recortándose si es necesario.
- color: #222222; define el color de texto por defecto (gris oscuro) para el contenido.

3.2 Canvas de partículas y centrado del formulario

Después defino el estilo del <canvas> donde dibujo la animación de partículas y el contenedor que centra el formulario. (Para hacer la animación de partículas he usado la ayuda de Chatgpt 5.1).

```
12
13  #bgCanvas {
14      position: fixed;
15      inset: 0;
16      z-index: -1;
17      pointer-events: none;
18  }
19
20
21  .page-wrapper {
22      flex: 1;
23      display: flex;
24      justify-content: center;
25      align-items: center;
26      padding: 20px;
27  }
```

Explicación:

- #bgCanvas:
  - position: fixed; hace que el canvas esté siempre fijo respecto a la ventana, ocupando toda la pantalla.
  - inset: 0; es equivalente a poner top: 0; right: 0; bottom: 0; left: 0;, es decir, el canvas ocupa todo el espacio.
  - z-index: -1; sitúa el canvas por detrás del resto de elementos de la página, de forma que el formulario queda “por encima” del fondo animado.
  - pointer-events: none; evita que el canvas interfiera con los clics del usuario (los clics pasan a los elementos que hay encima, como el formulario).
- .page-wrapper:
  - flex: 1; hace que este contenedor ocupe el espacio disponible entre el encabezado (si lo hubiera) y el footer.
  - display: flex; justify-content: center; align-items: center; centra el contenido tanto horizontal como verticalmente.
  - padding: 20px; añade un pequeño margen interior para que en pantallas más pequeñas el formulario no quede pegado al borde.

3.3 Tarjeta del formulario (.form-card) y título

```
29
30  .form-card {
31      width: 520px;
32      max-width: 100%;
33      background: rgba(255, 255, 255, 0.96);
34      border-radius: 24px;
35      padding: 28px 32px 26px 32px;
36      box-shadow: 0 20px 45px rgba(0, 0, 0, 0.45);
37      border: 1px solid rgba(255, 255, 255, 0.85);
38      backdrop-filter: blur(4px);
39  }
40
41  .form-card h1 {
42      margin: 0 0 22px 0;
43      font-size: 1.9rem;
44      text-align: center;
45  }
```

Explicación:

- .form-card:
  - width: 520px; max-width: 100%; fija un ancho base de 520 píxeles, pero permite que en pantallas pequeñas se reduzca hasta ocupar el 100% del ancho disponible.
  - background: rgba(255, 255, 255, 0.96); utiliza un fondo blanco casi opaco, dejando ver muy ligeramente el fondo de la página.
  - border-radius: 24px; redondea las esquinas de la tarjeta, dándole un aspecto más moderno.
  - padding: 28px 32px 26px 32px; añade espacio interior por todos los lados (arriba, derecha, abajo, izquierda).
  - box-shadow: 0 20px 45px rgba(0, 0, 0, 0.45); crea una sombra pronunciada que hace que el formulario parezca una tarjeta flotante sobre el fondo.
  - border: 1px solid rgba(255, 255, 255, 0.85); añade un borde suave y claro alrededor de la tarjeta.
  - backdrop-filter: blur(4px); aplica un ligero desenfoque al fondo que se ve a través de la tarjeta (en navegadores que soportan esta propiedad), creando un efecto “glassmorphism”.
- .form-card h1:
  - Controla el título “FORMULARIO”: margenes, tamaño y centrado.

3.4 Distribución de cada fila: .field-row, etiquetas y contenedores

El formulario está organizado por filas mediante un grid de 3 columnas: etiqueta, campo y botón de validar

```
47
48  .field-row {
49      display: grid;
50      grid-template-columns: 150px minmax(0, 1fr) auto;
51      /* 3 columnas */
52      column-gap: 20px;
53      /* hueco claro entre input y botón */
54      align-items: center;
55      margin-bottom: 14px;
56  }
57
58  .field-label {
59      font-size: 0.96rem;
60  }
61
62  .field-input {
63      width: 100%;
64  }
```

Explicación:

- .field-row:
  - display: grid; establece un sistema de rejilla para distribuir los elementos de la fila.
  - grid-template-columns: 150px minmax(0, 1fr) auto; define tres columnas:
    - Primera columna de ancho fijo (150px) para la etiqueta.
    - Segunda columna flexible (minmax(0, 1fr)) para el campo de texto, que ocupa todo el espacio disponible.
    - Tercera columna auto para el botón “Validar”.
  - column-gap: 20px; deja un espacio horizontal claro entre la etiqueta, el campo y el botón, evitando que se “pegue” visualmente el input con el botón.
  - align-items: center; alinea verticalmente los elementos de la fila.
  - margin-bottom: 14px; separa cada fila de la siguiente.
- .field-label:
  - Ajusta el tamaño de la fuente de las etiquetas de los campos.
- .field-input:
  - Ocupa todo el ancho de la columna central y actúa como contenedor del <input>.

3.5 Estilos de los campos de texto (.campo)

Todos los inputs del formulario comparten la clase .campo, lo que permite aplicarles el mismo estilo de forma consistente:

```
66
67  .campo {
68      width: 100%;
69      box-sizing: border-box;
70      padding: 7px 10px;
71      border-radius: 6px;
72      border: 1px solid #c4c8d0;
73      font-size: 0.95rem;
74      transition: border-color 0.2s, box-shadow 0.2s, background-color 0.2s;
75      background-color: #fdfdfd;
76      margin-right: 0;
77
78  }
79
80  .campo::placeholder {
81      color: #9ea3af;
82      font-size: 0.9rem;
83  }
84
85  .campo:focus {
86      outline: none;
87      border-color: #1e88e5;
88      box-shadow: 0 0 0 2px rgba(30, 136, 229, 0.25);
89      background-color: #f5f9ff;
90  }
```

Explicación:

- width: 100%; hace que el campo use el ancho completo de su contenedor .field-input.
- box-sizing: border-box; asegura que padding y borde se incluyan dentro del ancho total, evitando que el input “rompa” el layout cuando cambia el borde (por ejemplo, al marcar un error).
- padding: 7px 10px; añade espacio interno para que el texto no pegue con el borde.
- border-radius: 6px; redondea las esquinas del campo.
- border: 1px solid #c4c8d0; define un borde gris claro.
- transition: ... añade una transición suave cuando cambian el color del borde, la sombra o el fondo (por ejemplo, al enfocar o al marcar error).
- background-color: #fdfdfd; establece un fondo ligeramente gris claro.
- .campo::placeholder:
  - Define el color y el tamaño del texto del placeholder (los ejemplos “Ej: pepe@foc.es”, etc.).
- .campo:focus:
  - outline: none; elimina el contorno azul por defecto del navegador.
  - border-color: #1e88e5; resalta el campo con un borde azul al recibir el foco.
  - box-shadow: 0 0 0 2px rgba(30, 136, 229, 0.25); añade una sombra suave alrededor, como realce.

- background-color: #f5f9ff; cambia ligeramente el tono de fondo para indicar que el campo está activo.

3.6 Estilos de los botones

El formulario tiene dos tipos de botones: los pequeños de Validar junto a cada campo y los dos botones grandes de la parte inferior

```
120
121   .btn-primary,
122 v .btn-secondary {
123       padding: 9px 18px;
124       border-radius: 999px;
125       font-size: 0.95rem;
126       cursor: pointer;
127       font-weight: 600;
128       border: none;
129       transition: background-color 0.2s, box-shadow 0.2s, transform 0.05s;
130       justify-self: center;
131       min-width: 190px;
132   }
133
134 v .btn-primary {
135       background-color: #1e88e5;
136       color: #ffffff;
137       box-shadow: 0 6px 16px rgba(30, 136, 229, 0.5);
138   }
139
140 v .btn-primary:hover {
141       background-color: #1565c0;
142       box-shadow: 0 8px 20px rgba(21, 101, 192, 0.6);
143       transform: translateY(-1px);
144   }
145
146 v .btn-secondary {
147       background-color: #f5f5f5;
148       color: #333333;
149       border: 1px solid #c4c8d0;
150   }
151
152 v .btn-secondary:hover {
153       background-color: #e6e6e6;
154   }
155
156       Ctrl+I for Command, Ctrl+L for Agent
157 v .campo.errores {
158       border: 2px solid #e53935;
159       background-color: #ffebee;
160       box-shadow: 0 0 0 2px rgba(229, 57, 53, 0.2);
161   }
```

Explicación:

- .btn-small:
  - Botones pequeños “Validar”.
  - min-width: 90px; garantiza que todos tengan el mismo ancho mínimo y no queden más estrechos que el texto.
  - border-radius: 6px;, background-color: #f7f7f7; les da un aspecto discreto que no compite con los botones principales.
  - El :hover añade un pequeño cambio de color y un ligero movimiento hacia arriba.
- .field-row.field-row-buttons:

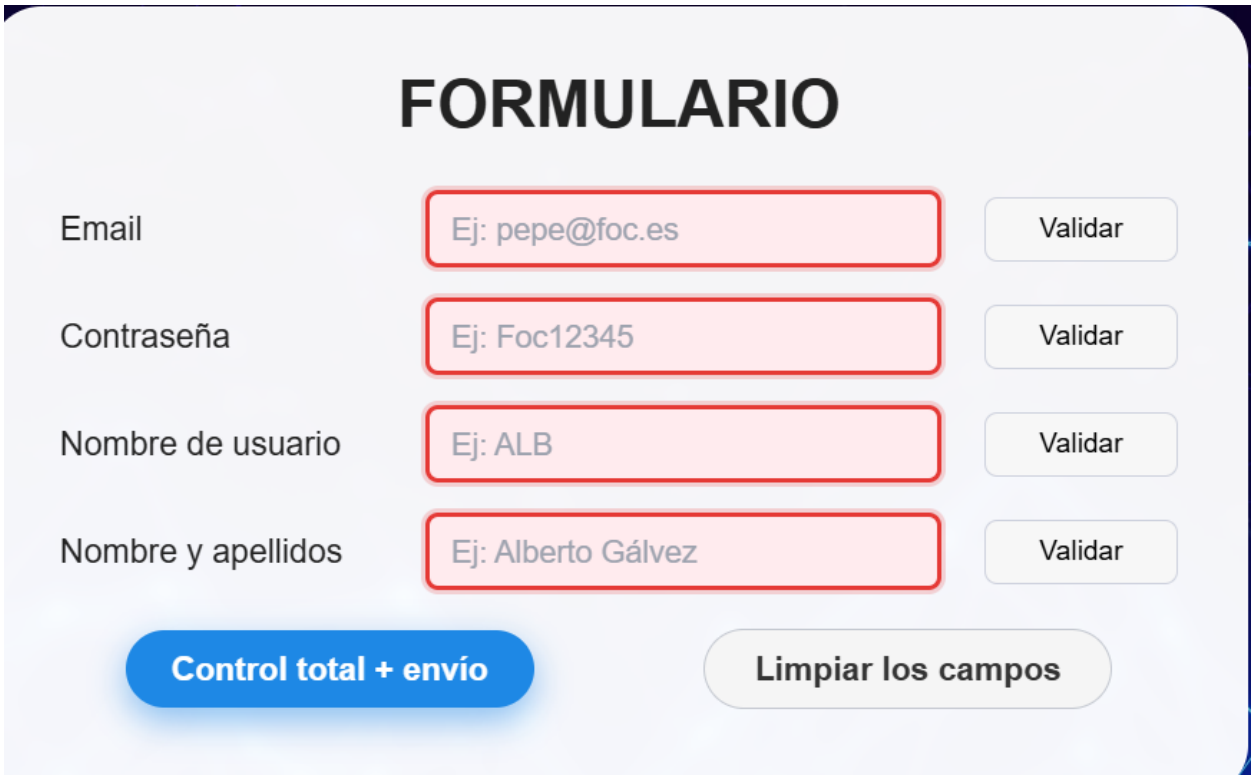
- Modifica la rejilla solo para la fila de los botones grandes: en lugar de 3 columnas hay 2, una para cada botón.

- .btn-primary:
  - Botón principal “Control total + envío”: color azul, texto blanco y sombra más fuerte.
- .btn-secondary:
  - Botón “Limpiar los campos”, con un estilo más neutro (fondo gris claro, borde).

3.8 Estilo de error en los campos (.campo.errores) RA03_f

Para cumplir específicamente el RA03_f (modificaciones sobre los estilos de un documento web cuando hay un error), he creado un estilo específico que se activa cuando hay un error de validación:

```
56
57  .campo.errores {
58      border: 2px solid #e53935;
59      background-color: #ffebee;
60      box-shadow: 0 0 0 2px rgba(229, 57, 53, 0.2);
61  }
```



Explicación:

- .campo.errores se aplica cuando el input tiene simultáneamente la clase campo y la clase errores.
- border: 2px solid #e53935; resalta el campo con un borde rojo fuerte.
- background-color: #ffebee; cambia el fondo a un tono rosado suave.
- box-shadow: 0 0 0 2px rgba(229, 57, 53, 0.2); añade un halo alrededor del campo para destacar aún más el error.

Este estilo se activa desde JavaScript llamando a la función marcarError(campoId), que sustituye la clase del input por "campo errores" cuando hay un dato incorrecto.

3.9 Estilos del footer

Por último, he definido el estilo del pie de página

```css
footer {
    text-align: center;
    padding: 10px 0 14px 0;
    font-size: 0.85rem;
    color: #f1f5ff;
    text-shadow: 0 1px 3px rgba(0, 0, 0, 0.6);
}
```

Explicación:

- footer:
  - Centra el texto, ajusta el tamaño de la letra y añade una ligera sombra para que se lea bien sobre el fondo.

4. Lógica de validación y control en Javascript

En este apartado describo todo el código JavaScript que he utilizado para controlar la validación de los campos del formulario, mostrar mensajes al usuario, limpiar los datos introducidos y gestionar la animación de partículas de fondo
El código JavaScript está incluido al final del archivo HTML, dentro de una etiqueta <script>, para asegurar que todo el HTML del formulario está cargado antes de ejecutar las funciones.

4.1 Funciones para marcar y limpiar errores

En primer lugar, he creado dos funciones auxiliares que se encargan de aplicar o quitar el estilo de error a un campo concreto:

```
66
67      <script>
68          function marcarError(campoId) {
69              var input = document.getElementById(campoId);
70              input.className = "campo errores";
71          }
72
73          function limpiarError(campoId) {
74              var input = document.getElementById(campoId);
75              input.className = "campo";
76          }
```

Explicación:

- marcarError(campoId):
  - Recibe como parámetro el identificador del campo (por ejemplo "email" o "contrasena").
  - Dentro de la función utilizo document.getElementById(campoId) para recuperar el elemento <input> correspondiente.
  - Asigno className = "campo errores", es decir, el input pasa a tener las clases campo y errores al mismo tiempo.
  - Gracias a esto, en el CSS se aplica el estilo .campo.errores, que muestra el campo con borde rojo y fondo rosado (RA03_f).
- limpiarError(campoId):
  - Realiza la operación inversa: vuelve a dejar la clase del campo como "campo", quitando el aspecto de error.
  - Se utiliza cuando el usuario corrige los datos o cuando se limpian los campos.

4.2 Funciones de validación de cada campo del formulario

Para cada uno de los campos del formulario he definido una función de validación que:

1. Obtiene el valor actual del campo.
2. Comprueba si cumple las condiciones exigidas.
3. Marca o desmarca el campo como error.
4. Devuelve un objeto con el resultado (ok verdadero o falso) y un mensaje descriptivo.

Las funciones son: comprobarEmail, comprobarContrasena, comprobarUsuario y comprobarNombre.

COMPROBAR EMAIL:

```
77
78      function comprobarEmail() {
79          var valor = document.getElementById("email").value.trim();
80          var textoError = "";
81
82          if (valor === "") {
83              textoError = "El email es obligatorio.";
84          } else {
85              var posArroba = valor.indexOf("@");
86              var posPunto = valor.lastIndexOf(".");
87              if (posArroba < 1 || posPunto < posArroba + 2 || posPunto >= valor.length - 1) {
88                  textoError = "El email no tiene un formato válido (ejemplo: pepe@foc.es).";
89              }
90          }
91
92          if (textoError !== "") {
93              marcarError("email");
94              return { ok: false, mensaje: textoError };
95          } else {
96              limpiarError("email");
97              return { ok: true, mensaje: "Email correcto." };
98          }
99      }
00
```

Explicación:

- Uso document.getElementById("email").value.trim() para obtener el valor del campo email y quitar espacios en blanco al principio y al final.
- Si el campo está vacío (valor === ""), el mensaje de error es “El email es obligatorio.”.
- Si no está vacío, compruebo si tiene formato válido:
    - posArroba = valor.indexOf("@") busca la posición del carácter @.
    - posPunto = valor.lastIndexOf(".") busca la posición del último punto ..
    - Compruebo que:
        - posArroba no esté en la primera posición (debe haber algo antes de @).
        - Que haya al menos dos caracteres entre @ y el punto.
        - Que el punto no sea el último carácter (debe haber algo después de .).
- Si alguna de estas condiciones falla, considero que el email no es válido y guardo el texto de error.
- Al final:
    - Si textoError !== "", llamo a marcarError("email") y devuelvo { ok: false, mensaje: textoError }.
    - Si todo está bien, llamo a limpiarError("email") y devuelvo { ok: true, mensaje: "Email correcto." }.

COMPROBAR CONTRASEÑA:

```
100
101     function comprobarContrasena() {
102         var valor = document.getElementById("contrasena").value;
103         var textoError = "";
104
105         if (valor.length < 5) {
106             textoError = "La contraseña debe tener al menos 5 caracteres.";
107         } else {
108             var tieneMayus = /[A-Z]/.test(valor);
109             var tieneMinus = /[a-z]/.test(valor);
110             var tieneNumero = /[0-9]/.test(valor);
111
112             if (!tieneMayus || !tieneMinus || !tieneNumero) {
113                 textoError = "La contraseña debe tener 1 mayúscula, 1 minúscula y 1 número.";
114             }
115         }
116
117         if (textoError !== "") {
118             marcarError("contrasena");
119             return { ok: false, mensaje: textoError };
120         } else {
121             limpiarError("contrasena");
122             return { ok: true, mensaje: "Contraseña correcta." };
123         }
124     }
```

Explicación:

- Recupero el valor del campo contraseña sin aplicar trim (porque los espacios podrían formar parte de la contraseña, aunque normalmente no se usan).
- Compruebo dos tipos de condiciones:
    1. Longitud mínima:
        - Si valor.length < 5, la contraseña es demasiado corta.
    2. Complejidad:
        - tieneMayus = /[A-Z]/.test(valor); comprueba si hay al menos una letra mayúscula.
        - tieneMinus = /[a-z]/.test(valor); comprueba si hay al menos una letra minúscula.
        - tieneNumero = /[0-9]/.test(valor); comprueba si hay al menos un número.
        - Si falta cualquiera de estos tres tipos de caracteres, se genera un mensaje de error.
- Igual que en el caso del email, si hay error, marco el campo (marcarError("contrasena")) y devuelvo ok: false. Si todo está correcto, limpio el error y devuelvo ok: true.

COMPROBAR USUARIO:

```
function comprobarUsuario() {
    var valor = document.getElementById("usuario").value.trim();
    var textoError = "";

    if (valor.length < 3) {
        textoError = "El nombre de usuario debe tener tres o más caracteres.";
    }

    if (textoError !== "") {
        marcarError("usuario");
        return { ok: false, mensaje: textoError };
    } else {
        limpiarError("usuario");
        return { ok: true, mensaje: "Usuario correcto." };
    }
}
```

Explicación:

- Leo el valor del campo de usuario y le aplico trim() para evitar espacios al principio o al final.
- En esta tarea solo exijo que el nombre de usuario tenga una longitud mínima de 3 caracteres.
- Si la longitud es menor, marco el error y devuelvo un mensaje informativo. Si no, limpio el error.

COMPROBAR NOMBRE

```
}

function comprobarNombre() {
    var valor = document.getElementById("nombre").value.trim();
    var textoError = "";

    if (valor === "") {
        textoError = "El nombre y apellidos son obligatorios.";
    } else if (valor.indexOf(" ") === -1) {
        textoError = "Escribe al menos un nombre y un apellido.";
    }

    if (textoError !== "") {
        marcarError("nombre");
        return { ok: false, mensaje: textoError };
    } else {
        limpiarError("nombre");
        return { ok: true, mensaje: "Nombre y apellidos correctos." };
    }
}
```

Explicación:

- Utilizo de nuevo trim() para evitar que solo haya espacios.
- Primera condición: si el valor está vacío, se informa de que “El nombre y apellidos son obligatorios.”.
- Segunda condición: si el texto no contiene ningún espacio (valor.indexOf(" ") === -1), significa que el usuario solo ha escrito una palabra (por ejemplo, solo el nombre). En ese caso, se pide que escriba al menos nombre y un apellido.
- Según el resultado, aplico el estilo de error o lo limpio.

4.3 Funciones asociadas a los botones Validar

Cada botón “Validar” que aparece a la derecha del campo en el formulario llama a una función JavaScript muy corta, que se encarga de invocar a la función de comprobación correspondiente y mostrar un mensaje con alert.

```
161
162     function validarEmail() {
163         var r = comprobarEmail();
164         alert(r.mensaje);
165     }
166
167     function validarContrasena() {
168         var r = comprobarContrasena();
169         alert(r.mensaje);
170     }
171
172     function validarUsuario() {
173         var r = comprobarUsuario();
174         alert(r.mensaje);
175     }
176
177     function validarNombre() {
178         var r = comprobarNombre();
179         alert(r.mensaje);
180     }
```

Explicación:

- Cada función (validarEmail, validarContrasena, etc.):
  - Llama a la función de comprobación correspondiente (comprobarEmail, comprobarContrasena, etc.).
  - La función de comprobación devuelve un objeto con dos propiedades:
    - ok: indica si la validación ha sido correcta o no.
    - mensaje: contiene el texto que quiero mostrar al usuario.
  - El alert(r.mensaje) muestra un cuadro de diálogo con el resultado de la validación.

Estas funciones están conectadas con el HTML mediante el atributo onclick de cada botón, por ejemplo:

```
<button type="button" class="btn-secondary" onclick="limpiarCampos()">
    Limpiar los campos
</button>
```

4.4 Función de control total del formulario

El botón “Control total + envío” no valida solo un campo, sino que revisa todos a la vez y muestra un resumen de errores.

```
function controlTotal() {
    var errores = "";

    var rEmail = comprobarEmail();
    if (!rEmail.ok) errores += "- " + rEmail.mensaje + "\n";

    var rPass = comprobarContrasena();
    if (!rPass.ok) errores += "- " + rPass.mensaje + "\n";

    var rUser = comprobarUsuario();
    if (!rUser.ok) errores += "- " + rUser.mensaje + "\n";

    var rNom = comprobarNombre();
    if (!rNom.ok) errores += "- " + rNom.mensaje + "\n";

    if (errores === "") {
        alert("Todos los datos son correctos. Formulario listo para enviar.");
    } else {
        alert("Se han encontrado los siguientes errores:\n\n" + errores);
    }
}
```

Explicación:

- Inicialmente defino una cadena de texto vacía errores = "".
- Llamo a cada función de comprobación:
    - comprobarEmail(), comprobarContrasena(), comprobarUsuario(), comprobarNombre() y guardo el resultado.
- Si alguna de ellas devuelve ok: false, concateno su mensaje de error a la variable errores, añadiendo un guion y un salto de línea para mostrarlos ordenados.
- Al final:
    - Si errores === "", significa que ninguno de los campos ha generado error, de modo que muestro el mensaje “Todos los datos son correctos. Formulario listo para enviar.”.
    - Si errores contiene texto, muestro un alert con todos los errores juntos, cada uno en una línea

Esta función está asociada al botón principal mediante:

```
<div class="field-row field-row-buttons">
    <button type="button" class="btn-primary" onclick="controlTotal()">
        Control total + envío
    </button>
```

Con esto se cumple la parte del enunciado que pide validar todos los campos al pulsar el botón correspondiente y mostrar todos los errores de forma conjunta

4.5 Función para limpiar los campos del formulario (RA03_e)

La tarea también pide un botón que limpie los campos y devuelva los estilos iniciales. Para eso he creado la función limpiarCampos():

```
203
204     function limpiarCampos() {
205         document.getElementById("email").value = "";
206         document.getElementById("contrasena").value = "";
207         document.getElementById("usuario").value = "";
208         document.getElementById("nombre").value = "";
209
210         limpiarError("email");
211         limpiarError("contrasena");
212         limpiarError("usuario");
213         limpiarError("nombre");
214     }
```

Explicación:

- Asigno una cadena vacía ("") a la propiedad value de cada input (email, contrasena, usuario y nombre), borrando así el contenido introducido por el usuario.
- A continuación llamo a limpiarError(...) sobre cada campo, de forma que:
  - Se elimina la clase errores.
  - El campo recupera el estilo visual normal definido por .campo.

Esta función está conectada con el botón secundario del formulario:

```
<button type="button" class="btn-secondary" onclick="limpiarCampos()">
    Limpiar los campos
</button>
```

De esta manera cumplo con el RA03_e, ya que se eliminan los datos del documento y se restauran los estilos delos elementos a su estado inicial

4.6 Código JavaScript de la animación de partículas

Por último, en la parte final del <script> se encuentra el código que controla la animación de partículas en el canvas bgCanvas. Este código crea un conjunto de partículas que se mueven suavemente, rebotan en los límites de la pantalla y reaccionan a la posición del ratón. Esto lo he hecho inspirándome en la página web https://antigravity.google/

```
215
216     var canvas = document.getElementById("bgCanvas");
217     var ctx = canvas.getContext("2d");
218     var ancho, alto;
219     var particulas = [];
220     var numParticulas = 120;
221     var raton = { x: null, y: null };
222
223     function redimensionarCanvas() {
224         ancho = canvas.width = window.innerWidth;
225         alto = canvas.height = window.innerHeight;
226     }
227
228     window.addEventListener("resize", redimensionarCanvas);
229     redimensionarCanvas();
230
231     function crearParticulas() {
232         particulas = [];
233         for (var i = 0; i < numParticulas; i++) {
234             particulas.push({
235                 x: Math.random() * ancho,
236                 y: Math.random() * alto,
237                 vx: (Math.random() - 0.5) * 0.6,
238                 vy: (Math.random() - 0.5) * 0.6,
239                 radioBase: 1.5 + Math.random() * 1.5
240             });
241         }
242     }
243
244     crearParticulas();
245
246     window.addEventListener("mousemove", function (e) {
247         raton.x = e.clientX;
248         raton.y = e.clientY;
249     });
250
251     window.addEventListener("mouseleave", function () {
252         raton.x = null;
253         raton.y = null;
254     });
255
256     function dibujar() {
257         ctx.clearRect(0, 0, ancho, alto);
258
259         for (var i = 0; i < particulas.length; i++) {
260             var p = particulas[i];
261
262             p.x += p.vx;
263             p.y += p.vy;
264
265             if (p.x < 0 || p.x > ancho) p.vx *= -1;
266             if (p.y < 0 || p.y > alto) p.vy *= -1;
267
268             var radio = p.radioBase;
269             var color = "rgba(255, 255, 255, 0.7)";
270
271             if (raton.x !== null) {
272                 var dx = p.x - raton.x;
273                 var dy = p.y - raton.y;
274                 var dist = Math.sqrt(dx * dx + dy * dy);
275                 var radioInfluencia = 200;
276
277                 if (dist < radioInfluencia && dist > 0) {
278                     var fuerza = (radioInfluencia - dist) / radioInfluencia;
279                     p.x += (dx / dist) * fuerza * 3.5;
280                     p.y += (dy / dist) * fuerza * 3.5;
281                     radio = p.radioBase + fuerza * 3;
282                     color = "rgba(144, 202, 249, " + (0.4 + fuerza * 0.6) + ")";
283                 }
284             }
```

```
            ctx.beginPath();
            ctx.arc(p.x, p.y, radio, 0, Math.PI * 2);
            ctx.fillStyle = color;
            ctx.fill();
        }

        var maxLinea = 140;

        for (var a = 0; a < particulas.length; a++) {
            for (var b = a + 1; b < particulas.length; b++) {
                var pa = particulas[a];
                var pb = particulas[b];
                var dx2 = pa.x - pb.x;
                var dy2 = pa.y - pb.y;
                var dist2 = Math.sqrt(dx2 * dx2 + dy2 * dy2);

                if (dist2 < maxLinea) {
                    var alpha = 1 - dist2 / maxLinea;
                    ctx.beginPath();
                    ctx.moveTo(pa.x, pa.y);
                    ctx.lineTo(pb.x, pb.y);
                    ctx.strokeStyle = "rgba(144, 202, 249, " + (alpha * 0.7) + ")";
                    ctx.lineWidth = 1;
                    ctx.stroke();
                }
            }
        }

        requestAnimationFrame(dibujar);
    }

    dibujar();
```

Explicación:

- Recupero el elemento <canvas> mediante getElementById.
- ctx = canvas.getContext("2d"); obtiene el contexto 2D que necesito para dibujar.
- ancho y alto guardarán el tamaño actual del canvas.
- particulas es un array donde almacenaré todas las partículas.
- numParticulas = 120; define cuántas partículas quiero dibujar.
- raton es un objeto donde guardaré las coordenadas actuales del ratón dentro de la ventana.
- Vacío el array particulas y relleno de nuevo con numParticulas objetos.
- Cada partícula tiene:
- x e y: posición inicial aleatoria dentro del canvas.
- vx y vy: velocidad horizontal y vertical, con valores pequeños para que el movimiento sea suave.
- radioBase: tamaño base del círculo que representa la partícula.
- Cuando el ratón se mueve (mousemove), guardo sus coordenadas en el objeto raton.
- Cuando el cursor sale de la ventana (mouseleave), pongo raton.x y raton.y a null para indicar que no hay interacción
- Limpiar el canvas.
- Actualizar la posición de cada partícula
- Modificar su tamaño/color si está cerca del ratón
- Dibujar líneas entre partículas cercanas
- Volver a llamarse a sí misma con requestAnimationFrame
- Explicación
- Movimiento:
- Actualizo la posición con p.x += p.vx; y p.y += p.vy;.
- Si la partícula se sale de los límites del canvas, invierto la velocidad para simular un rebote.
- Interacción con el ratón:

- Si el ratón está activo (raton.x !== null), calculo la distancia entre la partícula y el cursor.
- Si la distancia es menor que un radio de influencia, aplico un pequeño desplazamiento y aumento el radio y el brillo del color para crear el efecto visual de repulsión.
- Líneas entre partículas:
- Recorro todas las parejas de partículas y, si están cerca (dist2 < maxLinea), dibujo una línea entre ellas con una opacidad que depende de la distancia.
- Animación:
- Llamo a requestAnimationFrame(dibujar); para que la función se ejecute continuamente y la animación sea fluida.

Para esta parte de la animación he utilizado la idea visual del IDE Antigravity y la ayuda de ChatGPT 5.1 para construir la lógica del movimiento, adaptándola a mi diseño y a la estructura de mi proyecto.

5. Relación con los resultados de aprendizaje (RA03_c, RA03_d, RA03_e, RA03_f)

En este apartado explico cómo el trabajo realizado en el formulario (HTML + CSS + JavaScript) cumple con cada uno de los resultados de aprendizaje indicados en la tarea: RA03_c, RA03_d, RA03_e y RA03_f.

5.1 RA03_c) Selección y acceso de los diferentes elementos de un documento web

Este resultado de aprendizaje se refiere a la capacidad de localizar y acceder a los elementos del documento desde JavaScript, normalmente utilizando sus identificadores (id).

En mi proyecto cumplo este RA de la siguiente forma:

- Todos los campos del formulario tienen un id único:
  - id="email", id="contrasena", id="usuario", id="nombre", etc.
- Desde JavaScript accedo a esos elementos utilizando document.getElementById(...) en las funciones de validación y de limpieza:
  - document.getElementById("email").value para leer el valor del email.
  - document.getElementById("contrasena").value para la contraseña.
  - document.getElementById("usuario").value para el nombre de usuario.
  - document.getElementById("nombre").value para nombre y apellidos.
- También utilizo document.getElementById(campoId) en las funciones marcarError(campoId) y limpiarError(campoId) para cambiar las clases de cada input según su estado (correcto o con error).

  Explicación:
- He diseñado el formulario teniendo en cuenta desde el principio que cada input debía ser accesible fácilmente desde JavaScript.
- El uso de getElementById es constante en todo el código de validación, lo que demuestra que sé seleccionar los elementos del documento web y trabajar con ellos desde el script.

5.2 RA03_d) Creación y modificación de elementos de documentos web

Este RA está relacionado con la capacidad de crear o modificar el contenido y las propiedades de los elementos del documento.

En mi formulario cumplo este RA principalmente a través de la modificación de:

- El valor de los campos (value) cuando se limpian o cuando el usuario introduce datos.
- Las clases CSS (className) para cambiar su aspecto visual.
- El contenido de los mensajes que se muestran al usuario mediante alert.

Ejemplos concretos:

- En la función limpiarCampos() modifico el contenido de los inputs:
  - document.getElementById("email").value = "";
  - document.getElementById("contrasena").value = "";
  - document.getElementById("usuario").value = "";
  - document.getElementById("nombre").value = "";
- En las funciones de validación (comprobarEmail, comprobarContrasena, etc.) cambio las clases de los elementos:
  - input.className = "campo errores"; cuando hay un error.
  - input.className = "campo"; cuando el dato es correcto y quiero quitar el error.
- En la función controlTotal() construyo dinámicamente un texto con todos los errores encontrados y lo muestro en un alert, modificando así la información que recibe el usuario según los datos introducidos.

Explicación:

- Aunque en esta tarea no creo nuevos nodos HTML desde JavaScript, sí modifico activamente los elementos que ya existen (valores, clases, mensajes).
- Esto demuestra que sé manipular el documento web “en vivo”, adaptando el formulario al estado de los datos que introduce el usuario.

5.3 RA03_e) Eliminación de elementos de documentos web

En el contexto de este ejercicio, el RA03_e se trabaja principalmente a través de la eliminación de los datos introducidos y la restauración del formulario a su estado inicial.

La función clave para este RA es limpiarCampos():

- Deja los valores de todos los inputs vacíos:
  - Elimina el texto que el usuario ha escrito en email, contrasena, usuario y nombre.
- También elimina los estilos de error llamando a limpiarError(...) para cada campo:
  - Con esto, cualquier rastro visual de error (bordes rojos, fondo rosado, etc.) desaparece

Explicación:

- Aunque no borro nodos del DOM de forma literal, sí “elimino” los datos que el usuario ha introducido en el documento web
- Además, elimino el estado de error de los elementos al quitar la clase errores, devolviendo el formulario a su apariencia inicial
- De esta forma, el botón “Limpiar los campos” sirve para dejar el formulario completamente limpio y preparado para una nueva introducción de datos, cumpliendo el espíritu del RA03_e dentro de esta práctica

5.4 RA03_f) Modificación de estilos de un documento web

Este resultado de aprendizaje pide que se realicen cambios de estilo en el documento cuando se cumplan ciertas condiciones, por ejemplo, cuando se produce un error

En mi práctica cumplo este RA de forma clara con:

- La clase CSS .campo.errores, que define el estilo de los inputs cuando hay un error:
  - Borde rojo más grueso
  - Fondo rosado
  - Sombra alrededor del campo para resaltarlo
- Las funciones marcarError(campoId) y limpiarError(campoId), que añaden o eliminan la clase errores a cada campo según el resultado de la validación

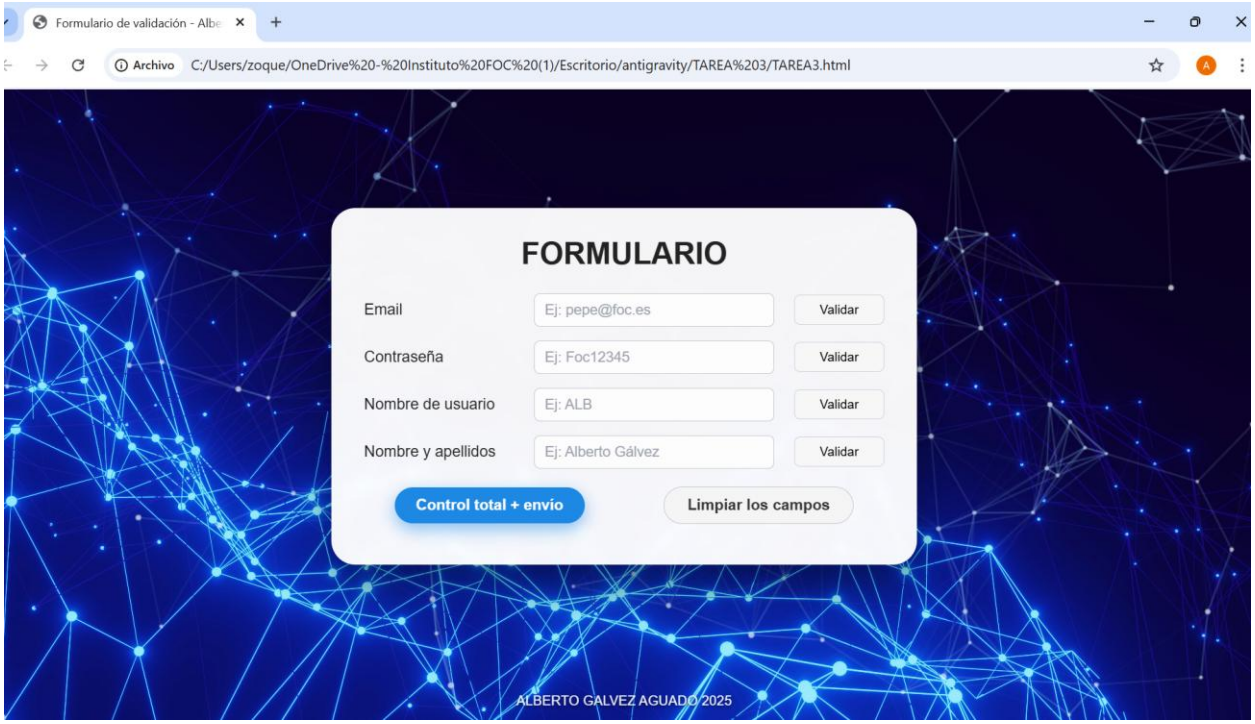
Funcionamiento:

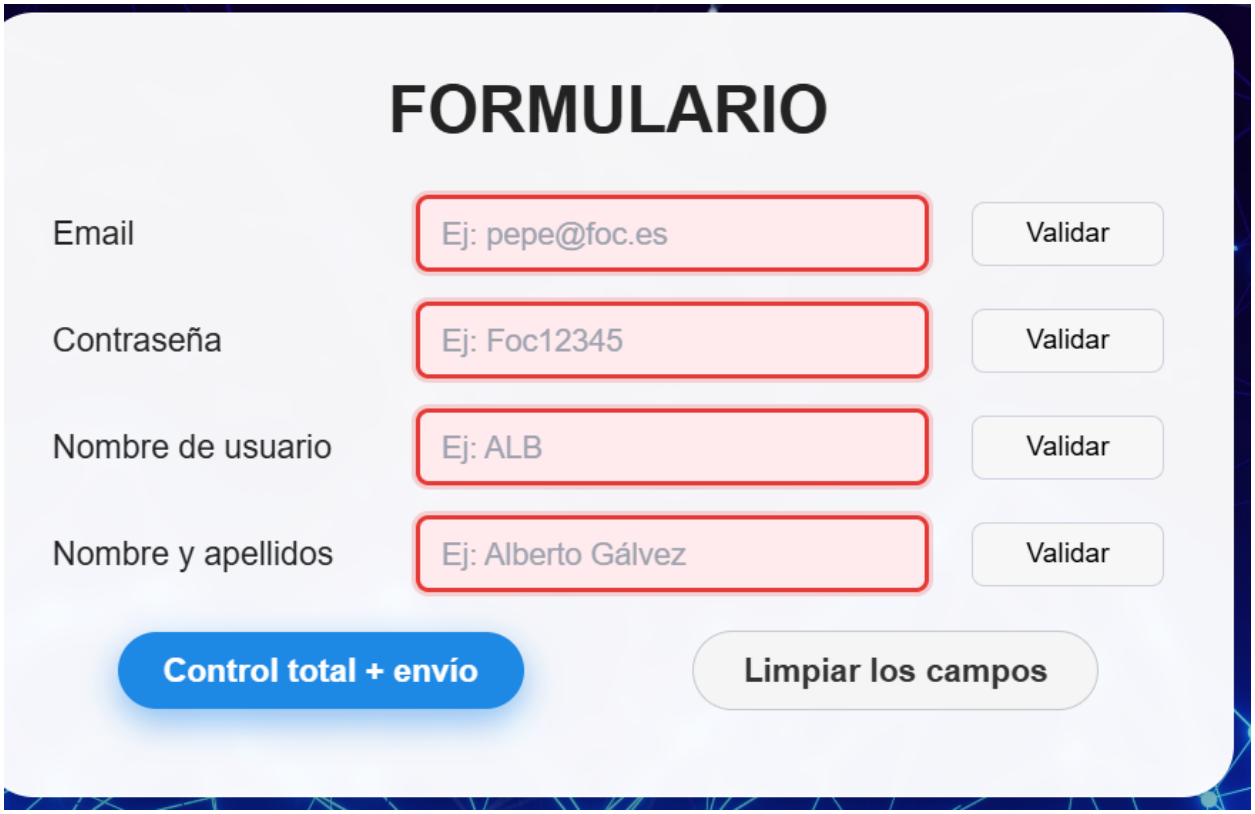
- Si un campo no supera las comprobaciones (por ejemplo, email sin @ o contraseña demasiado corta), la función de validación correspondiente llama a marcarError("idCampo")
- marcarError cambia la clase del input a "campo errores", y automáticamente el CSS aplica el estilo de error:
  - El usuario ve claramente qué campo está incorrecto sin necesidad de que aparezcan mensajes escritos debajo del input
- Cuando el dato es corregido o se limpian los campos, se llama a limpiarError("idCampo") y el campo vuelve a su estilo normal .campo

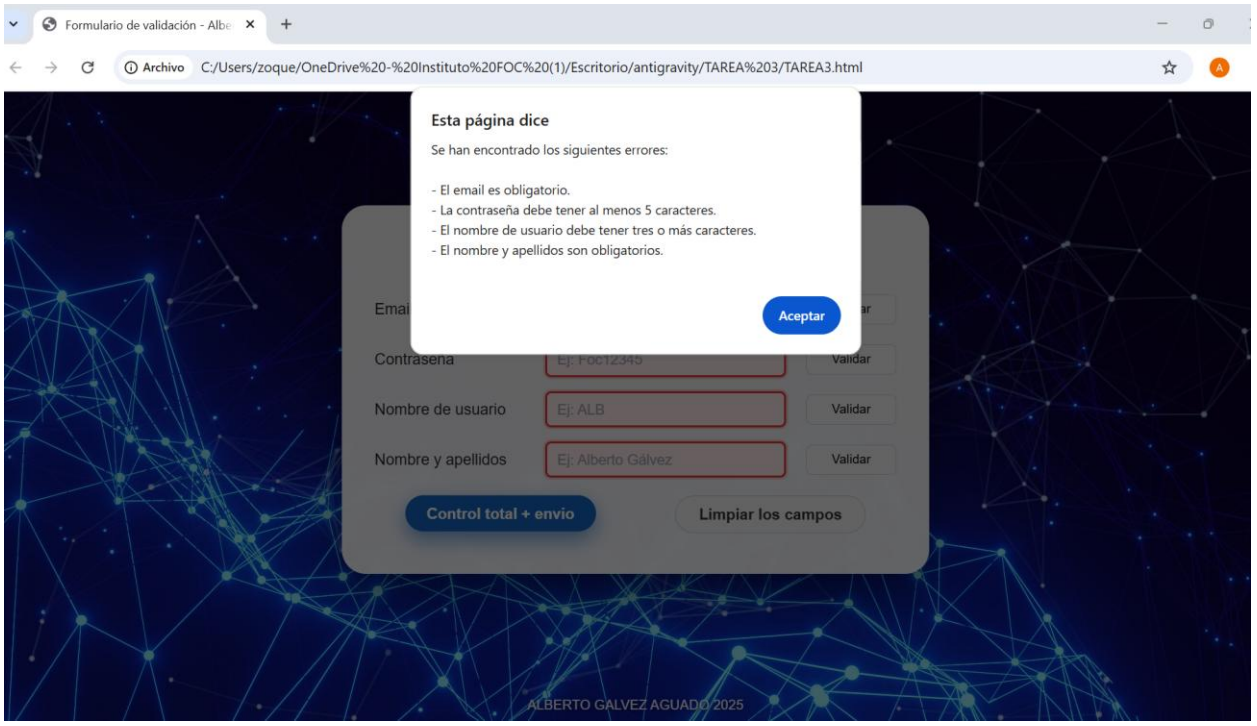
Explicación:

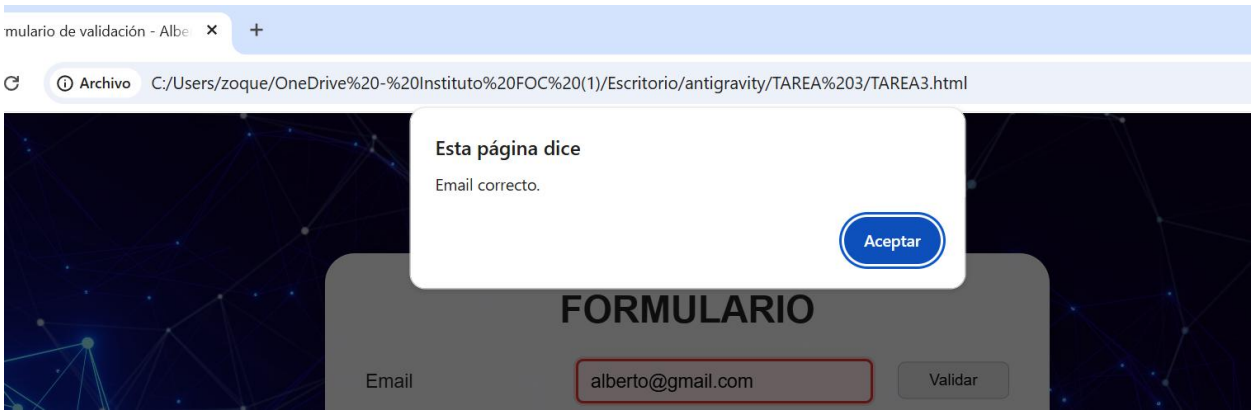
- El RA03_f no solo se cumple de manera “mínima”, sino que lo uso como parte clave de la experiencia visual del formulario
- El cambio de estilo sirve como feedback directo para el usuario, indicando visualmente en qué campo debe corregir los datos
- Esta combinación de CSS (clase .campo.errores) y JavaScript (funciones marcarError y limpiarError) es un ejemplo claro de cómo modificar estilos dinámicamente en función del estado de los datos

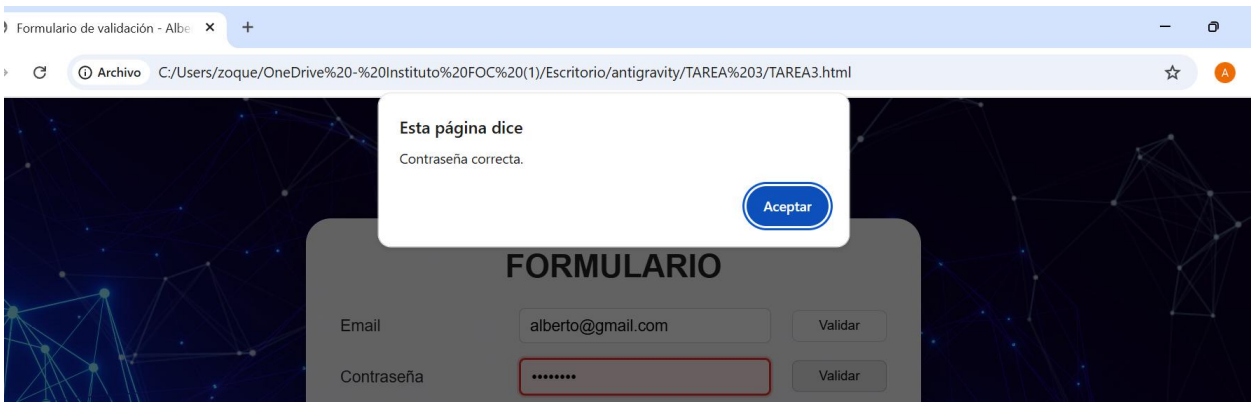
6. Capturas

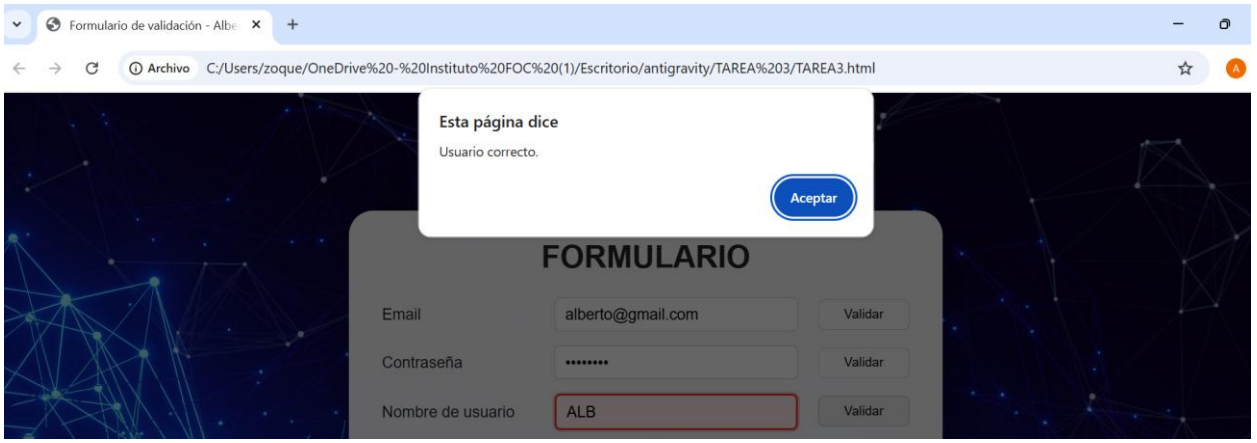

Formulario de validación - Albe
Archivo
C:/Users/zoque/OneDrive%20-%20Instituto%20FOC%20(1)/Escritorio/antigravity/TAREA%203/TAREA3.html
FORMULARIO
Email
Ej: pepe@foc.es
Validar
Contraseña
Ej: Foc12345
Validar
Nombre de usuario
Ej: ALB
Validar
Nombre y apellidos
Ej: Alberto Gálvez
Validar
Control total + envío
Limpiar los campos
ALBERTO GALVEZ AGUADO 2025

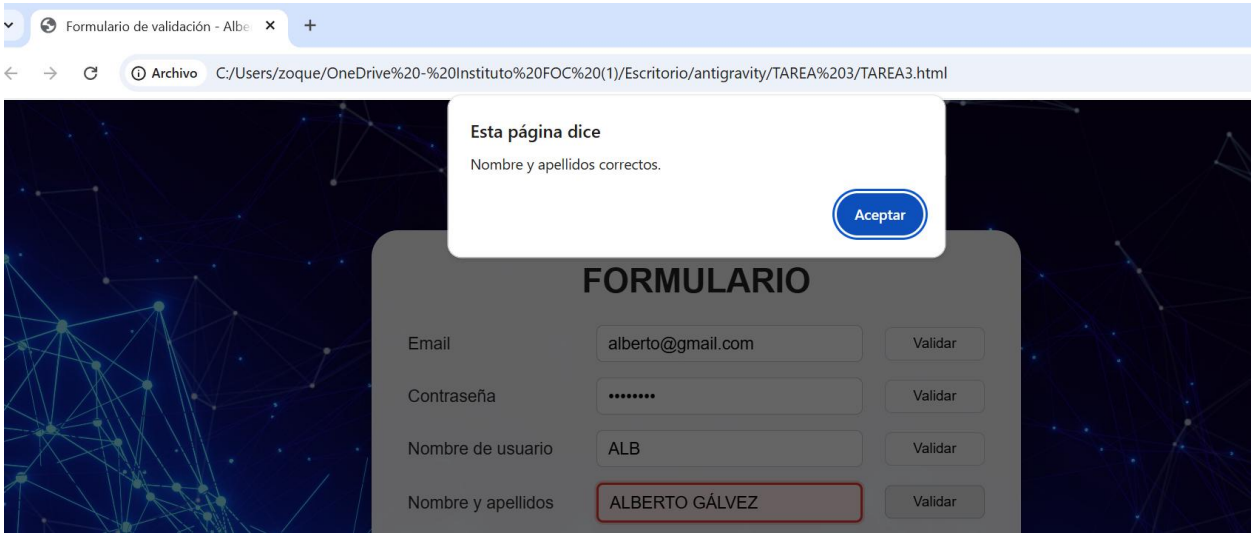

FORMULARIO
Email
Ej: pepe@foc.es
Validar
Contraseña
Ej: Foc12345
Validar
Nombre de usuario
Ej: ALB
Validar
Nombre y apellidos
Ej: Alberto Gálvez
Validar
Control total + envío
Limpiar los campos

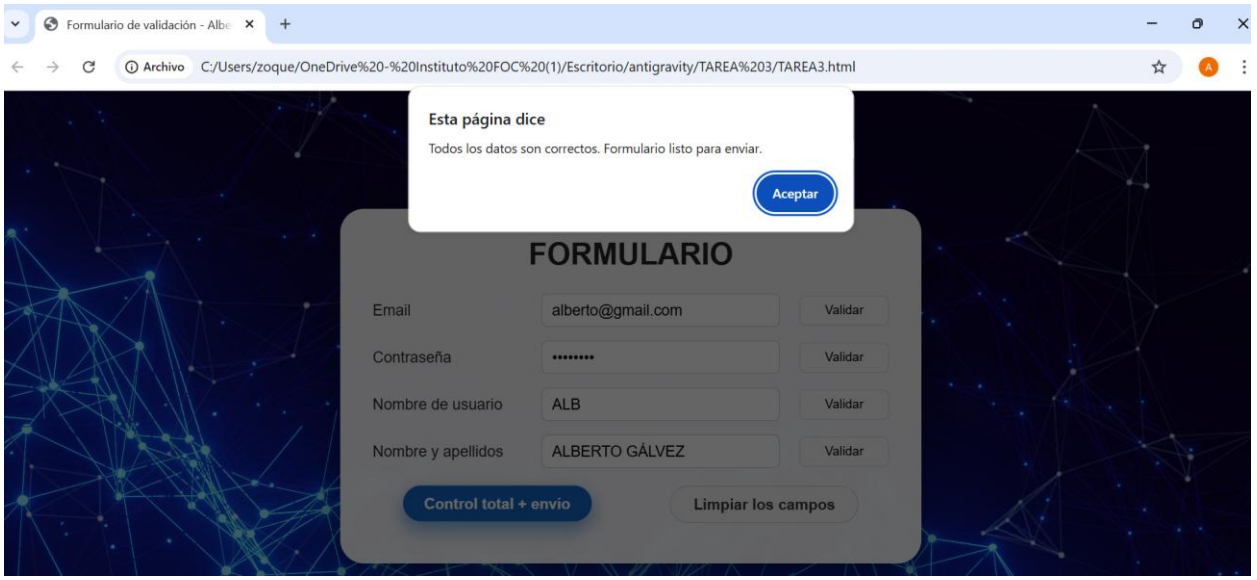

Formulario de validación - Albe
Archivo
C:/Users/zoque/OneDrive%20-%20Instituto%20FOC%20(1)/Escritorio/antigravity/TAREA%203/TAREA3.html
Esta página dice
Se han encontrado los siguientes errores:
- El email es obligatorio.
- La contraseña debe tener al menos 5 caracteres.
- El nombre de usuario debe tener tres o más caracteres.
- El nombre y apellidos son obligatorios.
Aceptar
Contraseña
Nombre de usuario
Ej: ALB
Validar
Nombre y apellidos
Ej: Alberto Gálvez
Control total + envio
Limpiar los campos
ALBERTO GALVEZ AGUADO 2025


mulario de validación - Albe
Archivo
C:/Users/zoque/OneDrive%20-%20Instituto%20FOC%20(1)/Escritorio/antigravity/TAREA%203/TAREA3.html
Esta página dice
Email correcto.
Aceptar
FORMULARIO
Email
alberto@gmail.com
Validar


Formulario de validación - Albe
Archivo
C:/Users/zoque/OneDrive%20-%20Instituto%20FOC%20(1)/Escritorio/antigravity/TAREA%203/TAREA3.html
Esta página dice
Contraseña correcta.
Aceptar
FORMULARIO
Email
alberto@gmail.com
Validar
Contraseña
Validar







En el zip adjuntado, se incluye el .html, el .css, la imagen de fondo y este mismo informe