

## Unidad 4. Gestión de eventos y formularios en JavaScript.

### Boletín de actividades

#### Contenido

Formularios .....	2
1. Fundamentos HTML y CSS.....	2
2. Validación nativa con HTML5.....	4
3. Fundamentos JavaScript. ....	6
4. Acciones comunes de JavaScript sobre formularios. ....	7
5. Validación con JavaScript y la API de validación de HTML5. ....	9
6. Uso de la API de validación.....	11
7. Restricciones complejas .....	13
8. Formularios para practicar .....	14

# Formularios

Los formularios son la forma en la que los usuarios se comunican con las aplicaciones web. Su diseño e implementación están estrechamente ligados a la experiencia del usuario en el sitio. Todos hemos encontrado formularios interminables o aquellos que no se envían y no sabemos por qué. En esta unidad aprenderemos el funcionamiento de los formularios, sus principios de diseño y formas de implementación.

## 1. Fundamentos HTML y CSS.

En esta unidad, además de JavaScript será muy importante conocer el HTML y CSS detrás de cualquier formulario ya que gran parte de su comportamiento será determinado por estos lenguajes. Más aún a partir de la aparición de los nuevos atributos de formularios en HTML5 que permiten validar los campos sin escribir una sola línea de JavaScript.

Para empezar vamos a escribir el HTML y CSS de un formulario y a verificar su comportamiento por defecto. Cómo se toman los datos y se envían a un servidor. En el servidor los datos suelen tratarse en un proceso que implica alguna operación con una base de datos.

### Actividad 1. Instala el servidor.

Para disponer fácilmente de un servidor utiliza el paquete XAMPP que permite instalar Apache, PHP y MariaDB desde un único ejecutable. Puedes descargarlo de la [página oficial de XAMPP](#). Para saber como instalarlo y arrancarlo consulta el [FAQ oficial de XAMPP para Linux](#). Si dispones de Windows el proceso es similar al de cualquier otro programa.

### Actividad 2. Estructura el formulario.

Implementa el siguiente formulario de registro en HTML. Puedes ver un ejemplo en el [artículo HTML Forms de W3Schools](#). Presta especial atención a la etiqueta label.

### Formulario de registro

Email:

Password:

Nombre:

Edad:

Género:  
☐ Hombre ☐ Mujer

País:

Intereses:  
☐ Coches ☐ Motos ☐ Barcos

### Actividad 3. Recoge los datos.

Implementa una aplicación en PHP que recoja los datos recibidos del formulario y devuelva al cliente un mensaje de confirmación.

Abre las herramientas del desarrollador del navegador y comprueba los datos de la petición HTTP enviada cuando se utiliza el método GET y el método POST.

### Actividad 4. La apariencia del formulario.

La apariencia es siempre un aspecto importante y más aún cuando hablamos de formularios en los que le pedimos al usuario que haga una tarea que normalmente no quiere realizar. Clarificar la identificación de cada campo o espaciar y escalar adecuadamente los elementos pueden marcar toda la diferencia. Enlaza una nueva hoja de estilo al formulario y añade las reglas CSS necesarias para conseguir una apariencia como la mostrada.

## Formulario de registro

Email:

Password:

Nombre:

Edad:

Género:  
☐ Hombre ☐ Mujer

País:

Intereses:  
☐ Coches ☐ Motos ☐ Barcos

### Actividad 5. Ayuda al usuario.

El formulario debe diseñarse para minimizar el número de errores del usuario y ayudarlo a entender que se desea que introduzca en cada campo. Una buena práctica es mostrar una indicación incluso un ejemplo de los datos que se deben introducir. Ten en cuenta que tampoco se debe añadir información que no sea necesaria para no distraer ni molestar al usuario.

Imagina que eres tú mismo el que tiene que rellenar el formulario. Identifica los campos que pueden crear algún problema y añade las indicaciones necesarias.

## 2. Validación nativa con HTML5.

Aunque agreguemos información de ayuda a un formulario todavía hay muchos errores que se pueden seguir produciendo, desde un simple error tipográfico al escribir la dirección de correo hasta un usuario que escriba su edad con letras.

Para asegurarnos que la información es correcta, los datos se pueden enviar al servidor, donde se validarían y en caso de que no fueran correctos, se podrían dar indicaciones al usuario para que los modificara. Este proceso, debido principalmente a que los datos deben viajar por la red, no es instantáneo. Para dar un feedback inmediato que mejore la experiencia del usuario, debemos implementar la validación en el lado del cliente aunque no debemos olvidar que por seguridad este proceso no elimina la necesidad de validar los datos en el servidor.

### Actividad 1. Restricciones de validación.

HTML5 incorpora nuevos atributos que permiten validar los campos de un formulario sin escribir código JavaScript. Esta característica de HTML5 habilita a los navegadores para validar campos de forma nativa. Es por tanto el método más seguro y rápido. Revisa el [artículo Validación de restricciones de MDN](#) y el [artículo HTML Input Attributes de W3Schools](#) para conocer estas características de validación.

A continuación introduce las modificaciones necesarias en el formulario para aplicar las siguientes restricciones:

- Los campos email y password son obligatorios. El usuario debe completarlos antes de enviar el formulario.
- En el campo email se debe introducir un dirección de correo válida.
- El password debe incluir al menos 1 número.
- El máximo número de caracteres que puede tener el nombre son 10.
- En el campo edad solo se pueden introducir números entre 18 y 100.

### Actividad 2. Apariencia de los campos con restricciones.

El aspecto de los campos en un formulario con restricciones HTML5 se puede cambiar con nuevos pseudoselectores CSS. Ahora podemos usar:

- `:required` y `:optional` que seleccionan los elementos que tengan el atributo `required` y que no lo tengan respectivamente.
- `:valid` e `:invalid` para los elementos válidos e inválidos en función de las restricciones especificadas.

Puedes ver un resumen de estos elementos en el [apartado Controlando el aspecto de los elementos del artículo Validación de restricciones de MDN](#).

Ahora incluye una nueva regla CSS en el formulario para que cuando un campo no cumpla las restricciones adquiera un color de borde rojo.

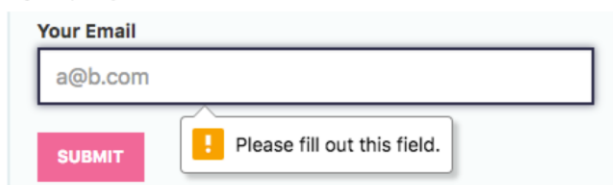
## ¿Es éste el mejor método de validación?

Pues como siempre, dependerá del proyecto en particular, anteriormente indicábamos que esta validación nativa con HTML5 era la más segura y rápida y es así, pero sin duda hay muchos aspectos que no podemos controlar. Las mayores desventajas de este método son:

- No se puede controlar el texto que se muestra en los mensajes de error o su apariencia.
- No es posible establecer restricciones de mayor complejidad, como que se haya escrito la misma contraseña en dos campos diferentes.
- No podemos determinar el momento en el que se validan las restricciones.

Además, no hay que olvidar que el comportamiento de la validación no será uniforme en todos los navegadores, cada uno realiza su propia implementación.

### Chrome



A screenshot of a web form in the Chrome browser. The form has a label "Your Email" above a text input field containing "a@b.com". Below the input field is a pink "SUBMIT" button. To the right of the button, a yellow tooltip with an exclamation mark icon displays the message "Please fill out this field."

### Firefox



A screenshot of a web form in the Firefox browser. The form has a label "Your Email" above a text input field containing "a@b.com". Below the input field, a pink tooltip displays the message "Please fill out this field."

### Safari



A screenshot of a web form in the Safari browser. The form has a label "Your Email" above a text input field containing "a@b.com". Above the input field, a pink tooltip displays the message "Fill out this field".

Para tener total control de la experiencia que tendrá el usuario debemos usar JavaScript. Los dos métodos recomendados son:

- Usar únicamente JavaScript para validar los campos. Esto es, tomar las cadenas de texto escritas por el usuario y otros campos y validarlos usando las funciones que incorpora el lenguaje.
- Usar JavaScript junto a la API de validación de formularios para seguir aprovechando las ventajas de la validación nativa de HTML5. Esto es, seguir utilizando las restricciones HTML en los campos y comprobar a través de la API si se cumplen o no para actuar en consecuencia usando JavaScript.

A continuación veremos los fundamentos de la gestión de formularios con JavaScript, la validación de formulario usando JavaScript y la API de validación y ejemplos de formularios para practicar.

### 3. Fundamentos JavaScript.

En este apartado veremos los fundamentos de la gestión de formularios con JavaScript. Para que no interfieran con las acciones que vamos a implementar en este apartado, elimina por ahora las restricciones de validación del HTML.

#### Actividad 1. Acceder al formulario.

Un formulario es un elemento HTML y como cualquier otro tendrá su representación en el DOM. Los métodos por tanto estudiados en la unidad anterior para acceder y modificar elementos serán igualmente válidos en formularios.

Enlaza un nuevo fichero JavaScript y obtén una referencia al formulario usando los métodos descritos en el *apartado 1.1.- Formas de selección del objeto Form* de los contenidos de la plataforma.

#### Actividad 2. Controlar el envío del formulario.

Cuando utilizamos JavaScript para mejorar la experiencia con un formulario, no queremos que los datos se envíen al servidor hasta que los datos no se hayan procesado por el cliente. Si el usuario pulsa sobre el botón `submit` los datos se enviarán automáticamente al servidor sin ninguna espera. Desde JavaScript podemos evitar este comportamiento por defecto y enviar los datos al servidor cuando nos interese.

Para desactivar el comportamiento por defecto del botón `submit` (y cualquier otra acción por defecto) se utiliza la función `preventDefault` y para enviar el formulario en cualquier momento se usa la función `submit`. Consulta los artículos [preventDefault\(\) Event Method](#) y [Form submit\(\) Method](#) de W3Schools para conocer su funcionamiento.

Vuelve ahora a nuestro formulario y desactiva el comportamiento del botón `submit` para que los datos no se envíen automáticamente al servidor. Ten en cuenta que cuando este botón se pulsa se genera un evento también llamado `submit` que puedo capturar para hacer la operación. Cuando se pulse el botón de `submit` debes mostrar un mensaje por consola.

#### Actividad 3. Acceder a los elementos del formulario.

Para acceder a los campos de un formulario podemos obtenerlos como cualquier otro elemento del DOM o hacer uso de la propiedad `elements` con la que cuentan todos los nodos de tipo formulario.

Utiliza la propiedad `elements` para recorrer los campos del formulario e ir mostrando el nombre de cada campo y el valor introducido por el usuario para conseguir una información similar a la enviada al servidor cuando se hace `submit`. Imprime por consola la información cuando se pulse el botón de enviar.

Consulta el apartado *1.3.- Acceso a propiedades y métodos del formulario* para ver un ejemplo de uso de la propiedad `elements`. Consulta también el apartado *2.- Objetos relacionados con formularios* y sus subapartados para conocer las propiedades de los diferentes campos del formulario. Como en `elements` no se encuentran los elementos `select`, obtén el elemento seleccionado del `select` e imprime sus datos al final por consola.

## 4. Acciones comunes de JavaScript sobre formularios.

### Actividad 1. Deshabilitar el botón de submit.

Un problema habitual en los formularios es la tendencia que tienen los usuarios a hacer clic varias veces en el botón de `submit`, lo cuál supone el envío de información al servidor en varias ocasiones. Una solución común es deshabilitar el botón de `submit` una vez el usuario haya enviado el formulario.

Implementa esta solución teniendo en cuenta que todos los campos cuentan con un atributo “disabled” que deshabilita el campo correspondiente.

Para que el usuario sepa que el botón ya no está activo, añade una nueva regla CSS que cambie el color de fondo del botón a gris cuando el botón esté deshabilitado. Utiliza el selector `:disabled`. Para que la operación sea sencilla de comprobar visualmente, detén el comportamiento por defecto del `submit`.

### Actividad 2. Dar el foco al primer campo.

Sobre cualquier campo del formulario podemos aplicar dos métodos: `focus()` y `blur()`. El primero le da el foco al elemento y el segundo se lo quita. Puedes ver más información y ejemplos de los métodos en el [artículo HTML DOM blur\(\) Method](#) de W3Schools.

Una práctica común es darle el foco al primer campo cuando la página se ha cargado. Implementa este comportamiento teniendo en cuenta que cuando la página ha terminado de cargarse se genera el evento `load` asociado al objeto `window`.

HTML5 también ha incorporado el atributo `autofocus` que tiene el mismo objetivo.

### Actividad 3. Elegir el momento de la validación.

Hay 2 eventos que se generan para todos los campos: `blur` y `focus`. El evento `focus` se genera cuando un elemento gana el foco, el evento `blur` cuando el elemento lo pierde. También es habitual hacer uso de los eventos `change` y `input` que se producen cuando hay una modificación en el valor del campo, sus diferencias son sutiles. En el [artículo HTML Element: Evento change](#) de MDN puedes encontrar más información.

Añadiendo manejadores a estos eventos podemos determinar el momento en el que validar un campo. El evento `blur` es el más utilizado.

Muestra una ventana emergente en el navegador si el dato introducido en el campo edad no es un número. Lleva a cabo esta tarea cuando el campo pierda el foco.

**Actividad 4. Filtrar la entrada de texto.**

Algunos campos de texto requieren de caracteres específicos. Desde JavaScript es posible bloquear aquellos caracteres no válidos de forma que directamente el usuario no pueda introducirlos.

Este comportamiento se puede conseguir con el evento `keypress` que se genera cada vez que se va a introducir un carácter y que la propiedad `charCode` del objeto `Event` permite conocer el carácter introducido. Consulta el [artículo KeyboardEvent charCode Property](#) de W3Schools para conocer el funcionamiento de la propiedad `charCode`.

Evita que en el campo `edad` puedan introducirse otros caracteres diferentes a números.

**Actividad 5. Generación dinámica de campos.**

Es común que algunos campos del formulario se creen dinámicamente en el cliente a partir de los datos proporcionados por el servidor. Siempre podemos crear y añadir nuevos elementos utilizando las funciones ya estudiadas del DOM pero además hay campos como los `select` que cuentan con métodos específicos `add` y `remove` para esta tarea.

En el documento HTML, localiza el campo `país` y borra las opciones del `select`. Genera automáticamente sus opciones (elementos de tipo `option`) a partir del siguiente array de objetos:

```
const aOpcionesPais = [  
  {text: "Portugal", value: "pt"},  
  {text: "Francia", value: "fr"},  
  {text: "Reino Unido", value: "uk"},  
  {text: "Alemania", value: "de"},  
  {text: "España", value: "es"}  
];
```

Los elementos `select` cuentan con la propiedad `selected` para indicar la opción seleccionada. Selecciona por defecto la opción de España.



## 5. Validación con JavaScript y la API de validación de HTML5.

HTML5 introduce 3 categorías de características para la validación de formularios:

- Los nuevos tipos y atributos HTML.
- Los nuevos pseudoselectores CSS.
- Una nueva **API de validación JavaScript** con funciones y propiedades para saber si los campos cumplen las restricciones impuestas.

Los pasos a seguir para implementar este método son:

1. Incluir en el HTML las restricciones que deben cumplir los campos.
2. Desactivar la notificación automática de errores por parte del navegador.
3. Utilizar JavaScript para, consultando la API de validación, saber si los campos cumplen las restricciones e informar al usuario convenientemente.

A continuación se indican artículos recomendados para conocer el funcionamiento e implementación de este método:

- El [artículo Why you should be using HTML5 form validation: a tour](#) de pageclip.co.
- El [artículo Constraint Validation: Native Client Side Validation for Web Forms](#) de html5rocks.com
- El [artículo The Constraint Validation API \(JavaScript\)](#) de css-tricks.com.

### Actividad 1. Añade las restricciones de validación.

El primer paso es añadir las restricciones de validación en el HTML. Introduce de nuevo en el formulario las restricciones descritas en el apartado de validación nativa con HTML5.

### Actividad 2. Desactiva la notificación automática de errores.

El segundo paso es desactivar la notificación de errores que realiza el navegador.

Realizar esta acción es tan sencillo como añadir el atributo `novalidate` al formulario. Puedes ver un ejemplo en el [artículo HTML <form> novalidate Attribute](#) de W3Schools.

También es posible añadir el atributo `formnovalidate` al botón de submit. Este enfoque nos permite tener un botón que use la notificación automática del navegador y otro que no para realizar pruebas. Puedes ver un ejemplo en el [artículo HTML <input> formnovalidate Attribute](#) de W3Schools.

Desactiva la notificación automática de errores en nuestro formulario de ejemplo.

El tercer y último paso es usar JavaScript junto a la API de validación para conocer si los campos cumplen las restricciones e informar al usuario convenientemente. Ya que JavaScript nos da la posibilidad de controlar todo el proceso cabe preguntarnos cuál sería la mejor forma de validar los campos por ejemplo ¿cómo mostrar los errores?, ¿en qué momento?

En este sentido la consultora *Nielsen Norman Group* ha elaborado un artículo con 10 líneas de diseño para el reporte de errores en formularios web basado en la investigación y experiencia. Lo usaremos como marco para nuestras validaciones. En este enlace puedes ver el [artículo original](#) y en este [otro artículo](#) un resumen en castellano del blog torresmuriel.com.

Volvamos a centrarnos en la implementación.

## 6. Uso de la API de validación.

Para conocer si los campos cumplen las restricciones especificadas en el HTML, desde JavaScript podemos usar la API de validación de HTML5. En el [artículo JavaScript Validation API](#) de W3Schools puedes ver una descripción de sus métodos y propiedades.

Para empezar fíjate en el método `checkValidity` y en la propiedad `validity`.

- El método `checkValidity` :
  - Permite comprobar si el valor introducido por el usuario cumple las restricciones impuestas en el HTML y guarda el resultado de la validación en la propiedad `validity`.
  - Se invoca automáticamente cuando se pulsa el botón `submit`.
  - Genera un evento `invalid` si el resultado de la validación a sido negativo.
- La propiedad `validity` :
  - Contiene a su vez las propiedades `customError`, `patternMismatch`, `rangeOverflow`, `rangeUnderflow`, `stepMismatch`, `tooLong`, `typeMismatch`, `valueMissing` y `valid`, que se corresponden con cada una de las restricciones que puedo aplicar sobre el HTML. Cada propiedad puede contener el valor `true` o `false`.
  - Por ejemplo, si al hacer `checkValidity` sobre un campo que esté requerido (atributo `required`), la restricción no se cumple, se guardará automáticamente el valor `true` en la propiedad `validity.valueMissing`. Si la restricción se cumple almacenará el valor `false`.

### Actividad 1. Implementa el momento de la validación.

El momento ideal para hacer la validación de un campo (método `checkValidity`) es cuando el usuario ha terminado de rellenar el campo y este pierde el foco (evento `blur`). Implementa este comportamiento en nuestro formulario.

Además ten en cuenta que no permitiremos que el formulario se envíe hasta que todos los campos tengan una validación positiva. Implementa también esta función en el formulario.

### Actividad 2. Actúa cuando se produce un error.

Si el resultado de la validación con `checkValidity` ha sido negativo, es decir el campo no cumple las restricciones especificadas en el HTML, se generará un evento `invalid`. Podemos usar un manejador de este evento para notificar el error al usuario.

La notificación del error debe consistir en el cambio de la apariencia del campo por ejemplo añadiendo un borde rojo y en un mensaje que le indique dónde se ha equivocado incluso cómo corregirlo. El mensaje debe aparecer junto al campo, en nuestro caso lo colocaremos justo debajo.

Notifica los errores a los usuarios en nuestro formulario, para cambiar la apariencia del campo utiliza una clase CSS.

**Actividad 3. Actúa cuando el campo vuelve a ser correcto.**

En el momento en el que usuario introduzca un valor correcto, la notificación del error debería desaparecer. El evento `input` se dispara cuando el `value` de un elemento `input`, `select` o `textarea` ha sido cambiado.

Utiliza un manejador para el evento `input` de forma que cuando el valor del campo cambie, se compruebe si el campo ya es válido y en tal caso se elimine la notificación del error. Para comprobar si un campo es válido recuerda que puedes usar la propiedad `validity.valid`.

## 7. Restricciones complejas

En ocasiones las restricciones disponibles en HTML5 no son suficientes para validar un campo. Un ejemplo común es el típico campo de repetición de la contraseña para asegurarnos que el usuario no se ha equivocado escribiéndola en el primer intento y pueda tener problemas de acceso en el futuro.

En la implementación del campo podríamos aplicar un `required` o incluso un `pattern` pero no tenemos forma de establecer una restricción que compruebe que las contraseñas escritas en ambos campos coinciden. Por suerte la API de validación de HTML5 cuenta con el método `setCustomValidity` que permite marcar un campo como inválido sea cuál sea la condición que deseamos que se cumpla.

El funcionamiento es bastante simple:

- Si indicamos un mensaje en el método, por ejemplo `setCustomValidity('invalid')` la propiedad del campo `validity.customError` tomará el valor `true`.
- Si indicamos una cadena vacía: `setCustomValidity('')` la propiedad del campo `validity.customError` tomará el valor `false`.

El establecimiento de un `customError` para un campo debería hacerse en el mismo lugar en el que se establecen el resto de restricciones, es decir, cuando se invoque al método `checkValidity`. Igualmente, antes de comprobar si el campo ya es `valid` se debería comprobar si la restricción personalizada ya se cumple para quitar el `customError`.

### Actividad 1. Campo de repetición de password.

Añade un nuevo campo “Repita el password” justo debajo del campo “Password” original. Implementa una restricción personalizada de forma que el campo no quede validado hasta que ambos passwords coincidan.

### Actividad 2. Género obligatorio y al menos 2 intereses.

Añade las restricciones necesarias para que el campo “Género” sea obligatorio y en el campo “Intereses” se deban marcar al menos 2 intereses.

## 8. Formularios para practicar

### Actividad 1. Formulario de datos personales.

Implementa un formulario con los siguientes campos y restricciones. Los mensajes de error se deben mostrar justo encima del campo.

- Campo *Nombre*. Obligatorio. Acepta valores entre 8 y 15 caracteres de longitud.
- Campo *Apellidos*. Obligatorio. Acepta valores entre 8 y 30 caracteres de longitud.
- Campo *Web personal*. Sólo permite direcciones url válidas.
- Campo *Edad*. Cuyo valor sólo admite números entre 18 y 60.
- Campo *Fecha de nacimiento*. Obligatorio. Solo acepta valores de fecha con formato DD/MM/AAAA.
- Campo *Ocupación*. Obligatorio. Tipo `select` con opciones “desempleado”, “en activo” y “jubilado”.
- Campo *Hijos*. Obligatorio. Tipo `radio` con opciones “sí” y “no”.
- Campo *Intereses*. Tipo `checkbox` con opciones “cine”, “música” y “teatro”.

### Actividad 2. Formulario de inscripción en cursos.

Implementa un formulario con los siguientes campos y restricciones. Los mensajes de error se mostrarán al inicio del formulario.

- Campo *Nombre*. Obligatorio. Acepta valores entre 8 y 15 caracteres de longitud.
- Campo *Apellidos*. Obligatorio. Acepta valores entre 8 y 30 caracteres de longitud.
- Campo *email*. Sólo admite direcciones de correo válidas.
- Campo *Comunidad autónoma*. Tipo `select` con las comunidades autónomas de España.
- Campo *Provincia*. Tipo `select` con las provincias de la comunidad autónoma seleccionada.
- Campo *Cursos*. Tipo `checkbox` con opciones “Introducción a HTML5”, “Introducción a CSS3”, “Introducción a JavaScript”, “HTML5 avanzado”, “CSS3 avanzado” y “JavaScript avanzado”.
- Campo *Precio Total*. Campo dinámico que se actualiza con la suma de los precios de los cursos marcados en el campo anterior. Los cursos de introducción tienen un precio de 100€ y los cursos avanzados de 150€. Si se seleccionan dos o más cursos se obtiene un descuento del 20%.
- Campo *Forma de pago*. Tipo `radio` con opciones “Tarjeta de débito o crédito”, “PayPal” y “Transferencia bancaria”.

**Actividad 3. Formulario de registro.**

Implementa un formulario con los siguientes campos y restricciones. Los mensajes de error se deben mostrar justo debajo del campo.

- Campo *Nombre de usuario*. Obligatorio. Acepta valores entre 8 y 15 caracteres de longitud.
- Campo *Contraseña*. Obligatorio. Debe tener al menos una longitud de 8 caracteres, una letra mayúscula y un guión bajo.
- Campo *Email*. Sólo admite direcciones de correo de Google.
- Campo *Teléfono*. Debe ser un teléfono válido.
- Campo *Teléfono alternativo*. Debe ser un teléfono válido y distinto al campo anterior.
- Campo *Descripción personal*. Tipo `textarea` con 400 caracteres de longitud máxima. Se debe informar al usuario de los caracteres restantes que le quedan por escribir.
- Campo *Intereses*. Tipo `checkbox` con opciones “Desarrollo”, “Diseño”, “Negocios”, “Pruebas”, “Análisis”. Es obligatorio que se marquen al menos 2 intereses.