<!--UPSLP--> API FORMS {

<Materia="Programación Web 1"/>

}

Equipo 1 {

EDWIN ADMAEL CASTILLO G.

ITI 181700

VALERI ROCHA

ITI 182335

ROBERTO EMIL MORALES

ITI 182377

AGUSTIN ROMO ARREDONDO

ITI 181634

CARMEN LISBETH SANCHEZ ROJAS

ITI 179719

}

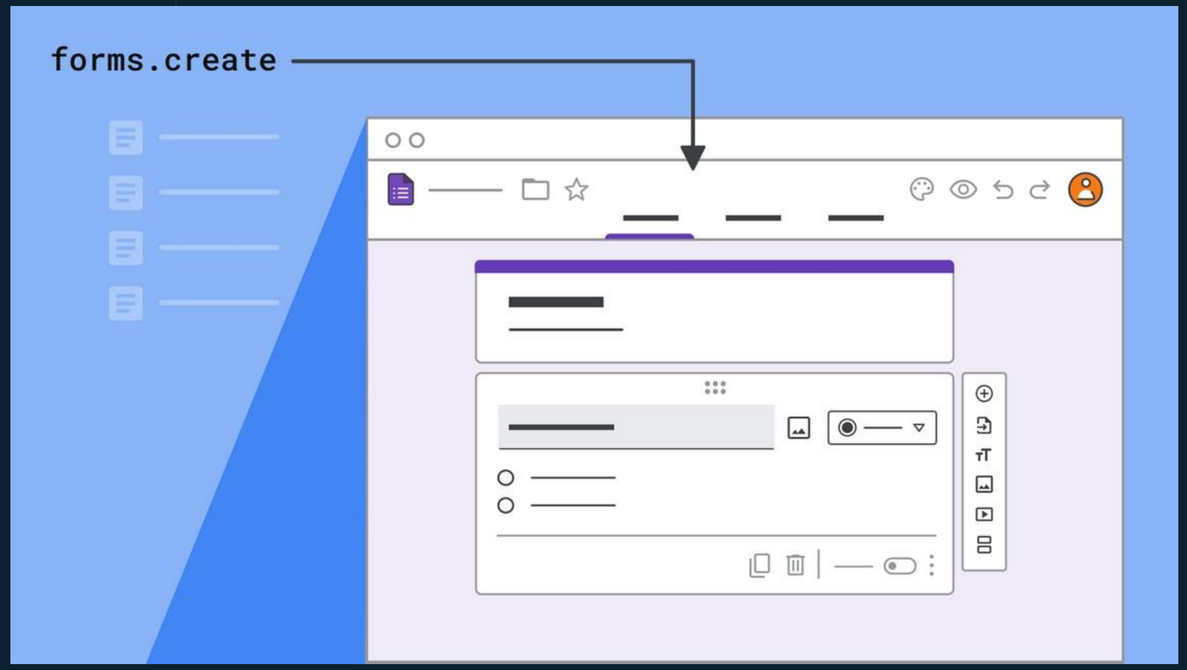
API Forms

Los formularios web son la interface más importante de todas, permiten a los usuarios insertar datos, tomar decisiones, comunicar información y cambiar el comportamiento de una aplicación.
Durante los últimos años, códigos personalizados y librerías fueron creados para procesar formularios en el ordenador del usuario.

Introducción {

Las API Forms son una forma de interactuar con formularios web utilizando una API (Application Programming Interface) para enviar y recibir datos.

En lugar de interactuar manualmente con un formulario web, se puede usar una API para automatizar el proceso y enviar los datos de forma programática.



}

Mas sobre las API {

Las API Forms son útiles para una variedad de casos de uso, como:

- Automatización de la entrada de datos. Integración de sistema.
  - Elimina la necesidad de
  - interactuar manualmente con formularios web.

Las API Forms también pueden ser utilizadas para:

- Validación de datos. Mejora de
- la precisión de los datos enviados a través de formularios.

Formas de uso.

Una forma común de utilizar API Forms es mediante el uso de solicitudes HTTP. Las solicitudes HTTP son utilizadas para enviar datos a través de la API a un servidor web

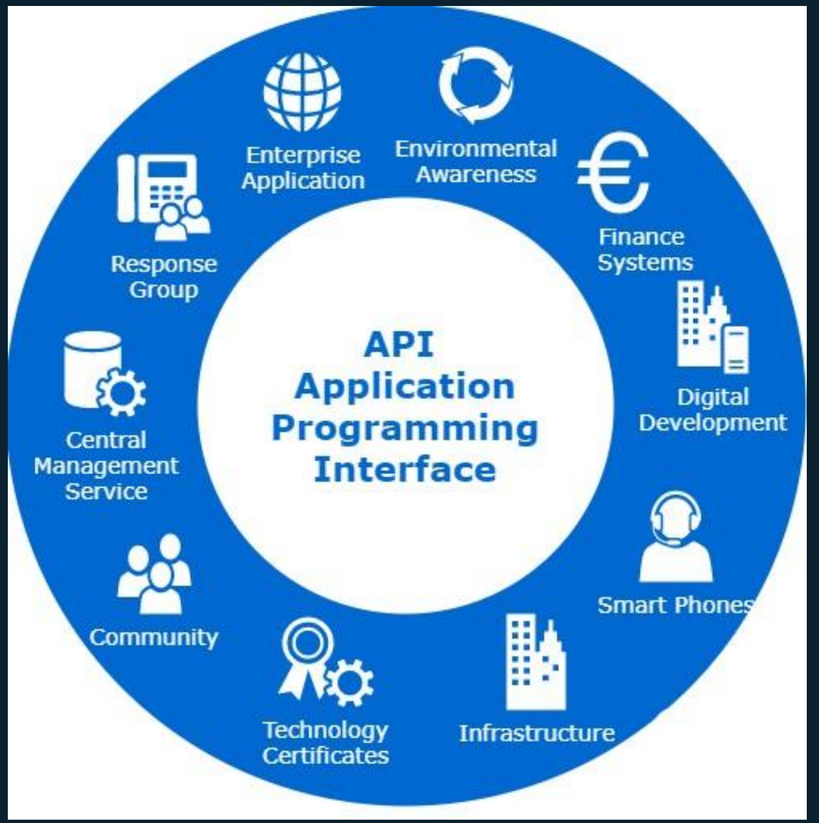
Para utilizar una API Form, primero se debe obtener una clave de API y una URL de la API. Luego, se puede utilizar una biblioteca de programación en un lenguaje como Python o JavaScript para enviar solicitudes HTTP a la API.

}

Datos relevantes {

Las API Forms son una herramienta poderosa para interactuar con formularios web de forma programática. Al automatizar el proceso de entrada y procesamiento de datos, se pueden mejorar la eficiencia y precisión de los sistemas que utilizan formularios web.

Esto es útil para casos en los que se necesita obtener información de un usuario, como en una encuesta o un formulario de registro.



El Código {

}

Los formularios en HTML no han cambiado mucho. La estructura sigue siendo la misma, pero HTML5 ha agregado nuevos elementos, tipos de campo y atributos para expandirlos tanto como sea necesario y proveer así las funciones actualmente implementadas en aplicaciones web. Siempre tendrá el elemento <form>.

El elemento más importante en un formulario es <input>. Este elemento puede cambiar sus características gracias al atributo type (tipo). Este atributo determina qué clase de entrada es esperada desde el usuario. Los tipos disponibles hasta el momento eran el multipropósitos text (para textos en general) y solo unos pocos más específicos como password o submit.

formulario_api_ejemplo.html > ...

```
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4   <title>Formularios</title>
5   </head>
6   <body>
7   <section>
8   <form name="miformulario" id="miformulario" method="get">
9   <input type="text" name="nombre" id="nombre">
10  <input type="submit" value="Enviar">
11  </form>
12  </section>
13  </body>
14  </html>
15
```

+ sobre el Código {

TIPO EMAIL

Casi todo formulario en la web ofrece un campo para ingresar una dirección de email, pero hasta ahora el único tipo de campo disponible para esta clase de datos era text.

El tipo text representa un texto general, no un dato específico, por lo que teníamos que controlar la entrada con código Javascript para estar seguros de que el texto ingresado correspondía a un email válido. Ahora el navegador se hace cargo de esto con el nuevo tipo email:

<input type="email" name="miemail"

formulario_api_ejemplo.html > ...

```
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4   <title>Formularios</title>
5   </head>
6   <body>
7   <section>
8   <form name="miformulario" id="miformulario" method="get">
9   <input type="text" name="nombre" id="nombre">
10  <input type="submit" value="Enviar">
11  </form>
12  </section>
13  </body>
14  </html>
15
```

id="miemail">

+ sobre el Código {

TIPO SEARCH

El tipo search (búsqueda) no controla la entrada, es solo una indicación para los navegadores. Al detectar este tipo de campo algunos navegadores cambiarán el diseño del elemento para ofrecer al usuario un indicio de su propósito.

<input type="search" name="busqueda" id="busqueda">

TIPO URL

Este tipo de campo trabaja exactamente igual que el tipo email pero es específico para direcciones web. Está destinado a recibir

formulario_api_ejemplo.html > ...

```
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4   <title>Formularios</title>
5   </head>
6   <body>
7   <section>
8   <form name="miformulario" id="miformulario" method="get">
9   <input type="text" name="nombre" id="nombre">
10  <input type="submit" value="Enviar">
11  </form>
12  </section>
13  </body>
14  </html>
15
```

solo URLs absolutas y retornará un error si el valor es inválido.

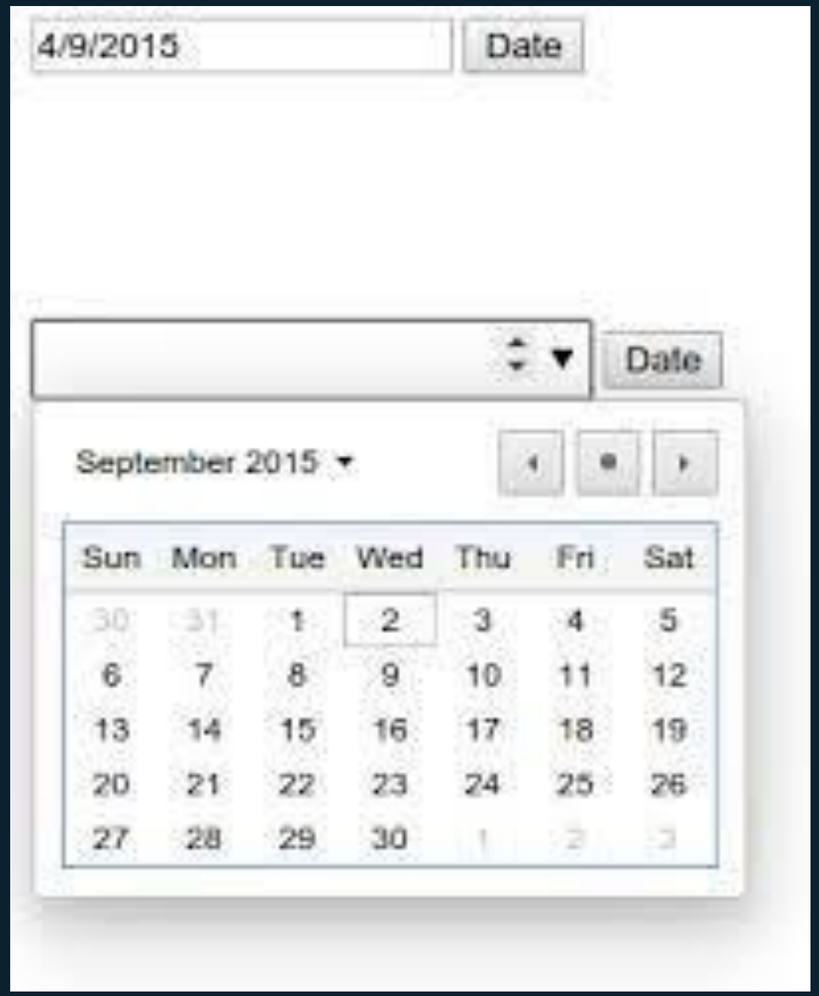
```
<input type="url" name="miurl" id="miurl">
```

+ sobre el Código {

TIPO DATE

Este es otro tipo de campo que genera una nueva clase de control. En este caso fue incluido para ofrecer una mejor forma de ingresar una fecha. Algunos navegadores muestran en pantalla un calendario que aparece cada vez que el usuario hace clic sobre elcampo.

El calendario le permite al usuario seleccionar un día que será ingresado en el campo junto con el resto de la fecha.



Gracias al tipo date ahora es el navegador el que se
encarga de construir un almanaque o las herramientas necesarias para facilitar el ingreso de este tipo de datos. **<input type="date" name="fecha" id="fecha">**

}

Ejemplo de Código {

formulario_api_ejemplo.html > ...

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de API Form</title>
</head>
<body>
    <h1>Enviar datos a través de una API Form</h1>

    <form id="my-form" action="" method="post">
        <label for="name">Nombre:</label>
        <input type="text" id="name" name="name"><br>

        <label for="email">Correo electrónico:</label>
        <input type="email" id="email" name="email"><br>

        <label for="phone">Teléfono:</label>
        <input type="text" id="phone" name="phone"><br>

        <button type="submit">Enviar</button>
    </form>

    <script>
        // Agregar un manejador de eventos para el envío del formulario
        document.getElementById("my-form").addEventListener("submit", function(event) {
            event.preventDefault(); // Prevenir el envío del formulario por defecto

            // Datos a enviar a través de la API Form
            var data = {
                name: document.getElementById("name").value,
                email: document.getElementById("email").value,
```

formulario_api_ejemplo.html > ...

```
                // Datos a enviar a través de la API Form
                var data = {
                    name: document.getElementById("name").value,
                    email: document.getElementById("email").value,
                    phone: document.getElementById("phone").value
                };

                // Enviar la solicitud HTTP utilizando la biblioteca Fetch de JavaScript
                fetch("https://api.example.com/form", {
                    method: "POST",
                    headers: {
                        "Content-Type": "application/json",
                        "Authorization": "Bearer your_api_key"
                    },
                    body: JSON.stringify(data)
                })
                .then(response => response.json())
                .then(data => {
                    console.log(data); // Imprimir la respuesta de la API en la consola del navegador
                })
                .catch(error => {
                    console.error(error); // Manejar cualquier error de la solicitud HTTP
                });
            });
        </script>
    </body>
    </html>
```

<!--Programacion Web 1 -->

Gracias por su atención{

<Por="Alumnos ITI "/>

}