WordPress: associare una Google Map ai post usando i custom field per gli indirizzi

In questo articolo vedremo come associare un indirizzo ai post di WordPress e visualizzarli utilizzando le API delle Google Maps.

L'API fondamentale da attivare nel nostro account Google, oltre a quella Maps, è l'API Geocoding che utilizzeremo per convertire un indirizzo in una coppia di coordinate (latitudine e longitudine).

Per visualizzare la mappa associata ad un post, sfrutteremo un custom field di WordPress e creeremo uno shortcode per poter inserire la mappa nel contenuto del post.

Al fine di mantenere il codice PHP il più lineare possibile, creeremo un template HTML che conterrà il codice HTML, CSS, e JavaScript per usare le API di Google. In tal senso creeremo le variabili segnaposto `<address>` e `<apikey>` per inserire dinamicamente l'indirizzo reperito dal custom field e la nostra chiave API, rispettivamente.

Il template HTML avrà questi contenuti:

```
<script>
function initMap() {
  const map = new
google.maps.Map(document.getElementById("my-map"), {
    zoom: 6,
    center: { lat: 41.9101776, lng: 12.4659587 },
    mapTypeControl: false
  });
  const geocoder = new google.maps.Geocoder();
  const marker = new google.maps.Marker({
    map
  });
```

```
  const address = "<address>";


  geocoder
    .geocode({ address })
    .then((result) => {
      const { results } = result;

      map.setCenter(results[0].geometry.location);
      marker.setPosition(results[0].geometry.location);
      marker.setMap(map);
    })
    .catch((e) => {
      console.log(e);
    });
}

window.initMap = initMap;
</script>

<style>
#my-map {
    width: 100%;
    height: 350px;
}
</style>


<div id="my-map"></div>

<script
      src="https://maps.googleapis.com/maps/api/js?key=
<apikey>&callback=initMap&v=weekly"
      defer
    >
</script>
```

La funzione JavaScript globale `initMap()` inizializza la mappa, quindi usa l'indirizzo contenuto nella variabile `address` per effettuare la

geolocalizzazione dell'indirizzo che, se ha successo, restituirà le coordinate che verranno usate per mostrare il marcatore sulla mappa.

A livello di codice PHP possiamo creare il seguente plugin:

```
<?php
/*
Plugin Name: My Post Location
Version: 1.0
Description: Creates a Google Map for a given post
Author: Me
Author URI: https://mysite.tld
*/

define( 'APIKEY', 'chiave-api' );
```

Nel plugin, il primo step da compiere è associare e salvare un custom field specifico per i post.

```
function my_register_meta_boxes() {
    add_meta_box( 'my', __( 'Indirizzo', 'my' ),
'my_display_callback', 'post' );
}
add_action( 'add_meta_boxes', 'my_register_meta_boxes' );

function my_display_callback() {
    global $post;
    ?>
    <style>
        .my-meta { padding-bottom: 8px }
        label[for^="my-"] {
            display: block;
            margin-bottom: 8px;
        }
        #my-location {
            display: block;
            width: 100%;
        }
    </style>
    <p class="meta-options my-meta">
```

```
        <label for="my-location"><?php _e( 'Indirizzo', 'my'
); ?></label>
        <input id="my-location" type="text"
name="my_location" value="<?php echo get_post_meta( $post-
>ID, 'my_location', true); ?>">
    </p>
<?php
}

function my_save_meta_box( $post_id ) {
    if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE )
return;
    if ( $parent_id = wp_is_post_revision( $post_id ) ) {
        $post_id = $parent_id;
    }
    $fields = [
        'my_location'
    ];
    foreach ( $fields as $field ) {
        if ( array_key_exists( $field, $_POST ) ) {
            update_post_meta( $post_id, $field,
sanitize_text_field( $_POST[$field] ) );
        }
     }
}
add_action( 'save_post', 'my_save_meta_box' );
```

A questo punto possiamo creare lo shortcode che leggerà il template HTML effettuando la sostituzione delle variabili segnaposto restituendo la stringa HTML risultante.

```
function my_map() {
    global $post;
    $content = '';
    $location = trim( get_post_meta( $post->ID,
'my_location', true ) );
    if( !$location ) {
        return $content;
    }
    $html_map = file_get_contents( plugins_url(
```

```
'src/map.html', __FILE__ ) );
    $sanitized_location = str_replace( '"', '', $location );
    $content = str_replace( [ '<address>','<apikey>' ], [
$location, APIKEY ], $html_map );
    return $content;
}

add_shortcode( 'my-map', 'my_map' );
```

Il file HTML si trova indicativamente nella directory `src` all'interno della directory del nostro plugin, quindi la costante PHP `__FILE__` permette alla funzione di WordPress `plugins_url()` di calcolare l'URL corretto a partire dal file PHP principale del nostro plugin.

Si tratta di un'implementazione relativamente semplice che può essere ulteriormente perfezionata.