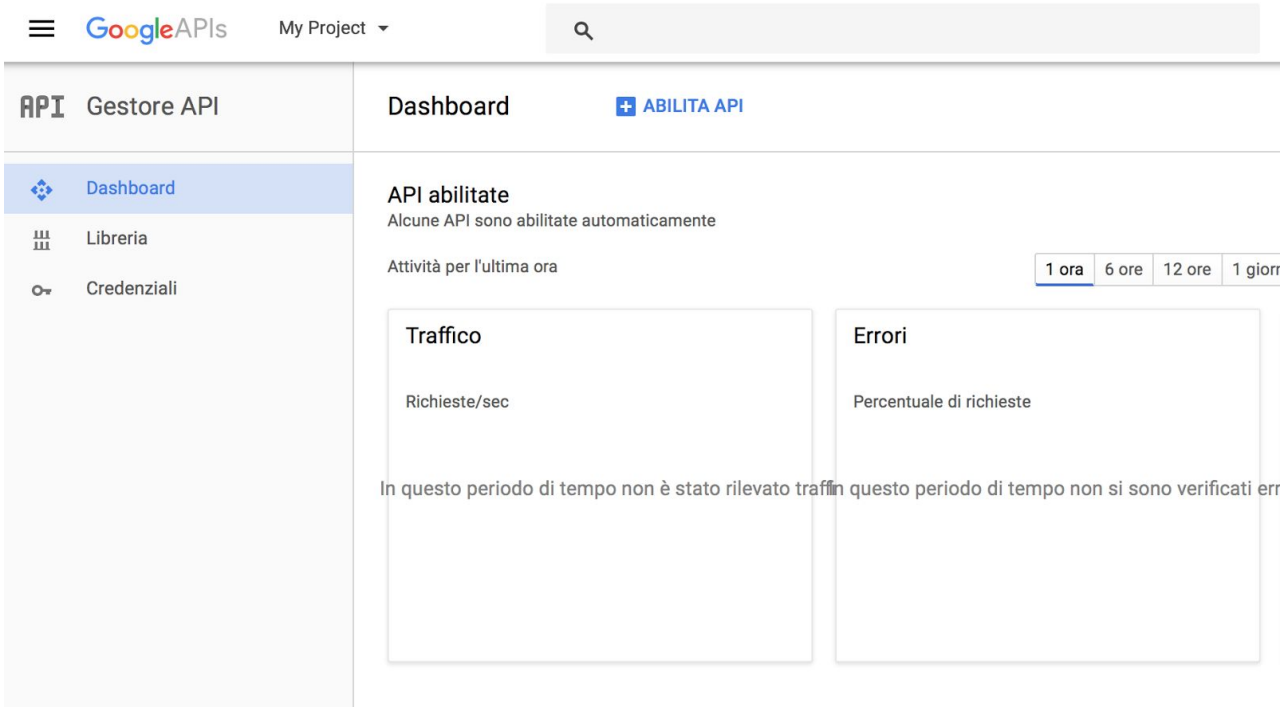
Esercitazione Google Maps

Esercizio 1

Visualizzare i risultati della API comuni.php su una mappa geografica.

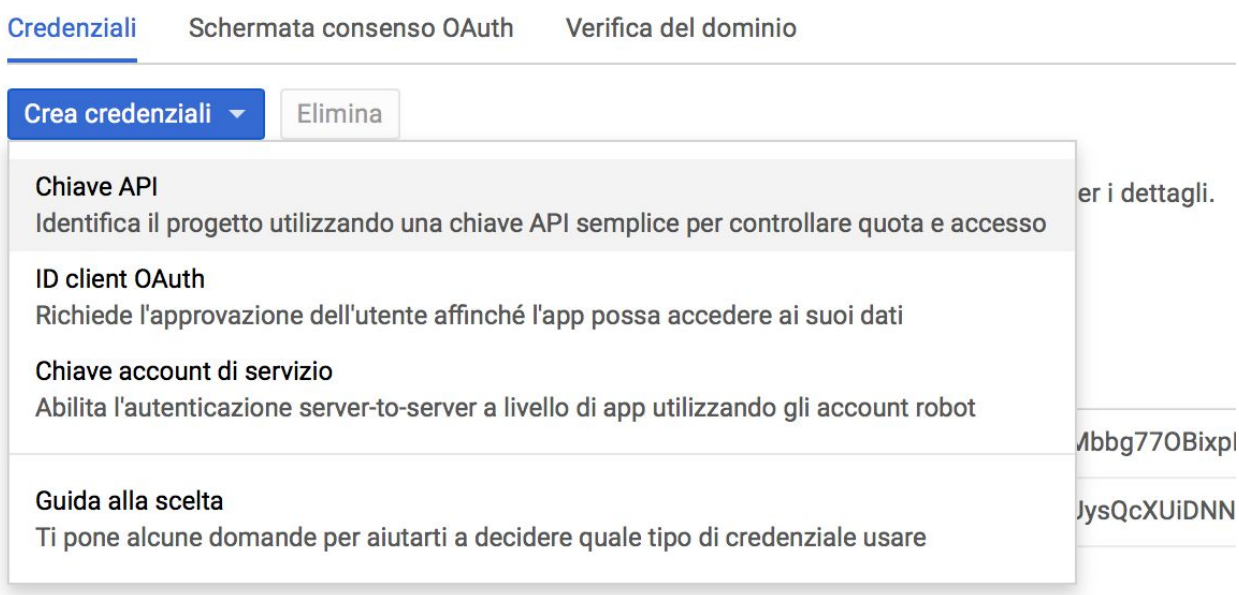
Soluzione

Come prima cosa, occorre creare una chiave di utilizzo della libreria Google Maps. Per fare questo, andiamo sul sito della Google API Console[1]. Se non abbiamo un account google, dobbiamo registrarci e poi possiamo loggarci. Una volta loggati al sistema, appare una schermata come quella mostrata nella figura seguente.



Nel menu a sinistra, selezionare Credenziali e poi Crea credenziali e Chiave API:

[1] https://console.developers.google.com/



A questo punto è stata creata una nuova chiave. Copiare il valore della nuova chiave da qualche parte in modo che poi possa essere utilizzata. Ora occorre abilitare la nuova chiave ad accedere alle API di Google Maps. Per fare questo, nel menu a sinistra selezionare Dashboard e quindi abilita API. Selezionare Google Maps Javascript API e poi Abilita.

Ora passiamo a modificare il file index.html in modo da includere la libreria di Google Maps. Nell'header del file, immediatamente dopo l'inclusione della libreria Highcharts e prima dell'inclusione del nostro script, aggiungiamo la seguente libreria:

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=MIA_CHIAVE&callback=in
izializza">
</script>
```

avendo l'accortezza di sostituire MIA_CHIAVE con la chiave che abbiamo appena creato. Ora dobbiamo creare un div che conterrà la mappa. Creiamolo subito dopo il div con id id_div:

```
<div id="mappa"></div>
```

Dobbiamo anche definire un foglio di stile css che specifica le dimensioni della mappa. Pertanto, prima dell'inclusione di tutte le librerie javascript, includiamo un foglio di stile:

```
<link rel="stylesheet" href="css/stile.css" type="text/css">
```

Ora il file index.html è pronto. Passiamo a definire il foglio di stile. Nella cartella css creiamo un file stile.css, al cui interno definiamo le dimensioni della div che abbiamo appena creato, ad esempio fissiamo la larghezza all'intero schermo e l'altezza a 400px:

```
#mappa {
        height: 400px;
        width: 100%;
}
```

Ora passiamo alla creazione della mappa. Apriamo il file main.js. Quando abbiamo incluso la libreria di Google Maps nel file index.html, abbiamo scritto anche callback=inizializza. Questo significa che la libreria di Google Maps eseguirà la funzione inizializza per la creazione della mappa. Se si vuole creare una mappa direttamente al caricamento della pagina, occorre scrivere tutto il codice dentro la funzione inizializza. Nel nostro caso, però, la mappa va caricata solo nel caso in cui venga selezionato il pulsante comuni e poi coordinate. Quindi definiamo la funzione inizializza dentro il file main.js, lasciandola vuota. La creeremo al momento opportuno:

```
function inizializza(){}
```

Ora dobbiamo creare la mappa in corrispondenza della pressione del tasto coordinate. Dobbiamo distinguere il caso in cui venga pressato il tasto coordinate da quello in cui venga pressato il tasto url. Nel caso in cui si preme il tasto coordinate, dovrà essere invocata una funzione crea_mappa, altrimenti verrà invocata la funzione crea_tabella:

```
if(type == 'coordinate')
     $.getJSON('api/comuni-get.php?type=' + type, crea_mappa);
else
     $.getJSON('api/comuni-get.php?type=' + type, crea_tabella);
```

Definiamo ora la funzione crea_mappa, in cui creiamo la mappa:

```
function crea_mappa(data)
{
     var opzioni = {
          zoom       : 8,
          center     : {lat: 43.029533, lng: 13.174643}
     };
       var map = new google.maps.Map(document.getElementById('mappa'),
opzioni);
}
```

Abbiamo definito una variabile opzioni, che è un array associativo contenente tutte le opzioni da passare alla mappa. In particolare abbiamo settato il centro della mappa (in uno dei comuni terremotati) e lo zoom. Poi creiamo la mappa invocando la funzione messa a disposizione da Google.
Vediamo ora come inserire dei marker nella mappa.

```
var marker = new google.maps.Marker({
    position: {lat: valore_lat, lng: valore_lng},
    map: map
});
```

Per creare un marker, occorre specificare almeno la posizione del marker e la mappa su cui si vuole inserire. Nel nostro caso vogliamo inserire tanti marker quanti sono i comuni terremotati. Per fare questo, occorre modificare l'API comuni.php in modo da preparare i dati direttamente per la creazione del marker. Apriamo il file comuni.php. Dobbiamo modificare il codice in modo che nel caso in cui si voglia invocare l'API con parametro coordinate lo script invochi una funzione get_risultati_mappa, altrimenti è chiamata la funzione get_risultati:

```
switch($type)
{
	case 'coordinate':
		$sql=costruisci_query(array('Comune',
		'latitude','longitude'),"ComuniTerremotati", null,"");
		$risultato = get_risultati_mappa($sql);
		break;
	case 'url':
		$sql = costruisci_query(array('Comune',
		'url'),"ComuniTerremotati", null,"");
		$risultato = get_risultati($sql);
		break;
	default:
		echo json_encode(array("tipo non riconosciuto"));
		exit(1);
}
echo $risultato;
```

Definiamo ora la funzione get_risultati_mappa, dentro il file funzioni.php. Questa funzione deve restituire un array numerico fatto da tanti array associativi, uno per ogni comune:

```
[
	{
		coordinate : {lat: 32.2, lng: 12.3},
		comune     : Pisa
	},

	{
		coordinate : {lat: 34.3, lng: 12.4},
		comune     : Roma
	},
	 ...
]
```

Per fare questo, definiamo un array $return in cui mettiamo i risultati, eseguiamo quindi la query tramite la select e scorriamo l'array $risultato tramite ciclo for:

```
function get_risultati_mappa($sql)
{
	include_once('../api/config.php');
	$risultato = select($db,$sql);
```

```
	$return = array();

	for($i = 0; $i < count($risultato); $i++)
	{
		$return[$i]['comune'] = $risultato[$i]['Comune'];
		$return[$i]['coordinate'] = array(
			'lat' => floatval($risultato[$i]['latitude']),
			'lng' => floatval($risultato[$i]['longitude']));

	}
	closeDB($db);
	return json_encode($return);
}
```

Ricordiamoci di trasformare latitudine e longitudine in float.
Ritorniamo al file main.js e aggiungiamo tanti markers quanti sono quelli contenuti in data:

```
for(var i = 0; i < data.length; i++)
{
	var marker = new google.maps.Marker({
			position: data[i].coordinate,
			map: map
		});
}
```

Aggiungiamo ora una infowindow per ogni marker, dentro il ciclo for:

```
var infowindow = new google.maps.InfoWindow({
	content: data[i].comune,
	position : data[i].coordinate
    });
```

Abbiamo specificato il contenuto della infowindow e la posizione. Associamo ora un ascoltatore alla infowindow:

```
google.maps.event.addListener(marker,'click',
(function(marker,content,infowindow){
	return function() {
			infowindow.setContent(content);
			infowindow.open(map,marker);
	};})(marker,data[i].comune,infowindow));
```

Abbiamo creato una funzione anonima che riceve in ingresso tre parametri: marker, content e infowindow e abbiamo invocato questa funzione passando i parametri attuali marker, data[i].comune e infowindow.