GOOGLE API

Scarichiamo la libreria ufficiale di ***Google*** per le API.
Link - https://github.com/googleapis/google-cloud-php-vision

- Nella *bash*:

```
composer require google/cloud-vision
```

- Creiamo la *Migration*:

```
php artisan make:migration add_googlevision_fields_to_announcement_images
```

Qui possiamo scegliere quali API utilizzare:
Link - https://cloud.google.com/vision/docs/labels
Nel nostro caso Detect Labels e Detect Explicit Content.

- Nella *Migration* appena creata:

```
public function up(){
	Schema::table('announcement_images', function(Blueprint $table){
		$table->text(' labels')->nullable();

		$table->string('adult')->nullable();
		$table->string('spoof')->nullable();
		$table->string('medical')->nullable();
		$table->string('violence')->nullable();
		$table->string('racy')->nullable();
	});
}

public function down(){
	Schema::table('announcement_images', function (Blueprint $table){
		$table->dropColumn(['labels', 'adult', 'spoof', 'medical', 'violence',
                                                                'racy']);
	});
}
```

- Creiamo il Job:

```
php artisan make:job GoogleVisionSafeSearchImage
```

Su Google Vision API, creiamo un *nuovo progetto* >
APIs&Services > Library > Cerchiamo le librerie scelte > Abilita su questo Progetto
Credentials > Cloud Vision API > No > Nome, Viewer, Json

Inserisco nel progetto il file generato e lo rinominiamo ***google_credential.json***

- Nel Job creato in precedenza:

```
class GoogleVisionSafeSearchImage implements ShouldQueue{
	use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

	private $announcement_image_id;
	public function  __construct($announcement_image_id){
		$this->announcement_image_id = $announcement_image_id;
	}
	public function handle(){
		$i = AnnouncementImage::find($this->announcement_image_id);

		if(!$i){
			return;
		}
		$image = file_get_contents(storage_path('/app/' . $i->file));

		putenv('GOOGLE_APPLICATION_CREDENTIALS=' .
                                        base_path('google_credential.json'));

		$imageAnnotator = new ImageAnnotatorClient();
		$response = $imageAnnotator->safeSearchDetection($image);
		$imageAnnotator->close();

		$safe = $response->getSafeSearchAnnotation();
		$adult = $safe->getAdult();
		$medical = $safe->getMedical();
		$spoot = $safe->getSpoof();
		$violence = $safe->getViolence();
		$racy = $safe->getRacy();

		echo json_encode([$adult, $medical, $spoof, $violence, $racy]);

		$likelihoodname = [
			'UNKNOWN', 'VERY_UNLIKELY', 'UNLIKELY', 'POSSIBLE', 'LIKELY',
                                                        'VERY_LIKELY'];

		$i->adult = $likelihoodName[$adult];
		$i->medical = $likelihoodName[$medical];
		$i->spoof = $likelihoodName[$spoof];
		$i->violence = $likelihoodName[$violence];
		$i->racy = $likelihoodName[$racy];

		$i->save();
	}
}
```

- Nel *Tinker* facciamo una prova:

```
php artisan tinker
$j = new \App\Jobs\GoogleVisionImage(idImage);
$j->handle();
$j->save();
```

- Nel *Controller* alla funzione di creazione dell'articolo inseriamo dopo il ***save()***, quindi dopo che abbiamo un *id* dell'immagine:

```
dispatch(new GoogleVisionSafeSearchImage($i->id));
```

- Attiviamo il: `php artisan queue:work`

- Nella view del Revisor inseriamo il risultato della valutazione di Google:

```
@foreach($announcement->images as $image)
  adult: {{ $image->adult}}
  spoof: {{ $image->spoof}}
@endforeach
```

- Creiamo un secondo *Job* per *Detect Labels* chiamato *GoogleVisionLabelImage*:
Copiamo il codice *PHP* all'interno della descrizione sul sito, e importiamola nel nostro nuovo *job.*

NB. Rimane tutto uguale fino a dopo il ***putenv()***, poi:

```
$imageAnnotator = new ImageAnnotatorClient();
$response = $imageAnnotator->labelDetection($image);
$labels = $response->getLabelAnnotations();

if ($labels) {
      $result = [];
      foreach ($labels as $label) {
            $result[] = $label->getDescription();
      }

      $i->labels= $result;
      $i->save();
}
$imageAnnotator->close();
}
```

- Nello stesso modo di prima istruisco il Controller:

```
dispatch(new GoogleVisionLabelImage($i->id));
```

- Istruisco il Model *AnnouncementImage* per chiedergli di convertire in un *array* il campo Labels del *json* che riceverà.
Per maggiori informazioni il rimando e' alla sezione ***Eloquent Mutators*** della documentazione di Laravel:

```
class AnnouncementImage extends Model{
	protected $casts = [
		'labels' => 'array',
	];
...
}
```

Nella *view* del *revisor*:

```
@if($image->labels)
	@foreach($image->labels as $label)
		<li>{{$label}}</li>
	@endforeach
@endif
```

NB. Ricordiamoci di riavviare il *work* ad ogni modifica dei *jobs*.