CROP IMMAGINI CARICATE

Nella *Bash* installiamo la libreria di ***Spatie*** e successivamente attiviamo le *code*:

```
composer require spatie/image
php artisan queue:table
php artisan queue:failed-table
```

Creiamo un nuovo *Job* nella *Bash*:

```
php artisan make:job ResizeImage
```

Nel *.env*:

```
QUEUE_CONNECTION=sync
```

Nel file *Job* appena creato:

```
class ResizeImage implements ShouldQueue{
	use Dispatchable, InteractsWithQueue, Queueable, SerializesModel;

	private $path, $fileName, $w, $h;

	public function __construct($filePath, $w, $h){
		$this->path = dirname($filePath);
		$this->fileName = basename($filePath);
		$this->w = $w;
		$this->h = $h;
	}

	public function handle(){
		$w = $this->w;
		$h = $this ->h;
		$srcPath = storage_path() . '/app/' . $this->path . '/' .
                                                        $this->fileName;
		$destPath = storage_path() . '/app/' . $this->path .
                                    "/crop{$w}x{$h}_" . $this->fileName;

		Image::load($srcPath)
                    ->crop(Manipulations::CROP_CENTER, $w, $h)
                    ->save($destPath);
	}
}
```

path **.** Percorso dove abbiamo memorizzato il file
fileName . Nome dell'immagine
w/h . Altezza e Larghezza dell'immagine.
Eseguiamo il crop del ***srcPath*** *e lo finito sara' il* ***destPath*** *con quell'impostazione di visualizzazione.*
Image:: ->crop . Effettua il Crop dell'immagine.

- Nella funzione di creazione dell'articolo aggiungiamo sotto lo *Storage::* il richiamo al Job:

```
...
Storage::move($image, $newFileName);

dispatch( new ResizeImage( $newFileName, 300, 150));
dispatch( new ResizeImage( $newFileName, 400, 300));

$i->file = $newFileName;
...
```

newFileName . Path di dove l'abbiamo creata.

Per creare il ***Thumbnail*** andiamo nella funzione *uploadImage*:

```
...
fileName= $request->file('file')->store("public/temp/{$uniqueSecret}");
dispatch(new ResizeImage($fileName, 120, 120));
...
```

- Nel Model *AnnouncementImage* inseriamo una funzione per modificare il *path* in modo da richiamare quello corretto quando andiamo a richiamare l'immagine.

```
class AnnouncementImage extends Model{
     static public function getUrlByFilePath($filePath, $w = null, $h = null){
          if(!$w && !$h){
              return Storage::url($filePath);
          }

          $path = dirname($filePath);
          $filename = basename($filePath);
          $file = "{$path}/crop{$w}x{$h}_{$filename}";

          return Storage::url($file);
    }

     public function getUrl($w = null, $h = null){
          return AnnouncementImage::getUrlByFilePath($this->file, $w, $h);
     }
}
```

Passiamo l'url attraverso questa funzione statica per averlo nel formato che preferiamo. Se le dimensioni non sono indicate saranno null, e il path rimarrà lo stesso.

- Modifichiamo nella funzione *getImages()* per creare i *Thumbnail* quando per visualizzarli nel box dell'upload passandolo attraverso la *Funzione Statica* creata in precedenza:

```
foreach($images as $image){
        $data[] = [
                'id' => $image,
                'src' => AnnouncementImage::getUrlByFilePath($image, 120,120)
        ];
}
```

- Per utilizzare questa funzione (*non quella statica)* nel punto in una *view* in cui vogliamo visualizzare questa immagine, facciamo come segue:

```
                                                @foreach ($announcement->images as $image)
                        <img src="{{ $image->getUrl(300, 150)}}" class="rounded float-right"
                                                                                alt="">
@endforeach
```

Se volessi chiamare solo la prima immagine:

```
<div class="row mb-s">
        <div class="col-md-4">
                <img src="{{$article->images->first()->getUrl(400,300)}}"
                                                class="rounded float-right" alt="">
        </div>
</div>
```

- Impostiamo adesso il database in modalità ***asincrona*** per attivare le code.

File *.env*:

```
QUEUE_CONNECTION = database
```

Nella *bash* faccio un restart al server (*ctrl+c* e *php artisan serve*) e successivamente:

```
php artisan queue:work
```