CARICARE IMMAGINI

Per caricare immagini agli articoli inseriamo nell'HTML degli attributi al *tag form* e cambiamo il *type* e il *name* all'input.

```
<form enctype="multipart/form-data">
<input type="file" name="img">
```

enctype . Abilitiamo il form all'invio di dati complessi.
Aggiungiamo al ***Model*** il *name* dell'*input* tra i **$fillable.**
Ora creiamo un file ***Migrate*** che ci permetta di aggiungere una nuova colonna senza dover fare il rollback delle colonne già inserite.

php artisan:make migration add_img_column_to_articles_table

```
class AddImgColumnToArticlesTable extends Migration{

	public function up(){
		Schema::table('articles', function (Blueprint$table){
$table->string('img')->nullable();
		});
	}

	public function down(){
		Schema::table('articles', function (Blueprint $table){
			$table->dropColumn('img');
		});
	}
}
```

Schema::table . Vogliamo aggiungere una colonna ad una tabella già esistente
string() . Diamo indicazione al database che dovrà gestire una stringa perché il database non conterrà l'immagine, ma il *path*.
nullable() . Da un valore di *null* ad ogni valore che non ha un dato.
dropColumn() . Nel rollback eliminerà soltanto la colonna indicata.

Effettuiamo la ***migrazione***
- php artisan migrate

Nel ***controller*** gestiamo il dato in entrata:

```
public function submit(Request $request){
	$article = Article::create([
		'title' => $request->file('img')->store('public/img');
	}
}
```

NB *** *Per l'inserimento dell'immagine nel corpo dell'HTML controlla il paragrafo "Chiamata al Database".*

Dobbiamo collegare ora la cartella *public* in ***Storage*** dove e' caricata l'immagine, alla cartella *public* che vede il ***Client.***

- *php artisan storage:link*

Abbiamo creato così una ***cartella link*** *public/storage/img* collegata alla cartella *storage/public/img.*
Ora visualizziamo l'immagine correttamente.