SETTING DEL DATABASE IN LARAVEL

Nel file *.env* modifichiamo i setting del DB (*database*) con i dati corrispondenti appena creati (*database, username e password*).

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_POST=3306
DB_DATABASE=nomeDatabase
DB_USERNAME=root
DB_PASSWORD=root
```

MIGRATION

Se lavoriamo su un progetto con altre persone dovremo condividere insieme ad altri la struttura del database: per farlo creiamo una ***migration***.

La *migration* e' il file che mi permette di *creare* e *modificare* le tabelle, e lo creiamo dalla bash:

*php artisan make:migration create_**articles**_table*

***NB:** I nomi delle tabelle si usano al plurale, e la sintassi va eseguita come nell'esempio, altrimenti non segui la Convention Over Configuration, e tutti i benefici che porta.*

Abbiamo creato una cartella in *app/database/migrations/* con dei file per le tabelle di Users e Password, e il nostro file.

```
class CreateArticlesTable extends Migration{

	public function up(){
		Schema::create('articles', function (Blueprint $table){
			$table-.id();
			$table->string('name');
			$table->string('name');
			$table->text('name');
			$table->timestamps();
		});
	}

public function down(){
	Schema::dropIfExists('articles');
	}
}
```

up() . Funzione per creare una tabella
Schema . Classe che crea una tabella
::create . Metodo statico che crea. Vuole due parametri (nome tabella e funzione)
Blueprint . Classe di laravel che permette di creare colonne attraverso oggetto ***$table$***
*->**id()*** . Crea una colonna id
*->**timestamps()*** . Crea automaticamente due colonne (*created_at* e *updated_at*)
*->**string('name', 100)*** . Accetta una stringa e due parametri: *nome* della colonna e lunghezza massima (opzionale). Il *nome* che inseriamo è quello della *key* dell'array key=>value che riceviamo dal form.
*->**text('nome')*** . Accetta un testo più lungo. *name* come *string*
down() . Funzione per eliminare una colonna o l'intera tabella. Deve essere l'esatto contrario di *up()*.

Nella documentazione alla voce *Database Migrations / Creating Columns* abbiamo tutti i tipi di colonne che possiamo creare.

Fino ad ora abbiamo solo scritto le regole di creazione della tabella, ma non l'abbiamo ancora creata. Per farlo nella bash diamo il comando:

php artisan migrate . *Funzioni di up() . Lancia tutte le migrazioni nel database.*
php artisan migrate:rollback . *Funzioni di down()*
php artisan migrate:rollback --step=1 . *Rollback sull'ultima migrazione*

Apro **TablePlus** e creo una *nuova connessione* e inserisco i dati corrispondenti a quelli inseriti nel file *.env* quando ho fatto il setting del Database.
(Nome scelto, Host, User, Password, Nome Database)
Ora vediamo la tabella sul database.

La cartella ***migrations*** è una tabella contenente una cronologia dei *migrate* effettuati.