CRUD, PROCESSI e NUOVI AUTOMATISMI

Laravel e' un *framework*, e in quanto tale cerca di **facilitare** le procedure del nostro lavoro.
I tre aspetti legati al database sono il *Modello*, la *Migration* e il *Controller*.

Quando creiamo un ***model*** Laravel ci permette di creare il file ***migrate*** e il file ***controller*** ad esso corrispondenti aggiungendo "*-mcr*" al comando.

```
php artisan make:model Travel -mcr
```

Il file *Controller* che crea ci mette a disposizione il ***CRUD*** (*Create, Read, Update and Delete),* ovvero delle funzioni che rappresentano tutto ciò che si puo' eseguire su un'entità.
Le funzioni ***create() store() show() edit() update() destroy().***

```
Route::get('form/create', [TravelController::class, 'create'])->name('travel.create');
Route::post('/form/store',[TravelController::class, 'store'])->name('travel.store');
Route::get('/index', [TravelController::class, 'index'])->name('travel.index');
Route::get('/show/{travel}', [TravelController::class, 'show'])->name('travel.show');
```

Nel *TravelController*:

```
public function create(){
return view('travel.form')
}
```

Nel *Model*:

```
class Travel extends Model{
	protected $fillable =[
		'title',
		'description',
		'price',
		'img'
	];
}
```

Nel *TravelController*:

```
public function store(Request $request){
	$travel = Travel::create([
		'title'=> $request->title,
		'description'=> $request->description,
		'price'=>$request->price,
		'img'=>$request->file('img')->store('public/img')
	]);
	return redirect(route(''));
}
```

NB: Possiamo richiamare i dati della $request ***senza*** l'uso dell'***input*** $request->title

Nel *Migrations* facciamo il ***mass assignment***:

```
public function up(){
	Schema::create('travel', function (Blueprint $table){
		$table->id();
		$table->string('title');
		$table->text('description');
		$table->string('price');
		$table->string('img')
	}
}
```

Nella *bash*:

```
php artisan migrate
```

Inseriamo nella funzione ***index*** la lista dei nostri articoli caricati attraverso il form. Scegliamo l'*index* del *Controller* corretto a seconda del posizionamento del ***middleware***.

Nell'*index* faccio la ***chiamata* al database**:

```
public function index(){
	$travels = Travel::all();
	return view('travel.index', compact('travels');
}
```

Nel *controller* scelto il *CRUD* fa già una Dependency Injection:

```
public function show(Travel $travel){
	return view('travel.show', compact('travel'));
}
```

RELAZIONI TRA TABELLE

Le relazioni tra tabelle sono dei ***collegamenti** di record* tra tabelle differenti.
Aggiungiamo una ***foreign key*** ad una tabella, collegata al record di un'altra tabella.
Nel caso di un ID, alla cartella User.

Ci sono diversi tipi di relazioni tra tabelle:
- **One to One** . Un record user viene collegato ad un solo record della tabella travel
- **One to Many** . Un record user viene collegato a più travel..
- **Many to Many** . Più utenti associati a più travel.