MANY TO MANY

La relazione Many To Many collega attraverso una tabella *Pivot* collegamenti multipli di due tabelle.
Nella tabella ***Pivot*** saranno contenuti oltre all'*id* proprio della tabella Pivot, anche gli *id* dei due dati che vogliamo collegare corrispondenti a quelli nella propria tabella.

Creiamo un nuovo ***model*** e ad esso una *migration* correlata per creare una nuova tabella:

```
php artisan make:model Flight -m
```

Nel file *migration* della nuova tabella:

```
class CreateFlightsTable extends Migration{

        public function up(){
              Schema::create('flights', function (Blueprint $table){
                    $table->id();
                    $table->string('destination');
                   $table->timestamps();
              });

        $destinations = ['Madrid', 'New York', 'Dubai', 'Gaeta'];

        foreach ($destinations as $destination){
              $flight = new Flight();
              $flight->destination = $destination;
              $flight->save();
        }

}
```

foreach . Stiamo ciclando ogni destinazione dell'array per creare un nuovo *record*, attraverso il *model flight.*

Nel *Model*:

```
class Flight extends Model{
      protected $fillable = ['destination'];
}
```

Creiamo ora la tabella ***pivot***.

```
php artisan make:migration create_flight_travel_table
```

NB La tabella Pivot va al Singolare (a differenza delle altre tabelle), e l'ordine in cui le richiamo deve essere in ordine alfabetico. flight_travel (**NO** travel_flight)

Nel *Migrate* dovremo creare le colonne *user_id* e *role_id* ovvero le ***foreign key***:

```
class CreateFlightTravelTable extends Migration{

	public function up(){
		Schema::create('flight_travel', function (Blueprint $table){
			$table->id();
			$table->unsignedBigInteger('flight_id');

			$table->foreign('flight_id')->references('id')->on('flights');
			$table->unsignedBigInteger('travel_id');
			$table->foreign('travel_id')->references('id')->on('travel');
			$table->timestamps();
		});

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

Come nel *One to Many* creiamo prima la colonna, poi dichiariamo che e' ***foreign*** e la ***reference*** *(la colonna)* e ***on*** *(la tabella)***.**

Ora dobbiamo mettere in relazione i *Models*:

```
class Flight extends Model{
	protected $fillable = ['destination'];

	public function travels(){
		return $this->belongsToMany(Travel::Class);
	}
}
```

NB: Nel *Model Travel* sara' la stessa funzione ma sostituendo Travel con Flight.

Nel *TravelController* facciamo la chiamata al *database*:

```
public function create(){
	$flights = Flight::all();
	return view('travel.form', compact('flights'));
}
```

Nella *view* 'travel.form' facciamo scegliere all'utente quale dei voli scegliere ciclandoli:

```
<div class="form-group">
     <label>Voli</label>
     <select name="destination">
     @foreach($flights as $flight)
          <option value="{{$flight->id}}">{{$flight->destination}}</option>
     @endforeach
     </select>
</div>
```

value . L'*id* del *Flight* scelto nel menù a tendina *select>option*

Nel *TravelController* gestiamo ora il dato ricevuto dal *form.submit*:

```
     public function store(Request $request){
          $flightId = $request->destination;

          $travel = Travel::create([
               '...'
          ]);
          $travel->flights()->attach($flightId);
     return redirect(route('travel.index'));
     }
```

$flightId . E' il valore della destinazione scelta nel form, quindi l'*id*.
$travel->flights()->attach($flightId) . Relazione il $travel che abbiamo appena creato, e collega al valore appena selezionato.

Facciamo una *query* per mostrare il collegamento all'utente:

```
    public function show(Travel $travel){
          $flights = Travel::find($travel->id)->flights()->get();
          return view('travel.show', compact('travel', 'flights'));
    }
```

Travel::find . Cerca tra i Travel quello con l'*id* selezionato e mostrami tutti i *flights* ad esso collegato.
get() . Perché ci restituirà una collection
compact . Ricordiamoci di inserire la variabile.

Nella *view* show cicliamo la collection che riceviamo (anche se con un solo oggetto):

```
     @foreach ($flights as $flight)
          <p>{{$flight->destination}}</p>
```

```
@endforeach
```

Per permettere una ***modifica***, aggiorno nel *TravelController:*

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

Faccio una query per richiamare tutti i Flights per sceglierne uno nuovo.

```
public function update(Request $request, Travel $travel){
	...
	$travel->flights()->attach($request->destination);

	return redirect(route('travel.show', compact('travel')));
}
```

Specifico nell'update di sovrascrivere l'id vecchio con la $request->destination nuova.