

Corso Laravel

Model e Migration

Create DB

Via terminale

```
mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.7.24 MySQL Community Server (GPL)

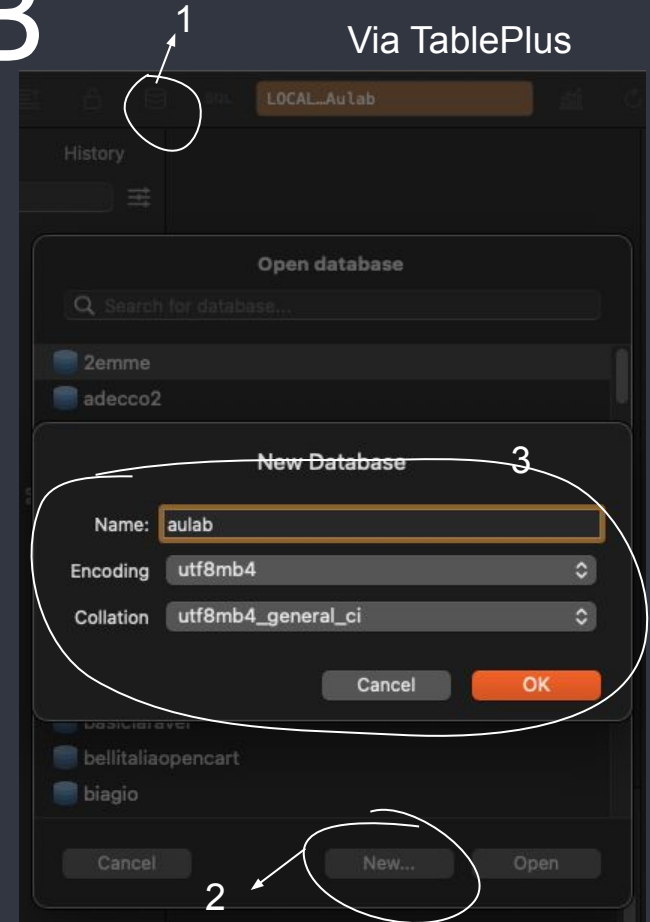
Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database aulab;
```

Via TablePlus



Collegare DB

Dentro il file .env andiamo ad impostare i dati precedentemente configurati in MySQL

```
DB_CONNECTION=mysql  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=database  
DB_USERNAME=root  
DB_PASSWORD=root
```

Schema Tabella

```
Table books{  
  id int [primary key]  
  name varchar(100)  
  pages int [default: null]  
  year int [default: null]  
}
```

books	
id 🔗	int
name	varchar(100)
pages 📖	int
year 📅	int

Comandi



#MAC

```
mysql -u root -p
```

#windows

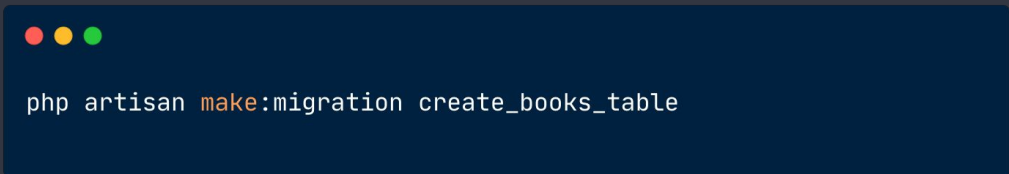
```
winpty mysql -u root -p
```

Migration per le tabelle

Utilizzare le Migrations nei tuoi progetti ti permetterà di tenere traccia, come succede in GIT, di tutte le modifiche effettuate al database da parte tua e dei tuoi colleghi.

Tutto ciò è possibile grazie alla **facade** Schema che, in modo agnostico, ti permetterà di creare e manipolare qualsivoglia tabella del Database.

Da console lancia il comando Artisan:



```
php artisan make:migration create_books_table
```

Recandoti in **database/migrations** potrai constatare tu stesso la comparsa di un nuovo file che avrà nel nome la data e l'ora di creazione concatenata al nome dichiarato nel comando, ad esempio:

AAAA_GG_MM_hhmmss_create_books_table.php

Convenzioni dei nomi

Cosa	Come	Giusto	Sbagliato
Controller	singolare	ArticleController	ArticlesController
Route	plurale	articles/1	article/1
Route name	snake_case con notazione punto	users.show_active	users.show-active, show-active-users
Model	singolare	User	Users
<u>Tabelle</u>	plurale	articles	article

Creare una tabella SQL

```
CREATE TABLE books (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    pages INT NOT NULL,  
    year INT NULL,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
);
```


Creare una tabella

```
public function up(): void
{
    Schema::create('books', function (Blueprint $table) {
        $table->id();
        $table->string('name', 100);
        $table->integer('pages');
        $table->integer('year')->nullable();
        $table->timestamps();
    });
}
```

La nostra tabella

```
Schema::create('books', function (Blueprint $table) {  
    $table->id();  
    $table->string('name', 100);  
    $table->integer('pages');  
    $table->timestamps();  
});
```

id	name	pages	created_at	updated_at
1	Divina Commedia	256	2023-0... ↕	2023-05... ↕

<https://laravel.com/docs/11.x/migrations#available-column-types>

Up() & down()

Migrate: Una volta definite le tabelle lato migration dovrai lanciarle.

Rollback: Per ripristinare l'ultima operazione di migrazione, è possibile utilizzare il comando rollback.


DA COMPLETARE con differenza tra rollback rollout e immagiin

```
public function up(): void
{
    Schema::create('books', function (Blueprint $table) {
        $table->id();
        $table->timestamps();
    });
}

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

Modificare una tabella

Come per la creazione, per modificare delle colonne già esistenti dovrai utilizzare il metodo `table` . Il metodo appena citato accetta due argomenti: il nome della tabella e la Closure Blueprint:



```
php artisan make:migration add_to_books_table
```

Model

Un Model non è altro che un file fisico .php in grado di astrarre il concetto di "tabella".

Scrivendo query in puro Eloquent potrai cambiare il Driver del database (MySQL, SQLite o PostgreSQL) senza impattare sulle query.

Di default potrai trovare i Model all'interno della cartella app/Models .

Tutti i modelli Eloquent estendono la classe **Illuminate\Database\Eloquent\Model** , questo significa che avrai accesso ad un numero sconfinato di metodi.

Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Book extends Model
{
    use HasFactory;

    protected $fillable = ['name', 'pages'];
}
```

Model -> Controller

```
public function index()
{
    $books = Book::all();

    return view('index', ['books' => $books]);
}
```

Store di un dato

Salvataggio PHP

```
<?php
```

```
public function store(Request $request){  
  
    $user = new User;  
    $user->firstname = $request->firstname;  
    $user->lastname = $request->lastname;  
    $user->save();  
}
```

Salvataggio Eloquent

```
<?php
```

```
public function store(Request $request){  
  
    //Primo Modo  
    Users::create($request->all());  
  
    //Secondo Modo  
    $request->validate([  
        'firstname'=> 'required',  
        'lastname'=> 'required',  
    ]);  
    Users::create( $request->validated());  
  
    //Secondo Modo  
    Properties::create([  
        'nome'=> $request->firstname,  
        'cognome'=> $request->lastname,  
    ]);  
}
```


Mass Assignment

```
protected $fillable = [  
    'name', 'pages'  
];
```

```
public function store(){  
    Books::create([  
        'name' => 'Libro 1',  
        'pages' => 45,  
        'gratis' => true  
    ])  
}
```

"gratis" verrà ignorato