

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/383948153>


# Laravel 11 Labs

Presentation · September 2024

CITATIONS  
0

READS  
976

1 author:



[Youcef Benabderrezak](#)  
University of Boumerdes

**131** PUBLICATIONS **13** CITATIONS

SEE PROFILE

Let's go step by step to set up a Laravel 11 project with PostgreSQL as the database, along with explanations in simple terms.

## Step 1: Install Laravel 11

### 1. Install Composer

- Composer is the tool that manages PHP dependencies, including Laravel.
- If Composer is not installed, install it by running:
  - **sudo apt update**
  - **sudo apt install curl php-cli php-mbstring git unzip**
  - **curl -sS https://getcomposer.org/installer | php**
  - **sudo mv composer.phar /usr/local/bin/composer**
- Check that Composer is installed:
  - **composer --version**

### 2. Create a New Laravel 11 Project

- Now that Composer is installed, create a new Laravel project by running:
  - **composer create-project --prefer-dist laravel/laravel my\_laravel\_app**
- Replace my\_laravel\_app with your project's name.
- Go to the new project directory:
  - **cd my\_laravel\_app**

### 3. Start Laravel Development Server

- Start Laravel's built-in development server by running:
  - **php artisan serve**
- This will start the server at <http://127.0.0.1:8000>.
- You can access your Laravel project in your browser.

## Step 2: Set Up PostgreSQL

### 1. Install PostgreSQL

- PostgreSQL is the database that we will use.

- To install PostgreSQL:
  - **sudo apt update**
  - **sudo apt install postgresql postgresql-contrib**

## 2. Set Up a PostgreSQL User and Database

- Log in to the PostgreSQL interactive terminal as the default user postgres:
  - **sudo -i -u postgres**
  - **psql**
- Create a new PostgreSQL user:
  - **CREATE USER my\_laravel\_user WITH PASSWORD 'secret\_password';**
- Create a new PostgreSQL database:
  - **CREATE DATABASE my\_laravel\_db;**
- Grant all privileges on the new database to the user:
  - **GRANT ALL PRIVILEGES ON DATABASE my\_laravel\_db TO my\_laravel\_user;**
- Exit PostgreSQL:
  - **\q**
  - **exit**

## Step 3: Connect Laravel to PostgreSQL

### 1. Configure the Database in .env

- Open the .env file in your project and change the following lines to match your PostgreSQL configuration:
  - **DB\_CONNECTION=pgsql**
  - **DB\_HOST=127.0.0.1**
  - **DB\_PORT=5432**
  - **DB\_DATABASE=my\_laravel\_db**
  - **DB\_USERNAME=my\_laravel\_user**
  - **DB\_PASSWORD=secret\_password**

## 2. Install PostgreSQL Driver for PHP

- Install the PostgreSQL driver for PHP by running:
  - **sudo apt install php-pgsql**

## Step 4: Create and Run Migrations

Migrations are how Laravel creates and modifies tables in your database.

### 1. Create a Migration for a Table

- To create a migration file for a books table (for a book's ISBN, title, description, and image):
  - **php artisan make:migration create\_books\_table**

### 2. Modify the Migration File

- Open the migration file in the **database/migrations** folder and define the schema for the books table:

```
public function up()
{
    Schema::create('books', function (Blueprint $table) {
        $table->id();
        $table->string('isbn');
        $table->string('title');
        $table->text('description');
        $table->string('image')->nullable();
        $table->timestamps();
    });
}
```

### 3. Run the Migration

- Run the migration to create the table in your PostgreSQL database:
  - **php artisan migrate**

### Step 5 : Create a Book Model and Controller

Laravel uses models to interact with the database.

#### 1. Create the Book Model

- **php artisan make:model Book**

#### 2. Create a Controller

- Create a controller for handling CRUD operations:
  - **php artisan make:controller BookController --resource**
- This command creates a BookController that includes basic methods like index, store, update, destroy, etc.

### Step 6: Define Routes for Books

#### 1. Add Routes to routes/web.php

- Define routes for handling book operations:

```
use App\Http\Controllers\BookController;  
Route::resource('books', BookController::class);
```

### Step 7: Implement CRUD Operations in the Controller

#### 7.1 Implement CRUD in BookController

- Open app/Http/Controllers/BookController.php and implement the CRUD methods.
- Here's an example for each method:

```
use App\Models\Book;  
use Illuminate\Http\Request;  
  
class BookController extends Controller  
{  
    // Display a list of books  
    public function index()  
    {  
        $books = Book::all();
```

```
        return view('books.index', compact('books'));
    }

    // Show the form for creating a new book
    public function create()
    {
        return view('books.create');
    }

    // Store a new book in the database
    public function store(Request $request)
    {
        $book = new Book();
        $book->isbn = $request->isbn;
        $book->title = $request->title;
        $book->description = $request->description;
        $book->image = $request->image;
        $book->save();
        return redirect()->route('books.index');
    }

    // Show a specific book
    public function show(Book $book)
    {
        return view('books.show', compact('book'));
    }

    // Edit a book
    public function edit(Book $book)
    {
        return view('books.edit', compact('book'));
    }
}
```

```
// Update a book
public function update(Request $request, Book $book)
{
    $book->isbn = $request->isbn;
    $book->title = $request->title;
    $book->description = $request->description;
    $book->image = $request->image;
    $book->save();
    return redirect()->route('books.index');
}

// Delete a book
public function destroy(Book $book)
{
    $book->delete();
    return redirect()->route('books.index');
}
}
```

## Step 8: Create Views

### 1. Create Blade Views for Books

- In the **resources/views/books** folder, create these view files:
  - index.blade.php (to list books).
  - create.blade.php (to create a new book).
  - edit.blade.php (to edit a book).
  - show.blade.php (to display a single book).

Page	Content
index	<pre> &lt;h1&gt;Book List&lt;/h1&gt;  &lt;!-- Link to add a new book --&gt; &lt;a href="{{ route('books.create') }}" class="btn btn-primary mb-3"&gt;Add New Book&lt;/a&gt;  @if(\$books-&gt;count()) &lt;table class="table table-bordered"&gt; &lt;thead&gt; &lt;tr&gt; &lt;th&gt;ISBN&lt;/th&gt; &lt;th&gt;Title&lt;/th&gt; &lt;th&gt;Description&lt;/th&gt; &lt;th&gt;Actions&lt;/th&gt; &lt;/tr&gt; &lt;/thead&gt; &lt;tbody&gt;     @foreach(\$books as \$book)         &lt;tr&gt;             &lt;td&gt;{{ \$book-&gt;isbn }}&lt;/td&gt;             &lt;td&gt;{{ \$book-&gt;title }}&lt;/td&gt;             &lt;td&gt;{{ \$book-&gt;description }}&lt;/td&gt;             &lt;td&gt;                 &lt;!-- Link to show book details --&gt;                 &lt;a href="{{ route('books.show', \$book-&gt;id) }}" class="btn btn-info"&gt;Show&lt;/a&gt;                  &lt;!-- Link to edit book --&gt;                 &lt;a href="{{ route('books.edit', \$book-&gt;id) }}" class="btn btn-warning"&gt;Edit&lt;/a&gt;                  &lt;!-- Delete form --&gt;                 &lt;form action="{{ route('books.destroy', \$book-&gt;id) }}" method="POST" style="display:inline;"&gt;                     @csrf                     @method('DELETE')                     &lt;button type="submit" class="btn btn-danger"&gt;Delete&lt;/button&gt;                 &lt;/form&gt;             &lt;/td&gt;         &lt;/tr&gt;     @endforeach &lt;/tbody&gt; &lt;/table&gt; </pre>



	<pre>         @else         &lt;p&gt;No books available.&lt;/p&gt;         @endif </pre>
create	<pre> &lt;!-- resources/views/books/create.blade.php --&gt;  @extends('layouts.app')  @section('content')     &lt;h1&gt;Add New Book&lt;/h1&gt;      &lt;!-- Book creation form --&gt;     &lt;form action="{{ route('books.store') }}" method="POST"     enctype="multipart/form-data"&gt;         @csrf          &lt;div class="mb-3"&gt;             &lt;label for="isbn" class="form-label"&gt;ISBN&lt;/label&gt;             &lt;input type="text" class="form-control" id="isbn" name="isbn" required&gt;         &lt;/div&gt;          &lt;div class="mb-3"&gt;             &lt;label for="title" class="form-label"&gt;Title&lt;/label&gt;             &lt;input type="text" class="form-control" id="title" name="title" required&gt;         &lt;/div&gt;          &lt;div class="mb-3"&gt;             &lt;label for="description" class="form-label"&gt;Description&lt;/label&gt;             &lt;textarea class="form-control" id="description" name="description"             required&gt;&lt;/textarea&gt;         &lt;/div&gt;          &lt;div class="mb-3"&gt;             &lt;label for="image" class="form-label"&gt;Book Image&lt;/label&gt;             &lt;input type="file" class="form-control" id="image" name="image"&gt;         &lt;/div&gt;          &lt;button type="submit" class="btn btn-primary"&gt;Submit&lt;/button&gt;     &lt;/form&gt;     @endsection </pre>
edit	<pre> &lt;!-- resources/views/books/edit.blade.php --&gt;  @extends('layouts.app') </pre>

```

@section('content')
    <h1>Edit Book</h1>

    <!-- Book editing form -->
    <form action="{{ route('books.update', $book->id) }}" method="POST"
    enctype="multipart/form-data">
        @csrf
        @method('PUT')

        <div class="mb-3">
            <label for="isbn" class="form-label">ISBN</label>
            <input type="text" class="form-control" id="isbn" name="isbn" value="{{
$book->isbn }}" required>
        </div>

        <div class="mb-3">
            <label for="title" class="form-label">Title</label>
            <input type="text" class="form-control" id="title" name="title" value="{{
$book->title }}" required>
        </div>

        <div class="mb-3">
            <label for="description" class="form-label">Description</label>
            <textarea class="form-control" id="description" name="description"
required>{{ $book->description }}</textarea>
        </div>

        <div class="mb-3">
            <label for="image" class="form-label">Book Image</label>
            <input type="file" class="form-control" id="image" name="image">
        </div>

        <button type="submit" class="btn btn-success">Update</button>
    </form>
@endsection

```

### Homework

- Set up a fresh Laravel last version project.
- Create a migration, model, and controller for Task with the following fields:
  - **id: Primary Key (Auto-increment)**

- **title: String, required**
- **description: Text, optional**
- **status: Enum ('pending', 'completed')**
- **created\_at: Timestamp**
- **updated\_at: Timestamp**
- Implement the following pages:
  1. Task List Page : List all tasks, with options to create a new task, edit, delete, and mark tasks as completed.
  2. Create Task Page : A form to add a new task.
  3. Edit Task Page : A form to edit an existing task.
  4. Show Task Page : Display detailed information about a single task.
- Set up routes to handle:
  - Viewing all tasks
  - Creating a new task
  - Editing an existing task
  - Deleting a task
  - Marking a task as completed
- Use the Eloquent ORM for database interactions.