Contents

LIVT stack (Laravel, Inertia, Vue, Tailwind).

Why this stack?

- Laravel: Handles the backend (database, routing, logic).
- Vue.js: Handles the frontend (interactivity).
- Inertia.js: The "glue." It allows you to build a Single Page Application (SPA) using classic server-side routing. You don't need a separate API; you return Vue pages directly from Laravel controllers.
- Tailwind: Utility-first CSS for styling.

Prerequisites

Before we start, ensure you have installed:

1. PHP and Composer
2. Node.js and NPM

To save hours of manual configuration, we will use Laravel Breeze. It creates the skeleton for this specific stack automatically.

Project 1: Demo Static Site

Setup

1. Create the project:

```
composer create-project laravel/laravel project1
cd project1
```

2. Add inertia packages

```
composer require inertiajs/inertia-laravel
```

3. Install the inertia middleware

```
php artisan inertia:middleware
```

In Laravel 11/12, append this middleware to your bootstrap/app.php file.

```
...
use App\Http\Middleware\HandleInertiaRequests;
...
...
  ->withMiddleware(function (Middleware $middleware): void {
     HandleInertiaRequests::class;
  })
...
```

4. Install vue packages and plugin

```
npm install @inertiajs/vue3 vue @vitejs/plugin-vue
```

Then register the plugin in your configuration file

vite.config.js

```
...
import vue from '@vitejs/plugin-vue'; // <--- 1. IMPORT THIS
...
...
...
    vue({ // <--- 2. ADD THIS FUNCTION CALL
      template: {
        transformAssetUrls: {
          base: null,
          includeAbsolute: false,
        },
      },
    }),
...
```

5. Install the Ziggy library (which allows Laravel routes to be used in JavaScript)

```
composer require tightenco/Ziggy
php artisan view:clear
# then restart your server
```

6. Initialize the App

Open resources/js/app.js and paste the standard initialization code:

```
import { createApp, h } from 'vue'
import { createInertiaApp } from '@inertiajs/vue3'

createInertiaApp({
 resolve: name => {
  const pages = import.meta.glob('./Pages/**/*.vue', { eager: true })
  return pages[`./Pages/${name}.vue`]
 },
 setup({ el, App, props, plugin }) {
  createApp({ render: () => h(App, props) })
   .use(plugin)
   .mount(el)
 },
})
```

7. Install tailwindcss and initialize

```
npm install -D tailwindcss@3 postcss autoprefixer
npx tailwindcss init -p
```

Update tailwind.config.js to look for your Vue files:

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./resources/**/*.blade.php",
    "./resources/**/*.js",
    "./resources/**/*.vue",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Add Directives:
Create a CSS file at resources/css/app.css and add:

```
(overwrite the existing code with the following)
@tailwind base;
@tailwind components;
@tailwind utilities;
```

(Ensure this CSS file is imported in your resources/js/app.js via import '../css/app.css';)

8. Start the servers (keep two terminal windows open):
    - Terminal A: php artisan serve (Backend)
    - Terminal B: npm run dev (Frontend compiler)

Root Template

Inertia needs a **Root Template** (a single Blade file) to load your Vue application.

resources/views/app.blade.php

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title inertia>{{ config('app.name', 'Laravel') }}</title>

    <link rel="preconnect" href="https://fonts.bunny.net">
    <link href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap" rel="stylesheet"
/>

    @routes
    @vite(['resources/js/app.js', "resources/js/Pages/{$page['component']}.vue"])
```

```
    @inertiaHead
  </head>
  <body class="font-sans antialiased">
    @inertia
  </body>
</html>
```

The Shared Layout (NavBar)

In Inertia, we use Layouts to keep the navigation persistent while the content changes.

Create a file: resources/js/Layouts/MainLayout.vue

```
<script setup>
import { Link } from '@inertiajs/vue3';
</script>

<template>
  <div class="min-h-screen bg-gray-100 font-sans">
    <nav class="bg-indigo-600 text-white p-4">
      <div class="max-w-4xl mx-auto flex justify-between items-center">
        <div class="font-bold text-xl">MyProject</div>
        <div class="space-x-4">
          <Link href="/" class="hover:text-indigo-200">Home</Link>
          <Link href="/content" class="hover:text-indigo-200">Content</Link>
          <Link href="/about" class="hover:text-indigo-200">About</Link>
        </div>
      </div>
    </nav>

    <main class="max-w-4xl mx-auto mt-6 p-6 bg-white shadow rounded">
      <slot /> </main>
  </div>
</template>
```

Creating the Pages

Create three files in resources/js/Pages/.
1. Home.vue

```
<script setup>
import MainLayout from '@/Layouts/MainLayout.vue';
import { Head } from '@inertiajs/vue3';
</script>
```

```
<template>
  <Head title="Home" />
  <MainLayout>
    <h1 class="text-3xl font-bold text-gray-800">Welcome Home</h1>
    <p class="mt-4 text-gray-600">This is the landing page built with Laravel and Vue.</p>
  </MainLayout>
</template>
```

2. Content.vue

```
<script setup>
import MainLayout from '@/Layouts/MainLayout.vue';
import { Head } from '@inertiajs/vue3';
</script>

<template>
  <Head title="Content" />
  <MainLayout>
    <h1 class="text-3xl font-bold text-green-600">Our Content</h1>
    <div class="mt-4 p-4 bg-green-50 border border-green-200 rounded">
      <p>Here is some featured content using Tailwind utility classes.</p>
    </div>
  </MainLayout>
</template>
```

3. About.vue

```
<script setup>
import MainLayout from '@/Layouts/MainLayout.vue';
import { Head } from '@inertiajs/vue3';
</script>

<template>
  <Head title="About Us" />
  <MainLayout>
    <h1 class="text-3xl font-bold text-blue-600">About Us</h1>
    <p class="mt-4">We are learning the TALL stack step-by-step.</p>
  </MainLayout>
</template>
```

Define Routes

Open routes/web.php. We map URLs directly to the Vue components using Inertia::render.

```
<?php

use Illuminate\Support\Facades\Route;
use Inertia\Inertia;
```

```
// Route 1: Home
Route::get('/', function () {
   return Inertia::render('Home');
});

// Route 2: Content
Route::get('/content', function () {
   return Inertia::render('Content');
});

// Route 3: About
Route::get('/about', function () {
   return Inertia::render('About');
});
```

Test it: Go to http://127.0.0.1:8000. Click the links. Notice the page does not reload (the browser icon doesn't spin)? That is Inertia working.

Project 2: Simple Form Submission Demo

1. Setup Laravel project same as Project 1
2. Setup Inertia, Vue, and Tailwind packages and configuration
3. Create routes

```
Route::get('/page1', [TestController::class, 'page1']);
Route::post('/page2', [TestController::class, 'page2']);
```

4. Create a vue page (form)

resources/js/Pages/Test/Page1.vue

```
<script setup>
import { useForm } from "@inertiajs/vue3";

const form = useForm({
  unitprice: "",
  quantity: "",
});

const submit = () => {
  form.post("/page2");
};
</script>

<template>
  <div>
    <h1>Page 1</h1>
```

```
  <form @submit.prevent="submit">
   <input v-model="form.unitprice" type="number" placeholder="Unit Price" />
   <input v-model="form.quantity" type="number" placeholder="Quantity" />
   <button type="submit">Submit</button>
  </form>
 </div>
</template>
```

5. Create controller

```
php artisan make:controller TestController
```

add the following functions

```
...
  public function page1()
  {
    return Inertia::render('Test/Page1');
  }

  public function page2(Request $request)
  {
    $total = $request['unitprice'] * $request['quantity'];
    return Inertia::render('Test/Page2')->with('total', $total);
  }
...
```

6. Create the page to display the output

resources/js/Pages/Test/Page2.vue

```
<script setup>
import { Link } from "@inertiajs/vue3";

defineProps({
 total: Number,
});
</script>

<template>
 <div>
  <h1>Page 2</h1>
  <p>Total: {{ total }}</p>
  <Link href="/page1" class="text-gray-500 text-sm mb-4 inline-block"
   >&larr; Back to list</Link
  >
 </div>
</template>
```

Project 3: Form Processing App

Setup & Database

1. Create project2 (follow the same installation steps as Project 1).
2. Set up your .env file to connect to a database (SQLite is easiest for learning, or MySQL).
3. Create the database model and migration:

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

4. Open the migration file in database/migrations/xxxx_create_feedback_table.php:

```
public function up(): void
{
   Schema::create('feedback', function (Blueprint $table) {
      $table->id();
      $table->string('name');
      $table->string('email');
      $table->text('message');
      $table->timestamps();
   });
}
```

5. Run migration: php artisan migrate

The Controller

We need logic to handle showing the form and saving it.

```
php artisan make:controller FeedbackController
```

Open app/Http/Controllers/FeedbackController.php:

```
<?php

namespace App\Http\Controllers;

use App\Models\Feedback;
use Illuminate\Http\Request;
use Inertia\Inertia;

class FeedbackController extends Controller
{
   // Page 1: Show the form
   public function create() {
      return Inertia::render('Feedback/Create');
   }
```

```
    // Logic: Store the data
    public function store(Request $request) {
        // 1. Validate
        $validated = $request->validate([
            'name' => 'required|max:50',
            'email' => 'required|email',
            'message' => 'required|min:10'
        ]);

        // 2. Save
        Feedback::create($validated);

        // 3. Redirect
        return redirect('/success');
    }

    // Page 2: Success Message
    public function success() {
        return Inertia::render('Feedback/Success');
    }

    // Page 3: List all feedback (Admin)
    public function index() {
        return Inertia::render('Feedback/Index', [
            'feedbackList' => Feedback::latest()->get()
        ]);
    }

    // Page 4: Show single feedback detail
    public function show($id) {
        return Inertia::render('Feedback/Show', [
            'feedback' => Feedback::findOrFail($id)
        ]);
    }
}
```

Note: Allow mass assignment in app/Models/Feedback.php: protected $guarded = [];

The Routes

Open routes/web.php:

```
use App\Http\Controllers\FeedbackController;

Route::get('/', [FeedbackController::class, 'create']); // Home is the form
Route::post('/feedback', [FeedbackController::class, 'store']); // Post action
Route::get('/success', [FeedbackController::class, 'success']);
```

```
Route::get('/admin/feedback', [FeedbackController::class, 'index']);
Route::get('/admin/feedback/{id}', [FeedbackController::class, 'show']);
```

The Vue Pages (The Frontend)

Create a folder resources/js/Pages/Feedback.
1. Create.vue (The Form) This uses the useForm hook, which is crucial for handling form data and errors in Inertia.

```
<script setup>
import { useForm } from '@inertiajs/vue3';

// Initialize form data
const form = useForm({
  name: '',
  email: '',
  message: ''
});

const submit = () => {
  form.post('/feedback');
};
</script>

<template>
  <div class="max-w-md mx-auto mt-10 p-6 bg-white shadow-lg rounded-lg">
    <h1 class="text-2xl font-bold mb-6">Submit Feedback</h1>

    <form @submit.prevent="submit">
      <div class="mb-4">
        <label class="block text-gray-700">Name</label>
        <input v-model="form.name" type="text" class="w-full border p-2 rounded" />
        <div v-if="form.errors.name" class="text-red-500 text-sm">{{ form.errors.name }}</div>
      </div>

      <div class="mb-4">
        <label class="block text-gray-700">Email</label>
        <input v-model="form.email" type="email" class="w-full border p-2 rounded" />
        <div v-if="form.errors.email" class="text-red-500 text-sm">{{ form.errors.email }}</div>
      </div>

      <div class="mb-4">
        <label class="block text-gray-700">Message</label>
        <textarea v-model="form.message" class="w-full border p-2 rounded"></textarea>
        <div v-if="form.errors.message" class="text-red-500 text-sm">{{ form.errors.message }}</div>
      </div>
```

```
        <button type="submit" :disabled="form.processing" class="bg-blue-600 text-white px-4 py-2
rounded hover:bg-blue-700">
          Submit
        </button>
      </form>
    </div>
</template>
```

2. Success.vue

```
<script setup>
import { Link } from '@inertiajs/vue3';
</script>
<template>
  <div class="text-center mt-20">
    <h1 class="text-4xl text-green-600 font-bold">Thank You!</h1>
    <p class="mt-4">Your feedback has been received.</p>
    <Link href="/" class="mt-6 inline-block text-blue-500 underline">Send Another</Link>
    <br>
    <Link href="/admin/feedback" class="mt-2 inline-block text-gray-500 text-sm">View Admin
Panel</Link>
  </div>
</template>
```

3. Index.vue (Admin List) Notice how we receive feedbackList as a prop. This data came from the Laravel Controller.

```
<script setup>
import { Link } from '@inertiajs/vue3';

defineProps({
  feedbackList: Array
});
</script>

<template>
  <div class="max-w-2xl mx-auto mt-10">
    <h1 class="text-2xl font-bold mb-4">Admin: Feedback List</h1>

    <div v-for="item in feedbackList" :key="item.id" class="border-b p-4 hover:bg-gray-50">
      <div class="flex justify-between">
        <span class="font-bold">{{ item.name }}</span>
        <span class="text-gray-500 text-sm">{{ new Date(item.created_at).toLocaleDateString() }}</span>
      </div>
      <p class="truncate text-gray-600">{{ item.message }}</p>
      <Link :href="` /admin/feedback/${item.id}` " class="text-blue-500 text-sm">View Details</Link>
```

```
    </div>
  </div>
</template>
```

4. Show.vue (Detail View)

```
<script setup>
import { Link } from '@inertiajs/vue3';

defineProps({
  feedback: Object
});
</script>

<template>
  <div class="max-w-xl mx-auto mt-10 p-6 border rounded shadow">
    <Link href="/admin/feedback" class="text-gray-500 text-sm mb-4 inline-block">&larr; Back to
list</Link>

    <h1 class="text-3xl font-bold">{{ feedback.name }}</h1>
    <p class="text-blue-600 mb-6">{{ feedback.email }}</p>

    <div class="bg-gray-100 p-4 rounded">
      {{ feedback.message }}
    </div>
  </div>
</template>
```