

Universidad San Carlos de Guatemala

Sistema de Base de Datos 2

Auxiliar: Saul Jafet Menchu Recinos

Guatemala, 20 de Junio de 2025



PROYECTO NO.1
(MANUAL TÉCNICO)
Grupo #8

Javier Isaías Coyoy Marín 3152-38488-0901

Eiler Rigoberto Gómez Figueroa 3249-40033-1332

MANUAL TÉCNICO

VUE

Vue.js es un framework utilizado para construir interfaces de usuario interactivas. Fue creado en el año 2014 y, desde entonces, se ha consolidado como una de las herramientas más utilizadas en el desarrollo frontend moderno. Vue destaca por su curva de aprendizaje suave, su enfoque declarativo, su sistema de reactividad eficiente y su arquitectura basada en componentes reutilizables, lo que facilita la construcción de interfaces modulares, escalables y mantenibles.

A diferencia de otros frameworks más complejos o rígidos, Vue puede integrarse de forma gradual en proyectos existentes, lo que permite su adopción parcial o total según las necesidades del equipo de desarrollo. Asimismo, la comunidad activa y la excelente documentación oficial han contribuido significativamente a su crecimiento y adopción tanto en entornos académicos como empresariales.

Vite

Para este proyecto se utilizó Vue combinado con Vite, una herramienta de desarrollo moderna creada por el mismo autor de Vue.js. Vite destaca por su rapidez al iniciar el servidor de desarrollo y por su sistema de recarga en caliente (Hot Module Replacement), que permite ver los cambios de forma instantánea sin recargar toda la página. A diferencia de herramientas tradicionales como Webpack, Vite utiliza módulos ES nativos durante el desarrollo y Rollup para la construcción final, lo que mejora notablemente el rendimiento y reduce los tiempos de espera. Gracias a su configuración sencilla y su eficiencia, Vite se ha convertido en una opción ideal para desarrollar aplicaciones web modernas con Vue.

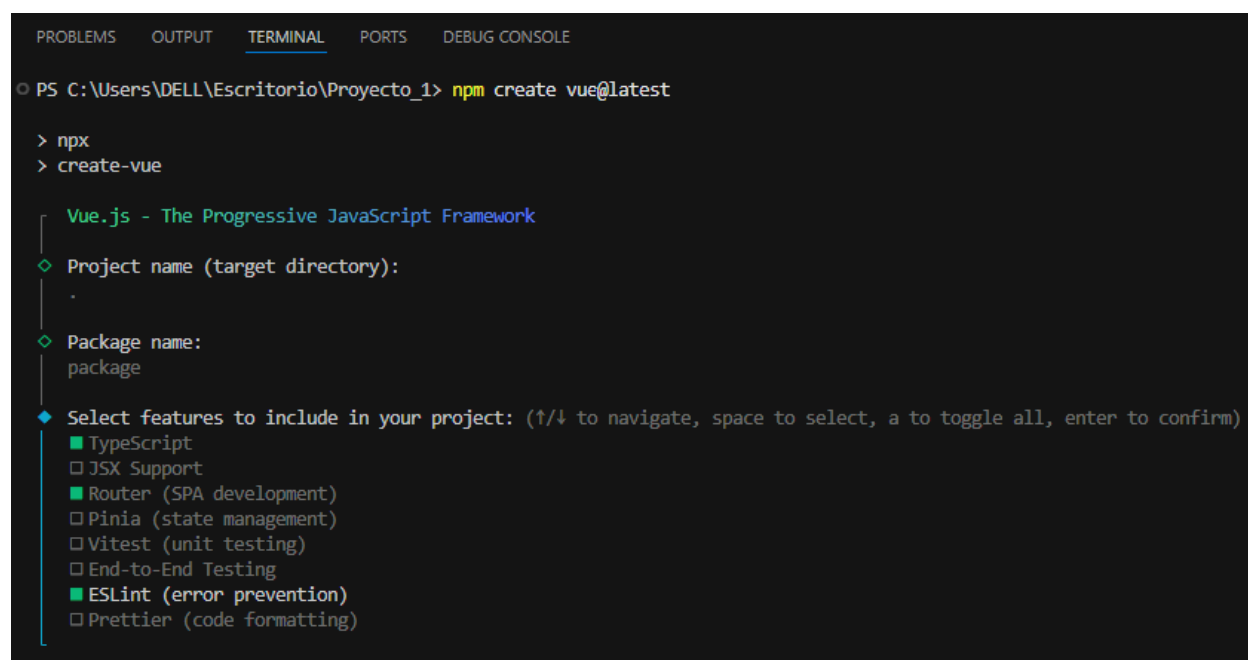
Vuetify

Para la construcción de la interfaz de usuario del proyecto se utilizó Vuetify, un framework de componentes que sigue las especificaciones de Material Design. Su integración con Vue.js es directa y potente, permitiendo el uso de componentes

visuales predefinidos que se adaptan automáticamente a diferentes resoluciones y dispositivos. Esto redujo significativamente el tiempo de desarrollo de la interfaz y facilitó mantener una apariencia profesional y coherente en toda la aplicación.

Además, Vuetify proporciona funcionalidades adicionales como validación de formularios, diseño responsivo con sistema y temas personalizables, lo que permitió construir vistas dinámicas y adaptables sin tener que recurrir a librerías CSS externas. Su enfoque en la accesibilidad y buenas prácticas de diseño también ayudó a mejorar la experiencia del usuario final de manera notable.

Para utilizar Vue.js en un proyecto, es necesario contar con un entorno de desarrollo básico configurado previamente. Esto incluye tener instalado Node.js y npm (Node Package Manager), ya que Vue se basa en módulos y herramientas que requieren un gestor de paquetes. También se recomienda utilizar un editor de código como Visual Studio Code, que ofrece integración con extensiones útiles para Vue. Una vez instalado Node. Para iniciar con el proyecto seleccionamos la carpeta y realizamos los siguiente:



```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
PS C:\Users\DELL\Escritorio\Proyecto_1> npm create vue@latest

> npx
> create-vue

  Vue.js - The Progressive JavaScript Framework
  ◇ Project name (target directory):
    .
  ◇ Package name:
    package
  ◆ Select features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm)
    ■ TypeScript
    □ JSX Support
    ■ Router (SPA development)
    □ Pinia (state management)
    □ Vitest (unit testing)
    □ End-to-End Testing
    ■ ESLint (error prevention)
    □ Prettier (code formatting)
```

Durante la creación del proyecto con `npm create vue@latest`, hicimos las siguientes selecciones:

Nombre del proyecto

En lugar de escribir un nombre, escribimos `.` (punto), lo que significa que el proyecto se creó directamente en la carpeta actual. Luego, como nombre del paquete, escribimos `"package"`, que es el identificador que quedó en el archivo `package.json`. Este nombre es útil si se quiere publicar el proyecto como un paquete en npm o simplemente para identificar el proyecto.

TypeScript

Seleccionamos TypeScript para tener tipado estático en nuestro código. Esto nos ayuda a detectar errores desde el editor, escribir código más seguro y mantener mejor el proyecto a medida que crece. Esto también agregó un archivo `tsconfig.json` y permite usar `lang="ts"` dentro de nuestros componentes `.vue`.

Vue Router (SPA Mode)

Agregamos Vue Router en modo SPA (Single Page Application), lo que nos permite tener varias "páginas" dentro de la aplicación sin necesidad de recargar toda la web. Con esto se crearon rutas básicas como `/` (Home) y `/about`, y se generó una carpeta `/views` con las vistas asociadas. También se añadió automáticamente el archivo `src/router/index.ts` donde definimos nuestras rutas y su comportamiento.

ESLint

Activamos ESLint, que es una herramienta que revisa nuestro código en busca de errores, malas prácticas o estilos inconsistentes. Esto nos ayuda a escribir un código más limpio y uniforme.

Dependencias

Para este proyecto, instalamos y utilizamos una serie de dependencias que permiten ampliar y mejorar las funcionalidades del entorno base generado por Vue. Estas dependencias fueron esenciales tanto para el diseño visual como para la gestión de datos y la representación gráfica. A continuación se detallan:

- **npm install:** Este comando se ejecutó inicialmente para instalar todas las dependencias definidas en el archivo package.json, incluyendo Vue y las herramientas seleccionadas durante la creación del proyecto como Vite, TypeScript, Vue Router y ESLint.
- **npm install vuetify@next @mdi/font:** Se instaló Vuetify, un framework de componentes basado en Material Design, compatible con Vue 3 (por eso se especifica @next). También se añadió @mdi/font, que proporciona el conjunto de íconos Material Design Icons, utilizados junto con Vuetify para mejorar la presentación visual de la interfaz.
- **npm install axios:** Axios es una librería que facilita la realización de peticiones HTTP. En este proyecto, la utilizamos para conectarnos a APIs o servicios backend, lo cual nos permitió consumir datos externos de manera sencilla.
- **npm install chart.js:** Esta dependencia se utilizó para incorporar gráficas interactivas dentro del sistema. Chart.js es una biblioteca liviana y muy popular para generar gráficos como barras, líneas, tortas, entre otros, totalmente personalizables desde Vue.
- **npm install chartjs-plugin-zoom:** Este complemento se integró con Chart.js para permitir funciones avanzadas como zoom y desplazamiento (pan) dentro de las gráficas. Esto mejoró la experiencia del usuario al visualizar grandes volúmenes de datos o realizar análisis más detallados en las visualizaciones.

Iniciar Proyecto

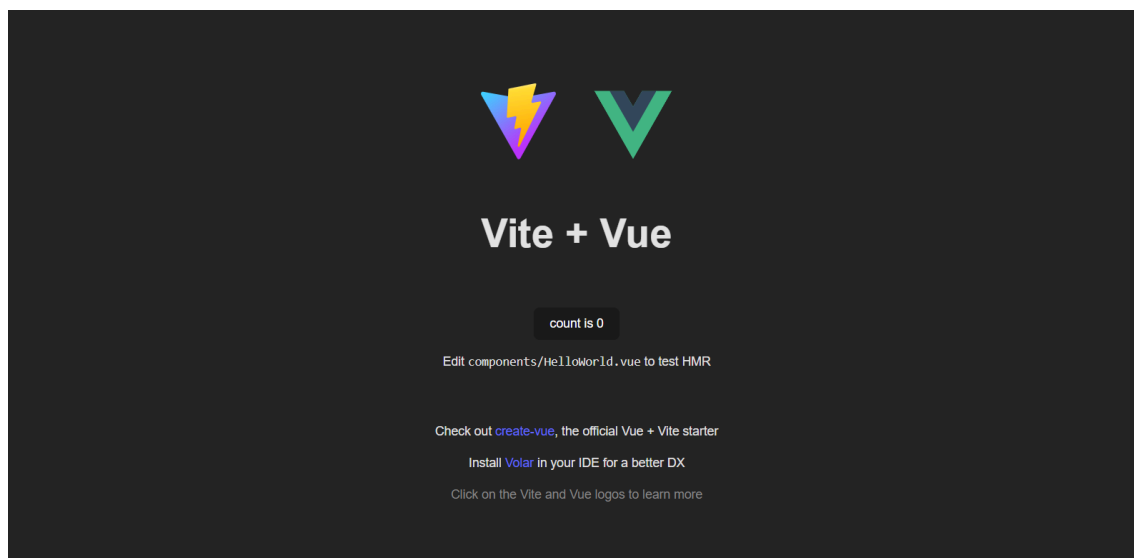
Para iniciar el proyecto en modo de desarrollo utilizamos el comando **npm run dev**. Este comando ejecuta el servidor de desarrollo proporcionado por Vite, permitiendo que la aplicación Vue se compile de forma rápida y se actualice automáticamente cada vez que se realicen cambios en el código. Gracias a esta funcionalidad, conocida como Hot Module Replacement (HMR), no es necesario recargar manualmente la página para ver los resultados, lo que agiliza considerablemente el proceso de desarrollo. Además, este entorno de desarrollo incluye herramientas útiles como mensajes de error detallados en consola y una vista previa en tiempo real en el navegador, lo que facilita la detección y corrección de errores durante la construcción de la aplicación.

```
PS C:\Users\DELL\Escritorio\Base de Datos 2 (Laboratorio)\Proyecto1\Frontend> npm run dev

> package@0.0.0 dev
> vite

VITE v6.3.5 ready in 2981 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ Vue DevTools: Open http://localhost:5173/__devtools__/ as a separate window
→ Vue DevTools: Press Alt(⌘)+Shift(⇧)+D in App to toggle the Vue DevTools
→ press h + enter to show help
```



Node js

Node.js es un entorno de ejecución para JavaScript construido sobre el motor V8 de Chrome, que permite ejecutar código JavaScript en el servidor. A diferencia de los navegadores, que ejecutan JavaScript en el cliente, Node.js permite desarrollar aplicaciones del lado del servidor utilizando un solo lenguaje: JavaScript. Esto lo hace ideal para crear APIs, servidores web, herramientas de línea de comandos y aplicaciones en tiempo real como chats o sistemas de notificaciones.

Node.js se destaca por su modelo de E/S (entrada/salida) no bloqueante y orientado a eventos, lo que le permite manejar múltiples conexiones de manera eficiente sin necesidad de múltiples hilos. Gracias a su ecosistema de paquetes gestionado por npm (Node Package Manager), ofrece una gran cantidad de bibliotecas y módulos reutilizables que aceleran el desarrollo de aplicaciones modernas y escalables.

Axios

Es una biblioteca basada en promesas que permite realizar solicitudes HTTP desde Node.js o el navegador. Es utilizada para enviar y recibir datos de servidores remotos, facilitando la integración con APIs RESTful. Soporta los métodos comunes como GET, POST, PUT y DELETE, y permite manejar encabezados, cuerpos de solicitud y respuestas de forma estructurada.

Esta biblioteca destaca por su facilidad de uso, manejo automático de respuestas JSON y capacidad de interceptar solicitudes o respuestas para aplicar lógica personalizada (como autenticación o manejo de errores). Es ideal para construir aplicaciones web que se comunican con servicios externos.

csv-parser

Es un módulo de Node.js que permite analizar archivos CSV de forma rápida y eficiente. Convierte cada fila del archivo en un objeto JavaScript, lo que simplifica el

procesamiento de grandes volúmenes de datos tabulares sin necesidad de cargar todo el archivo en memoria.

Esta herramienta es especialmente útil en entornos donde se trabaja con datos estructurados provenientes de hojas de cálculo u otras fuentes que exportan en formato CSV. Su enfoque basado en streams lo hace altamente eficiente para archivos grandes.

dotenv

Es una utilidad que carga variables de entorno desde un archivo `.env` al objeto `process.env` de Node.js. Esto permite separar la configuración sensible (como credenciales de base de datos o claves de API) del código fuente, facilitando una gestión segura y flexible de entornos de desarrollo y producción.

Su uso promueve buenas prácticas en el desarrollo de aplicaciones, ya que evita la exposición de datos confidenciales en el repositorio y permite cambiar configuraciones sin modificar el código. Es ampliamente adoptado en proyectos que requieren portabilidad y control de configuración.

express

Es un framework web para Node.js que simplifica la creación de servidores HTTP y APIs REST. Proporciona una arquitectura basada en rutas y middlewares que permite organizar el código de forma modular, facilitando el desarrollo, mantenimiento y escalabilidad de aplicaciones web.

Al ser minimalista, `express` permite al desarrollador agregar solo las funcionalidades necesarias mediante paquetes adicionales, manteniendo el control total sobre el flujo de datos y la lógica del servidor. Es ampliamente utilizado por su rapidez, flexibilidad y gran comunidad de soporte.

mongodb

Es el driver oficial para conectar aplicaciones Node.js con bases de datos MongoDB. Proporciona una API moderna y optimizada para ejecutar operaciones CRUD (crear, leer, actualizar y eliminar) sobre documentos y colecciones, utilizando una estructura orientada a objetos.

Este paquete facilita la interacción con MongoDB directamente desde el backend, permitiendo la gestión eficiente de datos en aplicaciones NoSQL. Soporta operaciones avanzadas como agregaciones, índices, transacciones y conexiones a clústeres, lo cual lo convierte en una herramienta robusta para trabajar con bases de datos escalables.

NOTAS:

para desplegar el proyecto debes clonar el repositorio de git compartido [Repositorio](#), puedes abrir una terminal en la carpeta llamada Backend, seguido puedes desplegar el proyecto con **npm start**, no sin antes instalar las dependencias con **npm install**, si se desea subir a un servidor es importante que se ejecuten estos dos comandos.