

Universidad San Carlos de Guatemala

Sistema de Base de Datos 2

Auxiliar: Saul Jafet Menchu Recinos

Guatemala, 20 de Junio de 2025



PROYECTO NO.2
(MANUAL TÉCNICO)
Grupo #8

Javier Isaías Coyoy Marín 3152-38488-0901

Eiler Rigoberto Gómez Figueroa 3249-40033-1332

MANUAL TÉCNICO

FRONTEND

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.

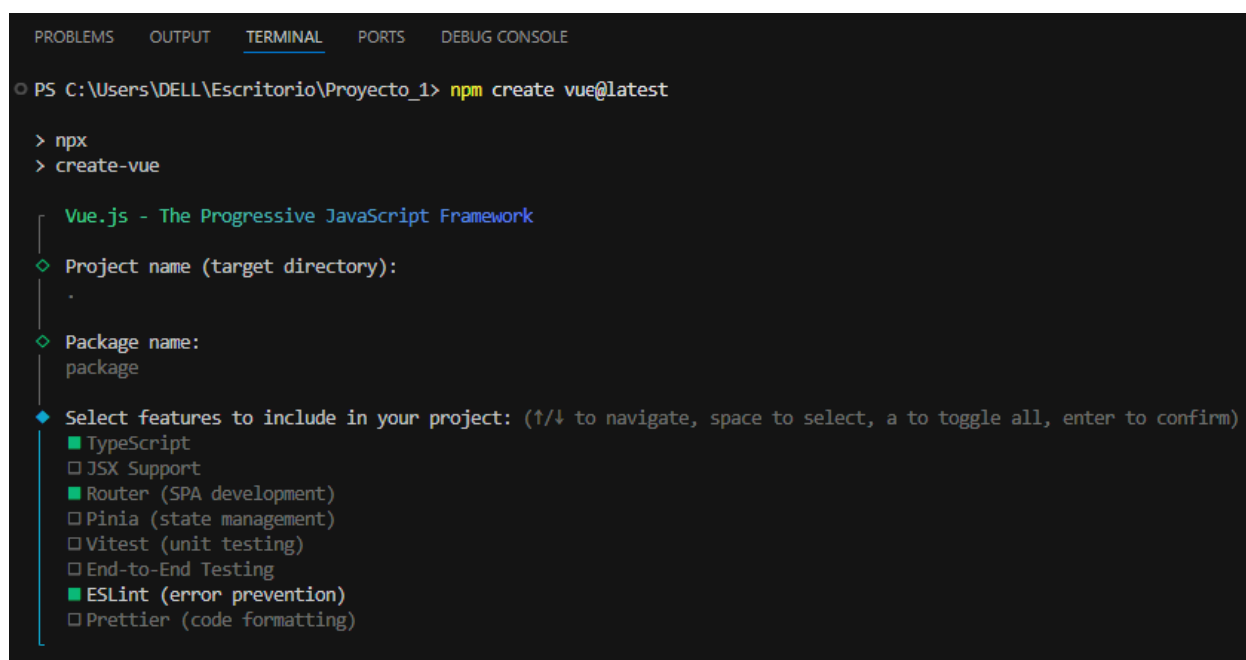
Vuetify

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.

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

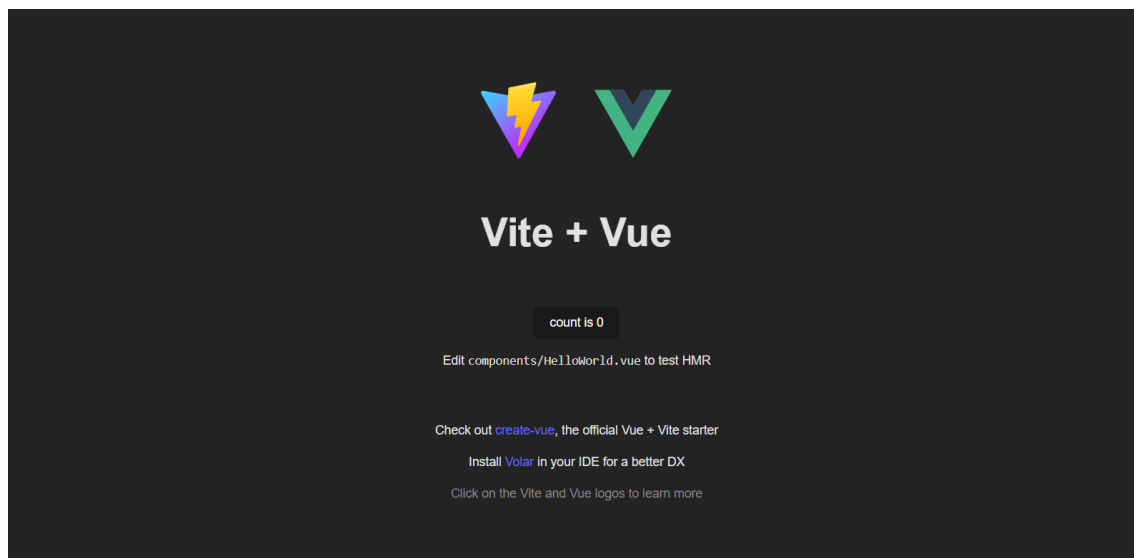
previa en tiempo real en el navegador, lo que facilita la detección y corrección de errores durante la construcción de la aplicación.

```
PROBLEMS 1 OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS C:\Users\DELL\Escritorio\Base de Datos 2 (Laboratorio)\Proyecto2\Frontend> npm run dev

> package@0.0.0 dev
> vite

VITE v6.3.5 ready in 2912 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
```



BACKEND

Para el proyecto backend se usaron las librerías que se describen a continuación, las cuales son instaladas al inicio cuando se realiza **npm install** si el proyecto se correrá localmente, luego use **npm start** para desplegar el proyecto en el puerto 3000 de la máquina local.

1. dotenv

Módulo que gestiona variables de entorno, cargando configuraciones desde un archivo .env al proceso de Node.js. Permite separar datos sensibles como claves y conexiones del código fuente, mejorando la seguridad y facilitando el despliegue en diferentes entornos. Es esencial para manejar configuraciones específicas de desarrollo, prueba y producción sin modificar el código.

2. express

Framework minimalista para construir aplicaciones web y APIs en Node.js. Proporciona un conjunto robusto de características para manejar rutas, middlewares y peticiones HTTP. Simplifica la creación de servidores y es la base para desarrollar aplicaciones backend escalables y eficientes. Ideal para implementar servicios RESTful.

3. express-validator

Biblioteca de validación y sanitización de datos de entrada en aplicaciones Express. Ofrece funciones para verificar campos, prevenir inyecciones de código y asegurar la integridad de los datos recibidos. Se integra directamente con Express para validar peticiones de forma declarativa.

4. prom-client

Cliente para Prometheus que permite recopilar y exponer métricas de aplicaciones Node.js. Crea contadores, gauges y histogramas para monitorear rendimiento y comportamiento del sistema. Las métricas se exponen en un endpoint /metrics para ser consumidas por Prometheus.

5. redis

Cliente oficial de Redis para Node.js, que permite interactuar con esta base de datos clave-valor. Se usa para cacheo, almacenamiento de sesiones, colas de mensajes y datos temporales. Ofrece alta performance y soporte para operaciones atómicas y pub/sub.

6. uuid

Generador de identificadores únicos universales (UUIDs) versión 4. Produce strings no secuenciales de 36 caracteres para usarse como IDs únicos en bases de datos o sistemas distribuidos. Elimina colisiones y no requiere coordinación centralizada.

7. cors

Middleware de Express que habilita el intercambio de recursos entre orígenes (CORS). Configura los headers HTTP necesarios para permitir peticiones seguras entre dominios diferentes. Previene errores de bloqueo en navegadores para APIs consumidas desde frontends externos.

NOTA: Este proyecto está planeado para dar datos en Prometheus y Grafana por lo que es dockerizado; en la raíz del proyecto busque el archivo llamado [Readme.md](#) el cual contiene instrucciones para desplegar el proyecto en docker.

dentro del proyecto abra una terminal y ejecute el comando: `docker-compose up --build -d`

por lo que desplegará los servicios en docker los cuales son:

```
**API Node.js:** http://localhost:3000
Punto de entrada para tus endpoints REST.

- **Redis:** puerto 6380
Base de datos en memoria usada por la API.

- **Prometheus UI:** http://localhost:9090
```



```
Plataforma para monitoreo y recolección de métricas.  
  
- **Grafana UI:** http://localhost:3001  
Plataforma para visualización de métricas.  
**Usuario:** admin  
**Contraseña:** admin123
```

El puerto de Redis ha sido cambiado a 6380 debido a que si se tiene redis instalado localmente esto generará problemas porque el nativo utiliza el puerto por defecto es decir el 6379.



redis

