Universidad del Valle de Guatemala

Ingeniería de Software 1

Prof. Erick Marroquín

Tarea Investigativa 2

Juan Diego Solís Martínez - 23720

Nils Muralles Morales - 23727

Víctor Manuel Pérez Chávez - 23731

Diego Oswaldo Flores Rivas - 23714

Isabella Recinos Rodríguez- 23003

Guatemala, 12 de marzo de 2025

Resumen

Vue.js y Ruby on Rails son frameworks que optimizan el desarrollo de aplicaciones web mediante estructuras organizadas y patrones de diseño eficientes. Vue.js, enfocado en el frontend, permite la creación de interfaces reactivas y modulares gracias a su arquitectura basada en componentes y su sistema de reactividad. Implementa principios como el enlace de datos, la sintaxis declarativa y el DOM virtual, facilitando la creación de aplicaciones interactivas.

Por otro lado, Ruby on Rails es un framework backend que sigue el patrón MVC, simplificando el desarrollo mediante convenciones y automatización de tareas. Su uso de Active Record para la gestión de bases de datos y su enfoque en RESTful permiten la creación de aplicaciones escalables y estructuradas.

Ambos frameworks incorporan patrones de diseño como el Observador, el Proxy y la composición en Vue.js, mientras que Rails emplea Active Record, Singleton y Service Objects para la organización del código. Aunque Vue.js y Rails tienen enfoques distintos, pueden integrarse para desarrollar aplicaciones web completas, combinando una interfaz dinámica con una gestión eficiente del servidor y la base de datos.

Introducción

En el desarrollo de aplicaciones web, los frameworks juegan un papel crucial al proporcionar herramientas y estructuras que optimizan el proceso de construcción de software. Dos de los frameworks más utilizados en la actualidad son Vue.js y Ruby on Rails, cada uno con enfoques y características distintas que los hacen adecuados para diferentes tipos de proyectos. Vue.js es un framework progresivo de JavaScript centrado en la creación de interfaces de usuario dinámicas y reactivas. Su arquitectura basada en componentes y su sistema de reactividad permiten desarrollar aplicaciones web interactivas con un código modular y reutilizable. Al ser un framework orientado al frontend, Vue.js facilita la manipulación del DOM y la integración con otras tecnologías para el desarrollo de aplicaciones modernas. Por otro lado, Ruby on Rails es un framework backend escrito en Ruby, diseñado para el desarrollo ágil de aplicaciones web mediante el patrón Modelo-Vista-Controlador (MVC). Su filosofía de "Convención sobre Configuración" (CoC) y "No te repitas" (DRY) optimiza la escritura de código limpio y eficiente, permitiendo el desarrollo rápido de aplicaciones escalables y mantenibles. Este trabajo analiza en detalle ambos frameworks, explorando sus principios de funcionamiento, patrones de diseño y arquitectónicos, casos de uso recomendados y sus diferencias con otras tecnologías similares.

Frameworks

VueJS

1. Descripción:

   Vue.js es un framework diseñado para construir interfaces de usuario de manera eficiente y dinámica. Se centra en mejorar la experiencia del desarrollo frontend al facilitar la creación de aplicaciones web interactivas (López, 2019).

   El objetivo de Vue.js es simplificar el desarrollo de aplicaciones web, proporcionando herramientas que permiten estructurar la interfaz de usuario de forma declarativa y reactiva. Se utiliza principalmente en el desarrollo frontend, permitiendo la creación de componentes reutilizables y la gestión eficiente del estado de la aplicación (Barragán, 2021).

2. Principios de funcionamiento y componentes:

   Entre los distintos principios de funcionamiento de VueJS se encuentran los siguientes:

   1. Reactividad: En un DOM (Document Object Model) virtual, Vue reacciona a los cambios que el usuario realiza en la interfaz de manera inmediata y se actualiza automáticamente.
   2. Basado en componentes: Una interfaz gráfica desarrollada utilizando Vue utiliza componentes reutilizables comúnmente declarados por los usuarios que cuentan con un estado (state) y tienen un ciclo de vida propio.
   3. Enlace de datos: Existen enlaces que pueden realizarse de dos maneras: Con v-bind, se pueden atar valores HTML con datos del componente de vue; y con v-model se puede sincronizar el estado de los componentes con la interacción que realice el usuario.
   4. Sintaxis declarativa: Utiliza sintaxis declarativa para generar HTML, la cual incluye directivas como v-if, v-for y v-show, las cuales modifican el DOM de manera sencilla y eficaz.

   Con respecto a los componentes, cada archivo de Vue es un bloque reutilizable que encapsula HTML, CSS y JavaScript, los cuales pueden comunicarse con otros componentes y escuchar eventos. Además de que cuenta con componentes que manejan el enrutamiento por medio de una SPA (Single Page Application) (Vue, 2024).

3. Tipo de framework y proceso que cubre:

   Vue.js es un framework progresivo de JavaScript. Esto significa que se puede adoptar de forma gradual, desde mejorar partes específicas de una aplicación hasta usarlo como el núcleo del desarrollo completo (VueJs, 2024).

   Vue.js se especializa en el desarrollo de:

   a. Front-end: Está enfocado en la construcción de interfaces de usuario y la manipulación del DOM de forma eficiente.
   b. Reactividad y gestión de estados: Ofrece un sistema de datos reactivos que actualiza automáticamente la UI cuando cambia el estado de la aplicación.
   c. Routing y navegación: Permite gestionar la navegación de aplicación de una sola pagína con Vue Router.
   d. Integración de APIs: Facilita la comunicación con APIs Rest o GraphQL a través de Axios y otras bibliotecas.
   e. SSR y Mobile: Soporta el renderizado del lado del servidor SSR con Nuxt.js y con Quasar o Ionic Vue se pueden crear aplicaciones móviles y de escritorio.

   (VueJs, 2024 & Barragán, 2021).

   No gestiona de forma directa bases de datos o lógica de servidores, por lo que dependiendo del proyecto se puede combinar con otras tecnologías:

   a. Para Backend: Puede combinarse con Node.js (Express.js), Django, Laravel u otras tecnologías que manejen la lógica de servidor y la base de datos.
   b. Para Bases de Datos: Se usa con MySQL, PostgreSQL, Firebase, MongoDB, entre otras.
   c. Para estilos: Puede usarse con CSS puro, Tailwind CSS, Bootstrap o Vuetify.

   (López, 2019 & García, 2024).

4. Patrones de diseño y arquitectónicos que implementa:

   Patrones de diseño:

   a) Patrón observador: El principio de reactividad ante los cambios realizados por el usuario en Vue es una implementación del patrón observador, ya que la UI se actualiza automáticamente por medio de componentes como ref y reactive, y eventos escuchados por el usuario.
   b) Patŕon de composición: Patŕon de modularización y organización de código implementado a partir de Vue 3, el cual permite separar el código en funciones que funcionan como componentes reutilizables.
   c) Patrón proxy: En sus versiones más recientes, Vue también se beneficia del patrón de diseño proxy para modificar el comportamiento de objetos y

hacerlos reactivos como con reactive, el cual envuelve el objeto y lo hace reaccionar ante los cambios del usuario.

(Movaffagh, 2022; Vue, 2024; Mozilla Foundation, s.f.).

Patrones arquitectónicos:

a) Arquitectura basada en componentes: En Vue, cada elemento es un componente con su propia lógica, estilo y pantalla, la cual permite que una aplicación desarrollada en este framework sea modular y escalable.
b) Arquitectura MVC/MVVM: Vue cuenta con modelos reactivos definidos por medio de declarativas específicas como data, ref y reactive; vistas hechas con template y viewmodel con componentes como computed y methods.
c) Arquitectura SPA: SPA, o single page application por sus siglas en inglés, es utilizado por medio del sistema de enrutadores de vue, que permite hacer lazyloading y cargar las vistas en una sola página.

(Vue, 2024).

5. Situaciones recomendadas para su uso:
    a. Aplicaciones de una sola página: Facilita la creación de aplicaciones dinámicas sin necesidad de recargar la página.
    b. Dashboards e interfaces interactivas: Actualiza los datos en tiempo real y gestiona los componentes reutilizables.
    c. Sistemas de gestión de contenido: Crea interfaces que se actualizan sin necesidad de recargar la página.
    d. Integración en proyectos existentes: Se puede integrar fácilmente en sistemas ya desarrollados, añadiendo nuevas funcionalidades sin rehacer toda la aplicación.
    e. Aplicaciones móviles y de escritorio: Se integra con frameworks como Quasar o Ionic Vue para crear aplicaciones móviles y de escritorio con un único código base.

(Vue, 2024 & Barragán, 2021).

6. Semejanzas y diferencias con otros frameworks
    - Vue, React y Svelte tienen arquitecturas basadas en componentes.
    - Vue y React utilizan un DOM virtual, mientras que Angular utiliza un DOM incremental y Svelte compila el código sin necesidad de DOM virtual.
    - Manejo de componentes de manera reactiva.
    - En comparación con Angular, React y Svelte, ambos permiten tanto SPA como SSR.

(Vue, 2024).

Ruby on Rails

1. Descripción

   Es un framework del lado del servidor para desarrollo de aplicaciones web, escrito en Ruby. Sigue el paradigma de arquitectura Modelo-Vista-Controlador (MVC), lo que facilita la organización y separación del código en componentes bien definidos. Además, Rails sigue otros paradigmas, como el de convenciones sobre configuraciones (CoC), lo que significa que ofrece patrones predefinidos para resolver problemas comunes, reduciendo la necesidad de escribir código repetitivo. Esta filosofía, junto con la amplia biblioteca de gemas (paquetes reutilizables), permite a los desarrolladores construir aplicaciones web robustas y escalables de manera rápida y eficiente (Ruby on Rails, 2025).

2. Principios de funcionamiento y componentes
   a. Convention over Configuration (CoC): Rails incluye una serie de convenciones por defecto, evitando que estas tengan que ser definidas por el desarrollador en configuraciones extensas. Esto significa que el framework tiene una opinión acerca de la mejor manera de hacer las cosas.

   b. Don't repeat yourselff (DRY): Para que el código sea mantenible, extensible y con menos bugs, se debe evitar la repetición a toda costa.

   c. Model-View-Controller (MVC): Organiza la aplicación en tres componentes principales. El Modelo gestiona la lógica de negocio y la interacción con la base de datos. La Vista se encarga de la presentación, mostrando los datos al usuario de manera dinámica. El Controlador actúa como intermediario, recibiendo las solicitudes del usuario, coordinando la lógica con el modelo y enviando los resultados a la vista. Esta separación facilita el desarrollo, la escalabilidad y el mantenimiento de la aplicación.

   (Ruby on Rails, 2025).

3. Tipo de framework y proceso que cubre

   Ruby on Rails es un framework de desarrollo web full-stack basado en el lenguaje Ruby. Sigue el patrón MVC (Modelo-Vista-Controlador), lo que permite organizar el código en capas separadas para mejorar la mantenibilidad y escalabilidad. Cubre todo el proceso de desarrollo web, desde la gestión de la base de datos hasta la presentación en la interfaz de usuario, facilitando el desarrollo ágil con convenciones y herramientas que automatizan tareas repetitivas (Ruby on Rails, 2025).

4. Patrones de diseño y arquitectónicos que implementa
    a. Patrones de diseño
        i. Active Record: Implementa el patrón Active Record, que permite mapear objetos de Ruby a tablas de bases de datos y simplifica las consultas mediante un ORM (Object-Relational Mapping).
        ii. Singleton: Se usa en la configuración de la aplicación y en la gestión de ciertas instancias únicas, como las conexiones a bases de datos.
        iii. Observer: Se usa en Active Record para ejecutar acciones cuando ocurren cambios en un modelo.
        iv. Decorator (Draper Gem): Permite añadir funcionalidad a los modelos sin modificar su código base.
        v. Service Objects: Se usa para encapsular lógica de negocio compleja en clases específicas en lugar de sobrecargar modelos o controladores.

    (Patrones de Diseño En Ruby, 2025).

    b. Patrones arquitectónicos
        i. Modelo-Vista-Controlador (MVC): Organiza la aplicación en tres capas separadas.
        ii. RESTful: Ruby on Rails fomenta el uso de arquitectura REST para estructurar las rutas y controladores, facilitando el desarrollo de APIs.
        iii. Inversión de Control (IoC) y Convención sobre Configuración: Rails reduce la necesidad de configuración explícita y usa convenciones predefinidas para simplificar el desarrollo.
        iv. Microservicios (posible integración): Aunque Rails es tradicionalmente un framework monolítico, puede integrarse en arquitecturas de microservicios con APIs REST y GraphQL.

    (Arquitectura | Guías de Make It Real, 2022).

5. Situaciones recomendadas para su uso
    a. APIs RESTful: Es útil para la construcción de APIs que permitan la comunicación entre diferentes aplicaciones.
    b. Prototipos rápidos: Es sumamente eficiente para la creación de prototipos funcionales en corto tiempo, esto gracias a su enfoque en rapidez y simplicidad.
    c. Aplicaciones web: Debido a su simplicidad es muy útil para realizar aplicaciones web para empresas emergentes.

(Cortés, 2024; Kodigo, s.f.).

6. Semejanzas y diferencias con otros frameworks
    a. Arquitectura Similar: Frameworks como Django y Laravel siguen el patrón MVC, separando la lógica de negocio, la interfaz de usuario y el control de flujo de forma similar a como lo hace Ruby on Rails.
    b. Lenguaje de programación: Ruby on Rails utiliza Ruby, conocido por su sintaxis elegante y legibilidad, mientras que Django utiliza Python y Laravel utiliza PHP.
    c. Ecosistema y bibliotecas: Cada framework tiene su propio conjunto de bibliotecas y herramientas. Por ejemplo, Ruby on Rails utiliza "gems" para extender funcionalidades, mientras que Django utiliza "packages" y Laravel utiliza "packages" o "composer packages".

(MoldStud, 2024; Morales, 2024; Techcronus, 2025).

Conclusiones

- Vue.js y Ruby on Rails están diseñados para optimizar el desarrollo de aplicaciones, ofreciendo estructuras organizadas y componentes reutilizables que mejoran la eficiencia.
- Mientras Vue.js se enfoca en la construcción de interfaces dinámicas y reactivas, Ruby on Rails cubre todo el proceso de desarrollo backend con una arquitectura MVC bien definida.
- Ambos frameworks aplican patrones de diseño y arquitecturas que favorecen la escalabilidad y el mantenimiento, permitiendo desarrollar aplicaciones robustas y eficientes.

Bibliografía

Barragán, A. (2021). Qué es Vue JS y qué lo diferencia de otros frameworks. OpenWebinars. https://openwebinars.net/blog/que-es-vue-js-y-que-lo-diferencia-de-otros-frameworks/

Cortés, D. (2024). ¿Qué es Ruby on Rails que puedo hacer? Blender Deluxe. https://blenderdeluxe.com/es/desarrollo-web/que-es-ruby-on-rails-que-puedo-hacer-271

García, F. (2024). Qué es Vue: por qué usarlo como framework JS de referencia. Arsys. https://www.arsys.es/blog/vuejs

Kodigo. (s.f.). Ruby on Rails: ¿qué es y para qué sirve? Kodigo. https://kodigo.org/hablemos-de-ruby-on-rails-ruby-on-rails-que-es-y-para-que-sirve

López, M. (2019). ¿Qué es Vue.js y cómo lo usamos? Encora. https://insights.encora.com/es/blog/qu%C3%A9-es-vue.js-y-c%C3%B3mo-lo-usamos

MoldStud. (2024). Ruby on Rails vs. Other Web Frameworks: Comparing Strengths and Weaknesses. Ruby on Rails vs. Other Web Frameworks: Comparing Strengths and Weaknesses | MoldStud

Morales, A. (2024). Laravel v.s. Ruby on Rails v.s. Django. Dev. Laravel v.s. Ruby on Rails v.s. Django - DEV Community

Movaffagh, E. (2022, February 14). Reactivity system in Vue.js. Medium. Retrieved March 11, 2025, from https://ehsan-movaffagh.medium.com/reactivity-system-in-vue-js-8c8093436eb1Movaffagh, E. (2022, February 14). Reactivity system in Vue.js. Medium. Retrieved March 11, 2025, from https://ehsan-movaffagh.medium.com/reactivity-system-in-vue-js-8c8093436eb1

Mozilla Foundation. (s.f.). Proxy - JavaScript | MDN. MDN Web Docs. Retrieved March 11, 2025, from https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Proxy

Ruby on Rails. (2025). Getting Started with Rails. https://guides.rubyonrails.org/getting_started.html

Techcronus. (2025). Ruby on Rails vs. other frameworks – A comparative Analysis. Choosing the Right Framework: Ruby on Rails vs. Other Leading Options

VueJs. (2024). ¿Qué es Vue.js? https://es.vuejs.org/v2/guide/

Anexos

Repositorio de GitHub:

https://github.com/Isabella334/Proyecto_DAPA.git

Enlace al Google Docs:

https://docs.google.com/document/d/1jKRI8ubJfgZbG3JWf75wXf4kOsbU0afuFe-FUFKk8M4/edit?usp=sharing

Gestión del tiempo:

Nro.	Tarea	Encargado	Fecha inicio	Fecha entrega
1	Redactar el resumen	Juan Solís	11/03/2025	11/03/2025
2	Redactar la introducción	Isabella Recinos	11/03/2025	11/03/2025
3	Nombre y descripción general de VueJS	Juan Solís	11/03/2025	11/03/2025
4	Principios de funcionamiento y componentes de VueJS	Nils Muralles	11/03/2025	11/03/2025
5	Tipo de framework y proceso que cubre VueJS	Juan Solís	11/03/2025	11/03/2025
6	Patrones de diseño y arquitectónicos que implementa VueJS	Nils Muralles	11/03/2025	11/03/2025
7	Situaciones recomendadas para el uso de VueJS	Juan Solís	11/03/2025	11/03/2025
8	Semejanzas y diferencias con otros frameworks	Nils muralles	11/03/2025	11/03/2025
9	Nombre y descripción general de Ruby on Rails	Victor Pérez	11/03/2025	11/03/2025
10	Principios de funcionamiento y componentes de Ruby on Rails	Victor Pérez	11/03/2025	11/03/2025
11	Tipo de framework y proceso que cubre Ruby on	Isabella Recinos	11/03/2025	11/03/2025

	Rails			
12	Patrones de diseño y arquitectónicos que implementa Ruby on Rails	Isabella Recinos	11/03/2025	11/03/2025
13	Situaciones recomendadas para el uso de Ruby on Rails	Diego Flores	11/03/2025	11/03/2025
14	Semejanzas y diferencias con otros frameworks	Diego Flores	11/03/2025	11/03/2025
15	Redactar las conclusiones	Todos	11/03/2025	11/03/2025
16	Elaborar la presentación del trabajo de investigación	Todos	11/03/2025	11/03/2025