**UNIVERSIDAD DEL VALLE DE GUATEMALA**
Ingeniería de Software 1
Sección 20

# Tarea investigativa 2: Frameworks

Facultad de Ingeniería

Iris Ayala, 23965
Gabriel Bran, 23590
David Domínguez, 23712
Luis Padilla, 23663
Anggie Quezada, 23643

Guatemala

2025

## Resumen

Los frameworks facilitan el desarrollo de software al proporcionar estructuras optimizadas. Este informe analiza Svelte, un framework frontend que mejora el rendimiento al compilar código en JavaScript optimizado, y Laravel, un framework backend en PHP que sigue el patrón Modelo-Vista-Controlador (MVC) y simplifica la gestión de bases de datos, autenticación y seguridad.

Svelte destaca por su reactividad sin necesidad de estados centralizados, su encapsulamiento de estilos y una sintaxis que integra HTML, CSS y JavaScript en un solo archivo. Laravel, por su parte, sobresale por herramientas como Eloquent ORM para bases de datos, Blade para plantillas y Artisan CLI para automatización.

Ambos frameworks mejoran la productividad y el rendimiento, pero se diferencian en su aplicación: Svelte es ideal para interfaces interactivas ligeras y rápidas, mientras que Laravel es adecuado para sistemas robustos con una arquitectura escalable y segura.


## Introducción

Los frameworks son herramientas esenciales en el desarrollo de software, ya que proporcionan una estructura optimizada para agilizar la creación de aplicaciones. Su uso permite mejorar la eficiencia, escalabilidad y mantenibilidad del código, facilitando la implementación de soluciones complejas con menor esfuerzo.

En este documento se presentan dos frameworks ampliamente utilizados en el desarrollo web: Svelte, para la creación de interfaces interactivas en el frontend, y Laravel, para la gestión del backend. Se describen sus principales características, principios de funcionamiento y patrones de diseño, destacando su impacto en la optimización del proceso de desarrollo.

## Fronted: Framework Svelte

### Descripción

Svelte es un framework front-end para la construcción de interfaces interactivas. Fue creado por Rich Harris y su primera versión se lanzó en 2016. A diferencia de otros frameworks como React o Vue, que dependen de un proceso de actualización del DOM, Svelte adopta un enfoque distinto: compila el código en JavaScript altamente optimizado en el momento de la construcción, eliminando el tiempo de ejecución para interpretar y renderizar componentes (Chaewonkong, 2023).

El propósito de Svelte es simplificar el desarrollo frontend proporcionando un sistema más eficiente y fácil de usar para la creación de aplicaciones web reactivas. Se enfoca en reducir la sobrecarga de procesamiento en el navegador, lo que resulta en aplicaciones más rápidas y ligeras (Chaewonkong, 2023).

**Principios de funcionamiento y componentes**

- **Compilación en tiempo de construcción**: En lugar de enviar un framework completo al navegador, Svelte traduce los componentes en código JavaScript optimizado antes de que la aplicación se ejecute.
- **Reactividad sin necesidad de estados centralizados**: En otros frameworks, la reactividad suele manejarse con funciones específicas o estados globales. En Svelte, simplemente actualizar una variable en el código automáticamente actualiza la interfaz de usuario.
- **Componentes**: un archivo .svelte que combina:
    - HTML para la estructura.
    - CSS para los estilos.
    - JavaScript para la lógica.

**Tipo de framework es y proceso del desarrollo que cubre:**

Svelte es un framework frontend que se enfoca en la construcción de componentes y la gestión de reactividad en el frontend.

Svelte cubre principalmente:

- Creación de componentes dinámicos e interactivos.
- Gestión de estado local y reactividad sin herramientas adicionales.
- Optimización del rendimiento gracias a su modelo de compilación.

Sin embargo, para aplicaciones más grandes, puede ser necesario combinar Svelte con otras tecnologías:

- Backend: Se puede integrar con cualquier backend basado en REST o GraphQL.
- Gestión de estado global: Para proyectos más complejos, se pueden usar herramientas como Svelte Stores o soluciones externas como Redux.
- Enrutamiento: Svelte no tiene un sistema de enrutamiento incorporado, pero se puede usar svelte-routing o frameworks como SvelteKit, que amplía las capacidades de Svelte para aplicaciones completas.

**Qué patrones de diseño y arquitectónicos implementa el framework seleccionado y en que parte de este son implementados:**

Svelte incorpora varios patrones de diseño y arquitectónicos en su estructura:

| Patrón | Descripción | Beneficios |
|---|---|---|
| Arquitectura Basada en Componentes | Archivos .svelte con HTML, JavaScript y CSS integrados. Componentes autónomos y reutilizables. | Desarrollo modular, mantenimiento simplificado |
| Reactividad Declarativa | Actualización de UI basada en asignación de variables. Sin necesidad de métodos como setState | Código más limpio. Actualizaciones automáticas |
| Encapsulamiento de Estilos | CSS definido dentro del componente. Estilos aislados por componente | Evita conflictos de CSS. No requiere BEM o CSS-in-JS |
| Compilación en Tiempo de Construcción | Transpilación a JavaScript optimizado. Elimina el DOM virtual | Menor sobrecarga en ejecución. Rendimiento mejorado. |

(Design Patterns for Building Reusable Svelte Components | Render Blog, 2022)

Dónde se puede implementar estos patrones dentro del framework:

| Patrón | Implementación en Svelte |
|---|---|
| Componentes | Definidos en archivos .svelte, combinando HTML, CSS y JavaScript en una unidad coherente. |
| Reactividad | Se logra mediante la asignación de valores a variables, lo que actualiza la UI de manera automática. |
| Encapsulamiento de Estilos | Los estilos CSS escritos dentro de un componente solo afectan a ese componente, evitando conflictos globales. |
| Compilación | Durante la construcción, Svelte genera JavaScript optimizado que manipula directamente el DOM. |

(Introducción a Svelte - Aprende Desarrollo Web | MDN, 2024)

**Situaciones donde se recomienda su uso:**

Svelte es particularmente útil en los siguientes escenarios:

- Aplicaciones que requieren alto rendimiento: Su enfoque sin DOM virtual reduce la carga en el navegador y mejora la velocidad de carga.
- Proyectos que buscan simplicidad y menor complejidad: la sintaxis clara y la estructura basada en componentes facilitan el desarrollo.

- Aplicaciones donde el tamaño del bundle es un factor crítico: La eliminación de dependencias innecesarias permite construir aplicaciones más ligeras.
- Equipos que prefieren una integración estrecha entre HTML, CSS y JavaScript: La combinación de estos lenguajes en un solo archivo mejora la coherencia del código y su mantenibilidad.

**Ejemplo de implementación:**

```
<script>
  let count = 0;

  function increment() {
    count += 1;
  }

  function decrement() {
    count -= 1;
  }
</script>

<main>
  <h1>Counter: {count}</h1>
  <button on:click={increment}>Increment</button>
  <button on:click={decrement}>Decrement</button>
</main>

<style>
  main {
    text-align: center;
    padding: 2em;
  }

  h1 {
    font-size: 2em;
  }

  button {
    font-size: 1em;
    margin: 0.5em;
    padding: 0.5em 1em;
  }
</style>
```

(Chaewonkong, 2023)

## Backend: Framework Laravel

**Descripción**

Laravel es un framework de desarrollo web basado en PHP que se distingue por su sintaxis elegante y expresiva. Su propósito principal es simplificar el proceso de creación de aplicaciones web proporcionando una estructura robusta y un conjunto de

herramientas que permiten a los desarrolladores enfocarse en la lógica de negocio sin preocuparse por la implementación de tareas comunes, como la gestión de bases de datos, autenticación de usuarios, enrutamiento y manejo de sesiones. (Laravel, 2023)

Este framework facilita el desarrollo ágil al ofrecer una arquitectura modular, herramientas avanzadas para pruebas, y una integración sencilla con bases de datos y servicios externos. Además, Laravel promueve las mejores prácticas del desarrollo web, como el uso del patrón MVC (Modelo-Vista-Controlador) y la inyección de dependencias, asegurando así aplicaciones más mantenibles y escalables. (Laravel, 2023)

Areas que resuelve:

- Desarrollo de aplicaciones web y APIs RESTful: Laravel facilita la construcción de aplicaciones web dinámicas y servicios API mediante su sistema de enrutamiento flexible, middleware y soporte para JSON.
- Gestión de bases de datos: A través de su ORM Eloquent, Laravel permite interactuar con bases de datos de manera intuitiva, además de ofrecer migraciones y seeders para una mejor gestión de datos.
- Autenticación y seguridad: Laravel incluye un sistema de autenticación y autorización listo para usar, con protección contra vulnerabilidades como inyecciones SQL y ataques CSRF.
- Enrutamiento y renderizado de vistas: Su sistema de rutas simplifica la definición de endpoints y permite manejar diferentes respuestas para controladores y vistas con facilidad.

**Principios de funcionamiento y componentes**

Principios

- **Convención sobre configuración**: Laravel minimiza la necesidad de configuraciones manuales al establecer convenciones predeterminadas para aspectos comunes del desarrollo. Esto permite reducir el tiempo de configuración y facilita la integración de nuevas funcionalidades.
- **Arquitectura Modelo-Vista-Controlador (MVC)**: Laravel adopta el patrón de diseño **MVC**, que separa la lógica de negocio (**Model**), la interfaz de usuario (**View**) y el control de flujo de la aplicación (**Controller**). Esta organización mejora el modularidad, la reutilización del código y la escalabilidad de las aplicaciones.
- **Eloquent ORM**: Laravel incluye un poderoso **Object-Relational Mapping (ORM)** llamado **Eloquent**, que permite a los desarrolladores interactuar con bases de datos utilizando modelos y métodos en lugar de escribir consultas SQL

manualmente. Esto hace que la gestión de datos sea más intuitiva y segura. (Laravel, 2023)

Componentes

- **Artisan**: Es la interfaz de línea de comandos (CLI) de Laravel, que permite ejecutar comandos para automatizar tareas como la generación de código, la creación de migraciones y la ejecución de pruebas.
- **Blade**: Motor de plantillas ligero y flexible que facilita la creación de vistas dinámicas, permitiendo reutilizar componentes y escribir código PHP de manera más organizada dentro de las plantillas.
- **Eloquent**: ORM integrado en Laravel que simplifica la gestión de bases de datos, proporcionando una sintaxis elegante para consultas, relaciones y migraciones.
- **Middleware**: Capa intermedia que permite filtrar y modificar solicitudes HTTP antes de que lleguen a los controladores. Se usa para funciones como autenticación, validación de permisos y protección contra ataques CSRF.
- **Routing**: Sistema de enrutamiento flexible que permite definir las rutas de la aplicación y asociarlas a controladores o acciones específicas, facilitando la organización del flujo de la aplicación.
- **Laravel Mix**: Herramienta de compilación que permite gestionar y optimizar archivos estáticos como **CSS**, **JavaScript** y **SASS**, simplificando la integración con frameworks front-end como Vue.js o React. (Laravel, 2023)

**Tipo de framework es y proceso del desarrollo que cubre**

Laravel es un framework backend basado en PHP que cubre múltiples aspectos del desarrollo web, desde la gestión de rutas y controladores hasta la interacción con bases de datos, APIs y la seguridad (Laravel, 2023).

Su arquitectura modular y enfoque en la convención sobre configuración permiten a los desarrolladores reducir el tiempo de implementación y enfocarse en la lógica de negocio, facilitando la construcción de aplicaciones escalables y mantenibles (Laravel, 2023).

Su proceso de desarrollo se centra en la rapidez y eficiencia, permitiendo definir rutas y controladores con facilidad, administrar datos mediante Eloquent ORM, generar vistas dinámicas con Blade y garantizar la seguridad con Middleware y autenticación integrada. Además, facilita la automatización de tareas comunes a través de Artisan CLI, optimizando el flujo de trabajo del desarrollador (Laravel, 2023).

**Qué patrones de diseño y arquitectónicos implementa el framework seleccionado y en que parte de este son implementados**

Laravel sigue el patrón Modelo-Vista-Controlador (MVC), asegurando una separación clara entre la lógica de negocio, la presentación y el control de flujo de la aplicación. Además, implementa el Active Record Pattern a través de Eloquent ORM, lo que simplifica la manipulación de bases de datos. También utiliza Middleware para gestionar peticiones HTTP y aplicar reglas de seguridad, así como la inyección de Dependencias para mejorar la reutilización y mantenimiento del código (Laravel, 2023).

**Situaciones donde se recomienda su uso:**

- Aplicaciones web con backend robusto:
    - Si necesitas manejar autenticación, bases de datos y lógica de negocio, Laravel ofrece herramientas como Eloquent ORM, autenticación con Sanctum o Passport, y migraciones de base de datos.
- APIs RESTful :
    - Laravel permite construir APIs fácilmente con Laravel Sanctum o Laravel Passport para autenticación.
- Aplicaciones empresariales:
    - Es útil para sistemas de gestión, comercio electrónico, CRM y cualquier app que requiera seguridad, escalabilidad y manejo de usuarios.
- Aplicaciones con plantillas Blade:
    - Laravel usa Blade para renderizar HTML de manera dinámica, útil si no quieres un frontend separado como React o Vue.
- Proyectos con integraciones complejas:
    - Laravel facilita la integración con servicios de terceros como Stripe, AWS, Firebase, y más.

Fastfwd and Fastfwd (2022).

**Ejemplo de implementación:**

```php
class CrearTablaCanciones extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('canciones', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string("nombre");
            $table->string("artista");
            $table->string("album");
            $table->integer("anio");
        });
    }

```

**Semejanzas y diferencias de los frameworks**

Semejanzas:

Desarrolladores enfocados en la productividad: Ambos frameworks buscan simplificar el desarrollo, Laravel en el backend y Svelte en el frontend.

Enfoque en rendimiento: Laravel usa caché y optimización de consultas; Svelte compila su código a JavaScript optimizado sin necesidad de un runtime.

Facilidad de uso: Ambos ofrecen una curva de aprendizaje más amigable que otras alternativas (Laravel vs Symfony en PHP, Svelte vs React/Vue en frontend).

**Diferencias:**

| Característica | Laravel | Svelte |
|---|---|---|
| Implementación | Framework backend (PHP) | Framework frontend (JavaScript) |
| Uso | Manejo de bases de datos, lógica de negocio, APIs | Creación de interfaces interactivas |
| Renderizado | Genera HTML con Blade o APIs JSON | Genera componentes dinámicos compilados en JS |
| Manejo de Estados | Manejado en controladores y modelos | Manejado con stores en el frontend |

| | | |
|---|---|---|
| Escalabilidad | Mejor para backend robusto y arquitecturas monolíticas o API-first | Mejor para SPAs y aplicaciones interactivas |

## Conclusiones

- Svelte es altamente eficiente en el rendimiento, ya que no utiliza un Virtual DOM y compila el código en JavaScript optimizado, reduciendo el peso y mejorando la velocidad de ejecución.
- Laravel proporciona una estructura robusta y organizada para el backend, con características como ORM, autenticación, migraciones y enrutamiento que facilitan el desarrollo de aplicaciones web escalables.
- Svelte es más fácil de aprender y escribir en comparación con otros frameworks frontend, ya que combina HTML, CSS y JavaScript en un solo archivo con una sintaxis clara y mínima configuración.
- Laravel es ideal gracias a su amplio ecosistema de paquetes y herramientas como Eloquent y Blade, que agilizan la gestión de datos y vistas.
- Svelte y Laravel pueden complementarse bien en el desarrollo full stack, donde Svelte maneja la interfaz de usuario de forma dinámica y eficiente, mientras que Laravel gestiona la lógica del servidor y la base de datos de manera estructurada.

## Referencias


Chaewonkong. (2023, 5 julio). The Rise of Svelte: Revolutionizing Front-end Development. Medium. https://medium.com/@chaewonkong/the-rise-of-svelte-revolutionizing-front-end-development-d6a3a469ff9d

Laravel. (2023). Documentación oficial de Laravel. Laravel. https://laravel.com/docs

Design patterns for building reusable svelte components | Render Blog. (10 de marzo 2022). https://render.com/blog/svelte-design-patterns

Introducción a Svelte - Aprende desarrollo web | MDN. (2024). MDN Web Docs. https://developer.mozilla.org/es/docs/Learn_web_development/Core/Frameworks_libraries/Svelte_getting_started

Fastfwd, & Fastfwd. (2022, March 14). *Why choose Laravel for your next web project*. Fastfwd. https://www.fastfwd.com/why-choose-laravel-for-your-next-web-project/

## Informe de gestión del tiempo de la tarea

Link para el historial: Tarea Investigativa 2, Frameworks

Link del excel: Gestion de tiempos.xlsx

Presentación:
https://www.canva.com/design/DAGhh9ruh7c/19dbGB2bodYwcVs18Fy4Pg/edit?utm_content=DAGhh9ruh7c&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

**Planificación de tareas:**

| Integrante | Tarea | Fecha Inicio | Fecha Fin |
|---|---|---|---|
| Bran | Resumen | 12/3/2025 | 13/3/2025 |
| David | Introduccion | 12/3/2025 | 13/3/2025 |
| Iris | Fronted- puntos del 1-3 | 11/3/2025 | 12/3/2025 |
| Anggie | Fronted – puntos del 4-5 | 11/3/2025 | 12/3/2025 |
| Bran | Backend – puntos 1-2 | 11/3/2025 | 12/3/2025 |
| David | Backend – puntos 3-4 | 11/3/2025 | 12/3/2025 |
| Luis | Backend – puntos 5 | 11/3/2025 | 12/3/2025 |
| Luis | Describir semejanzas y diferencias de los frameworks | 11/3/2025 | 12/3/2025 |
| Anggie | Conclusiones | 12/3/2025 | 13/3/2025 |
| Todos | Presentacion | 12/3/2025 | 12/3/2025 |