UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

ANÁLISIS Y DISEÑO 2

VACACIONES DEL SEGUNDO SEMESTRE 2023

GRUPO 2
DOCUMENTO DE DISEÑO

Nombre	Carnet
Mario Cesar Moran Porras	202010793
Gerson Rubén Quiroa del Cid	202000166
Esdras Rodolfo Toc Hi	201807373
Josué Daniel Mínchez Velásquez	202003926

Contenido

Frameworks de PHP

Sobre PHP y los Frameworks

PHP es un lenguaje de programación ampliamente utilizado en el desarrollo web, y hay varios frameworks disponibles que pueden ser adecuados para este proyecto. La elección del framework adecuado puede afectar significativamente el desarrollo, la escalabilidad, la seguridad y la facilidad de mantenimiento del sistema. Debido a que el sistema debe ser capaz de manejar un alto volumen de usuarios y transacciones simultáneas, sin degradar su rendimiento, y también debe ser seguro y capaz de manejar un alto volumen de transacciones, algunos de los frameworks de PHP que podrían ser adecuados son:

- Laravel: Es un framework PHP muy popular que ofrece una estructura sólida, una seguridad avanzada y una gran comunidad de desarrollo. Tiene una sintaxis clara y elegante, facilitando el desarrollo rápido y la implementación de prácticas de seguridad.

- Symfony: Es otro framework bien establecido y maduro. Ofrece una arquitectura flexible que permite la implementación de soluciones complejas. Symfony se centra en la reutilización de código y en la escalabilidad.

Estos dos frameworks también son los más utilizados en la comunidad de PHP, además, debido a la necesidad de garantizar la seguridad de los datos financieros y personales de los clientes, así como la capacidad de manejar un alto volumen de transacciones, podrían ser opciones sólidas. Estos frameworks ofrecen características de seguridad avanzadas, una estructura robusta y una amplia gama de herramientas para el desarrollo y la gestión de aplicaciones web complejas.

A continuación comparamos ambos frameworks para tomar una decisión sobre que se utilizará para el proyecto.

Cuadro Comparativo

Laravel	Symfony
Curva de Aprendizaje: Curva de aprendizaje de grado medio a fuerte.	**Curva de Aprendizaje**: Curva de aprendizaje un poco más severa.
Flexibilidad: Menos flexible en ciertos aspectos, pero proporciona muchas características listas para usar. Limitaciones de personalización. Tareas repetitivas automatizadas y migraciones de bases de datos simplificadas con Artisan	**Flexibilidad**: Altamente flexible y modular, permite la personalización extensa.
Rendimiento: Buen rendimiento, adecuado para aplicaciones de tamaño mediano a grande.	**Rendimiento**: Alto rendimiento, se adapta bien a aplicaciones de gran escala. Estable y robusto para aplicaciones empresariales a gran escala.
Arquitectura: Utiliza el patrón de arquitectura MVC.	**Arquitectura**: Basado en componentes, puede adaptarse a diferentes patrones de diseño (MVC, HMVC, etc.)
Seguridad: Ofrece características de seguridad integradas, pero requiere configuración adicional para algunas medidas específicas.	**Seguridad**: Enfoque en la seguridad, proporciona herramientas avanzadas para la protección de aplicaciones.
Mantenimiento: Facilidad para mantener y actualizar aplicaciones, gracias a su estructura clara.	**Mantenimiento**: Requiere un poco más de esfuerzo para mantener aplicaciones debido a su modularidad.
Escalabilidad: Escalable y permite el desarrollo de aplicaciones con crecimiento futuro.	**Escalabilidad**: Altamente escalable, adecuado para proyectos que necesitan crecer rápidamente.

Para este proyecto creemos que la mejor opción es Laravel, una de las razones de nuestra elección es que Laravel suele ser más amigable para los principiantes gracias a su sintaxis hace que sea más rápido comprender y comenzar a trabajar con el framework. Otra razón es que ofrece una gran cantidad de características listas para usar que permiten el desarrollo rápido de aplicaciones. Además, mantener y actualizar las aplicaciones es más fácil con Laravel que con Symfony.

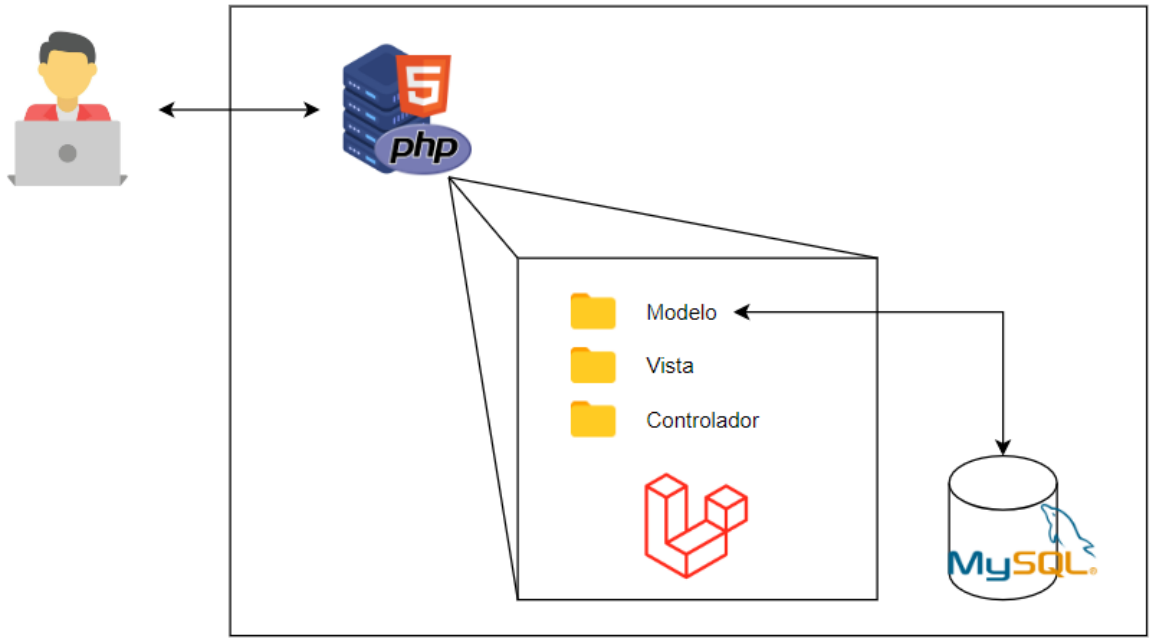
Arquitectura

Descripción

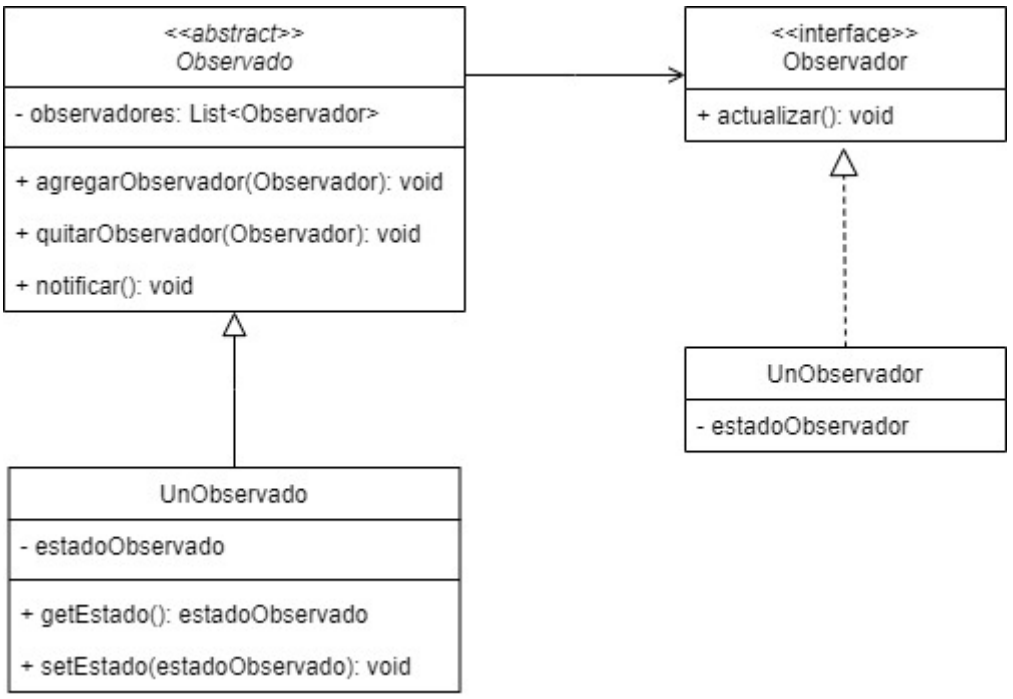
La arquitectura seleccionada para el desarrollo del proyecto se basa en el patrón Modelo-Vista-Controlador (MVC). Este enfoque arquitectónico se eligió debido a su capacidad para proporcionar una estructura organizada y modular que facilita el desarrollo, mantenimiento y escalabilidad de la aplicación. El patrón MVC permite la separación clara de las diferentes capas de la aplicación. El Modelo maneja la lógica de negocio y la interacción con la base de datos, la Vista se encarga de la presentación de la información al usuario, y el Controlador gestiona las solicitudes del usuario, interactuando con el Modelo y determinando qué Vista mostrar. Además, la estructura modular de MVC facilita la adición de nuevas funcionalidades o la modificación de las existentes sin afectar otras partes del sistema.

La elección del patrón Modelo-Vista-Controlador (MVC) se alinea con los requerimientos no funcionales cruciales para la tienda en línea. Usar esta arquitectura nos posibilita la optimización de cada componente de manera independiente para mejorar el rendimiento global de la aplicación. También permite un diseño centrado en la experiencia del usuario. Por último, nos proporciona una base sólida para la escalabilidad. La modularidad y la independencia entre componentes permiten agregar nuevas funcionalidades, manejar volúmenes crecientes de usuarios y adaptarse a cambios en el negocio sin afectar otras áreas del sistema.

Diseño de Arquitectura



Patrón de diseño: Observador



Componentes del patrón

Sujeto (Model):

- Es el objeto que está siendo observado.
- Mantiene una lista de observadores interesados en sus cambios de estado.
- Proporciona métodos para que los observadores se registren (suscriban) y se desregistran (cancelen la suscripción).

Observador (View):

- Define una interfaz para aquellos objetos que deben ser notificados de los cambios en el sujeto.
- Contiene un método de actualización que es llamado por el sujeto cuando su estado cambia.

Flujo de trabajo:

Registro de observadores:

- Los objetos interesados (observadores) se registran con el sujeto para recibir notificaciones sobre cambios.

Cambio de estado:

- El sujeto cambia su estado de alguna manera.

Notificación:

- El sujeto notifica a todos sus observadores registrados sobre el cambio de estado llamando a sus métodos de actualización.

Actualización:

- Cada observador concreto que implementa la interfaz de observador reacciona al cambio de estado realizando alguna acción específica.

Ventajas del patrón Observador:

Desacoplamiento:

- Permite que los sujetos y observadores interactúen sin necesidad de conocer detalles internos entre ellos, logrando un desacoplamiento.

Reusabilidad:

- Puedes reutilizar observadores en diferentes sujetos y viceversa, ya que están desacoplados.

Extensibilidad:

- Puedes agregar nuevos observadores sin modificar el sujeto, y viceversa, añadir nuevos sujetos sin afectar a los observadores existentes.

Diagrama de clases