UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

CIENCIAS DE LA COMPUTACION

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

INGENIERÍA DE SOFTWARE II

TEMA

VALIDAR LA ARQUITECTURA DE SOFTWARE PLANTEADA

GRUPO 4

INTEGRANTES

Caisaguano Ilaquiche Diana Verónica
Guashpa Bonilla Wilfrido Patricio
Luna Maza Karla Daniela
Mosquera Coronel Harlem Mateo
Troya Acosta Alexis Santiago

DOCENTE

Ing. Efraín Rodrigo Fonseca Carrera

18 junio del 2023

ÍNDICE

1. INTRODUCCIÓN

En la era actual, las tiendas en línea están creciendo significativamente, brindando a los usuarios una forma conveniente de acceder a una amplia variedad de productos y servicios desde la comodidad de sus hogares. Para aprovechar al máximo este próspero mercado y ofrecer una experiencia de compra única, es fundamental contar con un software de calidad que respalde y potencie las operaciones de una tienda en línea.

Por lo que se procederá al desarrollo de un software para una tienda en línea en la cual se visualizará los productos o servicios, para que los usuarios que navegan por internet los adquieran. Para una mejor comprensión de lo realizado en el siguiente proyecto, se presentarán el diagrama de uso del sistema a utilizar, diagrama de secuencia, arquitectura, etc.

2. OBJETIVOS

2.1 Objetivo general

Desarrollar un software el cual se tratará de una tienda en línea, poniendo en práctica los diagramas de casos de uso, arquitectura a utilizar para poder brindar a los usuarios una interfaz responsiva y de fácil uso.

2.2 Objetivos específicos

- Diseñar el software teniendo en cuenta las métricas de Métricas de usabilidad y las interfaces requeridas basadas en el paradigma de reutilización.
- Realizar las respectivas pruebas de caja blanca y caja negra para verificar su correcto funcionamiento.
- Revisar y analizar en detalle la estructura y diseño propuesto para identificar posibles problemas, deficiencias o errores en la arquitectura.

3. DESARROLLO

3.1 Problemática

La problemática que abordaremos en este caso práctico de estudio se basa en el desarrollo de un sistema de gestión de una tienda en línea. La tienda actualmente vende productos electrónicos y desea expandir su catálogo para incluir otros tipos de productos, como ropa, accesorios, libros, etc. Además, se espera que el sistema evolucione a medida que la tienda crezca y se requieran nuevas funcionalidades.

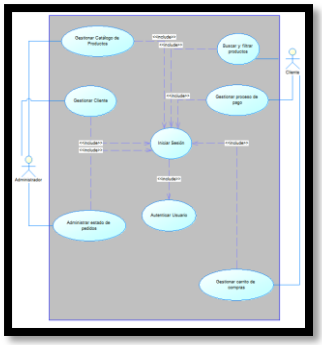
3.1.1 Requerimientos generales del sistema

a) Diagrama de casos de uso

La importancia del diagrama de caso de uso es que redacta todas las funciones básicas o compuestas del sistema desde el punto de vista de los usuarios internos y externos de manera que estos usuarios puedan entenderlos, describen el comportamiento de los usuarios por lo tanto es de vital importancia que al momento de crearlos el usuario final debe revisar y aceptar el caso de uso. (rvillarroel, 2017).

Por lo que a continuación se muestra el diagrama de caso de uso general de la tienda en línea.

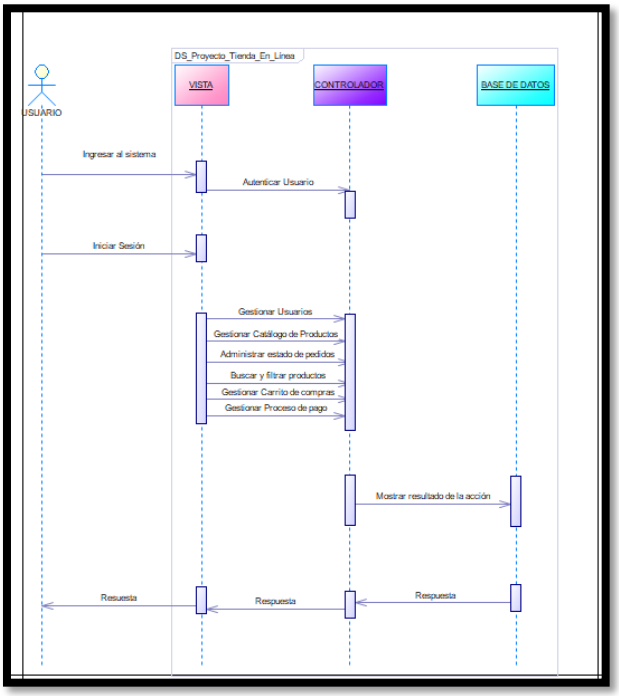
Ilustración 1 Diagrama de caso de uso general del sistema



Fuente: Autoría propia

b) Diagrama de Secuencia

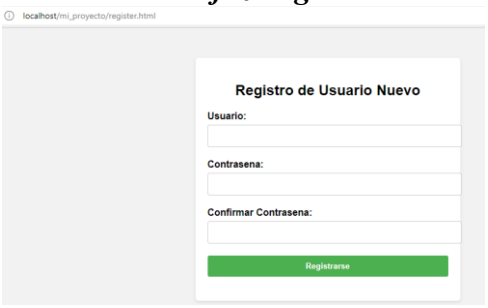
Ilustración 2 Diagrama de secuencia general del sistema



Fuente: Autoría propia

Diseñe el software con las interfases requeridas basados en el paradigma de reutilización.

Ilustración 3 Interfaz Registro de usuario



Fuente: Autoría propia

Ilustración 4 Interfaz de Inicio de Sesión

Fuente: Autoría propia

En nuestras interfaces, aplicamos un enfoque eficiente y consistente al reutilizar varios componentes del formulario de registro en el formulario de inicio de sesión. Esto implica aprovechar elementos como los cuadros de texto, botones, mensajes de indicación de entrada, textos y esquemas de colores. Al reutilizar estos componentes, garantizamos una experiencia de usuario coherente y familiar, lo que facilita la interacción y reduce la curva de aprendizaje.

Además de los elementos visuales, también reutilizamos ciertas validaciones en ambos formularios. Por ejemplo, implementamos validaciones comunes como los campos requeridos y los formatos de entrada para garantizar que los usuarios proporcionen la información necesaria en el formato adecuado. Esto no solo ahorra tiempo y esfuerzo en el desarrollo, sino que también brinda una experiencia consistente al usuario, ya que se enfrenta a los mismos requisitos en ambas funciones.

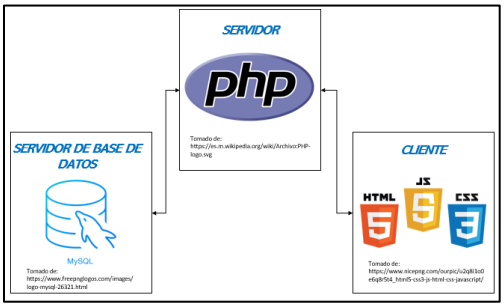
Así, al reutilizar componentes visuales y validaciones, buscamos optimizar la eficiencia y la usabilidad de nuestras interfaces, brindando una experiencia coherente y conveniente para los usuarios.

3.2 Depure la arquitectura del sistema propuesta

La arquitectura del sistema constara de:

- **Servicio de Base de Datos**
  - **MySQL:** La base de datos relacional MySQL nos permitirá almacenar y acceder los datos a través del motor de almacenamiento siendo capaz de replicar datos y particionar tablas para mejorar el rendimiento y la durabilidad.
- **Servidor**
  - **PHP:** Este lenguaje nos servirá para conectar ciertos elementos de la página con determinados datos, esto con el objetivo de que el contenido cambie con las peticiones de los usuarios y que esta petición se manda a un servidor que, a su vez, contacta con la base de datos de donde extrae la información.
- **Cliente**
  - **HTML:** Será el responsable de darle estructura a la información dentro del sitio.
  - **JS:** Se encarga del componente interactivo como: animaciones o respuestas, cuando se ejecute una acción.
  - **CSS:** Se encargará de agregar diseño como: colores, fuentes, sombras.

4. Ilustración 4.Arquitectura de Tienda en Línea



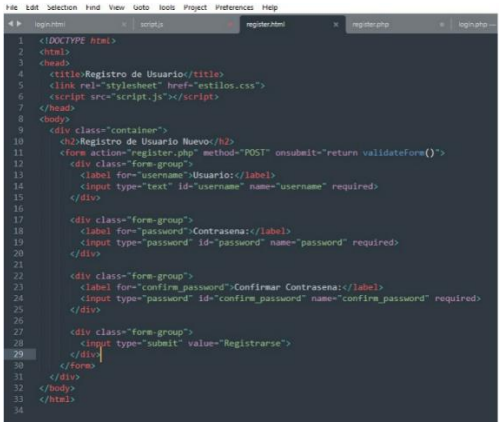
Fuente: Autoría propia

3.3 Proponga actualizaciones en el software y realice el mantenimiento inherente a las mismas.

3.4 Proponga casos de prueba de caja blanca y de caja negra y realice las pruebas (automáticas o manuales).

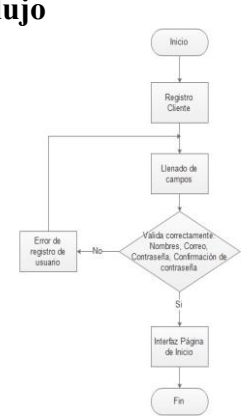
3.4.1 PRUEBA DE CAJA BLANCA

Extracto del código del RF2 de Registro de Usuario Nuevo

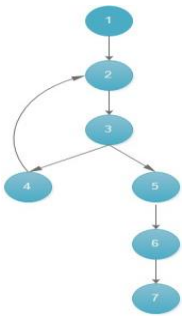


```
<!DOCTYPE html>
<html>
<head>
  <title>Registro de Usuario</title>
  <link rel="stylesheet" href="estilos.css">
  <script src="script.js"></script>
</head>
<body>
  <div class="container">
    <h2>Registro de Usuario Nuevo</h2>
    <form action="register.php" method="POST" onsubmit="return validateForm()">
      <div class="form-group">
        <label for="username">Usuario:</label>
        <input type="text" id="username" name="username" required>
      </div>

      <div class="form-group">
        <label for="password">Contrasena:</label>
        <input type="password" id="password" name="password" required>
      </div>

      <div class="form-group">
        <label for="confirm_password">Confirmar Contrasena:</label>
        <input type="password" id="confirm_password" name="confirm_password" required>
      </div>

      <div class="form-group">
        <input type="submit" value="Registrarse">
      </div>
    </form>
  </div>
</body>
</html>
```

3.4.2 Diagramas de Flujo



3.4.3 Grafo



Rutas:

R1: 1,2,3,4,2

R2: 1,2,3,5,6,7

Complejidad Ciclomática

Esta se calcula de la siguiente manera:

V(G) = E – N + 2

V(G) = 7-7+2

V(G) = 2

V(G) = P + 1

V(G) = 1 +1 = 2

Donde

Prueba De Caja Neg

Validación de Registro de Usuario Nuevo

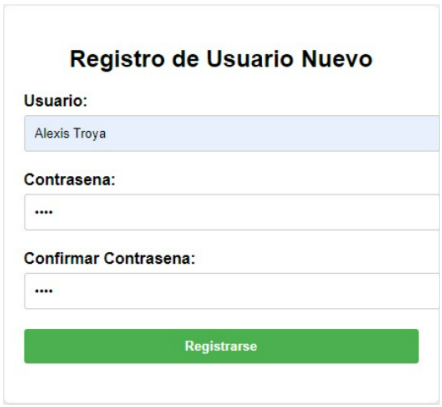
Requerimiento Funcional 2

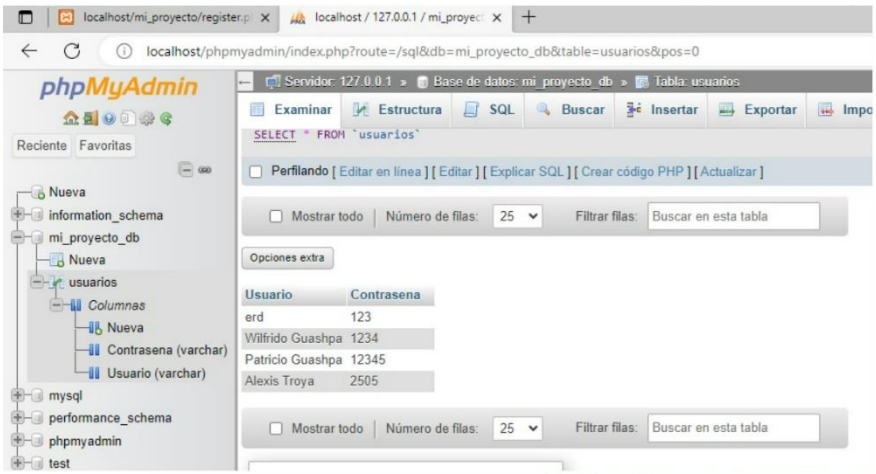
Partición de clases equivalentes

VARIABLE	CLASE DE EQUIVALENCIA	ESTADO	REPRESENTANTE
Nombres	EC1: Nombres == TextNombres	Válido	Alexis Troya
	EC2: Nombres!= TextNombres	No válido	----------
Correo electrónico	EC3: email == [a-zA-Z0-9]+@[a-zA-Z0-9]+([.][a-zA-Z09_]+)*[.][a-zA-Z]{1,5}	Válido	astroya1@espe.com
	EC4: email != [a-zA-Z0-9]+@[a-zA-Z0-9]+([.][a-zA-Z09_]+)*[.][a-zA-Z]{1,5}	No válido	astroya1@espe.com
Contraseña	EC5: if len($contrasena) > 8:	Válido	Alexis2505
	EC6: if len($contrasena) < 8	No válido	----------
Confirmación - Contraseña	EC5: if len($contrasena) > 8:	Válido	Alexis2505
	EC6: if len($contrasena) < 8	No válido	

Prueba de Campos Completos.

Si ingresamos todos los campos solicitados en el usuario podrá ingresar correctamente y se almacenará en la base de datos





Prueba de Campos No Completados

Si los campos de usuario o contraseña o confirmar contraseña se encuentran vacíos nos presente el mensaje “Rellene este campo”

Link del video:
https://youtu.be/qN-hQsJhdpQ

4. CONCLUSIONES

- La arquitectura MVC promueve una clara separación de responsabilidades entre el modelo, la vista y el controlador. Esto facilita el mantenimiento, la escalabilidad y la reutilización del código.
- Gracias al modularidad inherente de la arquitectura MVC, es más fácil reutilizar componentes de código en diferentes partes de la tienda en línea. Esto puede acelerar el desarrollo y mejorar la calidad del software.

- Al separar la lógica de presentación de la lógica de negocio, la arquitectura MVC puede mejorar la experiencia del usuario al garantizar una respuesta rápida y fluida a las acciones realizadas en la tienda en línea.

5. RECOMENDACIONES

- Comunicación efectiva: Establece una comunicación clara y abierta con el cliente para comprender completamente sus necesidades y expectativas. Programa reuniones regulares, entrevistas o talleres de trabajo para discutir y documentar los requerimientos.
- Escucha activa: Presta atención a las necesidades y requerimientos del cliente.
- Documentación detallada: Registra todos los requerimientos en detalle y de manera organizada.
- Validación y verificación: Una vez que hayas recopilado los
- Mantén la flexibilidad: Reconoce que los requerimientos pueden evolucionar a medida que avanza el proyecto.
- Pruebas y validación: Una vez que los requerimientos se implementen, realiza pruebas y validaciones para asegurarte de que cumplen con las expectativas del cliente.

BIBLIOGRAFÍA

¿Por qué es importante actualizar el sistema operativo? (Diciembre de 2022). Obtenido de Argentina.gob.ar: https://www.argentina.gob.ar/justicia/convosenlaweb/situaciones/por-que-es-importante-actualizar-el-sistema-operativo#:~:text=Las%20actualizaciones%20instalan%20mejoras%20en,solucionar%20vulnerabilidades

Los 4 tipos de mantenimiento de software. (s.f.). Obtenido de Thales: https://cpl.thalesgroup.com/es/software-monetization/four-types-of-software-maintenance#:~:text=El%20mantenimiento%20de%20software%20es,satisfacer%20las%20necesidades%20del%20cliente.

McConnell, S. (2004). *Code Complete: A Practical Handbook of Software Construction* . (2nd Edition). Microsoft Press.

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach.* (8th Edition). McGraw-Hill.

rvillarroel. (22 de enero de 2017). *ingenieriadesoftware*. Obtenido de https://ingenieriadesoftwareutmachala.wordpress.com/2017/01/22/casos-de-uso-importancia/

Shaw, M. &. (1996). *Software Architecture: Perspectives on an Emerging Discipline.* Prentice Hal.