Universidad de las Ciencias Informáticas

“Facultad 2”

Título: “Sistema web para la búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D’Prisa”

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Rafael Alejandro Ruano Rodríguez
Tutor(es): Ing. Glennis Tamayo Morales
Ing. José Alexei Leyva Desdín
Ing. Osmar Vega González

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: Sistema web para la búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Rafael Alejandro Ruano Rodríguez

_____

Firma del Autor

Ing. Glennis Tamayo Morales

_____

Firma del Tutor

Ing. José Alexei Leyva Desdín

_____

Firma del Tutor

Ing. Osmar Vega González

_____

Firma del Tutor

FRASE

Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido…"

Ché

DATOS DE CONTACTO

Tutores:

Ing. Glennis Tamayo Morales.

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

E-mail: gtamayo@uci.cu

Ing. Osmar Vega González

Universidad de las Ciencias Informáticas, Cuidad de la Habana, Cuba

E-mail: ovega@uci.cu

Ing. José Alexei Leyva Desdín

Universidad de las Ciencias Informáticas, Cuidad de la Habana, Cuba

DEDICATORIA

A Dios y a mis padres porque son mi espíritu y mi fuerza, a mis abuelos Francisco, Hilda y Eustaquio, Noelia.....

AGRADECIMIENTOS

Agradecerles a las personas más importantes de vida, mis padres Odalis y Eligio por el amor tan grande que siempre me han dado, por ser mis guías en todo momento, por ayudarme a superar cada día que pasa con sus consejos y apoyos, por ser la inspiración más grande por la cual se hace posible este sueño de ser ingeniero, de corazón se los debo todo, gracias por cada momento que han dedicado y aún dedican en forjar mis valores y educación para que sea un hombre de bien. Los amo con la vida.

Agradecerles a mis abuelos Daysis, Ramón y Nilda por todo el amor y el cariño que siempre me han dado aunque no sea su nieto de sangre, por los consejos y las enseñanzas que me han brindado durante todo mi vida.

Agradecerles a mis otras madres Alina, Ailé, Xiomara, Olga, Vivián por quererme como a un hijo y educarme a lo largo de todo estos años de vida.

Agradecerle a todos mis tíos en especial a mi tía Esperanza, Fe, Anairis, Madelaine, Tania, Berta, Alida, Encarnación y Aidita, a mi tío Gerencio, Luis, Ribiaux, Gito,

Alberto, Alexis y Denis por ser como padres en todo momento, aconsejándome y ayudándome demostrándome su amor y cariño.

Agradecerles a todos mis hermanos y primos en especial Anabel, Anaiya, Glendys, Gleidys, Leodán, Anneris, Maykel, Angelin, Serguey, Raulito, Alejandro, Jorgito, Tavo y Liván por brindarme tanto amor, cariño y respeto a este hermano que lo quiere mucho, por todos los consejos y enseñanzas que me han dado.

Agradecerle a Darien y Annia por ser como padres, por ayudarme a salir adelantes en la carrera, dándome apoyo y comprensión en todo momento.

Agradecerle a mi novia Alina por todo el amor y cariño que me ha dado desde que nos conocimos, por la comprensión y ayuda que me ha brindado en todo este proceso de la tesis. Además a su hermana y su mamá Mireya por el apoyo brindado a lo largo de todo este período.

Agradecerles a mis tutores Glennis, Osmar y Alexeis por el apoyo brindado durante todo este proceso de tesis, ayudándome para que todo saliera, siempres positivos, con una alegría y enseñándome en todo momento. Gracias de todo corazón.

Agradecerles a mis todos mis amigos en especial a Ernesto, Eric, Pastor, César, Edilberto, Fernando, Eliver, Leo, Ericito, César Eduardo, Jorge Luis, Ibrahim, Pocholo, Pablo, Abel, Hiram, Pedro, Leyan, Daniel, Daniel el Flaco, Kirian, Vicente por ser como mis

hermanos en todo momento, por estar en las buenas y en las malas y más en las malas, por la comprensión y el respeto que me han brindado en todo este tiempo que tenemos de conocernos.

Agradecerles Gendis, Gonzálo, Maykel por la su amistad incondicional, por los consejos que me han dado a lo largo de toda mi carrera.

Agradecerles a todos mis profesores desde la primaria hasta la universidad por haber contribuido en mi formación como profesional, por su apoyo, compresión y enseñanzas.

Agradecerles de manera general a todas las personas que de una forma u otra han formado parte de mi educación durante estos 5 años de carrera, ayudándome a lograr este sueño.

RESUMEN

En la Empresa de Tecnologías de la Información para la Defensa (XETID) pertenecientes a las Fuerzas Armadas Revolucionarias (FAR) se trabaja en el desarrollo de varios proyectos destinados a la informatización de los principales procesos del Ministerio de las FAR y de la sociedad. Formando parte de este conjunto de proyectos se encuentra la plataforma D´Prisa que surge como estrategia trazada por el centro de Sistemas Operativos (TeDis). En este proyecto se desarrollan diversos sistemas y módulos dirigidos a mejorar las diferentes esferas de la sociedad, así como sistemas de carácter militar como parte del desarrollo tecnológico de las Fuerzas Armadas Revolucionarias para la defensa del país. El presente trabajo de diploma tiene como objetivo desarrollar un Sistema web para de búsqueda de Productos en Tiendas que facilite la búsqueda de los clientes a la hora de realizar una compra de un producto. Con el estudio realizado sobre los buscadores de productos existentes en Cuba y en el mundo se definió la estructura básica del sistema web. Se seleccionó la Arquitectura de Microservicio y AUP-UCI como metodología de software, así como las distintas herramientas y tecnologías que fueron empleadas en el desarrollo de la solución, para el desarrollo del sistema se definió los requisitos funcionales y no funcionales del mismo. Se definió una estrategia de prueba quedando plasmados los tipos de prueba funcionales y aceptación, y los método de prueba escogido fue de Caja Blanca con la técnica de Camino Básico y la prueba de Caja de Negra con la técnica partición equivalente, todo con el objetivo de encontrar y corregir posibles errores.

PALABRAS CLAVE

Búsqueda, microservicios, plataforma, productos, sistema, tiendas,

ABSTRACT

The Defence Information Technology Company (XETID) belonging to the Revolutionary Armed Forces (FAR) is working on the development of several projects aimed at computerising the main processes of the Ministry of FAR and society. Forming part of this set of projects is the platform D´Prisa which arises as a strategy drawn up by the Centre for Operating Systems (TeDis). In this project several systems and modules are developed aimed at improving the different spheres of society. The present diploma work aims to develop a Product Search System in Shops that facilitates this task. The basic structure and main functionalities of the system to be carried out were defined with the study carried out on the existing product search engines in Cuba and in the world. The Microservice Architecture and AUP-UCI were selected as software methodology, as well as the different tools and technologies that were used in the development of the solution. A test strategy was defined with the functional and acceptance test types being captured, and the test method chosen was the White Box with the Basic Way technique and the Black Box test with the equivalent partition technique, all with the objective of finding and correcting possible errors.

KEYWORDS

Platform, System, Search, Products, Shops, Microservices

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo tecnológico que se ha producido ha propiciado lo que muchos autores denominan la nueva “revolución” social, con el desarrollo de "la sociedad de la información". Con ello, se hace alusión a que la materia prima: "la información" será el motor de esta nueva sociedad, y entorno a ella, surgirán profesiones y trabajos nuevos, o se readaptarán las profesiones existentes, y en este aspecto un elemento de marcada importancia lo constituyen las Tecnologías de la Información y de las Comunicaciones (TIC).

Las TIC han jugado un papel importantísimo en el desarrollo de la sociedad, estas forman parte de la mayoría de los sectores como: educación, cultura, política, salud y economía. Su crecimiento y aplicación en estos últimos años ha revolucionado la concepción tradicional de la comunicación, el aprendizaje, la economía y el mundo en general. En el sector económico comercial el impacto de las Tecnologías de la Información y la Comunicación (TIC) es tan importante y aborda tantos ámbitos que configura lo que ya se conoce como los pilares de la nueva economía o economía digital. La comprensión del impacto de la revolución que suponen estas tecnologías es un fenómeno complejo, objeto de estudios académicos de diversas disciplinas científico-técnicas y socioeconómicas, y está en el centro de las políticas económicas de países avanzados, desarrollados y en desarrollo. La evidencia muestra que las TIC son el núcleo duro de un proceso creciente de integración en los mercados globales, que aceleran la innovación y la gestión en las empresas, crean riqueza y, en definitiva, generan desarrollo y bienestar. (Prieto Paco, 2019)

Y es que el uso intensivo de las TIC ha generado cambios en la manera de divertirse, de trabajar, en la organización de las empresas y las administraciones públicas, cambios en la comunicación y nuevas formas de comprar y vender. Las nuevas tecnologías permiten crear, manipular, organizar, transmitir, almacenar y gestionar de manera ágil, flexible y, sobre todo, a coste muy bajo, en ocasiones casi cero, la información. Las empresas se enfrentan al reto de tener clientes cada vez más informados y exigentes por lo que necesitan buscar nuevos métodos para así poder promocionar sus productos y servicios, lo que acelera su competencia y con ello la necesidad de crear nuevas oportunidades de negocio sobre las bases de diferenciación, conocimiento, control de la gestión, comercialización y, en definitiva, innovación. (Prieto Paco, 2019)

Tecnologías como los sitios web forman parte de las alternativas o métodos que tienen las empresas para lograr acelerar su mercado en cuanto a compra, contratos y servicios. Estos son capaces de ser una guía para el cliente a la hora de realizar una búsqueda para comprar un producto, solicitar un servicio, realizar alguna gestión empresarial o cualquier otro uso según el contenido de la misma. Esta manera de facilitar la búsqueda del cliente se considera un comercio convencional que usa como medio principal para realizar sus transacciones un sitio web de internet. Estos sitios podrán utilizarse en la red nacional o internacional sin tener la necesidad de contar con la presencia física de usuarios interesados en las ofertas del sitio.

En la actualidad, estos sitios son frecuentados diariamente por las comodidades y ventajas que brindan, lo que permite que los clientes interesados en hacer una compra de un producto o solicitar un servicio realicen sus búsquedas sin necesidad de ir hasta el local o tienda física para saber si el producto o servicio se oferta o no. Entre los elementos que conforman un sitio web se encuentran los catálogos donde se muestra el abanico de productos o servicios ofertados.

En Cuba, que no se encuentra ajena a toda esta revolución del uso de las tecnologías para mejorar la calidad de vida de sus habitantes, existen un conjunto de aplicaciones web, así como móviles que brindan información sobre los productos y servicios que se ofrecen en las diferentes cadenas de tiendas y empresas del país. Si bien es cierto que existen aplicaciones que contienen información online de los productos y servicios que se ofrecen en las diferentes cadenas de tiendas, cabe resaltar que estas aun no satisfacen a los usuarios que la utilizan, debido a que ninguna garantiza la posibilidad de tener toda la información de los productos en un solo sitio web, así mismo en la mayoría de estos la mayor parte del tiempo la información se encuentra desactualizada y obsoleta lo cual hace que estas aplicaciones sean poco fiables.

Por lo anteriormente planteado se identifica como **problema a resolver:** ¿Cómo contribuir a la búsqueda de productos en tiendas empleando arquitectura de micro servicios sobre la plataforma D'Prisa?, enmarcándose en el **objeto de estudio**: tecnologías para la búsqueda en sistemas web, centrado en el **campo de acción:** la búsqueda de productos en aplicaciones de tiendas en línea.

Para darle solución al problema planteado, se define como **objetivo general:** desarrollar un sistema web para la búsqueda de productos en tiendas, que permita dar a conocer la información de un producto determinado a un cliente al realizar una compra.
Para darle cumplimiento al objetivo general se proponen las siguientes **tareas de la investigación**:

1. Caracterización de las diferentes tecnologías que son utilizadas para realizar búsquedas en sistemas web.
2. Caracterización de las diferentes aplicaciones de tiendas en líneas que se utilizan en Cuba.
3. Caracterización de la metodología, herramientas y tecnologías a utilizar en el desarrollo de la propuesta de solución.
4. Identificación de las funcionalidades de la propuesta de solución.
5. Diseño de la solución a partir de las funcionalidades identificadas.
6. Implementación de las funcionalidades identificadas.
7. Diseño de los casos de pruebas para la aplicación de estas a la solución desarrollada.
8. Aplicación de los casos de pruebas para validar que la solución desarrollada y responde a las necesidades del usuario final.

Para el desarrollo de la investigación se emplearon **métodos científicos**, estos fueron agrupados en métodos teóricos y métodos empíricos:

Métodos Teóricos

- ✓ **Análisis y síntesis:** Está integrado por el desarrollo del análisis y la síntesis, mediante el cual se descompone un objeto, fenómeno o proceso en los principales elementos que lo integran para analizar, valorar y conocer sus particularidades, y simultáneamente a través de la síntesis. El mismo se utilizó para el estudio y análisis de diferentes fuentes bibliográficas con el objetivo de obtener un amplio conocimiento acerca de los sistemas de búsqueda de productos sobre las funcionalidades que presentan y las tecnologías que utilizan.
- ✓ **Modelación:** Es justamente el método mediante el cual se crean abstracciones con vistas a explicar la realidad. La modelación es el método que opera en forma práctica o teórica con un objeto, no en forma directa, sino utilizando cierto sistema intermedio, auxiliar, natural o artificial.

Este método se utilizó para modelar los diagramas, la arquitectura y los distintos procesos que se realizan en la construcción del Sistema de búsqueda de productos en tiendas.

Métodos Empíricos:

- ✓ **Observación:** Con este método es posible distinguir directamente los hechos de la realidad objetiva. Permite conocer el proceso delimitado como objeto de estudio, lo cual contribuyó a tener un registro visual más detallado de lo que se quiere y hace falta hacer; y cómo hay que hacerlo. Mediante este método se observaron varios sistemas de búsqueda de productos y se obtuvo una guía para el desarrollo del nuevo sistema.
- ✓ **Entrevista:** Este método es una técnica de recopilación de información mediante una conversación profesional, con el que se adquiere información acerca de lo que se investiga. Esta puede estar estructurada o no mediante un cuestionario previamente elaborado donde la información que se obtiene resulta fácil de procesar. Mediante este método se pudieron definir parte de los requisitos funcionales del sistema, en una entrevista con los clientes (Ver anexo 1).

El documento se descompone en tres capítulos, estructurados de la siguiente forma:

Capítulo 1. Fundamentación teórica: Se analizan los conceptos y elementos que conforman el objeto de estudio de la presente investigación. Se abordan los fundamentos teóricos relacionados con el estado del arte a nivel internacional y nacional; y se realiza una descripción de la metodología, tecnologías y herramientas a utilizar para el desarrollo de la solución propuesta.

Capítulo 2. Análisis del Sistema: Se realiza una descripción del sistema a desarrollar; se abordan los fundamentos teóricos del modelo del dominio, se realiza una descripción del funcionamiento del sistema propuesto, y se representa la estructura de este con el empleo de la metodología seleccionada.

Capítulo 3. Diseño e implementación: Se realiza una descripción de la arquitectura del módulo a desarrollar, se especifican los patrones de diseño a utilizar por los desarrolladores y se elabora el diseño de la solución mediante los diagramas de clases del diseño, y se representa el diagrama de despliegue.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los elementos que corresponden al marco teórico de la investigación. Se realiza un análisis del estado de arte de los sistemas de búsqueda de productos en tiendas. Se caracterizan las herramientas y tecnologías para la elaboración de la propuesta de solución, así como la metodología de desarrollo de software a emplear.

1.1 Conceptos asociados a la investigación

D'Prisa

Es una plataforma desarrollada por la empresa la Empresa de Tecnologías de la Información para la Defensa (XETID), empresa militar perteneciente a las FAR (Fuerzas Armadas Revolucionaria) que trabaja en el desarrollo de soluciones informáticas para automatizar la defensa del país. D'Prisa tiene como objetivo principal el desarrollo rápido de aplicaciones a través de un espacio colaborativo donde los desarrolladores podrán compartir código listo para su empleo en forma de módulos y sistemas de software autónomos denominados "MicroServicios". Gracias a esta tecnología que se va imponiendo gradualmente en el mundo, se podrán crear aplicaciones distribuidas y escalables. (Nube", 2019)

Permitirá también aplicar la filosofía de "*MicroFramework*" donde, no se requerirá emplear un marco de trabajo completo para utilizar un par de funciones sencillas. Se podrá seleccionar única y exclusivamente los elementos necesarios para desarrollar la aplicación. Se propone, además, el desarrollo de "*MicroFrontends*" a partir de un conjunto de lineamientos que permitirán un desarrollo mucho más rápido de aplicaciones *Web* y aplicaciones Android, entre otros. Inicialmente constituirá una propuesta que será validada y enriquecida con la práctica y el uso diario. Otra de las funciones que pretende realizar D´Prisa es ofrecer una arquitectura básica de desarrollo de aplicaciones a partir de los conceptos mencionados que, si bien no es la solución para todo el universo de problemas que la informática tiene el deber de resolver, se puede acercar bastante a un 70% u 80% de los casos. (Santiago, 2018)

MicroServicios

Una arquitectura de microservicios es un enfoque para desarrollar una aplicación de *software* como una serie de pequeños servicios, cada uno ejecutándose de forma autónoma y comunicándose entre sí. Normalmente hay un número mínimo de servicios que gestionan cosas comunes para los demás (como el acceso a base de datos), pero cada microservicio es pequeño y corresponde a un área de negocio de la

aplicación. Además, cada uno es independiente y su código debe poder ser desplegado sin afectar a los demás. Incluso cada uno de ellos puede escribirse en un lenguaje de programación diferente, ya que solo exponen la API (una interfaz común, a la que le da igual el lenguaje de programación en la que el microservicio esté programado por debajo) al resto de microservicios.

Ventajas de usar microservicios:

Escalabilidad:
A diferencia de los monolíticos, donde se deben escalar toda la aplicación para soportar picos en una funcionalidad concreta, con los microservicios se puede escalar partes de manera independiente al resto, por lo cual ajustas la infraestructura a las necesidades concretas del momento sin sobre dimensionar como con los monolitos.

Modularidad:
Cada microservicio debe realizar una función acotada. Los desarrolladores trabajan en un código más aislado y limpio.

Tecnologías:
En cuanto a tecnologías, se tiene la ventaja también de utilizar la tecnología más adecuada para cada funcionalidad.

Para desarrollar con microservicios se ha de tener en cuenta que el tiempo y los costes son limitados. En este sentido, hay que encontrar el punto de inflexión adecuado en el que una arquitectura de este tipo es suficientemente ventajosa y beneficiosa, o, dicho de otra manera, que aporte valor.

MicroFrameworks

Microframework es el término que ha sido usado para referirse a *Frameworks* minimalistas de aplicaciones web, en contraste con *Frameworks* Full-Stack. Un *microframework* carece de la mayoría de funcionalidades que es común esperar en un Full-Stack, (como lo es Django) tales como manejo de cuentas, autenticación, autorizaciones, roles, permisos, abstracción de bases de datos a través de un mapeo de objetos relacionales, motores de plantillas *web*, entre otras. Puntualmente lo que se busca al usar este *microframework*, es facilitar la recepción de peticiones HTTP, entregando la petición al controlador apropiado, despachar, y devolver una respuesta HTTP. (Henao, 2018).

Sin embargo, esta idea no es nueva, en el pasado se llamaba Integración de Frontend para Sistemas Verticalizados o Sistemas Autocontenidos. Pero MicroFrontends es claramente un término más amigable y menos voluminoso. (Geers, 2018).

1.2 Análisis de soluciones existentes.

Sistemas de búsquedas de productos de tiendas en línea

Donde Hay

Donde Hay es un sitio *web* con descarga de aplicación androide perteneciente a la Tiendas Panamericanas de la firma Cimex.sa, su interfaz proporciona una búsqueda para los usuarios a través de un botón principal, además cuenta con otro botón en la esquina inferior derecha que permite limpiar la lista de la búsqueda de un cliente en caso de que quiera buscar otro producto y en el margen superior izquierdo aparece un menú que despliega una barra con dos opciones Filtro y Ayuda , el filtro se realiza por provincias, municipios, precios y productos en rebaja, la opción de ayuda nos brinda un tutorial de la aplicación y los contactos de los desarrolladores de la aplicación. El sitio es de gran ayuda para los clientes, pero actualmente presenta problemas de actualización con las bases de datos además que no cumple con todas las necesidades de los clientes debido a que la información que presenta solo pertenece a la cadena de Tiendas Panamericanas.

Ofertas

Ofertas es un sitio *web* cubano perteneciente a la Agencia de Información Nacional, el mismo tiene como objetivo principal promover los diferentes tipos de productos y servicios que en el país se ofertan por parte de instituciones estatales como de propietarios privados. En su página principal cuenta con un botón de que permite la búsqueda de los productos y servicios por parte de los usuarios, además contiene un botón que permite al usuario crear un anuncio, del producto o servicio que ofrece o necesita. En la misma se pueden apreciar los diferentes tipos de productos y servicios que se ofrecen divididos por categorías permitiéndole al usuario mayor usabilidad del sistema y garantizándole una mejor búsqueda. El sitio es de gran ayuda para los usuarios, pero no soluciona en ocasiones las necesidades de los usuarios debido a que todos los usuarios pueden crear un anuncio, ofreciendo un producto o servicio y muchas veces estos no son las cantidades suficientes para satisfacer las necesidades de los usuarios.

Revolico

Revolico es un sitio *web* cubano que permite promover los diferentes tipos de productos y servicios que se ofertan en el país por parte de propietarios privados. El mismo tiene mucha similitud con el sitio *web* Ofertas.cu, en su página principal contiene un botón que permite la búsqueda de los productos y servicios, además de uno que permite insertar un anuncio, los productos y servicios que el mismo muestra, salen divididos por categorías garantizando una búsqueda más fácil para el usuario. El sitio es muy usado por los ciudadanos cubanos. Pero tiene como principal inconveniente que su consulta no es autorizada dentro de instituciones estatales del país debido a que sus servidores no están ubicados en el país lo que trae como consecuencia que la información de los productos y servicios que en este se anuncian pueden ser legales o no, de manera que no se puede controlar la misma. Además de que muchos de los productos y servicios ofertados por un usuario no satisfacen las necesidades de todas las personas.

En la siguiente tabla se muestran los sistemas de búsqueda de productos.

Tabla 1.Funciones de las soluciones existentes.

Sitios	**Características**
Donde Hay	Permite realizar una búsqueda simple y avanzada, la búsqueda avanzada se realiza por filtros de provincias, municipios, precios y productos en rebajas, realiza un mapeo de los resultados obtenidos según la ubicación del mismo.
Ofertas	Permite realizar una búsqueda de los productos, se listan los productos por categorías, permite crear un anuncio de ofertando o solicitando un producto o servicio.
Revolico	Permite realizar una búsqueda de los productos, se listan por categorías, permite crear un anuncio de ofertando o solicitando un producto o servicio dado.

Conclusiones parciales del estudio

Después de realizar un análisis de las soluciones existentes se decide realizar un sistema de búsqueda de productos que tenga como objetivo principal la búsqueda de productos en tiendas. El cual sea capaz de dar respuesta a través de criterios de búsquedas simple o avanzada, que permita al usuario dar opiniones acerca de los productos y además que sea capaz de actuar como sitio para el debate entre los clientes a través de comentarios y respuestas. Además de contar con la arquitectura de microservicio necesaria para agregarlas a la plataforma D'Prisa.
Además, se definen como elementos fundamentales:

- ✓ Permitir mostrar los resultados de la búsqueda del usuario.
- ✓ Mostrar una interfaz principal que muestre nombre, logo y slogan del sitio así como los productos por categorías.
- ✓ Permitir la selección de los filtros.
- ✓ Permitir que el usuario pueda limpiar la búsqueda realizada.
- ✓ Mostrar al usuario las opiniones de un producto buscado.
- ✓ Permitir al usuario crear su opinión acerca de un producto.
- ✓ Mostrar a los usuarios todos los comentarios existentes.
- ✓ Permitir a los usuarios crear comentarios acerca de un tema de su interés.
- ✓ Permitir que el usuario pueda buscar un comentario de su interés.
- ✓ Permitir a los usuarios crear respuestas acerca de un comentario de su interés.
- ✓ Mostrar los productos por categorías.
- ✓ Mostrar la disponibilidad y otros datos de los productos encontrados mediante la búsqueda.
- ✓ Mostrar una información de ayuda al usuario, así como los datos de los desarrolladores del sitio.

Herramientas, tecnologías y metodología

Desarrollar un *software* implica que sea necesario definir las tecnologías, herramientas y metodología de desarrollo que se van a emplear para lograr el objetivo planteado (Cendejas, 2011). Para la selección de las herramientas, tecnologías y metodología de desarrollo a emplear en el desarrollo de la solución se tuvieron en cuenta las definidas por el cliente.

Metodología AUP-UCI

La metodología de desarrollo de software Proceso Unificado Ágil (AUP) para la UCI, es una adaptación que se definió en la universidad a partir de la metodología AUP y está definida como el escrito oficial que rige la actividad de producción de software en la universidad. La variación AUP-UCI tiene tres fases (Inicio, Ejecución y Cierre).

Descripción de las disciplinas:

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían Gestión de la configuración (CM), Planeación de proyecto (PP) y Monitoreo y control de proyecto (PMC) (Calvache, 2016).

De los cuatro escenarios que define la metodología, se selecciona para llevar a cabo esta investigación es el escenario 2, ya que el mismo está destinado para proyectos donde el principal objetivo es la gestión y presentación de la información. Este escenario exige para la modelación del negocio el Modelo Conceptual y para encapsular los requisitos los Casos de Uso del Sistema.

Herramientas y Tecnologías

Angular V. 7.3.3

Es un *framework* JavaScript, gratuito y *Open Source*, creado por Google y destinado a facilitar la creación de aplicaciones *web* modernas de tipo SPA (Single Page Application). Angular utiliza como lenguaje de programación principal TypeScript, un súper-conjunto de JavaScript/ECMAScript que facilita mucho el desarrollo. (Pascal Precht , 2016).

HTML 5

HTML5 es la última versión de HTML. El término representa dos conceptos diferentes: Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance. A este conjunto se le llama HTML5 y amigos, a menudo reducido a HTML5. (MDN Web Docs, 2019).

CSS 3

Se utilizará el estilo CSS el cual es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página. (Uniwebsidad, 2019).

TypeScript

Es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft, el mismo hace su aparición en 2012. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases. Typescript es un superset de JavaScript. Un superset de un lenguaje de programación, es cuando puede ejecutar programas de la tecnología, Typescript en este caso, y del lenguaje del que es el superset, JavaScript en este mismo ejemplo. En resumen, esto significa

que los programas de JavaScript son programas válidos de TypeScript, a pesar de que TypeScript sea otro lenguaje de programación. (S Somasegar, 2019).

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas *web* dinámicas. Una página *web* dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Node.JS versión 8.9.3

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

PostgreSQL 9.4.0

Es un sistema gestor de bases de datos relacionales, está orientado a objetos, es multiplataforma y *Open Source***.** Publicado bajo la licencia PostgreSQL. Dentro de sus características principales se encuentra que es multisistema, por tanto, PostgreSQL puede ser instalado en Microsoft Windows, GNU/Linux, MacOS, BSD y muchos otros sistemas operativos. Además de ser escalable y puede manejar bases de datos enormes, de más de 100 Terabytes y funciona bajo licencia libre.

PlpgSQL

PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL. Permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones, dando mucho más control automático que las sentencias SQL básicas. Desde PL/pgSQL se pueden realizar cálculos complejos y crear nuevos tipos de

datos de usuario. Como un verdadero lenguaje de programación, dispone de estructuras de control repetitivas y condicionales, además de la posibilidad de creación de funciones que pueden ser llamadas en sentencias SQL normales o ejecutadas en eventos de tipo disparador (trigger).

PgAdmin III

PgAdmin 3 es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados (*EnterpriseDB Postgres Plus Advanced* Server y *Greenplum Database*). Incluye interfaz administrativa gráfica, herramienta de consulta SQL (con un EXPLAIN gráfico), editor de código procedural, agente de planificación SQL/shell/batch y administración de Slony-I. Este se diseña para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. La interface gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris.

Visual Studio Code versión 1.33.1

Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto.

Visual Paradigm 8.0

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML. Es ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de Sistemas que tienen como objetivo la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Visual Paradigm también ofrece:

- ✓ Navegación intuitiva entre la escritura del código y su visualización.
- ✓ Potente generador de informes en formato PDF/HTML.
- ✓ Ambiente visualmente superior de modelado.
- ✓ Sincronización de código fuente en tiempo real.

Esta herramienta CASE es multiplataforma y su licencia se encuentra disponible de forma gratuita solo con fines académicos y no comerciales. Además, brinda un completo conjunto de herramientas de equipos de desarrollo de software necesario para la captura de requisitos, software de planificación y el modelado de datos (Software, 2013).

Conclusiones del capítulo

Después de realizar un análisis de las soluciones existentes se decide desarrollar un sistema web de búsqueda de productos que tenga como objetivo principal la búsqueda de productos en tiendas, el cual sea capaz de dar respuesta a través de criterios de búsquedas simple o avanzada, que permita al usuario dar opiniones acerca de los productos y además que sea capaz de actuar como sitio para el debate entre los clientes a través de comentarios y respuestas. Además de contar con la arquitectura de microservicio necesaria para agregarlas a la plataforma D'Prisa. Del estudio de las herramientas y tecnologías a emplear en la solución se define AUP-UCI como metodología de desarrollo de software, Visual Paradigm 8.0 como herramienta CASE y UML como lenguaje de modelado. Los lenguajes de programación a utilizar serán: Javascript , TypeScript, HTML 5. El frameworks a emplear será: Angular en su versión 7.3.3.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo se evidencian las principales características de la solución propuesta. Se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que el sistema web funcione correctamente. Además, se construyen los artefactos correspondientes al análisis y diseño acorde con la metodología seleccionada en el capítulo anterior.

2.1 Modelo de Dominio

El modelo de dominio, es el punto de partida para lograr un mejor diseño del sistema a desarrollar y una comprensión más clara del problema a resolver. En él se muestran las clases conceptuales más significativas asociadas al dominio del problema. Está considerado como parte del modelo de negocio, pues se centra en una parte del mismo relacionada con el ambiente del proyecto (Sosa, 2013).

En la siguiente figura se muestra el modelo de dominio asociado a la descripción del problema:

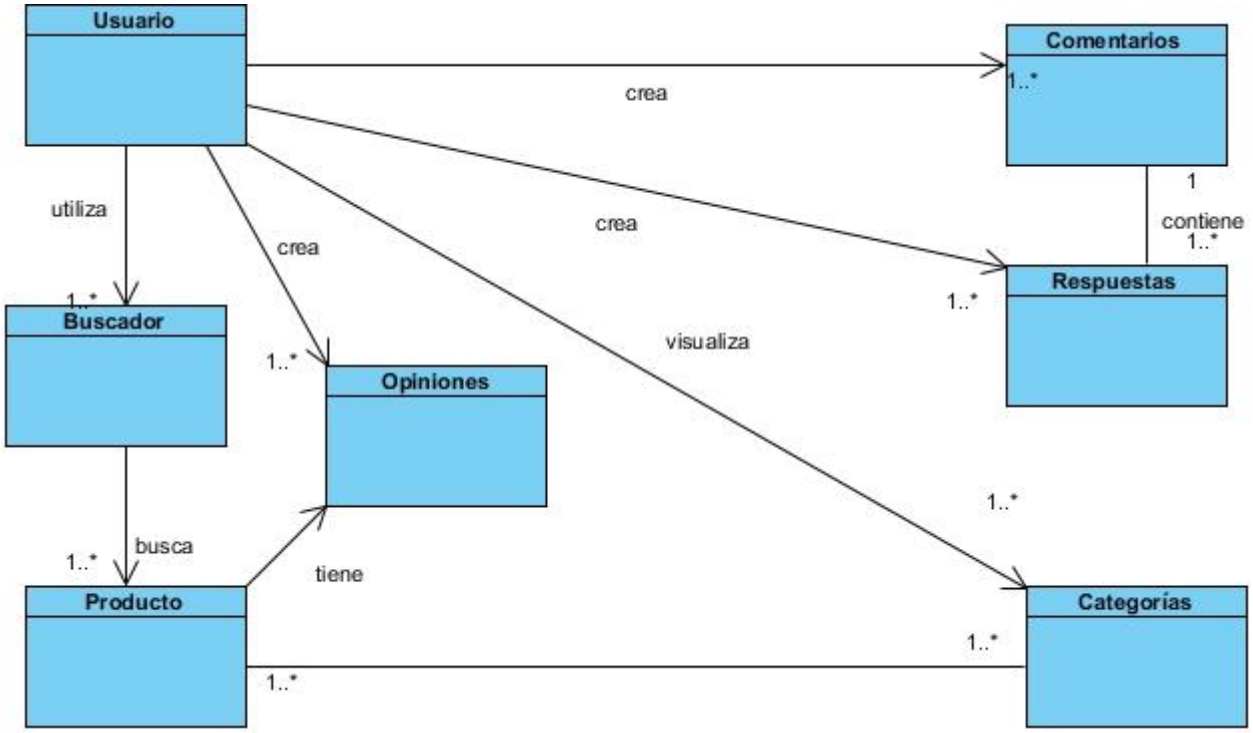


Figura 1.Modelo de Dominio

Definición de las clases del modelo de dominio

- **Buscador:** Herramienta que se utiliza para la búsqueda del usuario.
- **Producto:** Resultado que se muestra de una búsqueda.
- **Usuario:** Persona que interactúa con el sitio.
- **Categorías:** Permite dividir a los productos y ayuda a la búsqueda del usuario.
- **Comentarios:** Permite que el usuario comente de algún tema de su interés.
- **Respuestas:** Permite que el usuario responda a un comentario de su interés.
- **Opiniones:** Permite que el usuario de una opinión de un producto buscado.

2.2 Requisitos del Sistema

La especificación de requisitos es el proceso de desarrollo de software donde se analizan las necesidades de los usuarios y del sistema. Un correcto levantamiento de requisitos permite posteriormente la construcción de un software de calidad (Pressman, 2011). Los mismos se clasifican en requisitos funcionales y no funcionales.

Requisitos Funcionales

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Además, se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los requisitos funcionales definen las funciones que el sistema será capaz de realizar (Somerville, 2011). A continuación, se describen los requisitos funcionales que debe cumplir el Sistema de Búsqueda de productos en tiendas empleando arquitectura de microservicio sobre la plataforma D'Prisa.

RF1: Mostrar Categorías. Permite mostrar las categorías desde la vista principal.
RF2: Realizar búsqueda simple. Permite al usuario realizar una búsqueda del producto a través del nombre.
RF3: Realizar búsqueda avanzada. Permite al usuario realizar una búsqueda avanzada a través de diferentes filtros.
RF3.1: Por provincia. Permite al usuario realizar una búsqueda avanzada filtrando por provincia.
RF3.2: Por municipio. Permite al usuario realizar una búsqueda avanzada filtrando por municipio.
RF3.3: Por precios. Permite al usuario realizar una búsqueda avanzada filtrando por el precio.

RF3.4: Por categorías. Permite al usuario realizar una búsqueda avanzada filtrando por la categoría a la que pertenece el producto.

RF4: Mostrar resultado. Permite mostrar los resultados de la búsqueda del usuario.

RF5: Mostrar información del producto. Muestra al usuario toda la información referente al producto buscado

RF6: Mostrar opiniones de un producto. Permite ver al usuario todas las opiniones que existan acerca de un producto por parte de otros usuarios.

RF7: Crear opinión. Permite al usuario crear una opinión acerca de un producto.

RF8: Mostrar comentarios. Permite ver al usuario todos los comentarios creados por otros usuarios.

RF9: Crear comentario. Permite al usuario crear un comentario de su interés.

RF10: Mostrar respuestas. Permite ver al usuario todas las respuestas que existan de un comentario.

RF11: Crear respuesta. Permite al usuario crear una respuesta a un comentario dado.

RF12: Mostrar ayuda. Brinda al usuario una interfaz para la familiarización con el sitio a través de un video tutorial, así como datos de los desarrolladores y la empresa.

RF13: Buscar comentarios. Permite al usuario buscar un comentario creado a través del nombre del usuario del cliente.

Requisitos No Funcionales

Los requisitos no funcionales (NF) se refieren a funciones específicas que proporciona el sistema. Son restricciones de los servicios o funciones ofrecidas por el sistema (fiabilidad, tiempo de respuestas, capacidad de almacenamiento). Generalmente se aplican al sistema en su totalidad. Surgen de las necesidades del usuario (restricciones de presupuesto, políticas de la organización, necesidad de interoperabilidad). En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Están vinculados a requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser (Pressman, 2011).

A continuación, se describen los requisitos no funcionales que debe cumplir sistema de búsqueda de productos en tiendas.

Requisitos de software

RNF 1 Requisitos de software para la PC del cliente.

Para el uso del sistema la PC cliente debe contar con un navegador web Firefox versión 65.0.1 o navegador Google Chrome en su versión 72.0. En cualquiera de los dos casos puede usarse una versión superior.

RNF 2 Servidor para instalar la aplicación.

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos de software:

- SO: GNU/Linux Coloso.
- Usuarios con privilegios de administración del SO.

RNF 3 Servidor para instalar la base de datos.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos de software (puede ser el mismo donde estará la aplicación):

- SO: GNU/Linux Coloso.
- PostgreSQL versión 9.4.0 o superior.
- PgAdmin III o algún administrador para PostgreSQL.
- Usuarios con privilegios de administración del SO.

Requisitos de hardware.

RNF 4 PC Cliente

La PC cliente debe cumplir con los siguientes requisitos de hardware:

- Ordenador Dual Core o superior, con 1.7 GHZ de velocidad del microprocesador.
- Memoria RAM mínimo 1GB.

RNF 5 Servidor de aplicaciones

La PC donde se instalará el servidor de la aplicación debe cumplir con los siguientes requisitos de hardware:

- Ordenador i3 o superior, con 2.3 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 4GB.

- Disco Duro con capacidad de 100 GB de capacidad.

RNF 6 Servidor de Base de Datos

La PC donde se instalará la base de datos del sistema debe cumplir con los siguientes requisitos de hardware:

- Ordenador i3 o superior, con 2.3GHz de velocidad de microprocesador.
- Memoria RAM mínimo 4 GB.
- Disco Duro con capacidad de 100GB de capacidad.

Restricciones de diseño

RNF 7 Lenguaje para el desarrollo del sistema del lado del servidor.

El sistema deberá ser implementado en el lenguaje de programación Nodejs de Express, el cual trabaja con una arquitectura basada en microservicios a través de APIs.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF 8 Documentación de usuarios

Se entregará a los clientes un manual de usuario que guía paso a paso las acciones a seguir para trabajar con el sistema. Este manual estará escrito en español.

Requisitos de Interfaz

RNF 9 Interfaz

La interfaz del sistema contendrá los datos de forma estructurada, permitiendo la interpretación correcta de la información. Todos los textos y mensajes en pantalla serán mostrados en idioma español.

2.3 Patrones de Casos de Uso

Los patrones de casos de uso son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen como debería ser estructurados y organizados los casos de uso. Son patrones que capturan mejores prácticas para modelar casos de uso. A continuación, se explican los patrones de caso de uso utilizados.

Extensión: Consiste en dos casos de uso y una relación extendida entre ellos. Puede ser instalado en sí mismo, así como extendido en el caso de uso base. El referente puede ser concreto o abstracto. Este patrón se aplica cuando un flujo puede extender el flujo de otro caso de uso, así como ser realizado en sí mismo. El patrón extensión concreta es evidenciado en el caso de uso Mostrar información del producto ya que es una extensión del caso de uso Realizar búsqueda (Caso base), le incorpora funcionalidad al caso base sin alterar ni modificar a este, no siendo indispensable para su completamiento.

Inclusión: Se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instalado en sí mismo. El caso de uso base puede ser concreto o abstracto. El patrón inclusión es evidenciado en el caso de uso Mostrar resultado de la búsqueda ya que es una inclusión del caso de uso Realizar búsqueda (Caso base), le incorpora funcionalidad al caso base, pero si modifica a este, siendo indispensable para su completamiento.

Actores Múltiples: A un Caso de Uso ingresan más de dos actores y estos tienen un rol común. El patrón actores múltiples es evidenciado en los casos de uso “Mostrar Ayuda”, “Mostrar Categorías” y “Realizar Búsqueda” debido a que dos actores Usuario y Usuario Identificado tienen un rol en común.

2.4 Casos de uso del sistema propuesto

Los casos de uso se crean para refinar un conjunto de requisitos de acuerdo con una función o tarea. En lugar de la tradicional lista de requisitos que quizás no trate de forma directa el uso de la solución, los casos de uso reúnen requisitos comunes basados en el tipo de función u objetivo. Los casos de uso definen qué harán los usuarios o funciones en la solución y un proceso empresarial define cómo realizarán esas funciones. (Center, 2013).

2.4.1 Diagrama de casos de uso del Sistema

A continuación, se muestra el diagrama de casos de usos del sistema de búsqueda de productos en tiendas.

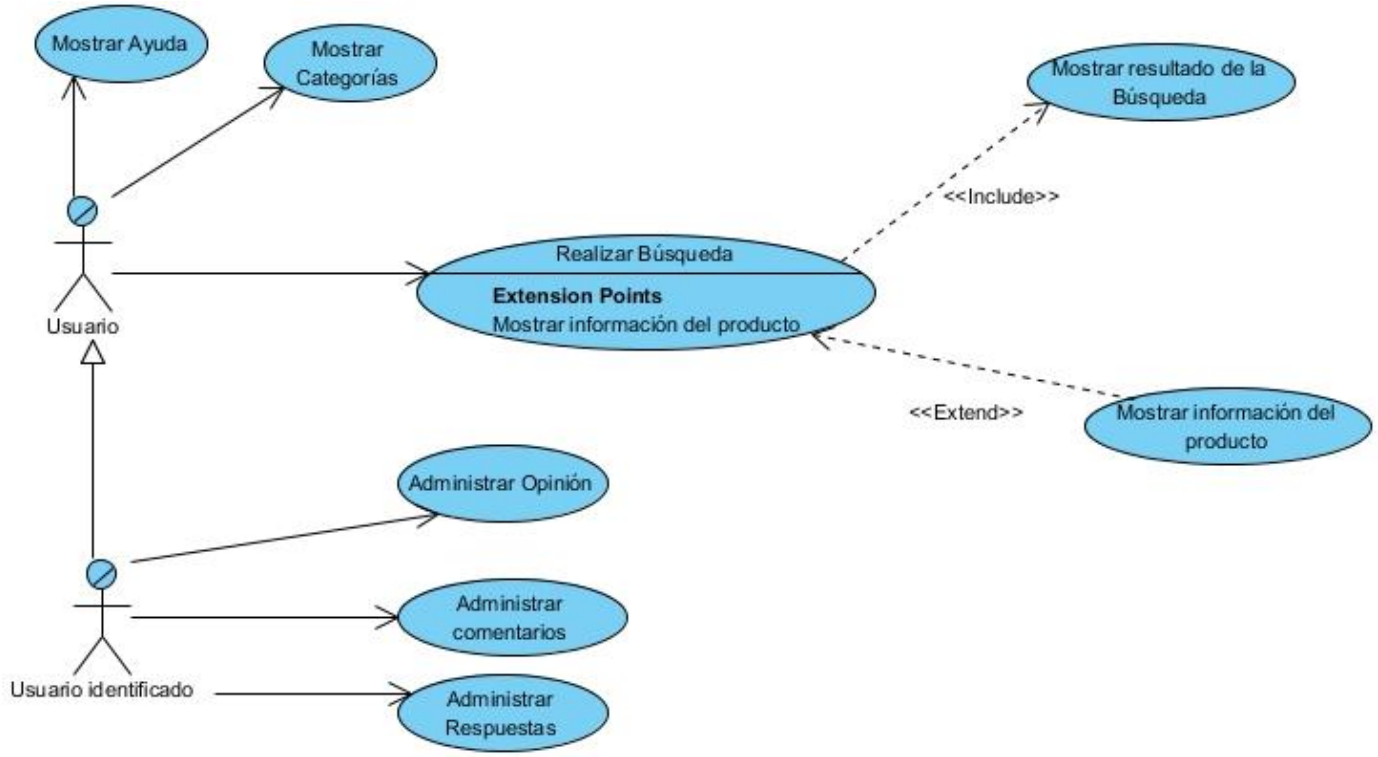


Figura 2.Diagrama de Casos de Uso del Sistema

2.4.2 Descripción textual de los casos de uso y del actor del sistema

Los actores del sistema son aquellos individuos u otros sistemas externos que interactúan con el sistema (Svitla Systems, 2016). Para el desarrollo de estos diagramas se han identificado los siguientes actores.

Tabla 2. Tabla de Descripción de los actores de los Casos de Usos.

Actor	Descripción
Usuario	Representa una generalización de todos los usuarios que interactúan con el sistema.
Usuario identificado	Representa los todos los usuarios que se identificados en el sistema a través de los nombres de usuarios que tienen en sus comentarios, respuestas y opiniones.

CU Mostrar Ayuda: Muestra al usuario una interfaz brindándole información acerca de cómo hacer un buen uso de las diferentes vistas del sitio, a través de un video tutorial.

CU Mostrar Categorías: Muestra al usuario 9 categorías por las cuales se dividen los productos en las tiendas permitiendo al usuario la búsqueda de los mismo a través de estas.

CU Realizar Búsqueda: Permite al usuario buscar un producto, a través del nombre o por filtros de Provincia, Municipio, Precio y Categoría.

CU Mostrar Resultado de la Búsqueda: Muestra al usuario un listado con el resultado de la búsqueda realizada.

CU Mostrar información del producto: Muestra al usuario la información de producto buscado: nombre, precio, disponibilidad, nombre de la tienda, dirección de la tienda, categoría a la que pertenece el producto, descripción de la categoría, provincia, municipio, tipo de entidad a la que pertenece el producto, así como el nombre de la misma.

CU Administrar Opinión: Permite al usuario ver las opiniones que hay de un producto, así como insertar una él.

CU Administrar Comentarios: Permite al usuario ver los comentarios que existen e insertar uno él.

CU Administrar Respuestas: Permite al usuario ver las respuestas que hay de un comentario y crear una él.

Tabla 3. Referencia Caso de Uso con Requisitos Funcionales.

Casos de uso	**Referencia RF**
Mostrar categoría	RF1
Mostrar Ayuda	RF 12
Realizar Búsqueda	RF2, RF3, RF3.1, RF3.2, RF3.3, RF3.4
Mostrar Resultado de la Búsqueda	RF4
Mostrar información del producto	RF5
Administrar Opinión	RF6, RF7
Administrar Comentarios	RF8, RF9, RF13
Administrar Respuestas	RF10, RF11

2.4.3 Especificación de casos de uso del sistema propuesto

Las siguientes tablas que se muestran corresponden a la descripción de los casos de usos “Realizar Búsqueda”, “Administrar Comentarios” y “Mostrar Categorías”.

Tabla 4.Descripción del Caso de Uso "Realizar Búsqueda".

Nombre	**Caso de Uso: Realizar Búsqueda**
Objetivo	Permitir al usuario realizar la búsqueda de productos.
Actor	Usuario
Resumen	El caso de uso inicia cuando el Usuario selecciona la opción “Buscar Producto” en la vista principal, el sistema lo remite a la vista de la opción en la cual podrá realizar una búsqueda del producto por el Nombre, filtros de Provincia, Municipio, Precio, Categoría o la combinación de todos. Una vez el Usuario haber definido porque criterio realizar su búsqueda hace click en el botón “Buscar” y así termina el caso de uso.
Complejidad	Alta
Prioridad	Alta
Referencias	RF1, RF2, RF3, RF3.1, RF3.2, RF3.3, RF3.4
Precondiciones	-
Poscondiciones	Se muestra el resultado de la búsqueda.
Flujo de eventos	
Flujo básico	
Actor	**Sistema**
	1. Muestra una interfaz principal con las opciones: ➢ Principal. ➢ Comentarios. ➢ Buscar Producto. ➢ Ayuda.
2. El usuario selecciona la opción “Buscar	3. El sistema muestra la interfaz de búsqueda del

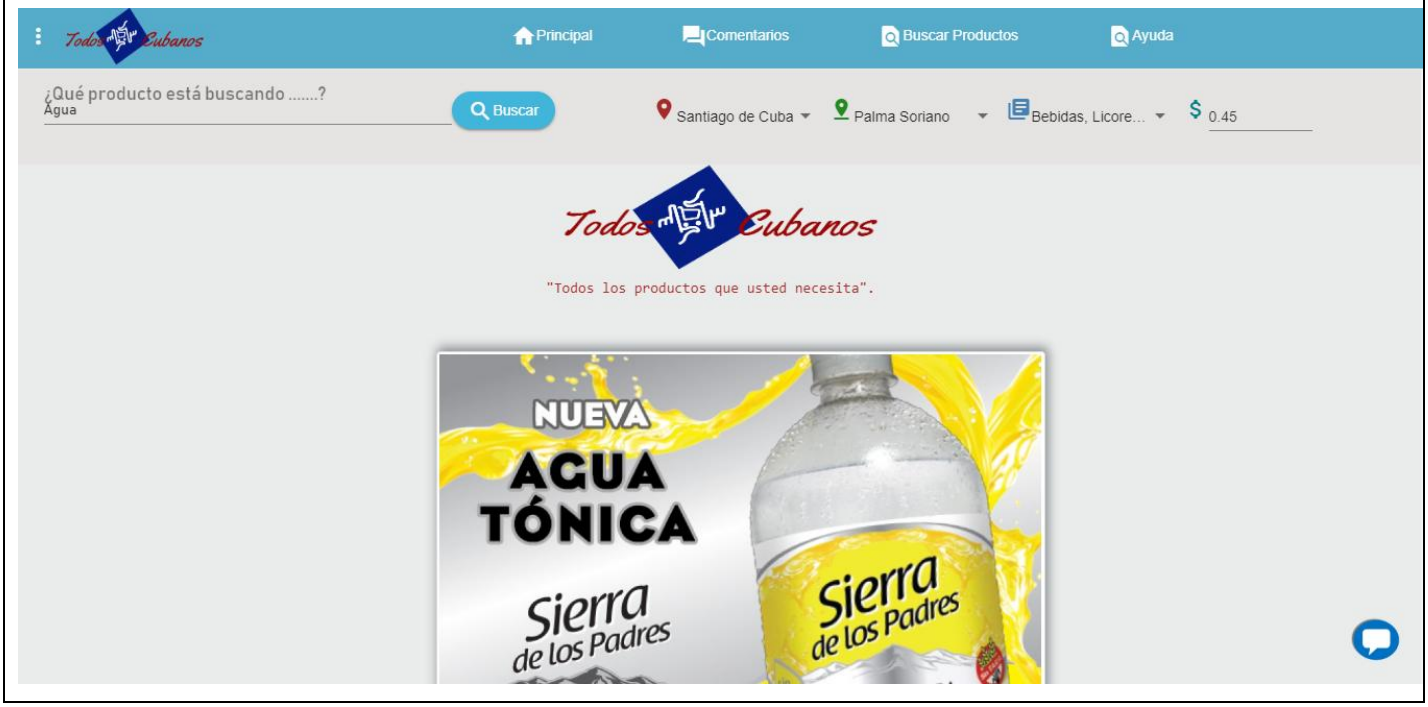
Producto” en la interfaz principal del sistema.	producto.
4. El usuario define el criterio por el cual realizará su búsqueda ya sea por nombre del producto o por los filtros de provincia, municipio, precio y categoría.	5. El sistema permite escribir el nombre del producto, seleccionar la provincia, municipio, precio o la categoría del mismo.
6. El usuario da clic en el botón “Buscar”.	7. El sistema muestra el resultado de la búsqueda del usuario.
Prototipo de la Interfaz: 	

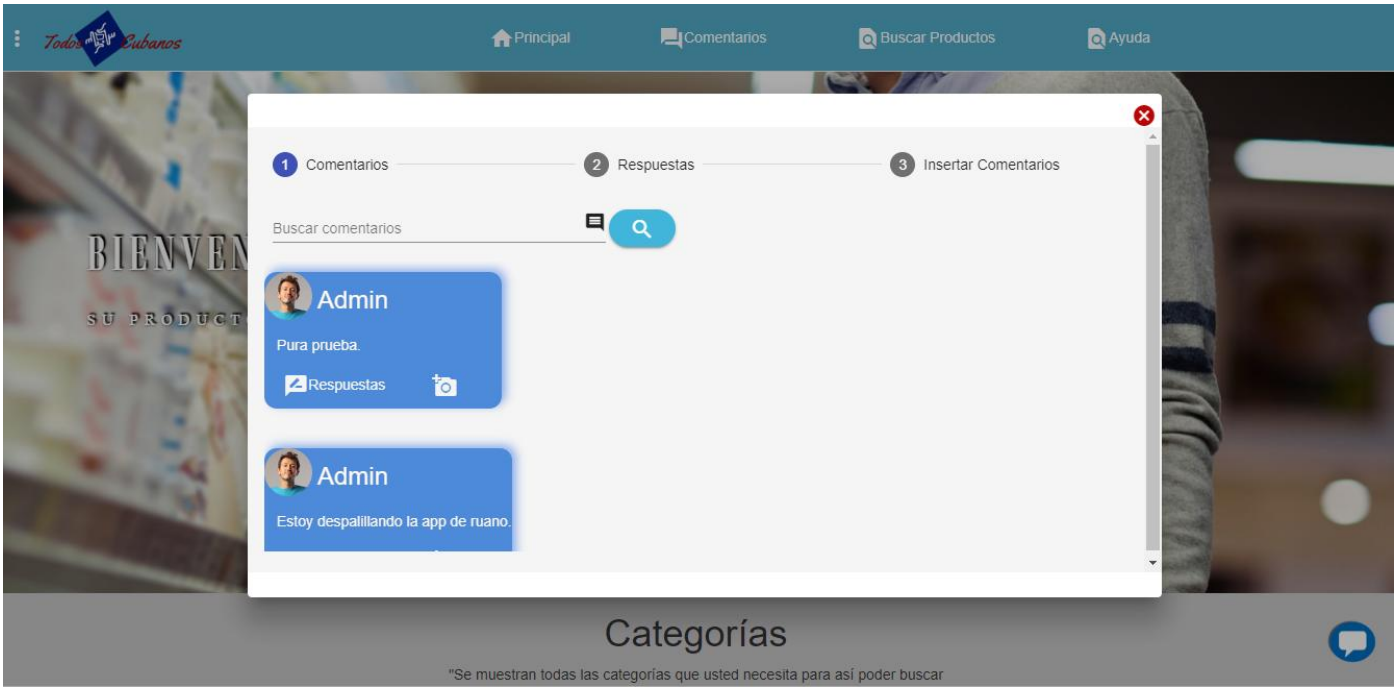
Tabla 5.Descripción del Caso de Uso "Administrar Comentario".

Nombre	**Caso de Uso: Administrar Comentarios.**
Objetivo	Mostrar e insertar comentario en el sistema.

Actor	Usuario
Resumen	El caso de uso inicia cuando el Usuario selecciona la opción “Comentarios” en la vista principal, el sistema levanta una ventana mostrando tres opciones “Comentarios”, “Respuestas” y “Insertar Comentarios”, el usuario selecciona en la opción “Comentarios” y se muestran todos los comentarios que se han realizado en el sistema por parte de otros usuarios, luego el mismo selecciona la opción “Insertar Comentarios”, y se muestra un formulario para que el usuario escriba su nombre y su comentario además de dos botones con la opción de Comentar y Cancelar. Una vez que el usuario complete los campos del formulario selecciona la opción Comentar y aparece un cartel de aviso de que el comentario se ha insertado correctamente el mismo posee un botón aceptar el cual el usuario da click y así termina el caso de uso.
Complejidad	Media
Prioridad	Media
Referencias	RF8, RF9
Precondiciones	El Usuario tiene que seleccionar la opción “Comentarios”, el usuario tiene que llenar el formulario para insertar un comentario.
Poscondiciones	Se muestra un mensaje de alerta de que el comentario se insertó correctamente y luego se muestra el comentario.
Flujo de eventos	
Flujo básico. Insertar Comentario	
Actor	**Sistema**
	1. Muestra una interfaz principal con las opciones: ➢ Principal. ➢ Comentarios.

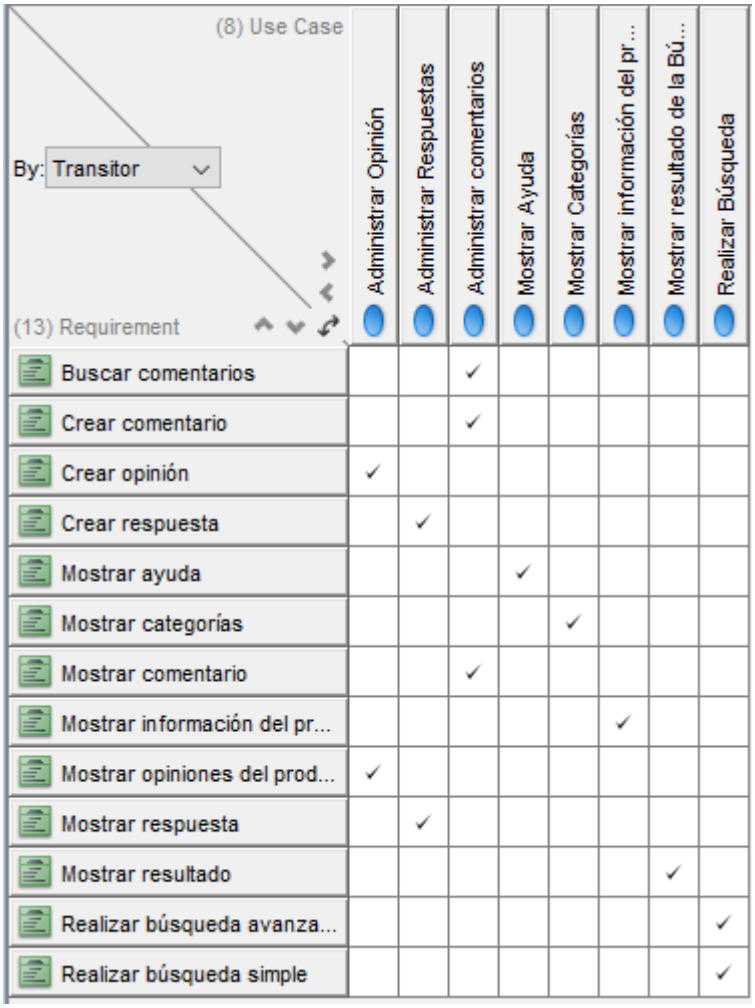
	➢ Buscar Producto. ➢ Ayuda.
2. El usuario selecciona la opción “Comentarios” en la interfaz principal del sistema.	3. El sistema muestra la interfaz de comentarios, con tres opciones: ➢ Comentarios. ➢ Respuestas. ➢ Insertar comentarios. ➢ Buscar comentario. Y todos los comentarios que se han realizado en el sitio.
4. El usuario selecciona la opción “Insertar Comentarios” para unirse al debate.	5. El sistema muestra un formulario con los campos de nombre de usuario y comentario además de dos botones Comentar y Cancelar.
6. El usuario llena los campos del formulario y da clic en el botón Comentar.	7. El sistema valida los datos introducidos por el usuario.
	8. El sistema muestra un mensaje de alerta informando que su comentario ha sido insertado correctamente y un botón Aceptar.
9. El usuario da clic en el botón Aceptar.	10. El sistema lo envía a la opción “Comentarios” y muestra el nuevo comentario.
Flujo Alterno. Insertar comentario	
6.1 El usuario presiona el botón Cancelar.	6.2 El sistema cierra el formulario y remite al usuario a la página anterior.
	7.1 Muestra en color rojo el campo que ha quedado vacío o que se ha llenado con datos incorrectos.

Prototipo de la Interfaz:



2.5 Matriz de Trazabilidad

La matriz de trazabilidad de requisitos actúa como un apoyo para mantener al día el estado de cada uno de los requisitos y especificaciones del proyecto. Este estado puede cambiar con el tiempo y en la matriz de trazabilidad se lleva el registro de estos cambios desde su inicio hasta su consecución. La matriz de trazabilidad mantiene dos versiones generales, un antes y un después, lo que es de gran utilidad. (MDAP , 2019). A conticnuación se muestran las matrices de trazabilidad creadas al sistema:



(13) Requirement \ (8) Use Case (By: Transitor)	Administrar Opinión	Administrar Respuestas	Administrar comentarios	Mostrar Ayuda	Mostrar Categorías	Mostrar información del pr...	Mostrar resultado de la Bú...	Realizar Búsqueda
Buscar comentarios			✓					
Crear comentario			✓					
Crear opinión	✓							
Crear respuesta		✓						
Mostrar ayuda				✓				
Mostrar categorías					✓			
Mostrar comentario			✓					
Mostrar información del pr...						✓		
Mostrar opiniones del prod...	✓							
Mostrar respuesta		✓						
Mostrar resultado							✓	
Realizar búsqueda avanza...								✓
Realizar búsqueda simple								✓

Figura 3. Matriz de Trazabilidad de Requisitos Funcionales y Casos de Uso.

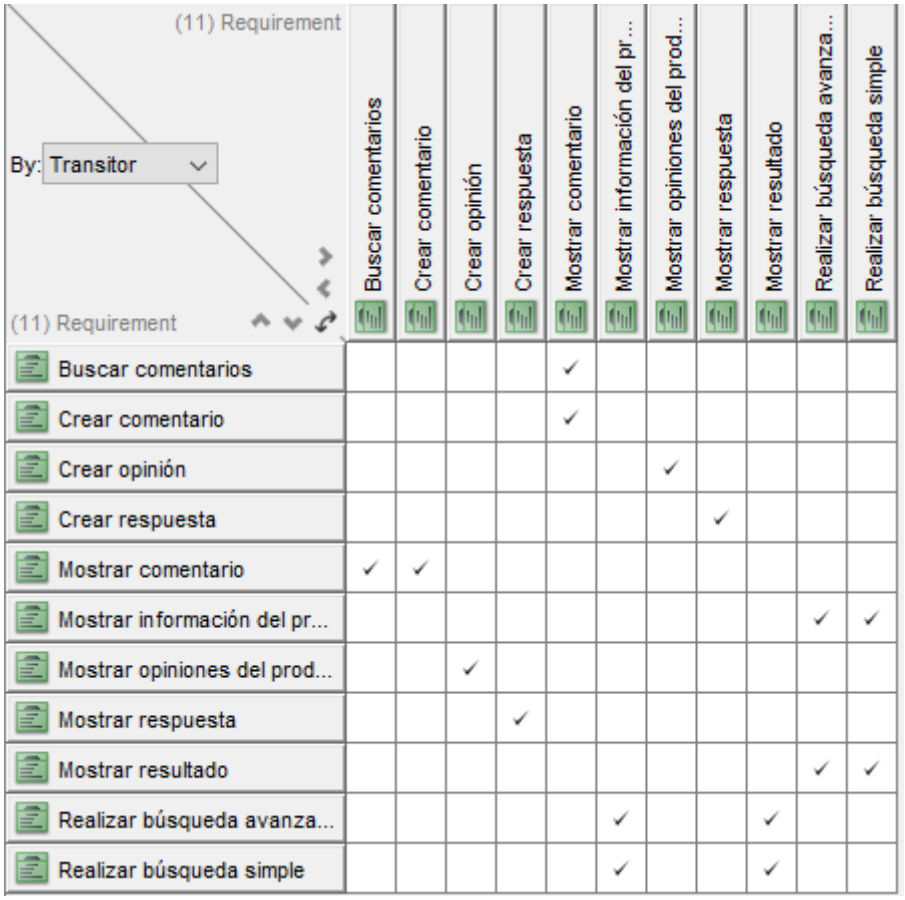


Figura 4.Matriz de Requisitos Funcionales y Requisitos Funcionales.

2.6 Patrones de arquitectura

La arquitectura de software se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para la guía en el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado.

Para dar a conocer la arquitectura del sistema lo primero que se debe comprender es que cada microservicio es un sistema autónomo, por lo que funciona en sí mismo como una aplicación autónoma, exponiendo, independientemente de la arquitectura de la plataforma a la que se integre su propia arquitectura, la cual se describe a continuación:

En el caso de los *microfrontend* sigue el patrón de arquitectura MVC (Modelo Vista Controlador) que permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control. Este patrón se ve usualmente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes (Larman, 2000).

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

En la siguiente figura se muestra un ejemplo de la estructura del patrón arquitectónico MVC (Modelo Vista Controlador):

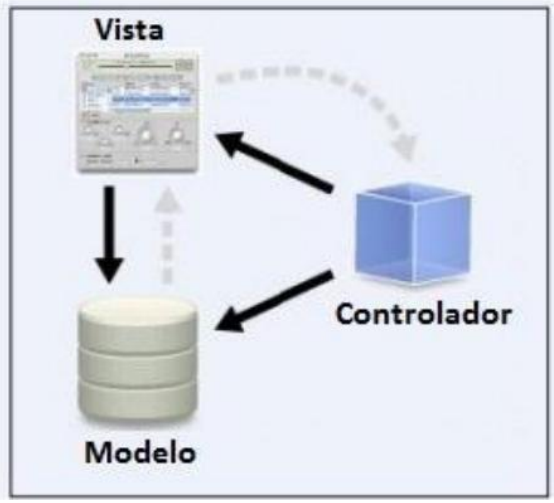


Figura 5.Patrón de Arquitectura MVC (Modelo Vista Controlador).

La principal ventaja de utilizar esta arquitectura es que existe una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, sin alterar la lógica de negocio. La separación de capas es fundamental para el desarrollo

de arquitecturas consistentes, reutilizables y mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos. Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- Agregar nuevas vistas.
- Modificar los objetos de negocios para migrar a otra tecnología.
- Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas.

Para el desarrollo de este sistema se tiene la arquitectura de microservicios, conocido por las siglas MSA (Micro Services Architecture), esta es un método de desarrollo de aplicaciones o sistemas de software, compuesto por una colección de servicios autónomos y pequeños, cada microservicios es independiente entre sí y cada uno se encarga de implementar una funcionalidad completa del negocio, la comunicación entre estos se realiza a través de API´s.

Como toda arquitectura posee una serie de ventajas y desventajas, a continuación de exponen las principales de cada una:

Ventajas

- **Implementaciones independientes**. Se puede implementar, revertir o poner al día un servicio si algo va mal sin volver a implementar toda la aplicación. Las correcciones de errores y las publicaciones de características son más fáciles de administrar y entrañan menos riesgo.
- **Desarrollo independiente**. Un solo equipo de desarrollo puede compilar, probar e implementar un servicio. El resultado es la innovación continua y un ritmo más rápido de publicación.
- **Aislamiento de errores**. Si un servicio deja de funcionar, no es necesario paralizar toda la aplicación.
- **Pilas de tecnología mixta**. Los microservicios permiten el uso de diferentes tecnologías y lenguajes.
- **Reutilización de código.** El desarrollador puede aprovechar las funcionalidades que ya han sido desarrolladas por terceros—no necesita reinventar la rueda, simplemente utilizar lo que ya existe y funciona.
- **Facilidad al desplegar.** Los microservicios pueden desplegarse según sea necesario, por lo que funcionan bien dentro de metodologías ágiles.

Desventajas

- **Complejidad**. Una aplicación de microservicios tiene más partes en movimiento que la aplicación monolítica equivalente. Cada servicio es más sencillo, pero el sistema como un todo es más complejo.
- **Desarrollo y pruebas**. El desarrollo con dependencias de servicios requiere un enfoque diferente. Las herramientas existentes no están necesariamente diseñadas para trabajar con dependencias de servicios.
- **Falta de gobernanza**. El enfoque descentralizado para la generación de microservicios tiene ventajas, pero también puede causar problemas. Puede acabar con tantos lenguajes y marcos de trabajo diferentes que la aplicación puede ser difícil de mantener.
- **Congestión y latencia de red**. El uso de muchos servicios pequeños y detallados puede dar lugar a más comunicación interservicios. Además, si la cadena de dependencias del servicio se hace demasiado larga (el servicio A llama a B, que llama a C...), la latencia adicional puede constituir un problema.
- **Integridad de datos**. Cada microservicio es responsable de la conservación de sus propios datos. Como consecuencia, la coherencia de los datos puede suponer un problema.

A continuación, se representa la comunicación entre el *microfrontend* y el microservicio del sistema *web* y a su vez la que posee con los otros microservicios contenidos en la Plataforma D´Prisa:

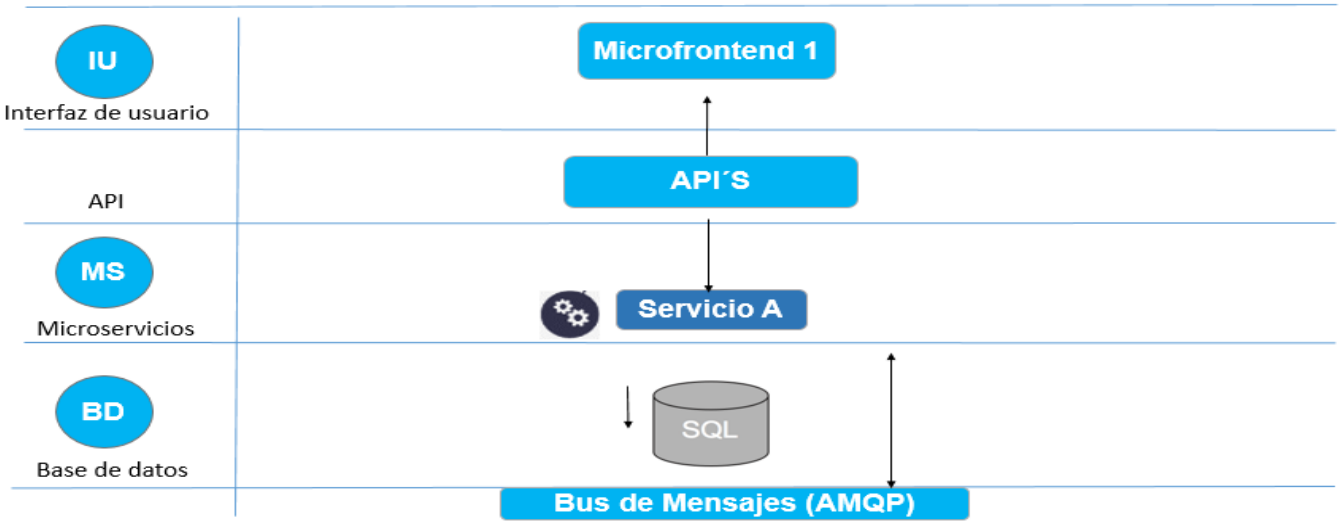


Figura 6. Arquitectura de microservicios de la Plataforma D´Prisa.

De la figura anterior se interpreta que el sistema *web* en la parte de la interfaz del usuario contendrá el microfrontend el cual se comunica a través de petición HTTP por los métodos GET y POST con las API´s que la misma a su vez salen del microservicio en el que están contenidas con la misma petición. El microservicio además mantiene una conexión a la base de datos a través de la ruta localhost con puerto 5432 y las consultas a la misma se realizan a través de SQL. Dentro de la Plataforma el sistema *web* como microservicio se comunicará con otros como Sistema *web* de proveedor de productos en tiendas y el Autenticar Usuario a través del Bus de Mensajes (AMQP).

2.7 Diagramas de clases de diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de una aplicación. Además, muestra las definiciones de entidades de software más que conceptos del mundo real y contiene información acerca de las clases, asociaciones, atributos, interfaces, métodos y relaciones (Unad, 2006).

2.7.1 Definición de los elementos de los diagramas de clases del diseño

Como el sistema propuesto constituye una aplicación web, en los diagramas de clases de diseño de cada uno de los casos de uso “Realizar Búsqueda” y “Administrar Comentarios”; se emplearon clases UML estereotipadas “Server Page”, “Client Page” y “Form”, utilizadas para representar código servidor, páginas clientes y formularios.

Tabla 6. Tabla de Descripción de los Estereotipos Web.

Estereotipos web para las clases del diseño	
ESTEREOTIPO	**DESCRIPCIÓN**
	**Server Page**: Constituye la página web que interactúa con las bases de datos, lógica del negocio, se encarga de la construcción (*build*) del resultado HTML, representado con las páginas clientes (*client page*).

	**Client Page**: Representa las páginas en formato HTML, contiene los formularios. Puede contener scripts que son interpretados por el navegador.
	**Form**: Constituye la colección de campos de entrada que forman parte de una página cliente. Los formularios envían sus datos al código servidor para ser procesados los pedidos (*submit*).

2.7.2 Definición de las relaciones entre los elementos del diseño

Seguidamente se explican las relaciones que se establecen entre los elementos de los diagramas de clases del diseño. En el caso de la propuesta de solución se emplearon: *link, build, submit.*

Tabla 7. Descripción de las Relaciones del Diagrama de Clase del Diseño.

Relaciones de los elementos del diseño	
RELACIÓN	**DESCRIPCIÓN**
link	Representa el tipo de relación que se establece desde una “client page” hacia otra “client page” o “server page”.
build	Es el tipo de relación que se efectúa entre una “server page” y una “client page”. Una “server page” puede crear varias “client page”, pero una “client page” sólo puede ser creada por una sola “server page”.
submit	Este tipo de relación se desarrolla siempre desde un “form” contenido en una “client page” hacia una “server page”, en donde son procesados los datos enviados.

Se realizó un diagrama de clases de diseño por cada caso de uso, en las siguientes figuras se muestran los diagramas de clases del diseño pertenecientes a los casos de uso “Realizar Búsqueda”, “Administrar Comentarios.”

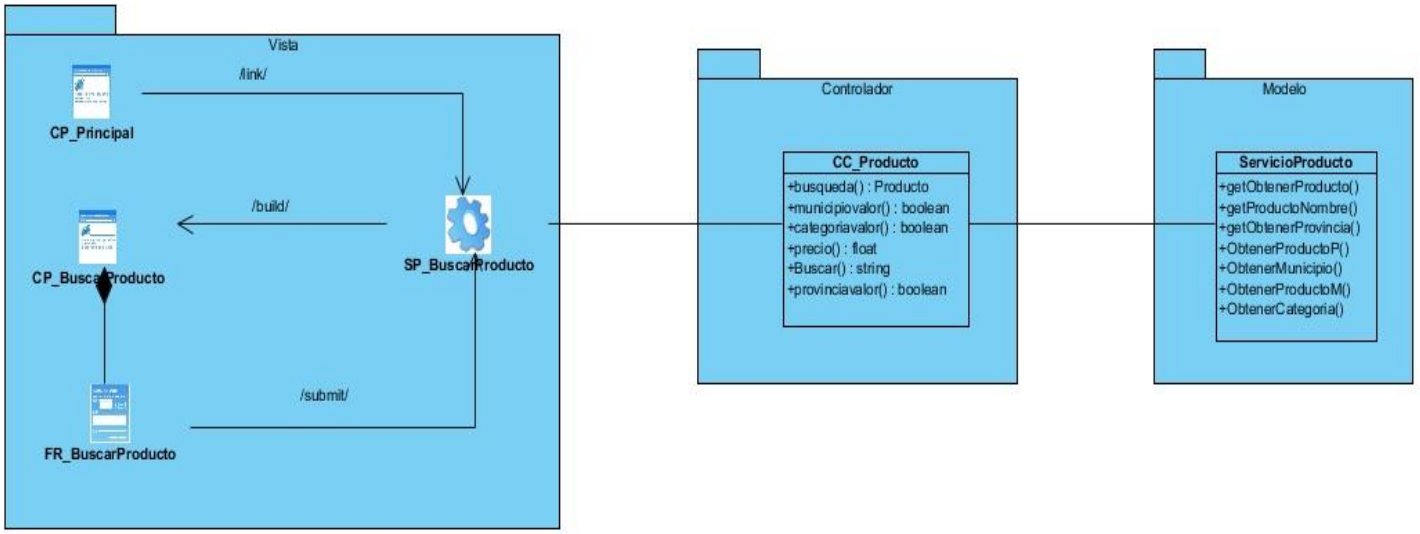


Figura 7.Diagrama de clase de diseño del caso de uso "Realizar Búsqueda".

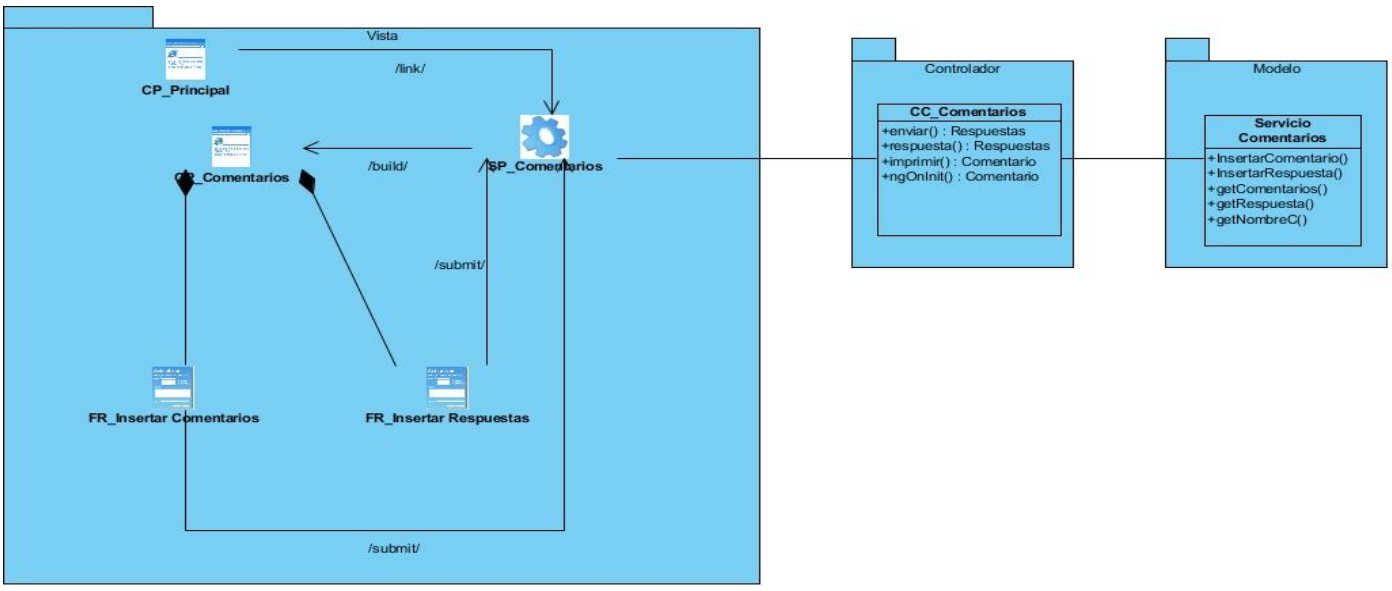


Figura 8.Diagrama de clase de diseño del caso de uso "Administrar Comentarios".

2.7.3 Descripción de las clases del Diagrama de clases de diseño

En los diagramas de clases de diseño de las figuras, se representaron las principales clases que se utilizaron para la realización del caso de uso “Realizar Búsqueda” y “Administrar Comentario”, estas quedaron representadas en una arquitectura MVC ya que es la utilizada por el framework Angular 7. A continuación, se describen las clases del Diagrama de clases de diseño:

En la Vista se encuentran las clases:

- **CP_Principal:** Representa la página principal del sistema, que el mismo muestra las opciones principales; “Comentarios”, “Buscar Producto” y “Ayuda”. Además también muestra las categorías en las que se distribuyen los productos.
- **CP_Buscar Producto:** Representa la vista de la opción “Buscar Producto” que la misma muestra las opciones de búsqueda que tiene el usuario: por nombre de producto, provincia, municipio, categoría, precio.
- **SP_Buscar Producto:** Gestiona las acciones las acciones del lado del cliente y las llamadas a la clase controladora Producto.
- **FR_Buscar Producto**: Representa el formulario contenido en la vista “Buscar Producto” que permite al usuario definir el criterio de búsqueda: por nombre, provincia, municipio, categoría, precio o cualquier combinación de estos.
- **CP_Comentarios:** Representa la vista de la opción “Comentarios”, que la misma muestra las opciones de “Comentarios” que muestra todos los comentarios, “Respuesta” que contiene las respuestas de un comentario y el formulario para insertar, además de la opción “Insertar Comentarios” que contiene el formulario para dicha acción.
- **SP_Comentarios:** Gestiona las acciones las acciones del lado del cliente y las llamadas a la clase controladora Comentarios.
- **FR_Insertar** Comentarios: Representa el formulario contenido en la opción “Insertar Comentarios” para realizar dicha acción.

- **FR_Insertar_Respuesta**: Representa el formulario contenido en la opción “Respuesta” para insertar una respuesta.

En el Controlador se encuentran las clases:

- **CC_Producto**: Representa la clase encargada de responder las peticiones realizadas del lado del cliente a través de métodos tales como búsqueda (), provinciavalor (), municipiovalor (), categoriavalor (), precio () y Buscar ().

- **CC_Comentario**: Representa la clase encargada de responder las peticiones realizadas del lado del cliente a través de métodos tales como enviar (), imprimir (), respuesta () y ngOninit ().

En el Modelo se encuentran las clases:

- **Servicio Producto**: Es la clase que contiene las peticiones http tales como getObtenerProducto (), getObtenerProvincia (), getProductoNombre (), ObtenerProductoP (), ObtenerMunicipio (), ObtenerProductoM (), ObtenerCategoría () representa de manera general la información realizada por las clases.

- **Servicio Comentarios**: Es la clase que contiene las peticiones http tales como Insertar Comentarios (), Insertar Respuesta (), getComentarios (), getRespuesta () y getNombreC (); representa de manera general la información realizada por las clases.

2.8 Patrones de Diseño

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. Los patrones de arquitectura expresan un esquema organizativo estructural fundamental para sistemas de software.

2.8.1 Patrones GRASP

Describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Larman, 2003). A continuación, se detallan los patrones GRASP utilizados en la solución.

Creador: Soluciona el problema de ¿quién debería ser responsable de crear una nueva instancia? El patrón creador guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea

muy común en sistemas orientados a objetos. En la clase Comentarios se evidencia este patrón ya que es la encargada de la creación de la instancia de la clase Respuestas.

Experto: La responsabilidad de realizar una labor específica es otorgada a la clase que contiene la información necesaria para realizar dicha labor (Larman, 2003). En la propuesta de solución un objeto de **tipo Comentarios** es el responsable de validar sus metadatos y exponer dicha evaluación a los demás objetos, pues la clase Comentario es el experto en esa información.

Bajo Acoplamiento: El diseño se debe realizar de forma tal que la dependencia entre clases sea baja. De esta forma se tiene clases menos dependientes, logrando una reducción del impacto de los cambios y garantizando un mayor grado de reutilización (Larman, 2003). Este patrón se pone en práctica al asociar a cada controlador un solo servicio, disminuyendo de esta forma el acoplamiento.

Alta Cohesión: Los elementos deben tener responsabilidades altamente relacionadas y efectuar la menor cantidad de operaciones posibles. Al existir una alta cohesión se hace más fácil el mantenimiento, entendimiento y la reutilización (Larman, 2003). La utilización de este patrón se evidencia en los controladores, estos mantienen su código lo más simple posible manejando solo los eventos del sistema y delegando las demás tareas, como la implementación de la lógica de negocio a los servicios.

Conclusiones del capítulo

Al realizar el análisis se obtuvieron 9 requisitos no funcionales y 13 requisitos funcionales, estos últimos agrupados en 8 casos de uso del sistema utilizando los patrones de casos de uso Extensión e Inclusión concreta. Durante el diseño se describe la arquitectura a través de los patrones de diseño GRASP: Creador, Experto, Bajo Acoplamiento y Alta Cohesión, con el uso del patrón arquitectónico MVC.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA WEB PARA LA BÚSQUEDA DE PRODUCTOS EN TIENDAS EMPLEANDO UNA ARQUITECTURA DE MICROSERVICIO SOBRE LA PLATAFORMA D'PRISA.

El presente capítulo está enfocado a la implementación de la aplicación para dar solución a los requisitos especificados. Se elaboran los diagramas de componentes que forman parte del modelo de implementación. Se define el estándar de codificación a utilizar en la implementación de la solución y se realiza el diagrama de despliegue. Además, se realizan pruebas de software con el objetivo de descubrir y corregir errores.

3.1 Diagrama de Componentes

Un componente es un módulo de software que puede ser código fuente, código binario, un ejecutable, o una librería con una interfaz definida. Una interfaz establece las operaciones externas de un componente, las cuales determinan una parte del comportamiento del mismo. Además, se representan las dependencias entre componentes o entre un componente y la interfaz de otro, es decir uno de ellos usa los servicios o facilidades del otro. Estos diagramas pueden incluir paquetes que permiten organizar la construcción del sistema de información en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías, etc. (Cillero, 2019). A continuación se muestra el diagrama de componentes referente al sistema de búsqueda de productos en tiendas.

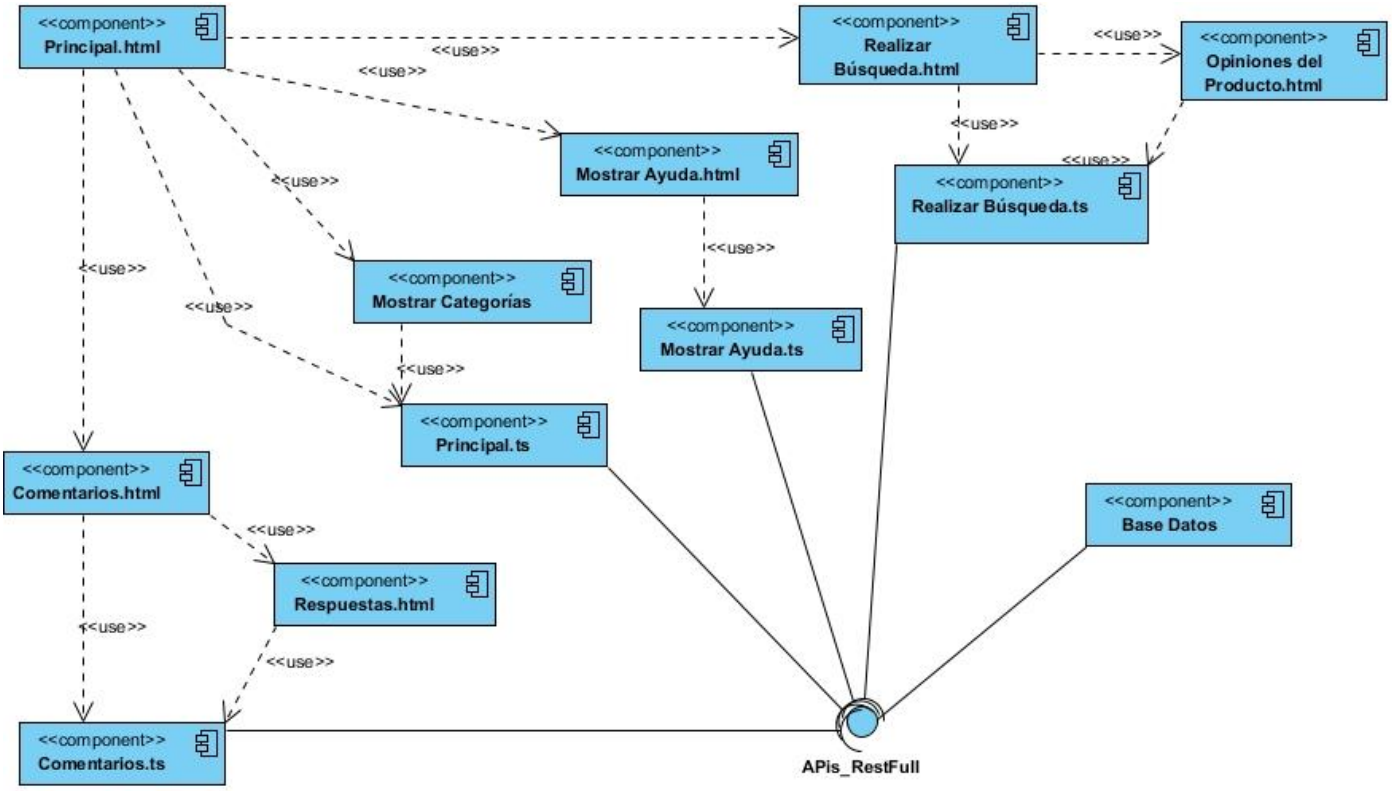


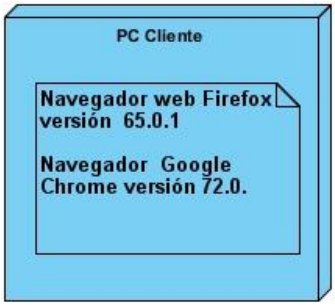
Figura 9. Diagrama de Componente.

3.2 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. (SYSTEMS, 2019).

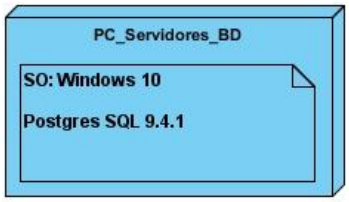
Tabla 8. Descripción de los Estereotipos Web.

Estereotipos web para las clases del diseño	
COMPONENTE	**DESCRIPCIÓN**



PC Cliente: este nodo representa el componente por el cual el usuario accederá al sistema propuesto con el uso de un navegador web.

PC Servidor: representa el componente donde se encontrará una parte del sistema, y además, es donde se establecerán las llamadas a los servicios del backend para el correcto funcionamiento del sistema.



PC Servidores Base Datos: constituye el componente en donde se encontrará ubicado el backend del sistema con las consultas SQL a la base de datos del sistema de gestión desarrollado, y que permitirá la gestión de los elementos de la base de datos.

A continuación, se muestra en la figura el Diagrama de Despliegue del sistema de búsqueda de productos en tiendas:

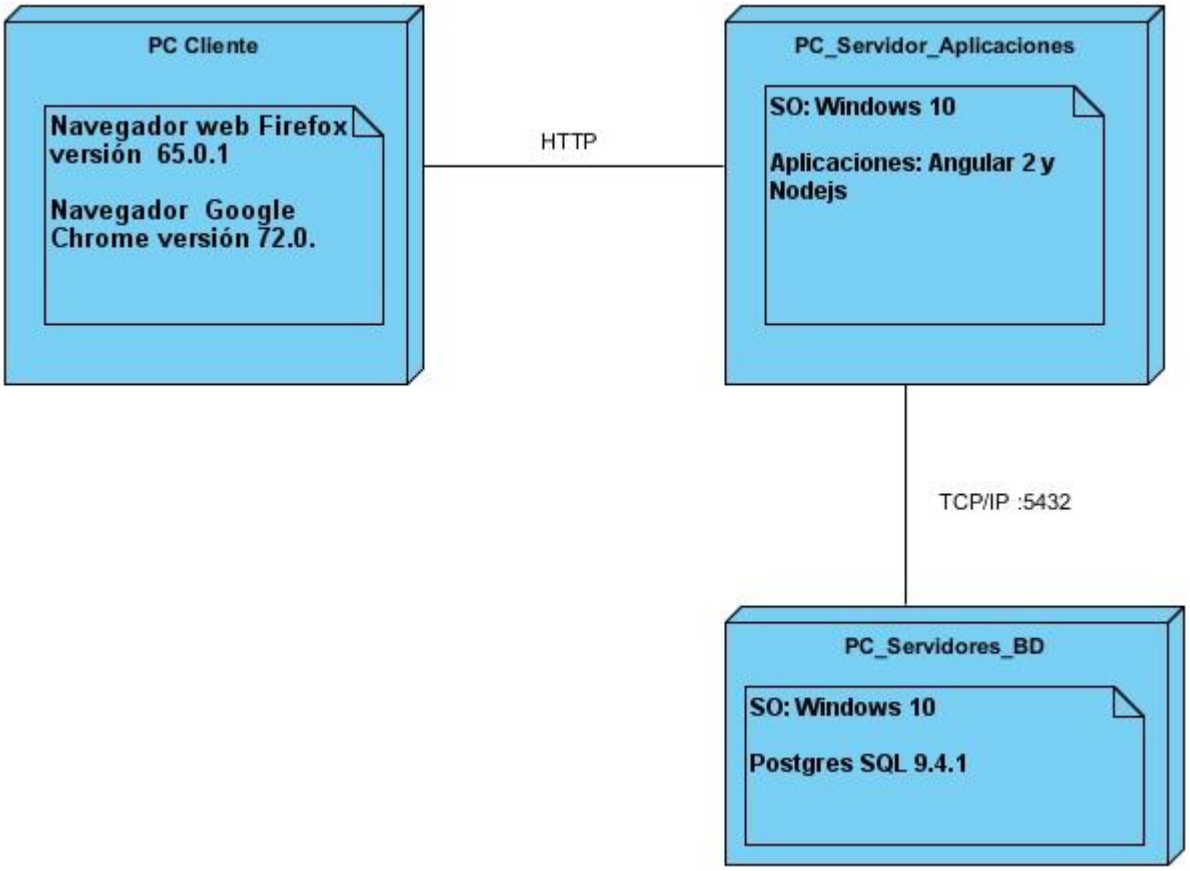


Figura 10. Diagrama de Despliegue.

3.3 Pruebas de Software

Las pruebas de software (en inglés software testing) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder. Es una actividad más en el proceso de control de calidad. Estas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo. (Enfoque para pruebas de unidad basado en la generación aleatoria de objetos, abril 2014)

3.4 Descripción del método de prueba Caja Blanca aplicado

La prueba de caja blanca se basa en el diseño de la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que se centran en evaluar la ejecución por lo menos una vez de cada sentencia del programa. (House, 2001)

Se centra en el estudio minucioso de la operatividad de una parte del sistema considerando los detalles procedurales (la lógica del sistema). Usa la estructura de control del diseño procedimental para obtener los casos de prueba.

• Estos casos deben garantizar:

- Que se ejercita por lo menos una vez todos los caminos Independientes de cada módulo.
- Que se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.

Prueba de Camino Básico:

La prueba del camino básico es una técnica de prueba de la Caja Blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. (Pressman, 2010)

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, obtiene el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo obtenido.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Para la realización del Método de Caja Blanca se escogió el método de Búsqueda de Producto por todos los criterios, el mismo representa el método más crítico debido a que el mismo tiene el objetivo principal del sistema.

Método:

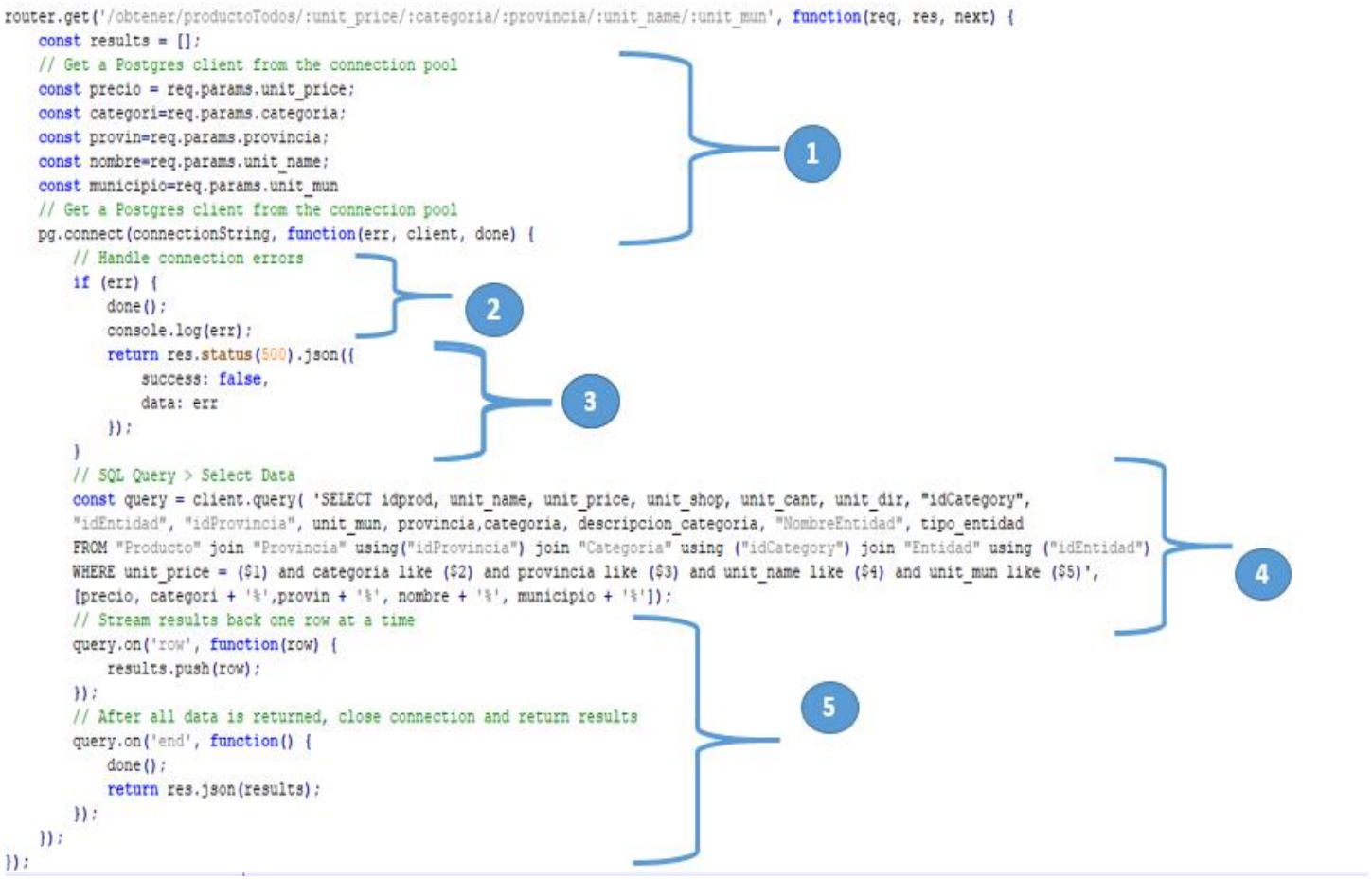


Figura 11. Método del API "Obtener Producto por Todos los Criterios".

Grafo Resultante:

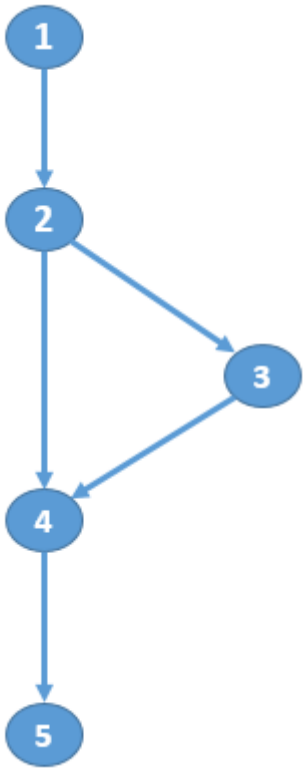


Figura 12. Grafo Resultante.

Leyenda:

- **A-# de aristas**
- **N- # de nodos**
- **P- regiones cerradas**

Complejidad Ciclomática:

V (G)= # de regiones **=**2 **V (G)=A-N+2 =**5-5+2**=**2 **V (G)= P+1 =**1+1**=**2
<u>Caminos lógicos:</u> **Camino #1: 1 2 4 5** **Camino #2: 1 2 3 4 5**

Descripción de los caminos obtenidos.

Tabla 9. Descripción del Camino #1.

Caso de prueba #1 para el camino básico:	1 2 4 5
Descripción:	No devuelve los valores requeridos de la consulta SQL, lanza un mensaje de Error.
Condición de Ejecución	-
Procedimiento prueba automatizada	
Datos de Entrada	Nombre , precio , categoría , provincia, municipio
Tipos de Datos Esperados	String, float, interger, interger, interger
Evalución del Caso de Prueba	Satisfactoria.

Tabla 10. Descripción del Camino #2.

Caso de prueba #1 para el camino básico:	1 2 3 4 5
Descripción:	La consulta SQL se realiza correctamente devolviendo todos los datos esperado.
Condición de Ejecución	-
Procedimiento prueba automatizada	
Datos de Entrada	Nombre , precio , categoría , provincia, municipio

Tipos de Datos Esperados	String, float, interger, interger, interger
Evalución del Caso de Prueba	Satisfactoria.

3.5 Pruebas Servicios Web Rest Crud

Estas pruebas son realizadas a los servicios web REST generados a partir de las entidades de la aplicación. Estos servicios son generados automáticamente por lo que se mostrarán las pruebas realizadas a un servicio web REST donde se solicite crear, editar, eliminar y leer datos de una entidad (CRUD). (Muñoz, 2013). Ayuda a satisfacer los requisitos de la integración que son críticos para construir sistemas en los que los datos se puedan combinar fácilmente, y para extender o construir un conjunto de servicios de RESTful básicos y crear algo mucho mayor.

Obtener Producto por Todos los Criterios de Búsqueda.

En esta prueba se realizó un llamado GET para obtener los datos de las instituciones en formato JSON.

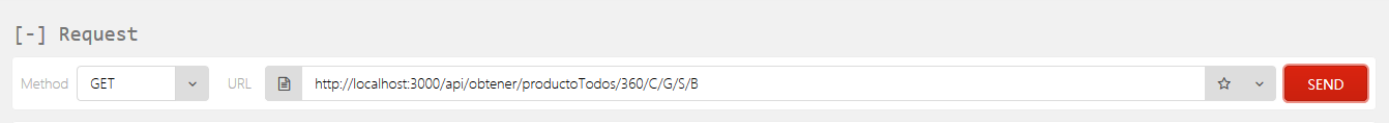


Figura 13. Prueba del API "Obtener por Todos los Criterios de Búsqueda"

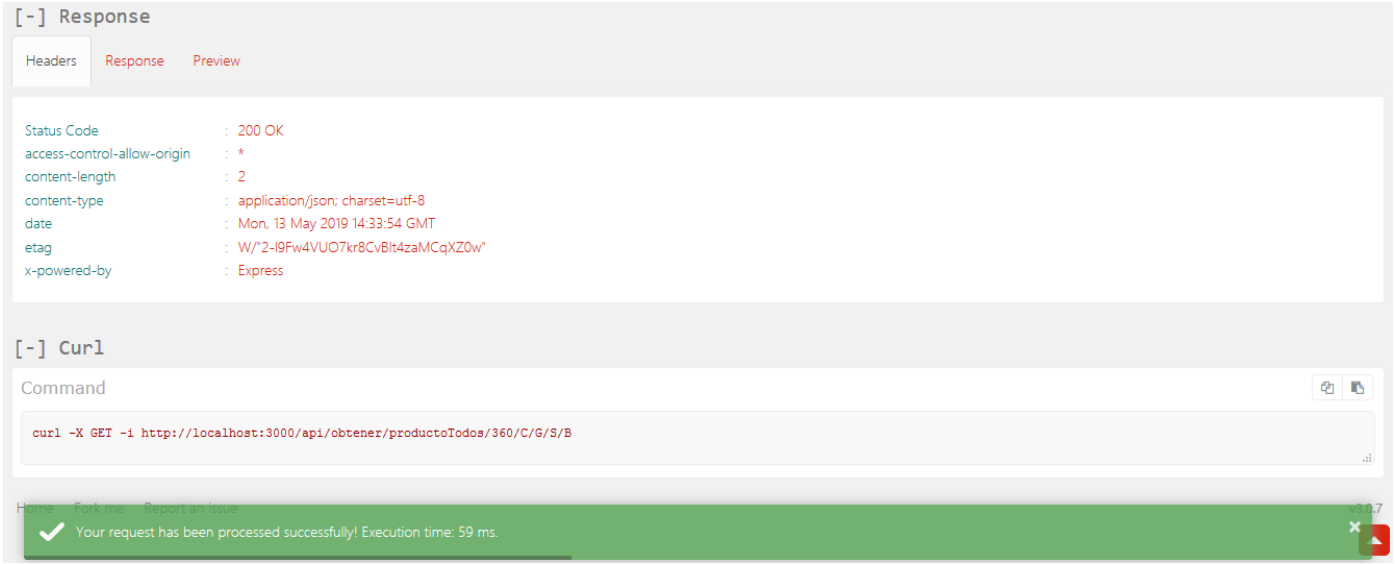


Figura 14. Resultad de la prueba del API "Obtener por Todos los Criterios de Búsqueda"

3.6 Descripción del método de prueba Caja Negra aplicada.

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2010).Analiza principalmente la compatibilidad entre sí, en cuanto a las interfaces, de cada uno de los componentes del software (no tiene en cuenta la lógica del sistema).

La prueba de caja negra intenta encontrar las siguientes categorías de errores (Pressman, 2010).

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

3.5.1 Técnica de partición equivalente:

Este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia y definir el menor número de casos de prueba que descubran clases de errores. (PRUEBAS FUNCIONALES USANDO TÉCNICAS DE CAJA NEGRA – PARTE I , 2013)

El diseño de casos de prueba según esta técnica consta de dos pasos:

1. Identificar las clases de equivalencia.

2. Identificar los casos de prueba.

• Una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realzada con cualquier otro valor de dicha clase.

• Si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error.

Tabla 11. Descripción del escenario de Prueba del Caso de Uso "Realizar Búsqueda".

Escenario	Descripción	Criterio de Búsqueda	Respuesta del modulo	Flujo Central
EC 1.1 Realizar Búsqueda.	El escenario de prueba permite al cliente buscar un producto a través de criterios de búsqueda como nombre, provincia, municipio, categorías y precio.	V Samsung Santiago de Cuba Palma Soriano Computadores, Celulares y accesorios.	El módulo verifica los datos y devuelve la lista de productos que coinciden.	1- Presionar el botón ‘Buscar Productos’. 2- Definir los criterios de búsqueda por nombre, provincia, municipio, categorías y precio. 3- Presionar el

				botón ‘Buscar’.
EC 1.2 Datos incorrectos del nombre del producto.	El escenario de prueba permite al cliente validar campos incorrectos. En caso de que el usuario realice la búsqueda por el criterio del nombre del producto.	I Samsung2%$	El módulo muestra un mensaje: “El valor entrado contiene caracteres no validos”.	<Lo mismo>
EC 1.3 Campo obligatorio vacío nombre, provincia, municipio, categoría y precio.	El escenario de prueba permite al cliente validar campos vacíos tanto sea por el nombre del producto o los filtros de provincia, municipio, categoría, precio.	I	El módulo muestra un mensaje: “No ha definido criterio para realizar su búsqueda”.	<Lo mismo>

Tabla 12.Descripción del escenario de Prueba del Caso de Uso "Administrar Comentarios".

Escenario	Descripción	Datos de los comentarios.	Respuesta del modulo	Flujo Central
EC 1.1 Crear Comentarios	El escenario de prueba permite que el usuario pueda crear un comentario.	V Usuario1 Casos de Pruebas.	El módulo verifica los datos que pasa el usuario e inserta el comentario.	1. Presionar el botón 'Comentarios'. 2. Presionar la opción 'Insertar Comentario'. 3. Rellenar los campos del formulario insertar comentario. 4. Presionar el botón Comentar.
EC 1.2 Datos incorrectos del nombre del usuario.	El escenario de prueba permite al cliente validar campos incorrectos.	I Nombre2%$!	El módulo muestra un mensaje: "El valor entrado contiene caracteres no validos".	<Lo mismo>

EC 1.3 Campo obligatorio vacío, nombre del usuario.	El escenario de prueba permite al cliente validar los campos vacíos de Nombre del usuario y Comentarios.	I	El módulo no habilita al usuario el botón para Comentar, evitando comentarios sin nombre y vacíos.	<Lo mismo>

3.7 Resultado de las pruebas funcionales aplicando el método de caja negra

Para validar el correcto funcionamiento del software se realizó la prueba de caja negra a través de los casos de pruebas asociados a cada caso de uso. Fueron detectadas u total de seis no conformidades con la calificación: 2 de funcionalidad, 2 opciones que no funciona y 2 errores ortográficos durante 2 iteraciones, en la primera se detectaron seis no conformidades y en la segunda no se encontró ninguna no conformidad. De forma general se excluyeron todos los errores culminando las pruebas con resultados satisfactorios.

Figura 15. Gráfico de No Conformidades.

Entiéndase por NC de funcionalidad cuando los resultados mostrados al realizar una acción no son los esperados, NC de interfaz a los errores visuales relacionados con el diseño de las interfaces, NC de recomendaciones son sugerencias hechas por los probadores y NC de opciones que no funcionan son las que no muestran ningún resultado al realizar alguna acción.

A continuación, en el **"Registro de defectos y dificultades detectados"** se muestran ejemplos de las no conformidades detectadas durante las pruebas a la aplicación, así como su estado (Pendiente (PD) o Resuelto (RA)) y la respuesta del equipo de desarrollo:

Tabla 13. Registro de defectos y dificultades detectados.

Clasificación	**No.**	**No conformidad (NC)**	**Estado NC**	**Resp. equipo de desarrollo**
NC funcionalidad	1	Al hacerse clic en la opción Respuesta de los Comentarios no se muestran las respuestas del mismo.	PD 17/4/19 RA 22/4/19	Se realizó el método de mostrar las respuestas, especificando el mismo por el ID de los comentarios.
NC opciones que no funcionan	2	Al hacerse clic en la opción Ayuda, desde la vista principal el mismo no remite a la vista de la opción.	PD 18/4/19 RA 22/4/19	Se especificó el método RouterLink, para redirigir a la Página al través del botón.
NC funcionalidad	3	Al hacerse clic en la opción Cancelar, desde la vista Comentarios, Insertar Comentarios, el mismo no cierra la vista.	PD 19/4/19 RA 23/4/19	Se agregó un método para cerrar la vista Comentarios.
NC errores ortográficos	4	La palabra de Categoría en la vista principal no estaba	PD 19/4/19	Se corrigió la palabra encontrada en la etiqueta

		acentuada.	RA 19/4/19	Categoría .
NC opciones que no funcionan	5	El video tutorial ubicado en la vista Ayuda, no cargaba.	PD 20/4/19 RA 24/4/19	Se revisó la etiqueta de video , la cual no tenía la propiedad src.
NC errores ortográficos	6	En la vista Comentarios la palabra Insertar Comentarios, estaba mal escrita.	PD 20/4/19 RA 20/4/19	Se corrigió la palabra encontrada en la etiqueta Insertar Comentarios .

3.8 Prueba de Carga y Estrés

Con el objetivo de garantizar la calidad en el software se realizan los siguientes tipos de pruebas:

- **Prueba de Carga:** enfocada a evaluar el rendimiento del sistema y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios o transacciones que se espera en el ambiente de producción.
- **Prueba de Estrés:** el objetivo de estas pruebas es someter al software a situaciones extremas, intentar que el sistema se colapse, para ver cómo se comporta, si es capaz de recuperarse o tratar correctamente un error grave. Permite encontrar el volumen de datos o el tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones realizadas. Son pruebas de carga, pero superando los límites esperados en el ambiente de producción.

Resultados de las pruebas de Carga y Estrés

Para la realización de dichas pruebas se hizo necesario tener en cuenta las condiciones del escenario, tanto del hardware como software, donde se encuentra la aplicación para obtener una correcta información de comportamiento y resultados en general.

Hardware

- Tipo de procesador: Intel(R) Core(TM) i3-6100 CPU @ 2.30GHz
- Memoria instalada RAM: 4,00 GB
- Tipo de Sistema: Sistema Operativo de 64bits, procesador x64
- Modelo: ACER Aspire

Software

- Plataforma: Windows 10 Home Single Language de 64 bits

La prueba de carga y estrés para los servicios web se realizó utilizando la herramienta JMeter mencionada en el Capítulo 1. La misma se configuró utilizando el árbol de configuración de la Figura 18 para que se llevaran a cabo 5 grupos de peticiones las cuales estaban dadas en el siguiente orden:

- Realizar búsqueda avanzada.
- Insertar comentario.
- Mostrar comentarios.
- Insertar respuesta.
- Mostrar respuesta.

Figura 16. Árbol de Configuración de JMeter.

JMeter permite establecer como variables principales a la hora de realizar una simulación las que se muestran seguidamente, y para una mejor comprensión ver Figura 19:

- **Número de hilos**: Se corresponde con el número de procesos o usuarios concurrentes que van a ejecutar las peticiones, en este caso para los tres primeros escenarios (50, 250, 400, 800) usuarios, agregando 1000 usuarios en el escenario 3.
- **Periodo de subida**: Es el tiempo en segundos que tardan en iniciarse todos los procesos, en este caso se emplearon 3, 2 y 1 segundos respectivamente en cada escenario.
- **Control de bucle**: Es el número de veces que se van a lanzar las iteraciones por los n usuarios determinados en el número de hilos siendo 1 en los 3 escenarios.

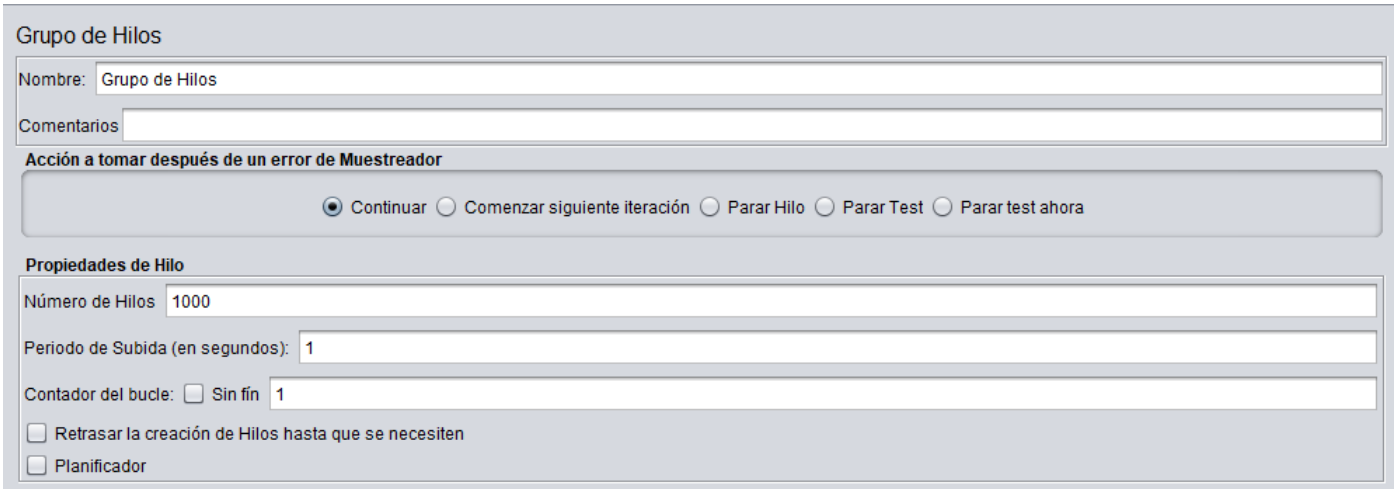


Figura 17.Configuración del JMeter en el Escenario 3 con 1000 hilos.

La tabla que se muestra a continuación corresponde a los datos de entrada para el escenario 3, además se muestran el resultado obtenido para la mayor cantidad de usuarios representado en las variables: cantidad total de solicitudes, tiempo medio de respuesta y el porciento de error.

Tabla 14. Resultado obtenido en el Escenario 3 para los casos de 50, 250, 400, 800 y 1000.

Escenario 3

Período de subida	Casos	Resultados para 1000 usuarios		
1 segundos	50, 250, 400, 800, 1000 usuarios	Solicitudes	Tiempo medio	Errores
		12500	1556 milisegundos	0%

Seguidamente se muestra el informe agregado correspondiente al escenario 3 y a su último caso (mayor cantidad de usuarios concurrentes), y se representan los valores obtenidos en las variables: número de muestras (solicitudes), tiempos de respuesta de la aplicación (media, mediana, máximo, mínimo y línea del 90%); errores de solicitudes no resueltas, rendimiento en muestras por milisegundo, la cantidad de kilobytes (Kb) que el servidor procesa y el total de cada una de estas variables.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimie...	Kb/sec	Sent KB/s...
Realizar B...	2500	1299	1227	2414	2662	2937	6	2992	0.00%	12.0/sec	24.31	1.55
Mostrar R...	2500	1741	1749	2645	2652	2662	14	2703	0.00%	11.6/sec	33.09	1.46
Mostrar C...	2500	1533	1616	2404	2420	2435	14	2444	0.00%	11.8/sec	33.82	1.49
Insertar R...	2500	1512	1600	2364	2621	2655	16	2673	0.00%	11.5/sec	32.88	1.45
Insertar C...	2500	1693	1696	2582	2613	2624	11	2637	0.00%	11.7/sec	33.43	1.47
TOTAL	12500	1556	1613	2475	2626	2667	6	2992	0.00%	57.4/sec	154.49	7.27

Figura 18. Informe agregado de Escenario 3 con1000 usuarios y periodo de subida de 1 segundos.

3.9 Pruebas de Integración

Las pruebas de integración son “una técnica sistemática para construir la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados a la interfaz” (Pressman, 2005).

Estas son definidas para verificar el correcto ensamblaje entre los distintos módulos que conforman un sistema informático. Las mismas validan que estos componentes realmente funcionan juntos, son llamados correctamente y, además, transfieren los datos correctos en el tiempo preciso y por las vías de comunicación establecidas (Sommerville, 2005).

Para la aplicación de las pruebas de integración se utilizan principalmente técnicas que verifican el correcto manejo de las entradas y salidas del software, es decir, las pruebas funcionales. Existen dos tipos de pruebas de integración (Suarez, 2012):

- Incremental
- No incremental

Dentro de las pruebas de integración incremental existen dos enfoques principales:

- Integración descendente (componentes funcionales).
- Integración ascendente (componentes de infraestructura).

Para la realización de pruebas de integración del Sistema *web* de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa, con el sistema *web* para la obtención de información de los productos que oferta la red de ventas minorista en el país, se eligió el tipo de prueba incremental, con un enfoque ascendente, la misma permite probar el programa en pequeñas porciones, lo que facilita la detección de los errores existentes.

Para probar la integración de los sistemas se realizaron los siguientes pasos:

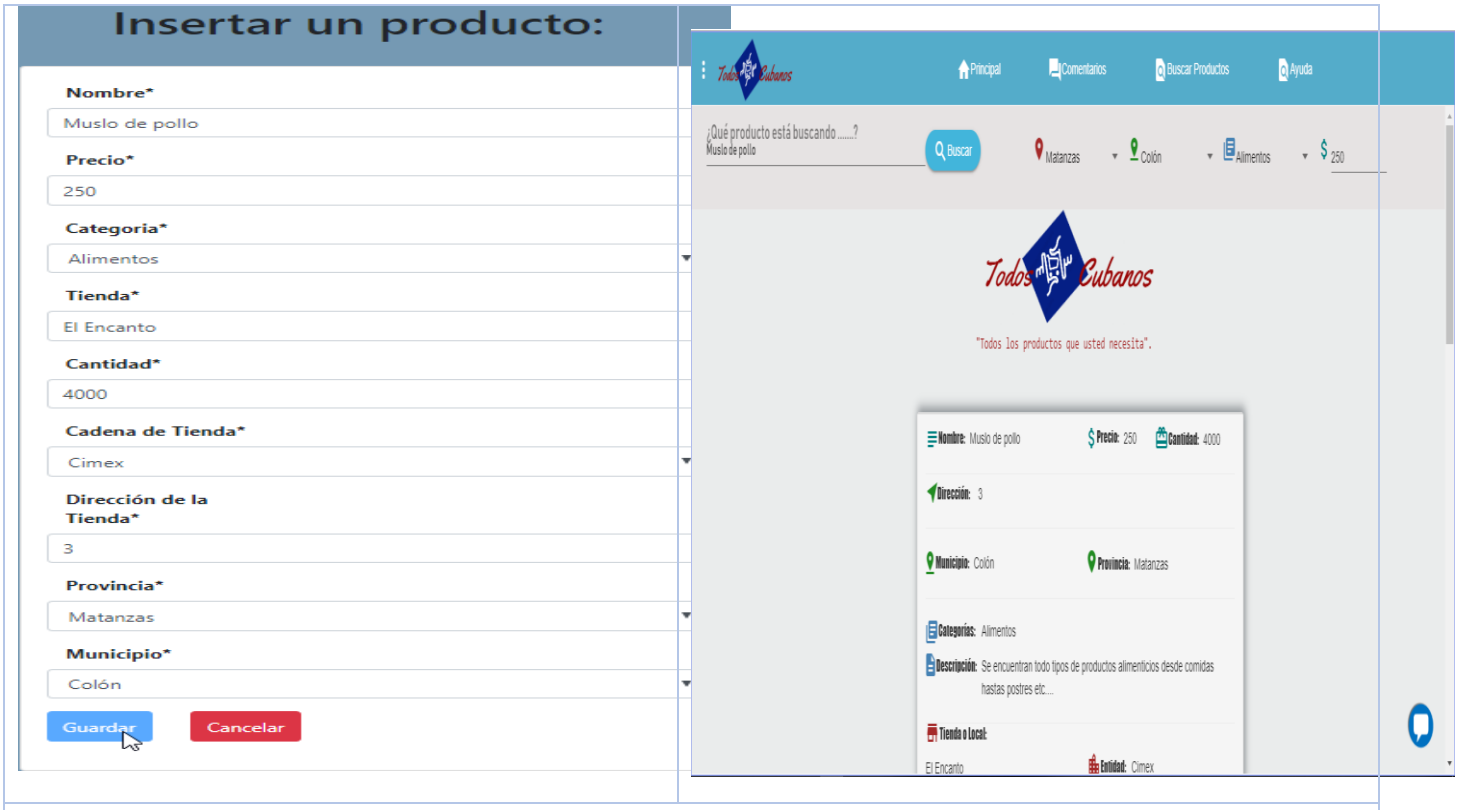
1. Se compiló el frontend del Sistema web de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa por el puerto 4200(localhost: 4200) y el backend por el puerto 3000(localhost:3000).
2. Se compiló el frontend del sistema web para la obtención de información de los productos que oferta la red de ventas minorista en el país por el puerto 4201(localhost: 4201) y el backend por el puerto 3001(localhost: 3001).
3. La base de datos de ambos módulos es la misma y se compiló por el puerto 5432(localhost: 5432)

A continuación, en la tabla se muestra el caso de prueba de integración Insertar Producto y Buscar Producto.

Tabla 15.Caso de prueba de integración Insertar Producto y Buscar Producto.

Caso de prueba de integración: Insertar Producto y Buscar Producto
Sistema al que se integra: Sistema web de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa.

<table>
<tr><td colspan="2">Condiciones de ejecución: Un proveedor inserta un cliente y un cliente lo busca.</td></tr>
<tr><td colspan="2">Descripción de la prueba: Comprobar que los sistema son capaces de comunicarse y que los productos se envíen al sistema web de búsqueda.</td></tr>
<tr><td colspan="2">Entradas/Pasos de ejecución:
Sistema web para la obtención de información de los productos que oferta la red de ventas minorista en el país.
1. Se selecciona la opción Insertar Producto.
2. Se inserta un producto.
Sistema web de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D’Prisa.
1. Se selecciona la opción Buscar Producto.
2. Se busca el producto.</td></tr>
<tr><td colspan="2">Resultado esperado: Los sistemas son capaces de enviar y recibir la información de los productos.</td></tr>
<tr><td>Interfaz del Proveedor:</td><td>Interfaz del Buscador:</td></tr>
</table>



Evaluación: Prueba satisfactoria.

Tras haber ejecutado el caso de prueba diseñado para la prueba de integración, los cuales arrojaron resultados satisfactorios, queda demostrada la correcta integración del módulo elaborado con el sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de microservicios sobre la plataforma D'Prisa.

3.10 Prueba de aceptación

Esta es la etapa final en el proceso de pruebas, antes de que el sistema se acepte para uso operacional. El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados. Las pruebas de aceptación revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba. Las pruebas de aceptación revelan problemas de requerimientos, donde las

instalaciones del sistema en realidad no cumplan las necesidades del usuario o cuando sea inaceptable el rendimiento del sistema (Sommerville, 2011).

Prueba de Aceptación de tipo alfa

Estas pruebas son realizadas por el cliente (Jefe del Proyecto) para certificar que el sistema es válido para él. Son básicamente pruebas funcionales sobre el sistema una vez terminado y buscan comprobar que se satisfacen los requisitos establecidos. Son realizadas en presencia del desarrollador, este en caso de que existan errores los registrará y posteriormente les dará solución. Esta prueba es ejecutada nuevamente, repitiendo el proceso hasta lograr que el producto quede libre de errores y el cliente obtenga el producto que solicitó (Valdez, et al., 2013). La misma fue realizada con el Jefe de la División de Tecnología del la Empresa XETID y el Jefe de Centro de producción TeDIS. Se redactaron dos actas confirmando la aceptación del cliente con el sistema web,las mismas se pueden apreciar en los ANEXOS.

Conclusiones del capítulo

De los 8 casos de uso definidos en la fase de análisis y diseño se logró implementar el 100% de los mismas. Se realizaron las pruebas funcionales y de aceptación a través del diseño de casos de pruebas, uno por cada caso de uso, utilizando el método caja negra con la técnica, partición de equivalencia y el método de caja blanca con Camino básico, arrojaron como resultado 6 no conformidades, que fueron solucionadas demostrando que la solución desarrollada cumple con las características especificadas por el cliente.

CONCLUSIONES

La realización del presente trabajo posibilitó cumplir con los objetivos y tareas propuestas para su desarrollo, arribándose las siguientes conclusiones:

- Al realizar el análisis se obtuvieron 9 requisitos no funcionales y 13 requisitos funcionales, estos últimos agrupados en 8 casos de uso del sistema utilizando los patrones de casos de uso Extensión e Inclusión concreta.

.

- Se implementó el Sistema de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa.

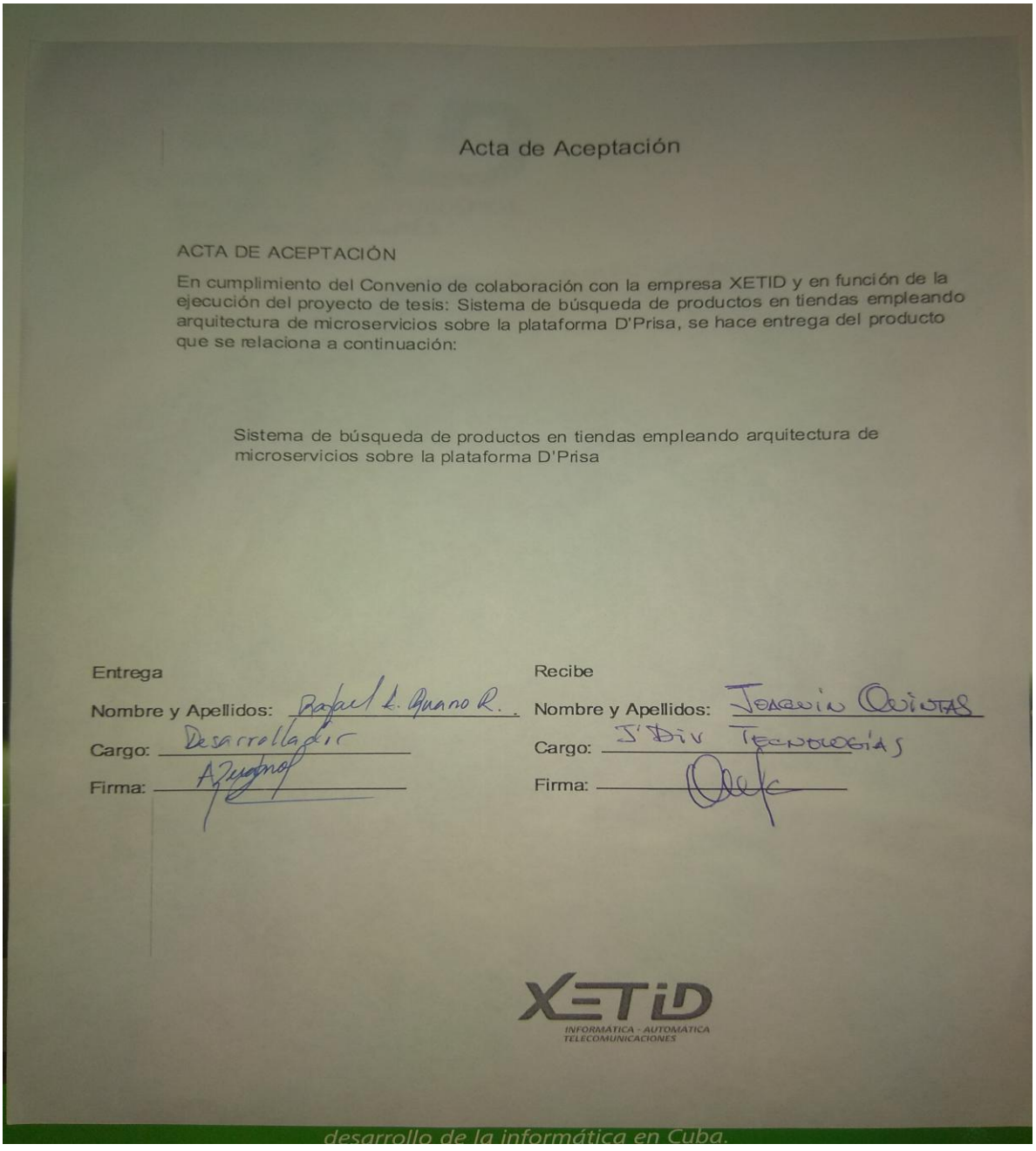
- Se validó la solución mediante la confección de casos de prueba uno por cada caso de uso y los métodos de caja blanca y caja negra, demostrando que la aplicación cumple con las funcionalidades identificadas.

RECOMENDACIONES

A lo largo de la presente investigación fueron identificados elementos que no se contemplaron en el desarrollo de la solución al no ser parte del alcance inicial, pero se considera que son relevantes e incrementarían la utilidad del sistema y la calidad de uso del mismo por lo que se recomienda:

- Se recomienda continuar el ciclo de desarrollo del sistema, realizando nuevas iteraciones y agregando nuevas funcionalidades para ganar en usabilidad, logrando así mejorar la competitividad en el mercado del software.
- Se recomienda el uso de tecnologías para la búsqueda en sistemas web tales como Apache Solr, Elasticsearch.

ANEXOS



Acta de Aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del Convenio de colaboración con la empresa XETID y en función de la ejecución del proyecto de tesis: Sistema de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa, se hace entrega del producto que se relaciona a continuación:

Sistema de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa

Entrega

Nombre y Apellidos: Rafael L. Guano R.

Cargo: Desarrollador

Firma:

Recibe

Nombre y Apellidos: Joaquín Quintas

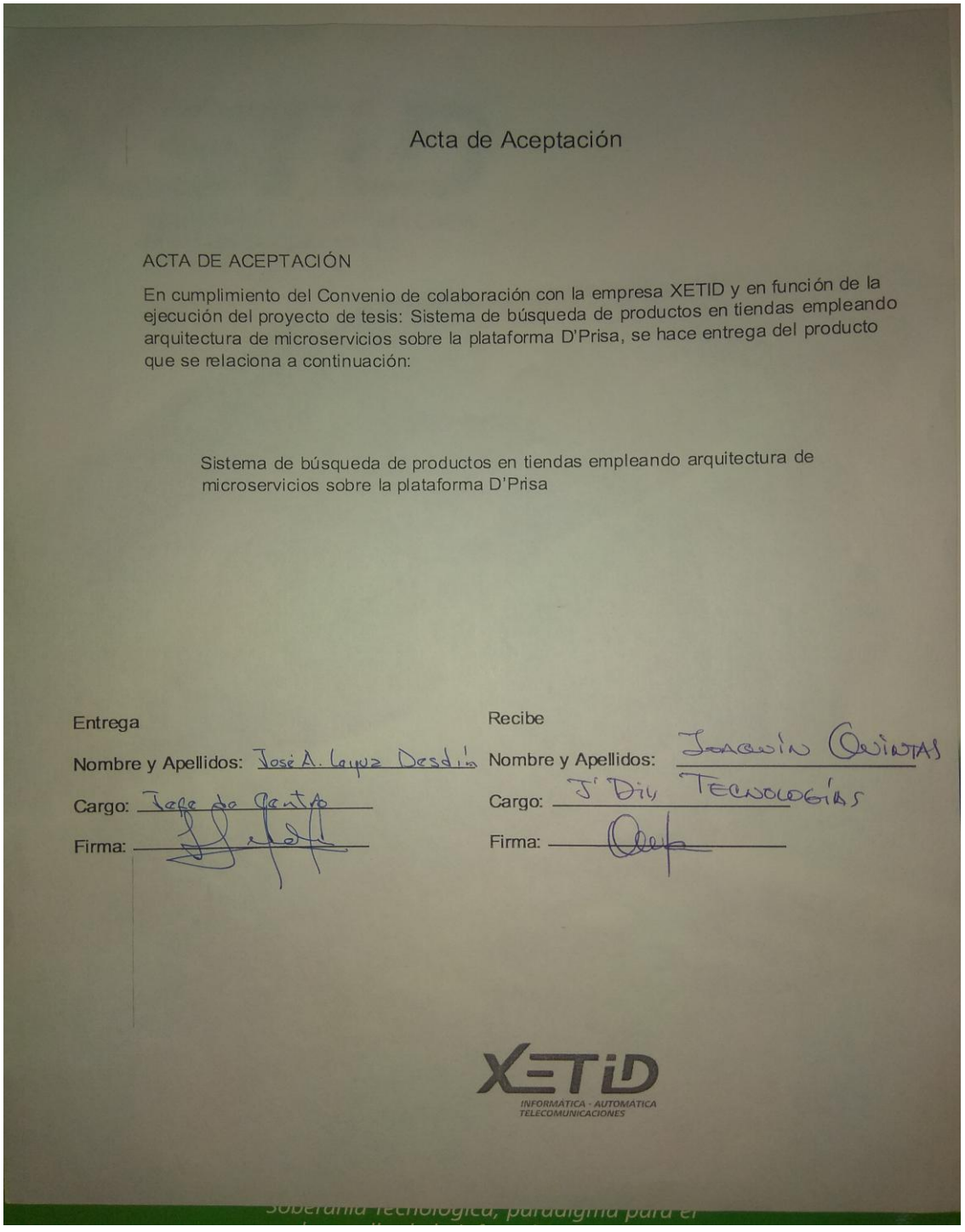
Cargo: J'Div Tecnologías

Firma:

XETID
INFORMÁTICA - AUTOMÁTICA
TELECOMUNICACIONES

desarrollo de la informática en Cuba.

Figura 19. Carta de Aceptación del Jefe de Tecnología presentada por el desarrollador.



Acta de Aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del Convenio de colaboración con la empresa XETID y en función de la ejecución del proyecto de tesis: Sistema de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa, se hace entrega del producto que se relaciona a continuación:

Sistema de búsqueda de productos en tiendas empleando arquitectura de microservicios sobre la plataforma D'Prisa

Entrega

Nombre y Apellidos: José A. Lapuz Desdín

Cargo: Jefe de Centro

Firma:

Recibe

Nombre y Apellidos: Joaquín Quintas

Cargo: J' Div. Tecnologías

Firma:

XETID
INFORMÁTICA · AUTOMÁTICA
TELECOMUNICACIONES

Figura 20.Carta de Aceptación del Jefe de Tecnología presentada por el Jefe de Centro.

REFERENCIAS BIBLIOGRÁFICAS.

1) **Barrientos, Pablo Andrés.**
2) **Belloch. 2012.** 2012.
3) **Calvache, Arias y. 2016.** 2016.
4) **Cendejas. 2011.** 2011.
5) **Center. 2013.** 2013.
6) **Cillero, Manuel. 2019.** manuel.cillero.es. [Online] 5 12, 2019.
7) **EcuRed. 2019.** EcuRed. [Online] 2019. https://www.ecured.cu/Patrones_de_Casos_de_Uso.
8) **2014.** Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. [Online] 2014.
9) *Enfoque para pruebas de unidad basado en la generación aleatoria de objetos.* **Barrientos, Pablo Andrés. abril 2014.** abril 2014.
10) **2018.** Estrategia Digital Orientada a Resultados, HTML 5. [Online] 2018. [Cited: 12 10, 2018.] https://www.arimetrics.com/glosario-digital/html5.
11) **GAVIRIA, BEATRIZ FLORIAN.** [Online]
12) **Geers, Michael. 2018.** What are Micro Frontends? [Online] 2018. [Cited: 11 23, 2018.] https://micro-frontends.org/.
13) **Henao, Jose Mario Valencia. 2018.** Diseño y aplicación de un sistema bajo arquitectura cliente servidor para la activación de riego automatizado en Arduino a traves de Twitter. [Online] 2018. [Cited: 11 20, 2018.] https://www.researchgate.net/publication/328174883_Diseno_y_aplicacion_de_un_sistema_bajo_arquitectura_cliente_servidor_para_la_activacion_de_riego_automatizado_en_Arduino_a_traves_de_Twitter.
14) **House. 2001.** 2001.
15) **Larman. 2003.** 2003.
16) **MDAP . 2019.** MDAP Executive Master Project Management. [Online] 5 11, 2019.
17) **Microsoft. 2016.** 2016.
18) **Muñoz. 2013.** 2013.
19) **Nube", Nativas de "La. 2019.** [Online] 2019.
20) **Oterino, Ana M. del Carmen García. 2015.** ¿Qué es eso de los microservicios? [Online] 2015. [Cited: 11 20, 2018.] http://www.javiergarzas.com/2015/06/microservicios.html.

21) **Pressman. 2010.** *Ingenieria de Software Un Enfoque Práctico.* 2010.
22) **—. 2011.** *Software Ingienery.* 2011.
23) **Prieto Paco. 2019.** *La Servilleta.* 2019.
24) **Prieto, Paco. 2019.** La Servilleta. [Online] 2019.
25) *PRUEBAS FUNCIONALES USANDO TÉCNICAS DE CAJA NEGRA – PARTE I .* **GAVIRIA, BEATRIZ FLORIAN. 2013.** 2013.
26) **Rivera and otros, y. 2010.** 2010.
27) **Salazar. 2012.** 2012.
28) **Santiago. 2018.** 2018.
29) **Santiago, Joaquín Quintas. 2018.** La Habana : s.n., 2018.
30) **Scielo. 2013.** 2013.
31) **Software. 2013.** 2013.
32) **Somerville. 2011.** 2011.
33) **Sosa. 2013.** 2013.
34) **Sosa, Dailyn. 2013.** Sitio Web Eumed.net. *Sitio Web Eumed.net.* [Online] 2013.
35) **Svitla Systems. 2016.** [Online] 2016.
36) **SYSTEMS, SPARX. 2019.** SPARX SYSTEMS. [Online] 5 11, 2019.
37) **Unad. 2006.** [Online] 2006.
38) **Valdez, Abner Gerardo, et al. 2013.** Sitio Web Pruebas de sistemas y pruebas de aceptación. *Sitio Web Pruebas de sistemas y pruebas de aceptación.* [Online] 2013

1