FACULTAD DE INGENIERÍAS

Desarrollo de una tienda online con análisis de indicadores de ventas por medio de un modelo de datos para la empresa Quimpec Químicas Cia. Ltda.

AUTOR:

Jorge David Peralta Godoy

Trabajo de Titulación previo a la obtención del título de:

INGENIERÍA EN SISTEMAS

CON ÉNFASIS EN SISTEMAS

TUTOR:

Ing. Luis Espinoza

Guayaquil, Ecuador

2021

CERTIFICADO DEL PORCENTAJE DE COINCIDENCIAS

Habiendo sido nombrado LUIS ENRIQUE ESPINOZA MENDOZA, tutor del trabajo de titulación" **Desarrollo de una tienda online con reportes de indicadores de ventas para la empresa Quimpec Química Cia. Ltda.** " elaborado por **Jorge David Peralta Godoy** **,** con mi respectiva supervisión como requerimiento parcial para la obtención del título de INGENIERO EN SISTEMAS CON ENFASIS EN SISTEMAS.

Se informa que el mismo ha resultado tener un porcentaje de coincidencias 6(%) mismo que se puede verificar en el siguiente link: https://secure.urkund.com/view/103912363-375789-568400#/details/sources. Adicional se adjunta print de pantalla de dicho resultado.

Luis E. Espinoza M.

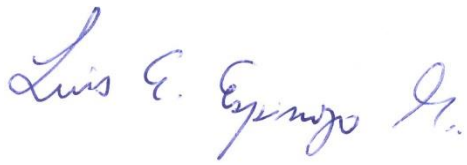
CERTIFICACION DE REVISION FINAL

QUE EL PRESENTE PROYECTO DE INVESTIGACIÓN TITULADO:

DESARROLLO DE UNA TIENDA ONLINE CON ANÁLISIS DE INDICADORES DE VENTAS POR MEDIO DE UN MODELO DE DATOS PARA LA EMPRESA QUIMPEC QUÍMICAS CIA. LTDA.

ACOGIÓ E INCORPORÓ TODAS LAS OBSERVACIONES REALIZADAS POR LOS MIEMBROS DEL TRIBUNAL ASIGNADO Y CUMPLE CON LA CALIDAD EXIGIDA PARA UN TRABAJO DE TITULACIÓN, POR LO QUE SE AUTORIZA A: **JORGE DAVID PERALTA GODOY,** QUE PROCEDA A SU PRESENTACION.

Samborondón, 5-Julio-2021



Tesis Lista Recibidos ×

LUIS ESPINOZA MENDOZA
para mí

Estimado después de revisar su tesis, cumple los requisitos para su sustentación.

Gracias

--

Ing. Luis Espinoza, M.Ed.

Docente Facultad de Ingenierías

PBX: 04 3723400 Ext.: 508

Resumen

Es importante adaptarse al cambio, e implementar las nuevas herramientas que las tecnologías de la información ofrecen en el mundo empresarial. El presente proyecto tiene como finalidad implementar una tienda online con reportes de indicadores de ventas para la empresa Quimpec Química Cia. Ltda. Utilizando la metodología de desarrollo ágil: Programación extrema (XP), esta metodología tiene como características principales: ser un modelo iterativo e incrementa, escribir las pruebas antes que el código, refactorización constante para mejorar la calidad del código, codificación en pares e integraciones continuas.

Se utilizó también la metodología de desarrollo orientado por pruebas (TDD, por sus siglas en inglés) lo cual permitió tener un producto terminado con pruebas automatizadas en su totalidad, estas pruebas de regresión facilitan las futuras mejoras que se le quiera añadir al producto entregado. Con la tranquilidad de agregar nuevas funcionalidades sin insertar errores (*bugs*) en las funcionalidades ya terminadas.

Los métodos ágiles tienen como objetivo adaptarse al cambio que se pueda dar en el proyecto durante el proceso de desarrollo. Este enfoque dio resultados pues se dieron nuevos cambios en el proceso de codificación lo cual no generó percances en la gestión del proyecto.

El desarrollo se dio con éxito, cumpliendo con todos los requerimientos expresados por el cliente y esto le ha permitido a Quimpec mejorar la capacidad instalada de ventas de la empresa, aumento de la cobertura de mercado y mostrar mejores indicadores para toma de decisiones a sus gerentes.

Abstract

An Online store and sales indicators reports were developed for Quimpec Químicas Company using the methodology Extreme Programming. This methodology has the following characteristics: it's an iterative and incremental model, follows the "write the tests first paradigm", continuous refactoring to improve code´s quality, pair programming and continuous integration.
It was also used the paradigm Test Driven Development (TDD), using this paradigm gave us a finished product with complete automated tests, this regression tests allow to write new functionality in the future, with the confidence of knowing that we are not inserting new bugs to the already finished functionalities.

The agile methods have as a purpose to help the project to adapt to the appearing changes during development. This Approach gave results because there were new requirements during coding, and because of the methodology used this was not a issue for the management of the project.

Development concluded with success, complying with all the requirements by the client. This Allowed Quimpec increase their sales installed capacity, their market coverage and showing better indicators their decision makers.

Índice

Tabla de Figuras

Índice de Tablas

Dedicatoria

A mis padres, Lety y Paco, las mejores guías que Dios me pudo dar.
Y a mis maestros de vida Ronny, Carmen y Sergio.

Agradecimientos

Le agradezco profundamente a todos quienes fueron mis profesores en la Universidad. Gracias por sus enseñanzas sobre desarrollo de software, pero, sobre todo, gracias por sus enseñanzas sobre la vida.

Introducción

Desde los inicios del comercio, los negocios han utilizado tecnología. En la década de 1970 las compañías empezaron a incorporar herramientas avanzadas de TI en sus modelos de negocio. A partir de la década de 1980, este tipo de tecnología penetró, al punto de convertirse en el corazón de los negocios; Transformando estructuras, procesos y toda la forma de hacer intercambios comerciales. (Sabki, 2020)

Castaño y Jurado (2016) definen el comercio electrónico como la utilización de técnicas del marketing tradicional en entornos digitales. Son varios los beneficios del comercio electrónico, entre ellos: crear nuevos canales de negocios, mejora la promoción del producto, reduce costos y permite tener una relación más cercana con el cliente.

En este contexto, es necesario que Quimpec Química Cia. Ltda. ofrezca una mayor presencia en la web a través de una tienda online que le permita competir y a la vez ofrecer sus productos en el mercado en las mismas o mejores condiciones que una tienda física, para ello se propone el desarrollo de una tienda online con indicadores de ventas, que aumente el volumen de estas ventas y ayude a la toma de decisiones.

Contexto histórico social del objeto de estudio

Antecedentes

En el año 2002, el Congreso Nacional del Ecuador aprobó la Ley del Comercio Electrónico firmas electrónicas y mensajes de datos, estableciendo la normativa sobre la cual rigen las operaciones comerciales en internet. En la que se aporta un marco legal a las comunicaciones y transacciones comerciales en internet, para de esta forma regular el comercio electrónico en el Ecuador.

En el Ecuador existen tiendas online o personas informales que venden sus productos usando las redes sociales, sin mayores controles de seguridad. El

Ministerio de Telecomunicaciones y de la Sociedad de la Información (Mintel) afirma que por datos de una encuesta. que un 60 por ciento de las personas no compran online debido a la desconfianza por motivos de seguridad y un 33 por ciento por la falta de conocimiento de uso (El Telégrafo, 2016).

Sin embargo, la implementación de herramientas de comercio electrónico sigue en aumento en nuestro país. Esparza (2016) afirma que actualmente el comercio electrónico se ha convertido en una estrategia organizacional que permite a las empresas obtener mayores ganancias monetarias y un impacto comercial más elevado que aquellas que no lo utilizan, además de permitir una gestión integral mediante los sistemas de información para tomar decisiones estratégicas apropiadas.

Planteamiento del problema a resolver

En la actualidad, para que una empresa pueda mantenerse competitiva, debe tener una fuerte presencia en la red, en palabras de Lee (2019): “no se puede competir en el mercado actual si no se usa y se entiende completamente las herramientas web que están disponibles para ayudar a su empresa.”

Actualmente la presencia de la empresa Quimpec en la web es limitada, solo cuenta con anuncios en su cuenta de la red social Instagram, pero no cuenta con ningún sistema que permita mostrar la oferta de sus productos, y procesar los pagos de forma automatizada.

Este sistema le permitirá a Quimpec aumentar sus ventas, abarcar un sector del mercado que actualmente tiene descuidado y apoyarse del reporte con indicadores de ventas para una mejor toma de decisiones.

Asensio (2017) sostiene que cualquier empresa que mantenga un sitio web o cualquier otro recurso digital, recibe información de sus posibles clientes en cada momento, en cada visita y en cada vista web. Esta es información que puede ser analizada para tener una percepción de los intereses y necesidades del cliente, que pueden ser utilizados para mejorar la relación con el consumidor.

Quimpec Química Cia. Ltda. Es una empresa galardonada por su innovación y por su eficiente gestión. Recibió el premio al primer lugar entre empresas manufactureras de la revista Ekos del año 2012 y el Bussiness Management Award en el 2013, por nombrar algunas. Sin embargo, su presencia en la web es limitada, y no cuenta con un sistema de ventas online ni con una forma de analizar el comportamiento de sus clientes en línea de acuerdo a hechos pasados.

Por lo tanto, para llegar a una situación óptima es necesario implementar un sistema con una tienda online web, la cual incluirá un tablero con reportes de los indicadores de ventas. que permitirá medir la rentabilidad online de Quimpec.

Ante todo, lo expuesto, con esta propuesta se pretende dar respuesta a:

¿Cómo mejorar la toma de decisiones de la empresa Quimpec con vista a incrementar las ventas?

Objetivo general

Desarrollar una tienda online con análisis de indicadores de ventas que permita mejorar la toma de decisiones de la empresa Quimpec Química Cia. Ltda.

Objetivos específicos

- Identificar referentes teóricos relacionados con el desarrollo de soluciones tecnológicas aplicadas al comercio electrónico.

- Diagnosticar la información asociada a la toma de decisiones en el proceso de ventas dentro de Quimpec Química Cia. Ltda.

- Definir los requerimientos de la tienda online de Quimpec Química Cia. Ltda.

- Desarrollar los diferentes módulos que componen la tienda en línea de Quimpec Química Cia. Ltda.

- Implementar pruebas de verificación las cuales comprenden las pruebas de unidad, y las pruebas de integración de la tienda online de Quimpec Química Cia. Ltda.
- Implementar pruebas de validación, conformadas por las pruebas de validación para la tienda online de Quimpec Química Cia. Ltda.

Marco Teórico

Capítulo 1

Patrones de diseño

Un patrón de diseño es una forma recurrente de manejar problemas similares. Usar patrones de diseño nos hace posible trabajar con código limpio, legible y reutilizable. Los patrones documentan conocimiento existente de diseño y nos ayudan a encontrar soluciones apropiadas a problemas actuales. Cada patrón con un problema específico y recurrente en el diseño o en la implementación de un sistema. Los patrones también pueden ser usados para construir arquitecturas de software con propiedades específicas (Buschmann, Meunier, Rohnert , Sommerlad, & Stal, 2001).

Tipo de patrones

Patrón 'Factory'

El patrón de diseño 'Factory' es un patrón creacional que usa métodos de construcción para lidiar con el problema de crear objetos sin tener que especificar la clase exacta del objeto que va a ser creado.

Esto se consigue llamando a métodos de construcción en el momento de crear objetos, ya sea especificado en una interfaz e implementado por una clase hija, o implementada en una clase base y sobrescrita por clases hijas. Todo esto en lugar de llamar al constructor (Kilicdagi, 2014).

Patrón Modelo-Vista-Controlador (MVC)

Es un patrón de diseño que se fundamenta en el principio de que la lógica de una aplicación debe estar separada de su presentación. Divide un software dado en 3 partes interconectadas con la finalidad de diferenciar las representaciones internas de información de la manera que la información es presentada al usuario.

- **Modelo**: Se puede definir de forma sencilla como la capa que maneja la información. El Modelo implementa la lógica del negocio de la aplicación, es decir que el Modelo es responsable de obtener información y convertirla en información mas valiosa que pueda ser manejada por otras capas de la aplicación y que sean enviadas de vuelta a las capas correspondientes.

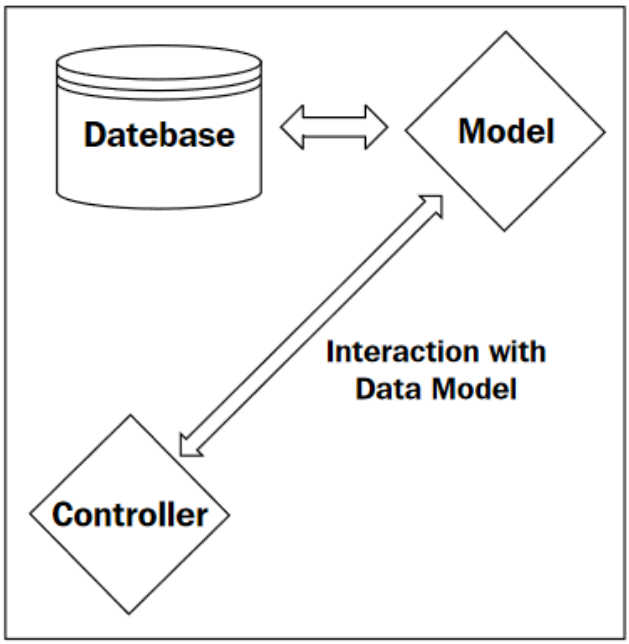


Figura 1 - El Modelo dentro del patrón MVC. Fuente: Laravel Design Patterns and Best Practices, Kilicdagi

Cabe recalcar que el Modelo no conoce de donde proviene la información y como es recibida.

Algunos beneficios de usar este patrón de diseño son que los mecanismos de propagación del cambio nos aseguran que todas las vistas simultáneamente reflejen el estado actual de un Modelo en específico. Por otro lado, los cambios que se hagan únicamente a la interfaz de usuario se vuelven más fáciles de realizar.

-**Vista**: La Vista consiste en los aspectos de la lógica de la presentación como: plantillas, caché, etc. Es lo que es presentado al usuario.

- **Controlador:** El Controlador se puede definir como la capa lógica de la aplicación. El controlador procesa las consultas que vienen desde otros extremos (consultas de usuarios, o de APIs), hace llamadas a los métodos correspondientes, hace las primeras validaciones, maneja la lógica de la consulta

y devuelve la información necesaria a la vista correspondiente o redirige al usuario a otra ruta.

El patrón Modelo-Vista-Controlador en Laravel

En Laravel hay algunas responsabilidades del Modelo que han sido separadas en otras clases, por ejemplo, la clase 'Validation' y la capa de conexión a bases de datos tiene su propia clase para cada driver de base de datos, esto convierte a los Modelos en Laravel extensibles, divisible en módulos y fácil de aplicar pruebas.

Laravel viene con una versión modificada del MVC tradicional, a diferencia de este, en Laravel la vista puede comunicarse con el controlador y el controlador con el modelo. (Kilicdagi, 2014). Ver figura 2.

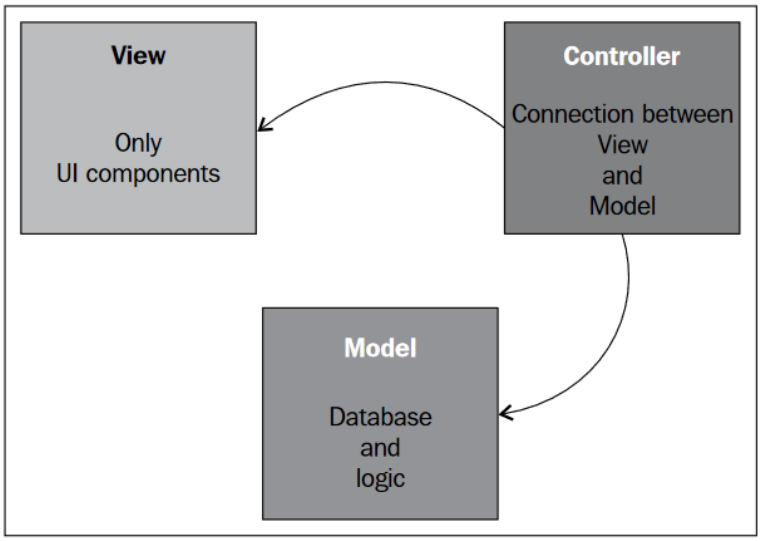


Figura 2 - MVC modificado de Laravel. Fuente: *Laravel Design Patterns and Best Practices, Kilicdagi*

Las vistas en Laravel se gestionan a través del gestor de plantillas Blade. Blade es un motor de plantillas simple y a la vez poderoso. A diferencia de otros motores de plantillas populares de PHP, Blade no impide utilizar código PHP plano en las vistas. De hecho, todas las vistas de Blade son compiladas en código PHP plano y almacenadas en caché hasta que sean modificadas, lo que significa que Blade no

añade sobrecarga a la aplicación. Los archivos de las vistas de Blade tienen la extensión ‘.blade.php’ y son usualmente almacenados en el directorio ‘resources/views’ (Laravel documentation, 2021).

Modelado de requerimientos

Métodos basados en clases

Los métodos basados en clases para el modelado de requerimientos fabrican una representación de una aplicación para que pueda ser entendida por partes interesadas sin conocimientos técnicos. En la medida que los requerimientos se van definiendo y expandiendo, estos se convierten en especificaciones que pueden ser utilizadas por los ingenieros de software en la creación del diseño del sistema (Pressman & Maxim, 2012).

El modelado basado en clases representa los objetos que el sistema va a manipular, as operaciones (métodos) que serán aplicadas a los objetos para ejecutar la manipulación, relaciones (algunas de ellas de tipo jerárquica) entre los objetos, y las colaboraciones que ocurren entre las clases definidas.

Los elementos de un modelo basado en clases incluyen clases, objetos, atributos, operaciones y modelado clase-responsabilidad-colaborador (CRC).

Modelado clase-responsabilidad-colaborador (CRC)

El modelado clase-responsabilidad-colaborador provee un medio simple para identificar y organizar las clases que son relevantes para los requerimientos del sistema.

Un modelo CRC es un grupo de tarjetas indexadas que representan clases. Las tarjetas tienen 3 divisiones. En la parte superior se escribe el nombre de la clase. En el cuerpo de la tarjeta a la izquierda se lista las responsabilidades de la clase y sus colaboradores a la derecha. (Ver figura 3).

Las responsabilidades son atributos y operaciones que son relevantes para la clase. Por otro lado, los colaboradores son aquellas clases necesarias para proveer a la clase con la información necesaria para cumplir la responsabilidad. En general, una colaboración consiste ya sea en una petición de información o una petición de alguna acción (Pressman & Maxim, 2012).

Clase: Automóvil	
Responsabilidades: - Acelerar en el momento que se presiona el acelerador. - Gira las llantas cuando se usa el volante. - Enciende las luces	Colaboradores: **- Volante** **-Llantas** **- Luminarias**

Figura 3 - Ejemplo de una tarjeta CRC. Fuente: Elaboración propia

Base de datos de un Sistema de información

En palabras de (Camps Paré, y otros, 2005): “una base de datos de un sistema de información es la representación integrada de los conjuntos de entidades del sistema de información y de sus interrelaciones.” Esta representación informática (o conjunto estructurado de datos) debe poder ser utilizado de forma compartida por muchos usuarios de distintos tipos.

Las principales ventajas de la implementación de una base de datos es que facilita la eliminación de la redundancia, asegurar la integridad, esto se consigue de varias formas, una de ellas es añadiendo reglas de integridad (o restricciones). Por otro lado, las bases de datos siguen las reglas de integridad del modelo de datos en el que, por ejemplo, no se permiten columnas repetidas.

Otra ventaja de la implementación de una base de datos es la seguridad, es decir el aseguramiento de la confidencialidad, las autorizaciones, los derechos de acceso, etc.

El modelo relacional

El modelo relacional es un modelo de datos y, como tal, tiene en cuenta los tres aspectos siguientes de los datos (Camps Paré, y otros, 2005):

1) La *estructura*, que debe permitir representar la información que nos interesa del mundo real.

2)La *manipulación*, a la que da apoyo mediante las operaciones de actualización y consulta de los datos.

3)La *integridad*, que es facilitada mediante el establecimiento de reglas de integridad; es decir, condiciones que los datos deben cumplir.

Metodologías de software

Un modelo de proceso de software es una representación simplificada de un proceso en específica. Cada modelo del proceso representa a otro desde una particular perspectiva y, por lo tanto, ofrece sólo información parcial acerca de dicho proceso (Sommerville, 2011).

También se puede entender una metodología como un conjunto de reglas a seguir que garantizan el éxito (Beck, 2005).

La metodología de Cascada

Este modelo tomas las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y, luego, los representa como

fases separadas del proceso, tal como especificación de requerimientos, diseño de software, implementación, pruebas, etc. (Sommerville, 2011).

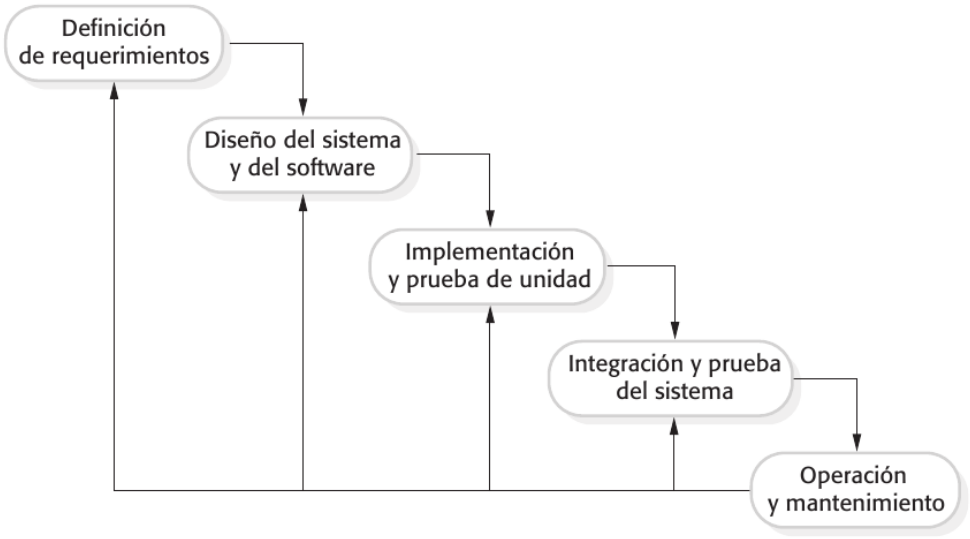


Figura 4 - Modelo de la cascada. Fuente: Sommerville, 2011

Las actividades fundamentales de desarrollo usando el método cascado son las siguientes:

1. Análisis y definición de requerimientos: Los servicios, las restricciones y las metas del sistema se establecen mediante consulta a los usuarios del sistema. Luego, se definen con detalle y sirven como una especificación del sistema.

2. Diseño del sistema y del software: El proceso de diseño de sistemas asigna los requerimientos, para sistemas de hardware o de software, al establecer una arquitectura de sistema global. El diseño del software implica identificar y describir las abstracciones fundamentales del sistema de software y sus relaciones.

3. Implementación y prueba de unidad: Durante esta etapa, el diseño de software se realiza como un conjunto de programas o unidades del programa. La prueba de unidad consiste en verificar que cada unidad cumpla con su especificación.

4. Integración y prueba de sistema: Las unidades del programa o los programas individuales se integran y prueban como un sistema completo para asegurarse de que se cumplan los requerimientos de software. Después de probarlo, se libera el sistema de software al cliente.

5. Operación y mantenimiento Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida, donde el sistema se instala y se pone en práctica. El mantenimiento incluye corregir los errores que no se detectaron en etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema e incrementar los servicios del sistema conforme se descubren nuevos requerimientos.

Metodología XP (Programación Extrema)

La metodología de programación extrema XP (del inglés *Extreme Programming*) fue el primero de los métodos Ágiles en volverse popular. Fue bautizado por su creador Kent Beck (2000) debido a que está metodología buscaba llevar a niveles “extremos” las prácticas reconocidas, como el desarrollo iterativo.

En la metodología XP los requerimientos se expresan como “Historias del usuario” que el equipo de desarrollo descompone en tareas y estima es esfuerzo y los recursos requeridos para implementar cada tarea. Esto involucra por lo general discusiones con el cliente para refinar los requerimientos. Entonces, para su

implementación, el cliente prioriza las historias y elige aquellas que pueden usarse inmediatamente para entregar apoyo empresarial útil.

Un precepto fundamental de la ingeniería de software tradicional es que se tiene que diseñar para cambiar. Esto es, deben anticiparse cambios futuros al software y diseñarlo de manera que dichos cambios se implementen con facilidad. Sin embargo, la programación extrema descartó este principio basado en el hecho de que al diseñar para el cambio con frecuencia se desperdicia esfuerzo. No vale la pena gastar tiempo en adicionar generalidad a un programa para enfrentar el cambio. Los cambios anticipados casi nunca se materializan y en realidad pueden hacerse peticiones de cambio diametralmente opuestas. Por lo tanto, el enfoque XP acepta que los cambios sucederán y cuando éstos ocurran realmente se reorganizará el software.

XP se rige bajo el principio de “Escribir las Pruebas primero”. Por tanto, tanto la prueba de aceptación como las unitarias son escritas antes que el mismo código. Y se promueve una continua integración (de preferencia diaria) del código de los diferentes programadores (Sommerville, 2011).

La metodología XP se basa en los siguientes principios:

1. Se basa en un desarrollo incremental, se apoya en pequeñas y frecuentes liberaciones del sistema. Los requerimientos se fundamentan en simples historias del cliente, o bien, en escenarios usados como base para decidir qué funcionalidad debe incluirse en un incremento del sistema.
2. El cliente debe estar incluido y comunicado constantemente con el equipo de desarrollo. El cliente, o su representante es el responsable de escribir las pruebas de aceptación para el sistema.
3. Se basa en la programación en pares, en la propiedad colectiva del código del sistema y en un proceso de desarrollo sustentable que no incluya jornadas de trabajo excesivamente largas.

4. Las liberaciones regulares del sistema a los clientes, escribir las pruebas primero, continuas refactorizaciones y la integración continua permiten la aceptación al cambio.
5. Las refactorizaciones del código constante mejoran la calidad del mismo y promueven la simplicidad, evitando repeticiones y líneas de código innecesarios, así como mejorando su legibilidad y mantenibilidad. Se usan diseños simples que no anticipan innecesariamente futuros cambios al sistema.

Los programadores trabajan en pares, por lo general uno escribe las pruebas para el otro y viceversa, para promover el conocimiento sobre la aplicación y la implementación a lo largo de todo el equipo.

Beck (2005) define al XP como una metodología de peso ligero. En XP solo se hace lo necesario para crear valor para el cliente. Beck considera que el XP es un es una metodología basada en abordar las limitaciones en el desarrollo de software. No se enfoca en la gestión del portafolio de proyectos, justificación financiera de proyectos, operaciones, marketing o ventas.

Si bien el XP puede tener relación con todas estas áreas, no las aborda de forma directa, Beck también afirma que la metodología en cuestión puede ser utilizada para proyectos y equipos tanto grandes como pequeños, y que ha sido puesta a prueba con éxito en ambos casos.

Una de las ideas principales del XP es adaptarse rápidamente a requerimientos imprecisos o muy cambiante. Por lo que XP es ideal para el cambio constante del mundo empresarial de nuestros días.

Valores XP

En la metodología XP se definen cinco valores que son fundamentales para el equipo durante un proyecto: comunicación, simplicidad, retroalimentación, valentía y respeto (Pressman, Ingeniería del Software, un enfoque práctico, 2010).

Comunicación: A fin de lograr la comunicación eficaz entre los ingenieros de software y otros participantes XP pone el énfasis en la colaboración estrecha pero informal (verbal) entre los clientes y los desarrolladores, en el establecimiento de “metáforas” para comunicar conceptos importantes, en la retroalimentación continua y en evitar la documentación voluminosa como medio de comunicación.

Simplicidad: XP restringe a los desarrolladores para que diseñen sólo para las necesidades inmediatas, en lugar de considerar las del futuro. El objetivo es crear un diseño sencillo que se implemente con facilidad en forma de código. Si hay que mejorar el diseño, se rediseñará en un momento posterior.

Retroalimentación: Se obtiene de tres fuentes: el software implementado, el cliente y otros miembros del equipo de software. Al diseñar e implementar una estrategia de pruebas eficaz, el software (por medio de los resultados de las pruebas) da retroalimentación al equipo ágil. XP usa la prueba unitaria como su táctica principal de pruebas. A medida que se desarrolla cada clase, el equipo implementa una prueba unitaria para ejecutar cada operación de acuerdo con su funcionalidad especificada. Cuando se entrega un incremento a un cliente, las historias del usuario o casos de uso que se implementan con el incremento se utilizan como base para las pruebas de aceptación. El grado en el que el software implementa la salida, función y comportamiento del caso de uso es una forma de retroalimentación. Por último, conforme se obtienen nuevos requerimientos como parte de la planeación iterativa, el equipo da al cliente una retroalimentación rápida con miras al costo y al efecto en la programación de actividades.

Valentía: Es necesaria la valentía para ir contra de los paradigmas tradicionales de desarrollo como son: hacer las pruebas al principio y no al final. Estar abiertos a cualquier cambio en la planificación y al aplicar la sencillez y no ceder ante la presión de programas funcionalidades para el futuro.

Respeto: Percibir a los demás miembros del equipo como personas y no sacar provecho de su trabajo ni extender a horarios de trabajo inhumanos.

El proceso de la metodología XP

La metodología de programación extrema tiene cuatro actividades principales: planeación, diseño, codificación y pruebas. (Ver Figura 5).

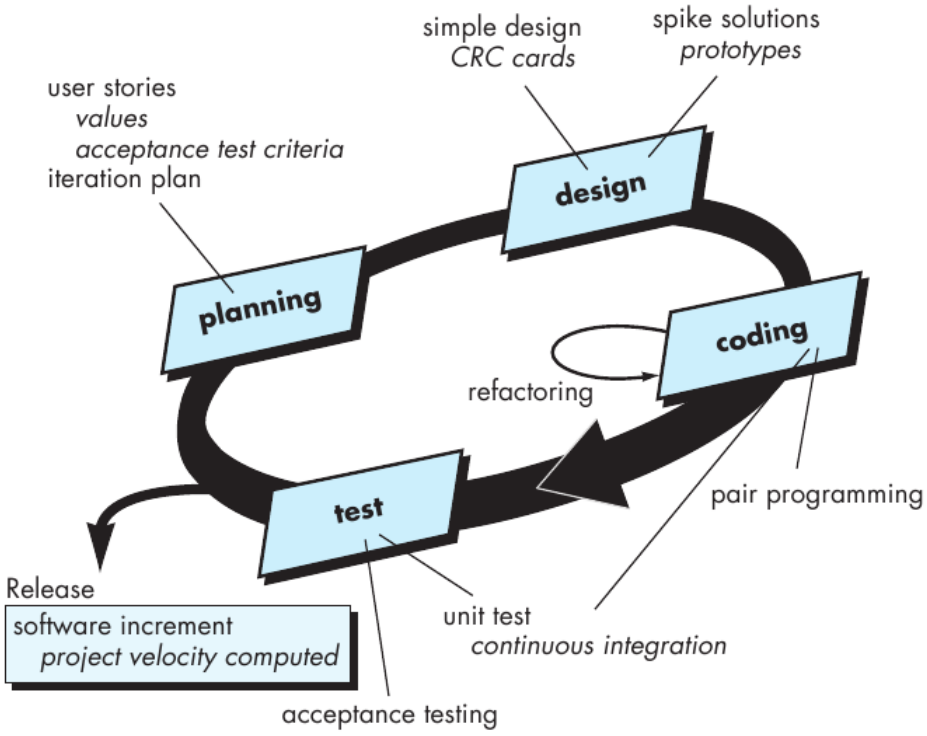


Figura 5 - El proceso de la programación extrema. Fuente: Pressman, 2015

Planeación: Esta fase consiste en el levantamiento de requerimientos del cliente. Con los requerimientos se crean las “Historias del usuario” que describen la salida necesaria, características y funcionalidad del software que se va a elaborar. Cada historia es escrita por el cliente y colocada en una tarjeta indizada. El cliente asigna un valor (es decir, una prioridad) a la historia con base en el valor general de la característica o función para el negocio.

Una vez diseñadas las tarjetas de historia, el equipo de desarrollo las descompone en tareas y estima el esfuerzo y los recursos requeridos para implementar cada tarea.

Diseño: El diseño en la metodología XP sigue rigurosamente el principio "*Mantenlo simple*" (*Keep it simple,* en inglés). XP no aconseja el desarrollo de funcionalidades mas complejas por que el desarrollador crea que se pueda necesitar después, se debe desarrollar lo que está en las tareas de historias de usuario, nada más ni nada menos.

XP estimula el uso de las tarjetas CRC como un mecanismo eficaz para pensar en el software en un contexto orientado a objetos. Las tarjetas CRC (clase-responsabilidad-colaborador) identifican y organizan las clases orientadas a objetos que son relevantes para el incremento actual de software. Las tarjetas CRC son el único producto del trabajo de diseño que se genera como parte del proceso XP.

Codificación: Una vez que las tareas han sido redactadas y que se han diseñado las tarjetas CRC, el equipo comienza a codificar, pero empezando siempre por las pruebas unitarias automatizadas. Una vez creada la prueba, el desarrollador está mejor capacitado para centrarse en lo que debe implementarse para pasar la prueba.

Una vez que el programador termina la tarea, se le aplica la prueba unitaria para comprobar que su código haga lo que se espera.

Además, la XP sugiere que el software debe refactorizarse continuamente. Esto significa que el equipo de programación busca posibles mejoras al software y las implementa de inmediato. Cuando un miembro del equipo observa un código que puede optimarse, realiza dichas mejoras, aun en situaciones donde no hay necesidad apremiante de ellas. Los ejemplos de refactorización incluyen la reorganización de una jerarquía de clases para remover un código duplicado, el

ordenamiento el cambio de nombre de atributos y métodos, y la sustitución de código con llamadas a métodos definidos en la librería de un programa.

Fowler (2000) define a la refactorización como:

> "Refactorización es el proceso mediante el cual se cambia un sistema de software en forma tal que no altere el comportamiento externo del código, pero sí mejore la estructura interna. Es una manera disciplinada de limpiar el código que minimiza la probabilidad de introducir errores (bugs). En esencia, cuando se refactoriza, se mejora el diseño del código después de haber sido escrito."

Pruebas: Las pruebas unitarias deben ser creadas usando un framework que les permita ser automatizadas y por la tanto puedan ser ejecutadas de forma sencilla y repetitiva. Para seguir una estrategia de pruebas de regresión y en consecuencia realizar las refactorizaciones con efectividad (Wells, 1999) .

Por otro lado, las pruebas de aceptación son especificadas por el cliente y enfocadas en las funcionalidades globales del sistema que son visibles y calificables por el cliente. Las pruebas de aceptación se derivan de las historias de usuario que han sido implementadas.

Metodología de pruebas

Desarrollo orientado por pruebas (TDD)

El desarrollo orientado por pruebas (TDD, del inglés Test Driven Development) lo define Astels (2003)como: un estilo de desarrollo donde se escriben las pruebas primero, estas pruebas determinan que código necesita ser escrito y en el que ningún código es puesto en producción si no tiene pruebas asociadas con él.

Mientras que Sommerville (2016) identifica los siguientes pasos en el TDD:

1. Se empieza identificando la funcionalidad requerida del incremento.
2. Se escribe una prueba para esta funcionalidad y se la implementa en una prueba automatizada.

3. Se ejecuta la prueba, junto a todas las otras pruebas que ya se hallan implementado. Como hasta este momento no se ha implementado la nueva funcionalidad, se espera que la prueba falle.
4. Se implementa la funcionalidad se refactoriza el código y se reejecutan las pruebas
5. Una vez que las pruebas se ejecutan con éxito, se avanza implementando las nuevas funcionalidades.

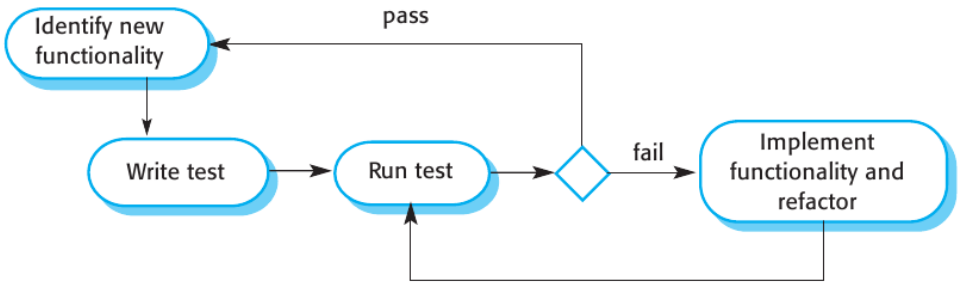


Figura 6 - Pasos del TDD, Fuente: Sommerville 2016

Elección de la metodología a utilizar

Para el presento proyecto fueron seleccionadas las metodologías XP y la programación orientada por pruebas (TDD). Por un lado, la metodología ágil XP fue seleccionada por la facilidad en adaptarse a los cambios durante todas las etapas del proyecto, ventaja que la diferencia de metodologías tradicionales como la metodología de cascada.

Ha sido elegida XP también por ser una metodología incremental, lo que permitirá hacer entregas pequeñas, esto permite conocer si el cliente está satisfecho con los avances, y corregir rápidamente en caso de ser necesario.

Otras de las ventajas por la que ha sido seleccionada la metodología XP es por su énfasis en las continuas refactorizaciones, lo que da como resultado un código de alta calidad y legibilidad.

También ha sido seleccionada la metodología XP por su familiaridad con la TDD, téngase en cuenta que la metodología TDD es una metodología derivada de los conceptos de XP.

Por su parte, la metodología TDD ha sido seleccionada por la alta cobertura de pruebas que genera sobre el código y por la calidad de las pruebas que a su vez sirven como documentación para futuros programadores que sean parte del proyecto.

Herramientas utilizadas

PHP

Es un lenguaje de programación de propósito general, que se adapta de forma particular al desarrollo web. Es un lenguaje rápido, flexible y pragmático (The PHP Group, 2021). PHP originalmente significaba Personal Home Page (Página personal), pero ahora significa el acrónimo recursivo PHP: Hypertext Preprocessor.

Si se busca adherir funcionalidad a una página HTML, PHP es ideal, porque permite hacer esto sin muchas líneas de código, a diferencia de C o Perl (The PHP Group, 2021). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final *<?php* y *?>* que permiten entrar y salir del "modo PHP".

La diferencia principal entre PHP y JavaScript es que PHP corre del lado del servidor. Es decir que ejecuta el código en el servidor la transforma a HTML y lo

envía al lado del cliente. Aunque también se puede utilizar como lenguaje de scripts desde línea de comandos, o escribir aplicaciones de escritorio (The PHP Group, 2021).

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

PHP es el lenguaje en el que están escritos sistemas de gestión de contenido (en inglés, Content Management System o CMS) como Wordpress, Joomla o la herramienta de gestión de aprendizaje Moodle. Cerca del 80% del internet está escrito en PHP (W3Techs, 2021).

Por la popularidad de este lenguaje, y por el excelente ambiente que posee (documentación, foros) ha sido elegido para este proyecto como lenguaje de backend.

JavaScript

JavaScript es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador. JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (Mozilla Developer Networks, 2021).

JavaScript es un dialecto del lenguaje ECMAScript inventado por Brendan Eich en la empresa Netscape (ECMA international, 2011). Javascript es un lenguaje que corre en el cliente y no requiere de ningún servidor web para ser ejecutado.

Javascript es responsable de la mayoría de interactividad en la página web como: cambio de imágenes, cambio del color del texto, mensajes de aviso y muchos otros. JavaScript comúnmente está incrustado en el HTML.

Laravel

Es un framework escrito en PHP con el patrón de diseño MVC (Modelo-Vista-Controlador). Fue diseñado para mejorar la calidad del software reduciendo costos al inicio del desarrollo, costos en el momento de mantenimiento y de mejorar la experiencia de trabajar con una aplicación proveyendo una sintaxis clara y todo un abanico de funcionalidades que pueden ayudar horas de implementación (McCool, 2012).

Laravel fue diseñado bajo la filosofía *"Convención sobre configuración".* Esto hace que haga predicciones sobre lo que se está tratando de conseguir con el objetivo de que el desarrollador consiga lo que desea con mucho menos código.

Se trata de framework de desarrollo con una curva de aprendizaje muy rápida y que maneja una sintaxis expresiva, elegante, con el objetivo de eliminar la molestia del desarrollo web facilitando las tareas comunes, como la autenticación, enrutamiento, sesiones y caché (Hostalia, 2020).

Laravel fue creado en el año 2011, con el paso del tiempo, ha ido ganando terreno a otros frameworks como Symfony o Zend framework. Para la versión de Laravel 4, el repositorio de Github del proyecto se volvió el repositorio más popular entre los frameworks de PHP (Bean, 2015).

Con su instalación ya se cuenta con componentes de pruebas unitarias, componentes de seguridad, un ORM (Object Relational Mapping) lo cual genera importante ahorro de tiempo y lo hace ideal para cualquier tipo de aplicación web.

Vue Js

Vue es un framework progresivo para crear interfaces de usuario. A diferencia de otros frameworks monolíticos está diseñado para ser adaptable de forma incremental. La librería principal está enfocada solamente en la capa de la "vista", es fácil de implementar e integrar con otras librerías o proyectos existentes. Por

otro lado, Vue es también capaz de generar *Single-Page-Applications (SPAs)* cuando es usado en combinación con herramientas modernas (vuejs.org, 2020).

Sass

Es un metalenguaje de Hojas de Estilo en Cascada (CSS). Es un lenguaje de script que es traducido a CSS, SassScript es el lenguaje de script en sí mismo. La sintaxis más reciente, SCSS, usa el formato de bloques como CSS. Este usa llaves para denotar bloques de código y punto y coma (;) para separar las líneas dentro de un bloque. (The Sass team, 2021)

Git

Git es un Sistema de control de versiones *open source*, diseñado para manejar desde pequeños a grandes proyectos con velocidad y eficiencia. Git es fácil de usar y tiene una pequeña huella sobre el proyecto (Git scm, 2020).

Git ha sido elegido debido a ser unas de las herramientas VCS con mayor popularidad en la actualidad. Y por su facilidad de vincularse a plataformas como GitLab o Github.

JetBrains PhpStorm

PhpStorm es un Entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Desarrolladores PHP, escrita sobre la plataforma de Intellij IDEA. Hereda todas las funcionalidades de Intellij IDEA relacionadas con la web como: edición en PHP, HTML, CSS, JavaScript, XML, sistema de manejo de versiones (VCS), y otras funcionalidades específicas del desarrollo web (Jetbrains, 2020).

El uso de un IDE con las funcionalidades de PhpStorm hace que sea mucho más fácil escribir código de calidad, por su detección de errores instantánea y ahorra tiempo de Desarrollo. Si bien PhpStorm normalmente es un software licenciado, Cuenta con un licenciamiento total para estudiantes por un tiempo de 4 años.

MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo (DB-engines, 2021). MySQL es también la base por defecto utilizada con Laravel.

Por este motivo ha sido elegida como SGBD a utilizarse en este proyecto, además MySQL posee un buen ambiente que incluye excelente documentación y soporte en foros en el momento de encontrar errores.

ORM (Asignación objeto-relacional)

El mapeo entre objetos y relaciones (ORM Object-Relational mapping). Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional.

Esto elimina la necesidad de crear una capa compleja para leer obtener información de un SGBD para convertirlos en objetos. Las Herramientas ORM también abstrae los detalles de la lógica del mapeo, los detalles de lectura y escritura, así como de los diferentes tipos de relaciones (uno-a-uno, uno-a-muchos).

En los motores de base de datos se dice que cada tabla es una relación, de ahí el nombre de base de datos relacionales, lo que implica que las tablas se encuentran relacionadas entre sí.

En una aplicación orientada a objetos se dice que tratamos con objetos y no con tablas. Cuando se agrega un registro a la tabla de personas, por ejemplo, en realidad decimos que agregamos un nuevo objeto “Persona”. Cuando decimos que un país está relacionado a varias personas, estamos diciendo que un objeto “País” contiene una colección de objetos “Persona”. Para esto, se crean clases que mapean cada relación de la base de datos y en lugar de interactuar

directamente con la base de datos, interactuamos con los objetos y el ORM se encargará de traducir lo necesario para hablar con la base de datos.

Con esto logramos una abstracción casi del 100% con relación al motor de base de datos, sin importar cuál sea, ya que hoy en día la mayoría de ellos se encuentran soportados por los ORM

Diferentes retos se presentan en el momento de intentar mapear objetos con tablas de una base de datos. Un desafío, por ejemplo, es intentar mapear estructuras de herencia de clases a tablas. Desde el punto de vista de la funcionalidad exiten diferentes aproximaciones para resolver este problema. A estas diferentes estrategias se les conoce como estrategiasa de mapeo. Cada estrategia tiene su diferente impacto en la calidad de la aplicación (Lorenz , Hesse, & Rudolph, 2021).

Eloquent ORM

Eloquent es el ORM incluido con Laravel, provee una implementación del patrón de diseño *Active Record*. En Eloquent cada tabla tiene su respectivo “Modelo” que es utilizado para interactuar con la base de datos. Y desde este modelo, Podemos hacer consultas o insertar datos a las tablas (Laravel documentation, 2021).

La ventaja de usar Eloquent en el campo de la seguridad, es que nos protege contra ataques de inyección de código SQL (Palacios, 2021).

Laragon

Es un ambiente de desarrollo portable, aislado, rápido y poderoso para PHP, Node.js, Python, Java, Go, Ruby. Es fácil de usar y fácil de extender. Laragon es ideal para crear y gestionar aplicaciones web modernas. Está enfocado en el rendimiento, estabilidad, simplicidad, flexibilidad y libertad (Laragon Documentation, 2021).

Laragon soporta los servidores web Apache y NGINX así como varios gestores de bases de datos como MySql, MongoDB, PostgreSQL, etc.

PhpUnit

Es un framework de código abierto para pruebas automatizadas para PHP, es una instancia de la arquitectura xUnit. PhpUnit se creó con la idea de que cuanto antes se encuentran los errores en el código, es más fácil y más rentable corregirlos. PHPUnit utiliza assertions para verificar que el comportamiento de una unidad de código es el esperado (Bergmann, 2006).

Comercio electrónico

Parkin (2014) en su libro Digital Marketing: Strategies for Online Succes. Define reglas para aplicar en el comercio electrónico sin importar el tamaño o la escala de la empresa:

1. No empezar hasta que se tenga una visión clara y un propósito del negocio.

2. Ver las cosas desde el punto de vista del consumidor, en otras palabras, ser "*cliente-céntrico".*

3. No introducir una reestructuración a las técnicas de comercio electrónico de forma radical, sino que los cambios sean en paralelo y de forma paulatina.

4. Usar herramientas de gestión de proyectos.

5. No percibir la aplicación de comercio electrónico como una inversión única, es decir que hay que estar pendiente cada año en mantenimiento, actualizaciones, etc.

6. Explorar nuevos consumidores y nuevos mercados.

Marco Metodológico

Capítulo 2

Tipo de investigación

La metodología de investigación será de naturaleza exploratoria y descriptiva. La investigación exploratoria permitirá conocer detalladamente y a fondo los procesos que se llevan actualmente dentro del proceso comercialización, para poder construir una solución que satisfaga las necesidades de la empresa.

La investigación descriptiva como afirma Salkind (1998) “se reseñan las características o rasgos de la situación o fenómeno objeto de estudio”. Este tipo de investigación nos permitirá conocer la situación actual de la empresa a través de levantamientos de información de los procesos que se manejan y así tener una visión más clara para profundizar en el tema.

Identificación y definición de variables

Existen diferentes tipos de variables entre los que destacan las variables dependientes e independientes. “Se denomina variable independiente a todo aquel aspecto, hecho situación, rasgo, etcétera, que se considera como la *causa de* una relación entre variables y variable dependiente al resultado o efecto producido por la acción de la variable independiente. (Bernal, 2010)

Variables independientes

El presente proyecto cuenta con dos variables independientes, la primera es la aplicación web de la tienda online para Quimpec Químicas Cia. Ltda. Y la segunda es el módulo de indicadores de ventas,

Variables dependientes

La variable dependiente es el modelo de información de ventas de la empresa, para la toma de decisiones gerenciales.

Universo y muestra

Universo

El universo o población son: “el conjunto de todos los elementos a los cuales se refiere la investigación. Se puede definir también como el conjunto de todas las unidades de muestreo” (Fracica, 1988). El Universo para el presente proyecto tiene las siguientes características:

Alcance: Todo el tiempo de operatividad de Quimpec Químicas Cia. Ltda.

Tiempo: de 1993 al 2021.

Elementos: Toda la actividad productiva de Quimpec Químicas Cia. Ltda. Incluyendo su información contable.

Unidades de muestreo: Cada año fiscal en que la empresa ha operado.

Muestra

Es la parte de la población que se selecciona, de la cual realmente se obtiene la información para el desarrollo del estudio y sobre la cual se efectuarán la medición y la observación de las variables objeto de estudio (Bernal, 2010).

Para el presente proyecto se ha seleccionado como muestra las actividades productivas y comerciales de la empresa Quimpec en el año fiscal 2021-2022.

Métodos empleados e instrumentos de la investigación

Para el presente proyecto se han seleccionado dos tipos de métodos empíricos, los métodos empíricos sirven para obtener conocimiento de los hechos fundamentales que caracterizan a los fenómenos. Se utilizaron dos de estos métodos:

Entrevista: Se planea tener reuniones contantes con el cliente del producto, el Ing. Jorge Puertas, gerente general de Quimpec Químicas.

Observación: Se planea aplicar este método al funcionamiento de la empresa, a su manejo de las ventas y a la toma de decisiones.

Situación Actual

Diagnóstico de las ventas en Quimpec Químicas cia tlda

Quimpec se ha visto afectada por el decrecimiento económico del país provocado por la pandemia de COVID-19. La tabla 3 muestra el total de ventas del año 2019 y del 2020 año en el que comienza la pandemia.

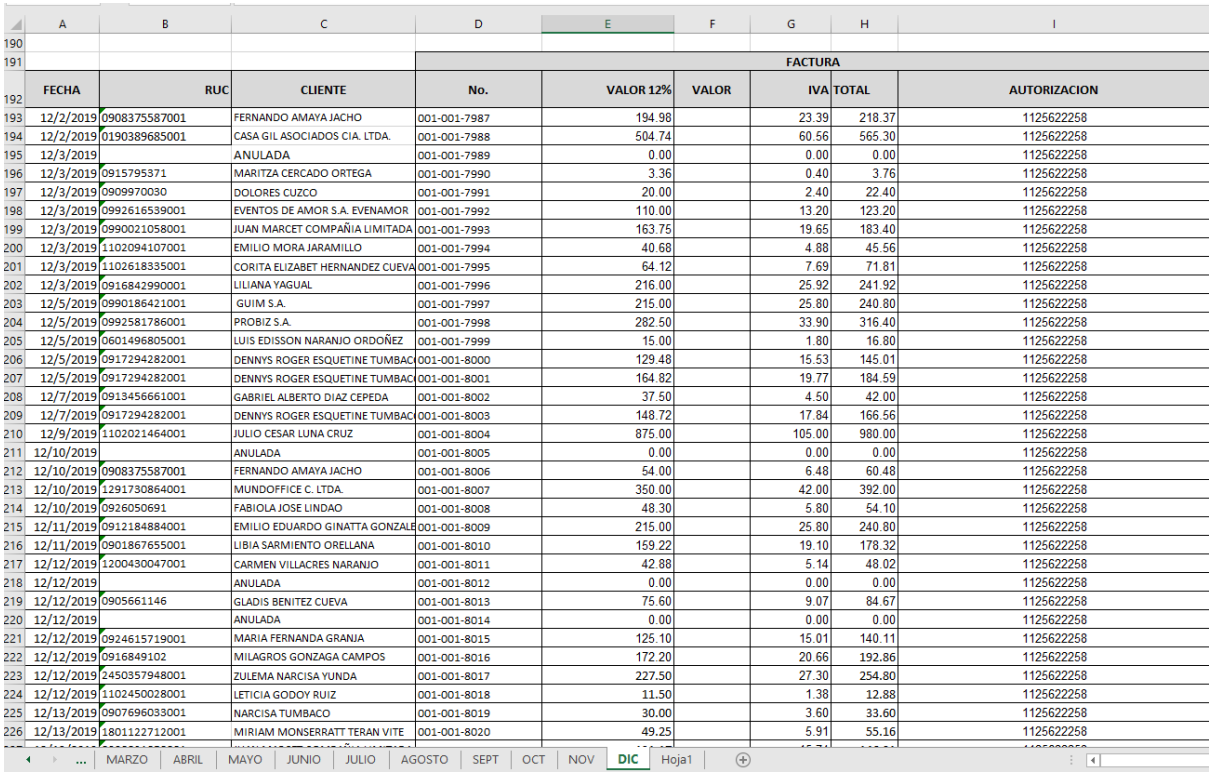
Año	Ventas Netas
2019	$ 131,041.05
2020	$ 57,204.07

Tabla 1 - Ventas Netas de Quimpec Químicas 2019,2020. Fuente: Contabilidad Quimpec Químicas

Lo cual nos muestra un decrecimiento de las ventas de un 56,3% entre el 2019 y 2020, es decir las ventas se redujeron en un poco más de la mitad. Por lo que se recomienda que Quimpec busque nuevos modelos de negocio relacionados al comercio electrónico para buscar un crecimiento en sus cifras de facturación.

Diagnóstico sobre los reportes y toma de decisiones

Actualmente en Quimpec la contabilidad se la lleva en Excel, con todas las limitaciones que esto conlleva. La figura 7 muestra una captura de pantalla de una hoja de Excel de la contabilidad del mes de diciembre 2019.



	A	B	C	D	E	F	G	H	I
190									
191				FACTURA					
192	FECHA	RUC	CLIENTE	No.	VALOR 12%	VALOR	IVA	TOTAL	AUTORIZACION
193	12/2/2019	0908375587001	FERNANDO AMAYA JACHO	001-001-7987	194.98		23.39	218.37	1125622258
194	12/2/2019	0190389685001	CASA GIL ASOCIADOS CIA. LTDA.	001-001-7988	504.74		60.56	565.30	1125622258
195	12/3/2019		ANULADA	001-001-7989	0.00		0.00	0.00	1125622258
196	12/3/2019	0915795371	MARITZA CERCADO ORTEGA	001-001-7990	3.36		0.40	3.76	1125622258
197	12/3/2019	0909970030	DOLORES CUZCO	001-001-7991	20.00		2.40	22.40	1125622258
198	12/3/2019	0992616539001	EVENTOS DE AMOR S.A. EVENAMOR	001-001-7992	110.00		13.20	123.20	1125622258
199	12/3/2019	0990021058001	JUAN MARCET COMPAÑIA LIMITADA	001-001-7993	163.75		19.65	183.40	1125622258
200	12/3/2019	1102094107001	EMILIO MORA JARAMILLO	001-001-7994	40.68		4.88	45.56	1125622258
201	12/3/2019	1102618335001	CORITA ELIZABET HERNANDEZ CUEVA	001-001-7995	64.12		7.69	71.81	1125622258
202	12/3/2019	0916842990001	LILIANA YAGUAL	001-001-7996	216.00		25.92	241.92	1125622258
203	12/5/2019	0990186421001	GUIM S.A.	001-001-7997	215.00		25.80	240.80	1125622258
204	12/5/2019	0992581786001	PROBIZ S.A.	001-001-7998	282.50		33.90	316.40	1125622258
205	12/5/2019	0601496805001	LUIS EDISSON NARANJO ORDOÑEZ	001-001-7999	15.00		1.80	16.80	1125622258
206	12/5/2019	0917294282001	DENNYS ROGER ESQUETINE TUMBAC	001-001-8000	129.48		15.53	145.01	1125622258
207	12/5/2019	0917294282001	DENNYS ROGER ESQUETINE TUMBAC	001-001-8001	164.82		19.77	184.59	1125622258
208	12/7/2019	0913456661001	GABRIEL ALBERTO DIAZ CEPEDA	001-001-8002	37.50		4.50	42.00	1125622258
209	12/7/2019	0917294282001	DENNYS ROGER ESQUETINE TUMBAC	001-001-8003	148.72		17.84	166.56	1125622258
210	12/9/2019	1102021464001	JULIO CESAR LUNA CRUZ	001-001-8004	875.00		105.00	980.00	1125622258
211	12/10/2019		ANULADA	001-001-8005	0.00		0.00	0.00	1125622258
212	12/10/2019	0908375587001	FERNANDO AMAYA JACHO	001-001-8006	54.00		6.48	60.48	1125622258
213	12/10/2019	1291730864001	MUNDOFFICE C. LTDA.	001-001-8007	350.00		42.00	392.00	1125622258
214	12/10/2019	0926050691	FABIOLA JOSE LINDAO	001-001-8008	48.30		5.80	54.10	1125622258
215	12/11/2019	0912184884001	EMILIO EDUARDO GINATTA GONZALE	001-001-8009	215.00		25.80	240.80	1125622258
216	12/11/2019	0901867655001	LIBIA SARMIENTO ORELLANA	001-001-8010	159.22		19.10	178.32	1125622258
217	12/12/2019	1200430047001	CARMEN VILLACRES NARANJO	001-001-8011	42.88		5.14	48.02	1125622258
218	12/12/2019		ANULADA	001-001-8012	0.00		0.00	0.00	1125622258
219	12/12/2019	0905661146	GLADIS BENITEZ CUEVA	001-001-8013	75.60		9.07	84.67	1125622258
220	12/12/2019		ANULADA	001-001-8014	0.00		0.00	0.00	1125622258
221	12/12/2019	0924615719001	MARIA FERNANDA GRANJA	001-001-8015	125.10		15.01	140.11	1125622258
222	12/12/2019	0916849102	MILAGROS GONZAGA CAMPOS	001-001-8016	172.20		20.66	192.86	1125622258
223	12/12/2019	2450357948001	ZULEMA NARCISA YUNDA	001-001-8017	227.50		27.30	254.80	1125622258
224	12/12/2019	1102450028001	LETICIA GODOY RUIZ	001-001-8018	11.50		1.38	12.88	1125622258
225	12/13/2019	0907696033001	NARCISA TUMBACO	001-001-8019	30.00		3.60	33.60	1125622258
226	12/13/2019	1801122712001	MIRIAM MONSERRATT TERAN VITE	001-001-8020	49.25		5.91	55.16	1125622258

Figura 7 - Contabilidad Quimpec Químicas diiembre 2019. Fuente: Contabilidad Quimpec Químicas

Las consultas que se le puede hacer a la contabilidad actual son bastante limitadas. No se puede conocer la provincia desde donde se hizo la venta ni los productos que han sido facturados.

Se recomienda a Quimpec Químicas migrar su información a una base de datos relacional y a partir de ella producir indicadores de ventas mensuales, por productos, por provincia o de cualquier tipo indicador que la gerencia necesite para mejorar su toma de decisiones.

Elección de la metodología a utilizar.

Para el presente proyecto se ha elegido la metodología XP por la facilidad que ofrece para adaptarse a los cambios que puedan darse en la fase de desarrollo. XP es recomendable para proyectos tanto grandes como pequeños. Y ha sido

elegida por la calidad de software que normalmente se produce siguiendo esta metodología.

Etapas de la Metodología XP

Planeación

En esta etapa se expresarán los requerimientos del cliente como “historias de usuarios”, luego estas historias de usuario serán descompuestas en tareas y a estas tareas se les asignará un valor. Se genera el plan de iteraciones en las que se asigna las tareas que se va a cumplir en casa entrega.

Diseño

Se crean las tarjetas CRC relacionadas con cada clase.

Codificación

Se escribe el código de las funcionalidades requeridas, siguiendo el desarrollo orientado por pruebas (escribir las pruebas primero). Por tanto, la prueba de aceptación como las unitarias son escritas antes que el mismo código.

También XP exige la constante refactorización del código, es decir, limpiar continuamente el código de repeticiones, variables innecesarias, etc. Para así mejorar la calidad y la legibilidad del mismo.

También en esta etapa se aconseja la continua integración del código de los diferentes miembros del equipo, además de la programación en pares. Estas dos últimas características no pueden ser aplicadas en el presente proyecto puesto que el equipo de desarrollo está conformado únicamente por el autor.

Aplicación de la metodología

Planeación

Historias del usuario

Título: Catálogo de producto	Número de historia: 1

	Relevancia (1-5): 5
Descripción: El usuario puede entrar a la página web a ver los productos, y guardar en un carrito los que le gustan.	

Tabla 2 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Título: Pago y cierre de venta	Número de historia: 2 Relevancia (1-5): 5
Descripción: El usuario debe poder pagar con tarjeta de crédito y que el programa me informe de los productos que ha comprado y de la dirección para poderle hacer el envío	

Tabla 3 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Título: Administración de los productos	Número de historia: 4 Relevancia (1-5): 4
Descripción: El programa debe permitirme a mi como administrador subir nuevos productos, subir nuevas fotos, crear nuevas categorías o borrar productos que ya no tenga en stock en que deje de fabricar.	

Tabla 4 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Título: Registro de clientes	Número de historia: 5 Relevancia (1-5): 3

Descripción: Los compradores deben poder crear su cuenta con sus datos de tal forma que no tengan que estar llenando toda la información, cada vez que quieran comprar. Pero si lo prefieren también debe haber la opción de comprar como “invitado” sin necesidad de registrarse.

Tabla 5 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Título: Códigos de descuento	Número de historia: 6 Relevancia (1-5): 4
Descripción: El programa debe permitirme crear promociones para los clientes. Estás promociones serán en forma de códigos de descuentos. Podré crear códigos que descuentes valores en efectivo o porcentaje de la compra	

Tabla 6 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Título: Gráficos con indicadores de ventas de la tienda	Número de historia: 7 Relevancia (1-5): 5
Descripción: Por medio de gráficos deseo ver los montos de ventas por provincia, ventas por meses y que productos son los que mas se han vendido, para en base a eso aumentar la producción	

Tabla 7 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Tareas

Tarea #1

Título: Maquetación de página web y tienda	Historia relacionada: 1
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 3
Responsable: Jorge David Peralta Godoy	
Descripción: Se necesita realizar la interfaz gráfica del inicio de la página web con un vínculo hacia la tienda. Lo anterior debe ser realizado con HTML y Sass Se necesita también la maquetación de la interfaz gráfica de la tienda, con sus respectivas partes: catálogo, carrito y pago.	

Tabla 8 - Tarea número 1

Tarea #2	
Título: Implementación de un arreglo de objetos que guarde los productos que el usuario guarde en el carrito.	Historia relacionada: 1
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 5
Responsable: Jorge David Peralta Godoy	
Descripción: Es necesario crear un controlador 'CartController' con el método para grabar en la memoria de la sesión los productos que el usuario guarde en su carrito de compras. Y conectar el método de ese controlador con los botones de la vista donde sea llamado.	

Tabla 9. Tarea 2

Tarea #3	
Título: Cargar imágenes de productos de prueba y configurar la paginación	Historia relacionada: 1
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 4
Responsable: Jorge David Peralta Godoy	

Descripción: Cargar imágenes de prueba y que se muestren de forma dinámica en la tienda y configurar la paginación para que solo cargue 9 productos por página y de esta forma mejorar el rendimiento puesto que no consultará a la base de datos todos los productos desde el primer momento. Para esto es necesario crear los Modelos 'Producto' y 'Categoría' con una relación entre ellos, donde cada producto pertenece a una o muchas categorías. Además de sus tablas en la base de datos por medio de migraciones.

Tabla 10 - Tarea 3

Tarea #4	
Título: Implementación de login para parte administrativa	Historia relacionada: 4
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 5
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe crear una ruta con un login para la sección de administración de la tienda online donde se encontrarán los CRUDS de: producto, categoría, cupones, usuarios y roles.	

Tabla 11 - Tarea 4

Tarea #5	
Título: Implementación de CRUD para Producto	Historia relacionada: 4
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 3
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe crear el mantenedor para producto donde se pueda ingresar: • Nombre del producto • Descripción • Precio • Imagen	

Para esto se debe crear un Modelo, un controlador y una tabla en la base de datos.

Tabla 12 - Tarea 5

Tarea #6	
Título: Implementación de CRUD para Categoría	Historia relacionada:
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 3
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe crear el mantenedor para Categoría donde se pueda ingresar el nombre de dicha categoría y la capacidad de asociarse con varios productos. Para esto se debe crear un Modelo, un controlador y una tabla en la base de datos.	

Tabla 13- Tarea 6

Tarea #7	
Título: Implementación de cupones de descuento	Historia relacionada: 6
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 2
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe implementar la opción de usar cupones de descuento, ya sea de un porcentaje o de un valor en dólares en específico. Para esto es necesario crear un controlador para cupones, un modelo y una tabla en la base de datos.	

Tabla 14 - Tarea 7

Tarea #8

Título: Registro y login de usuarios.	Historia relacionada: 5
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 4
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe crear el registro de usuarios para que lo compradores recurrentes no tengan que ingresar sus datos en cada compra. El registro tendrá los siguientes campos: • Nombre • Email • Contraseña Mientras que el login pedirá email y contraseña.	

Tabla 15 - Tarea 8

Tarea #9	
Título: Implementación de CRUD para Usuario y Roles	Historia relacionada: 4
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 4
Responsable: Jorge David Peralta Godoy	
Descripción: En la parte administrativa se debe crear un mantenedor para usuarios, así como a sus roles, que en el caso de este proyecto son dos: usuario y administrador.	

Tabla 16 - Tarea 9

Tarea #10	
Título: Guardar pedidos en base de datos	Historia relacionada: 2
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 3
Responsable: Jorge David Peralta Godoy	

Descripción: Se debe crear las tablas ‘ventas’ y ‘venta_items’ para almacenar la información de las ventas y los productos vendidos.

Tabla 17 - Tarea 10

Tarea #11	
Título: Integrar la tienda con PayPal	Historia relacionada: 2
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 3
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe integrar el sistema usando el SDK de PayPal para poder aceptar tarjetas de crédito	

Tabla 18 - Tarea 11

Tarea #12	
Título: Generar gráficos con indicadores de ventas	Historia relacionada: 7
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 5
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe realizar las APIs con la información de las ventas de la empresa para que luego sean consumidas por una librería de javascript para generar los siguientes gráficos: - Monto de ventas en dólares por provincia. - Monto de ventas en dólares por mes. - Productos vendidos el último mes.	

Tabla 19 - Tarea 12

Criterios de pruebas de aceptación

Funcionalidad Requerida	**Conforme**	**No conforme**
Mostrar el catálogo de productos		

Crear, editar y eliminar productos		
Crear, editar y eliminar categorías de productos		
Módulo de Mantenimiento de usuarios		
Crear pantalla de login para backend de tienda		
Implementar carro de compra		
Implementar el uso de tarjeta de crédito como método de pago por parte del cliente		
Registro de los clientes en la página, mediante la creación de cuenta de usuario.		
Crear códigos de descuento		

Tabla 20 - Pruebas de aceptación

Planificación de iteraciones

N°	Título de tarea	Iteración				
		1	2	3	4	5
1	Maquetación de página web y tienda	X				
2	Implementación de un arreglo de objetos que guarde los productos que el usuario guarde en el carrito.		X			
3	Cargar imágenes de productos de prueba y configurar la paginación		X			
4	Implementación de login para parte administrativa			X		

5	Implementación de CRUD para Producto			**X**		
6	Implementación de CRUD para Categoría			**X**		
7	Implementación de cupones de descuento			**X**		
8	Registro y login de usuarios.				**X**	
9	Implementación de CRUD para Usuario y Roles				**X**	
10	Guardar pedidos en base de datos					**X**
11	Integrar la tienda con PayPal					**X**
12	Crear gráficos con indicadores de ventas					**X**

Tabla 21 - Planificaión de iteraciones. Fuente: Elaboración propia

Plan de iteraciones

El tiempo total en el que se espera terminar la codificación es de 5 semanas, y será repartido por las iteraciones y las tareas según la siguiente tabla:

Iteración	**N° Tarea**	**Semanas**				
		1	**2**	**3**	**4**	**5**
1	1					
2	2					
	3					
3	4					

	5					
	6					
	7					
4	8					
	9					
5	10					
	11					
	12					

Tabla 22 - Plan de iteraciones

Codificación

Primera iteración

Como indica el plan de iteraciones, la primera entrega deberá incluir la Maquetación de página web y tienda.

Por lo tanto, Se crea el diseño de todas las vistas con HTML y Sass. Sin que esto incluya ningún tipo de funcionalidad. Ver figuras 8 y 9.

Las vistas elaboradas fueron: index, shop, product, cart y checkout. Los nombres de las vistas anteriores fueron nombrados en inglés para seguir el paradigma “Convención sobre Configuración” con el que está diseñado Laravel.

En la primera iteración no se crearon clases y por esta razón tampoco tarjetas CRC

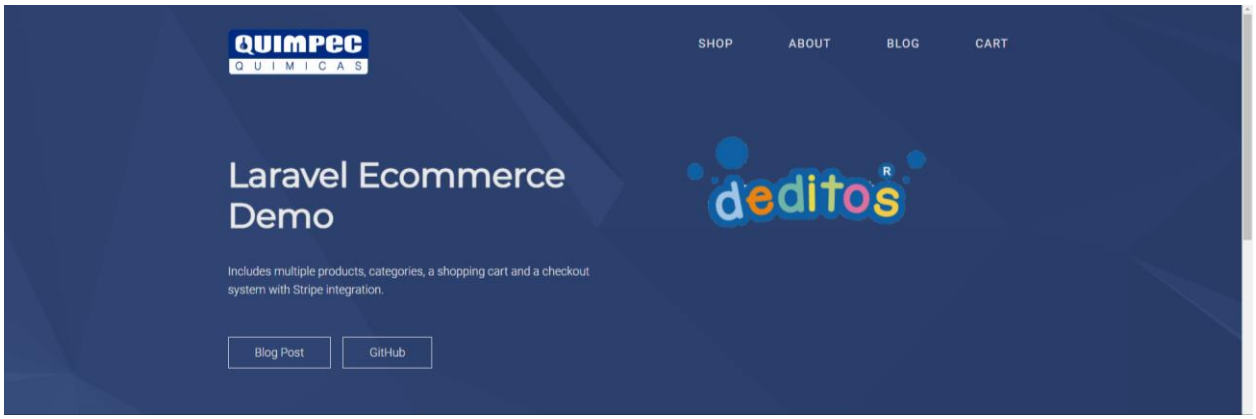


Figura 8 - Diseño del index. Fuente: Elaboración propia

Figura 9 - Diseño de vista 'Product'. Fuente: Elaboración propia

Segunda iteración

La segunda iteración tiene como objetivo la Implementación de un arreglo de objetos que guarde los productos que el usuario guarde en el carrito y cargar imágenes de productos de prueba y configurar la paginación

Para conseguir este objetivo, se creó los gráficos CRC para Producto, Categoría y Carrito y con ayuda de estos se procedió a agregar la funcionalidad al software.

Clase: Producto	
Responsabilidades: - Contener nombre, precio e imagen. - Es parte de una categoría. - Puede añadirse a un carrito - Muestra su precio en dólares	Colaboradores: **- Categoría** **-Carrito**

Tabla 23 - Diagrama CRC de clase Producto. Fuente: Elaboración propia

Clase: Categoría	
Responsabilidades: - Agrupas Productos. - Contener un nombre	Colaboradores: **- Producto**

Tabla 24 - Diagrama CRC de clase Categoría. Fuente: Elaboración propia

Clase: Carrito	
Responsabilidades: -Contener productos - Editar cantidad de productos productos. - Remover productos.	Colaboradores: **- Producto**

Tabla 25 - Diagrama CRC de clase Carrito. Fuente: Elaboración propia

Luego se creó las pruebas automatizadas para guardar y actualizar productos en el carrito. Ver figuras 10 y 11. Para el total de las pruebas realizadas en esta etapa ver Anexos.

```php
class CartTest extends TestCase
{
    use RefreshDatabase;

    /** @test */
    public function se_puede_aumentar_producto_al_carrito () //unit
    {
        $producto = factory(Product::class)->create();
        $cantidad = 1;
        Cart::instance('testing')->add($producto->id, $producto->name, $cantidad, $producto->price);

        $this->assertSame( $cantidad, Cart::instance('testing')->count() );

        Cart::instance('default');
    }
```

Figura 10 - Prueba para agregar productos al carrito. Fuente: Elaboración propia

```php
/** @test */
public function se_puede_actulizar_cantidad_del_carrito()
{
    $this->withoutExceptionHandling();
    $producto = factory(Product::class)->create();

    Cart::instance('testing');

   $this->post( uri: 'cart', [
            "id" => $producto->id,
            "name" => $producto->name,
            "price" => $producto->price,
    ]);

     $carrito =  Session::get('cart');
     $coleccion_carrito = $carrito["testing"];

    $response = $this->patch( uri: '/cart/'.$coleccion_carrito->first()->rowId , [
         'quantity' => 3
    ]);

    $response->assertSuccessful();
    Cart::instance('default');
}
```

Figura 11 - Prueba para verificar que no se puede actualizar los productos del carrito.

Fuente: Elaboración propia

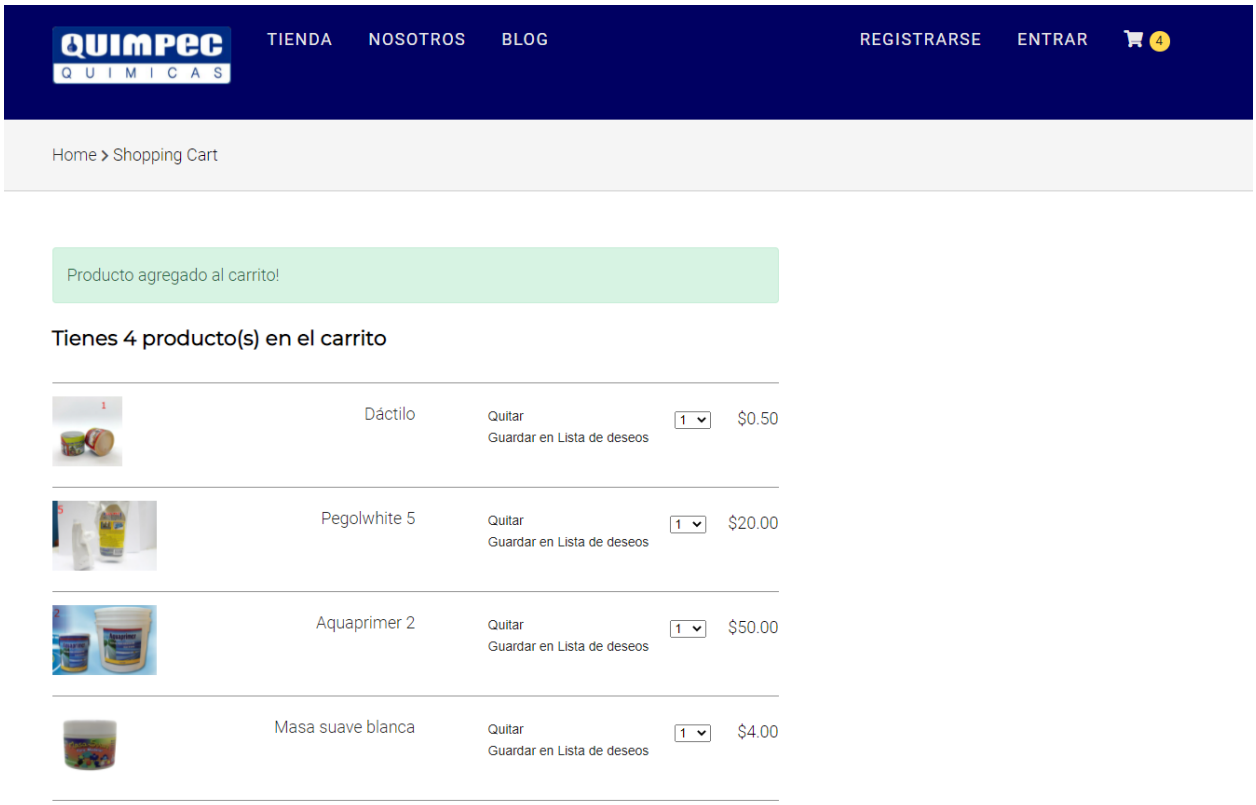


Figura 12 - Captura de pantalla de la funcionalidad de carrito. Fuente: Elaboración propia

Para finalmente comprobar que las pruebas estén pasando. Ver figura 13.

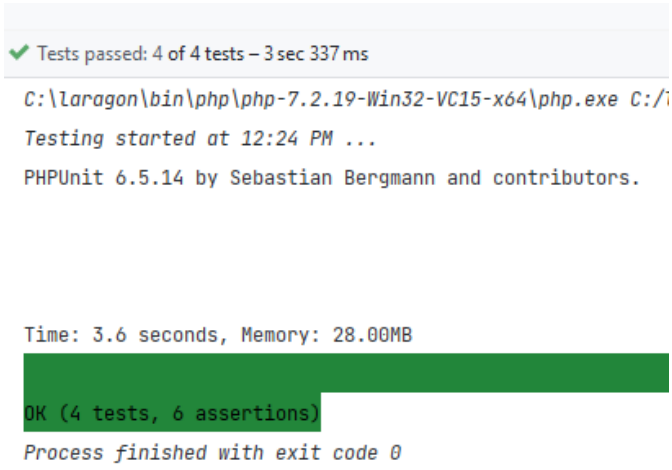


Figura 13 - Todas las pruebas pasando, después de la segunda iteración. Fuente: Elaboración propia

Tercera iteración

En la tercera iteración la planificación indica que debe implimentarse el login de la parte administrativa. Además de CRUDs para Producto, Categorías y cupones de descuento

Se escribieron las pruebas para el login administrativo. Ver figura 14 y 15.

```php
class AdminLoginTest extends DuskTestCase {
    use DatabaseMigrations;

    /** @test */
    public function se_puede_ingrear_al_backend_como_admin() {
        factory(Role::class)->create([
                'name' => 'admin',
                'display_name' => 'Administrador'
        ]);
        $admin = factory(User::class)->create([
                'role_id' => '1'
        ]);

        $this->browse(function (Browser $browser) use ($admin) {
            $browser->loginAs($admin)
                    ->visit( url: '/admin')
                    ->assertSee( text: 'Tablero');
        });

    }
```

Figura 14 - Prueba automatizada para entrar al login administrativo Fuente:

Elaboración propia

```php
    /** @test */
    public function no_se_puede_ingrear_al_backend_si_no_es_admin() {
        factory(Role::class)->create([
                'name' => 'admin',
                'display_name' => 'Administrador'
        ]);

        factory(Role::class)->create([
                'name' => 'user',
                'display_name' => 'Usuario normal'
        ]);
        factory(User::class)->create([
                'role_id' => '1'
        ]);

        $user = factory(User::class)->create([
                'role_id' => '2'
        ]);

        $this->browse(function (Browser $browser) use ($user) {
            $browser->loginAs($user)
                    ->visit( url: '/admin')
                    ->assertDontSee( text: 'Tablero');
        });
    }
}
```

Figura 15 - Prueba automatizada para entrar a la administración solo si es admin

Fuente: Elaboración propia

Luego se procedió a escribir en código la funcionalidad de login para la sección administrativa. La figura 16 muestra una captura de pantalla de dicha funcionalidad.

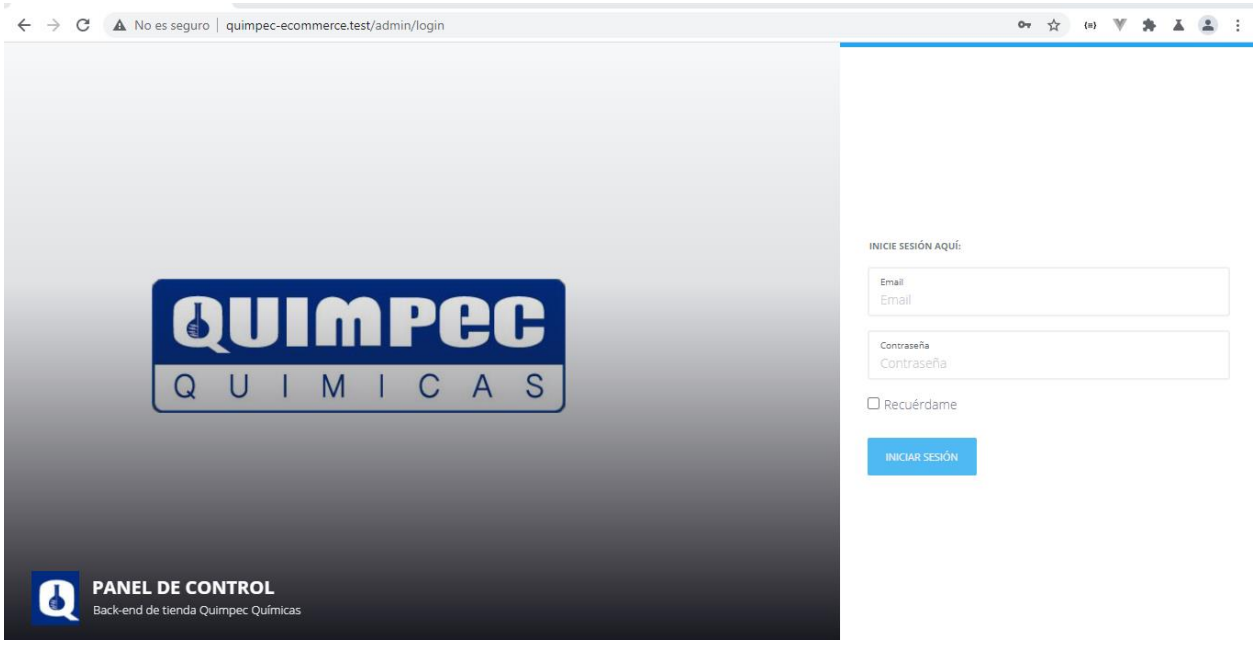


Figura 16 - Captura de pantalla del login administrativo Fuente: Elaboración propia

También en lo planificado en esta iteración se encontraba la Elaboración de CRUDs apara los Modelos de Producto, Categoría y Cupones. Siguiendo los principios de XP se procedió a escribir las pruebas automatizas primero. (Ver figuras 17 y 18). Para la totalidad de las pruebas ver los Anexos.

```php
class ProductoTest extends DuskTestCase
{
    use DatabaseMigrations;

    /** @test */
    public function se_puede_ver_lista_de_productos_creados_en_panel_de_control() {
        factory(Role::class)->create([
                'name' => 'admin',
                'display_name' => 'Administrador'
        ]);
        $admin = factory(User::class)->create([
                'role_id' => '1'
        ]);

        $this->browse(function (Browser $browser) use ($admin) {
            $browser->loginAs($admin)
                    ->visit( url: '/admin/products')
                    ->assertSee( text: 'Productos');
        });
    }
```

Figura 17 - Prueba automatizada para ver listado de productos Fuente: Elaboración propia

```php
/** @test */
public function se_puede_crear_un_nuevo_producto() {
    factory(Role::class)->create([
            'name' => 'admin',
            'display_name' => 'Administrador'
    ]);
    $admin = factory(User::class)->create([
            'role_id' => '1'
    ]);

    $this->browse(function (Browser $browser) use ($admin) {
        $browser->loginAs($admin)
                ->maximize()
                ->visit( url: '/admin/products/create')
                ->type( field: 'name',  value: 'Dactilo')
                ->type( field: 'slug',  value: 'dactilo')
                ->type( field: 'price',  value: 100);

                $browser->click( selector: 'button[type="submit"]')
                ->assertSee( text: 'Producto creado');
    });

}
}
```

Figura 18 - Prueba automatizada para ver crear nuevo producto Fuente: Elaboración propia

Se crearon los CRUDS para el Modelo ‘Product’, ‘Categoría’ y ‘Cupon’. En la figura 19 se muestra una captura del CRUD para Productos.

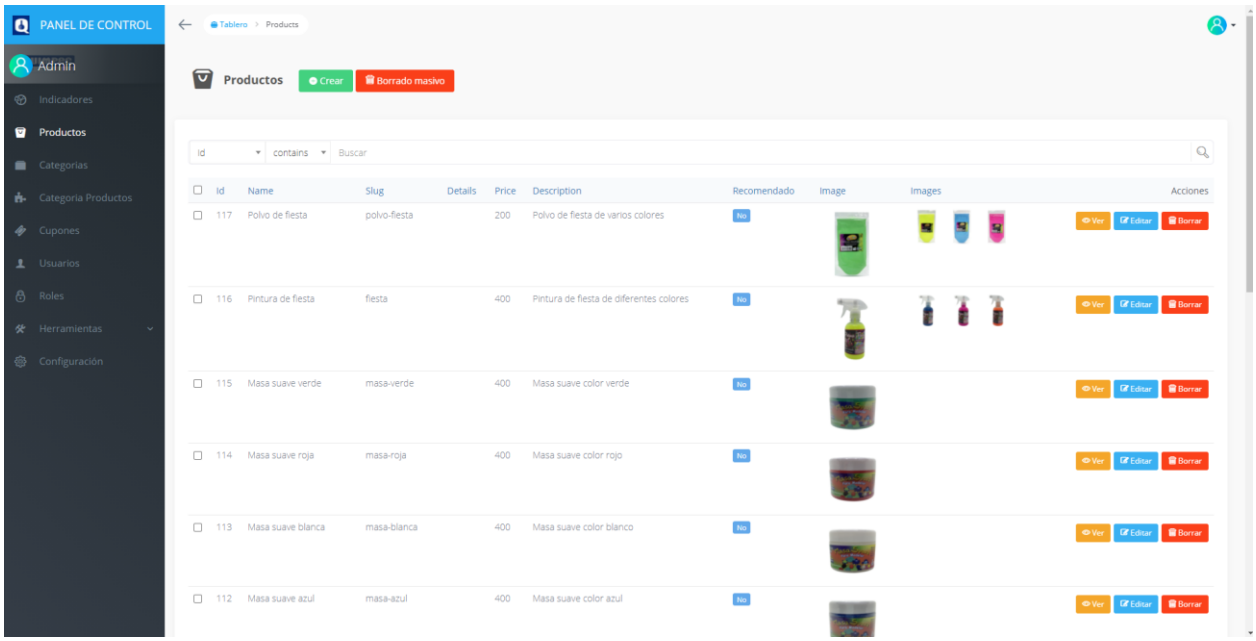


Figura 19 - Prueba automatizada para ver crear nuevo producto Fuente: Elaboración propia

Para finalmente comprobar que las pruebas escritas en un principio estén pasando.

Figura 20 - Todas las pruebas pasando después de la iteración Fuente: Elaboración propia

Nuevo requerimiento

En este punto del desarrollo, se mostró al propietario del software el resultado del último incremento, para lo que El Ing. Jorge Puertas dio a conocer un nuevo requerimiento junto con su historia de usuario (ver tabla 28). Lo cual no es algo que fuera de lugar, al contrario, es algo bastante común al momento de desarrollar software, y es de hecho el motivo de creación de los métodos ágiles: adaptarse con agilidad a los cambios que se presenten en el camino.

Título: Mostrar Varias fotos por producto	Número de historia: 8 Relevancia (1-5): 4
Descripción: El programa debe permitir a los usuarios ver fotos de todos los colores de mis productos (como una pequeña galería) y permitirles agrandar las fotos en los que ellos den click.	

Tabla 26 - Elaboración: Ing. Jorge Puertas, Gerente Quimpec

Dicha historia se tradujo en la siguiente Tarea:

Tarea #13	
Título: Creación de galería de imágenes en vista 'Product'	Historia relacionada: 8
Tipo de tarea: Desarrollo	Puntos estimados (1-5): 3
Responsable: Jorge David Peralta Godoy	
Descripción: Se debe cambiar cambiar la imagen mostrada por medio de Blade a una mini galería creada por medio de un componente de vue.js que dé la opción de cambiar las imágenes.	

Tabla 27 - Tarea 13

Lo anterior llevó a la necesidad de replanificar las iteraciones. Por motivos de dificultad se decidió mover la tarea 12 (gráficos de ventas) de la iteración número 6 a la siguiente iteración. Los cambios realizados se resumen en la siguiente tabla:

Iteración	N° Tarea	Semanas		
		4	5	6
4	12			
	13			
5	8			
	9			
6	10			
	11			

Tabla 28 - Nueva tabla de iteraciones

El nuevo requerimiento extendía el tiempo de desarrollo a un total de 6 semanas, esto se le comunicó al dueño del producto, y el manifestó que estaba de acuerdo con esta nueva planificación, puesto que la funcionalidad en cuestión (pequeña galería) era muy importante para el proyecto desde su punto de vista.

Cuarta iteración

Según la nueva planificación la cuarta iteración consiste en crear los gráficos con indicadores de ventas para el administrador y la creación de la galería de imágenes.

En el caso de la galería, se debe mostrar varias fotos de un mismo producto. Lo cual se consiguió con el componente ‘QuimpecProducto.vue’

QuimpecProducto.vue

```
27
28        name: "QuimpecProducto",
29        data() {
30          return {
31            imgUrlPrefix: '/storage/',
32            array_images: Array,
33            urlImagenMostrada: String,
34            productoElejidoActual: {
35              type: Number
36            }
37          }
38        },
39        props: {
40          producto: Object,
41        },
42        computed: {
43          imagenMostrada: function () {
44            return this.imgUrlPrefix + this.producto.image;
45          }
46        },
47        mounted: function () {
48          this.array_images = JSON.parse(this.producto.images);
49          this.urlImagenMostrada = this.imagenMostrada;
50          this.productoElejidoActual = -1;
51
52          console.log(this.producto)
53        },
54        methods: {
55          mostrarImagen(i) {
56            if (i >= 0) {
57              this.urlImagenMostrada = this.imgUrlPrefix + this.array_images[i];
58              this.productoElejidoActual = i;
59            } else if (i < 0) {
60              this.urlImagenMostrada = this.imgUrlPrefix + this.producto.image;
61              this.productoElejidoActual = i;
62            }
```

Figura 21 - Script del componente QuimpecProducto. Fuente: Elaboración propia

Finalmente se consiguió la funcionalidad esperada.

Figura 22 - Captura de pantalla de la galería de productos. Fuente: Elaboración propia

También en esta iteración se busca realizar los gráficos de Monto de ventas en dólares por provincia, Monto de ventas en dólares por mes y Productos vendidos el último mes. Para esto se crearon 3 APIs: 'venta-dolares-por-provincia', 'venta-dolares-por-mes', 'productos-vendidos-ultimo-mes'. Para ser consumidas por medio de axios y con la ayuda de la librería ChartJs poder generar los gráficos.

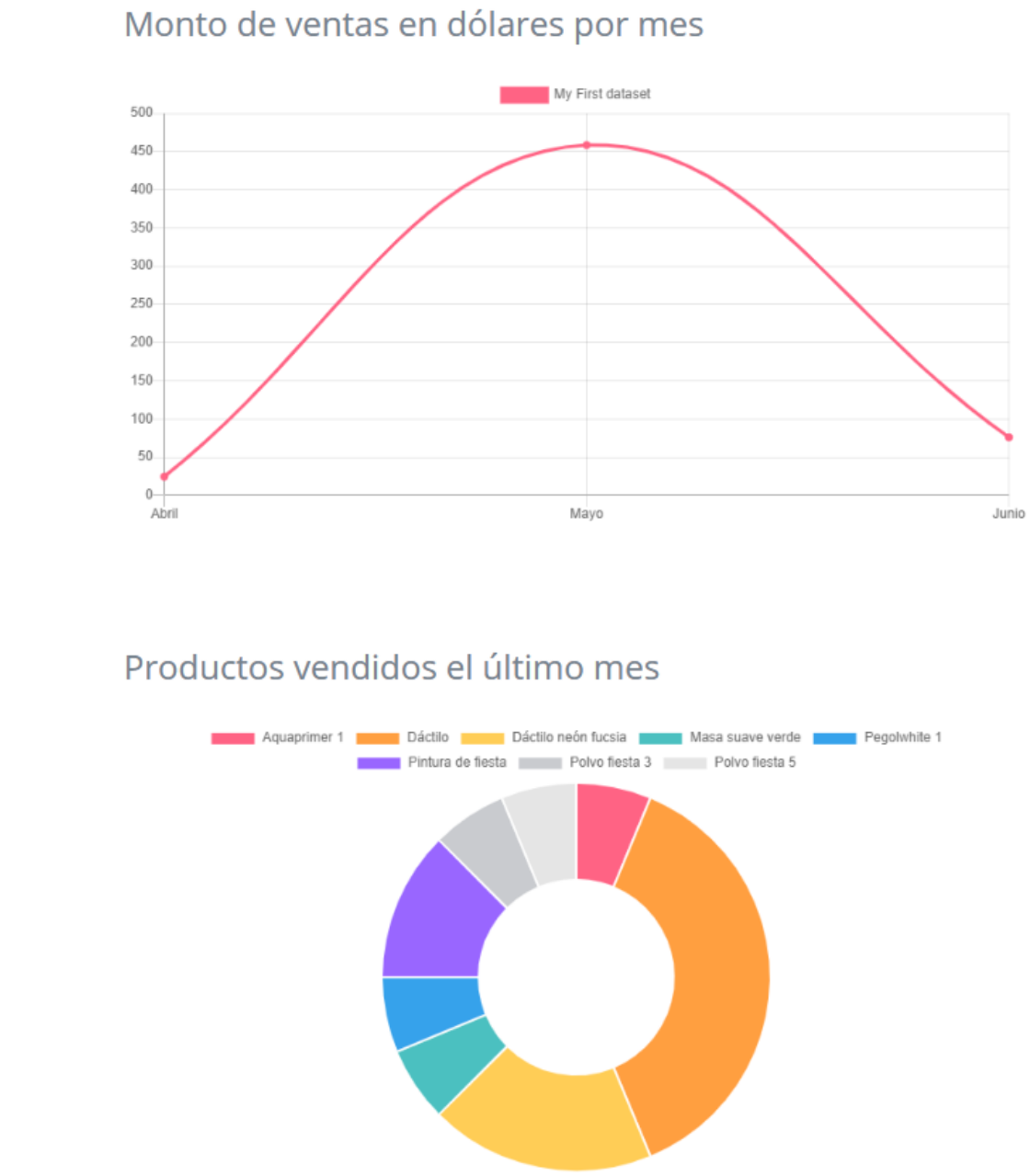


Figura 23 - Captura de pantalla gráficos de indicadores de venta. Fuente: Elaboración propia

Quinta iteración

La quinta iteración tiene como finalidad realizar las tareas 8 y 9 las cuales incluyen un registro y login de usuarios, y CRUDs para usuarios y sus roles desde la parte administrativa.

Algunas de las pruebas escritas para estas funcionalidades se muestran en las figuras 24 y 25.

```php
/** @test */
public function usuario_no_puede_ingresar_con_password_incorrecto()
{
    $user =  factory(User::class)->create([
            'password' => Hash::make($password = 'secreto'),
    ]);

    $response = $this->from($this->loginGetRoute())->post($this->loginPostRoute(), [
            'email' => $user->email,
            'password' => 'contraseña-incorrecta',
    ]);

    $response->assertRedirect($this->loginGetRoute());
    $response->assertSessionHasErrors( keys: 'email');
    $this->assertTrue(session()->hasOldInput( key: 'email'));
    $this->assertFalse(session()->hasOldInput( key: 'password'));
    $this->assertGuest();
}
```

Figura 24 - Prueba para comprobar que no se pueda ingresar con contraseña incorrecta. Fuente: Elaboración propia

```
class RegisterTest extends DuskTestCase
{
    use DatabaseMigrations;

    /** @test */
    public function se_puede_registrar_un_nuevo_usuario() {

        $this->browse(function (Browser $browser)  {
            $browser->visit( url: '/register')
                    ->type( field: 'name',  value: 'Jorge')
                    ->type( field: 'email',  value: 'jorge@jorge.com')
                    ->type( field: 'password',  value: 'secreto')
                    ->type( field: 'password_confirmation',  value: 'secreto')
                    ->click( selector: 'button[type="submit"]')
                    ->assertSee( text: 'Categorías');
        });

    }
}
```

Figura 25 - Prueba para comprobar que se pueda registrar nuevo usuario. Fuente: Elaboración propia

Se procedió a codificar la funcionalidad requerida, se muestran algunas capturas de pantalla al momento de terminar las tareas.

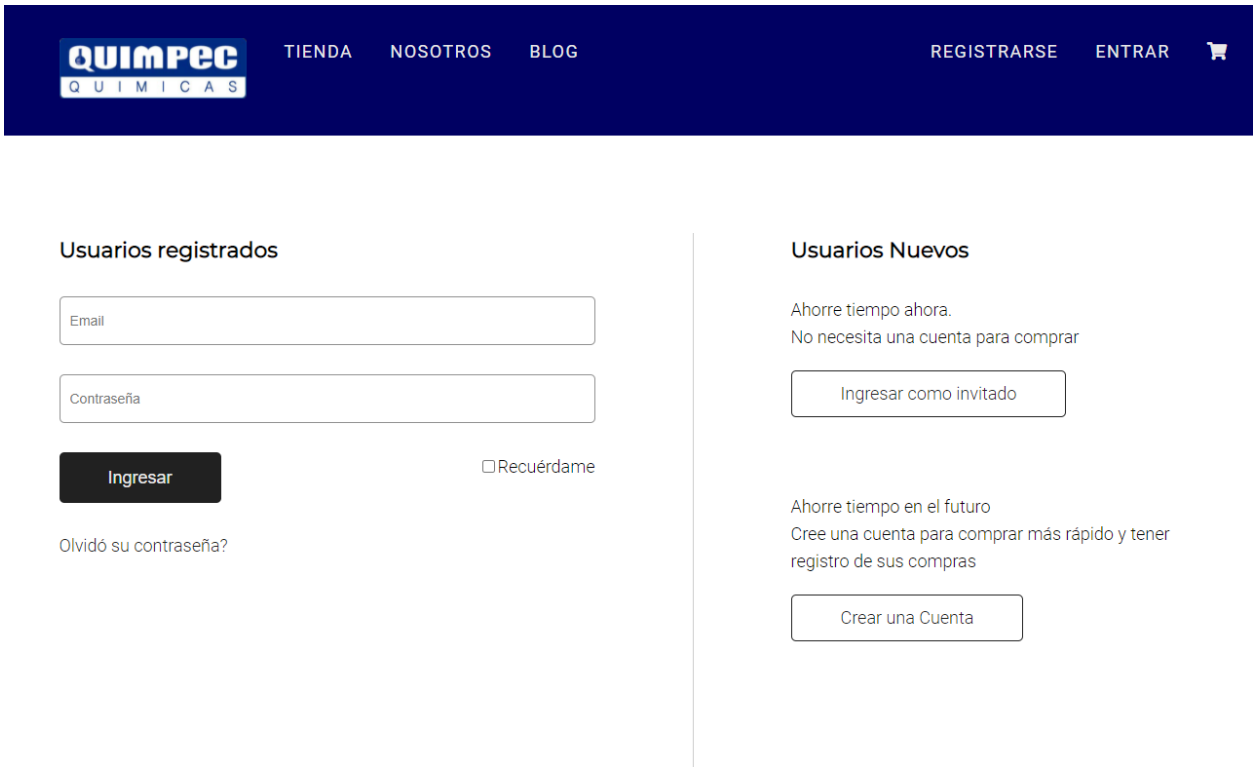


Figura 26 - Funcionalidad de login terminada. Fuente: Elaboración propia

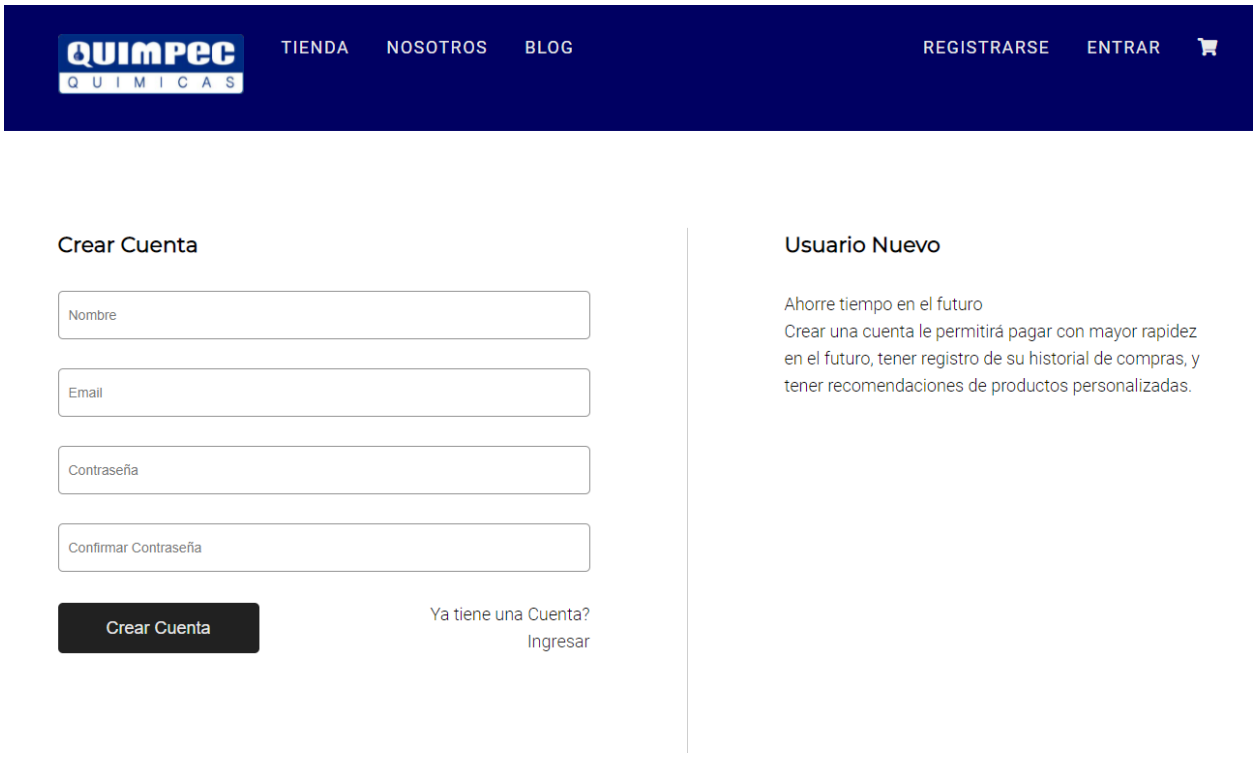


Figura 27 - Funcionalidad de registro de usuarios, terminada. Fuente: Elaboración propia

Finalmente se procedió a correr las pruebas y comprobar que todas están pasando.

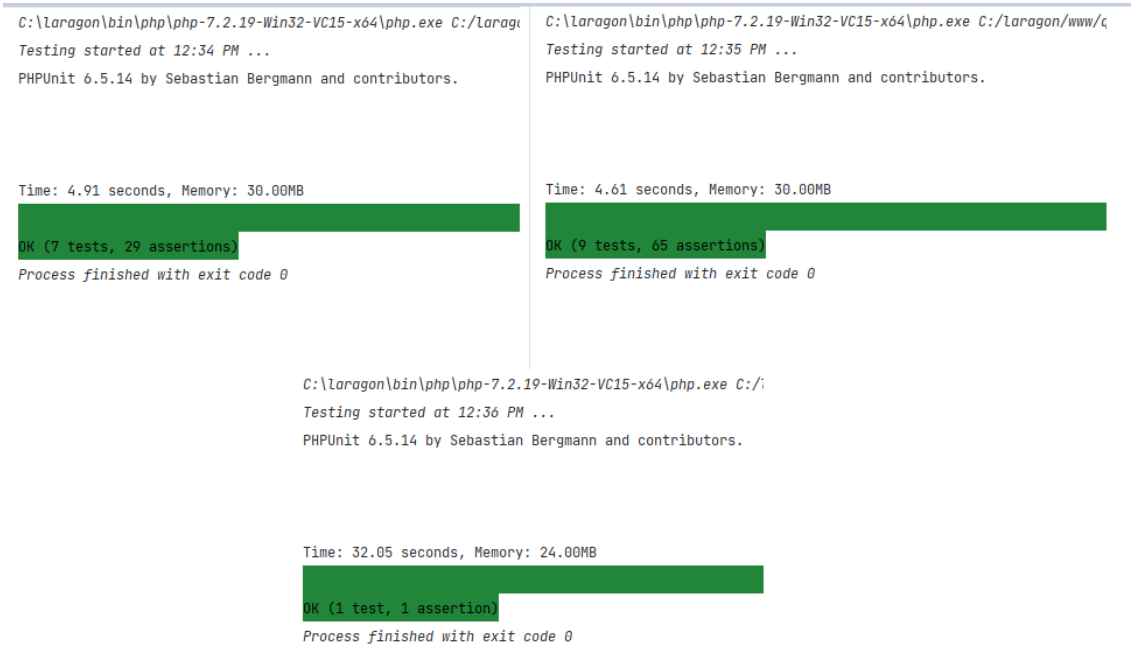


Figura 28 - Un total de 17 pruebas y 35 comprobaciones pasando. Fuente: Elaboración propia.

Sexta iteración

La sexta y última iteración tiene como finalidad realizar las tareas 10 y 11 que incluía la posibilidad de almacenar los pedidos que emita el cliente en la base de datos y la integración con el SDK de PayPal.

Una de las pruebas del API de ventas se muestra en la figura 29.

```php
/** @test */
function se_crea_una_nueva_venta_con_un_producto() {
    $this->withoutExceptionHandling();
    $producto = factory(Product::class)->create();
    $cantidad = 1;
    $provincia = 'Guayas';
    $ciudad = 'Guayaquil';
    $direccion = $this->faker->address;
    Cart::instance('testing')->add($producto->id, $producto->name, $cantidad, $producto->price);
    $cart = Cart::instance('testing')->content()->toArray();
    $response = $this->post( uri: 'api/venta', [
            'total' => Cart::instance('testing')->total(2, ".", ""),
            'provincia' => $provincia,
            'ciudad' => $ciudad,
            'direccion' => $direccion,
            'cantidad' => $cantidad,
            'cart' => $cart
    ]);
    $response->assertStatus( status: 201);
    $response->assertJsonStructure([
            'id', 'total', 'provincia', 'ciudad', 'direccion', 'created_at'
    ]);
    $response->assertJson([
            'total' => Cart::instance('testing')->total(2, ".", ""),
            'provincia' => $provincia,
            'ciudad' => $ciudad,
            'direccion' => $direccion,
    ]);
    $this->assertDatabaseHas( table: 'ventas', [
            'provincia' => $provincia,
            'total' => Cart::instance('testing')->total(2, ".", ""),
            'ciudad' => $ciudad,
            'direccion' => $direccion,
    ]);
    $this->assertCount( expectedCount: 1, VentaItem::all());
    Cart::instance('default');
```

Figura 29 - Prueba sobre el API Venta sobre la venta de un producto. Fuente: Elaboración propia.

Una vez escrita la prueba se procedió a implementar el API, para esto se crearon las migraciones de las tablas ‘venta’ y ‘venta_items’. Además de sus modelos y controladores. En la figura 30 se muestra el método ‘store’ del VentaController.

```php
class VentaController extends Controller {

    public function store(Request $request) {
        $data = $request->all();
        $venta = [];
        DB::transaction(function () use ($data, &$venta) {
            $venta = Venta::create([
                    'total' => $data['total'],
                    'provincia' => $data['provincia'],
                    'ciudad' => $data['ciudad'],
                    'direccion' => $data['direccion']
            ]);

            foreach ($data['cart'] as $item) {
                VentaItem::Create([
                        'id_venta' => $venta->id,
                        'id_producto' => $item['id'],
                        'cantidad' => $item['qty']
                ]);
            }

        });

        return response()->json($venta, status: 201);
    }
```

Figura 30 - Método 'store' del controlador de Venta. Fuente: Elaboración propia.

Por otro lado, la integración con Paypal se consiguió por medio de la SDK que provee paypal en su documentación oficial escrita en vue.js

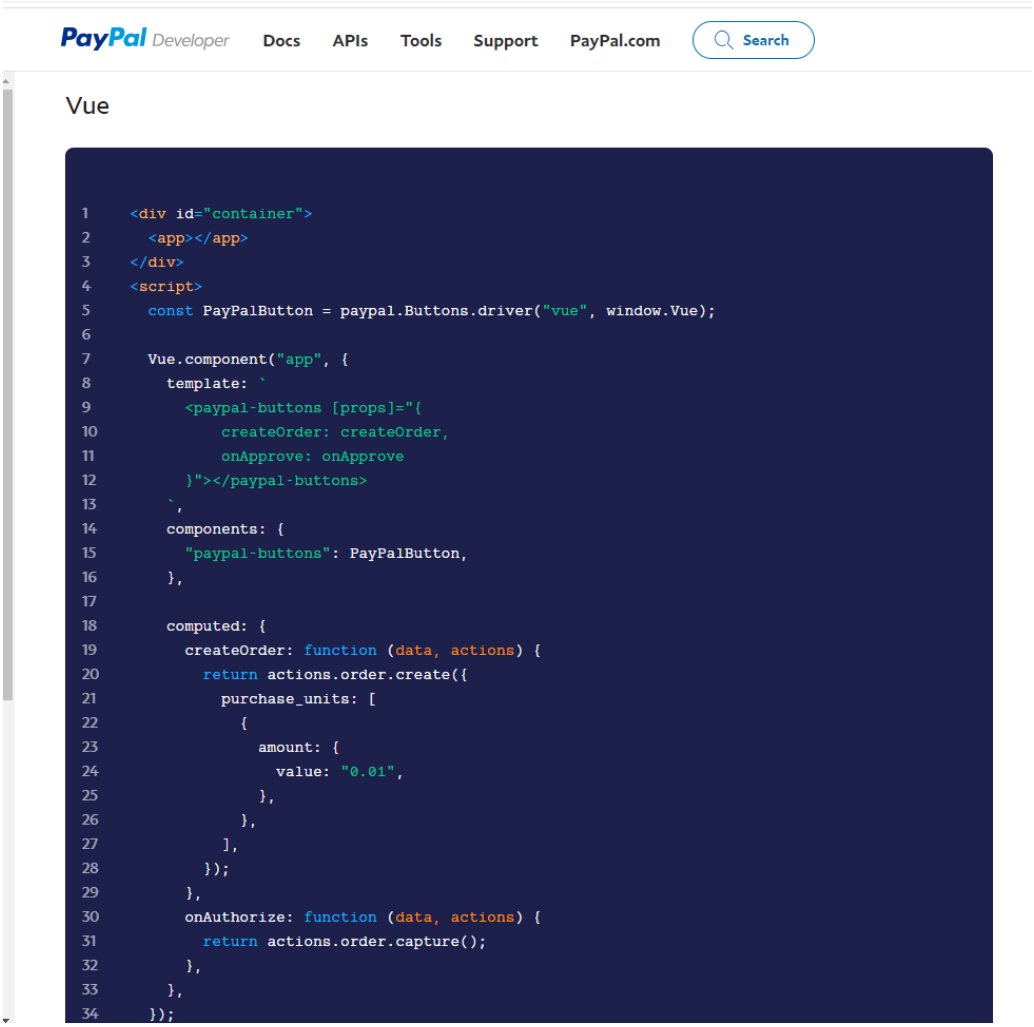


Figura 31 - Componente oficial de Paypal en Vue Fuente: Documentación Paypal Developer.

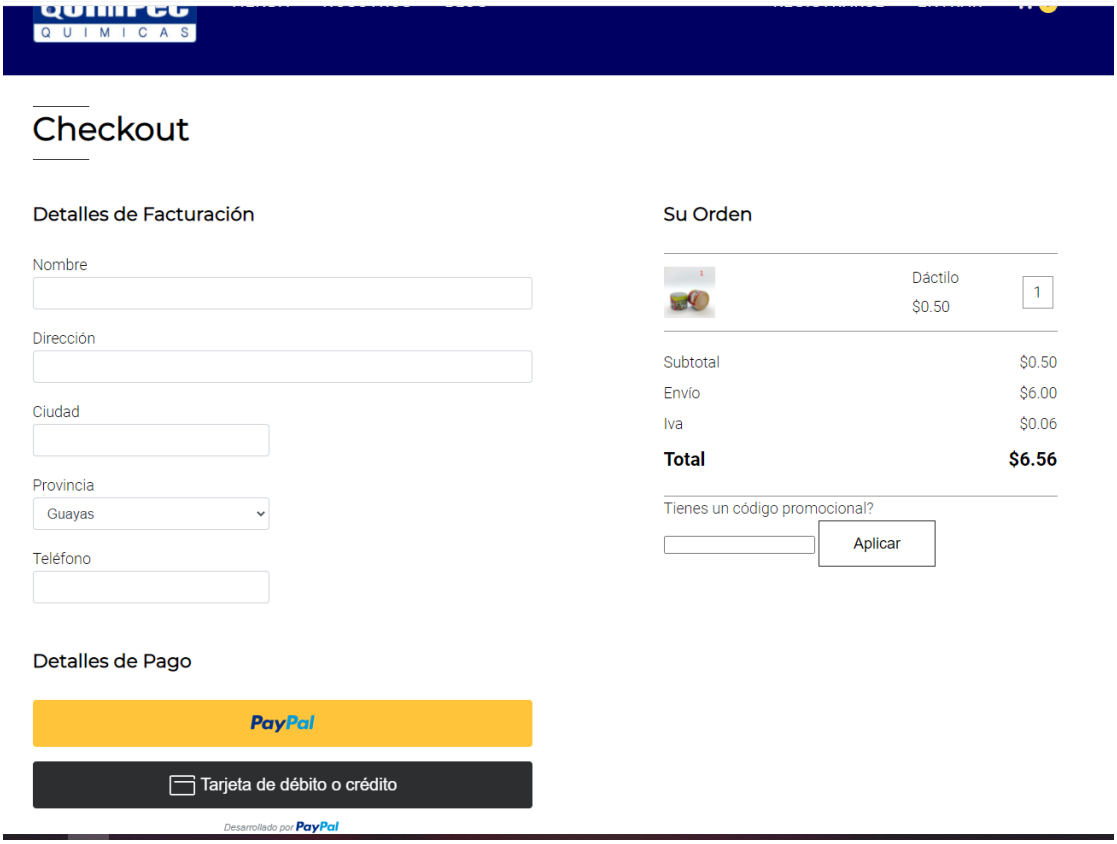


Figura 32 - Integración con el SDK de Paypal Fuente: Elaboración propia.

Para finalmente comprobar que las dos pruebas y 20 comprobaciones escritas en esta iteración estén pasando.

```
C:\laragon\bin\php\php-7.2.19-Win32-VC15-x64\php.exe C:/l
Testing started at 1:37 PM ...
PHPUnit 6.5.14 by Sebastian Bergmann and contributors.



Time: 5.3 seconds, Memory: 28.00MB

OK (2 tests, 20 assertions)
Process finished with exit code 0
```

Figura 33 - Pruebas sobre el API de ventas Fuente: Elaboración propia.

Análisis e interpretación de los resultados

Capítulo 3

Estrategia de pruebas

Toda estrategia de prueba de software incluye un plan de pruebas diseño de casos de pruebas, ejecución de pruebas y la información resultante de las evaluaciones. La estrategia para probar sistemas orientados a objetos consiste en comenzar con las pruebas unitarias, luego con las pruebas de integración y culminar con las pruebas de validación y las pruebas de sistema (Pressman & Maxim, 2012).

Para el presente proyecto se escogió la estrategia de pruebas conocida como: Desarrollo orientado por pruebas (TDD) porque funciona como una perfecta combinación para los métodos ágiles y es un componente de la metodología XP (Astels, 2003).

Pruebas de verificación

Las pruebas de verificación son aquellas que buscan responder a la pregunta: "¿Estamos construyendo el producto correctamente?". En el presente proyecto las pruebas de verificación abarcan las pruebas unitarias y las pruebas de integración.

Casos de pruebas unitarias

En la siguiente tabla se muestran los casos de prueba de pruebas unitarias, las funciones detalladas se encuentran en el directorio 'test/unit' del proyecto y se escribieron utilizando el framework de pruebas PhpUnit. Las mismas que se pueden ejecutar insertando el comando vendor/bin/phpunit en la consola.

Clase	Casos de prueba / Métodos
Venta	tabla_ventas_tiene_columnas_esperadas()
	se_puede_crear_una_venta()
	una_venta_puede_tener_un_item()
	una_venta_puede_tener_varios_venta_items()

VentaItem	tabla_venta_items_tiene_columnas_esperadas()
	se_puede_crear_venta_items()
	un_item_pertenece_a_una_venta()
User	tabla_users_tiene_columnas_esperadas()
	se_puede_crear_un_usuario()
Product	tabla_products_tiene_columnas_esperadas()
	se_puede_crear_un_producto()
	un_producto_puede_pertenecer_a_una_categoria()
	un_producto_puede_pertenecer_a_varias_categorias()
	la_funcion_as_dollars_devuleve_un_string()
	la_funcion_as_dollars_devuleve_valor_correcto_dado_5_dolares_90_centavos()
Envio	tabla_envios_tiene_columnas_esperadas()
	se_puede_crear_un_envio()
Cupon	tabla_cupones_tiene_columnas_esperadas()
	se_puede_crear_un_cupon_de_tipo_valor_fijo()
	se_puede_crear_un_cupon_de_tipo_porcentaje_descuento()
	funcion_descuento_devuelve_valor_de_cupon_de_tipo_valor_fijo()
	funcion_descuento_devuelve_valor_a_descontar_de_cupon_de_tipo_porcentaje_descuento()
	funcion_descuento_devuelve_0_si_cupon_no_es_del_tipo_correcto()
Categoria	tabla_categorias_tiene_columnas_esperadas()
	se_puede_crear_una_categoria()
	una_categoria_puede_contener_un_producto()
	una_categoria_puede_contener_muchos_productos()
CategoriaProducto	tabla_pivote_categoria_producto_tiene_columnas_esperadas()
	se_puede_crear_un_registro_en_categoria_producto()

Tabla 29 - Pruebas unitarias

Resultados de pruebas unitarias

El resultado de las pruebas unitarias se muestra en la figura 34. Son 29 pruebas unitarias con 57 verificaciones cuya ejecución es correcta.

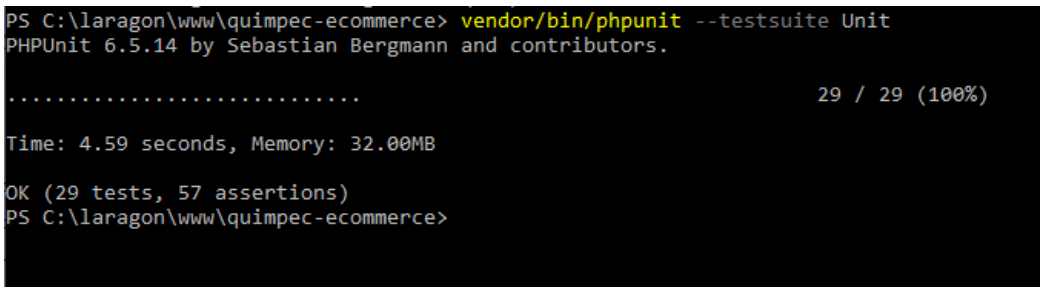


Figura 34 - Resultado de las pruebas unitarias. Fuente: Elaboración propia

Casos de pruebas de integración

En el caso de las pruebas de integración se utilizaron dos frameworks: Phpunit y Laravel Dusk. En la siguiente tabla se muestran los métodos de pruebas automatizadas escritas.

Módulo	Caso de prueba / Método
Login	se_puede_ingrear_al_backend_como_admin()
	no_se_puede_ingrear_al_backend_si_no_es_admin()
CRUD Categorías	se_puede_ver_lista_de_categorias_creados_en_panel_de_control()
	se_muestra_formulario_para_crear_nueva_categoria()
	se_puede_crear_una_nueva_categoria()
CRUD Producto	se_puede_ver_lista_de_productos_creados_en_panel_de_control()
	se_muestra_formulario_para_crear_nuevo_producto()
	se_puede_crear_un_nuevo_producto()
Registro	se_puede_registrar_un_nuevo_usuario()
	se_puede_aumentar_producto_al_carrito ()
	no_se_puede_aumentar_productos_repetidos_al_carrito()
	se_puede_actulizar_cantidad_del_carrito()

Carrito de compras	se_puede_quitar_productos_del_carrito()
Checkout	se_muestra_la_pagina_de_checkout_si_hay_un_producto_en_el_carrito()
	no_se_muestra_la_pagina_de_checkout_si_no_hay_productos_en_el_carrito()
Envio	devuelve_costo_de_envio_al_pasarle_provincia()
Login	usuario_puede_ver_formulario_de_login()
	usuario_no_puede_ver_formulario_de_login_si_esta_autenticado()
	usuario_puede_ingresar_con_credenciales_correctas()
	usuario_no_puede_ingresar_con_password_incorrecto()
	usuario_no_puede_ingresar_con_correo_no_registrado()
	usuario_puede_hacer_logout()
	usuario_no_puede_salir_si_no_ha_ingresado()
Registro	usuario_no_autenticado_puede_ver_formulario_de_registro()
	usuario_autenticado_no_puede_ver_formulario_de_registro()
	se_puede_registrar_un_nuevo_usuario()
	usuario_no_puede_resgistrarse_sin_nombre()
	usuario_no_puede_resgistrarse_sin_mail()
	usuario_no_puede_resgistrarse_con_mail_invalido()
	usuario_no_puede_resgistrarse_sin_password()
	usuario_no_puede_resgistrarse_sin_confirmacion_de_password()
	usuario_no_puede_resgistrarse_si_passwords_no_coinciden()
Compra	se_puede_visualizar_la_vista_shop()
	se_muestra_pagina_de_un_producto_especifico()
	si_producto_no_existe_se_muestra_un_404()
Venta	se_crea_una_nueva_venta_con_un_producto()
	se_crea_una_nueva_venta_con_dos_producto()

Vistas	se_puede_visualizar_la_vista_main()
	se_puede_visualizar_la_vista_shop()
	se_puede_visualizar_la_vista_product()
	se_puede_visualizar_la_vista_cart()
	se_puede_visualizar_la_vista_login()
	se_puede_visualizar_la_vista_register()
	invitado_puede_visualizar_la_vista_checkout()
	invitado_no_puede_ver_vista_checkout_si_no_hay_productos_en_el_carrito()
	usuario_registrado_puede_ver_vista_checkout_con_carrito_no_vacio()

Tabla 30 - Pruebas de integración

Resultados de Pruebas de integración

En la figura 35 se muestra el resultado de la ejecución de las pruebas de integración. Son un total de 37 pruebas con 161 verificaciones que se ejecutan correctamente.

Figura 35 - Resultados de las pruebas de integración automatizadas.Fuente: Elaboracipon propia

Pruebas de Validación

Las pruebas de validación tratan de responder a la pregunta: ¿Estamos construyendo el producto correcto? es decir son las que revisan que el software cumpla las necesidades y expectativas del cliente.

Las pruebas de validación son las pruebas de un orden mas alto que las de verificación y tienen como finalidad comprobar que el software cumpla con los requerimientos de funcionalidad, comportamiento y desempeño. En el presente proyecto se han elegido las pruebas de aceptación como pruebas de validación.

Estadística de pruebas

Con ayuda del IDE y de la extensión XDebug podemos generar un reporte de cobertura de las pruebas. Los resultados se muestran en la figura 36. Indicando un nivel de cobertura de las pruebas del 77% lo cual es un número bastante alto. Y se entiende que el porcentaje faltante escódigo escrito en HTML, CSS o en JavaScript, lo cual excede el alcance de las pruebas con PhpUnit.

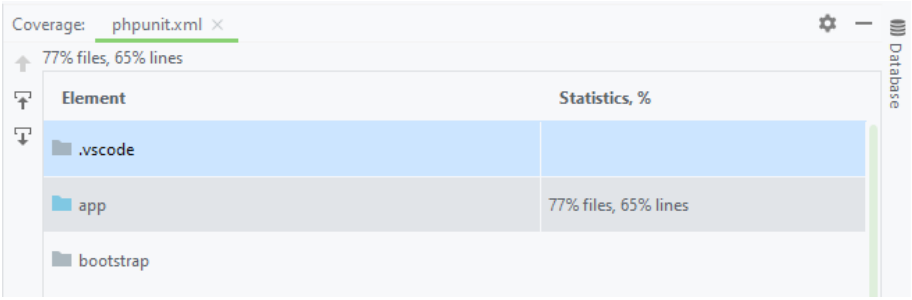


Figura 36 - Reporte de cobertura del código. Fuente: Elaboración propia

Pruebas de aceptación.

Los resultados de las pruebas de aceptación se muestran en la figura 37.

Quito, 26 de mayo de 2021

Aceptación de software por parte del cliente

Por la presente se hace constar que el señor Jorge Peralta Godoy, con número de cédula 0917667958, terminó satisfactoriamente el desarrollo del proyecto "**Desarrollo de una tienda online con reportes de indicadores de ventas**", cumpliendo con los requerimientos solicitados en cuanto a funcionalidad. Esta labor la realizó en el período del 1 de abril de 2021 al 16 de mayo de 2021

El detalle de la funcionalidad requerida, y aceptada es el siguiente:

Funcionalidad Requerida	**Conforme**	**No conforme**
Mostrar el catálogo de productos	✓	
Crear, editar y eliminar productos	✓	
Crear, editar y eliminar categorías de productos	✓	
Módulo de Mantenimiento de usuarios	✓	
Crear pantalla de login para backend	✓	
Implementar carro de compra	✓	
Implementar el uso de tarjeta de crédito como método de pago por parte del cliente	✓	
Registro de los clientes en la página, mediante la creación de cuenta de usuario.	✓	
Crear códigos de descuento	✓	

Se extiende la presente para los fines que el interesado convenga, en la ciudad de Quito, a los Veintiséis días del mes de mayo de dos mil veinte y uno

Atentamente,

Ing. Jorge Puertas
Gerente General Quimpec Químicas

Figura 37 - Carta de aceptación del cliente. Fuente: elaboración propia

Implementación de la solución tecnológica

Capítulo 4

Gráficos para la implementación

Existen una serie de gráficos UML que facilitaron la implementación del proyecto, los cuales se presentan a continuación:

Diagrama Entidad-relación

En la figura 38 se muestra el diagrama Entidad-Relación, con las diferentes conexiones en las tablas utilizadas en la base de datos del software. Tener en cuenta que la tabla ‘cupones’ no tiene relación con otra tabla en la implementación en el software, por tanto, no se le asigna ninguna relación en el diagrama. Así mismo, la tabla ‘migrations’ al ser una tabla propia del framework laravel tampoco tiene relación con otras tablas del sistema desarrollado.

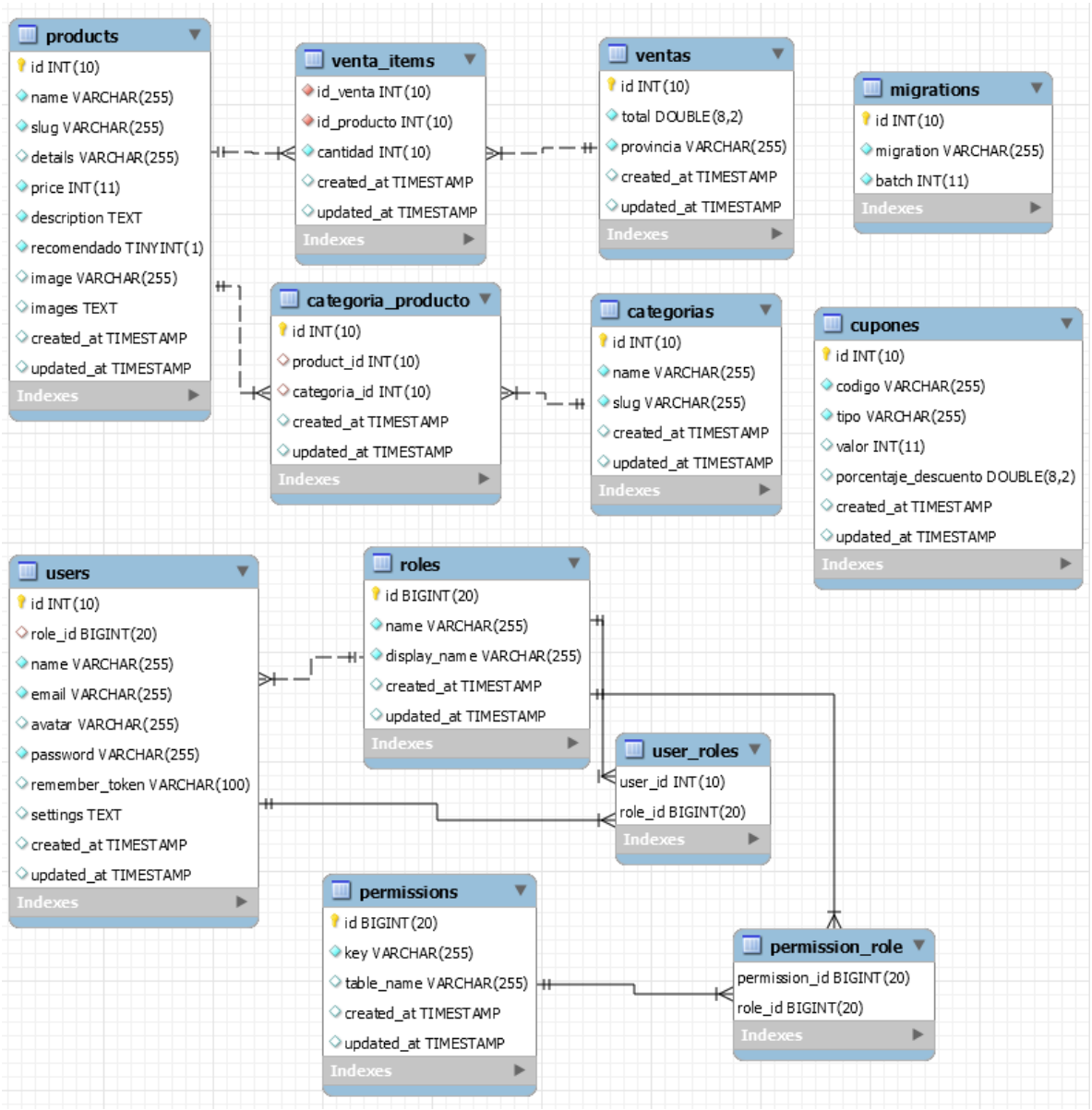


Figura 38 - Diagrama Entidad-Relación. Fuente: Elaboración propia.

Diagramas de clases de Modelos

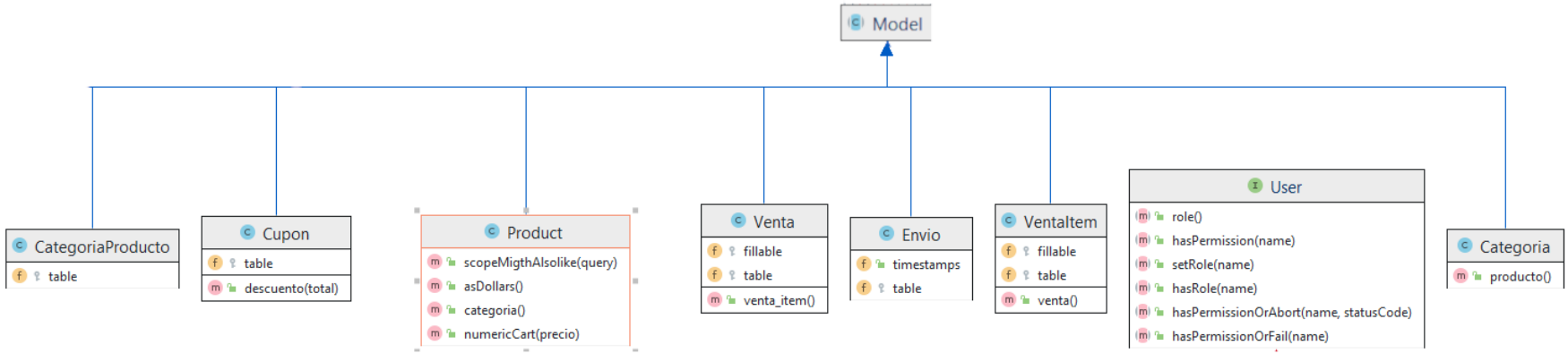


Figura 39 - Diagrama de Clases de Modelos. Fuente: Elaboración propia.

Diagrama de Clases de Controladores

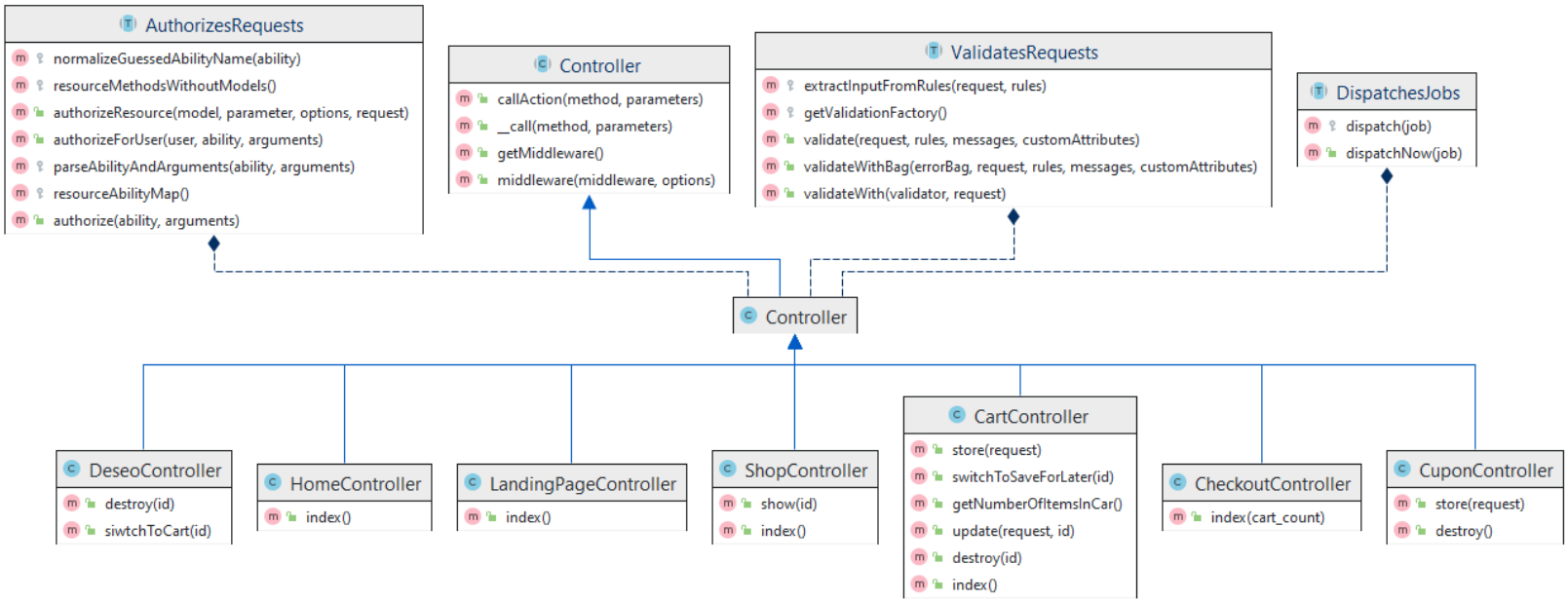


Figura 40 - Diagrama de Clases de Controladores. Fuente: Elaboración propia.

Diagrama de casos de uso del usuario

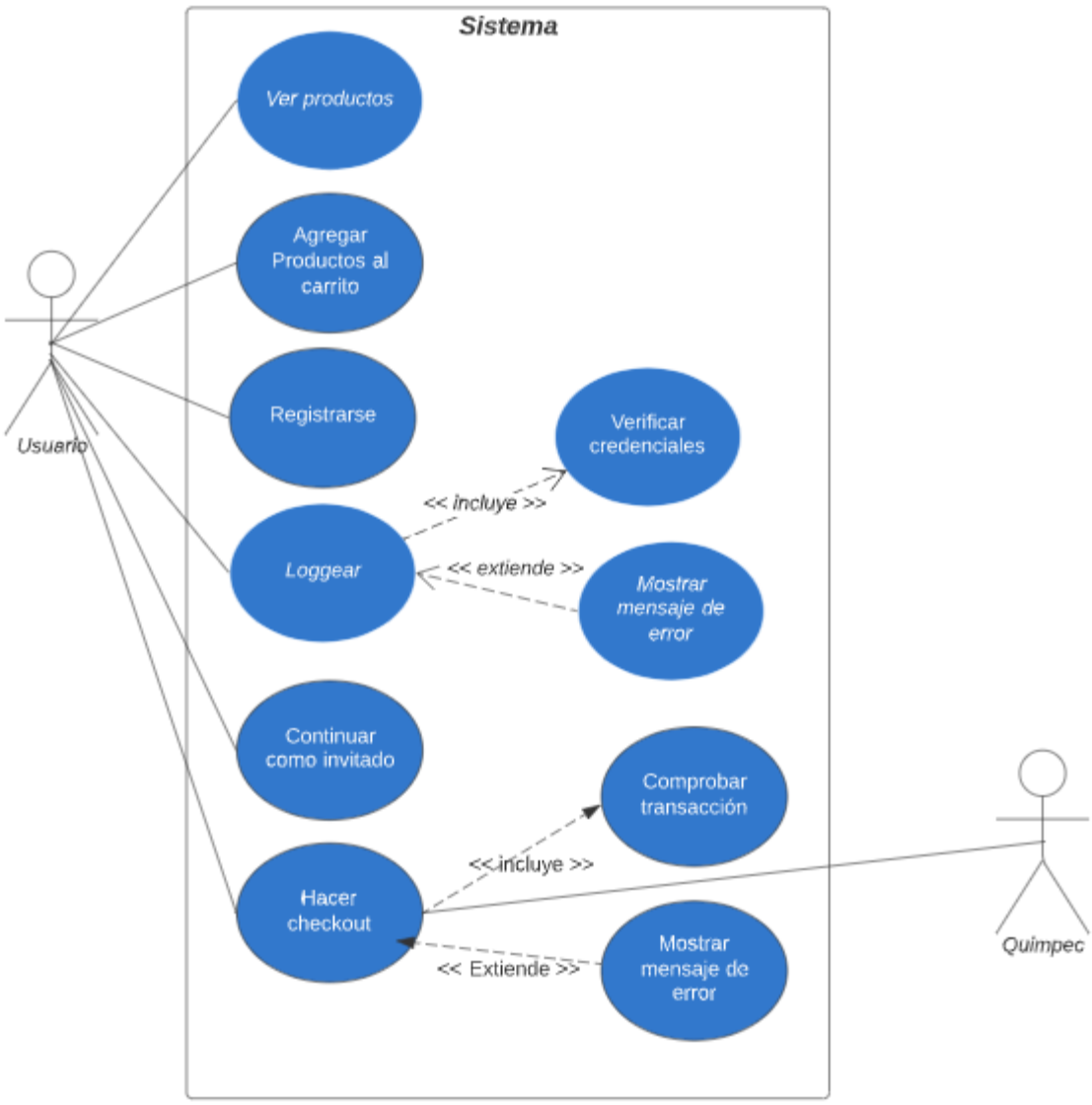


Figura 41 - Diagrama de casos de uso para el usuario. Fuente: Elaboración propia.

Diagrama de casos de uso de administrador

Figura 42 - Diagrama de casos de uso para administrador. Fuente: Elaboración propia

Implementación en ambiente de pruebas

Flujo de compras

Se muestran las capturas de pantalla que el usuario debe seguir en el momento de realizar compra.

Figura 43 - Pantalla de inicio. Fuente: Elaboración propia.

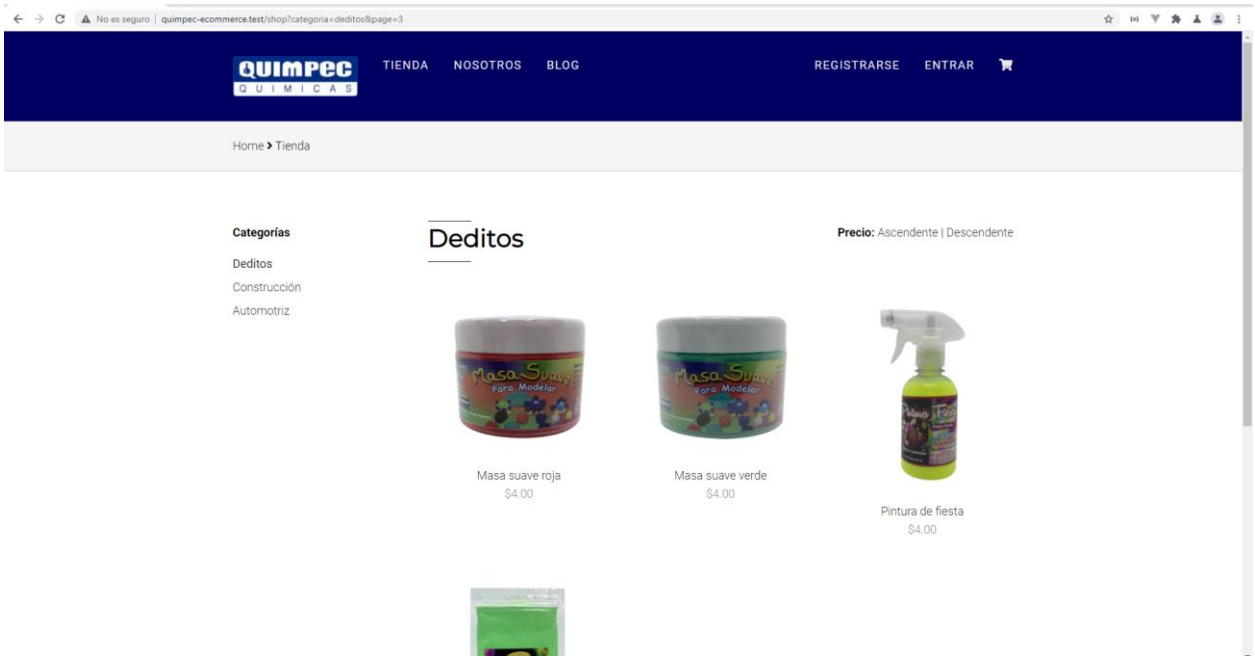


Figura 44 - Catálogo de productos. Fuente: elaboración propia

Figura 45 - Vista de producto. Fuente: Elaboración propia.

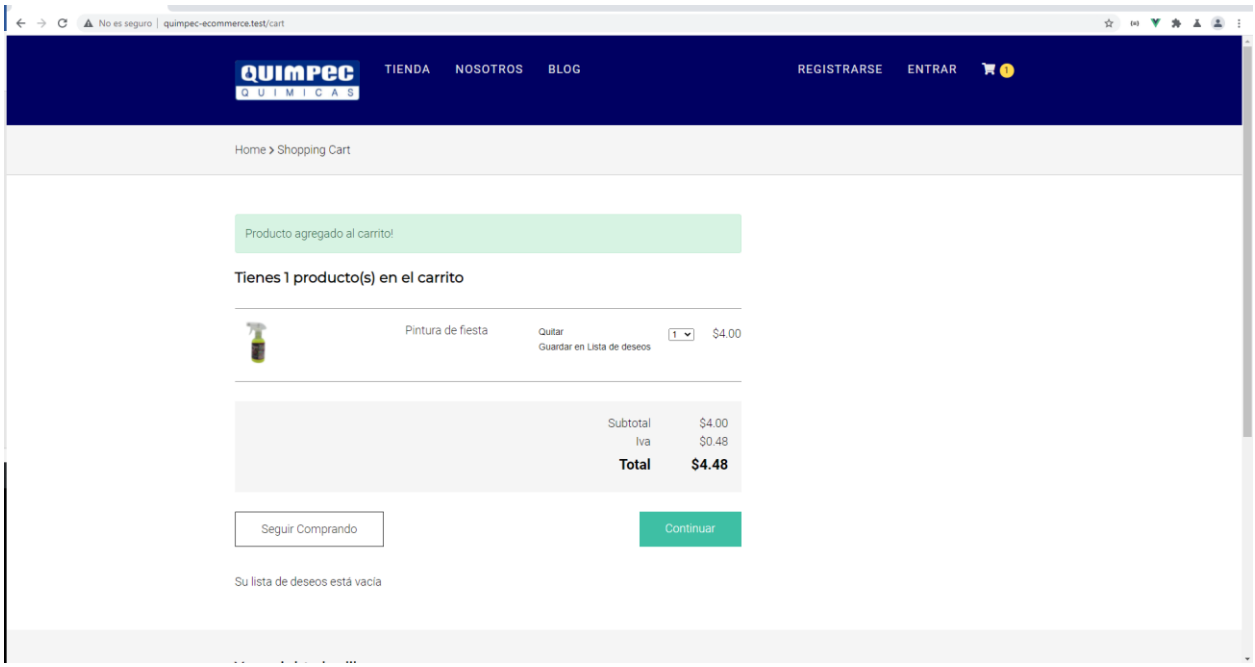


Figura 46 - Vista de carrito. Fuente: Elaboración propia.

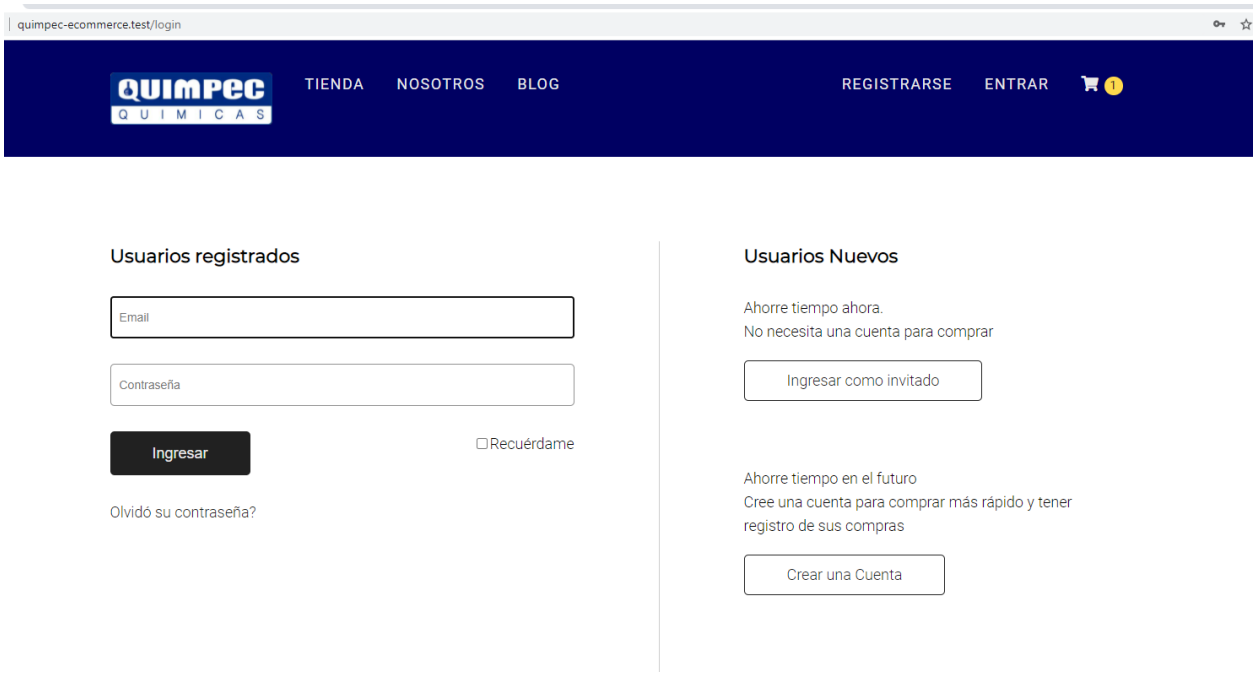


Figura 47 - Opción de login. Fuente: Elaboración propia

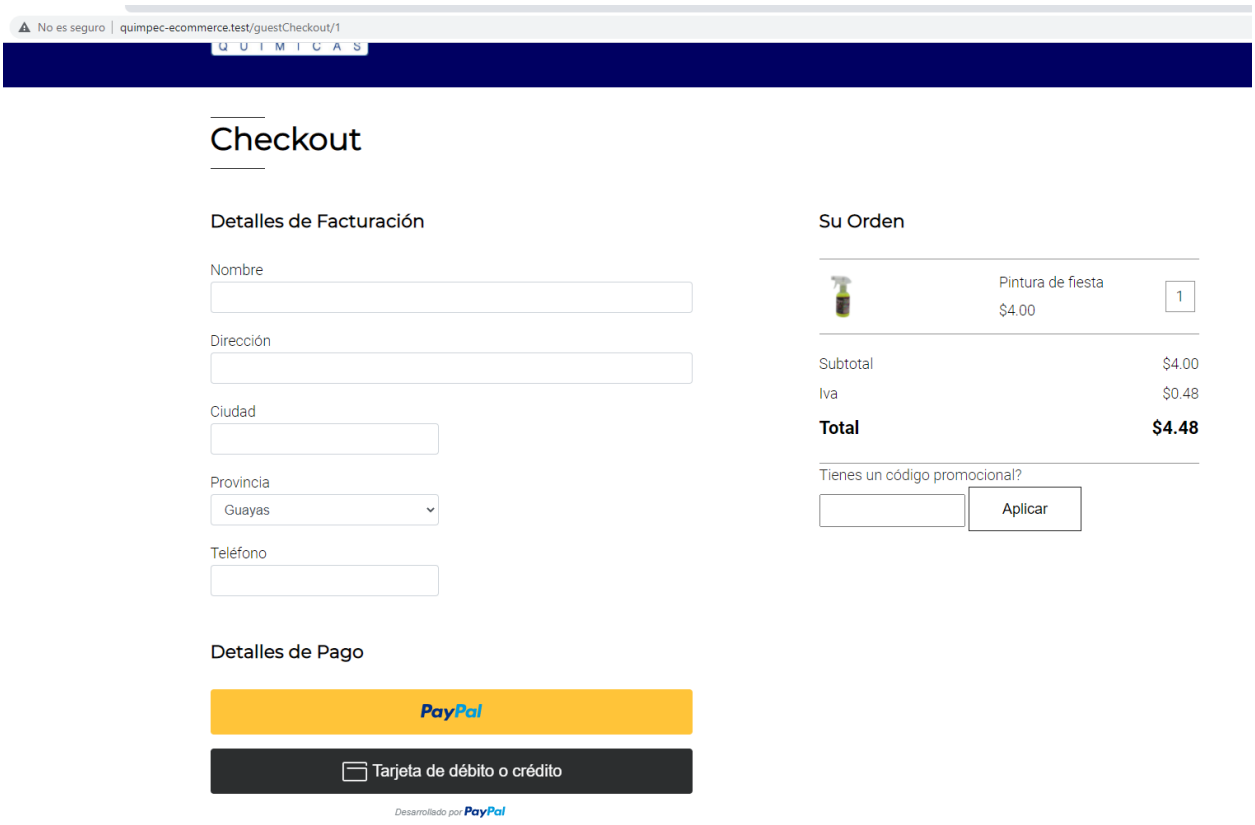


Figura 48 - Vista de pago. Fuente: Elaboración Propia

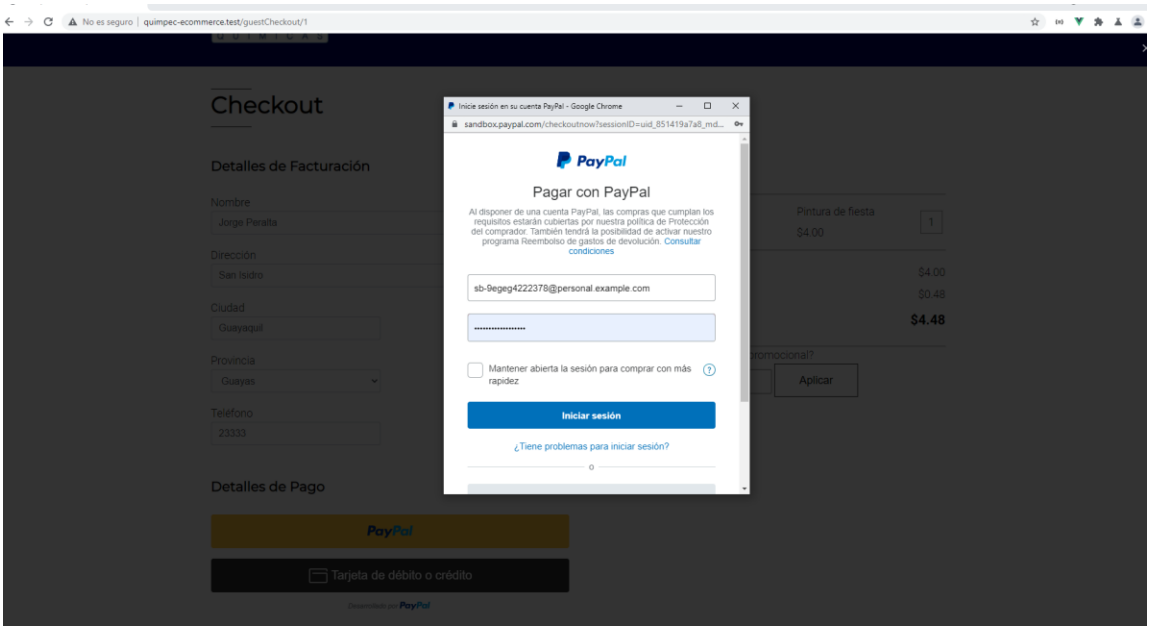


Figura 49 - Login en Paypal. Fuente: Elaboración propia

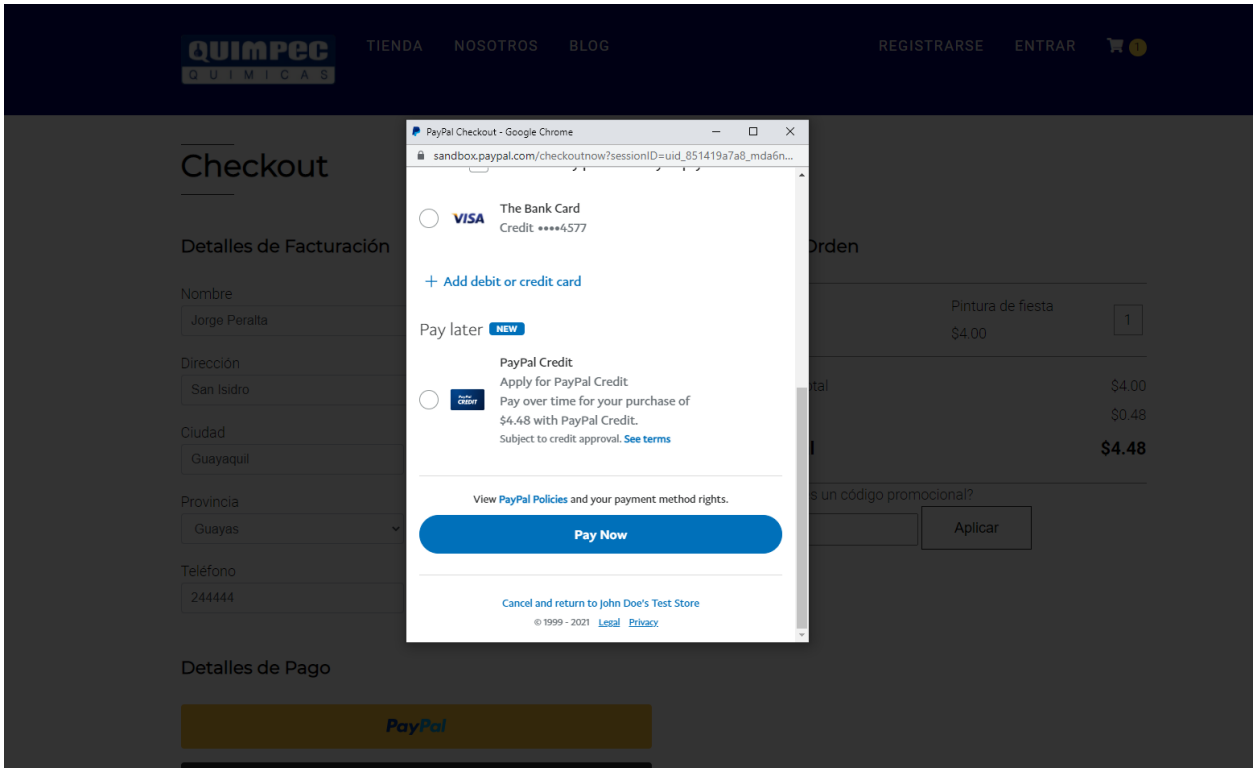


Figura 50 - Pagar con Paypal. Fuente: Elaboración propia

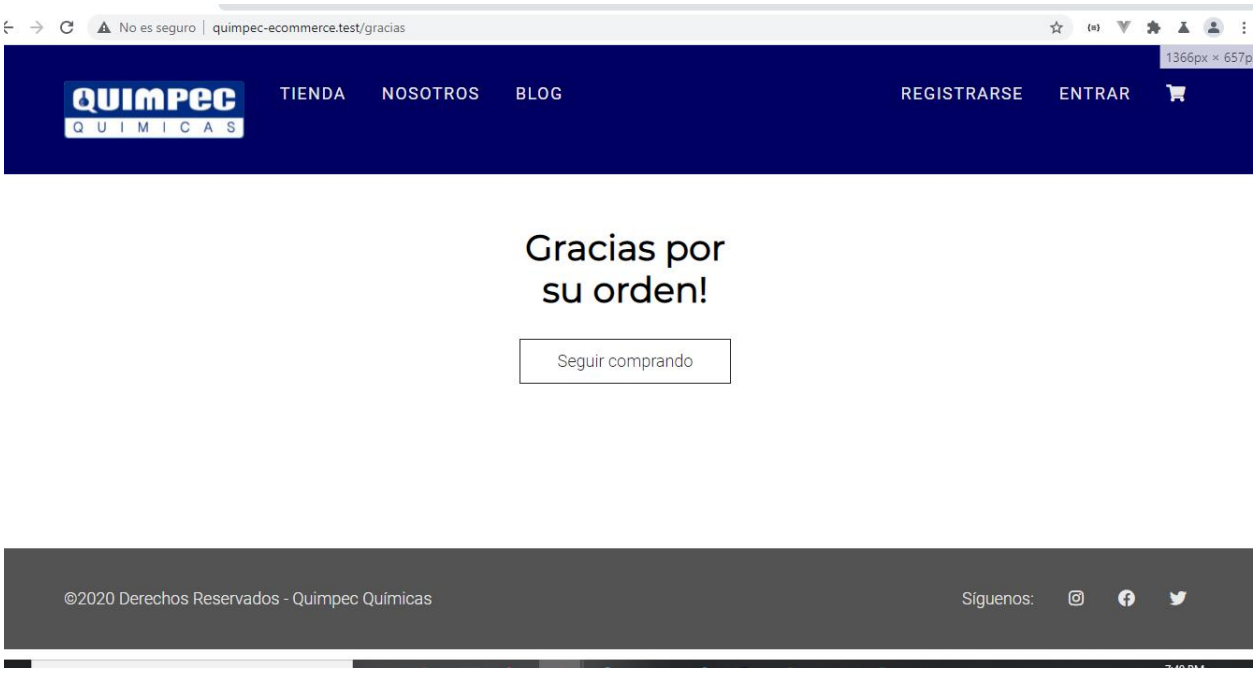


Figura 51 - Compra terminada. Fuente: Elaboración propia.

Flujo de ingreso de producto por parte del administrador

A continuación, se muestran algunas capturas para que el administrador puede agregar un producto al sistema.

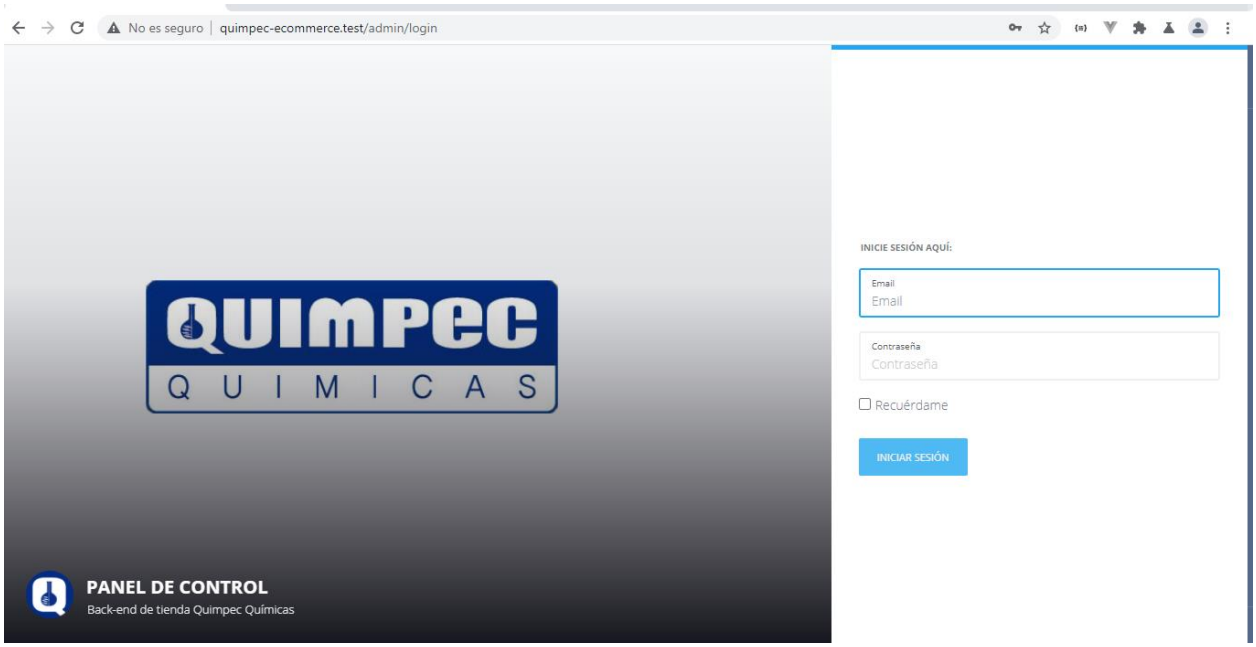


Figura 52 - Login de administrador. Fuente: Elaboración propia.

Figura 53 - Dashboard de administrador. Fuente: elaboración propia

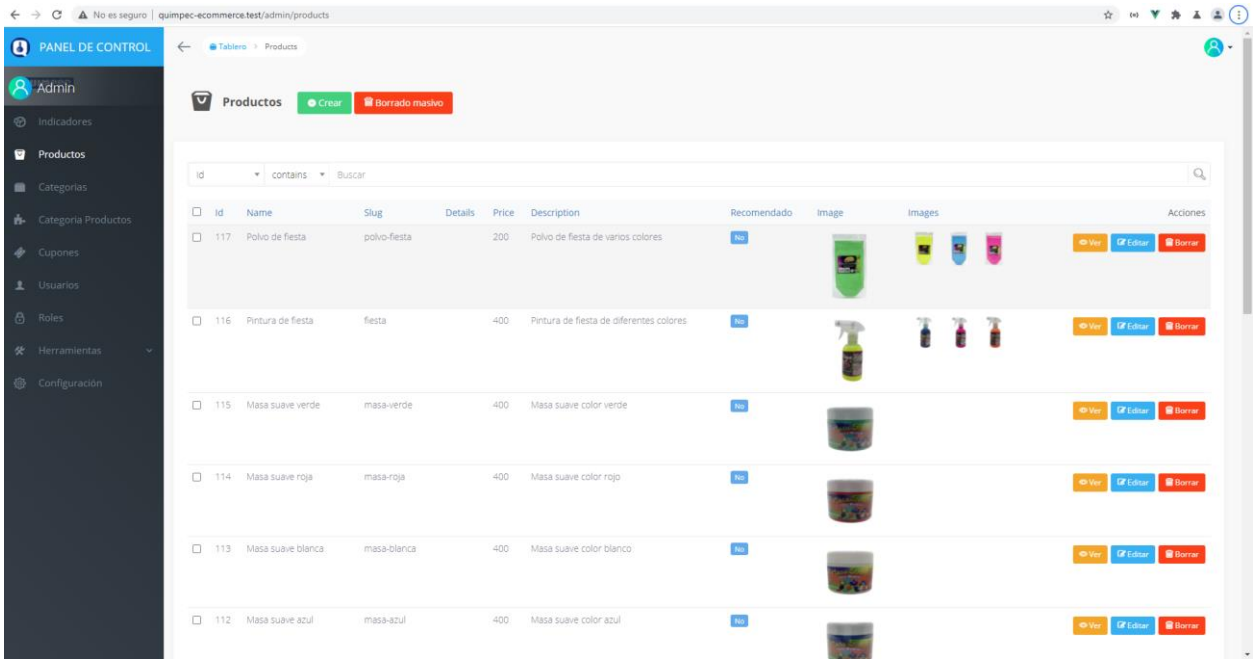


Figura 54 - CRUD de productos. Fuente: Elaboración propia

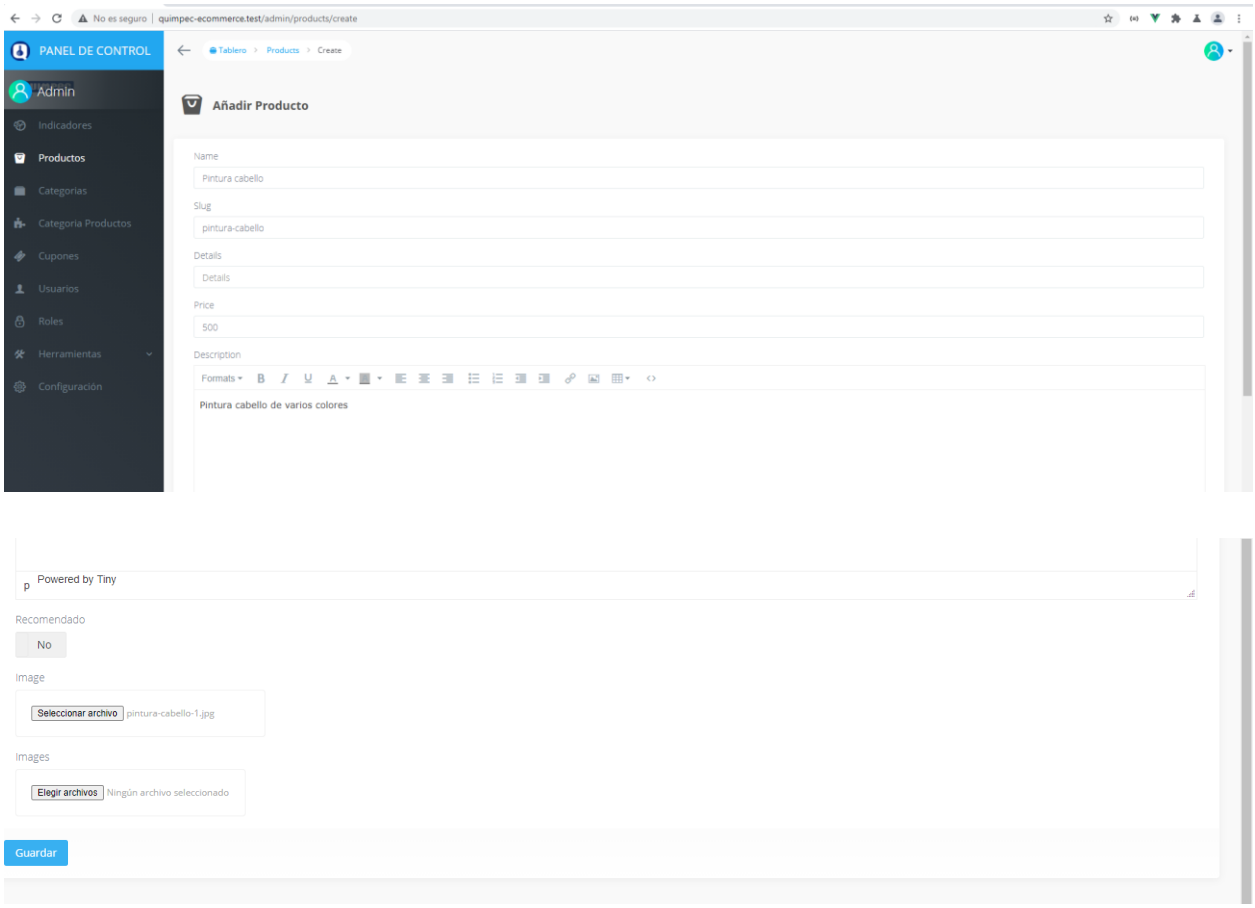


Figura 55 - Creación de un producto. Fuente: Elaboración propia.

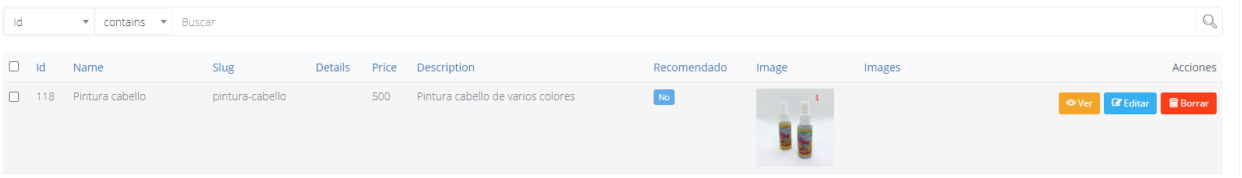


Figura 56 - Producto añadido. Fuente: Elaboración propia

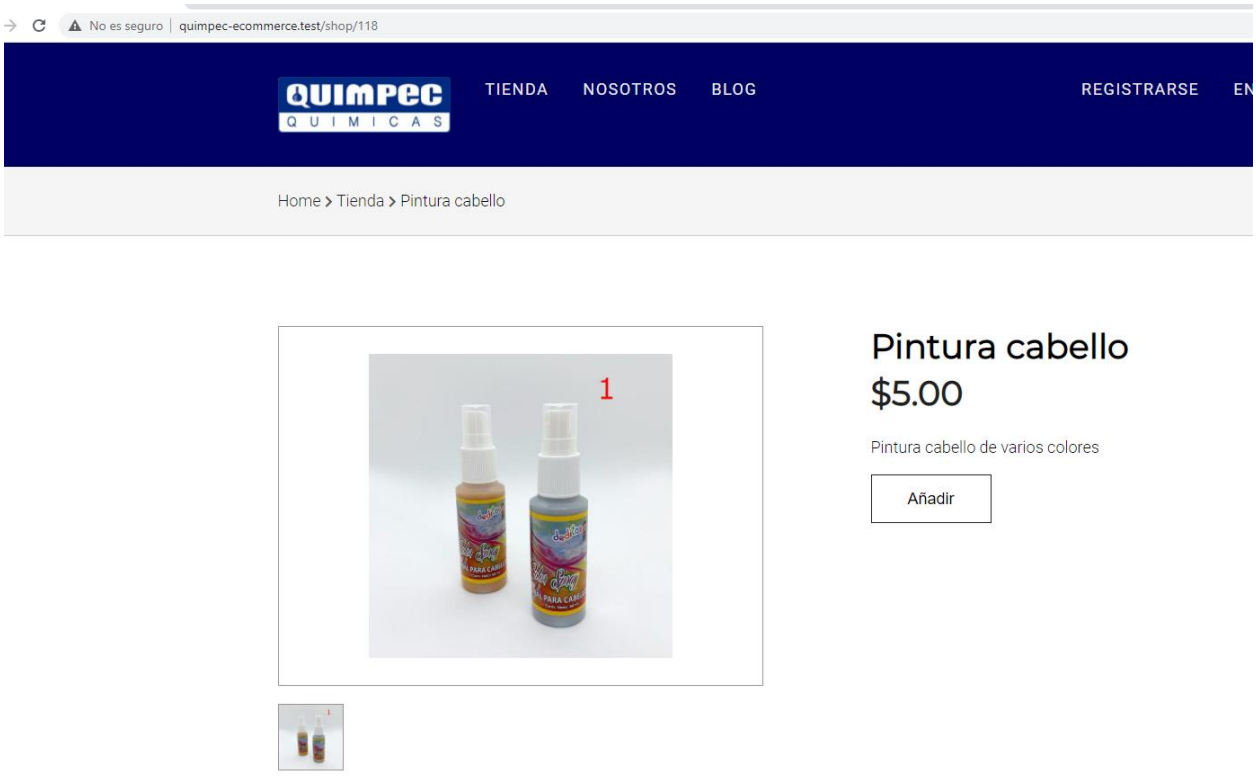


Figura 57 - Producto visible para el usuario. Fuente: Elaboración propia.

Mejora de la toma de decisiones

Tras la implementación se puede observar una mejora en la calidad de la información con la que cuentan los gerentes. Por medio de un modelo relacional se ha conseguido eliminar la redundancia y asegurar la integridad de la información

Quimpec Cia. Ltda. ha pasado de almacenar y mostrar su información en hojas de cálculo, a almacenarlo en bases de datos relacionales y por medio de este modelo la aplicación puede mostrar tres gráficos: Productos vendidos el presente mes (figura 58), monto de ventas en dólares por mes (figura 59) y monto de ventas en dólares por provincia (figura 60).

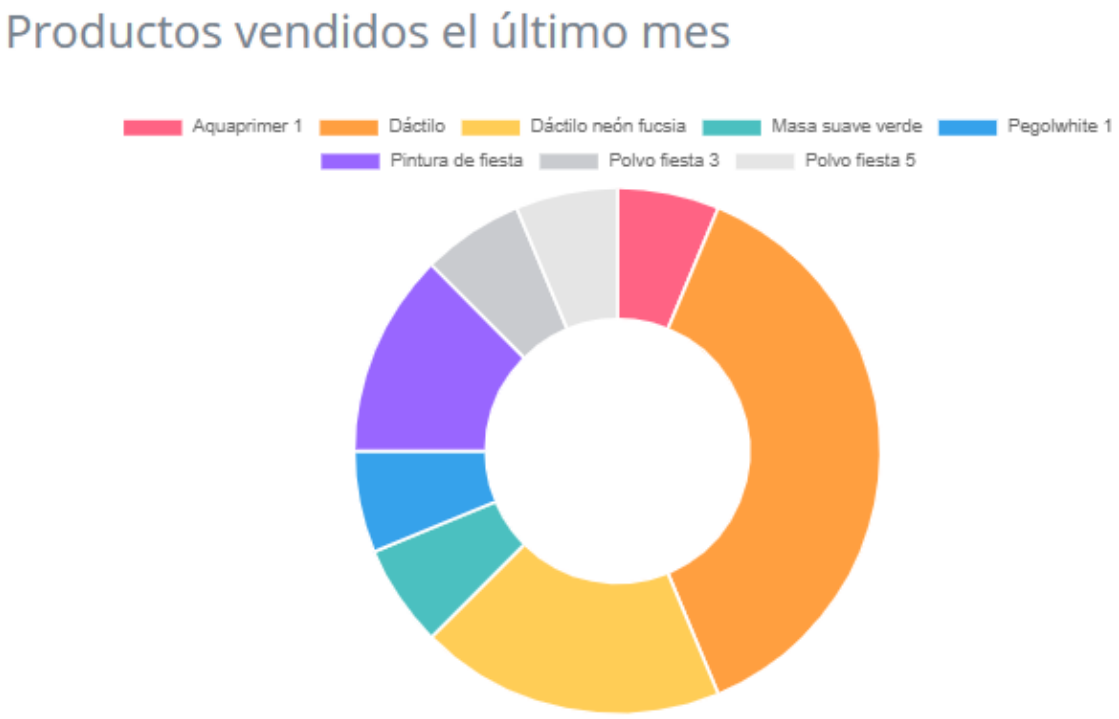


Figura 58 - Gráfico de productos vendidos en el último mes

Figura 59 - Gráfico de monto por mes

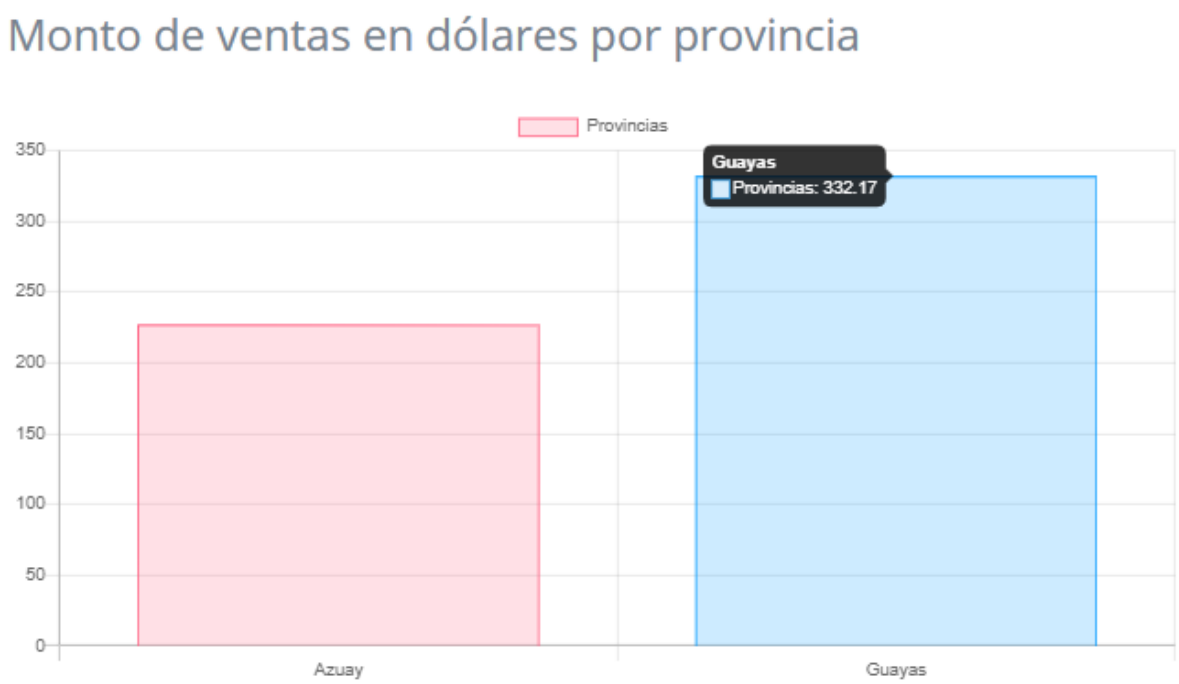


Figura 60 - Gráfico de Montos por provincia

Conclusiones

La investigación de publicaciones de autores como Asensio (2017) y Parkin (2014) demostraron la importancia del comercio electrónica en la administración gerencial actual y aportaron principios fundamentales que se debe seguir, si se desea tener éxito en la incursión en el comercio en la web.

El diagnóstico realizado a los recursos para tomar decisiones, demostró que Quimpec Cia. Ltda. tenía un campo de mejora en esta área. La información sobre las ventas se llevaba en hojas de cálculo sin restricciones para asegurar la consistencia de la información.

Se eligieron las metodologías de desarrollo XP y TDD como metodologías de pruebas. Con la guía de estás metodologías se desarrolló un plan de iteraciones para agregarle funcionalidades a la aplicación en cuestión, por medio de un modelo incremental.

Se cumplió con la planificación y con el plan de iteraciones, pero es importante destacar que este tuvo que redefinirse debido a que el cliente expresó un nuevo requerimiento en la mitad del desarrollo. Esto no significó un problema, puesto que las metodologías ágiles fueron diseñadas justamente para adaptarse de forma eficiente a los cambios que se dan en el momento del desarrollo.

El módulo de gráficos con indicadores genera 3 gráficos para la mejora de la toma de decisiones: Monto de ventas en dólares por provincia, monto de ventas en dólares por mes y productos vendidos al mes. Sin embargo, lo más relevante es el modelo de datos que queda diseñado dentro de la empresa desde el cual podrán extraerse otras informaciones con otros indicadores según lo que necesite la gerencia, apoyándose en la eliminación de redundancia y en la integridad de la información.

Finalmente, El proyecto termina con pruebas unitarias, de integración y de aceptación con resultados satisfactorios, lo cual no solo habla de la calidad del producto, sino que provee alternativas para hacer futuras mejoras deseadas con seguridad de no dañar el código ya existente.

Todo lo anterior les permitirá a los gerentes de Quimpec Cia. Ltda. contar con mejores herramientas para la toma de decisiones orientada al incremento de las ventas.

Recomendaciones

Como primer punto se recomienda poner el sistema en producción para poder aprovechar los beneficios de las características diseñadas. También se recomienda pasar toda la información de la contabilidad de Quimpec a bases de datos relacionales, no solo la generada por el presente sistema de comercio electrónico. Para de esta forma, aprovechar los beneficios y la diversidad de consultas que ofrece una base de datos por encima de una hoja de cálculo.

Por otro lado, es importante que Quimpec siga mejorando en sus herramientas de toma de decisiones, se aconseja que los gráficos de indicadores de ventas seas de todas las ventas de la empresa y no solo lo facturado por la tienda online.

Finalmente se recomienda que, en caso de aumentarle características al producto con otro equipo de desarrollo, se busque que dicho equipo escriba código limpio, legible, continuamente refactorizado y de preferencia con pruebas automatizadas para seguir manteniendo la calidad del producto desarrollado.

Bibliografía

Asensio, A. (2017). *World Wide Data : The Future of Digital Marketing, E-Commerce, and Big Data.* Business Expert Press.

Astels, D. (2003). *Test-Driven Development: a Practical Guide.* Prentice Hall.

Bean, M. (2015). *Laravel 5 Essentials.* Birmingham: Packt Publishing.

Beck, A. (2005). *Extreme Programming Explained, 2nd edition.* Boston: Addison-Wesley.

Bergmann, S. (2006). *PHPUnit pocket guide - Test driven development in PHP.* Sebastopol: O'Reilly.

Bernal, C. (2010). *Metodología de la investigación, administración, economía, humanidades y ciencias sociales. Tercera edición.* Bogotá: Pearson.

Buschmann, F., Meunier, R., Rohnert , H., Sommerlad, P., & Stal, M. (2001). *Pattern-Oriented software architecture, a sustem of patterns.* Chichester: Wiley.

Camps Paré, R., Casillas Santillán, L., Costal Costa, D., Gibert Ginestá, M., Martín Escofet, C., & Pérez Mora , O. (2005). *Bases de datos.* Barcelona: Fundació per a la Universitat Oberta de Catalunya.

DB-engines. (2021, Abril 20). *DB-Engines Ranking*. Retrieved from DB-Engines Ranking: https://db-engines.com/en/ranking

ECMA international. (2011). *ECMAScript Language Specification 5.1 edition.* Génova: Ecma international 2011.

El Telégrafo. (2016, Agosto 6). *El telégrafo noticias*. Retrieved from El telégrafo: https://www.eltelegrafo.com.ec/noticias/economia/4/menos-del-20-de-empresas-realiza-comercio-electronico

Esparza Cruz, N. K. (2016). Comercio Electrónico en el Ecuador. *Journal of Science and Research: Revista Ciencia E Investigación*, 29-32.

Fowler, B. B. (2000). *Refactoring: Improving the Design of Existing Code.* Boston: Addison-Wesley.

Fracica, G. (1988). *Modelo de simulación en muestreo.* Bogotá: Universidad de la Sabana.

Git scm. (2020, Junio 10). *index Git scm*. Retrieved from index git scm: https://git-scm.com/

Hostalia. (2020, Junio 10). *Hostalia*. Retrieved from Hostalia pressroom: https://pressroom.hostalia.com/contents/ui/theme/images/framework-laravel-wp-hostalia.pdf

Jetbrains. (2020, junio 10). *help PhpStorm*. Retrieved from Php Storm 2017.3 Help: https://www.jetbrains.com/help/phpstorm/2017.3/meet-phpstorm.html

Juan José Castaño, S. J. (2016). *Marketing digital (Comercio electrónico).* Editex.

Kilicdagi, A. (2014). *Laravel Design Patterns and Best Practices.* Birmingham: Packt Publishing.

Laragon Documentation. (2021, Abril 21). *What is Laragon?* Retrieved from What is Laragon?: https://laragon.org/docs/

Laravel documentation. (2021, Abril 21). *Eloquent: Getting Started*. Retrieved from Eloquent: Getting Started: https://laravel.com/docs/5.5/eloquent

Lee, J. (2019). *Business Hack : The Wealth Dragon Way to Build a Successful Business in the Digital Age.* John Wiley & Sons, Incorporated.

Lorenz , M., Hesse, G., & Rudolph, J.-P. (2021). *Object-relational Mapping Revised - A Guideline Review and Consolidation.* Postdam: Science and technology publications.

McCool, S. (2012). *Laravel Starter.* Birmingham: Packt Publishing.

Miller, E. (2000). *web site testing*. Retrieved from eValid - the web quality suite: http://www.e-valid.com/Technology/White.Papers/wpaper.testing.pdf

Mozilla Developer Networks. (2021, Abril 20). *Tecnología para desarrolladores web JavaScript*. Retrieved from Tecnología para desarrolladores web JavaScript: https://developer.mozilla.org/es/docs/Web/JavaScript

Palacios, D. (2021, Abril 21). *Constructor de consultas SQL de Laravel*. Retrieved from Constructor de consultas SQL de Laravel: https://styde.net/constructor-de-consultas-sql-de-laravel/

Parkin, G. (2014). *Digital Markting: Strategies form online succes.* London: New Holland Publishers.

Pressman, R. (2010). *Ingeniería del Software, un enfoque práctico.* México: Mc Graw Hill.

Pressman, R., & Maxim, B. (2012). *Software Engineering, a practitioner´s approach, eight edition.* New York: McGraw-Hill Education.

Sabki, P. G. (2020). *Developing an e-commerce solution: acase study of TimeXtra.* Emerald Publishing Limited.

Salkind. (1998). *Método de investigación.* México: Prentice-Hall.

Sommerville, I. (2011). *Ingeniería de software, 9na Ed.* México: Pearson Educación.

Sommerville, I. (2016). *Software Engineering, tenth edition.* Pearson.

The PHP Group. (2021, Abril 2021). *php.net*. Retrieved from PHP: Hypertext processor: https://www.php.net/

The Sass team. (2021, Abril 20). *Sass Documentation*. Retrieved from Sass Documentation: https://sass-lang.com/documentation

vuejs.org. (2020, Junio 06). *Vue Js guide*. Retrieved from Vue js guide: https://vuejs.org/v2/guide/

W3Techs. (2021, Abril 20). *Server-side Languages PHP*. Retrieved from Usage statistics of PHP for websites: https://w3techs.com/technologies/details/pl-php

Wells, D. (1999). *XP - Unit Tests*. Retrieved from http://www.extremeprogramming.org/: http://www.extremeprogramming.org/rules/unittests.html

Anexos

1. Pruebas sobre el Carrito de compras

```php
/** @test */
public function no_se_puede_aumentar_productos_repetidos_al_carrito()
{
    $producto = factory(Product::class)->create();

    $this->post( uri: 'cart', [
            "id" => $producto->id,
            "name" => $producto->name,
            "price" => $producto->price,
    ]);

    $response = $this->post( uri: 'cart', [
            "id" => $producto->id,
            "name" => $producto->name,
            "price" => $producto->price,
    ]);

    $response->assertRedirect( route( name: 'cart.index'));
    $response->assertSessionHas('warning_message');
}
```

Figura. Prueba para comprobar que no se agreguen productos repetidos al carrito. Fuente: Elaboración propia

```php
/** @test */
public function se_puede_quitar_productos_del_carrito()
{
    $producto = factory(Product::class)->create();
    Cart::instance('testing');

    $this->post( uri: 'cart', [
            "id" => $producto->id,
            "name" => $producto->name,
            "price" => $producto->price,
    ]);

    $carrito =  Session::get('cart');
    $coleccion_carrito = $carrito["testing"];

    $response = $this->delete( uri: '/cart/'.$coleccion_carrito->first()->rowId);
    $response->assertStatus( status: 302);
    Cart::instance('default');
}
```

Figura. Prueba para comprobar que se puede quitar productos del carrito. Fuente: Elaboración propia

2. Pruebas sobre el CRUD de productos

```php
/** @test */
public function se_muestra_formulario_para_crear_nuevo_producto() {
    factory(Role::class)->create([
            'name' => 'admin',
            'display_name' => 'Administrador'
    ]);
    $admin = factory(User::class)->create([
            'role_id' => '1'
    ]);
    $this->browse(function (Browser $browser) use ($admin) {
        $browser->loginAs($admin)
                ->visit( url: '/admin/products/create')
                ->assertSee( text: 'Añadir Producto');
    });
}
```

Figura. Prueba para mostrar formulario para crear producto. Fuente: Elaboración propia

3. Pruebas sobre el login de usuarios.

```php
class LoginTest extends TestCase
{
    use RefreshDatabase;

    /** @test */
    public function usuario_puede_ver_formulario_de_login(){
        $response = $this->get($this->loginGetRoute());

        $response->assertSuccessful();
        $response->assertViewIs( value: 'auth.login');
    }
```

Figura. Prueba para mostrar formulario de login. Fuente: Elaboración propia

```php
/** @test */
public function usuario_no_puede_ver_formulario_de_login_si_esta_autenticado()
{
    $user =  factory(User::class)->create();

    $response = $this->actingAs($user)->get($this->loginGetRoute());

    $response->assertRedirect($this->guestMiddlewareRoute());
}
```

Figura. Prueba para comprobar que usuario autenticado no pueda ver formulario de login.

Fuente: Elaboración propia

```php
/** @test */
public function usuario_puede_ingresar_con_credenciales_correctas()
{
    $user =  factory(User::class)->create([
            'password' => Hash::make($password = 'secreto'),
    ]);

    $response = $this->post($this->loginPostRoute(), [
            'email' => $user->email,
            'password' => $password,
    ]);

    $response->assertRedirect($this->successfulLoginRoute());
    $this->assertAuthenticatedAs($user);
}
```

Figura. Prueba que se pueda hacer login con credenciales correctas. Fuente: Elaboración propia

```php
/** @test */
public function usuario_no_puede_ingresar_con_password_incorrecto()
{
    $user =  factory(User::class)->create([
            'password' => Hash::make($password = 'secreto'),
    ]);

    $response = $this->from($this->loginGetRoute())->post($this->loginPostRoute(), [
            'email' => $user->email,
            'password' => 'contraseña-incorrecta',
    ]);

    $response->assertRedirect($this->loginGetRoute());
    $response->assertSessionHasErrors( keys: 'email');
    $this->assertTrue(session()->hasOldInput( key: 'email'));
    $this->assertFalse(session()->hasOldInput( key: 'password'));
    $this->assertGuest();
}
```

Figura. Prueba que no se pueda ingresar con contraseña incorrecta. Fuente: Elaboración propia

```php
/** @test */
public function usuario_no_puede_ingresar_con_correo_no_registrado()
{
    $response = $this->from($this->loginGetRoute())->post($this->loginPostRoute(), [
            'email' => 'correo-no-registrado@example.com',
            'password' => 'secreto',
    ]);

    $response->assertRedirect($this->loginGetRoute());
    $response->assertSessionHasErrors( keys: 'email');
    $this->assertTrue(session()->hasOldInput( key: 'email'));
    $this->assertFalse(session()->hasOldInput( key: 'password'));
    $this->assertGuest();
}
```

Figura. Prueba que no se puede ingresar con correo no registrado. Fuente: Elaboración propia

```php
/** @test */
public function usuario_puede_hacer_logout()
{
    $user =  factory(User::class)->create();
     $this->actingAs($user);

    $response = $this->post($this->logoutRoute());
    $response->assertRedirect($this->successfulLogoutRoute());
    $this->assertGuest();
}
```

Figura. Prueba que el logout está funcionando. Fuente: Elaboración propia

```php
/** @test */
public function usuario_no_puede_salir_si_no_ha_ingresado()
{
    $response = $this->post($this->logoutRoute());

    $response->assertRedirect($this->successfulLogoutRoute());
    $this->assertGuest();
}
```

Figura. Prueba que no se puede haer logout si no se ha ingresado primero Fuente: Elaboración propia