ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA DE GESTIÓN DE VENTA DE LICORES PARA LA LICORERÍA "LA NENITA"

DESARROLLO DE UN FRONTEND

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

KEVIN ALEXIS GALARZA JIMENEZ
kevin.galarza@epn.edu.ec

DIRECTOR: BYRON GUSTAVO LOARTE CAJAMARCA
byron.loarteb@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, **KEVIN ALEXIS GALARZA JIMENEZ** declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

KEVIN ALEXIS GALARZA JIMENEZ

kevin.galarza@epn.edu.ec

Certifico que el presente trabajo de integración curricular fue desarrollado por **KEVIN ALEXIS GALARZA JIMENEZ**, bajo mi supervisión.

Ing. BYRON LOARTE, MSc.
DIRECTOR

byron.loarteb@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

KEVIN ALEXIS GALARZA JIMENEZ

DEDICATORIA

Dedico este proyecto a mis padres y hermanos, por su constante apoyo y amor incondicional durante toda mi vida. Gracias por creer en mí y por impulsarme a seguir mis sueños. Sin su apoyo, no habría logrado llegar hasta aquí. A mi tutor de tesis, por su guía, paciencia y por compartir su conocimiento y experiencia. A mis amigos y compañeros de estudio, por su amistad y por compartir este camino conmigo.

KEVIN ALEXIS GALARZA JIMENEZ

AGRADECIMIENTO

Agradezco a todas las personas que me han ayudado a lo largo de este arduo proceso. En primer lugar, a mis profesores por su apoyo y asesoramiento en diferentes etapas de mi carrera y tesis.

Además, no puedo dejar de mencionar a mi familia y amigos, quienes siempre han estado ahí para mí, brindándome su amor, aliento y apoyo emocional. Gracias a ellos, he podido superar los momentos difíciles y mantener mi motivación en todo momento para culminar mi proyecto de titulación.

KEVIN ALEXIS GALARZA JIMENEZ

ÍNDICE DE CONTENIDO

RESUMEN

La licorería "La Nenita" es un establecimiento físico de venta de bebidas alcohólicas con más de tres años ubicado en Santa Rosa de Cuzubamba. Adicional a ello, cada uno de los pedidos se realizaban a través de varios medios como: redes sociales, llamadas telefónicas y la toma de notas a mano, lo que resulta ineficiente debido a la disponibilidad limitada del administrador y el riesgo de pérdida y duplicidad de información. Además, el negocio no disponía de un catálogo virtual que muestre su amplia variedad de productos a toda su clientela.

Para solucionar la problemática antes mencionada, el presente Trabajo de Integración Curricular desarrolla y pone en producción un *frontend* de gestión de ventas de licores para la licorería "La Nenita". Esta aplicación del lado de cliente que se conecta a través de un backend previamente desarrollado lo que permite a los clientes navegar y realizar compras mediante un catálogo virtual y un carrito de compras, proporcionando al administrador un mejor control del inventario y una gestión más eficiente de los pedidos, mejorando así la competitividad, posicionamiento del negocio en el mercado y una experiencia agradable a cada uno de los clientes ya que pueden realizar sus pedidos desde la comodidad de su hogar.

La organización del presente documento se estructura de la siguiente manera: en primer lugar, se identifica la problemática relacionada al negocio, y en base a ello, se establecen los objetivos, alcance y el marco teórico correspondiente. A continuación, se detalla la implementación de la metodología que se ha utilizado, que en este caso es Scrum, así como la creación de distintos prototipos y el uso de herramientas y librerías relevantes. Posteriormente, se proporciona una descripción detallada de cada una de las tareas que se han desarrollado en cada uno de los módulos, junto con los resultados que se han obtenido en cada iteración (Sprint). Finalmente, se presentan las conclusiones y recomendaciones que se han obtenido a lo largo del proceso de desarrollo del proyecto de Integración Curricular.

PALABRAS CLAVE: *Frontend*, Angular, Scrum, Tienda Virtual, Sprint, Pedidos.

ABSTRACT

The liquor store 'La Nenita' is a physical establishment selling alcoholic beverages, located in Santa Rosa de Cuzubamba for over three years. In addition to in-store purchases, orders were taken through various means such as social media, phone calls, and manual note-taking, which proved inefficient due to the limited availability of the administrator and the risk of information loss and duplication. Moreover, the business lacked a virtual catalog to showcase its wide range of products to its clientele.

To address the aforementioned issues, this Curricular Integration Project develops and implements a liquor sales management frontend for 'La Nenita' liquor store. This client-side application connects to a previously developed backend, enabling customers to browse and make purchases using a virtual catalog and a shopping cart. This solution provides the administrator with better inventory control and more efficient order management, thereby enhancing the business's competitiveness, market positioning, and providing a pleasant experience to customers who can place orders from the comfort of their homes.

The organization of this document is structured as follows: firstly, it identifies the business-related issues, establishes the corresponding objectives, scope, and theoretical framework. Next, it details the implementation of the methodology used, which, in this case, is Scrum, along with the creation of various prototypes and the use of relevant tools and libraries. Subsequently, a detailed description of each task developed in each module is provided, along with the results obtained in each iteration (Sprint). Finally, the conclusions and recommendations derived from the Curricular Integration Project's development process are presented.

KEYWORDS: *Frontend*, Angular, S1crum, Virtual Store, Sprint, Orders.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Licorería “La Nenita” es un establecimiento físico ubicado en Santa Rosa de Cuzubamba con una permanencia en el sector de más de tres años comercializando su amplia gama de productos. Su principal actividad comercial consiste en la venta de bebidas alcohólicas a clientes mayores de edad que se encuentran cerca de la localidad [1]. Así mismo, cada uno de los productos son adquiridos a través de pedidos los cuales se realizan por medio de llamadas telefónicas, mensajería instantánea, apuntes a mano, entre otros; para luego ser entregado en el lugar acordado. Sin embargo, en ocasiones, estos medios de comunicación no resultan eficientes, ya que el administrador no siempre está disponible para atender las llamadas o en ocasiones suele perder los apuntes lo que afecta negativamente a la reputación del negocio. Además, todos los pedidos solicitados y el inventario del negocio se registran en un cuaderno, lo que implica el riesgo de perder esta información en caso de que el cuaderno se extravíe. Del mismo modo, el negocio también carece de un catálogo virtual que muestre la amplia variedad de productos disponibles [1].

Por otra parte, la falta de una plataforma de pedidos en línea puede ser una desventaja significativa para cualquier tipo de negocio, ya que limita su capacidad de llegar a nuevos clientes y aumentar su base de consumidores. Además, la pandemia del COVID-19 impulso de una manera considerable las compras en línea, lo que hace que la falta de una plataforma de pedidos en línea sea necesaria e indispensable para cualquier tipo de negocio y actividad económica [2]. Por otra parte, una tienda virtual es una herramienta valiosa que permite atraer a una audiencia más amplia, ya que se encuentra disponible las 24 horas del día, sin depender de ningún tipo de horario comercial. Además, ayuda a reducir gastos, ya que no es necesario invertir en un local físico ni en personal para gestionar las compras, dado que la tienda virtual facilita este proceso [3]. A través de catálogos electrónicos, diversas formas de pago y carritos de compras, los usuarios pueden navegar, observar y adquirir productos con facilidad desde la comodidad de su hogar y utilizando cualquier dispositivo electrónico [4].

En base a lo expuesto anteriormente, este Trabajo de Integración Curricular desarrolla y pone a producción un sistema que opera en el lado del cliente, también conocido como *frontend* de gestión de ventas de licores para la licorería “La Nenita”. Permitiendo a los clientes navegar y adquirir productos por medio de un catálogo virtual y un carrito de compras, además permite que el usuario con perfil administrador tenga un mejor control sobre el inventario disponible y una gestión eficiente de los pedidos que se han generado por sus clientes, garantizando de esta manera que el negocio tenga un mayor número de

ventas, posicionamiento y competitividad en el mercado actual gracias al uso de modernas tecnologías de desarrollo de *software*.

1.1 Objetivo general

Desarrollar un *frontend* de gestión de venta de licores para la licorería "La Nenita".

1.2 Objetivos específicos

1. Definir los requerimientos funcionales del *frontend*.
2. Crear los prototipos del *frontend* en base a los requisitos que se han establecido.
3. Codificar los módulos correspondientes al *frontend*.
4. Ejecutar pruebas en el *frontend* para comprobar el correcto funcionamiento.
5. Desplegar el *frontend* a producción una vez que se han aprobado todas las pruebas y requisitos del dueño del producto.

1.3 Alcance

El comercio electrónico permite realizar transacciones sin salir de casa al igual que las empresas o pequeños negocios pueden establecer tiendas virtuales para ofertar catálogos electrónicos de diversos productos de una manera estratégica. Además, los clientes seleccionan los artículos de su interés y completan el proceso de pago de manera electrónica, lo que proporciona coherencia y comodidad. Por último, los pedidos se entregan en el domicilio o a través de medios digitales, según el producto seleccionado [5]. Por lo tanto, para la licorería “La Nenita” el disponer de una tienda virtual es indispensable para llegar a nuevos clientes y aumentar su base de consumidores utilizando la tecnología como un aliado estratégico [2].

El desarrollo del *frontend* desempeña un papel fundamental en la construcción de aplicaciones o sistemas *web*, ya que proporciona una experiencia agradable y navegación óptima al visitar una página. Adicional a ello, su relevancia se resume en su capacidad para no solo presentar una interfaz visual atractiva, sino también mejorar la accesibilidad y el rendimiento de la página [6]. Por ende, para cumplir los puntos mencionados anteriormente, se han empleado diversas herramientas, librerías y tecnologías como HTML, CSS y JavaScript, siendo el *Framework* Angular el principal de todas ellas.

El componente *frontend* como parte de este trabajo de Integración Curricular se desarrolla para gestionar las ventas de la licorería. Para ello, se utilizan las tecnologías mencionadas

previamente, un *Framework* que organiza el código fuente, archivos y directorios, en conjunto con una metodología ágil que sirve para controlar y cumplir con cada una de las iteraciones. A continuación, una etapa de pruebas que garanticen la calidad del producto terminado y la aprobación del propietario para luego ser desplegado a un ambiente de producción para que los clientes hagan uso del mismo.

Por último, existen 3 perfiles de usuario, cada uno con permisos diferentes para visualizar los distintos módulos como se muestra a continuación:

Perfiles que se establecen:

- Administrador.
- Empleado.
- Cliente.

En el *frontend* el perfil administrador consume varios *endpoints* que le permita:

- Iniciar sesión, cerrar sesión, modificar contraseña.
- Visualizar y modificar perfil del usuario.
- Gestionar categorías.
- Gestionar subcategorías.
- Gestionar productos.
- Gestionar pedidos.
- Visualizar el estado de los pedidos.
- Gestionar comentarios y/o sugerencias.

En el *frontend* el perfil empleado consume varios *endpoints* que le permita:

- Iniciar sesión, cerrar sesión, modificar contraseña.
- Visualizar y modificar perfil del usuario.
- Gestionar pedidos.
- Visualizar el estado de los pedidos.
- Seleccionar el pedido como entregado.

En el *frontend* el perfil cliente consume varios *endpoints* que le permita:

- Iniciar sesión, cerrar sesión, modificar contraseña.
- Visualizar y modificar perfil del usuario.
- Visualizar productos favoritos.
- Gestionar carrito de compras.
- Gestionar pedidos.
- Visualizar detalle de pedido
- Visualizar el historial de pedidos.
- Visualizar el estado de los pedidos.
- Enviar comentarios y/o sugerencias.

1.4 Marco teórico

En el desarrollo *web*, todas las tecnologías utilizadas en el lado del cliente, es decir, en el navegador *web* son ampliamente conocidas para el desarrollo de aplicaciones del lado del cliente. Se centra principalmente en tres lenguajes: HTML, CSS y JavaScript. Además, el *frontend* se encarga de la apariencia de las interfaces permitiendo que la información que se presente sea de manera coherente y comprensible para el usuario final [7].

Responsive Design, es una técnica que se usa actualmente para crear sitios *web* que se adapten a diversos dispositivos, como computadoras, tabletas, teléfonos inteligentes y cualquier otra resolución de pantalla existente. El objetivo es proporcionar una experiencia de usuario óptima y consistente en diferentes dispositivos, asegurando que el contenido y el diseño se ajusten de manera adecuada al tamaño y características de cada pantalla [8].

Lenguaje de marcado de hipertexto o HTML por su abreviatura, es un lenguaje de marcado estándar básico y fácil de usar que permite crear contenido estructurado y atractivo en forma de texto. Con HTML, es posible incluir enlaces o hipervínculos que remiten a otras páginas, recursos o fuentes de información afines, además de permitir la inserción de elementos multimedia como gráficos o sonidos [9].

CSS es la abreviatura de "*Cascading Style Sheets*" en inglés, que significa "hojas de estilo en cascada" en español. Se trata de un lenguaje utilizado para aplicar estilos a elementos escritos en un lenguaje de etiquetas como HTML. El principal objetivo de dicho lenguaje es lograr que el navegador asigne atributos específicos como: posicionamiento, colores, tipografías, animaciones, entre otros estilos a cada elemento de un documento HTML [10].

Para desarrollar páginas *web* interactivas y dinámicas se usa el lenguaje de programación JavaScript. A diferencia de otros lenguajes, JavaScript es interpretado, lo que significa que los programas escritos en JavaScript no requieren ser compilados antes de ejecutarse, permitiendo probar y ejecutar directamente los programas en cualquier navegador sin necesidad de pasos adicionales [11].

Un *Framework* es una colección de implementaciones, librerías, normas y prácticas recomendadas que buscan simplificar y estandarizar el desarrollo de un producto *software*. Su propósito es encapsular tareas comunes y repetitivas en módulos genéricos que pueden ser fácilmente reutilizados en diferentes proyectos [12].

Bootstrap es un *Framework frontend* ampliamente utilizado para desarrollar páginas *web* responsivas y compatibles con dispositivos móviles a través del etiquetado HTML, estilizado CSS y la interactividad JavaScript. Además, proporciona plantillas de diseño predefinidas que incluyen estilos y componentes ya creados, logrando con ello dar al usuario una experiencia más agradable cuando navega en una aplicación *web* [13].

Angular es un *Framework* que tiene como lenguaje de programación a TypeScript pero que se basa y compila a JavaScript. Su fundamento radica en la utilización de una estructura de componentes para construir aplicaciones *web* escalables y de una sola página. Además, proporciona un conjunto de recursos que facilitan la creación, compilación, evaluación y actualización del código fuente del producto *software* [14].

TypeScript es un lenguaje de programación compilado y que se basa en JavaScript, pero tiene varias características adicionales que mejoran la calidad del código. Estas características adicionales buscan reducir errores, simplificar el código, mantener la coherencia y facilitar las pruebas, lo que conduce a la creación de un código más limpio y robusto [15].

Angular CLI es una herramienta de línea de comandos desarrollada por el equipo del *Framework* de Angular. Esta herramienta permite crear proyectos de Angular y agiliza el proceso de generación de archivos y entidades, es decir, agregar módulos, componentes,

servicios o directivas directamente desde la línea de comandos, lo cual simplifica y acelera el proceso de desarrollo de aplicaciones [16].

Angular Material es un conjunto de componentes y módulos que se encuentran ya testeados y desarrollados por el equipo de angular. Su objetivo es facilitar el desarrollo de un proyecto en Angular, ya que permiten la reutilización de código y garantizan un comportamiento adecuado en diferentes resoluciones de pantalla [17].

Para transmitir información a través de la World Wide *Web* se usa el protocolo estándar HTTP (Protocolo de Transferencia de Hiper Textos), el cual define reglas y formatos de sintaxis y semántica informática que permiten establecer una comunicación efectiva para intercambiar información entre los diversos componentes de la arquitectura *web*, como servidores, clientes, proxies, etc. [18].

Una API tiene la capacidad de facilitar el proceso de comunicación para que distintos sistemas de *software* intercambien información entre sí, al proporcionar un conjunto estandarizado de protocolos y definiciones. Su arquitectura se encuentra fundamentada en la de cliente y servidor, es, decir, la aplicación que realiza cualquier tipo de solicitud es el cliente y el que responde es el servidor [19].

2 METODOLOGÍA

Se conoce como estudio de casos a la forma de investigación cuyo objetivo se centra en conseguir los resultados deseados a través de llevar a cabo una investigación efectiva. En otras palabras, es una estrategia basada en la observación y análisis de un caso específico, con el fin de llegar a conclusiones y encontrar la evidencia necesaria para respaldar la validez de dichas conclusiones [20].

En base a lo citado anteriormente, se ha realizado un estudio de casos obteniendo como resultado que la Licorería “La Nenita” no cuenta con una plataforma virtual, desaprovechando todas las ventajas que esta proporciona. Por ende, se ha desarrollado un *frontend* para que el administrador de dicha Licorería logre tener una mayor audiencia, aumento significativo en sus ventas y una gestión eficiente de su negocio y productos.

2.1 Metodología de Desarrollo

Se trata de una agrupación de técnicas que son utilizadas en periodos de tiempo determinados, con el único propósito de mejorar la eficiencia al momento de entregar un proyecto *software*. Con este enfoque, se logra realizar entregas incrementables a medida que se va cumpliendo cada etapa del proyecto, evitando esperar el entregable final hasta la finalización del proyecto [21].

En una reunión realizada en febrero del año 2001 en Utah-EEUU, se implementó el concepto “ágil” para desarrollar *software*. En esta reunión se estableció los valores y principios para que los equipos puedan desarrollar *software* de manera ágil, es decir, de forma rápida y con la capacidad de adaptarse a los cambios (tecnologías, requisitos, equipo, etc.) que puedan presentarse mientras se desarrolla un proyecto *software*. Por lo tanto, se estableció que es fundamental que la planificación no sea rígida, sino flexible y abierta a ajustes [22].

Por lo expuesto anteriormente, la metodología ágil *Scrum* se ha empleado en el proceso de desarrollo de este componente. La cual es una metodología que facilita el trabajo colaborativo, la entrega incremental de funcionalidades y la habilidad de acoplamiento a posibles cambios que durante el desarrollo del proyecto puedan aparecer [23]. A continuación, se presenta una explicación más exhaustiva sobre la implementación de esta metodología ágil y sus respectivas fases, roles y artefactos.

Roles

La metodología ágil *Scrum* se compone de tres roles importantes que son conocidos como *Product Owner*, *Development Team* y *Scrum Master*. Todos y cada uno de estos miembros cumplen con un rol en específico y deben contar con las habilidades necesarias para promover la adaptabilidad, innovación y la eficiencia durante el desarrollo del proyecto [24]. A continuación, se presenta los miembros asignados a cada rol.

Product Owner

Comúnmente llamado como dueño del producto, es el miembro del equipo encargado de optimizar el trabajo realizado por el equipo de desarrollo con la finalidad de maximizar su rendimiento, desempeñando el papel de representante del cliente final y en el *Product Backlog* es el que gestiona cada una de las prioridades. Además, todos los avances y resultados que presenta el equipo de desarrollo son revisados conjuntamente por el *Product Owner* [24]. En este contexto, la persona designada se muestra en la **Tabla 2.1.**

Scrum Master

Garantiza y verifica que se apliquen las buenas prácticas de *Scrum* durante todo el desarrollo del proyecto. Su función principal es eliminar cualquier obstáculo o impedimento que pueda afectar las habilidades del equipo. Además, se encarga de impulsar cambios que promuevan la productividad y el éxito del proyecto, brindando motivación y apoyo constante [24]. En este contexto, la **Tabla 2.1** muestra la persona designada.

Development Team

Esta principalmente compuesto por mínimo 3 y máximo 9 desarrolladores encargados de llevar a cabo la entrega de un producto o funcionalidad terminada. Estos profesionales se organizan y auto organizan para gestionar eficientemente el trabajo que deben realizar en cada una de las iteraciones para la entrega del proyecto [24]. En este contexto, la **Tabla 2.1** muestra la persona designada.

Tabla 2.1 Equipo *Scrum* con sus respectivos roles.

ROL	NOMBRE
Product Owner	María Irene Jiménez Flores
Scrum Master	Ing. Byron Loarte, MSc.
Development Team	Kevin Alexis Galarza Jimenez

Artefactos

En *Scrum* los artefactos son manifestaciones tangibles del trabajo realizado o del valor generado, y su objetivo es aumentar la transparencia de la información esencial. Como resultado, todas las personas que los examinan tienen acceso a la misma base de conocimiento y pueden adaptarse en consecuencia. Los tres artefactos que tienen más relevancia en *Scrum* son: Recopilación de requerimientos, *Product Backlog* y *Sprint Backlog* [24]. A continuación, como parte del desarrollo del *frontend* se presenta su respectiva implementación.

Recopilación de Requerimientos

Desempeña un papel crucial al determinar las necesidades del proyecto. Además, ayuda a estimar el tiempo y el alcance del proyecto, permitiendo una planificación adecuada de los entregables, logrando contribuir de esta manera a los objetivos del proyecto [25]. En la **Tabla 2.2** se han registrado todos los requerimientos funcionales del proyecto, los mismos que se ha logrado recopilar mediante varias reuniones con el *Product Owner*. Para una referencia más detallada, se puede consultar el **ANEXO II**, donde se encuentra la tabla completa.

Tabla 2.2 Requerimientos que se han obtenido.

RECOPILACIÓN DE REQUERIMIENTOS		
Tipo de sistema	**ID-RR**	**Enunciado del Ítem**
Frontend	RR005	Como usuario administrador necesita consumir varios *endpoints* para: • Gestionar categorías.
	RR006	Como usuario administrador necesita consumir varios *endpoints* para: • Gestionar subcategorías.

Historias de Usuario

Es una tarjeta que permite describir de manera detallada las funcionalidades que debe tener el producto *software*. Dentro de un marco ágil, se considera como la mínima unidad de trabajo. Estas tarjetas son simples y fáciles de recordar, siguen un patrón y tienen

elementos específicos [26]. La **Tabla 2.3** contiene la ejemplificación de una Historia de usuario relacionada con una funcionalidad del *frontend*, además esta Historia de usuario ha sido elaboradas siguiendo los requerimientos funcionales que se han recopilado y detallado anteriormente. Para una referencia más completa, en el **ANEXO II** se pueden encontrar las demás Historias de usuario.

Tabla 2.3 Formato para las Historias de usuario.

HISTORIA DE USUARIO	
Identificador: HU005	**Usuario:** Administrador
Nombre historia: Gestionar categorías	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 1	
Responsable (es): Kevin Galarza	
Descripción: El usuario administrador en el *frontend* debe consumir varios *endpoints* para: • Registrar categorías de productos. • Modificar categorías de productos. • Eliminar categorías de productos. • Visualizar lista de categorías de productos.	
Observación: El usuario administrador accede a dichas vistas para gestionar las categorías una vez que ha iniciado sesión.	

Product Backlog

Se refiere a una lista secuencial de tareas planificadas que tiene la intención de llevarse a cabo durante el desarrollo del proyecto. Dentro del *Product Backlog*, se incluyen tanto las actividades principales como las secundarias, organizadas en función de su prioridad en el desarrollo [27]. En ese sentido, se ha utilizado para la creación del *Product Backlog* el formato que se presenta en la **Tabla 2.4**, detallando la iteración, el estado y la prioridad de las diferentes tareas derivadas de las Historias de usuario. Para consultar la tabla completa, se puede hacer referencia al **ANEXO II.**

Tabla 2.4 Formato para el *Product Backlog*.

ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU-005	Gestionar categorías	2	Finalizado	Media
HU-006	Gestionar subcategorías	2	Finalizado	Media

Sprint Backlog

Para presentar el *Product Backlog* se ha usado una tabla que contiene las tareas identificadas por el equipo de desarrollo. Cada tarea debe completarse durante cada iteración, según su prioridad, lo que permite tener un detalle de cada una de las tareas por iteración o *Sprint* [28]. La plantilla con la que se ha elaborado el *Sprint Backlog* se muestra en la **Tabla 2.5**. Para consultar toda la tabla completa se puede revisar en el **ANEXO II**.

Tabla 2.5 Formato para el *Sprint Backlog*.

ELABORACIÓN DE *SPRINT BACKLOG*						
ID – SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREA**	**TIEMPO ESTIMADO**
SB000	Información principal del *frontend*	Módulo informativo	HU001	Visualizar página informativa	• Diseño e implementación de las interfaces para la página informativa (*Landing Page*).	20H

2.2 Diseño de interfaces

Es el proceso mediante el cual los diseñadores crean interfaces para un sistema *software* o dispositivos computarizados, poniendo énfasis en su apariencia y estilo. El objetivo es desarrollar interfaces que los usuarios encuentren intuitivas y atractivas visualmente,

facilitando su uso y generando una experiencia agradable [29]. A continuación, se presenta el *software* que se ha seleccionado para diseñar las interfaces.

Herramienta empleada en el diseño

Figma es una herramienta alojada en la *web*, la cual es usada principalmente para prototipados *webs* y como editor de gráficos vectoriales. Una de sus características distintivas es que, al ser basada en el navegador, permite compartir el proyecto con el equipo y realizar modificaciones en una misma mesa de trabajo colaborativa [30]. Es así que a través del uso de Figma se ha diseñado el prototipo del primer módulo donde se muestra una *Landing Page* de la licorería “La Nenita”, como se muestra en la **Figura 2.1**. Por otro lado, el **ANEXO II** contiene los demás prototipos**.**

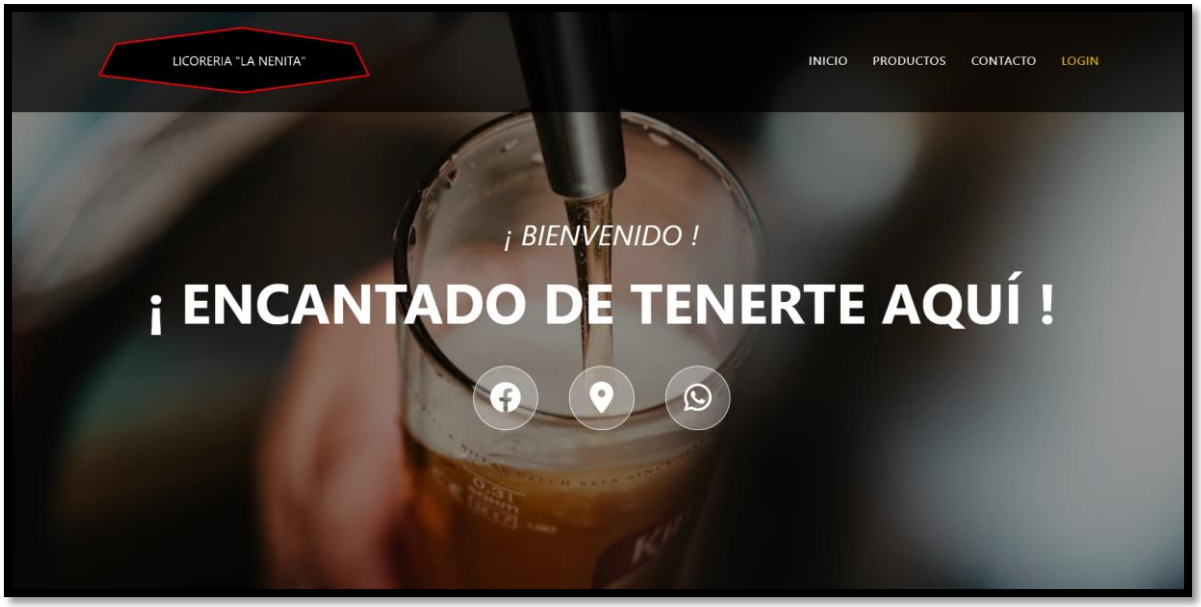

Figura 2.1 Prototipado de la *Landing Page***.**

2.3 Diseño de la arquitectura

Una aplicación o sistema *software* contiene componentes internos que interactúan entre sí, a esto hace referencia el diseño de la arquitectura [31]. El patrón arquitectónico que se ha utilizado en la codificación del *frontend* se muestra a continuación.

Patrón arquitectónico

El patrón de diseño MVC es ampliamente utilizado para desarrollar *software* que implementa interfaces, datos y lógica de control por separado. Es decir, busca separar la lógica del negocio de la visualización, promoviendo así una "separación de preocupaciones". Esta separación permite una mejor división del trabajo y facilita el

mantenimiento del código [32]. A continuación, se muestra la descripción de las capas mencionadas anteriormente:

- **Modelo**: Es responsable de administrar los datos y la lógica del negocio.
- **Vista**: Se encarga de la presentar visualmente la información al usuario.
- **Controlador**: Captura las interacciones que realiza el usuario y dirige los comandos al modelo o vista correspondiente.

En la **Figura 2.2** contiene el patrón que se ha implementado para codificar todo el *frontend*, el cual garantiza que exista compatibilidad entre otras herramientas y librerías.

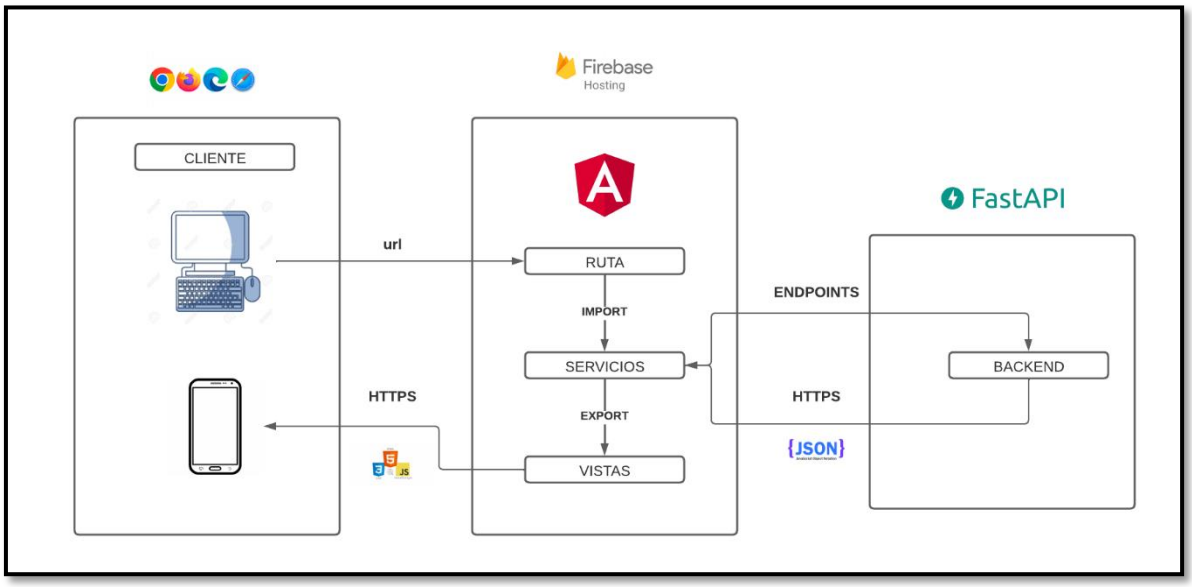

Figura 2.2 Representación del patrón de diseño MVC.

2.4 Herramientas de desarrollo

Las principales herramientas que se han empleado en el desarrollo de este componente han sido especialmente seleccionadas en función de la complejidad del proyecto, las mimas que se describen en la **Tabla 2.6**. Es importante recalcar que estas herramientas son indispensables para garantizar un desarrollo organizado en el *frontend*.

Tabla 2.6 Herramientas para el desarrollo de este componente.

HERRAMIENTA	JUSTIFICACIÓN
Angular	*Framework* de JavaScript que permite desarrollar el *frontend* de una forma escalable y de una sola página usando TypeScript [14].
Angular CLI	Permite desde la línea de comandos generar módulos, componentes, servicios o directivas que son útiles en el momento de la codificación del *frontend* [16].
Angular Material	Permite usar componentes y módulo ya creados para evitar diseñar y estilizar los componentes desde cero [17].
Bootstrap	Permite usar clases con propiedades ya establecidas para estilizar los elementos del *frontend*, evitando crear clases en los archivos CSS [13].

Librerías

En el desarrollo *frontend* se han empleado diversas librerías que proporcionan numerosas ventajas, asegurando la ausencia de errores durante la compilación, ejecución de pruebas y despliegue en producción. A diferencia de un *Framework*, una librería no proporciona una estructura específica de cómo debe realizarse, sino que se considera como un complemento adicional [33]. A continuación, la **Tabla 2.7** muestra las librerías que han sido utilizadas.

Tabla 2.7: Librerías para el *frontend*.

LIBRERÍA	DESCRIPCIÓN
datatables.net	Proporciona tablas interactivas que cuentan con búsqueda, filtros, ordenamiento y paginado [34].
ngx-spinner	Proporciona un spinner con más de 50 modelos para mostrarlo mientras se carga la información [35].
rxjs	Proporciona observables que permiten componer programas asíncronos y que se encuentran basados en eventos [36].

3 RESULTADOS

En esta etapa, se exhiben los logros que se han obtenido en cada uno de los segmentos y componentes visibles del *frontend*. Asi mismo, se presentan las pruebas que se han realizado y la implementación en un entorno de producción. Los resultados se muestran en *Sprints*, los cuales se han establecido previamente y se detallan en el **ANEXO II**.

3.1 *Sprint* 0. Configuración del ambiente de desarrollo.

A continuación, se exponen las actividades correspondientes al *Sprint* 0, las cuales han sido establecidas en el *Sprint Backlog*:

- Compilación de los requerimientos.
- Estructura del proyecto *frontend*.
- Roles de usuarios.

Recopilación de los requerimientos

Página informativa

Dentro del componente *frontend* el usuario administrador, empleado y cliente pueden visualizar una página informativa con las siguientes secciones: Inicio, productos, contacto y *login*. Además, no es necesario iniciar sesión en el *frontend* para visualizar dicha página.

Consumo de *endpoints* para el proceso de registro de usuarios

Dentro del componente *frontend*, los usuarios de tipo cliente tienen la capacidad de realizar el proceso de registro. Para lograrlo, es esencial que se consuman los *endpoints* que se han implementado en el componente *backend* para este proceso.

Consumo de *endpoints* para el proceso de inicio de sesión, cierre de sesión y modificación de contraseña

Dentro del componente *frontend*, los usuarios de tipo administrador, empleado y cliente pueden acceder mediante un email y contraseña. Para ello, es fundamental que el *frontend* consuma el *endpoint* que se ha proporcionado por el *backend*. Asi mismo, para permitir a los usuarios cerrar sesión, el *frontend* debe consumir el *endpoint* que se ha implementado por el *backend*, y para modificar la contraseña, el *frontend* debe consumir los *endpoints* respectivos que se han generado por el componente *backend*.

Consumo de *endpoints* para el proceso de modificar el perfil de usuario

Dentro del componente *frontend*, el usuario de tipo administrador, empleado y cliente tienen la capacidad de visualizar su perfil de usuario. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend*. Además, para realizar modificaciones en el perfil de estos usuarios, el *frontend* también debe consumir el *endpoint* respectivo.

Consumo de *endpoints* para el proceso de gestión de categorías

Dentro del componente *frontend*, el usuario de tipo administrador puede gestionar las categorías. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite visualizar las categorías existentes. Asi mismo, para realizar modificaciones en las categorías antes mencionadas, el *frontend* debe consumir el *endpoint* respectivo.

Consumo de *endpoints* para el proceso de gestión de subcategorías

Dentro del componente *frontend*, el usuario de tipo administrador puede gestionar las subcategorías. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite visualizar las subcategorías existentes. Asi mismo, para realizar modificaciones en las subcategorías mencionadas, el *frontend* debe consumir el *endpoint* respectivo.

Consumo de *endpoints* para el proceso de gestión de productos

Dentro del componente *frontend*, el usuario de tipo administrador puede gestionar los productos. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite visualizar los productos existentes. Asi mismo, para realizar modificaciones en los productos antes mencionados, el *frontend* debe consumir el *endpoint* respectivo.

Consumo de *endpoints* para el proceso de gestión de pedidos

Dentro del componente *frontend*, el usuario de tipo administrador y cliente pueden gestionar pedidos. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* el cual permite visualizar pedidos existentes. Asi mismo, para realizar modificaciones en los pedidos antes mencionados, el *frontend* debe consumir el *endpoint* respectivo.

Consumo de *endpoints* para la visualización de estado del pedido

Dentro del componente *frontend*, el usuario de tipo administrador y cliente pueden visualizar el estado de los pedidos. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite visualizar el estado de los pedidos existentes.

Consumo de *endpoints* para el proceso de gestión de comentarios y/o sugerencias

Dentro del componente *frontend*, el usuario de tipo administrador puede gestionar los comentarios y/o sugerencias que se han generado por parte del usuario cliente. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* respectivamente.

Consumo de *endpoints* para seleccionar el pedido como entregado

Dentro del componente *frontend*, el usuario de tipo empleado puede seleccionar el pedido como entregado. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* el cual permite seleccionar el pedido como entregado.

Consumo de *endpoints* para visualizar productos favoritos

Dentro del componente *frontend*, el usuario de tipo cliente puede visualizar los productos favoritos. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* respectivamente.

Consumo de *endpoints* para el proceso de gestión del carrito de compras

Dentro del componente *frontend*, el usuario de tipo cliente puede gestionar el carrito de compras. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite gestionar el carrito de compras. Asi mismo, para realizar modificaciones en el carrito de compras, el *frontend* debe consumir el *endpoint* respectivo.

Consumo de *endpoints* para la visualización del detalle del pedido

Dentro del componente *frontend*, el usuario de tipo administrador, empleado y cliente pueden visualizar el detalle de los pedidos. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite visualizar el detalle de los pedidos existentes.

Consumo de *endpoints* para el historial de pedidos

Dentro del componente *frontend*, el usuario de tipo administrador, empleado y cliente pueden visualizar el historial de los pedidos. Para lograrlo, el *frontend* debe consumir el *endpoint* que se ha generado por el componente *backend* que permite visualizar el historial de los pedidos existentes.

Consumo de *endpoints* para el proceso de envió de comentarios y/o sugerencias

Dentro del componente *frontend*, el usuario de tipo cliente puede enviar comentarios y/o sugerencias. Para lograrlo, el *frontend* debe consumir el *endpoint* generado por el componente *backend* que permite enviar comentarios y/o sugerencias.

Para terminar, en la **Figura 3.1** se exhibe un gráfico que muestra todas las funcionalidades de los usuarios de tipo administrador, empleado y cliente, con el propósito de mejorar la comprensión y el entendimiento de las funcionalidades disponibles.

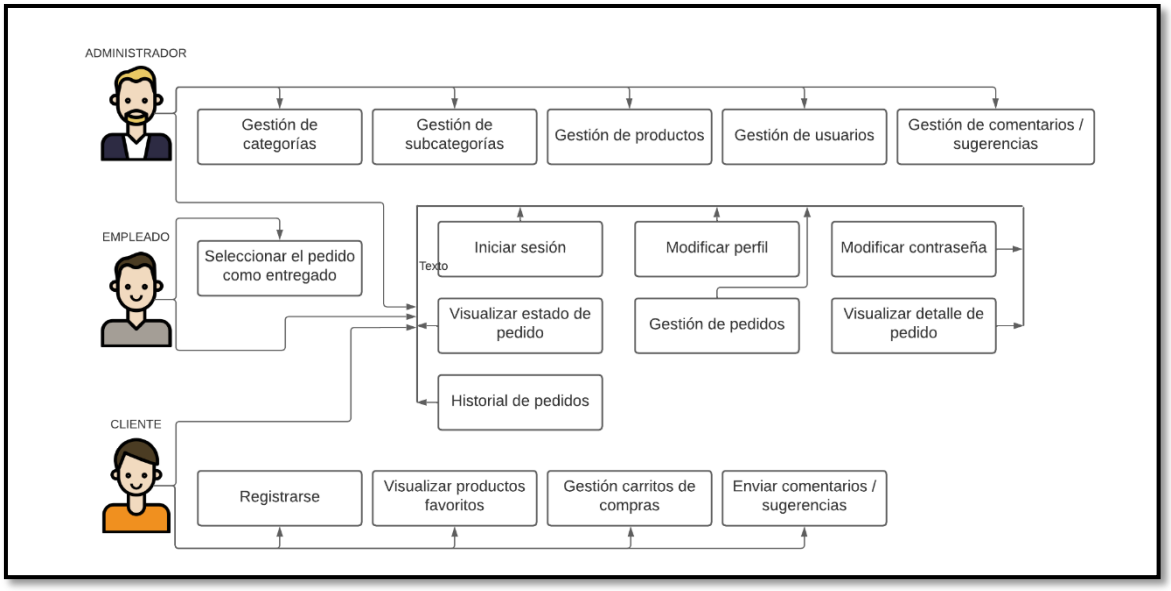

Figura 3.1 Tipos de usuarios y sus funcionalidades.

Estructura del proyecto *frontend*

El componente *frontend* ha sido desarrollado utilizando Visual Studio Code, un entorno de desarrollo que ha sido fundamental en la creación de la estructura de módulos, archivos, *scripts*, archivos de configuración y directorios necesarios para el desarrollo del *frontend*. En la **Figura 3.2** se muestra la organización del proyecto *frontend*, proporcionando una visión completa de la estructura del proyecto.

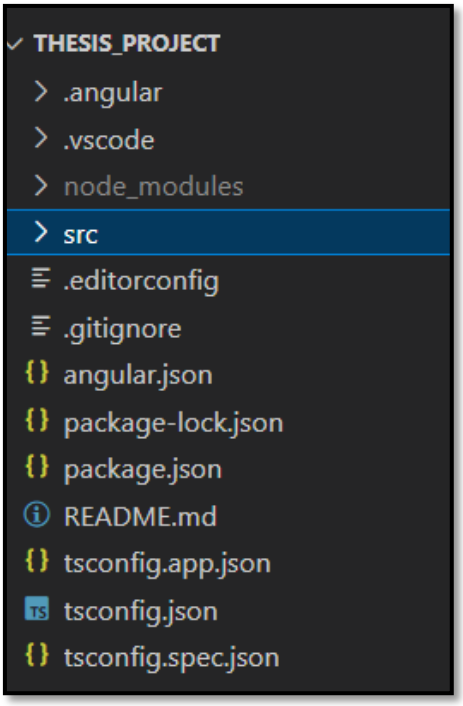

Figura 3.2 Estructura de directorios y archivos para el proyecto *frontend*.

Roles de usuario

A continuación, en la **Figura 3.3** se muestran los roles disponibles en el *frontend*, los cuales son: administrador, empleado y cliente. Esta representación visual tiene como objetivo mejorar la comprensión y el entendimiento de los diferentes roles y sus respectivas funcionalidades asociadas.

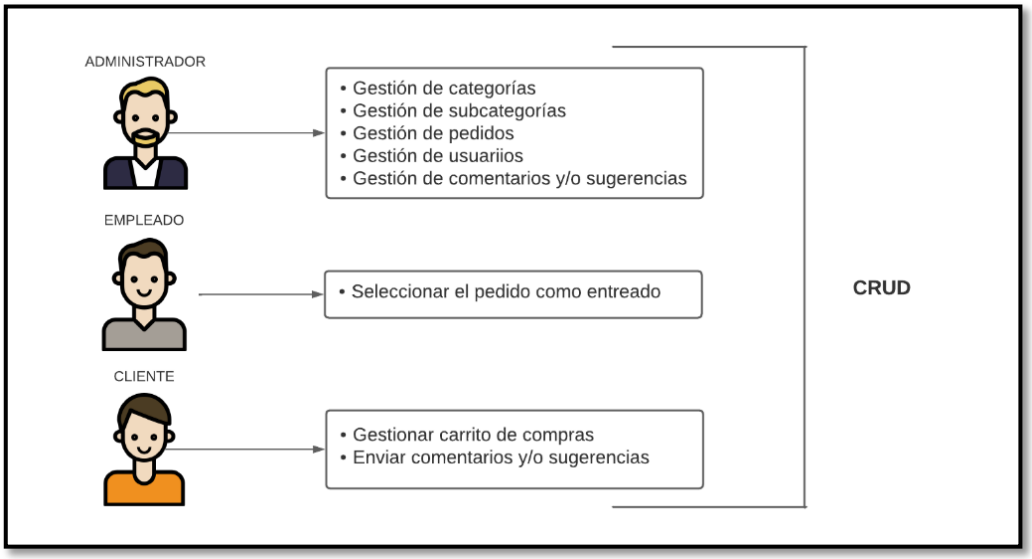

Figura 3.3 Roles y funcionalidades.

3.1 *Sprint* 1. Interfaces para el usuario de tipo administrador

El Sprint 1 se compone de las siguientes tareas:

- Consumo de *endpoints* para el proceso de iniciar sesión, cerrar sesión y modificar contraseña.
- Consumo de *endpoints* para el proceso de modificar y observar el perfil de usuario.
- Consumo de *endpoints* para el proceso de gestión de categorías.
- Consumo de *endpoints* para el proceso de gestión de subcategorías.
- Consumo de *endpoints* para el proceso de gestión de productos.
- Consumo de *endpoints* para el proceso de gestión de pedidos.
- Consumo de *endpoints* para el proceso de visualizar detalle de pedidos.
- Consumo de *endpoints* para el proceso de visualizar historial de pedidos.
- Consumo de *endpoints* para el proceso de visualizar estado de pedidos.
- Consumo de *endpoints* para el proceso gestión de comentarios y/o sugerencias.

Consumo de *endpoints* para el proceso de iniciar sesión, cerrar sesión y modificar contraseña.

En el *frontend*, el administrador puede iniciar sesión utilizando un email y contraseña proporcionados por el *backend*. Por otro lado, los empleados pueden iniciar sesión utilizando credenciales generadas por el administrador. Los usuarios clientes, por su parte, pueden iniciar sesión utilizando las credenciales con las que se han registrado. Una vez que los usuarios se encuentren dentro del *frontend*, tienen la opción de cerrar sesión, así como modificar o restablecer su contraseña. Para esto, deben completar los campos como: email, nueva contraseña y repetir contraseña. Estas funcionalidades son posibles gracias a los *endpoints* que se han generado en el *backend*. En las **Figura 3.4**, **Figura 3.5** y **Figura 3.6** se muestran los componentes visuales del *frontend*, mientras que en las **Figura 3.7**, **Figura 3.8** y **Figura 3.9** se presentan los resultados de las pruebas unitarias y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

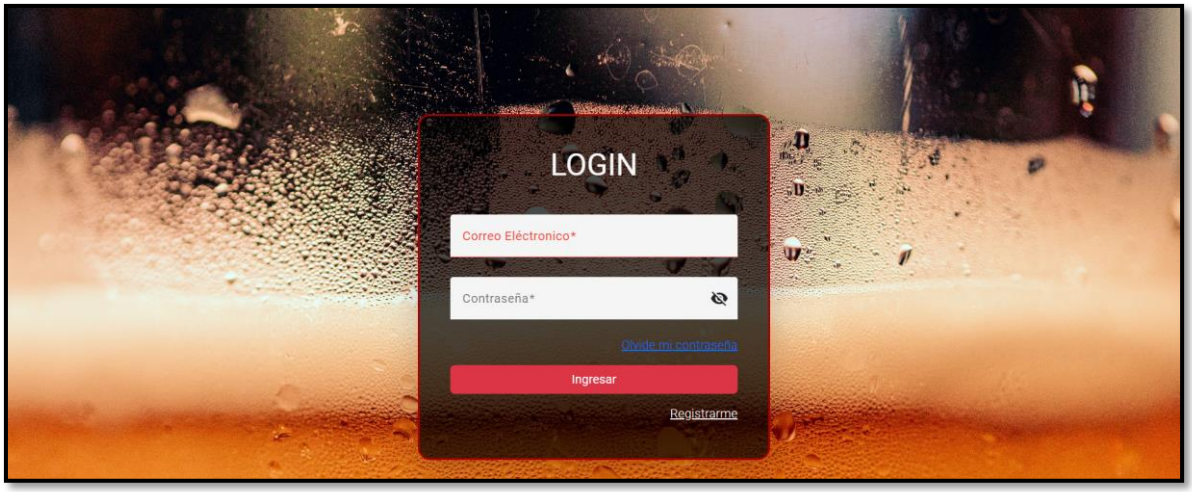

Figura 3.4 Proceso de inicio de sesión.

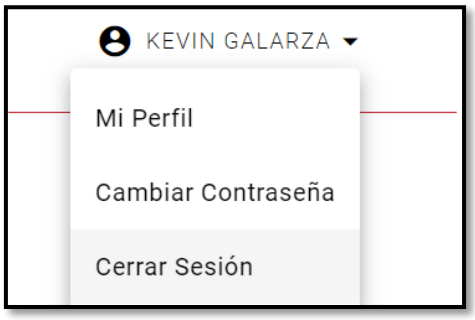

Figura 3.5 Proceso de cierre de sesión.

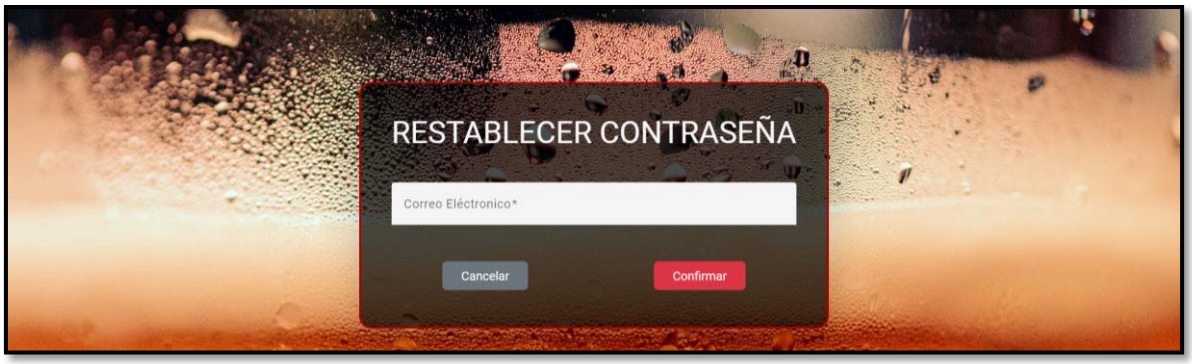

Figura 3.6 Proceso de modificación de contraseña.

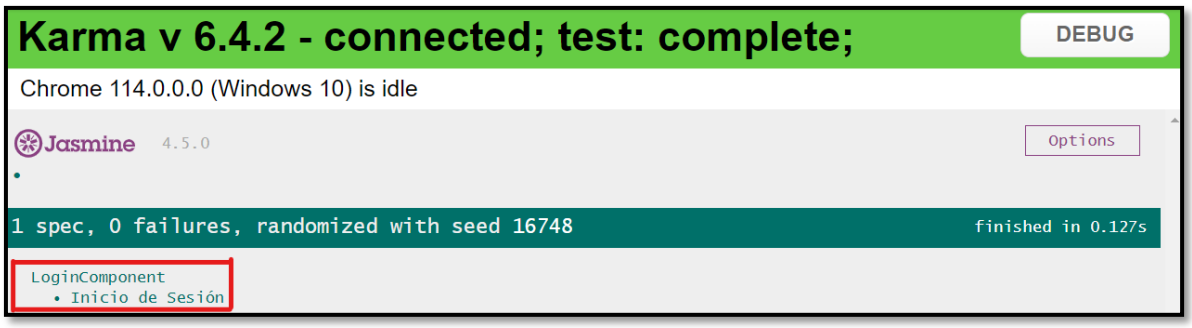

Figura 3.7 *Test* unitario del proceso de inicio de sesión.

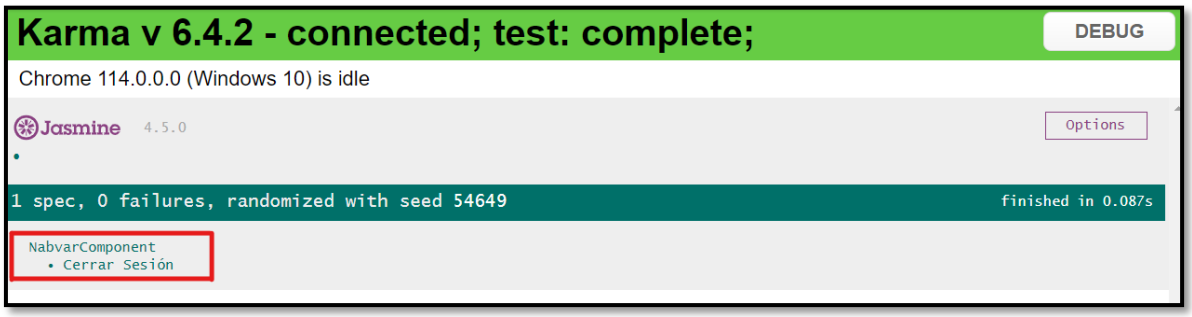

Figura 3.8 *Test* unitario del proceso de cierre de sesión.

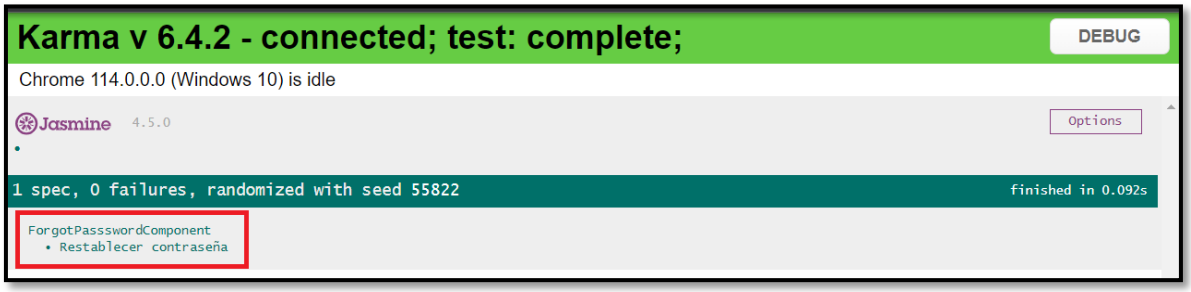

Figura 3.9 *Test* unitario del proceso de modificación de contraseña.

Consumo de *endpoints* para el para el proceso de modificar y observar el perfil de usuario

Dentro del *frontend*, los usuarios de tipo administrador, empleado y cliente tienen la capacidad de ver y editar su perfil. Este perfil incluye campos como nombres, apellidos, cédula, fecha de nacimiento, correo electrónico, celular y dirección. Todas estas acciones son posibles gracias a los *endpoints* que se han generado en el *backend*. En la **Figura 3.10** se puede observar el componente visual del *frontend* relacionado con el perfil, mientras que en la **Figura 3.11** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

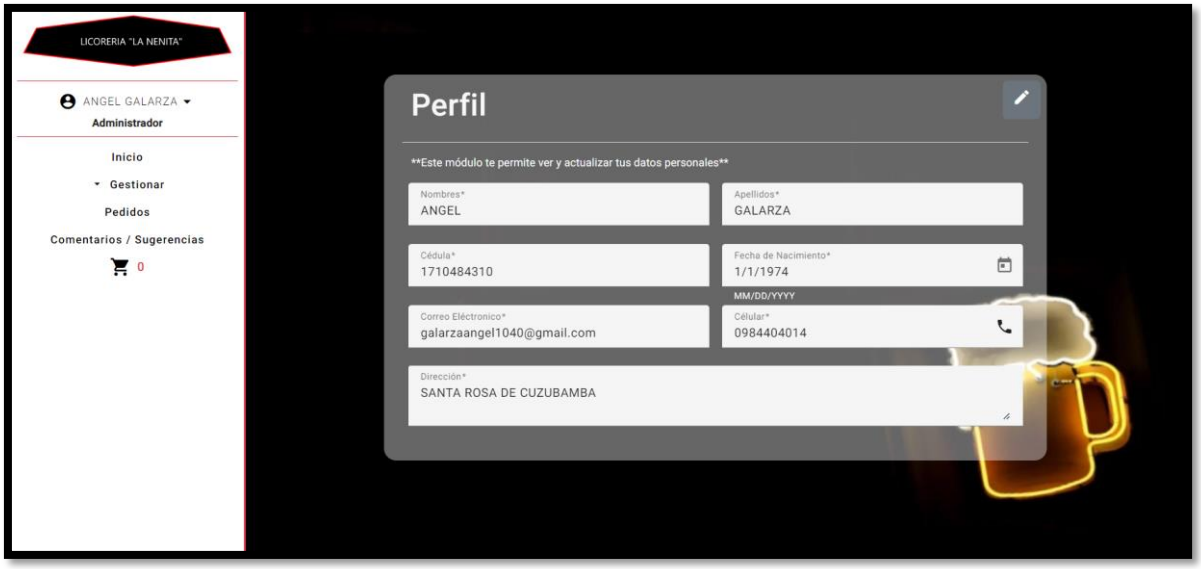

Figura 3.10 Proceso de modificar y observar el perfil de usuario.

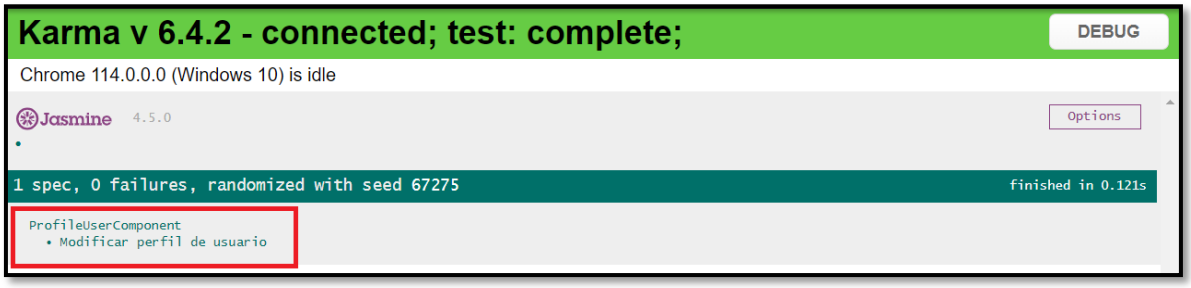

Figura 3.11 *Test* unitario del proceso de modificar y observar el perfil de usuario.

Consumo de *endpoints* para el proceso de gestión de categorías

En la parte del *frontend*, los usuarios de tipo administrador tienen la capacidad de gestionar categorías. Esto implica listar, buscar, registrar, modificar o eliminar categorías, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.12** se presentan el componente visual relacionado con la funcionalidad de listar categorías, en la **Figura 3.13** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

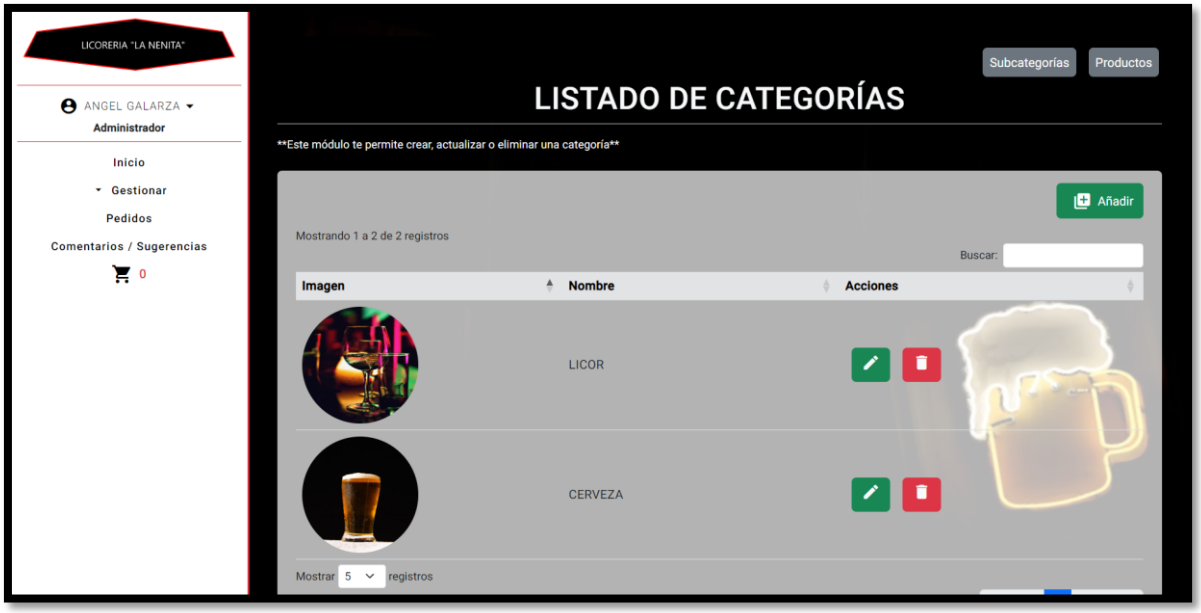

Figura 3.12 Proceso de gestión de categorías.

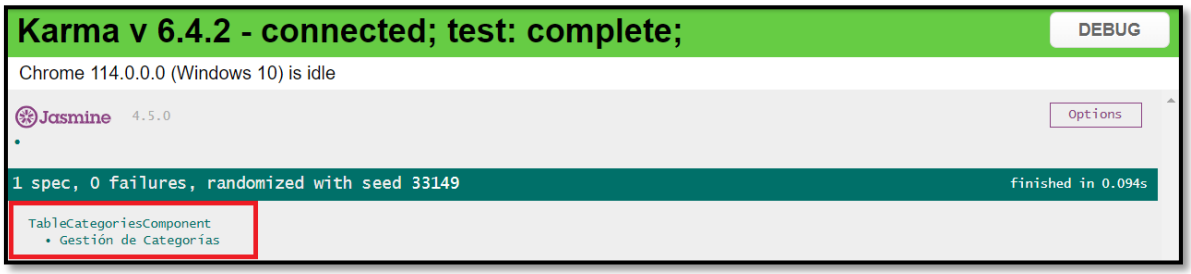

Figura 3.13 *Test* unitario del proceso de gestión de categorías.

Consumo de *endpoints* para el proceso de gestión de subcategorías

En la parte del *frontend*, los usuarios de tipo administrador tienen la capacidad de gestionar subcategorías. Esto implica listar, buscar, registrar, modificar o eliminar subcategorías, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.14** se presentan el componente visual relacionado con la funcionalidad de listar subcategorías, en la **Figura 3.15** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

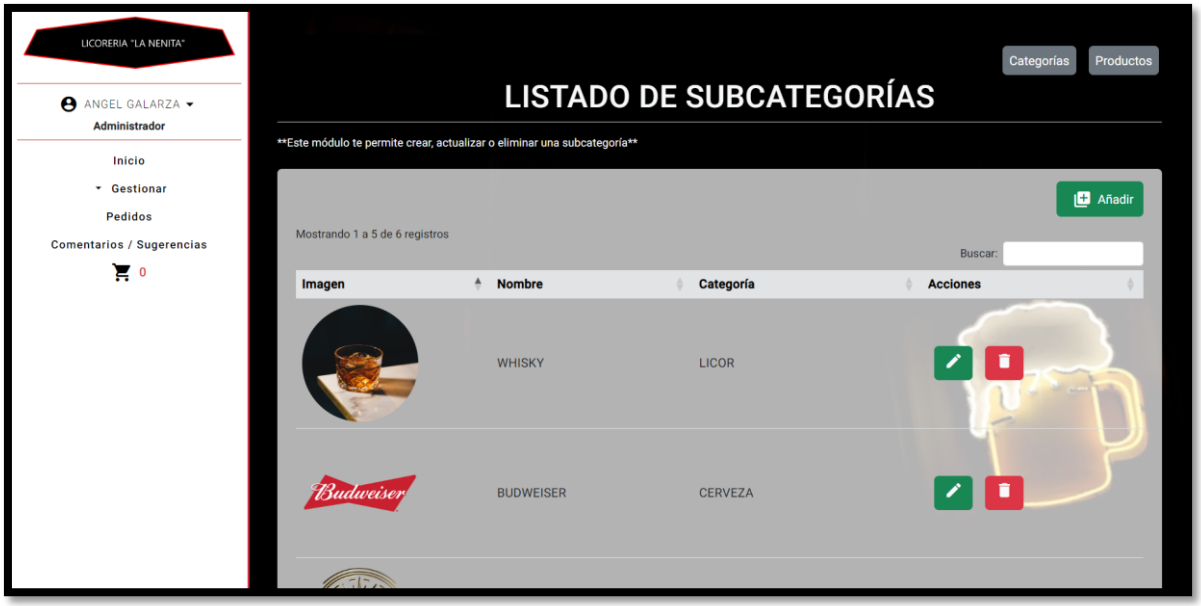

Figura 3.14 Proceso de gestión de subcategorías.

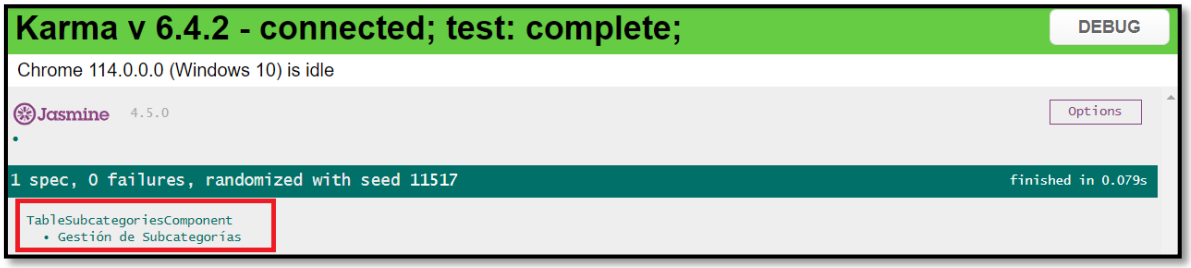

Figura 3.15 *Test* unitario del proceso de gestión de subcategorías.

Consumo de *endpoints* para el proceso de gestión de productos

En la parte del *frontend*, los usuarios de tipo administrador tienen la capacidad de gestionar productos. Esto implica listar, buscar, registrar, modificar o eliminar productos, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.16** se presentan el componente visual relacionado con la funcionalidad de listar productos, en la **Figura 3.17** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

Figura 3.16 Proceso de gestión de productos.

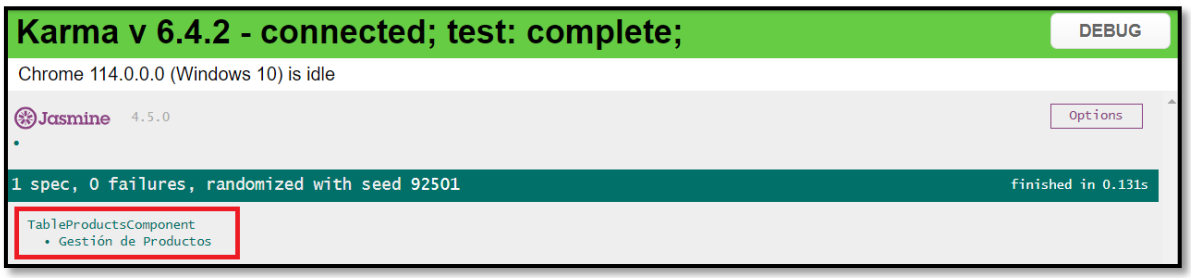

Figura 3.17 Test unitario del proceso de gestión de productos.

Consumo de *endpoints* para el proceso de gestión de pedidos

En la parte del *frontend*, los usuarios de tipo administrador tienen la capacidad de gestionar los pedidos. Esto implica listar, cambiar de estado, asignar a un empleado y dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. Por otro lado, el cliente solo puede cambiar de estado el pedido a anulado. En la **Figura 3.18** se presentan el componente visual relacionado con la funcionalidad de listar pedidos, en la **Figura 3.19** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

Figura 3.18 Proceso de gestión de pedidos.

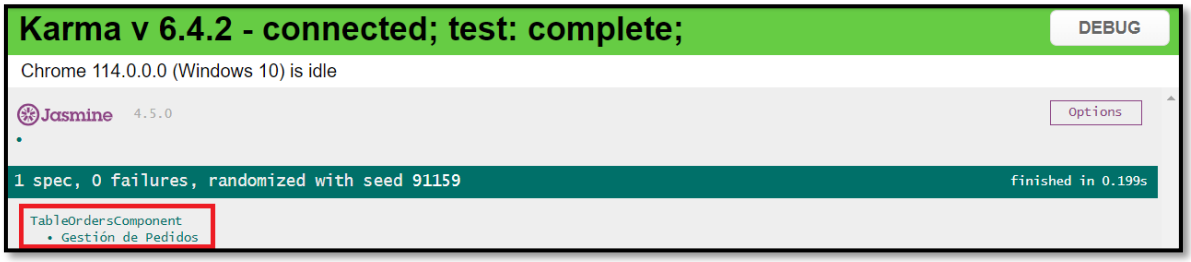

Figura 3.19 *Test* unitario del proceso de gestión de pedidos.

Consumo de *endpoints* para el proceso de visualizar detalle de pedidos

En la parte del *frontend*, los usuarios de tipo administrador, empleado y cliente tienen la capacidad de visualizar el detalle de los pedidos. Esto implica listar los detalles de cada pedido, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.20** se presentan el componente visual relacionado con esta funcionalidad, en la **Figura 3.21** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

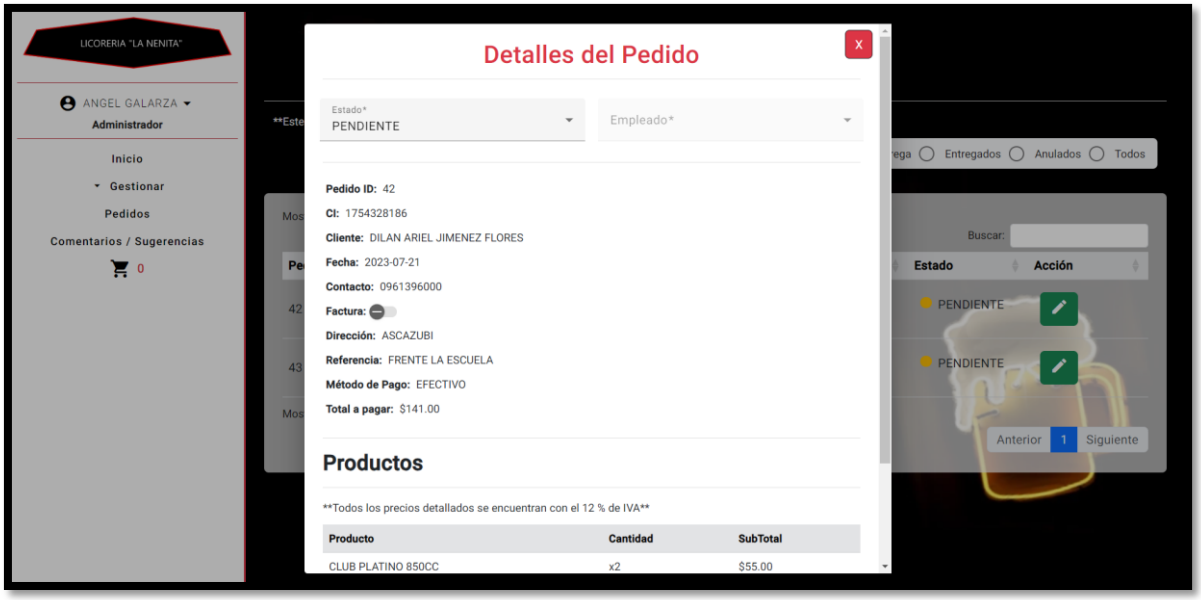

Figura 3.20 Proceso de visualizar detalle de pedidos.

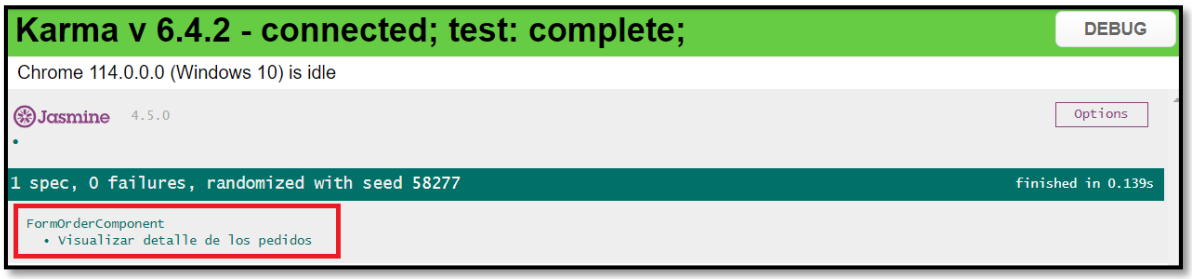

Figura 3.21 *Test* unitario del proceso de visualizar detalle de pedidos.

Consumo de *endpoints* para el proceso de visualizar historial de pedidos

En la parte del *frontend*, los usuarios de tipo administrador, empleado y cliente tienen la capacidad de visualizar historial de pedidos. Esto implica listar los pedidos existentes, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.22** se presentan el componente visual relacionado con esta funcionalidad, en la **Figura 3.23** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

Figura 3.22 Proceso de visualizar historial de pedidos.

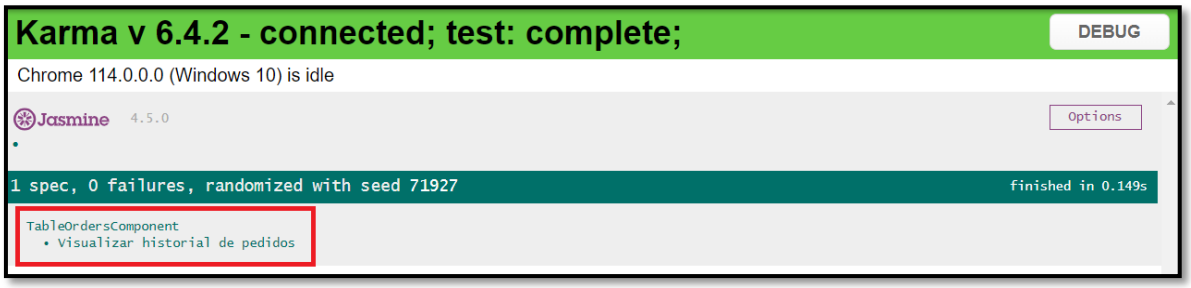

Figura 3.23 *Test* unitario del proceso de visualizar historial de pedidos.

Consumo de *endpoints* para el proceso de visualizar estado de pedidos

En la parte del *frontend*, los usuarios de tipo administrador, empleado y cliente tienen la capacidad de visualizar el estado de los pedidos. Esto implica ver el estado de los pedidos existentes, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.24** se presenta el componente visual relacionado con esta funcionalidad, en la **Figura 3.25** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

Figura 3.24 Proceso de visualizar estado de pedidos.

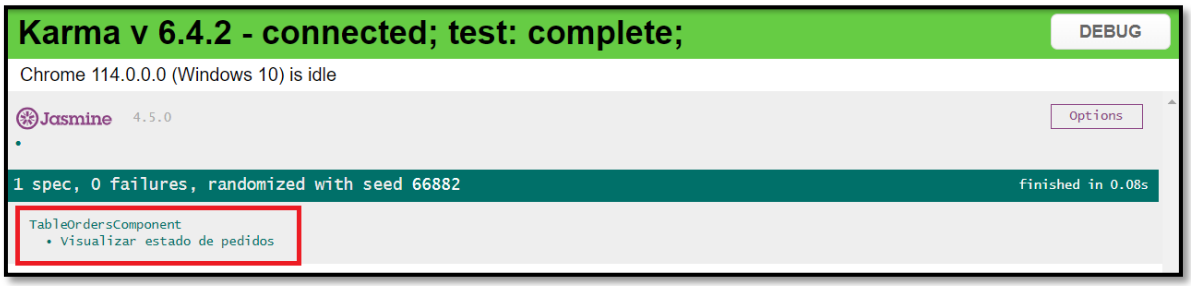

Figura 3.25 *Test* unitario del proceso de visualizar estado de pedidos.

Consumo de *endpoints* para el proceso de gestión de comentarios y/o sugerencias

En la parte del *frontend*, los usuarios de tipo administrador tienen la capacidad de gestionar comentarios y/o sugerencias. Esto implica ver el listado de los comentarios y/o sugerencias que han sido generados por los usuarios de tipo cliente, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.26** se presentan el componente visual relacionado con esta funcionalidad, en la **Figura 3.27** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

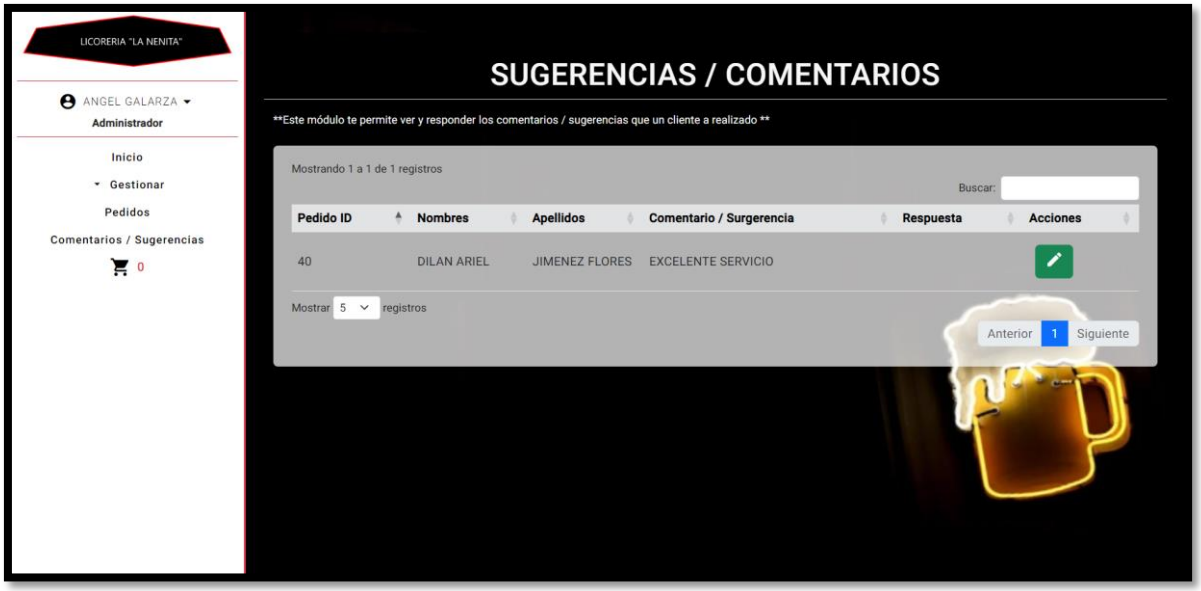

Figura 3.26 Proceso de gestión de comentarios y/o sugerencias.

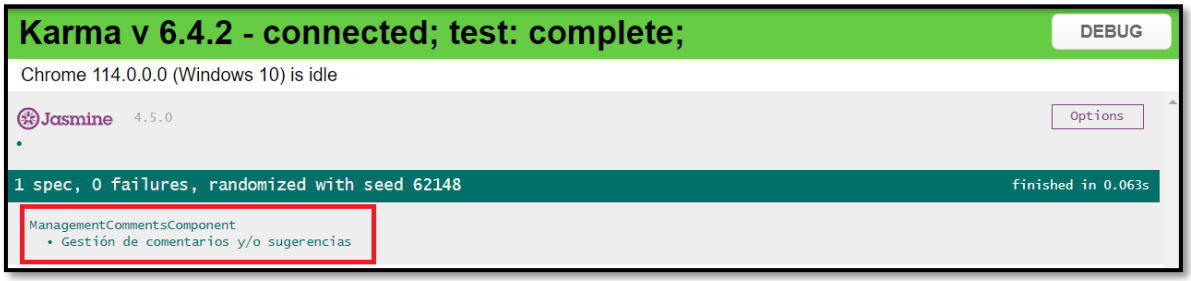

Figura 3.27 *Test* unitario del proceso de gestión de comentarios y/o sugerencias.

3.2 *Sprint* 2. Interfaces para el usuario de tipo empleado

El *Sprint* 2 se compone de la siguiente tarea:

- Consumo de *endpoints* para el proceso de seleccionar el pedido como entregado.

Consumo de *endpoints* para el proceso de seleccionar el pedido como entregado

En la parte del *frontend*, los usuarios de tipo empleado tienen la capacidad de seleccionar el pedido como entregado. Esto implica cambiar el estado del pedido a entregado, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.28** se presenta el componente visual relacionado con esta funcionalidad, en la **Figura 3.29** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

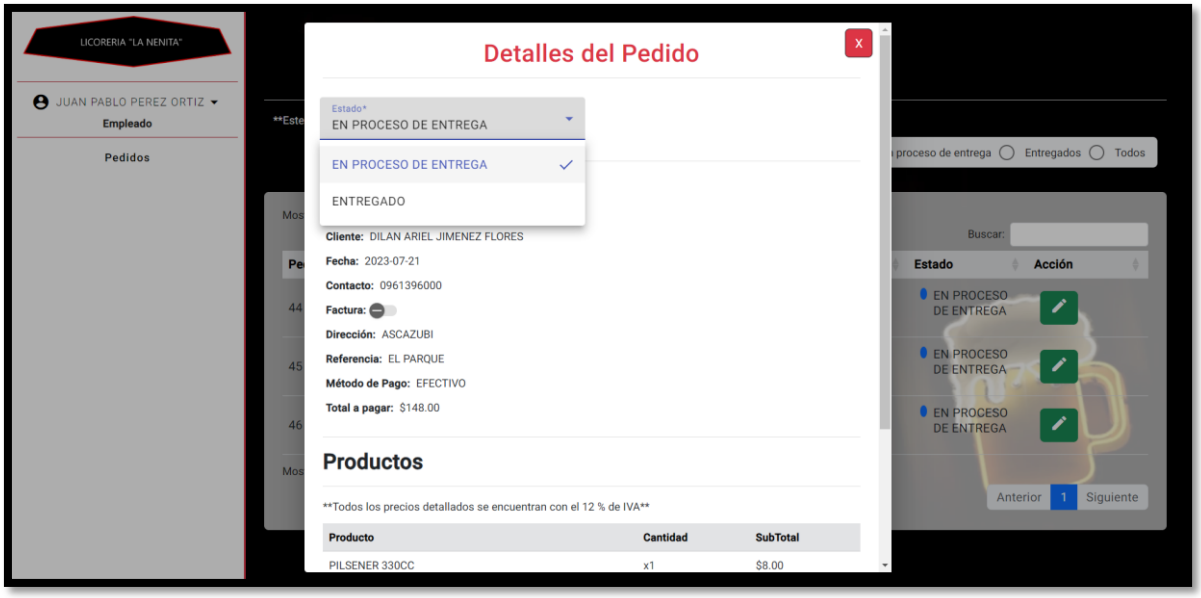

Figura 3.28 Proceso de gestión de seleccionar el pedido como entregado.

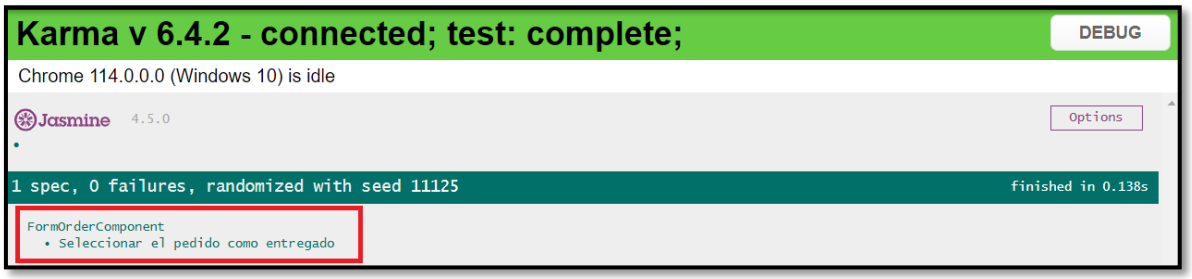

Figura 3.29 *Test* unitario del proceso de seleccionar el pedido como entregado.

3.3 *Sprint* 3. Interfaces para el usuario de tipo cliente

El *Sprint* 3 se compone de las siguientes tareas:

- Consumo de *endpoints* para el proceso de visualizar productos favoritos.
- Consumo de *endpoints* para el proceso de gestionar carrito de compras.
- Consumo de *endpoints* para el proceso de enviar comentarios y/o sugerencias.

Consumo de *endpoints* para el proceso de visualizar productos favoritos

En la parte del *frontend*, los usuarios de tipo cliente tienen la capacidad de visualizar los productos favoritos. Esto implica ver los productos más adquiridos por el cliente, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.30** se presenta el componente visual relacionado con esta funcionalidad, en la

Figura 3.31 se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

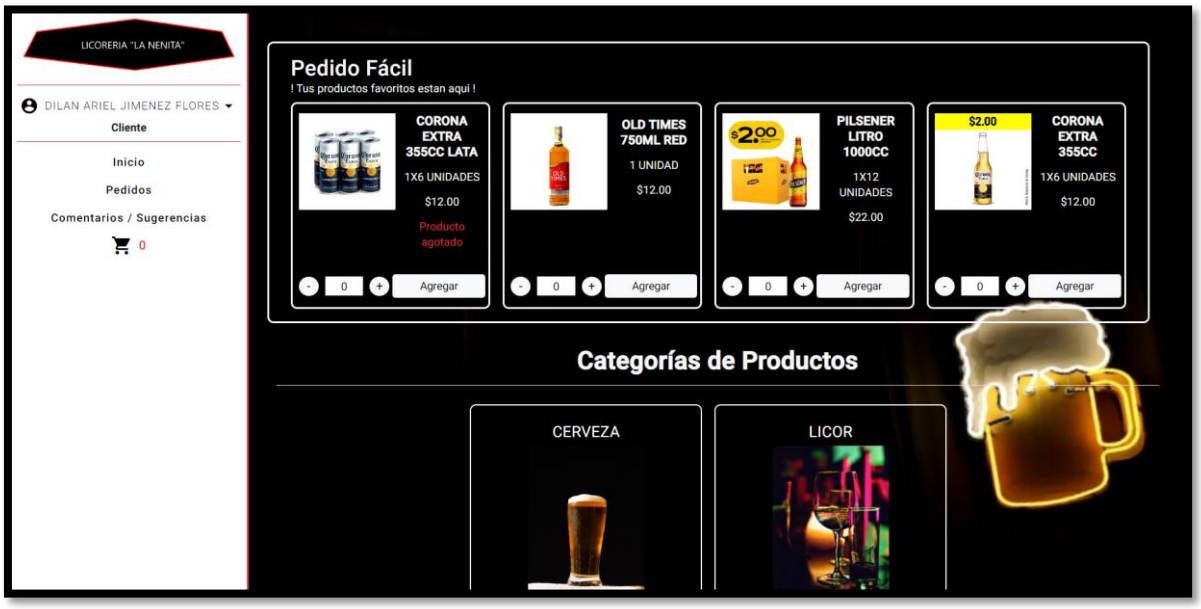

Figura 3.30 Proceso de gestión de visualizar productos favoritos.

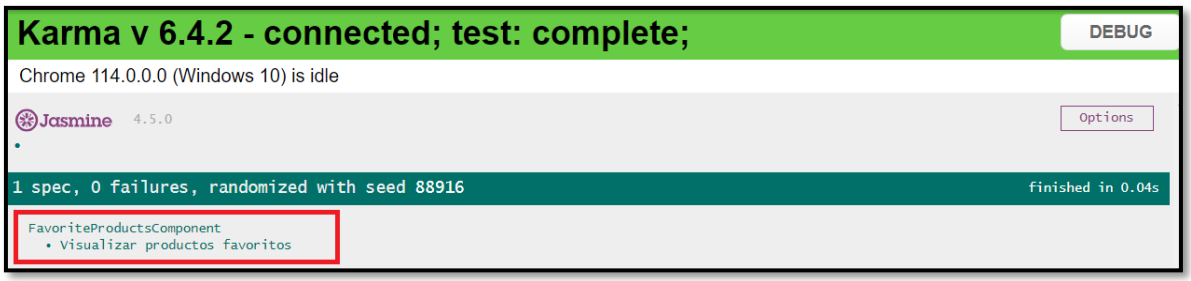

Figura 3.31 *Test* unitario del proceso de visualizar productos favoritos.

Consumo de *endpoints* para el proceso de gestionar carrito de compras

En la parte del *frontend*, los usuarios de tipo cliente tienen la capacidad de gestionar el carrito de compras. Esto implica ver, agregar, modificar la cantidad o el producto a adquirir, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.32** se presenta el componente visual relacionado con esta funcionalidad, en la **Figura 3.33** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO II**.

Figura 3.32 Proceso de gestión carrito de compras.

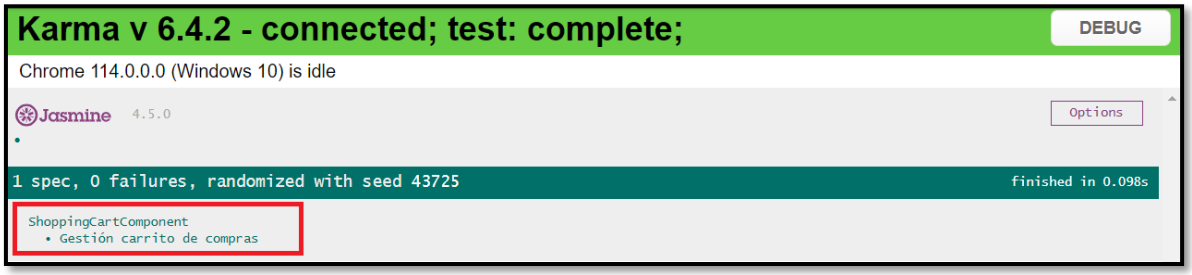

Figura 3.33 *Test* unitario del proceso de gestión carrito de compras.

Consumo de *endpoints* para el proceso de enviar comentarios y/o sugerencias

En la parte del *frontend*, los usuarios de tipo cliente tienen la capacidad de enviar comentarios y/o sugerencias. Esto implica generar comentarios o sugerencias que el administrador puede observar, dichas acciones se realizan utilizando los *endpoints* que se han generado en el *backend*. En la **Figura 3.34** se presenta el componente visual relacionado con esta funcionalidad, en la **Figura 3.35** se muestra el resultado de la prueba unitaria y para obtener un detalle más completo sobre el funcionamiento de esta tarea se encuentran en el **ANEXO III**.

Figura 3.34 Proceso de gestión de enviar comentarios y/o sugerencias.

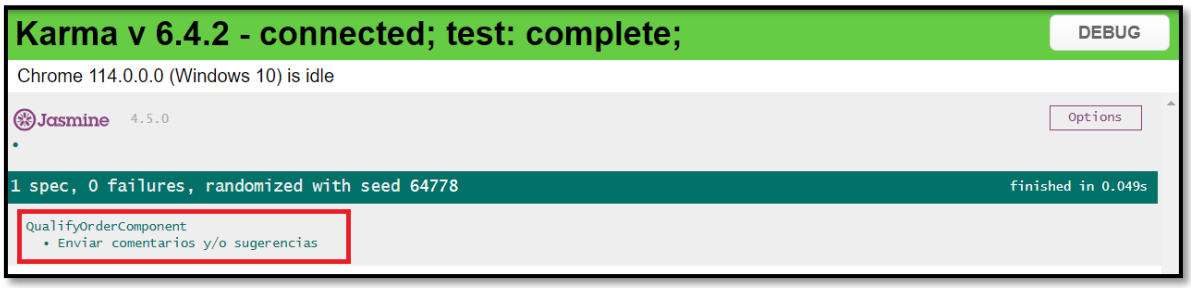

Figura 3.35 *Test* unitario del proceso de enviar comentarios y/o sugerencias.

3.4 *Sprint* 4. Pruebas para el componente *frontend.*

A continuación, el *Sprint* 4 se compone de las siguientes tareas:

- Realizar y proporcionar detalles sobre los resultados de las pruebas unitarias.
- Realizar y proporcionar detalles sobre los resultados de las pruebas de compatibilidad.
- Realizar y proporcionar detalles sobre los resultados de las pruebas de aceptación.
- Realizar y proporcionar detalles sobre los resultados de las pruebas de rendimiento.

Realizar y proporcionar detalles sobre los resultados de las pruebas unitarias

Las pruebas unitarias son pruebas que se realizan para verificar el correcto funcionamiento de cada uno de los componentes de forma aislada. Estas pruebas permiten evaluar el

código y asegurar que el mismo sea más confiable. Además, al ejecutar las pruebas unitarias, se puede identificar posibles errores y garantizar que los componentes se comporten según lo esperado, corrigiendo posibles vulnerabilidades o fallos en el código, lo que aumenta la robustez del sistema *software* [37].

La **Figura 3.36** muestra el código que se ha utilizado para implementar una de las pruebas unitarias. Además, la **Figura 3.37** muestra el resultado que se ha obtenido de dicha prueba y, por último, se puede encontrar un detalle completo de todos los resultados en el **ANEXO II.**

```
describe('LadingPageComponent', () => {
  let component: LadingPageComponent;
  let fixture: ComponentFixture<LadingPageComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ LadingPageComponent ],
      imports:[FormsModule, ReactiveFormsModule]
    })
    .compileComponents();

    fixture = TestBed.createComponent(LadingPageComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();      You, 2 months ago • Ladingpa
  });

  it('Página Informativa', () => {
    expect(component).toBeTruthy();
  });
});
```

Figura 3.36 Código ejemplar para la prueba unitaria.

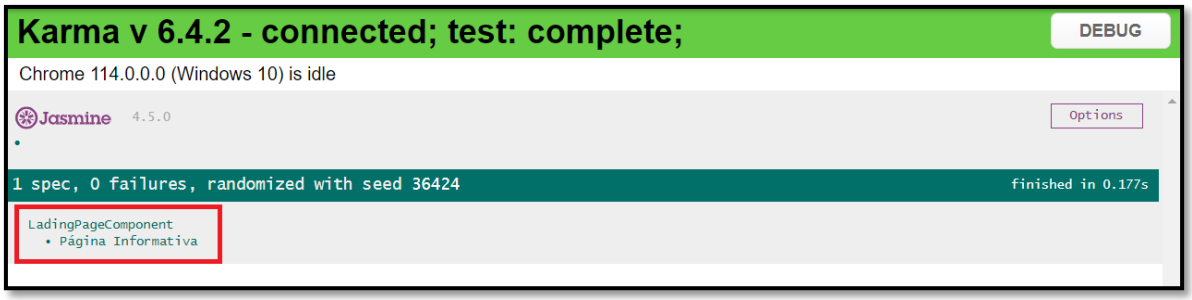

Figura 3.37 Resultado ejemplar obtenido de la prueba unitaria.

Después de llevar a cabo la prueba mencionada, se ha comprobado que todos los módulos operan de manera adecuada, dado que no se han detectado fallos en ninguna parte del código. Esto sugiere que las correcciones efectuadas han sido exitosas, logrando un funcionamiento adecuado en cada uno de los módulos.

Realizar y proporcionar detalles sobre los resultados de las pruebas de compatibilidad

Una prueba de compatibilidad es un proceso en el que se evalúa la capacidad de una aplicación o *software* para funcionar correctamente en diferentes entornos y dispositivos. Implica verificar si la aplicación es compatible con varios sistemas operativos, navegadores *web*, versiones de *software* y configuraciones de *hardware* [38]. Debido a esto, se ha llevado a cabo la prueba de compatibilidad en los navegadores mencionados en la **Tabla 3.1**. Además, en el **ANEXO II** se incluye información detallada sobre las demás pruebas de compatibilidad que se han realizado, junto con los resultados respectivos.

Tabla 3.1 Prueba de compatibilidad.

NAVEGADOR	VERSIÓN	OBSERVACIÓN
Brave	1.52.129	Operativamente sin problemas
Google Chrome	114.0.5735.199	Operativamente sin problemas
Microsoft Edge	117.0.2045.31	Operativamente sin problemas

Después de llevar a cabo la prueba mencionada, se ha comprobado que todos los módulos operan de manera adecuada en diversos entornos. Esto sugiere que las correcciones efectuadas han sido exitosas, logrando un funcionamiento adecuado de cada uno de los módulos en diferentes navegadores.

Realizar y proporcionar detalles sobre los resultados de las pruebas de aceptación

Estas pruebas se centran en determinar si un sistema de *software* cumple con los requisitos y expectativas que se han establecido por el propietario del producto final, y al mismo tiempo, asegurar que se ha desarrollado de acuerdo a los requerimientos que se han definido al inicio del proyecto [39]. Por esta razón, en la **Tabla 3.2** se muestra claramente la prueba de aceptación basada en los requisitos que se han establecido por el usuario, mientras que en el **ANEXO II** se proporciona información detallada sobre las demás pruebas de aceptación, junto con los resultados respectivos.

Tabla 3.2 Prueba de aceptación – Gestión de categorías.

PRUEBA DE ACEPTACIÓN	
Identificador (ID): PA005	**Identificador de historia de Usuario:** HU005
Nombre: Gestionar las categorías	
Descripción: El usuario administrador en el *frontend* debe consumir varios *endpoints* para: • Registrar categorías de productos. • Modificar categorías de productos. • Eliminar categorías de productos. • Visualizar lista de categorías de productos.	
Pasos de ejecución: Para gestionar las categorías: • Inicio de sesión como usuario de tipo administrador. • Visualizar el listado de las categorías. • Crear nuevas categorías. • Editar categorías existentes. • Eliminar categorías existentes.	
Resultado deseado: El *frontend* admite crear, actualizar, eliminar y listar categorías.	
Evaluación de la prueba: Resultado y conformidad del cliente al 100%.	

Después de llevar a cabo la prueba mencionada, se ha comprobado que todos los módulos cumplen con los requerimientos recopilados al inicio del proyecto. Además, cada uno de estos se encuentran aprobados por el *Product Owner*.

Realizar y proporcionar detalles sobre las pruebas de rendimiento

Las pruebas de rendimiento son evaluaciones que se llevan a cabo con el objetivo de medir el rendimiento que un sistema tiene al realizar una tarea específica en condiciones

particulares de funcionamiento [40]. Debido a esto, se ha llevado a cabo la prueba de rendimiento a través de la herramienta *PageSpeed* como se muestra en la **Figura 3.38**. Además, en el **ANEXO II** se incluye información detallada sobre esta prueba.

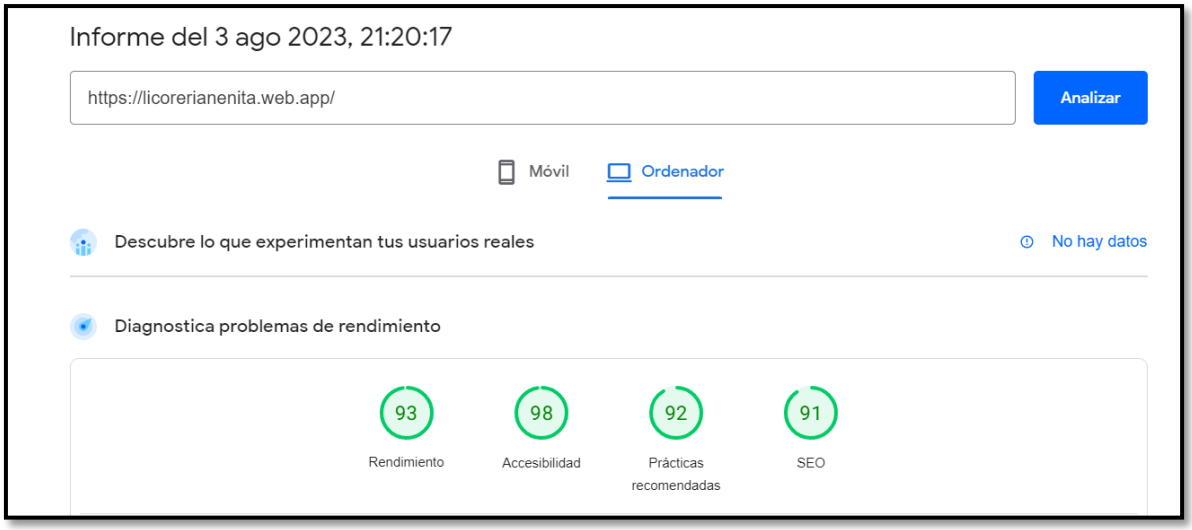

Figura 3.38 Resultado ejemplar de la prueba de rendimiento.

Después de llevar a cabo la prueba mencionada, se ha comprobado que todos los módulos operan con un rendimiento adecuado. Esto sugiere que las correcciones efectuadas han sido exitosas, logrando un correcto rendimiento de cada uno de los módulos.

3.4 *Sprint* 5. Despliegue hacia un entorno de producción

En esta tarea se hace el despliegue a producción del *frontend* haciendo uso de la herramienta *Firebase* y la URL para acceder al mismo es la siguiente:

https://licorerianenita.web.app/

Por último, el administrador de la Licoreria “La Nenita” ha otorgado un certificado que confirma el cumplimiento de todos los requisitos y funcionalidades del *frontend* que han sido solicitados al inicio del proyecto. El certificado se encuentra adjunto en el **ANEXO II**.

4 CONCLUSIONES

En esta sección, se presentan las conclusiones que se han alcanzado durante el desarrollo del proyecto de Integración Curricular.

- Los objetivos y el alcance que se han establecido como parte del proyecto han sido logrados satisfactoriamente. Como resultado, tanto el usuario administrador, empleado y cliente pueden realizar diversas acciones según su rol en el *frontend* logrando que la gestión de venta de licores se realice de forma adecuada por medio de la tecnología.

- El desarrollo del componente *frontend* se ha logrado finalizar en el tiempo que se ha establecido gracias a la integración de la SC.

- El establecimiento de los requisitos al comienzo del proyecto ha resultado fundamental, ya que ha permitido una selección adecuada de los artefactos, planificación de cada uno de los *sprints* y herramientas necesarias para cada una de las diferentes etapas de codificación, pruebas y despliegue.

- Mediante una clara separación de responsabilidades en el modelo, vista y controlador, la implementación del patrón MVC ha facilitado un desarrollo organizado en el proyecto en cuanto al desarrollo del componente *frontend*.

- Una adecuada implementación de librerías como herramientas para la compilación del *frontend* ha resultado de gran utilidad debido a su amplia variedad de componentes predefinidos. Estos componentes se encuentran listos para ser utilizados, lo cual agiliza considerablemente el desarrollo del *frontend*, permitiendo un enfoque más eficiente en la creación de interfaces de usuario y mejorando la experiencia del usuario final.

- Cada una de las pruebas que se han realizado al componente *frontend* han permitido corroborar el correcto funcionamiento de cada uno de los módulos, presentación del contenido, optimización y la aceptación del producto final.

- Al conectar el componente *frontend* con el componente *backend* a través del uso de APIs se ha logrado intercambiar información con facilidad, enviando y recibiendo datos según los módulos disponibles para cada tipo de usuario.

5 RECOMENDACIONES

En esta sección, se presentan las recomendaciones durante el desarrollo del proyecto de Integración Curricular.

- Antes de empezar a usar la aplicación por el lado del cliente, es recomendable capacitar a todos y cada uno de los usuarios para que realicen peticiones de manera correcta contribuyendo asi a una mejor experiencia de uso.
- Se sugiere desarrollar una versión móvil del *frontend* ya que a pesar de ser *responsive*, esta versión puede ofrecer una experiencia mejorada a los posibles clientes.
- Es aconsejable establecer políticas de seguridad en la gestión de la información desde el *frontend*.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] I. Jiménez. “Licorería la nenita”. Facebook. https://www.facebook.com/profile.php?id=100063883096253&mibextid=ZbWKwL.

[2] NIQ. "Rebalancing the COVID-19 effect on alcohol sales". NIQ. https://nielseniq.com/global/en/insights/analysis/2020/rebalancing-the-covid-19-effect-on-alcohol-sales.

[3] Á. Hódar. "8 Ventajas de tener una tienda online o e-Commerce”. Baética. https://baetica.com/ventajas-tienda-online.

[4] M. Mulford, L. Vergara y D. Plata de Plata. "Tienda virtual: social market Colombia". Multiciencias, vol. 14, n.º 3, p. 273, 2014.

[5] Horizonte. "La importancia de las tiendas Online o E-commerce". Horizonte. https://soyhorizonte.com/blog/la-importancia-de-las-tiendas-online-o-e-commerce.

[6] Next_u. "¿Qué es el front end y por qué tiene tanta popularidad actualmente? - Blog | NextU LATAM". NextU LATAM. https://www.nextu.com/blog/que-es-front-end-rc22.

[7] S. Pérez, J. Quispe, F. Mullicundo y D. Lamas. “HERRAMIENTAS Y TECNOLOGÍAS PARA EL DESARROLLO WEB DESDE EL FRONTEND AL BACKEND”. Chilecito: UNdeC, 2021.

[8] X. Pallerols. "Haz tu web Responsive Design y adáptala a todas las plataformas". Thinking for Innovation. https://www.iebschool.com/blog/que-es-responsive-web-design-analitica-usabilidad.

[9] R. Vara. “Introducción a HTML “. 2a ed., Ediciones Universidad de Salamanca, 2018.

[10] G. B. "¿Qué es CSS?". Tutoriales Hostinger. https://www.hostinger.es/tutoriales/que-es-css.

[11] J. Pérez. “Introducción a JavaScript”. 2a ed., Belcorp, 2009.

[12] Uniwebsidad. "Capítulo 5. Frameworks". Uniwebsidad. https://uniwebsidad.com/libros/css-avanzado/capitulo-5.

[13] S. Gaikwad. “A Review Paper on Bootstrap Framework”. 2a ed., CA Departmen, 2019.

[14] M. Gonçalves. "¿Qué es Angular y para qué sirve? - Blog de Hiberus Tecnología". Blog de Hiberus Tecnología. https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve.

[15] J. Chacón. "TypeScript: qué es, diferencias con JavaScript y por qué aprenderlo". Profile Software Services. https://profile.es/blog/que-es-typescript-vs-javascript.

[16] D. Djordjevic, S. Ollivier y W. Klein. “Desarrolle sus aplicaciones web con el framework JavaScript de Google”. Barcelona: ENI EDICIONES, 2020.

[17] J. Fatjó. "Introducción a Angular Material". Tribalyte Technologies. https://tech.tribalyte.eu/blog-introduccion-angular-material.

[18] Editorial Etecé. "HTTP - Concepto, para qué sirve y cómo funciona". Concepto. https://concepto.de/http.

[19] AWS. "¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS". Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api.

[20] Software DELSOL. "Metodología ¿Qué es?". Software DELSOL - Soluciones empresariales. https://www.sdelsol.com/glosario/metodologia.

[21] Zendesk. "¿Qué es la metodología ágil y cuáles son las más utilizadas?". Zendesk. https://www.zendesk.com.mx/blog/metodologia-agil-que-es.

[22] J. Canós y C. Penadés. "Repositorio institucional de la Universidad de Las Tunas: Métodologías Ágiles en el Desarrollo de Software". Repositorio institucional de la Universidad de Las Tunas: Home. http://roa.ult.edu.cu/handle/123456789/476.

[23] Proyectos Agiles. "Qué es SCRUM". Proyectos Ágiles. https://proyectosagiles.org/que-es-scrum.

[24] K. Schwaber y J. Sutherlan, “La guía de Scrum”. Julio 201. http://www.Scrumguides.org/docs/Scrumguide/v1/Scrum-GuideES.pdf.

[25] Asana, “Guía de 6 pasos para la recopilación de requisitos para asegurar el éxito de tu proyecto”. Available: https://asana.com/es/resources/requirementsgathering.

[26] Digite. "Historias De Usuarios: Qué Son Y Por Qué Y Cómo Usarlas". Nimblework. https://www.digite.com/es/agile/historias-de-usuarios.

[27] J. Ramos. "Scrum: ¿Qué es el Product Backlog?". Programación y más. https://programacionymas.com/blog/scrum-product-backlog.

[28] Integrait. "Sprint y Sprint Backlog: puntos esenciales de SCRUM". Integra IT Soluciones. https://integrait.com.mx/blog/sprint-y-sprint-backlog.

[29] F. Debernardi. "¿Qué es el diseño de la interfaz de usuario?" Linkedin. https://www.linkedin.com/pulse/qué-es-el-diseño-de-la-interfaz-usuario-diseñador-ui-ux/?originalSubdomain=es.

[30] CEI. "¿Qué es Figma y para qué sirve?". Escuela de Diseño y Marketing. https://cei.es/que-es-figma.

[31] Wordpress. "2.1 Arquitectura de las aplicaciones Web". Programacion Web. https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web.

[32] Mozilla. "MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN". MDN Web Docs. https://developer.mozilla.org/es/docs/Glossary/MVC.

[33] L. Bravo. “Framework o librerías: ventajas y desventajas”. Tithink. https://www.tithink.com/es/2018/08/29/framework-olibrerias-ventajas-y-desventajas.

[34] Cloud Tables. “Reference. DataTables | Table plug-in for jQuery”. Cloud Tables https://datatables.net/reference/index.

[35] Y. Chauhan. “ngx-spinner. npm”. npm. https://www.npmjs.com/package/ngx-spinner.

[36] RxJS. “RxJS Introduction”. RxJS. https://rxjs.dev/guide/overview.

[37] B. Bayhan. "¿Por qué es tan importante el testing en Frontend? | Apiumhub". Apiumhub. https://apiumhub.com/es/tech-blog-barcelona/por-que-es-tan-importante-el-testing-en-frontend.

[38] V. Gomez. "Pruebas de compatibilidad en diferentes navegadores con Coded UI". Linkedin. https://es.linkedin.com/pulse/pruebas-de-compatibilidad-en-diferentes-navegadores-con-gómez-adán.

[39] PYM. "Los diferentes tipos de Pruebas de software". Programación y más. https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software.