



# INSTITUTO POLITÉCNICO NACIONAL

## ESCUELA SUPERIOR DE CÓMPUTO



### “Proyecto: Dulcemprende”

**GRUPO:** 2CM16

**INTEGRANTES:** Briones Tapia Daniela - 2020630022<sup>1</sup>

Meza Vargas Brandon David - 2020630288<sup>2</sup>

Torres Jiménez Diego Antonio -2020630528<sup>3</sup>

**PROFESOR:** Dorantes González Marco Antonio

---

<sup>1</sup> dbrionest1400@alumno.ipn.mx

<sup>2</sup> brandonmv2001@gmail.com

<sup>3</sup> diegoantoniotj@gmail.com

# Índice

<b>Introducción.....</b>	<b>7</b>
<b>Estado del arte .....</b>	<b>8</b>
<b>Justificación .....</b>	<b>9</b>
<b>Objetivo general .....</b>	<b>9</b>
<b>Objetivos particulares.....</b>	<b>9</b>
<b>Capítulo 1.....</b>	<b>10</b>
<b>Marco Teórico .....</b>	<b>10</b>
<b>Capítulo 2.....</b>	<b>12</b>
<b>Análisis .....</b>	<b>12</b>
<b>Metodología .....</b>	<b>12</b>
<b>Técnica de Recolección de Requisitos.....</b>	<b>13</b>
<b>Arquitectura .....</b>	<b>13</b>
<b>Requerimientos del sistema .....</b>	<b>14</b>
<b>Reglas de negocio.....</b>	<b>14</b>
<b>Requerimientos funcionales .....</b>	<b>14</b>
<b>Requerimientos no funcionales .....</b>	<b>15</b>
<b>Requerimientos técnicos .....</b>	<b>16</b>
<b>Actores.....</b>	<b>16</b>
<b>Capítulo 3.....</b>	<b>17</b>
<b>Diseño del sistema .....</b>	<b>17</b>
<b>Casos de uso .....</b>	<b>17</b>
<b>Diagramas de clase .....</b>	<b>25</b>
<b>Diagramas de secuencia .....</b>	<b>28</b>
<b>Diagramas de estados.....</b>	<b>31</b>
<b>Diagramas de actividades .....</b>	<b>32</b>
<b>Capítulo 4.....</b>	<b>33</b>
<b>Codificación del sistema.....</b>	<b>33</b>
<b>Backend .....</b>	<b>33</b>
<b>Fronted .....</b>	<b>44</b>
<b>Capítulo 5.....</b>	<b>68</b>
<b>Prototipos .....</b>	<b>68</b>

<b>Administrador .....</b>	<b>68</b>	
<b>    Cliente.....</b>	<b>72</b>	
<b>Capítulo 6.....</b>	<b>81</b>	
<b>    Plan de pruebas Dulcemprende .....</b>	<b>81</b>	
<b>Propósito .....</b>	<b>81</b>	
<b>Objetivo.....</b>	<b>81</b>	
<b>Herramientas .....</b>	<b>81</b>	
<b>Entregables .....</b>	<b>81</b>	
<b>Software .....</b>	<b>81</b>	
<b>Hardware .....</b>	<b>81</b>	
<b>Características a ser probadas .....</b>	<b>82</b>	
<b>Características a no ser probadas.....</b>	<b>82</b>	
<b>Casos de prueba.....</b>	<b>82</b>	
<b>Criterios para el lanzamiento.....</b>	<b>86</b>	
<b>Resultados .....</b>	<b>86</b>	
<b>    Conclusión.....</b>	<b>95</b>	
<b>    Referencias.....</b>	<b>96</b>	
<b>    Anexos .....</b>	<b>97</b>	
<b>Cronograma.....</b>	<b>97</b>	
<b>Entregables .....</b>	<b>99</b>	
<b>Poster.....</b>	<b>99</b>	
<b>Tríptico.....</b>	<b>100</b>	
<b>Manual de usuario.....</b>	<b>102</b>	
1. Registro .....	4 .....	103
2. Iniciar sesión .....	4 .....	103
3. Operaciones Básicas .....	4 .....	103
3.1. Tienda .....	4 .....	103
3.2. Administración .....	5 .....	103
3.3. Chat .....	5 .....	103
1. OBJETIVO DE LA APLICACIÓN.....		104
Módulo de Registro.....		105
Módulo de Iniciar sesión.....		105
Módulo de Tienda.....		105

Módulo de Administración.....	105
Módulo de Chat .....	105
Modulo 1: Registro.....	105
Modulo 2: Inicio de Sesión .....	106
Modulo 3: Tienda .....	106
Modulo 4: Administración.....	110
Modulo 5: Chat.....	113

## Índice de figuras

Imagen 1. Ciclo de vida de la metodología cascada .....	12
Imagen 2. Esquema de la arquitectura horizontal.....	13
Imagen 3. Diagrama de casos de uso .....	18
Imagen 4. Caso de uso. Autentificar administrador.....	18
Imagen 5. Caso de uso. Dar de alta producto .....	19
Imagen 6. Caso de uso. Registro de usuario .....	22
Imagen 7. Caso de uso. Obtener estadísticas .....	24
Imagen 8. Diagrama de Clases. ....	25
Imagen 9. Diagrama de clases parte 1 .....	26
Imagen 10. Diagrama de clases parte 2 .....	27
Imagen 11. Diagrama de clases parte 3. ....	27
Imagen 12. Diagrama de secuencia iniciar sesión administrador.....	28
Imagen 13. Diagrama de secuencia iniciar sesión cliente.....	29
Imagen 14. Diagrama de secuencia registrar cliente. ....	29
Imagen 15. Diagrama de secuencia realizar envío.....	30
Imagen 16. Diagrama de secuencia ver estadísticas.....	30
Imagen 17. Diagrama de secuencia alta productos .....	31
Imagen 18. Diagrama de estados. ....	31
Imagen 19. Diagrama de actividades .....	32
Imagen 20. Registro administrador.....	33
Imagen 21. Inicio de sesión del administrador .....	34
Imagen 22. Cierre de sesión de administrador.....	34
Imagen 23. Registro de cliente.....	35
Imagen 24. Inicio de sesión del cliente .....	36
Imagen 25. API carrito primera parte .....	37
Imagen 26. API carrito segunda parte.....	38
Imagen 27. Parte principal de la API de categorías.....	39
Imagen 28. Parte esencial de la API de ordenes .....	40
Imagen 29. Primera parte del API de productos.....	41
Imagen 30 Segunda parte de la API del producto.....	42
Imagen 31.API de las estadísticas del producto.....	43
Imagen 32. Conexión a la BD.....	44
Imagen 33. Sección de productos primera parte.....	45
Imagen 34. Sección de productos segunda parte .....	46
Imagen 35. Sección de productos tercera parte .....	47
Imagen 36. Sección de productos cuarta parte .....	48
Imagen 37. Sección de administración de ordenes primera parte.....	49
Imagen 38. Sección de administración de ordenes segunda parte. ....	50
Imagen 39. Sección de administración de ordenes tercera parte. ....	51
Imagen 40. Componente de inicio de sesión de administrador .....	51
Imagen 41. Componente de registro de administrador .....	52

Imagen 42. Componente de Chat .....	53
Imagen 43. Componente de administración de negocio.....	54
Imagen 44. Parte principal de la tienda primera parte .....	55
Imagen 45. Parte principal de la tienda segunda parte .....	56
Imagen 46. Parte principal de la tienda tercera parte .....	57
Imagen 47. Componente de visualización de productos.....	58
Imagen 48. Componente de detalles del producto. ....	59
Imagen 49. Componente del carrito. ....	60
Imagen 50. Página de checkout primera parte .....	61
Imagen 51. Página de checkout segunda parte .....	62
Imagen 52. Página de checkout tercera parte .....	63
Imagen 53. Página de checkout cuarta parte.....	64
Imagen 54. Página de checkout quinta parte .....	65
Imagen 55. Página de checkout sexta parte .....	66
Imagen 56. Componente de las ordenes del cliente.....	67
Imagen 57. Pantalla de inicio de sesión del administrador .....	68
Imagen 58. Pantalla de registro del administrador.....	68
Imagen 59. Apartado de categorías .....	69
Imagen 60. Modal para agregar categoría.....	69
Imagen 61. Modal para eliminar categorías. ....	69
Imagen 62. Modal para editar categoría.....	70
Imagen 63. Apartado de productos. ....	70
Imagen 64. Modal para añadir nuevo producto. ....	71
Imagen 65. Modal con la información del producto.....	71
Imagen 66. Apartado de órdenes.....	72
Imagen 67. Pantalla principal del cliente no registrado.....	72
Imagen 68. Modal del inicio de sesión del cliente .....	73
Imagen 69. Modal para el registro del usuario .....	73
Imagen 70. Administración del negocio del cliente .....	74
Imagen 71. Apartado de chat.....	74
Imagen 72. Sobre Dulcemprende.....	75
Imagen 73. Sobre los creadores .....	75
Imagen 74. Parte principal de la tienda primera parte. ....	76
Imagen 75. Parte principal de la tienda segunda parte. ....	76
Imagen 76. Parte principal de la tienda tercera parte. ....	77
Imagen 77. visualización de los productos.....	77
Imagen 78. Información del producto. ....	78
Imagen 79. Apartado del carrito. ....	78
Imagen 80. Parte del pago .....	79
Imagen 81. Página con las órdenes del cliente .....	79
Imagen 82. Detalles de orden .....	80
Imagen 83. CP01 No. 1 exitoso .....	86
Imagen 84. CP01 No. 2 exitoso .....	87
Imagen 85. CP02 No. 1 exitoso .....	87

Imagen 86. CP02 No. 2 exitoso .....	88
Imagen 87. CP03 No.1 exitoso .....	88
Imagen 88. CP03 No. 2 exitoso .....	89
Imagen 89. CP04 No.1 exitoso .....	89
Imagen 90. CP05 exitoso .....	89
Imagen 91. CP06 exitoso .....	90
Imagen 92. CP07 exitoso .....	90
Imagen 93. CP08 No.1 exitoso .....	91
Imagen 94. CP08 No.2 exitoso .....	91
Imagen 95. CP09 No.1 exitoso .....	92
Imagen 96. CP09 No.2 exitoso .....	92
Imagen 97. CP010 exitoso .....	93
Imagen 98. CP011 No.1 exitoso .....	93
Imagen 99. CP011 No.2 exitoso .....	94
Imagen 100. CP012 exitoso .....	94
Imagen 101. Poster dulcemprende.....	99
Imagen 102. Tríptico Dulcemprende.....	100
Imagen 103. Tríptico Dulcemprende.....	101
Imagen 104. Pantalla de Inicio .....	105
Imagen 105, Pantalla de Registro.....	106
Imagen 106, Entrada de datos en el inicio de sesión.....	106
Imagen 107. Barra Lateral .....	106
Imagen 108. Tienda Inicio .....	107
Imagen 109. Artículos en la Tienda .....	107
Imagen 110. Vista del producto .....	108
Imagen 111. Vista de pago .....	108
Imagen 112. Métodos de Pago .....	109
Imagen 113, Pantalla de seguimiento.....	109
Imagen 114. Menú de usuario .....	110
Imagen 115. Ordenes del Usuario.....	110
Imagen 116. Inicio de Sesión.....	110
Imagen 117. Registro de Administradores.....	111
Imagen 118. Menús de Administrador .....	111
Imagen 119. Agregar categoría .....	111
Imagen 120. Eliminar categoría.....	112
Imagen 121. Actualizar categoría.....	112
Imagen 122. Lista de productos .....	112
Imagen 123. Formulario Productos.....	113
Imagen 124. Ventana de Chat.....	113

## **Introducción**

Día a día el internet toma control de más y más sectores. Este es el caso del comercio por internet, donde las empresas están optando por tiendas en línea y cada vez aumenta la cantidad de personas que hace uso de estas.

De acuerdo con el reporte sobre el impacto del Covid-19 en venta online, elaborado por la Asociación Mexicana de Venta Online (AMVO), 5 de cada 10 empresas en México están duplicando su crecimiento en internet, y de cada 10 registran incrementos del 300% en el volumen de negocios de ventas online.

De igual forma, cada día aumentan las personas que desean emprender y tener una fuente de ingresos extra o dedicarse de lleno a eso, un problema que presentan estas personas es en la organización de sus ventas e inversiones, además de la búsqueda de los lugares donde es más apto conseguir los productos que ellos pasan a vender después.

Es por eso que un sistema que aparte de tienda en línea, funcione como un panel de administración para aquellas personas que están emprendiendo o deseen a emprender es muy viable hoy en día.

## Estado del arte

La mayoría de los sistemas como el que se desea implementar que están en la web, solo funcionan como venta al por mayor de dulces, estos sitios no ofrecen una interfaz o panel de control para el emprendedor que le permita gestionar su negocio, además de arrojar recomendaciones. He aquí algunos sistemas:

Página	Ventajas	Desventajas	Plataforma
HS Comercial	-Sistema de envío -Precios competitivos -Promociones -Variedad de dulces	-Mal diseño -No ofrece un administrador de negocio	WEB
Azúcar dulcerías	-Sistema de envío nacional -Buenos precios -Variedad de dulces -Buenas ofertas	-No ofrece un administrador de negocio -Precios normales	WEB
CandyMaria	-Seguridad en los envíos -Diseño llamativo -Buen menú de precios	-Precios no tan buenos -No ofrece un administrador de negocio	WEB

## **Justificación**

A mediados de marzo de 2020 comenzó el confinamiento debido al COVID-19, lo anterior orilló a que negocios esenciales, como lo son los del sector salud y los que cumplen con las necesidades básicas de la población, permanecieran abiertos y los que no se consideran esenciales permanecieran cerrados. Estos últimos negocios se vieron muy afectados en cuestiones operativas y económicas. Según datos del INEGI, antes de la pandemia existían 4.9 millones de micro, pequeñas y medianas empresas (MiPymes), unos meses después de la crisis sanitaria, se registró el cierre de 1 millón de MiPymes. Esta crisis operativa y económica de los negocios se da, en gran medida, por el cierre de actividades dentro de ellos, lo que provoca que no tengan ventas y por lo tanto se queden sin ingresos.

Estos hechos conducen a un nuevo entendimiento del concepto de ventas y el mundo digital, el E-commerce en México creció aceleradamente, según el International Data Corp (IDC) el comercio digital creció en 2020, aproximadamente, un 60% que traducido en dinero serían 860,000 millones de pesos mexicanos. Lo anterior nos habla de la importancia de que los negocios se empiecen a preocupar por su digitalización.

El presente proyecto busca incentivar a las personas y estudiantes a comenzar emprendimientos en un sector popular y factible como lo son los dulces. Esta plataforma disminuirá el impacto económico que han tenido algunas personas, pues los precios serán accesibles.

La complejidad de este trabajo consta del diseño de la plataforma, además de la programación de los servicios que ofrecerá y el análisis de mercado que se realizará sobre los dulces.

## **Objetivo general**

Desarrollar un sistema de dulcería en línea para aquellas personas que deseen realizar un emprendimiento que funcione como tienda y panel de control para que los usuarios administren su negocio.

## **Objetivos particulares**

- Identificar las necesidades que tienen las personas de emprender
- Reconocer la importancia de un sistema en línea
- Analizar el mercado para determinar productos populares y con el mejor precio

# Capítulo 1

## Marco Teórico

Una dulcería es un establecimiento el cual distribuye dulces a sus clientes. Por otro lado tenemos las tiendas en línea, las cuales facilitan la compra/venta, a través del uso de una aplicación web.

Las aplicaciones web reciben este nombre porque se ejecutan en internet. Es decir que los datos o los archivos en los que trabajas son procesados y almacenados dentro de la web. Estas aplicaciones, por lo general, no necesitan ser instaladas en tu computador.

El concepto de aplicaciones web está relacionado con el almacenamiento en la nube. Toda la información se guarda de forma permanente en grandes servidores de internet y nos envían a nuestros dispositivos o equipos los datos que requerimos en ese momento, quedando una copia temporal dentro de nuestro equipo.

En cualquier momento, lugar y desde cualquier dispositivo podemos acceder a este servicio, sólo necesitamos una conexión a internet y nuestros datos de acceso, que por lo general son el nombre de usuario y contraseña.

En este caso, para facilitar el diseño y la implementación del proyecto se usó JavaScript, específicamente con la librería React.

React es una librería focalizada en el desarrollo de interfaces de usuario. Sin embargo, lo cierto es que en React encontramos un excelente aliado para hacer todo tipo de aplicaciones web, SPA (Single Page Application) o incluso aplicaciones para móviles. Para ello, alrededor de React existe un completo ecosistema de módulos, herramientas y componentes capaces de ayudar al desarrollador a cubrir objetivos avanzados con relativamente poco esfuerzo.

La creación de esta plataforma facilitará a los usuarios, más ahora en tiempos de pandemia, la compra y venta de dulces. En este caso como nos enfocamos a la distribución a empresas o tiendas de conveniencia.

Por otro lado, muchas personas piensan en crear su propio negocio por diversas razones, pero la principal de ellas es lograr su independencia económica y tener autonomía en la toma de decisiones de un negocio.

México es conocido por sus altos índices de emprendimiento y creación de pequeñas y medianas empresas (PyMEs). La Universidad Anáhuac informó que se abren cerca de 35 mil negocios mensualmente en el país. Con estos datos, cualquiera podría pensar que el país tiene un ambiente ideal para los emprendedores, sin embargo, este aspecto tiene diferentes puntos de vista.

El 75% de los negocios que hoy se abren no llegarán a su segundo año de vida, 2 de cada 3 emprendimientos en México no alcanzarán los cinco años de operaciones y solamente el 10% logrará sobrevivir más de una década. De hecho, de los emprendimientos que hoy se están haciendo solamente el 0.7% de ellos crecerá, según datos de la Bolsa Institucional de Valores (BIVA), recopilados por la revista Forbes.

A decir de la directora de esta institución, la razón detrás de estos datos está en la falta de acceso a tecnología y financiamiento para estas empresas, sin embargo la Universidad de Estudios Avanzados (UNEA) considera que actualmente existen los programas y apoyos suficientes para los emprendedores y que en realidad los retos radican en otros aspectos.

Ahora bien, si hablamos específicamente de dulcerías, tenemos que a nivel global, la industria de la confitería tiene un valor de mercado superior a US\$185,477 millones, siendo los habitantes de países industrializados quienes generan el mayor consumo.

México es reconocido como uno de los 10 países con más ventas de la industria confitera mundial. En 2011 obtuvo el tercer lugar en el continente americano, al sumar ingresos por US\$4,651 millones, sólo después de Estados Unidos y Brasil.

ProMéxico señala que ese año el 43% de las ventas fueron generadas por la categoría de dulces –que agrupa productos como mazapanes, caramelo suave y macizo, paletas, tamarindos y dulces típicos–, seguidas por la división de chicles, con el 33%, y la de chocolates, con el 24 por ciento.

De acuerdo con la Asociación Nacional de Tiendas de Autoservicio y Departamentales de México (Antad), el principal cliente potencial, por su margen de consumo, es un hombre o mujer menor de 20 años que se inclina por paletas y caramelos. También existe otro consumidor, mayor de 20, que busca chocolates y productos para refrescar el aliento.

A la par, está surgiendo una nueva categoría marcada por una fuerte influencia a lo saludable y light. De ahí que no debes perder de vista los productos bajos en calorías, endulzados con edulcorantes o reducidos en sal. En todo caso, considera que el cliente compra estos productos por impulso, motivado por la conveniencia, por el precio, el fácil acceso y el sabor.

Este es otro de los motivos por los cuales decidimos crear nuestra app, ya que con ella pretendemos acercarnos a clientes potenciales de una forma rápida. El contar con estas herramientas facilitará la búsqueda de emprendedores que decidan crear su propio negocio y de esta forma, les podemos apoyar con las diferentes herramientas que nuestro proyecto tiene para ofrecer.

## Capítulo 2

### Análisis

#### Metodología

La metodología para usar será la clásica waterfall o de cascada.

Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.

En la imagen 1 se puede ver el ciclo de vida de esta metodología:

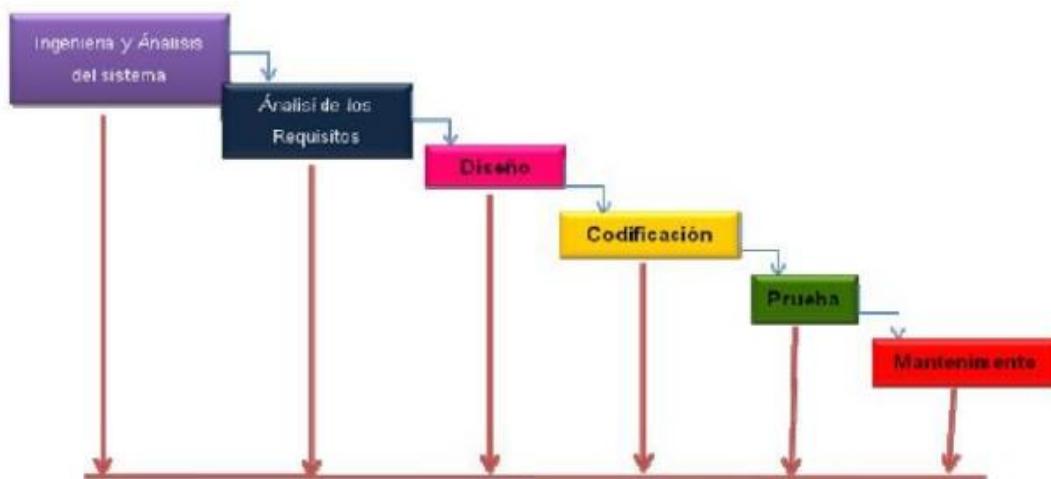


Imagen 1. Ciclo de vida de la metodología cascada.

**Ingeniería y Análisis del Sistema:** debido que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.

**Análisis de los requisitos del software:** el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software (Analistas) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.

**Diseño:** el diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedural y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

**Codificación:** el diseño debe traducirse en una forma legible para la máquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.

**Prueba:** una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

**Mantenimiento:** el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a que se han encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

## Técnica de Recolección de Requisitos

Debido a la situación actual en la que nos encontramos la técnica más apropiada para la recolección de requisitos será la aplicación de cuestionarios.

## Arquitectura

Al ser una plataforma web, es importante incorporar una arquitectura web organizada y coherente que facilite la indexación y el rastreo de nuestra web en los buscadores, además de que ayudará a encontrar a los usuarios aquello que buscan de manera sencilla.

Es por ello por lo que se optó por una arquitectura web horizontal, en este tipo de arquitectura se tienen pocos niveles de profundidad, es decir, los usuarios deben hacer un menor número de clics para encontrar cualquier apartado de nuestra plataforma. En la imagen 2 podemos ver el esquema de esta arquitectura.

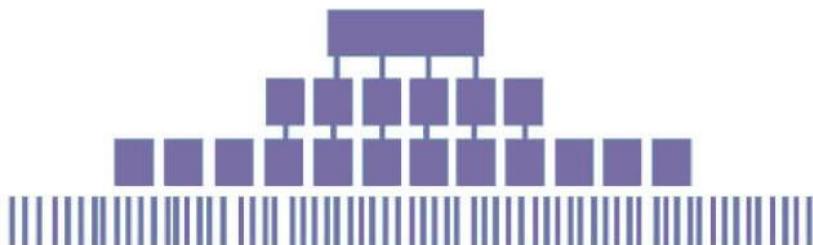


Imagen 2. Esquema de la arquitectura horizontal.

## Requerimientos del sistema

La información utilizada para el análisis de los requerimientos de nuestro sistema fue recopilada con la técnica de revisión de registros.

Esta revisión fue hecha en distintas tiendas en línea de dulces que existen, recabando información sobre lo que ofrecen y lo que no y verificar que nuestro sistema no se ha hecho antes y es algo nuevo.

## Reglas de negocio

- Todo cliente tiene autorizado realizar cualquier pedido solo si posee una cuenta en la página web del negocio.
- Ofrecer productos con descuento a los clientes frecuentes o con pedido mayor a \$1,000 pesos.
- El lapso de pago deberá ser estrictamente no mayor a 48 horas.
- Los envíos se realizan únicamente en zonas contenidas en el área metropolitana.
- Solo las personas con cuenta pueden tener acceso al chat
- Solo las personas con cuenta pueden tener accesos a su panel de administración

## Requerimientos funcionales

ID	Nombre	Descripción
RF1	Registrar usuario	Introducir usuario, contraseña, nombre y dirección del usuario
RF2	Autenticar usuario	Introducir usuario y contraseña correctos para entrar a la cuenta del sistema
RF3	Dar de alta a usuario	Almacenar los datos del usuario en una base de datos
RF4	Modificar datos de usuario	Modificar los datos previamente introducidos por el usuario dados de alta en la base de datos; nombre, correo, contraseña, dirección, forma de pago.
RF5	Dar de alta productos	Almacenar los dulces en una base de datos
RF6	Modificar productos	Modificar el stock o descripción de los dulces almacenados en la base de datos
RF7	Generar factura	Generar una factura al momento de que un usuario compre un producto
RF8	Consultar productos	Buscar por marca, categoría >o temporada

RF9	Obtener estadísticas	El usuario podrá ver sus estadísticas como sus ventas, compras, ganancias, etc.
RF10	Autentificar administrador	Introducir usuario y contraseña correctos para entrar al sistema
RF11	Ingresar datos de pago	El usuario ingresará sus datos de pago, puede ser por medio de paypal o transferencia electrónica
RF12	Verificar pago	Se verificará el pago que el usuario realizó para así poder proceder al envío del producto comprado
RF13	Realizar envío	Se contratará una paquetería para realizar los envíos de los productos
RF14	Realizar compra	El usuario realiza la compra de los productos deseados.

## Requerimientos no funcionales

ID	Nombre	Descripción
RNF1	Restricción	Solo el administrador podrá modificar los productos o hacerle cambios a la página
RNF2	Disponibilidad	El sistema estará disponible
RNF3	Confidencialidad	Las contraseñas, direcciones y forma de pago de los usuarios deben ser confidenciales, la contraseña deberá ser encriptada. El sistema deberá mantener la integridad de los datos personales de los usuarios registrados.
RNF4	Usabilidad	El sistema será fácil de usar, se implementarán interfaces sencillas y de fácil entendimiento para todo usuario
RNF5	Mantenibilidad	El sistema deberá tener un mantenimiento en caso de tener errores en cualquier aspecto
RNF6	Interfaz	Se dispondrá de una interfaz interactiva
RNF7	Seguridad	Todas las comunicaciones externas entre los servidores de datos, la aplicación y el cliente del sistema deben estar cifradas utilizando un algoritmo de encriptación
RNF8	Portabilidad	El sistema diseñado será compatible con todos los dispositivos pues será desplegado en web
RNF9	Desempeño	El sistema tendrá una respuesta rápida a las peticiones del usuario

## **Requerimientos técnicos**

El desarrolló de la plataforma debe apegarse a las siguientes especificaciones técnicas:

- Uso de HTML, HTML5, PHP, Javascript u otro lenguaje adecuado a consideración del programador
- El código fuente que se vaya generando se deberá cargar a un repositorio para mantener un control de versiones de la plataforma
- Contar con un equipo que trabaje en el proyecto'

## **Actores**

Administrador: Persona que interactúa con los productos de la página.

Clientes: Las personas que actúan como usuarios del sistema que comprarán y podrán tener su panel de control.

Repartidor: Persona que interactúa con el envío de los productos.

ACTIVIDAD	MAR	ABR	MAY	JUN
Análisis de requisitos				
Definición de objetivos generales y particulares				
Planeación de actividades del negocio				
Elaboración del diagrama de contexto				
Elaboración del diagrama de flujo de datos lógico				
Elaboración del diagrama de clases				
Elaboración del diagrama de clasificación y ensamble				
Planeación y diseño de página web				
Creación de página web				
Realización de pruebas (Hacer pedido)				
Entrega proyecto final				

## Capítulo 3

### Diseño del sistema

#### Casos de uso

A continuación, el diagrama de casos de uso de nuestro sistema Dulcemprende, comprendido por los autores: administrador, cliente y repartidor.

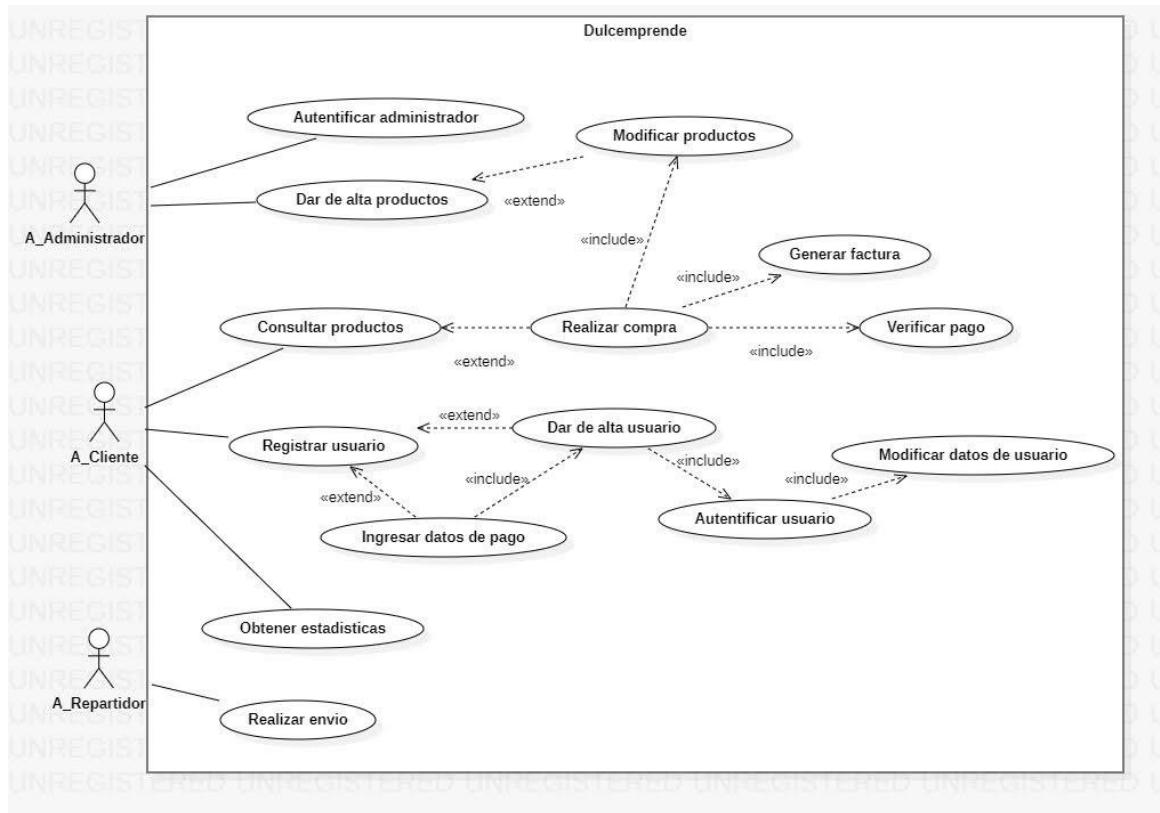


Imagen 3. Diagrama de casos de uso

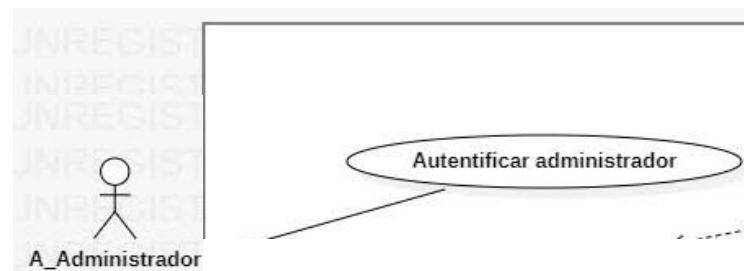


Imagen 4. Caso de uso. Autenticar administrador.

<b>Caso de Uso:</b>	CU1 Autenticar administrador
<b>Versión:</b>	1.0
<b>Actor(es):</b>	Administrador
<b>Propósito:</b>	Poder realizar acciones de administrador
<b>Resumen:</b>	Ver la interfaz y acciones del administrador
<b>Entradas:</b>	La aplicación por separado del administrador
<b>Salidas:</b>	Ver opciones de inicio de sesión y registro
<b>Precondiciones:</b>	Entrar a la aplicación del administrador
<b>Postcondiciones</b>	Iniciar sesión
<b>Autor:</b>	Brandon Meza
<b>Tipo:</b>	Primario
<b>Módulo:</b>	Iniciar sesión

Flujo de eventos:

#### Trayectoria Principal

1. El administrador entra a la aplicación del administrador
2. El sistema muestra el formulario para el inicio de sesión
3. El administrador llena sus datos
4. El sistema valida los datos [Trayectoria A]
5. El sistema da acceso a las acciones del administrador

-Fin de trayectoria

#### Trayectoria A (datos incorrectos)

6. El sistema muestra el mensaje “Contraseña incorrecta”
7. El sistema continua al paso 3 de la trayectoria principal

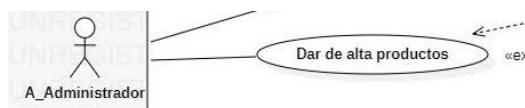


Imagen 5. Caso de uso. Dar de alta producto

<b>Caso de Uso:</b>	CU2 Dar de alta productos
<b>Versión:</b>	1.0
<b>Actor(es):</b>	Administrador
<b>Propósito:</b>	Poder añadir productos
<b>Resumen:</b>	Agregar productos para la visualización del usuario
<b>Entradas:</b>	Seleccionar opción añadir productos
<b>Salidas:</b>	Se ve el nuevo producto en una tabla
<b>Precondiciones:</b>	Iniciar sesión como administrador
<b>Postcondiciones</b>	Iniciar sesión
<b>Autor:</b>	Brandon Meza
<b>Tipo:</b>	Primario
<b>Módulo:</b>	Añadir producto

Flujo de eventos:

#### Trayectoria Principal

1. El administrador entra a la aplicación del administrador
2. El administrador va a la pestaña de agregar producto.
3. El sistema muestra el formulario para añadir el producto
4. El sistema valida los datos [Trayectoria A]
5. El sistema muestra el producto añadido en una tabla

-Fin de trayectoria

#### Trayectoria A (datos incompletos)

6. El sistema muestra el mensaje del dato faltante del producto
7. El sistema continua al paso 3 de la trayectoria principal

<b>Caso de Uso:</b>	CU3 Modificar productos
<b>Versión:</b>	1.0
<b>Actor(es):</b>	Administrador

<b>Propósito:</b>	Poder modificar productos
<b>Resumen:</b>	Modificar productos para la visualización del usuario
<b>Entradas:</b>	Seleccionar opción modificar productos
<b>Salidas:</b>	Se ve el producto modificado en una tabla
<b>Precondiciones:</b>	Iniciar sesión como administrador
<b>Postcondiciones</b>	Iniciar sesión
<b>Autor:</b>	Daniela Briones
<b>Tipo:</b>	Primario
<b>Módulo:</b>	Modificar producto

Flujo de eventos:

#### Trayectoria Principal

1. El administrador entra a la aplicación del administrador
2. El administrador va a la pestaña de agregar producto.
3. El administrador escoge la opción de modificar producto
4. El sistema muestra el formulario para modificar producto
5. El sistema valida los nuevos datos [Trayectoria A]
6. Se muestra el producto en una tabla con sus modificaciones

-Fin de trayectoria

#### Trayectoria A (datos incompletos)

7. El sistema muestra el mensaje del dato faltante del producto
8. El sistema continua al paso 3 de la trayectoria principal

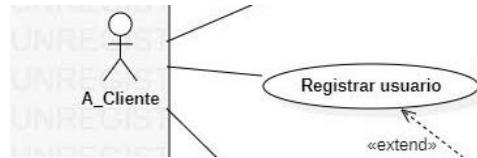


Imagen 6. Caso de uso. Registro de usuario

<b>Caso de Uso:</b>	CU4 Registro del cliente
<b>Versión:</b>	1.0
<b>Actor(es):</b>	Cliente
<b>Propósito:</b>	Poder realizar acciones como cliente
<b>Resumen:</b>	Inicio de sesión como cliente para realizar acciones en el sistema
<b>Entradas:</b>	Entrar a la aplicación del cliente
<b>Salidas:</b>	Cliente autentificado
<b>Precondiciones:</b>	Entrar al sistema
<b>Postcondiciones</b>	Registro
<b>Autor:</b>	Diego Torres
<b>Tipo:</b>	Primario
<b>Módulo:</b>	Registro de cliente

Flujo de eventos:

Trayectoria Principal

1. El cliente entra a la aplicación del cliente
2. El cliente va al menú de registrarse
3. El sistema muestra una ventana con el formulario de registro
4. El usuario ingresa sus datos
5. El usuario ingresa un correo [Trayectoria A]
6. El usuario ingresa una contraseña [Trayectoria B]

-Fin de trayectoria

Trayectoria A (correo no valida)

7. El sistema muestra el mensaje “Correo no valido”
8. El sistema continua al paso 3 de la trayectoria principal

Trayectoria B (contraseña no valida)

1. El sistema muestra el mensaje “La contraseña debe tener 6 o más caracteres”
2. El sistema continua al paso 3 de la trayectoria principal

<b>Caso de Uso:</b>	CU5 Inicio de sesión del cliente
<b>Versión:</b>	1.0
<b>Actor(es):</b>	Cliente
<b>Propósito:</b>	Poder realizar acciones como cliente
<b>Resumen:</b>	Inicio de sesión como cliente para realizar acciones en el sistema
<b>Entradas:</b>	Entrar a la aplicación del cliente
<b>Salidas:</b>	Cliente autenticado
<b>Precondiciones:</b>	Entrar al sistema
<b>Postcondiciones</b>	Iniciar sesión
<b>Autor:</b>	Diego Torres
<b>Tipo:</b>	Primario
<b>Módulo:</b>	Inicio de sesión de cliente

Flujo de eventos:

#### Trayectoria Principal

1. El cliente entra al sistema
2. El cliente va al apartado de iniciar sesión
3. El sistema muestra el formulario de inicio de sesión
4. El cliente ingresa sus datos [Trayectoria A]
5. El sistema valida los datos
6. Se da acceso al sistema

-Fin de trayectoria

#### Trayectoria A (datos incorrectos)

7. El sistema no muestra respuesta
8. El sistema continua al paso 3 de la trayectoria principal

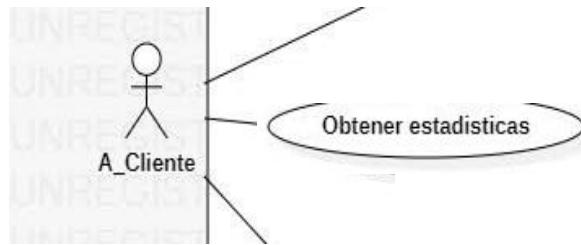


Imagen 7. Caso de uso. Obtener estadísticas

<b>Caso de Uso:</b>	CU6 Ver estadísticas
<b>Versión:</b>	1.0
<b>Actor(es):</b>	Cliente
<b>Propósito:</b>	Poder ver estadísticas del usuario
<b>Resumen:</b>	Ver estadísticas del cliente
<b>Entradas:</b>	Seleccionar opción panel de administración
<b>Salidas:</b>	Se ven las estadísticas del cliente
<b>Precondiciones:</b>	Iniciar sesión como cliente
<b>Postcondiciones</b>	Ver estadísticas
<b>Autor:</b>	Brandon Meza
<b>Tipo:</b>	Primario
<b>Módulo:</b>	Ver estadísticas

Flujo de eventos:

Trayectoria Principal

1. El cliente entra al sistema
2. El cliente inicia sesión o se registra
3. El cliente entra al apartado de administración
4. El sistema arroja las estadísticas del usuario
5. El usuario visualiza sus estadísticas

-Fin de trayectoria

## Diagramas de clase

El diagrama de clases de nuestro sistema se muestra a continuación en la imagen siguiente.

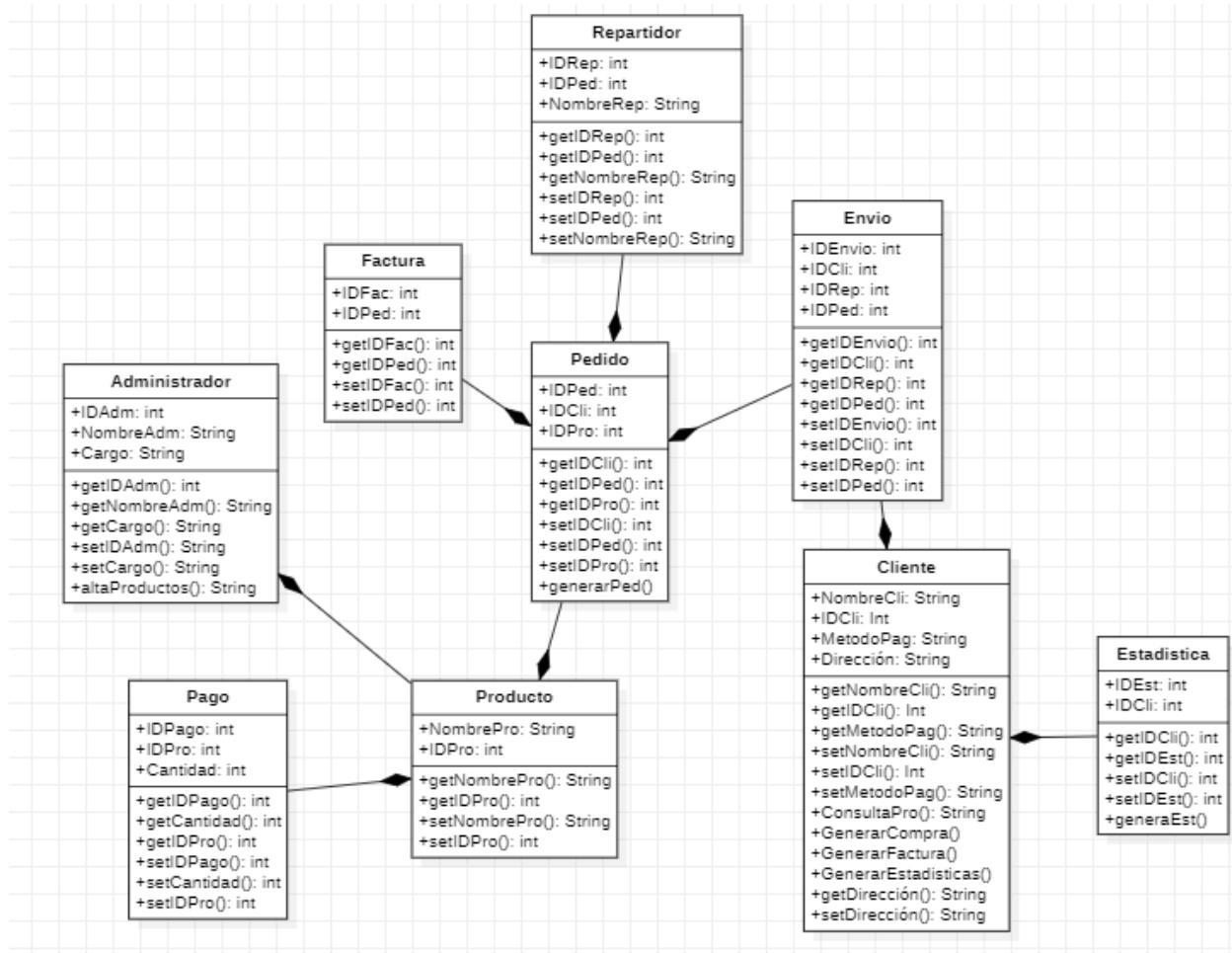


Imagen 8. Diagrama de Clases.

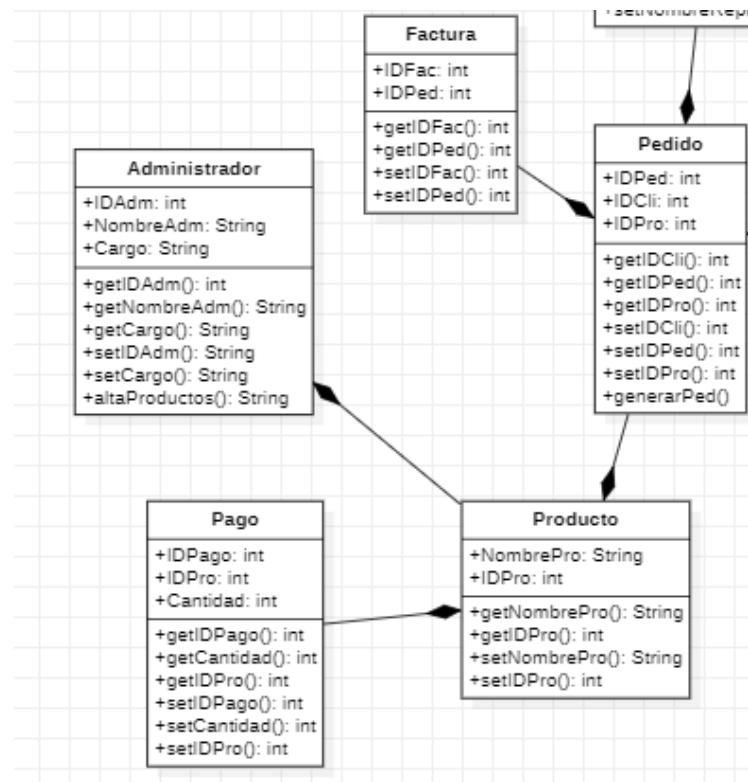


Imagen 9. Diagrama de clases parte 1

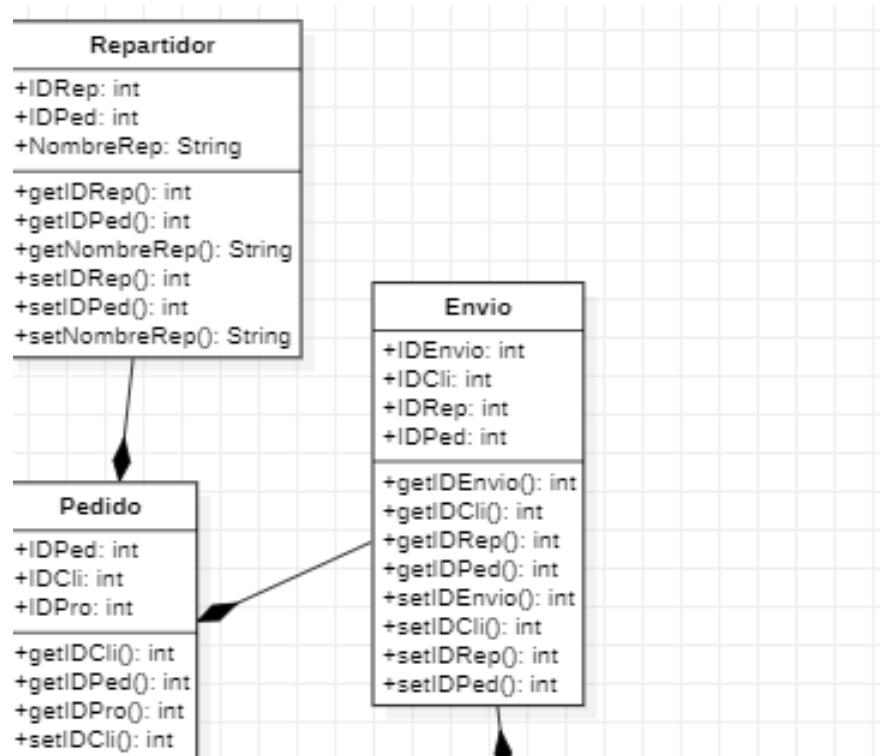


Imagen 10. Diagrama de clases parte 2

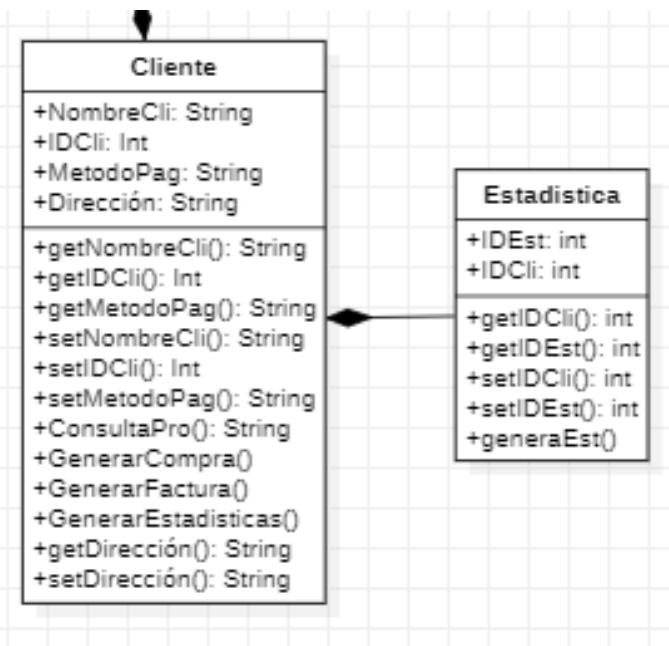


Imagen 11. Diagrama de clases parte 3.

## Diagramas de secuencia

A continuación los diagramas de secuencia de nuestro sistema Dulcemprende con los cuales se va completando cada vez más la parte del diseño de nuestro sistema.

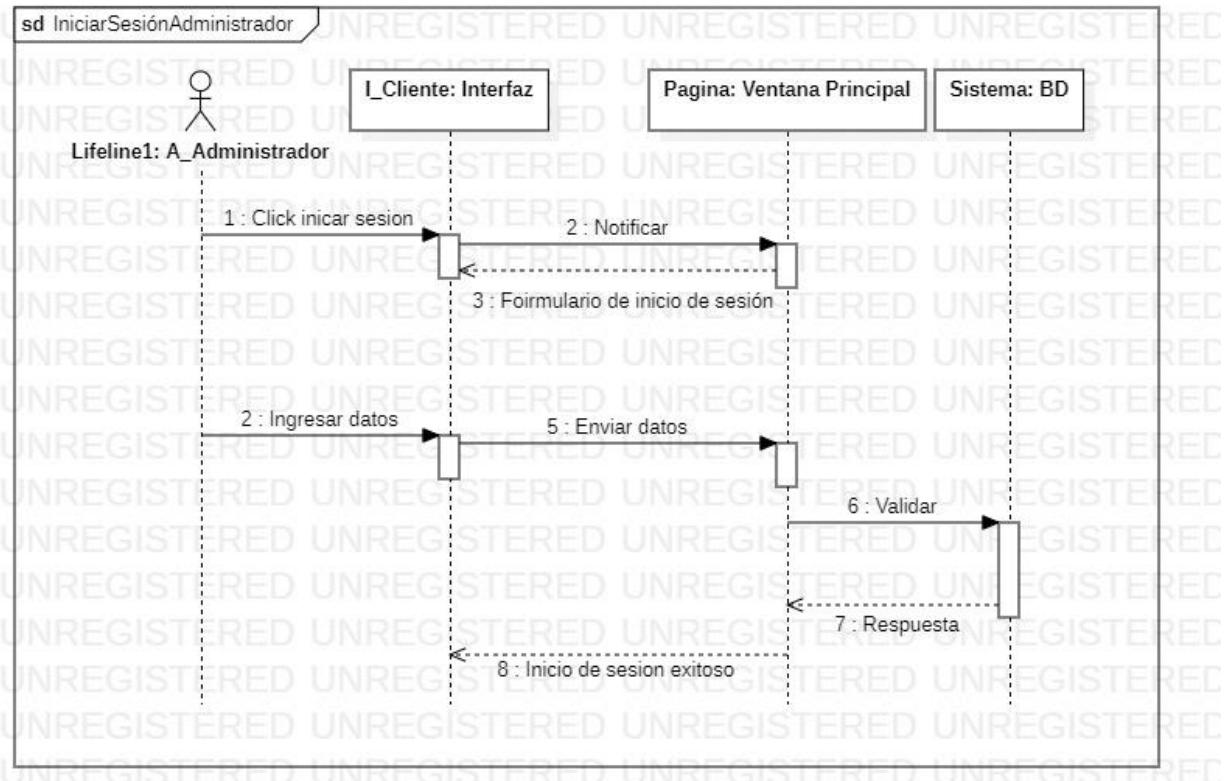


Imagen 12. Diagrama de secuencia iniciar sesión administrador.

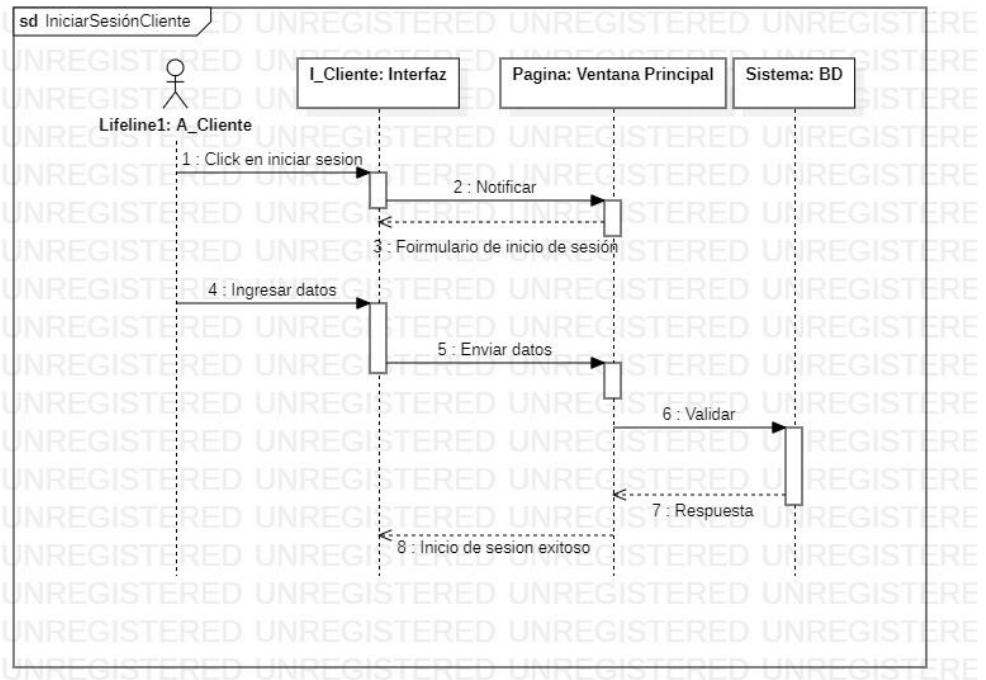


Imagen 13. Diagrama de secuencia iniciar sesión cliente.

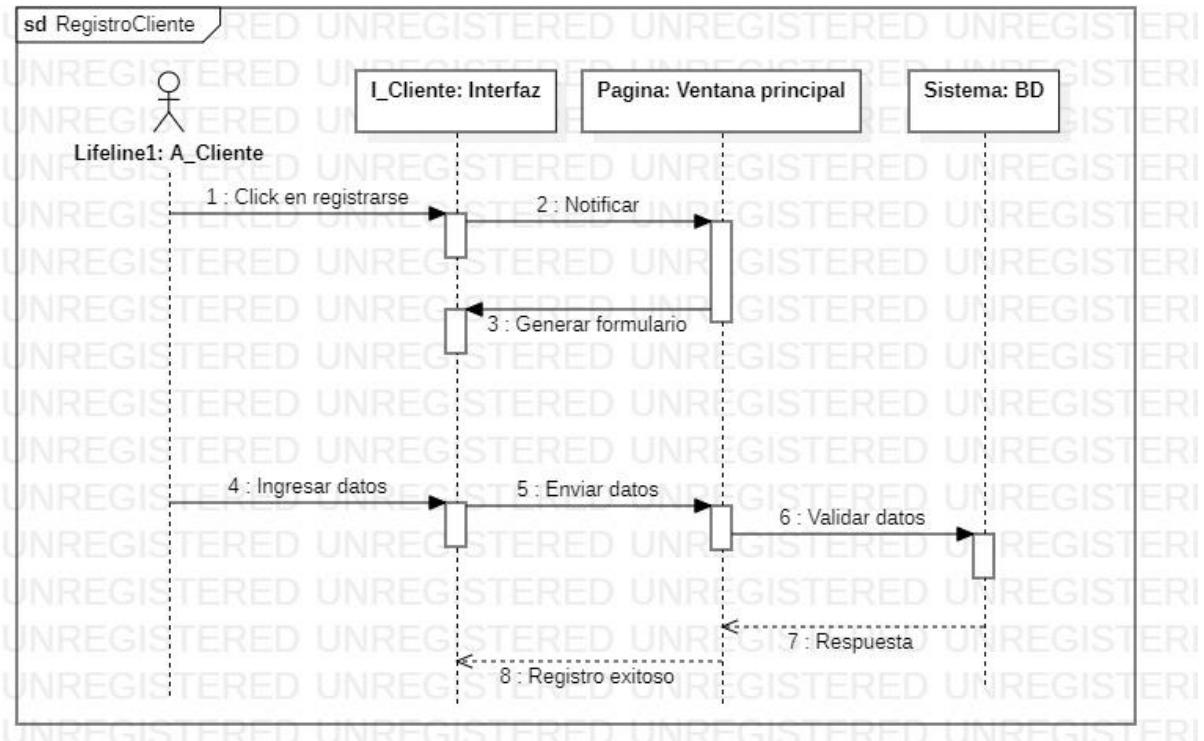


Imagen 14. Diagrama de secuencia registrar cliente.

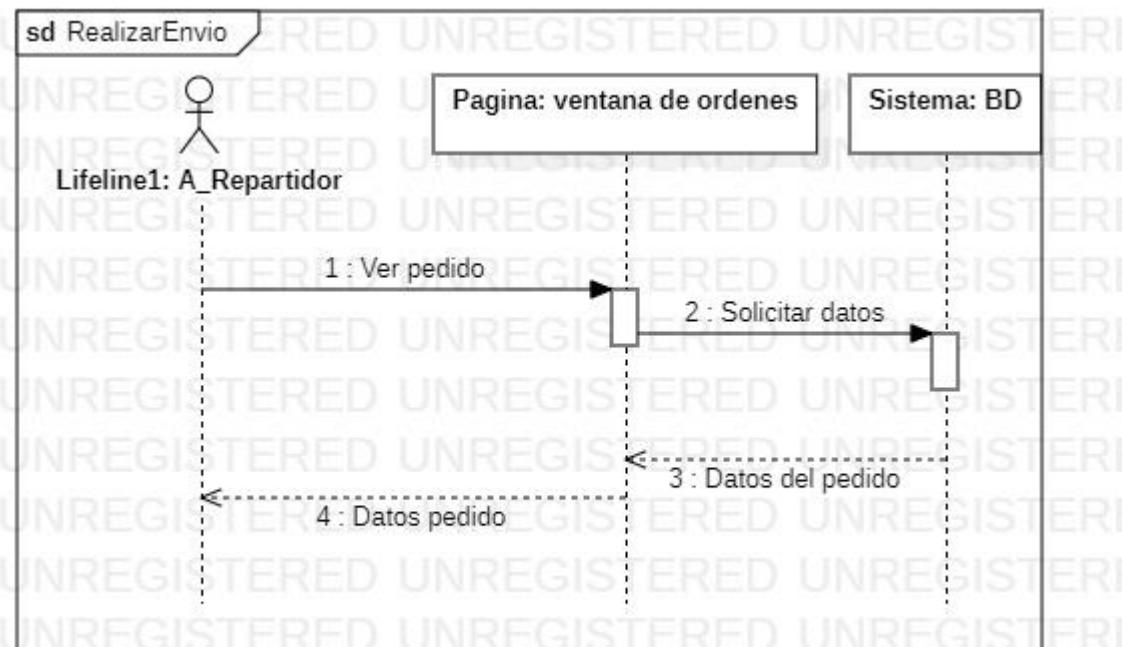


Imagen 15. Diagrama de secuencia realizar envío.

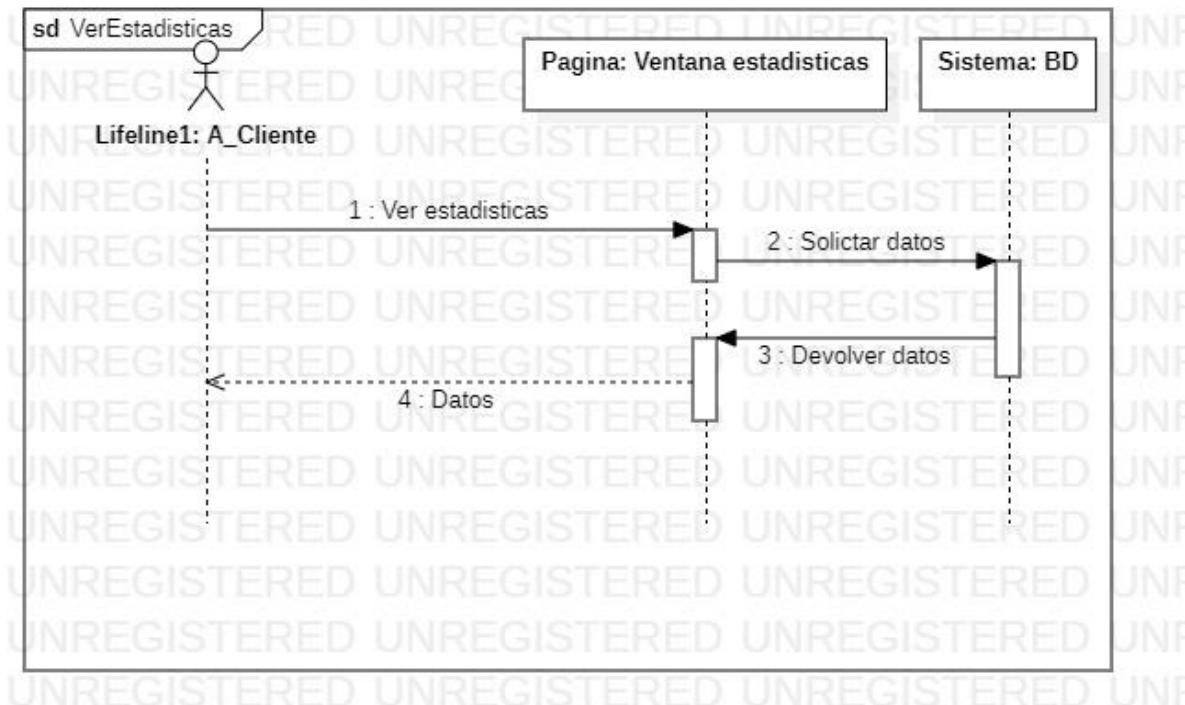


Imagen 16. Diagrama de secuencia ver estadísticas.

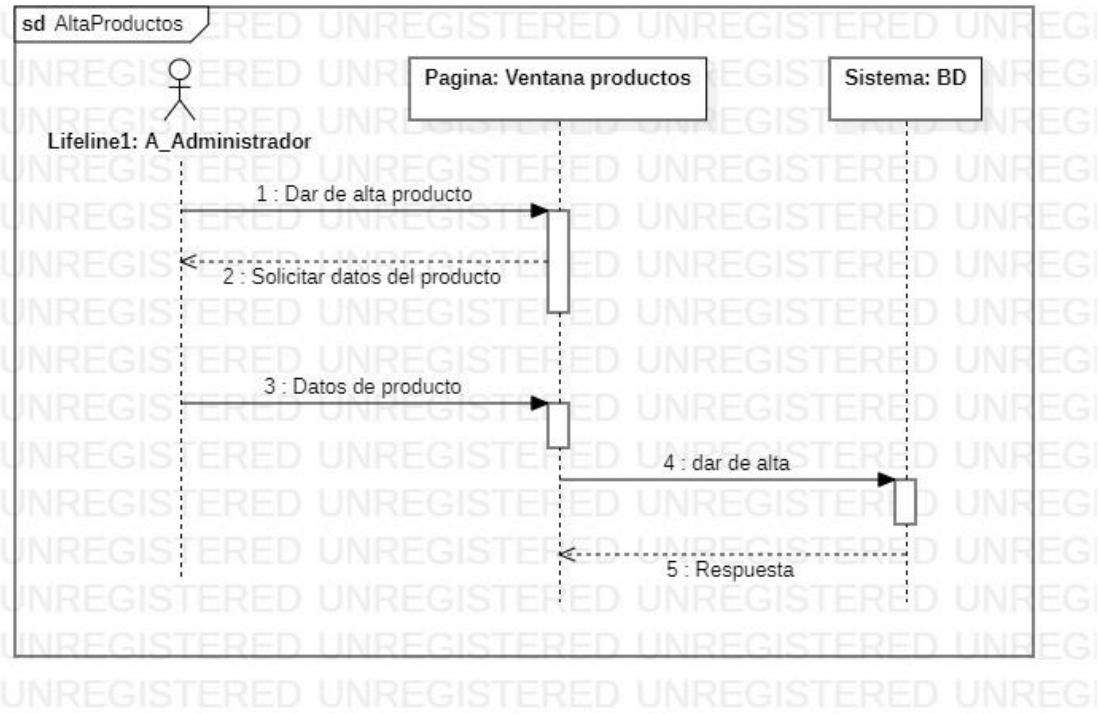


Imagen 17. Diagrama de secuencia alta productos

## Diagramas de estados

La imagen a continuación representa nuestro diagrama de estados, del cual solo requerimos uno.

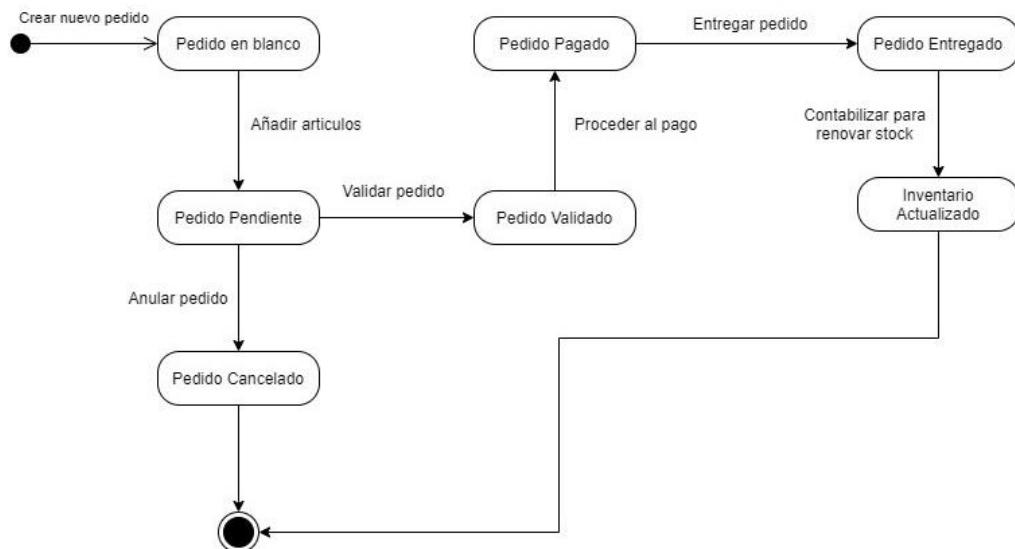


Imagen 18. Diagrama de estados.

## Diagramas de actividades

Al igual que el diagrama de estados, consideramos apropiado solo incluir un diagrama de actividades.

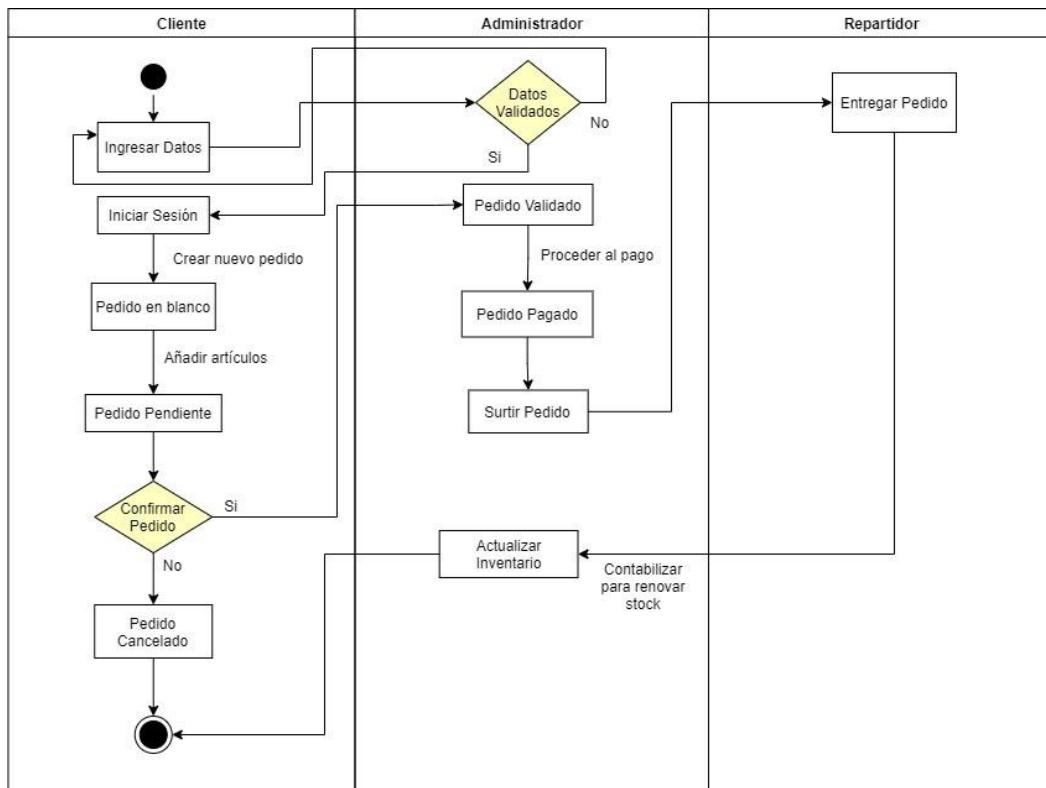


Imagen 19. Diagrama de actividades

## Capítulo 4

### Codificación del sistema

En este apartado se mostrarán las capturas más relevantes del sistema Dulcemprende con una explicación general.

#### Backend

En primer lugar, tenemos el api de autenticación del administrador que consiste en 3 partes, registro, inicio de sesión y cierre de sesión.

```
exports.signup = (req, res) => {
  User.findOne({ email: req.body.email }).exec((error, user) => {
    if (user)
      return res.status(400).json({
        message: "Ya te has registrado",
      });

    User.estimatedDocumentCount(async (err, count) => {
      if (err) return res.status(400).json({ error });

      const { firstName, lastName, email, password } = req.body;
      const hash_password = await bcrypt.hash(password, 10);
      const _user = new User({
        firstName,
        lastName,
        email,
        hash_password,
        username: shortid.generate(),
        role: 'admin',
      });

      _user.save((error, data) => {
        if (error)
          return res.status(400).json({
            message: "Algo salió mal. Intenta de nuevo",
          });
      }

        if (data) {
          return res.status(201).json({
            message: "Registro como administrador exitoso!",
          });
        }
      });
    });
  });
};
```

Imagen 20. Registro administrador

```

exports.signin = (req, res) => {
  User.findOne({ email: req.body.email }).exec(async (error, user) => {
    if (error) return res.status(400).json({ error });
    if (user) {
      const isPassword = await user.authenticate(req.body.password);
      if (
        isPassword &&
        user.role === "admin"
      ) {
        const token = jwt.sign(
          { _id: user._id, role: user.role },
          process.env.JWT_SECRET,
          { expiresIn: "1d" }
        );
        const { _id, firstName, lastName, email, role, fullName } = user;
        res.cookie("token", token, { expiresIn: "1d" });
        res.status(200).json({
          token,
          user: { _id, firstName, lastName, email, role, fullName },
        });
      } else {
        return res.status(400).json({
          message: "Contraseña incorrecta",
        });
      }
    } else {
      return res.status(400).json({ message: "Algo salio mal, intenta de nuevo" });
    }
  });
};

```

*Imagen 21. Inicio de sesión del administrador*

```

exports.signout = (req, res) => {
  res.clearCookie("token");
  res.status(200).json({
    message: "Cerraste sesión con éxito!",
  });
};

```

*Imagen 22. Cierre de sesión de administrador.*

De igual forma, contamos con el api de autenticación del cliente que consiste en registro, inicio de sesión y cierre de sesión.

```

exports.signup = (req, res) => {
  User.findOne({ email: req.body.email }).exec(async (error, user) => [
    if (user)
      return res.status(400).json({
        error: "Ya te has registrado",
      });

    const { firstName, lastName, email, password } = req.body;
    const hash_password = await bcrypt.hash(password, 10); You, 2 weeks ago .
    const _user = new User({
      firstName,
      lastName,
      email,
      hash_password,
      username: shortid.generate(),
    });

    _user.save((error, user) => {
      if (error) {
        return res.status(400).json({
          message: "Algo salió mal",
        });
      }

      if (user) {
        const token = generateJwtToken(user._id, user.role);
        const { _id, firstName, lastName, email, role, fullName } = user;
        return res.status(201).json({
          token,
          user: { _id, firstName, lastName, email, role, fullName },
        });
      }
    });
  ]);
};

```

Imagen 23. Registro de cliente

```

exports.signin = (req, res) => {
  User.findOne({ email: req.body.email }).exec(async (error, user) => {
    if (error) return res.status(400).json({ error });
    if (user) {
      const isPassword = await user.authenticate(req.body.password);
      if (isPassword && user.role === "user") {
        // const token = jwt.sign(
        //   { _id: user._id, role: user.role },
        //   process.env.JWT_SECRET,
        //   { expiresIn: "1d" }
        // );
        const token = generateJwtToken(user._id, user.role);
        const { _id, firstName, lastName, email, role, fullName } = user;
        res.status(200).json({
          token,
          user: { _id, firstName, lastName, email, role, fullName },
        });
      } else {
        return res.status(400).json({
          message: "Contraseña incorrecta",
        });
      }
    } else {
      return res.status(400).json({ message: "Algo salió mal" });
    }
  });
};

```

Imagen 24. Inicio de sesión del cliente

De igual forma se realizaron apis para el carrito del cliente, sus órdenes, estadísticas, productos y categorías, la parte esencial del código se ve en las siguientes capturas.

```

exports.addItemToCart = (req, res) => {
  Cart.findOne({ user: req.user._id }).exec((error, cart) => {
    if (error) return res.status(400).json({ error });
    if (cart) {
      //if cart already exists then update cart by quantity
      let promiseArray = [];

      req.body.cartItems.forEach(cartItem => {
        const product = cartItem.product;
        const item = cart.cartItems.find(c => c.product == product);
        let condition, update;
        if (item) {
          condition = { user: req.user._id, "cartItems.product": product };
          update = {
            $set: {
              "cartItems.$": cartItem,
            },
          };
        } else {
          condition = { user: req.user._id };
          update = {
            $push: {
              cartItems: cartItem,
            },
          };
        }
        promiseArray.push(runUpdate(condition, update));
        //Cart.findOneAndUpdate(condition, update, { new: true }).exec();
        // .exec((error, _cart) => {
        //   if(error) return res.status(400).json({ error });
        //   if(_cart){
        //     //return res.status(201).json({ cart: _cart });
        //     updateCount++;
        //   }
        // })
      });
      Promise.all(promiseArray)
        .then((response) => res.status(201).json({ response }))
        .catch((error) => res.status(400).json({ error }));
    }
  });
}

```

Imagen 25. API carrito primera parte

```

exports.getCartItems = (req, res) => {
  //const { user } = req.body.payload;
  //if(user){
    Cart.findOne({ user: req.user._id })
      .populate("cartItems.product", "_id name price productPictures")
      .exec((error, cart) => {
        if (error) return res.status(400).json({ error });
        if (cart) {
          let cartItems = {};
          cart.cartItems.forEach((item, index) => {
            cartItems[item.product._id.toString()] = {
              _id: item.product._id.toString(),
              name: item.product.name,
              img: item.product.productPictures[0].img,
              price: item.product.price,
              qty: item.quantity,
            };
          });
          res.status(200).json({ cartItems });
        }
      });
    //}
  };
};

// new update remove cart items
exports.removeCartItems = (req, res) => {
  const { productId } = req.body.payload;
  if (productId) {
    Cart.update(
      { user: req.user._id },
      {
        $pull: {
          cartItems: {
            product: productId,
          },
        },
      }
    ).exec((error, result) => {
      if (error) return res.status(400).json({ error });
      if (result) {
        res.status(202).json({ result });
      }
    });
  }
};

```

Imagen 26. API carrito segunda parte

```

exports.addCategory = (req, res) => {
  const categoryObj = {
    name: req.body.name,
    slug: `${slugify(req.body.name)}-${shortid.generate()}`,
    createdBy: req.user._id,
  };

  if (req.file) {
    categoryObj.categoryImage = "/public/" + req.file.filename;
  }

  if (req.body.parentId) {
    categoryObj.parentId = req.body.parentId;
  }

  const cat = new Category(categoryObj);
  cat.save((error, category) => {
    if (error) return res.status(400).json({ error });
    if (category) {
      return res.status(201).json({ category });
    }
  });
};

exports.getCategories = (req, res) => {
  Category.find({}).exec((error, categories) => {
    if (error) return res.status(400).json({ error });
    if (categories) {
      const categoryList = createCategories(categories);
      res.status(200).json({ categoryList });
    }
  });
};

exports.updateCategories = async (req, res) => {
  const { _id, name, parentId, type } = req.body;
  const updatedCategories = [];
  if (name instanceof Array) {
    for (let i = 0; i < name.length; i++) {
      const category = {
        name: name[i],
        type: type[i],
      };
      if (parentId[i] !== "") {
        category.parentId = parentId[i];
      }
    }
  }
};

```

Imagen 27. Parte principal de la API de categorías

```

exports.addOrder = (req, res) => {
  Cart.deleteOne({ user: req.user._id }).exec((error, result) => {
    if (error) return res.status(400).json({ error });
    if (result) {
      req.body.user = req.user._id;
      req.body.orderStatus = [
        {
          type: "ordenado",
          date: new Date(),
          isCompleted: true,
        },
        {
          type: "empacado",
          isCompleted: false,
        },
        {
          type: "enviado",
          isCompleted: false,
        },
        {
          type: "recibido",
          isCompleted: false,
        },
      ];
      const order = new Order(req.body);
      order.save((error, order) => {
        if (error) return res.status(400).json({ error });
        if (order) {
          res.status(201).json({ order });
        }
      });
    }
  });
};

exports.getOrders = (req, res) => {
  Order.find({ user: req.user._id })
    .select("_id paymentStatus paymentType orderStatus items")
    .populate("items.productId", "_id name productPictures")
    .exec((error, orders) => {
      if (error) return res.status(400).json({ error });
      if (orders) {
        res.status(200).json({ orders });
      }
    });
};

```

Imagen 28. Parte esencial de la API de ordenes

```

exports.createProduct = (req, res) => {
  //res.status(200).json( { file: req.files, body: req.body } );

  const { name, price, description, category, quantity, createdBy } = req.body;
  let productPictures = [];

  if (req.files.length > 0) {
    productPictures = req.files.map((file) => {
      return { img: file.filename };
    });
  }

  const product = new Product({
    name,
    slug: slugify(name),
    price,
    quantity,
    description,
    productPictures,
    category,
    createdBy: req.user._id,
  });

  product.save((error, product) => {
    if (error) return res.status(400).json({ error });
    if (product) {
      res.status(201).json({ product, files: req.files });
    }
  });
};

exports.getProductsBySlug = (req, res) => {
  const { slug } = req.params;
  Category.findOne({ slug: slug })
    .select("_id type")
    .exec((error, category) => {
      if (error) {
        return res.status(400).json({ error });
      }

      if (category) {
        Product.find({ category: category._id }).exec((error, products) => {
          if (error) {
            return res.status(400).json({ error });
          }
        });
      }
    });
};

```

*Imagen 29. Primera parte del API de productos.*

```

exports.getProductDetailsById = (req, res) => {
  const { productId } = req.params;
  if (productId) {
    Product.findOne({ _id: productId }).exec((error, product) => {
      if (error) return res.status(400).json({ error });
      if (product) {
        res.status(200).json({ product });
      }
    });
  } else {
    return res.status(400).json({ error: "Parametros requeridos" });
  }
};

// new update
exports.deleteProductById = (req, res) => {
  const { productId } = req.body.payload;
  if (productId) {
    Product.deleteOne({ _id: productId }).exec((error, result) => {
      if (error) return res.status(400).json({ error });
      if (result) {
        res.status(202).json({ result });
      }
    });
  } else {
    res.status(400).json({ error: "Parametros requeridos" });
  }
};

exports.getProducts = async (req, res) => {
  const products = await Product.find({ createdBy: req.user._id })
    .select("_id name price quantity slug description productPictures category")
    .populate({ path: "category", select: "_id name" })
    .exec();

  res.status(200).json({ products });
};

```

Imagen 30 Segunda parte de la API del producto

```

exports.addStatistics = (req, res) => {
  const { payload } = req.body
  if(payload.statistics){
    if(payload.statistics._id){
      UserStatistics.findOneAndUpdate(
        { user: req.user._id, "statistics._id": payload.statistics._id },
        {
          $set: {
            "statistics.$": payload.statistics
          }
        }
      ).exec((error, statistics) => {
        if (error) return res.status(400).json({error})
        if (statistics){
          res.status(201).json({statistics})
        }
      })
    } else{
      UserStatistics.findOneAndUpdate(
        { user: req.user._id },
        {
          $push: {
            statistics: payload.statistics,
          },
          { new: true, upsert: true }
        }
      ).exec((error, statistics) => {
        if(error) return res.status(400).json({ error })
        if(statistics){
          res.status(201).json({ statistics })
        }
      })
    }
  }else{
    res.status(400).json({ error: "Algo salio mal..." })
  }
}

exports.getStatistics = (req, res) => {
  UserStatistics.find({ user: req.user_id }).exec((error, statistics) => {
    if(error) return res.status(400).json({ error })
    if(statistics){
      res.status(200).json({ statistics })
    }
  })
}

```

Imagen 31.API de las estadísticas del producto

En la siguiente captura esta el código usado que permite la conexión a la base de datos usada, que en este caso fue MoongoDB

```

env.config();

// mongodb connection
//mongodb+srv://root:<password>@cluster0.8pl1w.mongodb.net/<dbname>?retryWrites=true&w=majority
mongoose
.connect(
  // mongodb+srv://root:<password>@cluster0.vvfi8.mongodb.net/myFirstDatabase?retryWrites=true&w=majority
  `mongodb+srv://${process.env.MONGO_DB_USER}:${process.env.MONGO_DB_PASSWORD}@cluster0.5vkcn.mongodb.net/<dbname>?retryWrites=true&w=majority`
  // `mongodb+srv://${process.env.MONGO_DB_USER}:${process.env.MONGO_DB_PASSWORD}@cluster0.cxwa7.mongodb.net/<dbname>?retryWrites=true&w=majority`
  {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
    useFindAndModify : false
  }
)
.then(() => {
  console.log("Conectado a la BD");
});

```

*Imagen 32. Conexión a la BD*

## Fronted

### Administrador

En este apartado se mostrará el código de las partes principales del administrador de la parte del fronted de nuestra aplicación

```

const Products = (props) => {
  const [name, setName] = useState("");
  const [quantity, setQuantity] = useState("");
  const [price, setPrice] = useState("");
  const [description, setDescription] = useState("");
  const [categoryId, setCategoryId] = useState("");
  const [productPictures, setProductPictures] = useState([]);
  const [show, setShow] = useState(false);
  const [productDetailModal, setProductDetailModal] = useState(false);
  const [productDetails, setProductDetails] = useState(null);
  const category = useSelector((state) => state.category);
  const product = useSelector((state) => state.product);
  const dispatch = useDispatch();

  const handleClose = () => {
    setShow(false);
  };

  const submitProductForm = () => {
    if(name.length < 1){
      alert("Ingresa un nombre del producto")
      return
    }
    if(quantity.length < 1){
      alert("Ingresa una cantidad del producto")
      return
    }
    if(price.length < 1){
      alert("Ingresa un precio del producto")
      return
    }
    if(description.length < 1){
      alert("Ingresa una descripción al producto")
      return
    }
    if(categoryId.length < 1){
      alert("Ingresa una categoría al producto")
      return
    }
    if(productPictures.length < 1){
      alert("Ingresa al menos una imagen al producto")
      return
    }
    const form = new FormData();
    form.append("name", name);
    form.append("quantity", quantity);
    form.append("price", price);
  };
}

```

Imagen 33. Sección de productos primera parte

```

    for (let pic of productPictures) {
      form.append("productPicture", pic);
    }

    dispatch(addProduct(form)).then(() => setShow(false));

    setName("")
    setQuantity("")
    setPrice("")
    setDescription("")
    setCategoryId("")
    setProductPictures([])
  };
  const handleShow = () => setShow(true);

  const createCategoryList = (categories, options = []) => {
    for (let category of categories) {
      options.push({ value: category._id, name: category.name });
      if (category.children.length > 0) {
        createCategoryList(category.children, options);
      }
    }

    return options;
  };

  const handleProductPictures = (e) => {
    setProductPictures([...productPictures, e.target.files[0]]);
  };

  const renderProducts = () => {
    return (
      <Table style={{ fontSize: 14 }} responsive="sm">
        <thead>
          <tr>
            <th>Nombre</th>
            <th>Precio</th>
            <th>Cantidad</th>
            <th>Categoria</th>
            <th>Acciones</th>
          </tr>
        </thead>
        <tbody>
          {product.products.length > 0
            ? product.products.map((product) => (
              <tr key={product._id}>

```

Imagen 34. Sección de productos segunda parte

```

        <tr key={product._id}>
          <td>{product.name}</td>
          <td>{product.price}</td>
          <td>{product.quantity}</td>
          <td>{product.category.name}</td>
          <td>
            <Button onClick={() => showProductDetailsModal(product)} variant="primary">
              <i className="fas fa-info"></i>
            </Button>
            <Button
              onClick={() => {
                const payload = {
                  productId: product._id,
                };
                dispatch(deleteProductById(payload));
              }}
              variant="danger"
            >
              <i className="fas fa-trash-alt"></i>
            </Button>
          </td>
        </tr>
      ))
    : null}
  </tbody>
</Table>
);
};

const renderAddProductModal = () => {
  return (
    <Modal
      show={show}
      handleClose={handleClose}
      modalTitle={"Añadir nuevo producto"}
      onSubmit={submitProductForm}
    >
      <Input
        label="Nombre"
        value={name}
        placeholder="Nombre del producto"
        onChange={(e) => setName(e.target.value)}
      />
    
```

Imagen 35. Sección de productos tercera parte

```

<Input
  label="Cantidad"
  value={quantity}
  placeholder={`Cantidad`}
  onChange={(e) => setQuantity(e.target.value)}
/>
<Input
  label="Precio"
  value={price}
  placeholder={`Precio`}
  onChange={(e) => setPrice(e.target.value)}
/>
<Input
  label="Descripción"
  value={description}
  placeholder={`Descripción`}
  onChange={(e) => setDescription(e.target.value)}
/>
<select
  className="form-control"
  value={categoryId}
  onChange={(e) => setCategoryId(e.target.value)}
>
  <option>select category</option>
  {createCategoryList(category.categories).map((option) => (
    <option key={option.value} value={option.value}>
      {option.name}
    </option>
  ))}
</select>
{productPictures.length > 0
  ? productPictures.map((pic, index) => (
    <div key={index}>{pic.name}</div>
  ))
  : null}
<input
  type="file"
  name="productPicture"
  onChange={handleProductPictures}
/>
</Modal>
);
};

```

Imagen 36. Sección de productos cuarta parte

```

const Orders = (props) => {
  const order = useSelector((state) => state.order);
  const [type, setType] = useState("");
  const dispatch = useDispatch();

  const onOrderUpdate = (orderId) => {
    const payload = {
      orderId,
      type,
    };
    dispatch(updateOrder(payload));
  };

  const formatDate = (date) => {
    if (date) {
      const d = new Date(date);
      return `${d.getFullYear()}-${d.getMonth() + 1}-${d.getDate()}`;
    }
    return "";
  };

  return (
    <Layout sidebar>
      {order.orders.map((orderItem, index) => (
        <Card
          style={{ margin: "10px 0", }}
          key={index}
          headerLeft={orderItem._id}
        >
          <div
            style={{
              display: "flex",
              justifyContent: "space-between",
              padding: "50px 50px",
              alignItems: "center",
            }}
          >
            <div>
              <div className="title">Productos</div>
              {orderItem.items.map((item, index) => (
                <div className="value" key={index}>
                  {item.productId.name}
                </div>
              ))}
            </div>
          </div>
        </Card>
      ))}
    </Layout>
  );
};

export default Orders;

```

Imagen 37. Sección de administración de órdenes primera parte.

```

    <span className="title">Costo total</span>
    <br />
    <span className="value">{orderItem.totalAmount}</span>
</div>
<div>
    <span className="title">Tipo de pago</span> <br />
    <span className="value">{orderItem.paymentType}</span>
</div>

</div>
<div
    className="container"
    style={{
        boxSizing: "border-box",
        padding: "10px",
        display: "flex",
        alignItems: "center",
    }}
>
    <div className="orderTrack">
        {orderItem.orderStatus.map((status) => (
            <div
                className={`orderStatus ${{
                    status.isCompleted ? "active" : ""
                }}`}
            >
                <div
                    className={`point ${status.isCompleted ? "active" : ""}`}
                ></div>
                <div className="orderInfo">
                    <div className="status">{status.type}</div>
                    <div className="date">{formatDate(status.date)}</div>
                </div>
            </div>
        )));
    </div>
    /* select input to apply order action */
    <div
        style={{
            padding: "0 50px",
            boxSizing: "border-box",
        }}
    >
        <select onChange={(e) => setType(e.target.value)}>
            <option value="">Status</option>
            {orderItem.orderStatus.map((status) => {
                return (

```

Imagen 38. Sección de administración de ordenes segunda parte.

```

        return (
          <>
            {!status.isCompleted ? (
              <option key={status.type} value={status.type}>
                {status.type}
              </option>
            ) : null}
          </>
        );
      )}
    </select>
  </div>
  {/* button to confirm action */}

  <div
    style={{
      padding: "0 50px",
      boxSizing: "border-box",
    }}
  >
    <button onClick={() => onOrderUpdate(orderItem._id)}>
      Confirmar
    </button>
  </div>
</div>
</Card>
));
);
</Layout>
);
};

export default Orders;

```

Imagen 39. Sección de administración de ordenes tercera parte.

```

return (
  <Layout sidebar>
    <Container style={{marginLeft: "10px"}}>
      <Row style={{ marginTop: '50px', fontSize: 30 }}>
        <Col md={{span: 6, offset: 3}}>
          <Form onSubmit={userLogin}>
            <Input
              label="Correo"
              placeholder="Correo"
              value={email}
              type="email"
              onChange={(e) => setEmail(e.target.value)}
            />

            <Input
              label="Contraseña"
              placeholder="Contraseña"
              value={password}
              type="password"
              onChange={(e) => setPassword(e.target.value)}
            />
            <Button variant="primary" type="submit">
              Iniciar sesión
            </Button>
          </Form>
          {user.message}
        </Col>
      </Row>
    </Container>
  </Layout>
)

```

Imagen 40. Componente de inicio de sesión de administrador

```

return (
  <Layout sidebar>
    <Container style={{marginLeft: "10px"}}>
      <Row style={{ marginTop: "50px", fontSize: 30 }}>
        <Col md={{ span: 6, offset: 3 }}>
          <Form onSubmit={userSignup}>
            <Row>
              <Col md={6}>
                <Input
                  label="Nombre"
                  placeholder="Nombre"
                  value={firstName}
                  type="text"
                  onChange={(e) => setFirstName(e.target.value)}>
                />
              </Col>
              <Col md={6}>
                <Input
                  label="Apellido"
                  placeholder="Apellido"
                  value={lastName}
                  type="text"
                  onChange={(e) => setLastName(e.target.value)}>
                />
              </Col>
            </Row>

            <Input
              label="Correo"
              placeholder="Correo"
              value={email}
              type="email"
              onChange={(e) => setEmail(e.target.value)}>
            />

            <Input
              label="Contraseña"
              placeholder="Contraseña"
              value={password}
              type="password"
              onChange={(e) => setPassword(e.target.value)}>
            />
            <Button variant="primary" type="submit">
              Registrarse
            </Button>
          </Form>
        </Col>
      </Row>
    </Container>
)

```

Imagen 41. Componente de registro de administrador

## Cliente

En esta parte del cliente se mostrarán los componentes principales como el de la tienda, el chat y panel de administración.

```
const ChatComponent = () =>{

  const uRef = useRef()
  const auth = useSelector((state) => state.auth);
  const messagesRef = firestore.collection('messages')
  const query = messagesRef.orderBy('createdAt').limit(25);
  const [messages] = useCollectionData(query, {idField: 'id'})
  console.log(messages);
  const [formValue, setformValue] = useState()
  const sendMessage = async(e) => {
    e.preventDefault()

    const { _id } = auth.user

    await messagesRef.add({
      text: formValue,
      createdAt: firebase.firestore.FieldValue.serverTimestamp(),
      _id
    })

    setformValue('')
    uRef.current.scrollIntoView({ behavior: 'smooth' })
  }
  return (
    <>
      <main className="chatMain">
        <h1>Iniciaste sesión como {auth.user.fullName}</h1>      You, seconds ago • Uncommitted changes
        <div className="divisor"></div>
        {messages && messages.map(msg => <ChatMessage key={msg.id} message={msg} />)}
        <div ref={uRef}></div>
        <span ref={uRef}></span>
      </main>
      <form onSubmit={sendMessage} className="chatForm">
        <input value={formValue} onChange={(e) => setformValue(e.target.value)} className="chatInput"/>
        <button type="submit" className="chatButton">Send</button>
      </form>
    </>
  )
}

const ChatMessage= (props) => {
  const auth = useSelector((state) => state.auth);

  const { text, uid } = props.message
  console.log(auth);
  const messageClass = uid === auth.user._id ? 'sent' : 'received'
  return(
    <div className={`message ${messageClass}`}>
      <img src={man} />
      <p className="para">{text}</p>
    </div>
  )
}
```

Imagen 42. Componente de Chat.

```

const loggedInInMain = () => {
  return (
    <div className="main__container">
      <div className="main__title">
        <div className="main__greeting">
          <h1>Hola {auth.user.fullName}</h1>
          <p>Bienvenido a Dulcemprende. Administra tu emprendimiento!</p>
        </div>
      </div>
      <div className="main__cards">
        <div className="card">
          <i className="fa fa-user fa-2x text-lightblue"></i>
          <div className="card_inner">
            <p className="text-primary-p">Vendido: </p>
            <form onSubmit={handleBenefits}>
              <span>
                <input
                  value={sell}
                  type="number"
                  onChange={(e) => setSell(e.target.value)}
                />
              </span>
              <button type="submit">★</button>
            </form>
          </div>
        </div>
      </div>
    </div>

    <div className="charts">
      <div className="charts_left">
        <div className="charts_left_title">
          <div>
            <h1>Ventas</h1>
            <p>Tu emprendimiento</p>
          </div>
          <i className="fas fa-dollar-sign"></i>
        </div>

        <div>
          <div className="card1">
            <h1>Inversión Total</h1>
            <h3>{invest} MXN</h3>
          </div>
          <div className="card2">
            <h1>Última Inversión</h1>
            <h3>{lastInvest} MXN</h3>
          </div>
        </div>
      </div>
      <div className="charts_right">
        <div className="charts_right_title">
          <div>
            <h1>Estadísticas</h1>
            <p>Así va tu negocio!</p>
          </div>
          <i className="fas fa-dollar-sign"></i>
        </div>
      </div>
    </div>
  )
}

```

Imagen 43. Componente de administración de negocio.

```

return (
  <>
  <div className="container">
    <Navbar
      sidebarOpen={setSidebarOpen}
      openSidebar={openSidebar}
      element={element}
    />

    <main>
      <Layout>
        <div className="headerStore">
          <div className="hcontainer">
            <div className="row">
              <div className="col-2">
                <h1>Empieza a emprender con Dulcemprende!</h1>
                <p>
                  El secreto de un gran negocio consiste en saber algo que
                  nadie más sabe. Aristoteles Onassis
                </p>
                <a href="#comienza" className="btn">
                  Comienza! <i className="fas fa-long-arrow-alt-right"></i>
                </a>
              </div>
              <div className="col-2">
                <img src={emprende} />
              </div>
            </div>
          </div>
        </div>
      </Layout>
      <div className="small-container">
        <h2 className="title" id="comienza">
          <Link
            to={`Dulces-mas-vendidos-1N_KvLZYf?cid=60b5cb7788b9c648043e6374&type=store`}
          >
            Productos Más Vendidos
          </Link>
        </h2>
        <div className="row">
          <div className="col-4">
            <img src={mamut} alt="" />
            <h4>Mini Mamut</h4>
            <p>$33</p>
          </div>
          <div className="col-4">
            <img src={carlos} alt="" />
            <h4>Carlos V stick</h4>
            <p>$23</p>
          </div>
          <div className="col-4">
            <img src={bubu} alt="" />
            <h4>Mini Bulbulubu</h4>
            <p>$45</p>
          </div>
          <div className="col-4">
            <img src={duraz} alt="" />
            <h4>Lucky Gummys Durazno</h4>
            <p>$60</p>
          </div>
        </div>
      </div>
    </main>
  </div>
)

```

Imagen 44. Parte principal de la tienda primera parte

```

91           |     </div>
92           |     </div>
93
94     <div className="small-container">
95       <h2 className="title">
96         <Link
97           to={`Dulces-mas-vendidos-1N_KvLZYf?cid=60b5cb7788b9c648043e6374&type=store`}
98         >
99           Productos en Oferta
100        </Link>
101      </h2>
102    <div className="row">
103      <div className="col-4">
104        <img src={nusi} alt="" />
105        <h4>Nucita Chocolate-Vainilla</h4>
106        <p>$12</p>
107      </div>
108      <div className="col-4">
109        <img src={roc} alt="" />
110        <h4>Mini Rocko</h4>
111        <p>$33</p>
112      </div>
113      <div className="col-4">
114        <img src={paya} alt="" />
115        <h4>Paleta Payaso</h4>
116        <p>$74</p>
117      </div>
118      <div className="col-4">
119        <img src={wa} alt="" />
120        <h4>Galleta Wafer Chocolate</h4>
121        <p>$13</p>
122      </div>
123      <div className="col-4">
124        <img src={boli} alt="" />
125        <h4>Bolitochas Sandía</h4>
126        <p>$58</p>
127      </div>
128      <div className="col-4">
129        <img src={va} alt="" />
130        <h4>Galleta Wafer Vainilla</h4>
131        <p>$13</p>
132      </div>
133    </div>
134  </div>
135
136  <div className="offer">
137    <div className="small-container">
138      <div className="row">
139        <div className="col-2">
140          <img src={dulces} className="offer-img" />
141        </div>
142        <div className="col-2">
143          <h1>Dulces al mejor precio!</h1>
144        </div>
145      </div>
146    </div>
147  </div>
148  <div className="testimonial">
149    <div className="small-container">

```

Imagen 45. Parte principal de la tienda segunda parte

```

151 <div className="col-3">
152   <i class="fas fa-quote-right"></i>
153   <p>
154     Aprende del negocio. Muchos emprendedores primerizos
155     fallan en esos pequeños detalles de organización que son
156     críticos para un buen modelo de negocios. Cuando no puedes
157     traducir tus ideas al plano financiero y económico, corres
158     el riesgo de no poder replicar el éxito ni hacerte
159     entender por otros interesados.
160   </p>
161 </div>
162 <div className="col-3">
163   <i class="fas fa-quote-right"></i>
164   <p>
165     Rodéate de gente talentosa. Un buen emprendedor sabe sumar
166     a su causa el personal necesario para complementar su
167     visión. Hay que ser paciente para conseguir personas que
168     amen hacer lo que tú odias y también les apasione lo que
169     quieras lograr. No temas despedir a alguien si no están en
170     sintonía con tu misión y valores.
171   </p>
172 </div>
173 <div className="col-3">
174   <i class="fas fa-quote-right"></i>
175   <p>
176     Prepárate para los imprevistos. Para conseguir todos los
177     beneficios del emprendedurismo, debes saber que habrá
178     obstáculos por superar. Si bien no puedes controlarlo
179     todo, asegúrate de que el negocio pueda funcionar sin
180     tener que estar encima de cada situación. Lo ideal es
181     tener un balance entre tu vida personal y profesional.
182   </p>
183 </div>
184 <div className="col-3">
185   <i class="fas fa-quote-right"></i>
186   <p>
187     Abraza tu pasión. Por lo general, un emprendimiento
188     surgirá de aquello que te apasiona o que más llama tu
189     atención. Lo más difícil es mantener la llama viva con el
190     tiempo, así que debes asegurarte de nutrirla cada día.
191     Enfócate en los problemas de tus clientes y las razones
192     que te impulsaron a dar soluciones inteligentes.
193   </p>
194 </div>
195 <div className="col-3">
196   <i class="fas fa-quote-right"></i>
197   <p>
198     Escucha a tus futuros compradores. La clave para que una
199     empresa sea bien recibida en el mercado está en darle al
200     cliente lo que necesita. ¿Sólo eso? No. Lo ideal es
201     superar sus expectativas. Uno de los errores más comunes
202     es creer que tu negocio será un éxito sólo porque a ti te
203     gusta.
204   </p>
205 </div>

```

Imagen 46. Parte principal de la tienda tercera parte

```

const ProductStore = (props) => {
  const product = useSelector((state) => state.product);
  const priceRange = product.priceRange;
  const dispatch = useDispatch();

  useEffect(() => {
    const { match } = props;
    dispatch(getProductsBySlug(match.params.slug));
  }, []);

  return (
    <>
      /* <div className="row row-2">
        <h2>Productos</h2>
        <select>
          <option>Por defecto</option>
          <option>Ordenar de mayor a menos precio</option>
          <option>Ordenar de menor a mayor precio</option>
        </select>
      </div> */
      {Object.keys(product.productsByPrice).map((key, index) => {
        return (
          <div className="small-container" key={index}>

            <div className="row">
              [product.productsByPrice[key].map((product, index) => (
                <Link
                  to={`/ ${product.slug}/${product._id}/p`}
                  key={index}
                  className="col-4"
                >
                  <img
                    src={generatePublicUrl(product.productPictures[0].img)}
                    alt=""
                  />
                  <h4>{product.name}</h4>
                  <div>
                    <Rating value="5" />
                  </div>
                  <p>{product.price} MXN</p>
                </Link>
              ))]
              You, 2 weeks ago • Dulcemprende restructuring
            </div>
          </div>
        );
      ))}
    </>
  );
};

export default ProductStore;

```

Imagen 47. Componente de visualización de productos.

```

const ProductListPage = (props) => {
  let element = "Tienda";
  const [sidebarOpen, setSidebarOpen] = useState(false);
  const openSidebar = () => {
    setSidebarOpen(true);
  };

  const closeSidebar = () => {
    setSidebarOpen(false);
  };

  const renderProduct = () => {
    console.log(props);
    const params = useParams(props.location.search);
    let content = null;
    switch (params.type) {
      case "store":
        content = <ProductStore {...props} />;
        break;
    }

    return content;
  };

  return (
    <div className="container">
      <Navbar
        sidebarOpen={setSidebarOpen}
        openSidebar={openSidebar}
        element={element}
      />

      <main>
        <Layout>{renderProduct()}</Layout>
      </main>
      <Sidebar
        sidebarOpen={sidebarOpen}
        closeSidebar={closeSidebar}
        element={element}
      />
    </div>
  );
};

export default ProductListPage;

```

Imagen 48. Componente de detalles del producto.

```

<main>
  <Layout>
    <br />
    <br />
    <br />
    <div className="small-container cart-page">
      <table>
        <tr>
          <th>Producto</th>
          <th>Cantidad</th>
          <th>Total</th>
        </tr>
        {Object.keys(cartItems).map((key, index) => (
          <CartItem
            key={index}
            cartItem={cartItems[key]}
            onQuantityInc={onQuantityIncrement}
            onQuantityDec={onQuantityDecrement}
            onRemoveCartItem={onRemoveCartItem}
          />
        ))}
      </table>
      <PriceDetails
        totalItem={Object.keys(cart.cartItems).reduce(function (
          qty,
          key
        ) {
          return qty + cart.cartItems[key].qty;
        },
        0)}
        totalPrice={Object.keys(cart.cartItems).reduce(
          (totalPrice, key) => {
            const { price, qty } = cart.cartItems[key];
            return totalPrice + price * qty;
          },
          0
        )}
      />
      <br />
      <br />

      <MaterialButton
        title="Pagar"
        onClick={() => props.history.push(`/checkout`)}
      />
    </div>
    <br />
    <br />
    <br />
    <br />
    <br />
    <br />
  </Layout>
</main>
<Sidebar
  sidebarOpen={sidebarOpen}
  closeSidebar={closeSidebar}
  element={element}
/>

```

Imagen 49. Componente del carrito.

```

✓ const CheckoutStep = (props) => {
  return (
    <div className="checkoutStep">
      <div
        onClick={props.onClick}
        className={`checkoutHeader ${props.active && "active"}`}
      >
        <div>
          <span className="stepNumber">{props.stepNumber}</span>
          <span className="stepTitle">{props.title}</span>
        </div>
      </div>
      {props.body && props.body}
    </div>
  );
};

✓ const Address = ({
  adr,
  selectAddress,
  enableAddressEditForm,
  confirmDeliveryAddress,
  onAddressSubmit,
}) => {
  return (
    <div className="addressContainer">
      <div>
        <input name="address" onClick={() => selectAddress(adr)} type="radio" />
      </div>
      <div className="flexRow sb addressinfo">
        {!adr.edit ? (
          <div style={{ width: "100%" }}>
            <div className="addressDetail">
              <div>
                <span className="addressName">{adr.name}</span>
                <span className="addressType">{adr.addressType}</span>
                <span className="addressMobileNumber">{adr.mobileNumber}</span>
              </div>
            <div>
              {adr.selected && (
                <Anchor
                  name="Editar"
                  onClick={() => enableAddressEditForm(adr)}
                  style={{
                    fontWeight: "500",
                    color: "#2874f0",
                  }}
                />
              )}
            </div>
            <div className="fullAddress">
              {adr.address} <br /> ${adr.state} - ${adr.pinCode}
            </div>
          </div>
        ) : (
          <div>
            <div>
              <span>Nombre: {adr.name}</span>
              <span>Número: {adr.mobileNumber}</span>
            </div>
            <div>
              <span>Dirección:</span>
              {adr.address}
            </div>
            <div>
              <span>Estado:</span>
              {adr.state}
            </div>
            <div>
              <span>Código postal:</span>
              {adr.pinCode}
            </div>
          </div>
        )}
      </div>
    </div>
  );
};

```

Imagen 50. Página de checkout primera parte

```

        margin: "10px 0",
    )}
    />
)
</div>
) : (
<AddressForm
withoutLayout={true}
onSubmitForm={onAddressSubmit}
initialData={adr}
onCancel={() => {}}
/>
)
</div>
</div>
);
};

const CheckoutPage = (props) => {
let element = "Tienda";
const [sidebarOpen, setSidebarOpen] = useState(false);
const openSidebar = () => {
setSidebarOpen(true);
};

const closeSidebar = () => {
setSidebarOpen(false);
};
const user = useSelector((state) => state.user);
const auth = useSelector((state) => state.auth);
const [newAddress, setNewAddress] = useState(false);
const [address, setAddress] = useState([]);
const [confirmAddress, setConfirmAddress] = useState(false);
const [selectedAddress, setSelectedAddress] = useState(null);
const [orderSummary, setOrderSummary] = useState(false);
const [orderConfirmation, setOrderConfirmation] = useState(false);
const [paymentOption, setPaymentOption] = useState(false);
const [confirmOrder, setConfirmOrder] = useState(false);
const [paypalButtons, setPaypalButtons] = useState(false);
const cart = useSelector((state) => state.cart);
const dispatch = useDispatch();

const onAddressSubmit = (addr) => {
setSelectedAddress(addr);
setConfirmAddress(true);
setOrderSummary(true);
};

const selectAddress = (addr) => {
//console.log(addr);
const updatedAddress = address.map((adr) =>
adr._id === addr._id
? { ...adr, selected: true }
: { ...adr, selected: false }
);
setAddress(updatedAddress);
};

```

Imagen 51. Página de checkout segunda parte

```

const confirmDeliveryAddress = (addr) => {
  setSelectedAddress(addr);
  setConfirmAddress(true);
  setOrderSummary(true);
};

const enableAddressEditForm = (addr) => {
  const updatedAddress = address.map((adr) =>
    adr._id === addr._id ? { ...adr, edit: true } : { ...adr, edit: false }
  );
  setAddress(updatedAddress);
};

const userOrderConfirmation = () => {
  setOrderConfirmation(true);
  setOrderSummary(false);
  setPaymentOption(true);
};

const onConfirmOrder = () => [
  const totalAmount = Object.keys(cart.cartItems).reduce(
    (totalPrice, key) => {
      const { price, qty } = cart.cartItems[key];
      return totalPrice + price * qty;
    },
    0
  );
  const items = Object.keys(cart.cartItems).map((key) => ({
    productId: key,
    payablePrice: cart.cartItems[key].price,
    purchasedQty: cart.cartItems[key].qty,
  }));
  const payload = {
    addressId: selectedAddress._id,
    totalAmount,
    items,
    paymentStatus: "pending",
    paymentType: "paypal",
  };
  You, 2 weeks ago • Dulcemprende restructuring
  console.log(payload);
  dispatch(addOrder(payload));
  setConfirmOrder(true);
];

useEffect(() => {
  auth.authenticate && dispatch(getAddress());
  auth.authenticate && dispatch(getCartItems());
}, [auth.authenticate]);

useEffect(() => {
  const address = user.address.map((adr) => ({
    ...adr,
    selected: false,
    edit: false,
  }));
  setAddress(address);
};

```

Imagen 52. Página de checkout tercera parte

```

useEffect(() => {
  if (confirmOrder && user.placedOrderId) {
    props.history.push(`/order_details/${user.placedOrderId}`);
  }
}, [user.placedOrderId]);

return (
  <div className="container">
    <Navbar
      sidebarOpen={setSidebarOpen}
      openSidebar={openSidebar}
      element={element}
    />

    <main>
      <Layout>
        <div className="cartContainer" style={{alignItems: "center" }}>
          <div className="checkoutContainer">
            {/* check if user logged in or not */}
            <CheckoutStep
              stepNumber={"1"}
              title={"Inicia sesión"}
              active={!auth.authenticate}
              body={
                auth.authenticate ? (
                  <div className="loggedInId">
                    <span style={{ fontWeight: 500 }}>
                      | {auth.user.fullName}
                    </span>
                    <span style={{ margin: "0 5px" }}>{auth.user.email}</span>
                  </div>
                ) : (
                  <div>
                    | <p>Debes iniciar sesión</p>
                  </div>
                )
              }
            />
            <CheckoutStep
              stepNumber={"2"}
              title={"Dirección de envío"}
              active={!confirmAddress && auth.authenticate}
              body={
                <>
                  {confirmAddress ? (
                    <div className="stepCompleted">${selectedAddress.name} ${selectedAddress.address} - ${selectedAddress.zipCode}</div>
                  ) : (
                    address.map((adr) => (
                      <Address
                        selectAddress={selectAddress}
                        enableAddressEditForm={enableAddressEditForm}
                        confirmDeliveryAddress={confirmDeliveryAddress}
                        onAddressSubmit={onAddressSubmit}
                        adr={adr}
                      />
                    ))
                  )}
                </>
              }
            />
          </div>
        </div>
      </Layout>
    </main>
  </div>
)

```

Imagen 53. Página de checkout cuarta parte

```

/* AddressForm */
{confirmAddress ? null : newAddress ? (
  <AddressForm
    onSubmitForm={onAddressSubmit}
    onCancel={() => {}}
  />
) : auth.authenticate ? (
  <CheckoutStep
    stepNumber={"+"}
    title={"Añadir dirección"}
    active={false}
    onClick={() => setNewAddress(true)}
  />
) : null}

<CheckoutStep
  stepNumber={"3"}
  title={"Resumen de orden"}
  active={orderSummary}
  body={
    orderSummary ? (
      <CartPage onlyCartItems={true} />
    ) : orderConfirmation ? (
      <div className="stepCompleted">
        {Object.keys(cart.cartItems).length} items
      </div>
    ) : null
  }
/>

{orderSummary && (
  <Card
    style={{
      margin: "10px 0",
    }}
  >
    <div
      className="flexRow sb"
      style={{
        padding: "20px",
        alignItems: "center",
      }}
    >
      <p style={{ fontSize: "12px" }}>
        La confirmación será enviada a{" "}
        <strong>{auth.user.email}</strong>
      </p>
      <MaterialButton
        title="Continuar"
        onClick={userOrderConfirmation}
        style={{
          width: "200px",
        }}
      />
    </div>
  </Card>
)
}

```

Imagen 54. Página de checkout quinta parte

```

<CheckoutStep
  stepNumber={"4"}
  title="Opciones de pago"
  active={paymentOption}
  body={
    paymentOption && (
      <div>
        <div
          className="flexRow"
          style={{
            alignItems: "center",
            padding: "20px",
          }}
        >
          <input
            type="radio"
            name="paymentOption"
            value="paypal"
            onClick={() => setPaypalButtons(true)}
          />
          <div>Paypal</div>
          {paypalButtons ? (
            <PayPal onConfirmOrder={onConfirmOrder} />
          ) : (
            ""
          )}
        </div>
        <MaterialButton
          title="Confirmar orden"
          onClick={onConfirmOrder}
          style={{
            width: "200px",
            margin: "0 0 20px 20px",
          }}
        />
      </div>
    )
  }
</div>

/* Price Component */
<PriceDetails
  totalItem={Object.keys(cart.cartItems).reduce(function (
    qty,
    key
  ) {
    return qty + cart.cartItems[key].qty;
  },
  0)}
  totalPrice={Object.keys(cart.cartItems).reduce(
    (totalPrice, key) => {
      const { price, qty } = cart.cartItems[key];
      return totalPrice + price * qty;
    },
    0
  )}

```

Imagen 55. Página de checkout sexta parte

```

return (
  <div className="container">
    <Navbar
      sidebarOpen={setSidebarOpen}
      openSidebar={openSidebar}
      element={element}
    />

    <main>
      <Layout>
        <div className="small-container">
          <h1>Ordenes</h1>

          </div>
          <div className="fle">
            {user.orders.map((order) => {
              return order.items.map((item, key) => (
                <div className="small-container" key={key}>
                  <div className="row">
                    <Link
                      to={`/order_details/${order._id}`}
                      className="col-4"
                    >
                      <img
                        src={generatePublicUrl(
                          item.productId.productPictures[0].img
                        )}
                      />
                      <h4>{item.productId.name}</h4>
                      <p>
                        {item.payablePrice * item.purchasedQty} MXN
                      </p>
                      <p>
                        Orden ID: {order._id}
                      </p>
                    </Link>
                  </div>
                );
              )));
            })
          </div>
        </Layout>
      </main>
    </div>
  )
)

```

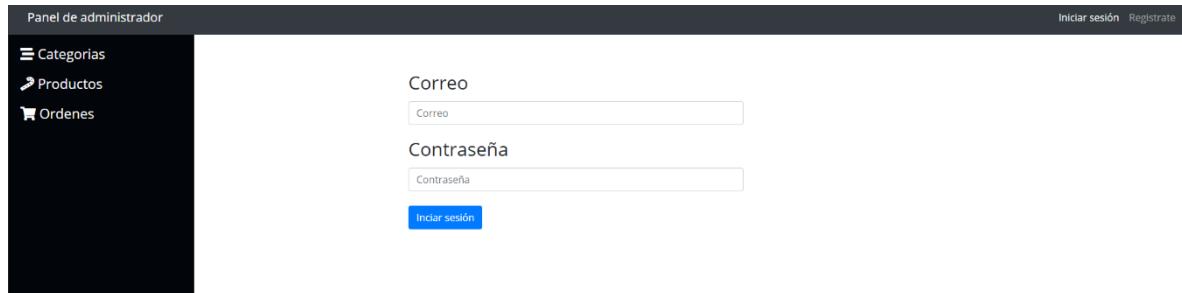
Imagen 56. Componente de las ordenes del cliente.

## Capítulo 5

### Prototipos

#### Administrador

##### Inicio de sesión del administrador

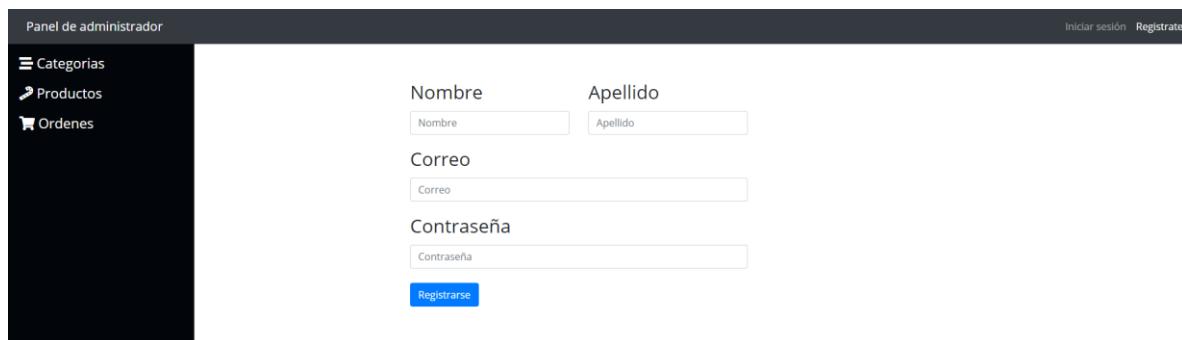


The screenshot shows the 'Panel de administrador' (Administrator Panel) interface. On the left is a dark sidebar with three menu items: 'Categorías' (Categories), 'Productos' (Products), and 'Ordenes' (Orders). The main area has a light background. At the top right are two buttons: 'Iniciar sesión' (Login) and 'Regístrate' (Register). Below these buttons is a form with two input fields: 'Correo' (Email) and 'Contraseña' (Password), each with its own input field. A blue 'Iniciar sesión' button is located at the bottom of the form.

Imagen 57. Pantalla de inicio de sesión del administrador

En la imagen 57 se puede ver la pantalla que muestra el formulario para que el administrador inicie sesión en el sistema

##### Registro del administrador



The screenshot shows the 'Panel de administrador' interface. The left sidebar contains the same three menu items: 'Categorías', 'Productos', and 'Ordenes'. The main area features a registration form. It includes two input fields for 'Nombre' (Name) and 'Apellido' (Last Name), which are grouped together. Below these are two more input fields: 'Correo' (Email) and 'Contraseña' (Password). A blue 'Registrarse' (Register) button is positioned at the bottom of the form. The overall layout is clean and organized.

Imagen 58. Pantalla de registro del administrador

En la imagen 58 podemos ver la imagen que muestra el apartado de registro del administrador.

##### Apartado de categorías

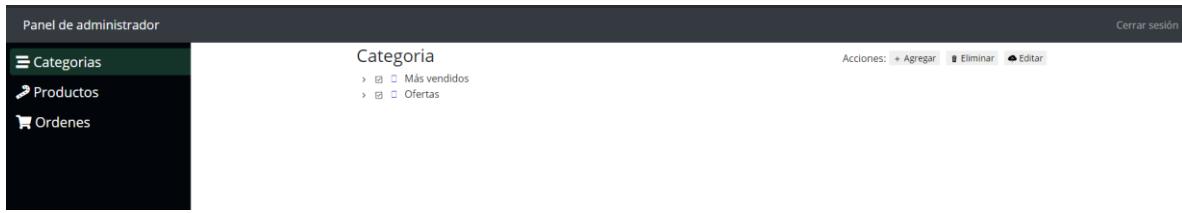


Imagen 59. Apartado de categorías

En la imagen 59 podemos ver el apartado de las categorias, el administrador puede agregar, eliminar y editar una categoría.

### Agregando categoría



Imagen 60. Modal para agregar categoría.

En la imagen 60 se ve el modal que se despliega al administrador cuando este quiere añadir una nueva categoría.

### Eliminando categoría

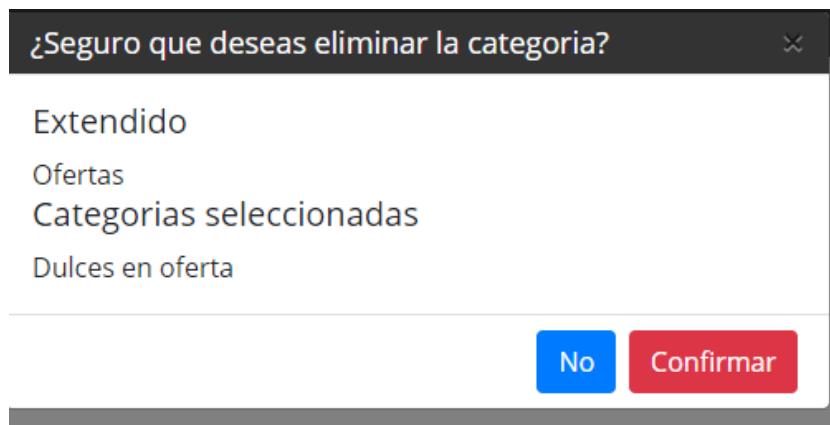


Imagen 61. Modal para eliminar categorías.

Como se ve en la imagen 61, al querer eliminar una categoría se muestra un mensaje de confirmación para la eliminación de las categorías seleccionadas o extendidas.

## Editando categoría

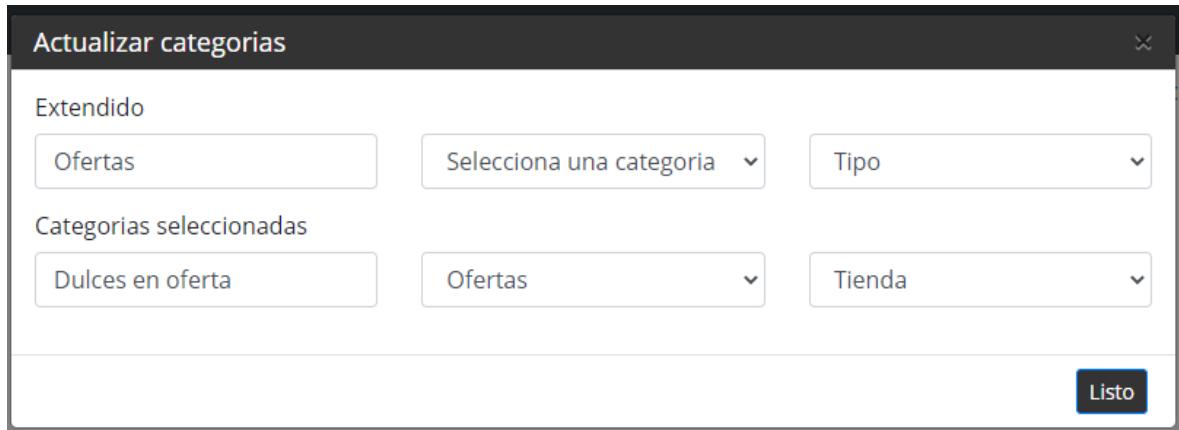


Imagen 62. Modal para editar categoría.

Como se ve en la imagen 62, al editar una categoría se abre un modal para modificar la categoría extendida o seleccionada.

## Apartado de productos

Productos					Añadir
Nombre	Precio	Cantidad	Categoría	Acciones	
Polvo Miguelito - 950 gramos	45	100	Dulces más vendidos		
Mini Mamut - 28 piezas	33	100	Dulces más vendidos		
Mazapán Azteca Rollo - 10 piezas	14	100	Dulces más vendidos		
Chocolate Bocadín - 50 piezas	49	100	Dulces más vendidos		
Mini Bubulubu - 25 piezas	45	100	Dulces más vendidos		
Chicle Canel's - 60 paquetes con 4 piezas c/u	25	100	Dulces más vendidos		

Imagen 63. Apartado de productos.

En la imagen 63 podemos ver el apartado de productos del administrador, el cual puede ver las características de cada producto, eliminarlo o añadir uno nuevo.

## Añadiendo nuevo producto

Añadir nuevo producto

Nombre

Cantidad

Precio

Descripción

select category

Seleccionar archivo Ningún archivo seleccionado

**Listo**

Imagen 64. Modal para añadir nuevo producto.

Al añadir un producto se abre un modal para ingresar sus características como se ve en la imagen 64.

### Información del producto.

Detalles del producto

<b>Nombre</b>	Polvo Miguelito - 950 Gramos	<b>precio</b>	45
<b>Cantidad</b>	100	<b>Categoría</b>	Dulces Más Vendidos
<b>Descripción</b>	950 Gramos. Azúcar Salada, Enchilada, Acidulada.		
<b>Imagenes del producto</b>			
 			

**Listo**

Imagen 65. Modal con la información del producto.

Como se ve en la imagen 65, al seleccionar la información de un producto se muestra una ventana modal para visualizar dicha información.

## Apartado de órdenes del cliente



Imagen 66. Apartado de órdenes.

Como se ve en la imagen 66, se muestra el apartado de las ordenes, donde el administrador puede editar el status de la orden de un cliente en específico.

## Cliente

### Pantalla principal del usuario no registrado



Imagen 67. Pantalla principal del cliente no registrado.

Como se ve en la imagen 67, se ve la pantalla que se le muestra a los usuarios no registrados y a la izquierda el panel donde pueden ver las diferentes acciones.

## Inicio de sesión del cliente

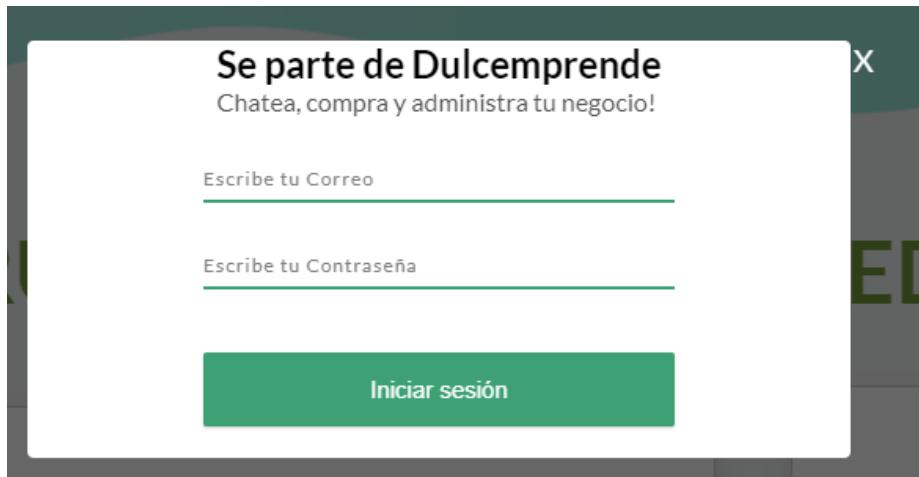


Imagen 68. Modal del inicio de sesión del cliente

En la imagen 68 podemos ver el formulario para el inicio de sesión del cliente de dulcemprende

## Registro del cliente

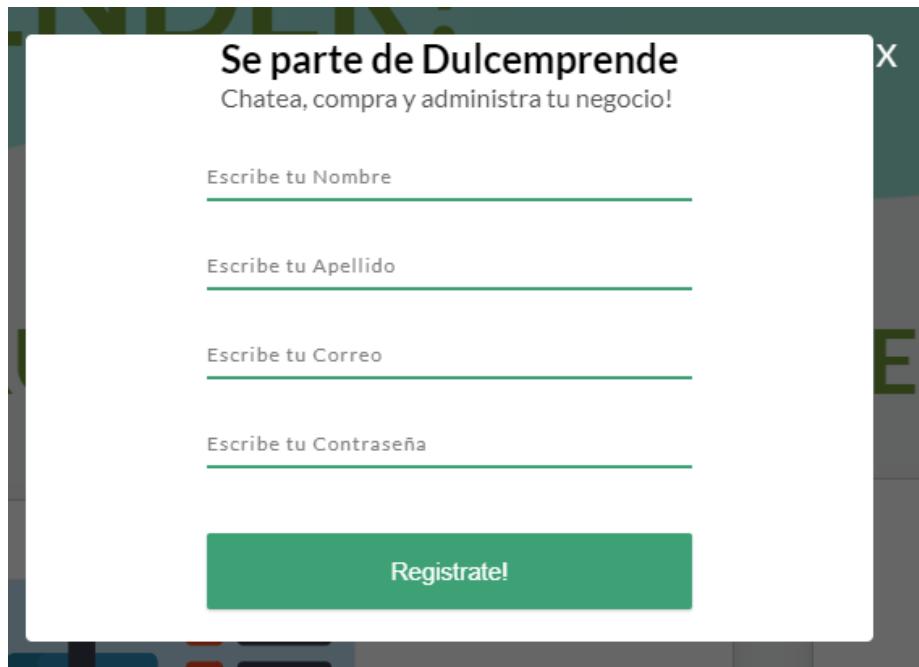


Imagen 69. Modal para el registro del usuario

En la imagen 69 podemos ver el formulario para el registro del cliente de dulcemprende

## Apartado de administración del negocio del cliente

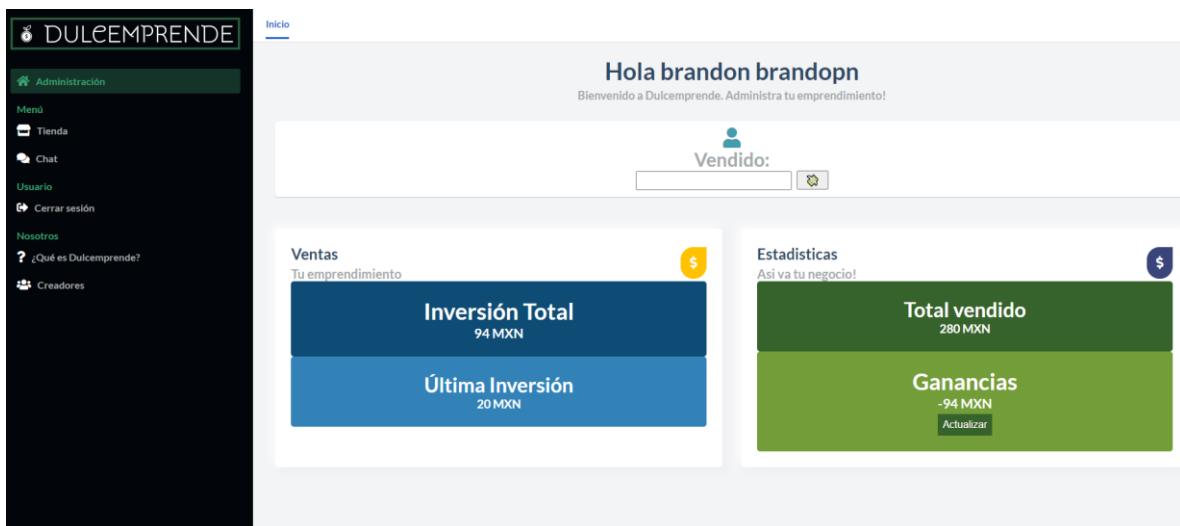


Imagen 70. Administración del negocio del cliente

Una vez dentro de su cuenta, el cliente puede ver esa parte de administración de su negocio donde se muestran varios componentes como su inversión total, ultima inversión, total vendido y sus ganancias.

## Apartado de chat

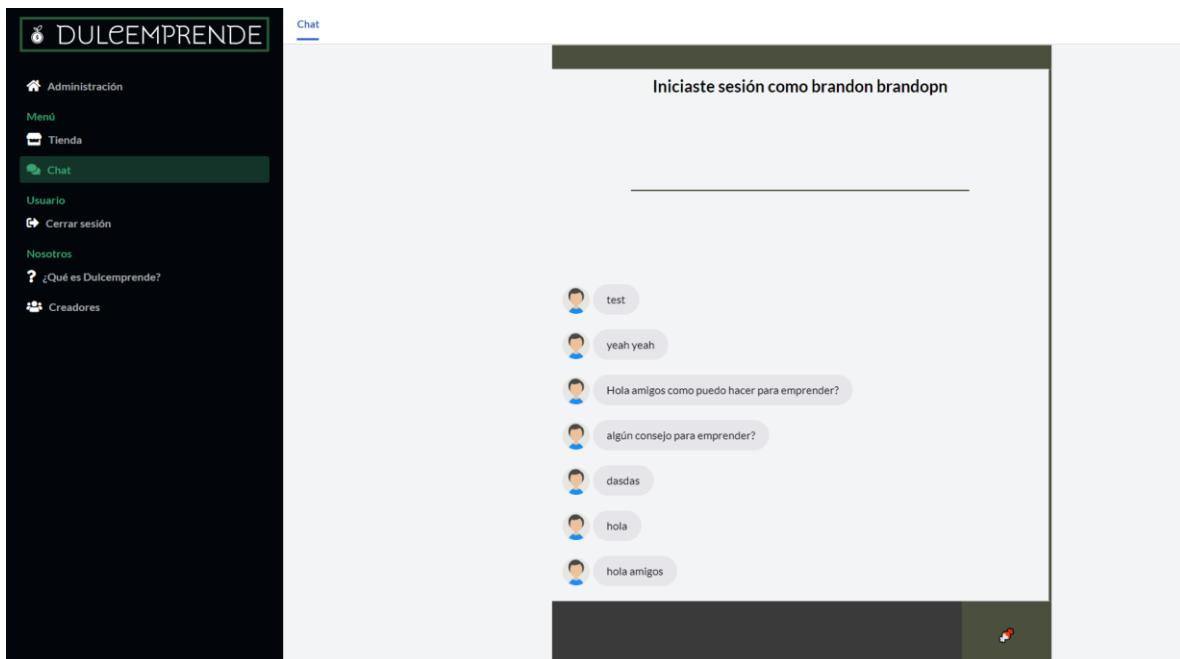


Imagen 71. Apartado de chat.

En la imagen 71 se ve el chat, el cual puede ser accedido por el usuario una vez que se haya registrado o iniciado sesión en su cuenta.

## Apartado de Sobre Dulcemprende

DULCEMPRENDE

SOBRE DULCEMPRENDE

Para aquellas personas emprendedoras

Dulcemprende es una plataforma que te permite comenzar con un emprendimiento del área de dulces, un mercado altamente demandado con futuro.

Ofrecemos distintas herramientas como lo es un completo ecommerce un chat para que los usuarios de Dulcemprende puedan compartir consejos, ideas y recomendaciones. Además de un apartado de administración donde el usuario puede ver lo que ha invertido, lo que ganado y lo que ha vendido de su negocio propio.

Repository on Github

Check the repository!

Checa el repositorio!

Puedes ver el repositorio del proyecto

DULCEMPRENDE

Otros Links

Sobre Dulcemprende

Sobre los creadores

Imagen 72. Sobre Dulcemprende

## Apartado de sobre los creadores

DULCEMPRENDE

Sobre los creadores

Creadores de Dulcemprende

Briones Tapia Daniela

Organizada y con gran motivación, adaptable a cualquier circunstancia y dando siempre lo mejor de mí en cualquier proyecto.

Meza Vargas Brandon David

Apasionado por la tecnología, siempre aprendiendo cosas nuevas y descubriendo lo que soy capaz.

Torres Jimenez Diego Antonio

Soy un tipo de trabajador que está acostumbrado al trabajo bajo presión, tengo varios años de experiencia en atención al cliente y en búsqueda de oportunidades comerciales. También me considero una persona resolutiva, con buen ánimo y capaz de resolver problemas fácilmente.

Administración

Tienda

Chat

Usuario

Cerrar sesión

Nosotros

¿Qué es Dulcemprende?

Creadores

Imagen 73. Sobre los creadores

## Pantalla de la vista principal de la tienda

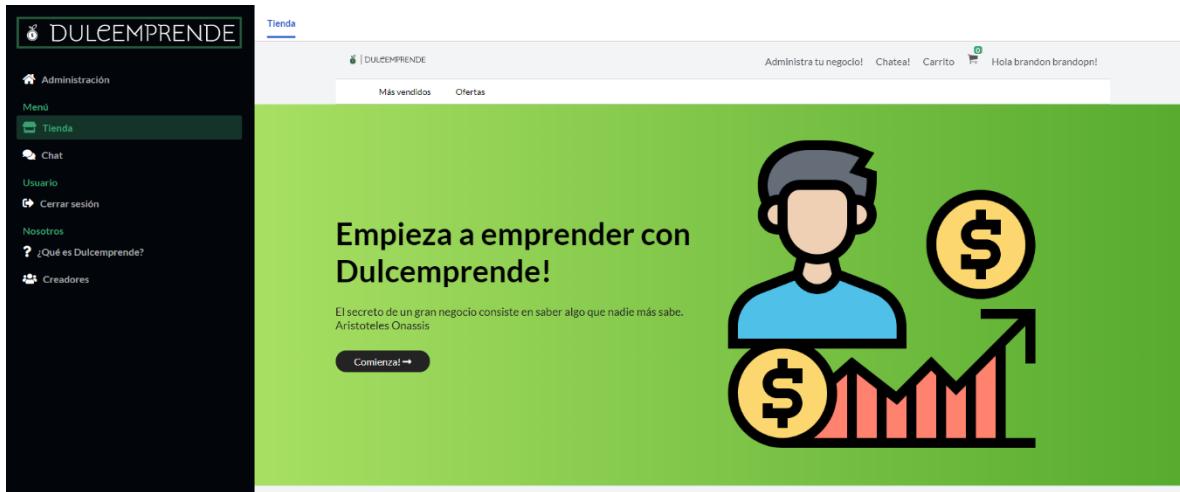


Imagen 74. Parte principal de la tienda primera parte.

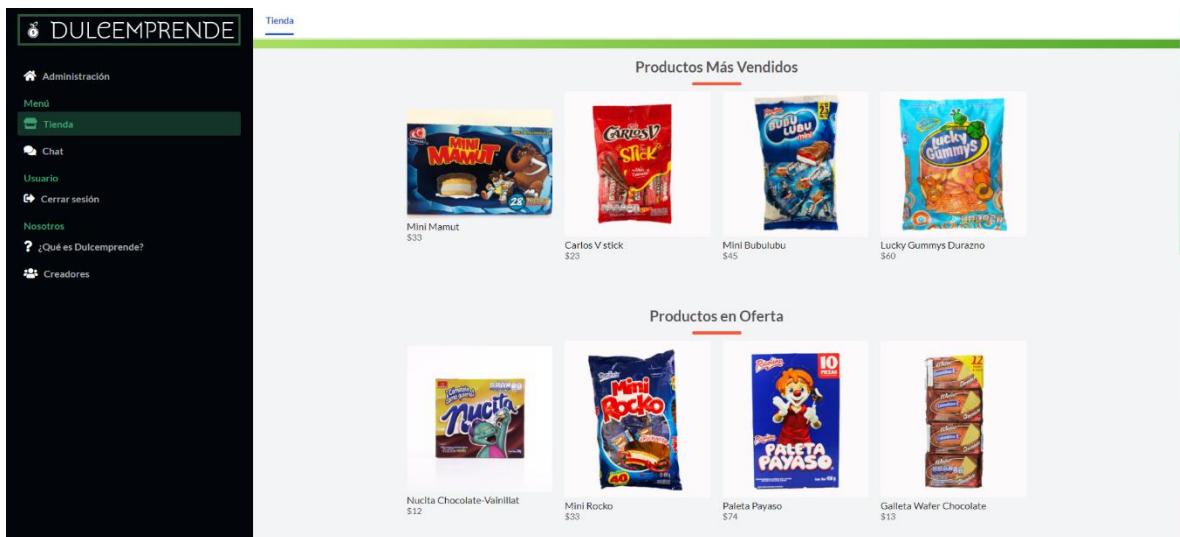


Imagen 75. Parte principal de la tienda segunda parte.

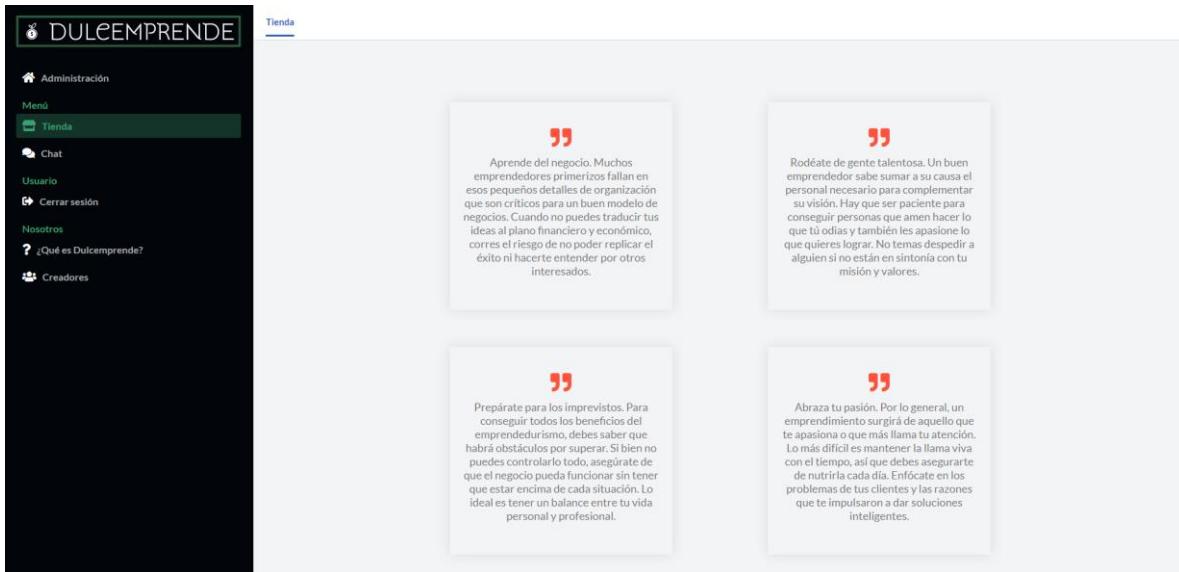


Imagen 76. Parte principal de la tienda tercera parte.

## Pantalla de la visualización de los productos

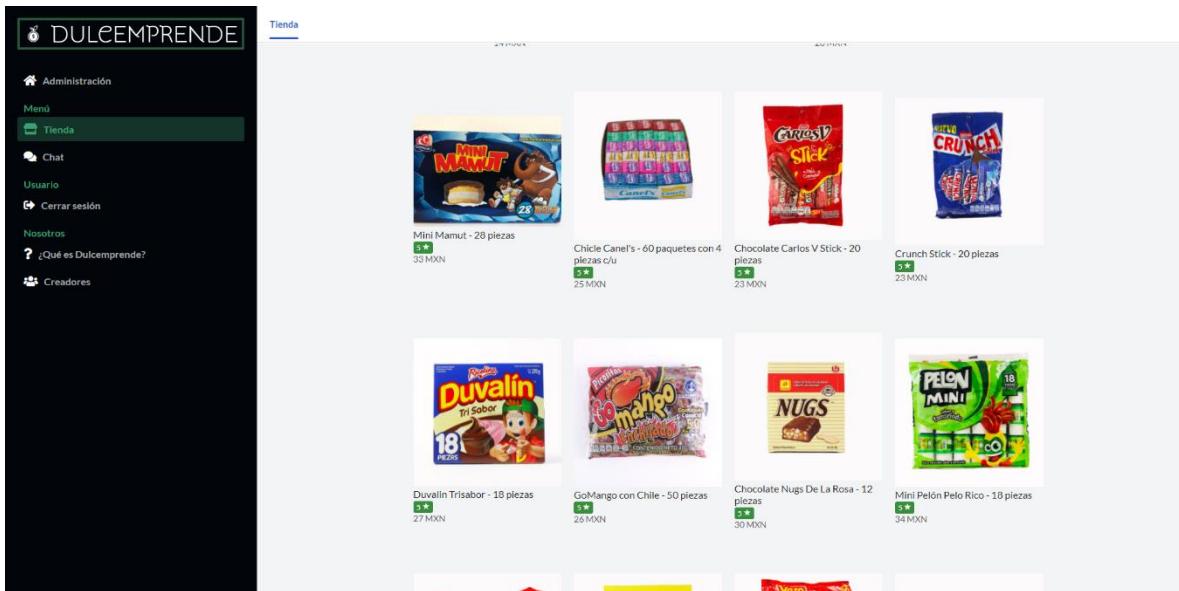


Imagen 77. visualización de los productos.

Como se ve en la imagen 77, dependiendo la categoría se mostrarán todos los productos pertenecientes a esa categoría.

## Pantalla de la información de un producto

The screenshot shows a website interface for 'DULCEMPRENDE'. On the left, a dark sidebar menu includes 'Administración', 'Menú', 'Tienda' (selected), 'Chat', 'Usuario', 'Cerrar sesión', 'Nosotros', '¿Qué es Dulcemprende?', and 'Creadores'. The main content area has a header with 'DULCEMPRENDE', 'Más vendidos', 'Ofertas', and user links. A large image of a 'MINI MAMUT' product box is displayed, showing a cartoon mammoth and the number '28'. Below the box are three smaller images of the product. The product details are listed: 'Productos / Mini Mamut - 28 piezas', 'Mini Mamut - 28 piezas', '33MXN', and a yellow 'Añadir al carrito' button. A 'Detalles del producto' section follows, containing the text '32 piezas. Galleta de cobertura sabor chocolate y malvavisco.' At the bottom, there's a 'Checa el repositorio!' link, social media icons, and a footer with 'DULCEMPRENDE', 'Otros Links', and copyright information.

Imagen 78. Información del producto.

En la imagen 78 podemos ver la página que se muestra cuando el usuario hace clic en un producto.

## Pantalla del carrito

The screenshot shows the same website interface as the previous one, but the main content area now displays a shopping cart. The cart summary table includes:

Producto	Cantidad	Total
Mini Mamut - 28 piezas Precio: 33MXN Eliminar	1	33
Subtotal		33
Envío		99MXN
Total		132

A green 'Pagar' button is at the bottom. The footer contains the same links and information as the previous screenshot.

Imagen 79. Apartado del carrito.

En la imagen 79 vemos la página del carrito del cliente.

## Pantalla de la página de pago

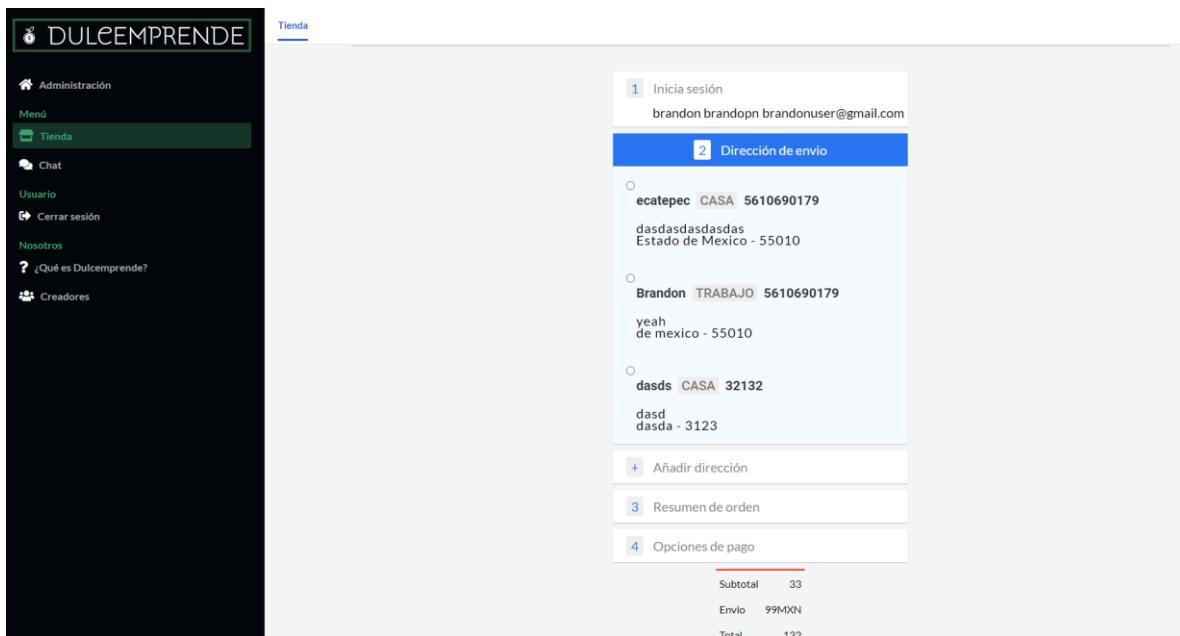


Imagen 80. Parte del pago

En la imagen 80 se ve el apartado para pagar del cliente, donde debe llenar y confirmar una serie de pasos antes de realizar el pago de su mercancía,

## Pantalla de las órdenes del cliente

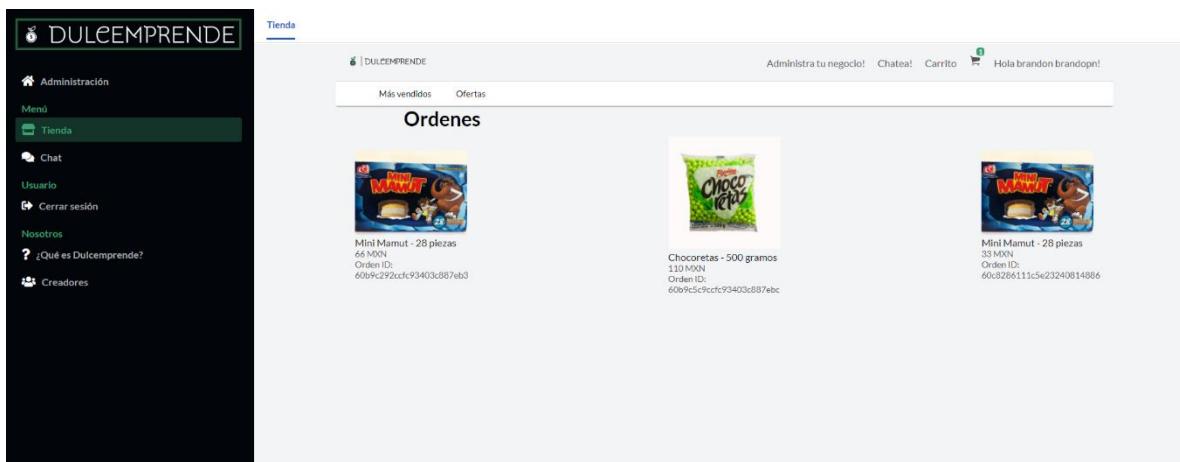


Imagen 81. Página con las órdenes del cliente

## Pantalla de la información de una orden del cliente

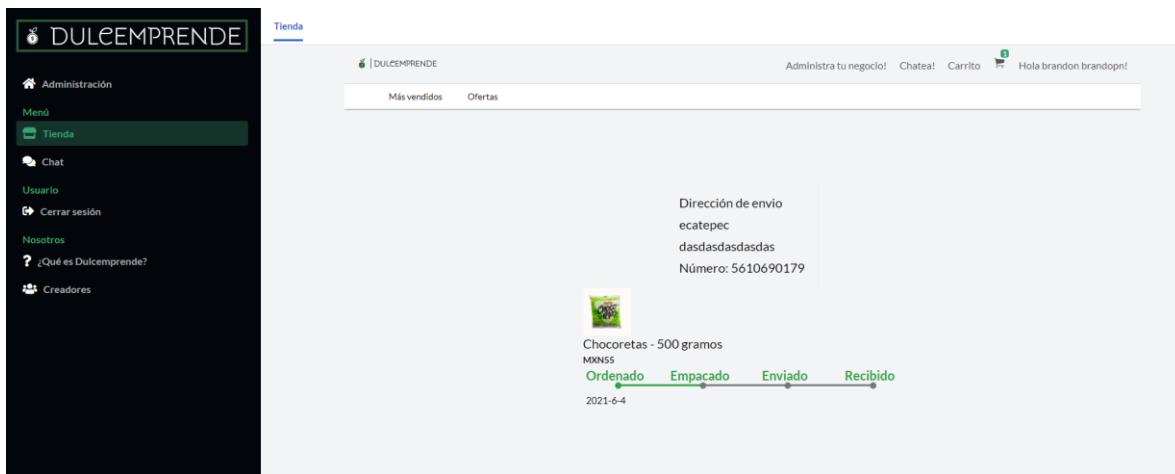


Imagen 82. Detalles de orden

En esta pantalla el cliente puede ver el status de una de sus órdenes en específico.

## Capítulo 6

### Plan de pruebas Dulcemprende

#### Propósito

El principal propósito de la evaluación es encontrar errores y defectos que puedan existir en el uso del sistema para así poder corregirlos a tiempo antes de la entrega. Verificar que los distintos módulos funcionen y se hagan las correctas validaciones para que no se ingresen datos que no son necesarios. Además, se quiere comprobar que el sistema cumple con los requerimientos establecidos y que tiene un rendimiento adecuado. Algo importante también es verificar el rol de los usuarios para que así no puedan hacer modificaciones indebidas.

#### Objetivo

El objetivo principal de realizar este plan de pruebas es realizar actividades para encontrar errores del sistema que puedan afectar la experiencia del usuario, así como encontrar defectos de nuestro sistema.

#### Herramientas

Como parte de las herramientas para las pruebas se usará postman, el cual es una muy buena herramienta para probar APIs, es gratuita y se puede obtener como extensión de google chrome. De igual forma usaremos gitlab, ya que permite tener una administración del proyecto, así como tener un seguimiento en los errores.

#### Entregables

Por cada prueba que se realice deberá anexarse una captura de pantalla o las necesarias para mostrar el resultado de la prueba independientemente si es favorable o no, además de una pequeña descripción puntuizando el estatus que dicha prueba arrojó.

#### Software

Nombre	Versión	Para que se usa
Postman	v8.5.0	Realizar pruebas con APIs
Gitlab	v13.12.1	Para control de versiones y seguimiento de errores.

#### Hardware

Recurso	Cantidad	Descripción
Disco duro	B	2 TB
Memoria RAM	2	8 GB
CPU	1	Intel i5-10400

### **Características a ser probadas**

- Correcto inicio de sesión
- Permitir al usuario navegar por el sitio para elegir los productos a comprar
- Cesta de productos almacene y actualice correctamente los elegidos por el usuario
- Se realice el pago correctamente una vez confirmado el pedido
- Agregar productos a la base de datos del sistema
- Permitir al administrador agregar categorías para clasificar los productos, así como editar el estatus del pedido del cliente
- Qué la parte del usuario sea responsiva.

### **Características a no ser probadas**

- Emisión de factura al realizar la compra: Esta característica no será probada puesto que la emisión no depende de nosotros sino del método de pago que se utilice.

### **Casos de prueba**

<b>ID caso de prueba: CP01</b>	<b>Autor del caso de prueba: Brandon Meza</b>		
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 05/06/21</b>		
<b>Versión 1.0</b>	<b>Fecha de ejecución: 05/06/21</b>		
<b>Flujo de pasos de la prueba:</b>			
No.	Descripción	Resultado esperado	Resultado obtenido
1	Ingreso de los datos del usuario al inicio de sesión	El sistema muestra el panel de administración del usuario.	Panel de administración mostrado correctamente.
2	Los datos ingresados por el usuario son erróneos	El sistema no realiza ninguna acción	El sistema no realizó ninguna acción.
<b>Decisión de aprobación del caso de prueba:</b> Aprobó: x Falló:			

<b>ID caso de prueba: CP02</b>	<b>Autor del caso de prueba: Daniela Briones</b>		
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 05/06/21</b>		
<b>Versión 1.0</b>	<b>Fecha de ejecución: 05/06/21</b>		
<b>Flujo de pasos de la prueba:</b>			
No.	Descripción	Resultado esperado	Resultado obtenido
1	Ingreso de los datos del usuario al registrarse	El sistema muestra el panel de administración del usuario.	Panel de administración mostrado correctamente.
2	El usuario ingresado ya está registrado.	El sistema muestra un mensaje diciendo que el usuario ya está registrado.	El sistema muestra el mensaje
<b>Decisión de aprobación del caso de prueba:</b> Aprobó: x Falló:			

<b>ID caso de prueba: CP03</b>	<b>Autor del caso de prueba: Diego Torres</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 05/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 05/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
1	Visualización de los productos del sistema	El sistema muestra una página con la vista de los productos disponibles	Se muestra la página con los productos disponibles
2	Visualización de la lista de categorías de productos	El sistema muestra un menú con la lista de categorías disponibles.	El sistema mostró el menú con la lista de categorías disponibles.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP04</b>	<b>Autor del caso de prueba: Brandon Meza</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 05/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 05/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
1	Visualización de las características de un producto	El sistema muestra una página con las características del producto seleccionado	Se mostró la página con las características del producto seleccionado.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP05</b>	<b>Autor del caso de prueba: Brandon Meza</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 06/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 06/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
1	Agregar al carrito un producto	Que el sistema agregue correctamente un producto al carrito del usuario	El sistema agregó de forma correcta los productos seleccionados por el usuario a su carrito.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP06</b>	<b>Autor del caso de prueba: Daniela Briones</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 06/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 06/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
-----	-------------	--------------------	--------------------

1	Ingreso de direcciones del usuario	Que el usuario ingrese sus direcciones y se puedan ver al momento de hacer una compra.	El usuario logró ver sus direcciones al momento de hacer una compra.
---	------------------------------------	--	--

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP07</b>	<b>Autor del caso de prueba: Diego Torres</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 06/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 06/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
1	Realización del pago por medio de PayPal.	Que se le muestre al usuario la pantalla de pago de paypal y al pagar se le redireccione a una página con su orden.	Se logró ver la pantalla del pago y se hizo un redireccionamiento correcto a las órdenes del usuario.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP08</b>	<b>Autor del caso de prueba: Brandon Meza</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 06/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 06/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
1	Estadísticas del usuario	Las inversiones, última inversión, la cantidad vendida y las ganancias del usuario se muestran en el apartado de administración.	Se lograron ver las estadísticas del usuario de forma correcta.
2	Actualización de las estadísticas del usuario.	Las estadísticas se van actualizando, dependiendo de las acciones del usuario (si este hace una inversión o vende su producto)	Se actualizaron las estadísticas del usuario de forma satisfactoria.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP09</b>	<b>Autor del caso de prueba: Diego Torres</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 05/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 05/06/21</b>

**Flujo de pasos de la prueba:**

No.	Descripción	Resultado esperado	Resultado obtenido
1	Inserción de datos del producto por parte del administrador a la base de datos	El sistema muestra confirmación de los datos	Se logró insertar nuevos productos y el usuario las logró ver en la página del usuario.
2	Campo vacío en el formulario de los productos	Se muestra un mensaje indicando que olvido llenar un dato.	Se mostró el mensaje al olvidar colocar un dato

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP010</b>	<b>Autor del caso de prueba: Brandon Meza</b>		
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 06/06/21</b>		
<b>Versión 1.0</b>	<b>Fecha de ejecución: 06/06/21</b>		
<b>Flujo de pasos de la prueba:</b>			
No.	Descripción	Resultado esperado	Resultado obtenido
1	Inserción de nuevas categorías a la base de datos	Se ingresa una nueva categoría a la base de datos y se muestra en la página del usuario	Se logró insertar nuevas categorías y el usuario las logró ver en la página del usuario.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP011</b>	<b>Autor del caso de prueba: Daniela Briones</b>		
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 06/06/21</b>		
<b>Versión 1.0</b>	<b>Fecha de ejecución: 06/06/21</b>		
<b>Flujo de pasos de la prueba:</b>			
No.	Descripción	Resultado esperado	Resultado obtenido
1	Visualización del estatus del pedido del usuario	Se ve el status del pedido que realiza un cliente	Se logró visualizar el status del pedido
2	actualización del estatus del cliente	Se logra editar el status del cliente y el cliente ve reflejado este cambio en sus órdenes.	Se logró actualizar el status correctamente y el cliente vió reflejado este cambio.

**Decisión de aprobación del caso de prueba:** Aprobó: x Falló:

<b>ID caso de prueba: CP012</b>	<b>Autor del caso de prueba: Brandon Meza</b>
<b>Versión del caso de prueba: 1</b>	<b>Fecha de creación: 05/06/21</b>
<b>Versión 1.0</b>	<b>Fecha de ejecución: 05/06/21</b>
<b>Flujo de pasos de la prueba:</b>	

No.	Descripción	Resultado esperado	Resultado obtenido
1	Cierre de sesión del usuario	El sistema muestra confirmación del cierre de sesión	El sistema vuelve a la página de inicio de sesión
<b>Decisión de aprobación del caso de prueba:</b> Aprobó: x Falló:			

### Criterios para el lanzamiento

- No existen errores de gravedad
- Los casos de prueba aprobados son mayores al 80%

### Resultados

#### CP01

##### 1. Datos correctos

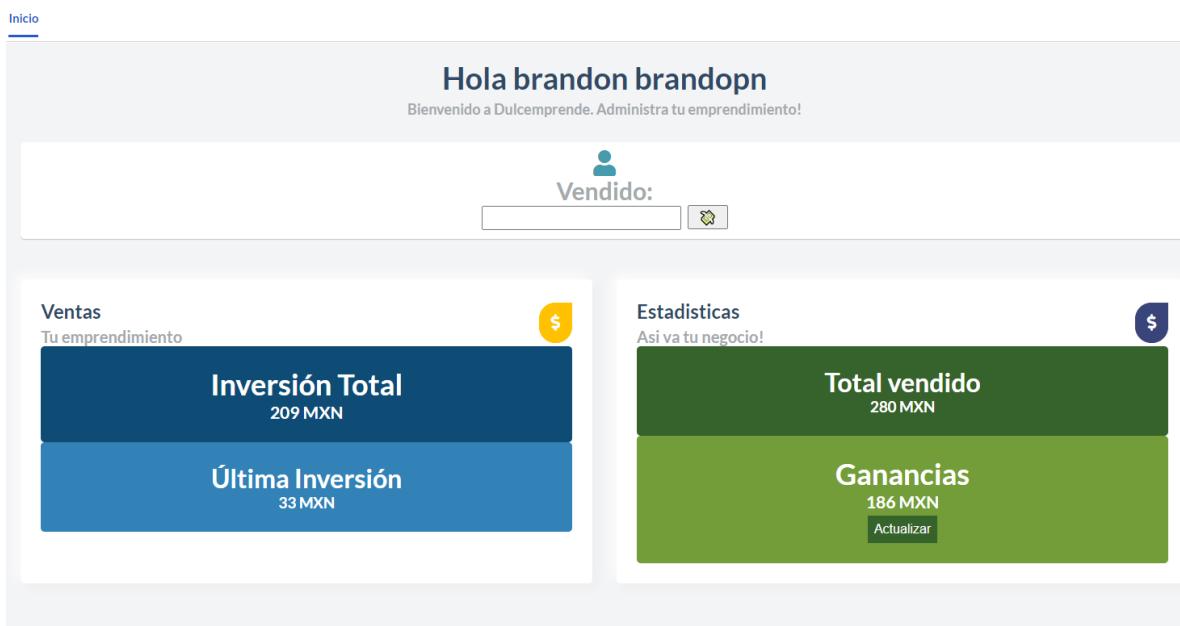


Imagen 83. CP01 No. 1 exitoso

##### 2. Datos incorrectos



Imagen 84. CP01 No. 2 exitoso

## CP02

### 1. Ingreso de datos

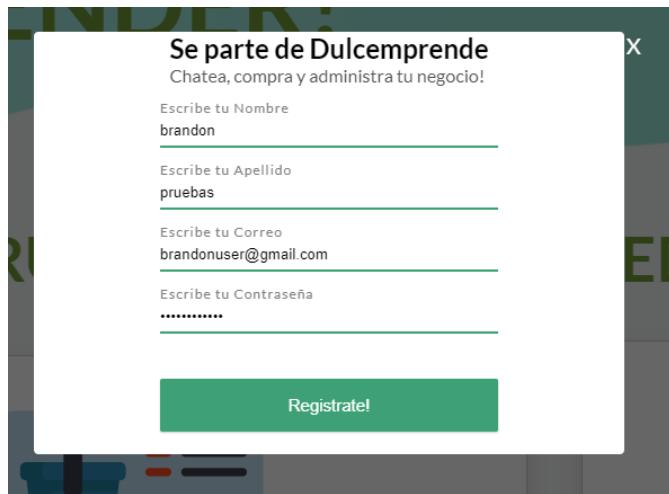


Imagen 85. CP02 No. 1 exitoso

### 2. Usuario registrado

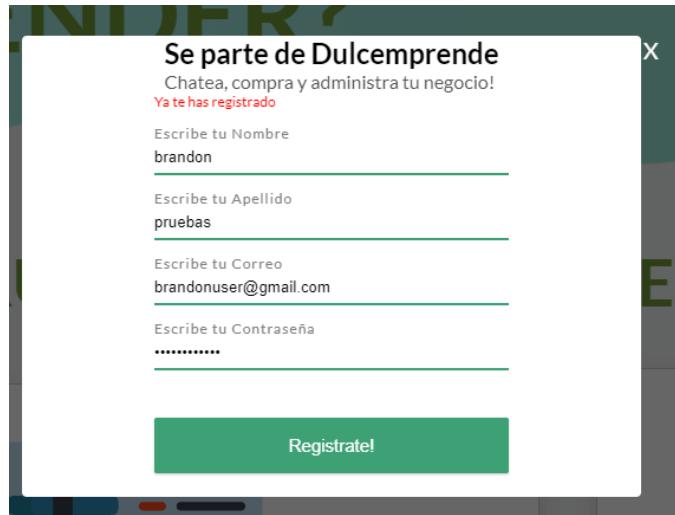


Imagen 86. CP02 No. 2 exitoso

## CP03

### 1. Visualización de los productos

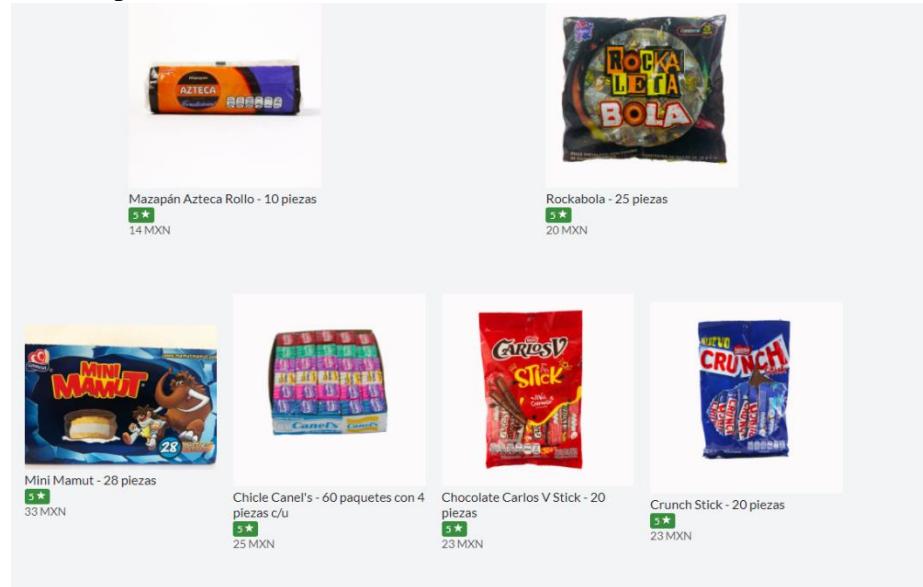


Imagen 87. CP03 No.1 exitoso

### 2. Visualización de la lista de categorías

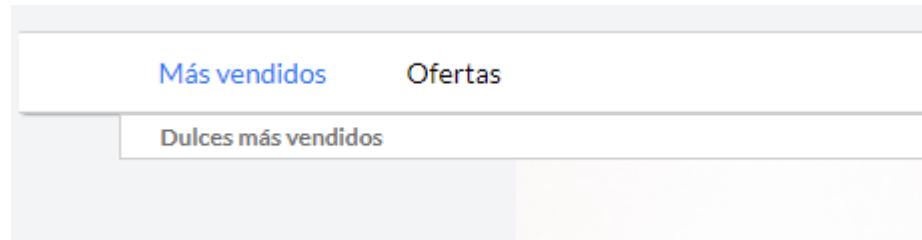


Imagen 88. CP03 No. 2 exitoso

## CP04

1. Visualización de las características de un producto

A product page for 'Mazapán Azteca Rollo - 10 piezas'. The page has a white background. On the left, there is a large image of the product: a roll of mazapan wrapped in orange and purple foil. Below it are three smaller images: the full roll, a single piece of mazapan, and a close-up of the mazapan itself. To the right of the images, the product name 'Mazapán Azteca Rollo - 10 piezas' is displayed in bold black text. Above the name, the price '14MXN' is shown. Below the name is a yellow button with the text 'Añadir al carrito' (Add to cart). Underneath the button, there is a link 'Detalles del producto' with a small info icon. At the bottom, a short description reads '10 piezas. Dulce de cacahuate estilo mazapan.'

Imagen 89. CP04 No.1 exitoso

## CP05

1. Agregar producto al carrito

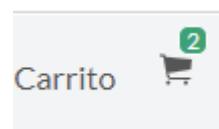


Imagen 90. CP05 exitoso

## CP06

1. Ingreso de direcciones del usuario

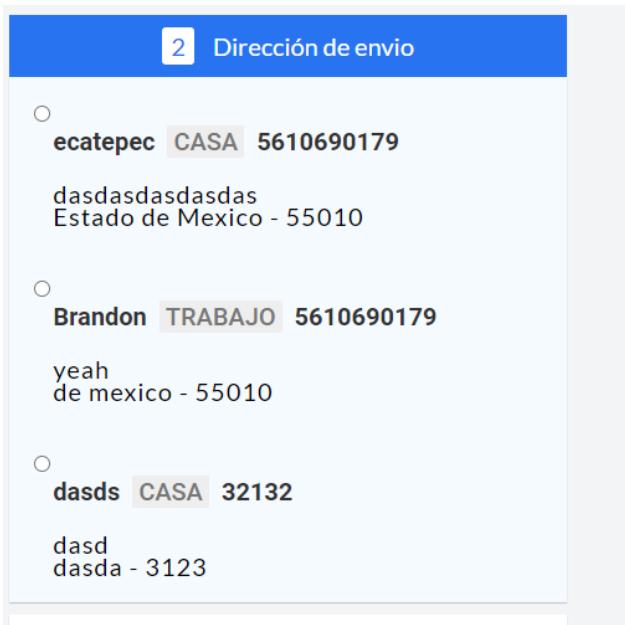


Imagen 91. CP06 exitoso

## CP07

### 1. Pago por medio de Paypal

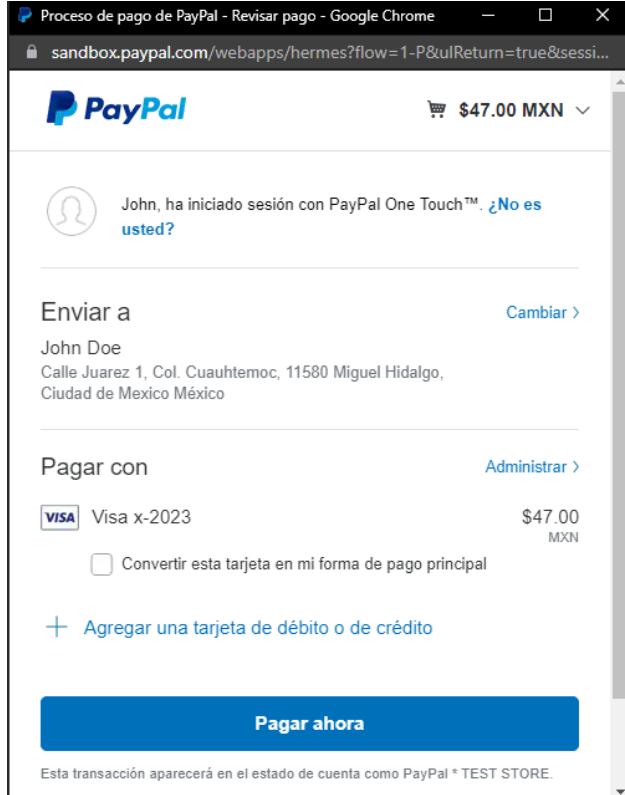


Imagen 92. CP07 exitoso

## CP08

### 1. Estadísticas del usuario



Imagen 93. CP08 No.1 exitoso

### 2. actualización de estadísticas.



Imagen 94. CP08 No.2 exitoso

## CP09

### 1. Inserción de datos del producto por parte del admin

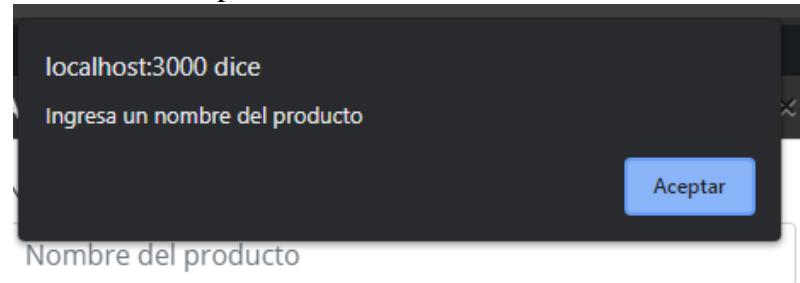
**Añadir nuevo producto**

Nombre	Producot
Cantidad	50
Precio	50
Descripción	este es un nuevo producto
Estado	Dulces en oferta
Imagen	logo.svg Seleccionar archivo logo.svg
<b>Listo</b>	

Producot      50      Dulces en oferta      i x

*Imagen 95. CP09 No.1 exitoso*

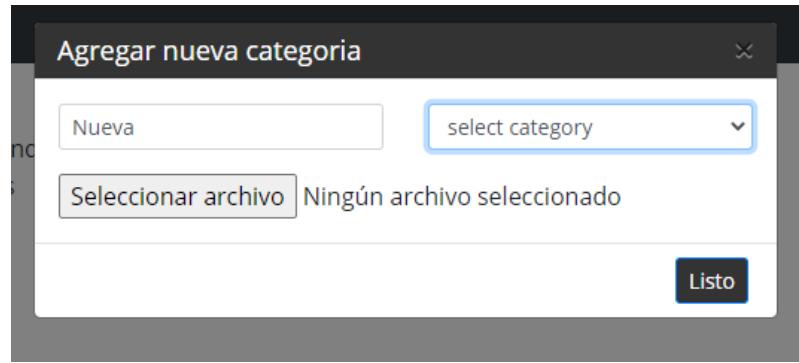
## 2. Campo vacío en el formulario de productos



*Imagen 96. CP09 No.2 exitoso*

## CP010

### 1. Inserción de nuevas categorías



## Categoría

- >   Más vendidos
- >   Ofertas
- Nueva

*Imagen 97. CP010 exitoso*

## CP011

### 1. Visualización del status del pedido

60b9c292ccfc93403c887eb3			
Productos Mini Mamut - 28 Piezas	Costo total 66	Tipo de pago Paypal	
<p>ordenado      empacado      enviado      recibido</p> <p>2021-6-4</p> <p>Status <input type="button" value="▼"/></p> <p><input type="button" value="Confirmar"/></p>			

*Imagen 98. CP011 No.1 exitoso*

## 2. Actualización del status del pedido



Imagen 99. CP011 No.2 exitoso

## COP012

### 1. Cierre de sesión



Imagen 100. CP012 exitoso

## **Conclusión**

Por medio de este proyecto pudimos comprobar lo importante que son los sistemas en línea hoy en día, con la situación mundial del COVID-19 el e-commerce tuvo un auge considerable dado que al implementar restricciones para las actividades comerciales y demás, esta forma de hacer negocios resultó la más cómoda para la mayoría de las personas afectadas. El e-commerce cuenta con destacables ventajas a comparación del comercio común pues se pueden realizar más ventas a diferentes partes del país e incluso fuera del mismo, disponibilidad de horario para el cliente e incluso la reducción de algunos costos por mencionar algunas.

El objetivo general de este proyecto era desarrollar un sistema de dulcería en línea para aquellas personas que deseen emprender su propio negocio. El sistema funciona como tienda en línea y a su vez como panel de control para que los usuarios administren ciertos factores del negocio como las ganancias. Tras varios meses invertidos en el desarrollo de esta idea, el proyecto se ha concluido. Con base en las pruebas realizadas podemos afirmar que el sistema cumple con los objetivos que se plantearon al inicio de su desarrollo.

## Referencias

- [1] H, Marianna, “Panorama de emprendimiento en México”, 2018. [En línea]. Disponible: <https://quickbooks.intuit.com/mx/recursos/comienza-tu-negocio/panorama-emprendimiento/>
- [2] Anónimo, “React”, 2021. [En línea]. Disponible: <https://es.reactjs.org/>
- [3] Anónimo, “¿Qué son las aplicaciones web?”, 2019. [En línea]. Disponible: <https://edu.gcfglobal.org/es/informatica-basica/que-son-las-aplicaciones-web/1/>
- [4] U, Erika, “Haz negocio con una dulcería boutique”, 2019. [En línea]. Disponible: <https://www.entrepreneur.com/article/269203>
- [5] saraclip, “Requerimientos de un proyecto”, 2017. [En línea]. Disponible: <https://www.saraclip.com/requerimientos-de-un-proyecto/>
- [6] “Importancia de la arquitectura web”, *Tokio School*, 2020. [En línea]. Disponible: <https://www.tokioschool.com/noticias/importancia-arquitectura-web/>
- [7] G, Maida, “Metodologías de desarrollo de software”, 2015. [En línea]. Disponible: <https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf>

## Anexos

### Cronograma

Nombre del alumno(a): Briones Tapia Daniela

Título del proyecto: Sistema de dulcería

ACTIVIDAD	MAR	ABR	MAY	JUN
Análisis de requisitos				
Definición de objetivos generales y particulares				
Planeación de actividades del negocio				
Elaboración del diagrama de contexto				
Elaboración del diagrama de flujo de datos lógico				
Elaboración del diagrama de clases				
Elaboración del diagrama de clasificación y ensamble				
Planeación y diseño de página web				
Creación de página web				
Realización de pruebas (Hacer pedido)				
Entrega proyecto final				

Nombre del alumno(a): Meza Vargas Brandon David

Título del proyecto: Sistema de dulcería

ACTIVIDAD	MAR	ABR	MAY	JUN
Análisis de requisitos				
Definición de objetivos generales y particulares				
Planeación de actividades del negocio				
Elaboración del diagrama de contexto				
Elaboración del diagrama de flujo de datos lógico				
Elaboración del diagrama de clases				
Elaboración del diagrama de clasificación y ensamble				
Planeación y diseño de página web				
Creación de página web				
Realización de pruebas (Hacer pedido)				
Entrega proyecto final				

Nombre del alumno(a): Torres Jiménez Diego Antonio

Título del proyecto: Sistema de dulcería

<b>ACTIVIDAD</b>	<b>MAR</b>	<b>ABR</b>	<b>MAY</b>	<b>JUN</b>
Análisis de requisitos				
Definición de objetivos generales y particulares				
Planeación de actividades del negocio				
Elaboración del diagrama de contexto				
Elaboración del diagrama de flujo de datos lógico				
Elaboración del diagrama de clases				
Elaboración del diagrama de clasificación y ensamble				
Planeación y diseño de página web				
Creación de página web				
Realización de pruebas (Hacer pedido)				
Entrega proyecto final				

## Entregables

### Poster



Imagen 101. Poster dulcemprende

## Tríptico



The graphic design for the Triptico Dulcemprende consists of three panels. The left panel features a yellow and orange decorative border at the top and bottom. It contains a section titled "OBJETIVO" with a descriptive text about developing an online system for candy stores. Below this is an icon of a shopping cart with a computer mouse. The middle panel has a decorative border at the top and bottom. It contains a section titled "FUNCIONES" with a bulleted list of three functions: streamlining purchases, allowing users to place orders without time restrictions, and offering a service that does not limit users to buying a fixed quantity of products. To the right of this list is a graphic of a computer monitor showing a dollar sign and a clock. The right panel has a decorative border at the top and bottom. It contains a section titled "JUSTIFICACIÓN" with a text explaining the context of the pandemic and how the platform aims to support small businesses by providing them with a digital storefront. Below this text is another graphic of a computer monitor and a clock.

**OBJETIVO**

Desarrollar un sistema de dulcería en línea para aquellas personas que deseen realizar un emprendimiento que funcione como tienda y panel de control para que los usuarios administren su negocio.

**FUNCIONES**

- Agilizar las compras de nuestros clientes por medio del e-commerce
- Permitir a los usuarios realizar pedidos sin restricción de horario
- Ofrecer un servicio que no limite al usuario a cumplir una cantidad de productos para poder realizar la compra

**JUSTIFICACIÓN**

A mediados de marzo de 2020 comenzó el confinamiento debido al COVID-19, lo anterior orilló a que negocios esenciales que cumplen con las necesidades básicas de la población, permanecieran abiertos y los que no se consideran esenciales permanecieran cerrados. Estos hechos conducen a un nuevo entendimiento del concepto de ventas y el mundo digital, es tiempo de que los negocios se empiecen a preocupar por su digitalización.

Imagen 102. Tríptico Dulcemprende.



Imagen 103. Tríptico Dulcemprende.

# **MANUAL DE USUARIO: DULCEMPRENDE**



**DULCEMPRENDE**



# Contenido

<b>I. Introducción .....</b>	<b>3</b>
1. Objetivo .....	3
2. Requerimientos .....	3
<b>II. Opciones Del Sistema .....</b>	<b>4</b>
1. Registro .....	4
2. Iniciar sesión .....	4
3. Operaciones Básicas .....	4
3.1. Tienda .....	4
3.2. Administración .....	5
3.3. Chat .....	5

## I. Introducción

### 1. OBJETIVO DE LA APLICACIÓN

Desarrollar un sistema de dulcería en línea para aquellas personas que deseen realizar un emprendimiento que funcione como tienda y panel de control para que los usuarios administren su negocio.

### 2. REQUERIMIENTOS

#### Software

Nombre	Versión	Para que se usa
Postman	v8.5.0	Realizar pruebas con APIs
Gitlab	v13.12.1	Para control de versiones y seguimiento de errores.

#### Hardware

Recurso	Cantidad	Descripción
Disco duro	1	2 TB
Memoria RAM	2	8 GB
CPU	1	Intel i5-10400

## II. Opciones Del Sistema

El presente Manual está organizado de acuerdo a la secuencia de ingreso a las pantallas del sistema de la siguiente manera:

1. Registro de Usuario
2. Ingreso al Sistema
3. Operaciones Básicas
  - 3.1 Tienda
  - 3.2 Administración
  - 3.3 Chat

## Módulo de Registro

Este módulo permite la creación de una nueva cuenta de usuario.

## Módulo de Iniciar sesión

Este módulo permite el ingreso de los usuarios al sistema, con el fin de mostrar sus datos y crear un entorno más personalizado.

## Módulo de Tienda

Este módulo los compradores potenciales podrán revisar la variedad de productos que hay registrados en la aplicación, donde podrán comprar la cantidad que necesiten.

## Módulo de Administración

En este módulo el usuario dueño de una dulcería podrá administrar sus productos desde el costo hasta la disponibilidad, así como añadir nuevos productos, mostrando imágenes y sus características. Adicionalmente se podrá ver el apartado de cliente. Para el caso de los compradores, verán una estadística con la inversión y las ventas que han hecho.

## Módulo de Chat

Este módulo permite la comunicación entre diferentes vendedores, esto con el fin de crear una comunidad donde entre ellos mismos, den a conocer información sobre productos o estrategias para mejorar su servicio.

## Modulo 1: Registro

Al ingresar al sistema lo primero que veremos será la pantalla de Bienvenida, donde se muestran algunas de las ventajas que ofrece nuestra aplicación



}

Y en el panel lateral podrá dar click a “Registrarse!” para acceder al módulo. Posteriormente solo debe llenar los campos con sus datos y ya quedara dado de alta en el sistema.

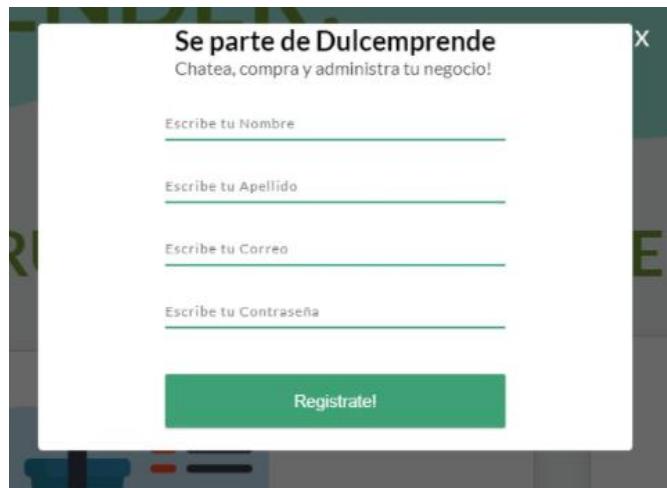


Imagen 105, Pantalla de Registro

## Modulo 2: Inicio de Sesión

En esta pantalla el usuario debe digitar el Correo y Clave con la que se registró y presionar sobre el botón “Iniciar Sesión” tal como se muestra en la figura siguiente, los datos que se ingresan se los proporciona al momento de registrarse, ¡si usted no está registrado debe volver al menú de opciones y seleccionar “Regístrate!” con el fin de otorgale un usuario y clave para que pueda ingresar al sistema.

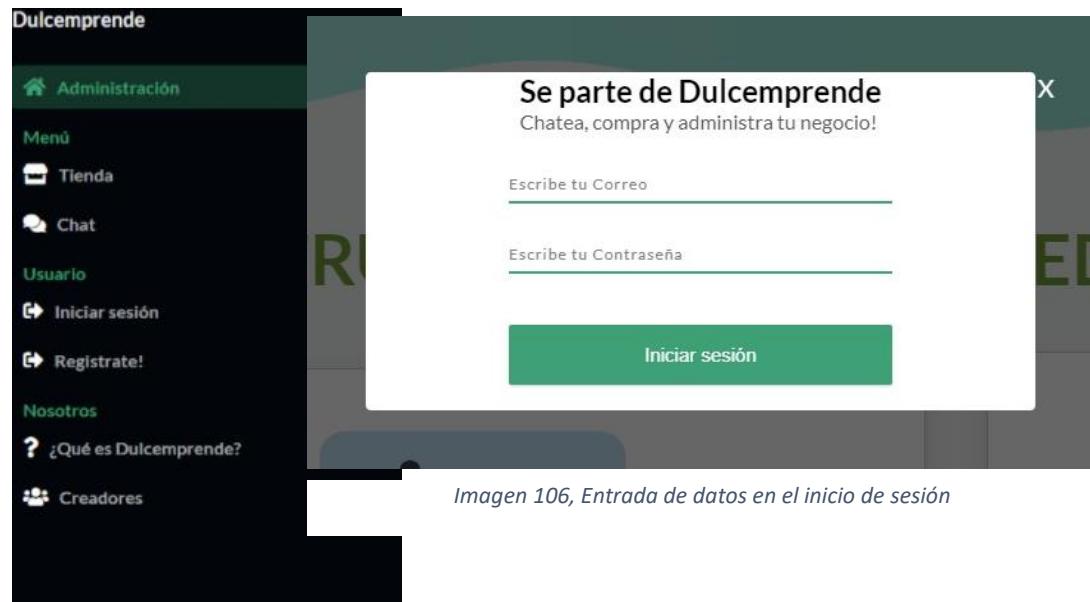


Imagen 106, Entrada de datos en el inicio de sesión

Imagen 107. Barra Lateral

## Modulo 3: Tienda

Una vez iniciada la sesión, el usuario puede navegar por el sitio libremente. Donde podrá acceder al módulo con el botón “Tienda” en la barra lateral.



Imagen 108. Tienda Inicio

Una vez ahí observamos 3 pestañas, las cuales guiaran al usuario a diferentes productos: por ejemplo, para el caso de para empezar, mostrara una sección de lo más recomendable para empezar un negocio. Lo más vendido, como su nombre lo indica, mostrará los dulces más vendidos en la aplicación. Y dentro de Ofertas hallará las promociones y descuentos para nuestros usuarios.

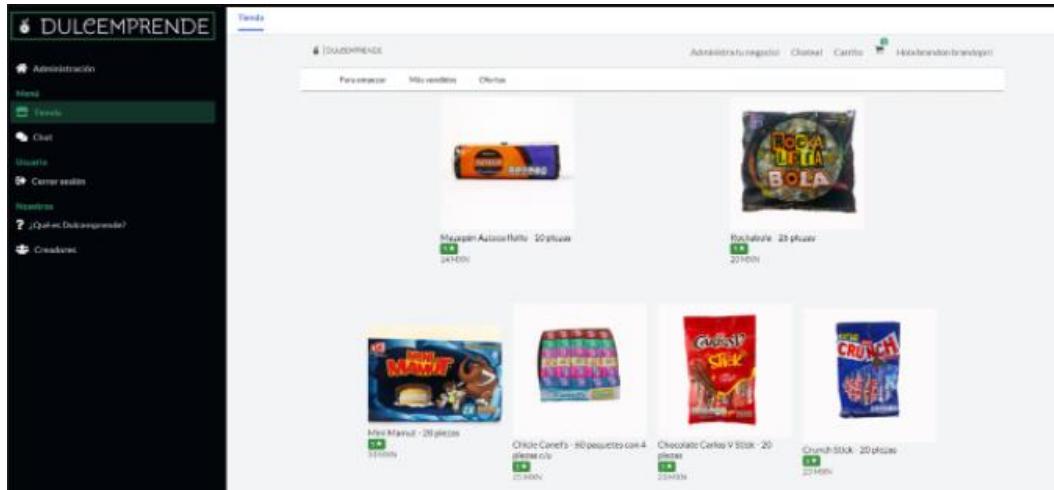


Imagen 109. Artículos en la Tienda

Ya una vez seleccionada la categoría podemos empezar a observar los diversos artículos que se venden, su precio y algunas otras características extra sobre el producto. Si hacemos click sobre uno. Podremos ver mas a detalle sus características, así como tenemos la opción de seleccionar varios y agregarlos al carrito.

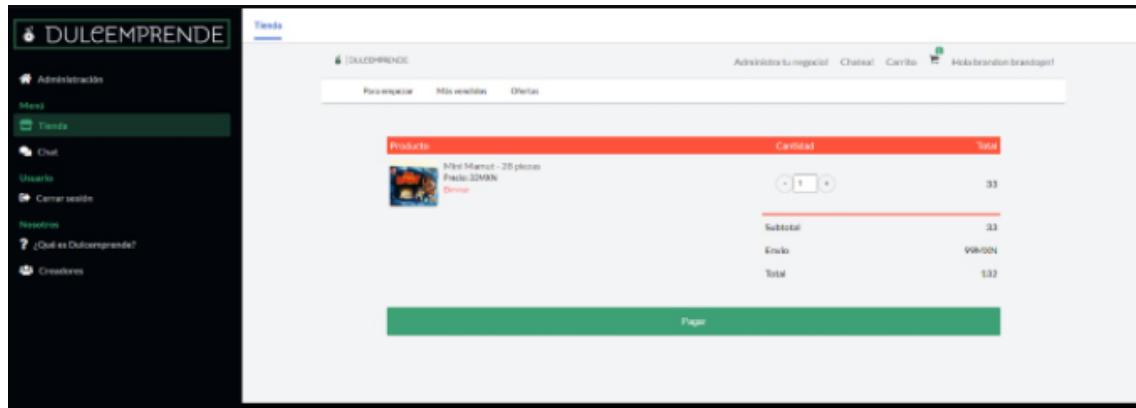


Imagen 110. Vista del producto

Una vez ya estén todos sus artículos deseados en el carrito, procedemos a ir para confirmar los datos de la compra. Se pedirán datos como dirección, su correo, y opciones de pago. Para al final concluir su compra.

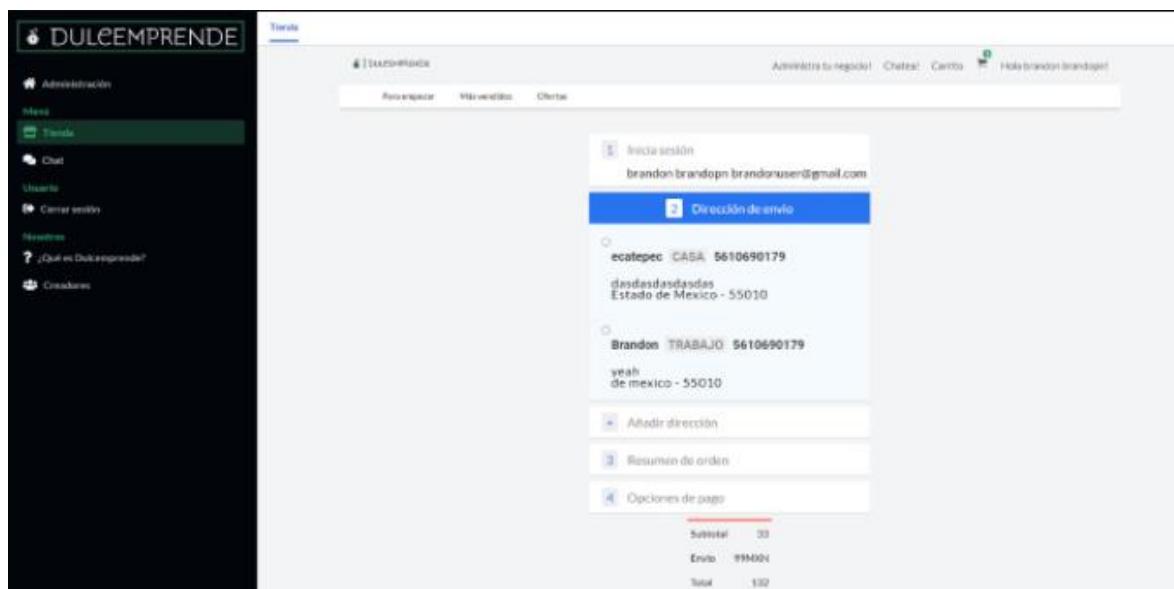


Imagen 111. Vista de pago

Dentro de las opciones de pago, se manejan dos diferentes: Tarjeta de débito o crédito o PayPal. Con el fin de hacer una transferencia segura y a la vez dar seguridad a nuestros clientes a la hora de pagar.

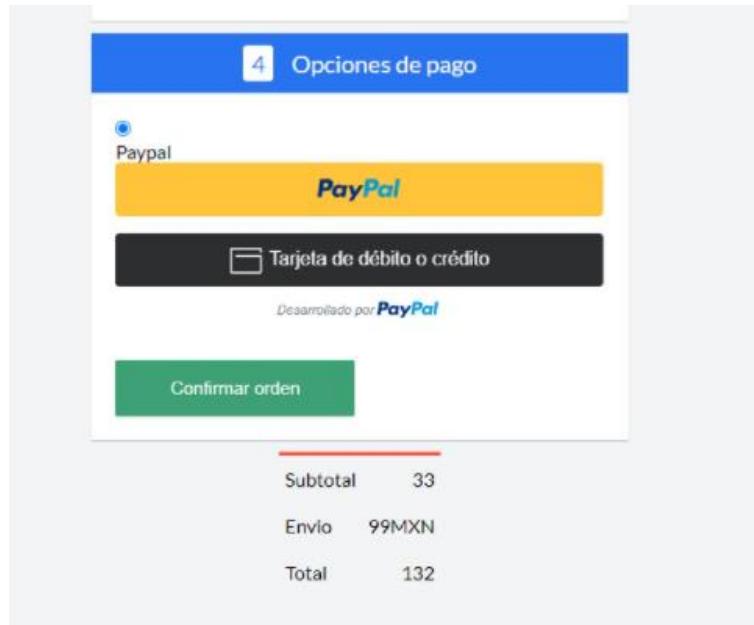


Imagen 112. Métodos de Pago

Para finalizar la compra ya solo se muestra un seguimiento del paquete, junto con la dirección de destino y un número de referencia para aclaraciones.



Imagen 113, Pantalla de seguimiento

Ya, por último, si lo que desea es consultar compras previas, solo pase su cursor sobre su nombre de usuario en la parte superior derecha y saldrá un menú, con la opción órdenes. Dentro veremos todas las ordenes que se han realizado dentro de la cuenta.

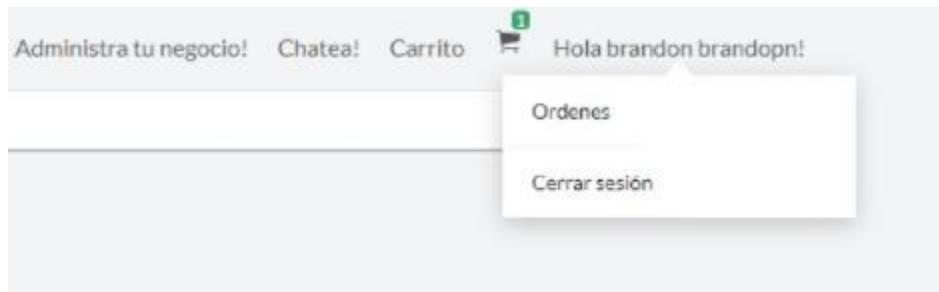


Imagen 14. Menú de usuario

Imagen 15 muestra la sección 'Ordenes' en la aplicación. Se presentan seis ordenes recientes con imágenes de los productos:

- Chocolate Bocadín - 50 piezas**: 147 MXN. Orden ID: 60b7c2e5bed5a51924a544f1
- Chocolate M&M Cacahuate - 6 piezas**: 195 MXN. Orden ID: 60b7c720bed5a51924a544fb
- Rellerindo - 65 piezas**: 120 MXN. Orden ID: 60b65bb6a1d10d2530f5fb72
- Pollo Miguelito - 950 gramos**: 45 MXN. Orden ID: 60b679dbdd10d2530f5fb7a
- Mazapán Azteca Rollo - 10 piezas**: 14 MXN. Orden ID: 60b67a11a1d10d2530f5fb82
- Chicle Canel's - 60 paquetes con 4 piezas c/u**: 25 MXN. Orden ID: 60b67da1a1d10d2530f5fb8c

Imagen 15. Ordenes del Usuario

## Modulo 4: Administración

En este módulo, los dueños de empresas podrán administrar su venta de dulces, agregando existencias y comprobando estrategias. Lo primero que aparecerá será el iniciar sesión o el registro dependiendo de si ya estén registrados en el sistema o no.



Imagen 16. Inicio de Sesión

The screenshot shows a registration form titled "Panel de administrador". On the left, there is a sidebar with three menu items: "Categorías", "Productos", and "Ordenes". The main area contains fields for "Nombre" (Name), "Apellido" (Last Name), "Correo" (Email), and "Contraseña" (Password). A blue "Registrarse" (Register) button is at the bottom.

Nombre	Apellido
<input type="text"/>	<input type="text"/>
Correo	
<input type="text"/>	
Contraseña	
<input type="password"/>	
<input type="button" value="Registrarse"/>	

Imagen 117. Registro de Administradores

Una vez iniciada la sesión, lo primero que encontramos es con una serie de menús dentro de los cuales podemos entrar. Para realizar diferentes acciones. Por ejemplo en categoría se puede administrar las categorías editándolas, eliminándolas o agregándolas.

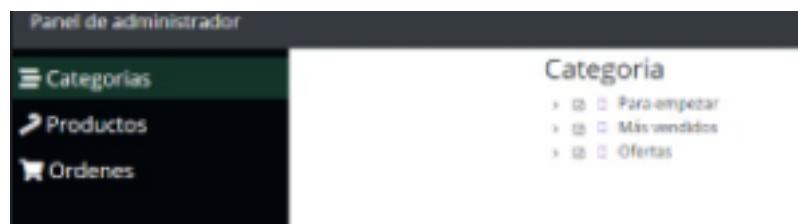


Imagen 118. Menús de Administrador

The screenshot shows a modal window titled "Agregar nueva categoría". It has two input fields: "Nombre de la categoría" (Category name) and "select category" (Select category). Below these is a file selection field: "Seleccionar archivo" (Select file) with the message "Ningún archivo seleccionado" (No file selected). At the bottom right is a blue "Listo" (Ready) button.

Imagen 119. Agregar categoría

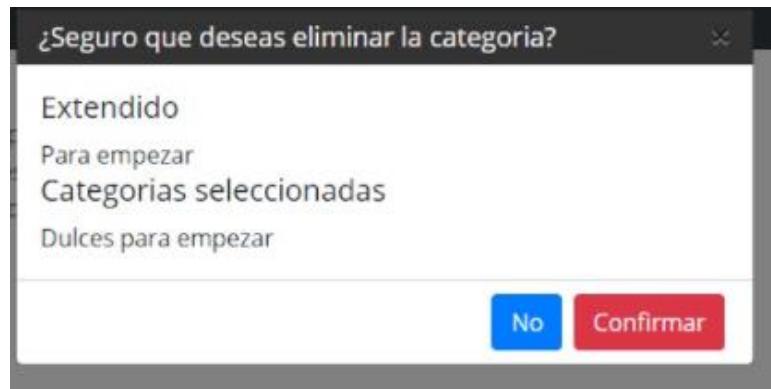


Imagen 120. Eliminar categoría

Imagen 121. Actualizar categoría

Ya, por último, tenemos el menú para que el usuario introduzca más de sus productos o cree existencias.

Productos					Añadir
Nombre	Precio	Cantidad	Categoría	Acciones	
Polo Vlieghe - 900 gramos	45	100	Dulces más vendidos		
M&M Marmal - 25 piezas	33	100	Dulces más vendidos		
Macapán Azteca Rollo - 10 piezas	14	100	Dulces más vendidos		
Chocolate Biscochito - 50 piezas	49	100	Dulces más vendidos		
Mini Gobuleta - 25 piezas	45	100	Dulces más vendidos		
Chicle Cereña - 10 paquetes con 4 piezas c/u	29	100	Dulces más vendidos		
Chocolate Carlos V Tíck - 20 piezas	29	100	Dulces más vendidos		
Flan dulce sabor fresa y chocolate - 50 piezas	45	100	Dulces más vendidos		

Imagen 122. Lista de productos

**Añadir nuevo producto**

Nombre	<input type="text" value="Nombre del producto"/>
Cantidad	<input type="text" value="Cantidad"/>
Precio	<input type="text" value="Precio"/>
Descripción	<input type="text" value="Descripción"/>
select category	<input type="button" value="▼"/>
<input type="button" value="Seleccionar archivo"/>	Ningún archivo seleccionado
<input type="button" value="Listo"/>	

Imagen 123. Formulario Productos

## Modulo 5: Chat

En este apartado el usuario puede conversar con otros usuarios que estén en línea en la plataforma, además de poder consultar sus dudas con otros usuarios usted puede tanto dar como recibir recomendaciones de los productos en venta.

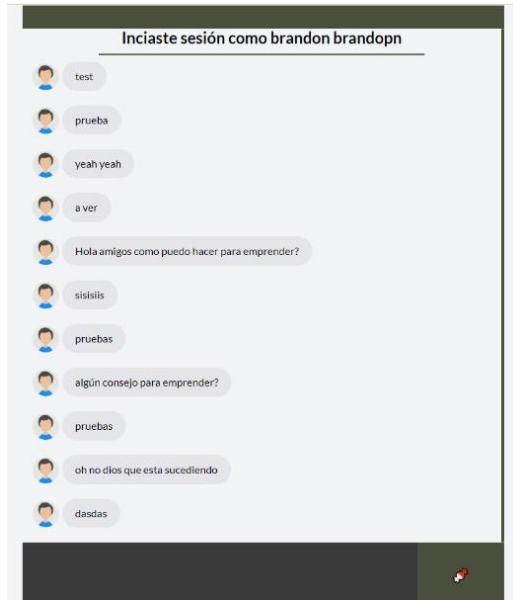


Imagen 124. Ventana de Chat