# PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
# FACULTAD DE INGENIERÍA

## ESCUELA DE SISTEMAS

TEMA

**"ANÁLISIS Y DESARROLLO DE UN PROTOTIPO FUNCIONAL PARA EL CONTROL DE LA GESTIÓN DE BARES Y RESTAURANTES. CASO DE ESTUDIO: HELARTE"**

AUTOR

**CÉSAR ELÍAS ARMENDARIZ RUANO**

**ÍNDICE GENERAL**

CAPÍTULO 1. DEFINICIÓN DEL PROYECTO 7
**1.1. Planteamiento del problema** ..... 7
**1.2. Objetivos** ..... 7
**1.2.1 Objetivo General** 7
**1.2.2 Objetivos Específicos** ..... 7
**1.3. Justificación** ..... 8
**1.4. Alcance** ..... 9
CAPÍTULO 2: MARCO TEÓRICO 9
**2.1 METODOLOGÍAS ÁGILES DE DESARROLLO '*AGILE*'** ..... 9
**2.2. *SCRUM*** ..... 9
**2.2.1 Equipo de *Scrum*** 9
**2.2.2 Artefactos de scrum** 10
**2.3 HERRAMIENTAS** ..... 11
**2.3.1 *Power Designer*** 11
**2.3.2 *Html5*** 11
**2.3.3 *CSS*** 12
**2.3.4 *JavaScript*** 12
**2.3.5 *TypeScript*** 13
**2.3.6 *Node.js*** 13
**2.3.7 *Angular Material*** 13
**2.3.8 *Angular*** 14
**2.3.9 *Express*** 14
**2.3.10 *PWA*** 14
**2.3.11 Paquetes y librerías** 15
**2.3.11.1 *NPM* (*Node Package Manager*)** 15
**2.3.11.2 *Flexbox Grid*** 15
**2.3.11.3 *ngx-auth-firebaseui*** ..... 15
**2.3.11.4 *html2canvas*** 15
**2.3.11.5 *jspdf*** 15
**2.3.12 *Git*** 16
**2.3.13 *GitHub*** 16

**2.3.14 *Cloud Firestore*** 16
**2.3.15 Resumen de uso de herramientas** 16
**2.4 MODELO VISTA CONTROLADOR** ..... 17
**2.5 MODELO VISTA – VISTA MODELO (MVVM)** ..... 17
**2.6. INVENTARIOS** ..... 18
**2.7. *E - Commerce*** ..... 19
CAPÍTULO 3: ANÁLISIS Y DISEÑO 19
**3.1 Introducción** ..... 19
**3.2 Perfiles de usuario** ..... 20
**3.3 Historias de usuario** ..... 23
**3.4 Requerimientos funcionales** ..... 26
**3.5 Requerimientos no funcionales** ..... 27
**3.6 Product Backlog o Lista del Producto** ..... 28
**3.7 Diagrama General de Módulos** ..... 31
**3.8 Diagrama General de casos de uso** ..... 32
**3.8 Modelos** ..... 33
**3.9 Diagramas de actividades** ..... 35
**3.10 Estructura de interfaces** ..... 44
CAPÍTULO 4: DESARROLLO Y CONTROL DE PROCESOS 48
**4.1 Creación de base de datos en *firestore*** ..... 48
**4.2 Desarrollo de las interfaces gráficas *UX* y *UI*** ..... 48
**4.3 Desarrollo de *Sprints*** ..... 48
**4.3.1 Primer *Sprint*** 49
**4.3.2 Segundo *Sprint*** 54
**4.3.3 Tercer *Sprint*** 57
**4.3.4 Cuarto *Sprint*** 67
CAPÍTULO 5: IMPLEMENTACIÓN Y PRUEBAS 71
**5.1 Código y desarrollo** ..... 71
**5.2 Pruebas de la aplicación** ..... 73
CAPÍTULO 6: CONCLUSIONES Y RECOMENDACIONES 78
**6.1 Conclusiones** ..... 78
**6.2 Recomendaciones** ..... 79
DICCIONARIO DE DATOS 80
BIBLIOGRAFÍA 83

# ÍNDICE DE TABLAS

Tabla 1 Resumen de uso de herramientas 10
Tabla 2 Perfiles de usuario 14
Tabla 3 Perfil de usuario: Cajera/o 15
Tabla 4 Perfil de usuario: Administrador/a 16
Tabla 5 Historia de usuario N.º 1 17
Tabla 6 Historia de usuario N.º 2 17
Tabla 7 Historia de usuario N.º 3 17
Tabla 8 Historia de usuario N.º 4 18
Tabla 9 Historia de usuario N.º 5 18
Tabla 10 Historia de usuario N.º 6 18
Tabla 11 Historia de usuario N.º 7 18
Tabla 12 Historia de usuario Nº 8 19
Tabla 13 Requerimientos funcionales 20
Tabla 14 Requerimientos no funcionales 20
Tabla 15 Product Backlog 21
Tabla 16 Product Backlog del primer sprint 42
Tabla 17 Sprint Backlog del módulo 6 Pedido de cliente 43
Tabla 18 Sprint Backlog del Módulo 7 Cotización de pedido 46
Tabla 19 Product Backlog del segundo sprint 48
Tabla 20 Sprint Backlog del Módulo 4 Lista de pedido 49
Tabla 21 Sprint Backlog del Módulo 5 Cobro de pedido 50
Tabla 22 Product Backlog del tercer Sprint 51
Tabla 23 Sprint Backlog del Módulo 1 Control de Inventario 52
Tabla 24 Sprint Backlog Módulo 2 Reabastecimiento de inventario 54
Tabla 25 Sprint Backlog módulo 3 Control de ventas 55
Tabla 26 Product Backlog del cuarto sprint 60
Tabla 27 Sprint Backlog Módulo 8 Pago de pedido 60
Tabla 28 Pruebas de Módulo 2 Reabastecimiento de inventario 65
Tabla 29 Pruebas de Módulo 1 Control de inventario 65
Tabla 30 Pruebas de Módulo 3 Control de ventas 65
Tabla 31 Módulo 3 Control de ventas 66

Tabla 32 Pruebas de Módulo 4 Lista de pedido 66
Tabla 33 Pruebas de Módulo 5 Cobro del pedido 67
Tabla 34 Pruebas del Módulo 6 Pedido de cliente 67
Tabla 35 Pruebas del Módulo 7 Cotización de pedido 68

## ÍNDICE DE ILUSTRACIONES

Ilustración 1 Perfiles de usuario 14
Ilustración 2 Perfil de usuario: Cajera/o 15
Ilustración 3 Perfil de usuario administrador/a 16
Ilustración 4 Diagrama General de Módulos **¡Error! Marcador no definido.**
Ilustración 5 Diagrama General de Casos de uso 25
Ilustración 6 Diagrama General Entidad Relación 26
Ilustración 7 Diagrama Base de datos basada en documentos (firestore) 27
Ilustración 8 Diagrama de actividades del módulo 1: Control de inventario 28
Ilustración 9 Diagrama de actividades del módulo 2: Reabastecimiento de inventario 29
Ilustración 10 Diagrama de actividades del módulo 3: Control de ventas 31
Ilustración 11 Diagrama de actividades del módulo 4: Lista de pedidos 32
Ilustración 12 Diagrama de actividades del módulo 5: Cobro de pedidos 34
Ilustración 13 Diagrama de actividades del módulo 6: Pedido de cliente 35
Ilustración 14 Diagrama de actividades del módulo 7: Cotización del pedido 36
Ilustración 15 Diagrama de actividades del módulo 8: Pago del pedido 37
Ilustración 16 Interfaces a desarrollar 38
Ilustración 17 Interfaz landing page de la aplicación 39
Ilustración 18 Interfaz Carrito de Compras 39
Ilustración 19 Interfaz de Administrador de la aplicación 40
Ilustración 20 Proceso 6.1 Lista de tipo de producto 43
Ilustración 21 Proceso 6.2 Lista de productos 44
Ilustración 22 Proceso 6.3 Producto seleccionado 44
Ilustración 23 Proceso 6.4 Ver lista de insumos 45
Ilustración 24 Proceso 6.5 Seleccionar insumos 45
Ilustración 25 Proceso 6.6 Agregar producto al carrito de compras 46
Ilustración 26 Proceso Módulo 7 Carrito de compra - Cotización de pedido 47
Ilustración 27 Proceso Módulo 4 Lista de pedido 49

Ilustración 28 Proceso 1.1 Loguearse como administrador/a 52
Ilustración 29 Proceso 1.2 Ingresar al componente administrador/a 53
Ilustración 30 Proceso 1.3 Ingresar al componente Inventario 53
Ilustración 31 Proceso 1.4 Observar el stock de cada insumo en el inventario 54
Ilustración 32 Proceso 2.4 Ingresar el stock en el insumo recibido 55
Ilustración 33 Proceso 3.3 Ingresar al componente Control de ventas 56
Ilustración 34 Proceso 3.4 Observar el valor total de la venta del día y mes vigente 57
Ilustración 35 Proceso 3.6 Ingresar al componente Productos 57
Ilustración 36 Proceso 3.7 Observar los detalles de los productos en venta 58
Ilustración 37 Proceso 3.8 Editar la información del producto 58
Ilustración 38 Proceso 3.9 Crear un nuevo producto 59
Ilustración 39 Proceso 8.3 Ingresar a la pestaña Comprobante de pago 61
Ilustración 40 Proceso 8.4 Ingresar los datos del cliente 62
Ilustración 41 Proceso 8.5 Ingresar el comprobante de pago en la base de datos de la aplicación 62
Ilustración 42 Proceso 8.6 Imprimir el comprobante de pago 63

# CAPÍTULO 1. DEFINICIÓN DEL PROYECTO

## 1.1. Planteamiento del problema

Debido a que en la actualidad el mayor ingreso económico del país son las pequeñas empresas (pymes) (Empresarial, 2017), de las cuales un 44% son empresas enfocadas al catering (Ecuador G. d., 2019), es necesario potenciar su crecimiento para que se pueda aumentar el empleo pleno y generar un incremento en el efecto multiplicador que son los restaurantes y bares (Ecuador U. A., 2019). Conociendo que la principal forma de incrementar las ganancias de estas pymes es controlando el porcentaje de costo, el tema propuesto permite analizar la rentabilidad de producto elaborado ya sea en comida o bebida preparada.

En el caso de estudio tomado para la realización del presente trabajo de titulación, para la heladería y cafetería "Helarte", se conoce que no disponen de procesos que ayuden a controlar los presupuestos, ni controlar el inventario. Debido a ello sus márgenes sustentables de ganancia no son claros, conociendo que se denota como una pyme en crecimiento, se requiere realizar un análisis de sus procesos para mejorar la capacidad de ventas junto con la automatización de los mismos para potenciar los dicho procesos, además, se requiere que se mejore la imagen de "Helarte" por medio de la incorporación de un sistema de atención al cliente actualizado y moderno, mediante la implementación de una aplicación móvil que permita generar una mejor y más rápida interacción entre los clientes y la heladería, con ello se espera obtener una ventaja competitiva frente a las demás pymes enfocadas en el mismo target de mercado.

## 1.2. Objetivos

### 1.2.1 Objetivo General

Desarrollar un prototipo funcional aplicando metodologías ágiles de desarrollo de plataformas *web* y móvil, para el control de la gestión de bares y restaurantes utilizando como caso práctico a la heladería Helarte.

### 1.2.2 Objetivos Específicos

- Analizar los procesos de inventario, venta y contabilidad de la heladería "Helarte" y mediante la metodología ágil "*SCRUM*" recabar los requisitos necesarios para la automatización de dichos procesos.
- Realizar el diseño lógico y físico de los procesos de control de ventas de producto, control de inventarios y control de costos de la heladería "Helarte".

- Desarrollar los procesos de: venta de productos, control de inventario, solicitud de materia prima y control de costos por medio del desarrollo de *software* con plataformas *web* y móvil para optimizar las ganancias de la heladería "Helarte".
- Realizar las pruebas de funcionalidad, del prototipo funcional, mediante pruebas *Alpha* y *beta* con los directos interesados, para verificar si hay una satisfacción aceptable del prototipo.

### 1.3. Justificación

Desde que las sociedades antiguas requerían almacenar grandes cantidades de alimentos para ser utilizados en tiempos de necesidad, el concepto del inventario nace como una forma de mantener un orden y determinar la cantidad de bienes como alimentos, granos, animales y subproductos que cada individuo posee como propiedad privada. (Amaya Mayra, 2012)

Como base de cualquier empresa comercial existe la compra y venta de bienes y servicios, por lo que el manejo de inventarios es casi obligatorio debido a que es necesario que la empresa maneje un control del estado contable de sus productos y la situación económica actual de la misma. (Hortensio, 2003)

Con la creciente demanda de calidad y cantidad de pedidos de consumo, como son comidas y bebidas que requieren de una preparación especial, se sugiere para los bares y restaurantes el desarrollo de un sistema capaz de manejar su inventario de materias primas, de una manera óptima y eficiente que les permita controlar el uso de dicha materia prima. (José Antonio Díaz-Batista, 2012)

Además, se requiere agilizar la atención al cliente mediante el uso de tecnología actual que le permita al bar o restaurante presentar sus menús de manera amigable e intuitiva a los clientes, para que puedan rápidamente realizar pedidos que se vinculen directamente con el inventario. (José Antonio Díaz-Batista, 2012)

Por lo cual el presente trabajo de titulación se justifica debido a que cubre la necesidad de los bares y restaurantes de mantener un control sobre sus materias primas, la atención más rápida y eficaz a los clientes y el ingreso por gramo o mililitro consumido dentro de las horas de trabajo.

Dentro del caso de estudio de la heladería "Helarte", se requiere un análisis previo sobre los procesos necesarios, ya que no cuenta con procesos establecidos con anterioridad. Los procesos por desarrollarse serán basados exclusivamente para el caso exclusivo de "Helarte".

Finalmente, gracias a la utilización de tecnología actual el desarrollo de un prototipo funcional que permita tener el control de inventario en tiempo real, el control sobre los pedidos de clientes y el control contable de la heladería "Helarte", se podrá obtener la información sobre las diferentes necesidades en su inventario o su sobrante en el mismo, aumentar su capacidad de venta y atención al cliente además de contar con un control sobre el sistema financiero personalizado.

### 1.4. Alcance

El presente proyecto de disertación de grado culminará con la entrega de los insumos generados del cumplimiento del ciclo de desarrollo de *software* de los procesos de inventario, venta, control de compra y venta de la heladería "Helarte" junto con la aplicación *web* funcionando.

## CAPÍTULO 2: MARCO TEÓRICO

### 2.1 METODOLOGÍAS ÁGILES DE DESARROLLO '*AGILE*'

En la actualidad se define a la metodología ágil de desarrollo como la manera de construir, validar y entregar de manera cíclica un proyecto de cualquier campo enfocándose sobre el campo del desarrollo de *software*. La metodología ágil es adaptable a los cambios que puedan ocurrir entre cada ciclo de desarrollo, haciendo que, durante el ciclo de vida del *software* (planificación, diseño, construcción, validación) se entregue el *software* con las necesidades que los clientes solicitan. (Jiménez, 2020)

### 2.2. *SCRUM*

*Scrum* es un *framework* (marco de trabajo) que utiliza la metodología ágil para desarrollar y mantener productos complejos. Consta de: roles, artefactos, eventos y reglas que se gobiernan entre ellos. Utilizando como base teórica el empirismo, iterativo e incremental optimizando el control del riesgo y su predictibilidad. Trabaja sobre el control de cada proceso utilizando la transparencia, inspección y adaptación. (Ken Schwaber, 2016)

#### 2.2.1 Equipo de *Scrum*

El equipo de *Scrum* es autoorganizado y multifuncional, no existe un líder general del equipo pues la resolución de problemas se decide en equipo. En *Scrum*, un equipo debe ser multifuncional ya que todos son necesarios para desarrollar una característica desde la idea hasta la implementación. Además, el proceso de desarrollo de *Scrum* está respaldado por dos roles específicos: el *ScrumMaster*, que es considerado el dirigente del proceso de desarrollo, el cual ayuda a los miembros del equipo para usar el proceso *Scrum*

y desempeñar su máximo nivel. Y El propietario del producto (PO) quien es el representante de la empresa, los clientes o usuarios, y además guía al equipo para lograr la construcción del producto correcto. (Goat., Software Mountain, 1998)

### 2.2.2 Artefactos de *scrum*

#### 2.2.2.1 *Product Backlog*

La Pila de Producto o *product backlog* es la lista priorizada de requisitos, historias o funcionalidades, entre otros requerimientos de la totalidad del proyecto. Cosas que el cliente quiere, y se las describe usando la terminología del cliente. A esto se le conoce como historias, o elementos de la Pila. Para que el equipo de desarrollo pueda convertir cada requerimiento en un módulo funcional del sistema es necesario que el dueño del producto junto con el *ScrumMaster* defina bien la prioridad y la funcionalidad de cada elemento de la lista, para trabajar en los elementos con mayor prioridad y ajustar en estimaciones de tiempo, recursos, alcance. (Kniberg, 2007)

#### 2.2.2.2 *Sprint Backlog*

El *Sprint Backlog* o pila del *Sprint* es una lista tomada de la lista del producto *Backlog* (lista del producto), esta lista de funcionalidades es en la que se va a trabajar por un periodo de tiempo de entre dos semanas a un mes, esta lista debe ser cumplida y entregada para sus pruebas en el tiempo establecido, la manera de determinar la cantidad de funcionalidades que se pueden hacer durante el *sprint* está definida por el equipo de desarrollo, el *ScrumMaster* y el dueño del producto. El dueño del producto decide la prioridad, es decir se puede elegir qué funcionalidades se requiere desarrollar en este periodo de tiempo y el equipo de desarrollo junto con el *ScrumMaster* deciden si pueden cumplir o no. Posteriormente se llega a un acuerdo y se empieza a trabajar en la lista definida para el *Sprint* actual. (Kniberg, 2007)

#### 2.2.2.3 Terminado

Es el resultado del *sprint*, durante cada ciclo de desarrollo el término "Terminado" significa que se entrega la o las funcionalidades terminadas y probadas al cliente para continuar con el siguiente ciclo y empezar un nuevo *Sprint*. (Kniberg, 2007)

## 2.3 HERRAMIENTAS

### 2.3.1 *Power Designer*

*Powerdesigner* es una herramienta de modelado de datos del *software* que permite construir modelos, planificando y gestionando arquitecturas de información compleja. (SAP, 2020)

### 2.3.2 *Html5*

#### 2.3.2.1 Antecedentes

Html es una herramienta de marcación de hipertexto creada en 1989 para la creación de páginas *web*. El lenguaje *html* tenía un lenguaje poco riguroso con tipado débil lo que provocó que los documentos creados en el lenguaje resultaran caóticos y poco adaptables a los buscadores de la época. Durante la etapa entre los años 1990-2010 *html* evoluciona incorporando diferentes mejoras con la finalidad de procesar la información de acuerdo con las nuevas necesidades que se presentan como: *Xhtml* que sustituye la sintaxis basada en *html* para facilitar la creación manual de documentos. *Xhtml2* que incorporaba un aumento significativo en la lógica y mejoras semánticas al lenguaje, *html* 3.2, *html* 4 y *html* 4.01 que ampliaron y mejoraron el lenguaje incluyendo al trabajo de la *W3C*. El trabajo realizado por el grupo de diseñadores de *Opera*, *Apple* y *Mozilla* el *Web Hypertext Application Technology Working Group*, para finalmente obtener la versión *Html* 5 que admite las versiones de *html*, *xhtml* y sus versiones lo que permite no solo como analizar los documentos sino interpretarlos y con la adaptación de los nuevos navegadores que corrigen la sintaxis el resultado es mucho más práctico poniendo en la tecnología de *html* la ingeniería inversa que los navegadores necesitaban para incorporarse y competir entre ellos. (Franganillo, 2011)

#### 2.3.2.2 Definición

*Html5* es un lenguaje de marcado creado a partir de los fundamentos de la *W3C* junto con la *WHATWG*, que utilizan principios de diseño como son:

- Compatibilidad

Los buscadores actuales pueden procesar el contenido creado con versiones anteriores y con las versiones más actuales de *HTML*. El lenguaje puede ser desarrollado en cualquier versión sin tener que agregar las funcionalidades más actuales, sin embargo, las versiones más antiguas se pueden fusionar con elementos de versiones más actuales. Si la tecnología existe, debe ser utilizada para abarcar casos específicos de uso, en lugar de desarrollar

una nueva. Por lo que tomar la tecnología que existe y diseñar características para adaptarlas al contenido antiguo genera una evolución del lenguaje *HTML*. (W3C, 2007)

- Utilidad

Los problemas tomados del contexto actual deben ser resueltos de manera inmediata, intentando resolver conflictos para múltiples circunscripciones tomando en cuenta que se deben jerarquizar la atención de esos problemas iniciando desde el cliente, hasta satisfacer a las razones teóricas. La documentación debe permitir la interacción entre documentos sin dejar de lado las restricciones de seguridad para lo cual *HTML* logra equilibrar la fluidez semántica junto con la utilidad de la presentación práctica y agradable. (W3C, 2007)

- Interoperabilidad

La implementación es libre al igual que la calidad de presentación, siguiendo las recomendaciones sobre el comportamiento de los autores de contenido. Las soluciones simples se prefieren a las complejas ya que son más fáciles de implementar para los desarrolladores y de fácil uso para los usuarios. El control de errores se da por la recuperación de errores y retroalimentación. (W3C, 2007)

- Acceso Universal

La fluidez del texto creado en *HTML* logra adecuarse a dimensiones de pantalla variables, por lo que puede funcionar en diferentes plataformas, dispositivos y medios. La compatibilidad con *Unicode* permite en la mayoría de los idiomas del mundo dando una fácil accesibilidad global. (W3C, 2007)

### 2.3.3 *CSS*

*CSS* son plantillas de documentos o mejor conocidas como hojas de estilo que complementa a los elementos de *HTML*, para separar la estructura de contenidos de la presentación en cualquier pantalla imaginable (móvil, monitor, *Tablet*, etc.), tecnologías de soporte (lectores de pantalla, líneas de braille). Mediante bloques lógicos *CSS* permite leer mejor los documentos y acortar los tiempos de carga en el *DOM*. (Schulz, 2008)

### 2.3.4 *JavaScript*

#### 2.3.4.1 Antecedentes

A finales de los años 80 e inicios de los años 90 la velocidad máxima alcanzada por los módems era de 28.8 kbps. Con la nueva aparición de aplicaciones *web* cada vez más robustas junto con la implementación de formularios y visualización dinámica se tuvo la necesidad de contar con un lenguaje de programación que se ejecutará en los navegadores

por lo que en el año 1995 Brendan Eich trabajador de *Netscape*, crea un lenguaje que podría solucionar el problema al cual se denominó *liveScript*. Posteriormente *Netscape* firma una alianza con *Sun Microsystems* y desarrollan *JavaScript* para que finalmente en 1997 el organismo *ECMA* (*European Computer Manufacturers Association*) estandarizó *JavaScript* como el lenguaje de programación para ejecución de aplicaciones web avalado por el estándar ISO/IEC-16262. (Pérez, 2009)

#### 2.3.4.2 Definición

*JavaScript* es un lenguaje de programación interpretado que es utilizado en su mayoría para crear aplicaciones *web* y móviles dinámicas, es decir que una página *web* pueda tener animaciones, efectos, ventanas emergentes y rapidez en el manejo de formularios. Las aplicaciones creadas con *JavaScript* pueden ser probadas directamente en cualquier navegador debido a que no requieren de un compilador. (Pérez, 2009)

### 2.3.5 *TypeScript*

*TypeScript* es un lenguaje de programación superset de *JavaScript*, con un tipado fuerte, desarrollado por *Microsoft*, con herramientas enfocadas en programación orientada a objetos. Al ser un superset de *JavaScript*, *TypeScript* puede ser ejecutado en cualquier navegador que esté basado en *JavaScript*, el navegador interpreta el lenguaje en *JavaScript* de manera normal, aunque el programa está basado en *TypeScript*. *TypeScript* puede ser usado para *back-end* con Express o *Front-end* con cualquier *framework* que soporte *JavaScript*. (Microsoft, 2012)

### 2.3.6 *Node.js*

*Node.js* es un entorno de ejecución de *JavaScript* del lado del servidor, basado en eventos, además de ser una librería que compila y ejecuta *JavaScript* a velocidad muy elevadas lo que proporciona una manera óptima de desarrollar aplicaciones *web* escalables. *Node.js* está diseñado para soportar una gran cantidad de tráfico y donde la lógica del lado del servidor y el procesamiento no presenten problemas al responder al cliente. (Abernethy, 2011)

### 2.3.7 *Angular Material*

*Angular Material* es un *kit* de herramientas de código abierto para desarrollar con *HTML*, *CSS* y *JS*. Desarrollado por el equipo de *Angular* para generar plantillas, prototipos y aplicaciones *FrontEnd* por medio del sistema de cuadrícula responsiva además de componentes preconstruidos y *plugins* basados en *jQuery*. (Google LLC, 2010)

### 2.3.8 *Angular*

*Angular* es un marco de trabajo (*Framework*) desarrollado por *Google* para el desarrollo de aplicaciones *web*, *web* móvil, móvil nativo y escritorio mediante la modelo vista controlador (MVC) utilizando *HTML*, *CSS*, *JavaScript* y *TypeScript*. *Angular* mediante el uso del *CLI Angular* inicializa, desarrolla y mantiene aplicaciones escalables. (Google, 2020)

### 2.3.9 *Express*

*Express* es un marco de aplicaciones *web node.js* mínimo y flexible que proporciona un sólido conjunto de características para aplicaciones *web* y móviles con métodos de utilidad *HTTP* y *middlewares* para crear o implementar *Apis* más robustas, rápidas y fáciles. (expressjs, 2017)

### 2.3.10 *PWA*

#### 2.3.10.1 Antecedentes

Al hablar de aplicaciones (*apps*) se debe mencionar que existen las aplicaciones *web* y móviles nativas para distintos dispositivos como son computadoras, *laptops*, *tablets*, *notebooks* y teléfonos inteligentes (*smartphones*). Las aplicaciones *web* son creadas con el fin de dar darle la posibilidad de trabajo remoto, enviando y recibiendo información de un servidor *web*, mediante el uso de navegadores en cualquier dispositivo con el uso de la tecnología responsiva, por otro lado, las aplicaciones móviles nativas permiten mantener la información en memorias caché del dispositivo y a su vez manejar todos sus sensores y periféricos mientras se utilizan diferentes aplicaciones. (Google Developers, 2020)

#### 2.3.10.2 Definición

Las *progressive web app* (*PWA*) o aplicaciones *web* progresivas que utilizan las características de una aplicación *web* como son la interacción de servidores *web* mediante intérpretes, las funcionalidades responsivas combinándolas con las ventajas de las aplicaciones móviles nativas como son la simulación de una aplicación instalada, la rápida respuesta al uso, el ingreso de información y su manipulación con cache del dispositivo por lo que permite tener un ahorro en la programación y una ventaja en tiempo de respuesta en el uso con el usuario. Sus estadísticas de uso permiten visualizar que las *PWA* ganan cada vez más usuarios por su rapidez, fiabilidad, consistencia y sus adaptabilidades a la hora de ahorrar costos de producción. (Platzi, 2020)

#### 2.3.10.3 Estadística de uso

Para el año 2014 el uso de aplicaciones para *smartphones* superó al uso de aplicaciones *web* debido, en su mayoría, a la portabilidad y facilidad de uso de los usuarios. Los usuarios que utilizan las aplicaciones *web,* lo hacen con un sentido enfocado al uso del servicio elegido, mientras que los usuarios de las aplicaciones móviles lo hacen buscando distracciones en su mayoría y en su minoría el uso enfocado mediante el uso de sensores del *smartphone*. (comscore, 2016)

Muchas compañías al actualizar sus aplicaciones pasándolas de ser simples *apps webs* a *PWA* incrementaron sus ganancias y ampliaron sus mercados, entre las marcas conocidas que implementaron *PAWs* están: *Starbucks*, *Trivago*, *Tinder*, *Uber*, *Pinterest*, *Forbes*, *Olx*, etc. Quienes incrementaron desde 20% al 50% sus ganancias a partir de la implementación de la tecnología *PWA*. (PWA stats, s.f.)

### 2.3.11 Paquetes y librerías

#### 2.3.11.1 *NPM* (*Node Package Manager*)

*NPM* es un gestor de librerías o paquetes de código abierto con licencia (*MIT*), desarrollado en lenguaje *JavaScript*, por el cual se puede obtener cualquier módulo necesario para el correcto funcionamiento de un proyecto. (Tavares, 2020)

#### 2.3.11.2 *Flexbox Grid*

*Flexbox Grid* es una librería *NPM*, que facilita el uso de las propiedades de *flexbox* en los módulos *Angular*. (Joseph, 2016)

#### 2.3.11.3 *ngx-auth-firebaseui*

*ngx-auth-firebaseui* es una librería *NPM*, que instala un módulo para facilitar la autenticación de usuarios, dentro de una aplicación desarrollada en Angular. (Nahas, 2020)

#### 2.3.11.4 *html2canvas*

*html2canvas* es una librería *NPM*, que permite tomar una captura de pantalla tomando como referencias partes del *DOM* visible en el momento de utilizarlo. (Nicklasvh, 2020)

#### 2.3.11.5 *jspdf*

*jspdf* es una librería *NPM*, que permite generar archivos tipo *PDF* desde *JavaScript*. (MrRio, 2020)

### 2.3.12 *Git*

*Git* es un Sistema de control de versiones. Creado por Linus Torvals, su uso garantiza la eficiencia y confiabilidad del versionamiento de un proyecto.

El uso de *Git* facilita el desarrollo de proyectos en grupos o individual a través del tiempo. (Camacho, 2020)

### 2.3.13 *GitHub*

*GitHub* es una plataforma que almacena y coordina proyectos en la nube, perteneciente a *Microsoft*. Su combinación con el control de versiones de *Git*, lo vuelve una herramienta potente para el desarrollo de aplicaciones en tiempo real. (Camacho, 2020)

### 2.3.14 *Cloud Firestore*

*Cloud firestore* es una base de datos *NoSQL* flexible y escalable que ayuda a la sincronización de datos entre las *apps* y los usuarios a tiempo real a través de dispositivos que soporten aplicaciones. (firestore.Google.com, 2020)

### 2.3.15 Resumen de uso de herramientas

*Tabla 1 Resumen de uso de herramientas*

| Herramienta | Uso en el proyecto |
|---|---|
| **2.3.1 *Power Designer*** | Desarrollar los diagramas *UML* de: diagrama de entidad relación, diagrama de casos de uso y diagrama de actividades. |
| **2.3.2 *Html5*** | Desarrollo de *Front-End* del proyecto mediante el *framework Angular*. |
| **2.3.3 *CSS*** | Desarrollo de estilos para visualización de *Html5* dentro del *framework Angular*. |
| **2.3.4 *JavaScript*** | Lenguaje de programación base utilizado por el *framework Angular* en el que se desarrollara el *Back-End* del proyecto. |
| **2.3.5 *TypeScript*** | *Superset* de *JavaScript* utilizado por *Angular* para interpretar tanto el *Back-End* y el *Front-End*. |
| **2.3.6 *Node.js*** | Librería utilizada por *JavaScript* para manejar de manera adecuada el tráfico de los usuarios del lado del cliente. |
| **2.3.7 *Angular Material*** | Herramienta que facilita el uso de grilla para ubicar cada elemento dentro de *Html5*. |
| **2.3.8 *Angular*** | *Framework* para el desarrollo del proyecto en el *Front-End* y en el *Back-End* |
| **2.3.9 *Express*** | Marco de aplicaciones para poder desarrollar aplicaciones *web* y móviles, que se utilizara para desarrollar el proyecto y entregar una aplicación multidispositivo. |
| **2.3.10 *PWA*** | Tecnología que permite la emulación de una aplicación *web* nativa o *app* nativa, por lo que será utilizada para la instalación de la aplicación en cualquier plataforma requerida. |
| **2.3.11.1 *NPM*** | Tecnología de alojamiento de librerías para *JavaScript*, de código abierto, que permite descargar cualquier herramienta necesaria para el funcionamiento de proyectos. |

| | |
|---|---|
| **2.3.11.2 *Flexbox Grid*** | Librería de *JavaScript* que permite el fácil uso de *flexbox* en cualquier proyecto. |
| **2.3.11.3 *ngx-auth-firebaseui*** | Librería de *JavaScript*, exclusiva para uso de la autenticación de *firebase*, en cualquier proyecto. |
| **2.3.11.4 *html2canvas*** | Librería de *JavaScript*, que permite tomar una captura de pantalla, tomando elementos exactos del DOM activado. |
| **2.3.11.5 *jspdf*** | Librería de *JavaScript*, que permite la creación de archivos PDF desde proyectos basados en *JavaScript*. |
| **2.3.12 *Git*** | Sistema de control de versiones, que permite manipular diferentes versiones del proyecto en el tiempo de desarrollo y actualización. |
| **2.3.13 *GitHub*** | Almacenamiento en la nube, donde se puede guardar las versiones del proyecto hechas en *Git*, para trabajar en tiempo real en los grupos de desarrollo. |

## 2.4 MODELO VISTA CONTROLADOR

La modelo vista controlador fue nombrado en el año 1979 por Trygve Reenskaug e introducido al lenguaje de programación de *Smalltalk* en los años 80, con el objetivo de reducir los esfuerzos necesarios en la implementación de sistemas múltiples y sincronizados con los datos, definiéndolo como un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, dando así la actualización y mantenimiento del *software* de manera fácil y en poco tiempo. (Yenisleidy Fernández Romero, 2012)

Este patrón permite separar los intereses entre las solicitudes del usuario que se enrutan a un controlador, que a su vez se encarga de trabajar con el modelo para realizar las acciones del usuario o recuperar los resultados de consultas. El controlador elige la vista para mostrar al usuario y proporciona cualquier dato de modelo que sea necesario. (Microsoft, 2020)

## 2.5 MODELO VISTA – VISTA MODELO (MVVM)

El modelo vista-vista modelo, es un patrón de arquitectura de *software* que se caracteriza por tratar de separar la interfaz de usuario de la lógica de programación.

El modelo vista-vista modelo (MVVM), es mostrado por primera vez por John Gossman en el año 2005, en un artículo *MSDN* de *Microsoft*, mostrándolo como una variación del patrón de diseño modelo vista controlador (MVC).

Lo componen los elementos de modelo que contiene la capa de datos y la lógica de negocio. Está totalmente desconectado de la interfaz de usuario. La vista que Representa la interfaz de usuario, e interpreta la información que llega a través de los elementos

visuales de los que está hecho. La vista muestra características activas, con eventos y enlaces a datos. Modelo de vista que es un actor que actúa como intermediario entre el modelo y la vista, comportándose como una abstracción de la interfaz de usuarios mientras recibe la información del modelo. (Microsoft, 2005)

## 2.6. INVENTARIOS

### 2.6.1 Definición

Los inventarios son la cantidad de bienes que una empresa mantiene en un periodo de tiempo determinado, ya sea para la venta directa o para consumo de bienes y servicios, Es la relación entre la producción y la venta de un producto además de representar una considerable inversión para la empresa. (Durán, 2012)

### 2.6.2 Tipos de inventarios

Los tipos de inventario están determinados por el sector en el que se enfoque, desde el sector manufacturero al sector de servicios. El inventario siempre estará en función de la incertidumbre, la variabilidad de la demanda, la incertidumbre o variabilidad en el proceso y la incertidumbre o variabilidad en el suministro. Tomando en cuenta que para el enfoque de bares y restaurantes, el inventario a utilizarse será el manufacturero los inventario a manejarse serán: el de materia prima que se encarga de prevenir la variabilidad en la cadena de suministro, el inventario de trabajo en proceso que incluyen los materiales para producir un bien, pero que aún no se encuentran en su forma terminada, los inventarios de productos terminados que involucran a cualquier bien o producto destinado al consumidor final y que forman parte de la cadena de distribución y los inventarios de suministros que son utilizados de apoyo a las operaciones pero que no forman parte del producto final. (Parada, 2006)

### 2.6.3 Manejo de *stock*

Debido a la variación de precios de adquisición que tienen los productos a lo largo del tiempo, la incidencia en el valor de las existencias finales o en el costo al momento de sus salidas presenta distintas situaciones. Si el volumen de los productos en almacenamiento es grande, se vuelve insostenible el registro de cada unidad debido al alto costo que esto supondría, por lo que es necesario establecer un criterio de valoración manteniendo el principio de precios de adquisición, sin necesidad de controlar el coste de cada unidad de producto. Los criterios que se manejan para mantener el principio de precios de

adquisición son: *FIFO* (*first in, first out*) que considera que las unidades que salen del almacén son las más antiguas de acuerdo con el principio de renovación (primero en entrar, primero en salir), por lo que las unidades que están en almacén son las más recientes. *LIFO* (*last in, last out*) que considera que las unidades que salen del almacén son las más recientes por lo que las unidades que se mantienen en almacén serán las más antiguas, lo que es lo contrario al principio de renovación. PRECIO ESTÁNDAR que considera que el coste se da en función de los componentes que lo integren como los materiales, la mano de obra y los gastos de fabricación. El precio estándar se fija por producto en base a un análisis que logra valorar los movimientos entre el almacén y los consumidores finales controlando los costos reales mediante una contabilidad lógica por cuentas intermedias. (Gutiérrez, 2007)

Para el presente proyecto se utilizó el modelo de almacenamiento *FIFO,* debido al tipo de producto que se manejó dentro de la heladería "Helarte".

### 2.7. *E - Commerce*

El *e-commerce* o comercio en línea también conocida como economía de plataforma es un método de compraventa de bienes, productos o servicios que utiliza el *internet* como el medio donde comercializar por lo que se vuelve un comercio *online*. Debido al crecimiento de las redes, la facilidad de compra y pago, esta modalidad de comercio se populariza y se espera que pronto se normalice como una forma de comercio formal. (sumup, 2020)

## CAPÍTULO 3: ANÁLISIS Y DISEÑO

### 3.1 Introducción

En el presente capítulo, se determina un documento donde se detalla la información sobre los perfiles de los usuarios, las historias de usuario que servirán para determinar los requerimientos funcionales y no funcionales, la pila del producto (*product backlog*) donde se detallan los módulos a trabajar con los datos de un identificador, nombre, importancia, estimación inicial, la comprobación y notas de información de cómo realizar cada una, además se detalla los diagramas de casos de uso, diagrama de clases, diagrama de entidad relación, diagrama físico, un diccionario de datos general, y los diagramas de actividades de cada módulo.

### 3.2 Perfiles de usuario

Los perfiles de usuario es la información que sintetiza las características recurrentes de un actor que interactúa con el *software* de uso. Los perfiles de usuario utilizados en el presente trabajo fueron: Cliente, Cajero y Administrador.

*Tabla 2 Perfiles de usuario*

| **Perfil de usuario: Cliente** | |
|---|---|
| **Característica** | Detalle |
| **Edad** | El cliente en general es mayor de edad, pero el rango de edad varía desde los mayores a 8 años. |
| **Ubicación** | El cliente frecuente se encuentra en un radio de hasta 5 kilómetros del local comercial “Helarte” |
| **Posición socioeconómica** | El cliente es de una situación socioeconómica variada, dependiendo de la cual demanda una preparación de productos más elaborados. |
| **Acompañamiento social** | El cliente frecuentemente acude al local acompañado de al menos una persona. Se puede observar también acompañamiento familiar de al menos 3 personas o solo asiste de manera individual. |

*Ilustración 1 Perfiles de usuario*

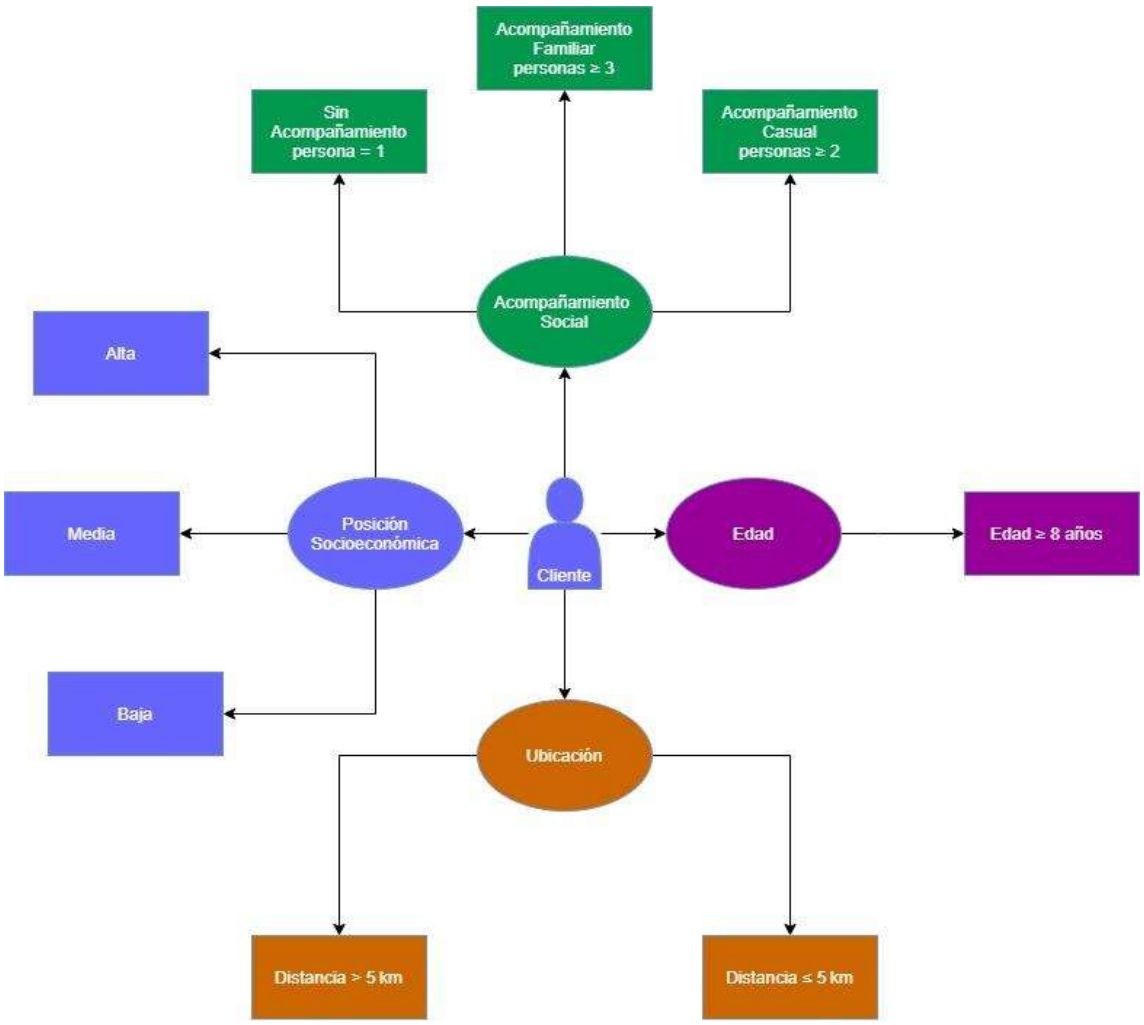



*Tabla 3 Perfil de usuario: Cajera/o*

| **Perfil de usuario: Cajera/o** | |
|---|---|
| **Característica** | Detalle |
| **Edad** | El cajero o la cajera debe ser mayor de edad. |
| **Ubicación** | El cajero o la cajera frecuente se encuentra en un radio de hasta 30 kilómetros del local comercial "Helarte", pero preferentemente se busca en un radio de hasta 5 kilómetros. |
| **Sexo** | El cajero o la cajera puede ser de sexo masculino, femenino u otro. |
| **Datos extra** | El cajero o la cajera debe disponer de su correo electrónico, un horario laboral y su aspiración salarial, la nacionalidad es de registro opcional |

*Ilustración 2 Perfil de usuario: Cajera/o*

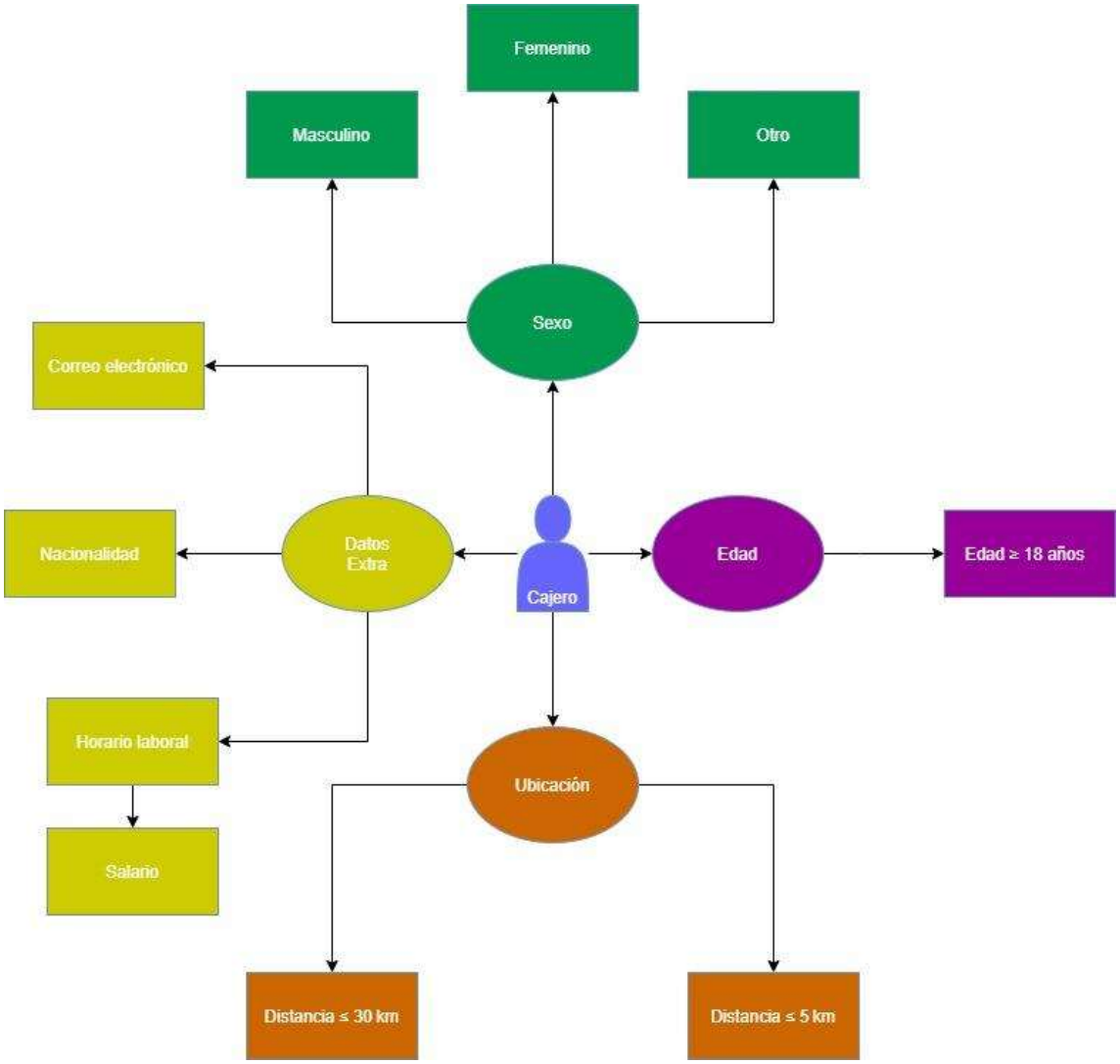
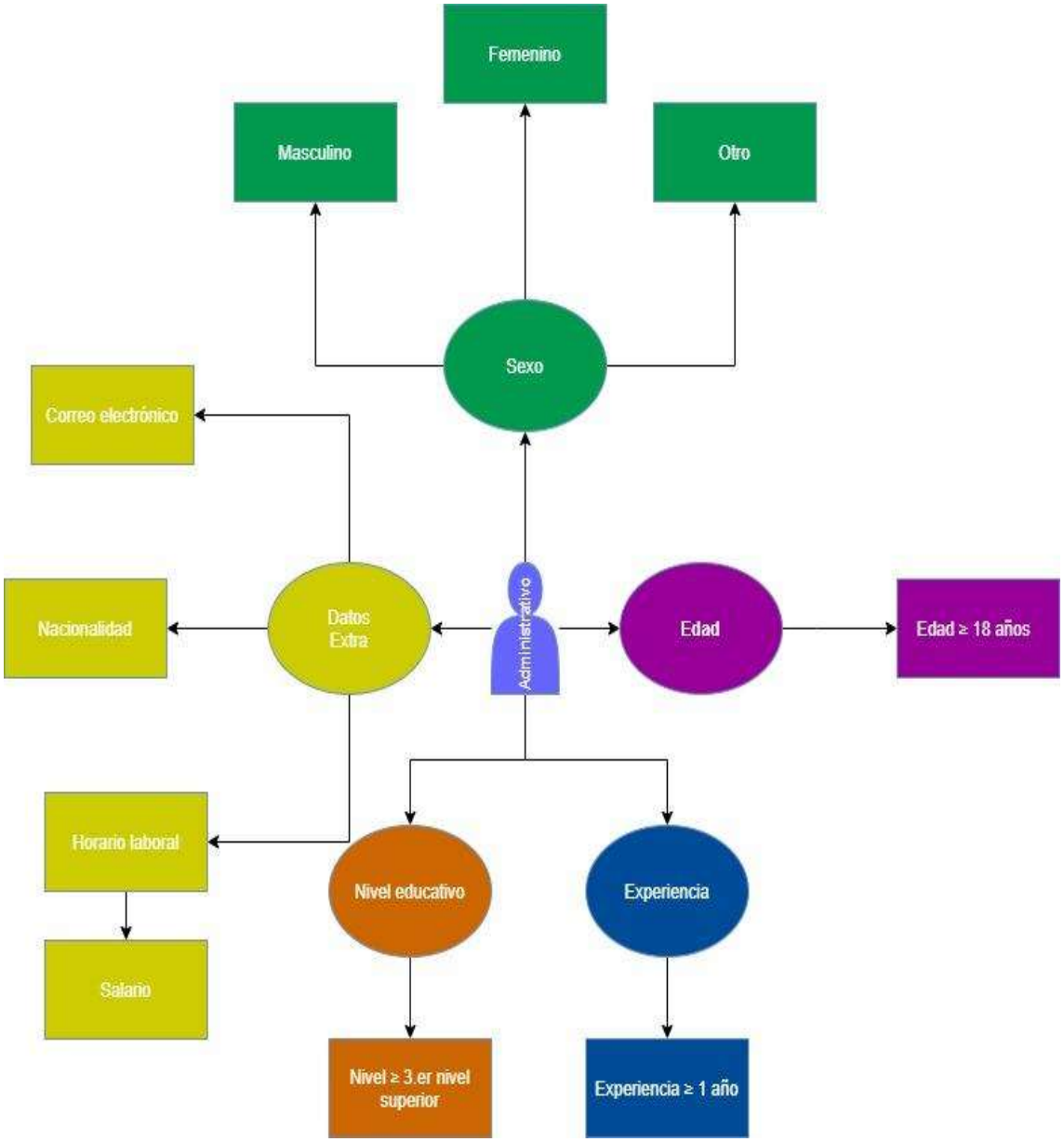



*Tabla 4 Perfil de usuario: Administrador/a*

| **Perfil de usuario: Administrador/a** | |
|---|---|
| **Característica** | Detalle |
| **Edad** | El administrativo debe ser mayor de edad. |
| **Sexo** | El administrativo puede ser de sexo masculino, femenino u otro. |
| **Datos extra** | El administrativo debe disponer de su correo electrónico, un horario laboral y su aspiración salarial, la nacionalidad es de registro opcional. |
| **Nivel educativo** | El administrativo debe poseer un grado académico de nivel tres o superior en administración o afines. |

| | |
|---|---|
| **Experiencia** | El administrativo debe disponer de una experiencia dirigiendo proyectos o entidades relacionadas a la venta de alimentos de al menos un año. |

*Ilustración 3 Perfil de usuario administrador/a*




### 3.3 Historias de usuario

Las historias de usuario son las solicitudes que los clientes piden al *Product Owner* para ser interpretadas con la ayuda del *Scrum Master*, posteriormente serán incluidas en el *Product Backlog* para ser implementadas y probadas por el equipo de desarrollo. En el presente trabajo se obtuvo las siguientes historias de usuario:

*Tabla 5 Historia de usuario N.º 1*

| **Historia de usuario** | |
|---|---|
| Número: 1 | Usuario: Administrador |
| Nombre de historia: Control de inventario | |
| Prioridad en negocio: alto | Riesgo en desarrollo: alto |
| Puntos estimados de prioridad: 9 | |
| Programador responsable: César Armendariz | |
| Descripción: Ver la cantidad de productos disponibles(unidades), código, nombre, descripción (caso de *snacks*, *bakeries*, conos, tulipanes, etc.). O cantidad de productos disponibles(gramos), código, nombre, descripción (sabores de helado). Para observar la cantidad de los productos disponibles | |
| Validación: Entrar a la aplicación, ingresar al módulo de inventario, observar todos los productos en *stock*, ingresar o retirar productos, actualizar la página y observar los cambios. | |

*Tabla 6 Historia de usuario N.º 2*

| **Historia de usuario** | |
|---|---|
| Número: 2 | Usuario: Administrador |
| Nombre de historia: Reabastecimiento de inventario | |
| Prioridad en negocio: medio | Riesgo en desarrollo: bajo |
| Puntos estimados de prioridad: 6 | |
| Programador responsable: César Armendariz | |
| Descripción: Entrar al inventario y realizar un reabastecimiento de productos faltantes. | |
| Validación: Entrar a la aplicación, ingresar al módulo de inventario, registrar el ingreso del nuevo producto al *stock* mediante el programa, actualizar datos, verificar cambios. | |

*Tabla 7 Historia de usuario N.º 3*

| **Historia de usuario** | |
|---|---|
| Número: 3 | Usuario: Administrado |
| Nombre de historia: Control de ventas | |
| Prioridad en negocio: alto | Riesgo en desarrollo: alto |
| Puntos estimados de prioridad: 7 | |
| Programador responsable: César Armendariz | |
| Descripción: Observar el total de ventas de los productos a diario para mantener el control de las ventas actualizada. | |
| Validación: Entrar a la aplicación, ingresar al módulo de estadísticas y verificar las ventas totales. | |

*Tabla 8 Historia de usuario N.º 4*

| **Historia de usuario** | |
|---|---|
| Número: 4 | Usuario: Cajero |
| Nombre de historia: Lista de pedidos | |
| Prioridad en negocio: alto | Riesgo en desarrollo: medio |
| Puntos estimados de prioridad: 7.5 | |

| |
|---|
| Programador responsable: César Armendariz |
| Descripción: Tener una lista de pedidos de los clientes para controlar la realización de los productos elaborados junto con su entrega y/o la de *snacks* o *bakeries*. |
| Validación: Entrar a la aplicación, ingresar al módulo de pedidos, verificar la cola de pedidos ingresados para realizar, realizar el pedido para la entrega |

*Tabla 9 Historia de usuario N.º 5*

| **Historia de usuario** | |
|---|---|
| Número: 5 | Usuario: Cajero |
| Nombre de historia: Cobro de pedidos | |
| Prioridad en negocio: alto | Riesgo en desarrollo: alto |
| Puntos estimados de prioridad: 8.5 | |
| Programador responsable: César Armendariz | |
| Descripción: Emitir notas de venta para realizar cobros a los clientes, de los pedidos que hayan realizado. | |
| Validación: Entrar en la aplicación, ingresar al módulo de pedidos, ingresar a la pestaña de pago, cobrar el pedido realizado por el cliente, ingresar datos del cliente para emitir una nota de venta que será enviada al correo del cliente. | |

*Tabla 10 Historia de usuario N.º 6*

| **Historia de usuario** | |
|---|---|
| Número: 6 | Usuario: Cliente |
| Nombre de historia: Pedido de cliente | |
| Prioridad en negocio: alto | Riesgo en desarrollo: medio |
| Puntos estimados de prioridad: 7 | |
| Programador responsable: César Armendariz | |
| Descripción: Observar la lista de los productos disponibles, junto con sus precios Para solicitar uno o varios productos ya sean preparados o de consumo inmediato. | |
| Validación: Entrar en la aplicación, seleccionar el tipo de producto que se quiere consumir, seleccionar el producto de la lista observable, ingresar al carrito de compras y verificar la entrada del producto en el módulo de carrito de compra. | |

*Tabla 11 Historia de usuario N.º 7*

| **Historia de usuario** | |
|---|---|
| Número: 7 | Usuario: Cliente |
| Nombre de historia: Cotización de pedido | |
| Prioridad en negocio: alto | Riesgo en desarrollo: alto |
| Puntos estimados de prioridad: 7.5 | |
| Programador responsable: César Armendariz | |
| Descripción: Observar el costo total de mi pedido previo a aceptarlo para tener la seguridad de poder pagarlo. | |
| Validación: Entrar en la aplicación, ingresar al módulo de carrito de compra, verificar los productos solicitados, observar el costo total de todos los productos del pedido, aceptar el costo, acercarse a la barra de entrega. | |

*Tabla 12 Historia de usuario N.º 8*

| **Historia de usuario** | |
|---|---|
| Número: 8 | Usuario: Cliente |
| Nombre de historia: Pago de pedido | |
| Prioridad en negocio: alto | Riesgo en desarrollo: alto |
| Puntos estimados de prioridad: 7.5 | |
| Programador responsable: César Armendariz | |
| Descripción: Quiero realizar el pago con efectivo y recibir una nota de venta electrónica de mi pedido (opcional). | |
| Validación: Entrar en la aplicación, Ingresar al módulo de pedidos, ingresar a la pestaña de pago, solicitar el pago al contado, realizar el pago, ingresar datos requeridos, verificar si la factura llego a la cuenta de correo electrónico entregada. | |

## 3.4 Requerimientos funcionales

Los requerimientos funcionales toman en cuenta las historias de usuario previamente obtenidas mediante el levantamiento de requerimientos para obtener los módulos de desarrollo los cuales se enlistan a continuación para luego ingresarlos a lista de producto (*producto backlog*), y posteriormente elaborados en cada *sprint* de desarrollo. Los módulos para trabajar se encuentran en la tabla a continuación.

*Tabla 13 Requerimientos funcionales*

| Desarrollo de la plataforma *web* (*responsive*), y móvil con los siguientes módulos funcionales | |
|---|---|
| Tienda | Administrador |
| *Landing Page* de productos<br>Helados<br>*Crepes*<br>*Shakes*<br>Cafes<br>Carrito de compras | *Login* de usuario<br>Categoria de productos<br>Lista de productos<br>Inventario de producto |

### 3.5 Requerimientos no funcionales

Los requerimientos no funcionales son las características que las aplicaciones deben tener para su funcionamiento intrínseco, dándole así características que elevan la calidad de la aplicación y por tanto dan una ventaja competitiva en el mercado.

*Tabla 14 Requerimientos no funcionales*

| Desarrollo de la plataforma con los siguientes requerimientos no funcionales | | | |
|---|---|---|---|
| Seguridad de datos y cuentas de usuario | Conectividad constante cuando existe uso de *internet* y mantención de datos cuando no se tenga uso de *internet* | Mantenibilidad del uso en todo momento | Desarrollado para multiplataforma de *software* |

### 3.6 Product Backlog o Lista del Producto

*Tabla 15 Product Backlog*

| **PRODUCT BACKLOG** | | | | | |
|---|---|---|---|---|---|
| **ID** | Módulo | Usuario | Puntos estimados de prioridad /100 puntos | Como comprobar | Notas |
| **1** | Control de inventario | Administrador | 90 | Entrar en la aplicación, ingresar al módulo de administrador y al componente de inventario, observar todos los insumos del inventario de productos. | Se requiere el diagrama de caso de uso y el diagrama de actividades. |
| **2** | Reabastecimiento de inventario | Administrador | 60 | Entrar en la aplicación, ingresar al módulo de administrador y al componente de inventario, ingresar nuevo *stock* al insumo requerido. | Se requiere el diagrama de caso de uso y el diagrama de actividades. |
| **3** | Control de ventas | Administrador | 70 | Entrar en la aplicación, ingresar al módulo de administrador, ingresar al componente de control de ventas y observar el valor total vendido en el día y el mes vigente. Además, observar el total de productos vendidos en el día y mes vigente. | Se requiere el diagrama de caso de uso y el diagrama de actividades. |
| **4** | Lista de pedidos | Cajero | 75 | Entrar en la aplicación, ingresar al módulo de carrito de compras, verificar la lista de productos en el pedido | Se requiere el diagrama de caso de uso y el diagrama de actividades. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | ingresado, solicitar una confirmación por parte del cliente | |
| **5** | Cobro de pedidos | Cajero | 85 | Entrar en la aplicación, ingresar al componente carrito de compra, ingresar a la pestaña de confirmación de pedido, cobrar el pedido realizado por el cliente. El cajero procederá posterior al pago a preparar los productos del pedido del cliente. | Se requiere el diagrama de caso de uso y el diagrama de actividades. |
| **6** | Pedido de cliente | Cliente | 70 | Entrar en la aplicación, seleccionar el tipo de producto que se quiere consumir, seleccionar el producto de la lista observable, dependiendo del producto seleccionado se deberá elegir los insumos con los que se debe preparar el producto. Ingresar al carrito de compras y verificar la entrada del producto en el componente de carrito de compras. | Se requiere el diagrama de caso de uso y el diagrama de actividades. |
| **7** | Cotización de pedido | Cliente | 75 | Entrar en la aplicación, ingresar al componente de carrito de compra, verificar los productos seleccionados por el cliente, observar el costo total del pedido y comunicárselo al cliente. | Se requiere el diagrama de caso de uso y el diagrama de actividades. |
| **8** | Pago de pedido | Cliente | 80 | Entrar en la aplicación, ingresar al componente de carrito de compra, ingresar a la pestaña de Comprobante, el cajero debe solicitar el pago al contado. El cliente debe realizar el pago, posteriormente el cajero debe | Se requiere el diagrama de caso de uso y el diagrama de actividades. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | ingresar datos requeridos por la aplicación, e ingresar el comprobante de pago a la base de datos y opcionalmente se puede descargar un comprobante de pago. | |

### 3.7 Diagrama General de Módulos

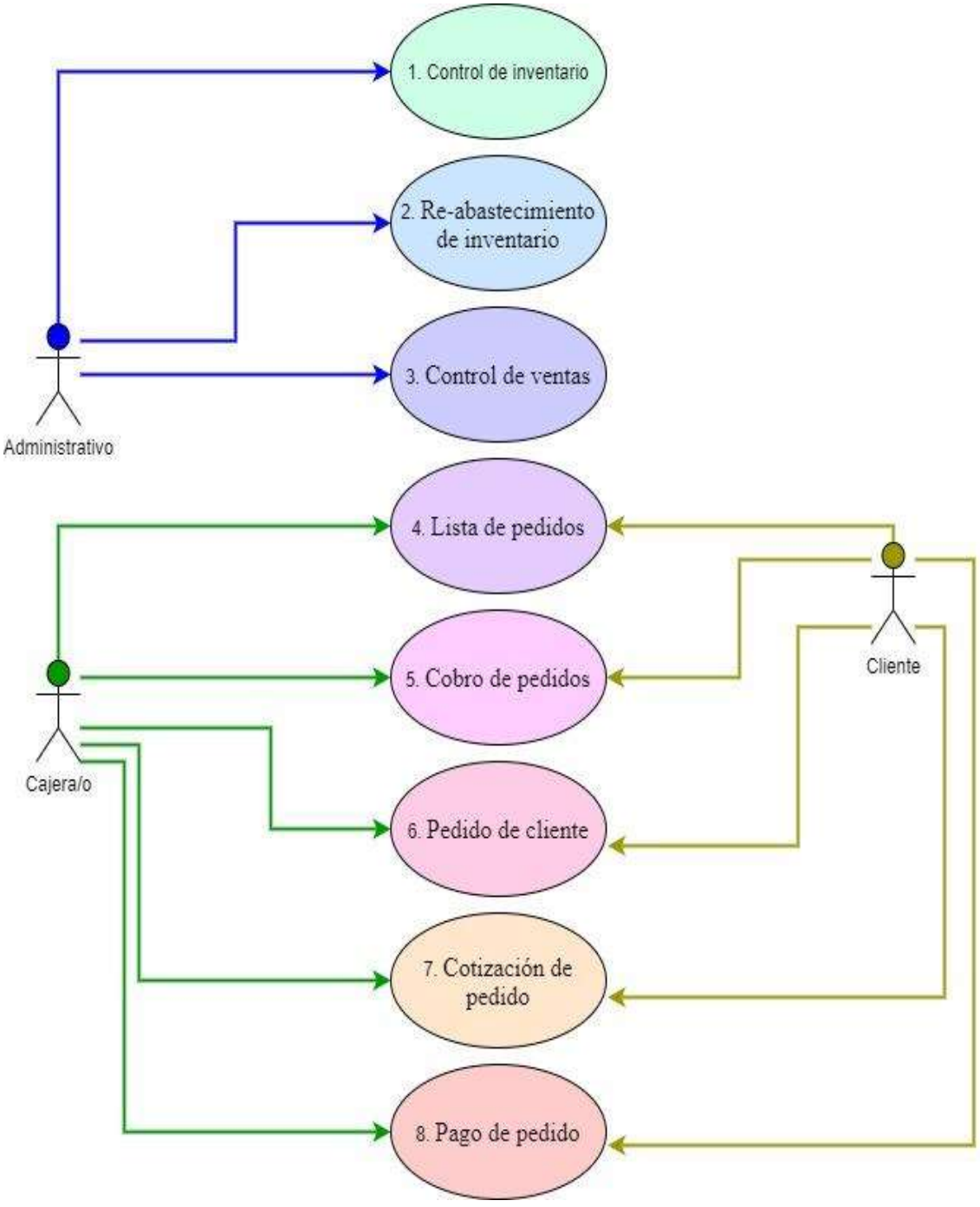
En el diagrama general de módulos a continuación, se puede observar los módulos a implementarse dentro del programa para el uso de todos los actores participantes, cumpliendo con el desarrollo planificado en el *Product Backlog*.

*Ilustración Diagrama General de Módulos*

### 3.8 Diagrama General de casos de uso

En el diagrama general de casos de uso a continuación, se puede observar los módulos donde los actores pueden ingresar, interactuar con el programa, ingresar solicitudes, además, en los casos del actor administrador y cajero se puede tener funcionalidades de *CRUD* básico en los datos.

*Ilustración 5Diagrama General de Casos de uso*

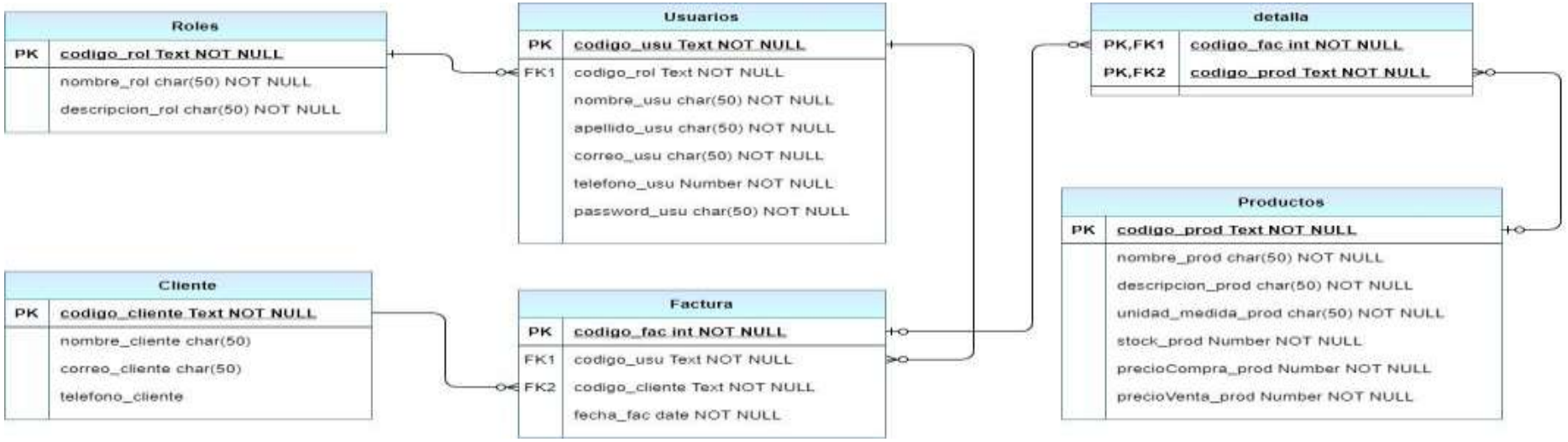


## 3.8 Modelos

### 3.8.1 Diagrama entidad relación

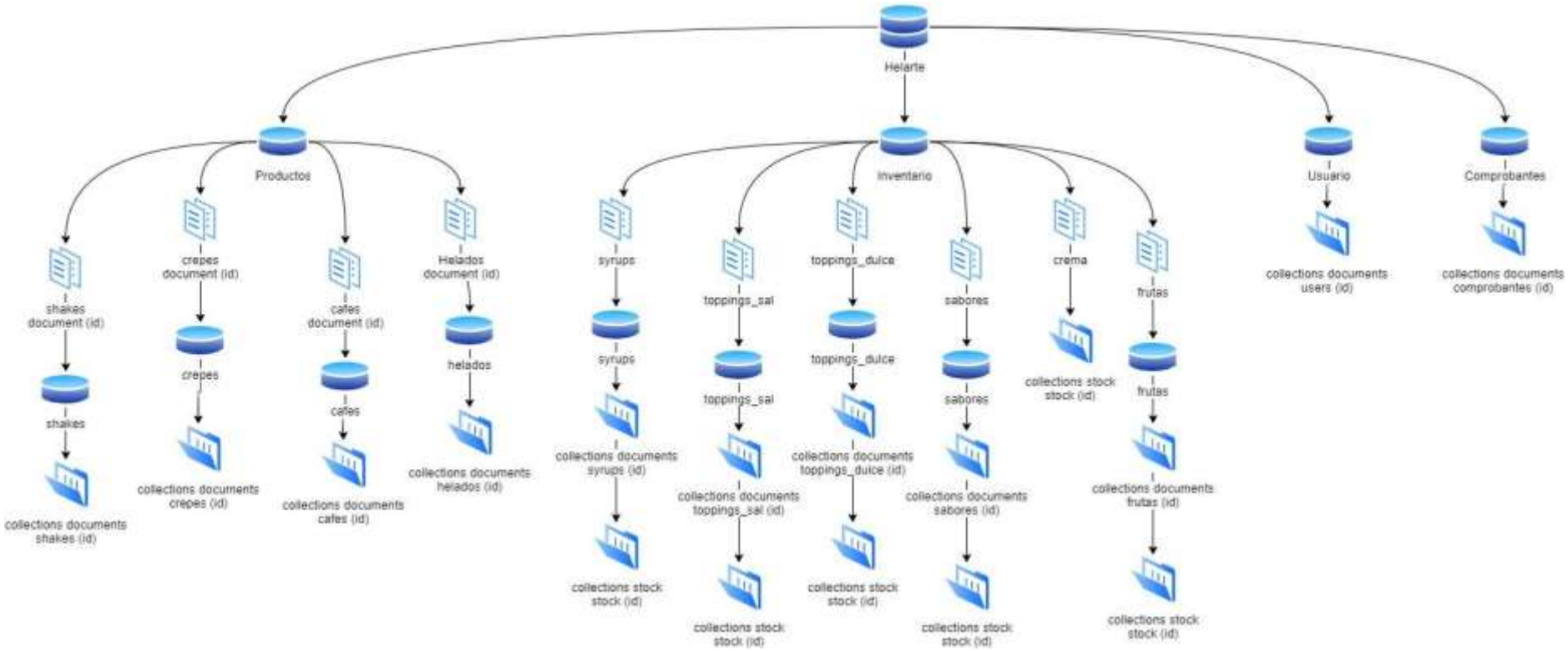
En el diagrama entidad relación a continuación, representa el modelo dentro de la arquitectura MVC que permitió entender la lógica de negocio y el manejo de datos.

*Ilustración 6 Diagrama General Entidad Relación*



### 3.8.2 Esquema de base de datos basada en documentos

*Ilustración 7 Diagrama Base de datos basada en documentos (firestore)*



## 3.9 Diagramas de actividades

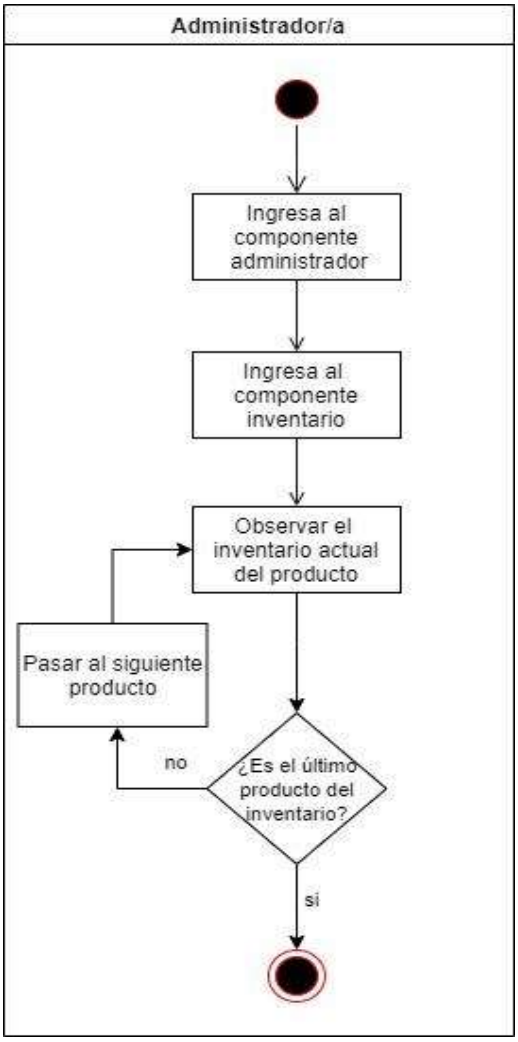
Los diagramas de actividades han sido realizados en base a los módulos establecidos en el diagrama de actividades. Muestran el comportamiento del sistema dentro de la ejecución de cada módulo detallando la secuencia de sus actividades.

### 3.9.1 Diagrama de Actividades del módulo 1: Control de inventario

**Descripción:** En el módulo 1 control de inventario el usuario administrador puede observar el inventario actual de cada insumo utilizado en los productos elaborados en la heladería.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 1 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 8 Diagrama de actividades del módulo 1: Control de inventario*



**Detalle del proceso**:

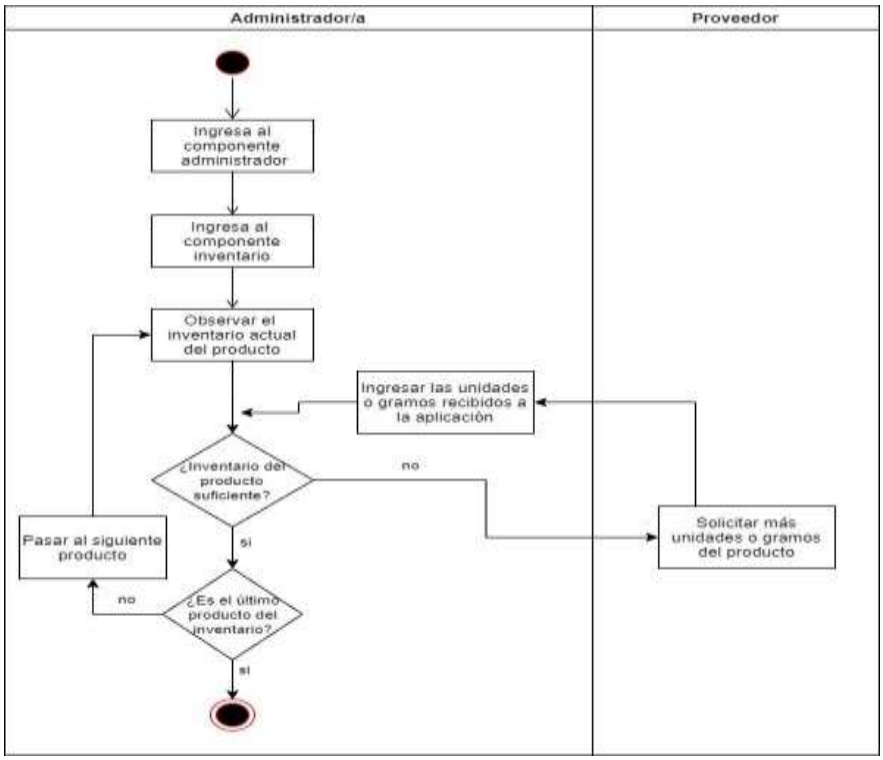
1. El usuario administrador/a ingresa a la aplicación al componente administrador.
2. El usuario administrador/a ingresa al componente inventario.
3. El usuario administrador/a Observa el *stock* del insumo actual en el inventario.
    a) Si es el último producto del inventario final del proceso.
    b) Si no es el último producto del inventario, pasa al siguiente producto y vuelve al paso 3.

### 3.9.2 Diagrama de Actividades del módulo 2: Reabastecimiento de inventario

**Descripción:** En el módulo 2 reabastecimiento de inventario, el usuario administrador/a tras haber realizado el módulo 1, puede verificar la existencia del *stock* de cada insumo y de ser requerido puede solicitar más *stock* del insumo faltante. Además, puede ingresar los gramos o unidades recibidos al *stock* del insumo en la aplicación.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 2 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 9 Diagrama de actividades del módulo 2: Reabastecimiento de inventario*




**Detalle del proceso**:

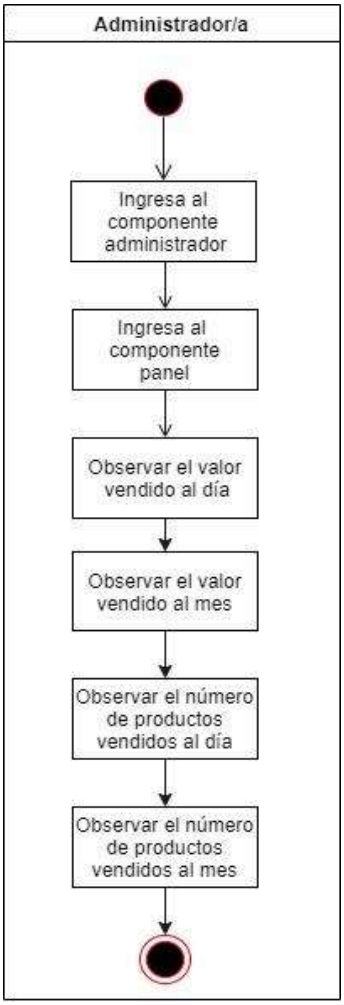
1. El usuario administrador/a ingresa a la aplicación al componente administrador.
2. El usuario administrador/a ingresa al componente inventario.
3. El usuario administrador/a Observa el *stock* del insumo actual en el inventario.
4. El usuario administrador/a verifica si el *stock* del insumo es suficiente.
    a. Si el *stock* es suficiente
        i. El usuario administrador/a verifica si es el último producto del inventario
            1. Si es el último producto del inventario fin del proceso.
            2. Si no es el último producto del inventario, pasa al siguiente producto y vuelve al paso 3.
    b. Si el *stock* no es suficiente
        i. El usuario administrador/a contacta con el proveedor y solicita más *stock* del insumo
        ii. El usuario administrador/a recibe el producto
        iii. El usuario administrador/a ingresa el nuevo *stock* dentro de la aplicación y vuelve al paso 4.

### 3.9.3 Diagrama de Actividades del módulo 3: Control de ventas

**Descripción:** En el módulo 3 control de ventas, el usuario administrador puede verificar el valor total vendido en el día y el mes actuales. Además, puede verificar la cantidad de productos vendidos en el día y mes actuales.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 3 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 10 Diagrama de actividades del módulo 3: Control de ventas*




**Detalle del proceso**:

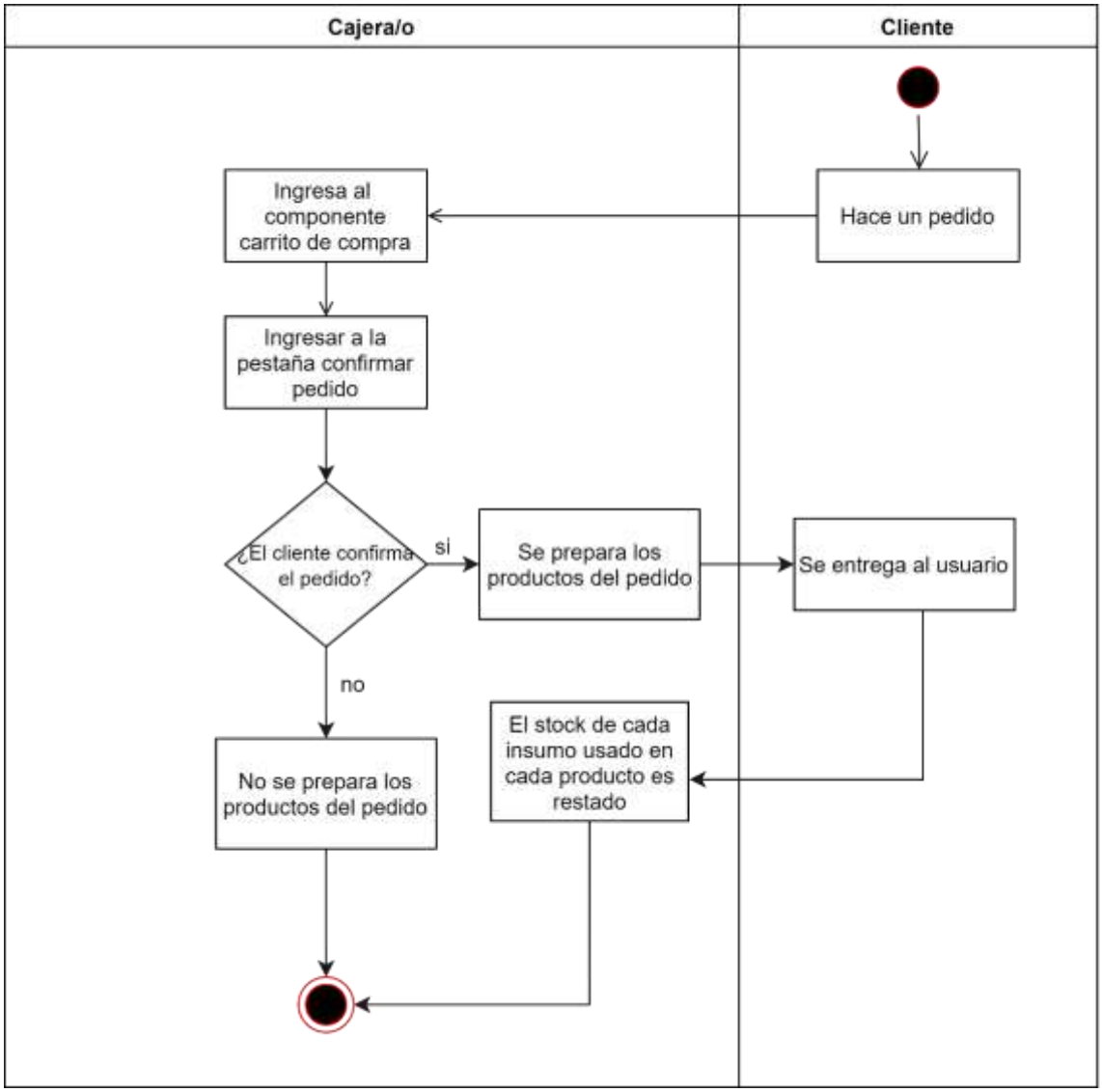
1. El usuario administrador/a ingresa a la aplicación al componente administrador.
2. El usuario administrador/a ingresa al componente panel.
3. El usuario administrador/a observa el valor vendido al día.
4. El usuario administrador/a observa el valor vendido al mes.

### 3.9.4 Diagrama de Actividades del módulo 4: Lista de pedidos

**Descripción:** En el módulo 4 lista de pedidos, el usuario cajero puede verificar los productos seleccionados y prepararlos para su entrega. Además, podrá confirmar el pedido lo que restará el stock gastado de cada insumo en los productos producidos.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 4 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 11 Diagrama de actividades del módulo 4: Lista de pedidos*




**Detalle del proceso**:

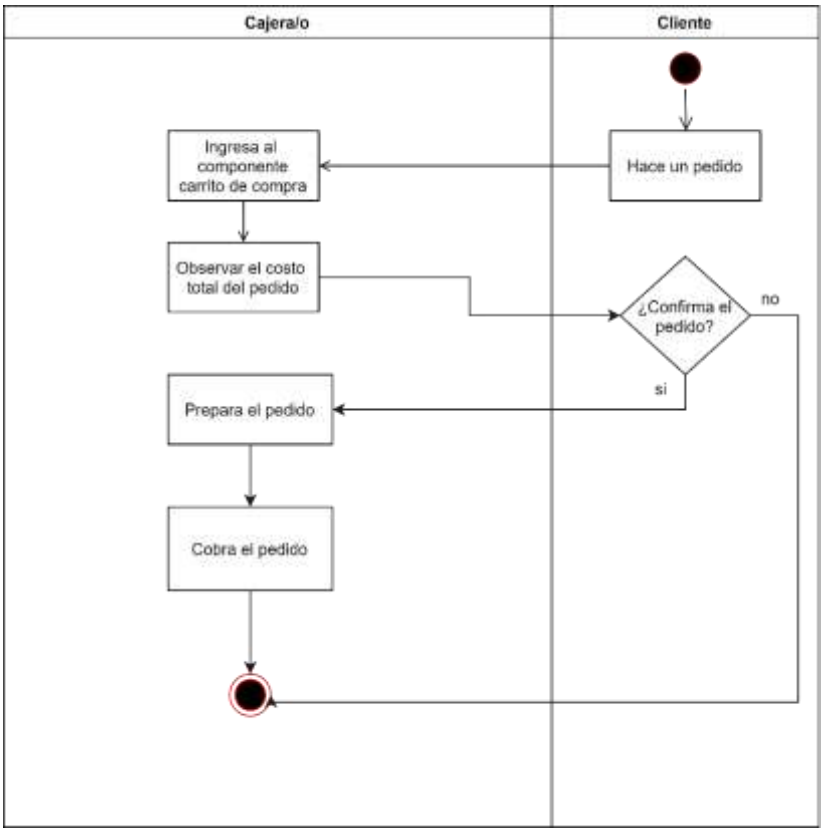
1. El usuario cliente hace un pedido.
2. El usuario cajera/o ingresa al componente carrito de compra.
3. El usuario cajera/o ingresa a la pestaña de confirmar pedido.
4. El usuario cajera/o confirma el pedido.
    a. Si el usuario cliente confirma el pedido
        i. El usuario cajera/o prepara los productos del pedido
        ii. El usuario cajera/o entrega el pedido al usuario cliente
        iii. El usuario cajera/o reduce el *stock* de cada insumo usado de la aplicación
    b. Si el usuario cliente no confirma el pedido
        i. El usuario cajera/o no prepara los productos del pedido

### 3.9.5 Diagrama de Actividades del módulo 5: Cobro de pedidos

**Descripción:** En el módulo 5 cobro de pedidos, el usuario cajero puede verificar el precio del pedido del cliente y podrá cobrar dicho precio en la caja.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 5 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 12 Diagrama de actividades del módulo 5: Cobro de pedidos*




**Detalle del proceso**:

1. El usuario cliente hace un pedido.
2. El usuario cajera/o ingresa al componente carrito de compra.
3. El usuario cajera/o ingresa a la pestaña de confirmar pedido.
4. El usuario cajera/o observa el costo total del pedido.
5. El usuario cliente confirma el pedido
    a. Si el usuario cliente confirma el pedido
        i. El usuario cajera/o prepara los productos del pedido.
        ii. El usuario cajera/o entrega el pedido al usuario cliente.
        iii. El usuario cajera/o cobra el pedido del usuario cliente.
    b. Si el usuario cliente no confirma el pedido
        i. El usuario cajera/o no prepara los productos del pedido.

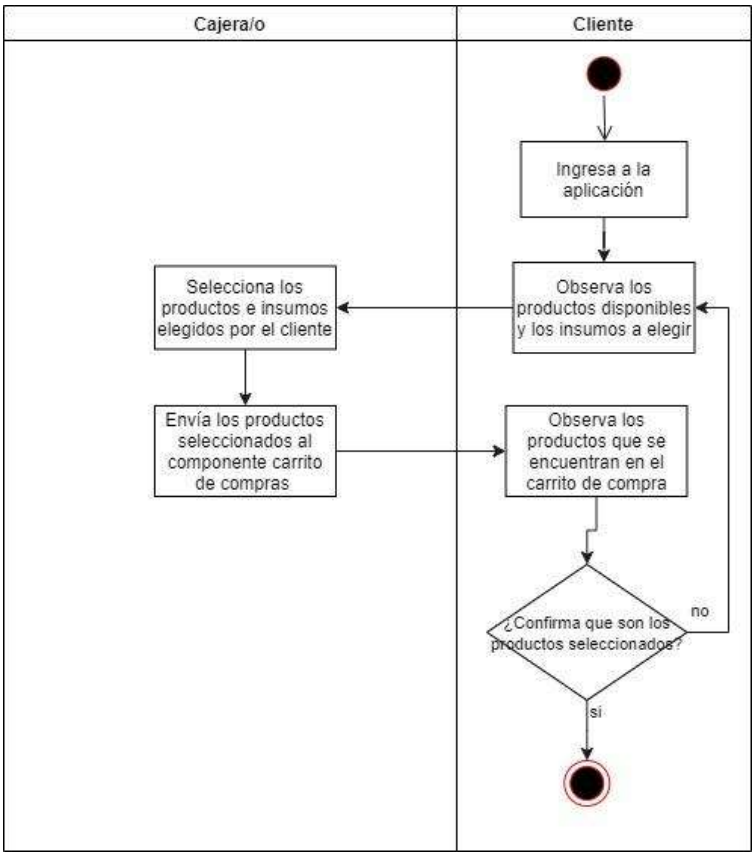
### 3.9.6 Diagrama de Actividades del módulo 6: Pedido de cliente

**Descripción:** En el módulo 6 pedido de cliente, el usuario cajero toma el pedido del usuario cliente, el usuario cajero busca los productos con los insumos solicitados en la

aplicación. El pedido es enviado al componente carrito de compras donde estarán todos los productos del pedido junto con los insumos que los componen.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 6 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 13 Diagrama de actividades del módulo 6: Pedido de cliente*




**Detalle del proceso**:

1. El usuario cliente ingresa a la aplicación.
2. El usuario cliente observa los productos disponibles y los insumos a elegir.
3. El usuario cajera/o selecciona los productos e insumos elegidos por el cliente.
4. El usuario cajera/o envía los productos seleccionados al componente carrito de compras.
5. El usuario cliente observa en el componente carrito de compra si están sus productos seleccionados.
    a. Si el cliente confirma que los productos en el carrito de compras son los productos que eligió, entonces termina el proceso.

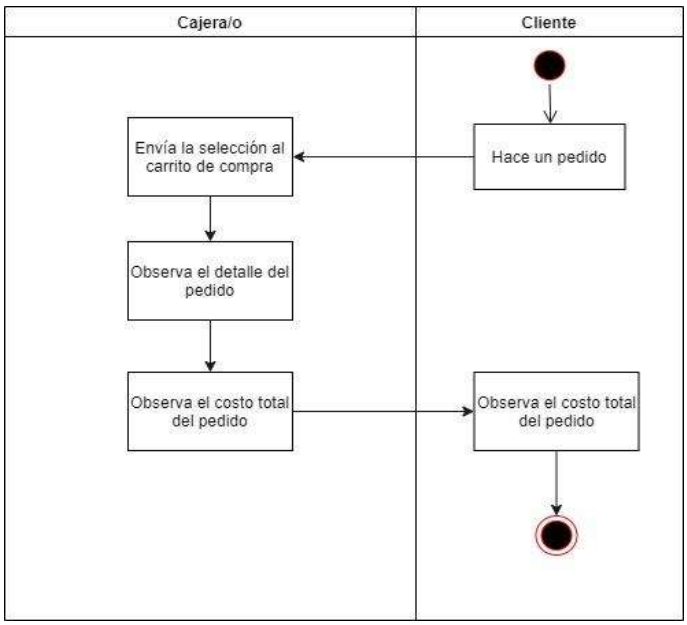
b. Si el cliente no confirma que los productos en el carrito de compras son los productos que eligió, entonces vuelve al paso 2.

### 3.9.7 Diagrama de Actividades del módulo 7: Cotización del pedido

**Descripción:** En el módulo 7 cotización del pedido, el usuario cajero toma el pedido del usuario cliente y lo envía al componente de carrito de compra. En el carrito de compra se detallan los productos y los insumos seleccionados por el usuario cliente. Además, puede observar el valor total del pedido.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 7 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 14 Diagrama de actividades del módulo 7: Cotización del pedido*




**Detalle del proceso**:

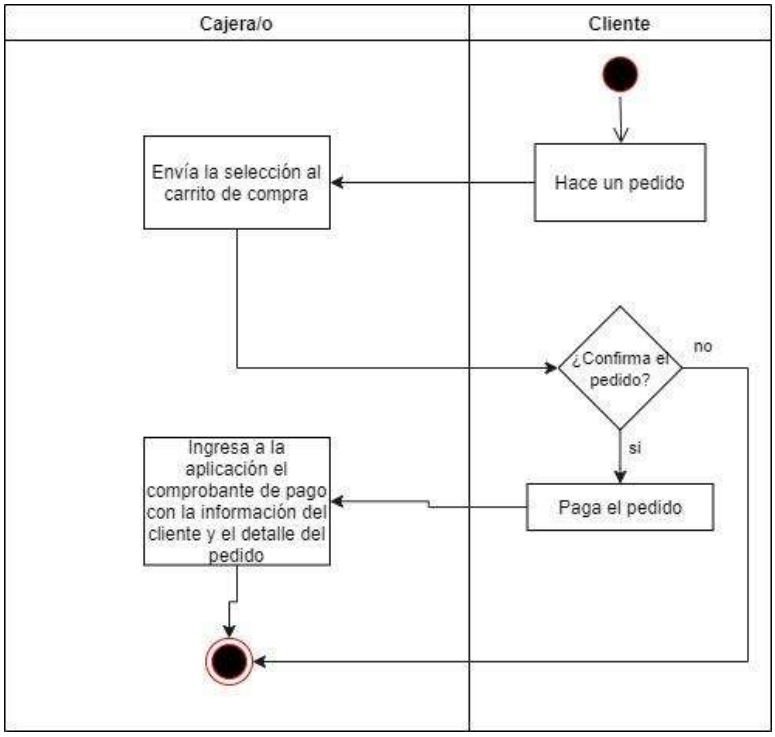
1. El usuario cliente hace un pedido.
2. El usuario cajera/o envía la selección al carrito de compra.
3. El usuario cajera/o observa el detalle del pedido.
4. El usuario cajera/o observa el costo total del pedido.
5. El usuario cajera/o muestra al usuario cliente el costo total del pedido.

### 3.9.8 Diagrama de Actividades del módulo 8: Pago del pedido

**Descripción:** En el módulo 8 pago del pedido, el usuario cliente confirma el pedido y lo cancela en efectivo, el usuario cajera/o ingresa el comprobante de pago al sistema con la información del usuario y el detalle de la compra.

**Diagrama de Actividades:** El diagrama de actividades representa la descripción de manera gráfica y muestra los procedimientos del módulo 8 por medio de la intervención de los usuarios en la aplicación.

*Ilustración 15 Diagrama de actividades del módulo 8: Pago del pedido*



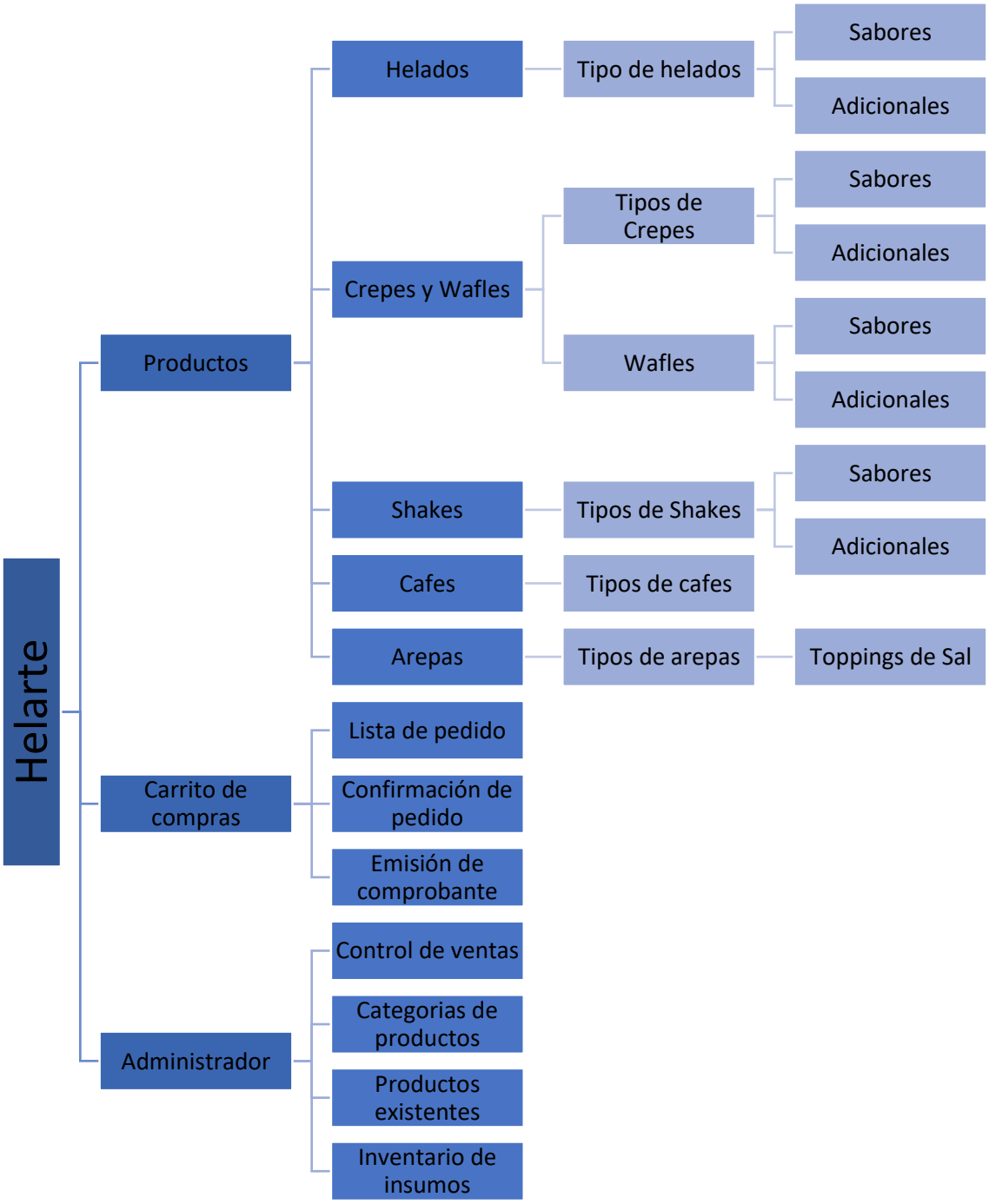

**Detalle del proceso**:

1. El usuario cliente hace un pedido.
2. El usuario cajera/o envía la selección al carrito de compra.
3. El usuario cajera/o pide una confirmación del pedido.
    a. Si el usuario cliente confirma el pedido, entonces paga el pedido
        i. El usuario cajera/o ingresa el comprobante de pago con la información del cliente y el detalle del pedido a la aplicación.
    b. Si el usuario no confirma el pedido, termina el proceso.

### 3.10 Estructura de interfaces

La estructura de interfaces será la siguiente:

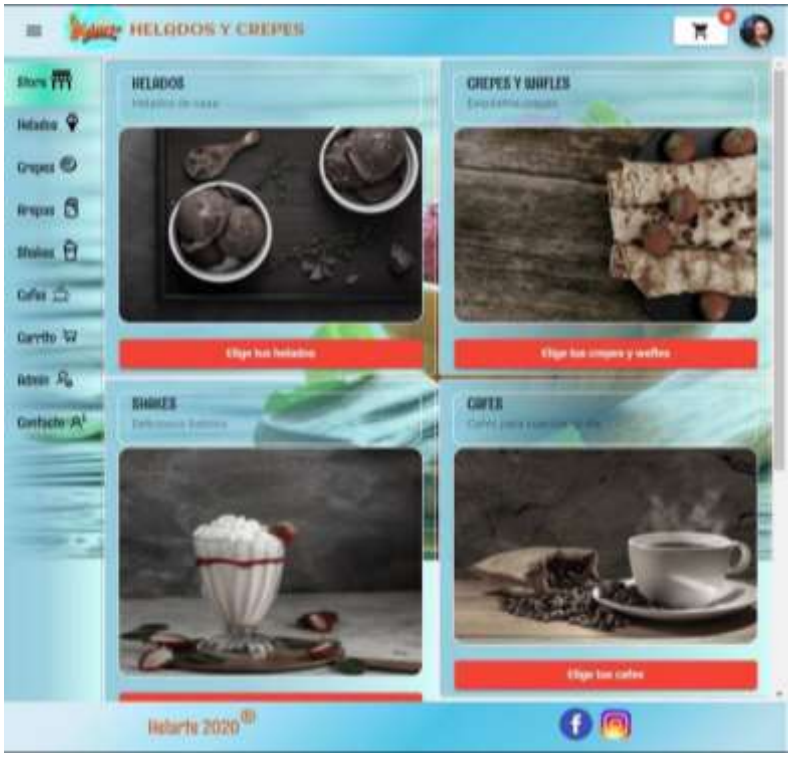
*Ilustración 16 Interfaz a desarrollar*



Se desarrolló las interfaces de la aplicación conforme al siguiente modelo:

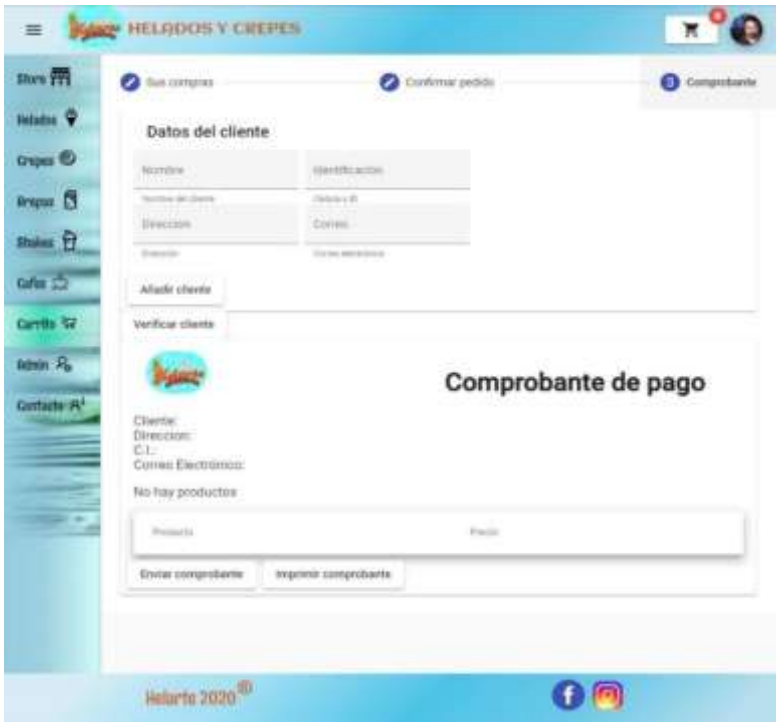
Productos

*Ilustración 17 Interfaz landing page de la aplicación*



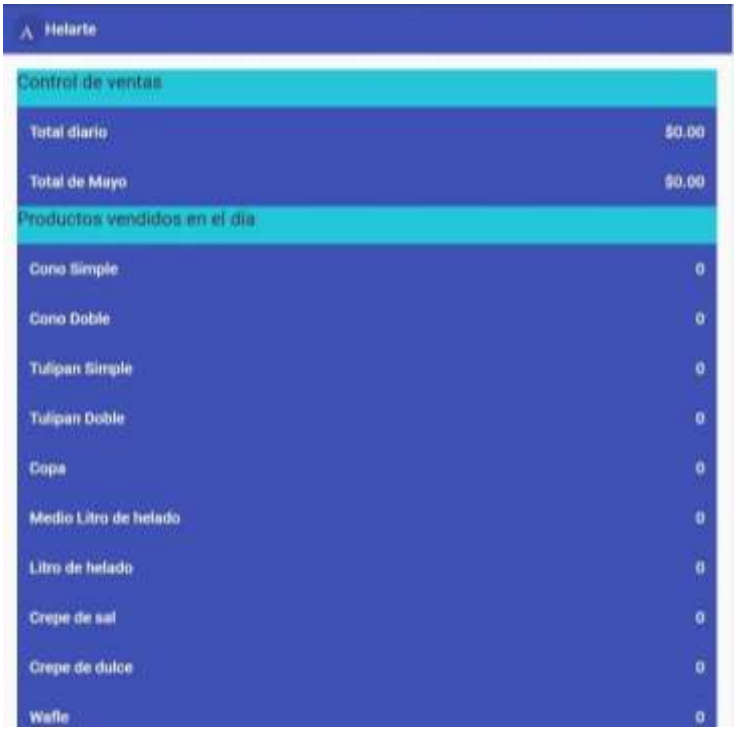

Carrito de compras

*Ilustración 18 Interfaz Carrito de Compras*




Administrador

*Ilustración 19 Interfaz de Administrador de la aplicación*



# CAPÍTULO 4: DESARROLLO Y CONTROL DE PROCESOS

## 4.1 Creación de base de datos en *firestore*

La creación de la base de datos en la *DBaaS* de *Firebase* (*firestore*). Se realizó en base al modelo de base de datos basado en documentos. Todos los procesos que involucran la información entre documentos son realizados por medio de transacciones en servicios programados dentro del *framework Angular*, lo que da una seguridad en el tratamiento de datos, ya que de presentarse errores en las transacciones los documentos no serían afectados.

El modelo fue generado con *Power Designer* y creado en la base de datos de *Firebase* (*firestore*).

## 4.2 Desarrollo de las interfaces gráficas *UX* y *UI*

Las interfaces fueron diseñadas previamente por los usuarios, tomando en cuenta la usabilidad, portabilidad y simplicidad en el manejo de la aplicación. El diseño de las interfaces ayuda a minimizar los *clics* necesarios para cumplir con las transacciones y con el cambio de pantallas.

La interfaz de usuario (*UI*) se desarrolló con la creación de un menú lateral proveyendo de navegación global dentro del componente de productos mientras que en la barra superior permite el acceso al componente carrito de compra. El acceso al componente de administrador se encuentra en el menú lateral y solo aparece si la aplicación reconoce el rol del usuario conectado como un rol de administrador.

La experiencia de usuario (*UX*) se muestra en la adaptabilidad de la aplicación a diferentes tamaños de pantallas, sin presentar problemas de interacción. Además, cuenta con la implementación de la tecnología *PWA* (*progressive web app* / aplicación de *web* progresiva), lo que permite la instalación de la aplicación dentro de los dispositivos como si fueran aplicaciones desarrolladas nativamente para el *SO* utilizado.

## 4.3 Desarrollo de *Sprints*

En el Desarrollo de *Sprint* se detalla la planificación de cada ciclo de desarrollo tomando en cuenta los puntos estimados de prioridad de cada historia de usuario para determinar las historias de usuario que serán desarrolladas en cada *sprint* junto con el tiempo requerido para cada uno. El presente proyecto será desarrollado en cuatro *Sprint* dando como resultado final la aplicación *web* junto con la aplicación móvil para el control de bares y restaurantes con el caso específico de la heladería helarte.

### 4.3.1 Primer *Sprint*

Durante el primer *sprint* de desarrollo se acuerda el desarrollo de los módulos 6: pedido de cliente, módulo número 7: cotización de pedido. El primer *Sprint* detalla la lista de módulos, además, de las etapas del proceso de cada módulo, necesarios para desarrollarlos. Además, se incluyó las interfaces de la aplicación que representan la usabilidad de los módulos 6 y 7.

#### 4.3.1.1 *Sprint Backlog*

Dentro del *sprint backlog* se analiza los módulos que se trabajaron durante el primer *sprint*, además de las etapas del proceso del módulo, necesarios para el correcto desarrollo de la aplicación.

*Tabla 16 Product Backlog del primer sprint*

| ***Product Backlog*** | | | | | |
|---|---|---|---|---|---|
| **ID** | Módulo | Usuario | Puntos estimados de prioridad /100 puntos | Como comprobar | Notas |
| **6** | Pedido de cliente | Cliente | 70 | Entrar en la aplicación, seleccionar el tipo de producto que se quiere solicitar, seleccionar el producto de la lista observable, ingresar al carrito de compras y verificar la entrada del producto en el módulo de pedidos. | |
| **7** | Cotización de pedido | Cliente | 75 | Entrar en la aplicación, ingresar al módulo de carrito de compra, verificar los productos solicitados, observar el costo total de todos los productos del pedido, aceptar el costo, acercarse a la barra de entrega. | |

#### 4.3.1.1.1 Módulo 6 Pedido de cliente.

*Tabla 17 Sprint Backlog del módulo 6 Pedido de cliente*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **6.1** | Ver lista de tipo de productos | Cliente | Ingresar a la *landing page*, observar los tipos de productos | Este visto en la ilustración 20 |
| **6.2** | Ver lista de productos | Cliente | Ingresar al tipo de producto requerido y observar la lista de productos | Este visto en la ilustración 21 |
| **6.3** | Seleccionar un producto | Cliente | Seleccionar el producto a solicitar | Este visto en la ilustración 22 |
| **6.4** | Ver lista de insumos | Cliente | Ingresar a las diferentes ventanas de los insumos que forman el producto | Este visto en la ilustración 23 |
| **6.5** | Seleccionar los insumos | Cajero | Seleccionar los insumos de los productos seleccionados | Este visto en la ilustración 24 |
| **6.6** | Agregar producto al carrito de compras | Cajero | Ingresar el producto con los insumos seleccionados al carrito de compras por medio de la aplicación. | Este visto en la ilustración 25 |

*Ilustración 20 Proceso 6.1 Lista de tipo de producto*

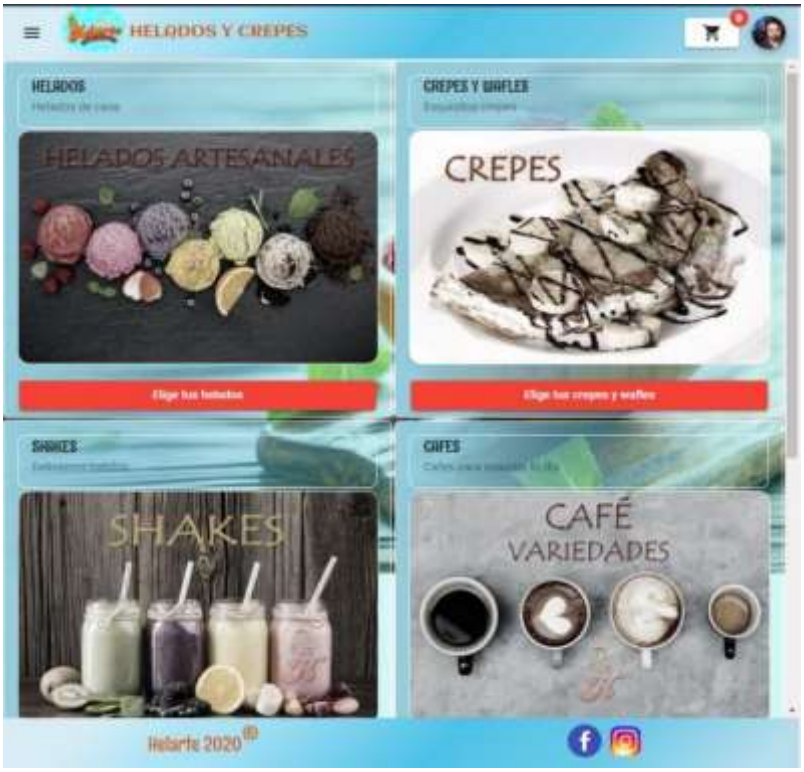


*Ilustración 21 Proceso 6.2 Lista de productos*

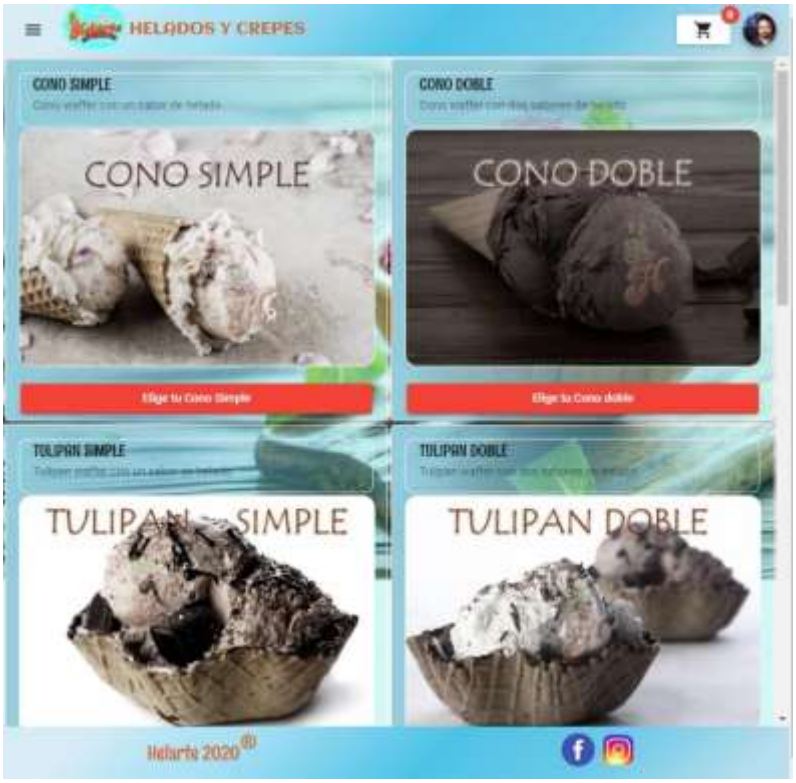



*Ilustración 22 Proceso 6.3 Producto seleccionado*

*Ilustración 23 Proceso 6.4 Ver lista de insumos*

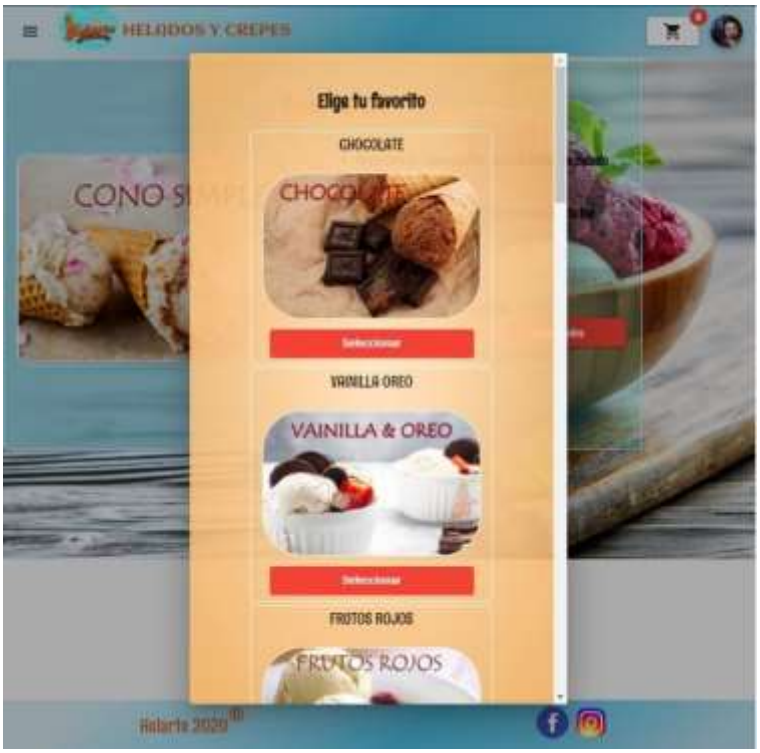



*Ilustración 24 Proceso 6.5 Seleccionar insumos*

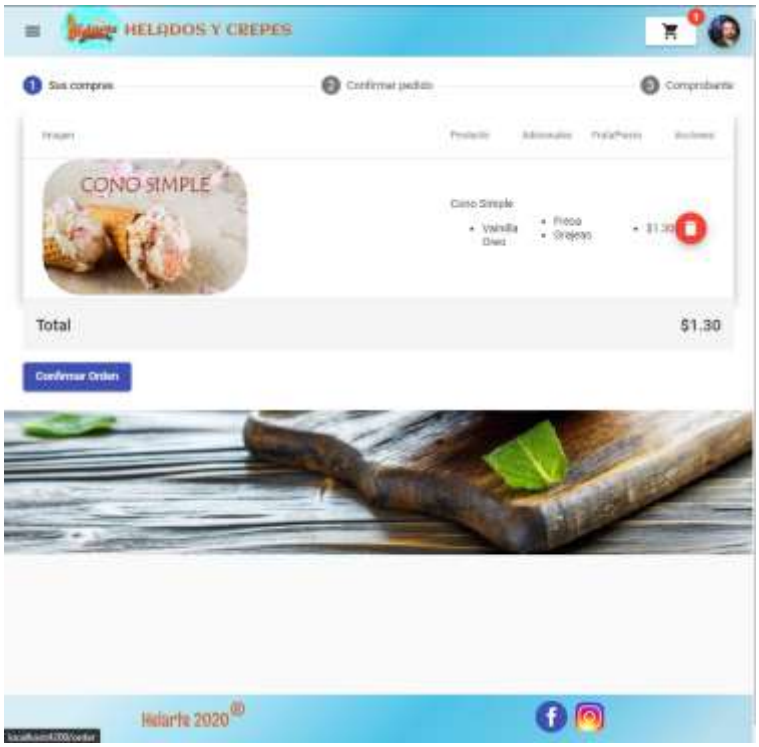
*Ilustración 25 Proceso 6.6 Agregar producto al carrito de compras*


### 4.3.1.1.2 Módulo 7 Cotización de pedido

*Tabla 18 Sprint Backlog del Módulo 7 Cotización de pedido*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **7.1** | Ingresa al componente de carrito de compra | Cajera/o | Entrar en la aplicación, loguearse con una cuenta de cajero, ingresar al componente carrito de compras | Este visto en la ilustración 26 |
| **7.2** | Ver lista de productos ingresados en el carrito de compras | Cajera/o | Observar los productos dentro de la primera pestaña del carrito de compras | Este visto en la ilustración 26 |
| **7.3** | Solicitar confirmación de pedido | Cajera/o | Solicitar una confirmación de todos los productos del pedido | Este visto en la ilustración 26 |
| **7.4** | Entregar el valor a cancelar por el pedido | Cajera/o | Se comunica el valor total del pedido a pagar | Este visto en la ilustración 26 |

*Ilustración 26 Proceso Módulo 7 Carrito de compra - Cotización de pedido*




### 4.3.2 Segundo *Sprint*

Durante el segundo *sprint* de desarrollo se acuerda el desarrollo de los módulos 4: lista de pedido y el módulo número 5: cobro de pedido. El segundo *Sprint* detalla la lista de módulos, además, de las etapas del proceso de cada módulo, necesarios para desarrollarlos. Además, se incluyó las interfaces de la aplicación que representan la usabilidad de los módulos 4 y 5.

#### 4.3.2.1 *Sprint Backlog*

Dentro del *sprint backlog* se analiza los módulos que se trabajaron durante el segundo *sprint*, además de las etapas del proceso del módulo, necesarios para el correcto desarrollo de la aplicación.

*Tabla 19 Product Backlog del segundo sprint*

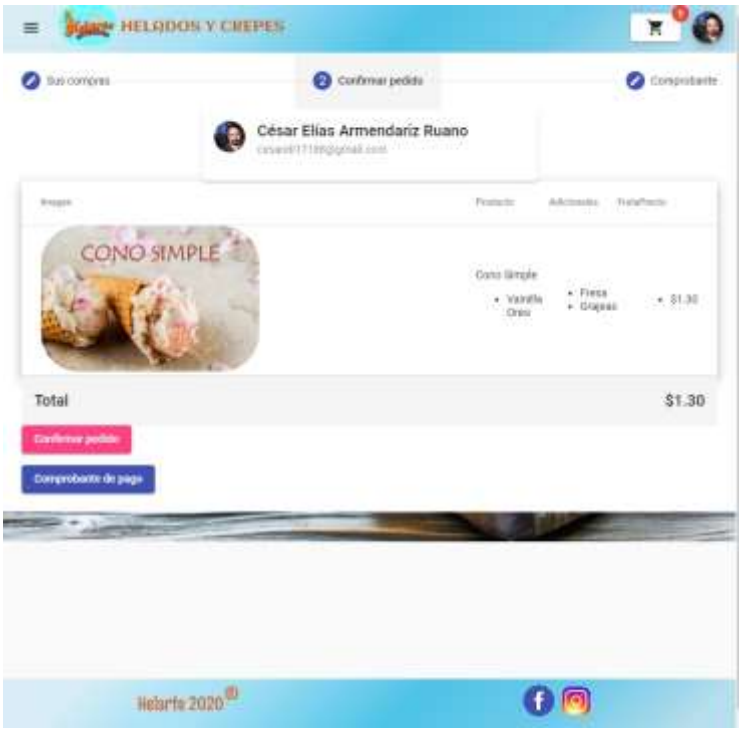
| ***Product Backlog*** | | | | | |
|---|---|---|---|---|---|
| **ID** | Módulo | Usuario | Puntos estimados de prioridad /100 puntos | Como comprobar | Notas |
| **4** | Lista de pedido | Cliente | 75 | Entrar en la aplicación, *loguearse* con una cuenta de cajero, seleccionar el tipo de producto que se quiere solicitar, seleccionar el producto de la lista observable, ingresar al carrito de compras y verificar la entrada del producto en el módulo de pedidos. | |
| **5** | Cobro de pedido | Cliente | 85 | Entrar en la aplicación, *loguearse* con una cuenta de cajero, Ingresar al módulo de carrito de compra, verificar los productos solicitados, observar el costo total de todos los productos del pedido, aceptar el costo, acercarse a la barra de entrega. | |

#### 4.3.2.1.1 Módulo 4 Lista de pedido

*Tabla 20 Sprint Backlog del Módulo 4 Lista de pedido*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **4.1** | Ingresa al componente de carrito de compra | Cajera/o | Entrar en la aplicación, *loguearse* con una cuenta de cajero, ingresar al componente carrito de compras. | Este visto en la ilustración 26 |
| **4.2** | Ingresar a la pestaña de confirmar pedido | Cajera/o | Ingresar a la pestaña de confirmar pedido dentro del componente carrito de compras. | Este visto en la ilustración 27 |
| **4.3** | Observar la lista de pedido del cliente | Cajera/o | Observar la lista confirmada de productos | Este visto en la ilustración 27 |
| **4.4** | Confirmar pedido | Cajera/o | Confirmar el pedido con la aplicación para reducir el *stock* de los insumos de los productos en la lista. | Este visto en la ilustración 27 |

*Ilustración 27 Proceso Módulo 4 Lista de pedido*



#### 4.3.2.1.2 Módulo 5 Cobro de pedido

*Tabla 21 Sprint Backlog del Módulo 5 Cobro de pedido*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **5.1** | Ingresa al componente de carrito de compra | Cajera/o | Ingresar a la *landing page* de la aplicación, Entrar en la aplicación, *loguearse* con una cuenta de cajero. Ingresar al componente carrito de compras. | Este visto en la ilustración 26 |
| **5.2** | Ingresar a la pestaña de confirmar pedido | Cajera/o | Ingresar a la pestaña de confirmar pedido dentro del componente carrito de compras. | Este visto en la ilustración 27 |
| **5.3** | Confirmar pedido | Cajera/o | Confirmar el pedido con la aplicación para reducir el *stock* de los insumos de los productos en la lista. | Este visto en la ilustración 27 |
| **5.4** | Cobrar pedido | Cajera/o | Cobrar el pedido del cliente en la caja. | |

### 4.3.3 Tercer *Sprint*

Durante el tercer *sprint* de desarrollo se acuerda el desarrollo de los módulos 1: Control de inventario, el módulo 2: Reabastecimiento de inventario y el módulo número 3: Control de ventas. El tercer *sprint* detalla la lista de módulos, además, de las etapas del proceso de cada módulo, necesarios para desarrollarlos. Además, se incluyó las interfaces de la aplicación que representan la usabilidad de los módulos 1,2 y 3.

#### 4.3.3.1 *Sprint Backlog*

Dentro del *sprint backlog* se analiza los módulos que se trabajaron durante el tercer *sprint*, además de las etapas del proceso del módulo, necesarios para el correcto desarrollo de la aplicación.

*Tabla 22 Product Backlog del tercer Sprint*

| ***Product Backlog*** | | | | | |
|---|---|---|---|---|---|
| **ID** | Módulo | Usuario | Puntos estimados de prioridad /100 puntos | Como comprobar | Notas |
| **1** | Control de Inventario | Administrador | 90 | Entrar en la aplicación, *loguearse* con una cuenta de administrador. Ingresar al componente Administrador. Entrar en el componente de Inventario, Observar el Inventario actual | |
| **2** | Reabastecimiento de inventario | Administrador | 60 | Entrar en la aplicación, *loguearse* con una cuenta de administrador. Ingresar al componente Administrador. Entrar en el componente de Inventario, ingresar el nuevo *stock* por medio de la aplicación | |
| **3** | Control de ventas | Administrador | 70 | Entrar en la aplicación, *loguearse* con una cuenta de administrador. Ingresar al componente Administrador. Entrar en el componente de control de ventas, observar el valor vendido en el día y en el mes en curso, además, se puede observar el número de productos vendidos en el día y mes vigentes. | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | Ingresar en el componente de Productos y verificar la información de los productos a la venta. | |

#### 4.3.3.1.1 Módulo 1 Control de Inventario

*Tabla 23 Sprint Backlog del Módulo 1 Control de Inventario*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **1.1** | *Loguearse* como administrador/a | Administrador/a | Ingresar a la *landing* page, *loguearse* por medio de la aplicación | Este visto en la ilustración 28 |
| **1.2** | Ingresar al componente administrador/a | Administrador/a | Ingresar al menú lateral y seleccionar la opción Admin | Este visto en la ilustración 29 |
| **1.3** | Ingresar al componente Inventario | Administrador/a | Mediante el menú lateral ingresar al componente Inventario | Este visto en la ilustración 30 |
| **1.4** | Observar el *stock* de cada insumo en el inventario | Administrador/a | Observar el *stock* actual de cada insumo | Este visto en la ilustración 31 |

*Ilustración 28 Proceso 1.1 Loguearse como administrador/a*

*Ilustración 29 Proceso 1.2 Ingresar al componente administrador/a*

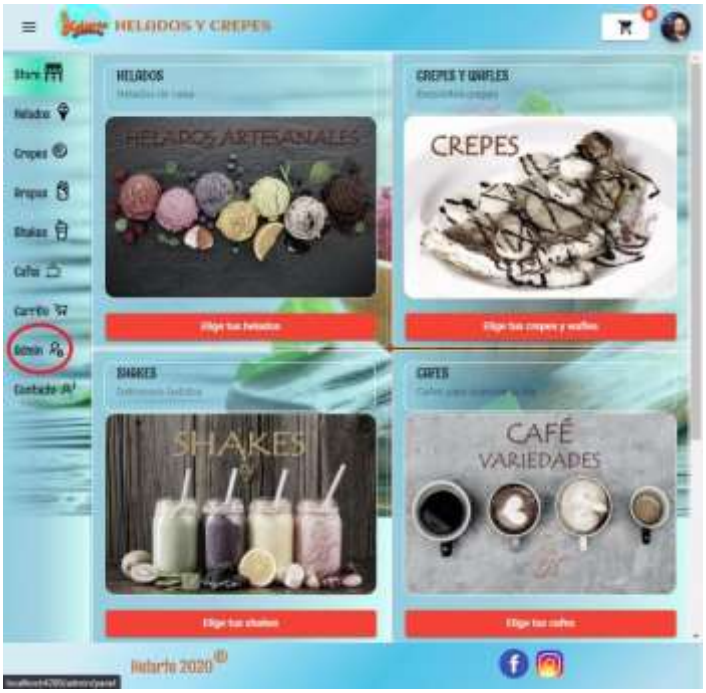



*Ilustración 30 Proceso 1.3 Ingresar al componente Inventario*

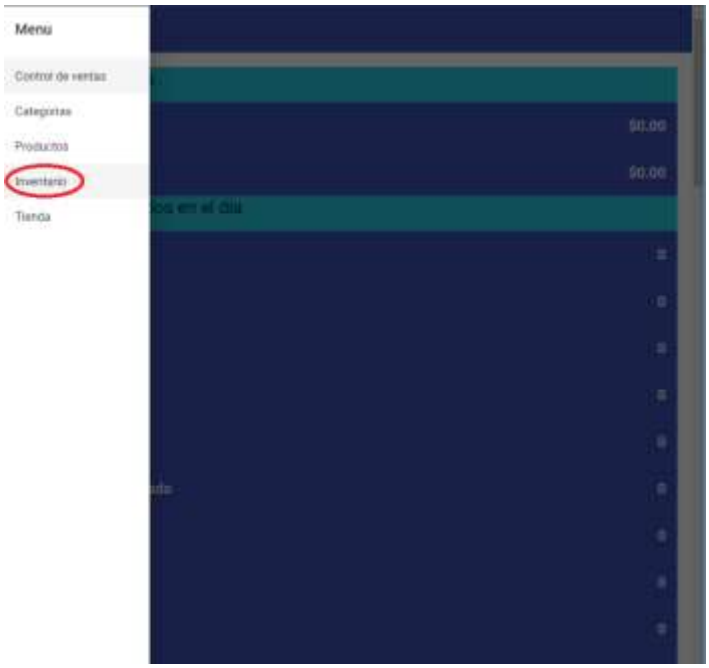
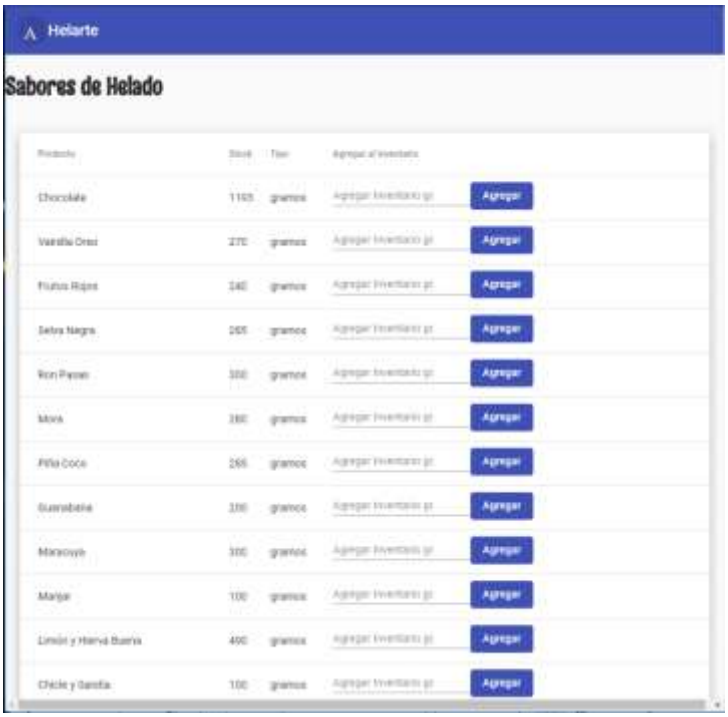


*Ilustración 31 Proceso 1.4 Observar el stock de cada insumo en el inventario*



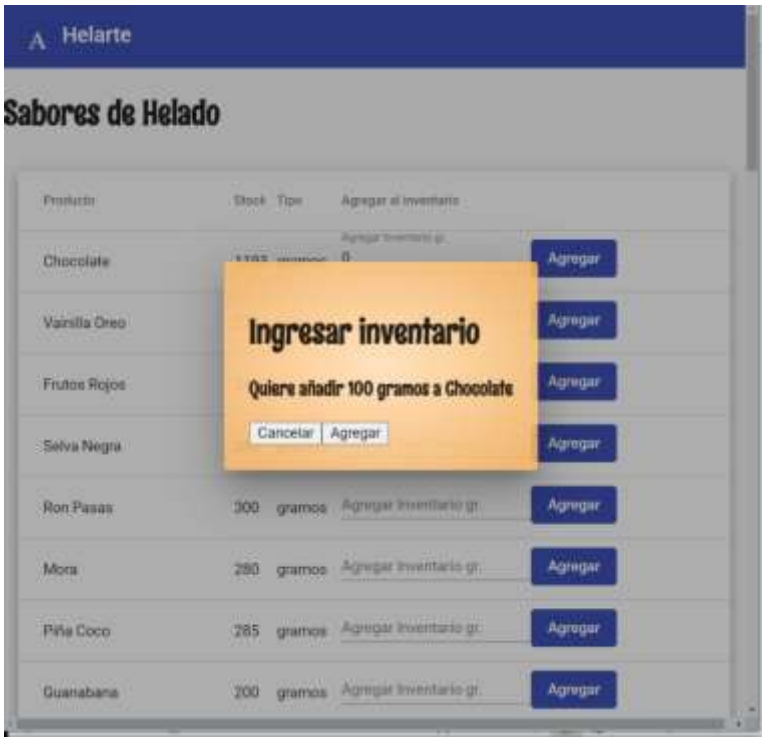

### 4.3.3.1.2 Módulo 2 Reabastecimiento de inventario

*Tabla 24 Sprint Backlog Módulo 2 Reabastecimiento de inventario*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **2.1** | *Loguearse* como administrador/a | Administrador/a | Ingresar a la *landing page*, *loguearse* por medio de la aplicación | Este visto en la ilustración 28 |
| **2.2** | Ingresar al componente administrador/a | Administrador/a | Ingresar al menú lateral y seleccionar la opción Admin | Este visto en la ilustración 29 |
| **2.3** | Ingresar al componente Inventario | Administrador/a | Mediante el menú lateral ingresar al componente Inventario | Este visto en la ilustración 30 |
| **2.4** | Ingresar el *stock* en el insumo recibido | Administrador/a | Ingresar el *stock* nuevo en cada insumo recibido | Este visto en la ilustración 32 |

Siguiendo las ilustraciones 28, 29, 30 y 31 el reabastecimiento de los insumos se realizaría mediante el formulario del lado derecho del componente Inventario.

*Ilustración 32 Proceso 2.4 Ingresar el stock en el insumo recibido*




### 4.3.3.1.3 Módulo 3 Control de ventas

*Tabla 25 Sprint Backlog módulo 3 Control de ventas*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **3.1** | *Loguearse* como administrador/a | Administrador/a | Ingresar a la *landing page*, *loguearse* por medio de la aplicación | Este visto en la ilustración 28 |
| **3.2** | Ingresar al componente administrador/a | Administrador/a | Ingresar al menú lateral y seleccionar la opción Admin | Este visto en la ilustración 29 |
| **3.3** | Ingresar al componente Control de ventas | Administrador/a | Mediante el menú lateral ingresar al componente Control de ventas | Este visto en la ilustración 33 |
| **3.4** | Observar el valor total de la venta del día y mes vigente | Administrador/a | Observar el valor total de la venta en el día y mes vigente. | Este visto en la ilustración 34 |

| | | | | |
|---|---|---|---|---|
| **3.5** | Observa el número de ventas totales de cada producto del día y del mes vigente | Administrador/a | Observar el número total de los productos vendidos en el día y mes vigente. | Este visto en la ilustración 34 |
| **3.6** | Ingresar al componente Productos | Administrador/a | Ingresar al menú lateral y seleccionar la opción Productos | Este visto en la ilustración 35 |
| **3.7** | Observar los detalles de los productos en venta | Administrador/a | Observar los productos dentro del componente Productos | Este visto en la ilustración 36 |
| **3.8** | Editar la información del producto | Administrador/a | Ingresar al componente de edición del producto con el botón de edición y actualizar los datos del producto | Este visto en la ilustración 37 |
| **3.9** | Crear un nuevo producto | Administrador/a | Ingresar al componente de edición del producto con el botón de Añadir producto y crear un nuevo producto. | Este visto en la ilustración 38 |
| **3.10** | Eliminar un producto | Administrador/a | Eliminar un producto con el botón de eliminación en el componente producto | Este visto en la ilustración 36 (botón de eliminación) |

*Ilustración 33 Proceso 3.3 Ingresar al componente Control de ventas*

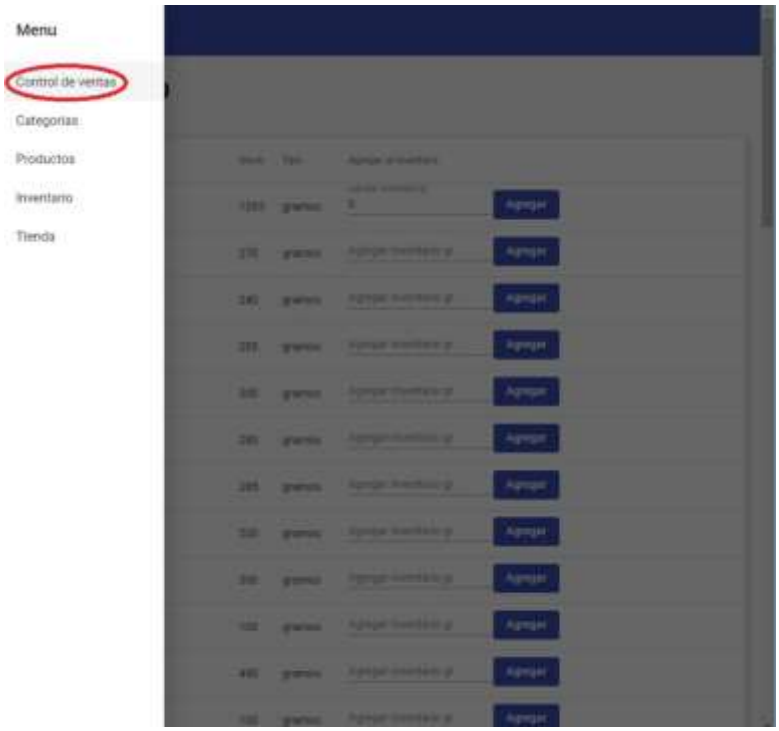


*Ilustración 34 Proceso 3.4 Observar el valor total de la venta del día y mes vigente*

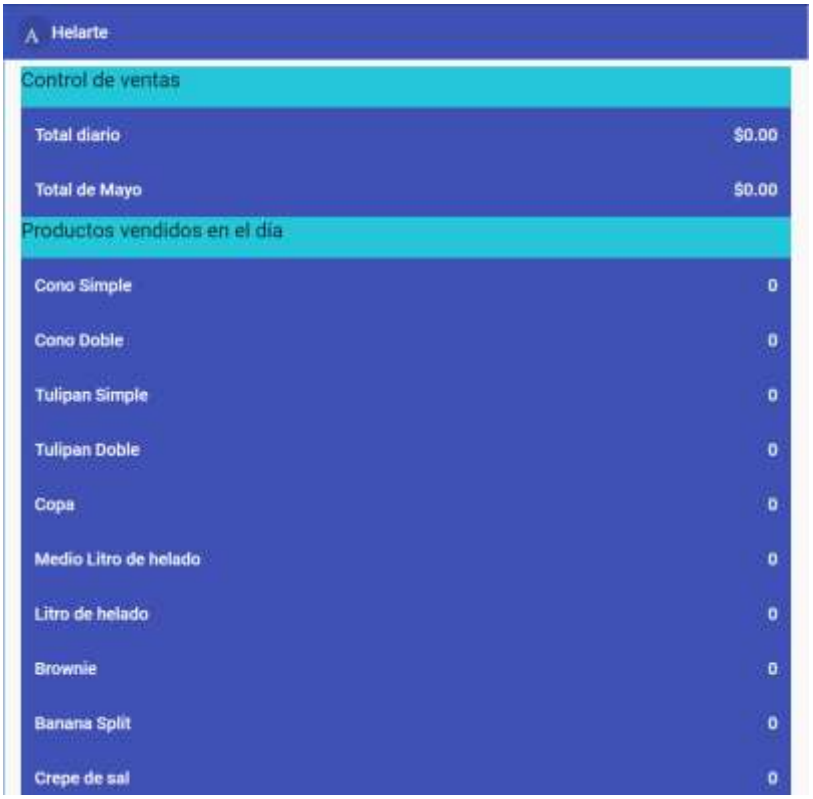



*Ilustración 35 Proceso 3.6 Ingresar al componente Productos*

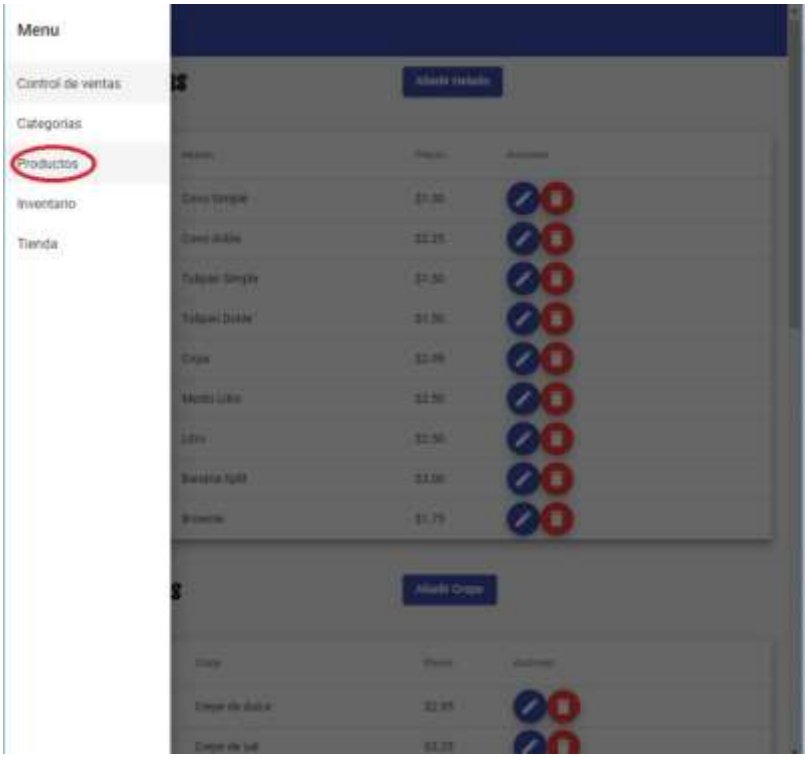


*Ilustración 36 Proceso 3.7 Observar los detalles de los productos en venta*

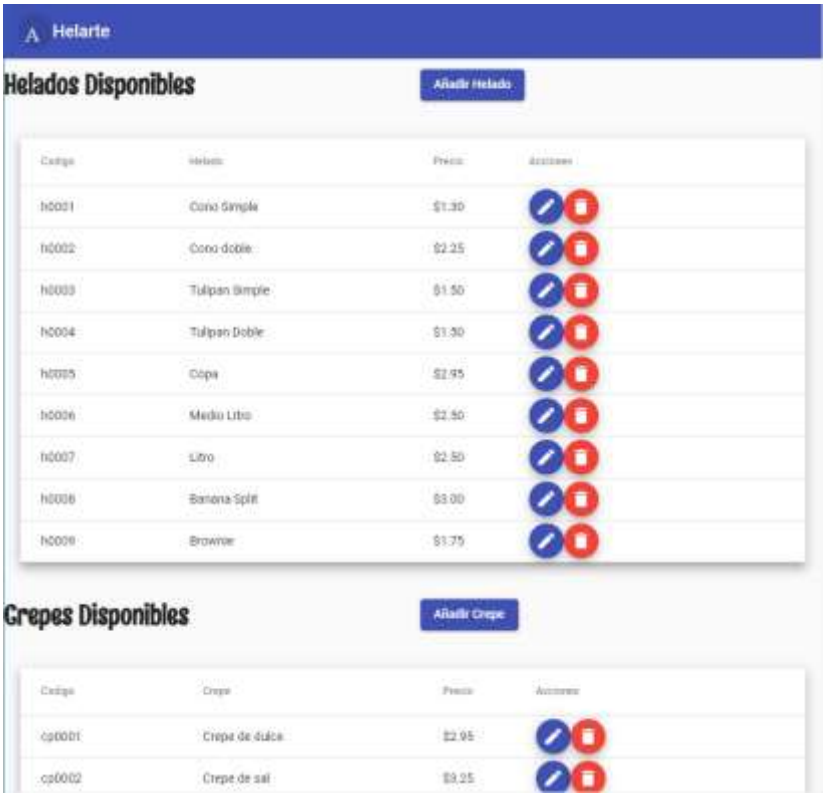


*Ilustración 37 Proceso 3.8 Editar la información del producto*

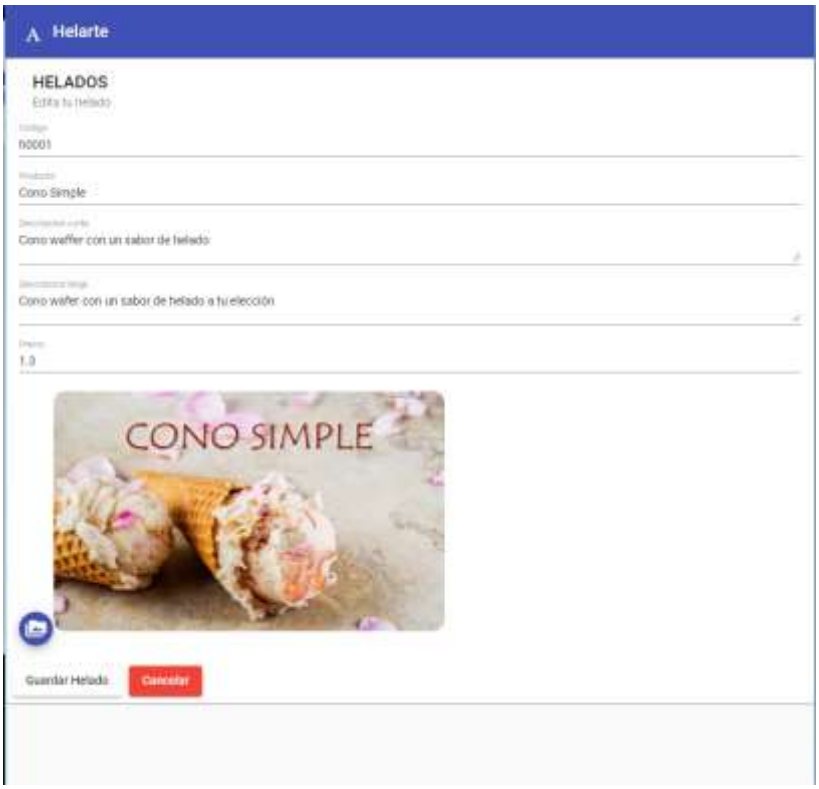
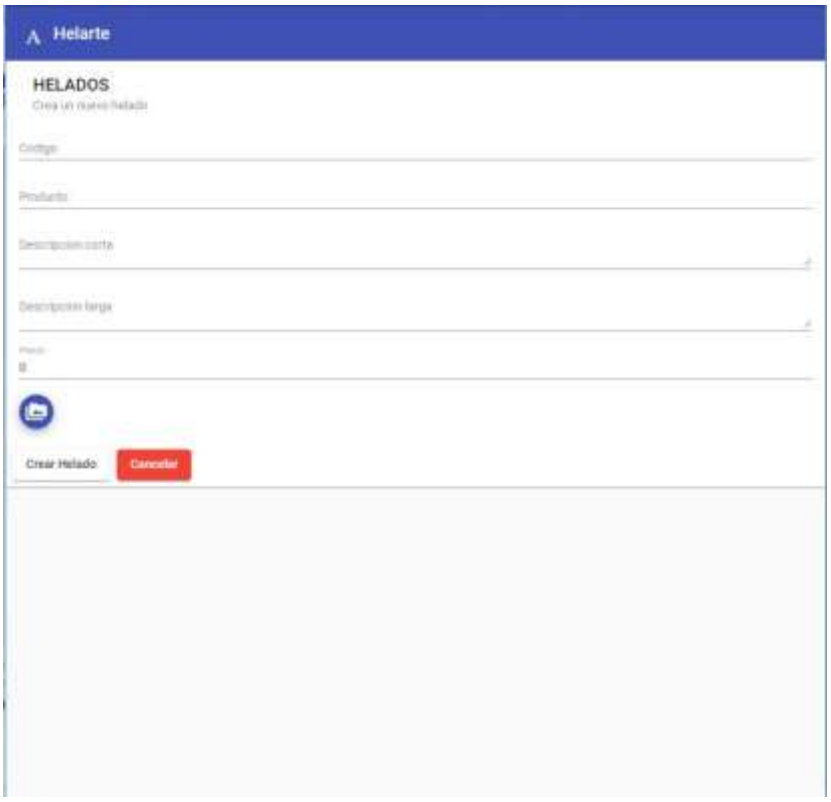


*Ilustración 38 Proceso 3.9 Crear un nuevo producto*




### 4.3.4 Cuarto *Sprint*

Durante el cuarto *sprint* de desarrollo se acuerda el desarrollo del módulo 8: Pago de pedido. El cuarto *sprint* detalla las etapas del proceso del módulo, necesarios para desarrollarlos. Además, se incluyó las interfaces de la aplicación que representan la usabilidad del módulo 8.

#### 4.3.4.1 *Sprint Backlog*

Dentro del *sprint backlog* se analiza los módulos que se trabajaron durante el cuarto *sprint*, además de las etapas del proceso del módulo, necesarios para el correcto desarrollo de la aplicación.

*Tabla 26 Product Backlog del cuarto sprint*

| ***Product Backlog*** | | | | | |
|---|---|---|---|---|---|
| **ID** | Módulo | Usuario | Puntos estimados de prioridad /100 puntos | Como comprobar | Notas |
| **8** | Pago de pedido | Cajera/o | 80 | Entrar en la aplicación, *loguearse* con una cuenta de cajero. Ingresar al componente carrito de compras. Entrar en la pestaña Comprobante. Ingresar la información del cliente. Ingresar el comprobante de pago en la base de datos de la aplicación y de ser requerido imprimir el comprobante de pago. | |

#### 4.3.4.1.1 Módulo 8 Pago de pedido

*Tabla 27 Sprint Backlog Módulo 8 Pago de pedido*

| ***Sprint Backlog*** | | | | |
|---|---|---|---|---|
| **ID** | Proceso | Usuario | Como comprobar | Notas |
| **8.1** | *Loguearse* como cajera/o | Cajera/o | Entrar en la aplicación, *loguearse* con una cuenta de cajero. | Este visto en la ilustración 28 |
| **8.2** | Ingresar al componente carrito de compra | Cajera/o | Ingresar en el componente carrito de compra mediante el menú lateral o desde el icono de carrito en la parte superior de la aplicación | Este visto en la ilustración 26 |
| **8.3** | Ingresar a la pestaña Comprobante de pago | Cajera/o | Dentro del componente carrito de compras ir a la pestaña de Comprobante | Este visto en la ilustración 39 |
| **8.4** | Ingresar los datos del cliente | Cajera/o | Ingresar los datos solicitados por la aplicación (opcional) | Este visto en la ilustración 40 |

| | | | | |
|---|---|---|---|---|
| **8.5** | Ingresar el comprobante de pago en la base de datos de la aplicación | Cajera/o | Mediante el botón de Enviar comprobante, ingresar el comprobante de pago dentro de la base de datos | Este visto en la ilustración 41 |
| **8.6** | Imprimir el comprobante de pago | Cajera/o | Si el cliente lo solicita se imprime el comprobante de pago y se le entrega al cliente por medio del botón imprimir comprobante | Este visto en la ilustración 42 |

*Ilustración 39 Proceso 8.3 Ingresar a la pestaña Comprobante de pago*

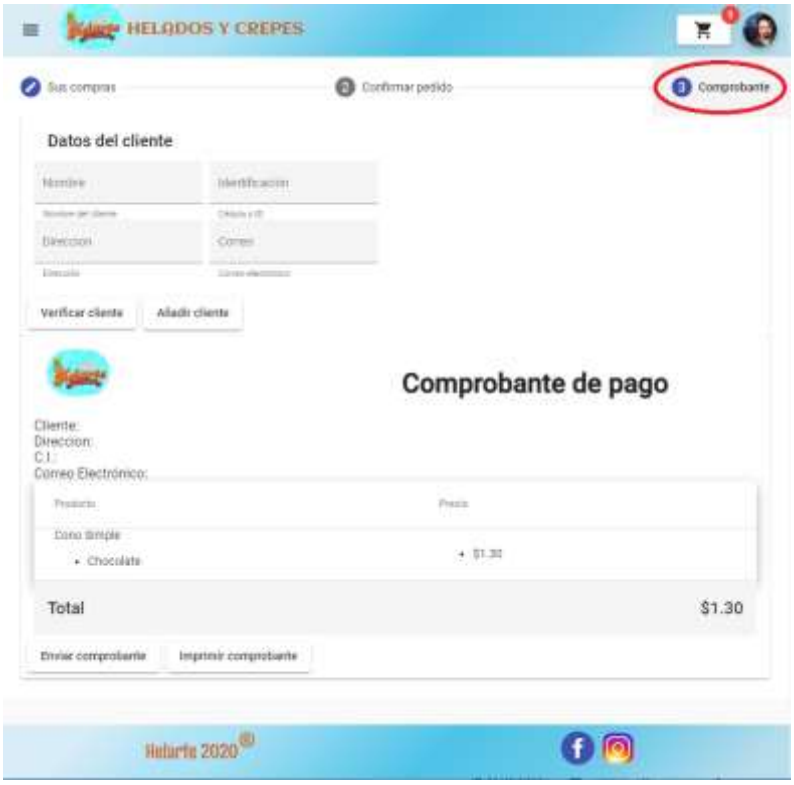


*Ilustración 40 Proceso 8.4 Ingresar los datos del cliente*

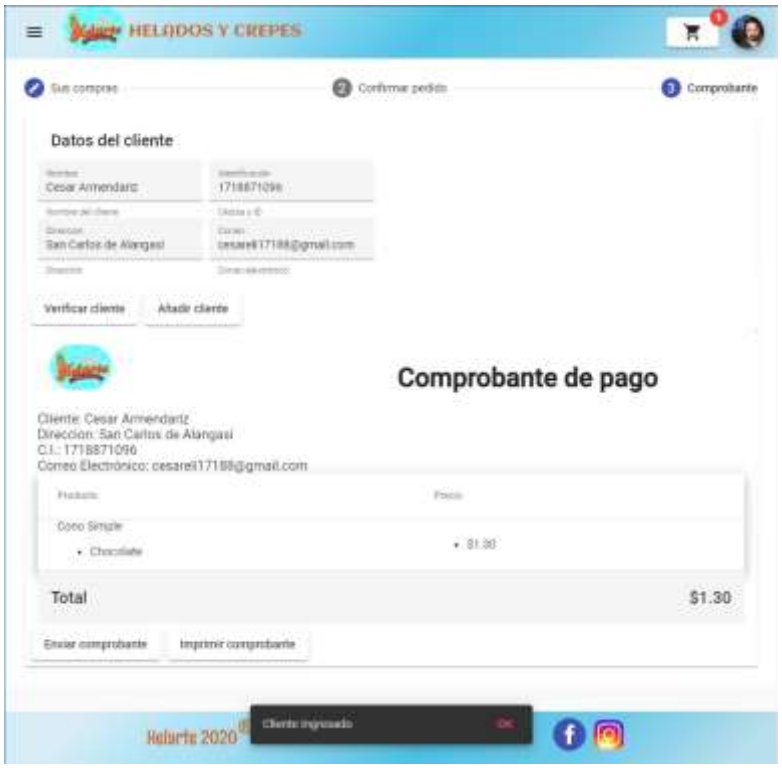



*Ilustración 41 Proceso 8.5 Ingresar el comprobante de pago en la base de datos de la aplicación*

*Ilustración 42 Proceso 8.6 Imprimir el comprobante de pago*


## CAPÍTULO 5: IMPLEMENTACIÓN Y PRUEBAS

### 5.1 Código y desarrollo

El código fue realizado siguiendo el patrón del modelo MVVM, utilizando el *framework Angular* con la versión 10.0, cuyo lenguaje principal es *Typescript*. Gracias a lo cual se implementó con varios complementos, servicios y componentes, entre las cuales se destacan:

- Módulos que pueden exportar un comportamiento y recibir el comportamiento de otros módulos
- Componentes que contienen la lógica de negocio, el archivo de maquetado y el archivo decorador, junto con un archivo para realizar pruebas unitarias.
- Servicios que funcionan como el componente que conecta la base de datos con la aplicación, transformando metadatos en datos legibles para la aplicación

- *RxJS* es una librería que simplifica el código asíncrono y conecta los servicios con los componentes en la lógica de negocio, haciendo más legible y rápido los datos recibidos.
- Guardianes que funcionan como *middlewares* ejecutables antes de cargar una ruta y determinar si se puede cargar dicha ruta o no.
- Routing Modules son módulos que permiten crear rutas de acceso conectando con los componentes que proveen del contenido al que se quiere acceder.

La interfaz gráfica se desarrolló utilizando la herramienta *Angular Material* en su versión 4.0, *framework* que es totalmente compatible con el *framework Angular* dándole múltiples características gráficas y elementos definidos a los archivos de marcado *HTML* que al acoplar con los archivos *SCSS*, que se encuentran en cada componente, da como resultado una interfaz gráfica interactiva y amigable al usuario.

Para la base de datos se utilizó la *DBaaS* de *Firebase* (*Firestore*) la cual se conecta a la aplicación mediante el *api angularfire* que permite:

- Leer, ingresar, modificar y eliminar colecciones de documentos y documentos de manera más nativa, ya que el *api* fue creado con la finalidad de interactuar con *Angular*.
- Autenticación de usuarios mediante el servicio de *firebase auth*, que permite crear usuarios mediante un *email* y contraseña o por medio de cuentas de *Google*, *Facebook*, *Twitter*, etc.

El servidor que aloja la aplicación es el de *Firebase* que permite entre otras funcionalidades darle seguridad de interacción en la web al usuario mediante el protocolo *HTTPS*. Además, permite la implementación de la tecnología *PWA* que da la oportunidad de instalar la aplicación en el Sistema operativo como si se tratara de una aplicación nativa del sistema operativo.

## 5.2 Pruebas de la aplicación

### 5.2.1 Módulo 1 Control de inventario

*Tabla 28 Pruebas de Módulo 2 Reabastecimiento de inventario*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar interactuar con el componente de administrador/a sin tener un usuario con rol administrador | El enlace de acceso al componente administrador no aparece, si se intenta ingresar por ruta directa el guardián del componente administrador lo impide | Correcto |

*Tabla 29 Pruebas de Módulo 1 Control de inventario*

### 5.2.2 Módulo 2 Reabastecimiento de inventario

*Tabla 30 Pruebas de Módulo 3 Control de ventas*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar interactuar con el componente de administrador/a sin tener un usuario con rol administrador | El enlace de acceso al componente administrador no aparece, si se intenta ingresar por ruta directa el guardián del componente administrador lo impide | Correcto |
| **2** | Ingresar nuevo *stock* a cualquier insumo con 0 unidades o gramos. | La aplicación no incrementa ningún valor al *stock* del insumo. | Correcto |
| **3** | Ingresar letras en el formulario de ingreso de *stock* en cualquier insumo | La aplicación devuelve una alerta de solo ingresar números. | Correcto |

### 5.2.3 Módulo 3 Control de ventas

*Tabla 31 Módulo 3 Control de ventas*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar interactuar con el componente de administrador/a sin tener un usuario con rol administrador | El enlace de acceso al componente administrador no aparece, si se intenta ingresar por ruta directa el guardián del componente administrador lo impide | Correcto |
| **2** | Editar un producto con información incorrecta en los formularios | La aplicación por medio de reactive forms devuelve un control de formularios por parte del usuario | Correcto |
| **3** | Crear un producto con información incorrecta en los formularios | La aplicación por medio de reactive forms devuelve un control de formularios por parte del usuario | Correcto |

### 5.2.4 Módulo 4 Lista de pedido

*Tabla 32 Pruebas de Módulo 4 Lista de pedido*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar enviar un producto al carrito de compras sin seleccionar los insumos necesarios para procesarlo | El botón de interacción no se activa si el producto no tiene los insumos necesarios para procesarlos | Correcto |
| **2** | Intentar interactuar con el componente de carrito de compras sin tener un usuario con rol administrador o cajero | La aplicación no permite el ingreso al componente carrito de compras si el guardián del componente no detecta un usuario con un rol de administrador o cajero. | Correcto |

| | | | |
|---|---|---|---|
| **3** | Los productos del pedido no son los correctos o se requiere otra interacción | La aplicación puede eliminar productos que sean incorrectos y puede regresar a la tienda para seleccionar los correctos | Correcto |

### 5.2.5 Módulo 5 Cobro del pedido

*Tabla 33 Pruebas de Módulo 5 Cobro del pedido*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar interactuar con el componente de carrito de compras sin tener un usuario con rol administrador o cajero | La aplicación no permite el ingreso al componente carrito de compras si el guardián del componente no detecta un usuario con un rol de administrador o cajero. | Correcto |
| **2** | Los productos del pedido no son los correctos o se requiere otra interacción | La aplicación puede eliminar productos que sean incorrectos y puede regresar a la tienda para seleccionar los correctos | Correcto |
| **3** | El costo del pedido no es aceptado | Se debe cancelar el pedido y regresar a la tienda principal para un nuevo pedido | Correcto |

### 5.2.6 Módulo 6 Pedido de cliente

*Tabla 34 Pruebas del Módulo 6 Pedido de cliente*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar enviar un producto al carrito de compras sin seleccionar los insumos necesarios para procesarlo | El botón de interacción no se activa si el producto no tiene los insumos necesarios para procesarlos | Correcto |
| **2** | Intentar interactuar con el componente de carrito de compras sin tener un usuario con rol administrador o cajero | La aplicación no permite el ingreso al componente carrito de compras si el guardián del componente no | Correcto |

| | | | |
|---|---|---|---|
| | | detecta un usuario con un rol de administrador o cajero. | |
| **3** | Los productos del pedido no son los correctos o se requiere otra interacción | La aplicación puede eliminar productos que sean incorrectos y puede regresar a la tienda para seleccionar los correctos | Correcto |

### 5.2.7 Módulo 7 Cotización de pedido

*Tabla 35 Pruebas del Módulo 7 Cotización de pedido*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar enviar un producto al carrito de compras sin seleccionar los insumos necesarios para procesarlo | El botón de interacción no se activa si el producto no tiene los insumos necesarios para procesarlos | Correcto |
| **2** | Intentar interactuar con el componente de carrito de compras sin tener un usuario con rol administrador o cajero | La aplicación no permite el ingreso al componente carrito de compras si el guardián del componente no detecta un usuario con un rol de administrador o cajero. | Correcto |
| **3** | El precio de los productos del pedido no son aceptados o se requiere otra eliminar productos. | La aplicación puede eliminar productos que sean necesarios | Correcto |

### 5.2.7 Módulo 8 Pago de pedido

*Tabla 36 Pruebas de Módulo 8 Pago de pedido*

| N.º | Descripción | Resultado Esperado | Estado |
|---|---|---|---|
| **1** | Intentar interactuar con el componente de carrito de compras sin tener un usuario con rol administrador o cajero | La aplicación no permite el ingreso al componente carrito de compras si el guardián del componente no detecta un usuario con un rol de administrador o cajero. | Correcto |

| | | | |
|---|---|---|---|
| **2** | Ingresar la información del cliente de manera incorrecta en los formularios | La aplicación no permite el ingreso de información incorrecta por medio de los componentes de *reactive forms* | Correcto |
| **3** | El comprobante se ingresa sin la información del usuario | La aplicación permite esta acción debido a que hay clientes que no desean ingresar sus datos. | Correcto |

# CAPÍTULO 6: CONCLUSIONES Y RECOMENDACIONES

## 6.1 Conclusiones

1. El desarrollo del prototipo funcional se pudo realizar de manera más comprensible gracias a la utilización de la metodología ágil *SCRUM* que permite la interacción con los *stakeholders* de manera más oportuna.
2. Los requerimientos funcionales de la aplicación fueron analizados junto a los usuarios principales, lo que produjo un entendimiento claro y un desarrollo de interfaz sencillo.
3. El desarrollo de servicios, mediante el diseño de la lógica de negocio, en los módulos de control de ventas, control de inventario y control de costos permitió la ejecución correcta de la aplicación.
4. El desarrollo con el *framework Angular* y la implementación de la tecnología *PWA*, permitió tener aplicaciones que pueden ser instaladas simulando aplicaciones nativas. Lo que permite la adaptabilidad de todos los módulos de la aplicación, en cualquier dispositivo que soporte la ejecución de aplicaciones inteligentes.
5. El uso de las herramientas como *Git*, *GitHub* y *NPM* permitió una optimización del proyecto, tanto en el desarrollo *backend* como en el *frontend*, dando como resultado una mejor funcionalidad, rapidez y visualización de la aplicación.
6. El uso del patrón de arquitectura de *software* MVVM, permitió tener una abstracción de los servicios, que conectan la aplicación con la base de datos, lo que permite una escalabilidad de la base de datos a futuro.
7. Las validaciones se realizaron del lado del servidor por medio de validaciones predeterminadas de *firebase* y *firestore*. Mientras que las validaciones de parte del cliente se realizaron desde los componentes que ofrece *Angular*, con ambas validaciones aseguramos seguridad de la conexión en la aplicación.
8. El diseño de la base de datos basada en documentos permitió adaptar la base de datos, para futuras creaciones de módulos y la actualización de los módulos vigentes.
9. Las pruebas de caja negra y caja blanca realizadas en cada módulo del proyecto, garantizaron la correcta funcionalidad de la aplicación.

10. Las pruebas finales se realizaron en la heladería "Helarte", dando satisfacción a los usuarios de este y solicitando actualizaciones de nuevos módulos para el futuro.

## 6.2 Recomendaciones

1. El uso de metodologías ágiles permite estar más conectados con los usuarios, de las aplicaciones actuales, con ello se disipan dudas y se pulen detalles en las aplicaciones.
2. El desarrollo de *software* actual mediante el conocimiento en distintos lenguajes, *frameworks*, servicios y microservicios, brinda una gama de posibilidades muy diversas, por lo que es necesario plantear cuáles serán las herramientas necesarias previo al desarrollo de cualquier proyecto.
3. Tener un arco de aprendizaje en una tecnología en específica y determinar qué tipo de desarrollo se debe utilizar previo al inicio de los ciclos de desarrollo.
4. Es recomendable utilizar las versiones de desarrollo de los navegadores como *Google Chrome dev, Firefox dev, Chromion dev,* para probar la aplicación, ya que eso dará una perspectiva del funcionamiento del proyecto en los navegadores *web*.
5. Es recomendable pagar las funcionalidades de seguridad que proporciona *Firebase* para aumentar el porcentaje de seguridad de la aplicación.
6. El uso de microservicios y librerías como *GitHub, npm* y servidores con seguridad, permitirá la optimización del desarrollo de las aplicaciones. Además, el correcto funcionamiento en producción de la aplicación.
7. Para cambios futuros o manteniendo es recomendable contratar a una persona con conocimientos en *Angular* y *firebase* además del manejo de metodologías ágiles.
8. Es recomendable implementar las funcionalidades de análisis estadístico de ventas y compras para optimizar las ganancias de la heladería.

## DICCIONARIO DE DATOS

***A***

**Administrador**: Usuario que tendrá acceso maestro a todos los módulos del programa

**Aplicación**: Programa informático creado para llevar a cabo o facilitar una tarea en un dispositivo informático.

***API***: Interfaz de programación de aplicaciones

***B***

***BackEnd***: capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. (Platzi, 2018)

***C***

**Cajero**: Usuario que tendrá acceso limitado a los módulos del programa.

**Cliente**: Usuario que podrá observar los productos y seleccionarlos junto con los insumos deseados.

***CSS***: *Cascading Style Sheets* (Schulz, 2008)

***CRUD***: funcionalidad que describe operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos de create(crear), read(leer), update(actualizar), delete(borrar).

**Cotización**: Valor total del pedido del cliente.

**Componente**: Bloque de código que consta de 3 archivos un *CSS, HTML y Typescript*

**Carrito de compras**: Módulo con los procesos de confirmación de pedido y emisión de comprobante de pago.

**Control de Ventas**: Módulo que procesa el valor total de las ventas en el día y en el mes vigente. Además, el número de ventas de productos en el día y mes vigentes.

**Comprobante**: Documento que muestra la información del cliente que realizó la compra y el detalle del pedido.

***D***

***DOM***: *Document Object Model* es una interfaz independiente de la plataforma y el lenguaje que permite a los programas y scripts acceder dinámicamente y actualizar el contenido, la estructura y el estilo de un documento. (W3C, 2007)

***DBaaS***: Bases de datos como servicio

**Día vigente**: Día actual en el que se realizan las transacciones con la aplicación.

***F***

***FrontEnd***: práctica de usar *HTML*, *CSS* y *JavaScript* para crear aplicaciones web del lado del cliente. (Bootstrap, 2020)

***Firebase***: plataforma móvil creada por *Google*, cuya principal función es desarrollar y facilitar la creación de aplicaciones de elevada calidad de una forma rápida

***Firestore***: *Cloud Firestore* es una base de datos flexible y escalable para el desarrollo en servidores, dispositivos móviles y la *Web* desde *Firebase* y *Google Cloud*. (firestore.Google.com, 2020)

***Firebase auth***: autenticación de usuarios de *firebase*.

***G***

**Guardianes**: *Middlewares* que provee *Angular* para resguardar el acceso a rutas específicas.

***Git***: Sistema de control de versiones.

***GitHub***: Repositorio de código en la nube.

***H***

***HTML***: *Hypertext Marques Lenguaje* (Franganillo, 2011)

***HTTP***: Protocolo de transferencia de hipertexto (*Hyper text transfer protocol*)

***HTTPS***: Protocolo de transferencia de hipertexto segura (*Hyper text transfer protocol secure*)

***I***

**Interesados o *Stakeholders***: Personas que tienen relación con el uso y el desarrollo del proyecto.

**Insumos:** Material utilizados en la preparación de productos.

**Inventario**: Lista de insumos utilizados.

***J***

***jQuery***: Conjunto de implementaciones funcionales ya definidas, desarrolladas y probadas que están listas para utilizar para *JavaScript*. (Bootstrap, 2020)

***L***

**Lista de producto:** Selección de productos por parte del cliente dentro del carrito de compras.

***Landing Page***: Página de lanzamiento, página inicial donde están los procesos principales.

***Loguearse***: Acceder a la aplicación con una cuenta registrada.

***M***

**Módulo:** Conjunto de componentes que muestran los procesos a desarrollar.

**Mes Vigente**: Mes actual donde se realizan las transacciones con la aplicación.

**MVC**: Modelo vista controlador

**MVVM**: Modelo vista – vista modelo

***N***

***NoSQL***: No solo *SQL* (not only *SQL*)

***P***

***Product Owner* o dueño del producto**: Persona encargada de dialogar con los

clientes e interpreta sus ideas y cambios para luego pasar la información al *Scrum Master* como requerimientos funcionales o no funcionales.

***Plugin***: Extensiones funcionales de una aplicación. (Bootstrap, 2020)

**Producto**: Producto procesado dentro de listado en la aplicación.

**Proceso:** Secuencia de acciones para lograr un objetivo específico.

**Pedido**: Lista de productos seleccionados por el cliente.

***PWA***: Aplicaciones de *web* progresivas (progressive web apps)

***R***

***Responsive***: Técnica de diseño *web* que busca la correcta visualización de una misma página en distintos dispositivos. (Bootstrap, 2020)

**Reabastecimiento de inventario**: Ingresar más *stock* en un insumo del inventario.

***S***

***SCRUM***: Marco de referencia de metodología de desarrollo.

***Scrum Master***: Persona encargada de organizar al equipo de desarrollo y mantener un diálogo con el *Product Owner* para plasmar las ideas junto con el equipo de desarrollo.

***Stakeholders***: Persona que tiene interacción con el producto final o tiene cierta relevancia para el desarrollo de este.

***Sprint***: Ciclo de desarrollo en *Scrum* cuya finalidad es el incremento del producto final.

***Superset***: lenguaje escrito sobre otro lenguaje. un lenguaje basado en el original. (Microsoft, 2012)

**Servicio**: Componente que conecta la base de datos con la aplicación transformando la metadata en datos legibles

***T***

***Team Development***: Equipo de desarrollo

***U***

**Usuario**: Persona que estará manejando el programa de manera directa.

***Unicode***: Estándar donde un número único para cada carácter, independientemente de la plataforma, dispositivo, aplicación o idioma. (Mark Davis, 2009)

***W***

***W3C***: *World Wide Web Consortium* (W3C, 2007)

***WHATWG***: *Web Hypertext Application Technology Working Group* (W3C, 2007)

## BIBLIOGRAFÍA

Abernethy, M. (14 de junio de 2011). *www.ibm.com*. Recuperado el 8 de marzo de 2020, de www.ibm.com: https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/index.html

Amaya Mayra, C. J. (12 de abril de 2012). *Universidad de Cuenca.* Recuperado el 28 de 01 de 2020, de Universidad de Cuenca: http://dspace.ucuenca.edu.ec/bitstream/123456789/1355/1/tcon652.pdf

Bootstrap. (2020). *getbootstrap.com*. Recuperado el 9 de marzo de 2020, de getbootstrap.com: https://getbootstrap.com/

Camacho, D. (2020). *Platzi.com*, 1. Recuperado el 24 de agosto de 2020, de Platzi.com: https://platzi.com/blog/que-es-github-como-funciona/

comscore. (2016). *www.comscore.com*, 1. (A. Fosk, Editor) Recuperado el 2018 de marzo de 2020, de www.comscore.com: https://www.comscore.com/lat/Prensa-y-Eventos/Presentaciones-y-libros-blancos/2020/El-estado-global-de-Mobile

Durán, Y. (2012). *Administración del inventario: elemento clave para la optimización de las utilidades en las empresas.* Universidad de los Andes, Administración del inventario. Mérida, Venezuela: Visión Gerencial. Recuperado el 10 de marzo de 2020, de https://www.redalyc.org/pdf/4655/465545892008.pdf

Ecuador, G. d. (2019). *proecudor*. Obtenido de www.proecuador.gob.ec: https://www.proecuador.gob.ec/

Ecuador, U. A. (2019). *eumed*. Obtenido de https://www.eumed.net/rev/oel/2018/04/pymes-ecuador-financiamiento.html

Empresarial, L. d. (2017). *ecuadorencifras*. Obtenido de www.ecuadorencifras.gob.ec: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Bibliotecas/Libros/Panorama%20Laboral%202017.pdf

expressjs. (2017). *http://expressjs.com/*. Recuperado el 10 de marzo de 2020, de http://expressjs.com/: http://expressjs.com/

firestore.Google.com. (28 de enero de 2020). *https://firebase.google.com/docs/firestore*, 1. Recuperado el 23 de agosto de 2020, de https://firebase.google.com/: https://firebase.google.com/docs/firestore

Franganillo, J. (11 de enero de 2011). *http://www.thinkepi.net/*, 5. Recuperado el 23 de febrero de 2020, de http://www.thinkepi.net/: https://franganillo.es/html5.pdf

Goat., Software Mountain. (1998). *www.mountaingoatsoftware.com*. Obtenido de www.mountaingoatsoftware.com: https://www.mountaingoatsoftware.com/agile/scrum

Google. (2020). *https://angular.io/*. Recuperado el 9 de marzo de 2020, de https://angular.io/: https://angular.io/

Google Developers. (6 de enero de 2020). *web.dev*, 1. (L. Sam Richard, Editor) Recuperado el 18 de 03 de 2020, de web.dev/what-are-pwas/#introduction: https://web.dev/what-are-pwas/#introduction

Google LLC. (01 de 01 de 2010). *Angular Material*, 12.2.3. (E. d. Angular, Editor, Google, Productor, & Google LCC) Recuperado el 23 de agosto de 2020, de Material Design for Angular: https://material.angular.io/

Gutiérrez, A. F. (2007). Gestión de Stocks en la logística de almacenes. En A. F. Gutiérrez, *Gestión de Stocks en la logística de almacenes.* (2 ed., Vol. 2, págs. 197-203). Madrid, España: Fundación Confemetal. Recuperado el 10 de febrero de 2020, de https://books.google.com.ec/books?hl=es&lr=lang_es&id=4oKwdF77cncC&oi=fnd&pg=PA5&dq=gestion+de+stock&ots=weTrQ8zTx9&sig=i30N3sVpfdTBINJnXYrY6aRX8sA&redir_esc=y#v=onepage&q=fifo&f=false

Hortensio, S. (2003). *Universidad de Santiago de Compostela.* doi:10.3989/hispania.2003.v63.i215

Jiménez, J. R. (2020). *Academia.edu*. Recuperado el 17 de febrero de 2020, de Metodologías de Desarrollo Ágil: https://www.academia.edu/1740166/Metodolog%C3%ADas_de_desarrollo_%C3%A1gil

José Antonio Díaz-Batista, D. P.-A. (09 de enero de 2012). *Investigación de operaciones optimización de los niveles de inventario en una cadena de suministro.* Recuperado el 30 de 01 de 2020, de http://rii.cujae.edu.cu/index.php/revistaind/article/view/379/450

Joseph, K. (2016). *github.com*, 6.3.1. Recuperado el 24 de agosto de 2020, de github.com: https://github.com/kristoferjoseph/flexboxgrid

Ken Schwaber, J. S. (Julio de 2016). *scrumguides.org.* Recuperado el 17 de febrero de 2020, de La Guía Definitiva de Scrum: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100

Kniberg, H. (2007). *SCRUM Y XP DESDE LAS TRINCHERAS* (Primera ed., Vol. 1). (D. Plesa, Ed., & Á. Medinilla, Trad.) Estados Unidos de América, Estados Unidos: C4Media Inc. Recuperado el 23 de febrero de 2020, de http://infoq.com/minibooks/ scrum-xp-from-the-trenches

Mark Davis, K. W. (19 de marzo de 2009). *http://www.unicode.org/.* Recuperado el 27 de febrero de 2020, de http://www.unicode.org/: http://www.unicode.org/L2/L2009/09137-pri145-uax15-30.pdf

Microsoft. (8 de octubre de 2005). *docs.microsoft.com/en-us/archive/blogs/johngossman*, 1. (J. Gossman, Editor) Recuperado el 22 de agosto de 2020, de docs.microsoft.com/en-us/archive/blogs/: https://docs.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps

Microsoft. (2012). *www.typescriptlang.org*. Recuperado el 8 de marzo de 2020, de www.typescriptlang.org: https://www.typescriptlang.org/index.html

Microsoft. (12 de febrero de 2020). *© Microsoft 2020.* (Microsoft, Ed.) Recuperado el 18 de marzo de 2020, de © Microsoft 2020: https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-3.1

MrRio. (2020). *github.com*, 2.3.1. Recuperado el 24 de agosto de 2020, de github.com: https://github.com/MrRio/jsPDF

Nahas, A. (2020). *github.com*, 5.3.0. Recuperado el 24 de agosto de 2020, de github.com: https://github.com/anthonynahas/ngx-auth-firebaseui

Nicklasvh. (2020). *github.com*, 1.3.2. Recuperado el 24 de agosto de 2020, de github.com: https://www.npmjs.com/package/html2canvas

Parada, J. E. (12 de Julio de 2006). *Sistemas de Inventario.* Caracas: Ediciones PuntoCero. Recuperado el 10 de marzo de 2020, de Sistemas de Inventario: https://s3.amazonaws.com/academia.edu.documents/48944075/inventarios.pdf?response-content-disposition=inline%3B%20filename%3DSistemas_de_Inventario.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20200310%2Fus-east-1%2Fs3%2Faw

Pérez, J. E. (25 de marzo de 2009). *http://www.librosweb.es.* Recuperado el 8 de marzo de 2020, de http://www.librosweb.es: http://www.librosweb.es/javascript

Platzi. (12 de marzo de 2018). *platzi*, 1. (N. Chapaval, Editor, & platzi) Recuperado el 15 de agosto de 2020, de platzi: https://platzi.com/blog/que-es-frontend-y-backend/

Platzi. (febrero de 2020). *platzi.com*, 1. (N. Molina, Editor) Recuperado el 18 de marzo de 2020, de platzi.com: https://platzi.com/clases/1818-pwa-angular/26024-caracteristicas-de-una-pwa/

PWA stats. (s.f.). *www.pwastats.com*, 1. Recuperado el 18 de marzo de 2020, de www.pwastats.com: https://www.pwastats.com/page3

SAP. (2020). *www.sap.com/latinoamerica*. Recuperado el 4 de marzo de 2020, de www.sap.com/latinoamerica:

https://www.sap.com/latinamerica/products/powerdesigner-data-modeling-tools.html#item_0

Schulz, R. G. (2008). *Diseño web con CSS* (Primera ed., Vol. 1). (V. P. Moreno, Trad.) Barcelona, España: Marcombo. Recuperado el 7 de marzo de 2020, de https://books.google.com.ec/books?hl=es&lr=lang_es&id=ZgI2WfHjPiEC&oi=fnd&pg=PA45&dq=Dise%C3%B1o+web+con+CSS&ots=7nWsjZRV1l&sig=6Q9XEfwnA_Loa-cO4sar-wMWbs0&redir_esc=y#v=onepage&q=Dise%C3%B1o%20web%20con%20CSS&f=false

sumup. (2020). *debitor*, 1. Recuperado el 23 de marzo de 2020, de debitor by sumup: https://debitoor.es/glosario/definicion-ecommerce

Tavares, O. B. (agosto de 2020). *Platzi.com*. Recuperado el 24 de agosto de 2020, de Platzi.com: https://platzi.com/clases/npm/

W3C. (26 de noviembre de 2007). *www.w3.org*. (W. Seltzer, Editor) Recuperado el 7 de marzo de 2020, de www.w3.org: https://www.w3.org/TR/html-design-principles/

Yenisleidy Fernández Romero, Y. D. (enero de 2012). https://biblat.unam.mx/pt/revista/telemtica-la-habana/articulo/patron-modelo-vista-controlador. *Telem@tica, 11*(1), 47-50. Recuperado el 18 de marzo de 2020, de biblat, bibliografía latinoamerica: https://biblat.unam.mx/pt/revista/telemtica-la-habana/articulo/patron-modelo-vista-controlador