



APLICACIÓN MÓVIL Y DE ESCRITORIO POS OFFLINE PARA LA EMPRESA FULL COLOMBIA S.A.S.

PROPONENTE

Santiago de la Vega Vargas

DIRECTOR

Ing. Harrizon Alexander Soler Galindo

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA DE SISTEMAS

SECCIONAL TUNJA

Tunja

Agosto 2026

CONTENIDO

1. FICHA TÉCNICA DE LA PASANTÍA.....	6
2. ANTECEDENTES DE LA EMPRESA.....	6
3. PLANTEAMIENTO DEL PROBLEMA	7
4. JUSTIFICACIÓN.....	9
5. OBJETIVOS	10
5.1. OBJETIVO GENERAL	10
5.2. OBJETIVOS ESPECÍFICOS	11
6. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS.....	11
6.1. LISTADO DE ACTIVIDADES REALIZADAS EN EL PROYECTO.....	11
6.2. HERRAMIENTAS UTILIZADAS EN EL DESARROLLO DEL PROYECTO	16
6.2.1. ENTORNO DE DESARROLLO Y COMPILACIÓN	16
6.2.2. BIBLIOTECAS CON INCIDENCIA EN LOS OBJETIVOS	17
6.2.3. GESTIÓN DEL TRABAJO.....	18
6.3. DESCRIPCIÓN GENERAL DEL SISTEMA INTERVENIDO.....	18
6.3.1. ACTORES Y FUNCIONES DEL SISTEMA	18
6.3.2. ARQUITECTURA DE LA APLICACIÓN	19
6.3.3. DESPLIEGUE Y ENTORNO DE OPERACIÓN	21
6.3.4. LEVANTAMIENTO DE REQUERIMIENTOS Y REALIMENTACIÓN	22
6.4. DESARROLLO DEL OBJETIVO ESPECÍFICO 1	23
6.4.1. ANÁLISIS DE LA ARQUITECTURA Y LÍNEA BASE DEL PROYECTO	23
6.4.2. ELABORACIÓN DE LOS MOCKUPS PARA ESCRITORIO	24
6.4.3. REFACTORIZACIÓN DE LA INTERFAZ Y CORRECCIÓN DE FALLOS	26
6.4.4. DESARROLLO DEL MÓDULO DE GESTIÓN DE TURNOS	31

6.4.5.	PERSISTENCIA EN SQLITE Y EVOLUCIÓN DEL ESQUEMA	33
6.4.6.	GENERACIÓN DE REPORTES DE VENTA SIN CONEXIÓN	35
6.4.7.	RESUMEN DE ACTIVIDADES DEL OBJETIVO 1	37
6.4.8.	LIMITACIONES IDENTIFICADAS	37
6.5.	DESARROLLO DEL OBJETIVO ESPECÍFICO 2	38
6.5.1.	CONFIGURACIÓN DEL ENTORNO DE COMPILACIÓN PARA ANDROID	38
6.5.2.	CRITERIO DE ADAPTACIÓN: DOS ESTRATEGIAS SEGÚN LA PANTALLA	38
6.5.3.	CONSTRUCCIÓN DEL MÓDULO POS MÓVIL	39
6.5.4.	CONSTRUCCIÓN DEL MÓDULO POS RESTAURANTE MÓVIL.....	43
6.5.5.	CORRECCIÓN DEL ACCESO A LA RED EN LA COMPILACIÓN DE DISTRIBUCIÓN	43
6.5.6.	DISTRIBUCIÓN Y ESTADO EN PRODUCCIÓN	43
6.5.7.	RESUMEN DE ACTIVIDADES DEL OBJETIVO 2	44
6.6.	DESARROLLO DEL OBJETIVO ESPECÍFICO 3	44
6.6.1.	EL CONTEXTO QUE CONDICIONA EL PROBLEMA.....	45
6.6.2.	LA SITUACIÓN PREVIA	46
6.6.3.	EL DESPEJE INVERSO	46
6.6.4.	EL ORDEN DE APLICACIÓN ENTRE DESCUENTO E IMPUESTO	47
6.6.5.	VERIFICACIÓN DE LA CONSISTENCIA DEL DESPEJE	48
6.6.6.	RECÁLCULO EN TIEMPO REAL.....	48
6.6.7.	INTERFAZ DE VERIFICACIÓN DETALLADA.....	49
6.6.8.	NORMALIZACIÓN DEL REDONDEO	52
6.6.9.	CORRECCIÓN DE DEFECTOS DE EXACTITUD	53
6.6.10.	VALIDACIÓN CRUZADA ENTRE CLIENTES.....	53
6.6.11.	LA FACTURA COMO COMPROBACIÓN DEL CÁLCULO	54
6.6.12.	LIMITACIONES IDENTIFICADAS	59
6.7.	DESARROLLO DEL OBJETIVO ESPECÍFICO 4	60

6.7.1.	POR QUÉ SCRUM Y NO OTRO MARCO DE TRABAJO	60
6.7.2.	CÓMO FUNCIONABA EL PROCESO EN LA PRÁCTICA.....	61
6.7.3.	LOS ARTEFACTOS QUE QUEDARON REGISTRADOS	61
6.7.4.	ESTRATEGIA DE RAMAS	64
6.7.5.	CICLO DE LIBERACIÓN	64
6.7.6.	SINCRONIZACIÓN ENTRE LA APLICACIÓN OFFLINE Y LOS SERVICIOS DEL BACKEND	65
6.7.7.	OBSERVABILIDAD COMO RESPUESTA A LA AUSENCIA DE PRUEBAS	69
6.7.8.	FORMALIZACIÓN DE PROCEDIMIENTOS	69
6.7.9.	INTEGRACIÓN CONTINUA: MEJORA IDENTIFICADA Y NO IMPLEMENTADA.....	70
6.7.10.	RESUMEN DE ACTIVIDADES DEL OBJETIVO 4	70
6.7.11.	LIMITACIONES IDENTIFICADAS	71
7.	<u>CONCLUSIONES</u>	<u>71</u>
8.	<u>REFERENCIAS</u>	<u>72</u>
9.	<u>MANUAL DE USUARIO</u>	<u>75</u>
9.1.	REQUISITOS E INSTALACIÓN	75
9.1.1.	VERSIÓN DE ESCRITORIO	75
9.1.2.	VERSIÓN ANDROID	75
9.2.	ACCESO AL SISTEMA.....	75
9.2.1.	INICIO DE SESIÓN	75
9.2.2.	SELECCIÓN DE SUCURSAL	76
9.2.3.	SINCRONIZACIÓN DE DATOS.....	77
9.2.4.	PANTALLA PRINCIPAL.....	78
9.3.	GESTIÓN DE TURNOS	80
9.3.1.	APERTURA DE TURNO	80
9.3.2.	CIERRE DE TURNO.....	81

9.3.3.	RECIBO DE CIERRE Y CONSULTA DE TURNOS ANTERIORES	82
9.4.	PUNTO DE VENTA COMERCIAL.....	85
9.4.1.	BÚSQUEDA Y SELECCIÓN DE ARTÍCULOS	85
9.4.2.	MODIFICACIÓN DE CANTIDADES Y DEL PRECIO	85
9.4.3.	APLICACIÓN DE DESCUENTOS	86
9.4.4.	VERIFICACIÓN DETALLADA DE VALORES	87
9.4.5.	COBRO DE LA VENTA	88
9.4.6.	IMPRESIÓN DEL DOCUMENTO	90
9.5.	PUNTO DE VENTA RESTAURANTE	91
9.5.1.	SECTORES Y MESAS	91
9.5.2.	APERTURA Y EDICIÓN DE CUENTA.....	92
9.5.3.	ADICIONES	94
9.5.4.	PRECUENTA Y COMANDA	94
9.6.	MÓDULOS DE APOYO	96
9.6.1.	CLIENTES	96
9.6.2.	ARTÍCULOS, INVENTARIO Y RESOLUCIONES	97
9.7.	SINCRONIZACIÓN Y DIAGNÓSTICO	98
9.7.1.	SINCRONIZACIÓN MANUAL.....	98
9.7.2.	EXPORTACIÓN DE REGISTROS PARA SOPORTE	98
9.8.	PARTICULARIDADES DE LA VERSIÓN ANDROID.....	99

1. FICHA TÉCNICA DE LA PASANTÍA

Tabla 1
Ficha técnica de la pasantía

Campo	Contenido
Título	Aplicación móvil y de escritorio POS offline para la empresa Full Colombia SAS
Nombre del estudiante	Santiago de la Vega Vargas
Documento de identidad	C.C. 1.032.796.953 expedida en Villa de Leyva
Correo electrónico del estudiante	Santiago.vega@usantoto.edu.co
Director	Ing. Harrizon Alexander Soler Galindo
Línea de investigación	Ingeniería del software
Empresa	Full Colombia S.A.S. — NIT 901.402.077-0
Ciudad donde se ubica la empresa	Tunja, Boyacá
Modalidad de la pasantía	Híbrida, en las instalaciones de la empresa
Tutor empresarial	Fabián Armando Castro Guerrero — Representante legal
Duración	8 meses (640 horas)
Palabras clave	desarrollo multiplataforma, facturación electrónica, persistencia de datos, punto de venta

Fuente: Autor

Los datos de empresa, ciudad, modalidad, tutor y duración corresponden al convenio específico de prácticas N.º 5110-20-2/0125, suscrito entre la Universidad Santo Tomás seccional Tunja y Full Colombia S.A.S.

2. ANTECEDENTES DE LA EMPRESA

Full Colombia S.A.S. es una empresa dedicada al desarrollo de soluciones tecnológicas en software, hardware y servicios de soporte, con enfoque en la digitalización empresarial y la facturación electrónica (Full Colombia, 2021). Sus dos líneas principales son Full EDS, orientada a estaciones de servicio, y Full Suit, un sistema de gestión empresarial que integra módulos de facturación, inventarios, ventas y reportes.

El crecimiento de los modelos de software como servicio y las exigencias normativas de la Dirección de Impuestos y Aduanas Nacionales han consolidado la posición de la empresa en el mercado de comercios y restaurantes. Esa entidad establece los lineamientos técnicos que las empresas deben cumplir para la emisión y validación de facturas electrónicas, y asegura con ello la trazabilidad y autenticidad de la información fiscal (Dirección de Impuestos y Aduanas Nacionales [DIAN], s. f.).

Dentro de la línea Full Suit conviven dos productos que comparten los mismos servicios de backend. El primero es una plataforma web. El segundo es una aplicación local desarrollada en Flutter y Dart, diseñada para operar aun sin conexión a internet mediante una base de datos SQLite que conserva el catálogo, los clientes, las resoluciones de facturación y las ventas realizadas, y que reconcilia esa información con los servicios de la empresa en cuanto el establecimiento recupera la red, de modo que un comercio pueda seguir facturando durante una caída de conectividad sin que ello comprometa la integridad de sus registros ni su cumplimiento tributario posterior. Este enfoque entre operación local y en línea es habitual en los sistemas distribuidos que deben garantizar continuidad del servicio sin conectividad permanente (Viscaino Quito & Serpa Andrade, 2022).

Esa aplicación local es el activo sobre el que se desarrolló la pasantía. Al llegar, la empresa ya tenía un tablero en GitHub Projects y las necesidades del producto relativamente identificadas, pero los requerimientos se fueron entregando de forma gradual, así que desde el inicio no se conocía todo el alcance. Había que rehacer la interfaz, que no correspondía con la línea gráfica vigente y hacía difícil usar filtros y listados, y llevar los turnos de cajero y los reportes de venta a la aplicación local. Lo que más pedían los clientes era poder editar el precio, seguido de la versión para Android. Esta última la pedía sobre todo uno de los clientes principales, un hotel de la región, para su zona de restaurante: querían que los meseros llevaran una tableta con el sistema y desde ahí pudieran tomar cuentas y facturar artículos.

3. PLANTEAMIENTO DEL PROBLEMA

La aplicación local de Full Suit funciona como punto de venta sin conexión. En muchos de los comercios que la usaban la conexión a internet era mala e intermitente, así que era preferible buscar conexión los pocos minutos que demoraba la sincronización que tener todo el día una conexión estable.

Además de ser pobre a nivel gráfico, la interfaz tenía muchos errores funcionales, sobre todo en los filtros de las listas. Muchas veces se limpiaban los campos de los filtros pero los registros no se actualizaban. También pasaba que al entrar al detalle de un registro y volver, el listado quedaba en la misma página, pero la paginación mostraba la página 1. Los controles de paginación muchas veces no funcionaban bien, e incluso había un servicio que filtraba en el frontend y no en el backend, lo que hacía la consulta supremamente lenta.

Los turnos existían, pero solo en la nube y no en la aplicación local, así que para los usuarios era tedioso tener que entrar al portal web a ver el registro de los turnos. Además, el registro no

quedaba bien, porque los turnos de la aplicación local llegaban a la nube solo al sincronizar, y no quedaba guardada la hora real en que había empezado y terminado cada turno.

Los comercios simplemente usaban la plataforma web o la aplicación local, pero solo en computadores. El uso en tabletas no existía antes y llegó con esta actualización.

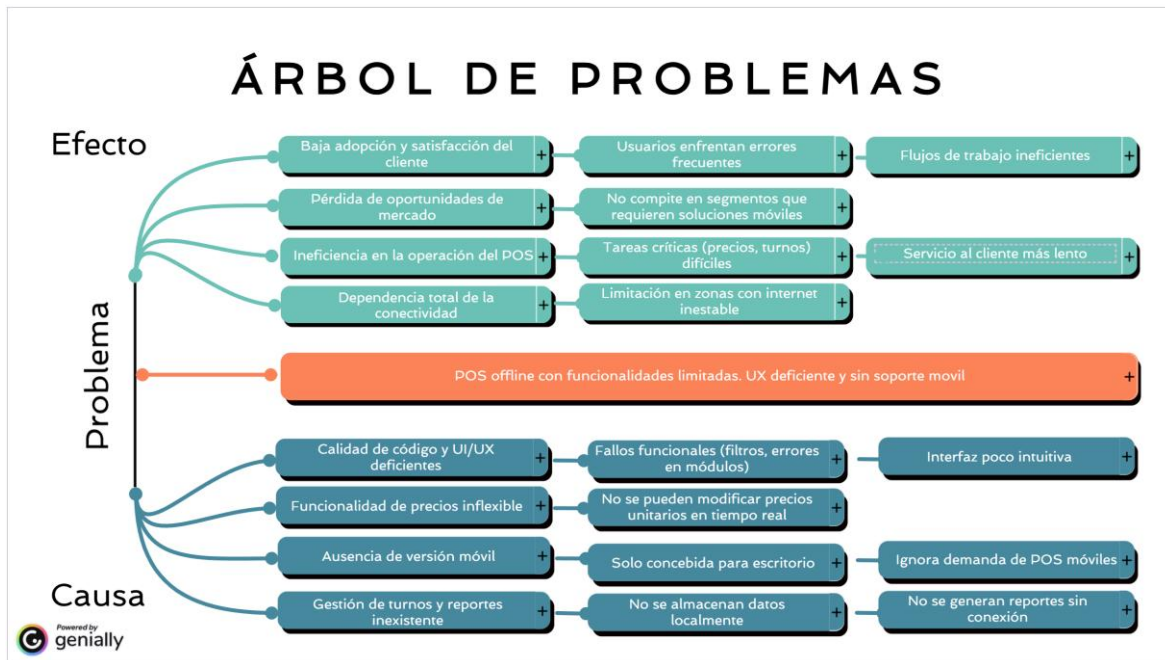
Tampoco se podía editar el precio en el momento de la venta. Para cambiar el precio final de un producto solo había dos opciones: crear un descuento para ese producto, pero en la factura electrónica salía como descuento junto al valor original, o modificar directamente el valor del producto, y entonces ese sería el precio del artículo en todas las ventas de ahí en adelante, hasta que se volviera a editar. Para resolverlo, el sistema tenía que dejar de calcular solo hacia adelante desde el precio de catálogo y poder sacar la base gravable, el descuento y el impuesto a partir del precio final que digita el cajero.

Durante la pasantía era muy común que llegaran mensajes al grupo de la empresa del representante legal, que era el mismo que cerraba los negocios con los comercios, diciendo que tal comercio había reportado un fallo, que al hacer una venta se quedaba cargando o que no le funcionaban los descuentos. Los clientes reportaban fallos críticos con frecuencia, y en una ocasión se supo de uno que se fue con otro proveedor porque era más accesible.

Siigo era uno de los competidores que más se escuchaban. Al tener un sistema más estable podía expandirse con mayor facilidad y ofrecer precios más accesibles, algo que Full Suit nunca pudo hacer por la falta de estabilidad de su sistema. El equipo pasaba la mayor parte del tiempo arreglando errores y atendiendo requerimientos de los clientes actuales, así que triplicar el número de clientes no era un escenario muy prometedor: no se habría dado abasto.

Para representar esta situación con mayor claridad se elaboró un árbol de problemas, en el que se identifica como problema central un punto de venta sin conexión con funcionalidades limitadas, experiencia de usuario deficiente y sin soporte móvil, junto con sus principales causas y efectos.

Figura 1
Árbol de problemas de la aplicación local de Full Suit



Fuente: Autor

En el árbol se observa que las causas se agrupan en tres frentes: las derivadas del diseño de la interfaz, las derivadas de la ausencia de funcionalidad local, y las derivadas de la limitación de plataforma. Los efectos, en cambio, confluyen en un mismo punto: la pérdida de eficiencia operativa del comercio que utiliza el sistema.

4. JUSTIFICACIÓN

La pasantía se orientó a optimizar un sistema del que dependen las operaciones diarias de numerosos comercios. La aplicación local de Full Suit es la herramienta con la que esos negocios venden, controlan inventarios, gestionan clientes y emiten los documentos de facturación electrónica que deben cumplir los estándares de la DIAN para tener validez fiscal (DIAN, s. f.).

Cuando un sistema de esa naturaleza presenta fallas de diseño, de funcionalidad o de sincronización, las consecuencias alcanzan al comerciante que factura con él y llegan hasta su relación con la autoridad tributaria, porque un documento emitido con valores inconsistentes retrasa la atención en el mostrador, puede ser rechazado durante la validación posterior y obliga entonces a rehacer el proceso completo con el cliente ya ausente del establecimiento. Los efectos se acumulan. Se traducen en errores de facturación con implicaciones tributarias, en descuadres de inventario y en pérdida de confianza hacia el producto. De ahí la necesidad de que la aplicación fuera confiable, sencilla de operar y capaz de crecer.

El aporte del trabajo se concentró en cuatro frentes. El primero fue mejorar la experiencia de uso mediante el rediseño de la aplicación y su adaptación a teléfonos y tabletas. El segundo, reducir la dependencia de la conexión a internet habilitando la gestión local de turnos, ventas y reportes. El tercero, fortalecer la integración entre la base de datos local y los servicios en la nube para evitar la pérdida o duplicación de información. El cuarto, hacer más competitivo el producto al ofrecer un punto de venta más flexible y adaptable a distintos escenarios de operación.

Desde el punto de vista técnico y profesional, la pasantía representó la oportunidad de aplicar conocimientos en desarrollo de aplicaciones multiplataforma con Flutter y Dart, tecnologías adoptadas ampliamente por su capacidad de operar de forma nativa en distintos sistemas y de mantener sincronización entre entornos con y sin conexión (Viscaino Quito & Serpa Andrade, 2022). El modelo de Flutter, que compila a código nativo y dibuja su propia interfaz en lugar de delegar en los componentes del sistema operativo, es lo que permite sostener una única base de código para escritorio y móvil sin renunciar al rendimiento (Windmill, 2020). El uso de bases de datos locales como SQLite permite garantizar la continuidad del servicio aun sin acceso a la red, y optimizar el almacenamiento y la recuperación de la información.

En ingeniería del software, intervenir un producto que ya está en producción, con clientes activos y sin poder detener la operación, se conoce como mantenimiento evolutivo, y es la fase que más esfuerzo concentra en la vida de un sistema (Sommerville, 2011; Pressman, 2010). Lo más difícil era no romper algo. Al llegar no había ningún tipo de documentación, ni la más mínima. A lo mucho, un comentario encima de cada método que decía qué hacía. Así que era muy común arreglar un problema en un lado y que ese supuesto arreglo generara uno o varios errores en otros. Probablemente por eso el cambio de precio fue más complejo de lo que suena, porque tocaba el núcleo del sistema y muchos módulos que giraban alrededor se rompieron con esos ajustes, como algunos reportes.

Se trabajó con SCRUM porque la empresa ya lo usaba y había que integrarse a su equipo. Hubo que adaptarse, aunque en la práctica era menos formal de lo que suena.

5. OBJETIVOS

5.1. Objetivo General

Ampliar una aplicación de punto de venta offline desarrollada en Flutter, mediante la refactorización de su base de código, la incorporación de nuevas funcionalidades y su

adaptación a dispositivos Android, con el fin de mejorar la eficiencia operativa e integrar la solución al ecosistema tecnológico de la empresa bajo un marco ágil SCRUM.

5.2. Objetivos Específicos

Tabla 2
Objetivos específicos

Nro.	Objetivo específico
1	Refactorizar la aplicación de escritorio en Flutter, mejorando la UI/UX, corrigiendo errores e implementando persistencia en SQLite, con el fin de habilitar la gestión de turnos de cajero y la generación de reportes de ventas en modo offline.
2	Desarrollar la versión móvil de la aplicación en Android con Flutter, aplicando diseño responsivo y reestructuración de módulos clave, para brindar una experiencia de usuario satisfactoria en dispositivos de menor tamaño.
3	Implementar un sistema de precios dinámicos en los módulos POS, capaz de recalcular en tiempo real impuestos y descuentos mediante fórmulas inversas, con el fin de garantizar exactitud en los valores finales y facilitar la correcta emisión de la factura electrónica a la DIAN.
4	Integrar las nuevas funcionalidades de la aplicación al flujo de trabajo SCRUM, sincronizando datos entre la aplicación offline, SQLite, la plataforma web y los servicios backend, para asegurar un ciclo de desarrollo paralelo, robusto y eficiente.

Fuente: Autor

Los objetivos anteriores se transcriben literalmente de la propuesta de pasantía aprobada. Su desarrollo se documenta en la sección siguiente, con un apartado dedicado a cada uno.

6. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

6.1. Listado de actividades realizadas en el proyecto

La tabla siguiente relaciona las actividades ejecutadas durante la pasantía, ordenadas de forma cronológica y con indicación del objetivo específico al que corresponde cada una. La última columna señala el apartado de este documento donde se presenta la evidencia del producto obtenido.

Las fechas se expresan en notación relativa de mes y semana, siguiendo el criterio del avance presentado durante la pasantía. Se adoptó esa notación porque el trabajo sobre varios de los objetivos se inició antes de que el convenio pudiera formalizarse, por razones administrativas ajenas al desarrollo, de modo que las fechas absolutas de ejecución no coinciden con las del documento contractual.

Los objetivos no se desarrollaron de forma secuencial. La propuesta los había proyectado como cuatro fases consecutivas, una detrás de otra, pero la dinámica real de un equipo que sostiene un producto en producción impuso otro orden, de modo que el trabajo sobre la versión

móvil comenzó mientras aún se refactorizaba la aplicación de escritorio, el sistema de precios dinámicos se extendió sobre ambas plataformas a medida que cada una quedaba disponible, y el cuarto objetivo atravesó todo el periodo por su naturaleza metodológica y de sincronización. Esa simultaneidad se refleja en la tabla.

Tabla 3
Listado consolidado de actividades realizadas

Descripción de la actividad	Obj.	Fecha inicio	Fecha entrega	Producto entregado
Análisis del código y detección de fallos	1	Mes 1, Sem 1	Mes 1, Sem 1	Bitácora técnica — Análisis de la arquitectura y línea base del proyecto
Creación de mockups para escritorio	1	Mes 1, Sem 1	Mes 1, Sem 2	Mockups en Figma — Elaboración de los mockups para escritorio
Configuración del entorno de compilación para Android	2	Mes 1, Sem 1	Mes 1, Sem 2	Proyecto compilando para Android — Configuración del entorno de compilación para Android
Incorporación al ciclo de reuniones diarias y al tablero de tareas	4	Mes 1, Sem 1	Mes 1, Sem 1	Participación en el flujo del equipo — Cómo funcionaba el proceso en la práctica
Refactorización de la interfaz en Flutter	1	Mes 1, Sem 2	Mes 2, Sem 3	Interfaz de escritorio actualizada — Refactorización de la interfaz y corrección de fallos
Adopción del esquema de ramas por funcionalidad y corrección	4	Mes 1, Sem 2	Mes 2, Sem 1	Ramas integradas a desarrollo — Estrategia de ramas
Implementación de validaciones y carga continua	1	Mes 1, Sem 3	Mes 2, Sem 2	Formularios y listados funcionales — Refactorización de la interfaz y corrección de

Construcción del módulo POS móvil	2	Mes 1, Sem 3	Mes 2, Sem 2	fallos Módulo pos-mobile operativo — Construcción del módulo POS móvil
Diálogo de cierre de venta y estado de cuenta en móvil	2	Mes 2, Sem 1	Mes 2, Sem 3	Flujo de cobro móvil funcional — Construcción del módulo POS móvil
Desarrollo del módulo de turnos sobre SQLite	1	Mes 2, Sem 2	Mes 2, Sem 4	Módulo de turnos operativo — Desarrollo del módulo de gestión de turnos
Captura global de excepciones y registro de peticiones	4	Mes 2, Sem 3	Mes 2, Sem 4	Archivo de diagnóstico local — Observabilidad como respuesta a la ausencia de pruebas
Corrección del acceso a la red en compilación de distribución	2	Mes 2, Sem 4	Mes 2, Sem 4	APK con conectividad verificada — Corrección del acceso a la red en la compilación de distribución
Encadenamiento de la sincronización con el cierre de turno	4	Mes 2, Sem 4	Mes 3, Sem 1	Cierre con subida forzada — Sincronización entre la aplicación offline y los servicios del backend
Construcción del módulo POS restaurante móvil	2	Mes 3, Sem 1	Mes 3, Sem 4	Módulo pos-restaurant-mobile operativo — Construcción del módulo POS restaurante móvil
Análisis del modelo de valores del artículo y del flujo de facturación	3	Mes 3, Sem 1	Mes 3, Sem 2	Documento de análisis — El contexto que condiciona el problema
Planteamiento de las fórmulas inversas en conjunto con la	3	Mes 3, Sem 2	Mes 3, Sem 4	Fórmulas acordadas entre clientes — El despeje inverso

plataforma web

Implementación del despeje inverso y del campo de precio editable	3	Mes 3, Sem 3	Mes 4, Sem 3	Recálculo funcional en POS comercial — Recálculo en tiempo real
Consulta del histórico de turnos y reporte de cierre	1	Mes 4, Sem 1	Mes 4, Sem 3	Recibo de cierre de caja — Generación de reportes de venta sin conexión
Extensión del recálculo al POS restaurante y a las adiciones	3	Mes 4, Sem 2	Mes 5, Sem 1	Recálculo funcional en POS restaurante — Recálculo en tiempo real
Precuenta y comanda para dispositivos móviles	2	Mes 4, Sem 2	Mes 5, Sem 1	Documentos PDF desde móvil — Construcción del módulo POS restaurante móvil
Exportación de la base de datos y de los registros de diagnóstico	4	Mes 4, Sem 2	Mes 5, Sem 2	Diagnóstico remoto habilitado — Observabilidad como respuesta a la ausencia de pruebas
Vista ampliada de descomposición de valores	3	Mes 4, Sem 4	Mes 5, Sem 2	Interfaz de verificación detallada — Interfaz de verificación detallada
Normalización del redondeo en cálculo, persistencia y envío	3	Mes 5, Sem 1	Mes 5, Sem 3	Valores consistentes en toda la cadena — Normalización del redondeo
Pruebas cruzadas con la plataforma web y validación contra la DIAN	3	Mes 5, Sem 2	Mes 6, Sem 3	Facturas aceptadas por la DIAN — Validación cruzada entre clientes
Versionado del esquema de la base de datos local	1	Mes 5, Sem 3	Mes 5, Sem 4	Migración de esquema sin pérdida de datos — Persistencia en SQLite y evolución del esquema

Corrección de los tres defectos de exactitud detectados	3	Mes 5, Sem 4	Mes 6, Sem 2	Valores fiscales correctos — Corrección de defectos de exactitud
Gestor centralizado de sesión y manejo de token expirado	4	Mes 6, Sem 2	Mes 7, Sem 1	Validación de sesión unificada — Sincronización entre la aplicación offline y los servicios del backend
Rediseño de la sincronización como proceso de siete pasos	4	Mes 6, Sem 3	Mes 7, Sem 2	Sincronización observable — Sincronización entre la aplicación offline y los servicios del backend
Reconciliación de duplicados y turnos cerrados	4	Mes 7, Sem 1	Mes 7, Sem 3	Fallo parcial tolerante — Sincronización entre la aplicación offline y los servicios del backend
Integridad transaccional en el registro de ventas	1	Mes 7, Sem 2	Mes 7, Sem 3	Escritura atómica de la venta — Persistencia en SQLite y evolución del esquema
Formalización de los procedimientos de liberación y de versionado	4	Mes 7, Sem 4	Mes 8, Sem 1	Documentos de procedimiento — Formalización de procedimientos

Fuente: Autor

En total se relacionan treinta y una actividades, distribuidas entre los cuatro objetivos específicos. Ocho corresponden al primero, seis al segundo, ocho al tercero y nueve al cuarto. Esa distribución no refleja el volumen de código de cada objetivo, ya que actividades como la refactorización de la interfaz o la construcción del módulo POS restaurante móvil concentran mucho más trabajo que otras de la misma tabla.

Además de las anteriores, durante el periodo se atendieron actividades propias de la operación de la empresa que no corresponden a los objetivos de la pasantía. La más extensa fue la integración de una báscula digital por puerto serie para la venta de productos a granel. Se

relacionan aquí por transparencia, aunque su desarrollo no se documenta en este informe por quedar fuera del alcance acordado.

6.2. Herramientas utilizadas en el desarrollo del proyecto

Las versiones que se relacionan a continuación corresponden a las efectivamente resueltas en el proyecto, tomadas del archivo de bloqueo de dependencias y de la configuración de compilación, no a los rangos declarados en el manifiesto. Hay dos grupos. Las versiones del marco de trabajo, del lenguaje, del sistema de compilación y de las bibliotecas están fijadas por el propio proyecto y permanecieron invariables durante todo el periodo, de modo que cualquiera que compile el repositorio obtiene las mismas. Las de los entornos de edición y de las herramientas de control de versiones corresponden en cambio a la instalación del equipo de desarrollo, que se actualiza de forma continua, y no condicionan el resultado del producto.

6.2.1. Entorno de desarrollo y compilación

Tabla 4
Herramientas del entorno de desarrollo y de compilación

Herramienta utilizada	Versión	Para qué la utilizó
Flutter	3.7.9	Marco de trabajo multiplataforma con el que se construyó la aplicación para Windows y Android
Dart	2.19.6	Lenguaje de programación del proyecto
FVM (Flutter Version Management)	—	Fijación de la versión de Flutter del proyecto, para que todo el equipo compilara sobre el mismo entorno
Visual Studio Code	1.132.0	Entorno de escritura, depuración y ejecución del código
Android Studio	2025.1	Herramientas del SDK de Android y emulador de dispositivo
Git	2.51.0	Control de versiones local, gestión de ramas por funcionalidad y de correcciones urgentes
Gradle	7.4	Sistema de compilación del proyecto Android
Android SDK	API 21 (Android 5.0) como mínimo	Plataforma objetivo de la versión móvil
Inno Setup	6.5.1	Ensamblado del instalador ejecutable de la versión de

escritorio. La configuración era anterior al periodo de la pasantía

Fuente: Autor

6.2.2. Bibliotecas con incidencia en los objetivos

El proyecto declara alrededor de treinta dependencias, la mayoría de las cuales resuelve asuntos de presentación sin relación con el trabajo aquí documentado. La tabla siguiente recoge únicamente aquellas de las que depende algún resultado descrito en este informe.

Tabla 5
Bibliotecas de las que dependen los resultados documentados

Herramienta utilizada	Versión	Para qué la utilizó
sqlite / sqlite_common_ff	2.2.8+4 / 2.2.5	Acceso a la base de datos SQLite local, en Android y en las plataformas de escritorio respectivamente
provider	6.1.5	Gestión del estado de la aplicación y notificación de cambios a la interfaz
http_interceptor	1.0.2	Interceptación de las peticiones al backend, para el registro de diagnóstico y el manejo de sesión expirada
connectivity_plus	3.0.6	Detección de disponibilidad de red antes de intentar la sincronización
flutter_secure_storage	5.1.2	Almacenamiento cifrado del token de autenticación
pdf / printing	3.10.1 / 5.10.4	Construcción de la factura, la precuenta y la comanda, y su envío a la impresora térmica
syncfusion_flutter_pdf	20.4.54	Generación del recibo de cierre de turno con discriminación de impuestos
flutter_libserialport	0.4.0	Comunicación por puerto serie con la báscula digital para productos a granel
intl	0.17.0	Formato de moneda y fecha según la configuración regional colombiana
uuid	3.0.7	Generación de los identificadores locales de venta, turno y línea de detalle

Fuente: Autor

6.2.3. Gestión del trabajo

Tabla 6
Herramientas de gestión y colaboración

Herramienta utilizada	Versión	Para qué la utilizó
GitHub	—	Repositorio remoto, integración de ramas y publicación de versiones
GitHub Projects	—	Tableros de planificación por sprint y de gestión de defectos reportados
Figma	—	Elaboración de los mockups de las pantallas sin propuesta previa y consulta de los entregados por el equipo de diseño

Fuente: Autor

6.3. Descripción general del sistema intervenido

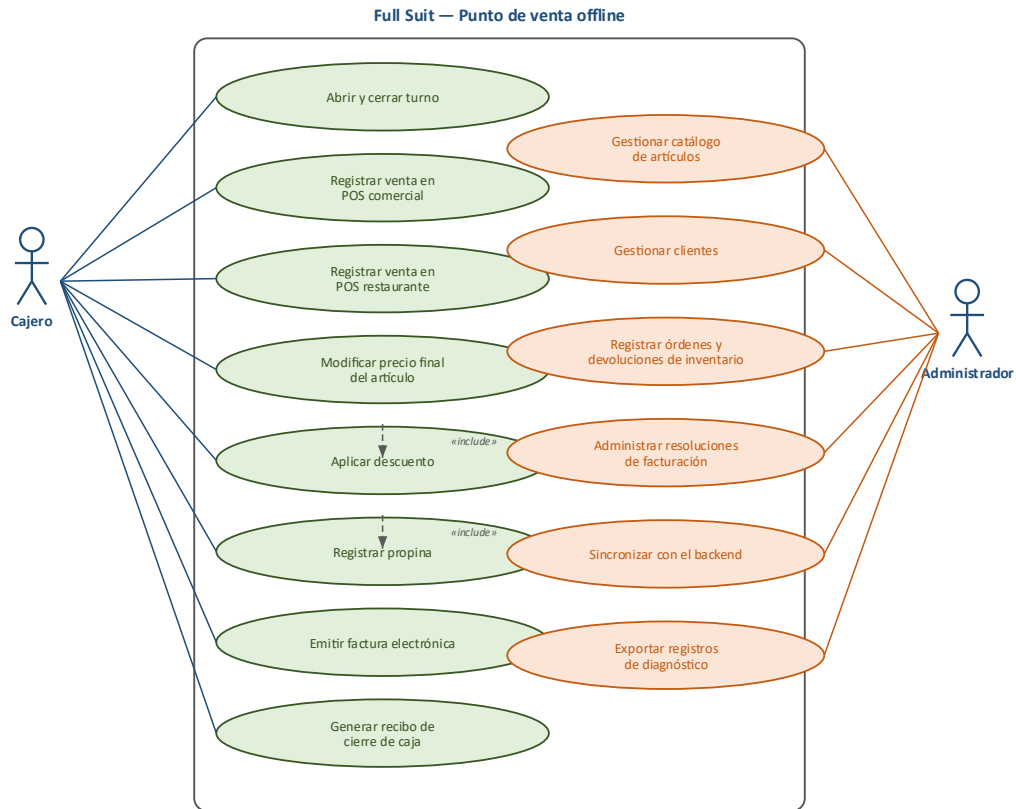
Antes del desarrollo de cada objetivo se presenta el sistema sobre el que se trabajó, porque varias de las decisiones técnicas de esos apartados solo se entienden a la luz de su arquitectura y de las restricciones que imponía.

Full Suit Offline es la aplicación de punto de venta que la empresa instala en los establecimientos de sus clientes. Su característica determinante es que debe seguir operando sin conexión a internet. Un comercio no puede dejar de facturar porque se caiga la red, de modo que la aplicación mantiene una base de datos local con el catálogo, los clientes, las resoluciones de facturación y las ventas, y reconcilia esa información con los servicios de la empresa cuando recupera conectividad.

6.3.1. Actores y funciones del sistema

La aplicación distingue dos perfiles de uso. El cajero opera el punto de venta durante la jornada, y el administrador gestiona el catálogo, el inventario y la configuración de facturación. Ambos comparten la aplicación y el mismo conjunto de datos locales.

Figura 2
Diagrama de casos de uso del sistema



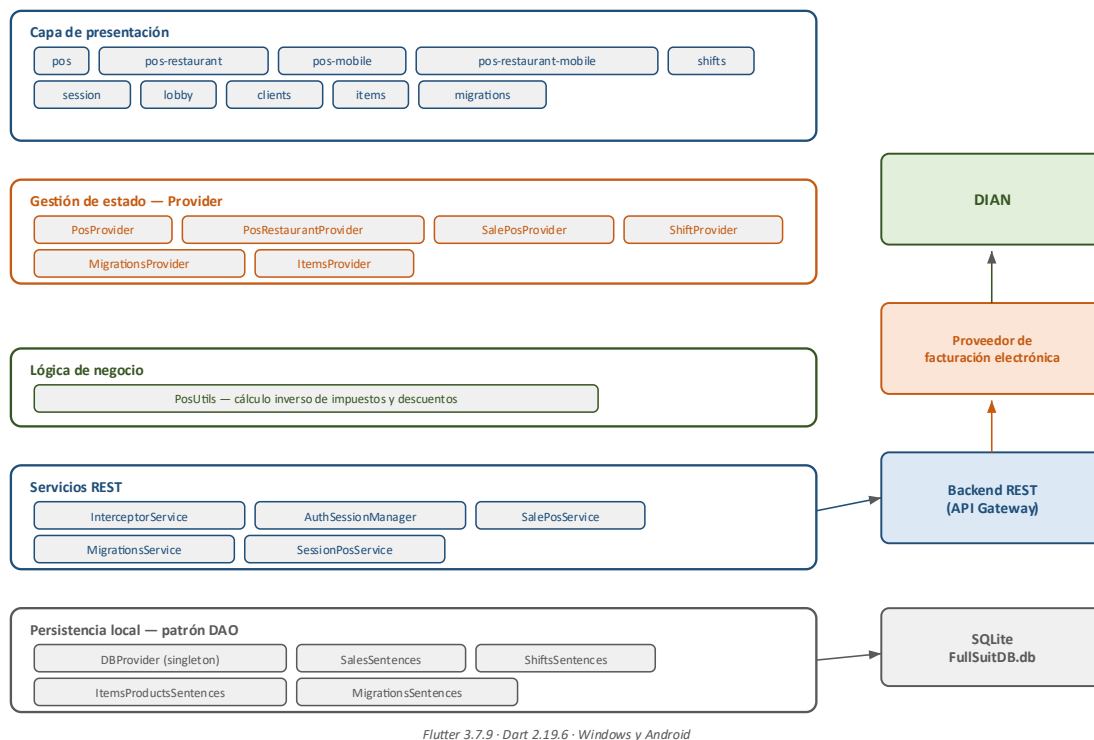
Fuente: Autor

En el diagrama se observa que las funciones del cajero giran alrededor del ciclo de la jornada: abrir turno, registrar ventas, cobrar y cerrar caja con su respectivo recibo. Las del administrador corresponden a la configuración que hace posible ese ciclo. Dos de los casos de uso del cajero (modificar el precio final del artículo y registrar propina) fueron incorporados durante la pasantía y se desarrollan en el tercer objetivo específico. La generación del recibo de cierre pertenece al primero.

6.3.2. Arquitectura de la aplicación

El proyecto está construido en Flutter sobre Dart, con una organización por capas que se mantuvo durante toda la intervención. La separación en capas acota el efecto de un cambio. Una modificación en la presentación no debería obligar a tocar la persistencia, ni a la inversa (Bass et al., 2012), y describir el sistema desde varias vistas complementarias (la lógica de los módulos, el despliegue físico y los escenarios de uso) responde a esa misma intención de separar preocupaciones que propuso Kruchten (1995).

Figura 3
Arquitectura de componentes de la aplicación



Fuente: Autor

La capa de presentación agrupa los módulos funcionales, cada uno en su propio directorio. Sobre ella opera la gestión de estado mediante el paquete Provider, donde cada módulo cuenta con uno o varios proveedores que mantienen el estado en memoria y notifican los cambios a la interfaz. El esquema corresponde al patrón observador, en el que un objeto notifica automáticamente a sus dependientes cuando cambia su estado (Gamma et al., 1995).

La lógica de cálculo está en una clase de utilidades aparte de la interfaz, que ya existía separada antes de la pasantía. Tenía métodos rotos, y con el requerimiento del cambio de precio hubo que refactorizarla casi en su totalidad. Como los cuatro módulos de venta (comercial y restaurante, en escritorio y en móvil) usan esa misma clase, todos hacen el mismo cálculo.

Los servicios REST del backend se consumen a través de un interceptor por el que pasan todas las peticiones. Este agrega el token y los datos de la sucursal y la organización, deja registro de cada petición y respuesta en un archivo local y, si el token expiró, limpia la sesión y manda al usuario al login. La persistencia sigue el patrón de objeto de acceso a datos, que separa el acceso al almacenamiento del resto de la aplicación (Alur et al., 2003). Hay un proveedor único, DBProvider, donde están registradas todas las tablas del sistema, y para cada

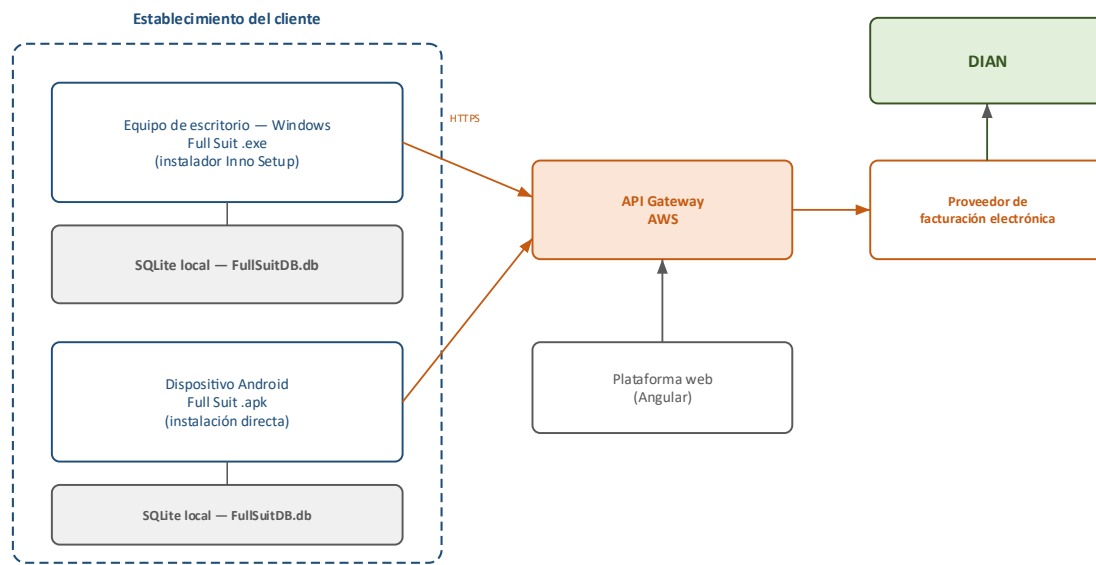
tabla hay un archivo en la capa de persistencia con las sentencias SQL que se usan. Ese proveedor único es una instancia compartida en toda la aplicación, esto es, el patrón de instancia única descrito por Gamma et al. (1995), decisión razonable cuando el recurso administrado es un archivo de base de datos que no admite conexiones concurrentes descontroladas.

El proyecto se mantuvo sobre Flutter 3.7.9 y Dart 2.19.6 (Flutter, s. f.; Dart, s. f.) durante todo el periodo. Es una versión publicada a comienzos de 2023, de modo que el trabajo se realizó sobre un entorno que ya llevaba más de dos años sin actualizarse, lo cual limitó las interfaces de programación disponibles. Eso es deuda técnica, porque se pospone el costo de actualizar y el aplazamiento acumula intereses, en forma de funcionalidades que no están disponibles y de una migración futura que cada mes resulta más cara (Cunningham, 1992).

6.3.3. Despliegue y entorno de operación

La distribución del producto se realiza de forma directa. La versión de escritorio se entrega como instalador ejecutable para Windows, y la versión Android como archivo de instalación que el cliente recibe e instala en su dispositivo, sin pasar por ninguna tienda de aplicaciones, decisión que simplifica la publicación y evita los tiempos de revisión de la plataforma, pero que traslada al equipo la responsabilidad de avisar de cada versión nueva, de verificar que el comercio la haya instalado y de sostener en producción instalaciones que pueden quedar varias versiones atrás sin que nadie lo advierta hasta que aparece un fallo.

Figura 4
Diagrama de despliegue del sistema



La aplicación opera sin conexión sobre la base local y reconcilia con el backend al recuperar red.

Fuente: Autor

Cada instalación mantiene su propia base de datos local. La comunicación con la empresa ocurre contra un servicio de puerta de enlace, que expone los servicios del backend. La misma organización puede estar operando simultáneamente desde la plataforma web, que consume esos mismos servicios desde una base de código independiente, y esa concurrencia entre clientes es la que origina buena parte de los problemas de sincronización que se abordan en el cuarto objetivo.

La emisión de la factura electrónica no ocurre directamente contra la DIAN. La venta se transmite a un servicio de la empresa que a su vez la presenta ante un proveedor tecnológico autorizado, y es ese proveedor quien la radica ante la autoridad tributaria. La aplicación, por tanto, no conversa con la DIAN, pero la exactitud de los valores que calcula determina si el documento será aceptado o rechazado al final de esa cadena.

6.3.4. Levantamiento de requerimientos y realimentación

Los requerimientos llegaban al equipo por dos vías, ambas registradas en tableros del repositorio.

La primera corresponde a las solicitudes de funcionalidad, redactadas como historias de usuario en la forma «como vendedor se solicita que...» y asignadas a un sprint con prioridad y estimación de tamaño. La segunda corresponde a los defectos reportados desde los

establecimientos en operación, que se registraban con fecha de descubrimiento, responsable asignado, fecha estimada de solución y fecha de entrega.

Durante el periodo de la pasantía quedaron asignados al autor treinta y cinco requerimientos de desarrollo y noventa y tres reportes de defecto, estos últimos correspondientes en su práctica totalidad a la aplicación móvil. Ese segundo conjunto constituyó el mecanismo principal de realimentación del producto, ya que el proyecto no contaba con un proceso formal de pruebas previas al despliegue. Las condiciones que originaron esa carencia se detallan en las conclusiones.

6.4. Desarrollo del objetivo específico 1

"Refactorizar la aplicación de escritorio en Flutter, mejorando la UI/UX, corrigiendo errores e implementando persistencia en SQLite, con el fin de habilitar la gestión de turnos de cajero y la generación de reportes de ventas en modo offline."

Este objetivo fue el más amplio de los cuatro en volumen de código, porque tocaba toda la aplicación: había que hacer los cambios gráficos de cada sección, revisar que cada campo de los formularios tuviera sus validaciones y corregir los errores de paginación de cada lista. En complejidad técnica y conceptual, en cambio, el más exigente fue el del cambio de precio.

6.4.1. Análisis de la arquitectura y línea base del proyecto

El código de Full Suit Offline acumulaba cerca de tres años de desarrollo por parte de otros integrantes del equipo. La aplicación usaba Flutter en el frontend, sincronizaba contra un backend remoto y tenía una base de datos SQLite local para operar sin conexión. La revisión fue gradual y los problemas fueron saliendo de a poco. Algunos se veían desde el propio código, pero los más evidentes se notaban al usar la aplicación, pasando por los distintos módulos, formularios y componentes.

En los listados, la paginación fallaba al avanzar o retroceder, y a veces los cambios hechos sobre un registro no se reflejaban en pantalla. Esto se notaba sobre todo en el catálogo de artículos, que es la pantalla más usada durante una jornada de venta.

Los formularios casi no validaban nada. En los campos numéricos muchas veces se podían escribir letras y caracteres especiales como @, ! o paréntesis, prácticamente cualquier carácter, y en los campos de solo letras pasaba lo mismo. Realmente sucedía en casi todos los formularios, y esos datos quedaban guardados tal cual en la base local.

El rediseño gráfico lo solicitó la empresa junto con la versión móvil, porque la interfaz de escritorio no correspondía con la línea gráfica vigente.

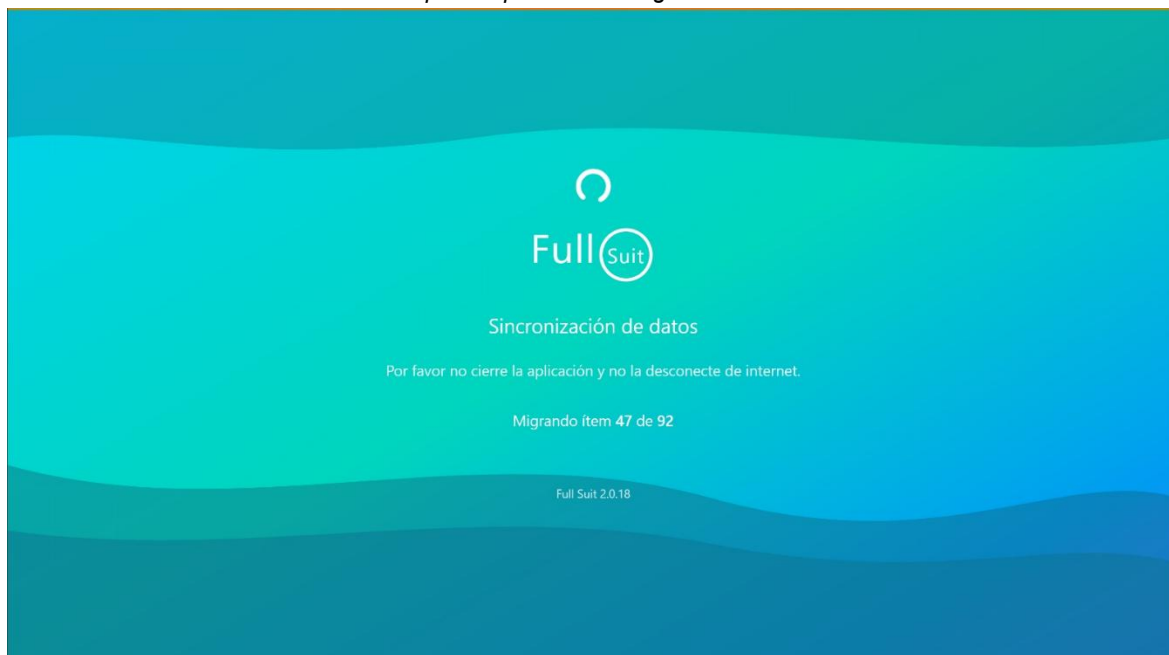
6.4.2. Elaboración de los mockups para escritorio

Con el fin de asegurar una interfaz coherente y adaptada al entorno de escritorio, el trabajo partió de un conjunto de mockups que definieron la línea visual, la distribución de los componentes y la organización general de la navegación. Una parte de esos diseños fue entregada por el equipo de diseño de la empresa. El resto se elaboró durante la pasantía, sobre la misma línea gráfica, para cubrir las pantallas que no contaban con propuesta previa.

En ambos casos el criterio fue el mismo: mejorar la claridad con que se presenta la información, facilitar la interacción y ordenar los elementos de manera que el flujo de uso resultara más directo. Esos diseños sirvieron como referencia durante toda la refactorización posterior.

Figura 5

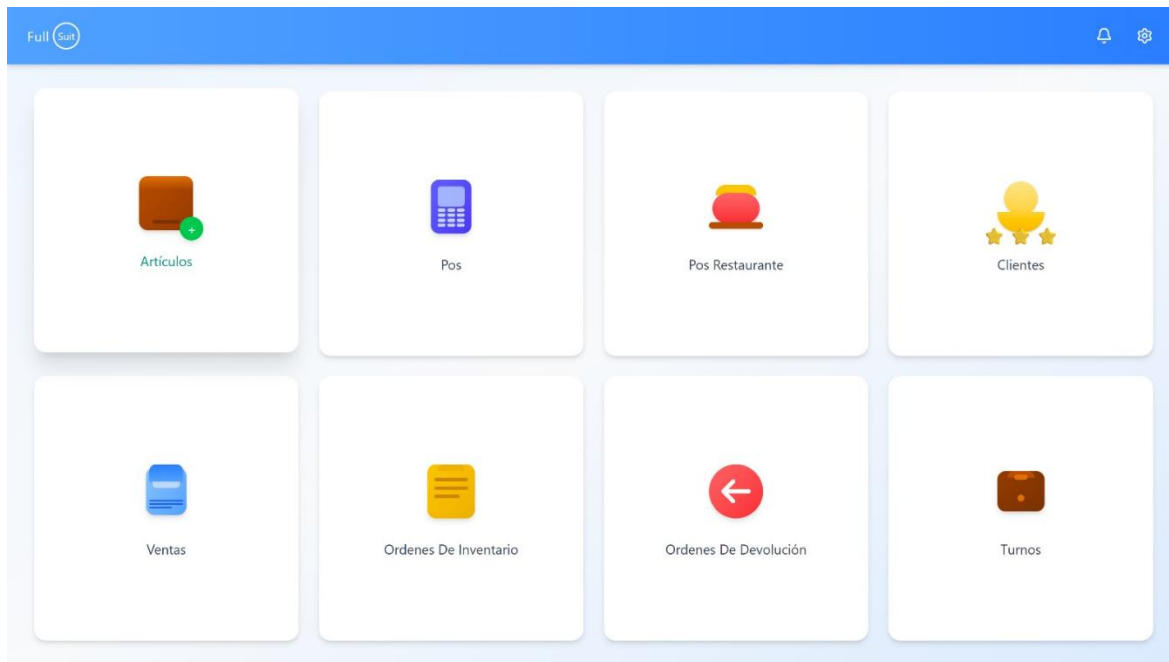
Mockup de la pantalla de migración de datos



Fuente: Autor

Figura 6

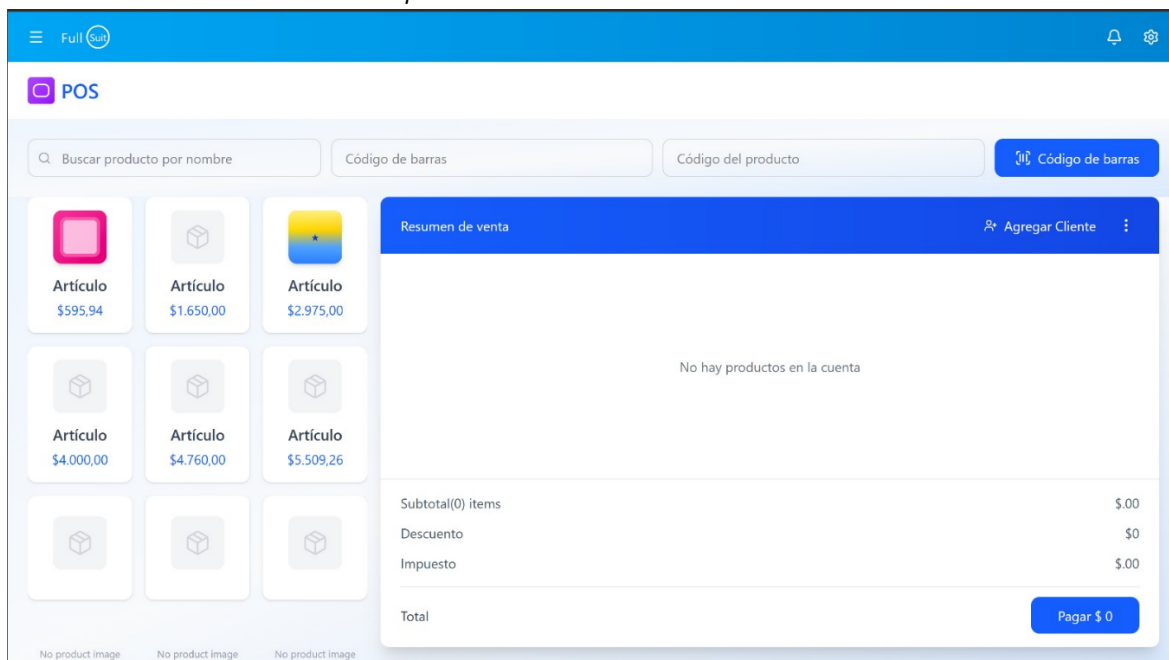
Mockup del nuevo lobby de la aplicación



Fuente: Autor

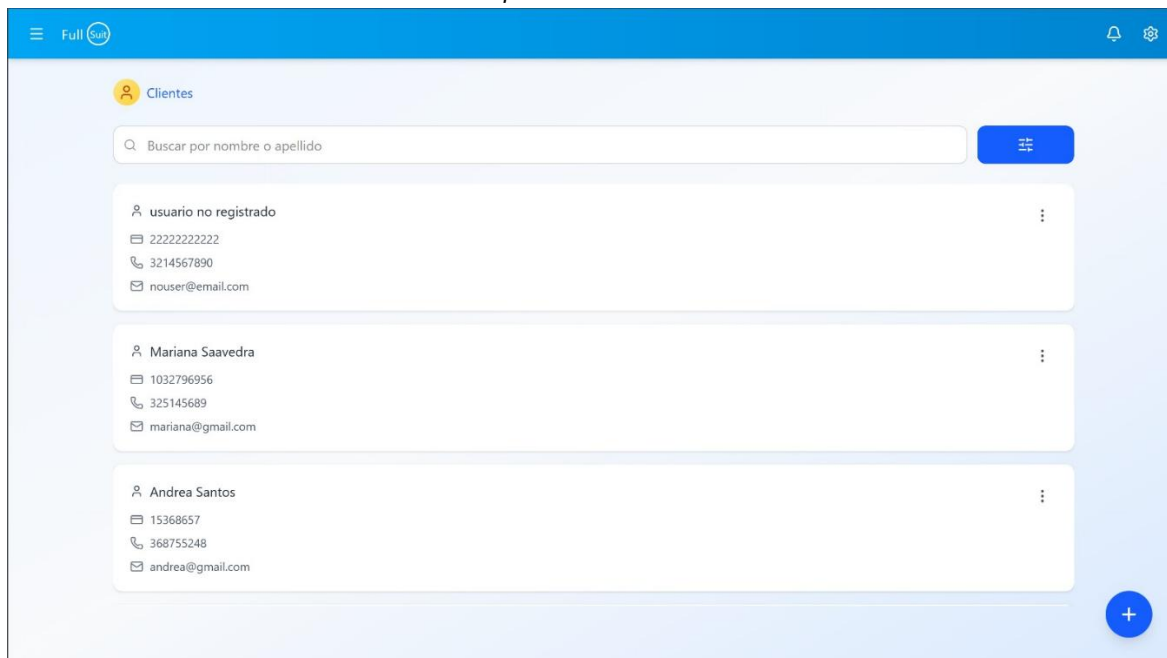
El lobby merece una aclaración. Se trata de una pantalla que no existía: la versión anterior ingresaba directamente al punto de venta después del acceso, sin ofrecer al usuario una vista intermedia desde la cual escoger módulo. Su incorporación reorganizó por completo la jerarquía de navegación de la aplicación.

Figura 7
Mockup del POS comercial en vista de escritorio



Fuente: Autor

Figura 8
Mockup del listado de clientes



Fuente: Autor

6.4.3. Refactorización de la interfaz y corrección de fallos

A partir de los mockups se rehizo cada pantalla. Además de los ajustes visuales, se revisaron los flujos internos y se corrigieron fallos acumulados, siguiendo el criterio de que la refactorización debe modificar la estructura interna del código sin alterar su comportamiento observable (Fowler, 2018). Los criterios de organización visual y de retroalimentación al usuario se apoyaron en los principios de usabilidad de interfaz (Nielsen, 1994). En el código, el trabajo partió de la idea de que la legibilidad determina el costo de toda modificación futura, porque un programa se lee muchas más veces de las que se escribe (McConnell, 2004).

Una de las primeras intervenciones consistió en reemplazar el sistema de paginación tradicional, que presentaba fallos recurrentes, por un mecanismo de carga continua. La modificación hizo más fluido el desplazamiento, redujo los errores al cargar información adicional y facilitó la actualización automática cuando se modificaban registros. El cambio se aplicó al catálogo de artículos, al directorio de clientes, al histórico de ventas, a las órdenes de inventario y a las devoluciones, en todos los casos llevando la paginación a las consultas SQLite con límite y desplazamiento en lugar de cargar la tabla completa en memoria.

Ese cambio arrastró consigo una cadena de problemas que tomó varios intentos resolver. Al desacoplar la limpieza de filtros de la recarga del listado, con el propósito de que cada pantalla decidiera por su cuenta cuándo volver a consultar la base de datos, las vistas que dependían

del comportamiento anterior dejaron de refrescarse, de modo que hubo que introducir un mecanismo que recordara desde qué contexto se había invocado la carga (el punto de venta o la administración de inventario) para que al limpiar los filtros la pantalla recuperara la vista que le correspondía y no la de la última consulta ejecutada. Fueron tres iteraciones. Las tres quedaron registradas en el control de versiones con el título de solución tentativa, lo que da una idea de cuánto costó aislar la causa.

Los formularios también se validaron. En el diálogo de creación de clientes, que es el más usado, se agregaron longitudes mínimas y máximas por campo, formato de correo, documento numérico dentro de un rango válido y teléfono de siete a diez dígitos. En la práctica no todas las validaciones quedaron bien a la primera: en algunos campos numéricos se bloqueaban las letras, pero se podía escribir la letra e, porque el sistema la tomaba como exponente. Cosas así hicieron que algunos formularios necesitaran varias pasadas de QA.

Lo mismo se hizo con la creación de resoluciones de facturación, donde se verifica el número, el prefijo, la clave técnica, los rangos de numeración y que las fechas de vigencia sean coherentes. Si una resolución quedaba mal creada y se enviaba una factura electrónica con ella, la DIAN rechazaba la venta indicando que la resolución tenía algún problema, y como sus respuestas muchas veces no eran muy intuitivas, no era fácil saber la causa.

La aplicación también se preparó para varias plataformas, con condiciones que muestran componentes distintos según el sistema operativo. Solo los POS cambian de estructura en móvil, porque toda la información que tienen en una sola vista no cabía en una pantalla pequeña, y el resto de pantallas mantiene la misma estructura. En lo visual, lo que más se notó fue algo simple: pasar de fondos monocromáticos a fondos con diseño. Lo que más trabajo dio fue corregir todos los errores de paginación y de filtros que había.

A continuación se presenta la comparación entre el estado anterior y el resultado de la refactorización en las pantallas más representativas.

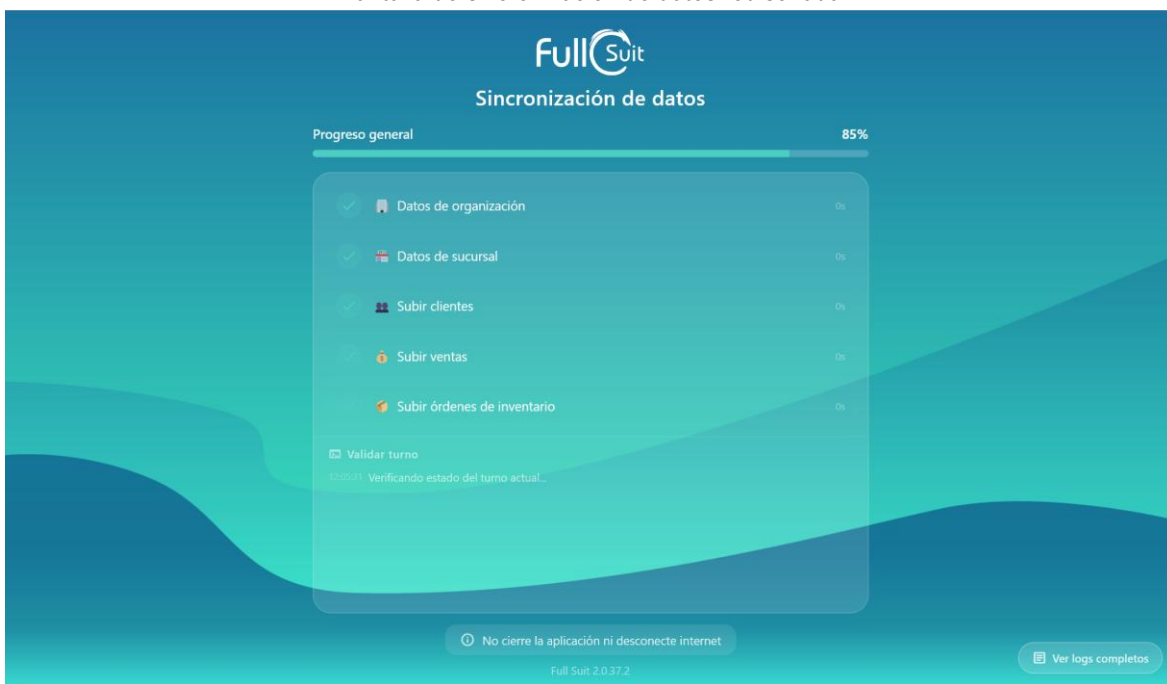
Figura 9
Pantalla de sincronización de datos antes de la refactorización



Fuente: Autor

Figura 10

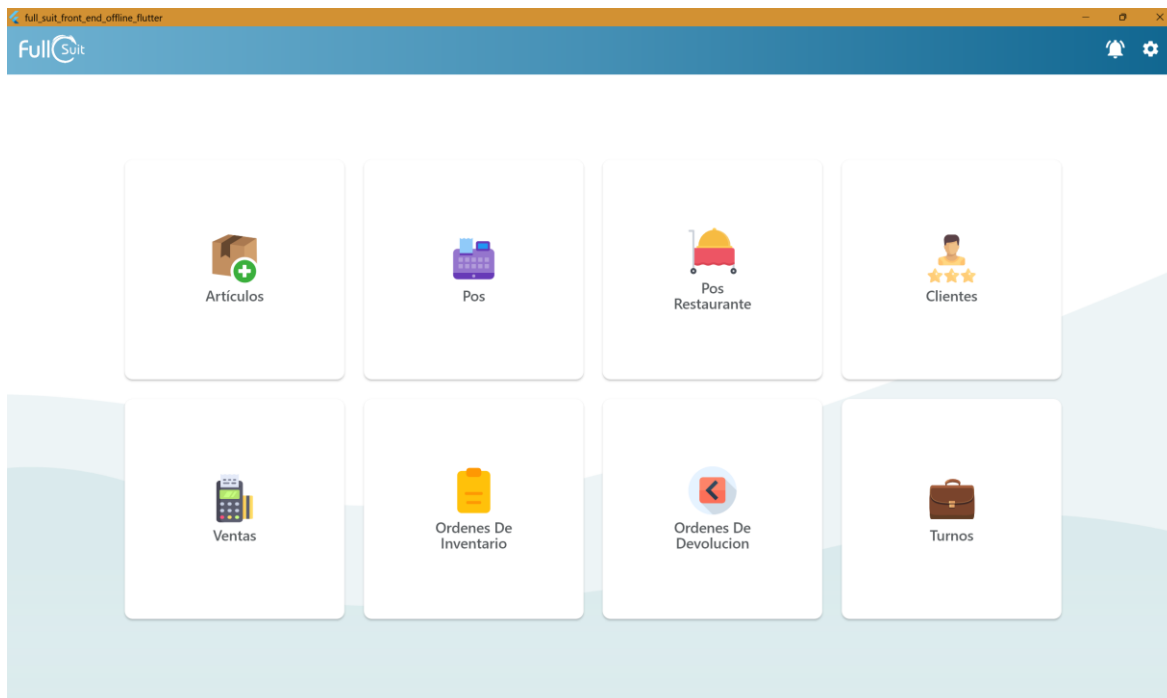
Pantalla de sincronización de datos rediseñada



Fuente: Autor

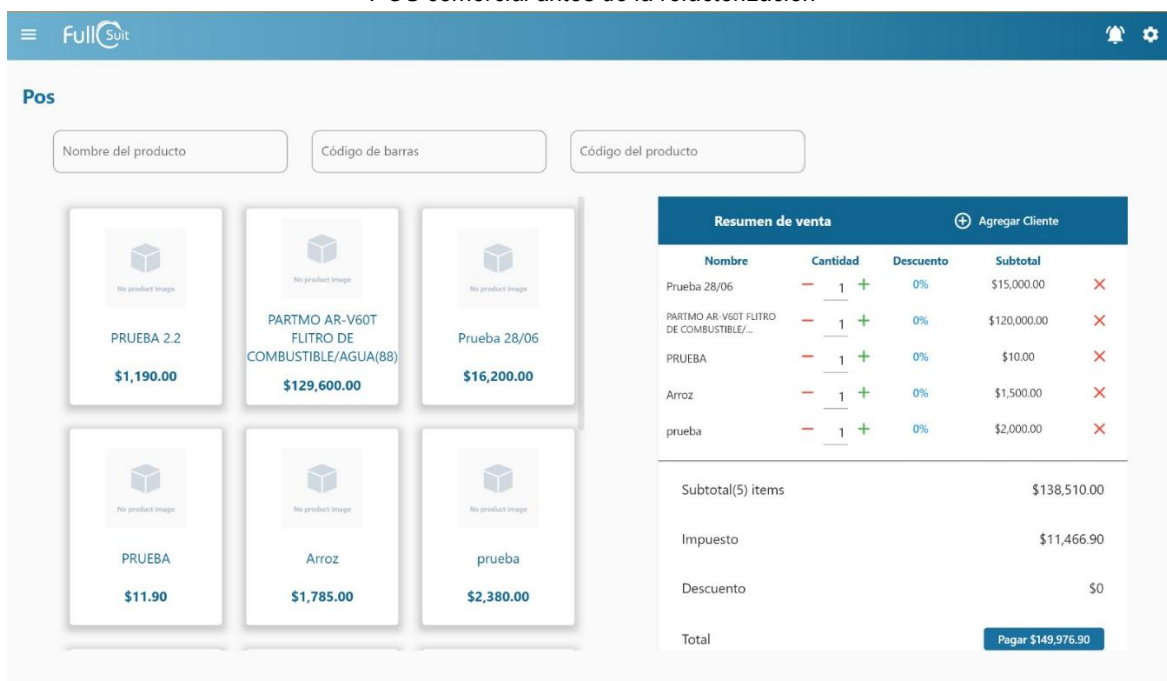
Figura 11

Nueva pantalla de lobby



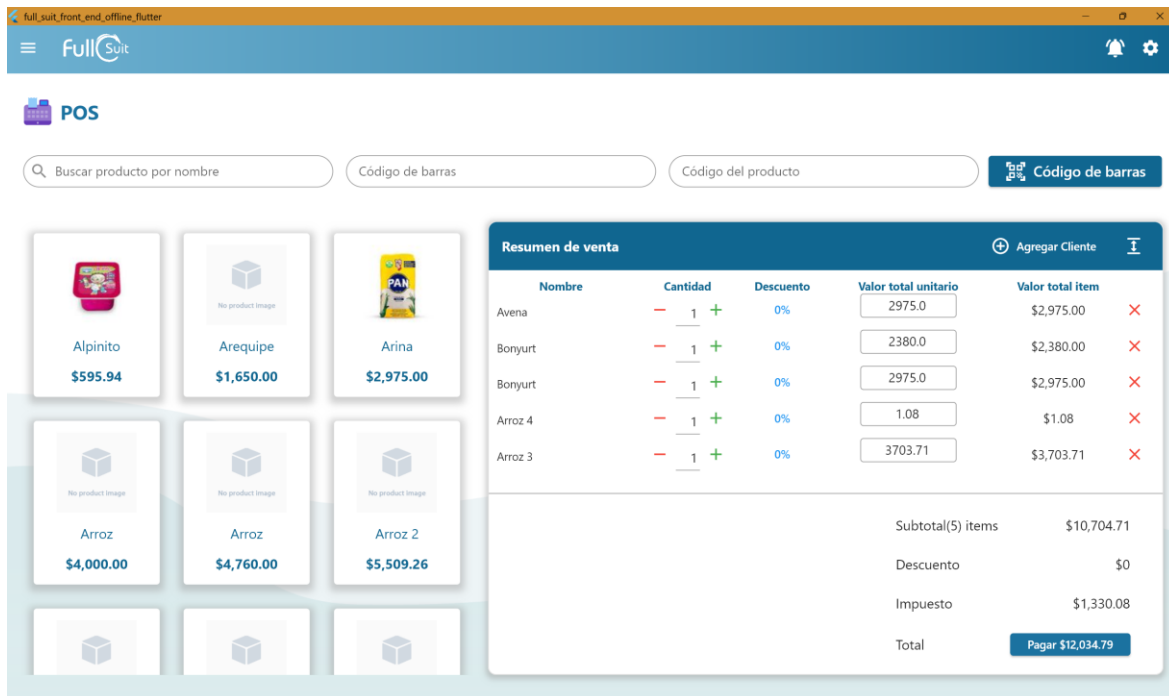
Fuente: Autor

Figura 12
POS comercial antes de la refactorización



Fuente: Autor

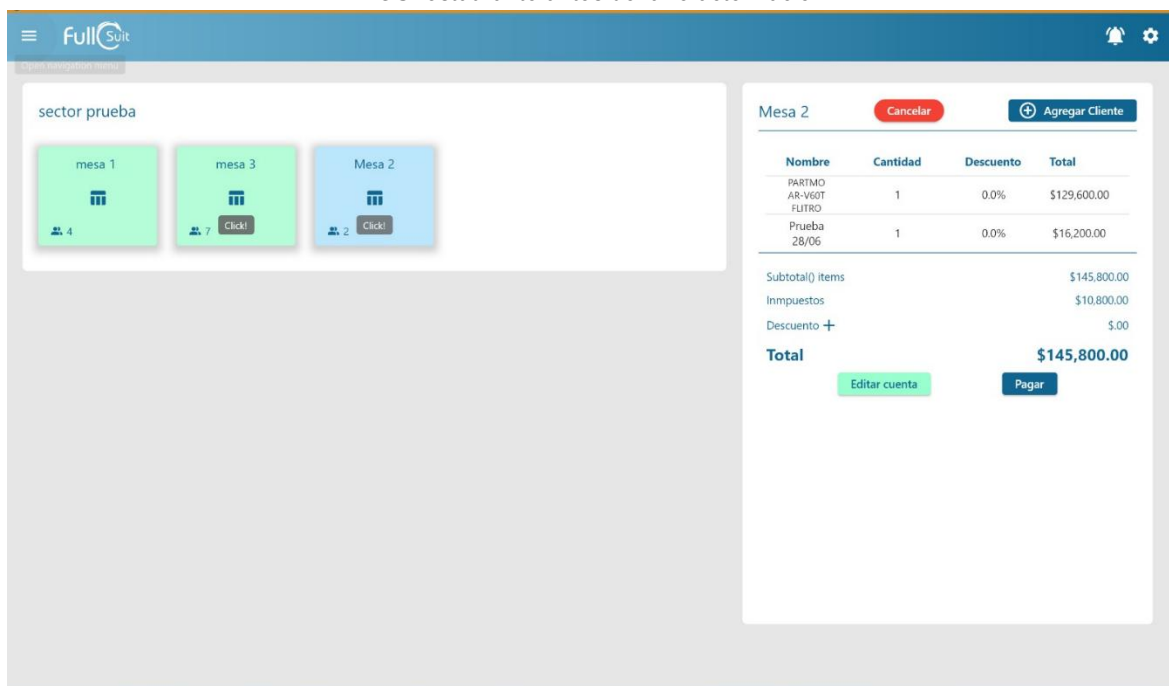
Figura 13
POS comercial rediseñado



Fuente: Autor

Figura 14

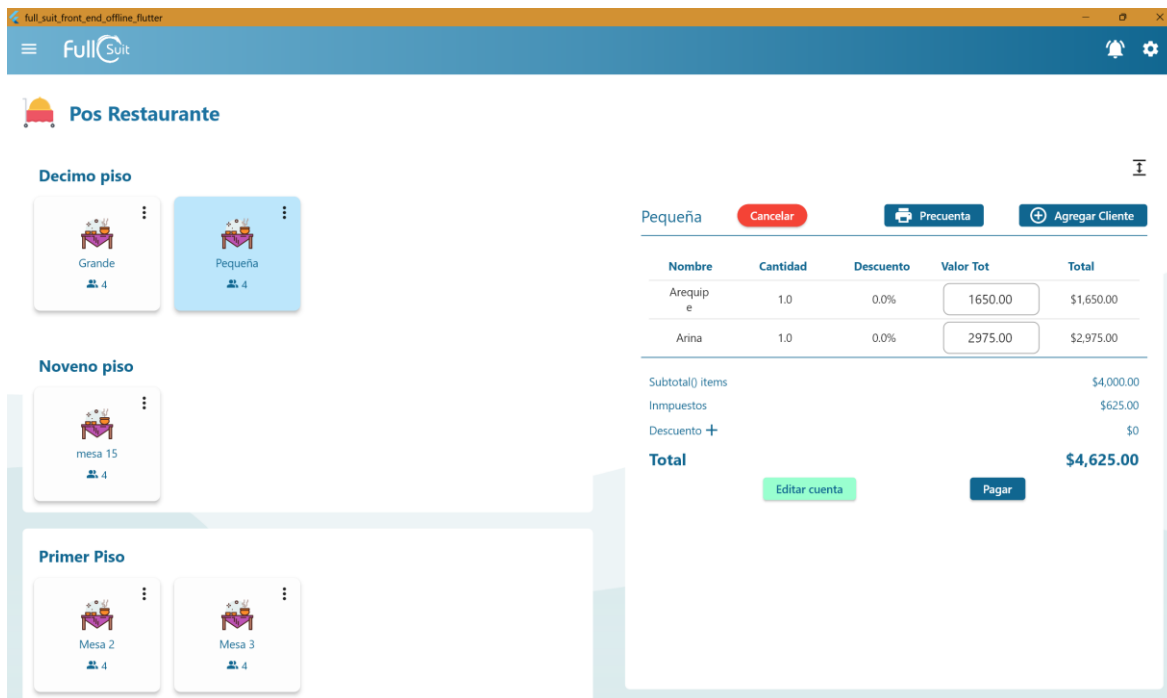
POS restaurante antes de la refactorización



Fuente: Autor

Figura 15

POS restaurante rediseñado



Fuente: Autor

En las comparaciones anteriores se aprecia el cambio de una interfaz de tablas densas y controles pequeños a una organizada en tarjetas, con mayor separación entre elementos y una jerarquía visual que distingue la acción principal de las secundarias.

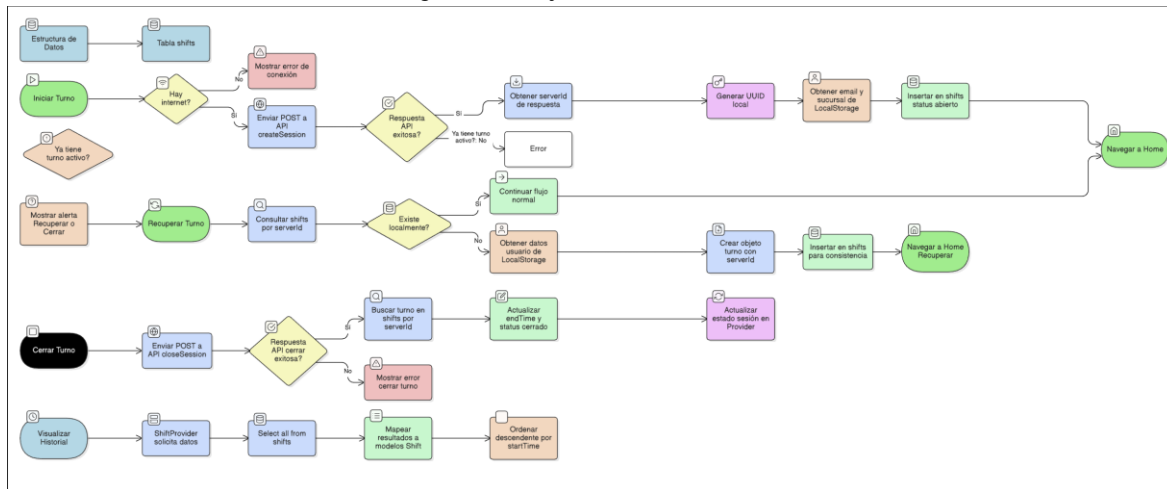
6.4.4. Desarrollo del módulo de gestión de turnos

Se construyó desde cero un módulo para la gestión de turnos de cajero, funcionalidad que no existía en la versión previa. El componente se diseñó bajo un enfoque de operación sin conexión, apoyado en SQLite como almacenamiento local, para que la información se conserve aunque el establecimiento pierda acceso a internet.

El funcionamiento se estructuró de la siguiente manera. La apertura del turno registra en la base local el usuario, la sucursal y la fecha y hora de inicio, junto con un identificador propio. A partir de ese momento cada venta queda asociada al turno activo. El cierre almacena la hora de salida y actualiza el estado del registro para dejar constancia del periodo trabajado.

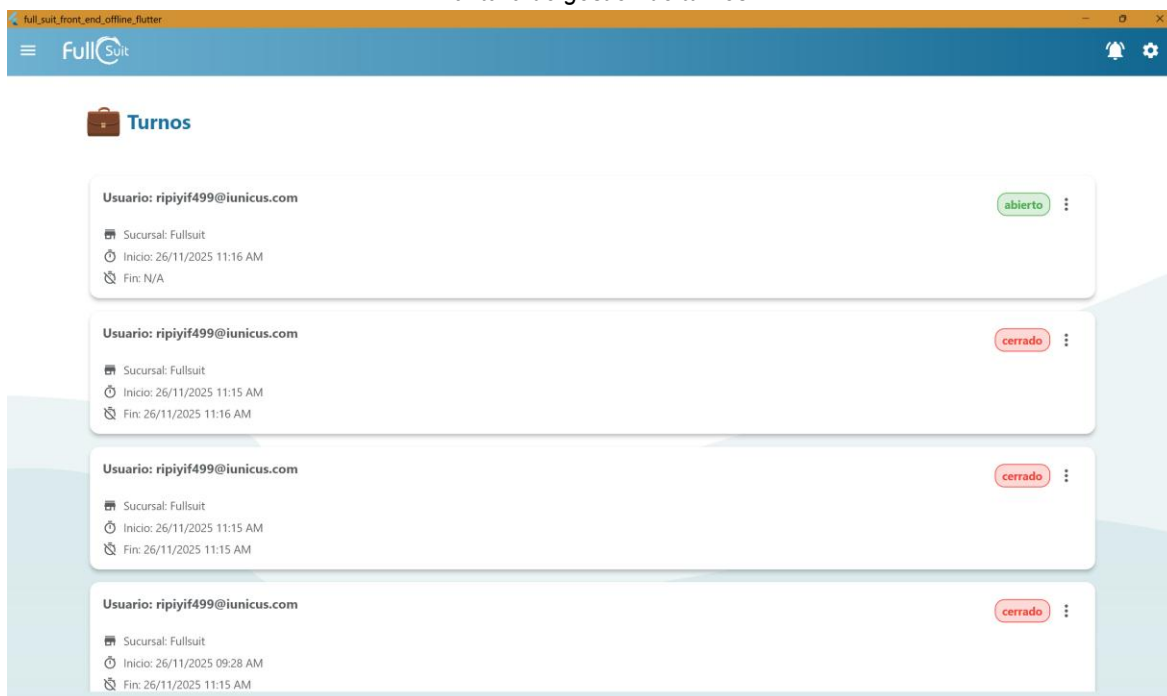
Cuando la aplicación recupera conexión, el proceso de sincronización consolida la información y enlaza las ventas con el turno correspondiente en el servidor, de modo que la operación offline y la registrada en la nube coincidan. El cierre de turno se encadenó con la sincronización, de manera que al cerrar caja se fuerza la subida de la información pendiente antes de generar el documento de cuadre.

Figura 16
Diagrama de flujo del módulo de turnos



Fuente: Autor

Figura 17
Pantalla de gestión de turnos



Fuente: Autor

El diagrama muestra los tres estados por los que pasa un turno y los puntos en que la información se escribe en la base local. La venta se asocia al turno en el momento de registrarla y no al cerrar el turno, lo que permite reconstruir el cuadro aunque el cierre se haga en otro dispositivo o después de un cierre inesperado de la aplicación.

Durante el desarrollo se atendió un caso que no estaba previsto. Cuando un turno se había abierto desde otro dispositivo o desde la plataforma web, la aplicación lo recuperaba del

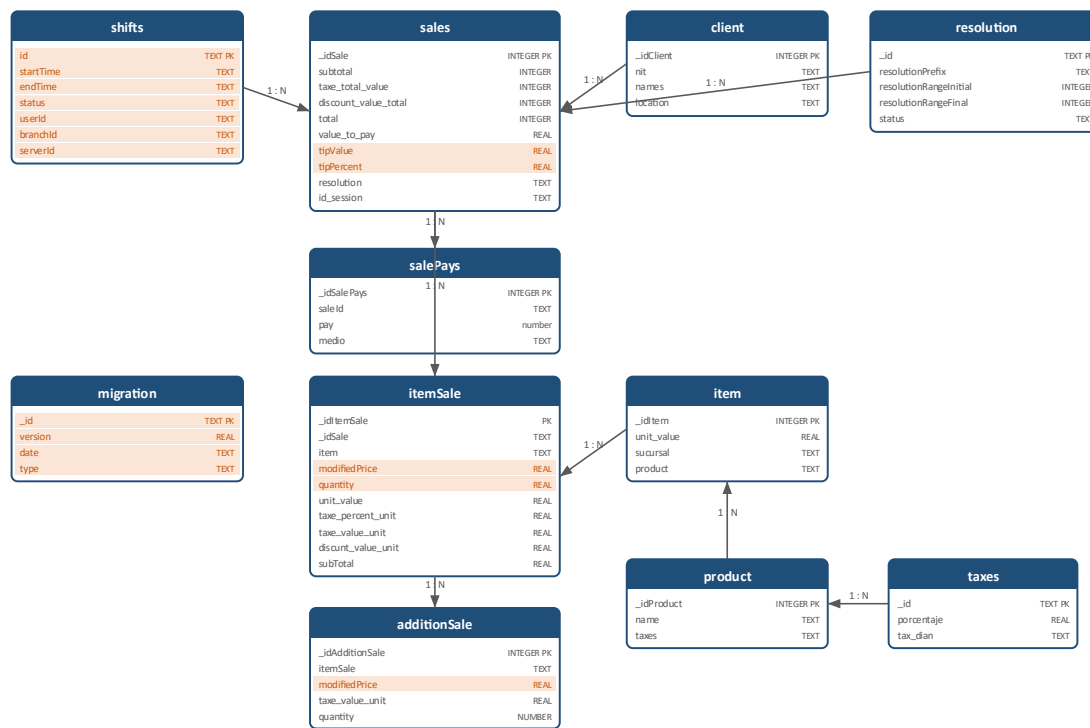
servidor pero no creaba la contraparte local, con lo cual quedaba sin registro en SQLite y sin posibilidad de generar reportes offline. Se añadió la construcción del registro local en ese escenario.

6.4.5. Persistencia en SQLite y evolución del esquema

El módulo de turnos requirió agregar al esquema de la base local una tabla nueva, con el identificador local del turno, la hora de inicio, la hora de cierre, el estado, el usuario, la sucursal y el identificador asignado por el servidor, que es el que permite reconciliar el turno local con el remoto al sincronizar.

El esquema completo de la base local tiene 40 tablas y 313 columnas. A continuación se presentan las 11 que sostienen la operación de venta y el control de turnos, que son las que se intervinieron durante la pasantía, con el tipo de dato tal como lo declara el motor.

Figura 18
Modelo entidad-relación de la base de datos local



Columna creada o modificada durante la pasantía

Tipos según PRAGMA table_info sobre FullSuitDB.db. Extracto de 11 de las 40 tablas: se omiten catálogo geográfico, inventario y configuración.

Fuente: Autor

En el diagrama se aprecia la estructura de una venta. La cabecera almacena los totales y se relaciona con el turno en que se registró, con el cliente y con la resolución de facturación

aplicada. De ella dependen las líneas de detalle, cada una con su propia descomposición de valores, y de cada línea dependen a su vez las adiciones en el caso del restaurante. La tabla de artículos conserva el precio por sucursal y apunta al producto de la organización, que es quien lleva la configuración tributaria.

Las columnas resaltadas son las que se crearon o se modificaron durante la pasantía. Cada una corresponde a un resultado descrito en este informe. La tabla shifts aparece completa: no existía antes y es el soporte del módulo de turnos. La columna migration.version es la que hace posible el versionado del esquema. En itemSale, la columna modifiedPrice almacena el precio final que fija el cajero y es el punto de partida del despeje inverso del tercer objetivo, mientras que quantity pasó de entero a real para admitir la venta de productos a granel. Las columnas tipValue y tipPercent de sales sostienen el manejo de propinas.

Cuando hubo que modificar el esquema por primera vez, al implementar los turnos, se hizo evidente que no había forma de alterar la estructura de la base en equipos que ya estaban instalados. SQLite tiene un mecanismo para esto, el número de versión de la base de datos junto con las sentencias de alteración de tabla (SQLite, s. f.), pero el proyecto no lo usaba. En esa actualización los clientes tuvieron que perder su información local. Por suerte no fue crítico, porque todo se había migrado a la nube y no se perdió nada. Para que no volviera a pasar se implementó el versionado del esquema, con una rutina que detecta la versión instalada y aplica los cambios necesarios sin borrar los datos. La primera migración agregó dos columnas a la tabla de ventas.

A partir de ese momento se pudieron agregar campos nuevos al modelo de datos, como los que después exigió el manejo de propinas, sin que ningún comercio perdiera el histórico de ventas guardado en su equipo. Ambler y Sadalage (2006) llaman a esta práctica diseño evolutivo de bases de datos: tratar el esquema como algo versionado que avanza con transformaciones sucesivas, en lugar de rehacerlo en cada cambio.

También se volvió atómico el registro de la venta. Más de una vez hubo incidentes en los que los clientes reportaban ventas duplicadas o corruptas, y la mayoría aparecían al sincronizar: la sincronización marcaba un error y, al revisarlo, era un registro de venta incompleto. Registrar una venta implica insertar la cabecera, su resolución de facturación, sus medios de pago, sus artículos y las adiciones de cada uno, y esas inserciones se hacían por separado, así que una interrupción a mitad del proceso dejaba la venta a medias. Se agruparon en una transacción, de modo que la operación se completa entera o no deja rastro. Es la propiedad de atomicidad,

que exige que el conjunto de operaciones se comporte como una unidad indivisible frente a cualquier fallo (Haerder & Reuter, 1983).

6.4.6. Generación de reportes de venta sin conexión

El objetivo contemplaba la generación de reportes, componente que se desarrolló en dos etapas.

La primera consistió en habilitar la consulta del histórico de turnos desde la propia aplicación, con filtros y con la posibilidad de recuperar el documento de cierre de cualquier turno anterior. La pantalla presenta el listado leído de la base local, de modo que la consulta funciona sin conexión.

La segunda incorporó el recibo de cierre de caja. Se trata de un documento pensado para impresión en papel térmico que reúne el detalle de los artículos vendidos durante el turno, el total de ventas, el total de descuentos, la separación entre documento POS y factura electrónica, la discriminación de impuestos por tarifa y el desglose de medios de pago. Su elaboración exigió resolver el ajuste de las columnas al ancho del papel, que varía según la impresora del establecimiento, mediante un reparto proporcional que garantiza que la suma de los anchos coincida con el ancho disponible.

Figura 19
Recibo de cierre de turno

FULL COLOMBIA SAS

Reporte de ventas

Nit: 901402077

Fecha de inicio:
13:54:21 13/08/2026

Fecha de fin:
15:09:15 13/08/2026

Punto de venta:

Full Suit prueba

Impreso por:

ripiyif499@iunicus.com

Colaborador:
prueba

Total ventas: 2

UD PRODUCT TOTAL

S O

3 ARROZ \$15.470,00
IVA 19

1 Chocorram \$1.000,00
o editado

1 Alpinito \$2.160,00

Total descuento: \$,00

Total: \$19.325,00

Discriminación por tipo de
venta

Tipo	Total
Electronica	\$,00
Pos	\$19.325,00

Discriminación de
impuestos

Tipo	BASE
IVA	\$2.470,00
INC	\$160,00

Discriminación por formas
de pago

EFFECTIVO \$19.325,00

Fuente: Autor

Se manejaron además los casos en que el reporte no puede generarse. Cuando el turno no registra ventas, o cuando las ventas todavía no se han sincronizado, la aplicación informa la causa concreta en lugar de mostrar un error genérico, e indica al cajero que sincronice antes de reintentar.

6.4.7. Resumen de actividades del objetivo 1

Tabla 7
Actividades realizadas para el objetivo específico 1

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto entregado
Análisis del código y detección de fallos	Mes 1, Sem 1	Mes 1, Sem 1	Bitácora técnica
Creación de mockups para escritorio	Mes 1, Sem 1	Mes 1, Sem 2	Mockups en Figma
Refactorización de la interfaz en Flutter	Mes 1, Sem 2	Mes 2, Sem 3	Interfaz de escritorio actualizada
Implementación de validaciones y carga continua	Mes 1, Sem 3	Mes 2, Sem 2	Formularios y listados funcionales
Desarrollo del módulo de turnos sobre SQLite	Mes 2, Sem 2	Mes 2, Sem 4	Módulo de turnos operativo
Consulta del histórico de turnos y reporte de cierre	Mes 4, Sem 1	Mes 4, Sem 3	Recibo de cierre de caja
Versionado del esquema de la base de datos local	Mes 5, Sem 3	Mes 5, Sem 4	Migración de esquema sin pérdida de datos
Integridad transaccional en el registro de ventas	Mes 7, Sem 2	Mes 7, Sem 3	Escritura atómica de la venta

Fuente: Autor

Las cinco primeras actividades corresponden al alcance reportado en el avance presentado durante la pasantía. Las tres últimas se desarrollaron con posterioridad, a medida que la operación en producción reveló necesidades que el alcance inicial no había contemplado.

6.4.8. Limitaciones identificadas

La refactorización no alcanzó la totalidad del código. Persisten pantallas que conservan patrones de la versión anterior, y la convivencia de ambos estilos es visible en módulos de uso menos frecuente.

Las sentencias SQL del proyecto arman las consultas concatenando cadenas en lugar de usar parámetros. No se recuerda un caso en que esto haya fallado en la práctica, pero un nombre de producto con un apóstrofo podría interrumpir una inserción. No se corrigió porque el cambio tocaba todo el proyecto.

6.5. Desarrollo del objetivo específico 2

"Desarrollar la versión móvil de la aplicación en Android con Flutter, aplicando diseño responsivo y reestructuración de módulos clave, para brindar una experiencia de usuario satisfactoria en dispositivos de menor tamaño."

La aplicación Full Suit Offline había sido concebida y compilada únicamente para escritorio durante sus primeros tres años de desarrollo. Llevar esa base de código a Android exigió dos tipos de trabajo: adaptar las pantallas que sí aguantaban una reducción de ancho y construir desde cero dos módulos nuevos para las que no.

6.5.1. Configuración del entorno de compilación para Android

La habilitación de la plataforma Android resultó menos costosa de lo previsto. Flutter es un entorno multiplataforma por diseño, de modo que bastó con activar el objetivo de compilación correspondiente y realizar algunos ajustes menores de configuración para obtener un archivo APK funcional en menos de una semana. El resultado de esa primera compilación, sin embargo, dejó claro que el trabajo real estaba en otra parte.

La aplicación arrancaba y respondía, pero ninguna de sus pantallas era usable. Los diseños habían sido calculados para monitores de escritorio, con anchos fijos y disposiciones de varias columnas que en una pantalla de teléfono producían desbordamientos de contenido en prácticamente todas las vistas, de manera que los textos quedaban cortados a media palabra, las tablas se salían del área visible por el costado derecho y los botones aparecían comprimidos hasta un tamaño que ningún dedo podía acertar. Nada de eso servía. La conclusión de esa prueba temprana fue que compilar para Android y funcionar en Android son dos cosas distintas.

Se estableció como versión mínima Android 5.0 (API 21). El proyecto se mantuvo sobre Flutter 3.7.9 y Dart 2.19.6, las versiones fijadas en el repositorio, lo que limitó las herramientas disponibles durante la adaptación.

6.5.2. Criterio de adaptación: dos estrategias según la pantalla

La decisión de qué pantallas se adaptaban y cuáles se rehacían fue simple. Las de listados y formularios se adaptaron, y solo se rehicieron las de los POS, por la cantidad de información y funciones que tenían en una sola vista.

Los módulos de listado y formulario (artículos, clientes, ventas, órdenes de inventario, devoluciones, turnos, sincronización y acceso) se resolvieron con diseño responsivo dentro del

mismo archivo, adaptando la disposición al espacio disponible en lugar de construir vistas independientes por tamaño de pantalla (Google, s. f.). Es el principio que Marcotte (2011) formuló para la web: una sola definición de la interfaz que responde a las dimensiones del dispositivo. Estas pantallas muestran la información en listas y campos y aguantan bien la reducción de ancho, así que bastó con ajustar márgenes, tamaños de letra y número de columnas.

Para eso se usó una condición booleana calculada a partir de la plataforma en ejecución, que cambia los valores de disposición sin duplicar la vista. En el lobby, por ejemplo, la cuadrícula de accesos tiene como máximo tres columnas en móvil y el espaciado entre elementos pasa de 24 a 16 píxeles. En el acceso al sistema, el formulario ocupa el 90 % del ancho en móvil, y en escritorio conserva un ancho fijo de 400 píxeles. Todo esto se hizo al mismo tiempo que el rediseño de esas pantallas.

Los módulos de punto de venta fueron el caso contrario. Tanto el POS comercial como el POS restaurante concentran en una sola vista la búsqueda de productos, la cuadrícula del catálogo, el resumen de la venta en curso, la edición de cantidades y descuentos, y el cierre de la transacción. Comprimir esa cantidad de información en la pantalla de un teléfono resultaba inviable sin destruir la experiencia de uso: todo quedaba ilegible. Por esa razón se construyeron dos módulos independientes, pos-mobile y pos-restaurant-mobile, que reorganizan el mismo flujo de trabajo en una secuencia de pantallas encadenadas en lugar de una vista única.

La interfaz que motivó esa decisión es la del POS comercial de escritorio (Figuras 12 y 13). En ella se ve la concentración de elementos de la vista de escritorio: los tres campos de búsqueda arriba, la cuadrícula de productos a la izquierda y el panel de resumen de venta a la derecha, con el detalle de cada línea, sus cantidades, descuentos y valores. Reproducir esa disposición en un dispositivo más pequeño habría obligado a reducir los elementos hasta hacerlos inoperables con el dedo. Nielsen y Budiu (2013) documentaron que trasladar sin más una interfaz de escritorio al móvil degrada la usabilidad de forma medible, y recomiendan reducir el contenido por pantalla y encadenar pasos antes que comprimir una vista densa.

6.5.3. Construcción del módulo POS móvil

El módulo pos-mobile divide la venta en tres pantallas. El catálogo de productos, con búsqueda y filtros por categoría, lleva al detalle del artículo, que es donde el cajero ajusta la cantidad, el precio unitario y el descuento. Desde ahí se pasa al estado de la cuenta y al cierre de la venta.

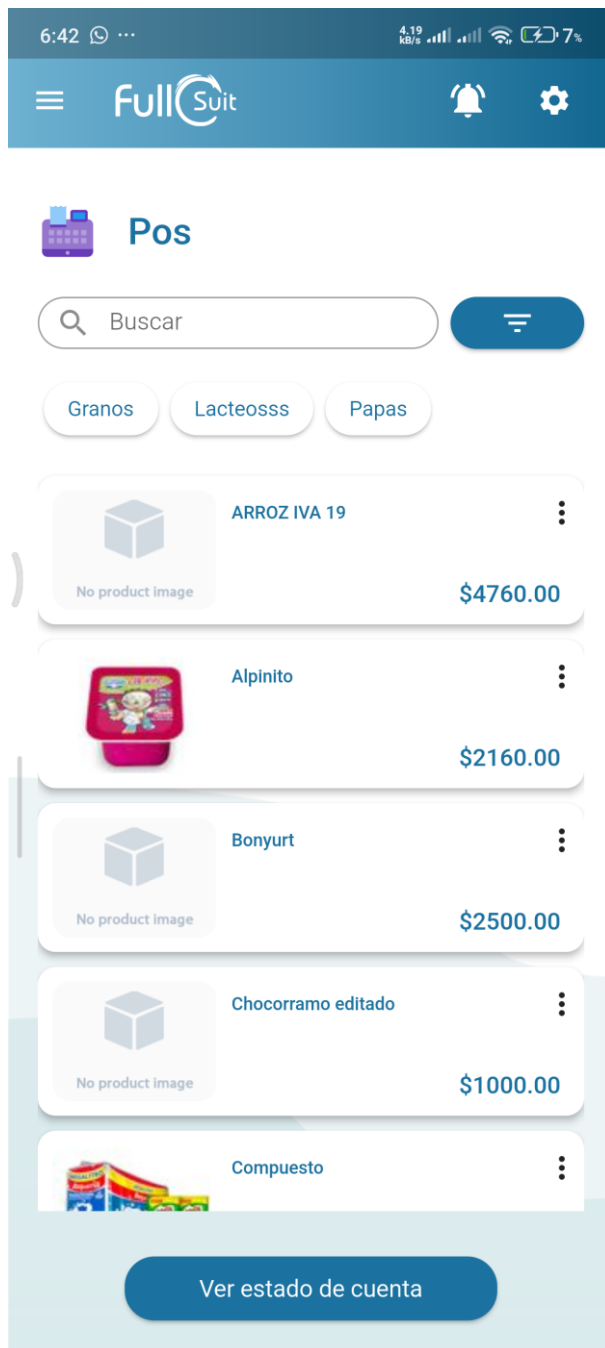
Los errores de paginación de este módulo se descubrieron durante el desarrollo, probando lo que se iba haciendo, y varios de ellos ponían muy lento el sistema. Uno venía del listado de productos: la primera implementación envolvía toda la columna en un contenedor desplazable, así que la lista no podía manejar su propia paginación y construía todos los elementos de una vez. Se corrigió dejando que el listado ocupara el espacio restante y manejara su propio desplazamiento, y con eso la carga por páginas volvió a funcionar.

Probando el módulo también se notó que con solo entrar a la pantalla de detalle el artículo quedaba agregado a la cuenta, así que si el cajero volvía atrás sin confirmar, el producto seguía en la venta sin que se diera cuenta. Se agregó un control de confirmación que descarta el artículo si se sale de la pantalla sin aceptar. También se reinicia el estado del módulo al entrar, liberando los controladores de texto y limpiando las listas de artículos, adiciones y descuentos, porque los datos de una venta anterior se pasaban a la siguiente.

El componente más extenso del módulo es el diálogo de cierre de venta, con 951 líneas de código. Reúne el resumen de totales, la selección de medios de pago, el cálculo del saldo y el cambio, y la validación de que la suma de los pagos no supere el valor de la venta.

La Figura 20 muestra el catálogo.

Figura 20
POS móvil: catálogo de productos



Fuente: Autor

El catálogo ocupa el alto completo del dispositivo y muestra los artículos en lista vertical, con el buscador y los filtros por categoría arriba y el acceso al estado de la cuenta abajo.

Figura 21

POS móvil: detalle de producto y edición de valores



Fuente: Autor

El detalle concentra los tres valores que el cajero puede intervenir (precio unitario, descuento y unidades) con el botón de confirmación al pie. Entre las dos figuras se aprecia cómo la información que en escritorio ocupaba una sola vista quedó distribuida en pantallas sucesivas, cada una con un propósito acotado y con controles dimensionados para la interacción táctil.

6.5.4. Construcción del módulo POS restaurante móvil

El POS restaurante suma a la venta la gestión de sectores, mesas, cuentas abiertas y adiciones por producto, y fue el trabajo más extenso del objetivo. Ninguna parte fue más difícil que las demás, pero en conjunto fue un requerimiento desafiante. Se construyeron 11 archivos nuevos, con cerca de 2.900 líneas, que cubren la selección de sector y mesa, el listado de artículos, el detalle de cada uno, la apertura y el estado de la cuenta, el diálogo de adiciones y la pantalla de cobro. En el historial de Git este trabajo quedó dentro de un lote registrado como corrección de errores, aunque más de la mitad de esos cambios son funcionalidad nueva. Después se agregaron la precuenta y la comanda de cocina, ambas con documento PDF para impresora térmica, igual que en la versión de escritorio.

6.5.5. Corrección del acceso a la red en la compilación de distribución

Al probar por primera vez la versión compilada para distribuir, la que se genera con `--release`, la aplicación no lograba conectarse. Durante todo el desarrollo se había comunicado sin problemas con el backend, pero en el APK de distribución arrancaba, la interfaz respondía y ninguna operación que necesitara conexión se completaba. Encontrar la causa tomó unas horas.

La causa estaba en la declaración de permisos de Android. Flutter incorpora el permiso de acceso a internet en el manifiesto que reserva para el desarrollo, porque lo necesita para sus propias herramientas de recarga en caliente y depuración, de modo que durante meses la aplicación dispuso de ese permiso sin que nadie lo hubiera declarado de forma explícita en el proyecto, y esa comodidad silenciosa fue justamente lo que mantuvo el defecto oculto hasta el momento de generar la primera versión destinada a un cliente. Ese manifiesto no participa en la compilación de distribución. El principal no declaraba el permiso. La corrección consistió en agregar la declaración de acceso a internet y de consulta del estado de la red al manifiesto principal del proyecto.

Desarrollar para escritorio y distribuir para Android no son lo mismo: el sistema operativo móvil impone restricciones que no se aplican durante la depuración y que solo aparecen en la compilación final, cuando el defecto ya va camino al cliente.

6.5.6. Distribución y estado en producción

La aplicación no se publicó en la tienda de aplicaciones de Google. La empresa optó por entregar el archivo APK directamente a cada cliente, que lo instala en su dispositivo. La razón

fue la estabilidad: el producto no alcanzó el grado que la empresa consideraba necesario para una publicación abierta, porque persistían defectos de diseño heredados de etapas anteriores del desarrollo.

La versión móvil se encuentra actualmente en operación en los comercios que la adoptaron y constituye la modalidad de uso habitual para varios de ellos. La última versión liberada durante la pasantía, identificada como 2.0.37.1, alcanzó un grado de estabilidad superior al de las anteriores.

6.5.7. Resumen de actividades del objetivo 2

Tabla 8
Actividades realizadas para el objetivo específico 2

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto entregado
Configuración del entorno de compilación para Android y primera prueba de APK	Mes 1, Sem 1	Mes 1, Sem 2	Proyecto compilando para Android
Construcción del módulo POS móvil	Mes 1, Sem 3	Mes 2, Sem 2	Módulo pos-mobile operativo
Diálogo de cierre de venta y estado de cuenta en móvil	Mes 2, Sem 1	Mes 2, Sem 3	Flujo de cobro móvil funcional
Construcción del módulo POS restaurante móvil	Mes 3, Sem 1	Mes 3, Sem 4	Módulo pos-restaurant-mobile operativo
Corrección del acceso a la red en compilación de distribución	Mes 2, Sem 4	Mes 2, Sem 4	APK con conectividad verificada
Precuenta y comanda para dispositivos móviles	Mes 4, Sem 2	Mes 5, Sem 1	Documentos PDF desde móvil

Fuente: Autor

El objetivo produjo 24 archivos nuevos y alrededor de 7.100 líneas de código en los dos módulos, además de las adaptaciones responsivas de las pantallas compartidas con la versión de escritorio.

6.6. Desarrollo del objetivo específico 3

"Implementar un sistema de precios dinámicos en los módulos POS, capaz de recalcular en tiempo real impuestos y descuentos mediante fórmulas inversas, con el fin de garantizar exactitud en los valores finales y facilitar la correcta emisión de la factura electrónica a la DIAN."

Este fue el objetivo que mayor dificultad técnica presentó, y su desarrollo se extendió por varios meses. Quienes lo llevaron a cabo eran dos practicantes sin experiencia ni conocimiento de los valores contables que se usan al emitir una factura electrónica, así que parte del tiempo se fue en explorar el requerimiento y aprender esos conceptos. Además, la plataforma web también permite hacer ventas, y lo que calculaba la versión web tenía que cuadrar al centavo con lo que calculaba la de escritorio, hecho de la misma forma. Por eso hubo que trabajar de la mano con el desarrollador de la aplicación web, que también era practicante. La parte lógica la lideró el autor, y fueron noches largas explicándole cómo llevar el requerimiento a Angular. Ayudó que allí la lógica de negocio estaba estructurada de forma parecida. A eso se sumaban los muchos casos de uso: las facturas grandes, con descuentos e impuestos distintos, solían destapar fallos.

6.6.1. El contexto que condiciona el problema

Full Suit no es una sola aplicación. La empresa mantiene dos clientes independientes sobre el mismo backend: una plataforma web desarrollada en Angular y la aplicación de escritorio y móvil objeto de esta pasantía, construida en Flutter. Son bases de código distintas, en lenguajes distintos, con interfaces deliberadamente parecidas, que consumen los mismos servicios y alimentan el mismo proceso de facturación.

La emisión de la factura electrónica se hace en tres pasos. La aplicación envía la venta a un servicio intermedio de la propia empresa, que actúa de puente con un proveedor tecnológico autorizado para emitir facturas electrónicas ante la DIAN. Por ese recorrido, un error de un centavo al calcular el impuesto de un artículo en el computador de una tienda no aparecía ahí ni en el servicio puente. Aparecía después, como un rechazo de la DIAN que no decía en qué valor de qué venta se había originado el descuadre.

El sistema separa producto de artículo. El producto pertenece a la organización y representa el bien en el catálogo, con su nombre, su categoría y su configuración tributaria. El artículo pertenece a la sucursal y es el que lleva el precio y las existencias. Un mismo producto, por ejemplo un arroz de cierta marca, existe como un artículo distinto en cada sede, y cada sede puede venderlo a un precio diferente.

Esa separación explica por qué el problema exigía un cálculo inverso. El impuesto se define en el ámbito de la organización, mientras que el precio se define en el ámbito de la sucursal. El único valor que el cajero controla en el punto de venta es el precio final que va a cobrar, y de

ese único valor dependen dos magnitudes que provienen de ámbitos distintos. Componer hacia adelante desde el catálogo dejaba de servir.

6.6.2. La situación previa

Antes de la intervención, todos los valores de una línea de venta se calculaban hacia adelante a partir del precio de catálogo del artículo. El subtotal se obtenía multiplicando el valor unitario por la cantidad y restando el descuento, y el impuesto se leía de un campo previamente almacenado. El cajero no podía intervenir el resultado.

En algunos comercios los precios cambian constantemente y se negocian con el cliente, como en una tienda de verduras, donde el precio depende de varios factores. El cajero necesitaba escribir directamente el valor que iba a cobrar, con impuesto y descuento ya incluidos, y que el sistema dedujera hacia atrás cada componente para que la factura electrónica los reportara bien.

6.6.3. El despeje inverso

Para eso se agregó al modelo de datos un campo nuevo que guarda el precio final unitario que fija el cajero. El despeje para hallar la fórmula inversa lo hizo principalmente el autor. A partir de ese valor se obtiene la base gravable dividiéndolo entre el producto de dos factores, uno con la tarifa del impuesto y otro con el descuento aplicado:

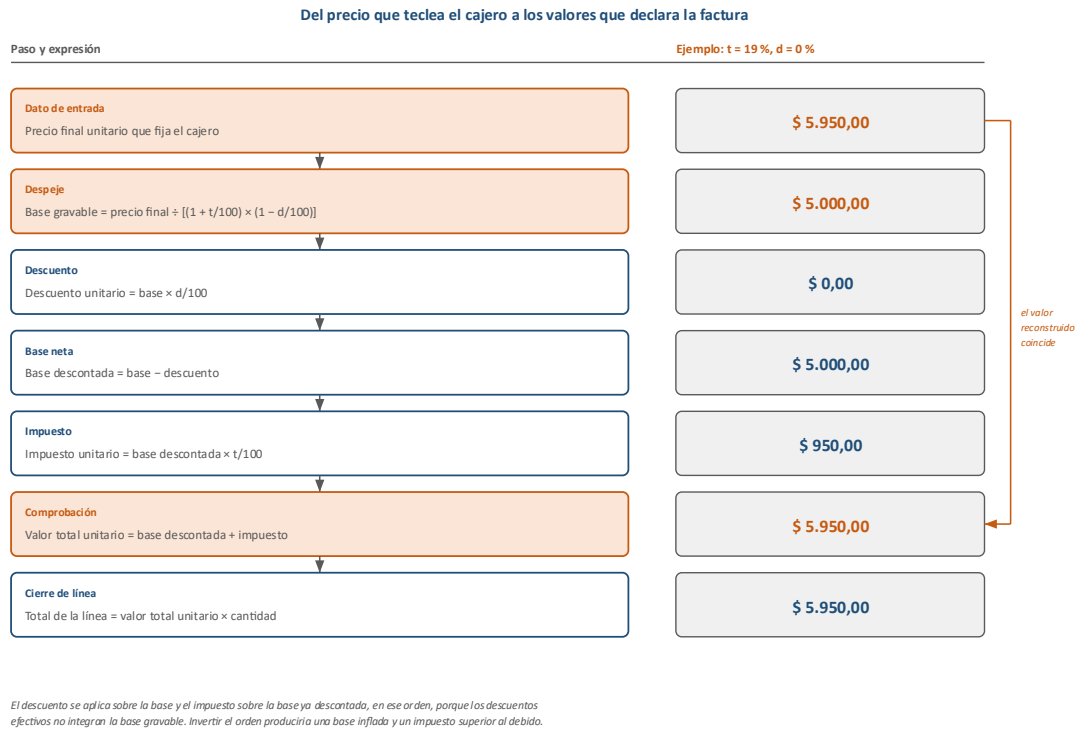
$$Base\ gravable = \frac{Precio\ final}{\left(1 + \frac{t}{100}\right)\left(1 - \frac{d}{100}\right)}$$

Donde t es la tarifa del impuesto en porcentaje y d el porcentaje de descuento. Es la inversa de la composición directa, en la que el precio final resulta de multiplicar la base por el factor de descuento y después por el de impuesto. El despeje quedó en una sola función de la capa de utilidades, aparte de la interfaz.

La función también controla los casos en que el denominador da cero, que son un impuesto de -100 % o un descuento del 100 %, para que la división no quede indeterminada.

A partir de esa base se calculan los demás componentes, en un orden determinado. El esquema siguiente muestra a la izquierda la expresión de cada paso y a la derecha el valor que da en un caso concreto: un artículo gravado al 19 %, sin descuento, cuyo precio final se fija a mano en \$5.950, que es la misma operación que aparece en las capturas de este apartado.

Figura 22
Descomposición del precio final en sus componentes fiscales



Fuente: Autor

El recorrido empieza por el único dato que controla el cajero, el precio final, y termina en el total de la línea. El despeje divide ese precio entre el producto de los dos factores y devuelve la base gravable. Sobre la base se calcula el descuento, y al restarlo queda la base descontada, que es sobre la que se aplica la tarifa del impuesto. Al sumar la base descontada y el impuesto se obtiene de nuevo el valor que tecleó el cajero, que es la comprobación resaltada en el esquema, y al multiplicar por la cantidad sale el total de la línea.

6.6.4. El orden de aplicación entre descuento e impuesto

El descuento se calcula sobre la base gravable, y el impuesto sobre la base ya descontada. Ese orden se aprendió en la investigación de los conceptos contables que hubo que hacer al comienzo del requerimiento.

El artículo 454 del Estatuto Tributario establece que los descuentos efectivos que consten en la factura no integran la base gravable, siempre que no estén sujetos a condición alguna y resulten normales según la costumbre comercial (Congreso de la República de Colombia, 1989). Aplicar el impuesto antes del descuento produciría una base inflada y un impuesto

mayor al que corresponde, y el documento quedaría inconsistente al transmitirlo para validación (DIAN, 2023).

En el código se calcula el valor del descuento aplicando su porcentaje sobre la base gravable, y con él un precio intermedio, la base menos ese descuento, que se guarda en una variable con nombre propio para que se vea en qué momento del cálculo se está. Solo sobre ese precio intermedio se aplica la tarifa del impuesto, y el nombre de la variable permite comprobar de un vistazo que el impuesto no se calculó antes de tiempo, que era el error que la DIAN terminaba devolviendo.

6.6.5. Verificación de la consistencia del despeje

Al cerrar la venta se guardan seis valores derivados por cada línea, y la relación entre ellos permite verificar algebraicamente que el despeje es correcto. Basta recomponer el precio unitario con sus componentes guardados: al sumar la base ya descontada y el impuesto calculado sobre ella, los dos términos comparten el factor de descuento, así que la suma se reduce a la base multiplicada por el factor de descuento y por el de impuesto, que es la composición directa de la que se partió.

El valor reconstruido coincide con el precio que escribió el cajero, y por eso la suma de subtotal, descuento e impuestos que viaja al servicio de facturación reproduce el valor cobrado, sin diferencias de centavos que la DIAN pudiera rechazar.

6.6.6. Recálculo en tiempo real

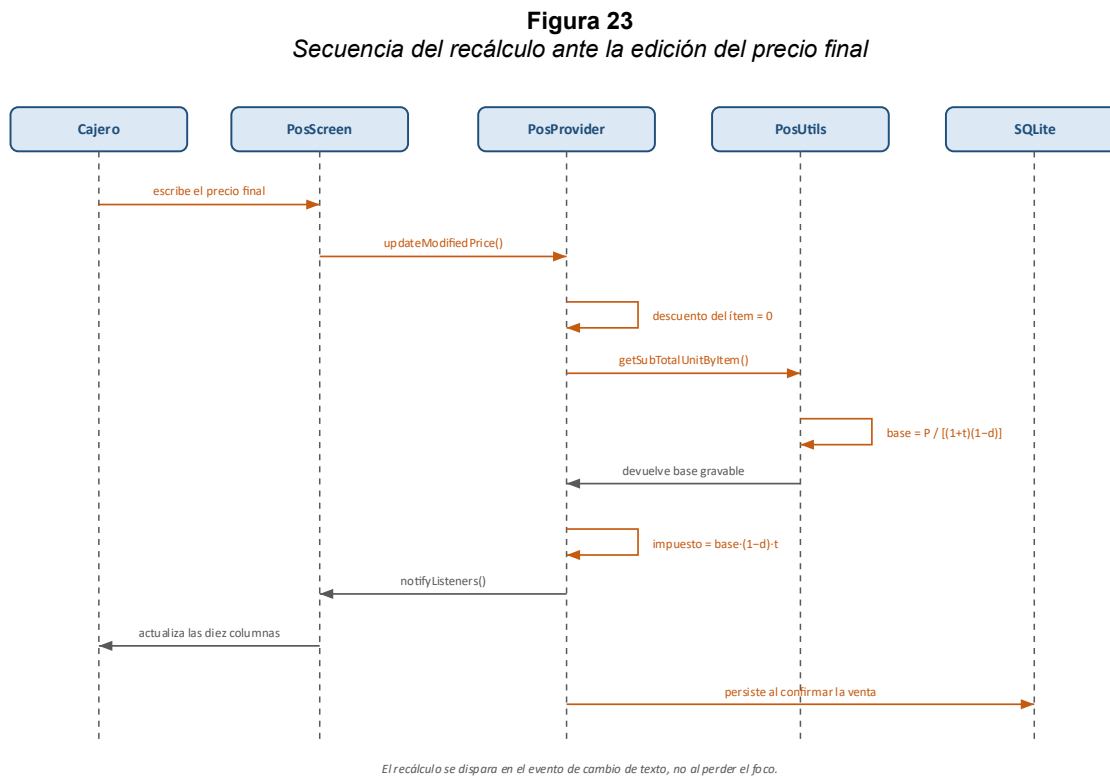
El recálculo debía ocurrir mientras el cajero interactúa con la pantalla y no al momento de cobrar, porque un valor que solo se actualiza al final no deja verificar lo que se está haciendo.

Se identificaron cuatro puntos de entrada por los que el usuario puede alterar el resultado de una línea, y se conectó cada uno con la rutina de recálculo. El primero es la edición directa del precio. Los otros tres corresponden a las formas de modificar la cantidad: los botones de incremento y decremento, el campo de texto de cantidad, y el reingreso de un artículo que ya estaba en la venta. Durante el desarrollo se detectó que la rutina existía pero no se invocaba desde ninguno de ellos, e incluso en un caso la llamada estaba anulada con un comentario que describía la necesidad de ejecutarla.

Además se modificó el momento de disparo de la edición del precio. La versión inicial recalculaba al perder el foco el campo, lo que obligaba al cajero a salir del recuadro para ver el

resultado. Se trasladó la lógica al evento de cambio de texto, de manera que los valores se actualizan mientras se escribe.

El recorrido completo de una modificación de precio se presenta en el siguiente esquema.



Fuente: Autor

La secuencia muestra que el despeje ocurre en la capa de utilidades, aislada de la interfaz, y que el resultado regresa al proveedor de estado antes de notificar a la pantalla. La anulación del descuento sucede antes del cálculo, lo cual garantiza que un precio impuesto manualmente no arrastre una rebaja porcentual previa. La escritura en la base local se produce al confirmar la línea, no en cada pulsación.

6.6.7. Interfaz de verificación detallada

Un sistema que deduce cinco valores a partir de uno solo necesita ser auditable por quien lo opera. Con ese fin se incorporó a la pantalla de venta un control que amplía el resumen a pantalla completa y presenta la descomposición completa de cada línea.

Se anuncia a continuación la vista compacta, que es la que el cajero utiliza durante la operación normal.

Figura 24
Resumen de venta en su presentación compacta

FullSuit

POS

Q

Buscar producto por nombre

Código de barras

Código del producto

Código de barras

Balanza

No product image

ARROZ IVA 19

\$4,760.00



Alpinito

\$2,160.00

No product image

Bonyurt

\$2,500.00

No product image

Chocorrano editado

\$1,000.00



Compuesto

\$4,000.00

No product image

Leche

\$2,750.00

No product image

No product image



Resumen de venta

+

Agregar Cliente

Nombre	Cantidad	Descuento	Valor total unitario	Valor total item
ARROZ IVA 19	- 1 +	0.0%	4760.00	\$4,760.00
Alpinito	- 1 +	0%	2160.0	\$2,160.00
Leche	- 1 +	0%	2750.0	\$2,750.00

Subtotal(3) items

\$8,500.00

Descuento

\$0

Impuesto

\$1,170.00

Total

Pagar \$9,670.00

Fuente: Autor

Al accionar el control de ampliación situado en el extremo superior derecho de la cabecera, la pantalla oculta el catálogo y los filtros, y despliega una tabla con diez columnas por artículo: nombre, subtotal unitario, porcentaje de descuento, valor del descuento unitario, subtotal unitario menos descuento, porcentaje de impuesto, valor del impuesto unitario, valor total unitario, cantidad y total de la línea.

Figura 25

Vista ampliada con la descomposición de valores por artículo

full_suit_front_end_offline_flutter

FullSuit

Resumen de venta

+

Agregar Cliente

Detalles de la venta

Nombre	Subtotal Unit.	% Desc	Valor Desc Unit.	Subtotal Unit. - Desc.	% Imp	Valor Imp Unit.	Valor Total Unit.	Cantidad producto	Total Item
ARROZ IVA 19	\$4,000.00	0%	\$0.00	\$4,000.00	19%	\$760.00	4760.0	- 1 +	\$4,760.00
Alpinito	\$2,000.00	0%	\$0.00	\$2,000.00	8%	\$160.00	2160.0	- 1 +	\$2,160.00
Bonyurt	\$2,500.00	0%	\$0.00	\$2,500.00	0%	\$0.00	2500.0	- 1 +	\$2,500.00

Subtotal(3) items

\$8,500.00

Descuento

\$0

Impuesto

\$920.00

Total

Pagar \$9,420.00

Fuente: Autor

Cada columna de esa tabla corresponde a un paso del despeje descrito anteriormente. El campo de precio final permanece editable dentro de la vista ampliada, de modo que al modificarlo se observa cómo se recalculan simultáneamente todas las demás columnas. La versión anterior de esta pantalla presentaba cinco columnas y ningún campo editable.

Para dejar constancia del comportamiento se registró una venta de prueba sobre la organización de ensayo y se forzó el precio final de un artículo gravado al 19 %. La figura siguiente muestra la pantalla justo después de esa edición.

Figura 26
Recálculo automático al forzar el precio final de una línea

full_suit_front_end_offline_flutter

FullSuit

Resumen de venta

+

Agregar Cliente

Detalles de la venta

Nombre	Subtotal Unit.	% Desc	Valor Desc Unit.	Subtotal Unit. - Desc.	% Imp	Valor Imp Unit.	Valor Total Unit.	Cantidad producto	Total Item
ARROZ IVA 19	\$5,000.00	0%	\$0.00	\$5,000.00	19%	\$950.00	5950.00	1	\$5,950.00
Alpinito	\$2,000.00	0%	\$0.00	\$2,000.00	8%	\$160.00	2160.0	1	\$2,160.00
Bonyurt	\$2,500.00	0%	\$0.00	\$2,500.00	0%	\$0.00	2500.0	1	\$2,500.00

Subtotal(3) items

\$9,500.00

Descuento

\$0

Impuesto

\$1,110.00

Total

Pagar \$10,610.00

Fuente: Autor

En la figura, el artículo costaba \$4.760, con una base de \$4.000 y un impuesto de \$760. Al escribir 5.950 en la columna de valor total unitario, la base pasó a \$5.000 y el impuesto a \$950 sin tocar esas dos casillas. Si se divide 5.950 entre 1,19 da 5.000, y el 19 % de esa base son 950, así que la fórmula se cumple en la propia interfaz. Las otras dos líneas, con tarifas del 8 % y del 0 %, no cambiaron, lo que muestra que el recálculo solo afecta la línea modificada. El mismo mecanismo se implementó en el POS de restaurante, donde las adiciones de cada plato aparecen como filas dependientes con su propia descomposición y su propio precio editable.

6.6.8. Normalización del redondeo

Los cálculos con punto flotante trajeron un problema que al principio no se veía. Los valores monetarios dejaban residuos en decimales lejanos, y esos residuos cambiaban según el camino que siguiera el dato. Es una propiedad del formato binario de doble precisión del estándar de punto flotante, que no puede representar de forma exacta la mayoría de las fracciones decimales y arrastra un error de redondeo en cada operación (Goldberg, 1991; Institute of Electrical and Electronics Engineers, 2019).

El caso más visible apareció al validar los medios de pago y se detectó en las pruebas. La comprobación de que la suma pagada no superara el total comparaba dos cantidades que venían de cálculos distintos, y un residuo de una milbillonésima bastaba para que un pago

exacto se tomara como mayor al total. El cajero recibía un mensaje de error y no podía cerrar una venta que estaba bien.

Se unificó el redondeo a dos decimales en los tres puntos por los que pasa un valor monetario: al escribirlo en la base de datos local, al calcularlo en memoria y al armar el mensaje que se envía al backend, que es el que alimenta la factura electrónica. En lo que teclea el cajero se dejó un redondeo hacia arriba al centavo, porque es la regla más común, y así no se pierden fracciones en contra de la empresa.

6.6.9. Corrección de defectos de exactitud

Haciendo muchas pruebas de ventas y con varias pasadas de QA aparecieron tres errores de cálculo que afectaban el valor declarado en la factura, y los tres venían de antes de la pasantía. El impuesto de una línea ya venía multiplicado por el número de unidades y al armar el registro de la venta se volvía a multiplicar, así que toda venta de más de una unidad reportaba más impuesto del que correspondía. El porcentaje de descuento se guardaba en la columna del valor del descuento, y entonces un 10 % sobre \$50.000 se enviaba como un descuento de \$10. Y el subtotal se calculaba sin restar el descuento, con lo que la base gravable reportada quedaba por encima de la real.

Cada uno se corrigió cambiando una línea, pero encontrarlos costó semanas, porque ninguno producía un fallo visible: la venta se registraba, el cajero cobraba el valor correcto y la pantalla mostraba cifras coherentes. El único síntoma era el rechazo posterior de un documento que ya había salido del punto de venta.

6.6.10. Validación cruzada entre clientes

Como la plataforma web y la aplicación de escritorio calculan de forma independiente sobre el mismo modelo, ambos debían producir resultados idénticos. Un mismo artículo, con el mismo impuesto, la misma cantidad y el mismo descuento, tenía que arrojar el mismo valor en las dos.

Las fórmulas se plantearon en paralelo con el desarrollador de la plataforma web. Las pruebas consistían en armar casos con la mayor cantidad de variables posible (un artículo con impuesto, varias unidades y un descuento) y comparar el resultado de las dos aplicaciones hasta el último decimal. Durante varias semanas eso fue mostrando diferencias que ninguno habría encontrado revisando solo su propio código, porque cada implementación era coherente por dentro, y solo al compararlas se veía que una de las dos redondeaba en el momento equivocado. En ocasiones una aplicación daba un valor y la otra uno distinto.

Cuando los valores no eran correctos, la validación final la hacía la propia DIAN, y por eso la verificación se hizo emitiendo muchas facturas electrónicas reales. El servicio devolvía errores por regla, como el rechazo FAW01, que aparece cuando una línea de la factura va con valor en cero sin precio de referencia, o la notificación FAX05, cuando no se informa una base gravable para un tributo. Esos mensajes permitieron encontrar inconsistencias que las pruebas locales no mostraban. Antes de estas correcciones se emitieron facturas con valores incorrectos que la DIAN no recibía.

6.6.11. La factura como comprobación del cálculo

Los valores despejados terminan en el documento que se entrega al cliente y que alimenta la facturación electrónica, así que la comprobación del trabajo solo queda completa cuando esos valores salen bien discriminados en el comprobante impreso. Para eso se hizo una operación de extremo a extremo sobre la organización de pruebas, con datos ficticios, desde la habilitación de la numeración hasta la impresión.

El módulo queda bloqueado mientras la sucursal no tenga una numeración habilitada, y el formulario de registro pide el número de la resolución, el prefijo, la clave técnica, el rango de numeración autorizado, las fechas de vigencia, el ambiente y la sede. Son los datos que la DIAN exige declarar para habilitar la numeración y para que el documento electrónico pase la validación previa (DIAN, 2020).

Figura 27
Formulario de creación de la resolución de facturación

Crear Resolución

Número*
000000000002

Prefijo*
PRUEBA

Llave Técnica*
LLAVETECNICADEPRUEBANOVALIDA

Tipo de resolución*
POS

Rango Inicial*
1

Rango Final*
1000

Valida desde*
13-08-2026

Valida hasta*
31-12-2026

Ambiente*
TEST

Sucursal*
Full Suit prueba

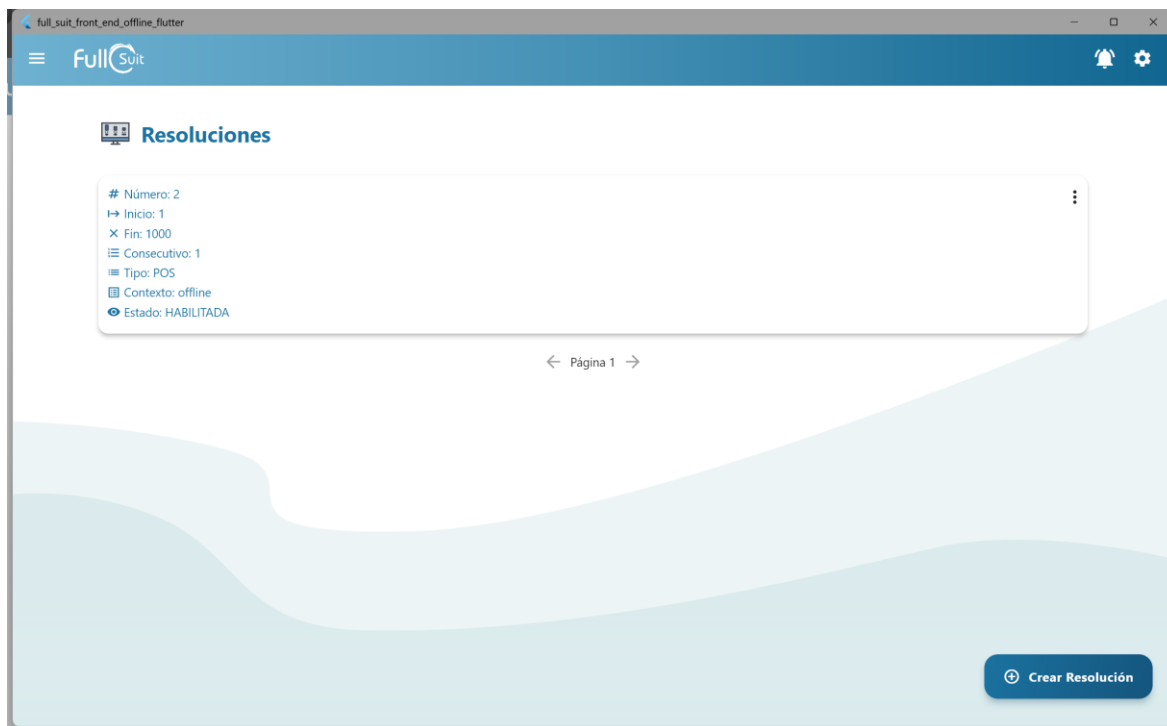
Crear Resolución

Fuente: Autor

Los valores de la figura son de prueba. Una vez registrada, la resolución queda deshabilitada hasta que se active, y la aplicación admite una sola resolución habilitada por sede y por tipo: al activar una, se desactiva la que estuviera vigente, para que dos numeraciones no consuman consecutivos al mismo tiempo y no haya saltos ni duplicados en la serie autorizada.

Figura 28

Listado de resoluciones con la numeración habilitada



Fuente: Autor

Con la numeración activa se registraron dos ventas. En la pantalla de cobro están el tipo de venta, los medios de pago, la propina, el cambio y el saldo pendiente. Se pueden combinar varios medios de pago en una misma operación, y la propina, el cambio y el saldo se recalculan mientras se digita el valor recibido.

Figura 29

Diálogo de cobro con medios de pago, propina y cambio

Crear venta

Tipo de venta: POS

Descripción:

Efectivo

Valor pagado con Efectivo: 10000

Cliente: usuario no registrado - 22222222222

Retención ☐ Crédito ☐

Propina

0% 5% 10% 15%

Porcentaje (%): 10

Valor: 695

Equivale al 10% de la venta

Cambio:	\$2,355.00
Saldo:	\$0.00
Total:	\$6,950.00
Total con propina:	\$7,645.00

Fuente: Autor

Sobre un total de \$6.950 se aplicó una propina del 10 % y se entregaron \$10.000 en efectivo, lo que dio un total con propina de \$7.645 y un cambio de \$2.355. La propina se suma al valor a pagar pero no entra en la base gravable, porque el artículo 512-9 del Estatuto Tributario establece que, por ser voluntaria, no hace parte de la base del impuesto al consumo (Congreso de la República de Colombia, 1989).

La Figura 30 muestra la factura resultante.

Figura 30
Factura POS con la discriminación de impuestos por tarifa

TU TIENDA LIBERTAD

NIT 222222222222-7

Sucursal: Full Suit prueba

cil 3 #4-45, Chiscas

Fecha: 8/13/2026

Hora: 14:41:06

INFORMACIÓN DEL CLIENTE

usuario no registrado

Doc: 222222222222-7

CÓDIGO	DESCRIPCIÓN		
CANT	V.UNIT	%DS	TOTAL
d984a743	ARROZ IVA 19		
1.0 un	\$5.950,00	%0.0	\$5.950,00
choc01	Chocorramo editado		
1.0 un	\$1.000,00	%0.0	\$1.000,00

PROPINA

Porcentaje: 10.0% \$695,00

Descuento en venta: \$,00

TOTAL: \$6.950,00

Total + propina: \$7.645,00

Medio de pago: EFECTIVO

Tipo de venta: Recibo

PAGOS

Efectivo \$7.645,00

Vendedor: prueba

DISCRIMINACIÓN DE IMPUESTOS

Tipo	Base	Impuesto
TARIFA	BASE	IMPUESTO
c=19%	\$5.000,00	\$950,00
EXE=0%	\$1.000,00	\$,00
Total	\$6.000,00	\$950,00

Fuente: Autor

El comprobante muestra el precio que fijó el cajero, \$5.950 para el artículo editado, y al pie discrimina las bases por tarifa: \$5.000 gravados al 19 % con \$950 de impuesto, y \$1.000 exentos, para un total de \$6.000 de base y \$950 de impuesto. La suma de la base y el impuesto da el total de la venta, y la propina aparece en un bloque aparte. Es la descomposición que la factura electrónica debe declarar, obtenida a partir de un precio que el cajero escribió a mano.

Tabla 9

Actividades realizadas para el objetivo específico 3

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto entregado
-----------------------------	--------------	---------------	--------------------

Análisis del modelo de valores del artículo y del flujo de facturación	Mes 3, Sem 1	Mes 3, Sem 2	Documento de análisis
Planteamiento de las fórmulas inversas en conjunto con la plataforma web	Mes 3, Sem 2	Mes 3, Sem 4	Fórmulas acordadas entre clientes
Implementación del despeje inverso y del campo de precio editable	Mes 3, Sem 3	Mes 4, Sem 3	Recálculo funcional en POS comercial
Extensión al POS restaurante y a las adiciones	Mes 4, Sem 2	Mes 5, Sem 1	Recálculo funcional en POS restaurante
Vista ampliada de descomposición de valores	Mes 4, Sem 4	Mes 5, Sem 2	Interfaz de verificación detallada
Normalización del redondeo en cálculo, persistencia y envío	Mes 5, Sem 1	Mes 5, Sem 3	Valores consistentes en toda la cadena
Pruebas cruzadas con la plataforma web y validación contra la DIAN	Mes 5, Sem 2	Mes 6, Sem 3	Facturas aceptadas por la DIAN
Corrección de los tres defectos de exactitud detectados	Mes 5, Sem 4	Mes 6, Sem 2	Valores fiscales correctos

Fuente: Autor

6.6.12. Limitaciones identificadas

La expresión del despeje quedó duplicada en la capa de presentación, en las columnas que muestran los valores calculados, en lugar de invocar en todos los casos la función común. Esa repetición contraviene el principio de mantener una única fuente para cada regla de negocio (Martin, 2008), y las copias además no heredaron los resguardos contra denominador nulo. Existen además dos variantes de la misma función de subtotal, una con validación de tipos y resguardos y otra sin ellos, y es la segunda la que interviene al persistir la venta.

Los módulos de precuenta del restaurante emplean una descomposición distinta de la del resto del sistema. La versión de escritorio omite el factor de descuento en el despeje, y la versión móvil calcula el descuento sobre el precio con impuesto en lugar de sobre la base. Ambas afectan únicamente a un documento informativo que se entrega al comensal antes de facturar, ya que la factura formal siempre atraviesa el cálculo verificado, pero es una inconsistencia real.

Tampoco se desarrollaron pruebas automatizadas que verifiquen las fórmulas. La validación se hizo de forma manual y mediante la comparación con la plataforma web, lo cual detecta discrepancias entre clientes, pero no protege frente a una modificación futura del cálculo.

6.7. Desarrollo del objetivo específico 4

"Integrar las nuevas funcionalidades de la aplicación al flujo de trabajo SCRUM, sincronizando datos entre la aplicación offline, SQLite, la plataforma web y los servicios backend, para asegurar un ciclo de desarrollo paralelo, robusto y eficiente."

Este objetivo tiene dos partes. Una es metodológica y tiene que ver con la forma de trabajo del equipo. La otra es técnica y comprende el mecanismo que mantiene coherente la información entre la base de datos local, la plataforma web y los servicios del backend.

La proporción entre ambos no es equilibrada, y responde a la del propio enunciado. El componente técnico ocupa la mayor parte del capítulo porque es donde se concentró el trabajo: reconstruir el proceso de sincronización y resolver los conflictos que aparecían cuando la misma organización operaba a la vez desde el punto de venta y desde la plataforma web. El componente metodológico se resuelve antes, y es más breve porque consistió en integrarse a un marco que ya existía.

6.7.1. Por qué SCRUM y no otro marco de trabajo

La empresa ya trabajaba con SCRUM desde antes de la pasantía, con un equipo pequeño que sostenía varios productos en producción, así que lo que tocaba era integrarse al equipo y a su ritmo desde el primer día. No fue muy duro, y como el otro desarrollador también era nuevo, la adaptación fue natural.

SCRUM define un conjunto acotado de eventos, roles y artefactos orientados a la entrega incremental de valor (Schwaber & Sutherland, 2020), y se inscribe en la corriente que privilegia el software funcionando y la respuesta al cambio sobre la documentación exhaustiva y el apego al plan (Beck et al., 2001). Para una pasantía individual, marcos como programación extrema o Kanban habrían ofrecido menos sobrecarga de ceremonias. Pero proponer un cambio de metodología a un equipo que ya trabajaba de otra manera, desde la posición de quien acaba de llegar y sobre un producto con clientes activos, no habría sido razonable. Kniberg (2007) describe con detalle esa clase de aplicación parcial del marco en equipos reales, donde las prácticas se adoptan de manera desigual según lo que cada organización puede sostener.

6.7.2. Cómo funcionaba el proceso en la práctica

El equipo tenía una reunión diaria virtual a las 9 de la mañana, de lunes a viernes. Normalmente duraba entre 20 y 30 minutos, y cuando se alargaba mucho no pasaba de una hora. Cada desarrollador reportaba lo que había hecho el día anterior, y si había terminado alguna tarea compartía pantalla y se la presentaba al director de desarrollo. Ahí se resolvían las dudas y se planificaba lo que se iba a hacer en el día.

La validación funcional quedaba a cargo de la persona de QA, que era personal administrativo de la empresa sin perfil técnico en aseguramiento de calidad. Cuando los requerimientos pasaban por QA se aprobaban y se sacaba una versión.

Las tareas se seguían en tableros de GitHub Projects. Al terminar una tarea se pasaba a Done para que la persona de QA probara ese requerimiento o bug. Al comienzo el tablero se mantenía al día, con la indicación explícita de actualizar el estado de cada tarea, pero con el tiempo el registro se volvió más irregular y en los últimos meses la documentación de sprints del aplicativo móvil dejó de mantenerse.

6.7.3. Los artefactos que quedaron registrados

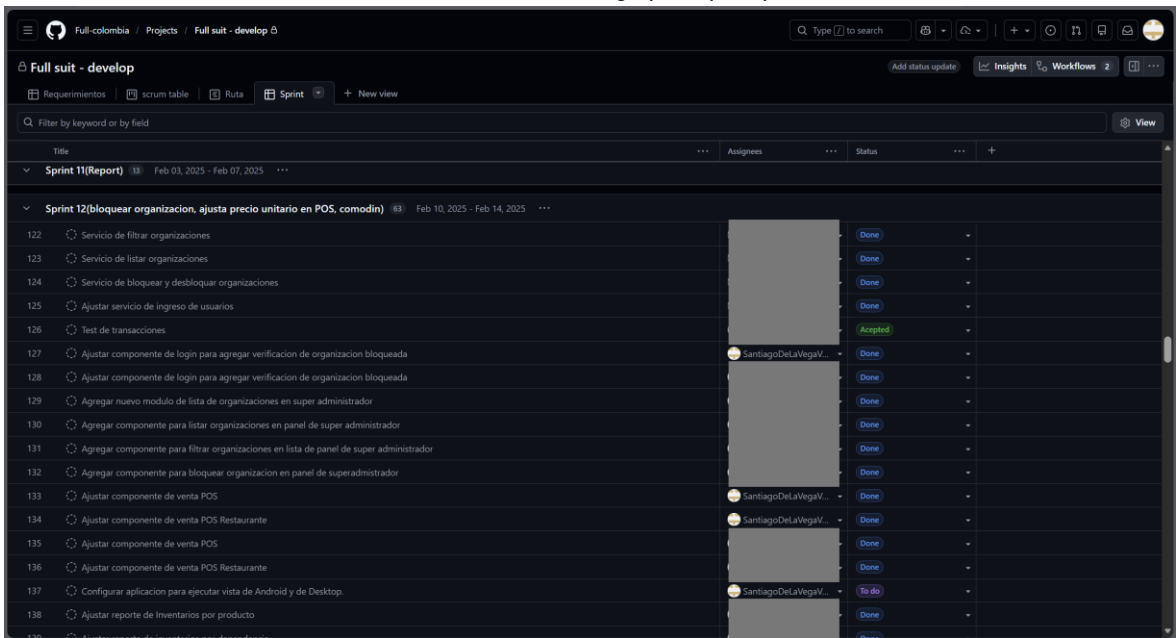
El tablero de desarrollo del producto registra 19 sprints completados, cada uno con título, fecha de inicio y duración. La configuración define iteraciones de 14 días que empiezan los lunes, aunque la duración real varió entre 3 y 17 días. La planificación de los sprints la hacían los responsables del equipo.

Varios de esos sprints se corresponden directamente con los objetivos de la pasantía. Uno de ellos lleva en su propio título la mención al ajuste del precio unitario en el punto de venta, que es el tercer objetivo específico, e incluye entre sus tareas la configuración de la aplicación para ejecutarse en Android además de escritorio, que es el segundo. Otro reúne los cambios en el punto de venta de restaurante, la impresión de comandas y la incorporación de clientes extranjeros.

Las tareas están registradas con los campos habituales de la gestión ágil: estado dentro del flujo, prioridad, estimación de tamaño, sprint asignado, fechas de inicio y fin, y responsable. Un buen número de ellas está redactado como historia de usuario, en la forma «como vendedor se solicita que...» o «como administrador se solicita que...». La fórmula nombra al rol y la necesidad. Es la que Cohn (2004) propone para mantener la conversación centrada en el valor para el usuario y no en la solución técnica.

La figura siguiente presenta la vista del tablero agrupada por iteración.

Figura 31
Tablero de desarrollo agrupado por sprint



Fuente: Autor

Se distingue la cabecera del sprint que reúne el bloqueo de organizaciones, el ajuste del precio unitario en el punto de venta y una tarea comodín, con su ventana de ejecución. Bajo ella aparecen, asignadas al autor, las tareas de ajuste del componente de venta del punto de venta comercial y del de restaurante, y la configuración de la aplicación para ejecutarse en Android además de escritorio, es decir, el tercer objetivo y el segundo, planificados en el mismo ciclo y registrados como tareas con responsable, estado y sprint asignado. Los nombres de los demás miembros del equipo se ocultaron por el deber de reserva que impone el convenio.

El detalle de ese mismo sprint muestra hasta qué punto el trabajo estaba descompuesto.

Figura 32
Componentes del punto de venta comercial asignados en el sprint

Title	Assignees	Status
168 Componente de listar compras de un cliente comprador	SantiagoDeLaVegaV...	Done
169 Componente de descargar recibo de compra de cliente comprador	SantiagoDeLaVegaV...	Done
170 Componente de descargar PDF de factura electronica de compra de usuario comprador	SantiagoDeLaVegaV...	Done
171 Componente de descargar ZIP de factura electronica compra de usuario comprador	SantiagoDeLaVegaV...	Done
172 Componente de listar notas de compra de un cliente comprador	SantiagoDeLaVegaV...	Done
173 Componente de descargar PDF de DIAN de notas de cliente comprador	SantiagoDeLaVegaV...	Done
174 Componente de descuentos en POS comercial	SantiagoDeLaVegaV...	Done
175 Componente de pagos de usuario en POS comercial	SantiagoDeLaVegaV...	Done
176 Componente de retenciones en POS comercial	SantiagoDeLaVegaV...	Done
177 Componente de valores negativos POS comercial	SantiagoDeLaVegaV...	Done
178 Componente decimales POS comercial	SantiagoDeLaVegaV...	Done
179 Componente lectura de codigo de barras POS comercial	SantiagoDeLaVegaV...	Done
180 Componente de filtro POS comercial	SantiagoDeLaVegaV...	Done
181 Servicios de editar roles de usuario	SantiagoDeLaVegaV...	Done
182 Ajustar componente de creacion de cliente	SantiagoDeLaVegaV...	Done
183 Ajustar componente de creacion de cliente	SantiagoDeLaVegaV...	Done
184 Error impuestos en reporte de venta #2	SantiagoDeLaVegaV...	To do

Fuente: Autor

Siete tareas consecutivas, todas a nombre del autor y todas cerradas, recorren una por una las piezas del cálculo que más adelante ocupó el tercer objetivo: descuentos, pagos del usuario, retenciones, valores negativos, decimales, lectura de código de barras y filtro. La descomposición por componente, y no por pantalla, explica por qué los defectos de exactitud pudieron aislarse después con precisión.

Un segundo tablero se dedica exclusivamente a la gestión de defectos. Registra la fecha de descubrimiento, la fecha de asignación, la fecha estimada de solución y la fecha de entrega, además del activo digital afectado.

Figura 33
Tablero de gestión de defectos con el ciclo de vida documentado

Title	Assignees	Status	Digital Asset	Date of discovery	Assignment date	Estimated date
1 listado de productos se queda en blanco (app) #15	SantiagoDeLaVegaV...	Accepted	App Movil	Jul 17, 2025	Jul 21, 2025	
2 validación de los datos para sincronizar antes de cerrar turno #32	SantiagoDeLaVegaV...	Accepted	App Movil	Jul 22, 2025	Jul 24, 2025	
3 Pago de una factura con medio de pago superior al total de la factura #36	SantiagoDeLaVegaV...	Accepted	App Movil	Jul 25, 2025	Jul 25, 2025	
4 tipo de venta al seleccionar el medio de pago #37	SantiagoDeLaVegaV...	Accepted	App Movil	Jul 25, 2025	Jul 25, 2025	Jul 29, 2025
5 historico de ventas #38	SantiagoDeLaVegaV...	Accepted	App Movil	Jul 30, 2025	Jul 25, 2025	Jul 29, 2025
6 Cambio de valor del descuento en pos restaurante #40	SantiagoDeLaVegaV...	Accepted	App Movil	Jul 25, 2025	Jul 25, 2025	Jul 30, 2025
7 Precarga de datos #46	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 5, 2025		
8 APK alertas #52	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
9 APK articulos boton de filtros #53	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
10 APK clientes #54	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
11 APK pos comercial #55	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
12 APK pos comercial, productos ocultos #56	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
13 APK pos comercial, categorias #57	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
14 APK pos comercial, boton de filtro #58	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
15 APK pos comercial, descuentos #59	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
16 APK pos comercial, facturación #60	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
17 APK pos comercial, seleccion de cliente en facturación #61	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		
18 APK pos comercial, insumos #62	SantiagoDeLaVegaV...	Accepted	App Movil	Aug 28, 2025		

Fuente: Autor

Filtrado a los elementos asignados al autor, el tablero arroja 93 defectos, casi todos marcados con el activo App Móvil, y todos cerrados. Las columnas de fecha permiten seguir cada uno desde que se descubrió hasta su entrega. Los títulos son descriptivos, por ejemplo «listado de productos se queda en blanco», «pago de una factura con medio de pago superior al total» o «cambio de valor del descuento en pos restaurante», y varios llevan el número de sprint, lo que enlaza este tablero con el de desarrollo.

6.7.4. Estrategia de ramas

El control de versiones siguió un esquema derivado de GitFlow, que organiza el trabajo en ramas de funcionalidad, de liberación y de corrección urgente sobre una rama de desarrollo y una principal (Driessen, 2010; Chacon & Straub, 2014). Cada funcionalidad se hacía en su propia rama feature, que después se integraba a la rama de desarrollo y de ahí pasaba a la principal al liberar. Las correcciones urgentes iban en ramas hotfix, etiquetadas al fusionarse.

El repositorio conserva las ramas de las funcionalidades desarrolladas durante la pasantía, entre ellas las del punto de venta móvil, la del lector de código de barras y la del trabajo sobre descuentos.

La aplicación del esquema fue laxa. Una parte considerable de los cambios se resolvió mediante ramas de corrección urgente, incluso cuando el contenido correspondía a funcionalidad nueva y no a un defecto en producción. Esa desviación se explica por las condiciones de trabajo descritas en las conclusiones.

6.7.5. Ciclo de liberación

La cadencia de entregas quedó registrada en el propio código, ya que la versión se declara como literal en las pantallas de arranque y sincronización. Durante el periodo documentado se publicaron nueve versiones.

Tabla 10
Versiones liberadas durante el periodo

Versión	Contenido principal
2.0.15	Precuenta en POS restaurante y exportación de la base de datos
2.1.0	Cierre del sprint de propinas, comandas y clientes extranjeros
2.0.18	Integración de báscula para productos a granel
2.0.19	Refuerzo del manejo de propinas y precuenta parcial

2.0.21	Exportación de registros de diagnóstico
2.0.22	Paso a entorno de producción
2.0.23	Gestión de sesiones y rediseño de la sincronización
2.0.36	Correcciones de sesión y persistencia
2.0.37.1	Versión final del periodo

Fuente: Autor

La numeración presenta discontinuidades, ya que algunas versiones intermedias se liberaron desde otras ramas del producto. El versionado fue una de las prácticas menos ordenadas del equipo: no existía una convención escrita sobre cuándo incrementar cada cifra, y los saltos de la tabla son la huella de esa ausencia.

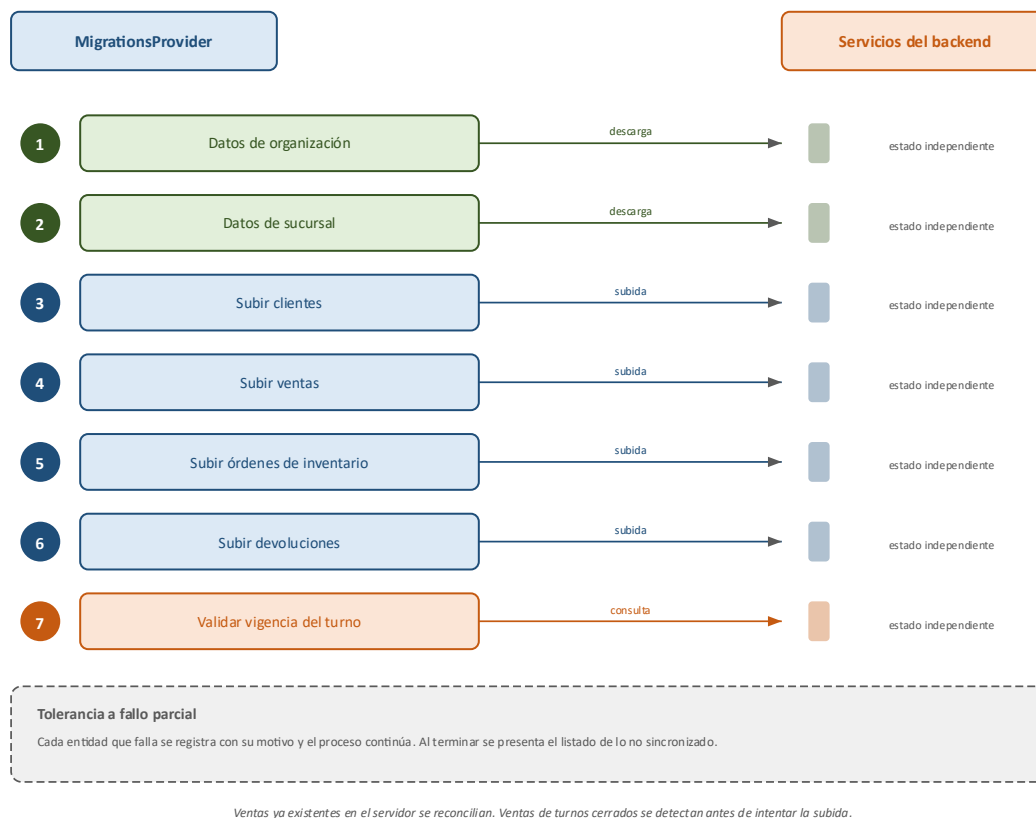
6.7.6. Sincronización entre la aplicación offline y los servicios del backend

Este es el componente técnico del objetivo y el de mayor volumen. La aplicación opera sobre una base de datos local y debe reconciliar su contenido con el servidor cada vez que recupera conexión, en un escenario donde la misma organización puede estar operando simultáneamente desde la plataforma web. Es un problema conocido. Los sistemas con réplicas que aceptan escrituras estando desconectadas pagan su disponibilidad con la necesidad de resolver después las divergencias entre copias, y esa resolución no puede delegarse en el motor de almacenamiento porque depende de reglas del negocio (Kleppmann, 2017). Una solución equivalente para sesiones alternadas entre operación con y sin red había sido descrita por Viscaino Quito y Serpa Andrade (2022).

El proceso original funcionaba como una caja negra. Mostraba un mensaje de texto mientras trabajaba, y ante cualquier fallo se detenía por completo sin indicar qué entidad lo había provocado, de modo que cuando un comercio reportaba que la sincronización no terminaba, la única información disponible era esa: que no terminaba, sin manera de saber si el problema estaba en una venta concreta, en un cliente con un documento mal formado, en una orden de inventario o en la propia conexión. Se rediseñó como una máquina de estados de siete pasos, cada uno con su propio progreso, su registro de eventos con marca temporal y su estado independiente de resultado.

Los siete pasos comprenden la descarga de datos de organización y de sucursal, la subida de clientes, de ventas, de órdenes de inventario y de devoluciones, y una validación final de vigencia del turno.

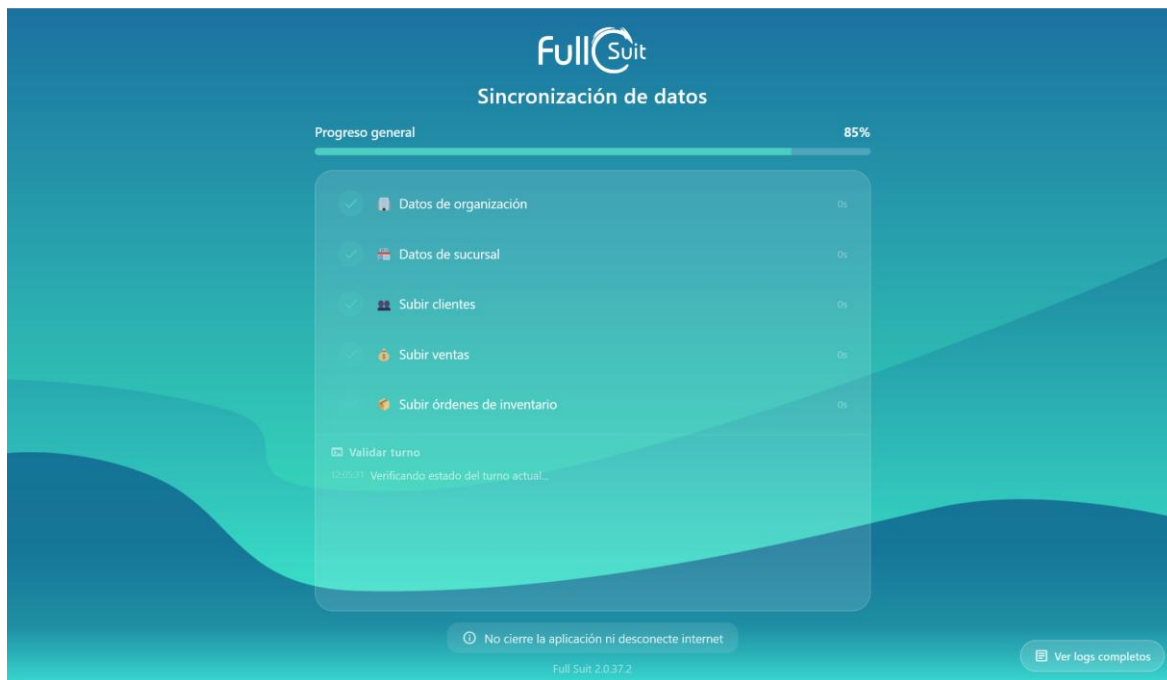
Figura 34
Secuencia del proceso de sincronización por pasos



Fuente: Autor

El esquema distingue los pasos de descarga de los de subida, y muestra que cada uno mantiene un estado propio de resultado. Esa independencia es la que permite el comportamiento descrito a continuación.

Figura 35
Pantalla de sincronización con el detalle por paso



Fuente: Autor

El cambio principal fue pasar de un modelo binario a uno tolerante al fallo parcial. Antes, una sola venta rechazada interrumpía toda la sincronización. Después, cada entidad que falla se registra de forma individual con su motivo, el proceso continúa con las siguientes, y al terminar se presenta el listado de lo que no pudo subirse.

Sobre esa base se resolvieron tres situaciones de la operación real. Dos de ellas tienen que ver con el turno de caja, que se desarrolló dentro del primer objetivo, porque el turno es donde aparecen los conflictos entre los dos clientes: es el contenedor al que pertenece cada venta y el que determina si esa venta puede subirse o no.

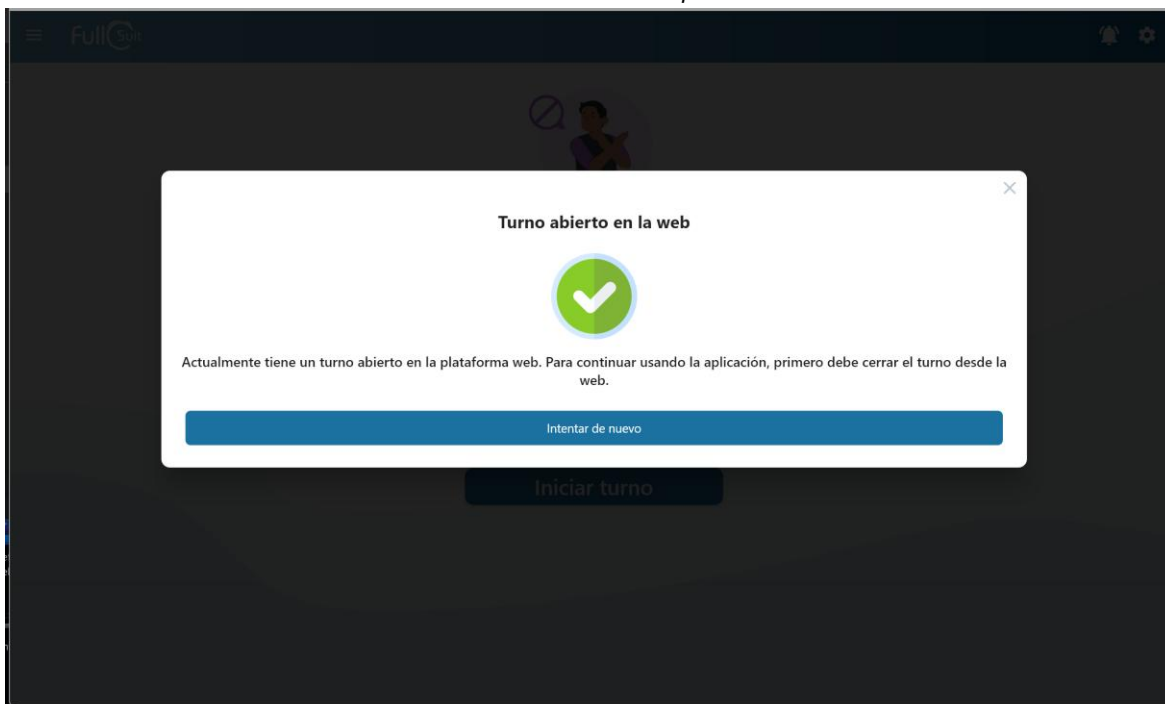
Una era la de las ventas duplicadas, que por lo general terminaban reportando los clientes y algunas veces la persona de QA. Cuando una venta ya existía en el servidor pero la aplicación no había recibido la confirmación, porque la conexión se cortó justo después de que el servidor la registrara y antes de que llegara la respuesta, la venta quedaba marcada como pendiente en la base local y se reintentaba en cada sincronización, indefinidamente, contra un servidor que la rechazaba por consecutivo repetido. Se agregó el reconocimiento de esa respuesta del backend, y ahora la venta se marca como subida en lugar de reintentarse.

También había ventas asociadas a turnos ya cerrados. Si un turno se cerraba desde la plataforma web mientras el punto de venta seguía operando, quedaban ventas locales huérfanas que el servidor rechazaba. El proceso ahora las identifica antes de intentar la subida

e informa la situación, con una opción de forzado para los casos en que la subida deba realizarse de todos modos.

Y estaba el conflicto entre sesiones. Un mismo usuario puede operar desde la aplicación de escritorio y desde la plataforma web, y un turno abierto en una de ellas invalida la apertura en la otra, porque las ventas quedarían repartidas entre dos turnos simultáneos y el cuadre de caja dejaría de corresponder con lo realmente vendido.

Figura 36
Detección de turno abierto en la plataforma web



Fuente: Autor

En la figura se ve el mensaje que muestra la aplicación de escritorio cuando detecta esa situación, con la indicación de cerrar primero el turno desde la plataforma web. Para resolverlo se construyó un gestor centralizado de sesión, AuthSessionManager, porque la aplicación no tenía un registro de las sesiones. El gestor evalúa en cascada la validez del token, la sucursal seleccionada, la disponibilidad de conexión y si hay un turno abierto en el servidor, y según el resultado decide a qué pantalla va la aplicación al arrancar, con nueve desenlaces posibles. Cuando no hay conexión pero existe un token válido se permite el acceso en modo local, porque impedirlo dejaría al comercio sin poder facturar.

6.7.7. Observabilidad como respuesta a la ausencia de pruebas

El proyecto no contaba con pruebas automatizadas ni con verificación previa al despliegue, y los dispositivos donde ocurrían los fallos estaban en los comercios de los clientes, fuera del alcance de depuración. Había muchísimos errores que no se sabía de dónde venían, ni siquiera si eran del backend o de la aplicación.

Por eso se construyeron los registros (logs). Lo primero fue capturar todas las excepciones del sistema, para que cualquier fallo no controlado quedara escrito en un archivo local en lugar de perderse al cerrar la aplicación. Después se registraron el cuerpo y la respuesta de las peticiones de venta, hasta llegar a registrar todas las peticiones y respuestas de la aplicación, reemplazando el token de autenticación antes de escribirlo para que el archivo se pudiera compartir sin exponer la sesión. Así se podía saber qué pasaba cuando algo fallaba.

El paso que cerró la línea fue hacer ese registro extraíble. Se añadió al menú una opción que entrega el archivo de registro al selector de archivos del sistema, con lo cual el cajero de un comercio puede enviarlo por sí mismo, sin cable, sin herramientas de desarrollo y sin conocimientos técnicos. Cinco semanas antes se había habilitado el mismo mecanismo para la base de datos local, de modo que un cálculo erróneo pudiera reproducirse con los datos reales del establecimiento.

Los registros terminaron siendo una de las herramientas más importantes del proyecto.

6.7.8. Formalización de procedimientos

Hacia el final del periodo se documentaron dos procedimientos que hasta entonces se ejecutaban de memoria.

Uno es el de liberación de versiones. Establece las fases del proceso, los archivos que deben modificarse, la matriz de configuración que distingue el entorno de pruebas del de producción en nombre de versión, base de datos, servicio de interfaz de programación y dirección de consulta de la factura, la regla de incremento del número de versión, y una confirmación obligatoria antes de aplicar cualquier cambio.

El otro fija la convención para organizar los cambios en el control de versiones, con las categorías ordenadas por capa, un límite recomendado de archivos por confirmación y la prohibición expresa de agregar todos los cambios de forma indiscriminada.

6.7.9. Integración continua: mejora identificada y no implementada

Durante el seguimiento académico se identificó la automatización del ensamblado del instalador mediante flujos de integración continua como una mejora deseable para el proyecto. La práctica consiste en llevar cada cambio a través de una cadena automatizada de compilación, verificación y empaquetado, de modo que el paso de código a entregable deje de depender de la intervención de una persona (Humble & Farley, 2010).

No se implementó. El ciclo de liberación dependía de una validación funcional manual previa que no era automatizable en las condiciones existentes, y la generación del instalador se apoyaba en una configuración anterior al periodo de la pasantía. Lo que sí se hizo fue formalizar el procedimiento de liberación en el documento descrito en el apartado anterior, que constituye la especificación manual de aquello que un flujo automatizado ejecutaría.

6.7.10. Resumen de actividades del objetivo 4

Tabla 11
Actividades realizadas para el objetivo específico 4

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto entregado
Incorporación al ciclo de reuniones diarias y al tablero de tareas	Mes 1, Sem 1	Mes 1, Sem 1	Participación en el flujo del equipo
Adopción del esquema de ramas por funcionalidad y corrección	Mes 1, Sem 2	Mes 2, Sem 1	Ramas integradas a desarrollo
Captura global de excepciones y registro de peticiones	Mes 2, Sem 3	Mes 2, Sem 4	Archivo de diagnóstico local
Encadenamiento de la sincronización con el cierre de turno	Mes 2, Sem 4	Mes 3, Sem 1	Cierre con subida forzada
Exportación de la base de datos y de los registros de diagnóstico	Mes 4, Sem 2	Mes 5, Sem 2	Diagnóstico remoto habilitado
Gestor centralizado de sesión y manejo de token expirado	Mes 6, Sem 2	Mes 7, Sem 1	Validación de sesión unificada
Rediseño de la sincronización como proceso de siete pasos	Mes 6, Sem 3	Mes 7, Sem 2	Sincronización observable
Reconciliación de duplicados y turnos cerrados	Mes 7, Sem 1	Mes 7, Sem 3	Fallo parcial tolerante

Formalización de los procedimientos de liberación y de versionado	Mes 7, Sem 4	Mes 8, Sem 1	Documentos de procedimiento
---	--------------	--------------	-----------------------------

Fuente: Autor

6.7.11. Limitaciones identificadas

El marco de trabajo se aplicó de forma adaptada. Se sostuvieron las reuniones diarias y la planificación por iteraciones, pero no se realizaron retrospectivas, ni refinamiento formal del listado de pendientes, ni seguimiento mediante métricas de avance. La documentación de sprints se interrumpió en las etapas finales.

El esquema de ramas se aplicó con desviaciones, resolviendo mediante ramas de corrección urgente trabajo que correspondía a funcionalidades nuevas.

No se implementaron flujos de integración continua ni pruebas automatizadas, así que el ensamblado y la verificación fueron manuales durante todo el periodo. La falta de pruebas sobre las fórmulas de cálculo es la limitación más importante, porque no permite detectar si una modificación posterior daña el cálculo (Beck, 2003).

7. CONCLUSIONES

Los cuatro objetivos específicos se cumplieron. La aplicación de escritorio se refactorizó y ahora tiene el módulo de turnos con su recibo de cierre, así que los turnos se consultan en la propia aplicación sin entrar a la plataforma web. La versión Android está en operación, y la usa, entre otros, el hotel de la región que la había pedido para su restaurante. El sistema de precios dinámicos quedó en los cuatro módulos de venta, y las facturas con precio editado a mano pasan la validación de la DIAN. Ese objetivo terminó siendo más grande de lo previsto, porque hubo que coordinar cada fórmula con el desarrollador de la plataforma web para que las dos aplicaciones dieran el mismo valor al centavo. El trabajo se integró al flujo SCRUM del equipo, y la sincronización ahora sigue adelante aunque falle un registro.

Lo más difícil fue el contexto de trabajo. Al llegar no había documentación, el producto estaba en producción con clientes facturando todos los días, y arreglar algo en un lado muchas veces rompía otra cosa. El equipo pasaba la mayor parte del tiempo corrigiendo errores y atendiendo requerimientos, y no había pruebas automatizadas ni una persona de QA con perfil técnico. Por eso se construyeron los registros, para poder saber qué había pasado cuando algo fallaba en un comercio.

Trabajar en un entorno un poco caótico enseñó a trabajar bajo presión y a entender que ningún sistema es perfecto y que siempre va a haber errores, incluso en producción. También se aprendió a trabajar en equipo: explicarle a un compañero cómo reproducir un error o cómo enviarle el cuerpo de la respuesta de una petición desde el inspector, cosas sencillas que no se practican en la universidad. En lo técnico quedaron el desarrollo multiplataforma con Flutter, la persistencia local con migraciones del esquema y el cálculo aplicado a la facturación electrónica. Fue una experiencia muy rica en conocimiento, aunque un poco estresante, y gracias a ella el autor trabaja hoy, antes de graduarse, como uno de los desarrolladores principales de un broker.

Antes de las correcciones de cálculo se emitían facturas con valores incorrectos que la DIAN rechazaba, porque el impuesto se multiplicaba dos veces por la cantidad y el descuento se guardaba como porcentaje en la columna del valor. Una vez corregidos esos errores, los documentos pasaron la validación. La versión Android le dio a la empresa un canal que no tenía, el versionado del esquema permite cambiar el modelo de datos sin que los clientes pierdan su histórico, y con la exportación de registros se pueden atender incidencias sin ir al comercio. Los beneficiados son los cajeros que usan el sistema y los comercios que dependen de él para facturar.

Quedan pendientes la integración continua para generar el instalador y las pruebas automatizadas que protejan las fórmulas de cálculo ante cambios futuros.

8. REFERENCIAS

Alur, D., Crupi, J., & Malks, D. (2003). Core J2EE patterns: Best practices and design strategies (2.^a ed.). Prentice Hall.

Ambler, S. W., & Sadalage, P. J. (2006). Refactoring databases: Evolutionary database design. Addison-Wesley.

Bass, L., Clements, P., & Kazman, R. (2012). Software architecture in practice (3.^a ed.). Addison-Wesley.

Beck, K. (2003). Test-driven development: By example. Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifiesto por el desarrollo ágil de software. <https://agilemanifesto.org/iso/es/manifesto.html>

Chacon, S., & Straub, B. (2014). Pro Git (2.^a ed.). Apress. <https://git-scm.com/book/es/v2>

Cohn, M. (2004). User stories applied: For agile software development. Addison-Wesley.

Congreso de la República de Colombia. (1989). Decreto 624 de 1989, por el cual se expide el Estatuto Tributario de los impuestos administrados por la Dirección General de Impuestos Nacionales. Artículo 454: Los descuentos efectivos no integran la base gravable. <https://estatuto.co/454>

Cunningham, W. (1992). The WyCash portfolio management system. En Addendum to the proceedings on object-oriented programming systems, languages, and applications (OOPSLA '92) (pp. 29–30). Association for Computing Machinery. <https://doi.org/10.1145/157709.157715>

Dart. (s. f.). Dart documentation. Google. <https://dart.dev/guides>

Dirección de Impuestos y Aduanas Nacionales. (2020). Resolución 000042 del 5 de mayo de 2020, por la cual se desarrollan los sistemas de facturación, los proveedores tecnológicos, el registro de la factura electrónica de venta como título valor, se expide el anexo técnico de factura electrónica de venta y se dictan otras disposiciones. <https://www.dian.gov.co/normatividad/Normatividad/Resoluci%C3%B3n%20000042%20de%2005-05-2020.pdf>

Dirección de Impuestos y Aduanas Nacionales. (2023). Resolución 000165 de 2023, por la cual se desarrolla el sistema de facturación electrónica y se expide el anexo técnico. <https://www.dian.gov.co/normatividad/Paginas/Resoluciones.aspx>

Dirección de Impuestos y Aduanas Nacionales. (s. f.). Normatividad del sistema de facturación electrónica. <https://micrositios.dian.gov.co/sistema-de-facturacion-electronica/normatividad/>

Driessen, V. (2010). A successful Git branching model. <https://nvie.com/posts/a-successful-git-branching-model/>

Flutter. (s. f.). Flutter documentation. Google. <https://docs.flutter.dev/>

Fowler, M. (2018). Refactoring: Improving the design of existing code (2.^a ed.). Addison-Wesley.

Full Colombia. (2021). Full Colombia. <https://fullcolombia.co/>

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys, 23(1), 5–48. <https://doi.org/10.1145/103162.103163>

Google. (s. f.). Material Design 3: Applying layout. <https://m3.material.io/foundations/layout/>

Haerder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4), 287–317. <https://doi.org/10.1145/289.291>

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley.

Institute of Electrical and Electronics Engineers. (2019). IEEE standard for floating-point arithmetic (IEEE Std 754-2019). <https://doi.org/10.1109/IEEESTD.2019.8766229>

Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media.

Kniberg, H. (2007). *Scrum and XP from the trenches*. InfoQ.

Kruchten, P. (1995). Architectural blueprints—The «4+1» view model of software architecture. *IEEE Software*, 12(6), 42–50. <https://doi.org/10.1109/52.469759>

Marcotte, E. (2011). *Responsive web design*. A Book Apart.

Martin, R. C. (2008). *Clean code: A handbook of agile software craftsmanship*. Prentice Hall.

McConnell, S. (2004). *Code complete: A practical handbook of software construction* (2.^a ed.). Microsoft Press.

Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.

Nielsen, J., & Budiu, R. (2013). *Mobile usability*. New Riders.

Pressman, R. S. (2010). *Software engineering: A practitioner's approach* (7.^a ed.). McGraw-Hill.

Schwaber, K., & Sutherland, J. (2020). *La Guía de Scrum: Las reglas del juego*. <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf>

Sommerville, I. (2011). *Software engineering* (9.^a ed.). Pearson.

SQLite. (s. f.). SQLite documentation. <https://www.sqlite.org/docs.html>

Viscaino Quito, J. A., & Serpa Andrade, L. J. (2022). Synchronization procedure for data collection in offline-online sessions. En *AHFE 2022 International Conference on Applied Human Factors and Ergonomics*. <https://doi.org/10.54941/ahfe1001461>

Windmill, E. (2020). *Flutter in action*. Manning Publications.

9. MANUAL DE USUARIO

Este manual describe la operación de la aplicación Full Suit Offline en sus dos presentaciones, la de escritorio para Windows y la de dispositivos Android. Está dirigido al personal que opera el sistema en el establecimiento, principalmente el cajero y el administrador del punto de venta, y no requiere conocimientos técnicos previos.

Las funcionalidades incorporadas durante la pasantía se señalan a lo largo del texto cuando corresponde, con el fin de que el lector pueda ubicarlas.

9.1. Requisitos e instalación

9.1.1. Versión de escritorio

La versión de escritorio se distribuye como un archivo instalador ejecutable para Windows. Al ejecutarlo, el asistente copia la aplicación y crea el acceso directo correspondiente. No requiere instalación de bases de datos ni de componentes adicionales, ya que el motor de almacenamiento local viaja dentro de la propia aplicación.

9.1.2. Versión Android

La versión para dispositivos móviles se distribuye como archivo de instalación de Android. El comercio lo recibe directamente de la empresa y lo instala en el dispositivo. Como no proviene de la tienda de aplicaciones, el sistema solicita autorización para instalar aplicaciones de origen desconocido, permiso que debe concederse una sola vez.

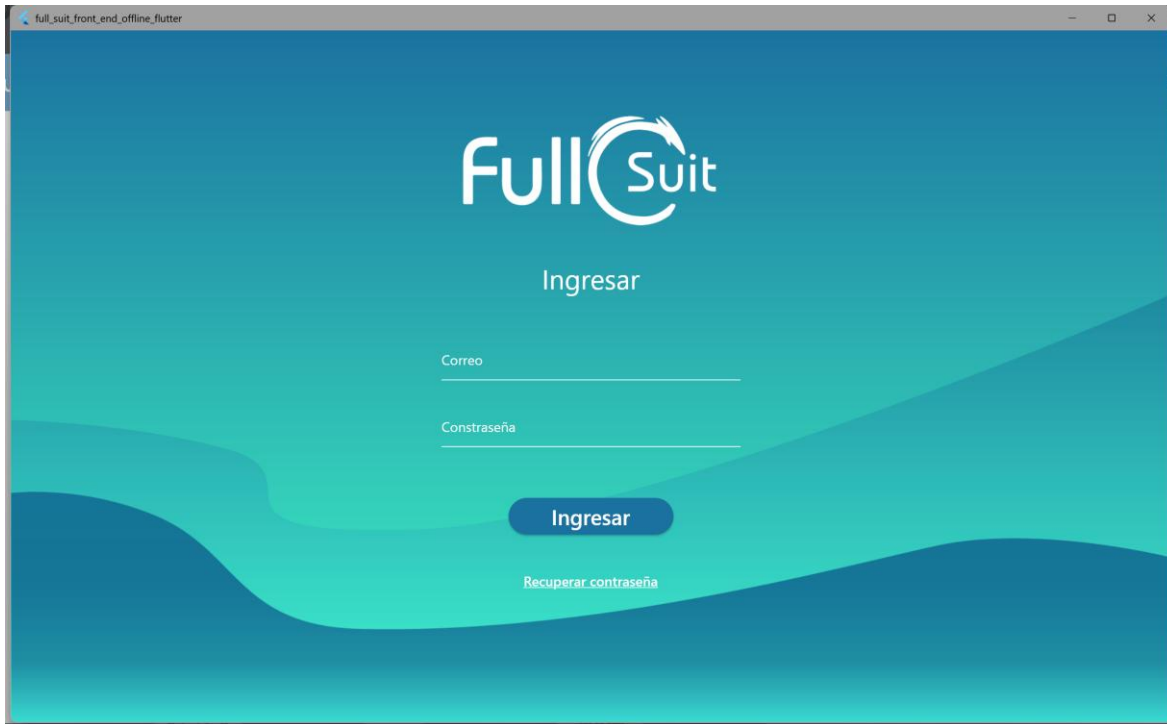
La versión mínima admitida es Android 5.0.

9.2. Acceso al sistema

9.2.1. Inicio de sesión

Al abrir la aplicación se presenta la pantalla de acceso, donde se ingresan el correo y la contraseña asignados al usuario.

Figura 37
Pantalla de inicio de sesión



Fuente: Autor

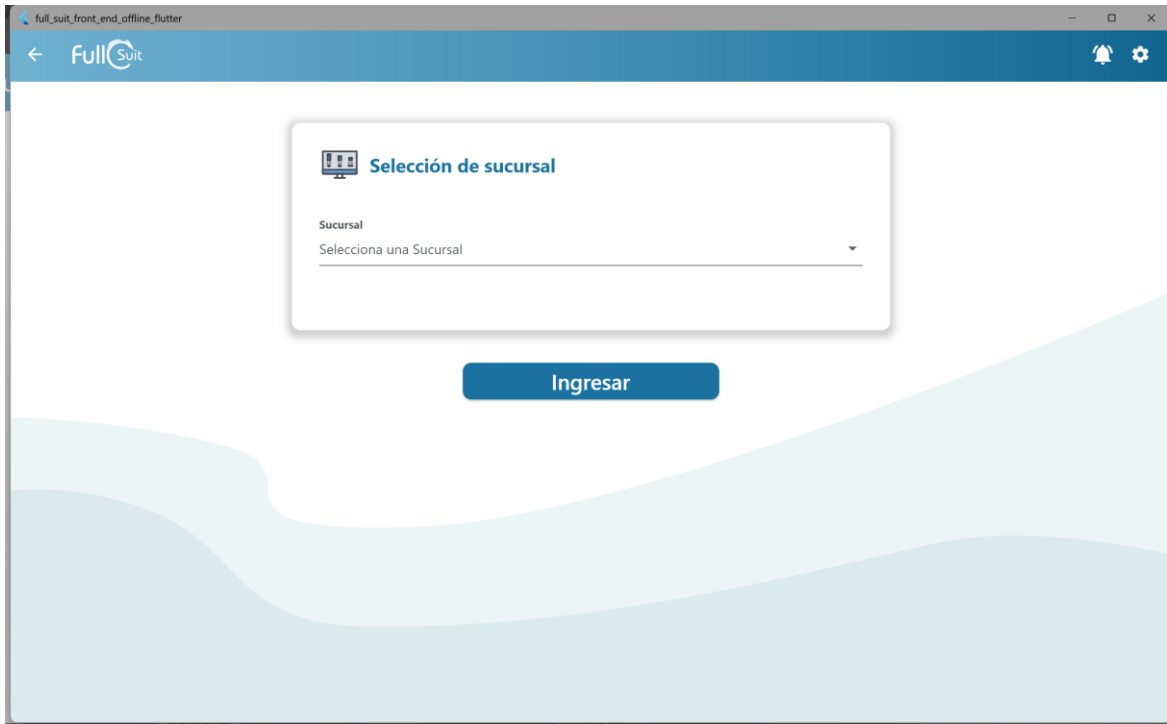
Si la organización se encuentra bloqueada por la empresa, el sistema lo informa con un mensaje específico en lugar de indicar que las credenciales son incorrectas, de modo que el usuario sepa que debe comunicarse con soporte.

Cuando el dispositivo no dispone de conexión pero existe una sesión previa válida, la aplicación permite el ingreso en modo local. Esta decisión es deliberada, porque un punto de venta que exigiera conexión para dejar entrar al cajero convertiría cualquier caída de red del establecimiento en una interrupción total del servicio, obligando a suspender la atención hasta que el proveedor restituyera el enlace, que es justo lo que una aplicación concebida para operar sin conexión debe evitar.

9.2.2. Selección de sucursal

Un mismo usuario puede tener acceso a varias sedes de la organización. Tras el ingreso, la aplicación presenta las sucursales disponibles y solicita elegir aquella en la que se va a operar.

Figura 38
Selección de sucursal



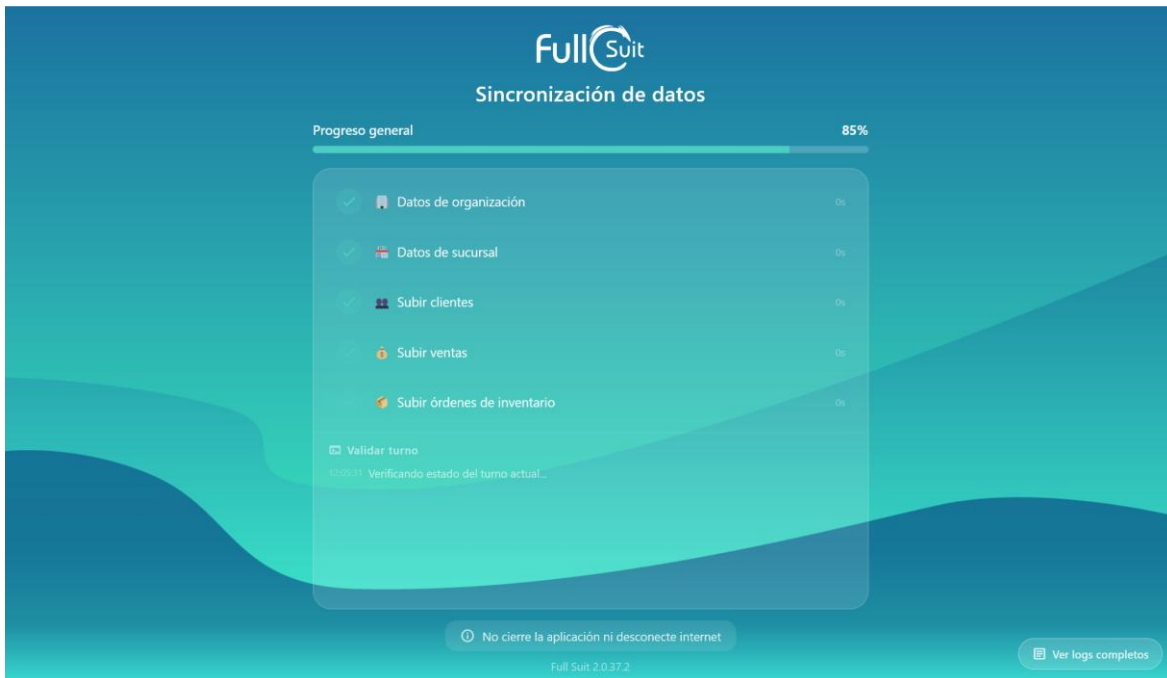
Fuente: Autor

La elección determina el catálogo de artículos, los precios y las resoluciones de facturación que se utilizarán durante la jornada, ya que el precio de un mismo producto puede diferir entre sedes.

9.2.3. Sincronización de datos

Antes de operar, la aplicación descarga la información de la organización y de la sucursal, y sube al servidor lo que haya quedado pendiente de envíos anteriores.

Figura 39
Pantalla de sincronización de datos



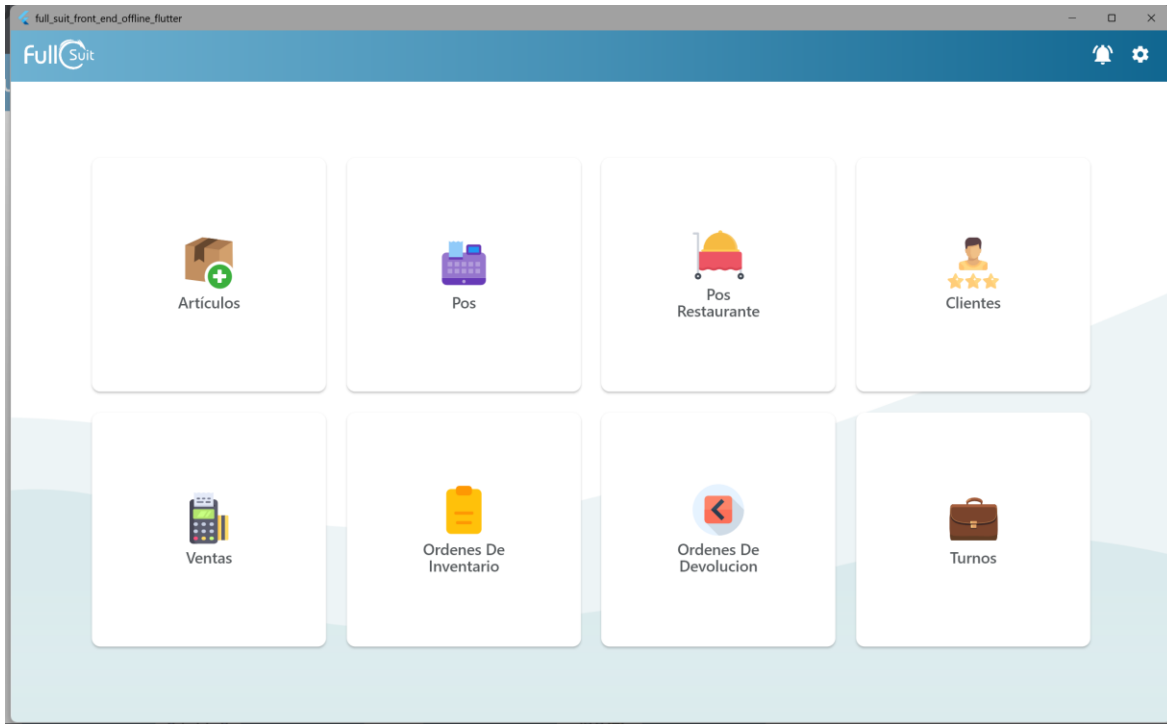
Fuente: Autor

El proceso se compone de siete pasos que se ejecutan en orden y cuyo avance se muestra de forma individual. Si alguno falla, el proceso continúa con los siguientes y al terminar presenta el listado de los registros que no pudieron sincronizarse, con el motivo de cada uno, de manera que el usuario sepa qué venta, qué cliente o qué orden de inventario quedó pendiente en lugar de enfrentarse a un mensaje de error genérico que le obligue a repetir toda la operación sin saber qué la provocó.

9.2.4. Pantalla principal

Superada la sincronización, la aplicación presenta el menú principal con los módulos disponibles.

Figura 40
Pantalla principal o lobby

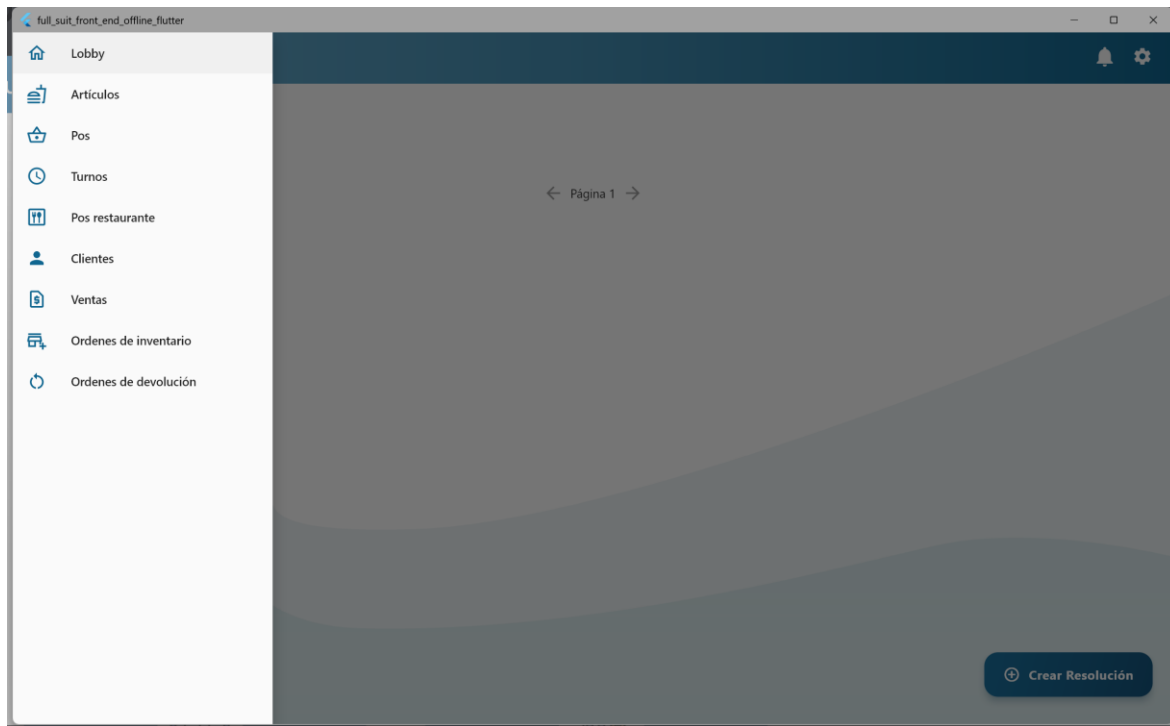


Fuente: Autor

En la versión de escritorio los accesos se distribuyen en una cuadrícula amplia. En dispositivos móviles la misma cuadrícula se reduce a un máximo de tres columnas para conservar un tamaño de pulsación cómodo.

Los mismos módulos están disponibles en todo momento desde el menú de navegación, que se despliega con el botón situado en el extremo superior izquierdo.

Figura 41
Menú de navegación



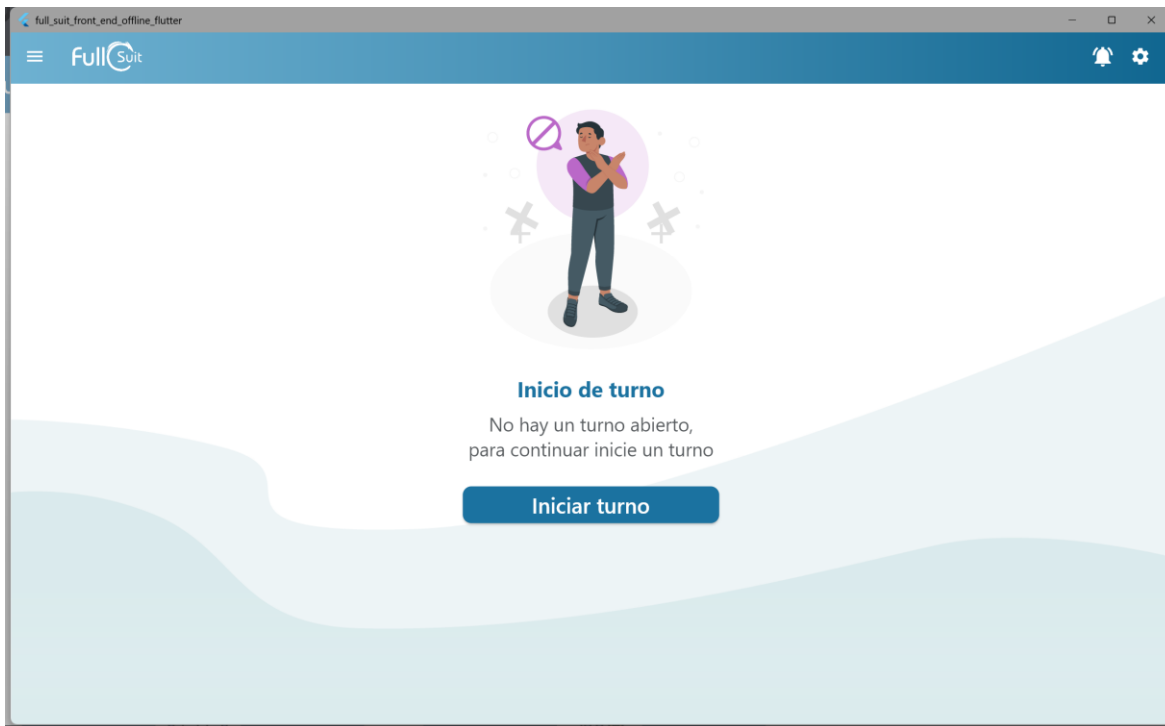
Fuente: Autor

9.3. Gestión de turnos

9.3.1. Apertura de turno

Antes de registrar cualquier venta es necesario abrir el turno de caja. La aplicación lo solicita al ingresar al punto de venta si no encuentra un turno activo.

Figura 42
Apertura de turno



Fuente: Autor

Al confirmar la apertura, el sistema registra el usuario, la sucursal y la fecha y hora de inicio. Desde ese momento, cada venta que se registre queda asociada al turno.

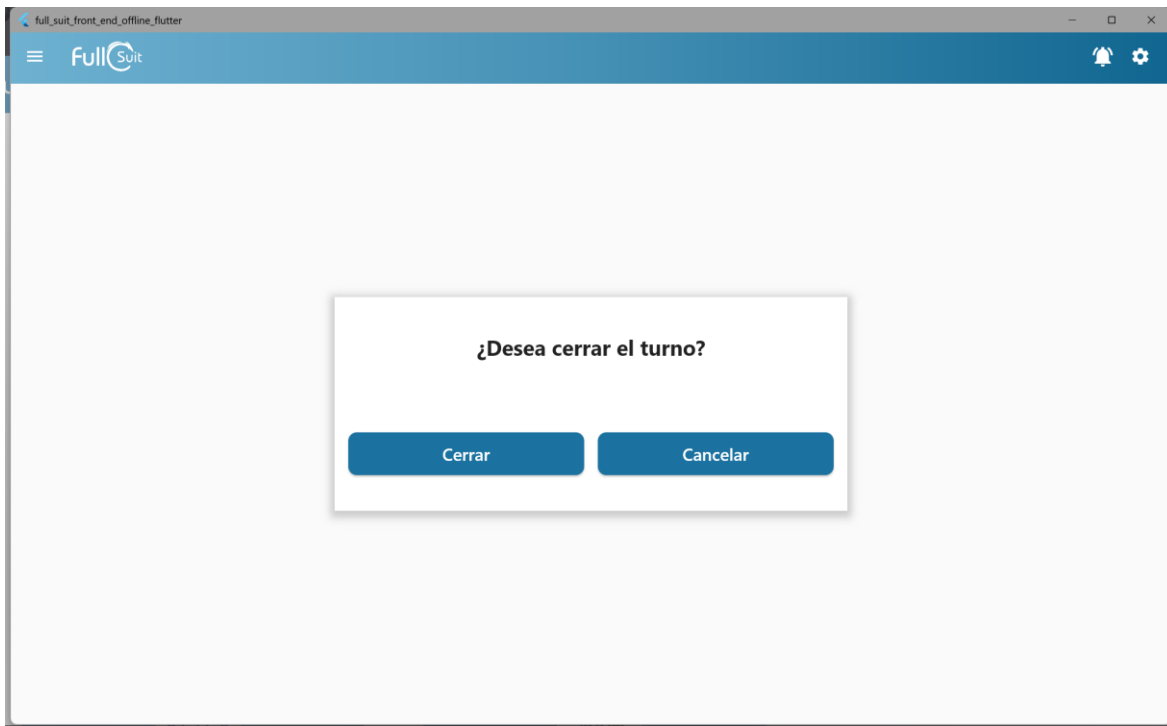
Si el usuario tiene un turno abierto en la plataforma web, la aplicación lo advierte y no permite abrir uno nuevo hasta que aquel se cierre, con el fin de evitar que las ventas queden repartidas entre dos turnos simultáneos.

Cuando existe un turno abierto en el servidor pero no en el dispositivo, por haberse iniciado desde otro equipo, se ofrece la opción de recuperarlo.

9.3.2. Cierre de turno

El cierre se realiza desde el menú superior de la aplicación. Antes de completarlo, el sistema ejecuta la sincronización para asegurar que todas las ventas de la jornada hayan sido enviadas al servidor.

Figura 43
Cierre de turno



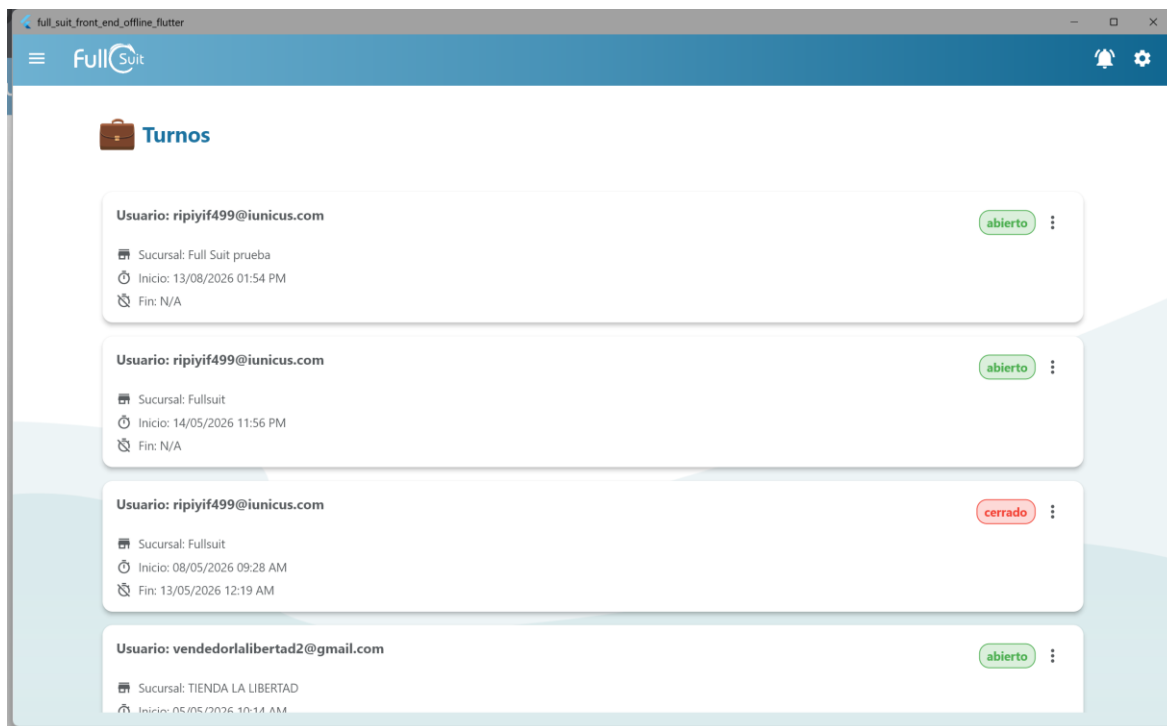
Fuente: Autor

Concluido el cierre, la aplicación ofrece la descarga del documento de cuadre de caja.

9.3.3. Recibo de cierre y consulta de turnos anteriores

El módulo de turnos presenta el histórico de la sucursal y permite generar el recibo de cualquier turno anterior.

Figura 44
Histórico de turnos



Fuente: Autor

Cada registro muestra el usuario que abrió el turno, la sucursal, la hora de inicio, la de cierre cuando existe, y su estado. Desde el menú de cada tarjeta se accede a la generación del documento de cuadre y del recibo.

Figura 45
Recibo de cierre de turno

FULL COLOMBIA SAS

Reporte de ventas

Nit: 901402077

Fecha de inicio:
13:54:21 13/08/2026

Fecha de fin:
15:09:15 13/08/2026

Punto de venta:

Full Suit prueba

Impreso por:

ripiyif499@iunicus.com

Colaborador:
prueba

Total ventas: 2

UD PRODUCT TOTAL

S O

3 ARROZ \$15.470,00
IVA 19

1 Chocorram \$1.000,00
o editado

1 Alpinito \$2.160,00

Total descuento: \$,00

Total: \$19.325,00

Discriminación por tipo de
venta

Tipo	Total
Electronica	\$,00
Pos	\$19.325,00

Discriminación de
impuestos

Tipo	BASE
IVA	\$2.470,00
INC	\$160,00

Discriminación por formas
de pago

EFFECTIVO \$19.325,00

Fuente: Autor

El recibo reúne el detalle de los artículos vendidos, el total de ventas, el total de descuentos, la separación entre documento POS y factura electrónica, la discriminación de impuestos por tarifa y el desglose de los medios de pago recibidos. Su formato se ajusta al ancho del papel de la impresora del establecimiento.

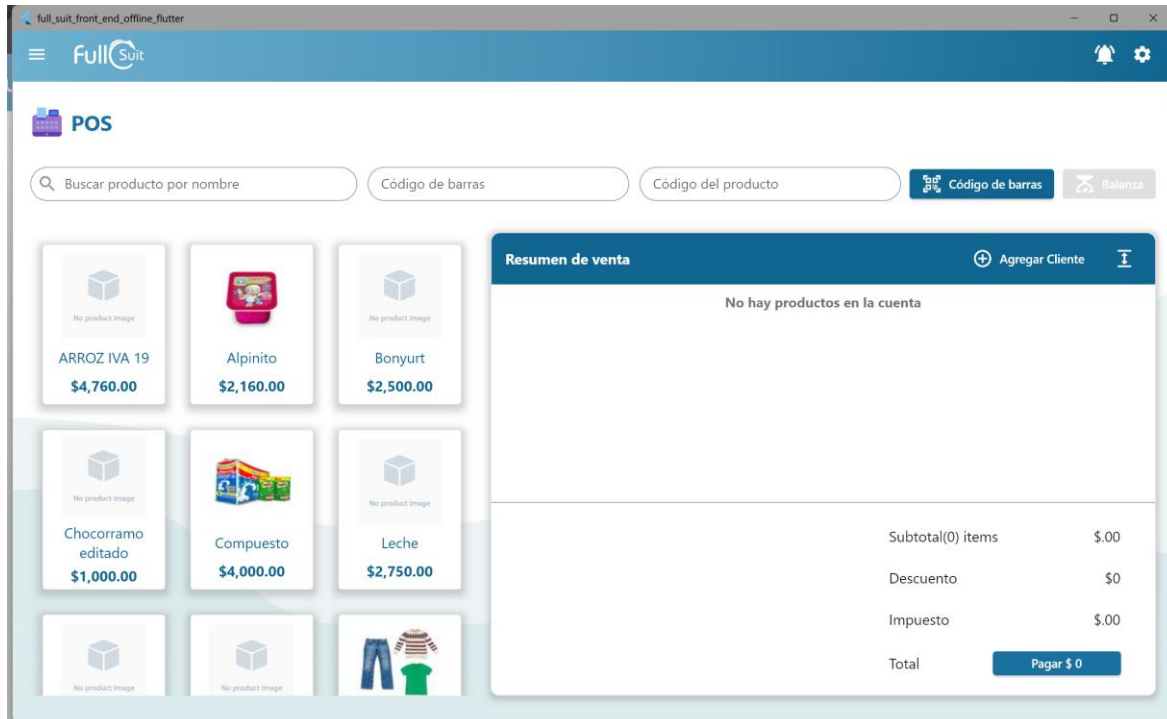
Cuando el turno no registra ventas, o cuando estas no se han sincronizado todavía, la aplicación informa la causa concreta e indica que debe sincronizarse antes de reintentar.

9.4. Punto de venta comercial

9.4.1. Búsqueda y selección de artículos

La pantalla del punto de venta presenta el catálogo de la sucursal y tres formas de localizar un artículo: por nombre, por código de barras y por código de producto. Adicionalmente, el botón de código de barras habilita la captura mediante lector físico.

Figura 46
Búsqueda de artículos en el punto de venta



Fuente: Autor

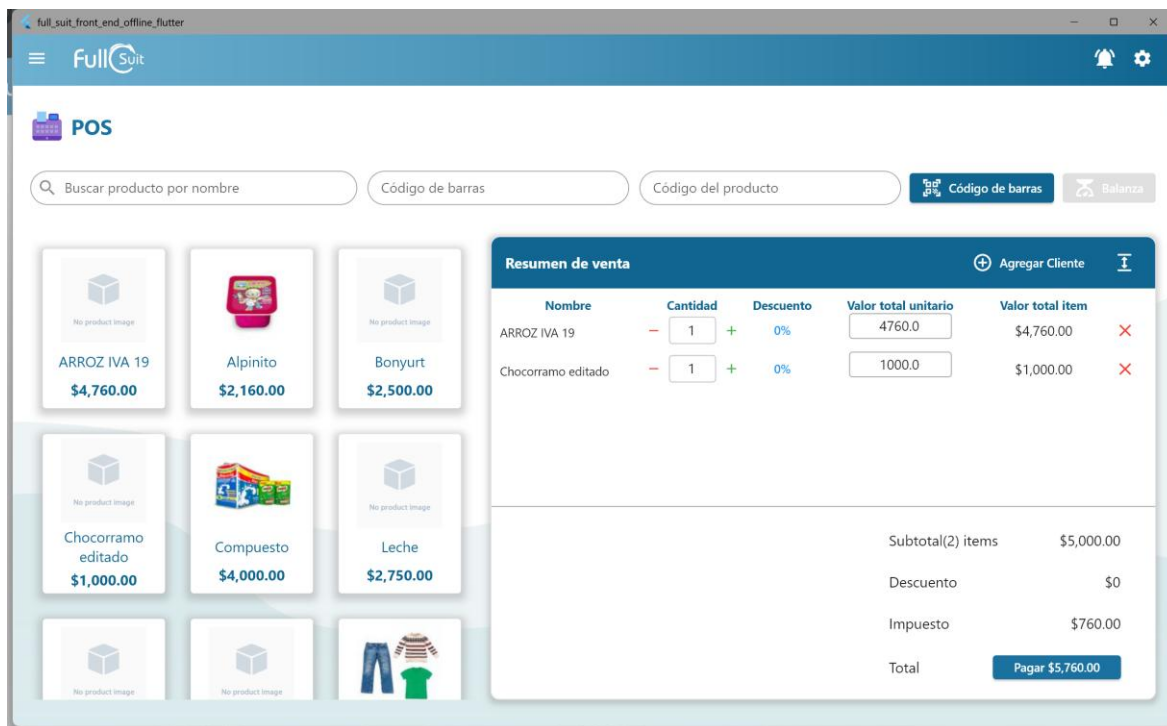
Con el modo de escaneo activo, la aplicación recibe los caracteres que emite el lector y agrega el artículo a la venta al detectar el fin de la lectura. Si el usuario selecciona cualquier campo de texto, el escaneo se desactiva automáticamente para no interferir con la escritura.

Al pulsar sobre un artículo del catálogo, este se agrega al resumen de venta con cantidad uno.

9.4.2. Modificación de cantidades y del precio

Cada línea del resumen de venta permite ajustar la cantidad mediante los botones de incremento y decremento, o escribiéndola directamente. También admite cantidades decimales, necesarias para la venta por peso.

Figura 47
Resumen de venta



Fuente: Autor

La columna de valor total unitario es editable. En ella se escribe el precio final que se va a cobrar por unidad, con el impuesto ya incluido. Al modificarlo, el sistema recalcula de inmediato la base gravable, el descuento y el impuesto correspondientes, y actualiza los totales de la venta.

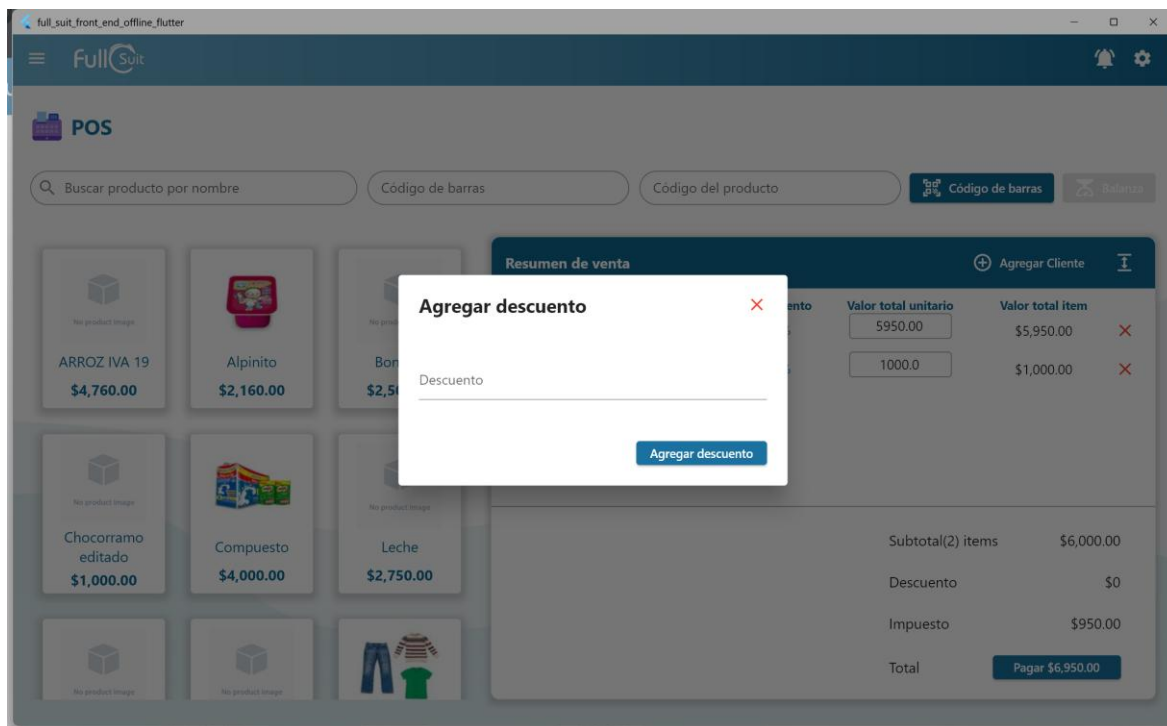
Escribir un precio manualmente anula el descuento porcentual que tuviera la línea. Los dos mecanismos son excluyentes por diseño, porque un precio impuesto a mano ya incorpora la rebaja que el cajero haya decidido conceder, de modo que aplicar encima un porcentaje descontaría dos veces sobre una base que el propio usuario acaba de alterar, y el valor resultante dejaría de corresponder con lo pactado con el cliente.

Si el precio se deja vacío o en cero, el campo se marca en rojo y el botón de cobro permanece inhabilitado hasta que se corrija. La aplicación tampoco permite facturar artículos cuyo valor total sea cero.

9.4.3. Aplicación de descuentos

El porcentaje de descuento de cada línea se modifica pulsando sobre el valor correspondiente.

Figura 48
Diálogo de descuento



Fuente: Autor

El descuento admite decimales y acepta tanto la coma como el punto como separador. Si el artículo tiene configurada una tasa máxima de descuento, el sistema limita el valor a ese tope.

9.4.4. Verificación detallada de valores

El botón situado en el extremo superior derecho del resumen de venta amplía la vista y presenta la descomposición completa de cada línea.

Figura 49

Vista ampliada con la descomposición de valores por artículo

Nombre	Subtotal Unit.	% Desc	Valor Desc Unit.	Subtotal Unit. - Desc.	% Imp	Valor Imp Unit.	Valor Total Unit.	Cantidad producto	Total Item
ARROZ IVA 19	\$4,000.00	0%	\$0.00	\$4,000.00	19%	\$760.00	4760.0	1	\$4,760.00
Alpinito	\$2,000.00	0%	\$0.00	\$2,000.00	8%	\$160.00	2160.0	1	\$2,160.00
Bonyurt	\$2,500.00	0%	\$0.00	\$2,500.00	0%	\$0.00	2500.0	1	\$2,500.00

Subtotal(3) items	\$8,500.00
Descuento	\$0
Impuesto	\$920.00
Total	Pagar \$9,420.00

Fuente: Autor

La tabla muestra, para cada artículo, el subtotal unitario, el porcentaje y el valor del descuento, el subtotal una vez descontado, el porcentaje y el valor del impuesto, el valor total unitario, la cantidad y el total de la línea. El campo de precio permanece editable dentro de esta vista, de modo que al modificarlo se observa cómo se recalculan todas las columnas.

Esta pantalla resulta útil para verificar la composición de un valor antes de facturar, en particular cuando se ha negociado un precio con el cliente.

9.4.5. Cobro de la venta

El botón de pago abre el diálogo de cierre de la transacción.

Figura 50
Diálogo de cobro

Crear venta

Tipo de venta: POS

Descripción:

Efectivo

Valor pagado con Efectivo: 10000

Cliente: usuario no registrado - 22222222222

Retención ☐ Crédito ☐

Propina

0% 5% 10% 15%

Porcentaje (%): 10

Valor: 695

Equivale al 10% de la venta

Cambio:	\$2,355.00
Saldo:	\$0.00
Total:	\$6,950.00
Total con propina:	\$7,645.00

Fuente: Autor

En él se selecciona el cliente, se elige el tipo de documento entre documento POS y factura electrónica, y se registran los medios de pago. Es posible repartir el cobro entre varios medios.

La aplicación valida que la suma de los pagos no exceda el valor total. La única excepción es el pago en efectivo como medio único, en cuyo caso se admite un valor superior y se calcula el cambio.

Si la organización tiene habilitada la propina, el diálogo presenta la sección correspondiente.

Figura 51
Registro de propina

Crear venta

Tipo de venta: POS

Descripción:

Medios de pago:

Cliente: usuario no registrado - 22222222222

Retención ☐ Crédito ☐

Propina

0% 5% 10% 15%

Porcentaje (%) 0

Valor 0

Equivalente al 0% de la venta

Cambio: \$0.00

Saldo: \$6,950.00

Total: \$6,950.00

Total con propina: \$6,950.00

Fuente: Autor

La propina puede indicarse como porcentaje o como valor, y ambos campos se mantienen sincronizados entre sí. Si después se modifica la venta, la propina se recalcula conservando el porcentaje. El valor de la propina se suma al total a cobrar pero no forma parte de la base gravable, tal como corresponde.

9.4.6. Impresión del documento

Registrada la venta, la aplicación presenta la vista previa del documento.

Figura 52
Vista previa de la factura

TU TIENDA LIBERTAD

NIT 222222222222-7

Sucursal: Full Suit prueba

cil 3 #4-45, Chiscas

Fecha: 8/13/2026

Hora: 14:41:06

INFORMACIÓN DEL CLIENTE

usuario no registrado

Doc: 222222222222-7

CÓDIGO	DESCRIPCIÓN		
CANT	V.UNIT	%DS	TOTAL
d984a743	ARROZ IVA 19		
1.0 un	\$5.950,00	%0.0	\$5.950,00
choc01	Chocorramo editado		
1.0 un	\$1.000,00	%0.0	\$1.000,00

PROPINA

Porcentaje: 10.0% \$695,00

Descuento en venta: \$,00

TOTAL: \$6.950,00

Total + propina: \$7.645,00

Medio de pago: EFECTIVO

Tipo de venta: Recibo

PAGOS

Efectivo \$7.645,00

Vendedor: prueba

DISCRIMINACIÓN DE IMPUESTOS

Tipo	Base	Impuesto
TARIFA	BASE	IMPUESTO
c=19%	\$5.000,00	\$950,00
EXE=0%	\$1.000,00	\$,00
Total	\$6.000,00	\$950,00

Fuente: Autor

El documento incluye los datos del establecimiento y del cliente, el detalle de los artículos con su descuento e impuesto, el bloque de discriminación de impuestos por tarifa, los medios de pago empleados y el código de consulta de la factura electrónica.

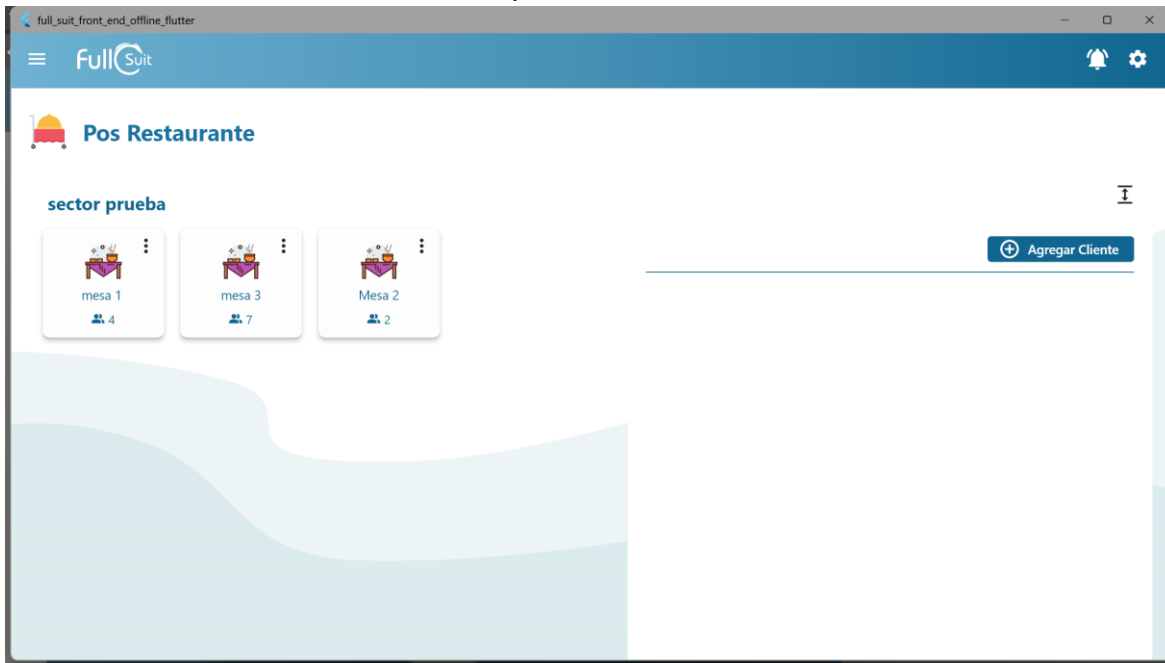
Desde esta pantalla se puede imprimir o iniciar una nueva venta. Al elegir la segunda opción, la aplicación regresa al punto de venta desde el que se originó la transacción.

9.5. Punto de venta restaurante

9.5.1. Sectores y mesas

El punto de venta de restaurante organiza la operación por sectores y mesas.

Figura 53
Sectores y mesas del restaurante



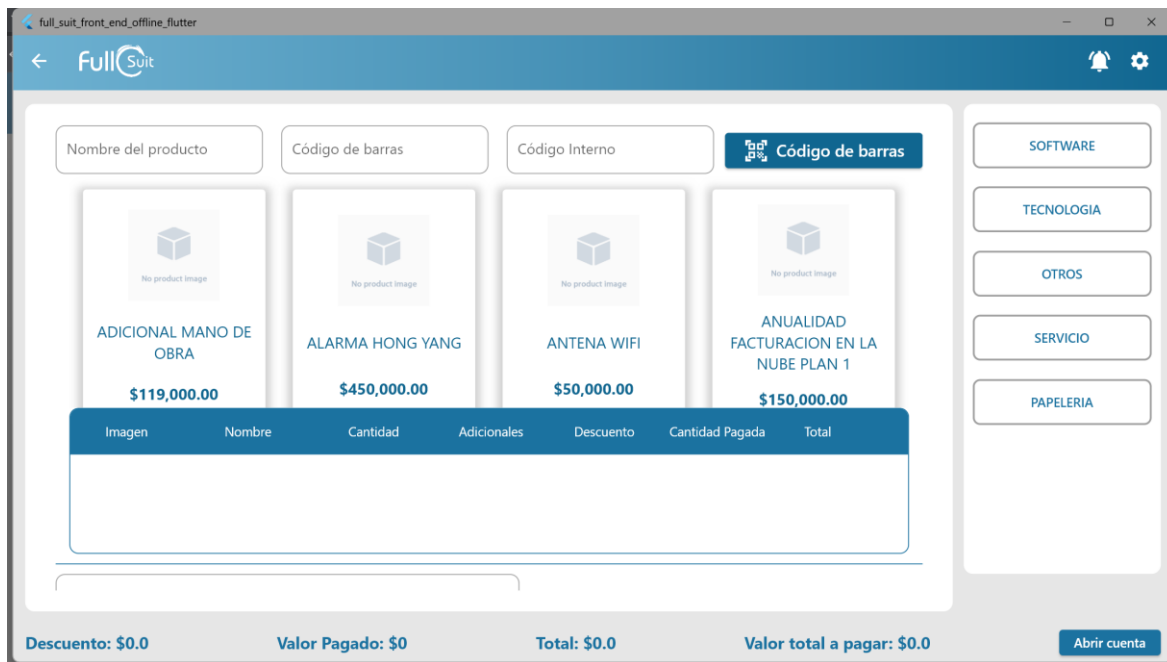
Fuente: Autor

Las mesas ocupadas se distinguen de las libres, y desde el menú de cada una se accede a las operaciones de cambio de mesa y unificación de cuentas.

9.5.2. Apertura y edición de cuenta

Al seleccionar una mesa libre se abre una cuenta nueva, indicando el número de comensales.

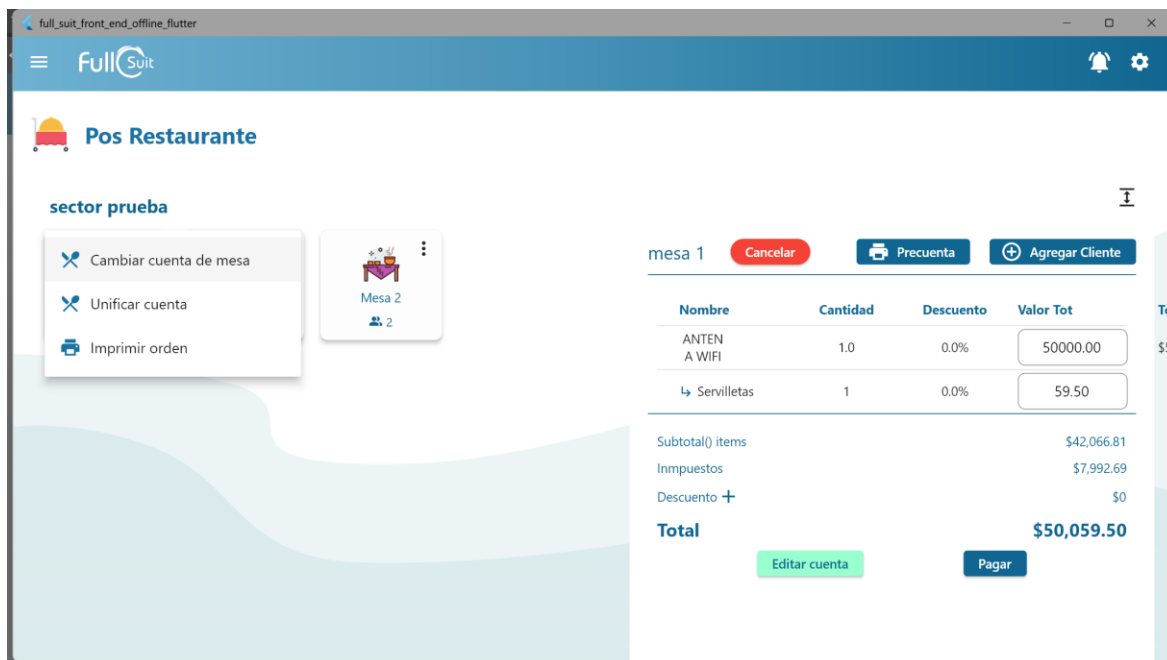
Figura 54
Apertura de cuenta



Fuente: Autor

La cuenta permanece almacenada en el dispositivo, de modo que puede consultarse y modificarse durante el servicio aunque la aplicación se cierre. El sistema impide abrir dos cuentas simultáneas sobre la misma mesa.

Figura 55
Edición de la cuenta



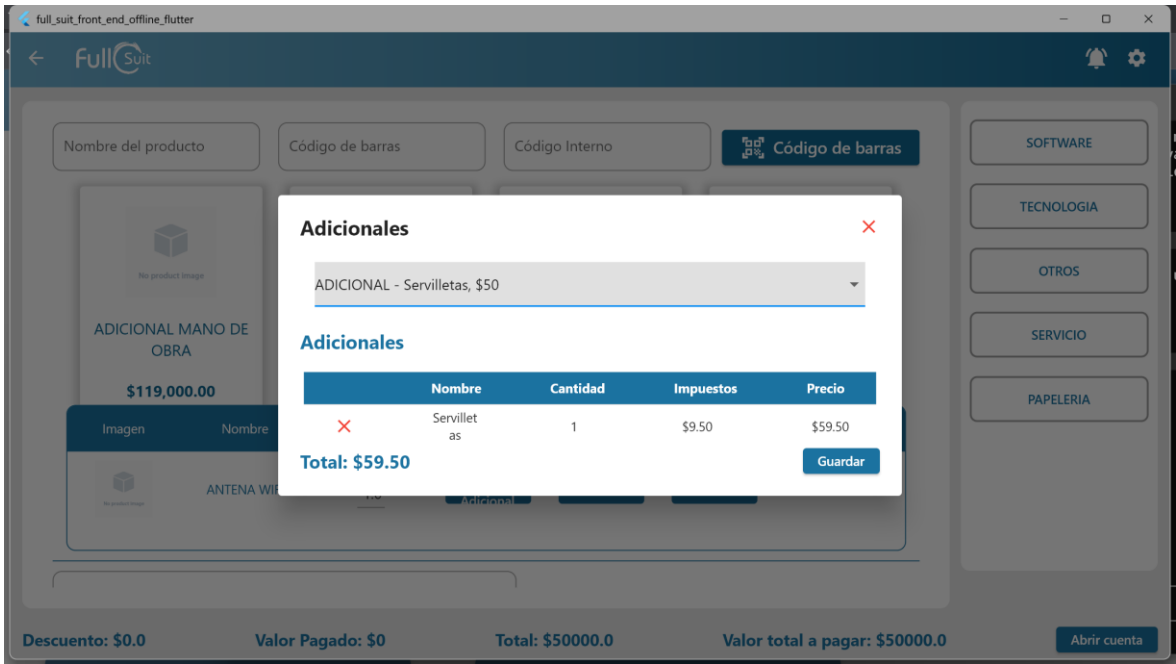
Fuente: Autor

Los artículos se agregan igual que en el punto de venta comercial, con la misma edición de precio y descuento por línea.

9.5.3. Adiciones

Cada artículo admite adiciones, que se registran desde el botón correspondiente de su fila.

Figura 56
Adiciones sobre un artículo



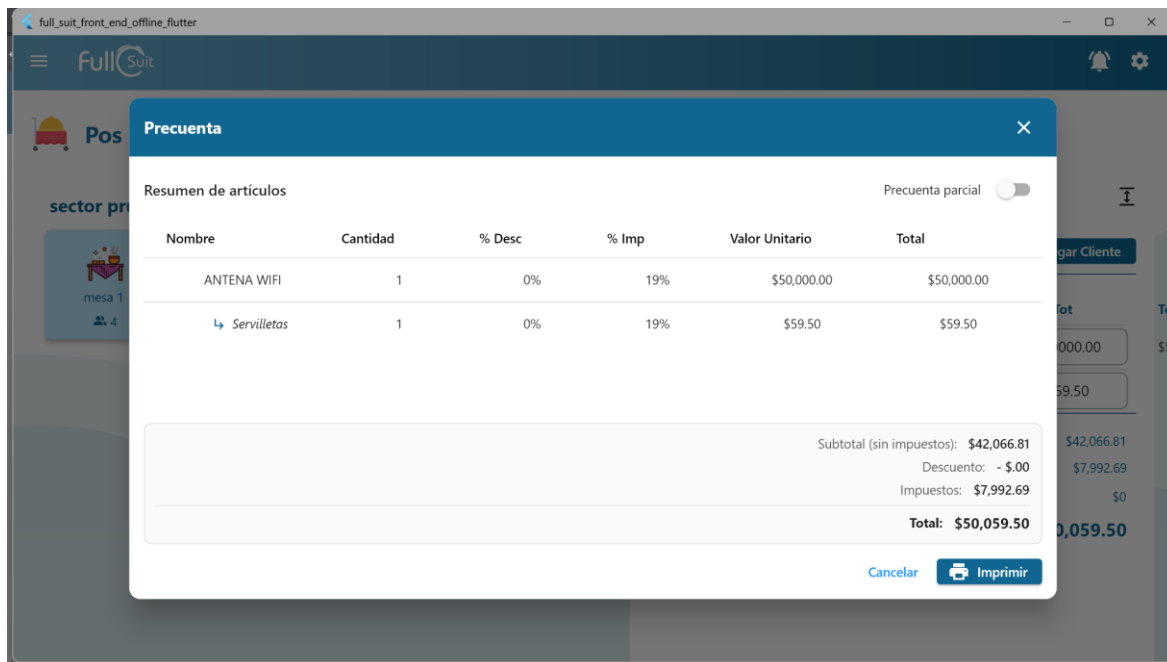
Fuente: Autor

Las adiciones llevan su propio precio, su propio impuesto y su propio campo editable, y se muestran como líneas dependientes del artículo al que pertenecen.

9.5.4. Precuenta y comanda

Antes de cobrar puede entregarse al comensal la precuenta, que no constituye documento fiscal.

Figura 57
Precuenta

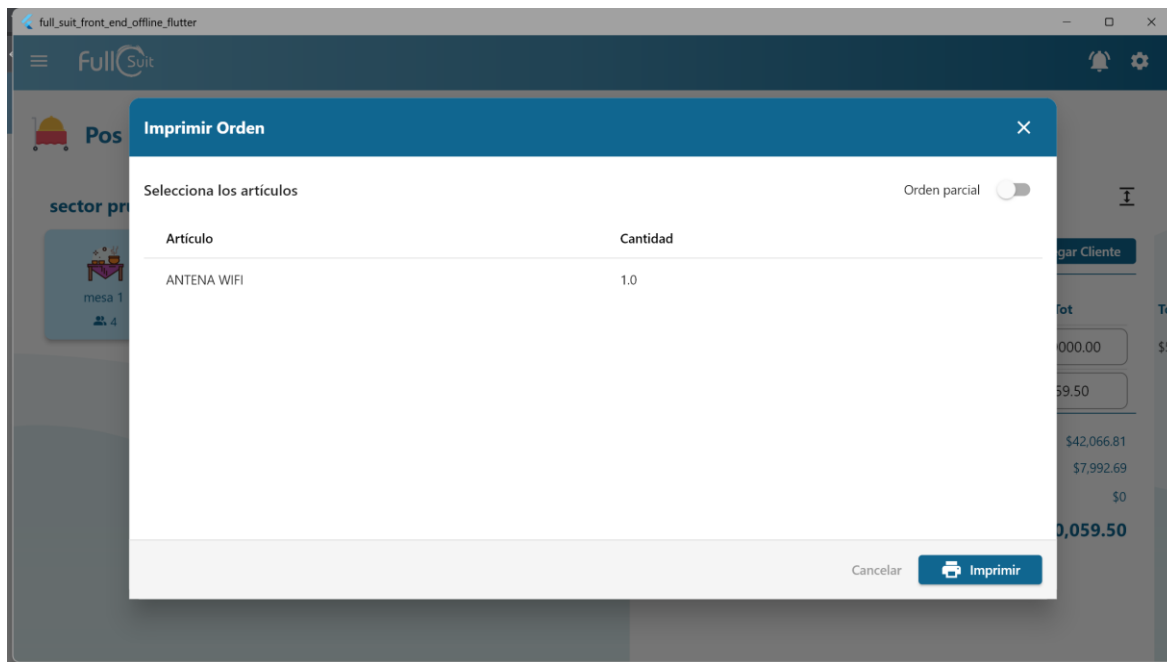


Fuente: Autor

La precuenta admite selección parcial por cantidad, lo que permite dividir el consumo entre varios comensales, y al indicar que se cobra únicamente una fracción de un artículo el sistema prorratea sus adiciones en la misma proporción, de modo que quien paga la mitad de un plato compartido asume también la mitad de los acompañamientos que se le agregaron y no el total.

La comanda de cocina se genera de forma independiente y permite seleccionar qué artículos se envían a preparación.

Figura 58
Comanda de cocina



Fuente: Autor

9.6. Módulos de apoyo

9.6.1. Clientes

El módulo de clientes permite consultar el directorio local y registrar clientes nuevos.

Figura 59
Creación de cliente

Crear cliente

Autocompletar

Cliente extranjero

Nombres*

Apellido (opcional)

Email*

Tipo de documento*

Número de Documento*

Dígito de verificación

0

Número de Teléfono*

Dirección*

Seleccione un departamento*

Seleccione una ubicación*

Crear Cliente

Fuente: Autor

El formulario valida el formato de cada campo antes de permitir el registro: longitud de nombres y apellidos, estructura del correo electrónico, documento numérico dentro de un rango válido y teléfono entre siete y diez dígitos. La ubicación se completa automáticamente a partir de los datos de la organización.

Para clientes extranjeros se dispone de una opción específica que sustituye la ubicación nacional por el país de origen y habilita el tipo de documento correspondiente.

9.6.2. Artículos, inventario y resoluciones

El módulo de artículos presenta el catálogo de la sucursal con búsqueda por nombre, código y código de barras, y filtros por categoría. Los módulos de órdenes de inventario y de devoluciones permiten registrar los movimientos correspondientes. El módulo de resoluciones

administra las resoluciones de facturación autorizadas, con validación de los rangos de numeración y de las fechas de vigencia.

El módulo de ventas presenta el histórico con filtros por fecha y por cliente, y permite reimprimir cualquier documento emitido.

9.7. Sincronización y diagnóstico

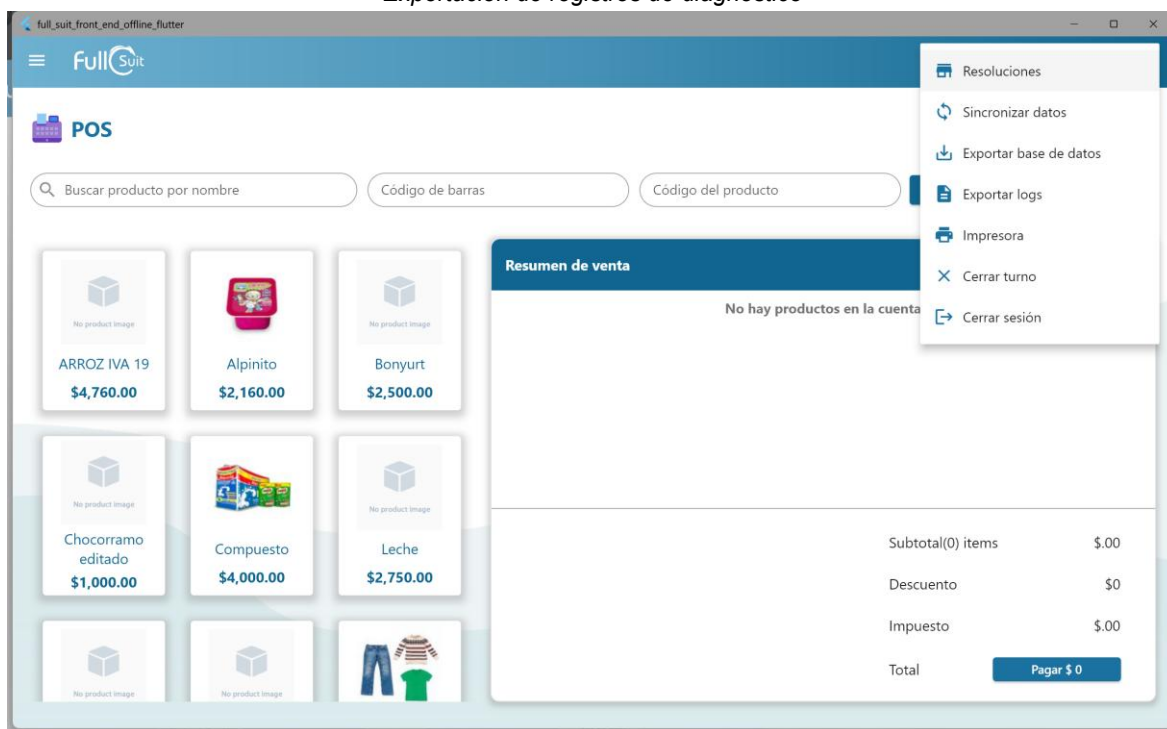
9.7.1. Sincronización manual

Además de la sincronización automática al ingresar y al cerrar turno, el proceso puede ejecutarse en cualquier momento desde el menú superior. Es recomendable hacerlo tras un periodo prolongado sin conexión.

9.7.2. Exportación de registros para soporte

La aplicación registra en un archivo local las operaciones realizadas y los errores encontrados. Cuando se presenta un comportamiento anómalo, el personal de soporte puede solicitar ese archivo.

Figura 60
Exportación de registros de diagnóstico



Fuente: Autor

Desde el menú superior se selecciona la opción de exportar registros y el sistema entrega el archivo mediante el selector del dispositivo, desde donde puede enviarse por el medio habitual. La operación no requiere conocimientos técnicos ni herramientas adicionales.

De la misma forma puede exportarse la base de datos local, cuando el diagnóstico requiere reproducir una situación con los datos reales del establecimiento.

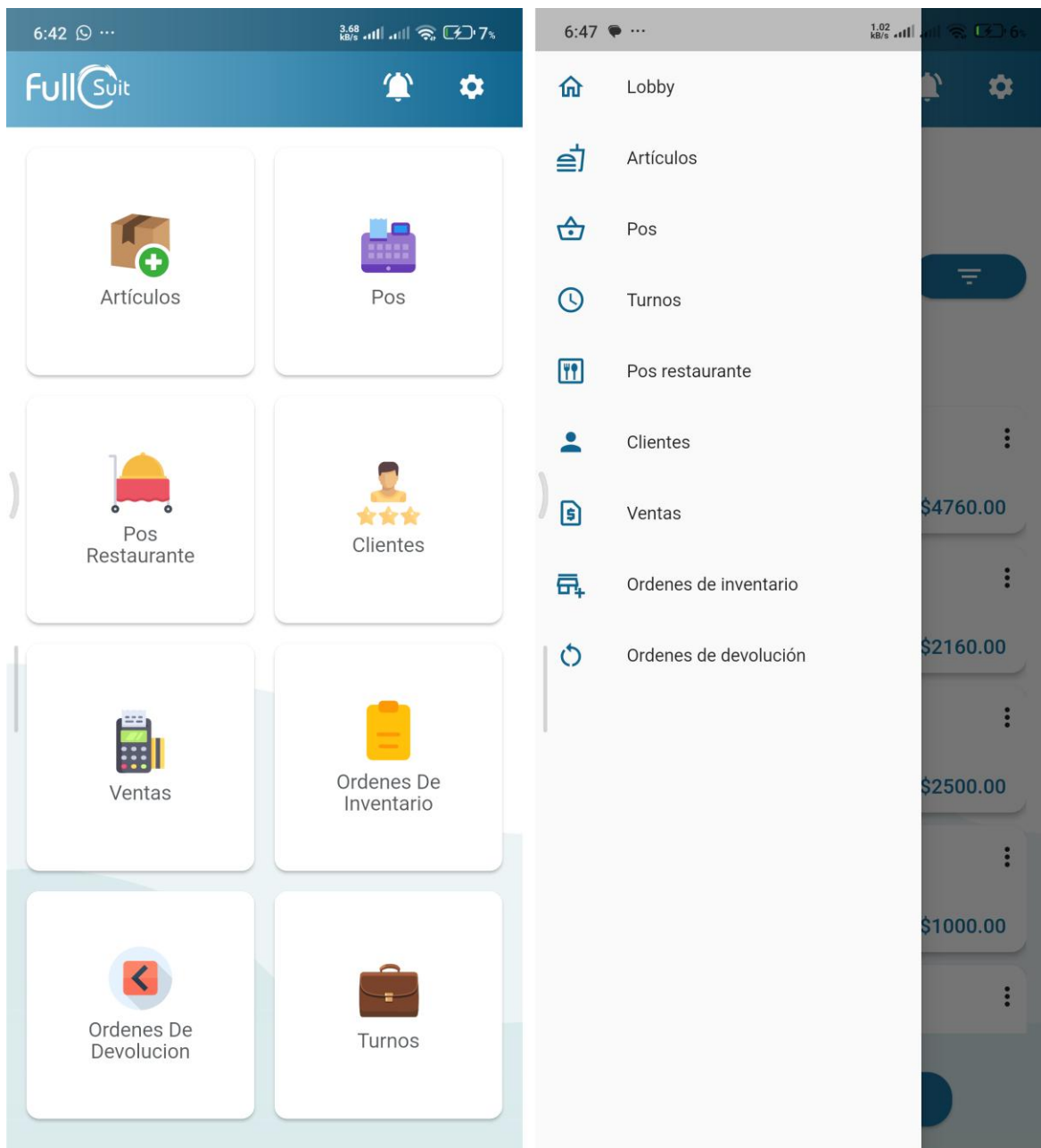
9.8. Particularidades de la versión Android

La versión para dispositivos móviles conserva la funcionalidad de la de escritorio, con las diferencias de presentación que impone el tamaño de pantalla.

Los módulos de consulta y los formularios adaptan su disposición al ancho disponible. La pantalla principal presenta los accesos en dos columnas en lugar de cuatro, y la navegación lateral ocupa el alto completo del dispositivo en vez de mantenerse plegada junto al contenido.

Figura 61

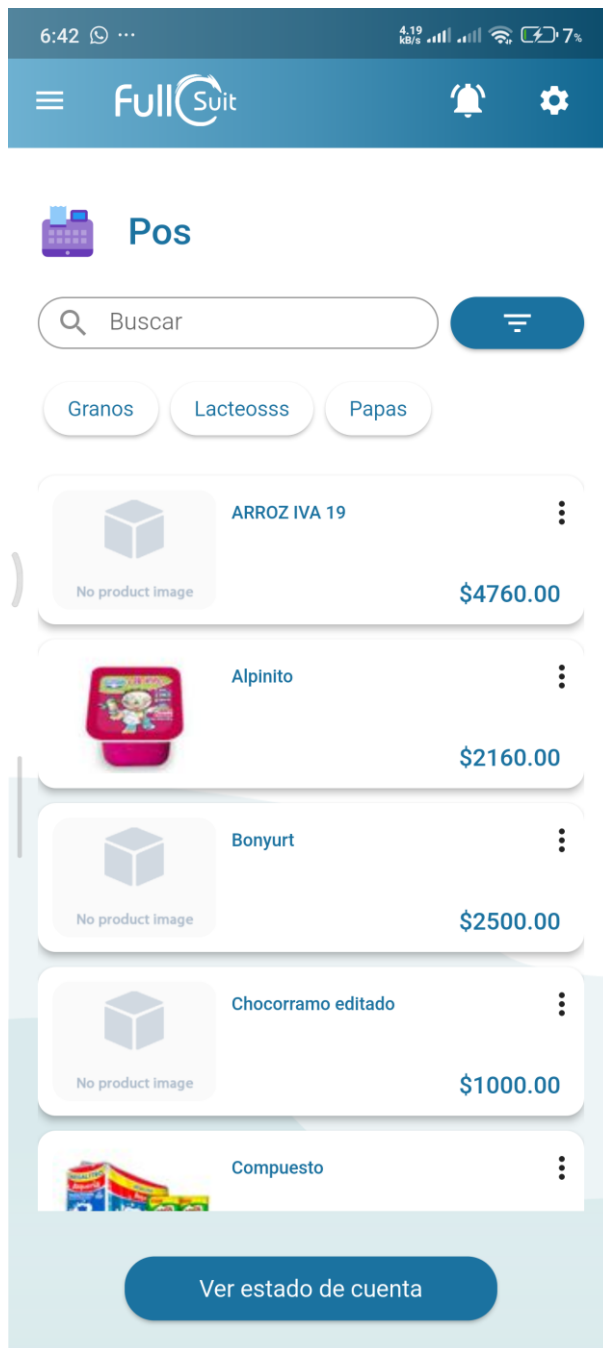
Pantalla principal y navegación en la versión Android



Fuente: Autor

Los dos puntos de venta, en cambio, no se limitan a reacomodar los elementos: se reorganizan en pantallas sucesivas. El catálogo, el detalle del artículo y el estado de la cuenta se presentan por separado en lugar de compartir una sola vista, porque los tres no caben simultáneamente en un teléfono sin volverse ilegibles.

Figura 62
Punto de venta comercial en la versión Android



Fuente: Autor

El catálogo ocupa la pantalla completa. Presenta los artículos en lista vertical, con el buscador y los filtros por categoría arriba y el acceso al estado de la cuenta fijado abajo, de manera que el cajero pueda consultar el total sin abandonar la búsqueda. Al tocar un artículo se abre su detalle.

Figura 63
Detalle de artículo en la versión Android



Fuente: Autor

En la pantalla de detalle se ajustan la cantidad, el precio unitario y el descuento del artículo seleccionado. El precio se confirma antes de incorporarlo a la cuenta, y si se abandona la pantalla sin confirmar, el artículo no queda registrado en la venta.

Figura 64
Estado de la cuenta en la versión Android



Fuente: Autor

El estado de la cuenta recoge los artículos incorporados y, en la franja inferior, el subtotal, el descuento y el impuesto separados del total a cobrar. El cálculo es el mismo de escritorio. Un artículo ofrecido en \$4.760 aparece con una base de \$4.000 y un impuesto de \$760, que es el resultado de aplicar hacia atrás la tarifa del 19 % sobre el precio de venta, igual que en el equipo del mostrador.