



ÍNDICE DE CONTENIDO

[1. INTRODUCCIÓN 6](#_heading=h.yfzm7qy8uz93)

[1.1. Contexto del Problema 6](#_heading=h.yq20r64f03n)

[2. OBJETIVOS 6](#_heading=h.1c1r5idbms76)

[2.1. Objetivo General 6](#_heading=h.j0193d48gcwc)

[2.2. Objetivos Específicos 6](#_heading=h.f2nartcm2ck4)

[3. ALCANCE DEL PROYECTO 7](#_heading=h.yaqpyeh2zzrg)

[3.1. Funcionalidades Implementadas 7](#_heading=h.u1x1f19s48q6)

[3.2. Integraciones 7](#_heading=h.1c9qbxuww1m6)

[4. ARQUITECTURA DEL SISTEMA 8](#_heading=h.th1sgaep1qwv)

[4.1. Arquitectura General 8](#_heading=h.s9hswlia52yt)

[4.2. Arquitectura en Capas 10](#_heading=h.azb84tpuzg8w)

[4.3. Tecnologías Utilizadas 12](#_heading=h.af542pvlfyr)

[5. SISTEMA GENERAL COMERCIALIZADORA DE ELECTRODOMÉSTICOS 12](#_heading=h.4smzejnziauv)

[5.1. Descripción General 12](#_heading=h.t9109fkws4kv)

[5.2. Modelo de Datos - BanQuito 13](#_heading=h.bdc2w3wrs1ha)

[5.2.1. Diagrama Entidad-Relación 13](#_heading=h.tpjdftrzutgc)

[6. PARTE PRÁCTICA 13](#_heading=h.tgxg5rmmciqk)

[6.1. Instalación de herramientas 13](#_heading=h.5ej9stuzmky9)

[6.1.1. Estructura de proyecto 13](#_heading=h.wx3osfrwqtzm)

[6.1.2. Instalación y configuración de herramientas 14](#_heading=h.urvvuxtqu2hq)

[6.1.3. Instalación y configuración de microsoft SQL Server 14](#_heading=h.z2b4zakckqv4)

[6.1.4. Instalación y configuración de SQL Server Management Studio 21 22](#_heading=h.qkw6h0k6vodq)

[6.1.5. Creación de la base de datos 23](#_heading=h.21z9wwgyqcoj)

[6.1.6. Aplicación Servidor 25](#_heading=h.ps84vr8m9w2r)

[6.1.7. Estructura del proyecto 26](#_heading=h.6hj7swr0a1jz)

[6.1.8. Creación de la Base De Datos 28](#_heading=h.pbjaoyigj1qq)

[7. CODIFICACIÓN DE LOS SERVIDORES 30](#_heading=h.flmcddjunfsp)

[7.1. Servidor BanQuito 30](#_heading=h.t7xi0i46vnsy)

[7.1.1. Capa de Servicio (Carpeta WS) 30](#_heading=h.oh6s95oanwu2)

[7.1.2. Capa de Lógica de Negocio (Carpeta Services) 34](#_heading=h.r3mtwg1pd7uy)

[7.1.3. Capa de Acceso a Datos (Carpeta DataAccess) 39](#_heading=h.lud3z9o2zeyp)

[7.1.4. Capa de Modelos y DTOs (Carpetas Models y DTOs) 44](#_heading=h.fy24xnqakqdd)

[7.2. Servidor Comercializadora 46](#_heading=h.chmp3exb6ef5)

[7.2.1. Capa de Servicio (Carpeta WS) 46](#_heading=h.dl3dmxmkqxeu)

[7.2.2. Capa de Lógica de Negocio (Carpeta Services) 47](#_heading=h.1yu6fmwbmosc)

[7.2.3. Capa de Acceso a Datos (Carpeta DataAcces) 54](#_heading=h.aiucyygtmou3)

[7.2.4. Capa de Modelos y DTOs (Carpetas Models y DTOs) 58](#_heading=h.e9tfy9en0o8k)

[8. PROYECTO CLIENTE WEB 62](#_heading=h.50pqi7vq0c4c)

[8.1. Funcionalidades Principales 62](#_heading=h.a3whohucrnww)

[8.2. Capturas de Pantalla del Sistema 62](#_heading=h.yqq2mtx4idez)

[8.3. Integración con los Servicios SOAP 67](#_heading=h.xicu9oe2rw5g)

[9. PROYECTO CLIENTE CONSOLA 67](#_heading=h.nsz3z3xlryr0)

[9.1. Funcionalidades Implementadas 68](#_heading=h.40k8af7fw6wf)

[9.2. Capturas de Pantalla del Sistema de Consola 68](#_heading=h.a9ui1j5atpsr)

[9.3. Integración con los Servicios SOAP 78](#_heading=h.rfpn3onm6qrz)

[10. PROYECTO CLIENTE ESCRITORIO 79](#_heading=h.lj91axswb2ok)

[10.1. Funcionalidades Principales 79](#_heading=h.56kwegjgfz9g)

[10.2. Capturas de Pantalla del Sistema de Escritorio 79](#_heading=h.p59ggx1hrs8r)

[10.3. Integración con los Servicios SOAP 83](#_heading=h.9org53yyg36b)

[11. PROYECTO CLIENTE MÓVIL 84](#_heading=h.nh4qbn3zhx4b)

[11.1. Funcionalidades Principales 84](#_heading=h.nbp9gxfqz0e8)

[11.2. Capturas de Pantalla del Sistema Móvil Android 84](#_heading=h.ag53kjvfww9)

[11.3. Integración con los Servicios SOAP en Android 94](#_heading=h.ncvgndj4fatg)

[12. CONCLUSIONES 95](#_heading=h.n3syncw86i5i)

[13. REFERENCIAS 96](#_heading=h.w72u01q5jh6j)

ÍNDICE DE FIGURAS

[Figura 1. Arquitectura de Software del Sistema Distribuido de Comercialización de Electrodomésticos 8](#_heading=h.50qn0nje3t5d)

[Figura 2. Arquitectura en capas del Sistema BanQuito 10](#_heading=h.i9mlfjatmo1t)

[Figura 5. Creación de la carpeta principal del examen 13](#_heading=h.pz6ru8hjhes4)

[Figura 6. Estructura de las carpetas del proyecto 14](#_heading=h.3d3rqxzgrl5g)

[Figura 7. Página para descargar SQL Server 14](#_heading=h.al8drue2lwko)

[Figura 8. Seleccionador del tipo de instalación 15](#_heading=h.ldj2s8h8qsdb)

[Figura 9. Mensaje de recomendación cambiar al inglés 15](#_heading=h.q2bn0kt40era)

[Figura 10. Estructura de las carpetas del proyecto. 15](#_heading=h.qb9pevfbusp6)

[Figura 11. Descarga del paquete SQL Server. 16](#_heading=h.wm9erhueae0a)

[Figura 12. Aplicación principal de SQL Server Installation Server 16](#_heading=h.azd4dtcojylw)

[Figura 13. Aceptar términos y condiciones. 16](#_heading=h.sntswnyeuyyc)

[Figura 14. Selección de actualizaciones automáticas. 17](#_heading=h.sg8pg0nd9ihy)

[Figura 15. Resumen de instalación. 17](#_heading=h.wviwx56ms0ip)

[Figura 16. Desmarcar instalación de Azure y cancelar la instalación 18](#_heading=h.w1gggigkhx64)

[Figura 17. Selección de características a instalar desde SQL Server 18](#_heading=h.zh1kfj8uvi49)

[Figura 18. Configuración de instancias 19](#_heading=h.13z460l3qh2s)

[Figura 19. Configuración del servidor 19](#_heading=h.hlcf6bhdcj85)

[Figura 20. Configuración de credenciales 20](#_heading=h.6x5ivxhnkums)

[Figura 21. Creación de carpeta de ubicación de datos. 20](#_heading=h.9s9det7nr41z)

[Figura 22. Selección de carpeta para almacenamiento de base de datos. 21](#_heading=h.y7d6wnqt0mdt)

[Figura 23. Resumen de instalación. 21](#_heading=h.fnt20xuqv1j3)

[Figura 24. Página de descarga de SQL Server Management Studio 21 22](#_heading=h.eiqa0ksw593e)

[Figura 25. Selección de la ubicación de instalación 22](#_heading=h.6m119ew10fej)

[Figura 26. Instalación completa de SQL Server Management Studio 21 23](#_heading=h.sjebfg6vfm9b)

[Figura 27. Conexión al gestor 24](#_heading=h.sy2qlc33xxe7)

[Figura 28. Explorador de soluciones del gestor 24](#_heading=h.xhgnaswxusfx)

[Figura 29. Creación de un nuevo query 25](#_heading=h.b2xrk5yx4ksi)

[Figura 30. Creación de base de datos 25](#_heading=h.rc2d54qq8brs)

[Figura 31. Creación de un proyecto de servicio WCF en Visual Studio 26](#_heading=h.ciqd8vjodjpk)

[Figura 32. Estructura Comercializadora 28](#_heading=h.xzyvukyiw7rg)

[Figura 33. Consulta de Factura por Número 63](#_heading=h.ssefx19psj00)

[Figura 34. Tabla de Amortización del Crédito 63](#_heading=h.khc8flxrpm0o)

[Figura 35. Gestión de Productos 64](#_heading=h.xb7xgr8rf0l)

[Figura 36. Confirmación de Venta Completada 65](#_heading=h.80nvfab2ya48)

[Figura 37. Página Principal del Sistema 65](#_heading=h.mnyqzhy0q9n9)

[Figura 38. Consulta de Facturas por Cliente 66](#_heading=h.dramat8fjh4i)

[Figura 39. Formulario de Nueva Venta a Crédito 66](#_heading=h.htdpvl78zams)

[Figura 40. Menú Principal de la Aplicación de Consola 68](#_heading=h.t05ecqev5dv8)

[Figura 41. Menú de Gestión de Productos 69](#_heading=h.khsafk4bzhf)

[Figura 42. Listado Completo de Productos 70](#_heading=h.f80eyvyb17he)

[Figura 43. Búsqueda de Producto por ID 71](#_heading=h.tz9i8c10p36l)

[Figura 44. Búsqueda de Productos por Categoría 71](#_heading=h.sdnpqwksj8zd)

[Figura 45. Creación de Nuevo Producto 72](#_heading=h.rska0xlccc74)

[Figura 46. Actualización de Producto Existente 72](#_heading=h.ir9irk3eqxe)

[Figura 47. Menú de Facturación 73](#_heading=h.vwc8x0150ff2)

[Figura 48. Proceso de Venta en Efectivo con Descuento 74](#_heading=h.i8ktnk37nbjv)

[Figura 49. Error en Venta a Crédito - Cliente con Crédito Activo 75](#_heading=h.9k945osdjbut)

[Figura 50. Consulta de Factura por Número 75](#_heading=h.wck6kquj4b3r)

[Figura 51. Consulta de Facturas por Cliente 76](#_heading=h.nklinu4aarou)

[Figura 52. Consulta de Facturas por Rango de Precio 77](#_heading=h.lmxsclzccbwi)

[Figura 53. Consulta de Facturas de Ventas a Crédito 77](#_heading=h.8fmj0bnh0khk)

[Figura 54. Tabla de amortización 78](#_heading=h.sefthokg5xm1)

[Figura 63. Consulta de Factura por Número 80](#_heading=h.4dxbi8sc3nb0)

[Figura 64. Tabla de Amortización del Crédito 80](#_heading=h.p5vupz1evc90)

[Figura 65. Gestión de Productos 81](#_heading=h.yfdtrdn1lks6)

[Figura 66. Confirmación de Venta Completada 81](#_heading=h.8dpzr0dkb9g0)

[Figura 67. Página Principal del Sistema 82](#_heading=h.skbl3l37fse)

[Figura 68. Consulta de Facturas por Cliente 83](#_heading=h.a1hj37hwryx4)

[Figura 69. Formulario de Nueva Venta a Crédito 83](#_heading=h.g29igv2j11fy)

[Figura 70. Pantalla Principal y Configuración 85](#_heading=h.hlduv6d6v1ks)

[Figura 71. Nueva Venta - Paso 1: Cliente Validado 86](#_heading=h.ilt7dem3tj3s)

[Figura 72. Nueva Venta - Paso 2: Carrito Confirmado 87](#_heading=h.xt8z43iof4ai)

[Figura 73. Consultas Generales - Factura por Número 88](#_heading=h.s4nntpljjqr0)

[Figura 74. Gestión de Productos Móvil 89](#_heading=h.6c3026k6o0k)

[Figura 75. Lista de Productos en Tarjetas 90](#_heading=h.9bpowvzfhz02)

[Figura 76. Consultas - Facturas por Cliente 91](#_heading=h.wkz1opfxf1tf)

[Figura 77. Nueva Venta - Paso 3: Venta Completada con Crédito 92](#_heading=h.lskf00rul0k9)

[Figura 78. Detalles de Compra y Totales 93](#_heading=h.69jhyzggn7l6)

[Figura 79. Consulta de Tabla de Amortización Móvil 94](#_heading=h.ul8r5gbrffl1)

ÍNDICE DE TABLAS

[Tabla 1. Tecnologías utilizadas en el proyecto 12](#_heading=h.wqn6yvwxqdum)

[Tabla 2. Cadenas de conexión para los servicios 29](#_heading=h.f2kzj43r2hrl)

[Tabla 3. Ejemplo de script T-SQL para la creación e inserción de datos 29](#_heading=h.uof26og2rwnl)

# **1. INTRODUCCIÓN**

El presente documento describe el desarrollo e implementación de un sistema distribuido para la comercialización de electrodomésticos con financiamiento bancario. El sistema está compuesto por dos aplicaciones independientes que se comunican mediante servicios bajo el protocolo SOAP:

1. **BanQuito Server**: Sistema bancario que gestiona clientes, cuentas y créditos
2. **Comercializadora Server**: Sistema de ventas que gestiona productos y facturación.

## **1.1. Contexto del Problema**

El Banco BanQuito ha establecido una alianza estratégica con una compañía comercializadora de electrodomésticos con el objetivo de proveer una línea de crédito directo a sus clientes, facilitando la adquisición de productos de alto valor. Para materializar esta alianza, se requiere una solución tecnológica que conecte de manera fluida y automatizada el sistema de ventas de la comercializadora con el módulo de créditos del banco.

El problema central es la ausencia de un sistema integrado que permita verificar la elegibilidad crediticia de un cliente en tiempo real durante el proceso de compra. Sin esta automatización, el proceso sería manual, lento, propenso a errores y resultaría en una experiencia de cliente deficiente, limitando el éxito de la alianza estratégica y desaprovechando oportunidades de venta

# **2. OBJETIVOS**

## **2.1. Objetivo General**

Desarrollar una solución de software integral basada en una Arquitectura Orientada a Servicios (SOA) que conecte el sistema de ventas de una comercializadora de electrodomésticos con el módulo de créditos del Banco BanQuito, con el fin de automatizar y agilizar el proceso de adquisición de productos financiados.

## **2.2. Objetivos Específicos**

* **Implementar el servicio web del Módulo de Crédito (Banquito)**, capaz de validar si un cliente es sujeto de crédito, calcular su monto máximo autorizado y procesar el otorgamiento de préstamos con su respectiva tabla de amortización.
* **Desarrollar el servicio web del Sistema de la Comercializadora**, que gestione el catálogo de productos y el proceso de facturación para las dos modalidades de pago requeridas: Efectivo y Crédito Directo.
* **Establecer una comunicación robusta entre ambos servicios utilizando tecnología SOAP**, permitiendo que el sistema de la comercializadora invoque los servicios del banco en tiempo real para la toma de decisiones durante la venta.
* **Diseñar y crear esquemas de bases de datos independientes** (banquito\_db y comercializadora\_db) para cada servicio, garantizando el desacoplamiento y la autonomía de cada dominio de negocio conforme a los principios de SOA.
* **Demostrar la funcionalidad completa** a través de una aplicación cliente que orquesta el flujo de venta a crédito, consumiendo los servicios de ambos sistemas de manera secuencial y lógica.

# **3. ALCANCE DEL PROYECTO**

## **3.1. Funcionalidades Implementadas**

El proyecto abarca el desarrollo de dos sistemas de backend independientes pero interconectados, cada uno con un conjunto específico de funcionalidades.

* **Gestión de Clientes:** Almacenamiento de la información personal de los clientes del banco.
* **Gestión de Transacciones:** Registro de depósitos y retiros, que sirven como base para el análisis crediticio.
* **Gestión de Créditos:**
  + **Validar Sujeto de Crédito:** Expone un servicio web que verifica si un cliente cumple con las 4 reglas de negocio definidas:
    1. Ser cliente activo del banco.
    2. Tener al menos un depósito en el último mes.
    3. Ser mayor de 25 años si su estado civil es "Casado".
    4. No tener un crédito activo previamente.
  + **Obtener Monto Máximo:** Provee un servicio web que calcula la capacidad máxima de endeudamiento de un cliente basado en la fórmula: ((Promedio Depósitos 3 meses – Promedio Retiros 3 meses) \* 60%) \* 9.
  + **Otorgar Crédito:** Expone un servicio que formaliza un préstamo con una tasa de interés del 16% anual y un plazo de 3 a 24 meses, registrando el crédito y generando su tabla de amortización.
  + **Obtener Tabla de Amortización:** Permite consultar el plan de pagos de un crédito ya otorgado.
* **Gestión de Productos:**
  + Funcionalidad completa para el mantenimiento (CRUD) del catálogo de electrodomésticos disponibles para la venta.
  + Control de stock para cada producto.
* **Gestión de Facturación:**
  + **Facturación en Efectivo:** Genera una factura aplicando un descuento automático del 33% sobre el precio de venta.
  + **Facturación con Crédito Directo:** Gestiona el proceso de facturación para compras financiadas, interactuando con el sistema BanQuito para la aprobación.
  + **Consulta de Facturas:** Permite buscar y visualizar facturas previamente emitidas.

## **3.2. Integraciones**

La integración es el núcleo funcional del proyecto, permitiendo que los dos sistemas colaboren para ejecutar el flujo de venta a crédito.

1. **Inicio en Comercializadora:** El cliente selecciona los productos que desea comprar a través de una aplicación que consume los servicios de la Comercializadora.
2. **Cálculo del Monto a Financiar:** La aplicación solicita al servicio Comercializadora que calcule el total de la compra (sin descuento).
3. **Consulta a BanQuito:** La aplicación invoca los servicios del sistema Banquito para:  
   a. Validar si el cliente es sujeto de crédito.  
   b. Si es válido, obtener el monto máximo de crédito que se le puede aprobar.
4. **Validación del Préstamo:** La aplicación cliente compara el total de la compra con el monto máximo aprobado. Si el monto de la compra es menor o igual, el proceso continúa.
5. **Otorgamiento del Crédito:** La aplicación solicita al servicio Banquito que otorgue un crédito por el monto exacto de la compra. Banquito procesa la solicitud y devuelve un número de crédito único.
6. **Finalización en Comercializadora:** La aplicación invoca nuevamente al servicio Comercializadora para generar la factura final, especificando la forma de pago como "CREDITO" e incluyendo el número de crédito proporcionado por Banquito.

**Nota:** Los sistemas son **independientes**. El cliente (o aplicación intermedia) es responsable de orquestar las llamadas.

# **4. ARQUITECTURA DEL SISTEMA**

## **4.1. Arquitectura General**

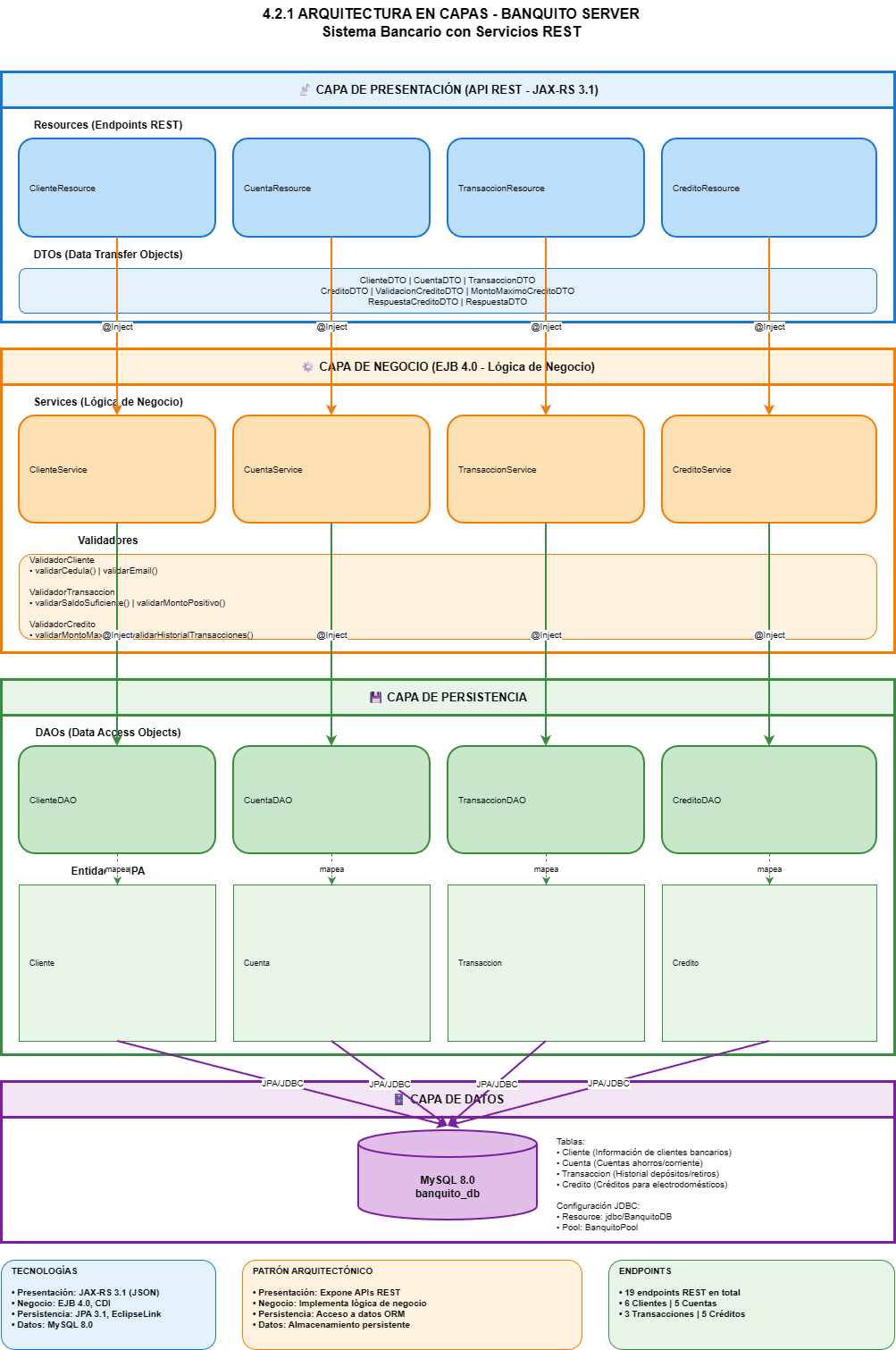
## 

##### ***Figura 1.*** *Arquitectura de Software del Sistema Distribuido de Comercialización de Electrodomésticos*

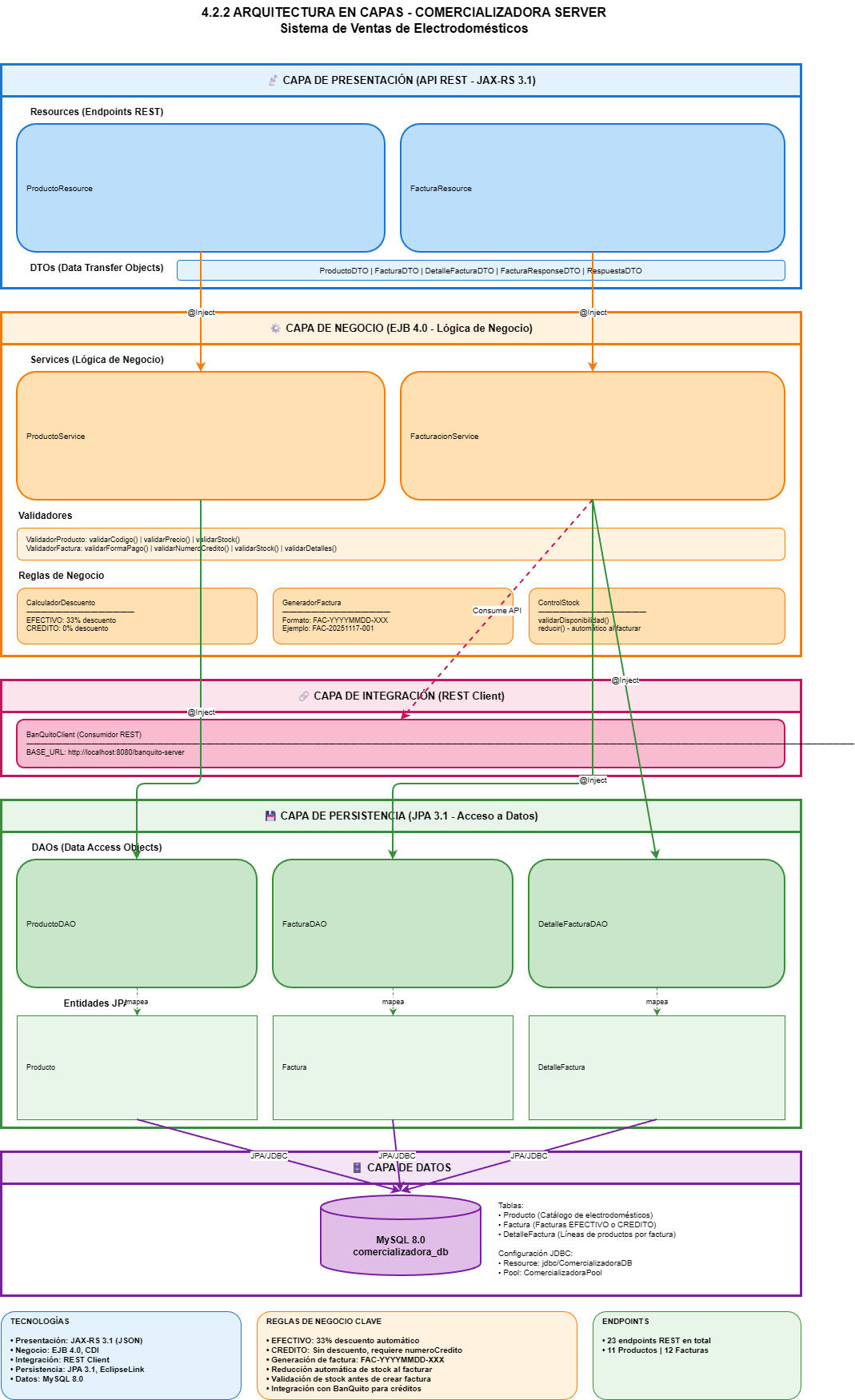
El sistema se ha diseñado siguiendo el paradigma de **Arquitectura Orientada a Servicios (SOA)**, que se enfoca en la creación de componentes de software modulares, reutilizables y desacoplados. La estructura se divide en tres capas lógicas principales, como se muestra en el diagrama de arquitectura:

* **Capa de Presentación (Usuarios):**
  + Compuesta por las aplicaciones cliente (Consola, Escritorio, Web, Móvil) que actúan como consumidores de los servicios de negocio. Esta capa es responsable de la interacción con el usuario y de orquestar las llamadas a los servicios para implementar los flujos de negocio.
* **Capa de Servicios (Lógica de Negocio):**
  + Es el corazón de la arquitectura. Contiene dos servicios web SOAP completamente autónomos:
    - **Servicio Banquito:** Encapsula y expone toda la lógica de negocio del dominio financiero y crediticio.
    - **Servicio Comercializadora:** Encapsula la lógica de negocio relacionada con la gestión de productos y la facturación.
  + Cada servicio se despliega de forma independiente y se comunica a través de contratos estandarizados (WSDL), lo que garantiza la interoperabilidad y facilita el mantenimiento.
* **Capa de Datos (BDD):**
  + Implementa el principio de **"base de datos por servicio"** para asegurar el máximo desacoplamiento y autonomía:
    - **banquito\_db:** Base de datos SQL Server dedicada exclusivamente a persistir los datos del sistema bancario.
    - **comercializadora\_db:** Base de datos SQL Server que almacena los datos del sistema de la comercializadora.
  + Esta separación evita el acoplamiento a nivel de base de datos, una práctica fundamental en arquitecturas distribuidas modernas.

## **4.2. Arquitectura en Capas**



##### ***Figura 2.*** *Arquitectura en capas del Sistema BanQuito*

***Figura 3.*** *Arquitectura en capas del Sistema de Electrodomésticos*

## **4.3. Tecnologías Utilizadas**

| Categoría | Tecnología | Versión | Uso |
| --- | --- | --- | --- |
| **Lenguaje** | C# (C Sharp) | 12 | Desarrollo de la lógica de negocio para los servicios backend y la aplicación cliente. |
| **Framework** | .NET Framework / .NET | 8.0 | Backend: Creación de los servicios web SOAP con WCF. <br> Cliente: Construcción de la aplicación de consola. |
| **Servidor** | IIS / IIS Express | 10.0 | Alojamiento y exposición de los servicios web SOAP (WCF) para su consumo. |
| **ORM** | Entity Framework | 6.5.1 | Mapeo Objeto-Relacional para la interacción entre el código C# y la base de datos SQL Server. |
| **Base de Datos** | Microsoft SQL Server | 2022 | Almacenamiento y persistencia de todos los datos de los sistemas BanQuito y Comercializadora. |
| **Build Tool** | MSBuild | Integrado | Compilación del código fuente, gestión de dependencias y empaquetado de las aplicaciones. |
| **IDE** | Microsoft Visual Studio | 2022 | Entorno de Desarrollo Integrado para escribir, compilar, depurar y desplegar la solución completa. |

###### Tabla 1. Tecnologías utilizadas en el proyecto

# **5. SISTEMA GENERAL COMERCIALIZADORA DE** ELECTRODOMÉSTICOS

## **5.1. Descripción General**

El sistema general es una solución integral que materializa la alianza estratégica entre la Comercializadora de Electrodomésticos y el Banco BanQuito. Está diseñado como un ecosistema de servicios interconectados donde cada componente tiene una responsabilidad de negocio claramente definida.

El Sistema de la Comercializadora actúa como el front-end del negocio. Su responsabilidad es gestionar el catálogo de productos, interactuar con el cliente durante el proceso de compra y generar la facturación final. Este sistema soporta dos modalidades de pago:

* **Pago en Efectivo:** Aplica un descuento directo del 33%.
* **Pago con Crédito Directo:** Activa el flujo de integración con el banco para financiar la compra.

El Módulo de Crédito de BanQuito funciona como el back-end financiero. Su rol es proveer los servicios necesarios para analizar, aprobar y gestionar los créditos solicitados por los clientes de la comercializadora. Es una entidad de negocio separada que expone sus capacidades a través de una API de servicio web SOAP.

En conjunto, ambos sistemas colaboran para ofrecer una experiencia de compra fluida donde un cliente puede seleccionar un producto, solicitar financiamiento, ser aprobado y recibir su factura en un único proceso automatizado, orquestado por la aplicación cliente.

## **5.2. Modelo de Datos - BanQuito**

La base de datos del sistema BanQuito, banquito\_db, fue diseñada para soportar tanto las operaciones bancarias centrales como las funcionalidades específicas del nuevo Módulo de Crédito. El esquema se compone de las siguientes entidades principales:

* **Cliente:** Entidad central que almacena la información personal y demográfica de los clientes del banco. Es el punto de anclaje para todas las demás entidades.
* **Cuenta:** Representa las cuentas bancarias (Ahorros, Corriente) que pertenecen a un cliente.
* **Movimiento:** Registra todas las transacciones (depósitos y retiros) asociadas a una cuenta. Esta tabla es fundamental para el análisis de la capacidad de pago del cliente.
* **Credito:** Nueva entidad que almacena toda la información de un préstamo otorgado a un cliente, incluyendo el monto, la tasa de interés, el número de cuotas y el estado actual.
* **CuotaAmortizacion:** Entidad que detalla el plan de pagos de un crédito específico. Cada registro representa una cuota mensual, desglosando el capital, el interés y el saldo pendiente.

El siguiente diagrama Entidad-Relación ilustra la estructura de la base de datos banquito\_db. Se visualizan las tablas, sus respectivos campos, y las relaciones establecidas a través de claves primarias (PK) y foráneas (FK).

Se destaca cómo las nuevas tablas Credito y CuotaAmortizacion se integran con el modelo de datos CORE a través de una relación directa con la tabla Cliente, cumpliendo con el requisito de extender el sistema existente sin alterarlo fundamentalmente. Las relaciones clave son:

* Un Cliente puede tener una o varias Cuentas.
* Una Cuenta puede tener muchos Movimientos.
* Un Cliente puede tener uno o varios Creditos.
* Un Credito se compone de muchas CuotasAmortizacion.

## 5.3. Modelo de Datos - Comercializadora

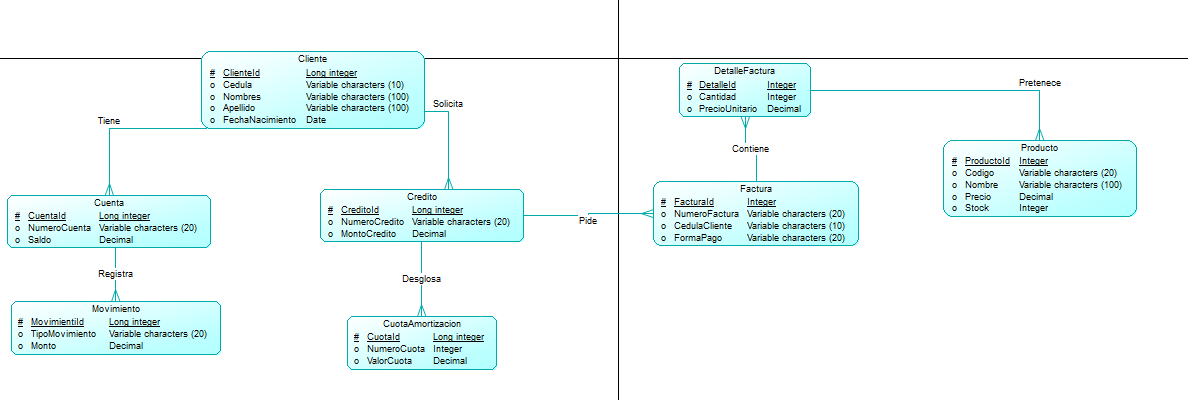
La base de datos del sistema Comercializadora, comercializadora\_db, está diseñada específicamente para gestionar el catálogo de productos y el ciclo de vida completo de la facturación. Su estructura es simple pero robusta, enfocada en las necesidades transaccionales de un punto de venta. Las entidades principales son:

* Producto: Almacena toda la información del inventario de electrodomésticos disponibles para la venta. Incluye campos como el código, nombre, descripción, precio y, fundamentalmente, el Stock, que se actualiza con cada venta.
* Factura: Registra la cabecera de cada transacción de venta. Contiene los datos del cliente (CedulaCliente, NombreCliente), la FormaPago, los totales (Subtotal, Descuento, Total) y el campo NumeroCredito, que vincula la factura con el préstamo otorgado por BanQuito en caso de ser una venta financiada.
* DetalleFactura: Actúa como una tabla de enlace que resuelve la relación muchos a muchos entre Factura y Producto. Cada registro en esta tabla representa una línea de producto dentro de una factura, especificando la Cantidad y el PrecioUnitario al momento de la venta.

### **5.2.1. Diagrama Entidad-Relación**

El siguiente diagrama presenta el modelo de datos completo del sistema general, consolidando los esquemas de las bases de datos de BanQuito y de la Comercializadora. Se evidencia la separación de dominios: por un lado, las entidades financieras y de clientes del banco; y por otro, las entidades de inventario y facturación del comercio.

Es importante destacar que no existen relaciones directas (claves foráneas) entre las tablas de ambas bases de datos. La integración se realiza a nivel de la capa de servicios, donde el campo NumeroCredito en la tabla Factura actúa como un identificador lógico que vincula una venta con un préstamo, pero no como una restricción de integridad referencial a nivel de base de datos.

*****Figura 4.*** *Modelo entidad relación del sistema general*

# 6. PARTE PRÁCTICA

Para la realización de la parte práctica de este proyecto, es necesario realizar los siguientes pasos detallados a continuación y adaptarlo a las necesidades de cada individuo.

Las herramientas necesarias para desarrollar esta práctica son los entornos de desarrollo SQL Server, Visual Studio 2026 y Android Studio.

## 6.1. Instalación de herramientas

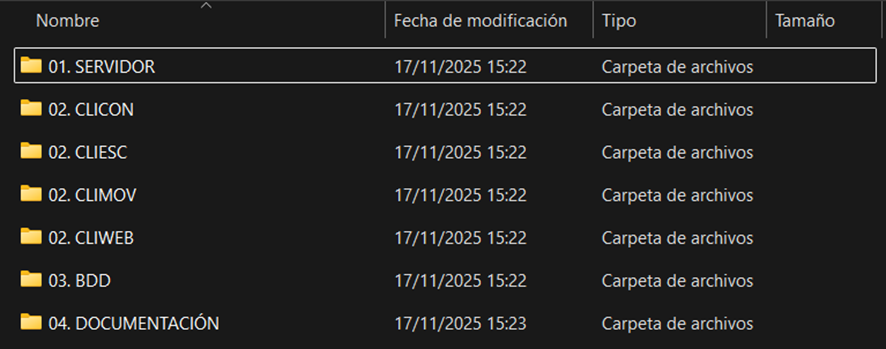
### 6.1.1. Estructura de proyecto

Para realizar el proyecto, primero se debe construir una carpeta donde se almacene los archivos necesarios para su funcionamiento.



##### ***Figura 5.*** Creación de la carpeta principal del examen

Dentro de la carpeta principal se ubica la siguiente subestructura.



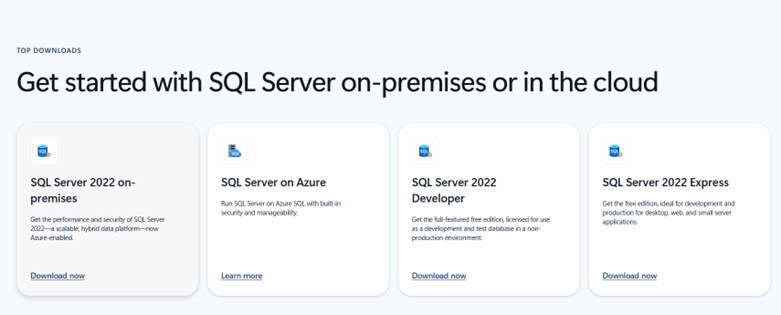
##### Figura 6. Estructura de las carpetas del proyecto

### **6****.1.2. Instalación y configuración de herramientas**

Para comenzar el desarrollo de la práctica, primero se deben instalar y configurar las herramientas necesarias.

### **6****.1.3. Instalación y configuración de microsoft SQL Server**

Para iniciar con la instalación del SQL Server, primero se debe descargar en el siguiente enlace: https://www.microsoft.com/en-us/sql-server/sql-server-downloads y se procede a descargar la opción de Express



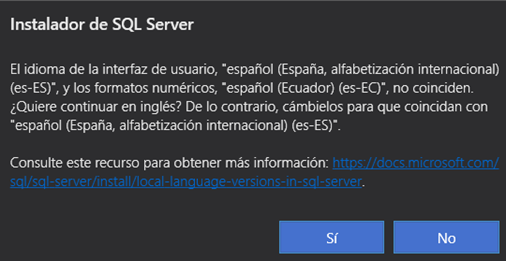
##### Figura 7. Página para descargar SQL Server

Una vez que se tenga descargado se ejecuta el instalador el cual iniciará con la siguiente pantalla.



##### Figura 8. Seleccionador del tipo de instalación

En la pantalla de selección de instalación se le dará clic en personalizada. Se recomienda instalación en inglés. Si en el caso de que se tenga el idioma español, saldrá un mensaje que se recomienda cambiar al inglés.



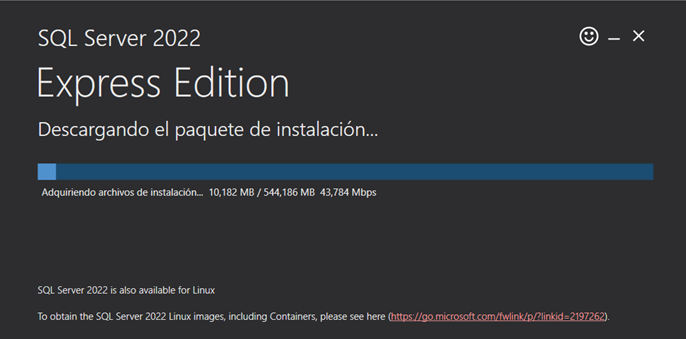
##### Figura 9. Mensaje de recomendación cambiar al inglés

En la siguiente pantalla, es necesario configurar opciones como el idioma y la ubicación, y luego hacer clic en "Instalar”.



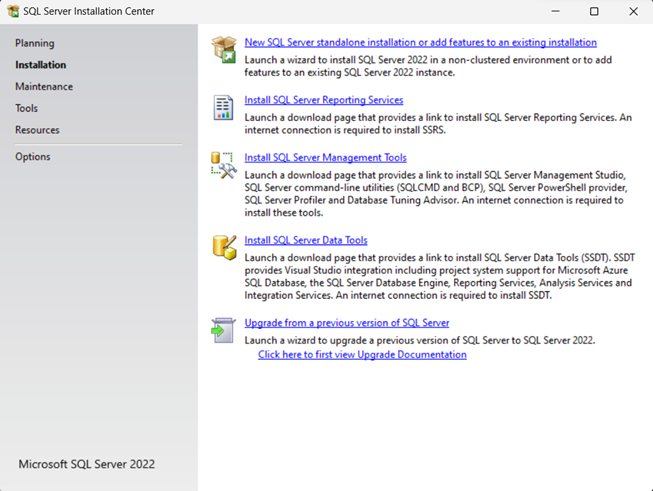
##### Figura 10. Estructura de las carpetas del proyecto.

Una vez hecho clic en instalar se mostrará la siguiente pantalla donde se indica que se está descargando lo necesario para la instalación.



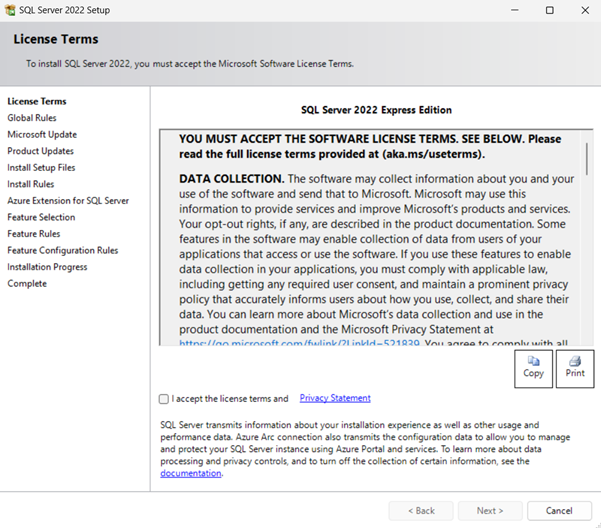
##### Figura 11. Descarga del paquete SQL Server.

Una vez finalizada la descarga se tiene la siguiente aplicación la cual es “SQL Server Installation Center” desde la cual se va a proceder con la instalación y configuración de las herramientas.



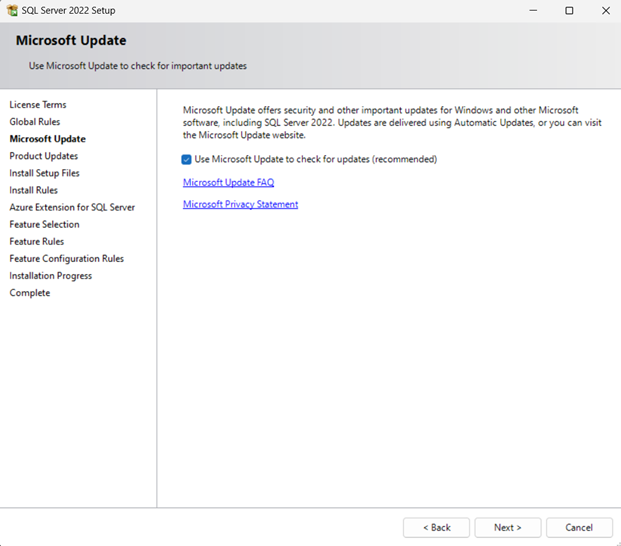
##### Figura 12. Aplicación principal de SQL Server Installation Server

Se deberá seleccionar la opción “Install SQL Server Reporting Services” y aceptar los términos y condiciones.



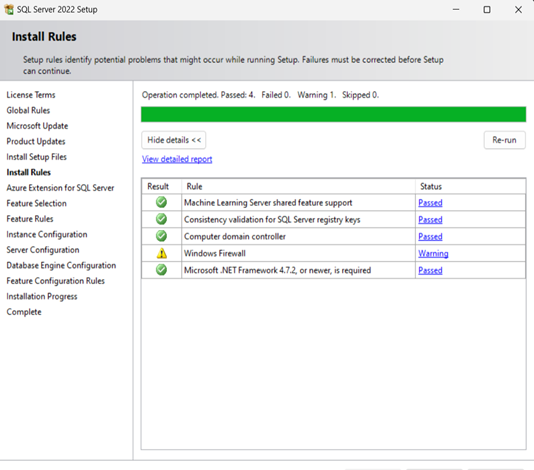
##### Figura 13. Aceptar términos y condiciones.

Se marca la opción de obtener actualizaciones por parte de Microsoft.



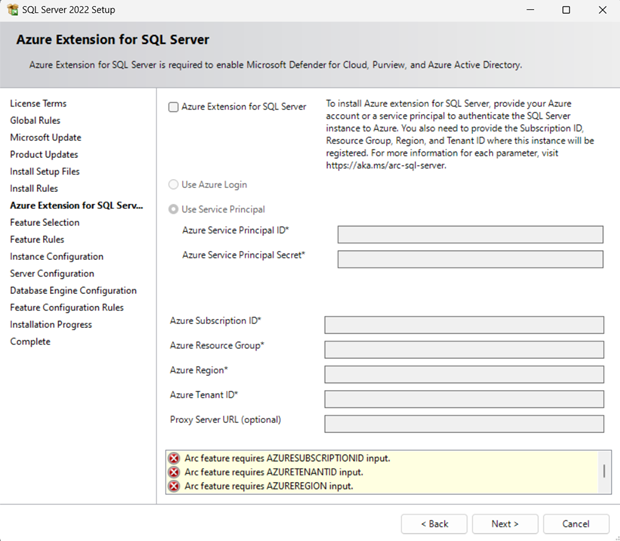
##### Figura 14. Selección de actualizaciones automáticas.

Seguido de eso se mostrará las reglas de instalación y se procede hacer clic en siguiente.



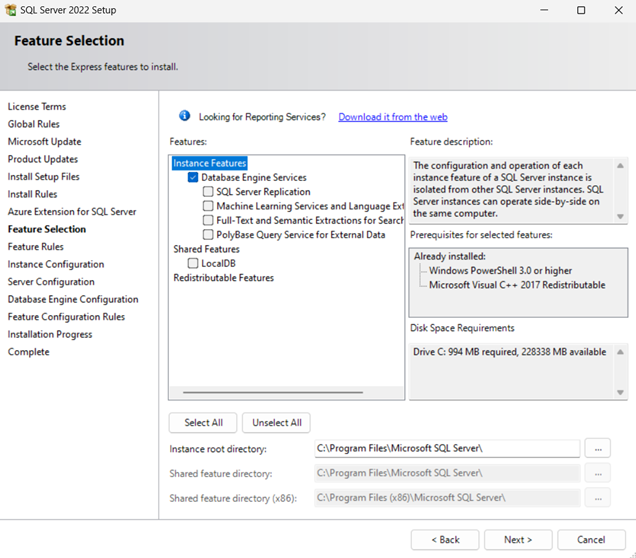
##### Figura 15. Resumen de instalación.

Se procede a desmarcar la opción de Azure Extension for SQL Server ya que no es necesario para el desarrollo de la práctica.



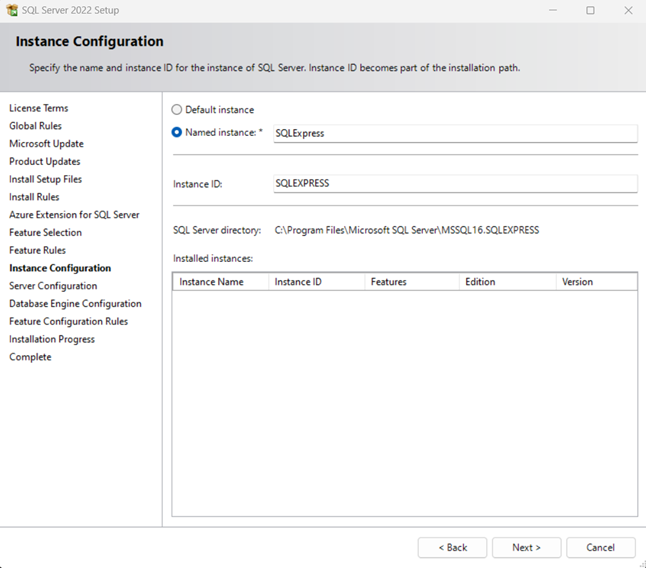
##### Figura 16. Desmarcar instalación de Azure y cancelar la instalación

En la siguiente pantalla se deberá realizar la selección de características que se requieren en la instalación, lo cual depende del tipo de práctica, por lo tanto, únicamente se seleccionará Database Engine Services.



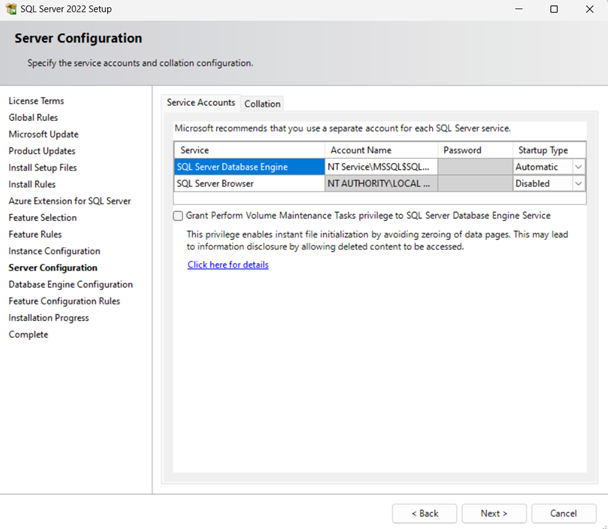
##### Figura 17. Selección de características a instalar desde SQL Server

En la siguiente pantalla se deberá realizar la configuración de instancias.



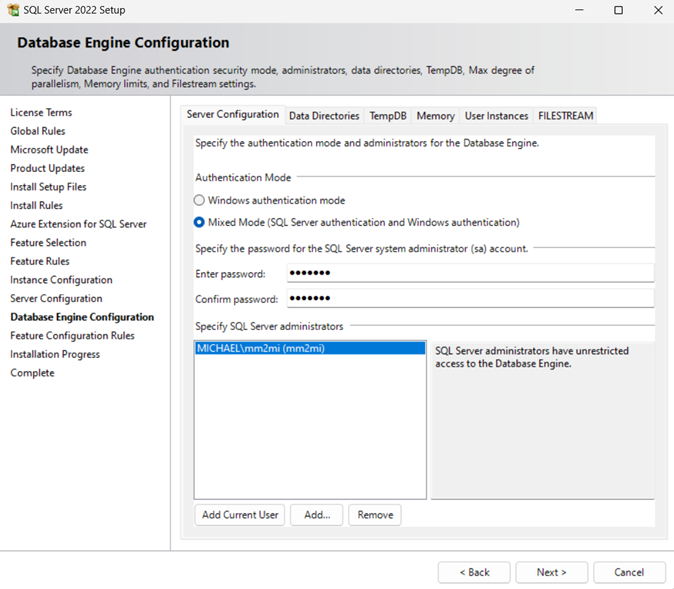
##### Figura 18. Configuración de instancias

Una vez configurado las instancias, se deberá realizar la configuración del servidor para lo cual se seleccionará el tipo de inicialización de los servicios.



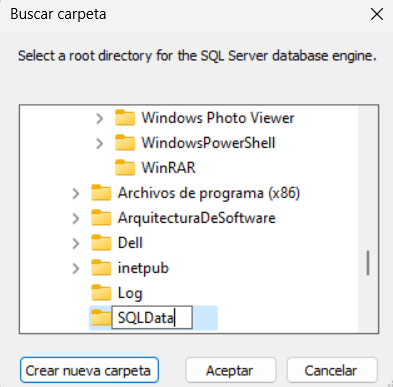
##### Figura 19. Configuración del servidor

Se procede a realizar la configuración del tipo de autenticación, y credenciales de inicio.

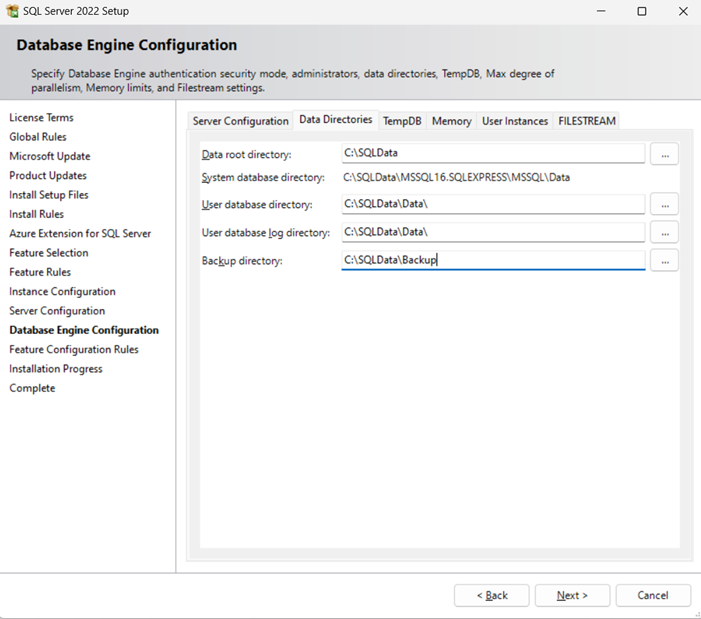


##### Figura 20. Configuración de credenciales

En el siguiente apartado se deberá configurar la dirección de las bases de datos, del backup, logs y demás para lo cual se deberá crear una nueva carpeta en la raíz del disco local C.



##### Figura 21. Creación de carpeta de ubicación de datos.



##### Figura 22. Selección de carpeta para almacenamiento de base de datos.

Finalmente se procede a realizar clic en siguiente lo cual comenzará con la instalación de la herramienta.



##### Figura 23. Resumen de instalación.

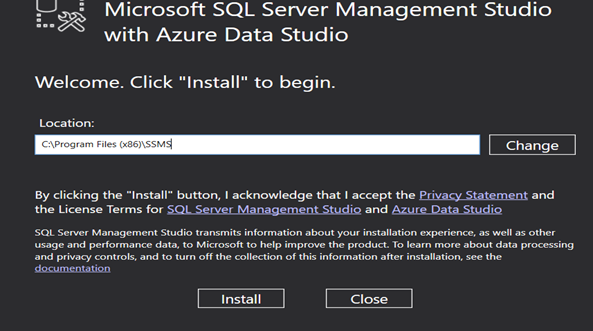
### **6****.1.4. Instalación y configuración de SQL Server Management Studio 21**

Para comenzar con la instalación y configuración de la herramienta SQL Server Management Studio 21 se procede a ingresar al siguiente enlace: https://learn.microsoft.com/es-es/ssms/install/install . Se da clic en el enlace de “Descargue SSMS 22”



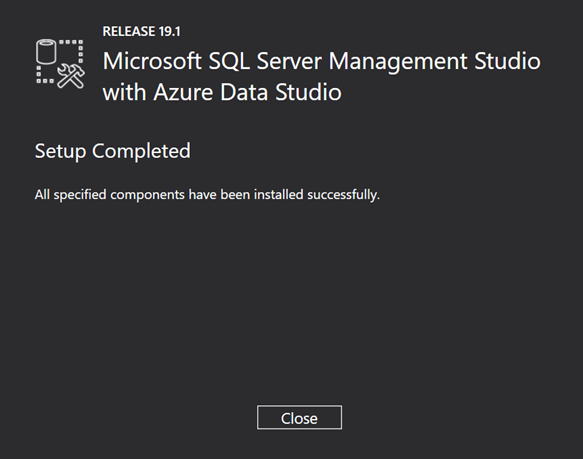
##### Figura 24. Página de descarga de SQL Server Management Studio 21

Una vez finalizada la descarga se procede a la instalación de la herramienta, donde como primer punto se deberá seleccionar la ruta de instalación y después dar clic en “instalall”.



##### Figura 25. Selección de la ubicación de instalación

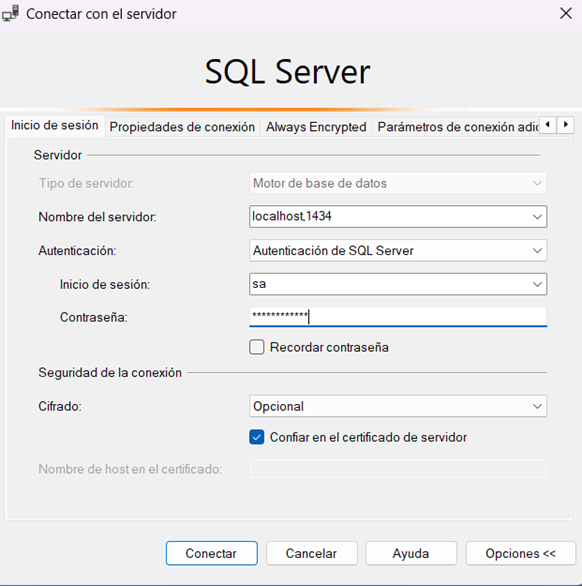
Una vez finalizada la instalación se mostrará la siguiente pantalla.



##### Figura 26. Instalación completa de SQL Server Management Studio 21

### **6****.1.5. Creación de la base de datos**

Para comenzar con el desarrollo de la práctica se procede a realizar la creación de la base de datos, para lo cual se deberá ingresar a la herramienta instalada anteriormente donde se pedirá el tipo de autenticación.



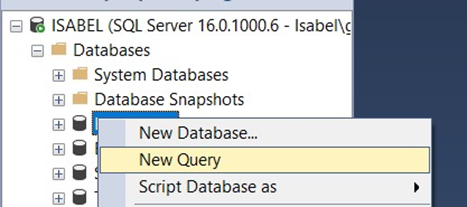
##### Figura 27. Conexión al gestor

Se tiene la siguiente estructura inicial donde se mostrarán las bases de datos creadas o existentes.



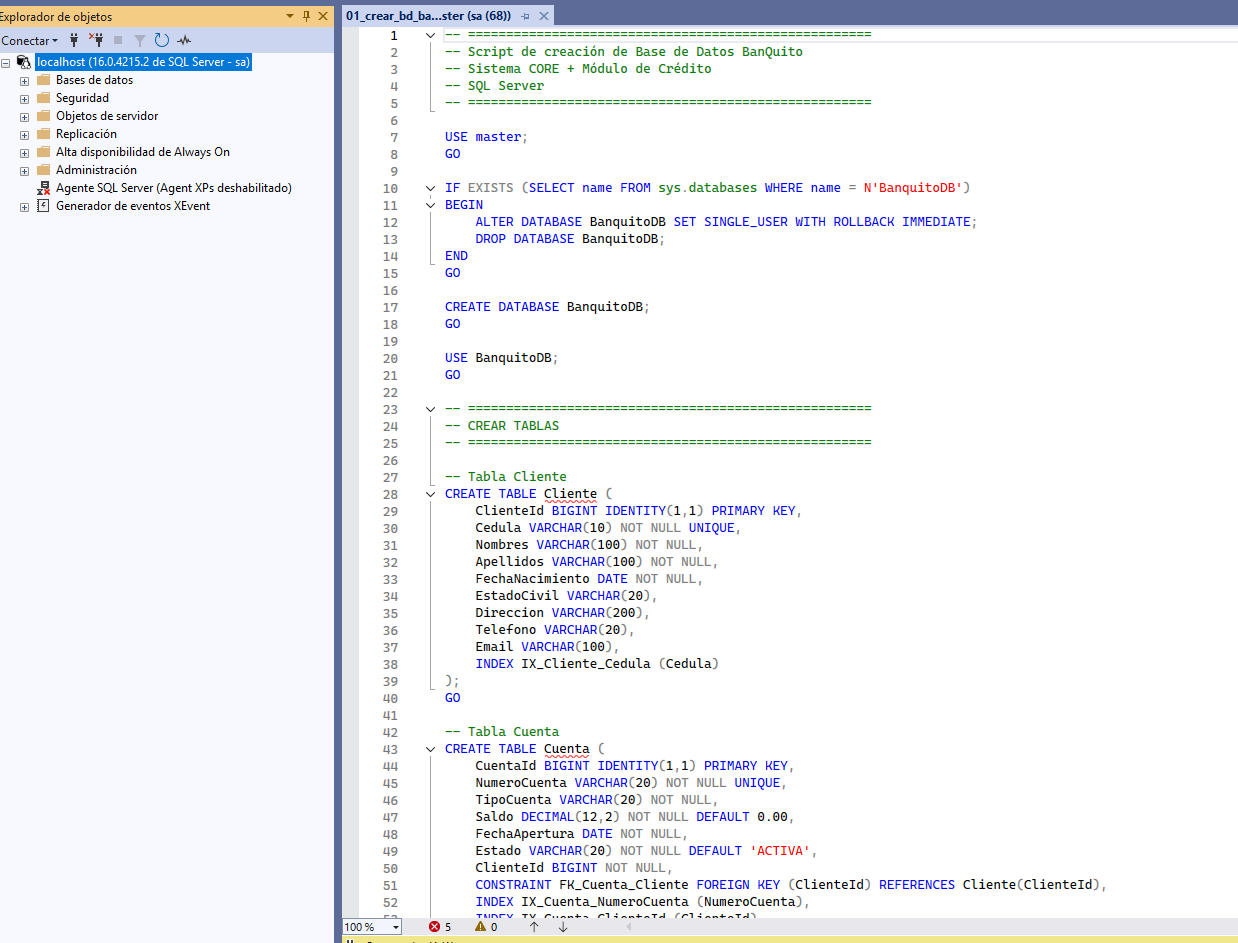
##### Figura 28. Explorador de soluciones del gestor

Para crear la base de datos se deberá hacer clic derecho sobre el espacio de trabajo, y seleccionar New Query.



##### Figura 29. Creación de un nuevo query

Se procede a registrar el query para la creación de la base de datos y se procede a ejecutar el query.33



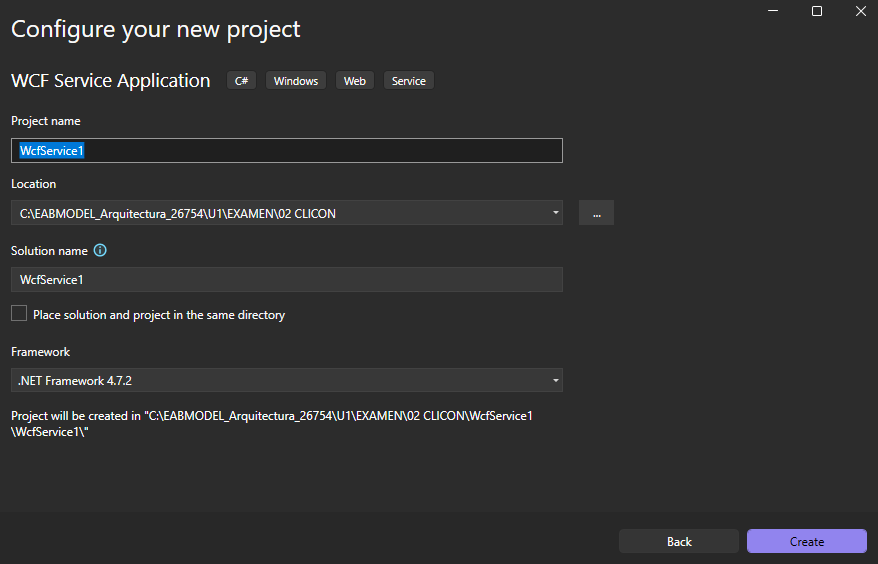
##### Figura 30. Creación de base de datos

### **6****.1.6. Aplicación Servidor**

CREACIÓN DEL PROYECTO SERVIDOR

Para comenzar con la creación de los proyectos de servidor (Banquito y Comercializadora), se utilizó Microsoft Visual Studio 2022. El tipo de proyecto seleccionado para ambos casos fue una Aplicación de servicio WCF de ASP.NET (.NET Framework).

Esta plantilla proporciona la estructura base para desarrollar y exponer servicios web basados en el protocolo SOAP, lo cual es un requisito fundamental de la arquitectura del sistema. La configuración inicial crea los archivos .svc que actúan como los endpoints públicos del servicio y las interfaces de contrato (IService) que definen las operaciones expuestas.



##### Figura 31. Creación de un proyecto de servicio WCF en Visual Studio

El servicio web spring initializr permitirá descargar un comprimido el cual se podrá abrir con un entorno de desarrollo para java y Maven compatible.

### **6****.1.7. Estructura del proyecto**

A diferencia del patrón MVC (Modelo-Vista-Controlador), que está orientado a aplicaciones con interfaces de usuario, la estructura del proyecto se organiza siguiendo una arquitectura en N-Capas (Layered Architecture), ideal para la creación de servicios de backend robustos y mantenibles.

Esta arquitectura separa las responsabilidades del sistema en las siguientes capas lógicas, que se corresponden directamente con la organización de carpetas en el proyecto:

**Capa de Servicio o Presentación (Carpeta WS):**

Contiene los archivos de servicio (.svc) y sus implementaciones (.svc.cs). Actúa como la puerta de entrada al sistema, exponiendo las operaciones definidas en los contratos WSDL a las aplicaciones cliente.

**Capa de Lógica de Negocio (Carpeta Services o BusinessLogic):**

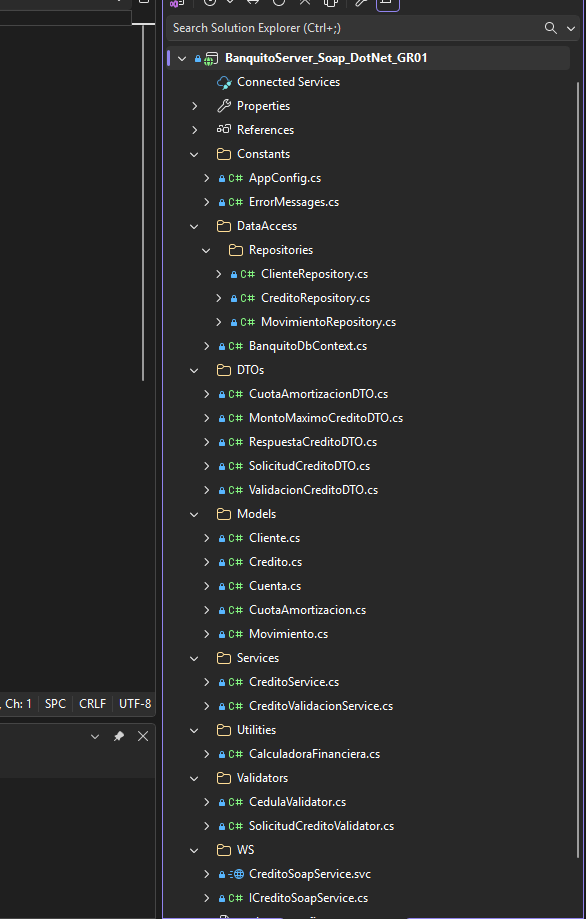
Encapsula todas las reglas de negocio y la lógica de la aplicación. Aquí residen clases como CreditoService y FacturacionService, que orquestan las operaciones y toman decisiones.

**Capa de Acceso a Datos (Carpeta DataAccess):**

Gestiona toda la comunicación con la base de datos. Se implementa utilizando el patrón Repository (ej. ClienteRepository, ProductoRepository) sobre el ORM Entity Framework, abstrayendo la lógica de las consultas SQL.

**Capa de Modelos y DTOs (Carpetas Models y DTOs):**

Define las estructuras de datos del proyecto. La carpeta Models contiene las clases de entidad que mapean directamente a las tablas de la base de datos. La carpeta DTOs (Data Transfer Objects) contiene las clases diseñadas para transferir datos de manera eficiente entre el cliente y el servidor.

  
Figura 32. Estructura BanQuito

##### Figura 32. Estructura Comercializadora

### **6****.1.8. Creación de la Base De Datos**

A diferencia de enfoques que utilizan herramientas de migración automatizada como Liquibase integradas en el proyecto, la estrategia para este proyecto se basa en la creación y gestión de la base de datos a través de scripts T-SQL explícitos. Este método, a menudo referido como "Database First", establece el esquema de la base de datos como la fuente de verdad, y la aplicación se desarrolla para interactuar con él.

El motor de base de datos seleccionado para ambos servicios es Microsoft SQL Server.

La conexión entre los servicios web y sus respectivas bases de datos no se define en un archivo de propiedades, sino en el archivo de configuración principal de la aplicación ASP.NET: Web.config. Dentro de este archivo, la sección <connectionStrings> especifica los parámetros necesarios para que Entity Framework pueda establecer la comunicación con el servidor de base de datos.

###### Tabla 2. Cadenas de conexión para los servicios

| <!-- Figura X. Definición de la cadena de conexión en Web.config -->  <connectionStrings>  <!-- Conexión para el servicio Banquito -->  <add name="BanquitoDb"  connectionString="Server=localhost,1433;Database=BanquitoDB;User Id=sa;Password=Admin1234;TrustServerCertificate=True;MultipleActiveResultSets=True;Encrypt=False;"  providerName="System.Data.SqlClient" />    <!-- Conexión para el servicio Comercializadora -->  <add name="ComercializadoraDb"  connectionString="Server=localhost,1433;Database=ComercializadoraDB;User Id=sa;Password=Admin1234;TrustServerCertificate=True;MultipleActiveResultSets=True;Encrypt=False;"  providerName="System.Data.SqlClient" />  </connectionStrings> |
| --- |

Para cada servicio (Banquito y Comercializadora), se ha proporcionado un script .sql dedicado que contiene todas las sentencias T-SQL necesarias para construir y poblar la base de datos desde cero.

Las responsabilidades de estos scripts incluyen:

Creación de la base de datos: Se asegura de que la base de datos exista.

Definición de Tablas: Crea cada una de las tablas (Cliente, Credito, Producto, Factura, etc.) especificando columnas, tipos de datos, restricciones (NOT NULL), claves primarias (PRIMARY KEY) y claves foráneas (FOREIGN KEY).

Creación de Índices: Define índices en columnas clave (como Cedula o NumeroFactura) para optimizar el rendimiento de las consultas.

Inserción de Datos de Prueba: Puebla las tablas con un conjunto inicial de datos (clientes, productos, movimientos) para permitir la prueba y validación inmediata de los servicios.

A continuación, se muestra un extracto del script de la base de datos de BanQuito como ejemplo:

###### **Tabla 3.** Ejemplo de script T-SQL para la creación e inserción de datos

| -- Figura X. Extracto del script de creación de la base de datos BanquitoDB  -- Creación de la tabla Cliente  CREATE TABLE Cliente (  ClienteId BIGINT IDENTITY(1,1) PRIMARY KEY,  Cedula VARCHAR(10) NOT NULL UNIQUE,  Nombres VARCHAR(100) NOT NULL,  Apellidos VARCHAR(100) NOT NULL,  FechaNacimiento DATE NOT NULL,  EstadoCivil VARCHAR(20),  Direccion VARCHAR(200),  Telefono VARCHAR(20),  Email VARCHAR(100),  INDEX IX\_Cliente\_Cedula (Cedula)  );  GO  -- Inserción de un cliente de prueba  INSERT INTO Cliente (Cedula, Nombres, Apellidos, FechaNacimiento, EstadoCivil, Direccion, Telefono, Email)  VALUES ('1234567890', 'Juan Carlos', 'Pérez González', '1985-03-15', 'Casado', 'Av. Amazonas N34-451', '0998765432', 'juan.perez@email.com');  GO |
| --- |

# 7. CODIFICACIÓN DE LOS SERVIDORES

La solución de backend está compuesta por dos servicios web independientes pero complementarios, ambos desarrollados sobre Microsoft .NET Framework 4.7.2 utilizando Windows Communication Foundation (WCF) para exponer sus funcionalidades a través de endpoints SOAP. Cada servidor sigue una arquitectura en N-Capas bien definida para separar las responsabilidades, mejorar la mantenibilidad y facilitar las pruebas. A continuación, se detalla la codificación de los componentes más relevantes de cada servidor.

## 7.1. Servidor BanQuito

Este servidor encapsula toda la lógica de negocio relacionada con el módulo de créditos del banco.

### 7.1.1. Capa de Servicio (Carpeta WS)

Actúa como la fachada del sistema, exponiendo las operaciones de negocio a través de un contrato de servicio SOAP.

* Contrato de Servicio (ICreditoSoapService.cs): Define las cuatro operaciones principales que el servicio ofrecerá a los clientes consumidores.

| // Archivo: WS/ICreditoSoapService.cs  [ServiceContract]  public interface ICreditoSoapService  {  [OperationContract]  ValidacionCreditoDTO ValidarSujetoCredito(string cedula);  [OperationContract]  MontoMaximoCreditoDTO ObtenerMontoMaximo(string cedula);  [OperationContract]  RespuestaCreditoDTO OtorgarCredito(SolicitudCreditoDTO solicitud);  [OperationContract]  List<CuotaAmortizacionDTO> ObtenerTablaAmortizacion(string numeroCredito);  } |
| --- |

* **Implementación del Servicio (CreditoSoapService.svc.cs):** Implementa el contrato, delegando la lógica de negocio a la capa correspondiente y manejando la creación de las respuestas.

| public class CreditoSoapService : ICreditoSoapService  {  private readonly CreditoValidacionService \_validacionService;  private readonly CreditoService \_creditoService;  public CreditoSoapService()  {  \_validacionService = new CreditoValidacionService();  \_creditoService = new CreditoService();  }  /// <summary>  /// WS 1: Validar si una persona es sujeto de crédito  /// GET /CreditoSoapService.svc  /// Parámetro: cedula  /// </summary>  public ValidacionCreditoDTO ValidarSujetoCredito(string cedula)  {  try  {  // Validar formato de cédula  if (!CedulaValidator.Validar(cedula, out string mensajeError))  {  return new ValidacionCreditoDTO  {  EsValido = false,  Mensaje = mensajeError,  Cedula = cedula,  NombreCompleto = null  };  }  var resultado = \_validacionService.ValidarSujetoCredito(cedula);  return new ValidacionCreditoDTO  {  EsValido = resultado.EsValido,  Mensaje = resultado.Mensaje,  Cedula = cedula,  NombreCompleto = resultado.Cliente?.NombreCompleto  };  }  catch (Exception ex)  {  return new ValidacionCreditoDTO  {  EsValido = false,  Mensaje = string.Format(ErrorMessages.ErrorInterno, ex.Message),  Cedula = cedula,  NombreCompleto = null  };  }  }  /// <summary>  /// WS 2: Obtener el monto máximo de crédito  /// GET /CreditoSoapService.svc  /// Parámetro: cedula  /// </summary>  public MontoMaximoCreditoDTO ObtenerMontoMaximo(string cedula)  {  try  {  // Validar formato de cédula  if (!CedulaValidator.Validar(cedula, out string mensajeError))  {  return new MontoMaximoCreditoDTO  {  Cedula = cedula,  MontoMaximo = 0,  PromedioDepositos = 0,  PromedioRetiros = 0,  Mensaje = mensajeError  };  }  var resultado = \_validacionService.CalcularMontoMaximo(cedula);  return new MontoMaximoCreditoDTO  {  Cedula = cedula,  MontoMaximo = resultado.MontoMaximo,  PromedioDepositos = resultado.PromedioDepositos,  PromedioRetiros = resultado.PromedioRetiros,  Mensaje = resultado.Mensaje  };  }  catch (Exception ex)  {  return new MontoMaximoCreditoDTO  {  Cedula = cedula,  MontoMaximo = 0,  PromedioDepositos = 0,  PromedioRetiros = 0,  Mensaje = string.Format(ErrorMessages.ErrorInterno, ex.Message)  };  }  }  /// <summary>  /// WS 3: Otorgar un crédito y generar tabla de amortización  /// POST /CreditoSoapService.svc  /// Body: SolicitudCreditoDTO  /// </summary>  public RespuestaCreditoDTO OtorgarCredito(SolicitudCreditoDTO solicitud)  {  try  {  // Validar y convertir la solicitud  if (!SolicitudCreditoValidator.ValidarYConvertir(solicitud, out decimal precio, out int cuotas, out string mensajeError))  {  return new RespuestaCreditoDTO  {  Exito = false,  Mensaje = mensajeError,  Cedula = solicitud?.Cedula  };  }  var resultado = \_creditoService.OtorgarCredito(  solicitud.Cedula,  precio,  cuotas);  var response = new RespuestaCreditoDTO  {  Exito = resultado.Exito,  Mensaje = resultado.Mensaje,  Cedula = solicitud.Cedula  };  if (resultado.Exito && resultado.Credito != null)  {  response.NumeroCredito = resultado.Credito.NumeroCredito;  response.MontoCredito = resultado.Credito.MontoCredito;  response.NumeroCuotas = resultado.Credito.NumeroCuotas;  response.CuotaMensual = resultado.Credito.CuotaMensual;  response.TasaInteres = resultado.Credito.TasaInteres;  // Convertir tabla de amortización a DTO  response.TablaAmortizacion = resultado.TablaAmortizacion.Select(c => new CuotaAmortizacionDTO  {  NumeroCuota = c.NumeroCuota,  ValorCuota = c.ValorCuota,  Interes = c.Interes,  CapitalPagado = c.CapitalPagado,  Saldo = c.Saldo  }).ToList();  }  return response;  }  catch (Exception ex)  {  return new RespuestaCreditoDTO  {  Exito = false,  Mensaje = string.Format(ErrorMessages.ErrorInterno, ex.Message),  Cedula = solicitud?.Cedula  };  }  }  /// <summary>  /// WS 4: Obtener la tabla de amortización de un crédito  /// GET /CreditoSoapService.svc  /// Parámetro: numeroCredito  /// </summary>  public List<CuotaAmortizacionDTO> ObtenerTablaAmortizacion(string numeroCredito)  {  try  {  var tabla = \_creditoService.ObtenerTablaAmortizacion(numeroCredito);  return tabla.Select(c => new CuotaAmortizacionDTO  {  NumeroCuota = c.NumeroCuota,  ValorCuota = c.ValorCuota,  Interes = c.Interes,  CapitalPagado = c.CapitalPagado,  Saldo = c.Saldo  }).ToList();  }  catch (Exception ex)  {  // Retornar lista vacía en caso de error  return new List<CuotaAmortizacionDTO>();  }  }  } |
| --- |

### 7.1.2. Capa de Lógica de Negocio (Carpeta Services)

Contiene las reglas de negocio críticas del sistema de créditos.

* **Servicio de Validación (CreditoValidacionService.cs):** Implementa las reglas para determinar si un cliente es sujeto de crédito y para calcular su capacidad de endeudamiento.

| using System;  using BanquitoServer\_Soap\_DotNet\_GR01.Constants;  using BanquitoServer\_Soap\_DotNet\_GR01.DataAccess;  using BanquitoServer\_Soap\_DotNet\_GR01.DataAccess.Repositories;  using BanquitoServer\_Soap\_DotNet\_GR01.Models;  using BanquitoServer\_Soap\_DotNet\_GR01.Utilities;  namespace BanquitoServer\_Soap\_DotNet\_GR01.BusinessLogic  {  /// <summary>  /// Servicio para validación de crédito y cálculo de monto máximo  /// </summary>  public class CreditoValidacionService  {  private readonly ClienteRepository \_clienteRepository;  private readonly MovimientoRepository \_movimientoRepository;  private readonly CreditoRepository \_creditoRepository;  public CreditoValidacionService()  {  var context = new BanquitoDbContext();  \_clienteRepository = new ClienteRepository(context);  \_movimientoRepository = new MovimientoRepository(context);  \_creditoRepository = new CreditoRepository(context);  }  /// <summary>  /// Validar si una persona es sujeto de crédito  /// Reglas:  /// 1. Es cliente del banco  /// 2. Tiene depósito en último mes  /// 3. Si casado, >= 25 años  /// 4. No tiene crédito activo  /// </summary>  public ResultadoValidacion ValidarSujetoCredito(string cedula)  {  // Regla 1: Verificar si es cliente del banco  Cliente cliente = \_clienteRepository.FindByCedula(cedula);  if (cliente == null)  {  return new ResultadoValidacion  {  EsValido = false,  Mensaje = ErrorMessages.ClienteNoEncontrado,  Cliente = null  };  }  // Regla 2: Verificar que tenga al menos un depósito en el último mes  if (!\_movimientoRepository.TieneDepositoEnUltimoMes(cedula))  {  return new ResultadoValidacion  {  EsValido = false,  Mensaje = ErrorMessages.SinDepositosRecientes,  Cliente = null  };  }  // Regla 3: Si es casado, debe tener al menos 25 años  if (!string.IsNullOrEmpty(cliente.EstadoCivil) &&  cliente.EstadoCivil.ToLower() == "casado")  {  int edad = CalculadoraFinanciera.CalcularEdad(cliente.FechaNacimiento);  if (edad < 25)  {  return new ResultadoValidacion  {  EsValido = false,  Mensaje = string.Format(ErrorMessages.EdadMinimaCasado, edad),  Cliente = null  };  }  }  // Regla 4: No debe tener un crédito activo  if (\_creditoRepository.TieneCreditoActivo(cedula))  {  return new ResultadoValidacion  {  EsValido = false,  Mensaje = ErrorMessages.CreditoActivo,  Cliente = null  };  }  // Si pasó todas las validaciones  return new ResultadoValidacion  {  EsValido = true,  Mensaje = ErrorMessages.ClienteValido,  Cliente = cliente  };  }  /// <summary>  /// Calcular el monto máximo de crédito  /// Fórmula: ((Promedio Depósitos – Promedio Retiros) \* 60%) \* 9  /// </summary>  public ResultadoMontoMaximo CalcularMontoMaximo(string cedula)  {  // Primero validar si es sujeto de crédito  var validacion = ValidarSujetoCredito(cedula);  if (!validacion.EsValido)  {  return new ResultadoMontoMaximo  {  MontoMaximo = 0,  PromedioDepositos = 0,  PromedioRetiros = 0,  Mensaje = validacion.Mensaje  };  }  // Calcular promedios de los últimos 3 meses  DateTime fechaInicio = DateTime.Now.AddMonths(-3);  DateTime fechaFin = DateTime.Now;  decimal promedioDepositos = \_movimientoRepository.PromedioDepositosPorPeriodo(cedula, fechaInicio, fechaFin);  decimal promedioRetiros = \_movimientoRepository.PromedioRetirosPorPeriodo(cedula, fechaInicio, fechaFin);  // Obtener configuración desde AppConfig  decimal porcentaje = AppConfig.CreditoPorcentajeCapacidad;  int multiplicador = AppConfig.CreditoMultiplicador;  // Calcular monto máximo  decimal diferencia = promedioDepositos - promedioRetiros;  decimal montoMaximo = Math.Max(0, diferencia \* porcentaje \* multiplicador);  montoMaximo = Math.Round(montoMaximo, 2);  return new ResultadoMontoMaximo  {  MontoMaximo = montoMaximo,  PromedioDepositos = Math.Round(promedioDepositos, 2),  PromedioRetiros = Math.Round(promedioRetiros, 2),  Mensaje = ErrorMessages.MontoMaximoCalculado  };  }  }  /// <summary>  /// Clase para resultado de validación  /// </summary>  public class ResultadoValidacion  {  public bool EsValido { get; set; }  public string Mensaje { get; set; }  public Cliente Cliente { get; set; }  }  /// <summary>  /// Clase para resultado de monto máximo  /// </summary>  public class ResultadoMontoMaximo  {  public decimal MontoMaximo { get; set; }  public decimal PromedioDepositos { get; set; }  public decimal PromedioRetiros { get; set; }  public string Mensaje { get; set; }  }  } |
| --- |

### 7.1.3. Capa de Acceso a Datos (Carpeta DataAccess)

Gestiona la comunicación con la base de datos BanquitoDB utilizando Entity Framework y el patrón Repository.

* **Contexto de la Base de Datos (BanquitoDbContext.cs):** Define la sesión con la base de datos y mapea las clases de entidad a las tablas.

| public class BanquitoDbContext : DbContext  {  /// <summary>  /// Constructor - usa la cadena de conexión "BanquitoDb" del Web.config  /// </summary>  public BanquitoDbContext() : base("name=BanquitoDb")  {  // Deshabilitar inicializador de base de datos  // (no queremos que EF cree o modifique la BD automáticamente)  Database.SetInitializer<BanquitoDbContext>(null);  }  // DbSets - representan las tablas de la base de datos  public virtual DbSet<Cliente> Clientes { get; set; }  public virtual DbSet<Cuenta> Cuentas { get; set; }  public virtual DbSet<Movimiento> Movimientos { get; set; }  public virtual DbSet<Credito> Creditos { get; set; }  public virtual DbSet<CuotaAmortizacion> CuotasAmortizacion { get; set; }  protected override void OnModelCreating(DbModelBuilder modelBuilder)  {  base.OnModelCreating(modelBuilder);  // Configurar precisión para campos decimales  modelBuilder.Entity<Cuenta>()  .Property(c => c.Saldo)  .HasPrecision(12, 2);  modelBuilder.Entity<Movimiento>()  .Property(m => m.Monto)  .HasPrecision(12, 2);  modelBuilder.Entity<Movimiento>()  .Property(m => m.SaldoAnterior)  .HasPrecision(12, 2);  modelBuilder.Entity<Movimiento>()  .Property(m => m.SaldoNuevo)  .HasPrecision(12, 2);  modelBuilder.Entity<Credito>()  .Property(c => c.MontoCredito)  .HasPrecision(12, 2);  modelBuilder.Entity<Credito>()  .Property(c => c.TasaInteres)  .HasPrecision(5, 4);  modelBuilder.Entity<Credito>()  .Property(c => c.CuotaMensual)  .HasPrecision(12, 2);  modelBuilder.Entity<CuotaAmortizacion>()  .Property(c => c.ValorCuota)  .HasPrecision(12, 2);  modelBuilder.Entity<CuotaAmortizacion>()  .Property(c => c.Interes)  .HasPrecision(12, 2);  modelBuilder.Entity<CuotaAmortizacion>()  .Property(c => c.CapitalPagado)  .HasPrecision(12, 2);  modelBuilder.Entity<CuotaAmortizacion>()  .Property(c => c.Saldo)  .HasPrecision(12, 2);  // Configurar relaciones  modelBuilder.Entity<Cuenta>()  .HasRequired(c => c.Cliente)  .WithMany(cl => cl.Cuentas)  .HasForeignKey(c => c.ClienteId)  .WillCascadeOnDelete(false);  modelBuilder.Entity<Movimiento>()  .HasRequired(m => m.Cuenta)  .WithMany(c => c.Movimientos)  .HasForeignKey(m => m.CuentaId)  .WillCascadeOnDelete(false);  modelBuilder.Entity<Credito>()  .HasRequired(cr => cr.Cliente)  .WithMany(cl => cl.Creditos)  .HasForeignKey(cr => cr.ClienteId)  .WillCascadeOnDelete(false);  modelBuilder.Entity<CuotaAmortizacion>()  .HasRequired(ca => ca.Credito)  .WithMany(cr => cr.CuotasAmortizacion)  .HasForeignKey(ca => ca.CreditoId)  .WillCascadeOnDelete(true); // Cascade delete para cuotas  }  } |
| --- |

* **Repositorio de Crédito (CreditoRepository.cs):** Abstrae las consultas a la base de datos relacionadas con la entidad Credito.

| {  private readonly BanquitoDbContext \_context;  public CreditoRepository()  {  \_context = new BanquitoDbContext();  }  public CreditoRepository(BanquitoDbContext context)  {  \_context = context;  }  /// <summary>  /// Verificar si el cliente tiene un crédito activo  /// </summary>  public bool TieneCreditoActivo(string cedula)  {  var tieneCredito = (from cr in \_context.Creditos  join cl in \_context.Clientes on cr.ClienteId equals cl.ClienteId  where cl.Cedula == cedula  && cr.Estado == "ACTIVO"  select cr).Any();  return tieneCredito;  }  /// <summary>  /// Guardar un crédito con su tabla de amortización  /// </summary>  public void Save(Credito credito)  {  try  {  \_context.Creditos.Add(credito);  \_context.SaveChanges();  }  catch (DbEntityValidationException ex)  {  // Construir mensaje detallado de errores de validación  StringBuilder sb = new StringBuilder();  sb.AppendLine("Errores de validación de Entity Framework:");  foreach (var validationErrors in ex.EntityValidationErrors)  {  foreach (var validationError in validationErrors.ValidationErrors)  {  sb.AppendLine($" - Propiedad: {validationError.PropertyName}, Error: {validationError.ErrorMessage}");  }  }  throw new Exception(sb.ToString(), ex);  }  }  /// <summary>  /// Buscar crédito por número de crédito  /// </summary>  public Credito FindByNumeroCredito(string numeroCredito)  {  return \_context.Creditos  .Include("CuotasAmortizacion")  .FirstOrDefault(c => c.NumeroCredito == numeroCredito);  }  } |
| --- |

### 7.1.4. Capa de Modelos y DTOs (Carpetas Models y DTOs)

Define las estructuras de datos del servicio.

* **Modelo de Entidad (Credito.cs):** Representa la tabla Credito de la base de datos.

| [Table("Credito")]  public class Credito  {  [Key]  [Column("CreditoId")]  [DatabaseGenerated(DatabaseGeneratedOption.Identity)]  public long CreditoId { get; set; }  [Required]  [Column("NumeroCredito")]  [MaxLength(20)]  [Index("IX\_Credito\_NumeroCredito", IsUnique = true)]  public string NumeroCredito { get; set; }  [Required]  [Column("MontoCredito")]  public decimal MontoCredito { get; set; }  [Required]  [Column("TasaInteres")]  public decimal TasaInteres { get; set; }  [Required]  [Column("NumeroCuotas")]  public int NumeroCuotas { get; set; }  [Required]  [Column("CuotaMensual")]  public decimal CuotaMensual { get; set; }  [Required]  [Column("FechaOtorgamiento")]  public DateTime FechaOtorgamiento { get; set; }  [Required]  [Column("Estado")]  [MaxLength(20)]  [Index("IX\_Credito\_Estado")]  public string Estado { get; set; }  [Column("Descripcion")]  [MaxLength(200)]  public string Descripcion { get; set; }  // Foreign Key  [Required]  [Column("ClienteId")]  [ForeignKey("Cliente")]  [Index("IX\_Credito\_ClienteId")]  public long ClienteId { get; set; }  // Propiedades de navegación  public virtual Cliente Cliente { get; set; }  public virtual ICollection<CuotaAmortizacion> CuotasAmortizacion { get; set; }  public Credito()  {  CuotasAmortizacion = new HashSet<CuotaAmortizacion>();  Estado = "ACTIVO";  FechaOtorgamiento = DateTime.Now;  }  } |
| --- |

* **Data Transfer Object (SolicitudCreditoDTO.cs):** Objeto plano utilizado para recibir datos del cliente.

| // Archivo: DTOs/SolicitudCreditoDTO.cs  [DataContract]  public class SolicitudCreditoDTO  {  [DataMember]  public string Cedula { get; set; }  [DataMember]  public string PrecioElectrodomestico { get; set; }  [DataMember]  public string NumeroCuotas { get; set; }  } |
| --- |

## 7.2. Servidor Comercializadora

Este servidor maneja la lógica de negocio para la gestión de productos y la facturación de ventas.

### 7.2.1. Capa de Servicio (Carpeta WS)

Expone las operaciones de facturación y gestión de productos.

* Contrato de Servicio (IFacturacionSoapService.cs): Define las operaciones relacionadas con la facturación.

| [ServiceContract]  public interface IFacturacionSoapService  {  /// <summary>  /// Calcula el total de una factura SIN generarla  /// Útil para conocer el monto ANTES de solicitar crédito  /// </summary>  [OperationContract]  CalculoFacturaDTO CalcularTotalFactura(SolicitudCalculoDTO solicitud);  [OperationContract]  FacturaDTO GenerarFactura(SolicitudFacturaDTO solicitud);  [OperationContract]  List<FacturaDTO> ObtenerFacturasPorCliente(string cedula);  [OperationContract]  FacturaDTO ObtenerFacturaPorNumero(string numeroFactura);  } |
| --- |

### 7.2.2. Capa de Lógica de Negocio (Carpeta Services)

Implementa la lógica para la creación de facturas y la gestión del catálogo.

* **Servicio de Facturación (FacturacionService.cs):** Contiene la lógica para generar facturas, aplicando descuentos y actualizando el stock.

| public class FacturacionService  {  private readonly ProductoRepository \_productoRepository;  private readonly FacturaRepository \_facturaRepository;  public FacturacionService()  {  \_productoRepository = new ProductoRepository();  \_facturaRepository = new FacturaRepository();  }  /// <summary>  /// Calcula el total de una factura SIN generarla  /// Útil para conocer el monto ANTES de solicitar crédito en BanQuito  /// NO actualiza stock, NO crea factura en BD  /// </summary>  public CalculoFacturaDTO CalcularTotalFactura(SolicitudCalculoDTO solicitud)  {  try  {  // Validaciones básicas  if (solicitud == null)  throw new Exception("La solicitud de factura es nula");  if (solicitud.Items == null || !solicitud.Items.Any())  throw new Exception("La solicitud de factura no contiene productos. Debe enviar al menos un producto.");  decimal subtotal = 0;  var detalles = new List<DetalleCalculoDTO>();  // Calcular subtotal y validar productos  foreach (var item in solicitud.Items)  {  var producto = \_productoRepository.GetById(item.ProductoId);  if (producto == null)  throw new Exception($"Producto con ID {item.ProductoId} no encontrado");  if (!\_productoRepository.TieneStock(item.ProductoId, item.Cantidad))  throw new Exception($"Stock insuficiente para el producto '{producto.Nombre}'. Stock disponible: {producto.Stock}, solicitado: {item.Cantidad}");  var subtotalItem = producto.Precio \* item.Cantidad;  subtotal += subtotalItem;  detalles.Add(new DetalleCalculoDTO  {  ProductoId = item.ProductoId,  NombreProducto = producto.Nombre,  Cantidad = item.Cantidad,  PrecioUnitario = producto.Precio,  Subtotal = subtotalItem  });  }  // Retornar cálculo exitoso  return new CalculoFacturaDTO  {  Exitoso = true,  Mensaje = "Cálculo realizado exitosamente",  Total = subtotal,  Detalles = detalles  };  }  catch (Exception ex)  {  // Retornar error con mensaje comprensible  return new CalculoFacturaDTO  {  Exitoso = false,  Mensaje = ex.Message,  Total = 0,  Detalles = new List<DetalleCalculoDTO>()  };  }  }  /// <summary>  /// Genera una factura (EFECTIVO con 33% descuento o CREDITO sin descuento)  /// </summary>  /// <param name="solicitud">Solicitud con FormaPago: EFECTIVO o CREDITO.  /// Si es CREDITO, NumeroCredito es requerido (obtenido desde BanQuito)</param>  public FacturaDTO GenerarFactura(SolicitudFacturaDTO solicitud)  {  try  {  // Validaciones comunes  if (solicitud == null)  throw new Exception("La solicitud de factura es nula");  if (solicitud.Items == null || !solicitud.Items.Any())  throw new Exception("La solicitud de factura no contiene productos. Debe enviar al menos un producto.");  if (string.IsNullOrWhiteSpace(solicitud.CedulaCliente))  throw new Exception("La cédula del cliente es requerida");  if (string.IsNullOrWhiteSpace(solicitud.NombreCliente))  throw new Exception("El nombre del cliente es requerido");  // Validar FormaPago  if (string.IsNullOrWhiteSpace(solicitud.FormaPago))  throw new Exception("La forma de pago es requerida. Valores permitidos: EFECTIVO, CREDITO");  string formaPago = solicitud.FormaPago.ToUpper();  if (formaPago != "EFECTIVO" && formaPago != "CREDITO")  throw new Exception("Forma de pago inválida. Valores permitidos: EFECTIVO, CREDITO");  // Validar NumeroCredito solo si es CREDITO  if (formaPago == "CREDITO" && string.IsNullOrWhiteSpace(solicitud.NumeroCredito))  throw new Exception("El número de crédito es requerido para pagos a CREDITO. Debe obtenerlo desde el servicio de BanQuito.");  // Crear factura  var factura = new Factura  {  NumeroFactura = \_facturaRepository.GenerarNumeroFactura(),  CedulaCliente = solicitud.CedulaCliente,  NombreCliente = solicitud.NombreCliente,  FormaPago = formaPago,  NumeroCredito = formaPago == "CREDITO" ? solicitud.NumeroCredito : null,  FechaEmision = DateTime.Now  };  decimal subtotal = 0;  var detalles = new List<DetalleFactura>();  // Procesar cada item  foreach (var item in solicitud.Items)  {  var producto = \_productoRepository.GetById(item.ProductoId);  if (producto == null)  throw new Exception($"Producto con ID {item.ProductoId} no encontrado");  if (!\_productoRepository.TieneStock(item.ProductoId, item.Cantidad))  throw new Exception($"Stock insuficiente para el producto {producto.Nombre}");  var subtotalItem = producto.Precio \* item.Cantidad;  subtotal += subtotalItem;  var detalle = new DetalleFactura  {  ProductoId = item.ProductoId,  Cantidad = item.Cantidad,  PrecioUnitario = producto.Precio,  Subtotal = subtotalItem  };  detalles.Add(detalle);  // Actualizar stock  \_productoRepository.ActualizarStock(item.ProductoId, item.Cantidad);  }  // Calcular descuento según forma de pago  factura.Subtotal = subtotal;  if (formaPago == "EFECTIVO")  {  // EFECTIVO: 33% de descuento  factura.Descuento = subtotal \* 0.33m;  factura.Total = subtotal - factura.Descuento;  }  else  {  // CREDITO: Sin descuento  factura.Descuento = 0;  factura.Total = subtotal;  }  factura.Detalles = detalles;  // Guardar factura  var facturaCreada = \_facturaRepository.Create(factura);  // Mapear a DTO  return MapearFacturaADTO(facturaCreada);  }  catch (Exception ex)  {  throw new Exception($"Error al generar factura: {ex.Message}");  }  }  /// <summary>  /// Obtener facturas por cédula de cliente  /// </summary>  public List<FacturaDTO> ObtenerFacturasPorCliente(string cedula)  {  var facturas = \_facturaRepository.GetByCedulaCliente(cedula);  return facturas.Select(f => MapearFacturaADTO(f)).ToList();  }  /// <summary>  /// Obtener factura por número  /// </summary>  public FacturaDTO ObtenerFacturaPorNumero(string numeroFactura)  {  var factura = \_facturaRepository.GetByNumeroFactura(numeroFactura);  return factura != null ? MapearFacturaADTO(factura) : null;  }  // Mapear Factura a DTO  private FacturaDTO MapearFacturaADTO(Factura factura)  {  var facturaDto = new FacturaDTO  {  FacturaId = factura.FacturaId,  NumeroFactura = factura.NumeroFactura,  CedulaCliente = factura.CedulaCliente,  NombreCliente = factura.NombreCliente,  FormaPago = factura.FormaPago,  Subtotal = factura.Subtotal,  Descuento = factura.Descuento,  Total = factura.Total,  NumeroCredito = factura.NumeroCredito,  FechaEmision = factura.FechaEmision,  Detalles = new List<DetalleFacturaDTO>()  };  if (factura.Detalles != null)  {  foreach (var detalle in factura.Detalles)  {  var producto = \_productoRepository.GetById(detalle.ProductoId);  facturaDto.Detalles.Add(new DetalleFacturaDTO  {  ProductoId = detalle.ProductoId,  NombreProducto = producto?.Nombre ?? "Producto no encontrado",  Cantidad = detalle.Cantidad,  PrecioUnitario = detalle.PrecioUnitario,  Subtotal = detalle.Subtotal  });  }  }  return facturaDto;  }  } |
| --- |

### 7.2.3. Capa de Acceso a Datos (Carpeta DataAcces)

Gestiona la comunicación con la base de datos ComercializadoraDB.

* Repositorio de Producto (ProductoRepository.cs): Abstrae las consultas a la base de datos para la entidad Producto.

| public class ProductoRepository  {  private readonly ComercializadoraDbContext \_context;  public ProductoRepository()  {  \_context = new ComercializadoraDbContext();  }  public ProductoRepository(ComercializadoraDbContext context)  {  \_context = context;  }  // CREATE  public Producto Create(Producto producto)  {  \_context.Productos.Add(producto);  \_context.SaveChanges();  // El ID ya se asigna automáticamente por Identity después de SaveChanges  return producto;  }  // READ ALL  public List<Producto> GetAll()  {  return \_context.Productos  .Where(p => p.Estado == "ACTIVO")  .OrderBy(p => p.Nombre)  .ToList();  }  // READ BY ID  public Producto GetById(int id)  {  return \_context.Productos.Find(id);  }  // READ BY CODIGO  public Producto GetByCodigo(string codigo)  {  return \_context.Productos  .FirstOrDefault(p => p.Codigo == codigo);  }  // READ BY CATEGORIA  public List<Producto> GetByCategoria(string categoria)  {  return \_context.Productos  .Where(p => p.Categoria == categoria && p.Estado == "ACTIVO")  .OrderBy(p => p.Nombre)  .ToList();  }  // READ BY PRECIO RANGE  public List<Producto> GetByPrecioRange(decimal precioMin, decimal precioMax)  {  return \_context.Productos  .Where(p => p.Precio >= precioMin && p.Precio <= precioMax && p.Estado == "ACTIVO")  .OrderBy(p => p.Precio)  .ToList();  }  // UPDATE  public Producto Update(Producto producto)  {  var existing = \_context.Productos.Find(producto.ProductoId);  if (existing != null)  {  \_context.Entry(existing).CurrentValues.SetValues(producto);  \_context.SaveChanges();  return existing;  }  return null;  }  // DELETE (Logical)  public bool Delete(int id)  {  var producto = \_context.Productos.Find(id);  if (producto != null)  {  producto.Estado = "INACTIVO";  \_context.SaveChanges();  return true;  }  return false;  }  // DELETE (Physical)  public bool DeletePermanent(int id)  {  var producto = \_context.Productos.Find(id);  if (producto != null)  {  \_context.Productos.Remove(producto);  \_context.SaveChanges();  return true;  }  return false;  }  // Verificar si código existe  public bool ExisteCodigo(string codigo, int? excludeId = null)  {  var query = \_context.Productos.Where(p => p.Codigo == codigo);  if (excludeId.HasValue)  {  query = query.Where(p => p.ProductoId != excludeId.Value);  }  return query.Any();  }  // Verificar stock disponible  public bool TieneStock(int productoId, int cantidad)  {  var producto = \_context.Productos.Find(productoId);  return producto != null && producto.Stock >= cantidad;  }  // Actualizar stock  public bool ActualizarStock(int productoId, int cantidad)  {  var producto = \_context.Productos.Find(productoId);  if (producto != null && producto.Stock >= cantidad)  {  producto.Stock -= cantidad;  \_context.SaveChanges();  return true;  }  return false;  }  } |
| --- |

### 7.2.4. Capa de Modelos y DTOs (Carpetas Models y DTOs)

Define las estructuras de datos para la comercializadora.

* Modelo de Entidad (Factura.cs): Representa la tabla Factura de la base de datos.

| [Table("Factura")]  public class Factura  {  [Key]  [DatabaseGenerated(DatabaseGeneratedOption.Identity)]  public int FacturaId { get; set; }  [Required]  [MaxLength(20)]  [Index("IX\_Factura\_NumeroFactura", IsUnique = true)]  public string NumeroFactura { get; set; }  [Required]  [MaxLength(10)]  public string CedulaCliente { get; set; }  [Required]  [MaxLength(200)]  public string NombreCliente { get; set; }  [Required]  [MaxLength(20)]  public string FormaPago { get; set; } // EFECTIVO o CREDITO  [Required]  public decimal Subtotal { get; set; }  public decimal Descuento { get; set; }  [Required]  public decimal Total { get; set; }  [MaxLength(20)]  public string NumeroCredito { get; set; } // Solo si es crédito  [Required]  public DateTime FechaEmision { get; set; }  // Navigation properties  public virtual ICollection<DetalleFactura> Detalles { get; set; }  public Factura()  {  FechaEmision = DateTime.Now;  Descuento = 0;  Detalles = new HashSet<DetalleFactura>();  }  } |
| --- |

* Data Transfer Object (SolicitudFacturaDTO.cs): Estructura compleja para solicitar la creación de una factura, incluyendo una lista de los productos a comprar.

| [DataContract]  public class FacturaDTO  {  [DataMember]  public int FacturaId { get; set; }  [DataMember]  public string NumeroFactura { get; set; }  [DataMember]  public string CedulaCliente { get; set; }  [DataMember]  public string NombreCliente { get; set; }  [DataMember]  public string FormaPago { get; set; }  [DataMember]  public decimal Subtotal { get; set; }  [DataMember]  public decimal Descuento { get; set; }  [DataMember]  public decimal Total { get; set; }  [DataMember]  public string NumeroCredito { get; set; }  [DataMember]  public DateTime FechaEmision { get; set; }  [DataMember]  public List<DetalleFacturaDTO> Detalles { get; set; }  }  [DataContract]  public class DetalleFacturaDTO  {  [DataMember]  public int ProductoId { get; set; }  [DataMember]  public string NombreProducto { get; set; }  [DataMember]  public int Cantidad { get; set; }  [DataMember]  public decimal PrecioUnitario { get; set; }  [DataMember]  public decimal Subtotal { get; set; }  }  [DataContract]  public class SolicitudFacturaDTO  {  [DataMember(IsRequired = false, Order = 1)]  public string CedulaCliente { get; set; }  [DataMember(IsRequired = false, Order = 2)]  public string FormaPago { get; set; } // EFECTIVO o CREDITO  [DataMember(IsRequired = false, Order = 3)]  public List<ItemFacturaDTO> Items { get; set; }  [DataMember(IsRequired = false, Order = 4)]  public string NombreCliente { get; set; }  [DataMember(IsRequired = false, Order = 5)]  public string NumeroCredito { get; set; } // Solo para CREDITO - Obtenido desde BanQuito  public SolicitudFacturaDTO()  {  Items = new List<ItemFacturaDTO>();  }  }  [DataContract]  public class ItemFacturaDTO  {  [DataMember(Order = 1)]  public int Cantidad { get; set; }  [DataMember(Order = 2)]  public int ProductoId { get; set; }  }  /// <summary>  /// DTO específico para calcular total de factura  /// Solo requiere los productos (Items)  /// </summary>  [DataContract]  public class SolicitudCalculoDTO  {  [DataMember(IsRequired = true, Order = 1)]  public List<ItemFacturaDTO> Items { get; set; }  public SolicitudCalculoDTO()  {  Items = new List<ItemFacturaDTO>();  }  } |
| --- |

# **8****. PROYECTO CLIENTE WEB**

El proyecto cliente web es la interfaz de usuario desarrollada en ASP.NET Web Forms que permite a los usuarios finales interactuar con el sistema de comercialización de electrodomésticos. Esta aplicación web consume los servicios SOAP previamente implementados y proporciona funcionalidades completas para la gestión de ventas a crédito, consultas de facturas, visualización de productos y tablas de amortización.

## 8.1. Funcionalidades Principales

La aplicación web cliente implementa las siguientes funcionalidades clave:

Consulta de Facturas: Permite buscar facturas por número y visualizar todos los detalles de la compra.

Tablas de Amortización: Muestra el desglose completo de las cuotas de un crédito, incluyendo interés, capital y saldo.

Gestión de Productos: Interfaz administrativa para visualizar el catálogo completo de electrodomésticos.

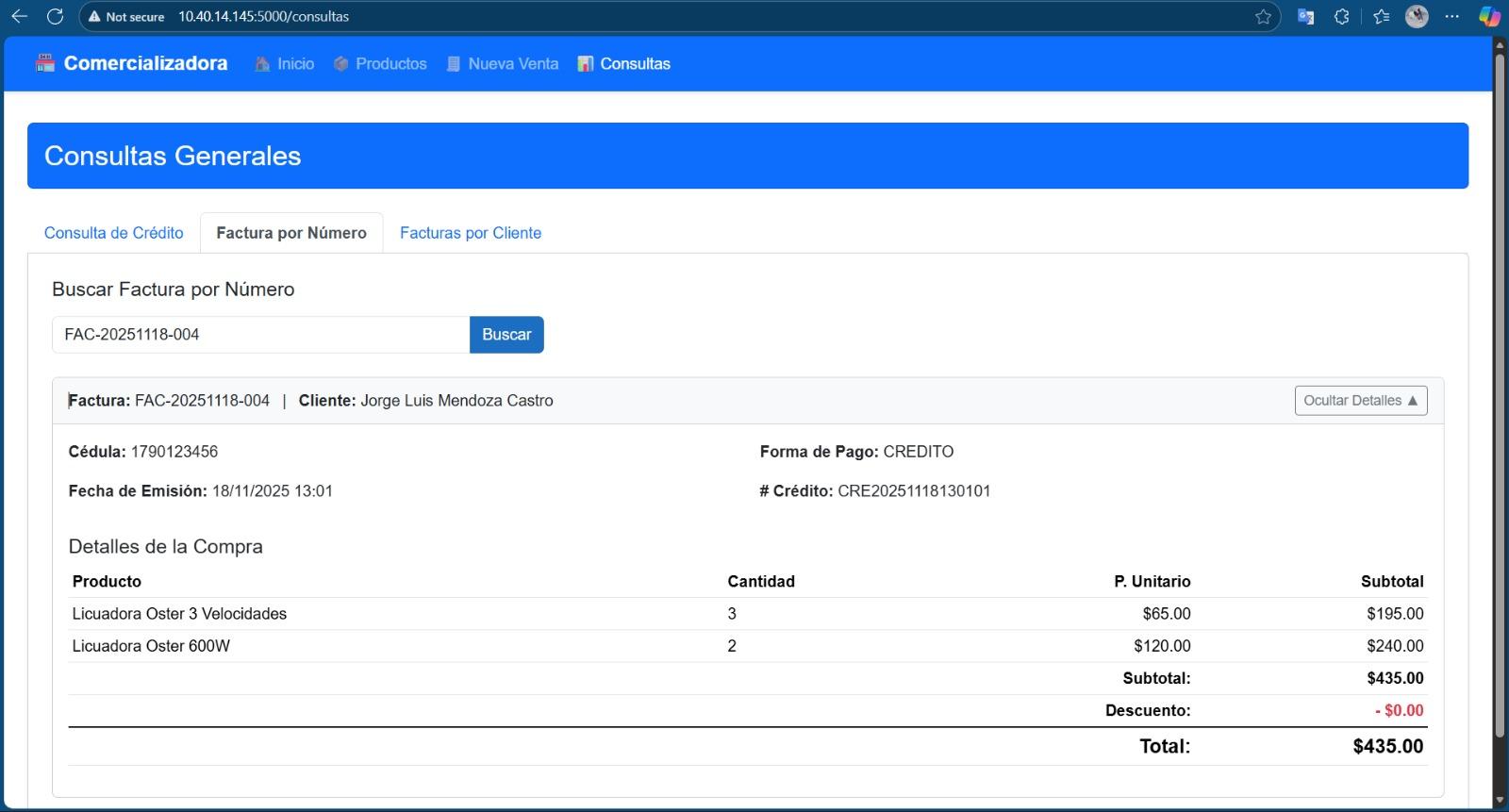
Nuevas Ventas a Crédito: Proceso completo para registrar ventas con financiamiento.

Consultas por Cliente: Búsqueda de facturas utilizando la cédula del cliente.

Página Principal: Dashboard centralizado con acceso rápido a todas las funcionalidades.

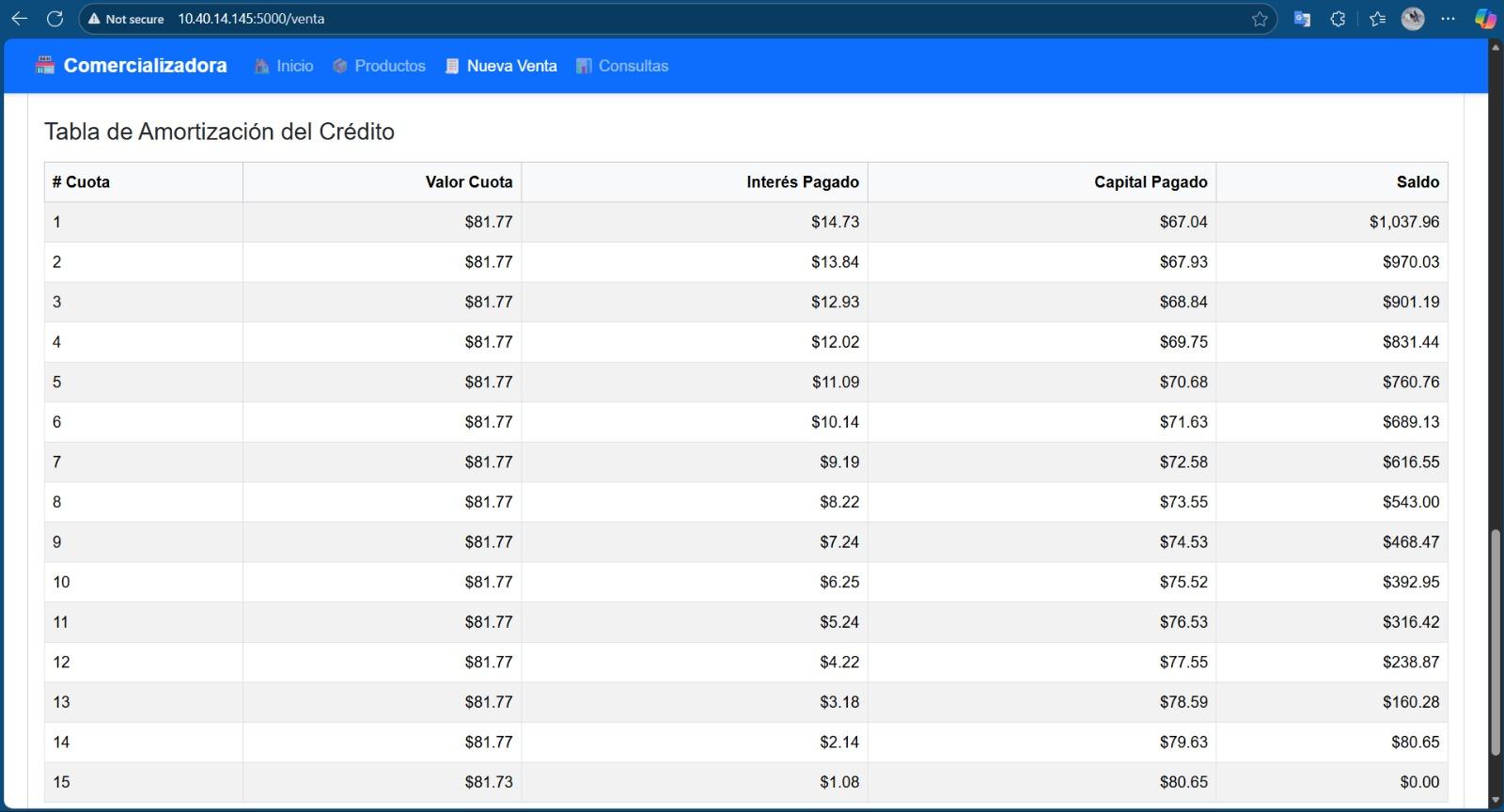
## 8.2. Capturas de Pantalla del Sistema

A continuación se presentan las interfaces principales del sistema web:



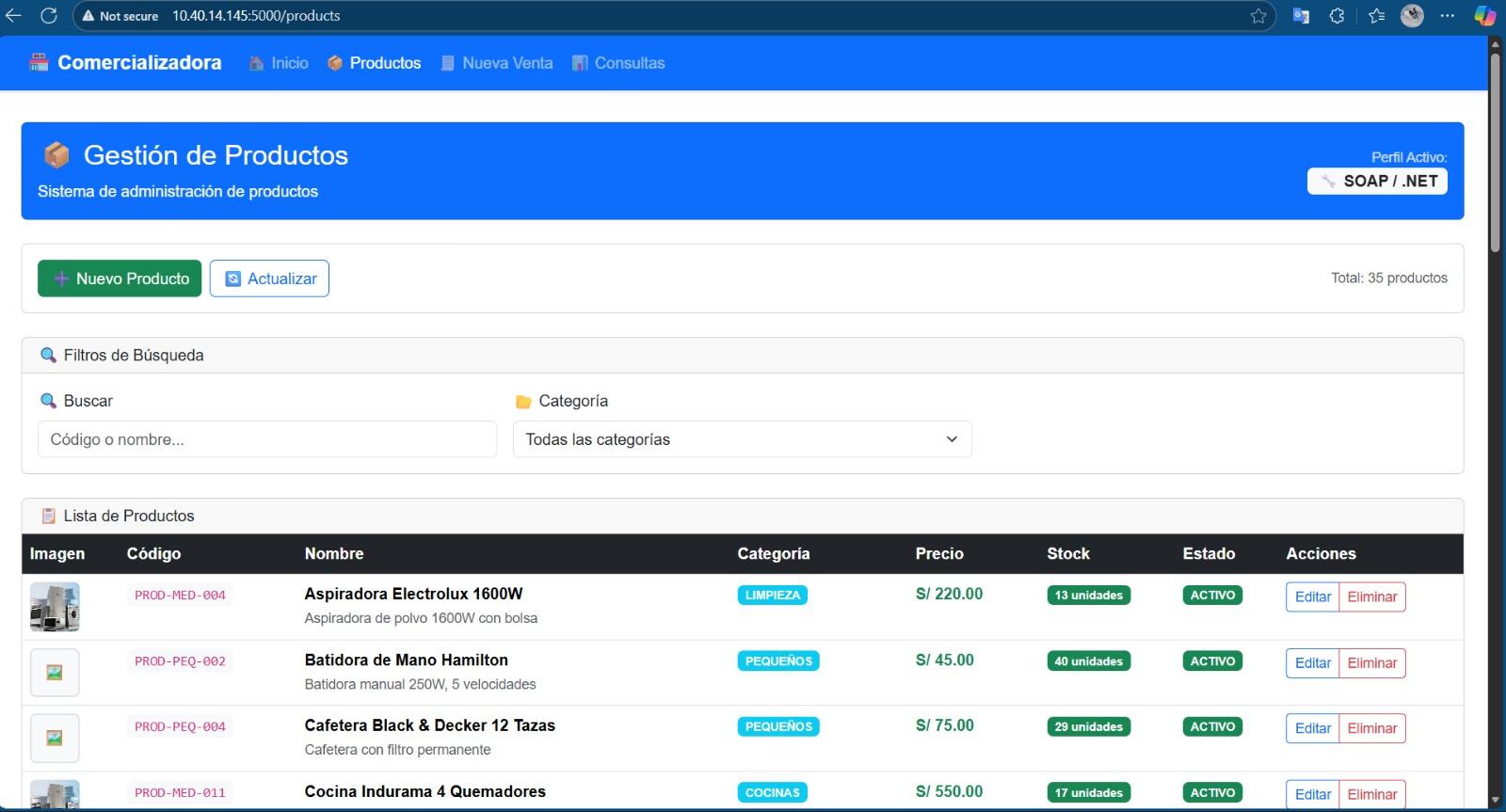
##### Figura 33. Consulta de Factura por Número

La Figura 33 muestra la interfaz de consulta de facturas por número. El usuario ingresa el número de factura (por ejemplo, FAC-20251118-004) y el sistema despliega la información completa incluyendo: datos del cliente (cédula, nombre completo), detalles de la transacción (fecha de emisión, forma de pago, número de crédito), y el desglose de productos adquiridos con sus cantidades, precios unitarios y subtotales. Al final se presenta el total de la factura.



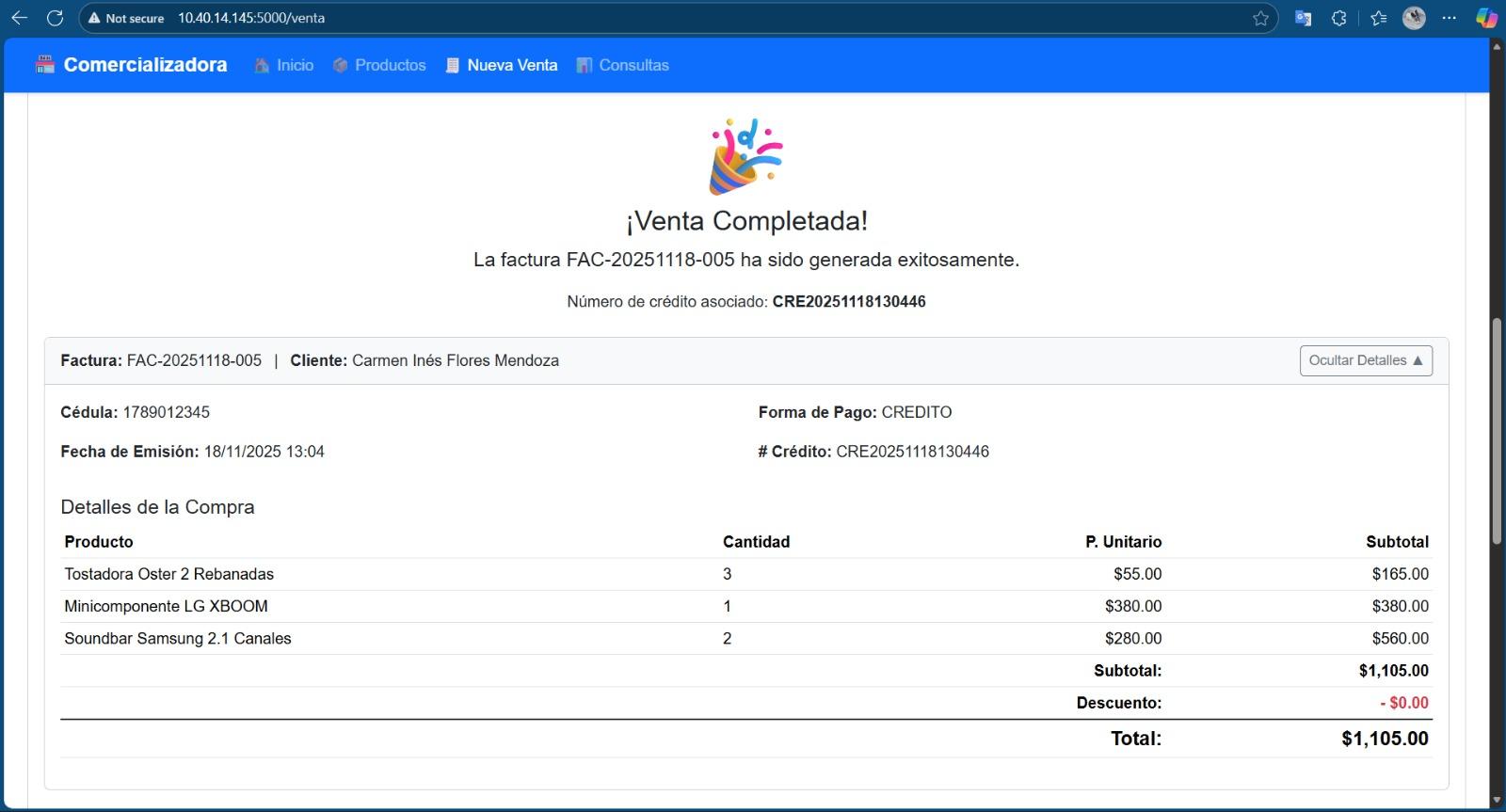
##### Figura 34. Tabla de Amortización del Crédito

La Figura 34 presenta la tabla de amortización asociada a un crédito específico. Esta interfaz muestra el número de crédito consultado y despliega una tabla detallada con las siguientes columnas: número de cuota, valor de la cuota, interés pagado, capital pagado y saldo restante. Esta información permite al usuario visualizar cómo se distribuyen los pagos a lo largo del período del crédito, mostrando la amortización progresiva del capital y la reducción gradual del saldo.



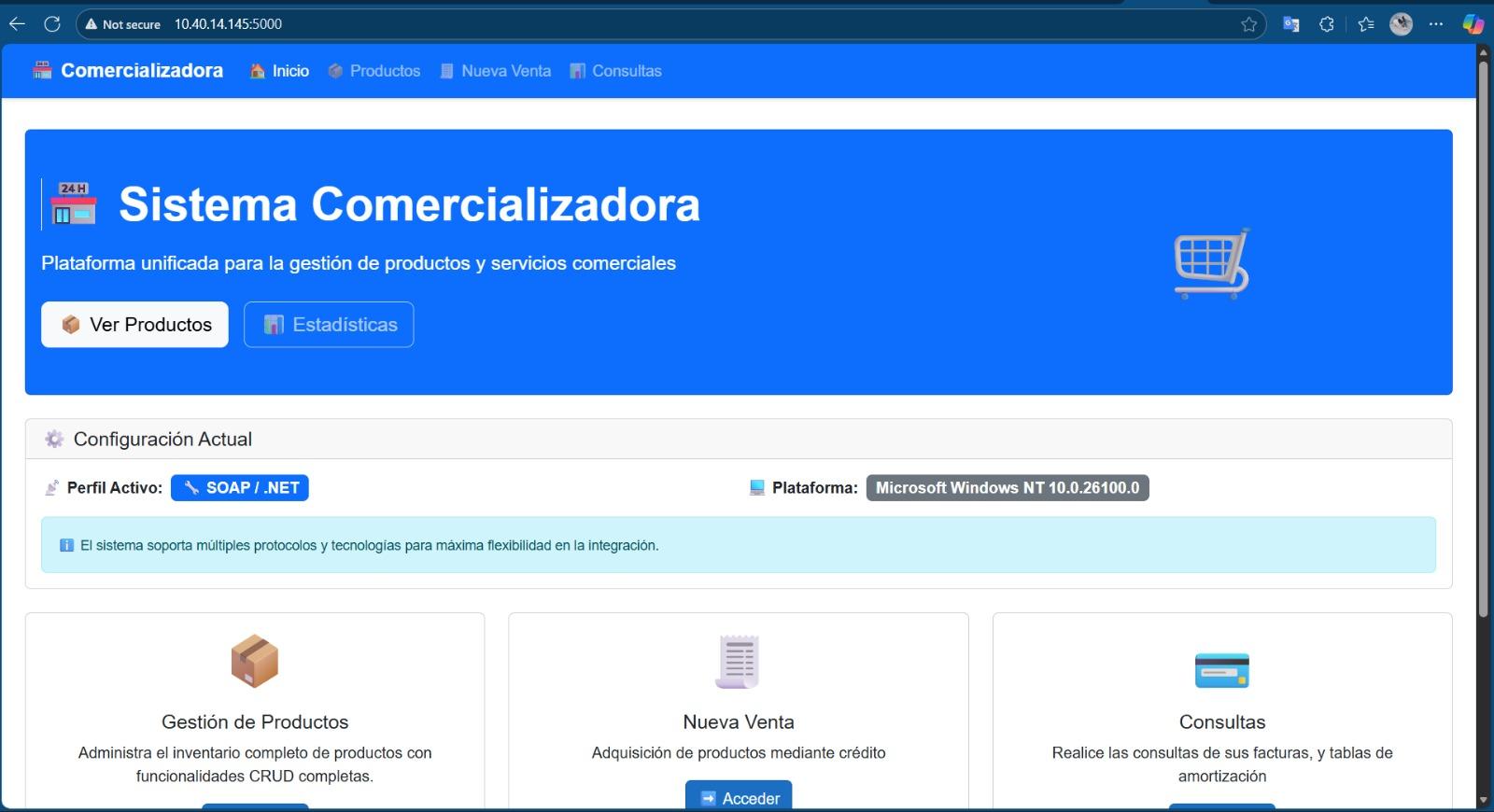
##### Figura 35. Gestión de Productos

La Figura 35 muestra la interfaz de gestión de productos del sistema. Esta página administrativa presenta un listado completo del catálogo de electrodomésticos disponibles, mostrando para cada producto: imagen representativa, código único, nombre del producto, descripción breve, categoría a la que pertenece, precio unitario, cantidad en stock, estado (activo/inactivo) y acciones disponibles (editar, eliminar). La interfaz incluye filtros de búsqueda por código/nombre y por categoría, además de botones para agregar nuevos productos o actualizar el catálogo.



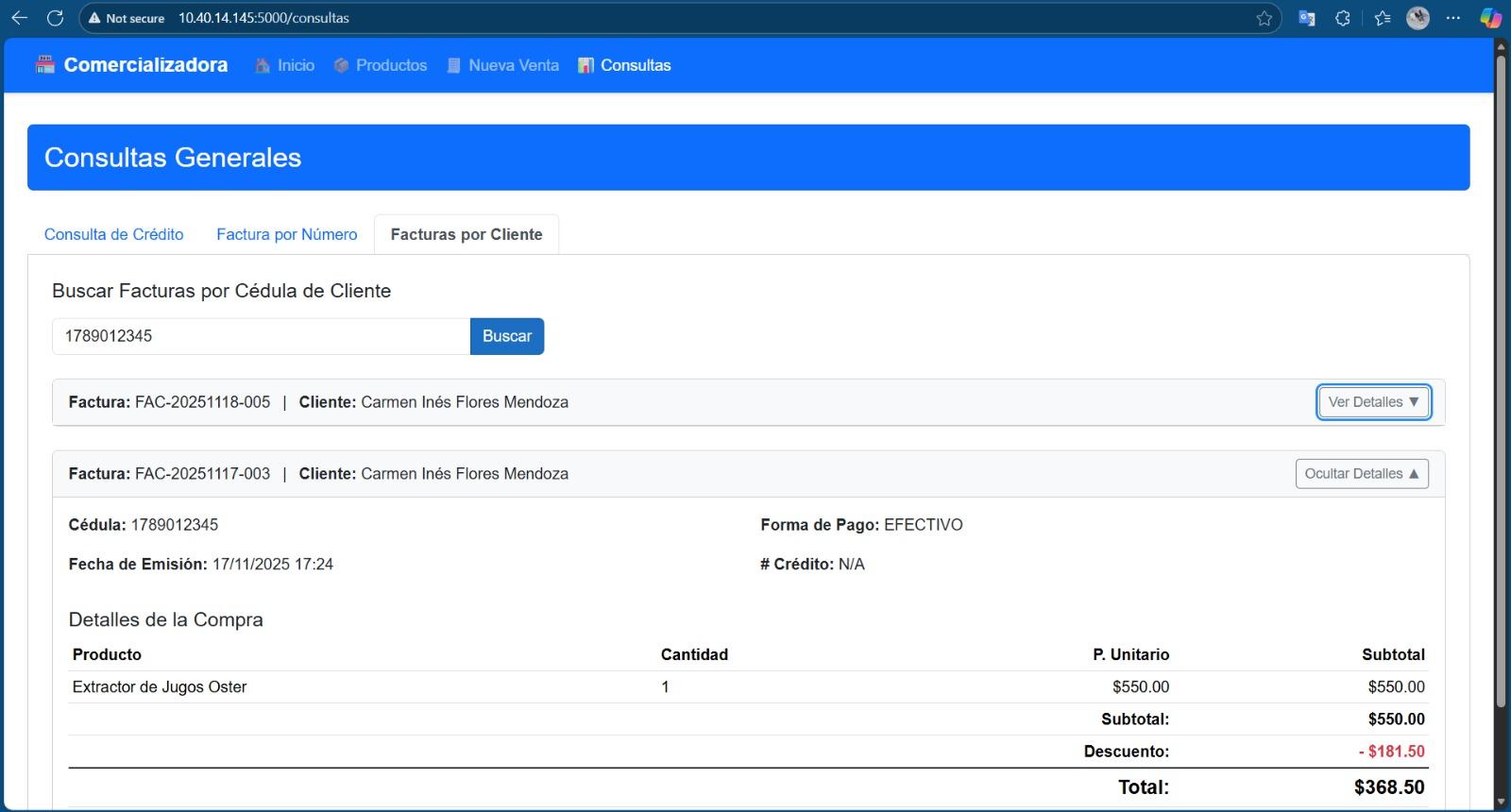
##### Figura 36. Confirmación de Venta Completada

La Figura 36 presenta la pantalla de confirmación que se muestra al usuario después de completar exitosamente una venta a crédito. El mensaje "¡Venta Completada!" indica el éxito de la operación. La interfaz muestra el número de factura generado (por ejemplo, FAC-20251118-005), el número de crédito asociado, datos completos del cliente, forma de pago utilizada (CRÉDITO), y el desglose detallado de los productos vendidos con sus cantidades, precios unitarios y el total de la transacción. Esta pantalla sirve como comprobante inmediato de la operación realizada.



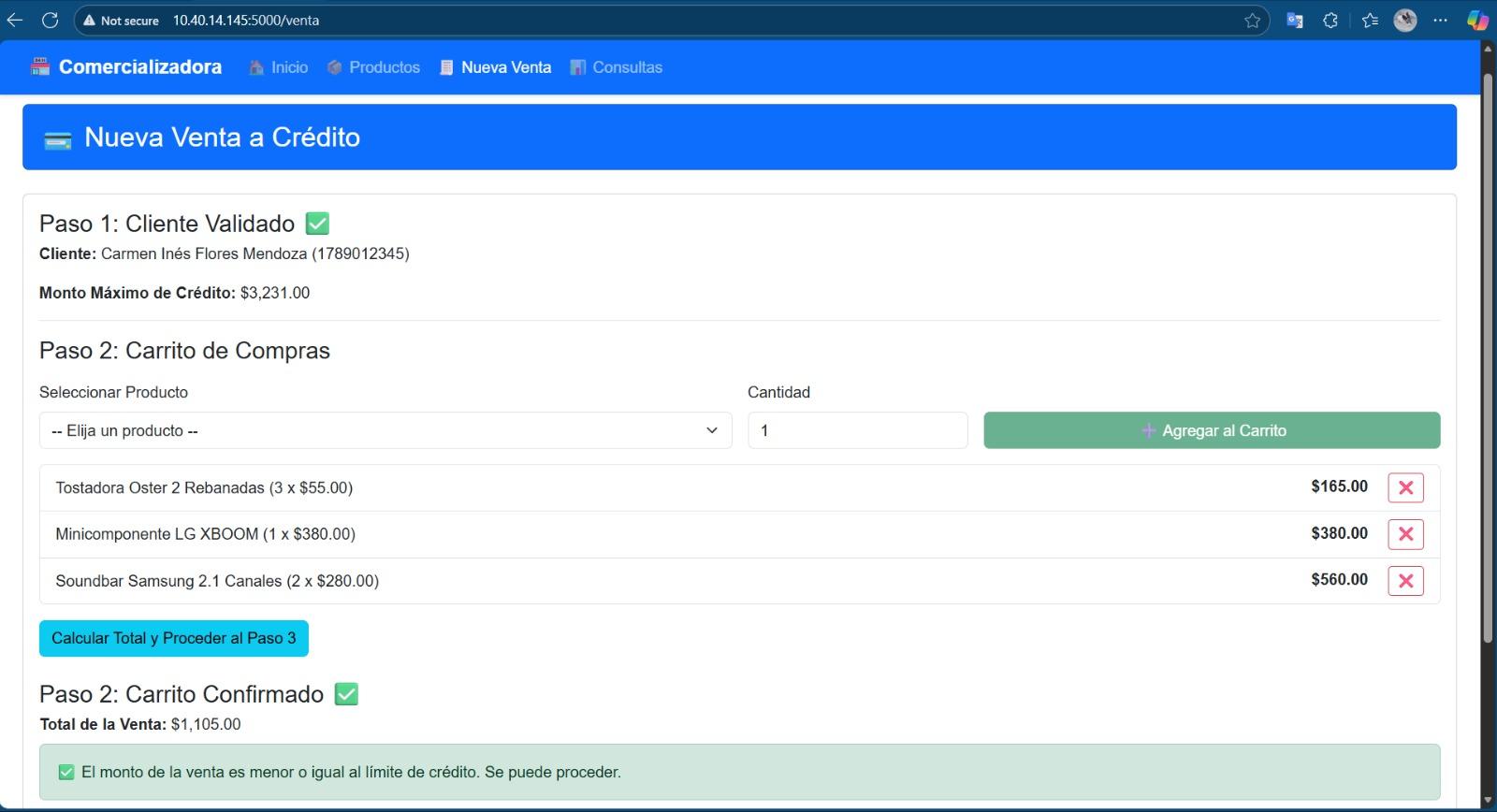
##### Figura 37. Página Principal del Sistema

La Figura 37 muestra la página principal o dashboard del Sistema Comercializadora. Esta interfaz de inicio presenta el logotipo y nombre del sistema, acompañado de una descripción que indica que es una "Plataforma unificada para la gestión de productos y servicios comerciales". El dashboard incluye información de configuración actual del sistema, mostrando el perfil activo (SOAP / .NET) y la plataforma utilizada (Microsoft Windows NT). Además, presenta tres módulos principales accesibles mediante tarjetas: Gestión de Productos, Nueva Venta (con botón de acceso destacado), y Consultas. Esta página sirve como punto de entrada centralizado a todas las funcionalidades del sistema.



##### Figura 38. Consulta de Facturas por Cliente

La Figura 38 presenta la interfaz de consulta de facturas por cédula de cliente. Esta funcionalidad permite al usuario ingresar el número de cédula de un cliente específico y obtener un listado de todas las facturas asociadas a dicho cliente. La pantalla muestra los resultados de la búsqueda, desplegando para cada factura: número de factura, datos completos del cliente (cédula, nombre), fecha de emisión, forma de pago utilizada, número de crédito (si aplica), productos adquiridos con sus cantidades y precios, subtotal, descuentos aplicados y el total de cada factura. Además, incluye botones para ver detalles completos u ocultar información de cada factura.



##### Figura 39. Formulario de Nueva Venta a Crédito

La Figura 39 muestra el formulario completo para registrar una nueva venta a crédito. La interfaz está dividida en pasos claramente definidos: Paso 1 muestra la validación del cliente con su nombre completo, cédula y el monto máximo de crédito disponible; Paso 2 presenta el carrito de compras donde se pueden seleccionar productos de un menú desplegable, especificar cantidades y agregarlos al carrito, mostrando un listado de los productos seleccionados con sus precios y subtotales, junto con un botón para calcular el total y proceder al paso siguiente; finalmente, la interfaz confirma que el monto de la venta es menor o igual al límite de crédito, permitiendo proceder con la transacción. Esta interfaz integra la funcionalidad completa del proceso de venta desde la selección de productos hasta la confirmación final.

## 8.3. Integración con los Servicios SOAP

El proyecto cliente web ASP.NET establece conexiones con los servicios SOAP implementados previamente (BanQuito y Comercializadora) mediante referencias web. Las principales integraciones incluyen:

Validación de Cliente: Antes de procesar una venta a crédito, el sistema consulta al servicio BanQuito para validar la existencia del cliente mediante su cédula y obtener su monto máximo de crédito disponible.

Registro de Créditos: Al completar una venta a crédito, el sistema invoca al servicio BanQuito para registrar el nuevo crédito y generar automáticamente la tabla de amortización correspondiente.

Gestión de Facturas: El servicio Comercializadora es utilizado para crear nuevas facturas, consultar facturas existentes por número o por cliente, y gestionar el catálogo de productos.

Todas estas operaciones se realizan mediante llamadas síncronas a los métodos web expuestos por los servicios SOAP, garantizando la consistencia de los datos entre los diferentes componentes del sistema.

# 9. PROYECTO CLIENTE CONSOLA

El proyecto cliente consola es una aplicación de línea de comandos desarrollada en C# .NET que proporciona una interfaz alternativa para interactuar con los servicios SOAP del sistema. Esta aplicación permite realizar todas las operaciones de gestión de productos, facturación y consultas de manera rápida y eficiente mediante menús de texto interactivos.

## 9.1. Funcionalidades Implementadas

La aplicación de consola implementa las siguientes funcionalidades principales:

Gestión de Productos: Listar, crear, buscar, actualizar y eliminar productos del catálogo.

Facturación: Crear ventas en efectivo y ventas a crédito con aplicación automática de descuentos.

Consultas: Buscar facturas por número o por cédula de cliente, consultar tablas de amortización.

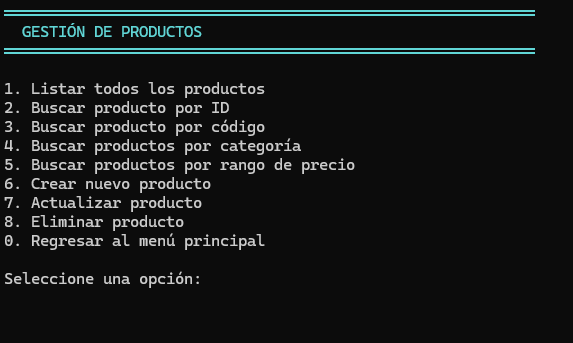
Menús Interactivos: Navegación mediante opciones numéricas con validaciones de entrada.

## 9.2. Capturas de Pantalla del Sistema de Consola

A continuación se presentan las interfaces principales del sistema de consola:

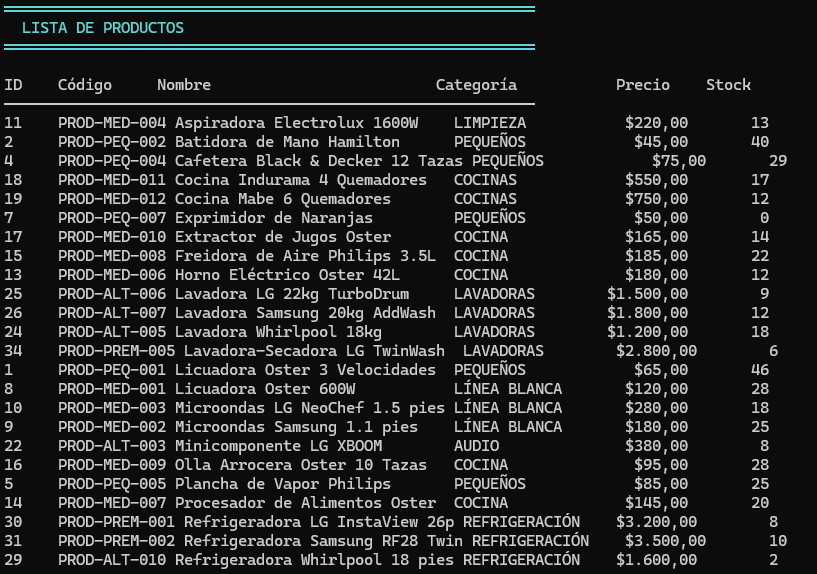
##### Figura 40. Menú Principal de la Aplicación de Consola

La Figura 40 muestra el menú principal del sistema de consola "COMERCIALIZADORA DE ELECTRODOMÉSTICOS - CLIENTE CONSOLA". Este menú presenta las tres opciones principales del sistema: (1) Gestión de Productos, (2) Facturación, (3) Consultas, y (0) Salir. El usuario selecciona una opción ingresando el número correspondiente, lo que permite navegar a los submódulos específicos del sistema.



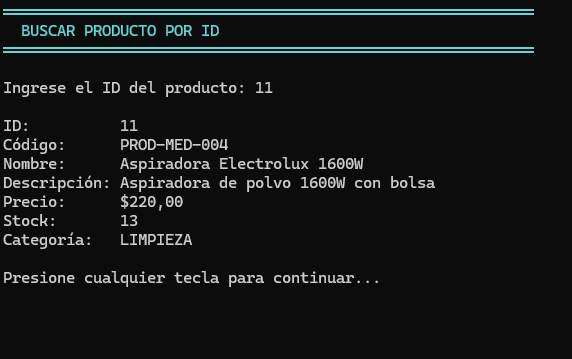
##### Figura 41. Menú de Gestión de Productos

La Figura 41 presenta el submenú de "GESTIÓN DE PRODUCTOS" que incluye ocho opciones: (1) Listar todos los productos, (2) Buscar producto por ID, (3) Buscar producto por código, (4) Buscar productos por categoría, (5) Buscar productos por rango de precio, (6) Crear nuevo producto, (7) Actualizar producto, (8) Eliminar producto, y (0) Regresar al menú principal. Este menú permite realizar todas las operaciones CRUD sobre el catálogo de productos.



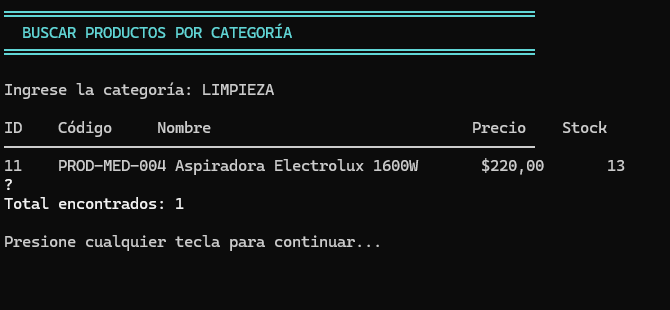
##### Figura 42. Listado Completo de Productos

La Figura 42 muestra el resultado de la opción "LISTA DE PRODUCTOS", desplegando una tabla con todos los productos del catálogo. La tabla incluye las columnas: ID, Código, Nombre, Categoría, Precio y Stock. Esta vista permite al usuario ver de un vistazo toda la información del inventario disponible, facilitando la toma de decisiones sobre qué productos gestionar o incluir en las ventas.



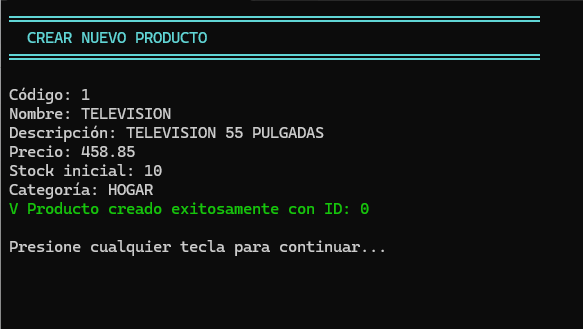
##### Figura 43. Búsqueda de Producto por ID

La Figura 43 presenta la funcionalidad "BUSCAR PRODUCTO POR ID". El usuario ingresa el ID del producto (por ejemplo, 11) y el sistema despliega toda la información detallada del producto incluyendo: ID, Código, Nombre, Descripción, Precio, Stock y Categoría. Esta búsqueda permite obtener rápidamente los detalles completos de un producto específico cuando se conoce su identificador único.



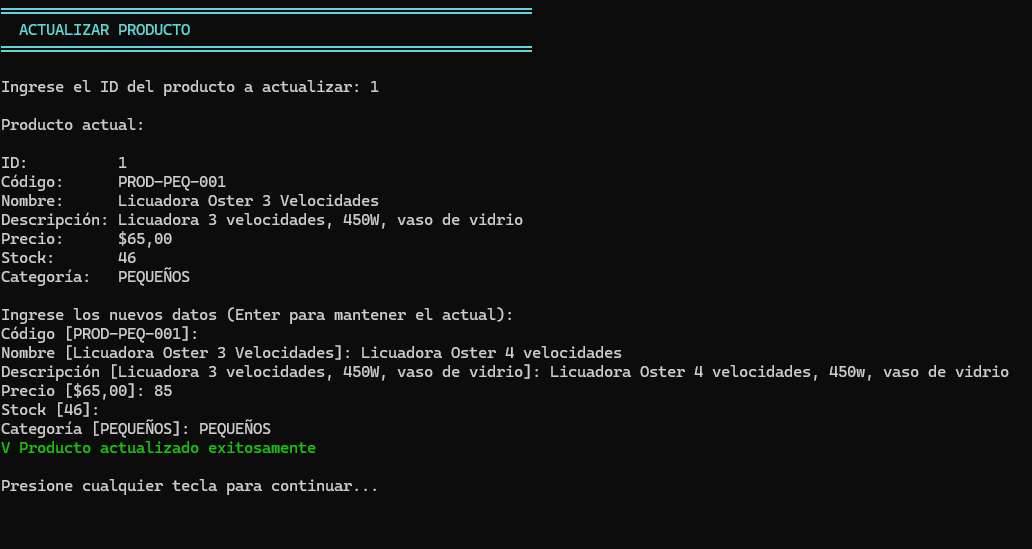
##### Figura 44. Búsqueda de Productos por Categoría

La Figura 44 ilustra la funcionalidad "BUSCAR PRODUCTOS POR CATEGORÍA". El usuario ingresa una categoría (por ejemplo, "LIMPIEZA") y el sistema muestra una tabla filtrada con todos los productos que pertenecen a esa categoría, indicando el total de productos encontrados. La tabla muestra ID, Código, Nombre, Precio y Stock de los productos que coinciden con el criterio de búsqueda.



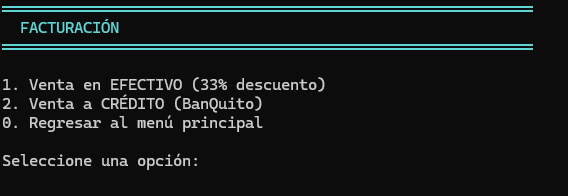
##### Figura 45. Creación de Nuevo Producto

La Figura 45 muestra el formulario "CREAR NUEVO PRODUCTO" donde el usuario ingresa los datos del producto: Código, Nombre, Descripción, Precio, Stock inicial y Categoría. Al completar el proceso, el sistema confirma con el mensaje "V Producto creado exitosamente con ID: 0", indicando que el producto ha sido registrado correctamente en la base de datos.



##### Figura 46. Actualización de Producto Existente

La Figura 46 presenta la funcionalidad "ACTUALIZAR PRODUCTO". El sistema primero muestra los datos actuales del producto seleccionado (por ejemplo, ID: 1, "Licuadora Oster 3 Velocidades") y luego permite al usuario ingresar los nuevos datos, manteniendo los valores actuales si se presiona Enter sin ingresar nada. Al finalizar, el sistema confirma "V Producto actualizado exitosamente", indicando que los cambios se han guardado correctamente.



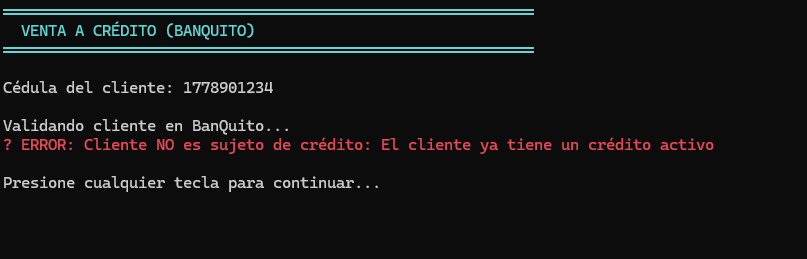
##### Figura 47. Menú de Facturación

La Figura 47 muestra el submenú de "FACTURACIÓN" que ofrece tres opciones: (1) Venta en EFECTIVO (33% descuento), (2) Venta a CRÉDITO (BanQuito), y (0) Regresar al menú principal. Este menú permite al usuario seleccionar el tipo de venta que desea realizar, aplicando automáticamente el descuento del 33% para ventas en efectivo o gestionando el crédito a través del servicio BanQuito.



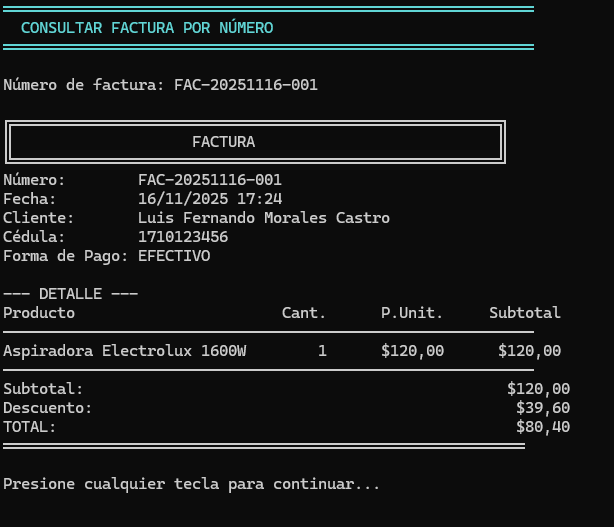
##### Figura 48. Proceso de Venta en Efectivo con Descuento

La Figura 48 muestra el proceso completo de una "VENTA EN EFECTIVO (33% DESCUENTO)". El usuario ingresa la cédula del cliente, nombre, y luego va agregando productos por ID con sus cantidades. El sistema muestra el descuento del 33% aplicado automáticamente y genera la factura final mostrando todos los detalles: número de factura, fecha, cliente, forma de pago (EFECTIVO), desglose de productos, subtotal, descuento aplicado y total final. El mensaje "V ¡Factura generada exitosamente!" confirma la operación exitosa.



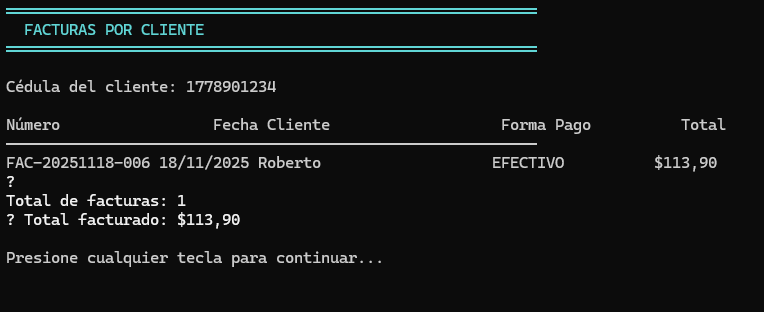
##### Figura 49. Error en Venta a Crédito - Cliente con Crédito Activo

La Figura 49 presenta un caso de validación en "VENTA A CRÉDITO (BANQUITO)". Al intentar procesar una venta a crédito para un cliente, el sistema consulta al servicio BanQuito para validar la elegibilidad. En este caso, el sistema muestra el mensaje de error "? ERROR: Cliente NO es sujeto de crédito: El cliente ya tiene un crédito activo", impidiendo que se procese la venta y protegiendo así las reglas de negocio del sistema de créditos.



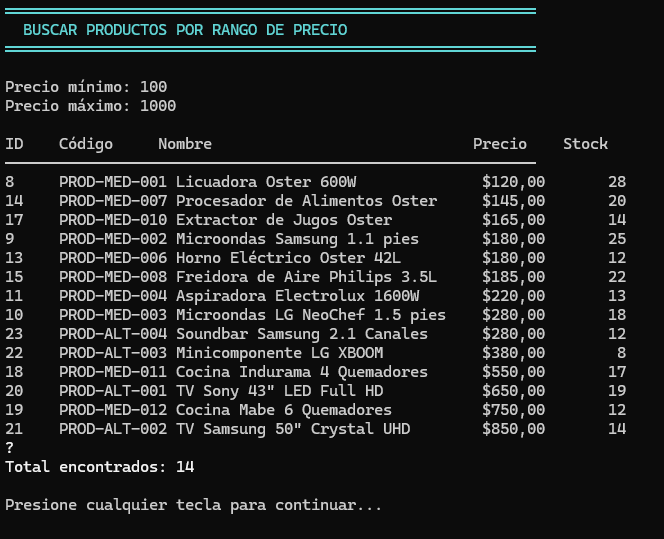
##### Figura 50. Consulta de Factura por Número

La Figura 50 muestra la funcionalidad "CONSULTAR FACTURA POR NÚMERO". El usuario ingresa el número de factura (por ejemplo, "FAC-20251116-001") y el sistema despliega todos los detalles: número, fecha, cliente (nombre y cédula), forma de pago, desglose de productos con cantidades y precios unitarios, subtotal, descuento aplicado y total. Esta consulta permite verificar rápidamente cualquier transacción previamente registrada.



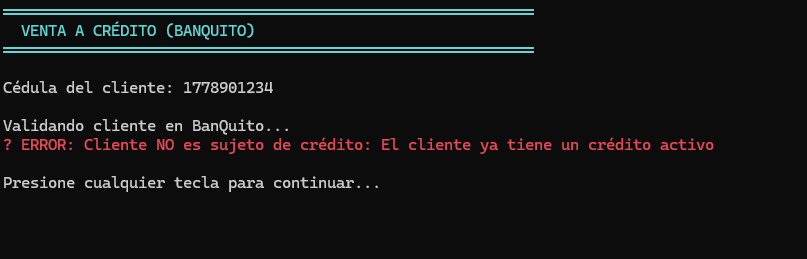
##### Figura 51. Consulta de Facturas por Cliente

La Figura 51 presenta la consulta "FACTURAS POR CLIENTE" donde el usuario ingresa una cédula (por ejemplo, "1778901234") para obtener un resumen de todas las facturas asociadas a ese cliente. El sistema muestra: Número de factura, Fecha, Cliente, Forma de Pago y Total. Además, indica el total de facturas encontradas y el total facturado acumulado para ese cliente específico.



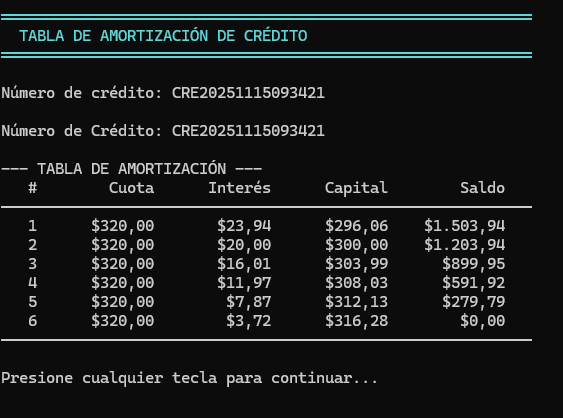
##### Figura 52. Consulta de Facturas por Rango de Precio

La Figura 52 muestra la funcionalidad "FACTURAS POR RANGO DE PRECIO" donde el usuario ingresa una fecha específica (por ejemplo, "100 a 1000") y el sistema recupera todas las transacciones realizadas en ese rango. La consulta despliega un listado completo con el número de factura, cliente asociado, forma de pago y el monto total de cada transacción. Esta función es útil para generar reportes diarios y análisis de ventas por período.



##### Figura 53. Consulta de Facturas de Ventas a Crédito

La Figura 53 presenta la consulta "FACTURAS DE VENTAS A CRÉDITO" que filtra y muestra únicamente las transacciones realizadas mediante el sistema BanQuito. El listado incluye el número de factura, identificación del cliente, número de crédito asignado, fecha de la transacción y el monto total financiado. Esta consulta permite dar seguimiento específico a las operaciones de crédito y facilita la conciliación con el servicio bancario.



##### Figura 54. Tabla de amortización

La Figura 54 muestra la funcionalidad de visualizar la tabla de amortización de clientes por número de crédito. El usuario ingresa el número de crédito (por ejemplo, "CRE20251115093421") y el sistema recupera y despliega toda la información asociada: cuota, interés, capital y saldo. Esta consulta es esencial para verificar datos de clientes durante el proceso de facturación o actualización de información.

## 9.3. Integración con los Servicios SOAP

La aplicación de consola interactúa directamente con los servicios web SOAP desarrollados en .NET para ejecutar todas las operaciones del sistema. Cada funcionalidad del menú (gestión de productos, facturación, consultas, gestión de clientes y reportes) invoca métodos específicos del servicio web, el cual procesa las solicitudes y retorna la información desde la base de datos local. Este cliente de consola demuestra cómo consumir servicios SOAP desde una aplicación .NET, implementando toda la lógica de negocio de forma remota y centralizada.

# 10. PROYECTO CLIENTE ESCRITORIO

El proyecto cliente de escritorio es una aplicación Windows Forms desarrollada en C# .NET que proporciona una interfaz gráfica de usuario (GUI) para interactuar con los servicios SOAP del sistema. Esta aplicación ofrece las mismas funcionalidades que el cliente web, pero optimizada para entornos de escritorio con controles nativos de Windows, permitiendo una experiencia de usuario más fluida y responsive en computadoras de escritorio.

## 10.1. Funcionalidades Principales

La aplicación de escritorio implementa las siguientes funcionalidades principales:

Gestión de Productos: Interfaz gráfica para listar, buscar, crear y actualizar productos del catálogo.

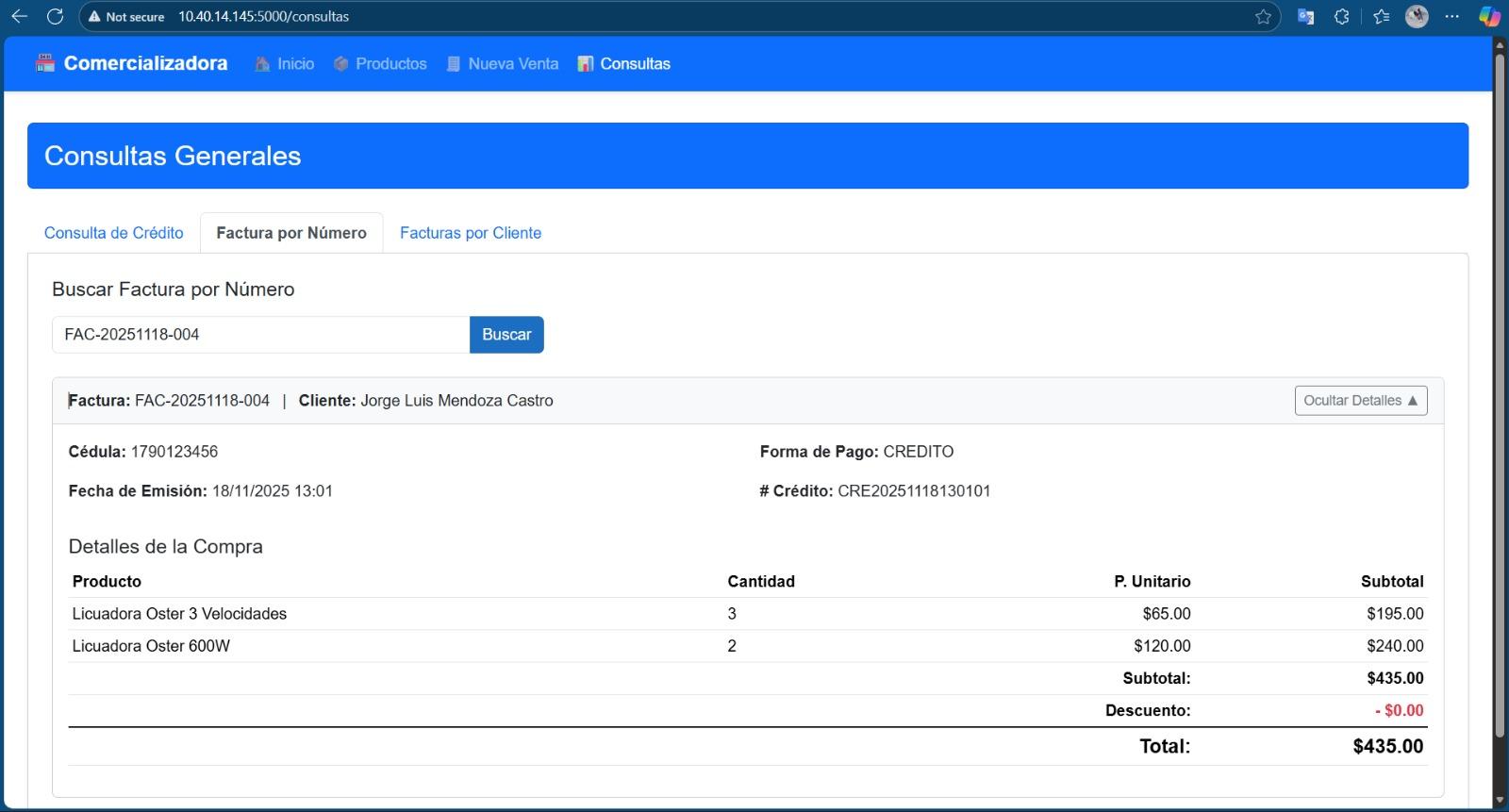
Facturación: Módulos para crear ventas en efectivo con descuento automático del 33% y ventas a crédito mediante integración con BanQuito.

Consultas: Visualización de facturas por número, por cliente y tablas de amortización de créditos.

Interfaz Windows Forms: Diseño nativo de Windows con controles como DataGridView, TextBox, Button y ComboBox para una experiencia de usuario optimizada.

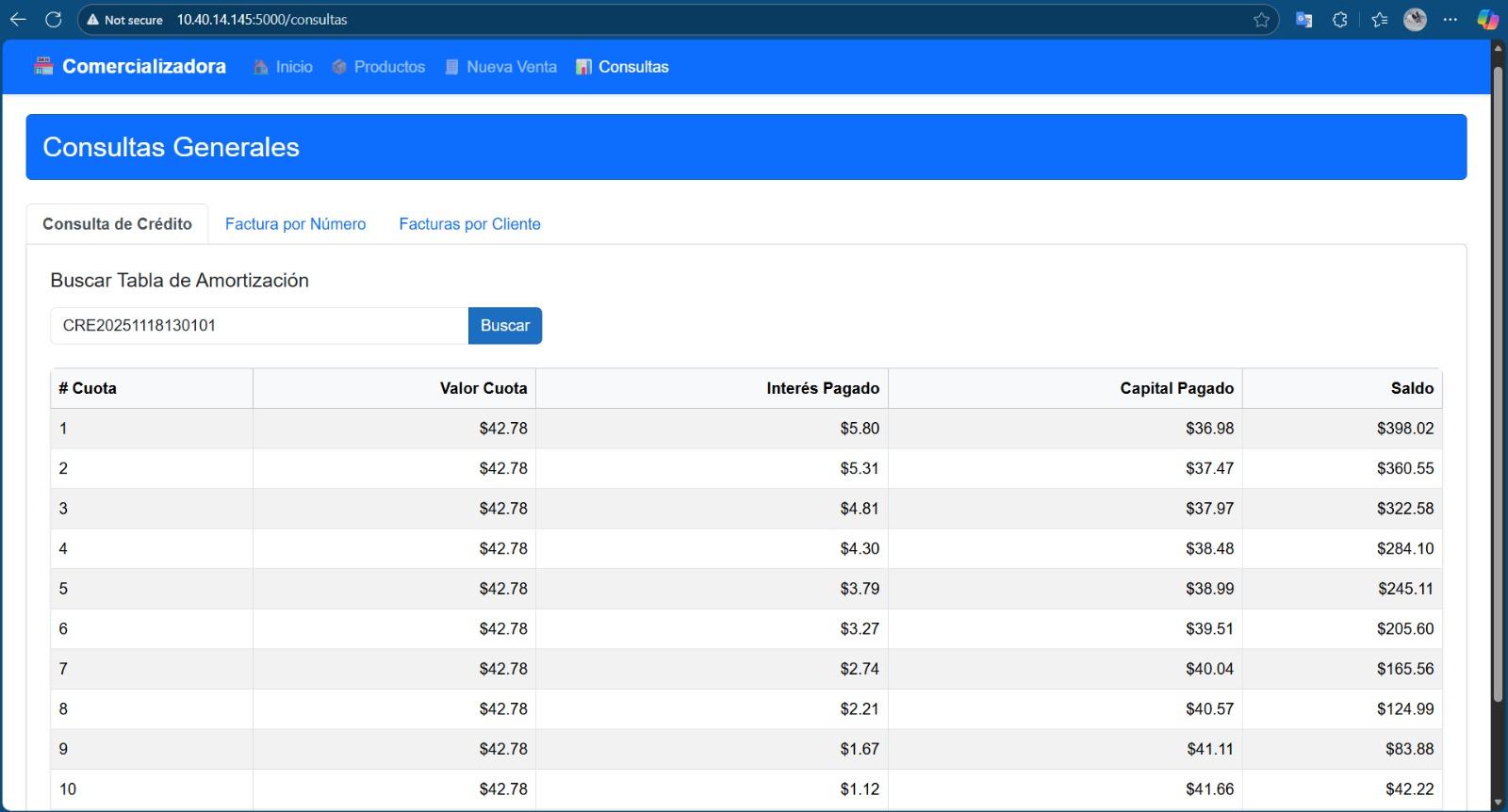
## 10.2. Capturas de Pantalla del Sistema de Escritorio

A continuación se presentan las interfaces principales del sistema de escritorio:



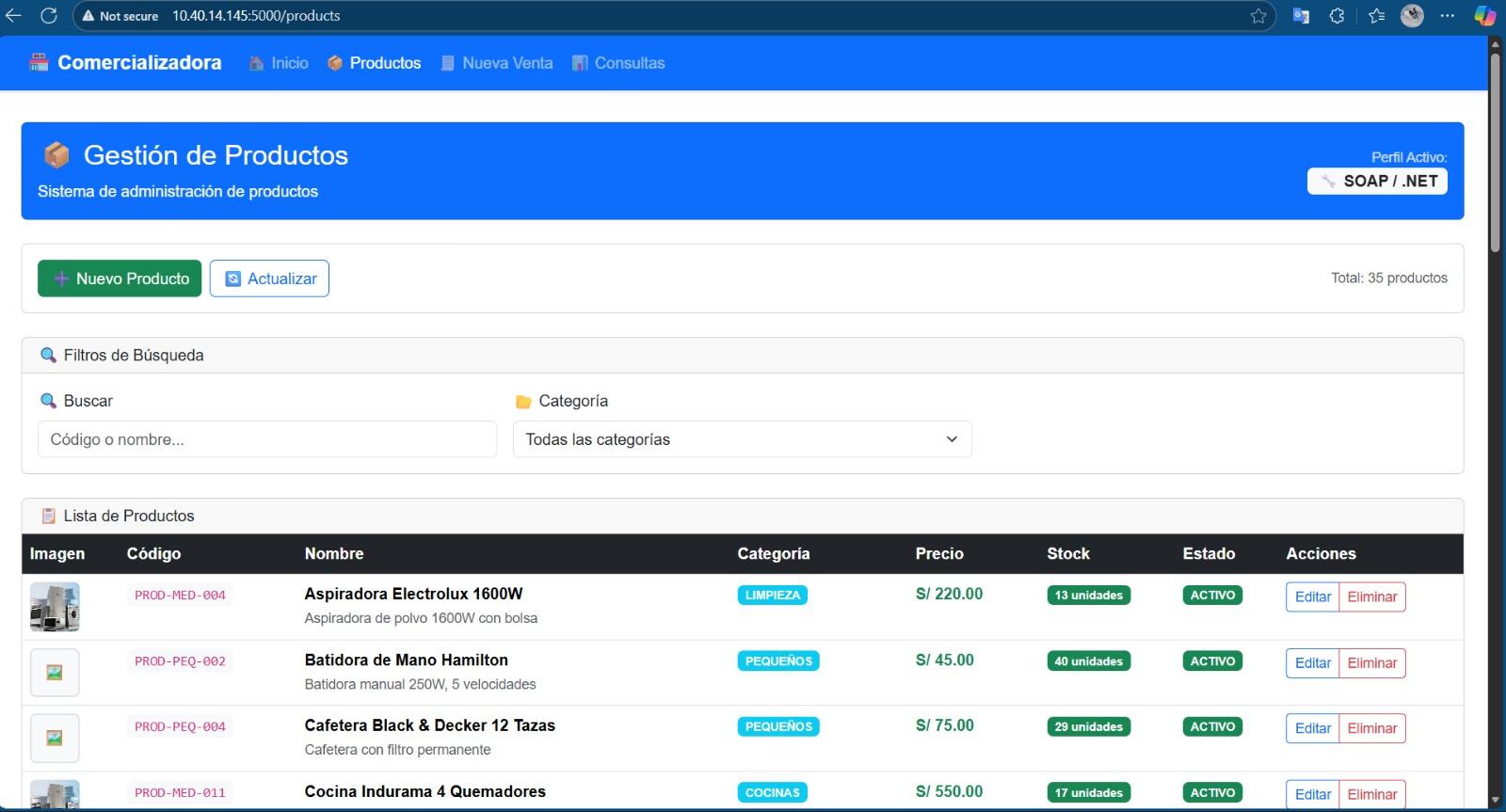
##### Figura 63. Consulta de Factura por Número

La Figura 63 muestra la interfaz de consulta de facturas por número en la aplicación de escritorio. El usuario ingresa el número de factura (por ejemplo, FAC-20251118-004) en un campo de texto y presiona el botón de búsqueda. El sistema despliega en un DataGridView la información completa incluyendo: datos del cliente (cédula, nombre completo), detalles de la transacción (fecha de emisión, forma de pago, número de crédito), y el desglose de productos adquiridos con sus cantidades, precios unitarios y subtotales. Al final se presenta el total de la factura en un panel de resumen.



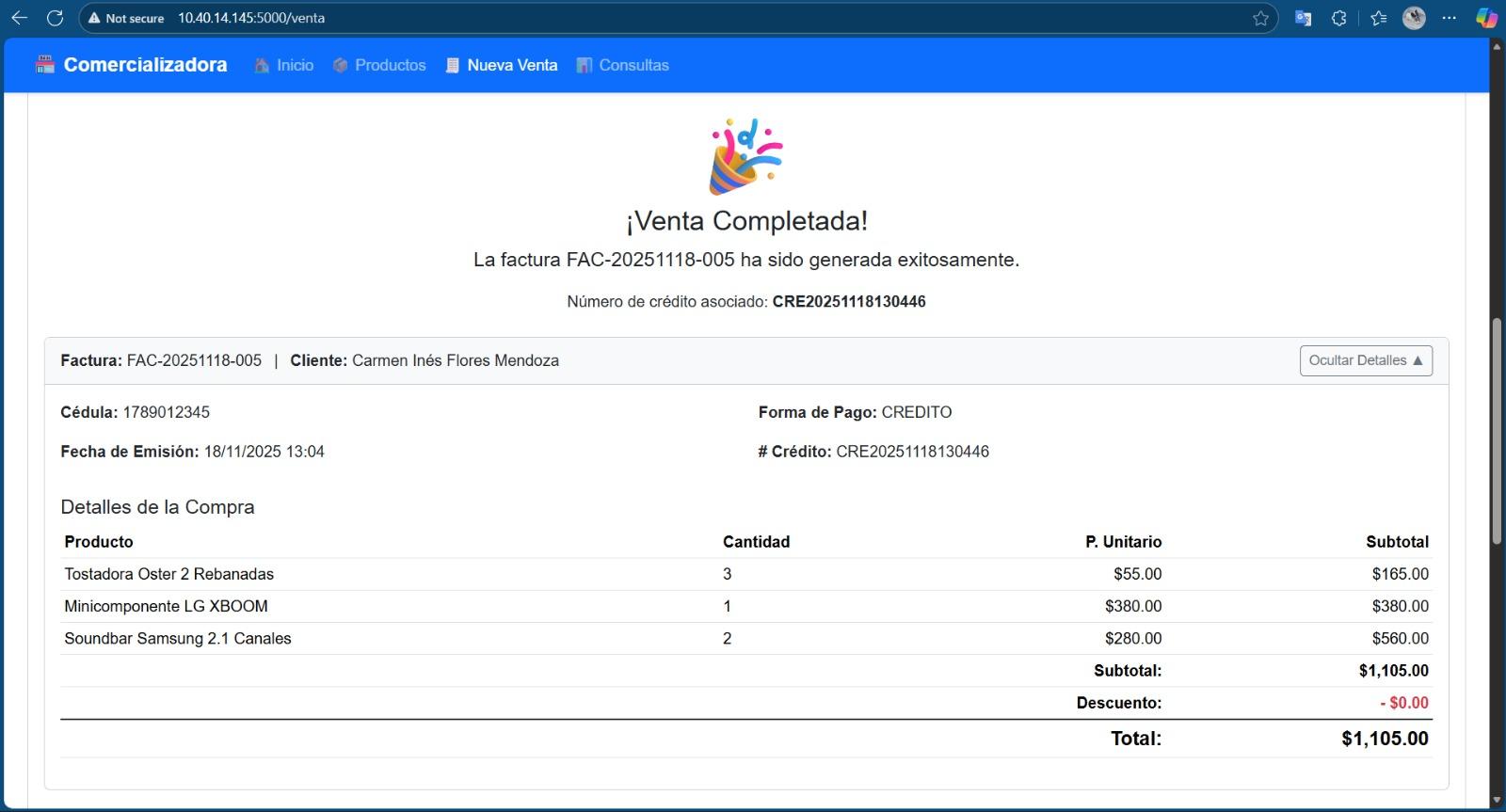
##### Figura 64. Tabla de Amortización del Crédito

La Figura 64 presenta la ventana de visualización de la tabla de amortización para créditos en la aplicación de escritorio. El usuario ingresa el número de crédito (por ejemplo, "CRE20251115093421") en un TextBox y el sistema recupera y despliega en un DataGridView toda la información del plan de pagos: cuota, interés, capital y saldo. Los datos se presentan en formato tabular con columnas claramente definidas y con opciones de ordenamiento. Esta interfaz permite a los usuarios verificar rápidamente el estado de amortización de cualquier crédito.



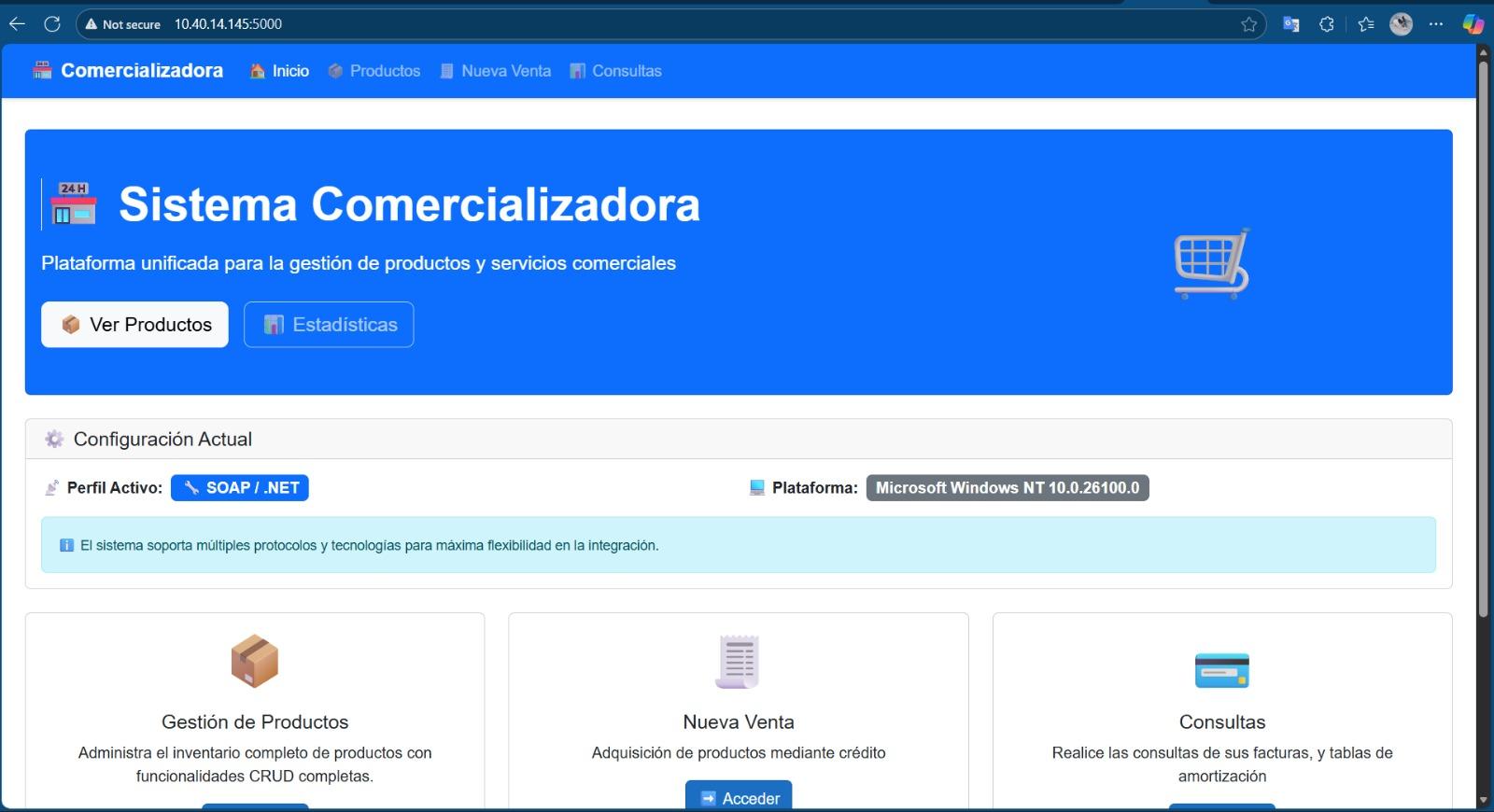
##### Figura 65. Gestión de Productos

La Figura 65 muestra el formulario de gestión de productos en la aplicación Windows Forms. La interfaz incluye un DataGridView principal que lista todos los productos disponibles con columnas para ID, Código, Nombre, Descripción, Precio, Stock y Categoría. En la parte superior hay controles de búsqueda y filtrado, y botones para las operaciones CRUD: "Nuevo Producto", "Editar", "Eliminar" y "Actualizar". Al seleccionar un producto de la grilla, sus datos se cargan automáticamente en los campos de edición en el panel lateral.



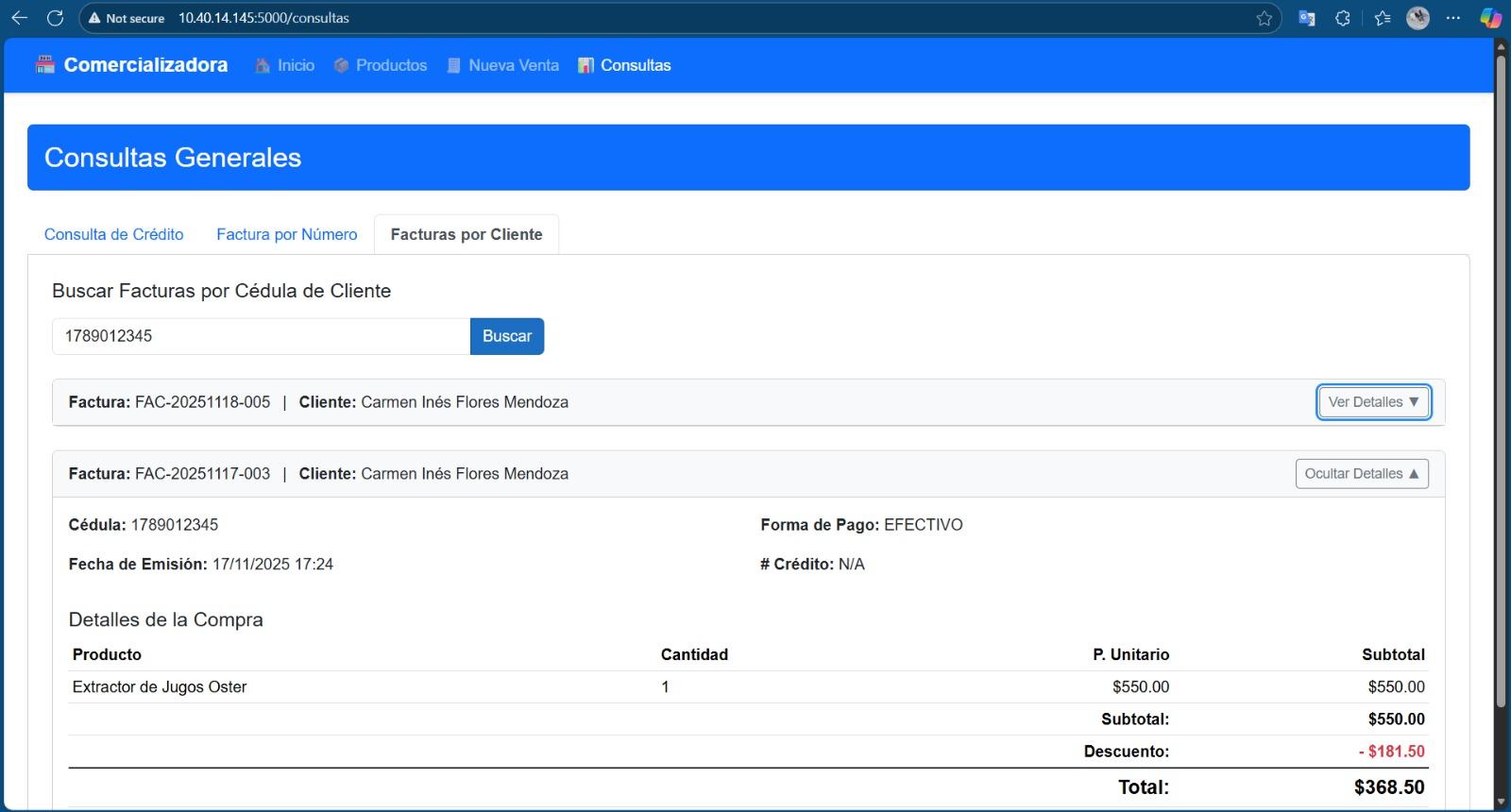
##### Figura 66. Confirmación de Venta Completada

La Figura 66 presenta el cuadro de diálogo de confirmación que aparece después de completar exitosamente una venta. El mensaje muestra "Venta realizada con éxito" junto con el número de factura generado (por ejemplo, "FAC-20251118-005"), el total de la venta, la forma de pago utilizada (EFECTIVO o CRÉDITO), y el descuento aplicado. El diálogo incluye botones para "Imprimir Factura", "Nueva Venta" o "Cerrar". Esta ventana proporciona retroalimentación inmediata al usuario sobre la transacción realizada.



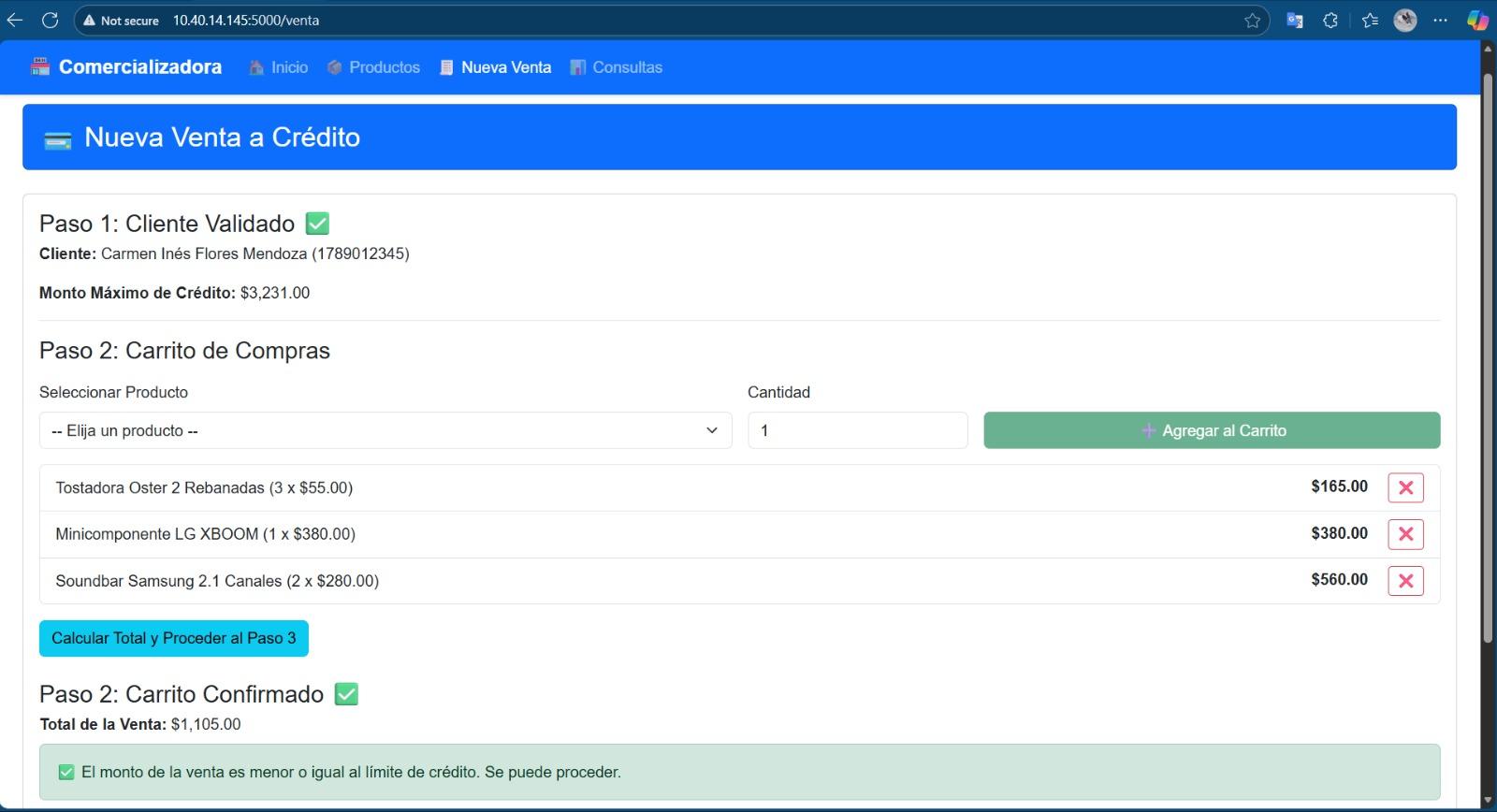
##### Figura 67. Página Principal del Sistema

La Figura 67 muestra la ventana principal (MDI - Multiple Document Interface) de la aplicación de escritorio. La interfaz incluye una barra de menús superior con las opciones principales: "Productos", "Facturación", "Consultas", "Clientes" y "Ayuda". También cuenta con una barra de herramientas con botones de acceso rápido representados por iconos para las funciones más utilizadas. El área de trabajo central puede contener múltiples ventanas secundarias abiertas simultáneamente. En la parte inferior se encuentra una barra de estado que muestra información del usuario conectado, fecha/hora actual y estado de conexión con el servidor SOAP.



##### Figura 68. Consulta de Facturas por Cliente

La Figura 68 presenta el formulario de consulta de facturas filtradas por cliente. El usuario ingresa la cédula del cliente (por ejemplo, "1778901234") en un campo de búsqueda y el sistema despliega en un DataGridView todas las facturas asociadas a ese cliente. La grilla muestra: Número de factura, Fecha, Cliente (nombre y cédula), Forma de Pago, Total y Estado. Los registros se pueden ordenar por cualquier columna haciendo clic en el encabezado. La interfaz incluye botones para "Exportar a Excel", "Imprimir Reporte" y "Ver Detalle" de la factura seleccionada.



##### Figura 69. Formulario de Nueva Venta a Crédito

La Figura 69 muestra el formulario completo para procesar una nueva venta a crédito en la aplicación de escritorio. La interfaz está dividida en tres secciones principales: Paso 1 - Selección de cliente donde se valida la elegibilidad del cliente ingresando su cédula; Paso 2 - Carrito de compras con un DataGridView donde se agregan productos seleccionándolos de un ComboBox, especificando cantidades y visualizando subtotales; Paso 3 - Confirmación y pago donde se muestra el total, se verifica que el monto no exceda el límite de crédito del cliente, y se procesa la transacción generando automáticamente el crédito en BanQuito con su tabla de amortización. La interfaz incluye validaciones en tiempo real y mensajes informativos en cada paso.

## 10.3. Integración con los Servicios SOAP

La aplicación de escritorio Windows Forms consume los servicios web SOAP mediante referencias de servicio agregadas al proyecto. Cada formulario instancia los clientes del servicio (BanQuitoClient y ComercializadoraClient) para invocar los métodos remotos correspondientes. Las principales integraciones incluyen:

Validación de Cliente: Antes de procesar una venta a crédito, la aplicación consulta al servicio BanQuito para validar la existencia del cliente mediante su cédula y obtener su monto máximo de crédito disponible.

Registro de Créditos: Al completar una venta a crédito, la aplicación invoca al servicio BanQuito para registrar el nuevo crédito y generar automáticamente la tabla de amortización correspondiente.

Gestión de Facturas: El servicio Comercializadora es utilizado para crear nuevas facturas, consultar facturas existentes por número o por cliente, y gestionar el catálogo de productos.

Manejo de Errores: La aplicación implementa try-catch para capturar excepciones de los servicios SOAP y mostrar mensajes amigables al usuario mediante cuadros de diálogo (MessageBox).

Todas estas operaciones se realizan de forma asíncrona cuando es posible para mantener la interfaz gráfica responsive, utilizando patrones async/await de C# para evitar bloqueos de la UI durante las llamadas a servicios web.

# 11. PROYECTO CLIENTE MÓVIL

El proyecto cliente móvil es una aplicación Android desarrollada con .NET MAUI (Multi-platform App UI) que proporciona acceso completo al sistema de comercialización de electrodomésticos desde dispositivos móviles. Esta aplicación permite a los usuarios realizar todas las operaciones de gestión de productos, facturación y consultas directamente desde smartphones y tablets Android, consumiendo los mismos servicios SOAP que los demás clientes del sistema.

## 11.1. Funcionalidades Principales

La aplicación móvil Android implementa las siguientes funcionalidades principales:

Gestión de Productos: Interfaz táctil optimizada para listar, buscar y administrar el catálogo completo de productos.

Nueva Venta a Crédito: Proceso completo de venta en múltiples pasos con validación de cliente, selección de productos y generación automática de crédito mediante BanQuito.

Consultas Móviles: Acceso rápido a facturas por número, por cliente y visualización de tablas de amortización de créditos.

Interfaz Adaptativa: Diseño responsive que se ajusta a diferentes tamaños de pantalla con navegación mediante CardViews, RecyclerViews y controles nativos de Android.

Configuración de Perfil: Panel de configuración que permite seleccionar entre diferentes protocolos de comunicación (SOAP/.NET) y visualizar la plataforma actual.

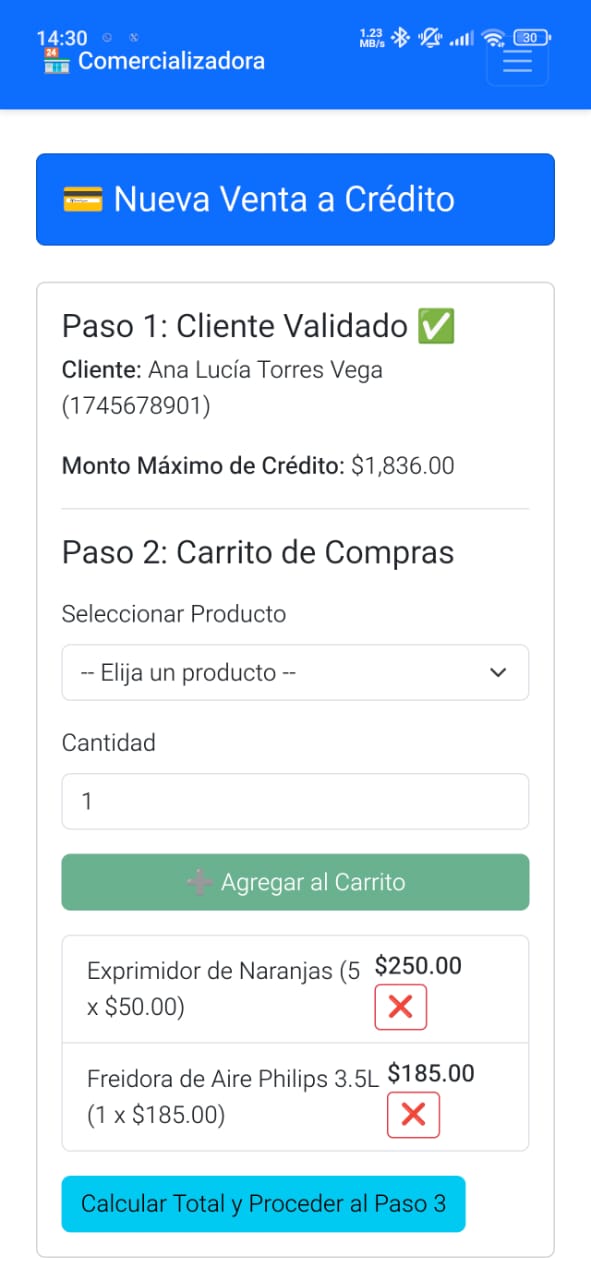
## 11.2. Capturas de Pantalla del Sistema Móvil Android

A continuación se presentan las interfaces principales del sistema móvil:



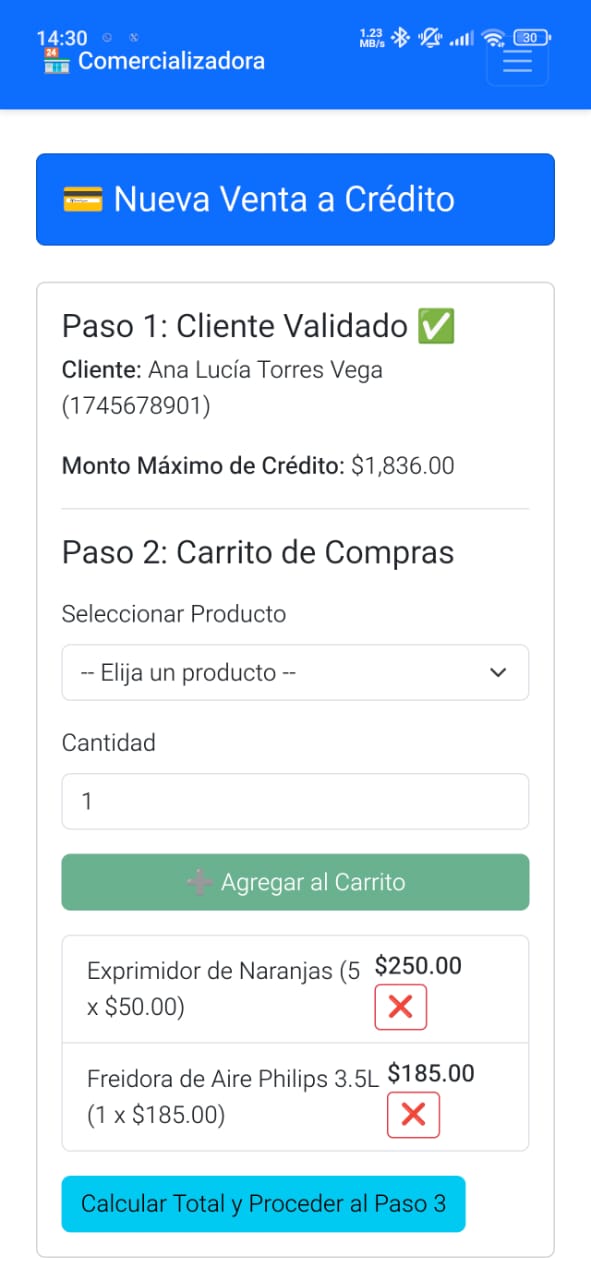
##### Figura 70. Pantalla Principal y Configuración

La Figura 70 muestra la pantalla principal de la aplicación móvil "Comercializadora" con el panel de "Configuración Actual" en la parte superior. Este panel indica el "Perfil Activo" configurado como "SOAP / .NET" con un icono de herramientas, y la "Plataforma" detectada como "Android - 15". Debajo aparece un mensaje informativo en color celeste explicando que el sistema soporta múltiples protocolos y tecnologías para máxima flexibilidad. La interfaz principal presenta dos tarjetas (cards) principales: "Gestión de Productos" con un icono de caja y descripción de funcionalidades CRUD, y "Nueva Venta" con un icono de factura para adquisición de productos mediante crédito, ambas con botones "Acceder" en azul.



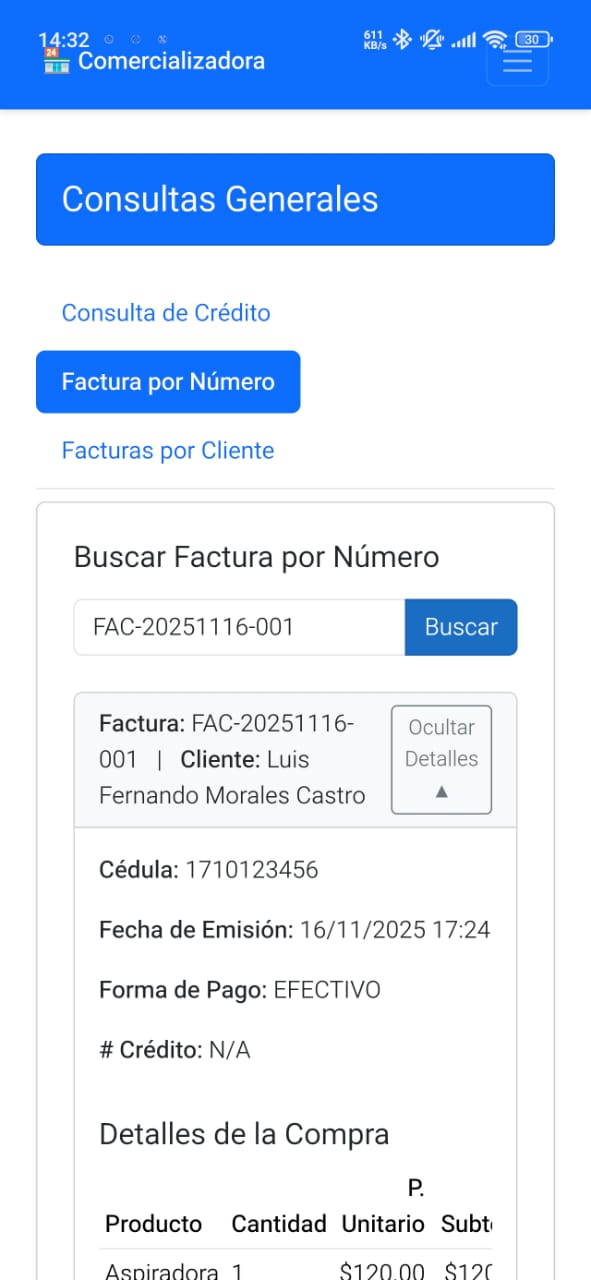
##### Figura 71. Nueva Venta - Paso 1: Cliente Validado

La Figura 71 muestra el proceso de "Nueva Venta a Crédito" en el Paso 1 con el encabezado en azul y un icono de tarjeta de crédito. La interfaz muestra "Paso 1: Cliente Validado" con una marca de verificación verde, indicando que la cliente "Ana Lucía Torres Vega (1745678901)" ha sido validada exitosamente. El sistema despliega el "Monto Máximo de Crédito: $1,836.00" disponible para esta cliente. A continuación aparece el "Paso 2: Carrito de Compras" con un dropdown para "Seleccionar Producto" (mostrando "-- Elija un producto --"), un campo de "Cantidad" con valor 1, y un botón verde "Agregar al Carrito". Debajo se visualiza el carrito con dos productos agregados: "Exprimidor de Naranjas (5 x $50.00)" por $250.00 y "Freidora de Aire Philips 3.5L (1 x $185.00)" por $185.00, cada uno con un botón rojo "X" para eliminar. Al final hay un botón celeste "Calcular Total y Proceder al Paso 3".



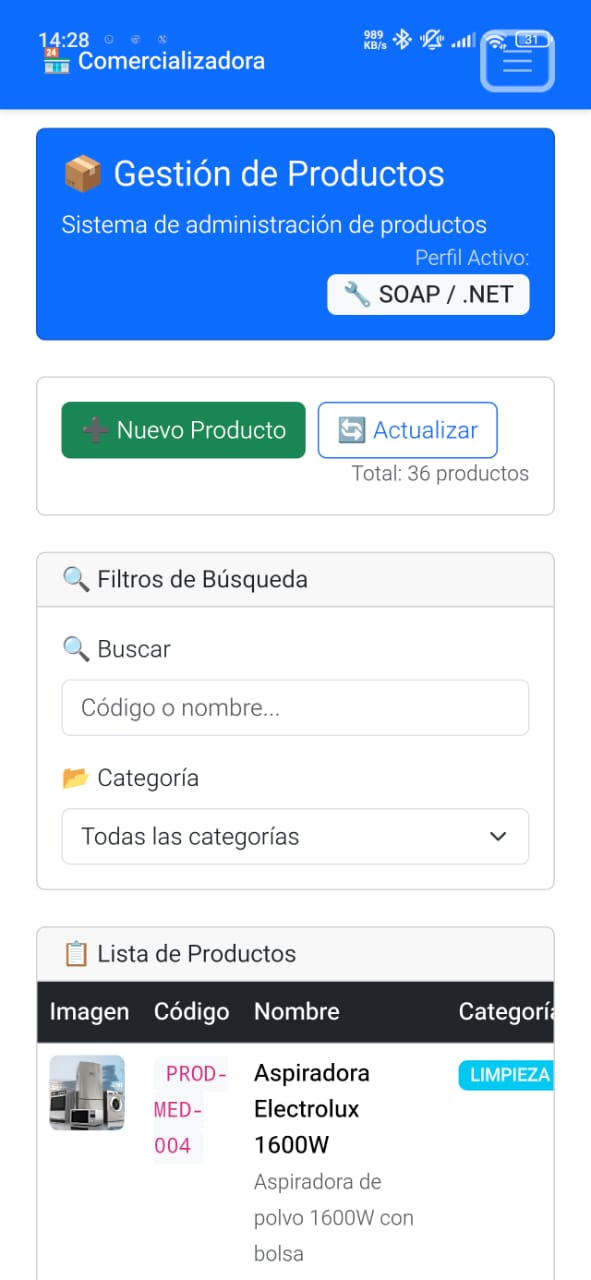
##### Figura 72. Nueva Venta - Paso 2: Carrito Confirmado

La Figura 72 presenta el estado del "Paso 2: Carrito Confirmado" con una marca de verificación verde. La pantalla muestra "Total de la Venta: $935.00" en texto destacado. Debajo aparece un mensaje de validación en un panel verde con texto: "El monto de la venta es menor o igual al límite de crédito. Se puede proceder." A continuación se presenta el "Paso 3: Otorgar Crédito y Facturar" con un campo de texto para ingresar el "Número de Cuotas (3-24)" mostrando el valor "12". Finalmente se encuentra un botón rojo grande con el texto "FINALIZAR VENTA Y OTORGAR CRÉDITO" para completar la transacción.



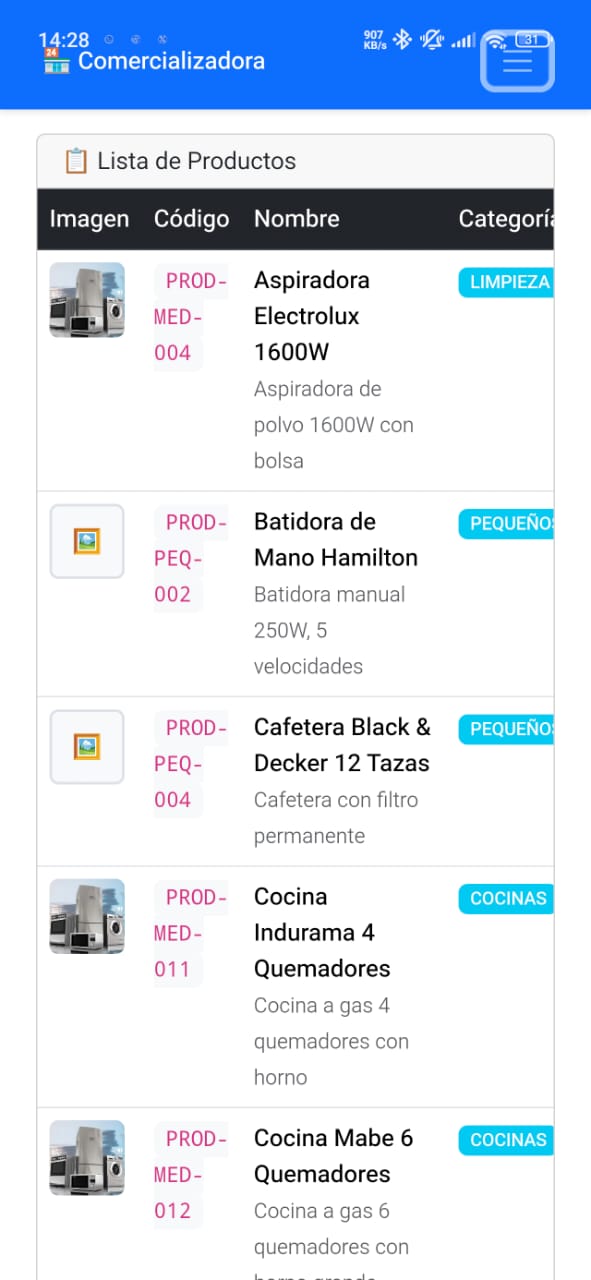
##### Figura 73. Consultas Generales - Factura por Número

La Figura 73 muestra el módulo de "Consultas Generales" con un encabezado azul. La pantalla presenta tres opciones de consulta: "Consulta de Crédito" en texto azul, "Factura por Número" en un botón azul destacado, y "Facturas por Cliente" en texto azul. En la sección principal aparece el formulario "Buscar Factura por Número" con un campo de texto mostrando "FAC-20251116-001" y un botón azul "Buscar" a la derecha. Debajo se despliega el resultado de la búsqueda mostrando: "Factura: FAC-20251116-001 | Cliente: Luis Fernando Morales Castro" con un botón "Ocultar Detalles" que permite expandir/contraer la información. Los detalles adicionales incluyen: "Cédula: 1710123456", "Fecha de Emisión: 16/11/2025 17:24", "Forma de Pago: EFECTIVO", "# Crédito: N/A", y la sección "Detalles de la Compra" con una tabla mostrando Producto, Cantidad, Precio Unitario y Subtotal para cada artículo (por ejemplo, "Aspiradora" con cantidad 1, precio $120.00 y subtotal $120.00).



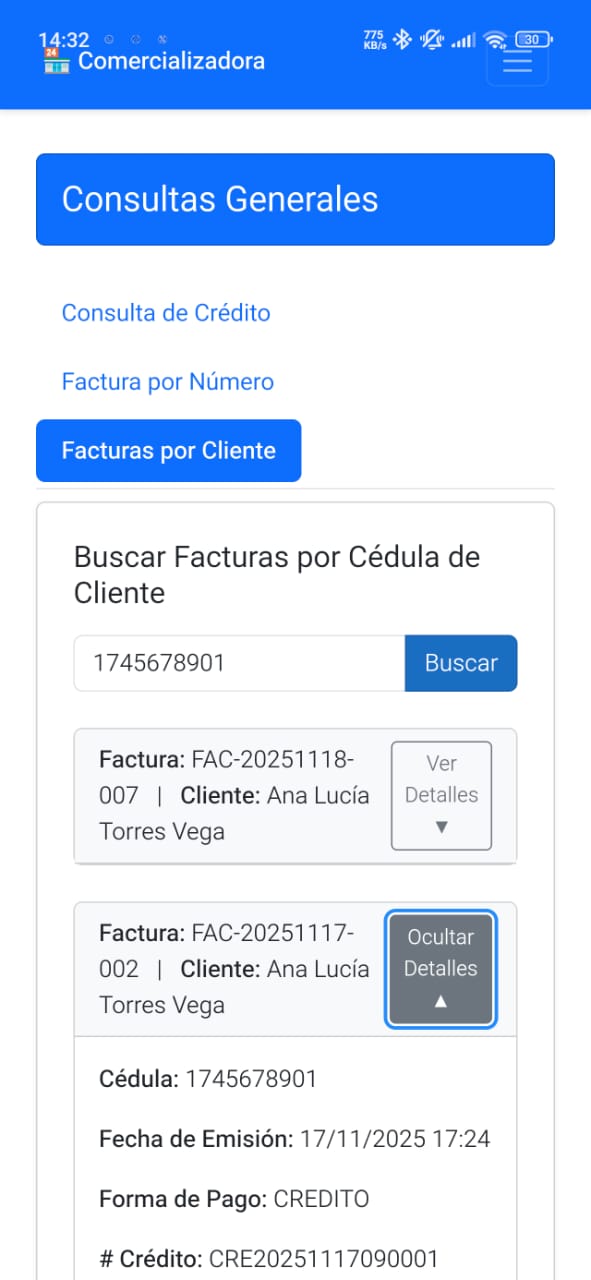
##### Figura 74. Gestión de Productos Móvil

La Figura 74 presenta el módulo de "Gestión de Productos" con un encabezado azul que incluye el icono de una caja y el texto "Sistema de administración de productos". En la esquina superior derecha se muestra "Perfil Activo: SOAP / .NET" con un icono de herramienta. La interfaz incluye dos botones principales: "Nuevo Producto" en verde y "Actualizar" en azul con bordes, mostrando "Total: 36 productos" en texto gris. Debajo aparece la sección "Filtros de Búsqueda" con un icono de lupa, conteniendo un campo de búsqueda con placeholder "Código o nombre..." y un dropdown de "Categoría" configurado en "Todas las categorías". La "Lista de Productos" se presenta en formato de tarjetas con columnas para Imagen, Código, Nombre y Categoría. El primer producto mostrado es "Aspiradora Electrolux 1600W" con código "PROD-MED-004", una imagen del producto a la izquierda, y una etiqueta celeste "LIMPIEZA" indicando su categoría.



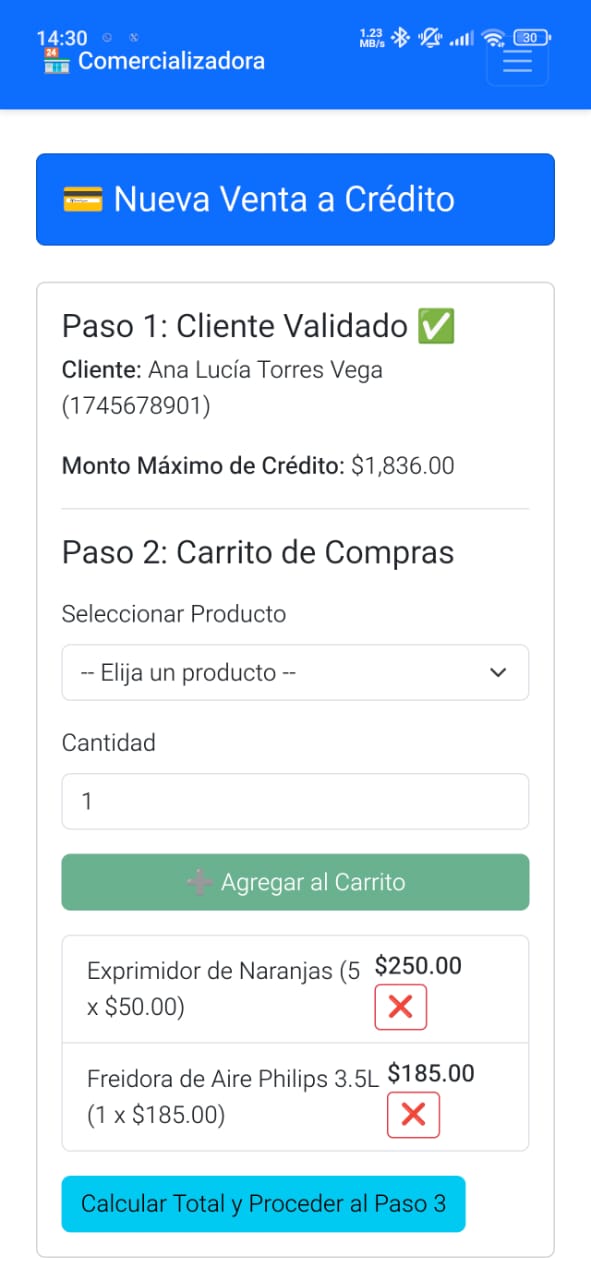
##### Figura 75. Lista de Productos en Tarjetas

La Figura 75 muestra la visualización completa de la lista de productos en formato de tarjetas deslizables (RecyclerView). Cada tarjeta de producto incluye una imagen miniatura a la izquierda, el código del producto en color rosa (por ejemplo, "PROD-MED-004", "PROD-PEQ-002", "PROD-PEQ-004"), el nombre completo del producto en negro (como "Aspiradora Electrolux 1600W", "Batidora de Mano Hamilton", "Cafetera Black & Decker 12 Tazas", "Cocina Indurama 4 Quemadores", "Cocina Mabe 6 Quemadores"), una breve descripción del producto en texto gris, y una etiqueta de categoría en celeste ("LIMPIEZA", "PEQUEÑOS", "COCINAS"). La interfaz permite scroll vertical para navegar por todos los productos del catálogo de manera fluida y optimizada para dispositivos táctiles.



##### Figura 76. Consultas - Facturas por Cliente

La Figura 76 presenta la pantalla de "Consultas Generales" con la opción "Facturas por Cliente" seleccionada (botón azul). El formulario muestra "Buscar Facturas por Cédula de Cliente" con un campo de texto conteniendo "1745678901" y un botón azul "Buscar". Los resultados se despliegan en tarjetas expandibles mostrando múltiples facturas: "Factura: FAC-20251118-007 | Cliente: Ana Lucía Torres Vega" con un botón "Ver Detalles" expandido mostrando información completa (Cédula, Fecha de Emisión, Forma de Pago, # Crédito), y "Factura: FAC-20251117-002 | Cliente: Ana Lucía Torres Vega" con un botón "Ocultar Detalles" que permite contraer la información. Cada factura muestra su cédula (1745678901), fecha de emisión, forma de pago (CRÉDITO) y número de crédito asociado.



##### Figura 77. Nueva Venta - Paso 3: Venta Completada con Crédito

La Figura 77 muestra la pantalla de confirmación después de completar exitosamente una venta a crédito. En la parte superior se visualiza el "Paso 3: Otorgar Crédito y Facturar" con el campo "Número de Cuotas (3-24)" mostrando el valor "6", y el botón rojo "FINALIZAR VENTA Y OTORGAR CRÉDITO". Debajo aparece un mensaje de éxito con un icono de celebración y el texto "¡Venta Completada!" en grande. El mensaje indica: "La factura FAC-20251118-007 ha sido generada exitosamente. Número de crédito asociado: CRE20251118143054". A continuación se muestra un resumen de la factura con: "Factura: FAC-20251118-007 | Cliente: Ana Lucía Torres Vega", un botón "Ocultar Detalles", y la información completa incluyendo Cédula (1745678901), Fecha de Emisión (18/11/2025 14:30), Forma de Pago (CRÉDITO), # Crédito (CRE20251118143054).



##### Figura 78. Detalles de Compra y Totales

La Figura 78 presenta la sección de "Detalles de la Compra" en formato de tabla mostrando cada producto adquirido. La tabla incluye columnas para Producto, Cantidad, Precio Unitario y Subtotal. En este ejemplo se muestran dos productos: "Cocina Mabe 6 Quemadores" (cantidad 1, precio unitario $750.00, subtotal $750.00) y "Freidora de Aire Philips" (cantidad 1, precio unitario $185.00, subtotal $185.00). Al final de la tabla se presenta el resumen financiero con tres líneas: "Subtotal: $935.00", "Descuento: - $0.00" en color rojo, y "Total: $935.00" en texto grande y negrita con una línea superior de separación. Esta sección proporciona transparencia total sobre el monto de la transacción.



##### Figura 79. Consulta de Tabla de Amortización Móvil

La Figura 79 muestra el módulo de "Consultas Generales" con la opción "Consulta de Crédito" seleccionada (botón azul). El formulario presenta "Buscar Tabla de Amortización" con un campo de texto conteniendo el número de crédito "CRE20251115093421" y un botón azul "Buscar". Los resultados se despliegan en una tabla responsive optimizada para móvil con columnas: # Cuota, Valor Cuota, Interés Pagado, Capital Pagado y Saldo. La tabla muestra las primeras 6 cuotas del plan de amortización: Cuota 1 ($320.00, interés $23.94, capital $296.06, saldo $1,503.94), Cuota 2 ($320.00, interés $20.00, capital $300.00, saldo $1,203.94), y así sucesivamente hasta la Cuota 6 ($320.00, interés $3.72, capital $316.28). La tabla permite scroll horizontal para ver todas las columnas en pantallas pequeñas y proporciona una visualización clara del plan de pagos del crédito.

## 11.3. Integración con los Servicios SOAP en Android

La aplicación móvil Android desarrollada con .NET MAUI consume los servicios web SOAP mediante la tecnología HttpClient y serialización XML para comunicarse con los servicios BanQuito y Comercializadora. La arquitectura de la aplicación implementa el patrón Repository para abstraer las llamadas a los servicios SOAP y facilitar el mantenimiento del código.

Las principales características de la integración móvil incluyen:

Cliente SOAP Personalizado: Se implementó una clase SoapClient genérica que construye los envelopes SOAP manualmente, envía las peticiones HTTP POST al endpoint del servicio, y parsea las respuestas XML utilizando XDocument y LINQ to XML.

Manejo Asíncrono: Todas las llamadas a servicios se realizan de forma asíncrona utilizando async/await para mantener la interfaz de usuario responsive y evitar bloqueos durante las operaciones de red, especialmente importantes en dispositivos móviles con conexiones variables.

Validación de Cliente Móvil: Antes de procesar una venta a crédito, la app consulta al servicio BanQuito mediante el método ValidarCliente, enviando la cédula del cliente y recibiendo el monto máximo de crédito disponible junto con los datos del cliente.

Registro de Créditos: Al confirmar una venta, la aplicación invoca al servicio BanQuito para registrar el nuevo crédito con el método CrearCredito, enviando la cédula, monto, número de cuotas y tasa de interés, recibiendo como respuesta el ID del crédito y la tabla de amortización generada.

Gestión de Facturas Móvil: El servicio Comercializadora es utilizado para crear nuevas facturas mediante CrearFactura, consultar facturas por número con ObtenerFacturaPorNumero, buscar facturas por cliente usando ObtenerFacturasPorCliente, y gestionar el catálogo completo de productos con ListarProductos.

Manejo de Errores y Conectividad: La aplicación implementa try-catch para capturar excepciones de red (timeout, sin conexión a internet), errores del servicio SOAP (SoapFault), y errores de serialización XML. Los mensajes de error se muestran al usuario mediante Toast notifications y AlertDialogs nativos de Android.

Configuración Adaptable: El sistema permite configurar dinámicamente la URL base de los servicios SOAP mediante el panel de configuración, facilitando el cambio entre ambientes de desarrollo, pruebas y producción sin necesidad de recompilar la aplicación.

Esta arquitectura asegura una integración robusta y eficiente con los servicios SOAP desde dispositivos móviles Android, proporcionando una experiencia de usuario fluida incluso con condiciones variables de red.

# 12. CONCLUSIONES

Se implementó exitosamente una arquitectura de servicios web SOAP utilizando .NET Framework y WCF (Windows Communication Foundation), demostrando la capacidad de crear servicios web empresariales robustos y seguros con soporte completo para transacciones distribuidas y mensajería confiable.

Los dos sistemas (BanQuito y Comercializadora) son completamente independientes, comunicándose únicamente mediante protocolos SOAP con XML bien estructurado. Esto permite que cada sistema evolucione de forma independiente manteniendo contratos de servicio estáticos definidos mediante WSDL, garantizando la interoperabilidad entre diferentes plataformas y tecnologías.

Se desarrolló un ecosistema completo de clientes heterogéneos que demuestran la versatilidad de SOAP: Cliente Web ASP.NET que consume servicios SOAP mediante referencias web tradicionales. Cliente de Consola .NET con interfaz de línea de comandos para operaciones administrativas. Cliente de Escritorio Windows Forms con interfaz gráfica nativa y experiencia de usuario optimizada. Cliente Móvil Android desarrollado con .NET MAUI que consume servicios SOAP mediante HttpClient y serialización XML personalizada.

Se completó la lógica del negocio con características avanzadas: Cálculo automático de descuento del 33% para pagos en EFECTIVO implementado en el servicio Comercializadora. Validación estricta para pagos a CRÉDITO que requiere número de crédito válido y verificación de elegibilidad del cliente mediante invocación del servicio BanQuito. Generación automática de tabla de amortización con fórmula de cuota francesa (sistema francés) calculada en el servidor. Reducción automática de stock en tiempo real al facturar, con control de concurrencia mediante Entity Framework.

Se utilizó SQL Server LocalDB como sistema de gestión de base de datos relacional, procedimientos almacenados y vistas para optimizar consultas complejas. La arquitectura implementa el patrón Repository y Unit of Work para abstraer el acceso a datos.

Se generó documentación completa del proyecto incluyendo: Diagramas de arquitectura distribuida y modelo entidad-relación de bases de datos. Especificación detallada de operaciones SOAP con ejemplos de envelopes XML de petición y respuesta. Documentación de WSDL generados automáticamente por WCF. Guías paso a paso de configuración de IIS, SQL Server y despliegue de servicios web. Capturas de pantalla y descripciones funcionales de todos los clientes desarrollados.

# 13. REFERENCIAS

1. Microsoft .NET Framework Documentation. https://docs.microsoft.com/en-us/dotnet/framework/ Documentación oficial de .NET Framework 4.8
2. Windows Communication Foundation (WCF) Documentation. https://docs.microsoft.com/en-us/dotnet/framework/wcf/ Especificación y guías de WCF para servicios SOAP
3. ASP.NET Web Services Documentation. https://docs.microsoft.com/en-us/aspnet/web-forms/ Documentación de ASP.NET Web Forms y servicios web
4. Entity Framework Documentation. https://docs.microsoft.com/en-us/ef/ Documentación oficial de Entity Framework para acceso a datos
5. SQL Server LocalDB Documentation. https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb Documentación de SQL Server Express LocalDB
6. SOAP Protocol Specification. https://www.w3.org/TR/soap/ Especificación del protocolo SOAP 1.2 del W3C
7. WSDL (Web Services Description Language). https://www.w3.org/TR/wsdl20/ Especificación WSDL 2.0 para descripción de servicios
8. IIS (Internet Information Services) Documentation. https://docs.microsoft.com/en-us/iis/ Documentación de configuración y despliegue en IIS
9. C# Programming Guide. https://docs.microsoft.com/en-us/dotnet/csharp/ Guía oficial de programación en C#
10. Visual Studio Documentation. https://docs.microsoft.com/en-us/visualstudio/ Documentación de Visual Studio IDE
11. .NET MAUI Documentation. https://docs.microsoft.com/en-us/dotnet/maui/ Documentación de .NET Multi-platform App UI
12. SOAP Web Services Best Practices. https://www.w3.org/TR/ws-arch/ Arquitectura y mejores prácticas de servicios web
13. Design Patterns: Elements of Reusable Object-Oriented Software. Gamma, Helm, Johnson, Vlissides (Gang of Four). Addison-Wesley, 1994. Patrones de diseño aplicados en la arquitectura