UNIVERSIDAD DE COLIMA

Bitácora de desarrollo del sistema Corporativo

Luis Esteban Maciel Ceballos

Facultad: Telemática, ITI

Maestra:

JARAMILLO VELASCO RICARDO

27-11-2025

Bitácora de desarrollo del sistema Corporativo

Proyecto: Sistema Corporativo de Gestión de Clientes, Productos y Ventas - Región A y B

Herramienta de desarrollo: Microsoft Visual Studio 2022

Fecha de elaboración: noviembre de 2025

El sistema está organizado en módulos operativos que permiten administrar clientes, productos, ventas y visualizar dashboards corporativos. A continuación, se describen las vistas principales y sus funcionalidades.

Módulos y funcionalidades principales

El sistema está estructurado en módulos claros para la gestión operativa y un área de *dashboard* para el monitoreo de alto nivel.

1. Módulo de bienvenida y acceso Operativo

- **Vista:**

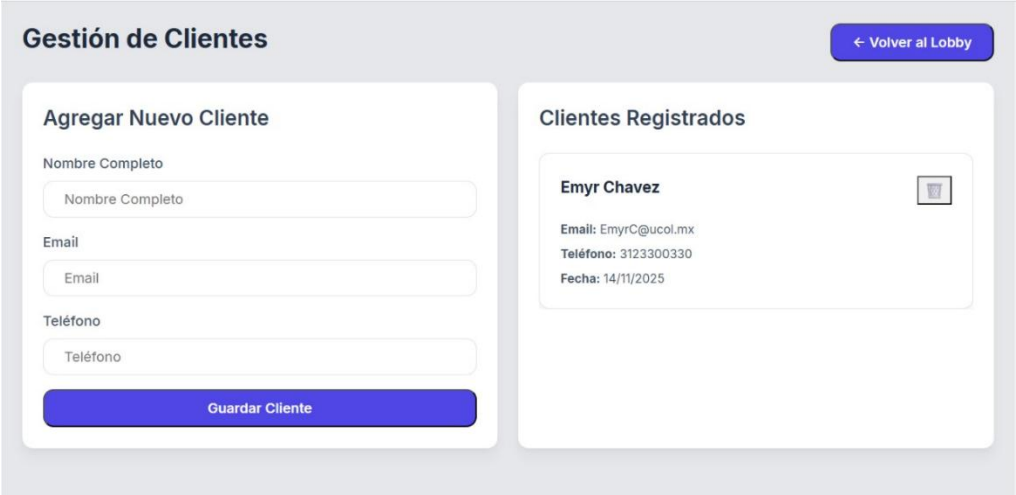
- **Propósito:** Proporcionar un punto de acceso centralizado para el operador, permitiendo la selección de las tareas diarias.

Funcionalidades:

  - **Gestión de clientes:** Acceso al módulo de administración de la base de datos de clientes (contactos e historial de compras).
  - **Gestión de productos:** Acceso al módulo de administración del inventario (productos, precios, *stock* y categorías).
  - **Registro de ventas:** Herramienta para el registro rápido y eficiente de nuevas transacciones diarias.
  - **Sucursal:** Muestra la sucursal del operador (**"PLAZA SENDERA"**), indicando el contexto operativo.

2. Gestión de clientes

- **Vista:**



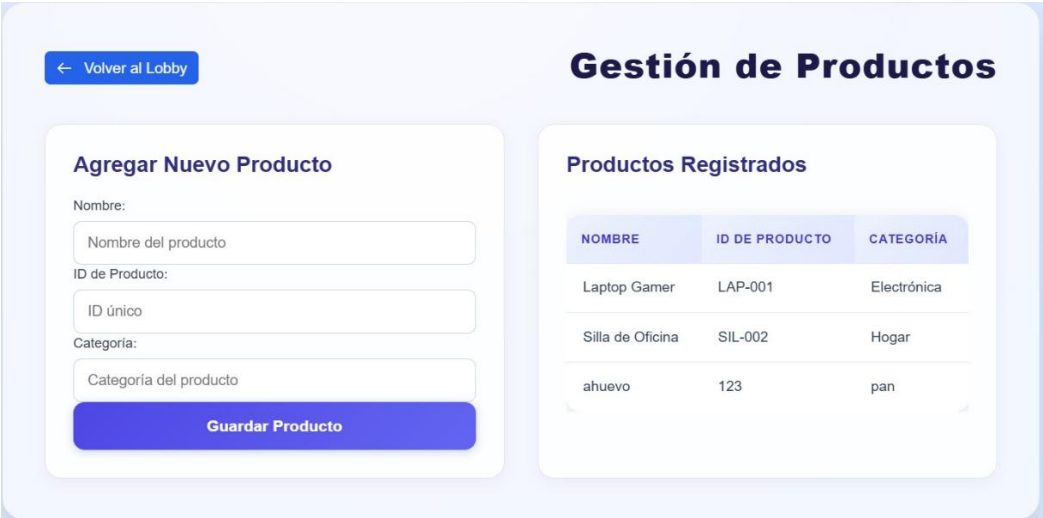
- **Propósito:** Administrar y mantener actualizada la base de datos de clientes.

Funcionalidades:

  - **Agregar nuevo cliente:** Formulario para capturar y guardar nuevos registros con campos para **Nombre completo**, **Email** y **Teléfono**.
  - **Clientes registrados:** Lista o panel que muestra la información básica de los clientes existentes (ej: Emyr Chavez) y una opción para **Eliminar** (ícono de bote de basura).

3. Gestión de Productos

- **Vista:**



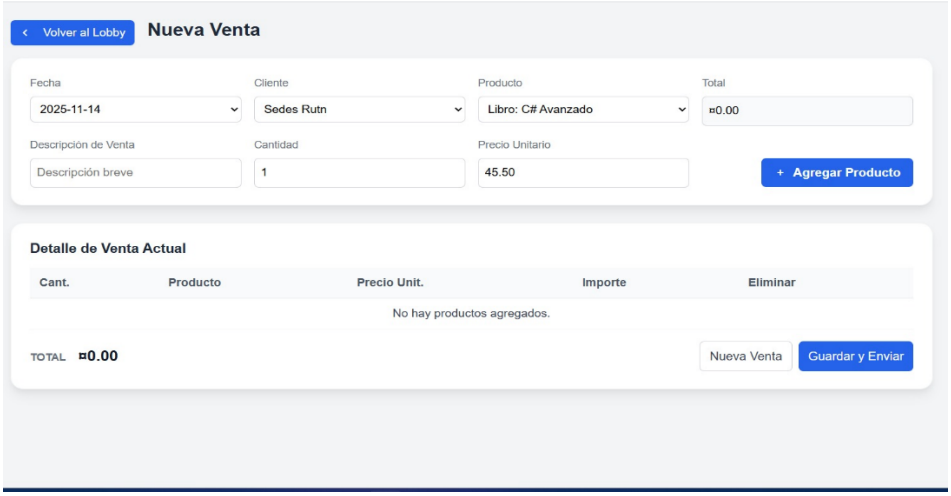
- **Propósito:** Administrar el catálogo de productos y el inventario.

Funcionalidades:

- **Agregar nuevo producto:** Formulario para registrar nuevos productos con campos para **Nombre**, **ID de producto** (único) y **categoría**.
- **Productos registrados:** Tabla o lista que muestra el **NOMBRE**, **ID DE PRODUCTO** y **CATEGORÍA** de los productos existentes (ej: Laptop Gamer, Silla de Oficina).

4. Registro de Ventas

- **Vista:**



- **Propósito:** Procesar y registrar transacciones de venta.

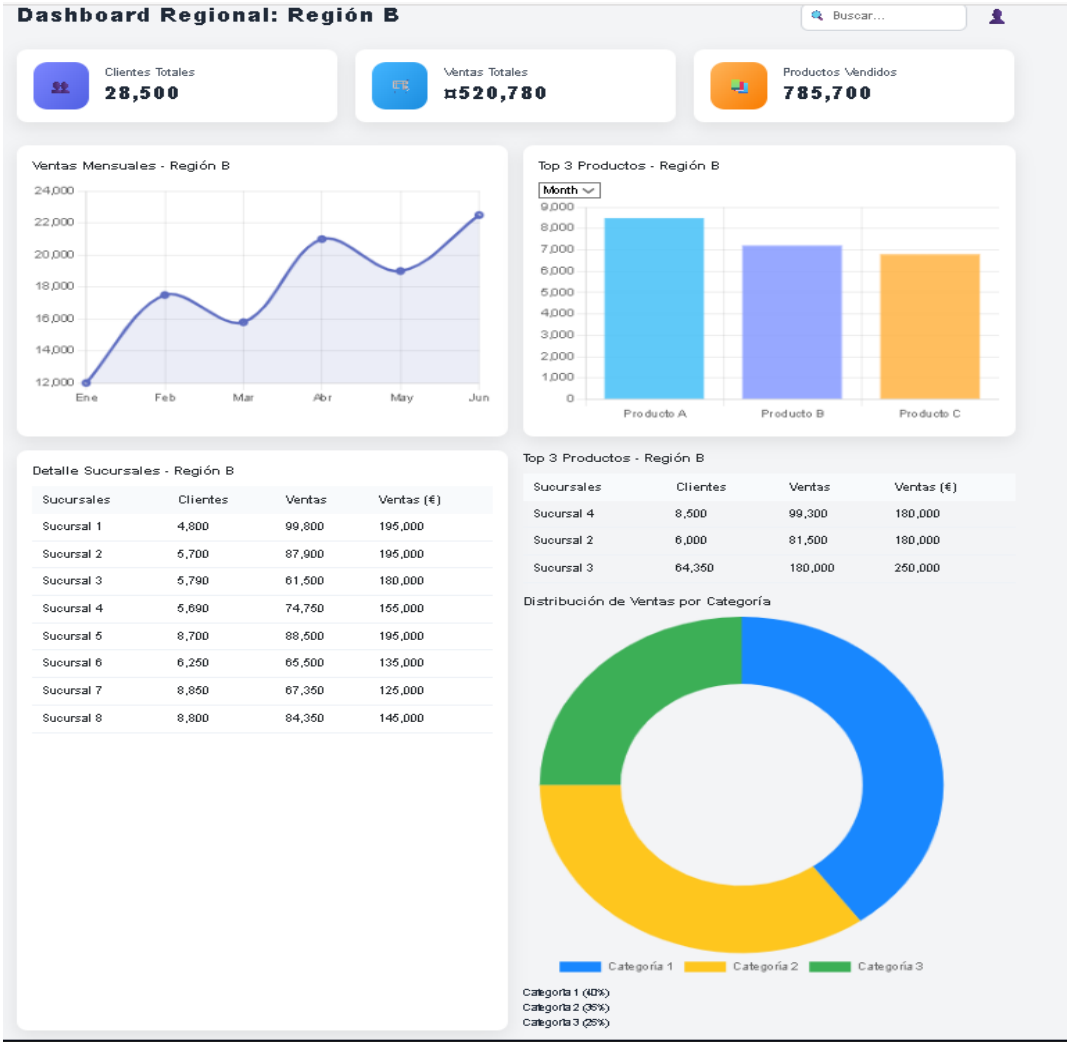
Funcionalidades:

- **Detalle de la venta:** Campos de entrada para **fecha**, selección de **cliente** y **producto**.
- **Cálculo de Ítems:** Inclusión de **cantidad**, **precio unitario** y **descripción de venta** breve, con un botón para **agregar producto** al detalle.
- **Total y envío:** Muestra el **TOTAL** de la venta actual y permite **guardar y enviar** la transacción o crear una **nueva venta**.

Dashboard y monitoreo corporativo

5. Dashboard regional: Región A y B

- **Vista:**



- **Propósito:** Visualizar los indicadores clave de rendimiento (**KPIs**) y el desempeño de las sucursales dentro de la región A y B.

Indicadores clave (KPIs):

- **Clientes totales: 28,500**
- **Ventas totales: $529,780**
- **Productos vendidos: 785,700**

Visualizaciones:

- **Ventas mensuales - región A y B:** Gráfico de línea que muestra la tendencia de ventas de Enero a Junio.
- **Top 3 productos - región A y B:** Gráfico de barras que compara el volumen de ventas de **producto A** (mayor), **producto B** y **producto C**.
- **Distribución de ventas por categoría:** Gráfico de dona que muestra la participación porcentual de las categorías (**categoría 1: 40%, categoría 2: 35%, categoría 3: 25%**).
- **Detalle sucursales - región A y B:** Tabla con métricas por sucursal (Clientes, Ventas y Ventas (€)).

6. Dashboard Corporativo

- **Vista:**

- **Propósito:** Proporcionar una visión consolidada y de alto nivel de las métricas de negocio.

Indicadores clave (KPIs):

- **Clientes: 12,548**
- **Ventas: 16,748**
- **Productos: 5,124**

Visualizaciones:

- **Días con + ventas:** Muestra el porcentaje de ventas por día de la semana (Lunes: **79.39%**, Martes: **11.40%**, Miércoles: **18.99%**). *Nota: Los porcentajes de los días suman más de 100%, lo que sugiere que el dato representa la proporción de días en los que se superó un objetivo de ventas, no el volumen total*.
- **Productos + vendidos:** Compara la venta de productos por tipo de usuario (**iPhone user: 1,586** vs. **Android user: 2,974**).
- **Ventas totales por sucursal:** Gráfico circular que compara la contribución de cada sucursal (Sucursal 1 a Sucursal 5).
- **Categorías más vendidas** y **nivel de inventario** (Gráficos pendientes de visualización detallada, pero con espacio para tendencia y barras).
-

Concluida la sección dedicada al diseño en CSS, es momento de dejar atrás los estilos, los píxeles rebeldes y las clases que parecen trabalenguas, para movernos al terreno donde realmente empieza la acción: la construcción de la API de sucursal. A partir de este punto, el enfoque cambia del aspecto visual al funcionamiento interno, donde cada endpoint y cada integración cuentan. Aquí arranca el análisis técnico del desarrollo de la API correspondiente a la operación de sucursales, detallando su estructura, lógica y decisiones implementadas.

Reporte de Desarrollo de la API "API_SUCURSAL"

Introducción

El presente reporte describe el desarrollo de la API *API_SUCURSAL*, versión 1.0, la cual fue creada con el objetivo de administrar la información relacionada con ventas, clientes y reportes dentro del sistema corporativo. La API sigue un diseño RESTful, utiliza formato JSON y codificación UTF-8. Su Base URL es: **/corporativo/api/**.

Diseño y Estructura General

La API fue construida utilizando métodos HTTP estándar: **GET, POST, PUT, DELETE y PATCH**. Está conectada a una base de datos cuyo modelo se diseñó en DrawSQL, contemplando entidades principales como *ventas*, *clientes*, *detalles de venta* y su respectiva relación.

Funcionalidades Implementadas

Módulo de Ventas

Incluye operaciones completas de CRUD, además de acciones específicas como:

- Consulta general y por ID.
- Creación de ventas.
- Marcado de ventas enviadas al sistema regional.
- Cancelación de ventas.

Se implementaron respuestas de error como *404 Not Found*, *409 Conflict* y *500 Internal Server Error* para una mejor gestión de fallos.

Módulo de Clientes

Permite administrar el catálogo de clientes mediante:

- Lista general.
- Consulta por ID.
- Creación, actualización y eliminación.

Se validan campos obligatorios y se manejan errores comunes como datos inválidos o clientes con ventas asociadas.

Detalles de Venta

Ofrece endpoints para:

- Obtener detalles de una venta.
- Agregar productos a una venta.
- Actualizar detalles existentes.

Incluye validaciones para evitar modificaciones en ventas ya enviadas o canceladas.

Módulo de Reportes

Se desarrollaron endpoints para consultas específicas:

- Ventas pendientes por enviar.
- Ventas registradas en una semana determinada del año.

Esto permite generar reportes locales de manera rápida y estructurada.

Resultados y Conclusiones

La API quedó completamente funcional y estructurada, cumpliendo con los criterios REST y con un manejo adecuado de errores y validaciones. La documentación del contrato facilita su integración futura en otros sistemas, como el sistema regional.
Actualmente permite una administración completa del flujo de ventas y clientes y deja bases sólidas para futuras ampliaciones.

Finalizado el reporte del desarrollo de la API de sucursal y habiendo dejado claras sus funciones, límites y particularidades, avanzamos hacia la siguiente capa de la arquitectura: la API regional. Este cambio no solo implica un salto en alcance, sino también en responsabilidades, ya que ahora el sistema debe coordinar información proveniente de múltiples sucursales. En esta nueva sección se documentará el diseño, operación y mecanismos que permiten que la región funcione como un cerebro organizador dentro de la plataforma.

Reporte del Desarrollo de la API "API_REGIONAL"

Introducción

El proyecto tuvo como finalidad el desarrollo de la API **API_REGIONAL**, una plataforma encargada de administrar toda la información operativa de una región, incluyendo productos, categorías, sucursales, inventarios y reportes semanales.
Esta API sirve como el sistema interno de cada región y, además, provee datos a la API Corporativa para consolidar estadísticas globales.

Objetivo del Desarrollo

El objetivo principal fue crear una API robusta y ordenada que permitiera:

- Gestionar productos, categorías y sucursales.
- Controlar existencias e inventarios por producto y por sucursal.
- Registrar reportes semanales con su respectivo detalle de ventas.
- Proporcionar información a la API Corporativa mediante endpoints de lectura.
- Contar con endpoints estadísticos tipo dashboard para consultas rápidas.

Funcionalidades Desarrolladas

Gestión de Productos

La API incluye operaciones para:

- Listar productos filtrando por ID, categoría o sucursal.
- Crear nuevos productos.
- Actualizar sus datos o existencia.
- Desactivarlos sin eliminarlos físicamente.

Los productos contienen atributos como:
id_producto, nombre, categoría, precio, existencia, activo.

Gestión de Categorías

Se implementaron endpoints para:

- Listar categorías.
- Crear nuevas categorías.
- Actualizarlas.
- Desactivarlas.

La estructura es sencilla, enfocada exclusivamente en el nombre de la categoría.

Gestión de Sucursales

La API permite:

- Listar sucursales y consultar sus detalles.
- Crear nuevas sucursales.
- Actualizar información como nombre, dirección o teléfono.
- Desactivar sucursales sin perder su historial.

Inventarios

El sistema maneja inventarios desde dos perspectivas:

- **Inventario general:** existencia total y sucursales donde hay disponibilidad.
- **Inventario por sucursal:** existencias individuales por producto.
- Actualización de existencia vía PATCH.

Esta sección permite controlar correctamente el stock regional.

Reportes Semanales

Se desarrolló un módulo completo para:

- Registrar reportes semanales con su detalle de productos vendidos.
- Consultar todos los reportes.
- Obtener un reporte específico.
- Filtrar reportes por sucursal o por mes del año.

Cada reporte incluye:
id_sucursal, año, mes, total_ventas, fecha_recepcion, id_cliente, detalles.

Dashboard y Estadísticas

Se agregaron endpoints de análisis regional que proporcionan:

- Resumen general de ventas, productos activos y rendimiento.
- Ventas agrupadas por sucursal.
- Productos más vendidos.
- Estadísticas mensuales.
- Sucursales con mayor volumen de ventas.

Estos endpoints permiten una rápida toma de decisiones dentro de la región.

API Sucursal y API Corporativo

La API expone endpoints de solo lectura para que:

- Las sucursales consulten catálogos y existencias.
- El corporativo pueda obtener información regional como:
    - sucursales activas,
    - productos disponibles,
    - reportes mensuales,
    - estadísticas generales.

Esto permite una integración completa entre sistemas.

Base de Datos

La API utiliza un modelo relacional documentado en DrawSQL, integrando tablas para:

- Productos
- Categorías
- Sucursales
- Inventarios
- Reportes semanales
- Detalles de reporte

El diseño permite escalabilidad y mantiene integridad referencial entre módulos.

Manejo de Errores

Se incluyeron códigos HTTP y mensajes estándar para:

- **400** Datos incompletos o inválidos
- **404** Elemento no encontrado
- **409** Reportes duplicados
- **500** Error interno del servidor

Esto asegura claridad y consistencia en todas las respuestas de la API.

Conclusión

El desarrollo de la API_REGIONAL cumplió con todos los requisitos funcionales. Ofrece un sistema sólido para gestionar productos, sucursales, inventarios y reportes, además de proporcionar información valiosa al corporativo. Con una arquitectura REST completa y endpoints bien estructurados, esta API cubre las necesidades operativas y administrativas de cada región de manera eficiente, escalable y organizada.

Con el análisis de la API regional debidamente completado y su comportamiento general explicado, corresponde ampliar la visión hacia el nivel superior: la API corporativa. Esta etapa marca el punto donde toda la información converge y la operación adquiere un carácter global. El siguiente apartado aborda la arquitectura, procesos y decisiones técnicas del desarrollo corporativo, mostrando cómo se integran los datos regionales para ofrecer una gestión centralizada y eficiente.

Reporte del Desarrollo de la API “API_CORPORATIVO”

Introducción

El proyecto consistió en el desarrollo de la API **API_CORPORATIVO**, diseñada para centralizar la información proveniente de distintas regiones del corporativo y gestionar reportes mensuales de ventas enviados por cada una.
La API fue implementada bajo arquitectura REST, utilizando JSON como formato principal de intercambio de datos.

Objetivo del Desarrollo

El propósito fue construir una API capaz de:

- Registrar, consultar, actualizar y desactivar regiones corporativas.
- Recibir y almacenar reportes mensuales generados por cada región.
- Proveer endpoints para análisis específicos, como reportes por región, por mes y la obtención de detalles enriquecidos con información de APIs externas regionales.

Endpoints Implementados

Gestión de Regiones

La API permite:

- Listar todas las regiones registradas.
- Consultar información de una región específica.
- Registrar nuevas regiones.
- Actualizar datos existentes.
- Desactivar regiones sin eliminarlas físicamente (borrado lógico).

Cada operación utiliza los métodos HTTP correspondientes (GET, POST, PUT, DELETE) y maneja validaciones básicas como datos repetidos, campos faltantes o regiones inexistentes.

Los campos principales manejados son:
id_region, nombre, api_url, activa.

Reportes Mensuales

Se desarrollaron módulos para:

- Registrar reportes mensuales enviados desde cada región.
- Listar todos los reportes del corporativo.
- Consultar un reporte específico.
- Consultar reportes filtrados por región.
- Obtener un reporte por mes y año.

Estos reportes incluyen datos como:
id_region, año, mes, total_ventas, fecha_recepcion.

El sistema también valida la existencia de reportes duplicados en el mismo año/mes.

Detalles Enriquecidos desde API Regional

Se implementó un endpoint especial que consulta directamente la API de cada región para obtener:

- Nombre de sucursal
- Productos vendidos
- Cantidades
- Subtotales

Este módulo combina datos locales con datos externos para generar reportes completos y enriquecidos.

Base de Datos

La API utiliza una base de datos estructurada con las tablas necesarias para:

- Regiones
- Reportes mensuales
- Relaciones entre ambas

El modelo se encuentra diagramado y documentado en DrawSQL.

Manejo de Errores

Se implementaron respuestas estándar con códigos HTTP claros:

- **200** Operación exitosa
- **201** Registro creado
- **400** Datos inválidos o incompletos
- **404** No encontrado
- **409** Conflicto (duplicados)
- **500** Error interno del servidor
- **502** Error en consulta a API externa

Reporte de la implementación de API_RegionalInterna

El proyecto API_RegionalInterna se desarrolló como una API centralizada para la gestión de información operativa dentro del entorno regional. Su estructura refleja una organización orientada a módulos funcionales, separando controladores, modelos, migraciones y configuración en distintos directorios para mantener claridad y escalabilidad. La API está diseñada para interactuar con información de sucursales, productos, inventarios, reportes y estadísticas, facilitando la comunicación entre niveles: sucursal, regional y corporativo.

Controllers

El directorio de controladores concentra los endpoints que exponen la lógica del sistema hacia el exterior. Entre ellos se incluyen controladores especializados para diferentes áreas:

- **Categoría y Productos**, con archivos separados para lógica de sucursal y lógica corporativa, asegurando que cada nivel administrativo acceda solo a su ámbito correspondiente.
- **Dashboard y Estadísticas**, orientados a la consulta de información agregada para toma de decisiones.
- **Inventario e Información de Sucursales,** encargados de manejar existencias, movimientos y datos estructurales de cada unidad operativa.
- **Reportes semanales y mensuales,** diferidos entre sucursal y regional para mantener un flujo de datos jerárquico.
- **RegionController**, que administra la información geográfica y organizacional del nivel regional.
- Se conserva también un **WeatherForecastController**, típico de plantillas iniciales, utilizado inicialmente para pruebas de estructura y funcionamiento.

En conjunto, estos controladores representan la capa donde se articula la lógica de negocio y las rutas disponibles para los distintos consumidores de la API.

Data

El archivo **RegionalDbContext.cs** define el contexto de base de datos para Entity Framework. Aquí se configuran las entidades principales del sistema, sus relaciones y el puente entre la lógica de la API y la base de datos. Este contexto es la pieza central para permitir consultas, inserciones y actualizaciones sobre los datos regionales.

Migrations

El proyecto incluye varias migraciones que documentan la evolución del esquema de base de datos:

- **20251106141501_Inicial**
- **20251111213503_InicialAzure**

Estas migraciones muestran variaciones y ajustes al prepararse el entorno de despliegue en Azure. El archivo adicional **RegionalDbContextcs** sugiere una configuración o corrección paralela al contexto principal, posiblemente generada para resolver diferencias entre entornos locales y remotos.

Models

Los modelos del proyecto representan las entidades fundamentales para la operación regional:

- **Categoría, Producto e Inventario**, esenciales para el flujo de ventas y abastecimiento.
- **Cliente**, en caso de requerir identificación de usuarios o compras personalizadas.
- **Región y Sucursal**, que definen la estructura administrativa y geográfica.
- **Reportes y Detalles de reporte**, que almacenan la información consolidada para análisis semanal o mensual.

Estos modelos establecen la base del esquema de datos sobre el cual trabajan los controladores y el contexto de base de datos.

Configuración y Archivos Base

El proyecto contiene, además:

- **Program.cs**, donde se inicializa el servidor web, se configuran los servicios, la inyección de dependencias y las rutas principales.
- **API_RegionalInterna.csproj**, que define las dependencias, SDK y configuraciones del proyecto.
- **appsettings.json y su variante de desarrollo**, donde se configuran cadenas de conexión, parámetros del entorno, opciones de logging y cualquier variable necesaria para la operación de la API.
- **launchSettings.json**, que establece cómo se ejecuta la API en entornos locales de desarrollo.
- **WeatherForecast.cs**, común en plantillas iniciales para pruebas de respuesta básica.

Conclusión

La estructura del proyecto está organizada para soportar múltiples niveles administrativos—sucursal, regional y corporativo—mediante el uso de controladores separados, modelos bien definidos y una capa de datos consistente con migraciones controladas. Esto permite mantener una API limpia, escalable y alineada con las necesidades operativas del sistema, facilitando tanto la integración como el mantenimiento a largo plazo.

Bitácora / Reporte de la Estructura del Proyecto API_RegionalInterna

El proyecto API_RegionalInterna está organizado bajo una arquitectura típica de una API en .NET, estructurada para mantener orden, escalabilidad y facilidad de mantenimiento. Cada carpeta cumple un rol definido y permite ubicar rápidamente controladores, modelos, configuración y migraciones de la base de datos.

1. Controllers

Ubicados dentro de RegionalDataBases/Controllers/, estos archivos concentran la lógica de entrada de la API. Cada controlador expone endpoints específicos según el módulo:

- **CategoriasController.cs:** Manejo de categorías.
- **ClientesController.cs:** Administración de clientes.
- **InventarioController.cs:** Gestión de existencias y movimientos.
- **ProductosController.cs:** Operaciones CRUD de productos.
- **SucursalesController.cs:** Control de sucursales.
- **VentasController.cs:** Procesamiento de ventas.
- **WeatherForecastController.cs:** Controlador por defecto para pruebas.

Estos controladores siguen el patrón MVC tradicional, permitiendo separar la lógica de negocio de la lógica de comunicación con el cliente.

2. Data

En RegionalDataBases/Data/ se encuentra el núcleo de la comunicación con la base de datos:

- **Context.cs:** Clase DbContext principal. Define las tablas (DbSets), reglas de configuración y el enlace con Entity Framework Core.

3. Migrations

Ubicado en RegionalDataBases/Migrations/, contiene las migraciones generadas por Entity Framework:

- Archivos **InitialCreate.cs** y **.Designer.cs:** Creación inicial del esquema de la base de datos.
- **ApplicationDbContextModelSnapshot.cs:** Fotografía del estado actual del modelo. Es lo que EF usa para detectar cambios.

Este módulo permite controlar la evolución del esquema sin perder trazabilidad.

4. Models

En RegionalDataBases/Models/ se encuentran las clases que definen la estructura de los datos del sistema:

- **Models.cs:** Conjunto de entidades utilizadas por el sistema.
- **WeatherForecast.cs:** Modelo de ejemplo para pruebas.
- **Properties/:** Configuración para publicación y perfiles de despliegue.

Cada modelo representa una tabla o entidad del sistema, y se usa tanto en el contexto como en los controladores.

5. Archivos de Configuración

Aquí se incluyen parámetros clave del proyecto:

- **appsettings.json** y **appsettings.Development.json:** Configuración de cadenas de conexión, logs y variables de entorno.
- **launchSettings.json:** Definición de perfiles para ejecutar la API en distintos entornos.
- **RegionalDataBases.http:** Archivo de pruebas rápidas de endpoints.

Estos archivos permiten adaptar el proyecto sin tocar el código central.

6. Archivos del Proyecto

- **Program.cs:** Punto de entrada de la aplicación y configuración de servicios.
- **RegionalDataBases.csproj / .user:** Configuración del proyecto .NET.
- **RegionalDataBases.slnx:** Solución global que agrupa todos los componentes.

Conclusión

La estructura del proyecto está ordenada conforme a las buenas prácticas de .NET. Cada componente está dividido por responsabilidad, lo que facilita el mantenimiento, la escalabilidad y el trabajo en equipo. Los controladores administran la lógica de entrada, los modelos definen el esquema de datos, el contexto enlaza con la base de datos y las migraciones documentan la evolución del sistema. La organización general permite comprender rápidamente cómo fluye la información dentro de la API y cómo está diseñado el backend para soportar procesos de ventas, clientes, productos y sucursales.

GUÍA DE ESTRUCTURA

La estructura completa abarca **tres grandes módulos**:

1. **RegionalDataBases (API de Regiones)**
2. **API_RegionalInterna (API Corporativa y Sucursales)**
3. **Corporativo (Proyecto Web Corporativo / Dashboard / Reportes)**

Cada uno contiene controladores, modelos, contexto de base de datos y configuraciones. Todo organizado como una solución .NET.

1. REGIONALDATABASES – API DE REGIONES

Carpeta raíz: API_RegionalInterna A / RegionalDataBases

Proyecto API encargado del manejo de datos regionales: clientes, inventarios, productos, sucursales y ventas.

Controllers

Controladores que exponen endpoints REST:

- **CategoriasController.cs** – CRUD de categorías.
- **ClientesController.cs** – Manejo de clientes.
- **InventarioController.cs** – Altas/bajas/cambios de inventario.
- **ProductosController.cs** – Gestión de productos.
- **SucursalesController.cs** – Información de sucursales.
- **VentasController.cs** – Registro y consulta de ventas.
- **WeatherForecastController.cs** – Controlador de prueba autogenerado.

Data

- **Context.cs** – DbContext principal que conecta el API con la base SQL.
- **Migrations/** – Historial de migraciones de Entity Framework.
    - Archivos InitialCreate – creación inicial de tablas.
    - ApplicationDbContextModelSnapshot.cs – estado actual del modelo.

Models

- **Models.cs** – Define las entidades del API: Categoría, Producto, Cliente, etc.

Properties

- **PublishProfiles/** – Configuración de despliegue Web Deploy (Azure/IIS).
- **launchSettings.json** – Configuración de ejecución local (Swagger, puerto, etc.)

Archivos raíz

- **Program.cs** – Configuración de servicios, middlewares, CORS, EF Core.
- **RegionalDataBases.csproj** – Configuración del proyecto.
- **RegionalDataBases.http** – Pruebas rápidas de endpoints desde VSCode.
- **appsettings.json / Development** – Cadenas de conexión y configuración.

2. API_RegionalInterna – API CORPORATIVA, SUCURSALES Y REPORTES

Es la API "grande" que integra **Corporativo, Sucursales y Regiones**. Contiene todos los módulos de negocio.

Controllers

Controladores segmentados por área:

Regiones y Sucursales

- **RegionController.cs**
- **SucursalesController.cs**
- **SucursalesControllerCorporativo.cs**
- **CategoriaControllerSucursal.cs**
- **ProductosControllerSucursal.cs**

Corporativo

- **CategoriasControllers.cs**
- **ProductosControllerCorporativocs.cs**
- **InventarioController.cs**

Reportes

- **DashboardController.cs** – Datos globales para dashboards.
- **EstadisticasController.cs** – Métricas generales.
- **ReportesMensualesController.cs**
- **ReportesSemanalesController.cs**
- **ReportesSemanalesControllerSucursal.cs**

Pruebas

- **WeatherForecastController.cs**

Data

- **RegionalDbContext.cs** – DbContext que administra todas las tablas corporativas/regionales.

Migrations/

Historial de migraciones:

- Inicial – creación de base local.
- InicialAzure – migración adaptada para Azure.

Models

Entidades principales:

- **Categoria.cs**
- **Cliente.cs**
- **Inventario.cs**
- **Producto.cs**
- **Region.cs**
- **ReporteSemanalDetalle.cs**
- **ReportesVentas.cs**
- **Sucursal.cs**

Archivos raíz

- **API_RegionalInterna.csproj** – Configuración del proyecto.
- **Program.cs** – configuración de servicios.
- **appsettings.json / Development** – conexiones y parámetros.
- **Contrato API ... .pdf** – Documentación contractual entre APIs y BD.
- **API_RegionalInterna.http** – pruebas de endpoints.

3. CORPORATIVO – APLICACIÓN WEB / DASHBOARD CENTRAL

Este es un **proyecto web** que consume las APIs anteriores y genera reportes.

.vs/

Archivos internos del entorno de Visual Studio:

- Configuración del IDE.
- Índices de proyectos.
- Historial de sesiones.

(No afecta el código del proyecto, pero VS lo usa).

Controllers/

- **RegionesController.cs** – Control para consumir datos de regiones.
- **ReporteController.cs** – Generación de reportes mensuales/semanales.

Data/

- **CorporativoContext.cs** – Contexto EF Core para la base corporativa.

Migrations/

- InitialCreate – migración base del proyecto Corporativo.

Models/

- **DetalleReporte.cs**
- **Regiones.cs**
- **ReporteMensual.cs**

Entidades usadas por el dashboard corporativo.

Utils/

DTOs usados para la creación de reportes:

- **CrearDetalleDTO.cs**
- **CrearReporteDTO.cs**

bin/ y obj/

Carpetas generadas automáticamente:

- **bin/** – Compilaciones (Debug/Release).
- **obj/** – Datos temporales de compilación.

Incluyen DLLs, dependencias .NET, endpoints de Razor, assets, etc.

Archivos raíz

- **Corporativo.csproj** – Archivo principal del proyecto.
- **Program.cs** – Configuración de middlewares y servicios web.
- **appsettings.json / Development** – cadenas de conexión, variables del sistema.
- **web.config** – configuración para despliegues en IIS.

Otros

- **adjuntos.zip** – materiales varios.
- **TUTORIAL DE INSTALACIÓN DE CSS...** – Guía de estilos.
- **CSS A IMPLEMENTAR** – Estilos definidos para la interfaz.

Conclusiones generales

El desarrollo completo del sistema corporativo —desde el diseño visual hasta las tres capas de API— logró consolidar un ecosistema sólido, escalable y coherente. Las vistas proporcionan una operación clara y funcional; la API_SUCURSAL garantiza la captura y administración de ventas locales; la API_REGIONAL articula múltiples sucursales y unifica la operación; y finalmente, la API_CORPORATIVO integra la información global para análisis ejecutivo.

El proyecto destaca por su estructura organizada, su arquitectura REST bien definida, su documentación consistente y su enfoque gradual que permite escalar el sistema sin comprometer su estabilidad. Con esto, el corporativo cuenta con una solución integral para la gestión operativa, regional y centralizada de toda su información clave.