Factuapp

Universidad Modelo

Proyectos III

Alejandro de Jesús Ayala Díaz [10], Miguel Ángel Martín Ortiz [10], Daniela Mora Ayuso [10], Ángel Mauricio Moreira Palma [10], Julio Santamaría Sánchez [10]

Kenia Nayrhovy Osorio López

30 de septiembre de 2025

ÍNDICE

RESUMEN

El proyecto Factuapp se centra en crear una aplicación para escritorio que facilite la descarga masiva y automatizada de las facturas electrónicas (CFDI) del portal del SAT. El objetivo es solucionar el inconveniente del proceso manual, que es lento y propenso a errores, el cual actualmente padecen contadores y empresas, mejorando los tiempos de trabajo y minimizando los riesgos de equivocaciones.

La solución que se propone incluye una arquitectura segura que resguarda las credenciales cifradas en el dispositivo del usuario, implementa. NET 8 en C# para la lógica de negocio, utiliza Electron para la parte visual, y se apoya en SQLite como sistema de almacenamiento local, todo ello con herramientas de libre uso.

Se ofrecerá un MVP operativo que podrá autenticarse, crear lotes para descarga, manejar colas de trabajo y exportar resultados en los formatos XML, CSV y JSON. El proyecto es viable desde el punto de vista técnico, se anticipa que será beneficioso para las empresas al automatizar tareas repetitivas, incrementar la seguridad de la información y reducir el esfuerzo administrativo.

INTRODUCCIÓN

Hoy en día, la gestión de comprobantes fiscales digitales representa un proceso de gran importancia para las empresas. Sin embargo, la descarga manual desde el portal del Servicio de Administración Tributaria (SAT) suele ser un proceso tardado, repetitivo e ineficiente, especialmente al manejar grandes cantidades de documentos.

A raíz de esta situación, surge la necesidad de desarrollar una aplicación que permita descargar de forma masiva, automatizada y simple, facturas electrónicas.

Con la realización de este proyecto se busca optimizar el proceso de descarga de facturas, a través de una interfaz amigable para el usuario y un sistema que ofrezca seguridad.

ANTECEDENTES

La facturación electrónica en México constituye el eje de la formalización tributaria y de la trazabilidad de las transacciones. El Comprobante Fiscal Digital por Internet (CFDI) es un archivo XML con validez legal que incorpora identificadores únicos (UUID), sellos digitales, datos del emisor y del receptor, importes e impuestos, y cuya estructura y sintaxis están normalizadas por el Anexo 20 del Servicio de Administración Tributaria (SAT). La versión 4.0 del CFDI entró en vigor el 1 de enero de 2022 y, tras un período de convivencia con la 3.3, es la única versión válida desde el 1 de abril de 2023; el cambio introdujo campos obligatorios del receptor (nombre o razón social, régimen fiscal y código postal) y ajustes que incrementan la calidad de los datos (SAT, 2022; SAT, 2023). Estas reglas técnicas y de negocio enmarcan cualquier iniciativa de software que procese facturas electrónicas y condicionan los requisitos de calidad, seguridad e interoperabilidad del sistema propuesto.

Desde el punto de vista legal, la emisión y recuperación del CFDI se sustenta en los artículos 29 y 29-A del Código Fiscal de la Federación, que obligan a expedir comprobantes mediante documentos digitales y determinan los requisitos mínimos de validez (Cámara de Diputados, 2025). En paralelo, la Ley Federal de Protección de Datos Personales en Posesión de los Particulares exige medidas de seguridad administrativas, técnicas y físicas proporcionales al riesgo, aspecto crucial cuando se manejan llaves y certificados de firma electrónica avanzada (Cámara de Diputados, 2025). Por ello, cualquier arquitectura que procese CFDI debe justificar explícitamente su modelo de protección de credenciales y su esquema de resguardo local.

En la práctica, los contribuyentes recuperan sus comprobantes por dos vías oficiales: el portal de “Consulta y recuperación de comprobantes” y el Servicio Web de descarga masiva. El portal permite autenticarse con e.firma o contraseña, buscar por filtros y descargar XML y metadatos, pero establece límites operativos y ventanas de atención, lo que restringe su uso en escenarios de gran volumen. El Servicio Web, por su parte, habilita la recuperación masiva autenticada con e.firma, mediante un ciclo de cuatro pasos: autenticar, solicitar paquetes, verificar estado y descargar. Aun con topes por petición y tiempos máximos de preparación, esta vía formaliza la automatización y desplaza la intervención manual hacia la configuración de parámetros y la supervisión de errores (SAT, s. f.). Esta asimetría entre portales interactivos y servicios automatizables justifica una solución local con gestión de estados y reintentos para organizaciones con alto volumen.

Desde la perspectiva económica y productiva, la digitalización tributaria y la factura electrónica son palancas de cumplimiento y eficiencia. La evidencia comparada de la OCDE muestra que los sistemas de e-invoicing, al estandarizar facturas estructuradas y su intercambio con la autoridad, facilitan prellenados, reducen cargas administrativas y mejoran la calidad de la base tributaria (OECD, 2022; OECD, 2025). En América Latina, el Banco Interamericano de Desarrollo ha documentado impactos sobre formalización, trazabilidad y control del fraude, con México como caso de referencia regional por su adopción temprana y alcance (Barreix y Zambrano, 2018). A nivel operativo, automatizar la descarga y organización de CFDI reduce horas-hombre dedicadas a búsquedas manuales, minimiza omisiones y duplica la consistencia de la información para cierres contables y auditorías; así, desplaza trabajo de bajo valor hacia análisis financiero, conciliación y planeación, con efectos positivos en costos y tiempos de ciclo (OECD, 2022; Barreix y Zambrano, 2018). En contextos de alta presión regulatoria y fiscalización digital, contar con repositorios completos y actualizados de CFDI mediante procesos automáticos disminuye riesgos de sanciones y mejora la capacidad de respuesta ante revisiones electrónicas (OECD, 2025).

La propuesta tecnológica se estructura como una aplicación de escritorio con interfaz en Electron, núcleo de negocio en .NET 8, almacenamiento local en SQLite y acceso a datos con Dapper, orquestada por un módulo de trabajo en segundo plano (worker) que encapsula el ciclo de autenticación, solicitud, sondeo y descarga del SAT. La elección de Electron responde a la necesidad de una interfaz multiplataforma y a su ecosistema web; sin embargo, su adopción exige seguir recomendaciones de endurecimiento: deshabilitar la integración de Node en renderer, habilitar aislamiento de contexto, activar el sandbox de procesos y evitar cargar contenido remoto con privilegios. Estas medidas, recomendadas en la guía oficial de seguridad, reducen la superficie de ataque y separan la UI del código privilegiado (Electron, 2025a; Electron, 2025b). En este esquema, la interfaz interactúa con servicios locales bien delimitados, mientras las operaciones sensibles (cifrado, IO y cliente del SAT) residen en procesos controlados.

El uso de .NET 8 para el core técnico se justifica por su rendimiento, la disponibilidad de soporte de larga duración y su biblioteca madura para XML. Con LINQ to XML, el sistema puede cargar, validar y transformar CFDI 3.3/4.0 con código conciso y consultas expresivas, lo que agiliza la extracción de metadatos (UUID, RFC, importes e impuestos) y reduce errores típicos del procesamiento basado en cadenas (Microsoft, 2023; Microsoft, 2025a; Microsoft, 2025b). Esta base tecnológica se complementa con SQLite como base de datos embebida, una opción transaccional ACID de cero configuración y buen desempeño en escenarios de cliente local, con persistencia en un único archivo

y comportamiento estable ante fallos de energía (SQLite Consortium, 2025a; SQLite Consortium, 2025b). El acceso a datos con Dapper, micro-ORM ligero y de alto rendimiento, permite consultas y escrituras eficientes sin imponer la complejidad de un ORM completo, manteniendo control directo sobre SQL cuando el rendimiento lo exige (Dapper Team, 2025; Microsoft, 2024).

Dado que el cuello de botella principal proviene de la disponibilidad del SAT y de sus límites de atención, la solución debe contemplar un worker robusto capaz de gestionar reintentos con backoff exponencial, colas de solicitudes, límites de concurrencia y registros de auditoría. Estos mecanismos, unidos a un manejo explícito de estados (pendiente, en proceso, listo, fallido), amortiguan la latencia del servicio y permiten recuperación ordenada tras fallos intermitentes. El diseño se ancla en las reglas de operación publicadas por el SAT y en el ciclo formal autenticar–solicitar–sondear–descargar; al vincular la arquitectura a dichas reglas se reduce el riesgo funcional y se evitan supuestos ad hoc frágiles ante cambios menores en la plataforma (SAT, s. f.). En paralelo, la seguridad de credenciales es un eje transversal: la e.firma —compuesta por certificado (.cer), llave privada (.key) y contraseña— habilita actos jurídicos electrónicos y, por su carácter de identificación y no repudio, requiere controles estrictos. En términos de mejores prácticas, se recomienda cifrar localmente las credenciales con algoritmos simétricos robustos, derivar y custodiar llaves de manera segura, y eliminar trazas sensibles en registros; estas decisiones están alineadas con la regulación mexicana de datos personales y con la orientación general sobre resguardo de identidades digitales (Cámara de Diputados, 2025; SAT, s. f.).

Finalmente, la calidad y la mantenibilidad se sostienen con pruebas automatizadas. xUnit, marco abierto y ampliamente adoptado en .NET, permite estructurar suites de pruebas unitarias y de integración para casos de parsing, exportación, control de errores transitorios y rutas críticas del worker. Las guías oficiales de Microsoft detallan las prácticas para organizar proyectos de pruebas y ejecutar escenarios que sirvan como red de seguridad ante refactorizaciones y actualizaciones del framework (xUnit.net, 2025; Microsoft, 2025c). Incorporar pruebas desde el inicio reduce el riesgo de defectos funcionales que afecten la descarga o integridad de los CFDI y favorece el versionado controlado frente a futuros cambios normativos o técnicos.

DEFINICIÓN DEL PROBLEMA

Hoy en día, las empresas se ven obligadas a invertir tiempo y esfuerzo en la descarga manual de comprobantes fiscales digitales desde el portal del SAT. Este procedimiento, aparte de ser lento y tedioso, se vuelve impráctico cuando el volumen de facturas es elevado.

Al carecer de una herramienta en que apoyarse, las empresas sufren de retrasos de gestión administrativa, la probabilidad de errores aumenta, y complica el control de la información fiscal. Es por esto que surge la necesidad de implementar un programa de software que facilite la descarga automatizada de múltiples archivos de CFDI, ofreciendo una interfaz sencilla de entender y medidas de seguridad necesarias.

JUSTIFICACIÓN

Para toda empresa, sin importar su tamaño, descargar manualmente las facturas electrónicas a través del portal del SAT implica una inversión considerable de tiempo y esfuerzo. Además, siempre existe el riesgo de cometer errores al trabajar con grandes cantidades de estos documentos.

Pese a que ya existen algunos servicios orientados a atender esta problemática, muchos de ellos pecan de tener un costo de uso elevado, ser demasiado complejos o no ofrecer todas las funciones necesarias. Por lo anterior, desarrollar una aplicación de descarga automatizada de facturas que sea, a la vez, eficiente y accesible resulta de gran importancia.

Es así como ofrecer una solución práctica y confiable que cubra las necesidades de los clientes no solo permitirá optimizar procesos administrativos al reducir tiempos, sino que también disminuirá la incidencia de errores humanos.

OBJETIVOS

General

Crear una aplicación de escritorio que facilite el proceso de obtención de facturas electrónicas a contadores y empresas, a través de una interfaz intuitiva para el usuario y un sistema que resguarde la seguridad y confidencialidad de la información fiscal.

Específicos

1. Implementar un sistema que se conecte a la API del SAT y permita el procesamiento y exportación de facturas electrónicas.
2. Desarrollar una función de descarga de facturas, que ofrezca opciones de descarga en diferentes formatos (XML, JSON y PDF).
3. Integrar a la aplicación una interfaz gráfica de usuario (GUI) intuitiva que facilite el uso de la aplicación a usuarios sin conocimientos técnicos.

METODOLOGÍA

La entrega del proyecto está prevista para finales de noviembre. El entregable será un MVP capaz de realizar descargas masivas de facturas electrónicas. A partir de la investigación inicial entendimos que el reto principal no es sólo técnico, sino de lógica de producto: la funcionalidad crítica depende de una integración directa con el SAT, un proveedor con restricciones duras, ventanas de mantenimiento y respuestas intermitentes. Por esa razón la arquitectura debe ser simple, segura y muy clara en el manejo de estados y errores.

Durante el prototipado comparamos alternativas web y de escritorio. Optamos por una aplicación de escritorio para priorizar la seguridad: la e.firma y el certificado digital son datos extremadamente sensibles y, al mantenerlos cifrados en el equipo del usuario, reducimos superficie de riesgo y dependencia de terceros. La aplicación validará credenciales solo en el momento de comunicarse con el SAT y no expondrá secretos en tránsito ni en registros.

Para la interfaz elegimos Electron (HTML, CSS y JavaScript). Consideramos MAUI en un inicio, pero implicaba aprender un framework nuevo de UI con la curva de adopción que eso conlleva dentro del tiempo disponible. En el núcleo de negocio usaremos .NET 8 en C#, porque el proyecto gira en

torno a XML y .NET ofrece librerías maduras para parseo, validación, transformación y acceso nativo al sistema de archivos. El sistema se organizará en cuatro partes: el core, donde vivirá la lógica de negocio, servicios, reglas y manejo de RFC; la infraestructura, que encapsulará la comunicación con servicios externos como los Web Services del SAT y el almacenamiento local; un worker responsable de ejecutar tareas pesadas en segundo plano con control de concurrencia y reintentos; y la interfaz de usuario en Electron que expone un flujo simple de uso.

Los componentes tecnológicos serán locales y sin costo de licenciamiento: SQLite para la base de datos embebida y Dapper como micro-ORM, ambos de libre acceso. El único servicio externo será el propio Web Service del SAT, también de libre acceso, con el que nos integraremos mediante cliente HTTP/SOAP. Las credenciales (archivos .cer y .key y su contraseña) se almacenarán cifradas en disco usando AES-256 con una llave derivada de forma segura, y la rotación de llaves quedará prevista para endurecer la seguridad. No registraremos secretos ni valores sensibles en los logs.

Sabemos por la investigación previa que el proceso de descarga no depende exclusivamente de nuestra velocidad, sino de la propia infraestructura del SAT. Por ello incluiremos un módulo worker que procese lotes en segundo plano, gestione la cola de tareas, aplique reintentos con backoff exponencial cuando el SAT limite la tasa o esté intermitente y priorice la finalización confiable por encima de la agresividad en concurrencia. La calidad se asegurará con un proyecto de pruebas automatizadas en C# con xUnit: verificaremos el cifrado y descifrado, el parseo de CFDI 3.3 y 4.0, la exportación a CSV y JSON y, mediante dobles de prueba del cliente del SAT, simularemos casos raros como XML corruptos, caídas del servicio y RFC inválidos.

La metodología de trabajo será Scrumban, una mezcla ligera de planificación por hitos con flujo continuo en un tablero Kanban. Esta elección nos permite mantener foco en entregables semanales sin sobrecargar al equipo con ceremonias. La Fase 1, de definición y diseño, ya dejó como resultado la decisión de arquitectura, el modelo de datos y las políticas de seguridad. La Fase 2, de construcción, se desarrollará a lo largo de octubre y noviembre con la siguiente cadencia orientativa considerando la capacidad real del equipo: una persona en backend con cuatro horas diarias y dos desarrolladores junior en frontend con una hora diaria cada uno. En la primera quincena de octubre levantaremos la columna vertebral del proyecto: creación de la solución en .NET, estructura del monorepo, almacenamiento cifrado de credenciales y el “caso feliz” del SAT (autenticar, solicitar, sondear y descargar un lote pequeño). En la segunda quincena activaremos el worker con control

de estados y reintentos, integraremos la interfaz de Electron para crear lotes y visualizar el progreso y consolidaremos el historial de lotes con búsqueda básica.

Durante la primera semana de noviembre incorporaremos el parser de metadatos esenciales del CFDI hacia SQLite y habilitaremos las exportaciones a CSV y JSON. En los días siguientes afinaremos la resiliencia: reintentos por elemento fallido, manejo explícito de errores recurrentes del SAT, sanitización de logs, purga de archivos temporales y trazabilidad mediante auditoría mínima. A mitad de noviembre realizaremos pruebas con volúmenes más altos dentro de lo razonable para un MVP, mediremos tiempos, tasas de error y estabilidad y corregiremos bordes. Si el calendario lo permite, añadiremos la representación en PDF como "impresión" del XML; si no, quedará programada para la siguiente iteración sin afectar el objetivo del MVP.

Con esta planificación y la capacidad indicada, el alcance del MVP es viable para finales de noviembre. Aun así, corresponde ser transparentes: el comportamiento del SAT puede introducir demoras ajenas a nuestro control. Si se presentaran bloqueos por cambios del Web Service o por restricciones de acceso, proponemos dos caminos claros. El primero es congelar el alcance en lo esencial y priorizar la estabilidad: descarga masiva con ZIP y exportación a CSV y JSON, historial con reintentos y seguridad completa, dejando la impresión en PDF y mejoras cosméticas para una iteración posterior. El segundo es extender una semana adicional para endurecimiento y pruebas si la institución lo permite, manteniendo el mismo alcance, pero asegurando un acabado más robusto en rendimiento y experiencia de usuario.

La última fase será la entrega. Cerraremos con un instalador firmado, el manual breve de instalación y uso, un video corto que demuestre el flujo completo desde la conexión del RFC hasta la descarga del lote, y un reporte de pruebas que documente los casos automatizados, las simulaciones de error y las métricas de rendimiento. De este modo, el MVP queda listo para su evaluación académica y para evolucionar, después de la entrega, hacia funcionalidades adicionales como la impresión en PDF, reportes analíticos y validaciones ampliadas.

DISEÑO CONCEPTUAL

Descripción detallada del proyecto

- **Cliente:** El usuario podrá crearse una cuenta e iniciar sesión utilizando su correo electrónico y una contraseña. Ingresando sus datos: certificado de sello digital, e-firma y contraseña del

portal (si aplica), el usuario podrá ver sus facturas. En la pantalla de visualización de las facturas, el usuario podrá ver cuáles son deducibles, así como descargar las que le interesen.

Requerimientos del cliente

Los requerimientos establecidos por el cliente se muestran en la Tabla 1, 2 y 3.

Tabla 1. Requerimientos del cliente.

ELEMENTO	PROCESO	REQUERIMIENTO (qué hace el sistema)	USUARIO	CRITERIOS / NOTAS
Autenticación	Iniciar sesión	Permitir login con email+password.	Cliente	3 intentos fallidos → bloqueo 15 min.
	Conectar RFC	Registrar/editar **CIEC** o **e.firma** (FIEL/CSD) para un RFC.	Cliente	Encriptar en reposo; validación con prueba de conexión.
	Nueva descarga	Crear **lote** indicando: RFC, periodo (fecha inicio/fin), tipo (CFDI 3.3/4.0, ingresos/egresos/pagos, retenciones), estatus (vigentes/cancelados), y "búsqueda por RFC emisor/receptor" opcional.	Cliente	Validar rangos de fechas; máximo **N** días por lote (p. ej., 90).
Descargas	**Enviar**	Colocar el lote en **cola** y mostrar estado: Pendiente → En progreso → Completado/Fallido/Parcial.	Cliente	Barra de progreso con conteo estimado y últimos mensajes.
	Descargar ZIP	Al terminar, descargar ZIP con XML; opcional CSV con metadatos (UUID, fecha, RFCs, total, efecto).	Cliente	ZIP firmado con hash y nombre normalizado RFC_YYYYMMDD_hhmm.zip.

Tabla 2. Requerimientos del operador.

ELEMENTO	PROCESO	REQUERIMIENTO	USUARIO	CRITERIOS / NOTAS
Panel	Tablero de cola	Ver **toda la cola** de procesamiento (multicliente): estados, prioridad, tiempo estimado.	Operador	Ordenar por estado/fecha/prioridad.
	Forzar reintento	Reintentar un lote o sólo los fallidos.	Operador	Registrar quién operó y cuándo.
	Pausar/Cancelar	Pausar o cancelar procesamiento de un lote.	Operador	Estados: Pausado/Cancelado con motivo.

Soporte	Ver logs	Consultar logs de errores del SAT (throttle, credenciales inválidas, timeouts).	Operador	Ocultar secretos; filtros por RFC/fecha.
	Ver credenciales	Ver estado (válidas/por expirar) y fecha de expiración de certificados.	Operador	**Nunca** mostrar secretos en claro; sólo estatus.

Tabla 3. Requerimientos del cliente.

ELEMENTO	PROCESO	REQUERIMIENTO	USUARIO	CRITERIOS / NOTAS
Usuarios	Alta/Baja/Edición	CRUD de usuarios y asignación de rol (Cliente/Operador/Admin).	Admin	Doble confirmación en bajas.
Tenants	Planes/Límites	Definir límites por cliente: **concurrencia**, **tamaño de lote** (p. ej. hasta 50,000 XML), **descargas/día**.	Admin	Validación en UI y en backend.
Seguridad	Rotación de llaves	Rotar claves de cifrado de credenciales.	Admin	Registrar auditoría.
Auditoría	Bitácora	Reporte de eventos (login, creación de lote, descargas, reintentos).	Admin	Exportable a CSV.
Sistema	Salud del servicio	Ver métricas básicas: jobs activos, tasa de éxito, latencias.	Admin	Endpoint /health y panel simple.

Requerimientos por botones / enlaces (UI MVP):

Conectar RFC: abre modal para capturar CIEC o archivos .cer/.key + contraseña, probar conexión.

Nueva descarga: formulario de lote (RFC, fechas, tipo CFDI/retenciones, estatus, filtro opcional por RFC tercero).

Enviar: crea el lote y redirige al Detalle del lote.

Descargar ZIP: descarga el paquete generado; Exportar CSV metadatos.

Reintentar fallidos: relanza sólo los UUID con error del lote.

Historial: lista de lotes con filtros y paginado.

Detalle del lote: vista con progreso, contadores y tabla de resultados/errores.

Cerrar sesión: invalida token.

Diseño de pantallas

Para factuapp, se ha optado por que el color de la marca sea el azul. A continuación, se muestran los diseños preliminares de cómo quedarían las pantallas principales de la aplicación, en las cuales también se muestra el logotipo preliminar en la esquina superior izquierda.

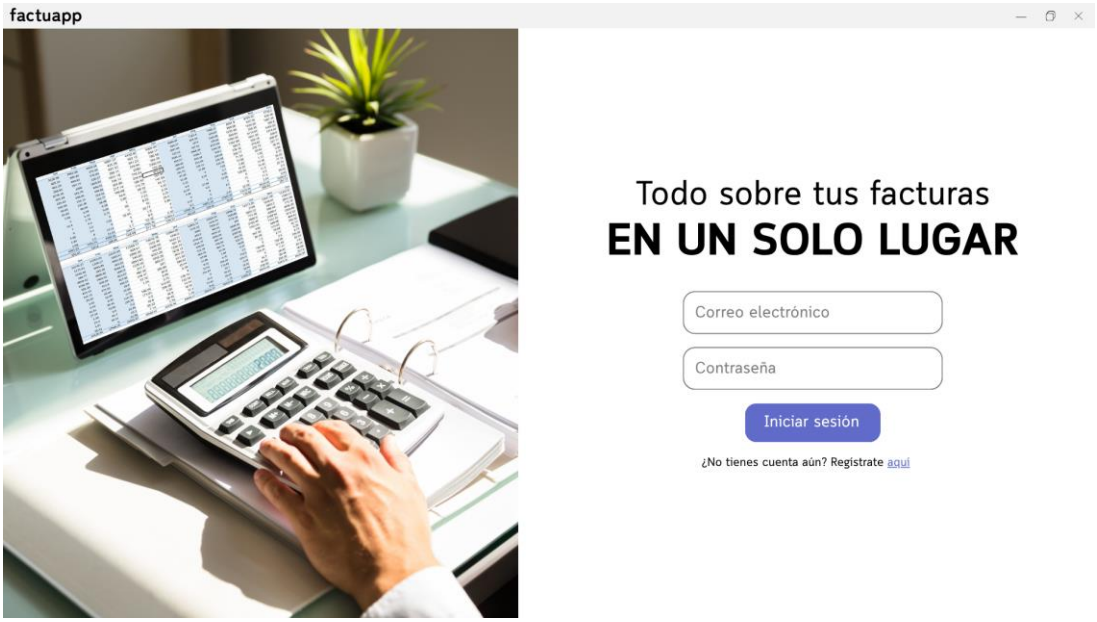

Figura 1. Vista de la pantalla inicial, en la que se encuentran los campos para iniciar sesión a través de un correo electrónico y contraseña. De igual manera, en la parte inferior hay un enlace que redirige a la página de registro.

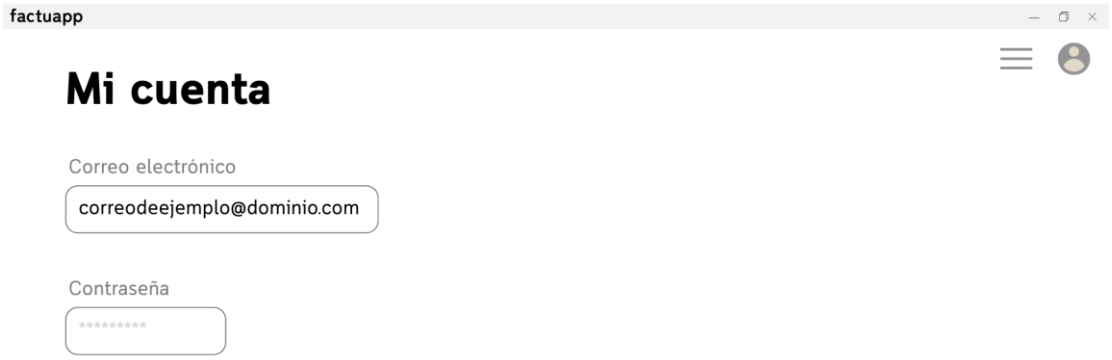

Figura 2. Pantalla de cuenta. En esta se mostrarán los datos de la cuenta con la que se ha iniciado sesión.

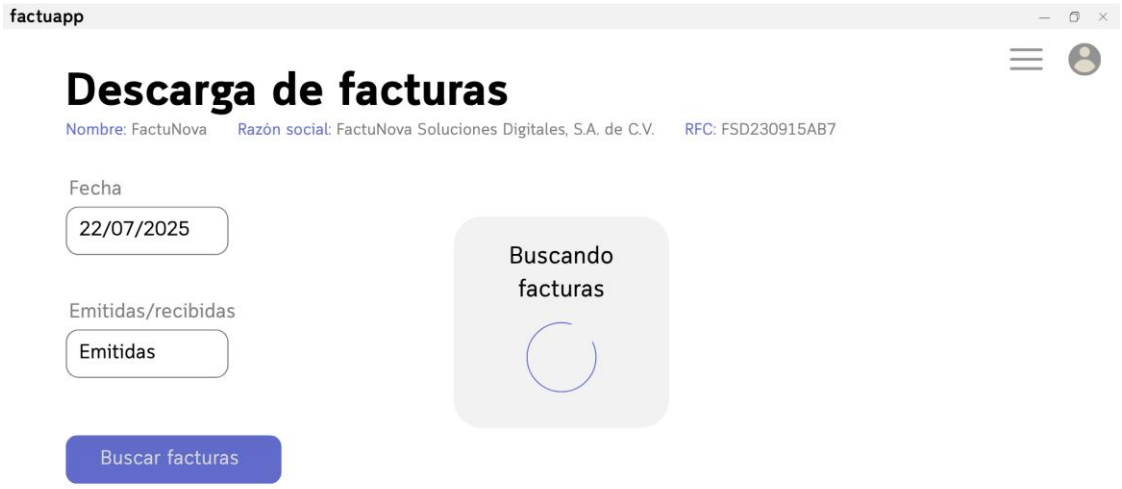

Figura 3. Vista de la pantalla para descargar facturas. En ella hay dos campos: fecha de la factura y si esta es emitida o recibida. Debajo se encuentra el botón buscar facturas, que buscará facturas asociadas al cliente que cumplan las condiciones establecidas.

Figura 4. Tras presionar el botón de buscar facturas, aquellas que coincidan con lo ingresado serán mostradas en un panel en la sección derecha de la pantalla. En este panel se mostrarán las facturas halladas, las cuales se podrán seleccionar para su descarga.

Figura 5. Pantalla de configuración de descarga. Esta pantalla contiene algunas opciones de personalización para la descarga, de modo que esta se adecúe a la necesidad del usuario.

Figura 6. Pantalla complementaria a las anteriores. Consiste únicamente en una ventana pop up que muestra el avance del proceso de descarga.

Herramientas a utilizar

Para el desarrollo, pruebas y operación del prototipo se utilizarán las siguientes herramientas:

Frontend: Para el desarrollo del frontend se utilizará el framework para escritorio Electron, el cual permite la creación de aplicaciones de escritorio completas utilizando código HTML, CSS y JavaScript.

Backend: El backend estará construido sobre el framework ASP.NET, el cual contiene herramientas y librerías diseñadas específicamente para el desarrollo de software como servicio. El lenguaje de programación a utilizar en el framework es C#.

Base de datos: Para la base de datos se utilizará SQLite, un sistema abierto de gestión de bases de datos relacionales.

Repositorios para el desarrollo del proyecto

Los archivos de la aplicación serán alojados en un repositorio de GitHub. De este modo, cada miembro del equipo podrá realizar aportaciones al proyecto sin interferir con el flujo de trabajo de los demás, optimizando la producción.

Organización del equipo de trabajo

Documentación: Miguel Martín.

Frontend: Daniela Mora y Julio Santamaría.

Backend: Alejandro Ayala y Mauricio Moreira.

Pruebas: Miguel Martín.

PLAN DE TRABAJO

Tabla 2. Diagrama de Gantt con las actividades a realizar para el desarrollo de la aplicación...

No.	Actividades	Responsable	Entrega		Agosto				Septiembre				Octubre				Noviembre				Diciembre			
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20
	Proyecto - FactuApp																							
1	Creación del anteproyecto.	Miguel	Sept. 2																					
2	Definir roles.	Miguel	Sept. 9																					
3	Entrevista y requisitos.	Alejandro A.	Sept. 16																					
4	Análisis de requerimiento.	Alejandro A.	Sept. 23																					
5	Diseño de base de datos	Alejandro A.	Oct. 14																					
6	Diseño de la arquitectura del sistema.	Mauricio M.	Oct. 21																					
7	Diseño de la interfaz del usuario.	Mauricio M.	Oct 28.																					
8	Planificación del desarrollo.	Alejandro y Mauricio	Nov. 4																					
9	Codificación del backend.	Alejandro y Mauricio	Nov. 18																					
10	Codificación del frontend.	Julio y Daniela	Nov. 25																					
11	integración fronend-backend.	Daniela y Alejandro	Dic. 2																					
12	Pruebas unitarias y del sistema.	Miguel	Dic. 9																					
13	Documentación del sistema.	Miguel	Dic. 16																					
14	Presentación de entrega y presentación final	Julio	Dic. 23																					

COSTOS

El costeo integral para el proyecto Factuapp, requiere una inversión total de 1,125,690 pesos para cubrir el desarrollo, lanzamiento y las fases iniciales del producto. Este presupuesto está diseñado para asegurar la creación de una aplicación robusta y su exitosa introducción en el mercado.

La mayor parte de la inversión se destina a los recursos humanos, con un total de 612,000 pesos. Este monto cubre un equipo multidisciplinario durante aproximadamente tres meses, compuesto por un Líder Técnico, dos Desarrolladores Full Stack, un Desarrollador Frontend, un Ingeniero de Seguridad, un Analista de Calidad (QA), un Diseñador UX/UI y un Product Owner para gestionar el proyecto. A esto se suman los costos de infraestructura tecnológica y herramientas, que ascienden a 31,400 pesos, contemplando servidores de desarrollo y pruebas, licencias de software especializado, repositorios de código y un certificado digital de firma de código para garantizar la seguridad de la aplicación.

Además de los costos de desarrollo, se han considerado los gastos operativos y administrativos necesarios durante esta fase, que totalizan 79,500 pesos. Esta cifra incluye el espacio de oficina o coworking, servicios básicos como internet y electricidad, y la amortización del equipo de cómputo. También se ha asignado una partida crucial de 60,000 pesos para cubrir los aspectos legales y fiscales, como el registro de la marca, la asesoría legal para la redacción de contratos y términos de uso, y consultoría fiscal específica para la interacción con el SAT.

Para asegurar una entrada exitosa al mercado, se ha presupuestado un fondo de 122,000 pesos para marketing y comercialización. Esto abarca el desarrollo de un sitio web corporativo, material de branding, una campaña de lanzamiento digital y la creación de contenido como videos tutoriales. Adicionalmente, se invertirán 73,000 pesos en la capacitación y el soporte inicial, lo que incluye la creación de manuales, la ejecución de un programa piloto con empresas beta y la configuración del equipo de soporte técnico para los primeros meses.

Finalmente, para mitigar riesgos y manejar imprevistos, el presupuesto incluye un fondo de contingencia y reservas por un total de 147,790 pesos. Esta reserva estratégica asegura la estabilidad financiera del proyecto ante cualquier desviación del plan, completando así el presupuesto total necesario para llevar Factuapp de una idea a una realidad operativa en el mercado.

Plan de negocios

El modelo de negocio de Factuapp es de tipo Freemium con suscripciones (SaaS), diseñado para atraer una amplia base de usuarios con una versión gratuita y funcional, para luego convertirlos en clientes de pago. A través de planes escalonados (Profesional, Empresarial y Enterprise), se genera un flujo de ingresos mensuales recurrentes. Para financiar el desarrollo y lanzamiento, se buscará una inversión inicial de $1,125,690 MXN a través de capital semilla, presentando este plan de negocio a inversionistas ángeles o fondos de capital de riesgo.

La recuperación de dicha inversión se logrará a través de las ganancias acumuladas generadas por las suscripciones de pago. Aunque el primer año proyecta una pérdida neta mientras se consolida la base de clientes, el crecimiento constante de los ingresos permitirá alcanzar el punto de equilibrio operativo rápidamente. Según las proyecciones financieras, se estima que la inversión inicial se recuperará en su totalidad en un plazo de 18 a 20 meses, momento a partir del cual el proyecto comenzará a generar un retorno de inversión significativo y a ser plenamente rentable.

Delimitaciones

El proyecto Factuapp se centrará exclusivamente en el diseño y desarrollo de un Producto Mínimo Viable (MVP) funcional. La responsabilidad del equipo culmina con la entrega final de este software probado y su aceptación por parte del cliente, asegurando que cumple con las funcionalidades acordadas.

Para mantener las expectativas alineadas, el alcance del proyecto tiene delimitaciones claras. El desarrollo no incluirá el mantenimiento continuo ni el soporte técnico a usuarios finales una vez entregado el producto. Tampoco cubrirá futuras actualizaciones que sean necesarias por cambios en la plataforma del SAT, el desarrollo de funcionalidades personalizadas que no estén en el acuerdo inicial, o cualquier estrategia de marketing y comercialización. Finalmente, la empresa desarrolladora no asume ninguna responsabilidad fiscal o legal derivada del uso que los clientes finales le den a la aplicación, ya que esta recae enteramente en el usuario.

CONCLUSIONES

En conclusión, el proyecto se presenta como una propuesta sólida y prometedora, gracias a su objetivo de optimizar la descarga masiva de comprobantes fiscales digitales y por las características que lo destacan de otros softwares del mercado.

Entre sus principales ventajas frente a otros competidores destacan un costo menor, una interfaz de usuario intuitiva y un alto nivel de seguridad en el resguardo y manejo seguro de los datos de los usuarios.

Finalmente, el diagrama de Gantt realizado nos muestra que con el seguimiento correcto de los pasos, el proyecto estaría listo aproximadamente en noviembre de 2025.

REFERENCIAS

Barreix, A., & Zambrano, R. (2018). Electronic invoicing in Latin America. Inter-American Development Bank. https://publications.iadb.org/publications/english/document/Electronic-Invoicing-in-Latin-America.pdf Publicaciones

Cámara de Diputados. (2025). Código Fiscal de la Federación. https://www.diputados.gob.mx/LeyesBiblio/pdf/CFF.pdf Cámara de Diputados

Cámara de Diputados. (2025). Ley Federal de Protección de Datos Personales en Posesión de los Particulares. https://www.diputados.gob.mx/LeyesBiblio/pdf/LFPDPPP.pdf Cámara de Diputados

Dapper Team. (2025). Dapper: a simple object mapper for .NET. https://github.com/DapperLib/Dapper GitHub

Electron. (2025a). Security recommendations. https://electronjs.org/docs/latest/tutorial/security Electron

Electron. (2025b). Process sandboxing. https://electronjs.org/docs/latest/tutorial/sandbox Electron

Microsoft. (2023). XDocument class overview (LINQ to XML). https://learn.microsoft.com/en-us/dotnet/standard/linq/xdocument-class-overview Microsoft Learn

Microsoft. (2024). Microsoft.Data.Sqlite overview. https://learn.microsoft.com/en-us/dotnet/standard/data/sqlite/ Microsoft Learn

Microsoft. (2025a). System.Xml.Linq – XDocument. https://learn.microsoft.com/en-us/dotnet/api/system.xml.linq.xdocument Microsoft Learn

Microsoft. (2025b). The official .NET support policy. https://dotnet.microsoft.com/en-us/platform/support/policy Microsoft

Microsoft. (2025c). Unit testing C# in .NET using xUnit. https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-csharp-with-xunit Microsoft Learn

Organisation for Economic Co-operation and Development. (2022). Tax Administration 3.0 and electronic invoicing. https://www.oecd.org/en/publications/tax-administration-3-0-and-electronic-invoicing_2ffc88ed-en.html OECD

Organisation for Economic Co-operation and Development. (2025). Welcome to the tech-enabled tax administration of the future! https://www.oecd.org/en/blogs/2025/06/welcome-to-the-tech-enabled-tax-administration-of-the-future.html OECD

Servicio de Administración Tributaria. (2022). Anexo 20: Guía de llenado de los CFDI. https://omawww.sat.gob.mx/tramitesyservicios/Paginas/documentos/Anexo_20_Guia_de_llenado_CFDI.pdf SAT México

Servicio de Administración Tributaria. (2023). Servicio de facturación CFDI versión 4.0 (vigente a partir del 1 de enero de 2022). https://wwwmat.sat.gob.mx/aplicacion/75169/servicio-de-facturacion-cfdi-version-4.0-(vigente-a-partir-del-1-de-enero-de-2022) SAT Matricula

Servicio de Administración Tributaria. (s. f.). Consulta y recuperación de comprobantes. https://wwwmat.sat.gob.mx/consultas/42968/consulta-y-recuperacion-de-comprobantes-(nuevo) SAT Matricula

Servicio de Administración Tributaria. (s. f.). Descarga masiva de CFDI y CFDI de retenciones e información de pagos: Manual de usuario. https://www.sat.gob.mx/cs/Satellite?blobkey=id&blobwhere=1461174995051&ssbinary=true SAT

Servicio de Administración Tributaria. (s. f.). e.firma: preguntas frecuentes. https://omawww.sat.gob.mx/informacion_fiscal/preguntas_frecuentes/Paginas/firma_electronica_preguntas.aspx SAT México

SQLite. (s. f.). SQLite is transactional (ACID). https://www.sqlite.org/transactional.html SQLite

SQLite. (2025a). Features of SQLite. https://www.sqlite.org/features.html SQLite

xUnit.net. (2025). About xUnit.net. https://xunit.net/ xUnit.net

ANEXOS

1. ¿Cuál es exactamente el problema que están viviendo hoy con la facturación electrónica?

El volumen. Cada mes gestionamos miles de CFDI de distintos clientes y el portal del SAT no está pensado para flujos masivos: hay límites por búsqueda, tiempos de espera y descargas manuales que consumen horas-hombre. Eso retrasa cierres contables, conciliaciones y auditorías internas.

2. ¿Por qué no seguir usando el portal con más personal o turnos?

Porque escalar personas a un proceso manual no escala la calidad ni reduce errores. Aumenta el costo y mantiene el riesgo: omisiones de XML, duplicados, clasificaciones inconsistentes. Necesitamos automatización confiable y auditable.

3. ¿Cuáles son las opciones actuales que vislumbran para mejorar este proceso?

Hemos estado pensando en varias soluciones pero siempre se complicaría por la conexión que hay con el SAT, es esencial tener de alguna manera la información a la mano directa del proveedor que en este caso es el SAT. Hemos pensado en IA, hasta analizado sí generar una unidad especializada para este tema en particular pero creemos que una aplicación dedicada a este tema sería lo ideal.

4. ¿Qué requisitos consideran no negociables para cualquier tipo de solución?

Pues lo principal la seguridad de nuestros clientes, hemos tenido varias situaciones desagradables que no nos gustaría repetir. Lo segundo, que verdaderamente sea una mejora ya que hemos invertido en “soluciones” que al final no se entregan, no esperaríamos mucho solo que cumpla con lo que promete.

5. ¿Qué integración necesitan con sus procesos contables?

Exportaciones consistentes. Con el CSV/JSON por lote ya podemos alimentar nuestro sistema contable y hacer conciliaciones. Más adelante quizá integremos directo con el ERP, pero el MVP debe asegurar metadatos limpios.

6. ¿Cómo medirán el éxito del MVP una vez entregado?

Tres indicadores: tiempo de ciclo para 1,000 XML, tasa de errores/reintentos por lote y cobertura (porcentaje de CFDI esperados vs. descargados). Si logramos cerrar el mes con 100% de comprobantes, menos retrabajo y trazabilidad completa, el MVP cumplió.

7. ¿Qué necesitan en términos de uso y adopción interna?

Una interfaz simple: conectar RFC/e.firma, crear lote y descargar. Onboarding corto con un manual de 6–8 páginas y un video. Roles básicos para evitar que cualquiera toque credenciales y, de ser posible, un modo “solo lectura” para quien solo consulta reportes.

8. ¿Qué expectativas tienen sobre soporte y mantenimiento tras la entrega?

Una ventana de soporte para la puesta en marcha, correcciones de estabilidad si el SAT cambia detalles operativos, y una guía para renovar certificados sin romper la configuración. No pedimos características nuevas inmediatas; primero estabilidad y seguridad.

9. ¿Cómo visualizan el retorno de inversión?

Ahorro de horas en descarga y clasificación, cierres más rápidos y menos incidencias en auditorías. Si el equipo deja de “pelearse” con el portal y dedica esas horas a análisis y planeación, ya ganamos. El ROI se medirá en tiempo liberado y en reducción de errores.

10. ¿Qué riesgos ven y cómo los quieren mitigar?

La seguridad es un riesgo, pero si logran hacerlo para que corra de manera local sin conectarse a internet, creo que con eso se mitigaría mucho riesgo. Otro tema además de la seguridad es el mismo SAT, no muchas personas que han trabajado con nosotros en algún tipo de innovación o diseño de procesos ha sabido adaptarse a la parte contable, como los términos contables específicos.