Módulo 1. Introducción a las automatizaciones en hojas de cálculo

CÓRDOBA CREA FUTURO

Introducción

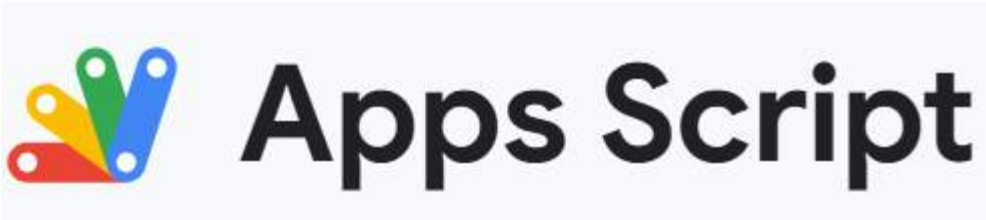
En términos simples, automatizar significa lograr que tareas que normalmente haríamos manualmente (como copiar datos, hacer cálculos o enviar reportes) se ejecuten de forma automática y repetible mediante herramientas o scripts. Estas automatizaciones son altamente relevantes en entornos laborales modernos y, como venimos viendo, son cada vez más sencillas de implementar.

Por el lado de las hojas de cálculo, estas siguen siendo una herramienta dominante para gestionar procesos empresariales en múltiples áreas (desde finanzas hasta recursos humanos), incluso en la era de la inteligencia artificial y la nube (Automation Anywhere, s. f.). Su automatización permite ahorrar cantidades significativas de tiempo en tareas rutinarias y minimizar errores humanos: muchas horas de trabajo manual (susceptibles a descuidos) pueden transformarse en segundos de ejecución precisa gracias a macros o scripts (Automation Anywhere, s. f.). Esto no solo mejora la eficiencia, sino que aumenta la calidad y

consistencia de los resultados. Un proceso automatizado realiza siempre los mismos pasos bajo las mismas reglas, elimina la variabilidad y reduce el riesgo de errores que, acumulados, podrían impactar negativamente en el negocio (Automation Anywhere, s. f.).

Es por esto que, si en una empresa se trabaja con múltiples hojas de cálculo, con procesos repetitivos y manuales, es de gran valor conocer las opciones de automatización de estas tareas.

Figura 1: Apps Script



Fuente: [imagen sin título del logo de Apps Script] (s. f.)
https://revolucionatupyme.com/google-apps-script/

1. Introducción a Apps Script

Google Apps Script (a veces llamado simplemente Apps Script) es la plataforma de scripting de Google diseñada para automatizar y extender las aplicaciones de la suite Google Workspace (Sheets, Docs, Gmail, Drive, Forms, etc.). En esencia, Apps Script proporciona un entorno de desarrollo basado en JavaScript que permite añadir funcionalidad personalizada e integrar servicios en Google Workspace de forma relativamente sencilla.

¿Qué es Apps Script y para qué sirve? Es un servicio en la nube de Google que nos permite programar acciones automáticas dentro de las herramientas de Google.

Por ejemplo, podríamos crear un pequeño programa que tome datos de una hoja de cálculo de Google y, con un solo

clic, genere un documento de Google Docs con un informe formateado, o envíe correos electrónicos automáticos a una lista de contactos usando Gmail. La gran ventaja es que Apps Script está totalmente integrado en el ecosistema Google: se puede acceder a él directamente desde Google Sheets, Docs, Forms, etc., y conectarlo con prácticamente cualquier producto de Google o de terceros. De hecho, la plataforma ofrece API y servicios listos para usar con dichas aplicaciones, lo que facilita la creación de automatizaciones que involucren múltiples aplicaciones. En síntesis, Apps Script permite "programar aplicaciones para crear automatizaciones de acciones en las que intervienen las herramientas del ecosistema de Google" (Maestro, s. f.) de forma muy fluida.

Posibilidades dentro de Google Workspace: Apps Script abre un amplio abanico de casos de uso en entornos laborales. A continuación, mencionamos algunos ejemplos típicos aplicables en el mundo profesional:

- **Envío de correos automáticos y personalizados:** es posible utilizar Apps Script para leer datos desde una hoja de cálculo (por ejemplo, nombre, dirección de correo electrónico y mensaje personalizado para cada cliente) y enviar de manera automática un correo personalizado a cada destinatario a través de Gmail. Esto permite realizar en segundos el envío masivo de comunicaciones que manualmente tomarían horas (Maestro, s. f.). Un caso de uso real es el de generar y mandar notificaciones

de pago, newsletters internas o saludos personalizados en fechas especiales, tomando la información de la hoja y combinándola con plantillas de Gmail.

- **Generación de informes y actualización de datos:** Apps Script puede automatizar la creación de reportes periódicos. Por ejemplo, una empresa puede tener un script que tome cifras de distintas hojas de cálculo, las procese (calculando totales, promedios, gráficos, etc.) y genere automáticamente un informe de Google Docs o un PDF listo para distribuir cada fin de mes. Asimismo, se pueden programar actualizaciones automáticas de datos: si varias hojas están vinculadas (p. ej., respuestas de Google Forms acumuladas en Sheets), un script puede consolidar esa información, limpiar datos duplicados o erróneos, y preparar un resumen ejecutivo cada día sin intervención manual.

- **Workflows integrados entre aplicaciones:** gracias a Apps Script, es posible orquestar flujos de trabajo complejos que involucren varias aplicaciones de Google. Por ejemplo, pensemos en la gestión de un evento corporativo: se puede crear un formulario de registro para participantes y, al enviarlo, un script almacena las respuestas en Sheets, envía automáticamente un correo de confirmación con detalles personalizados a cada participante, genera un Docs con su agenda personalizada y añade los eventos correspondientes a su calendario (Maestro, s. f.). Todo este proceso ocurriría en segundos, sin necesidad de que un usuario copiara y pegara información entre distintas aplicaciones. Este tipo de automatización integral ahorra muchísimo esfuerzo en coordinaciones y garantiza que no haya inconsistencias (porque todos los pasos están vinculados por el código).

Por supuesto, las capacidades de Google Apps Script van mucho más allá de estos ejemplos. Se pueden conectar con API externas (por ejemplo, consumir un servicio externo y volcar datos en la hoja), crear add-ons personalizados para Google Sheets, interactuar con bases de datos e incluso construir pequeñas aplicaciones web internas aprovechando la infraestructura de Google. En resumen, Apps Script convierte a Google Sheets (y el resto de Workspace) en una plataforma programable, donde un profesional con algo de creatividad puede resolver multitud de necesidades específicas y así automatizar flujos de trabajo propios de su organización.

Limitaciones

A pesar de sus ventajas, es importante destacar algunas limitaciones clave de esta herramienta:

- **Lenguaje y curva de aprendizaje:** Apps Script utiliza JavaScript como lenguaje de programación. Si bien JavaScript es muy popular y ampliamente documentado, esto implica que para explotar Apps Script más allá de ejemplos básicos, el usuario necesitará familiarizarse con conceptos de programación en este lenguaje. No es una herramienta no-code pura; es decir, los usuarios sí deben escribir código. Sin embargo, como veremos más adelante, apoyándose en la IA para desarrollar el código,

cualquier persona sin conocimientos puede implementar estas automatizaciones.

- **Entorno de ejecución y cuotas:** los scripts de Apps Script se ejecutan en la nube de Google, no en el equipo local del usuario. Esto conlleva restricciones impuestas por Google en cuanto a tiempo de ejecución y consumo de recursos. Por ejemplo, actualmente un script de Apps Script tiene un tiempo máximo de ejecución continuo de 6 minutos por invocación (Google for Developers, s. f.). Asimismo, existen cuotas diarias y límites para el uso de servicios: número de correos que se pueden enviar por día mediante un script, llamadas API de Google, tamaño de datos procesados, etc. (si se exceden, el script lanzará una excepción y se detendrá) (Google for Developers, s. f.). Estos límites se amplían con planes avanzados de Google Workspace.

- **Escalabilidad y complejidad:** si bien Apps Script permite crear soluciones muy útiles de forma rápida, cuando los proyectos crecen en complejidad, pueden volverse difíciles de mantener. Al no ser un entorno de desarrollo tradicional pleno, carece de herramientas avanzadas para proyectos grandes (control de versiones robusto, depuración avanzada, pruebas unitarias integradas, etc.). Además, no está pensado para construir aplicaciones empresariales a gran escala; para ello, sería más conveniente una plataforma de desarrollo más robusta (por ejemplo, un software a medida en un servidor propio). A pesar de esto, es más que suficiente para desarrollar automatizaciones de gran valor en el entorno laboral.

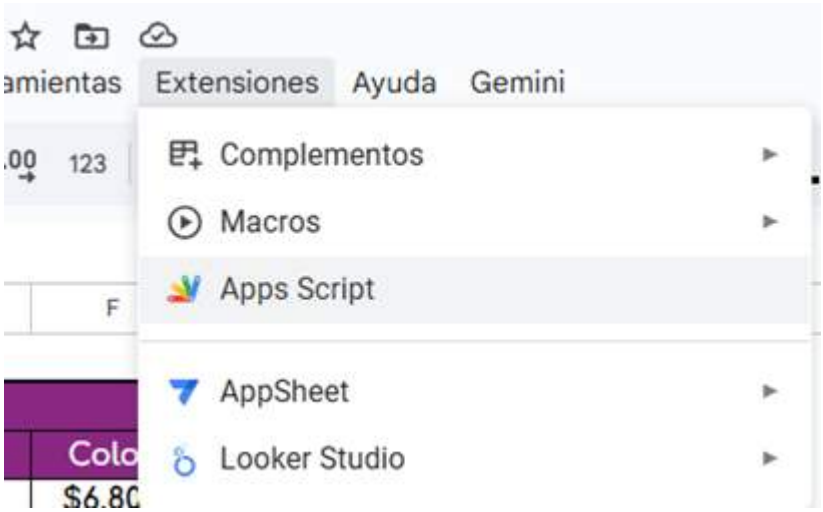
Cómo acceder a Google Apps Script desde Google Sheets

Aunque Apps Script se puede abrir desde distintas rutas, en la práctica, el camino más sencillo (y el que más se usa para

automatizar hojas de cálculo) es arrancar desde una hoja de cálculo de Google Sheets:

1. Abrir la hoja de cálculo con la que queremos trabajar (por ejemplo, un reporte de ventas en Google Sheets).
2. Ir al menú superior y hacer clic en “Extensiones”.
3. Dentro de ese menú, seleccionar “Apps Script”.

Figura 2: Google Sheets



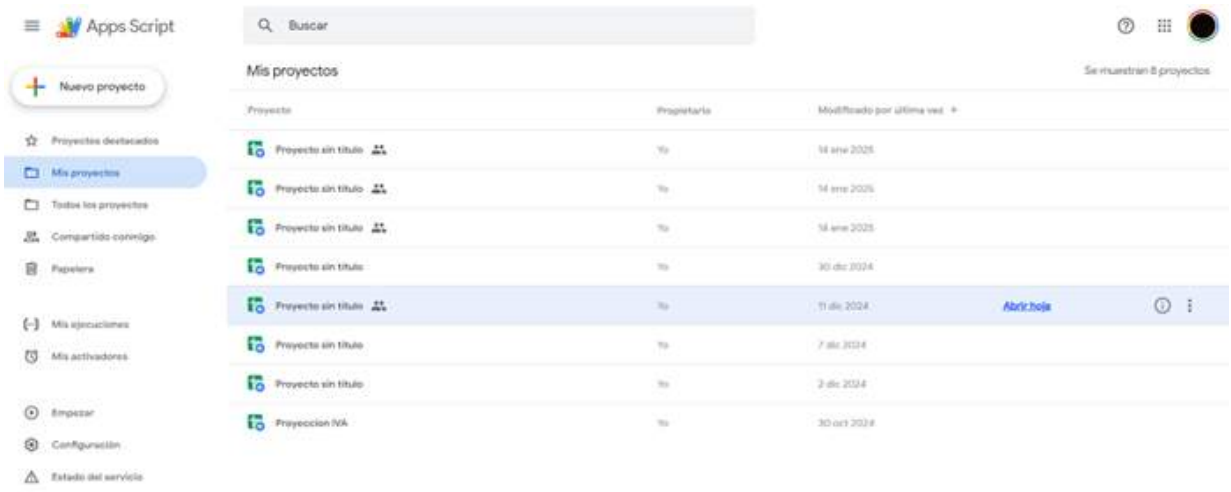
Fuente: captura de pantalla de Google Sheets.

Al hacer esto, se abre una nueva pestaña del navegador con el editor de Apps Script. Ese proyecto queda

“vinculado” a la hoja desde la que lo abrimos, lo que significa que el código que escribamos ahí va a poder leer y modificar directamente esa hoja de cálculo.

También existen opciones de entrar a Apps Script como desde la URL general (script.google.com) y crear un proyecto “en blanco”, no vinculado a ningún archivo en particular. Desde Google Drive, añadiendo un nuevo archivo, se puede seleccionar App Script, o desde otro tipo de documento (como Docs, Forms o Slides), pero para la mayoría de las automatizaciones con Sheets, es más práctico entrar desde la propia planilla, como acabamos de describir.

Figura 3: Apps Script



Fuente: captura de pantalla de Apps Script.

Cuando se abre Apps Script por primera vez, la interfaz puede intimidar un poco, pero en realidad se organiza en pocas zonas clave:

Barra superior del proyecto

En la parte de arriba vemos lo siguiente:

- El nombre del proyecto (por defecto, algo como “Proyecto sin título”), que se puede cambiar haciendo clic sobre él.
- Botón de “Guardar” (es importante guardar cada cambio).
- Botones “Ejecutar”, “Depurar” y acceso a “Registros/Ejecuciones”.
- El botón “Implementar” (o “Deploy”), que se usa cuando queremos publicar el proyecto como app web, complemento, etc. Para automatizaciones simples dentro de Sheets, muchas veces ni hace falta tocarlo.

Barra lateral izquierda

Es la columna donde se organiza el proyecto. Lo más importante que se verá es lo siguiente:

- Editor de código/archivos: por defecto, aparece un archivo llamado “Code.gs” o “Código.gs”. Ahí es donde se escribe el código en JavaScript. Se pueden crear más archivos si la solución crece (por ejemplo, “utilidades.gs”, “email.gs”, etc.).
- Bibliotecas y servicios (dependiendo de la versión): permiten conectar el proyecto con API adicionales o servicios avanzados, pero para un curso inicial se pueden mencionar solo como “algo que existe y se usa en casos más complejos”.
- Ejecuciones: muestra el historial de corridas del script, si hubo errores, cuánto tardaron, etc.
- Activadores: es donde se configuran los disparadores (por ejemplo: “Ejecutar esta función todos los días a las 9:00” o “Ejecutar cuando se edita la hoja”). Esta sección es clave cuando se quiere pasar de “lo corro a mano” a “esto corre solo”.

Área central de edición

Es el corazón del entorno: un editor de texto donde aparece el código. Al crear un proyecto nuevo suele mostrarse una

función de ejemplo, parecida a lo siguiente:

```
function myFunction() {

}
```

Desde ahí, el usuario empieza a escribir o pegar las funciones que generó con ayuda de la IA. El editor incluye numeración de líneas, coloreado de sintaxis (palabras clave en distintos colores) y autocompletado básico.

Figura 4: Barra lateral izquierda de Apps Script

Fuente: captura de pantalla de Apps Script.

Un detalle importante para incluir es qué pasa la primera vez que se ejecuta una función de Apps Script:

- El usuario escribe o pega una función (por ejemplo, una que lea datos de la hoja activa).
- En la barra superior, el usuario debe seleccionar la función que vaya a ejecutar (si hay varias) y hacer clic en el botón “Ejecutar”.

La primera vez, Google muestra un diálogo de autorización:

- Le pide al usuario elegir la cuenta de Google (la cuenta que elijamos será la que ejecutará el script; por ejemplo, si es para enviar correos electrónicos, se enviarán desde la cuenta que ejecute la función).
- Muestra qué tipos de acceso va a tener el script (leer y modificar hojas de cálculo, enviar correos, etc.).
- El usuario debe aceptar para que el código pueda interactuar con sus archivos. Esta autorización solo se pide la primera vez que se ejecuta.

Este paso es clave para entender que Apps Script actúa con los permisos del usuario: no es una “magia externa”, sino un programa que, con nuestro permiso, automatiza

acciones que nosotros mismos podríamos hacer a mano en Google Workspace.

Si la ejecución falla, en la parte inferior o en la sección de "Ejecuciones", se puede ver el error concreto (por ejemplo, "TypeError: Cannot read properties of null"). Ese mensaje de error es oro para trabajar con IA: el usuario puede copiarlo y pegarlo en ChatGPT/Gemini para pedir ayuda en depuración.

2. Introducción a VBA

Ahora pasemos al mundo de Microsoft Excel, donde la herramienta clásica para automatizaciones es VBA (Visual Basic for Applications).

VBA es el lenguaje de programación incorporado en las aplicaciones de Microsoft Office (como Excel, Word, Access, etc.), que permite crear macros, es decir, secuencias de instrucciones automatizadas, para manipular documentos y datos de manera personalizada (García de Zúñiga, 2025). En el contexto de Excel, VBA nos permite escribir y ejecutar macros para automatizar prácticamente cualquier tarea dentro de una hoja de cálculo. Una macro de Excel puede hacer casi todo lo que haría un usuario manualmente (escribir en celdas, aplicar formato, crear gráficos, filtrar información, importar/exportar datos, etc.), pero de forma programada. Y lo que es mejor: si algo puede hacerse

una vez mediante VBA, entonces puede repetirse mil veces con la misma facilidad, cambiando solo los datos de entrada. Esta es la mayor potencia de VBA: tomar operaciones repetitivas o complejas y ejecutarlas de manera automática y consistente personalizada (Microsoft Learn, 2023).

Las posibilidades son enormes: prácticamente cualquier cosa que necesites hacer en Excel se puede automatizar con una macro, incluso se pueden llegar a extender funcionalidades de Excel más allá de lo estándar (García de Zúñiga, 2025).

Enumeremos ahora algunas aplicaciones comunes de VBA en entornos laborales:

- **Validación y manipulación de datos:** si en un libro de Excel se ingresa regularmente información (por ejemplo, registros de clientes, pedidos, etc.), una macro puede recorrer las filas para detectar errores o inconsistencias (campos vacíos, formatos incorrectos, valores fuera de rango, duplicados), y alertar al usuario o bien corregirlos automáticamente. También se pueden crear macros para transformar datos: por ejemplo, tomar un listado y

separarlo en varias hojas según criterios, combinar columnas, eliminar duplicados, etc., tareas que a veces no están disponibles mediante las funciones estándar de Excel pero que con unas líneas de código VBA se resuelven.

- **Integración con otras aplicaciones de Office:** VBA permite que Excel converse con otras aplicaciones de la suite Office, lo cual habilita flujos de trabajo integrados. Un ejemplo típico es generar comunicaciones por correo electrónico desde Excel: mediante VBA, se puede tomar una lista de contactos en una hoja y hacer que Outlook envíe un correo personalizado a cada uno (quizá adjuntando el PDF de reporte generado, como mencionamos antes). O, al revés, desde Outlook pasar datos a Excel. También es posible controlar Word o PowerPoint desde Excel —por ejemplo, crear documentos de Word rellenando plantillas con datos de la hoja, o armar presentaciones de PowerPoint automáticamente con tablas y gráficos provenientes de Excel—. Esta interacción entre aplicaciones expande aún más lo que se puede lograr en términos de automatización de tareas de oficina (Microsoft Learn, 2023).

En definitiva, VBA empodera al usuario avanzado de Excel para que la herramienta haga prácticamente cualquier cosa que se imagine, como lo hace App Script en Workspace pero dentro del entorno de Office: desde cálculos complejos con solo pulsar un botón, hasta pequeñas interfaces personalizadas (formularios con botones, cuadros de diálogo) para facilitar tareas específicas. Las macros pueden desencadenarse con atajos, botones o automáticamente ante eventos (por ejemplo, al abrir un archivo, al modificar un valor, etc.); ofrecen así mucha flexibilidad para adaptar Excel a las necesidades exactas de un proceso de negocio.

Una aclaración importante: no es necesario escribir un código en VBA para automatizar procesos. Existe la opción de grabar macros, llevando a cabo manualmente la serie de pasos para que quede guardada y luego se ejecute automáticamente. Si bien esta función puede ser más simple para tareas sencillas, limita la posibilidad de establecer condiciones más complejas dentro del flujo, cosa que podemos hacer escribiendo el código en VBA.

Limitaciones de VBA

A pesar de su gran poder, VBA tiene ciertas limitaciones y consideraciones importantes que debemos tener en cuenta:

- **Dependencia de la aplicación de escritorio:** las macros de VBA solo funcionan en las versiones de escritorio de Excel (Windows o Mac). No es posible crear ni ejecutar macros en Excel Online (la versión web) ni en las apps móviles. Esto significa que, si un archivo de Excel con macros (.xlsm) se abre en el navegador (por ejemplo, desde OneDrive o SharePoint en modo web), las macros simplemente no estarán disponibles. Para usar las automatizaciones, es necesario descargar el archivo y abrirlo en la aplicación Excel instalada en el equipo. Esta dependencia del entorno de escritorio limita la portabilidad de las soluciones VBA en entornos colaborativos en la nube.

- **Entorno propietario:** VBA es una tecnología propiedad de Microsoft. En la práctica, esto implica que solo puede ejecutarse dentro de Excel (u otras apps de Office); no se puede usar una macro de Excel para llamar las API de aplicaciones externas así como lo hace App Script.

- **Seguridad y políticas corporativas**: una limitación práctica de las macros VBA es que, debido a su poder, representan un riesgo de seguridad si se usan con fines maliciosos. Por ello, muchas organizaciones bloquean o restringen la ejecución de macros por defecto en los documentos. Al abrir un archivo con macros, Excel suele mostrar advertencias de seguridad, y solo usuarios con cierta confianza habilitan su ejecución. En entornos corporativos, es común que el Departamento de TI establezca políticas para deshabilitar macros descargadas de Internet o habilitarlas solo si están firmadas digitalmente por proveedores confiables. Todo esto significa que, a pesar de que técnicamente una macro puede hacer muchas cosas, es posible

que el entorno de trabajo no permita usarla libremente sin configuraciones adicionales, lo que puede frenar la adopción de automatizaciones vía VBA si no se gestionan adecuadamente las cuestiones de seguridad.

- **Mantenimiento y escalabilidad:** similar a Apps Script, las soluciones VBA pueden volverse difíciles de mantener si crecen mucho. El código VBA suele vivir dentro de un archivo de Excel específico; manejar versiones del código, depuración avanzada o colaboración entre varios desarrolladores es complicado en este modelo. Además, las macros ejecutándose dentro de Excel comparten recursos con la aplicación; si una macro consume mucha memoria o CPU, puede hacer que Excel quede temporalmente congelado mientras corre. Para grandes volúmenes de datos o procesos muy pesados, VBA puede no ser la opción más eficiente. No obstante, para la gran mayoría de automatizaciones de complejidad media o baja en Excel, VBA es suficiente y ha demostrado ser una herramienta fiable a lo largo de décadas.

Acceder al editor

Para poder trabajar con macros, es necesario conocer dónde vive VBA dentro de Excel y qué se ve al abrir el entorno de programación.

Activar la pestaña “Desarrollador”

En muchas instalaciones de Excel, la pestaña Desarrollador (“Developer”) no aparece visible por defecto en la cinta de opciones. Esta pestaña es importante porque allí se

encuentran accesos directos tanto al Editor de Visual Basic como a la Grabadora de macros.

El proceso típico para habilitarla (puede variar levemente según la versión) es:

- Ir a “Archivo” → “Opciones”.
- Seleccionar “Personalizar cinta de opciones”.
- En el listado de pestañas, marcar la casilla “Desarrollador”.
- Confirmar con aceptar.

A partir de ese momento, en la parte superior de Excel aparecerá la pestaña “Desarrollador”, que será el punto de partida para la mayoría de las tareas con VBA.

Acceso al Editor de Visual Basic

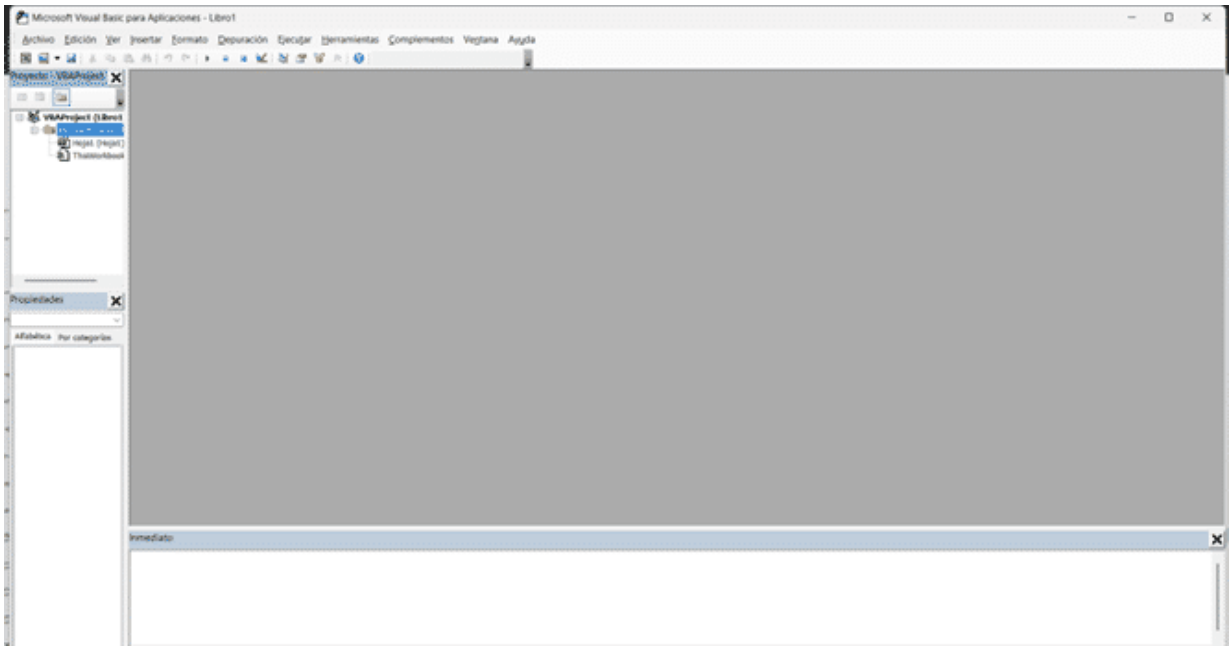
Hay dos caminos principales para entrar al editor donde se escribe y se ve el código:

- Desde el menú, haciendo clic en “Desarrollador” → “Visual Basic”.

- Usando el atajo de teclado ALT + F11, que abre directamente el Editor de Visual Basic (VBE).

Al hacerlo, se abre una nueva ventana separada de Excel. Esa ventana es el entorno donde se gestionan los módulos, formularios y el código VBA del archivo actual.

Figura 5: Microsoft Visual Basi

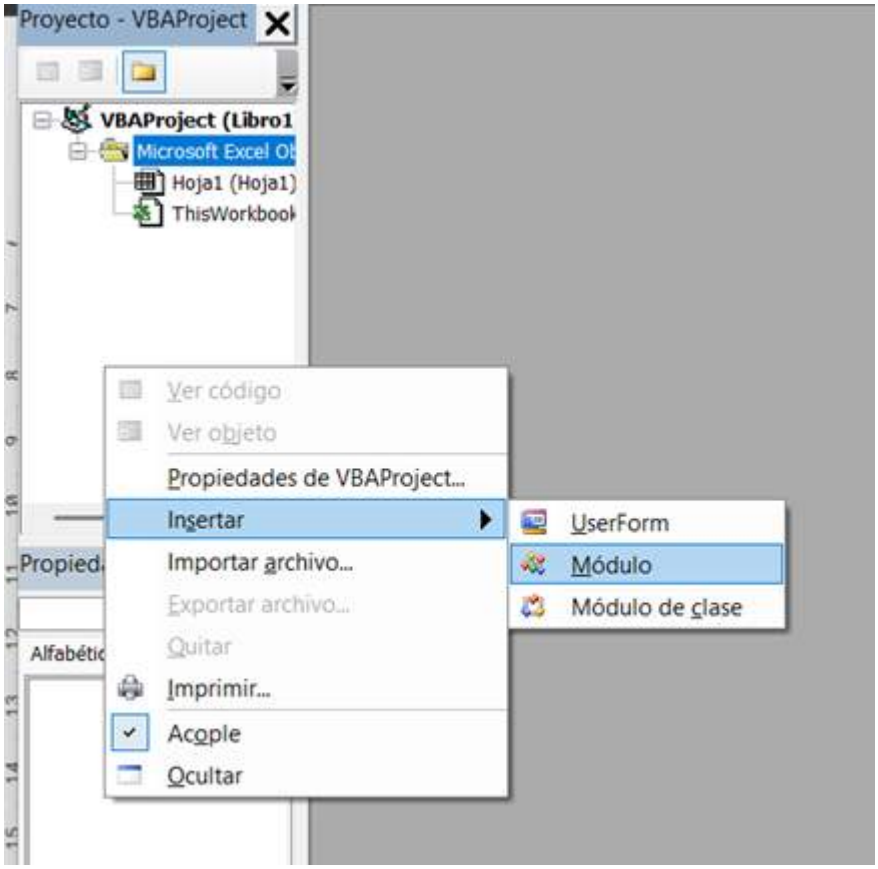


Fuente: captura de pantalla del Microsoft Visual Basic.

Una vez en esta pestaña, al hacer clic derecho sobre la ventana izquierda, podemos insertar un nuevo módulo, que

será donde podremos escribir o pegar nuestro código de VBA.

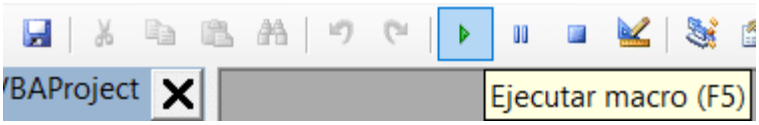
Figura 6: Inserción de nuevo módulo



Fuente: captura de pantalla de Microsoft Visual Basic.

En la barra superior, entre muchas opciones, tenemos la de Guardar el proyecto" (importante, ya que no se guarda automáticamente) y "Ejecutar el código".

Figura 7: Ejecutar el código



Fuente: captura de pantalla de Microsoft Visual Basic.

3. El papel de la IA en estas automatizaciones

Habiendo explorado las herramientas clásicas de automatización (Apps Script y VBA), surge una pregunta importante: ¿qué rol juega la inteligencia artificial en este contexto?

Es crucial aclarar que ni Google Apps Script ni VBA incorporan IA de forma nativa en su funcionamiento. Es decir, cuando escribimos un script de Apps Script o una macro en VBA, estamos programando secuencias de instrucciones determinísticas, sin ninguna magia de inteligencia artificial detrás de escena; son un código tradicional escrito por un humano.

Sin embargo, la IA puede intervenir como asistente para facilitar la creación de ese código. Aquí es donde entran en

juego modelos de lenguaje avanzados como ChatGPT. Herramientas de IA generativa pueden entender instrucciones en lenguaje natural y producir código en distintos lenguajes de programación. En la práctica, esto significa que hoy en día es posible pedirle a una IA que escriba por nosotros un script de Apps Script o una macro de Excel, simplemente describiendo lo que queremos lograr con palabras comunes (Tovar, 2023). Por ejemplo, un usuario podría decirle a ChatGPT: “Genera un código en Google Apps Script que lea una hoja de cálculo con nombres en la columna A y direcciones en la columna B, y envíe un correo electrónico personalizado a cada dirección”, y el modelo producirá un guion en JavaScript listo para copiar y pegar en Apps Script. De igual modo, alguien sin conocimientos de Visual Basic podría solicitar: “Escríbeme una macro de Excel que tome la tabla activa y la exporte a un PDF ordenado por fecha”, y obtendría un código VBA funcional como respuesta.

Lo realmente revolucionario de este enfoque es que pone la automatización al alcance de cualquier persona, incluso sin saber programar. Antes, la barrera para automatizar era conocer el lenguaje (JavaScript, VBA, etc.) y la sintaxis exacta de las funciones. Ahora, la IA hace de puente entre la idea en la mente del usuario y el código necesario en la máquina. Cualquier profesional puede describir su problema o necesidad, y obtener un script funcional generado por IA.

Desde luego, sigue siendo recomendable que el usuario revise y pruebe el código generado antes de implementarlo en producción, pero la carga pesada de escribir la estructura del programa se ve tremendamente reducida.

Otra ventaja importante de utilizar IA como asistente de código es la iteración y la depuración guiada. Supongamos que le pedimos a ChatGPT una macro y, al probarlo en Excel, algo falla o produce un error. En lugar de abandonar, el usuario puede proporcionarle a la IA el mensaje de error o describir el comportamiento incorrecto, y pedirle que corrija el código. ChatGPT es capaz de entender el error y ajustar el código en consecuencia, refinándolo paso a paso (Tovar, 2023). Este ciclo iterativo (escribir código, probar, depurar) se acelera enormemente con un asistente que habla nuestro idioma y conoce las soluciones habituales. En esencia, la IA actúa como un tutor o compañero de programación, que nos ayuda a superar obstáculos técnicos sin necesidad de buscar en foros durante horas.

Es importante señalar que, si bien la IA genera el código, la responsabilidad final de su correcto funcionamiento recae en nosotros. La IA puede cometer equivocaciones o malinterpretar alguna instrucción sutil, por lo que sigue siendo necesario validar que el script hace exactamente lo deseado y ajustarlo si es preciso. Aun así, la productividad que brinda es altísima: tareas de codificación que podrían tomar un día entero a un novato o que alguien sin conocimiento directamente no podría realizar ahora pueden resolverse en minutos con asistencia de la inteligencia artificial. Para este tipo de usos de la IA, hay que poner de relieve cuán importante son las instrucciones que le demos y lo detallistas que seamos a la hora de describir el proceso que queremos automatizar, ya que cada flujo es distinto y tiene sus excepciones, y ChatGPT no puede adivinarlas.

A partir de esto, creamos un GPT especializado en asistir en el desarrollo de automatizaciones con App Script y VBA. Una vez descrita la tarea, el asistente hará algunas preguntas para obtener más información y brindará el código necesario listo para pegarse en el editor.

Figura 8: Asistente IA de automatizaciones

Fuente: captura de pantalla de Asistente de automatizaciones (https://chatgpt.com/g/g-lNbzyijPy-asistente-de-automatizaciones).

Ahora la falta de conocimientos de programación ya no es un impedimento para aprovechar al máximo el potencial de automatizar nuestras hojas de cálculo: con una buena descripción en lenguaje natural y un par de iteraciones guiadas por IA, cualquiera puede construir soluciones que antes requerían un experto en código (Tovar, 2023).

Nota aclaratoria sobre uso de IA

Este material fue asistido con herramientas de IA generativa para tareas de borrador, síntesis, reescritura y apoyo en la organización de contenidos. Cada sección fue revisada, editada y validada por el equipo humano, que verificó la precisión conceptual, la coherencia pedagógica y las fuentes

citadas. Se invita a contrastar con las referencias bibliográficas incluidas y la documentación oficial. Dado que los modelos de IA evolucionan con rapidez, ciertas especificaciones técnicas podrían actualizarse; este texto refleja el estado del conocimiento al momento de su elaboración.

Referencias

Automation Anywhere. (s. f.). *¿Qué es la Automatización de Excel?*. https://www.automationanywhere.com/la/rpa/excel-automation

García de Zúñiga, F. (2025). *¿Qué es VBA y cómo funciona en Excel? Arsys*. https://www.arsys.es/blog/que-es-vba-y-como-funciona-en-excel

Google for Developers. (s. f.). *Cuotas para los servicios de Google*. https://developers.google.com/apps-script/guides/services/quotas

Google Workspace. (s. f.). *Apps Script*. https://workspace.google.com/intl/es/products/apps-script/

Maestro, L. (2020). *Automatiza tus tareas de ofimática con Google Apps Script*. Revoluciona tu Pyme. https://revolucionatupyme.com/google-apps-script/

Microsoft Learn. (2023). *Introducción a VBA en Office*. https://learn.microsoft.com/es-es/office/vba/library-reference/concepts/getting-started-with-vba-in-office

Tovar, E. (2023). *Usando ChatGPT de OpenAI para crear una macro de Excel para un modelo inmobiliario*. Adventures in CRE. https://www.adventuresincre.com/usando-chatgpt-de-openai-para-crear-una-macro-de-excel-para-un-modelo-inmobiliario/

CONTINUAR