Guía paso a paso para incrustar un agente de Copilot en una app de Power Apps

(buscando cero costo en desarrollo)

1) Crea tu entorno gratuito de desarrollo

- Da de alta el **Power Apps Developer Plan** (gratis). Te da un entorno personal con Power Apps, Power Automate y Dataverse para **construir y probar** sin costo. Microsoft Learn+1

2) Prepara tu app para Copilot

- Abre tu **canvas app** en Power Apps Studio.
- Ve a **Settings → Updates → Preview** y activa **App Copilot** (o "Copilot component"). Esto habilita el **Copilot control** dentro de tu app. Microsoft Learn+1

3) Crea el agente en Copilot Studio (sin pagar durante la fase de prueba)

- Regístrate en **Copilot Studio** con una **prueba gratuita** (self-service). Podrás **crear y publicar** un agente en tu entorno. La prueba puede extenderse y, si expira, el agente sigue funcionando por un periodo de gracia. Microsoft Learn+1

4) Publica y comparte el agente en el mismo entorno

- Publica el agente y compártelo con tu usuario "maker" (y con quienes vayan a editar/probar). Debe existir **en el mismo entorno** que tu app para conectarlo como "custom copilot". Microsoft Learn

5) Inserta el Copilot control en la app y conéctalo al agente

- Inserta el control **Copilot** en tu lienzo y, en sus propiedades, selecciona tu agente publicado. Opcional: activa "Edit in Copilot Studio" para refinar el comportamiento sin salir de Power Apps. Microsoft Learn+1

Con eso, en **desarrollo** ya tienes la experiencia de chat del agente **dentro** de tu canvas app, sin costos de licencias productivas mientras pruebas.

¿Se puede crear y consumir la app sin pagar en producción?

Corto y directo: hoy no.

- El Developer Plan es **solo para desarrollo/pruebas** individuales. Para **compartir** la app con usuarios finales necesitas planes de Power Apps (p. ej., **Premium** por usuario o por app). MicrosoftMicrosoft Learn

- **Copilot Studio** requiere licencia/ capacidad para **publicar y operar** agentes en canales productivos (sean "message packs" o **pago por mensaje**). Hay derechos limitados si ya tienes **Microsoft 365 Copilot**, pero **no** cubren crear/publicar agentes "en cualquier canal" ni el uso general fuera del contexto M365. Microsoft Learn+1
- El **Copilot control** de canvas app usa capacidades premium; en la práctica, los **usuarios que consumen la app** necesitan licencia Premium (o capacidad equivalente) para acceder a esa funcionalidad. Microsoft Learn+1

Ruta de menor costo recomendada para producción

1. **Power Apps Premium por app** para los **consumidores** de la app (licencia mínima por cada usuario de negocio de esa app). Microsoft Learn
2. **Copilot Studio** con **pago por mensaje** (PAYG) o **message packs** según el volumen (el "mensaje" es la unidad de consumo del agente). Úsalo solo en los escenarios donde aporte claro valor. Microsoft
3. Mantén el resto de la solución en **Dataverse** (governance) y define DLP/Managed Environment si aplica (ten en cuenta sus requisitos de licencia). Microsoft Learn

---

Alternativas "híbridas" si no usas Copilot Studio

A) ChatGPT / Azure OpenAI detrás de Power Automate o un conector personalizado

Arquitectura:
Power Apps (canvas) → **Power Automate flow** (HTTP) o **Custom Connector** → **Azure Function / APIM** → **Azure OpenAI** (o **OpenAI API**).

Pasos concretos (resumen):

1. Crea un **flow** con acción **HTTP** que reciba prompt y devuelva la respuesta del modelo.
2. Desde la app, llama al flow con PowerAutomate.Run(UserPrompt) y pinta la respuesta en un control de texto.
3. En Azure Function/APIM agrega: *prompt templates*, recorte de tokens, cache, y *content filtering*.
4. Comienza con modelos "mini" para **optimizar costo** y sube solo si hace falta.

Costos: son **pago por tokens**. Refiere precios oficiales de **Azure OpenAI** o **OpenAI API** y elige el modelo más económico viable para tu caso. Microsoft AzureOpenAI

Ventajas: control fino de costos, despliegue sin Copilot Studio.
Consideraciones: gobierno (DLP), manejo de secretos, latencia.

B) MCP (Model Context Protocol) para conectar datos/acciones de forma estandarizada

- **MCP** es un protocolo abierto para que los LLMs accedan a **recursos** (datos) y **herramientas** de forma uniforme. Ya está **GA** en Copilot Studio, e incluso existe **Dataverse MCP server**. modelcontextprotocol.ioMicrosoftMicrosoft Learn

Dos formas de usarlo:

1. **Con Copilot Studio**: extiende tu agente conectándolo a servidores MCP (por ejemplo, Dataverse MCP), y luego **incrústalo** en tu app con el Copilot control. Microsoft Learn
2. **Sin Copilot Studio**: usa un **cliente MCP** (p. ej., en VS Code / apps propias) y expón tus **fuentes internas** como servidores MCP; el LLM (incluido ChatGPT compatible/otros) consumirá esos recursos/acciones con menos *glue code*. modelcontextprotocol.io

Ventaja: portabilidad y menor dependencia de SDKs propietarios para "federar" contexto y herramientas.
Consideración: aún requerirás pagar por el modelo subyacente (Azure OpenAI / OpenAI), pero con **MCP** escalarás más limpio.

Recomendación final (pragmática)

1. **Desarrolla y prueba gratis**: Developer Plan + prueba de Copilot Studio + Copilot control en canvas app. Microsoft Learn+2Microsoft Learn+2
2. **Pasa a producción con bajo gasto**:
    - Usuarios finales en **Power Apps Premium por app** (solo para quienes realmente usan la app). Microsoft Learn
    - **Copilot Studio PAYG por mensaje** para el agente (dimensiona por casos de uso críticos). Microsoft
3. Si el presupuesto sigue apretado o necesitas más control, arranca con la **arquitectura ChatGPT/Azure OpenAI vía Power Automate** y evalúa **MCP** para estandarizar el acceso a datos/acciones. Microsoft Azuremodelcontextprotocol.io