


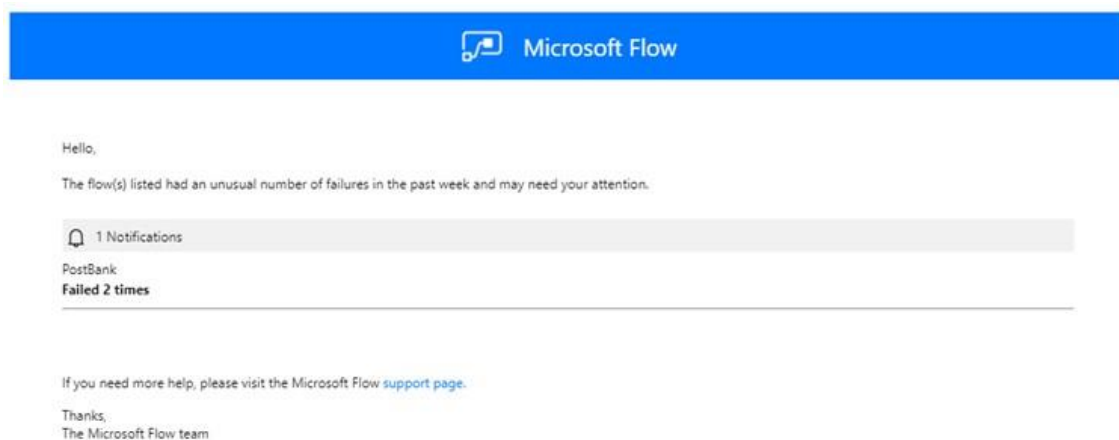
Gestión de errores avanzada en Power Automate

En [Power Automate](#) podemos diseñar multitud de procesos y como tales, están sujetos a errores como cualquier otro proceso automatizado. Es aconsejable que todo proceso diseñado en Power Automate gestione los posibles errores que pueden producirse para realizar acciones alternativas o de notificación.

De manera predeterminada, Power Automate nos ofrece un historial de ejecución de los últimos 28 días de nuestro proceso donde visualmente podemos saber si su ejecución ha finalizado correctamente o con errores

| 28-day run history ⓘ | | |  All runs |
|------------------------------|----------|-----------|--|
| Start | Duration | Status | |
| May 12, 10:18 AM (1 min ago) | 43 ms | Succeeded | |
| May 12, 10:18 AM (2 min ago) | 33 ms | Failed | |
| May 12, 10:18 AM (2 min ago) | 04 ms | Succeeded | |
| May 12, 10:18 AM (2 min ago) | 22 ms | Succeeded | |

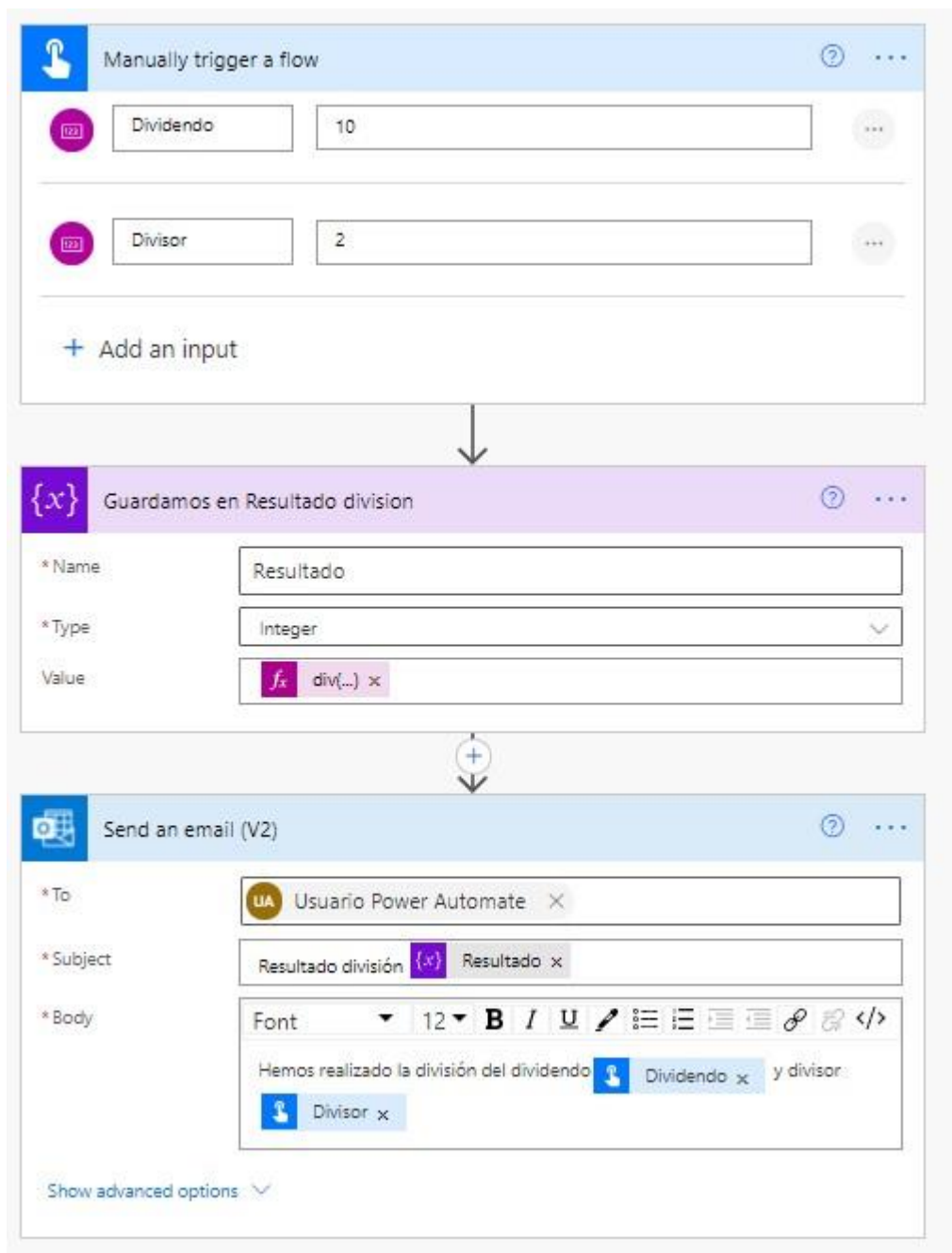
Adicionalmente, Power Automate nos envía un email semanalmente con el listado de nuestros procesos que han finalizado de manera incorrecta. El correo que recibimos nos indica el nombre del proceso y las veces que ha fallado.



Si queremos gestionar dentro de nuestro proceso los errores que se producen, Power Automate nos ofrece diversas acciones para conseguirlo. Para verlas, lo haremos sobre un proceso muy sencillo en Power Automate que realizará la división entre dos números y enviará por email el resultado. Esto es un ejemplo muy básico, pero nos permite ver rápidamente como gestionar las excepciones y todo lo que hagamos se puede emplear para cualquier proceso que diseñemos.

Nuestro proceso de división tendrá un trigger manual donde solicitará al usuario que lo ejecuta dos números, el dividendo y divisor. Una vez solicitado, nos creamos una variable también numérica llamada “Resultado”, donde guardaremos el resultado de la división. Para hacer la división haremos uso de expresiones. En nuestro caso la expresión, es `div(triggerBody()['number'],triggerBody()['number_1'])`

Nuestro proceso quedará de la siguiente manera



Si lo ejecutamos tendremos los siguientes resultados:

- El usuario indica valor 10 como dividendo y 2 como divisor. El proceso finaliza correctamente.
- El usuario indica valor 10 como dividendo y 0 como divisor. El proceso finaliza con error.

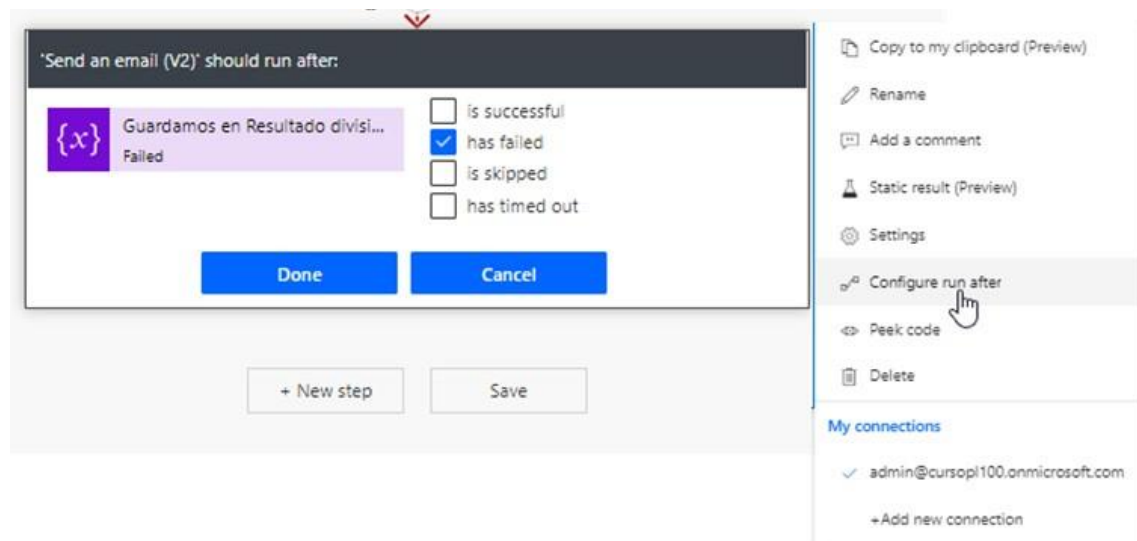
Este proceso sólo sabríamos si ha fallado viendo el historial o por el correo semanal.

Power Automate nos permite implementar el famoso "try-catch-finally", donde podemos intentar ejecutar una acción, captura el error y ejecutar acciones al finalizar. Esto lo

conseguimos a través de la combinación de emplear la acción “Scope” y de la opción “**Configure run after**” que disponemos en las acciones.

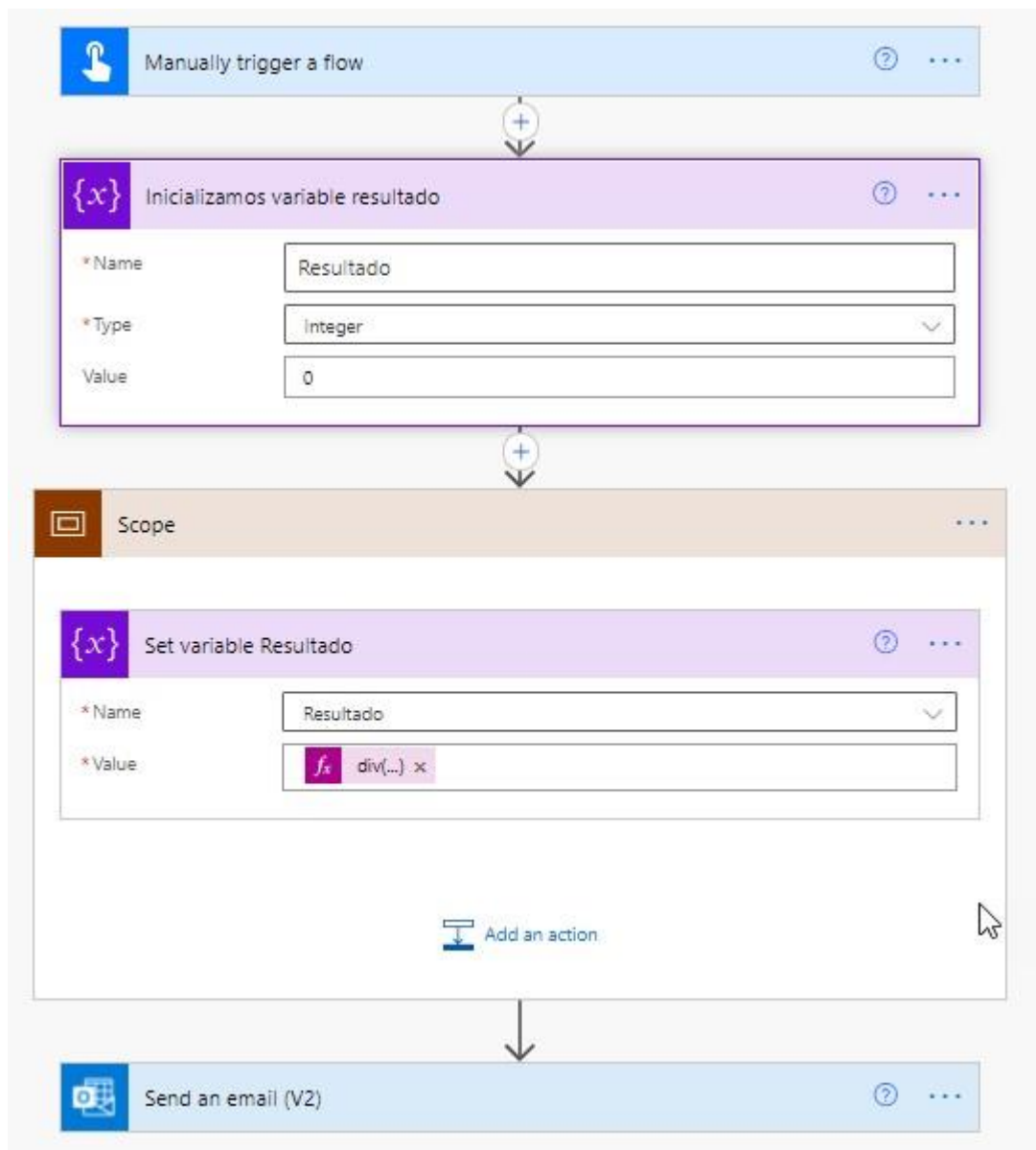
La acción “Scope” es un contenedor de acciones donde podemos añadir las acciones que queremos realizar y capturar si hay un error. La opción “**Configure run after**” nos permite indicar si una acción debe de ejecutar en función del resultado de ejecución de su acción predecesora. Disponemos de cuatro opciones posibles:

- Ejecutar mi acción si su predecesora se ha ejecutado correctamente.
- Ejecutar mi acción si su predecesora ha tenido un error
- Ejecutar mi acción si su predecesora ha dado error de timeout
- Ejecutar mi acción si su predecesora se ha omitido



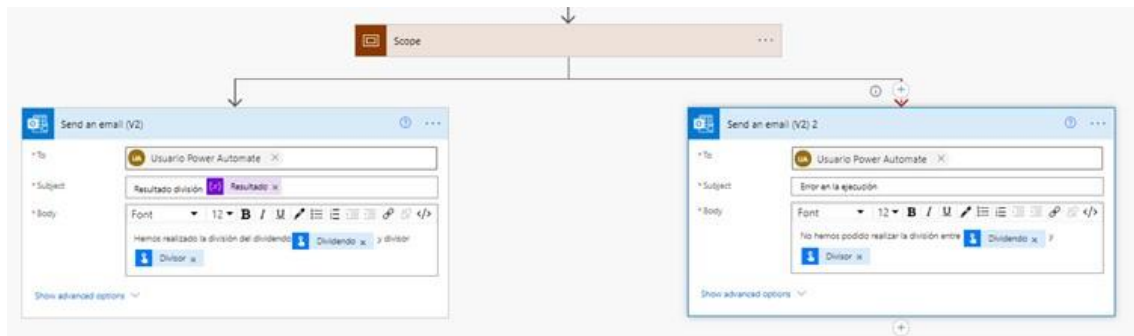
En base a lo anterior, vamos a modificar nuestro proceso para capturar el error de la acción de la división, para ello vamos a añadir una acción de “Scope”, que sería el “try”. Una de las limitaciones que tiene “Scope” es que no podemos incluir la declaración de variable, por tanto, después de inicializar la variable “Resultado” incluimos la acción “Scope”, en la inicialización de la variable resultado quitamos la expresión de la división y establecemos un valor inicial como puede ser el 0. Dentro de “Scope” agregamos la acción de guardar en la variable resultado la división con la misma expresión que teníamos. Este ejemplo solo tenemos una única acción dentro de “Scope” pero puede haber múltiples acciones.

Nuestro proceso quedaría de la siguiente manera



Si ejecutamos el proceso y generamos un error, veremos que estamos en la misma situación que al principio. Queda pendiente añadir la ejecución de acciones en caso de error. Para ello después de la acción “Scope” añadimos una rama paralela, en nuestro ejemplo será un email notificando que se ha producido un error, pero podríamos realizar cualquier acción. En las opciones de la acción de enviar email que hemos añadido en la rama paralela establecemos en “Configure run after” lo haga después de que haya un error en su acción anterior, en nuestro caso el “Scope”.

Ahora tenemos el proceso de la siguiente manera

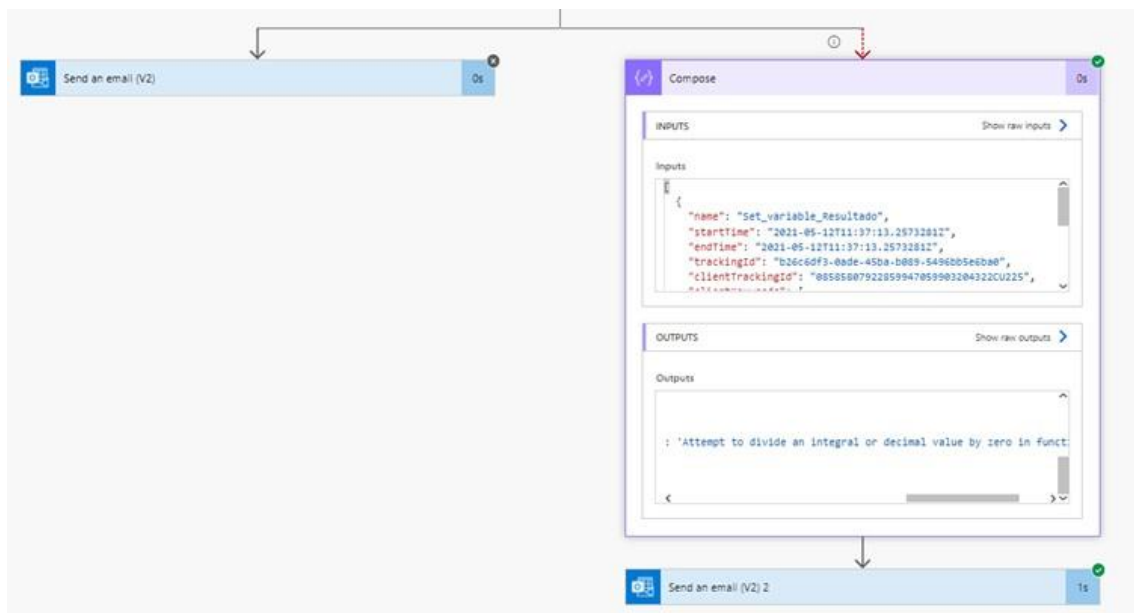


Por un lado, si el “Scope” se ejecuta correctamente envía el email con el resultado y si hay error, enviar un email notificando del error.

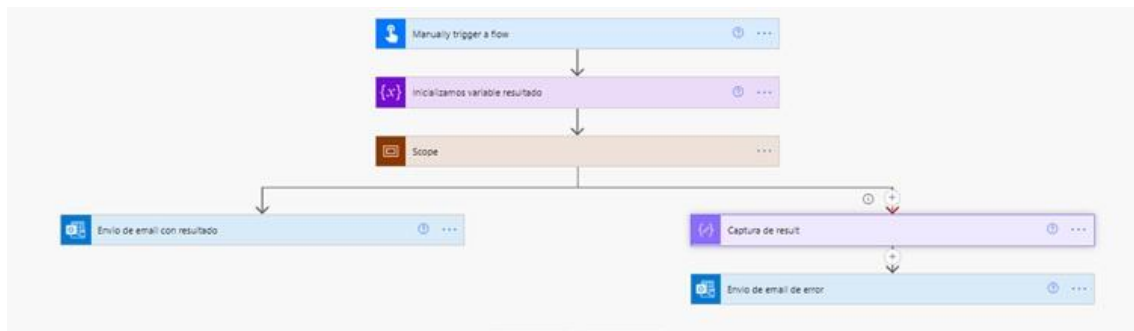
Si queremos ver más detalle de cuál ha sido el error que se ha producido porque por ejemplo queremos incluirlo en el email de aviso o en función del mismo realizar determinadas acciones, tenemos una función llamada “Result()”, el cual nos devuelve el resultado de ejecución de un “Scope”.

Si quiero ver esta información en nuestro ejemplo, podemos añadir una acción “Compose” antes de enviar el email de error y establecer como valor la expresión “Result(‘Scope’)”. Ahora hemos de volver a indicar que esta acción debe de ejecutarse tras el error del “Scope”

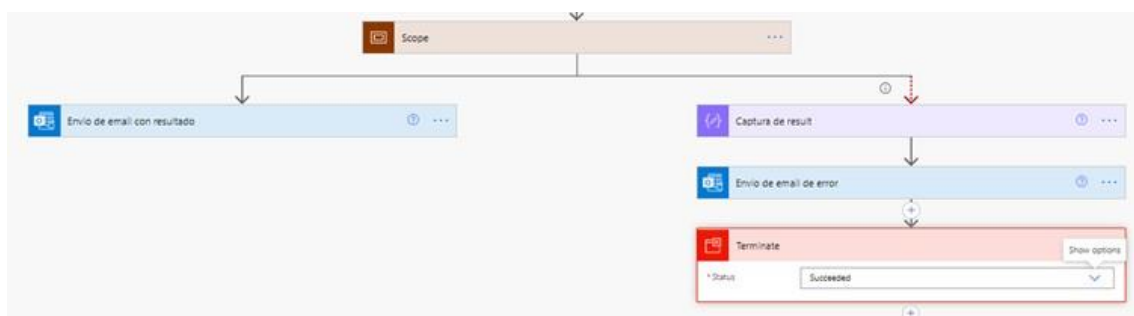
En nuestro ejemplo quedaría de la siguiente manera. Como vemos, nos da información de que acción ha fallado



Finalmente, nuestro proceso ha quedado de la siguiente manera



Si queremos evitar que cuando haya un error en nuestro proceso, quede registrado en el historial de ejecución como proceso finalizado con error. Power Automate nos permite a través de la acción “**Terminate**” establecer que nuestro proceso ha finalizado correctamente.



De esta manera, si lo ejecutamos y se genera un error, este es capturado y además marcamos que el proceso se ha ejecutado correctamente porque el error ha sido capturado y procesado.