Trabajo en Grupo 2º Trimestre

ÍNDICE

1. TEORÍA

1.1. ¿QUÉ ES?

Low-code es un enfoque de desarrollo de software que permite crear aplicaciones con poca necesidad de escribir código manualmente. En lugar de programar línea por línea, los desarrolladores usan interfaces visuales para diseñar y construir aplicaciones arrastrando y configurando componentes.

1.2. ¿CUÁL FUE LA PRIMERA PLATAFORMA LOW CODE?

La primera plataforma Low-code reconocida en el mercado fue OutSystems, fundada en 2001.
Sin embargo, la idea de simplificar el desarrollo sin escribir código viene desde los años 80 y 90 con herramientas como:

- **CASE (Computer-Aided Software Engineering)**: Herramientas para automatizar el desarrollo de software.
- **4GL (Fourth-Generation Languages)**: Lenguajes de programación que simplificaban tareas comunes, como SQL.
- **RAD (Rapid Application Development)**: Modelos como Visual Basic y Delphi que aceleraban la creación de aplicaciones.

Evolución del Low-code

- **2001** → Nace OutSystems, considerada la primera plataforma Low-code moderna.
- **2005-2010** → Aparecen plataformas como Mendix y **Appian**, enfocadas en aplicaciones empresariales.
- **2015-2020** → Microsoft Power Apps, Retool y otras herramientas hacen el Low-code más accesible.
- **Hoy** → El Low-code está en auge y se combina con IA para facilitar aún más el desarrollo.

1.3. ¿PARA QUÉ SE SUELE UTILIZAR?

1. **Aplicaciones empresariales**
    - **CRM y ERP personalizados** → Sistemas de gestión de clientes o recursos internos.
    - **Dashboards y reportes** → Visualización de datos en tiempo real.
    - **Gestión de empleados** → Apps de seguimiento de desempeño, vacaciones y evaluaciones.
    - **Herramientas internas** → Aplicaciones para mejorar la productividad dentro de una empresa.

    Ejemplo: Una empresa crea un CRM en Power Apps para gestionar clientes sin depender de un equipo de desarrollo.

2. **Automatización de procesos y flujos de trabajo**
    - **Automatización de aprobaciones** → Solicitudes de vacaciones, compras, reembolsos.
    - **Gestión de documentos** → Flujos de aprobación de contratos o generación de informes automáticos.
    - **Bots y asistentes virtuales** → Chatbots para atención al cliente sin necesidad de IA avanzada.

    Ejemplo: Un banco usa Appian para automatizar la validación de documentos en solicitudes de crédito.

3. **Aplicaciones para clientes y usuarios finales**
    - **Portales de autoservicio** → Clientes pueden actualizar datos o hacer solicitudes sin intervención humana.
    - **Aplicaciones móviles y web** → Apps sencillas para servicios, reservas o encuestas.
    - **Sistemas de ticketing y soporte** → Plataformas para gestionar incidencias de clientes.

    Ejemplo: Un restaurante lanza una app para reservas creada en OutSystems en solo unas semanas.

4. **Integraciones y conectividad entre sistemas**
    - **Conexión con APIs y bases de datos** → Sin necesidad de escribir código complejo.
    - **Automatización entre herramientas** → Integraciones con ERP, CRM, Google Sheets, Slack, etc.
    - **Manejo de IoT** → Control de dispositivos conectados sin desarrollar un backend desde cero.

    Ejemplo: Una empresa usa Zapier (No-code) para sincronizar información entre Salesforce y Google Drive.

5. **Creación de MVPs y prototipos rápidos**
    - Validación de ideas antes del desarrollo completo.
    - Construcción rápida sin necesidad de inversión en programación avanzada.
    - Iteración y mejora sin depender de programadores constantemente.

    Ejemplo: Una startup crea su primer producto en Mendix para validar su idea antes de invertir en desarrollo tradicional.

1.4. VENTAJAS Y DESVENTAJAS FRENTE A OTRAS MANERAS DE PROGRAMAR

Ventajas del low-code

1. **Desarrollo e implementación acelerados:** Las plataformas de código bajo reducen significativamente la complejidad de la codificación, lo que permite un desarrollo e implementación más rápidos de las aplicaciones. Este ciclo de desarrollo rápido es un cambio radical para las empresas que necesitan automatizar o gestionar procesos y están interesadas en la eficiencia operativa.
2. **Reducción de los requisitos de habilidades técnicas:** Al simplificar el proceso de desarrollo, las plataformas de código bajo permiten que personas sin amplios conocimientos de codificación contribuyan al desarrollo de aplicaciones. Esta democratización del desarrollo puede generar resultados más colaborativos e innovadores.
3. **Rentabilidad:** El desarrollo de código bajo puede reducir sustancialmente los costos asociados con el desarrollo de aplicaciones al reducir la necesidad de un gran equipo de desarrolladores altamente capacitados. Esta eficiencia lo convierte en una opción atractiva para empresas de todos los tamaños.
4. **Mayor agilidad:** La facilidad de uso y la flexibilidad que ofrecen las plataformas de bajo código significan que las empresas pueden ajustar más fácilmente sus aplicaciones en respuesta a los comentarios o los requisitos cambiantes, lo que garantiza que sigan siendo ágiles en un panorama competitivo.
5. **Mayor productividad de TI:** Con plataformas de bajo código, los departamentos de TI pueden crear e iterar aplicaciones rápidamente, liberando tiempo para centrarse en proyectos más complejos y estratégicos que requieren una experiencia técnica más profunda.
6. **Facilidad de uso:** Como el desarrollo no comienza desde cero, el proceso se vuelve más simple, lo que le permite centrarse en los deseos y requisitos del usuario.
7. Los usuarios de Accelerated Development pueden personalizar y adaptar rápidamente los elementos clave y el código fuente de su

aplicación. Además, se pueden integrar y conectar aplicaciones, flujos de trabajo y procesos de sistemas existentes. Según la empresa de investigación y desarrollo de TI Forrester, las plataformas de código bajo pueden acelerar los proyectos hasta 20 veces más rápido que la codificación tradicional.

8. **Mayor automatización:** mediante reglas de decisión básicas, los usuarios pueden controlar automáticamente los flujos de trabajo, que luego pueden integrarse en varios sistemas de información. En muchos casos, las herramientas de código bajo utilizan inteligencia artificial y aprendizaje automático para automatizar los flujos de trabajo.

Limitaciones del low-code

1. **Preocupaciones sobre el rendimiento y la escalabilidad:** En el caso de aplicaciones que requieren altos niveles de rendimiento o escalabilidad, las plataformas de código reducido pueden resultar insuficientes. Las aplicaciones complejas, en particular aquellas con integraciones intrincadas con Salesforce, pueden requerir un desarrollo personalizado para satisfacer necesidades comerciales específicas.
2. **Personalización limitada:** Si bien las plataformas de código bajo ofrecen una variedad de opciones de personalización, es posible que no sean suficientes para empresas con requisitos altamente especializados. El director ejecutivo de RIBERATEC, Stepan Ovchinnikov, enfatiza la importancia del desarrollo personalizado en tales escenarios para lograr la funcionalidad y la experiencia de usuario deseadas.
3. **Dependencia de los proveedores de plataformas:** El uso de plataformas de código reducido genera una dependencia de los proveedores en lo que respecta a actualizaciones, seguridad y mejoras de funciones. Esta dependencia puede suponer riesgos si la hoja de ruta de la plataforma no se alinea con las necesidades cambiantes de la empresa.

4. **Pasar por alto las mejores prácticas:** La simplicidad de las plataformas de bajo código puede llevar a pasar por alto las mejores prácticas en el desarrollo de software, lo que potencialmente afecta la calidad y la capacidad de mantenimiento de la aplicación a lo largo del tiempo.
5. **Potencial de deuda técnica:** Un desarrollo rápido sin una consideración exhaustiva de la arquitectura subyacente puede generar deuda técnica. A medida que las empresas crecen, esta deuda puede convertirse en un problema importante, que requiere un esfuerzo considerable para abordarlo.

¿Cuándo elegir cada opción?

- **Low-code:**

  ✅ **Casos de uso:**

  - **Desarrollo rápido de aplicaciones** → Prototipos, MVPs o proyectos con plazos ajustados.
  - **Automatización de procesos empresariales** → RPA, integraciones entre sistemas sin mucha personalización.
  - **Falta de experiencia técnica** → Equipos sin programadores expertos.
  - **Aplicaciones internas** → Dashboards, formularios, CRM personalizados, etc.
  - **Reducción de costos** → Menos recursos en desarrollo y mantenimiento.

  ⚠️ **Limitaciones:**

  - **Poca flexibilidad y personalización** → Difícil para proyectos muy específicos o avanzados.
  - **Dependencia de la plataforma** → Vendor lock-in, licencias costosas.
  - **Menos control sobre rendimiento y seguridad** → Puede no ser ideal para aplicaciones críticas.

- **Programación tradicional:**

  ✅ **Casos de uso:**

  - **Proyectos altamente personalizados** → Aplicaciones con lógica de negocio compleja.
  - **Sistemas escalables y de alto rendimiento** → Aplicaciones con gran volumen de datos o tráfico.
  - **Desarrollo de software a largo plazo** → Equipos con experiencia técnica, mantenimiento y mejoras constantes.
  - **Mayor control sobre seguridad y arquitectura** → Crítico en fintech, salud, etc.
  - **Integraciones avanzadas** → APIs personalizadas, interacciones con hardware, IA, etc.

  ⚠️ **Limitaciones:**

  - **Mayor costo y tiempo de desarrollo** → Requiere más recursos y planificación.
  - **Necesidad de programadores experimentados** → No apto para equipos sin conocimiento técnico.

OUTSYSTEMS

OutSystems es una plataforma de desarrollo de aplicaciones *low-code*. Su propósito principal es facilitar y acelerar la creación de aplicaciones web y móviles sin necesidad de escribir grandes cantidades de código manualmente.

¿Qué se puede hacer con OutSystems?

- **Desarrollar aplicaciones rápidamente:** Puedes diseñar aplicaciones visualmente, arrastrando y soltando componentes.
- **Integración fácil:** Conectarte a bases de datos, servicios externos y APIs.
- **Despliegue automatizado:** OutSystems permite desplegar aplicaciones en la nube o en servidores locales.
- **Escalabilidad:** Es adecuada tanto para aplicaciones simples como complejas que necesitan escalar.

¿Para quién es útil?

- **Desarrolladores:** Para quienes desean optimizar su flujo de trabajo sin programar todo desde cero.
- **Empresas:** Que necesitan aplicaciones internas o para clientes de forma ágil.
- **Equipos de TI:** Que buscan mantener una infraestructura sólida para el desarrollo ágil.

Node-RED

Node-RED es una plataforma de desarrollo basada en flujos (*flow-based development*) diseñada para conectar dispositivos, servicios y APIs de manera sencilla.

¿Para qué sirve Node-RED?

- **Automatización:** Ideal para automatizar procesos en aplicaciones IoT, integración de sistemas y flujos de datos.
- **Conexión de dispositivos:** Puedes integrar sensores, bases de datos, APIs, servidores y servicios en la nube.
- **Procesamiento de datos:** Procesa y transforma datos en tiempo real.

Características clave

- **Desarrollo visual:** Permite arrastrar y soltar nodos (bloques de funcionalidad) en una interfaz gráfica.
- **Flexibilidad:** Basado en Node.js, lo que permite extender su funcionalidad con paquetes NPM.
- **Escalabilidad:** Puede ejecutarse desde un dispositivo pequeño (como una Raspberry Pi) hasta servidores robustos.

¿Para quién es útil?

- **Desarrolladores IoT:** Para quienes necesitan prototipar o integrar dispositivos rápidamente.
- **Empresas:** Que buscan automatizar procesos sin necesidad de programar en profundidad.
- **Entusiastas del hogar inteligente:** Ideal para automatizar sistemas de domótica.

2. SINTAXIS

El término "sintaxis" en el ámbito del low code se aleja de la idea tradicional de escribir código línea por línea en lenguajes como Python o Java, y se centra en un enfoque visual e interactivo para construir aplicaciones. Las plataformas de low code, como las que podrías haber explorado en tu investigación, ofrecen interfaces gráficas que funcionan como un lienzo digital. En este entorno, los usuarios no necesitan dominar comandos complejos ni estructuras sintácticas rígidas; en su lugar, trabajan con herramientas intuitivas que simplifican el proceso de desarrollo a través de una metodología de "arrastrar y soltar".

Dentro de la sintaxis nos encontramos con tres partes importantes, que forman la estructura básica de la programación Low Code. Es importante destacar que al ser un ámbito tan amplio nos centraremos en la plataforma node red.

Entrada (Inject): El primer componente clave es la Entrada, representada en Node-RED por el nodo "Inject". Este nodo actúa como el punto de partida de cualquier flujo, iniciando el proceso al enviar un mensaje, que puede entenderse como un paquete de datos. Por ejemplo, el nodo Inject puede simular una entrada del mundo real, como la lectura de un sensor de temperatura, un clic en un botón o incluso un evento programado, como un mensaje enviado cada minuto. Los usuarios configuran este nodo mediante una interfaz sencilla, definiendo si se activa manualmente (con un botón) o automáticamente según un temporizador. Esta flexibilidad convierte al nodo Inject en el disparador esencial que pone en marcha la lógica de la aplicación, sin requerir que el usuario escriba código para capturar eventos.

Procesamiento (Function): El mensaje pasa al nodo Function, que lo modifica. El function puede ser de muchos tipos pero modifican el contenido del paquete datos.

Existen multitud de nodos function, los mas usados son “Change”, “Delay”, “Function” y “Switch”.

Por ejemplo, el nodo "Change" permite modificar directamente el contenido del mensaje (como reemplazar un valor), mientras que "Delay" introduce una pausa antes de pasar al siguiente paso. El nodo "Function" es especialmente poderoso, ya que permite al usuario introducir pequeñas porciones de código JavaScript para personalizar la lógica, como sumar valores o filtrar datos. Por su parte, "Switch" actúa como un interruptor, dirigiendo el

mensaje a diferentes caminos según condiciones predefinidas. Esta etapa de procesamiento es el corazón del flujo, donde la creatividad y las reglas de negocio toman forma de manera visual y modular.

Salida (Debug): Procesa el mensaje final y lo presenta en la pestaña de depuración.

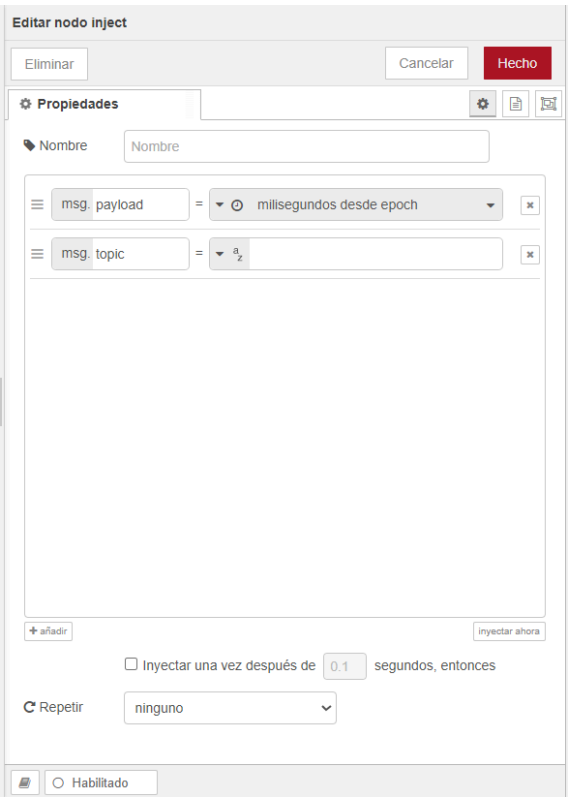
Finalmente, llegamos a la **Salida**, representada en este caso por el nodo "Debug". Este nodo recibe el mensaje final tras su transformación y lo muestra en una pestaña de depuración dentro de la interfaz de Node-RED. Es una herramienta esencial para verificar que el flujo funciona como se espera, presentando los datos procesados en tiempo real. Por ejemplo, si el flujo comenzó con un Inject que simulaba un sensor y pasó por un Function que calculó un promedio, el Debug mostrará el resultado final. Aunque "Debug" es un nodo de salida básico, Node-RED también permite conectar otros nodos de salida, como "HTTP" para enviar datos a una web o "MQTT" para comunicarse con dispositivos IoT, ampliando las posibilidades de aplicación.

3. EJEMPLO PRÁCTICO

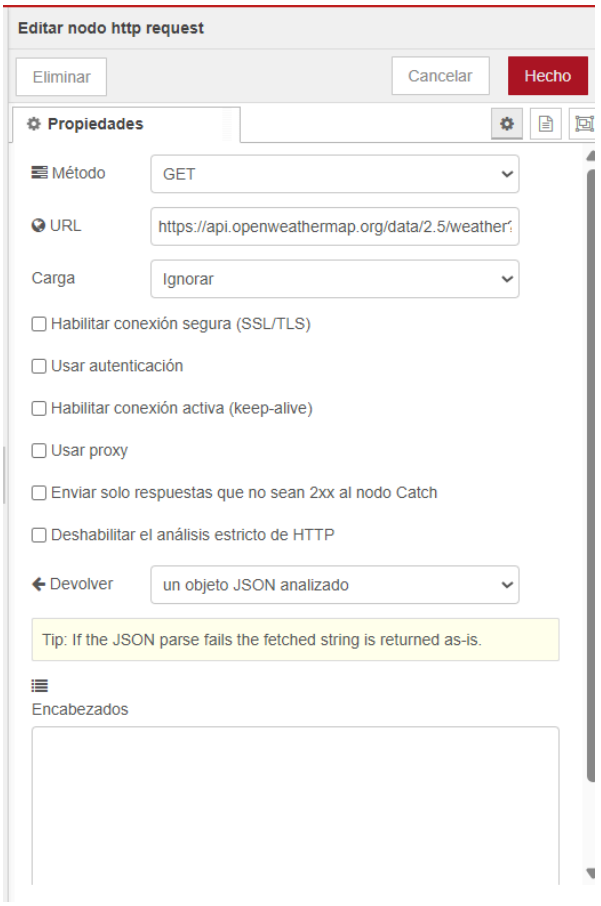
En cuanto al ejemplo práctico propuesto, se trata de una automatización sencilla mediante node-red, vamos a hacer un flujo configurado para recibir información del tiempo.

- Preparar el Entorno de Node-RED
  Abre Node-RED en tu navegador (normalmente accesible en http://localhost:1880 si está ejecutándose localmente). Asegúrate de tener el nodo "http request" instalado.

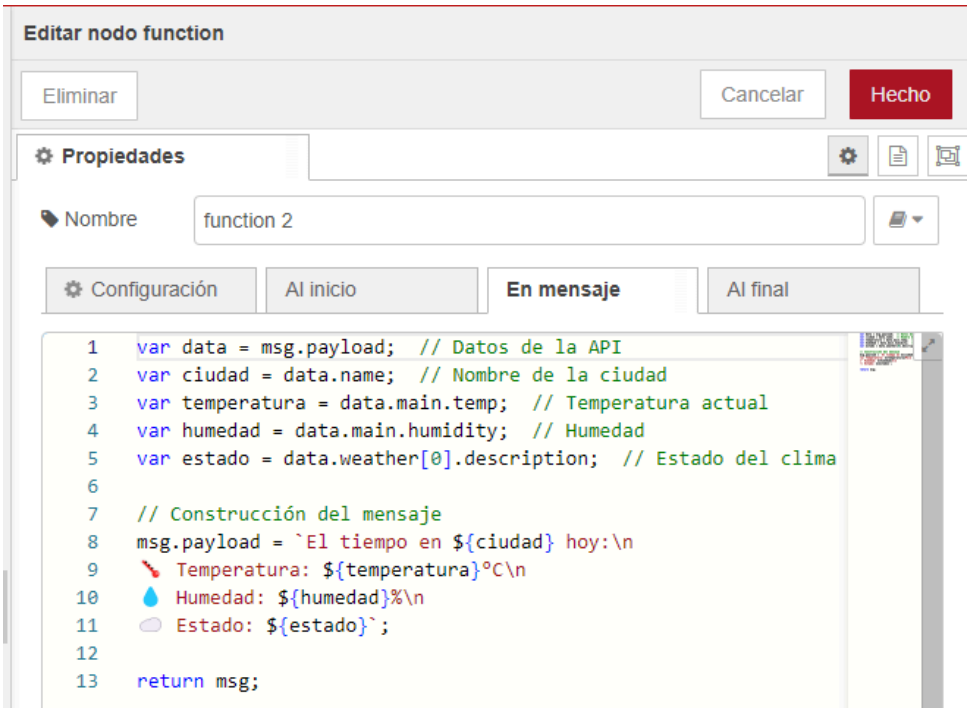
- Añadir el Nodo de Entrada (Inject - Marca Tiempo)
  Arrastra un nodo "Inject" desde la paleta a la izquierda al área de trabajo y nómbralo "marca tiempo".
  Haz doble clic para editar sus propiedades:
  En "Payload", selecciona "timestamp" (esto enviará la hora actual como dato inicial).
  En "Repeat", activa la opción "Injectar una vez después de" y establece un intervalo (por ejemplo, 60 segundos) para que el flujo se ejecute periódicamente.
  Habilita el nodo y haz clic en "Hecho".



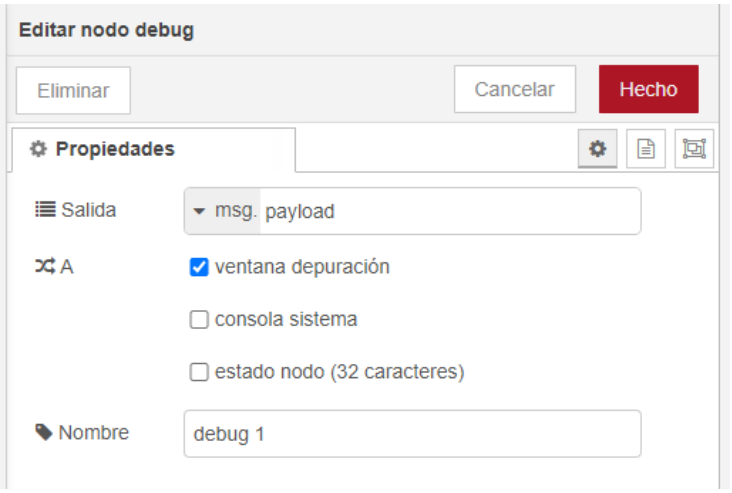
- Conectar el Nodo de Solicitud HTTP
  Arrastra un nodo "http request" desde la paleta y nómbralo "solicitud http".
  Conecta el nodo "marca tiempo" al nodo "solicitud http" con un cable.
  Edita las propiedades del nodo:
  Método: Selecciona "GET".
  URL: Ingresa
  https://api.openweathermap.org/data/2.5/weather?q=Seville&appid=tu_api_key,
  reemplazando "tu_api_key" con una clave válida de OpenWeatherMap.
  Retorno: Elige "un objeto JSON analizado".
  Desmarca "Habilitar conexión activa (keep-alive)" si no es necesario.
  Haz clic en "Hecho".



- Añadir el Nodo de Procesamiento (Function 2)
  Arrastra un nodo "Function" y nómbralo "function 2".
  Conecta el nodo "solicitud http" al nodo "function 2".
  Edita las propiedades:



- Añadir el Nodo de Salida (Debug 1)
  Arrastra un nodo "Debug" y nómbralo "debug 1".
  Conecta el nodo "function 2" al nodo "debug 1".
  Edita las propiedades:
  En "Salida", selecciona "msg.payload".
  Asegúrate de que "Ventana de depuración" esté marcada para mostrar los resultados.
  Haz clic en "Hecho".
  Este nodo mostrará el mensaje procesado en la pestaña de depuración.



Al ejecutar debe aparecer un mensaje como este:

4. OTROS EJERCICIOS BÁSICOS PROPUESTOS

4.1. Ejercicios encontrados o retos a proponer.

Reto propuesto:

Automatizar un mensaje para que se envíe a discord.

Resolver:

1. Crear un Webhook en Discord

1. Abre Discord y ve al servidor donde quieres enviar el mensaje.
2. Ve a **Configuración del servidor** > **Integraciones** > **Webhooks**.
3. Crea un nuevo webhook y copia la URL.

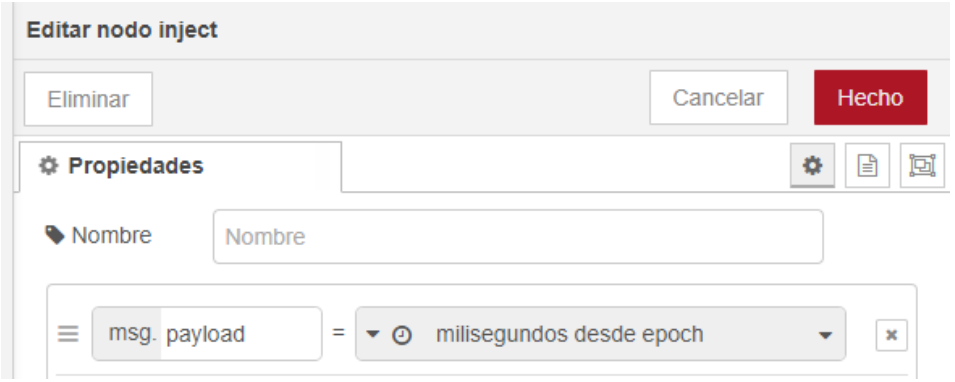
2. Configurar Node-RED

1. **Añadir un nodo inject** (para iniciar el envío del mensaje).
2. **Añadir un nodo function** (para configurar el mensaje).
3. **Añadir un nodo http request** (para enviar el mensaje a Discord).
4. **Añadir un nodo debug** (para ver la respuesta del servidor).

3. Configurar los nodos en Node-RED

Nodo `inject`

- Configurar con un temporizador o activación manual.

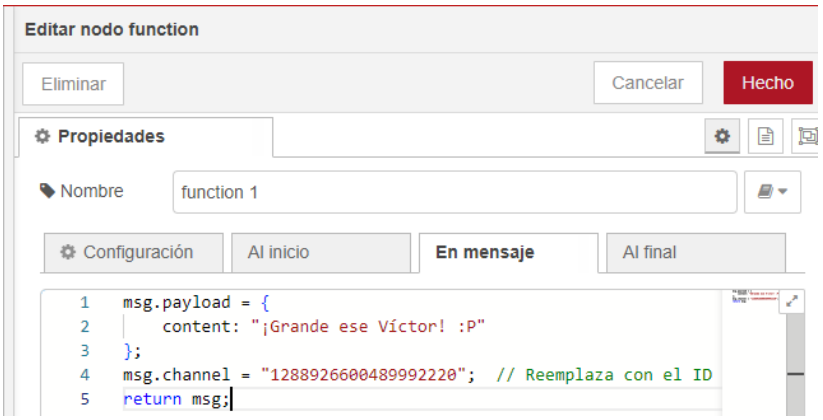


Nodo `function` (Configura el mensaje)

En este nodo, escribe lo siguiente:

```
msg.payload = {
    content: "¡Grande ese Víctor! :P"
};
```

```
msg.channel = "1288926600489992220";  // Reemplaza
con el ID del canal
return msg;
```



Nodo debug

- Para ver la respuesta de Discord.

🔧 Resultado

Cuando actives el flujo en Node-RED, enviará automáticamente el mensaje a tu canal de Discord.

4.2. Proyectos grandes encontrados.

Caso de Éxito: Heineken y OutSystems

- **Desafío**: Necesitaban digitalizar procesos de negocio en varias áreas, incluyendo logística, ventas y producción, sin depender exclusivamente de desarrolladores tradicionales.
- **Solución**: Implementaron **OutSystems**, una plataforma low-code, para crear múltiples aplicaciones internas.
- **Resultados**:
  - Redujeron el tiempo de desarrollo de meses a semanas.
  - Mejoraron la eficiencia operativa.
  - Crearon aplicaciones escalables sin comprometer la calidad.

💡 **Otras empresas que usan low-code en proyectos grandes**:

- **Schneider Electric** (automatización industrial con Mendix).
- **Airbus** (aplicaciones internas con Appian).
- **KLM Airlines** (gestión de procesos con Betty Blocks).

5. CONCLUSIONES

FINALIDAD, PARA QUÉ SE UTILIZA :

El Low-Code se usa para muchos tipos de proyectos, sobretodo cuando se quiere realizar rápidamente programas, o cuando se quiere realizar un programa que no sea tan grande ni requiera de una escalabilidad (entonces para que complicarse la vida escribiendo código, pudiendo hacerlo más rápido con low-code), otro caso es que si personas que no saben mucho de programación o que se dedican a algo que no tiene que ver con la programación pero en algún momento quieren una página web o una app para su producto, pues pueden realizarlo con low-code sin ser un profesional en programación y sin tener conocimientos de algún lenguaje de código.

Algunos de los proyectos más comunes donde se utiliza low-code son los siguientes:

- Aplicaciones de negocios:

- Gestión de procesos empresariales: Automatización de flujos de trabajo, gestión de inventarios, y control de proyectos. Las plataformas low-code permiten crear aplicaciones de

administración de empresas sin la necesidad de un equipo de desarrollo de software extensivos.

- CRM y ERP personalizados:
  Las plataformas low-code permiten crear soluciones de gestión de relaciones con clientes (CRM) o planificación de recursos empresariales (ERP) adaptados a las necesidades específicas de cada empresa.

- Aplicaciones móviles y web simples:

- Creación de aplicaciones móviles o sitios web con funcionalidades estándar como formularios, paneles de control, y funciones de administración sin necesidad de desarrollarlos desde cero.

- Aplicación de integración:

- Integración de sistemas existentes: muchas plataformas low-code ofrecen conectores para integrar aplicaciones ya existentes sin tener que escribir todo el código manualmente.

- Prototipos y MVPs (producto mínimo viable):

- Se pueden crear prototipos funcionales y MVPs rápidamente para validar ideas de negocio antes de invertir en desarrollos más complejos.

- Automatización de tareas:

- Creación de aplicaciones que ayuden a automatizar tareas repetitivas o de bajo valor, como gestión de aprobaciones, generación de informes, o envíos automáticos de correos electrónicos.

- Soluciones internas para empresas:

- Herramientas para uso interno en las organizaciones, como aplicaciones de seguimiento de empleados, gestión de tareas, gestión de tiempo, etc.

- Aplicaciones con funcionalidades limitadas o específicas

- proyectos donde se necesita una solución sencilla con funcionalidades muy específicas y sin necesidad de interfaces o lógicas demasiado complejas.

USO PRÁCTICO

en el uso práctico si es factible pero antes de ponerte a desarrollar hay que invertir un poco de tiempo a aprender para qué sirve cada nodo y a usar más o menos la interfaz gráfica de la plataforma low-code que vayamos a utilizar. Además, también tienes que aprender el lenguaje que necesites para cuando quieras escribir algunas líneas de código. Por ejemplo, en node-red puedes escribir código en javascript, pues tendrías que aprender javascript para cuando quieras escribir algunas líneas de código (este tiempo de aprendizaje es mucho menor que por ejemplo el tiempo de aprendizaje que habría que dedicar para aprender un lenguaje de código como java o python por ejemplo).

El low-code se utiliza para una amplia gama de proyectos, sobretodo renta mucho cuando se quiere realizar aplicaciones rápido sin tanta limitación porque puedes escribir un poco de código.

PROS VS CONS → Enfocado en NodeRed.

PROS de Node-RED:

- **Es fácil trabajar en equipo**: Node-RED facilita la colaboración entre miembros de un equipo gracias a su interfaz visual, que permite compartir flujos de trabajo y colaborar de manera intuitiva, sin necesidad de un conocimiento profundo en programación.

- **Gran comunidad de usuarios y desarrolladores activos**: La plataforma tiene una comunidad activa que ofrece soporte constante, comparte módulos y ofrece recursos como tutoriales, lo que facilita resolver problemas y mejorar tus proyectos.

- **Es compatible con dispositivos IoT**: Node-RED es especialmente útil para proyectos IoT, ya que permite integrar y controlar dispositivos fácilmente mediante nodos preconfigurados, lo que acelera la implementación de soluciones conectadas.

- **Se puede automatizar tareas rápido**: Node-RED permite automatizar flujos de trabajo y tareas repetitivas con facilidad, lo que ahorra tiempo y reduce el riesgo de errores manuales, ideal para proyectos donde la automatización es clave.

- **Interfaz gráfica intuitiva y sencilla**: La interfaz de Node-RED permite diseñar aplicaciones mediante arrastrar y soltar nodos, lo que hace que sea fácil para cualquier persona entender y crear flujos sin necesidad de ser experto en programación. Esto acelera el desarrollo y hace que la creación de aplicaciones sea accesible incluso para novatos.

CONS de Node-RED:

- **Dificultad para proyectos complejos**: A medida que los proyectos se vuelven más grandes o complejos, la gestión de los flujos de trabajo en Node-RED puede volverse confusa, dificultando el mantenimiento y la comprensión del sistema.

- **Falta de control sobre el código generado**: Aunque Node-RED ofrece mucha flexibilidad, el código generado por la plataforma es abstracto, lo que puede ser una limitación si necesitas un control detallado o personalizaciones avanzadas.

- **Limitación de personalización**: A pesar de su flexibilidad, Node-RED tiene límites cuando se trata de personalizar profundamente los flujos o los nodos sin necesidad de codificar, lo que podría no ser suficiente para proyectos que requieren ajustes a nivel de bajo nivel.

VENTAJAS Y DESVENTAJAS

Ventajas de Lowcode frente a Nocode:

- **Mayor flexibilidad y personalización**: Las plataformas *low-code* como Node-RED permiten personalizar el código y extender las funcionalidades con programación. Esto ofrece más flexibilidad para adaptarse a necesidades específicas en comparación con las

plataformas *no-code*, que suelen tener limitaciones en cuanto a personalización.

- **Escalabilidad**: Las soluciones *low-code* son más escalables, ya que permiten escribir código adicional cuando se necesita expandir una aplicación. Las plataformas *no-code* son generalmente más rígidas y pueden enfrentar problemas al intentar escalar proyectos complejos.

- **Mejor para programadores experimentados**: Los programadores experimentados pueden sacar más provecho de las plataformas *low-code*, ya que pueden añadir personalizaciones y optimizaciones de código según lo necesiten. En plataformas *no-code*, la capacidad de intervención del programador es muy limitada.

Ventajas de Nocode frente a Lowcode:

- **Más fácil de aprender**: Las plataformas *no-code* están diseñadas para ser extremadamente accesibles para usuarios sin experiencia técnica, permitiendo a cualquiera crear aplicaciones sin aprender a programar. Esto facilita el proceso de aprendizaje y la adopción.

- **Se desarrolla más rápido**: Dado que las plataformas *no-code* eliminan la necesidad de escribir cualquier código, el desarrollo de aplicaciones es mucho más rápido, lo que es ideal para proyectos simples o para quienes buscan soluciones inmediatas sin complicaciones.

- **Mejor adaptación para personas que están empezando en la programación**: Las plataformas *no-code* son perfectas para quienes están empezando a aprender sobre desarrollo, ya que no requieren conocimientos de programación, pero permiten entender cómo se estructuran las aplicaciones y los flujos de trabajo de

manera intuitiva.

Ventajas de Lowcode frente a Code:

- **Menor complejidad**: Las plataformas *low-code* abstraen muchos de los detalles técnicos y de infraestructura que normalmente se manejan en programación tradicional. Esto hace que los desarrolladores puedan concentrarse en la lógica de negocio sin complicarse con los aspectos técnicos más complejos.

- **Escalabilidad**: Las plataformas *low-code* permiten una escalabilidad más sencilla que las soluciones *no-code*. Además, permiten integrar y escribir código personalizado cuando la aplicación crece, lo que facilita su expansión sin la complejidad de codificar todo desde cero.

- **Más rápido de desarrollar**: En comparación con la programación tradicional, las plataformas *low-code* reducen considerablemente el tiempo de desarrollo. Los desarrolladores pueden usar componentes preconstruidos y centrarse en la lógica de negocio, lo que acelera la creación de aplicaciones.

Ventajas de Code frente a Lowcode:

- **Control sobre el hardware**: Al programar directamente, tienes control completo sobre el hardware y la infraestructura. Esto es crucial cuando necesitas interactuar de forma específica con el sistema o realizar ajustes a nivel de hardware que no son posibles con plataformas *low-code*.

- **Mayor rendimiento y optimización**: La programación tradicional te permite optimizar tu código para obtener el mejor rendimiento posible, controlando aspectos como la gestión de memoria o el

uso de recursos. Las plataformas *low-code* generan código más genérico que puede no ser tan eficiente.

- **Mejor comprensión de los conceptos fundamentales**: Programar desde cero te obliga a comprender los conceptos fundamentales de la informática, como estructuras de datos, algoritmos y administración de memoria. Esto da una base sólida que no se obtiene con plataformas *low-code*, que abstraen gran parte de esos detalles.

6. BIBLIOGRAFÍA

- Mashable: https://nl.mashable.com/business/9775/the-5-advantages-and-limitations-of-low-code-platforms

- Riveratec: https://riberatec.com/

- Node-red en docker: https://nodered.org/docs/getting-started/docker

- Docker instalador: https://docs.n8n.io/hosting/installation/docker/

- Mendix. (2024). *Guía Low-Code: ¿Qué es el desarrollo Low-Code?*. Disponible en: https://www.mendix.com/es/low-code/que-es-low-code/

- Digital 360 Iberia: https://www.computing.es/a-fondo/low-code-el-camino-mas-corto-al-exito/

- Iberdrola:

https://www.iberdrola.com/innovacion/low-code#:~:text=El%20movimiento%20no%20code%20o,%E2%80%9D%20o%20%E2%80%9Cprogramaci%C3%B3n%20visual%E2%80%9D.

- Parte de la información ha sido organizada con la asistencia de ChatGPT para la mejor comprensión de los conceptos.