Saturn Studio — Referencia Técnica

Arquitectura, Variables, Tipos de Datos, Expresiones y Referencia de Comandos

> **Alcance del Documento:** Este manual técnico abarca detalladamente la arquitectura interna de Saturn Studio, el modelo de ejecución de sus agentes, el sistema integral de variables, los tipos de datos admitidos, los operadores y expresiones de control, así como la referencia de comandos para sus módulos principales.

1. Arquitectura de Saturn Studio

Saturn Studio es un entorno de computación en la nube diseñado para la orquestación integral de agentes. Dentro de su ecosistema, cada componente asume una función específica en el ciclo de vida de las automatizaciones.

1.1 Capas del Ecosistema Rocketbot

El ecosistema se distribuye en cinco capas operativas bien definidas:

Capa	Función y Descripción
Saturn Studio	Núcleo central de orquestación donde convergen la lógica de negocio, los agentes de IA y RPA. Es un entorno 100% cloud accesible mediante el navegador en studio.rocketbot.com.
RPA Studio	Entorno multiplataforma (Windows, Linux, macOS) dedicado al desarrollo de bots tradicionales enfocados en interfaces de escritorio y procesos estructurados.
AI Studio	Capa especializada en la extracción y ordenamiento de datos no estructurados (tales como documentos, audios o correos) para alimentar los flujos de Saturn Studio.
Orchestrator	Centro de gobernanza, programación (scheduling) y supervisión en tiempo real que administra la fuerza laboral digital (instancias, colas, procesos y triggers).
Xperience	Interfaz humano-máquina que provee formularios dinámicos y colas transaccionales para la interacción de los usuarios con los agentes.

1.2 Modelo de Ejecución de un Agente

El motor interno de Saturn Studio procesa la ejecución de los agentes a través de las siguientes fases secuenciales:

1. **Trigger:** El agente es activado mediante un webhook, de manera manual, por planificación cron o invocado por otro agente.
2. **Inicialización:** Se cargan las variables de alcance local y global del agente, y se resuelven las credenciales de manera segura desde el vault.
3. **Ejecución secuencial de Moons:** El motor procesa cada Moon según el orden de sus conexiones, llamando al comando del módulo respectivo y almacenando el resultado en la variable de destino.
4. **Evaluación de condiciones:** Las Moons de tipo If/Else examinan expresiones booleanas bifurcando el flujo según el resultado obtenido.
5. **Human-in-the-loop (si aplica):** El flujo se detiene temporalmente a la espera de una acción humana (como respuestas o aprobaciones de formularios), quedando suspendido hasta recibir el input.
6. **Finalización:** El agente concluye su ciclo registrando en el log si finalizó de manera exitosa o con fallos.

1.3 Tipos de Trigger (Disparadores)

Los flujos pueden iniciarse mediante múltiples mecanismos de activación:

Tipo de Trigger	Comportamiento y Características
Manual	Ejecución directa por el usuario desde el panel de control o el editor, ideal para pruebas.
Webhook	Activación instantánea mediante llamadas HTTP (GET, POST, PUT, DELETE) desde sistemas externos. Permite configurar reintentos ante fallos (0 a 10) y tiempos de espera entre ellos.
Schedule (Cron)	Ejecución planificada mediante expresiones cron (por ejemplo, ejecuciones recurrentes en días hábiles a una hora fija).
Trigger por otro agente	Encadenamiento de flujos independientes donde la finalización de un agente activa el siguiente.
Xperience (Formulario)	El envío de un formulario por parte de un usuario inicia el agente, usando los datos cargados como contexto de entrada.
Múltiples triggers	Disparadores personalizados integrados en los módulos, como el de Gmail, que arranca el proceso al detectar nuevos correos.

2. Variables y Tipos de Datos

Las variables actúan como el canal de transferencia de información entre las distintas Moons. El sistema permite trabajar tanto con variables locales como globales.

2.1 Tipos de Variables Disponibles

Saturn Studio admite los siguientes tipos de datos para sus variables:

Tipo	Descripción y Uso Común	Valor de Ejemplo
String	Texto plano para contenidos de correos, IDs textuales, mensajes o respuestas de IA.	'Hola mundo'
Number	Valores numéricos enteros o decimales para contadores, montos o bucles.	42 o 3.14
Boolean	Valores lógicos de verdadero o falso para banderas de control y condicionales.	true / false
Object / JSON	Estructuras de datos anidadas de clave-valor para respuestas de APIs o formularios.	Acceso: ${{variable}.campo.subcampo}
Array / List	Listas indexadas y ordenadas para colecciones de correos o filas SQL.	Acceso: ${{variable}[0]}
Credential	Referencia segura a credenciales del vault que oculta el valor real.	Uso: {nombre_variable}
File	Referencias a elementos almacenados en el File Storage (PDFs, imágenes, etc.).	N/A
Null / Vacío	Variables sin valor asignado. Requieren validación previa para evitar fallos.	N/A

2.2 Alcance (Scope) de las Variables

La visibilidad y persistencia de las variables varían según su definición:

- **Variable local:** Definida y accesible únicamente dentro de un agente específico. Se configura en el panel de Variables y se reinicia en cada ejecución.
- **Variable global:** Declarada en el menú de Variables Globales de la cuenta, accesible por cualquier agente y persistente entre ejecuciones. Se invoca mediante {{NOMBRE_VARIABLE}}.
- **Variable de credencial:** Variable local de tipo *Credential* que apunta al vault seguro. Se emplea omitiendo las llaves dobles, usando el formato {nombre_variable}.
- **Variable de entorno:** Variable global configurada para segmentar recursos entre diferentes ambientes de trabajo (por ejemplo, Desarrollo vs. Producción).

2.3 Sintaxis de Referencia

Para invocar o inspeccionar variables se aplican las siguientes reglas sintácticas:

Expresión	Descripción
{mi_variable}	Acceso al valor completo de una variable local o global.
{VARIABLE_GLOBAL}	Convención formal recomendada para variables globales (en mayúsculas).
${{respuesta}.campo}	Sintaxis para acceder a una propiedad específica dentro de un objeto JSON.
"${{respuesta}.items[0].nombre}"	Acceso a un índice de una lista JSON y a su atributo interno.
{mi_credencial}	Referencia directa a una variable configurada como Credential.

3. Expresiones y Operadores

Las expresiones condicionales rigen el comportamiento de las Moons de tipo If/Else y los comandos parametrizados con lógica condicional.

3.1 Operadores de Comparación

Permiten contrastar valores dentro de los flujos operacionales:

Operador	Descripción	Ejemplo de Uso
==	Igual a	"{estado}" == 'aprobado'
!=	Distinto de	"{resultado}" != null (si resultado es string)
>	Mayor que	{monto} > 1000
<	Menor que	{contador} < 10
>=	Mayor o igual que	{score} >= 0.8
<=	Menor o igual que	{reintentos} <= 3
contains	Evaluación de inclusión (cadenas y listas)	"{mensaje}".includes('error')
is empty	Comprobación de valor nulo, vacío o indefinido	"{respuesta}" == null (si respuesta es string)
is not empty	Comprobación de existencia de contenido	"{datos}" != null (si datos es string)

3.2 Operadores Lógicos

Utilizados para combinar múltiples condiciones en una sola expresión:

- **AND / &&:** Verdadero únicamente si todas las condiciones se cumplen (Ejemplo: {activo} == true && {monto} > 0).
- **OR / ||:** Verdadero si al menos una de las condiciones es válida (Ejemplo: "{estado}" == 'error' || {reintentos} >= 3).
- **NOT / !:** Invierte el sentido lógico de la condición evaluada (Ejemplo: !({procesado} == true)).

3.3 Moons de Control de Flujo

Módulos nativos encargados de guiar y estructurar el mapa de ejecución:

Moon / Comando	Función Operativa
Start	Nodo de inicio obligatorio en todo flujo. Dirige la ejecución hacia la primera Moon lógica.
If / Else	Evalúa una condición booleana bifurcando el flujo en ramas True o False. Cuenta con un nodo *End If* para unificar el camino posterior.
While / Loop	Cíclico. Itera un bloque de Moons mientras se mantenga la condición de verdad. Finaliza mediante un *End loop*.
For Each	Itera sobre cada uno de los elementos de un Array, finalizando la secuencia con un *End loop*.
System Wait / Sleep	Suspende temporalmente la ejecución por los segundos indicados. Es idóneo para mitigar la saturación de APIs externas.
Try / Catch	Gestión de excepciones. Si ocurre un fallo en el bloque *Try*, la ejecución salta al bloque *Catch*. Concluye en un *End Try*.
Log Message	Inserta textos informativos en el registro del sistema, herramienta clave para auditoría y depuración.
Set Var	Asigna valores a una variable (estáticos, dinámicos o basados en expresiones).
Break	Interrumpe de forma inmediata el bucle en ejecución y salta fuera de su bloque de iteración.

4. Referencia de Comandos por Módulo

Cada módulo provee comandos específicos con parámetros de entrada. Los campos acompañados de un asterisco (*) son requeridos obligatoriamente para su funcionamiento.

4.1 Módulos de Inteligencia Artificial y Conectividad

Módulo: OpenAI

Comando	Parámetros Principales	Descripción / Retorno
Text Completion	credential*, model*, prompt*, max_tokens, temperature, system_prompt	Genera respuestas de texto a partir del modelo seleccionado. Retorna un *String*.
Generate Image	credential*, prompt*, size, quality, n	Devuelve la URL de la imagen creada mediante IA. Retorna un *String*.
Analyze Image	credential*, image_url*, prompt*, model	Genera un análisis o descripción de la imagen provista. Retorna un *String*.
List Models	credential*	Devuelve un listado (Array) con todos los modelos disponibles en la cuenta.

Comando	Parámetros Principales	Descripción / Retorno
Manage Assistants	credential*, action*, assistant_id, name, instructions, model	Ejecuta acciones de creación, actualización o baja de asistentes. Retorna el objeto asistente.
Message Assistant	credential*, assistant_id*, thread_id, message*	Entabla comunicación con un asistente, devolviendo la respuesta y gestionando los hilos.

Módulo: Request HTTP (Genérico)

Comando	Parámetros Principales	Descripción / Retorno
Simple HTTP Request	url*, method (GET/POST/PUT/DELETE), result_variable*	Invoca una URL específica y vuelca la respuesta como texto (*String*) en la variable elegida.
Call API Advanced	url*, method*, headers (JSON), body (JSON), auth, proxy, result_variable*	Realiza peticiones complejas configurando encabezados y cuerpos. Retorna una estructura *JSON*.
Execute CURL	curl_command*, result_variable*	Procesa comandos estructurados bajo el estándar cURL de forma directa.

Módulo: Webhooks

Comando	Parámetros Principales	Descripción / Retorno
Receive Webhook	http_method*, response_mode, result_variable*, retry_on_error, wait_between_retries	Captura llamadas HTTP entrantes guardando el cuerpo, encabezados y queries en la variable.
Send Webhook Response	task_id*, response (JSON/Text), status_code, headers	Envía una respuesta HTTP personalizada directamente al origen del webhook.

4.2 Módulos de Bases de Datos

Módulo: MySQL

- **Query MySQL:** credential*, query*, result_variable* | Ejecuta instrucciones SQL y devuelve un Array de objetos donde cada uno representa una fila de resultados.
- **Insert MySQL:** credential*, table*, data (JSON)*, result_variable | Realiza la inserción de un registro y devuelve el ID autogenerado (last_insert_id).
- **Update MySQL:** credential*, table*, data (JSON)*, where*, result_variable | Modifica registros existentes y retorna la cantidad de filas afectadas.
- **Delete MySQL:** credential*, table*, where*, result_variable | Remueve registros según el criterio establecido y retorna el número de filas borradas.

Módulo: PostgreSQL

- **Query PostgreSQL:** credential*, query*, result_variable* | Corre sentencias SQL estructuradas sobre el puerto predeterminado 5432. Retorna un Array de objetos.
- **Insert PostgreSQL:** credential*, table*, data (JSON)*, result_variable | Añade una fila en la base de datos, retornando el objeto recién guardado junto con su ID corporativo.
- **Update PostgreSQL:** credential*, table*, data (JSON)*, where*, result_variable | Actualiza de forma masiva o selectiva las filas que cumplan las condiciones del where.

Módulo: SQL Server

- **Query SQLServer:** credential*, query*, result_variable* | Ejecuta lenguaje T-SQL a través del puerto por defecto 1433, retornando colecciones de objetos dentro de un Array.

4.3 Módulos de Mensajería y Comunicación

Módulo: Gmail

Comando	Parámetros Principales	Descripción / Retorno
Send Email	credential*, to*, subject*, body*, cc, bcc, attachments	Despacha correos electrónicos adjuntando archivos opcionales. Retorna confirmación.
Read Emails	credential*, folder, max_results, query, result_variable*	Extrae correos en formato Array con detalles clave (from, subject, body, attachments).
Search Emails	credential*, query*, max_results, result_variable*	Localiza mensajes mediante filtros de búsqueda avanzada de Gmail (ej. has:attachment).
Move to Folder	credential*, message_id*, folder*	Reubica o etiqueta un correo específico dentro del árbol de carpetas de la cuenta.
Delete Email	credential*, message_id*	Envía el correo a la papelera o efectúa su borrado definitivo.

Módulo: PDF Reader

- **Extract Text:** file_path* (o file_variable*), result_variable* | Transforma el contenido íntegro del documento PDF en una sola variable tipo *String* (puede presentar limitaciones si el archivo es escaneado).
- **Extract Pages:** file_path*, page_numbers, result_variable* | Segrega y procesa de forma exclusiva las páginas parametrizadas, entregando el texto de cada una.
- **Extract Tables:** file_path*, result_variable* | Aplica algoritmos de

reconocimiento para estructurar tablas internas dentro de un Array de Arrays.

Módulo: Xperience

- **List Forms:** credential*, result_variable* | Lista todos los formularios activos enlazados a la credencial del entorno.
- **Get Queue Records:** credential*, form_id*, queue_id, filters, result_variable* | Descarga los registros transaccionales presentes en las colas de trabajo aplicando filtros de estado.
- **Update Record Status:** credential*, form_id*, record_id*, new_status*, result_variable | Cambia de forma lógica el estado operacional de una transacción (ej: *Procesado*, *Error*).
- **Get Job Detail:** credential*, job_id*, result_variable* | Devuelve información minuciosa sobre una tarea/job dentro de la cola transaccional.
- **Download Attachment:** credential*, form_id*, record_id*, result_variable* | Descarga de forma local los archivos que hayan sido adjuntados en un registro determinado.

4.4 Módulos de Productividad

Módulo: Notion

Comando	Parámetros Principales	Descripción / Retorno
Query Database	credential*, database_id*, filter (JSON), sorts, result_variable*	Consulta repositorios o bases de datos vinculadas formalmente con la integración de Notion.
Create Page	credential*, parent_id*, title*, properties (JSON), result_variable	Da de alta una nueva página web dentro de Notion, devolviendo su ID y el objeto creado.
Update Page	credential*, page_id*, properties (JSON), result_variable	Modifica de forma directa las propiedades o metadatos de una página existente.
Get Page	credential*, page_id*, result_variable*	Extrae el contenido y la estructura completa de propiedades de una página por su identificador.

Módulo: Asana

- **List Tasks:** credential*, project_id, assignee, completed, result_variable* | Descarga un Array con las tareas asociadas a proyectos o personas específicas.
- **Create Task:** credential*, name*, project_id, assignee, due_date, notes, result_variable | Da de alta una tarea en la plataforma devolviendo el objeto e ID de Asana.

- **Update Task:** credential*, task_id*, name, completed, assignee, due_date, result_variable | Modifica atributos clave en tareas previamente guardadas.
- **Complete Task:** credential*, task_id*, result_variable | Cambia el estado de una tarea directamente al valor de concluida o finalizada.
- **List Projects:** credential*, workspace_id, result_variable* | Lista de forma detallada los proyectos creados dentro de un espacio de trabajo.

Módulo: GitHub

- **Subscribe Webhook:** Configurable externamente en GitHub Settings; requiere result_variable* | Monitorea eventos del repositorio (tales como *push*, *pull_request*, *issues*).
- **Get Repository:** credential*, owner*, repo*, result_variable* | Trae datos del repositorio como descripciones, cantidad de estrellas, ramas y su nombre formal.
- **Get Pull Request:** credential*, owner*, repo*, pr_number*, result_variable* | Inspecciona el estado de un PR, incluyendo títulos, autores y listados de ficheros modificados.
- **List Commits:** credential*, owner*, repo*, branch, since, result_variable* | Retorna un listado estructurado de cambios (SHA, mensaje, autor y fecha).

5. Campos Técnicos de Credenciales

El almacenamiento en el vault seguro requiere parámetros específicos estructurados según el mecanismo de autenticación del servicio.

5.1 Parámetros de Configuración por Tipo de Autenticación

API Key Simple

- **Credential name:** Identificador único interno para su posterior selección (Ej: 'OpenAI Produccion'). No debe contener caracteres especiales.
- **API Key:** Código o token privado extraído directamente del proveedor. Se debe registrar omitiendo espacios adicionales.

OAuth 2.0 (Client ID + Client Secret)

Campo	Descripción y Obtención
Credential name	Identificador interno único.
Client ID	ID público asignado a la aplicación en la plataforma del proveedor externo.

Campo	Descripción y Obtención
Client Secret	Clave privada del aplicativo. Se recomienda respaldar de inmediato ya que suele ocultarse tras la primera visualización.
Redirect URI	Dirección fija y obligatoria: https://studio.rocketbot.com (exacta, sin parámetros ni diagonales finales).
Scopes	Permisos y accesos requeridos. Deben parametrizarse previamente en la consola del proveedor.

AWS IAM (Access Key + Secret)

- **Credential name:** Nombre descriptivo interno.
- **AWS Access Key ID:** Código de acceso alfanumérico del usuario IAM. Formato estándar: AKIA + 16 caracteres.
- **AWS Secret Access Key:** Firma o clave secreta del perfil IAM. Solo se despliega al crear la credencial y requiere resguardo inmediato.
- **AWS Region:** Región geográfica de AWS donde operan los servicios deseados (Ej: us-east-1, sa-east-1).

Bases de Datos (Host + Puerto + Credenciales)

Campo	Especificación Técnica
Server address	Dirección de IP pública o dominio del servidor de datos. **Bajo ningún motivo debe configurarse como localhost o 127.0.0.1.**
Server port	Puerto TCP de escucha: MySQL = 3306, PostgreSQL = 5432, SQL Server = 1433, MongoDB = 27017.
User	Cuenta con privilegios estrictamente acotados (evitar el uso de credenciales maestras como sa o root).
Password	Contraseña de la cuenta de base de datos configurada.
Database	Nombre de la base de datos (sensible a mayúsculas/minúsculas según el motor).
Encrypt Connection	(*SQL Server*) True activa el cifrado TLS, mandatorio en entornos productivos.
Trust Server Certificate	(*SQL Server*) True salta la validación permitiendo certificados autofirmados; restringido a desarrollo.

5.2 Límites y Comportamientos Críticos del Sistema

Al diseñar integraciones dentro de la plataforma, se deben prever los siguientes comportamientos:

- **Vigencia de Claves del Orquestador:** Las API Keys del Orquestador de Rocketbot tienen un periodo de validez estricto de 2 años. Al expirar, todo agente que dependa de ellas detendrá su ejecución por error de autenticación.
- **Revocación de Accesos:** Al dar de baja una credencial compartida con terceros, la desvinculación es inmediata y provocará el fallo del agente

en su ciclo subsiguiente.
- **Client Secret en Proveedores:** Portales como GitHub, LinkedIn o Mercado Libre muestran el *Client Secret* una única vez; es imperativo guardarlo antes de abandonar la pantalla.
- **Uso Sintáctico de Credenciales:** Las variables que referencian elementos del vault se llaman obligatoriamente con una única llave {nombre_variable}. Las dobles llaves se reservan exclusivamente para variables de datos ordinarias.
- **Conectividad de Base de Datos:** Debido a que Saturn Studio es un servicio nativo de la nube, las bases de datos externas deben poseer una IP pública accesible y tener el puerto TCP correspondiente desbloqueado en el firewall corporativo.

6. Webhooks: Referencia Técnica

Los webhooks proveen una vía ágil para recibir alertas y datos en tiempo real provenientes de sistemas periféricos.

6.1 Estructura de los Eventos Entrantes

Al recibir una llamada HTTP en el endpoint del webhook, Saturn Studio expone los siguientes datos estructurados dentro de la variable de resultados asignada:

- **body:** Contiene el cuerpo principal del mensaje, comúnmente formateado como un objeto JSON.
- **headers:** Metadatos técnicos de la petición HTTP (IP del cliente, Content-Type, marcas de tiempo y tokens).
- **query:** Parámetros adjuntos en la URL de llamada (Ej: si la ruta incluye ?id=123&estado=aprobado, se procesará como query.id = '123').
- **params:** Variables lógicas extraídas de segmentos de rutas dinámicas.
- **method:** El verbo HTTP empleado en la petición (GET, POST, PUT, DELETE).

6.2 Modos de Respuesta (Response Mode)

Dependiendo de las necesidades de integración, el comportamiento de respuesta HTTP se divide en tres configuraciones:

Modo	Comportamiento Técnico
Immediately	Devuelve un código HTTP 200 de forma instantánea al recibir el paquete, antes de procesar el flujo interno. Es la opción recomendada para integraciones con políticas de timeout muy estrictas (ej. GitHub, WhatsApp, Mercado Pago).
After execution	Mantiene la sesión HTTP abierta y responde únicamente cuando el flujo del agente se ha completado. Requiere que el emisor tolere tiempos de espera elevados (superiores a 30 segundos).

Modo	Comportamiento Técnico
Manual (Send Response)	Delega la respuesta al comando *Send Webhook Response*, permitiendo computar datos de forma interna para enviarlos de regreso en el cuerpo de la respuesta HTTP.

6.3 Panel de Supervisión y Monitoreo de Webhooks

El entorno administrativo ofrece un listado donde se detalla el agente vinculado, el método HTTP, la URL asignada y los controles de gestión del webhook. Adicionalmente, permite auditar visualmente los flujos en cola, completados con éxito o con errores.

A través de la sección **Tasks**, es posible realizar un análisis forense de cada petición que ingresa a la plataforma utilizando los siguientes datos en pantalla:

Columna / Control	Información Desplegada
Header count	Cuantifica la cantidad de encabezados recibidos en la cabecera HTTP.
Body size	Peso total en bytes de la carga útil (payload), útil para identificar paquetes vacíos o masivos.
Query	Muestra los parámetros incluidos en la URL de la consulta.
Data	Visualización de los datos procesados.
Status	Muestra el estado operativo final (*Completado* o *Con error*).
Updated	Muestra el tiempo transcurrido desde el momento exacto de la recepción.
Ojo (Ver)	Abre el inspector de datos crudos (*raw data*), mostrando el desglose de: body, headers, query, params y response. Panel crítico para depuración y debugging.

Consejo de Desarrollo: Para consultar de forma extendida y actualizada los detalles específicos de cada comando, capturas de interfaz y guías de implementación paso a paso, acceda al portal de documentación oficial en docs.rocketbot.com.