**📑 Documento para Iván – Formularios → Vercel → Make**

**1. 🌍 Introducción (para no técnicos)**

**En la web de Snipe hay distintos formularios: Contacto general, Soporte, y en el futuro más (feedback, embajadores, etc.).  
Todos necesitan que la información llegue segura, organizada y con avisos automáticos al equipo.**

**👉 Para esto montamos un sistema en 3 pasos:**

1. **El usuario rellena un formulario web (ej. soporte).**
2. **Nuestra API en Vercel recoge y valida esos datos.**
3. **Make recibe la información, la guarda en Notion y avisa en Slack.**

**2. ⚡ Beneficios del sistema**

* **Seguridad → El webhook de Make nunca se expone en el frontend. Todo pasa primero por Vercel.**
* **Escalabilidad → Un solo endpoint en Vercel para todos los formularios.**
* **Automatización → Make organiza los datos en Notion y lanza avisos en Slack.**
* **Clasificación automática → Los tickets de soporte se clasifican (crítico, neutro, positivo) sin intervención humana.**
* **Flexibilidad → Podemos añadir más formularios sin rehacer nada. Solo se cambia el source.**

**Ejemplo:**

* **web-contact → Contactos en Notion + Slack #contact.**
* **web-support → Tickets en Notion + Slack #soporte.**

**3. 🛠️ Qué ya está hecho**

* **HTML de formularios → apuntan al endpoint en Vercel (https://api.joinsnipe.com/contact).**
* **API en Vercel (contact.js) → valida y reenvía los datos a Make con un campo source.**
* **Payload definido y probado → ya manda datos como name, email, message, category, attachmentUrl, consent, ts, ip, ua.**

**4. 📥 Qué falta (tareas en Make)**

1. **Crear bases de datos en Notion:**
   * **Contactos (para formulario de contacto).**
   * **Tickets Soporte (para formulario de soporte).**
2. **Montar escenario en Make:**
   * **Webhook (recibir datos).**
   * **Validación básica (consent=true y campos obligatorios).**
   * **Set variables (normalizar texto para análisis).**
   * **Router (3 ramas: crítico 🟥, neutro 🟨, positivo 🟩).**
   * **Crear item en Notion.**
   * **Notificación en Slack con enlace al ticket en Notion.**
3. **Probar con ejemplos reales (un caso crítico, uno neutro y uno positivo).**
4. **Añadir extras: auto-reintentos, control de duplicados, vistas en Notion.**

**5. 🔎 Detalle técnico (para seguir paso a paso)**

**Payload que llega a Make**

**{**

**"source": "web-support",**

**"name": "Nombre Apellido",**

**"email": "user@example.com",**

**"phone": "666777888",**

**"category": "Pagos | Envíos | Pujas | Cuenta | Otro",**

**"message": "Texto libre…",**

**"attachmentUrl": "https://… (opcional)",**

**"consent": true,**

**"ts": "2025-09-25T16:40:53.123Z",**

**"ip": "x.x.x.x",**

**"ua": "User-Agent…"**

**}**

**Reglas de clasificación (Router en Make)**

* **🟥 Crítico → “no funciona”, “error”, “pago rechazado”, “bloqueado”, “estafa”, “reembolso”…**
* **🟨 Neutro → “sugerencia”, “idea”, “cómo hago”…**
* **🟩 Positivo → “gracias”, “me encanta”, “genial”, “funciona bien”…  
  👉 Prioridad: rojo > amarillo > verde.**

**Estructura Notion (Tickets Soporte)**

* **Asunto (Title) → [#{Categoría}] {Nombre} – {Fecha}**
* **Nombre, Email, Teléfono**
* **Categoría (Select)**
* **Mensaje (Rich text)**
* **Adjunto (URL)**
* **Severidad (Select: Crítico, Neutro, Positivo)**
* **Estado (Nuevo, En curso, Resuelto)**
* **Fecha (date, de ts)**
* **IP, UA, Consent, Origen**

**Plantillas Slack**

* **🟥 :red\_circle: Nuevo ticket crítico [{{category}}] – Abrir en Notion: {{url}}**
* **🟨 :large\_yellow\_circle: Nuevo ticket [{{category}}] – Abrir en Notion: {{url}}**
* **🟩 :green\_circle: Nuevo feedback positivo [{{category}}] – Abrir en Notion: {{url}}**

**6. 🌐 Qué más podemos hacer con Vercel (gratis)**

* **Formularios adicionales → /api/feedback.js, /api/bug-report.js.**
* **Stripe → /api/checkout.js (crear pagos) + /api/stripe-webhook.js (escuchar eventos).**
* **Páginas dinámicas → /api/drop/[id].js para subastas.**
* **Links cortos → /link/tiktok, /drop/mikel.**
* **Cron jobs → reportes diarios, recordatorios, limpieza de datos.**
* **Integraciones → guardar directo en Notion o Google Sheets.**
* **KV/Blob → contadores o archivos pequeños.**
* **Slack/Discord → avisos automáticos.**
* **Analíticas → /api/track.js para clics, campañas, etc.**

**👉 Todo esto gratis mientras no pasemos:**

* **125k invocaciones/mes.**
* **100 GB de ancho de banda/mes.**
* **1 GB de almacenamiento.**

**7. ✅ Checklist para Iván**

1. **Revisar el documento y entender el flujo (formularios → Vercel → Make).**
2. **Crear en Notion las dos bases de datos.**
3. **Montar escenario en Make con webhook único + router.**
4. **Configurar las 3 ramas (crítico, neutro, positivo).**
5. **Mapear campos a Notion.**
6. **Crear mensajes Slack con emojis y enlace.**
7. **Hacer pruebas con 3 tickets (uno de cada tipo).**

**🚀 Intro para Iván – Flujo Formularios → Vercel → Make**

**Hemos montado un único endpoint general en Vercel (/api/contact) que recibe todos los formularios de la web.  
La lógica es:**

1. **Formulario HTML (Contacto, Soporte, etc.) → siempre apunta al mismo endpoint en Vercel.**
2. **Endpoint en Vercel → recibe, valida y reenvía el payload completo a Make.**
   * **Aquí incluimos un campo source que identifica el origen (web-contact, web-support …).**
3. **Webhook en Make → recibe todos los datos en bruto.**
4. **Router en Make → separa el flujo según source.**
   * **source = web-contact → Notion (tabla Contactos) + Slack (#contact).**
   * **source = web-support → Notion (tabla Tickets) + Slack (#soporte).**

**De esta forma:**

* **No exponemos la URL de Make en el frontend (seguridad).**
* **Centralizamos todo en un único endpoint en Vercel.**
* **En Make solo hay que usar un router con filtros por source para enviar cada formulario a su destino.**

**👉 En resumen: un solo webhook en Make, múltiples ramas según el formulario.**

**✅ Ya hecho**

1. **Formulario HTML → apunta al endpoint en Vercel (/api/contact).**
2. **contact.js en Vercel → recibe datos, valida, y tiene el espacio para reenviar a Make.  
   (Falta solo pegar la URL real del webhook de Make en el archivo JSON).**
3. **Estructura del payload → definida y clara.**

**🔜 Pendiente (Make)**

1. **Crear la base en Notion (PASO 1) con las propiedades exactas.**
2. **Crear el escenario en Make (PASO 2–5):**
   * **Webhook → recibir payload.**
   * **Validación básica (consent, campos no vacíos).**
   * **Set variables (normalizar texto).**
   * **Router con filtros (rojo, amarillo, verde).**
   * **Notion → Create item con el mapeo.**
   * **Slack → Post message con link a Notion.**
3. **Probar con 3 casos de ejemplo: crítico, neutro, positivo.**
4. **Añadir extras: auto-retry, evitar duplicados, vistas en Notion.**

**🚩 Lo que NO cambias más**

* **El HTML del formulario → queda apuntando a Vercel.**
* **El try/catch del fetch en el HTML → ya no lleva email fijo de prueba (eso era solo para debug).**
* **Las validaciones principales → ahora las gestiona contact.js y Make, no el formulario.**

**👉 En otras palabras:  
La URL del HTML no se toca.  
Lo único que falta es crear el Webhook en Make y pegarlo en contact.js.**

**🌐 Cosas que puedes montar en Vercel (gratis) y aplicables a Snipe**

**1. Formularios y Webhooks (lo que ya hicimos)**

* **✅ Lo que tenemos: api/contact.js → recibe datos del form, valida y reenvía a Make.**
* **🚀 Próximos usos:**
  + **/api/feedback.js → recoger opiniones rápidas de usuarios.**
  + **/api/ambassadors.js → gestionar captación de embajadores (nombre, red social, etc.).**
  + **/api/bug-report.js → centralizar reportes de errores desde la app.**

**2. Pasarela de pagos con Stripe**

* **Stripe tiene webhooks que puedes recibir en Vercel:**
  + **/api/stripe-webhook.js → escucha pagos, suscripciones, reembolsos.**
  + **Al confirmarse un pago → actualizas estado del usuario en Notion o lanzas un aviso a Slack.**
* **También puedes crear un /api/checkout.js → genera sesión de pago de Stripe sin exponer la clave en el front.  
  👉 Así todo lo crítico (claves privadas) queda en el backend seguro (Vercel).**

**3. Generación dinámica de páginas**

* **Imagina un link único por drop/subasta:**
  + **/api/drop/[id].js → genera una página JSON/HTML en vivo con la información.**
  + **No dependes de un backend pesado, solo de un JSON en Notion/DB.  
    👉 Sirve para compartir subastas o items fuera de la app.**

**4. Enlaces cortos y redirecciones**

* **Snipe puede usarlo para marketing:**
  + **/link/tiktok → redirige al perfil oficial.**
  + **/drop/mikel → lleva al drop de Mikel Serrano.**
* **No pagas Bitly ni servicios externos.**

**5. Automatizaciones con Cron Jobs**

* **Ejemplos:**
  + **/api/daily-report.js → cada mañana calcula métricas de la app y las envía a Slack.**
  + **/api/clean-temp.js → borra datos antiguos de pruebas.**
  + **/api/reminder.js → envía recordatorios (ej. drops que expiran hoy).**

**6. Integraciones con Notion / Google Sheets**

* **Igual que con Make, pero directo si quieres:**
  + **/api/notion.js → guardar data sin pasar por Make.**
  + **/api/sheets.js → añadir fila en Google Sheets automáticamente.**

**7. Autenticación ligera**

* **Ejemplo: /api/login.js → validar tokens JWT o sesiones.**
* **Útil para paneles internos (equipo) sin montar un backend enorme.**

**8. Almacenamiento en Vercel KV o Blob**

* **Gratis hasta cierto límite.**
* **KV → guardar pares simples (ej. contador de visitas de un drop).**
* **Blob → subir fotos o adjuntos pequeños.  
  👉 Puede servir para prototipar sin pagar base de datos.**

**9. Notificaciones en Slack / Discord**

* **Cualquier evento → lo rediriges a Slack vía webhook.**
* **Ejemplo:**
  + **Nueva subasta creada → Slack #equipo.**
  + **Pago fallido → Slack #alertas.**
  + **Feedback positivo → Slack #snipe-vibes.**

**10. Analíticas simples**

* **Con un endpoint /api/track.js, capturas:**
  + **Qué botón clicó el usuario.**
  + **Desde qué campaña llegó (UTM).**
* **Almacenas en Notion, Sheets o incluso Vercel KV.  
  👉 Tienes métricas sin instalar Google Analytics.**

**💡 Ejemplos específicos para Snipe**

* **Stripe → subastas premium o venta directa de drops.**
* **Feedback rápido → mini formularios de 2 clics para medir comunidad.**
* **Embajadores → landing privada con formulario → API Vercel → Notion.**
* **Slack alerts → “Drop de artista X activo” → notificación automática al equipo.**
* **Links cortos → sni.pe/medusa → acceso directo a campaña en festival.**
* **Daily KPIs → cada mañana Slack recibe: “Nuevos usuarios: 150 | Nuevos drops: 20 | Pujas: 73”.**

**📌 Todo esto entra gratis en Vercel mientras no superes:**

* **125k invocaciones/mes (de funciones serverless).**
* **100 GB de ancho de banda/mes.**
* **1 GB de almacenamiento en blobs/KV.**

**Prompt para instaruar Forulario en make**

Quiero configurar en **Make** un flujo que recibe tickets desde mi API en Vercel y los guarda en **Notion**, con aviso en **Slack**. No quiero emails ni añadir campos nuevos al formulario. La “satisfacción” debe inferirse automáticamente (semáforo) a partir del **mensaje** y la **categoría**.

**Contexto técnico**

* Formulario web → POST https://api.joinsnipe.com/api/contact
* El endpoint valida reCAPTCHA y reenvía a **Make (webhook)**.
* Payload que llega a Make:

{

"source": "web-support",

"name": "Nombre Apellido",

"email": "user@example.com",

"phone": "666777888",

"category": "Pagos | Envíos | Pujas | Cuenta | Otro",

"message": "Texto libre…",

"attachmentUrl": "https://… (opcional)",

"consent": true,

"ts": "2025-09-25T16:40:53.123Z",

"ip": "x.x.x.x",

"ua": "User-Agent…"

}

**Lo que necesito paso a paso**

1. **Reglas de clasificación (sin IA, solo reglas) para semáforo**:
   * 🟥 **Crítico** si message contiene palabras clave negativas (ej.: “no funciona”, “error”, “pago rechazado”, “no me deja”, “bloqueado”, “estafa”, “devolución”, “no llega”, “reembolso”, “fallo”, “no puedo pagar/pujar”) o si category es “Pagos” y hay “error/fallo”.
   * 🟨 **Neutro** si message contiene “sugerencia”, “podríais”, “idea”, “mejoraría”, “cómo hago”.
   * 🟩 **Positivo** si message contiene “gracias”, “me encanta”, “buena”, “genial”, “funciona bien”.
   * Empates: priorizar 🟥 > 🟨 > 🟩.
   * Entregarme estas reglas como **filtros Make** (Text Contains / Regex) o **Text Parser**.
2. **Esquema Notion (database “Tickets Soporte”)**:
   * Título (fórmula/concat): [#{{category}}] {{name}} – {{date}}
   * Nombre (texto), Email (email), Teléfono (texto)
   * Categoría (select: Pagos, Envíos, Pujas, Cuenta, Otro)
   * Mensaje (rich text)
   * Adjunto (url)
   * **Severidad** (select: Crítico, Neutro, Positivo)
   * Estado (select: Nuevo, En curso, Resuelto)
   * Fecha (date, de ts)
   * IP (texto), UA (texto), Consent (checkbox), Origen (texto: “web-support”)
3. **Escenario Make (módulos exactos)**:
   * Webhooks → **Custom Webhook** (trigger)
   * **Module de “Set variables”** o “Text aggregator” → normalizar texto (lowercase, trim)
   * **Router (3 ramas)** según reglas:
     + 🟥 Rama Crítico
     + 🟨 Rama Neutro
     + 🟩 Rama Positivo
   * Notion → **Create database item** (mapear campos + Severidad según la rama)
   * Slack → **Post message** a #notificaciones-formulario con un mensaje **minimalista**:
     + 🟥/🟨/🟩 (emoji), [#Categoría] Nuevo ticket, y **enlace a Notion** (sin duplicar el contenido).
   * (Si lo ves mejor) primero crear en Notion y después usar el **URL** devuelto por Notion en el mensaje de Slack.
4. **Mapeo campo a campo** (payload → Notion):
   * name → Nombre
   * email → Email
   * phone → Teléfono
   * category → Categoría
   * message → Mensaje
   * attachmentUrl → Adjunto
   * ts → Fecha
   * ip → IP
   * ua → User-Agent
   * consent → Consent (checkbox)
   * “Severidad” → valor fijo según rama del Router
   * Título → fórmula/concat que me indiques
5. **Plantilla Slack (minimalista, con semáforo y link)**
   * 🟥 Nuevo ticket crítico [{{category}}] – Abrir en Notion: {{notion\_url}}
   * 🟨 Nuevo ticket [{{category}}] – Abrir en Notion: {{notion\_url}}
   * 🟩 Nuevo feedback positivo [{{category}}] – Abrir en Notion: {{notion\_url}}
6. **Filtros y seguridad**
   * En el Webhook o tras él: procesar **solo si** consent == true y name/email/category/message no vacíos.
   * Activar **Auto-retry** y **Error handler** en Make.
   * Evitar duplicados (idempotencia simple con ts+email si hace falta).
7. **Vistas/KPIs en Notion**
   * Hoy
   * Pendientes 48h (Estado=Nuevo y Fecha ≤ now-48h)
   * Por categoría
   * Por severidad (Crítico/Neutro/Positivo)
   * Por asignado (si añado campo “Responsable” después)
8. **Política de datos**
   * Marcar PII: name, email, phone, ip, ua
   * Retención 12 meses, borrado bajo solicitud
   * Acceso solo equipo de soporte
   * Registrar ts, origen (web-support) y (opcional) versión de payload
9. **Checklist de pruebas**
   * Enviar 3 casos desde el form: uno crítico (“no puedo pagar”), uno neutro (“tengo una sugerencia”), uno positivo (“gracias!”)
   * Ver en Make: ramas correctas
   * Crear ítems en Notion con severidad correcta
   * Ver mensajes en Slack con emoji y link a Notion

**Entregables:**

* Orden de módulos Make y configuración de cada uno (capturas mentales: dónde clic y qué opción elegir).
* Reglas de texto (contains/regex) para la clasificación.
* Mapeo a Notion y fórmula del Título.
* Plantillas Slack.
* Buenas prácticas (rate-limit, errores, duplicados).

Asume que ya tengo Make, Notion y Slack conectados.

DESAROOLLO PROTOCOLO

**PASO 1 — Crear la base en Notion (una sola vez)**

Crea una base de datos llamada **“Tickets Soporte”** con estos **campos**:

* **Asunto** → *Title* (propiedad obligatoria de Notion).  
  Lo rellenaremos desde Make con: [#{Categoría}] {Nombre} – {Fecha}.
* **Nombre** → *Text*
* **Email** → *Email*
* **Teléfono** → *Text* (si prefieres, *Phone number*)
* **Categoría** → *Select*: Pagos, Envíos, Pujas, Cuenta, Otro
* **Mensaje** → *Rich text*
* **Adjunto** → *URL*
* **Severidad** → *Select*: Crítico, Neutro, Positivo
* **Estado** → *Select*: Nuevo, En curso, Resuelto (valor inicial: **Nuevo**)
* **Fecha** → *Date* (mapearemos ts)
* **IP** → *Text*
* **UA** → *Text*
* **Consent** → *Checkbox*
* **Origen** → *Text* (pondremos web-support)

✅ (Opcional) Si quieres **ver también una fórmula** con el mismo “Asunto”, añade un campo *Formula* llamado **Título (auto)** con:

concat("[#", prop("Categoría"), "] ", prop("Nombre"), " – ", formatDate(prop("Fecha"), "YYYY-MM-DD HH:mm"))

Pero el que usaremos como título visible será **Asunto (Title)**, rellenado por Make.

**PASO 2 — Make: crear escenario y módulos iniciales**

**2.1 Webhook (Trigger)**

* Crea un **Escenario** nuevo → **Webhooks** → **Custom webhook** → *Add* (nómbralo tickets\_web).
* Copia la **URL** del webhook (la usará tu API en Vercel para reenviar el payload).
* (Ya que tu API valida reCAPTCHA y reenvía, aquí solo “escuchamos”.)

**2.2 Validación básica (Filtro inmediato)**

Añade un **Filtro** justo después del webhook, llamado **“Validación básica”**:

* consent == true
* length(trim(name)) > 0
* length(trim(email)) > 0
* length(trim(category)) > 0
* length(trim(message)) > 0

*(En Make: usa funciones trim() y length(). Campos mapeados desde el webhook.)*

**2.3 Set variables (normalizar texto)**

Añade módulo **Tools → Set variables** y define:

* **msg\_raw** = {{1.message}}
* **msg\_norm** = lower( replace( replace( replace( replace( replace( trim({{1.message}}); "á"; "a"); "é"; "e"); "í"; "i"); "ó"; "o"); "ú"; "u") )
* **cat\_norm** = lower( trim({{1.category}}) )

Esto solo pone el texto en minúsculas y sin tildes para que las reglas de palabras clave funcionen mejor.

**(A guardar para el siguiente paso) — Reglas de semáforo**

Cuando montemos el **Router** (PASO 3), usaremos **estos filtros** (copiar/pegar):

**🟥 Rama CRÍTICO (poner esta rama primera)**

Filtro (cualquiera de estas dos condiciones):

1. msg\_norm **Matches RegEx**:

(?i)(no funciona|errores?|pago(?:s)? rechazad(?:o|a|os|as)|no me deja|bloquead(?:o|a)|estaf(?:a|ado|ada)|devolucion(?:es)?|no (?:me )?llega|reembolso(?:s)?|fall(?:o|a)|no puedo (?:pagar|pujar))

**O**  
2) cat\_norm = "pagos" **Y** msg\_norm **Matches RegEx**:

(?i)(error|fallo)

**🟨 Rama NEUTRO (segunda)**

Filtro msg\_norm **Matches RegEx**:

(?i)(sugerencia|podr[ií]ais|ideas?|mejorar(?:ia|íais)?|c[oó]mo hago)

**🟩 Rama POSITIVO (tercera)**

Filtro msg\_norm **Matches RegEx**:

(?i)(gracias|me encanta|buena|genial|funciona bien)

**Desempates** ya quedan resueltos por el **orden**: primero rojo, luego amarillo, luego verde. Si no encaja en ninguna, puedes crear una **cuarta rama** por defecto: Neutro.

Si todo correcto hasta aquí, el siguiente bloque será:

* **PASO 3 — Router + Notion + Slack**
  + Router con 3 ramas (Crítico/Neutro/Positivo)
  + **Notion → Create database item**: mapeo campo a campo y **Asunto** [#{category}] {name} – {date}
  + **Slack → Post message** con plantilla minimalista y enlace al **URL** devuelto por Notion
  + *(Opcional)* Idempotencia con **Data store** (email|ts), Auto-retry y manejadores de error.

**🔹 PASO 2 – Webhook + Normalización**

1. En Make, crea un **nuevo escenario**.
2. Añade un **Webhook → Custom Webhook** → copia la URL que Make te da.
3. En tu API de Vercel, haz que después de validar el formulario, haga un **POST a ese webhook** con el payload.
4. En Make, añade un módulo **Tools → Set variable** para poner el message en minúsculas y recortar espacios.
5. Aplica un **filtro**: procesar solo si consent = true y los campos name, email, category, message no están vacíos.

**🔹 PASO 3 – Clasificación (Router)**

1. Inserta un **Router** en Make (3 ramas).
2. Configura filtros de texto:
   * **Crítico 🟥**: message contiene alguna palabra clave negativa (regex o “text contains”).
   * **Neutro 🟨**: message contiene sugerencias o preguntas abiertas.
   * **Positivo 🟩**: message contiene agradecimientos o valoraciones positivas.
   * ⚠️ Si varias coinciden → el orden de prioridad es 🟥 > 🟨 > 🟩.

**🔹 PASO 4 – Guardar en Notion**

1. En cada rama del Router, añade módulo **Notion → Create Database Item**.
2. Mapea cada campo del payload:
   * name → Nombre
   * email → Email
   * phone → Teléfono
   * category → Categoría
   * message → Mensaje
   * attachmentUrl → Adjunto
   * ts → Fecha
   * ip → IP
   * ua → User-Agent
   * consent → Consent (checkbox)
   * Severidad → fijo según la rama (Crítico / Neutro / Positivo)
3. Fórmula para el **Título**:
4. [#{{category}}] {{name}} – {{formatDate(ts;"YYYY-MM-DD")}}

**🔹 PASO 5 – Notificación en Slack**

1. Justo después de crear en Notion, añade módulo **Slack → Post message**.
2. Plantilla:
   * 🟥 :red\_circle: Nuevo ticket crítico [{{category}}] – Abrir en Notion: {{notion\_url}}
   * 🟨 :large\_yellow\_circle: Nuevo ticket [{{category}}] – Abrir en Notion: {{notion\_url}}
   * 🟩 :green\_circle: Nuevo feedback positivo [{{category}}] – Abrir en Notion: {{notion\_url}}
3. Usa el **URL devuelto por Notion** para enlazar directo al ticket.

**🔹 PASO 6 – Extras (opcional pero recomendable)**

* Activar **Auto-retry y Error Handler** en Make.
* Añadir control de duplicados (ejemplo: usar combinación ts + email como ID único).
* En Notion, crear **vistas filtradas**:
  + Tickets de hoy.
  + Pendientes +48h.
  + Por categoría.
  + Por severidad.