Dario, hier ist eine **klare, praxistaugliche Checkliste**, damit dein **ElevenLabs Voice Agent ↔ Replit-Frontend ↔ n8n** sauber zusammenspielt – inkl. **Barge-in (Unterbrechen während der Sprachausgabe)**.

**1) Verbindungs- und Security-Basics (macht oft alles kaputt)**

1. **Domain-Allowlist setzen (Embedding):** In ElevenLabs unter **Agent → Customization → Authentication → Allowlists** deine **Replit-Domain** eintragen (z. B. https://<dein-name>.replit.dev). Ohne Eintrag blockt der Agent die Einbettung; in der Browser-Konsole siehst du dann „host not allowed“. Max. 10 Hostnames möglich.
2. **Widget korrekt einbetten:** Nutze das offizielle **Convai-Widget** (Snippet siehe unten). Prüfe, dass agentId stimmt und die Seite über **HTTPS** läuft (Replit: ja).
3. **n8n-Webhook öffentlich & „Production“ verwenden:** In ElevenLabs-Tools **immer** die **Production-URL** deines n8n-Webhooks verwenden (nicht die Test-URL), und im n8n-Webhook **Response Mode = „Respond via Respond to Webhook“** aktivieren. Genau so wird es in Praxisguides beschrieben.  
   *Deine Exportdatei zeigt responseMode:"responseNode" am Webhook und einen angebundenen* ***Respond to Webhook****-Node – gut. Der Workflow ist im Export aber active:false; live muss er* ***aktiv*** *sein.*

**2) Richtiger Payload zwischen Agent ↔ n8n (häufigster Stolperstein)**

* In deinem **Haupt-Workflow „JARVIS“** liest der Agent Text aus **$json.body.query**. Wenn dein ElevenLabs-Tool stattdessen message, text o. ä. sendet, kommt bei dir **nichts** an. Lösung:  
  **Body-Param „query“** als **string (required)** im ElevenLabs-Webhook-Tool definieren.
* Verwende **Content-Type: application/json** und sende ggf. weitere Felder (z. B. email, time, …). Dieses Pattern (Query-Param + Body-Params) wird in Tutorials genauso erklärt.
* In n8n **Antwort serialisieren** (z. B. mit Set-Node oder direkt im **Respond to Webhook**) – das wird in deinen Transkript-Notizen als „stringify, um Sonderzeichen-Fehler zu vermeiden“ erwähnt.

**Kurztest per Terminal (funktioniert auch ohne ElevenLabs):**

curl -X POST "<DEINE\_N8N\_PRODUCTION\_WEBHOOK\_URL>" \

-H "Content-Type: application/json" \

-d '{"query":"Sag mir meinen nächsten Termin"}'

Wenn du eine sinnvolle JSON-Antwort bekommst, ist n8n korrekt verkabelt.

**3) ElevenLabs: Barge-in & Gesprächsfluss wie „menschlicher Dialog“**

Damit du den Agent **unterbrechen** kannst:

1. **Client-Event „Interruption“ aktivieren:** In **Agent → Customization → Conversation flow** unter **Advanced → Client events** „**interruption**“ auswählen. Erst dann akzeptiert der Agent echte Unterbrechungen („barge-in“).
2. **Turn-Taking/VAD (Sprachaktivität) einstellen:** ElevenLabs nutzt **Voice Activity Detection** und Turn-Taking-Logik für flüssige Redewechsel; das ist die technische Basis fürs Barge-in. (Hintergrund & Best-Practices siehe Turn-Taking/VAD-Beiträge.)
3. **Widget verwenden (Full-Duplex):** Das offizielle **Convai-Widget** unterstützt laufende Mikrofonaudio-Streams parallel zur TTS-Wiedergabe (Multi-Context WebSocket), was Barge-in ermöglicht.

**4) ElevenLabs → n8n Tool korrekt anlegen (Webhook-Tool)**

**In ElevenLabs (Agent → Tools → Webhook):**

* **Method:** POST
* **URL:** **n8n-Production-Webhook** des **Haupt-Workflows** (nicht der Child-Agents).
* **Headers:** Content-Type: application/json (+ optional eigener Shared-Secret-Header, den du im n8n-Webhook prüfst).
* **Query Parameter (optional):** z. B. request für Routing.
* **Body Parameters:** mind. query als **string**. Weitere Felder nach Bedarf. (Genau so ist es in Demos beschrieben, inkl. Hinweis Test↔Production-URL.)

**5) n8n-Seite: was dein Export verrät (und was du prüfen solltest)**

* **Webhook-Node:** path: "n8n", **Response via Respond to Webhook** – passt. Achte nur darauf, dass die URL, die du in ElevenLabs einträgst, exakt zur **Production-URL** dieses Webhooks gehört.
* **Agent-Prompt „Jarvis“:** Textinput wird aus **$json.body.query** gelesen (siehe oben).
* **Child-Agents (Email/Calendar/Contact/Content):** In den Exporten sind alle **active:false** – live müssen die **Sub-Workflows aktiviert** sein, sonst schlagen **Tool-Workflow-Aufrufe** still fehl.

**6) Häufige Fehlerursachen & Fix**

| **Symptom** | **Wahrscheinliche Ursache** | **Fix** |
| --- | --- | --- |
| Widget lädt, aber keine Audio/Verbindung | Replit-Domain nicht in **Allowlist** | Domain in **Authentication → Allowlists** eintragen. |
| n8n bekommt keine Nutzereingabe | ElevenLabs sendet message statt query | Im Tool **Body-Param query** setzen oder in n8n auf $json.body.message umstellen. |
| Agent antwortet, lässt sich aber nicht **unterbrechen** | **Client-Event „interruption“** nicht aktiv | In **Conversation flow → Advanced → Client events** „interruption“ aktivieren. |
| Funktioniert im Test, nicht live | **Test-URL** statt **Production-URL** | In ElevenLabs **Production-URL** verwenden; n8n-Workflow **aktivieren**. |
| 4xx/5xx aus n8n | CORS/Auth/Header fehlen | Content-Type: application/json, optional Shared-Secret-Header; in n8n prüfen. |
| Missmatch bei Tool-Namen/Routing | Tool-Name im Prompt ≠ Tool-Name im Node | Tool-Namen 1:1 abgleichen (deine Transkripte heben das hervor). |

**7) Minimal-Snippet: Offizielles ElevenLabs-Widget + dein Sci-Fi-Look**

**Einbettung (HTML, minimal):**

<!doctype html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>JARVIS</title>

<style>

body{margin:0;background:#05080d;height:100vh;display:grid;place-items:center;font-family:system-ui}

.ring{position:relative;width:320px;height:320px;border-radius:50%;

box-shadow:0 0 40px 10px rgba(0,255,255,.15) inset, 0 0 60px rgba(0,255,255,.15);

background:radial-gradient(circle at 50% 50%, #3fe0ff 0%, #0aa9c7 40%, #062a33 60%, transparent 61%);}

.ring::before{content:"";position:absolute;inset:6%;border-radius:50%;border:2px solid rgba(120,255,255,.35);

animation:spin 14s linear infinite;}

.core{position:absolute;inset:28%;border-radius:50%;

background:radial-gradient(circle,#7ee6ff 0%,#2bc4de 50%,#0f2d35 100%);box-shadow:0 0 30px #36daff;}

.pulse{position:absolute;inset:22%;border-radius:50%;box-shadow:0 0 80px 10px rgba(0,255,255,.15);

animation:pulse 2.4s ease-in-out infinite}

.label{position:absolute;bottom:-90px;left:50%;transform:translateX(-50%);

color:#bff7ff;font-weight:600;letter-spacing:.08em;background:#0a1b22;padding:14px 22px;border-radius:18px;border:1px solid #1d5a66;box-shadow:0 0 20px rgba(0,255,255,.15)}

@keyframes spin{to{transform:rotate(360deg)}}

@keyframes pulse{0%,100%{opacity:.45}50%{opacity:.95}}

</style>

</head>

<body>

<div class="ring">

<div class="pulse"></div>

<div class="core"></div>

<div class="label">🎤 Talk to JARVIS</div>

</div>

<script src="https://cdn.elevenlabs.io/convai-widget/index.js"></script>

<script>

window.ConvaiUI({

agentId: "YOUR\_AGENT\_ID", // <- ElevenLabs Agent ID

apiKey: undefined, // kein Client-Key nötig

// optional: startBehaviour: "press\_to\_talk" | "open\_mic"

});

</script>

</body>

</html>

*Hinweis:* Das **Barge-in** stellst du **im Agent (Client events: interruption)** ein, nicht im Snippet.

**8) „Replicate“ vs. „Replit“**

Falls du wirklich **Replicate.com** meinst (Modell-Inference): Rufe Replicate **serverseitig in n8n** auf (HTTP-Request-Node mit API-Key) und **nicht** direkt aus dem Browser/Widget (CORS/Secrets). Das ElevenLabs-Tool triggert dann nur deinen n8n-Webhook. (Allgemeine Integrationslogik zu Webhooks/Tools: )

**Nächste Schritte (kurz & konkret)**

1. **Allowlist**: Replit-Domain im Agent hinterlegen.
2. **Tool prüfen**: Body-Param **query** (string, required) + JSON Header. (Passt zu $json.body.query in deinem Workflow.)
3. **Production-Webhook & Active**: n8n-Workflow aktivieren und **Production-URL** im Tool setzen.
4. **Barge-in aktivieren**: Client-Event **„interruption“** einschalten.
5. **End-to-End-Test**: curl-Test → dann im Widget prüfen.

Wenn du willst, passe ich dir den n8n-Haupt-Workflow so an, dass er **sowohl query als auch message** akzeptiert (fällt robuster aus) und antworte mit schön formatiertem JSON.

**Quellen / Links**

* ElevenLabs **Agents Overview** – agents-platform/overview
* ElevenLabs **Add a voice agent to your app (Widget)** – agents-platform/how-to/adding-a-voice-agent-to-your-app
* ElevenLabs **How tools work** – tools
* ElevenLabs **Authentication → Allowlists** – customizing-your-agent/authentication
* ElevenLabs **Conversation flow → Client events (Interruption)** – customizing-your-agent/conversation-flow
* ElevenLabs **Multi-Context Websocket** – agents-platform/agents-core-tech/multi-context-websocket
* ElevenLabs Blog **Turn-taking & VAD** – voices/turn-taking … /conversational-ai-2-0 …
* **n8n ↔ ElevenLabs** Integration Guide – integrations/n8n
* **n8n Webhook** (Test vs Production) – n8n-Community Thread
* **Deine Dateien**: JARVIS.json, Contact/Calendar/Email/Content-Agent-Workflows (Mapping & Aktivierung), Transcripts (Webhook-Respond & Stringify) – zitiert im Text.

URLs:  
<https://elevenlabs.io/docs/agents-platform/overview>  
<https://elevenlabs.io/docs/agents-platform/how-to/adding-a-voice-agent-to-your-app>  
<https://elevenlabs.io/docs/agents-platform/tools>  
<https://elevenlabs.io/docs/agents-platform/customizing-your-agent/authentication>  
<https://elevenlabs.io/docs/agents-platform/customizing-your-agent/conversation-flow>  
<https://elevenlabs.io/docs/agents-platform/agents-core-tech/multi-context-websocket>  
<https://elevenlabs.io/blog/turn-taking-in-conversational-ai/>  
<https://elevenlabs.io/blog/conversational-ai-2-0/>  
<https://elevenlabs.io/docs/integrations/n8n>  
<https://community.n8n.io/t/webhook-test-url-vs-production-url/1188>

Kurzantwort: **Fast alles korrekt.** Dein aktueller Workflow ist lauffähig – **ein Punkt blockiert aber die ElevenLabs-→n8n Übergabe**:

* In deinem **Jarvis-Agent** liest du den Text aus **$json.body.message** ein.
* In ElevenLabs hast du im Tool (Webhook) den **Body-Parameter query (String, Required)** konfiguriert (siehe Screenshot). Ergebnis: **Namens-Mismatch → es kommt kein Text im Agent an.**

**Was du ändern sollst (Anfänger-freundlich, 5 Minuten)**

**1) Entweder n8n an ElevenLabs anpassen (einfachste Option)**

* Öffne in n8n den **Jarvis-Agent-Node**.
* Feld **Text** (Prompt) von  
  ={{ $json.body.message }}  
  **ändern auf**  
  ={{ $json.body.query }}  
  (Dann passt es 1:1 zu deinem ElevenLabs-Tool.)

Danach speichern & aktivieren → Test per curl oder Widget.

**2) Oder robust machen (empfohlen)**

So akzeptierst du **query und message** (plus Fallbacks):

**a) Set-Node einfügen (zwischen Webhook und Jarvis)**

* Name: **Normalize Input**
* **Keep Only Set:** an
* **Neues Feld:** userText → **Expression**:
* {{$json.body?.query || $json.body?.message || $json.query || $json.message}}

**b) Jarvis-Agent → Text auf**  
={{ $json.userText }} **ändern.**

Vorteil: Egal wie das Frontend/Tool sendet – dein Agent bekommt Text.

**Weitere kleine Checks (alles andere sieht gut aus)**

* **Webhook Node**: POST + **path** n8n, **Response via Respond to Webhook** – **korrekt angeschlossen.**
* **Workflow aktiv:** active: true ✅
* **Child-Agent-Tools** (Email/Calendar/Contact/Content): Die Verknüpfungen sind da. Achte darauf, dass **diese Workflows in n8n auch aktiviert** sind – sonst „verschluckt“ sich der Agent beim Tool-Aufruf.
* **Model/Creds:** OpenAI-Credentials sind zugewiesen (Name: „OpenAi account 2“). Prüfe ggf. die **Credential-Freigabe** in n8n Cloud (falls du die bekannte Meldung „credentials not shared“ siehst).
* **Tavily-API-Key** steckt im JSON-Body (Dev-Key). Für Produktion: **API-Key als Credential/Secret** hinterlegen, nicht im Klartext im Node.

**Mini-Test (ohne ElevenLabs)**

Wenn du Option 1 nutzt (query):

curl -X POST "https://nexperts.app.n8n.cloud/webhook/n8n" \

-H "Content-Type: application/json" \

-d '{"body":{"query":"Was steht heute im Kalender?"}}'

Wenn du Option 2 nutzt (robust, userText):

curl -X POST "https://nexperts.app.n8n.cloud/webhook/n8n" \

-H "Content-Type: application/json" \

-d '{"body":{"message":"Was steht heute im Kalender?"}}'

Du solltest eine sinnvolle JSON-Antwort vom **Respond to Webhook** erhalten.

**Bonus: Sicherheit & Barge-in**

* **Shared Secret**: In ElevenLabs-Tool unter **Headers** einen Secret-Header setzen (z. B. X-EL-Secret) und in n8n nach dem Webhook mit einem **IF-Node** prüfen. (Verhindert fremde Webhook-Aufrufe.)
* **Barge-in**: In ElevenLabs hast du **Client Event interruption** bereits aktiviert – das ist genau richtig für **Unterbrechen während der Sprachausgabe**. (Im Widget musst du nichts extra tun.)

Wenn du willst, baue ich dir gern die **robuste Variante** als kleinen Patch (Webhook → Set „Normalize Input“ → Jarvis → Respond). Sag kurz **Option 1** (einfach) oder **Option 2** (robust), dann bekommst du den passenden JSON-Diff.

**Quellen/Referenzen (für dich zum Nachlesen):**  
<https://elevenlabs.io/docs/agents-platform/how-to/adding-a-voice-agent-to-your-app>  
<https://elevenlabs.io/docs/agents-platform/customizing-your-agent/conversation-flow>  
<https://elevenlabs.io/docs/agents-platform/tools>  
<https://community.n8n.io/t/webhook-test-url-vs-production-url/11889>

**Interne Belege (dein Workflow):** JARVIS (6).json

**Deine Konfiguration im Überblick: Schritt für Schritt erklärt**

Du bist dabei, einen KI-Sprachassistenten namens **JARVIS** zu erstellen. Stell ihn dir wie einen Angestellten vor, dem du gerade beibringst, wie er sich verhalten, sprechen und welche Aufgaben er erledigen soll.

**1. Die Persönlichkeit und Hauptaufgabe (Tabs: Workflow & Alpha)**

Hier legst du das Gehirn und die Persönlichkeit deines Agenten fest.

* **Sprache:** Dein Agent ist auf **Deutsch** eingestellt.
* **Erste Nachricht:** Wenn ein Gespräch beginnt, sagt dein Agent: *"Systeme betriebsbereit. Wie kann Ihnen heute helfen, Sir?"*. Das ist der erste Eindruck, den ein Nutzer bekommt.
* **System Prompt (Die Persönlichkeit):** Du hast ihm eine klare Persönlichkeit gegeben, die an J.A.R.V.I.S. aus Iron Man erinnert: witzig, trocken, hilfsbereit und clever. Das ist die wichtigste Anweisung, denn sie bestimmt, *wie* der Agent auf Anfragen antwortet.
* **LLM (Das Sprachmodell):** Hier wählst du aus, welches KI-Modell die "Denkarbeit" leistet. In deinem Screenshot ist noch keins ausgewählt. Normalerweise ist hier standardmäßig ein Modell von ElevenLabs oder OpenAI (wie GPT-4) eingestellt.
* **Tools (Die Fähigkeiten):** Das ist der wichtigste Teil! Ein "Tool" gibt deinem Agenten eine Fähigkeit, die über reines Reden hinausgeht. Du hast ein "Custom Tool" namens **"n8n"** eingerichtet.
  + **Was es tut:** Dieses Tool soll die Anfrage des Nutzers an eine externe Anwendung (deinen n8n-Webhook) senden. Stell es dir so vor: Der Agent hört dem Nutzer zu, versteht, was er will (z.B. "Suche nach dem Wetter in Berlin"), und leitet diese Anfrage dann an n8n weiter, damit dort etwas damit passiert.
  + **Der "Body":** Du hast festgelegt, dass die Anfrage des Nutzers als "query" im Body der Nachricht an n8n gesendet wird. Das ist korrekt.

**❗️ Wichtiger Fehler zum Beheben:** In der Konfiguration deines "n8n"-Tools gibt es eine Fehlermeldung: **"Header name cannot be empty"** und **"Secret id cannot be empty"**.

* **Was sind "Headers"?** Das sind zusätzliche Informationen, die mit einer Anfrage gesendet werden, oft zur Authentifizierung. Stell es dir wie einen Ausweis vor, den dein Agent vorzeigen muss, damit n8n weiß, dass die Anfrage erlaubt ist.
* **Was ist ein "Secret"?** Das ist eine sichere Methode, um Passwörter oder API-Schlüssel (geheime Schlüssel) zu speichern, damit sie nicht offen in der Konfiguration stehen.

**Lösung:**

1. Du hast einen "Header" vom Typ "Secret" hinzugefügt, aber die Felder leer gelassen.
2. **Entweder:** Wenn dein n8n-Webhook keine Authentifizierung benötigt, klicke einfach auf den **"Delete"**-Knopf neben diesem leeren Header.
3. **Oder:** Wenn dein n8n-Webhook einen Sicherheitsschlüssel erwartet, musst du hier den Namen des Headers (z.B. Authorization oder X-API-Key) und den dazugehörigen geheimen Schlüssel (das "Secret") eintragen. Woher du diese Informationen bekommst, steht in der Dokumentation deines n8n-Workflows.

**Solange dieser Fehler besteht, kann dein Agent seine Hauptaufgabe (die Anfrage an n8n senden) nicht ausführen.**

**2. Die Stimme (Tab: Voice)**

Hier geht es darum, wie dein Agent klingt.

* **Stimme:** Du hast eine Stimme namens **"JARVIS"** ausgewählt. Das passt perfekt zur Persönlichkeit.
* **Use Flash:** Diese Einstellung ist aktiviert. Das bedeutet, du nutzt ein schnelleres Sprachmodell, das für Konversationen mit geringer Verzögerung optimiert ist. Gute Wahl für einen interaktiven Assistenten.
* **Stabilität & Klarheit:** Die Regler für "Stability" und "Clarity" sind in einer mittleren Position. Das ist ein guter Ausgangspunkt, um eine Stimme zu haben, die sowohl natürlich als auch verständlich ist.

**3. Sicherheit und Zugriff (Tab: Security)**

Hier legst du fest, wer deinen Agenten nutzen darf und wie.

* **Allowlist (Erlaubnis-Liste):** Du hast hier eine URL eingetragen: jarvis-ultimate-assistant-dariomohtachem.replit.app. Das ist eine sehr wichtige Sicherheitseinstellung! Sie bedeutet, dass **nur** von dieser Webseite aus eine Verbindung zu deinem Agenten hergestellt werden kann. Anfragen von anderen Webseiten werden blockiert.
* **Post-Call Webhook:** Du hast das Fenster geöffnet, um einen Webhook nach dem Anruf zu erstellen, ihn aber nicht konfiguriert. Ein solcher Webhook könnte nach jedem Gespräch automatisch eine Zusammenfassung oder Daten an ein anderes System senden.

**4. Gesprächsregeln (Tab: Advanced)**

Hier konfigurierst du die Feinheiten des Gesprächsablaufs.

* **Turn Timeout (7s):** Wenn der Nutzer spricht und dann aufhört, wartet der Agent 7 Sekunden auf eine weitere Eingabe, bevor er selbst wieder spricht oder das Gespräch als beendet ansieht.
* **Silence and end call timeout (-1):** Wenn der Nutzer für eine bestimmte Zeit komplett schweigt, beendet der Agent den Anruf. -1 bedeutet, dass diese Funktion aktuell deaktiviert ist und der Agent unendlich lange auf eine Antwort wartet.
* **Max conversation duration (600s):** Ein Gespräch wird nach 600 Sekunden (10 Minuten) automatisch beendet. Das ist eine Schutzmaßnahme, um endlose Anrufe und damit Kosten zu vermeiden.

**5. Einbettung auf einer Webseite (Tab: Widget)**

Hier konfigurierst du, wie dein Agent als Chat-Fenster auf einer Webseite erscheint.

* **Embed Code:** Dies ist der HTML-Code, den du auf deiner Webseite (wahrscheinlich der oben genannten Replit-Seite) einfügen musst, damit das Chat-Fenster dort erscheint.
* **Realtime transcript:** Du hast die Echtzeit-Abschrift aktiviert. Das bedeutet, der Nutzer kann live mitlesen, was gesprochen wird.
* **Terms and conditions:** Du hast aktiviert, dass Nutzer erst Nutzungsbedingungen zustimmen müssen, bevor sie den Chat starten können. Das ist wichtig für den Datenschutz.

**Zusammenfassung und deine nächsten Schritte**

1. **Dein Ziel:** Du baust einen J.A.R.V.I.S.-Assistenten, der auf einer Webseite lebt und Anfragen an dein n8n-System weiterleiten kann. Die Persönlichkeit und die Stimme hast du bereits super passend konfiguriert.
2. **Wichtigste Aufgabe:** **Behebe den Fehler im "Custom Tool"**. Gehe zum Tab Workflow -> Custom tools -> n8n und korrigiere den leeren "Header". Entweder löschst du ihn, oder du füllst ihn mit den korrekten Daten, die dein n8n-System erwartet. **Das ist der Grund, warum dein Agent wahrscheinlich noch nicht wie gewünscht funktioniert.**
3. **Speichern:** Du hast in vielen Fenstern die Meldung **"You have unsaved changes"**. Vergiss nicht, auf den **"Save"**-Knopf zu klicken, nachdem du die Fehler behoben und deine Einstellungen überprüft hast.
4. **Testen:** Sobald alles gespeichert ist, benutze den **"Test Agent"**-Knopf oben rechts, um auszuprobieren, ob dein Agent jetzt richtig auf Anfragen reagiert und diese an n8n weiterleitet.