DOCUMENT DE DESIGN

Solo Brain

+ Claude Code Mesh

Un assistant personnel multicanal et un maillage de sessions Claude Code distribuées, pilotables depuis l'iPhone, hébergés sur ton serveur GPU.

POUR
Michael Jouannigot — Electrosens

SYNTHÈSE EN 60 SECONDES

Le problème

Solo, multi-contrats, beaucoup de déplacements, beaucoup d'idées techniques, pas le temps de tout capturer. En plus, plusieurs sessions **Claude Code** en cours sur deux machines (serveur + PC perso) qui ne peuvent avancer que devant un clavier.

La proposition

Trois briques sur le même socle : un **cerveau de capture vocale** qui range tout sans effort, un **maillage Claude Code** pilotable depuis l'iPhone, et une **PWA mobile-first**. Le tout sur `assistant.electrosens.fr`, à côté de tes 11 vhosts existants, sans toucher à WordPress.

Pourquoi maintenant

INFRA VIVANT

Whisper, Kokoro, Ollama `qwen3.6:35b`, Onyx, Dify, n8n, LiteLLM — déjà debout. Et **11 sous-domaines** en service avec nginx + Let's Encrypt. On suit le même pattern.

GPU × 2

2 × RTX 5070 Ti, 125 Go RAM, 24 cœurs, 1.1 To dispo, IP publique stable. Le LLM tourne en local pour tout ce qui touche aux clients/NDA.

IPHONE 1 GESTE

Siri Shortcut depuis le lock screen, fonctionne en CarPlay. La capture coûte un appui long. Réponse vocale TTS retour.

i

Ce document

Une proposition d'architecture v1 à valider/corriger. Les options techniques explicitement laissées ouvertes sont listées en **§ 9 — Décisions à prendre**. Le reste est tranché par défaut, pour que tu n'aies pas à arbitrer 12 axes en parallèle.

Le profil utilisateur

Toi. Solo, Electrosens, électronique embarquée + IA. Tes douleurs : tu jongles entre mails, WhatsApp, suivi de contrats, idées R&D, et tu passes une partie de tes journées en voiture. Tu as une boîte à outils technique large mais peu de temps pour la mettre au service de **toi-même**.

Tes outils côté hardware (à part) : Raspberry Pi 3/4/5 (avec NPU 26 TOps), IMX8MP, RK3576, FPGA. Ces cartes sont **pour piloter des objets clients**, pas pour héberger l'assistant. L'assistant pourra les commander via MCP/n8n.

Ce qui tourne déjà sur le serveur

Couche	Service	Rôle
Hardware	24 CPU · 125 Go RAM · 2× RTX 5070 Ti · 1.8 To	host
Réseau	IP `178.170.43.134` · nginx host + Docker	host
STT	`whisper-stt` · faster-whisper-server CUDA	container
TTS	`kokoro-tts` · GPU FastAPI	container
LLM local	Ollama `:11434` · `qwen3.6:35b` (Q4_K_M)	host
LLM gateway	LiteLLM `:14000` · multi-providers	host
RAG	Onyx (api, web, model servers, Vespa, MinIO, Postgres, Redis)	stack
LLM-app	Dify 1.13.3 (api, worker, web, Weaviate, sandbox)	stack
Workflow	`n8n-automation`	container
Reranker	`infinity-reranker`	container
Search	`searxng` · meta-search privé	container
UI LLM	Open-WebUI `:4000`	host
DB	Postgres host `:5432` · MariaDB `:3306`	host

Conséquence directe

L'assistant n'a **aucune** nouvelle dépendance lourde à installer. On code uniquement un **orchestrateur** (FastAPI) dans `/root/assistant/api/` et une **PWA Next.js** dans `/root/assistant/web/` qui appellent ces services via leurs API HTTP locales.

L'écosystème de sous-domaines

Tu as déjà **11 sous-domaines** `*.electrosens.fr` servis par nginx host avec certificats Let's Encrypt automatiques. C'est exactement le pattern qu'on suit pour `assistant.electrosens.fr`.

Sous-domaine	Backend	Statut
`chat.electrosens.fr`	Open-WebUI :4000	ACTIF
`dify.electrosens.fr`	Dify (docker-nginx)	ACTIF
`n8n.electrosens.fr`	n8n :5678 + webhooks WhatsApp	ACTIF
`qgaudio.electrosens.fr`	QGAudio API :8011	ACTIF
`llmrouter.electrosens.fr`	LiteLLM :14000	ACTIF
`mcp.electrosens.fr`	Server MCP	ACTIF
`mkdeals.electrosens.fr`	App mkdeals	ACTIF
`search.electrosens.fr`	SearXNG	ACTIF
`gendocs.electrosens.fr`	Génération de docs	ACTIF
`music.electrosens.fr`	Front audio	ACTIF
`wordpress.electrosens.fr`	PHP 8.3-FPM + MariaDB	TRAFIC +
`assistant.electrosens.fr`	**à créer (cible v1)**	À VENIR

WordPress — sanctuaire intouchable

Le contexte

Site servi par nginx host + PHP 8.3-FPM + MariaDB. Fichiers dans `/var/www/wordpress/`. Vhost déjà **durci** (deny `xmlrpc.php`, deny `wp-config.php`, rate-limit `wp-login.php`). **Trafic en croissance**.

L'engagement

On **ne touche pas** à la vhost WP, ni à `/var/www/wordpress`, ni à la base MariaDB. L'assistant utilise un **schéma Postgres séparé** sur l'instance Postgres host. Les builds front lourds tournent en heures creuses.

Opportunités phase ultérieure (non engagées)

RAG sur le contenu publié — ingérer les articles WordPress dans Onyx via l'API REST `/wp-json/wp/v2/posts` pour que l'assistant connaisse ce que tu as publié. **Publication assistée** — depuis une note bien rédigée, créer un brouillon WP via POST sur `/wp-json/wp/v2/posts` (status `draft`). Jamais de publication automatique sans validation.

QGAudio — déjà connecté à WhatsApp

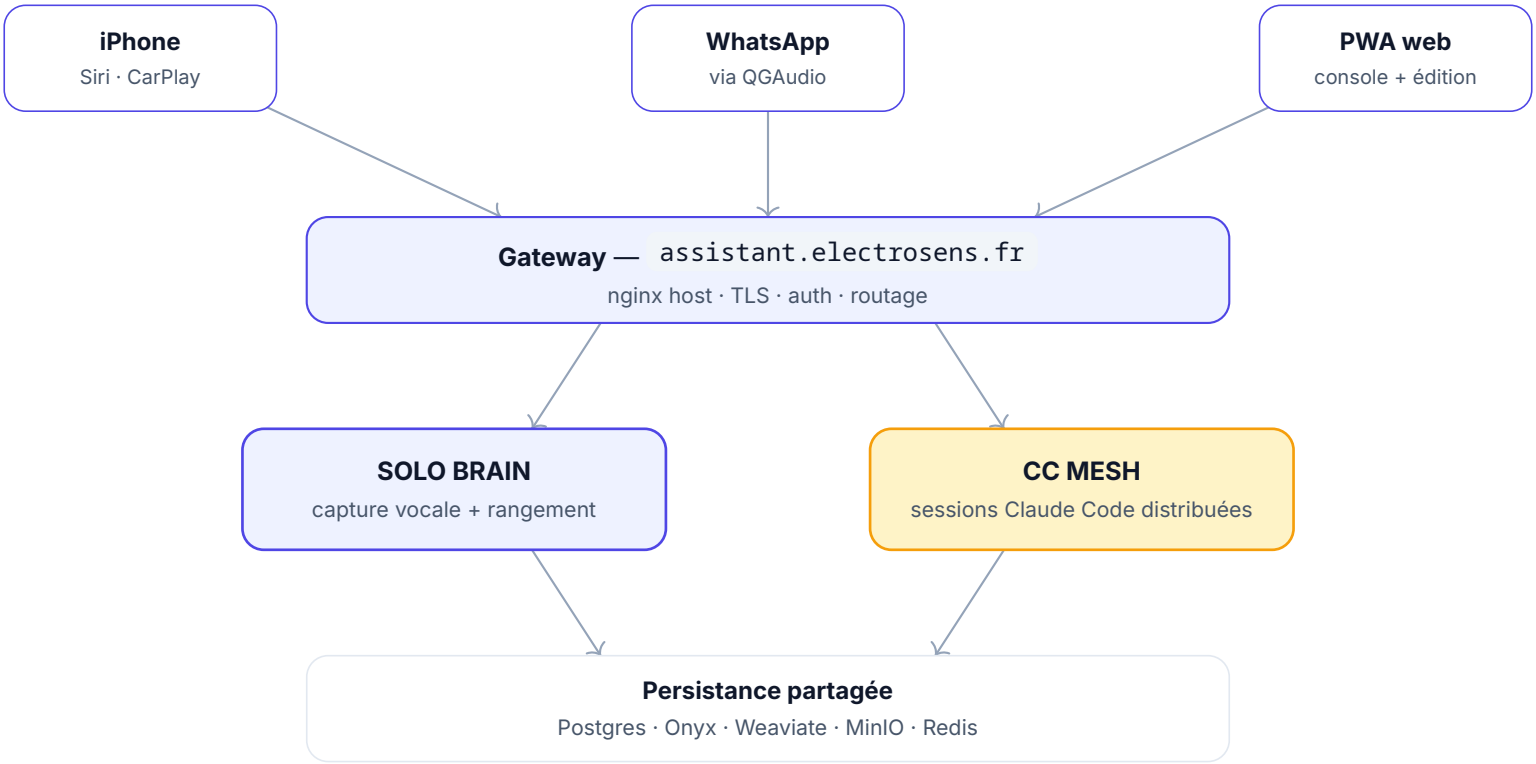
L'API Python `qgaudio-api.service` (port 8011, racine `/root/QGAudio/`) gère déjà :

- `n8n.electrosens.fr/webhook/whatsapp` — challenge GET + forward POST vers n8n
- `n8n.electrosens.fr/webhook/suno-callback`

Cadeau d'infra

Tu as déjà fait **la moitié du chemin WhatsApp Business**. La phase 2 du Solo Brain n'a qu'à se brancher sur le pipeline existant : QGAudio reçoit, dispatch vers `assistant.electrosens.fr/api/capture` au lieu (ou en plus) de n8n. Économie nette : **5 à 10 jours de plomberie WhatsApp Cloud API**.

VUE GLOBALE



Lecture du schéma

Une seule porte d'entrée publique (Gateway) qui authentifie + route vers les deux domaines fonctionnels. **Solo Brain** gère la capture et l'organisation. **CC Mesh** gère le pilotage des sessions Claude Code à distance. Les deux partagent la persistance pour qu'une note vocale puisse devenir un prompt Claude Code, et qu'une réponse Claude Code puisse alimenter le RAG.

Le scénario cible (en voiture)

1

Tu actives un Siri Shortcut depuis le lock screen

Une commande Siri (« Hey Siri, note assistant ») ou un appui long sur le bouton latéral. Aucune app à ouvrir, aucun écran à débloquer. Fonctionne avec CarPlay.

2

L'iPhone enregistre l'audio et le POSTe sur ton endpoint

POST chiffré vers `https://assistant.electrosens.fr/api/capture` avec un token statique. Aucun cloud Apple n'a vu le contenu en clair (la transcription se fait chez toi sur Whisper GPU).

3

L'orchestrateur traite la capture en pipeline

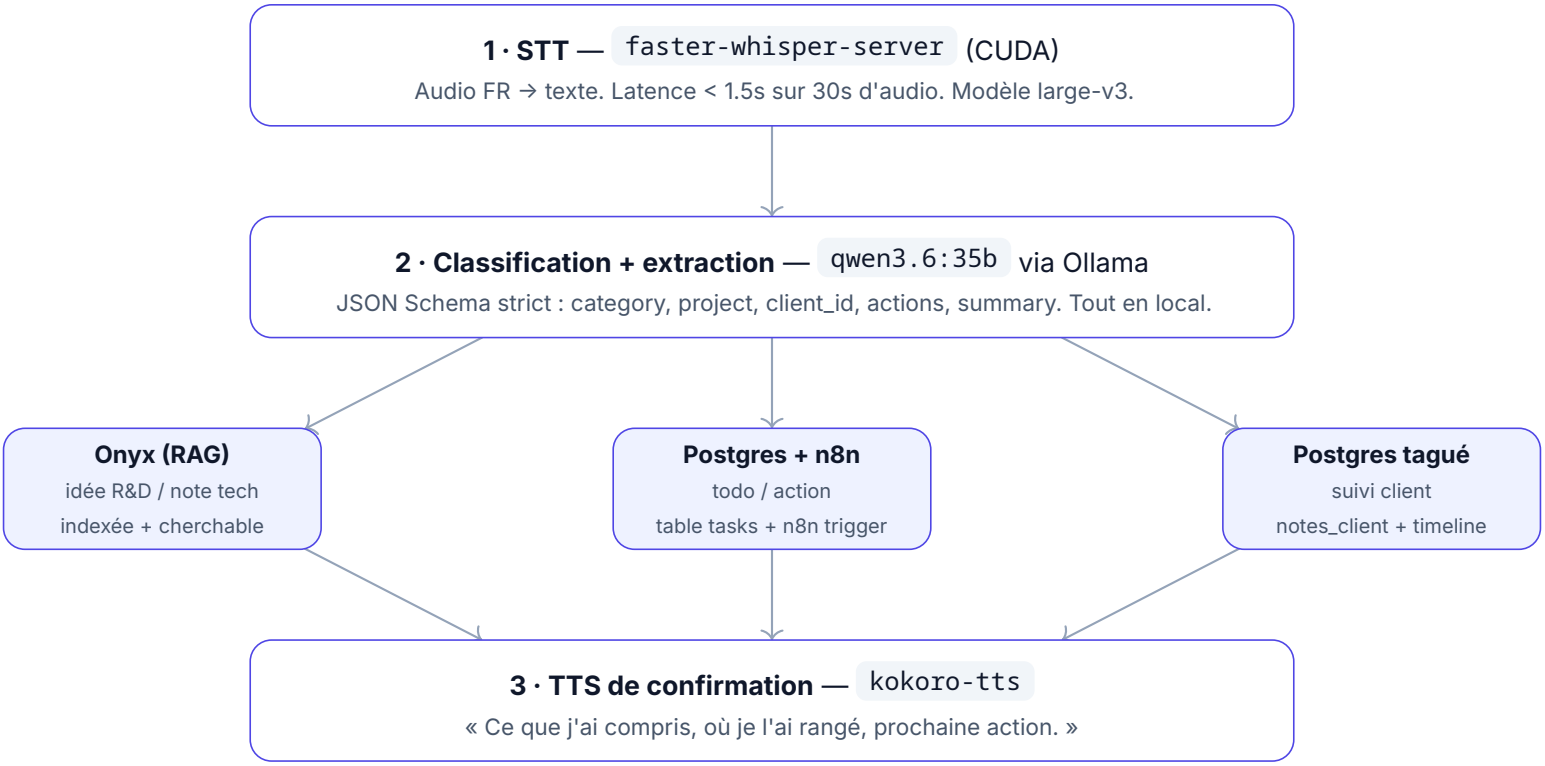
Whisper transcrit, Qwen 35B classifie, le routeur dispatch vers le bon stockage. Tout dure quelques secondes.

4

Réponse vocale TTS Kokoro de retour

« Capté, idée R&D rangée dans le projet IMX8MP, action notée pour vendredi. » Tu sais que c'est bien parti, tu ne perds pas l'info.

Le pipeline de traitement



Pourquoi ce flow

Coût cognitif zéro côté toi : tu balances en désordre, le LLM range. **Sécurité** : tout ce qui contient du contenu client passe par le LLM local, jamais par DeepSeek ou Ollama Cloud. **Réversible** : si tu n'aimes pas le rangement, la note reste accessible dans une « inbox » que tu peux re-classer manuellement depuis la PWA.

Schéma d'API

```
POST https://assistant.electrosens.fr/api/capture
Authorization: Bearer <token>
Content-Type: multipart/form-data

audio: <wav|mp3|m4a>
hint:  "client meeting"     # optionnel, biaise la classification
mode:  "voice" | "text"

→ 200 OK
{
  "capture_id": "cap_01HXY...",
  "transcript": "...",
```

```
  "category":   "idee_rd",
  "project":    "imx8mp-vision",
  "client_id":  null,
  "actions": [
    { "text": "benchmarker NPU 26 TOps", "due": "2026-05-17" }
  ],
  "stored_in": ["onyx://...", "pg:tasks/42"],
  "tts_url":   "https://assistant.electrosens.fr/audio/cap_01HXY.mp3"
}
```

Bonus QGAudio

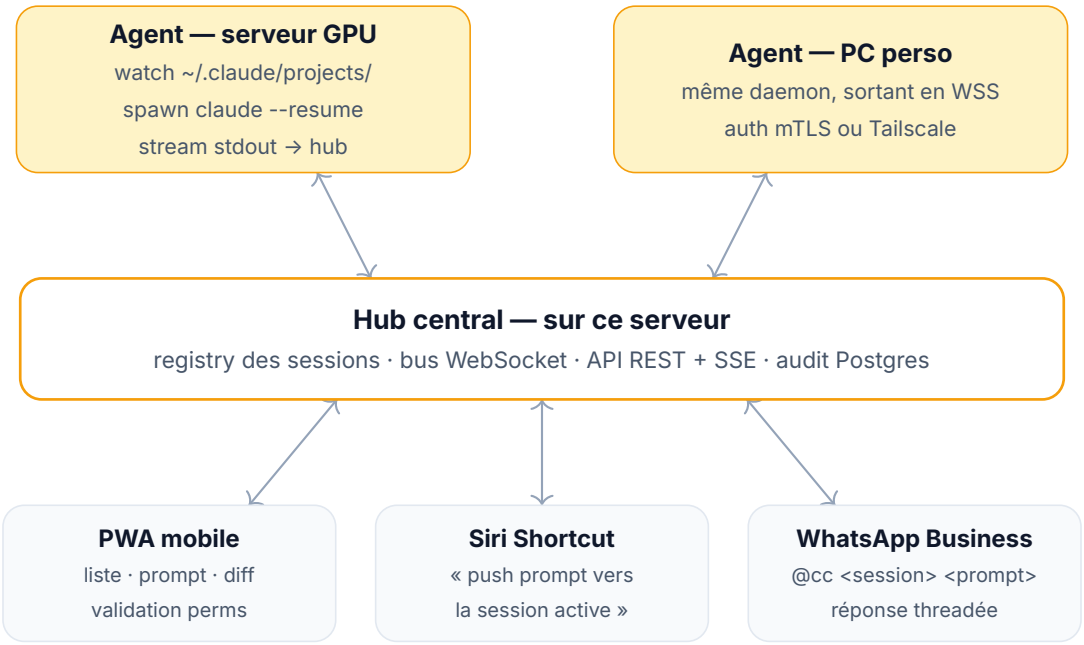
Pour la phase 2, le webhook WhatsApp Business est **déjà reçu** par QGAudio sur `n8n.electrosens.fr/webhook/whatsapp`. On le détourne en parallèle (ou à la place) vers `/api/capture`. Un seul backend, plusieurs canaux d'entrée.

Le problème concret

Tes sessions Claude Code vivent dans `~/.claude/projects/<encoded-path>/*.jsonl` sur **chaque machine**. Aujourd'hui, pour faire avancer une session du PC perso, tu dois être devant le PC perso. Pour faire avancer une session du serveur, tu dois être devant un terminal SSH. **Si tu sors** (RDV client, voiture, soir), tout s'arrête.

L'idée : un **hub central** qui voit toutes les sessions des deux machines, et un canal mobile pour envoyer un prompt à n'importe laquelle d'entre elles.

L'architecture



Le quotidien avec CC Mesh

Cas 1 — En voiture

Ta session `assistant-orchestrator` tourne sur le serveur, en attente. Tu dictes par Siri : « pour la session orchestrator, ajoute un endpoint `/health` ». Le hub reçoit, route au bon agent, l'agent injecte le prompt dans la session via le SDK Claude Agent. La réponse t'est lue à voix haute (Kokoro) si tu as activé le mode vocal.

Cas 2 — Chez le client

Tu vois la liste de tes sessions actives dans la PWA. Tu reprends celle qui patche le firmware IMX8MP, tu valides un tool-use en attente (parce que Claude Code a demandé une permission), tu envoies un prompt complémentaire. Tout passe par le hub, ton PC perso reste éteint si non sollicité.

Subtilités à régler

Permissions Claude Code à distance — valider un tool-use depuis l'iPhone implique de relayer le prompt système de permission via le hub. À cadrer en phase 2. **PC perso derrière NAT** — l'agent du PC perso initie la connexion sortante (WSS), donc pas besoin d'ouvrir un port. **Concurrence** — si tu pousses un prompt depuis l'iPhone alors que tu es aussi dans le terminal de la session, gérer le verrou (warning UI, file d'attente).

Schéma d'API (extraits)

```
GET  /sessions
→ [{ id, agent, project, last_msg_at, status, pending_permissions: [...] }]

POST /sessions/:id/prompt
{ "text": "...", "voice_reply": true }
→ { prompt_id, eta_seconds }

GET  /sessions/:id/stream    (SSE)
→ event: message
  data: { role, content, tool_uses, ... }

POST /sessions/:id/permissions/:pid/approve
```

Le but

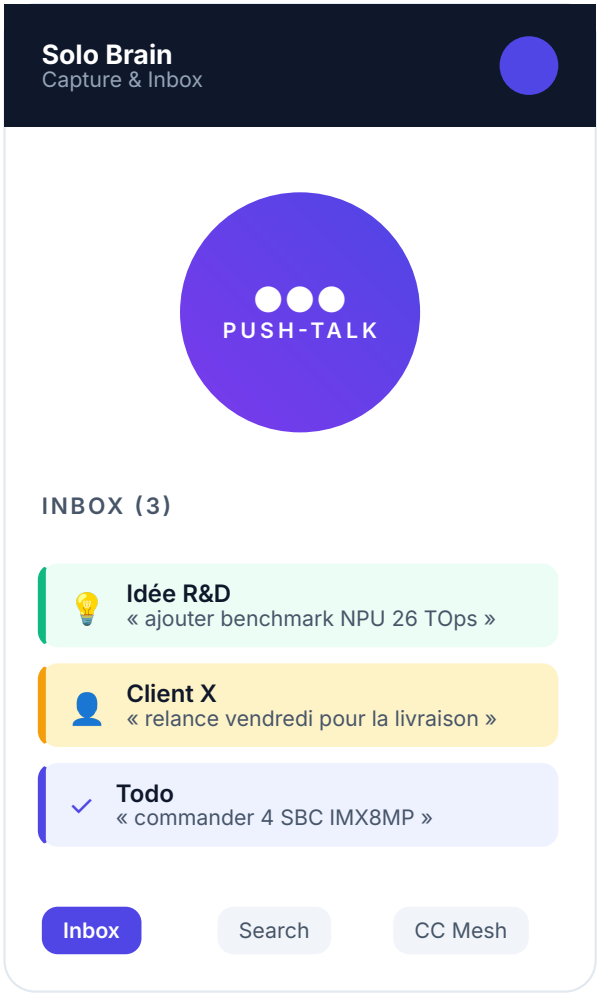
Une **PWA** (Progressive Web App) installable depuis Safari sur l'iPhone **et** utilisable depuis n'importe quel navigateur. Une seule URL, `assistant.electrosens.fr`, qui sert :

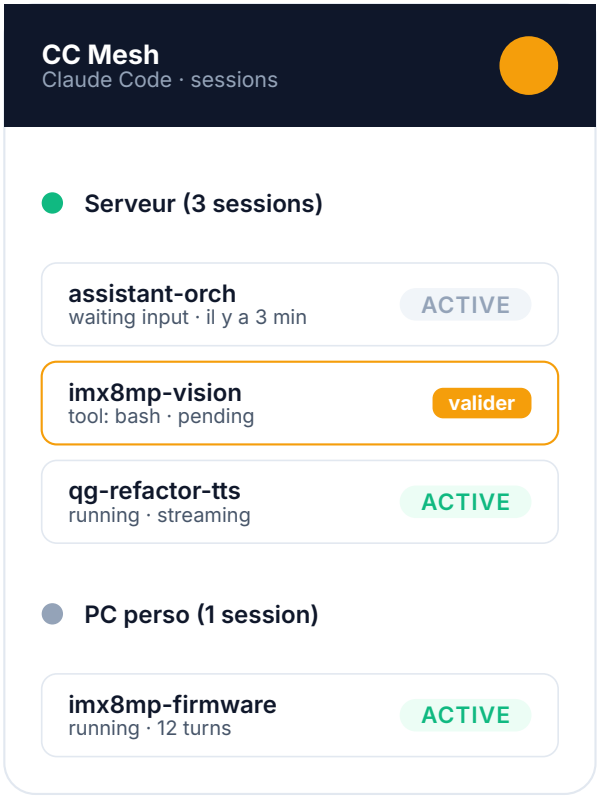
- le **dashboard** de capture et de tri (Solo Brain)
- la **console** de pilotage des sessions Claude Code (CC Mesh)
- la **configuration** (clients, projets, intégrations, tokens)

Le stack — « dernières technos » sans gadget

Couche	Choix	Pourquoi
Framework	**Next.js 15** App Router	RSC, Server Actions, Turbopack
UI runtime	**React 19** + **TypeScript 5.7+**	Actions, useOptimistic, types stricts
Style	**Tailwind CSS v4** (CSS-first)	OKLCH par défaut, config dans le CSS
Composants	**shadcn/ui** + **Lucide** + **Sonner**	Beauté par défaut, code possédé
PWA	**@serwist/next** (Service Worker)	Offline shell, install iOS, Web Push
Tooling	**Bun** + **Biome**	Vitesse réelle vs npm + ESLint + Prettier
Data	**Drizzle ORM** + **Postgres**	Schéma `assistant`, typé, migrations propres
Auth	**Better Auth** (passkeys)	Multi-device, support iOS natif
Fetch / cache	**TanStack Query v5**	Pour les composants client SSE/WS
API backend	**FastAPI** + **Pydantic v2** + **uv**	Endpoint `/api/capture` direct (latence audio min)

Maquettes des écrans clés





Comportement PWA sur iPhone

Install

Safari → Partager → "Sur l'écran d'accueil". Icône custom, splash screen, pas de barre Safari.

Offline shell

App-shell cachée par Service Worker, l'UI ouvre instantanément même hors ligne. Les requêtes mutent une queue qui se rejoue à la reconnexion.

Notifications push

Web Push iOS 16.4+ pour : permission Claude Code en attente, deadline qui approche, brief client.

Trois entrées convergentes

La PWA, le Siri Shortcut et le webhook WhatsApp tapent **le même backend FastAPI** (`/api/capture` , `/api/sessions/...`). Pas de logique dupliquée. La PWA est juste **un client de plus** — la plus visuelle, mais pas la plus rapide pour la capture vocale en voiture (Siri reste plus immédiat).

STACK TECHNIQUE CONSOLIDÉ

Brique	Choix	Pourquoi
Orchestrateur	**FastAPI** + **uvicorn** + **Pydantic v2**	ASGI rapide, types stricts
STT	`faster-whisper-server` (déjà déployé)	GPU CUDA, FR natif, latence < 1.5s
TTS	Kokoro FastAPI GPU	Voix naturelle 24 kHz, FR/EN
LLM principal	Ollama `qwen3.6:35b` en local	Souverain, classification structurée
LLM appoint	DeepSeek + Ollama Cloud via LiteLLM	Tâches non sensibles, recherche enrichie
RAG	Onyx (Vespa + Postgres + MinIO)	Recherche hybride, déjà debout
Workflows	n8n	Calendrier, mails, relances, IoT
Front PWA	Next.js 15 + Tailwind v4 + shadcn/ui	PWA installable iOS, RSC
Bus temps réel	WebSocket + SSE	Latence basse, simple à débugger
DB métier	Postgres schéma `assistant`	Notes, tasks, sessions CC, audit
Sécurité	nginx + Let's Encrypt + Better Auth	TLS auto, passkeys, JWT
Interop CC	`claude-agent-sdk` (Python) + watcher	Officiel, supporte resume, headless, MCP

Souverain par défaut

Tout ce qui touche à du contenu client **reste sur ce serveur**. Les modèles locaux (Whisper, Qwen 35B, Kokoro) couvrent 100% du flow Solo Brain. Les API externes (DeepSeek, Ollama Cloud) sont **opt-in par tag** et utilisées uniquement pour des tâches publiques ou de la recherche web.

ROADMAP PAR PHASES

PHASE 1

Squelette Solo Brain + PWA — 1-2 SEMAINES

- Scaffold `/root/assistant/api/` (FastAPI + Pydantic v2 + uv) et `/root/assistant/web/` (Next.js 15 + Tailwind v4 + shadcn/ui + Bun)
- Endpoint `/api/capture` (audio → Whisper → Qwen 35B → Postgres `inbox`)
- Auth Better Auth + token statique pour Siri Shortcut
- Siri Shortcut iOS qui POSTe vers `/api/capture` + reçoit TTS Kokoro
- PWA minimale : push-to-talk, liste inbox, re-classement, install iOS
- Vhost `assistant.electrosens.fr` + cert Let's Encrypt (déjà fait pour le PDF)

PHASE 2

Routage intelligent + WhatsApp via QGAudio + WP — 2-3 SEM. APRÈS V1

- Catégorisation fine (client / idee_rd / todo / note_tech / admin / perso)
- Push vers Onyx pour les notes longues + recherche fédérée depuis la PWA
- n8n triggers pour todos (Google Calendar, rappels, mails sortants)
- Branche QGAudio `/webhook/whatsapp` vers `/api/capture`
- Ingestion `wordpress.electrosens.fr` dans Onyx (lecture seule)
- Briefing client avant RDV

PHASE 3

CC Mesh — agents distribués — 3-4 SEMAINES

- Daemon `ccmesh-agent` (Rust ou Go) qui watch `~/.claude/projects/`
- Hub central WebSocket sur ce serveur
- PWA : vue liste sessions, prompt rapide, stream SSE des réponses
- Approbation des permissions tool-use depuis l'iPhone
- Connexion sortante WSS du PC perso (NAT-friendly)

PHASE 4

Voix complète + cartes IoT — OUVERTURE

- Mode conversationnel continu en voiture (push-to-talk hands-free permanent)
- Pilotage des cartes IMX8MP / RK3576 via MCP / n8n quand l'intent matche
- Connecteurs Onyx (mails IMAP, Gmail, drive client)
- Briefing matinal vocal automatique au démarrage de la voiture

DÉCISIONS À PRENDRE

Tranché

Hébergement — ce serveur. **Exposition** — `assistant.electrosens.fr` via nginx host + Let's Encrypt (déjà actif pour le PDF). **Privacy LLM** — tout le contenu client passe par Qwen 35B local. **Stack frontend** — Next.js 15 + Tailwind v4 + shadcn + Bun.

1. Authentification iPhone ↔ serveur

Recommandation v1 — token Bearer statique long (256 bits) stocké dans le Siri Shortcut + le keychain. **Évolution v2** — passkeys via Better Auth pour la PWA, et JWT à courte durée pour les Siri Shortcuts régénérés depuis la PWA.

2. Stockage des audio bruts

Recommandation — conservation 30 jours dans MinIO (déjà déployé pour Onyx) puis purge. Permet de re-transcrire si Whisper rate ou si le prompt de classification change. **Alternative** — purge immédiate après transcription.

3. PC perso — quel canal vers le hub ?

Recommandation — daemon `ccmesh-agent` qui ouvre un WebSocket sortant authentifié mTLS. Fonctionne même en NAT, pas besoin d'IP publique côté PC. **Alternative** — Tailscale + REST direct (plus simple à débugger, dépendance Tailscale).

4. Granularité du suivi client

Question ouverte — fiche client structurée (raison sociale, projets, contrats, deadlines) ou simplement un **tag client** sur les notes/tâches ? La fiche est plus utile pour le briefing avant RDV mais demande un effort initial de saisie.

5. Profondeur de l'intégration WordPress

Question ouverte — (a) lecture seule : Onyx ingère le contenu publié, ou (b) bidirectionnel : l'assistant prépare aussi des brouillons via POST `/wp-json/wp/v2/posts` (status `draft`) ? Pas obligé de choisir maintenant.

6. Bun vs npm pour le front

Recommandation — Bun (plus rapide, « dernières technos » demandées). **Alternative** — rester sur npm (Node 20 déjà installé). Impact marginal en prod, sensible en dev.

Comment on procède ensuite

Tu lis ce document à ton rythme. Si la **forme générale** (Solo Brain + CC Mesh sur le même socle) te convient, on passe au bootstrap de la **phase 1** dans `/root/assistant`. Si tu veux ajuster, on liste les ajustements et on itère sur ce document avant d'écrire la moindre ligne de code.

Document généré le 2026-05-10 · Typst 0.14 · `architecture.typ`