

Universitat de Girona
Escola Politècnica Superior

Grau en Enginyeria Informàtica

PROJECTE FINAL DE GRAU

**Automatització intel·ligent de processos
mitjançant n8n i serveis contenitzats**

Autor/a:
Alex García-Conde López

Tutor/a:
Antoni Bardera Reig

MEMÒRIA

Convocatòria:
Juny 2026

Departament :
INFORMÀTICA, MATEMÀTICA APLICADA I ESTADÍSTICA

Projecte: Projecte Final de Grau
Document: Memòria
Títol: Automatització intel·ligent de processos mitjançant n8n i serveis contenitzats
Autor: Alex García-Conde López
Data: Juny 2026

Estudi:
Grau en Enginyeria Informàtica
Universitat de Girona

Supervisor:
Antoni Bardera Reig
Universitat de Girona
Email: anton.bardera@udg.edu
Web: [Link](#)

Agraïments

En primer lloc, vull expressar el meu més sincer agraïment a la meva família, pel suport constant que m'han brindat al llarg de tota la meva trajectòria acadèmica. Des de l'inici de la carrera fins a l'actualitat, han estat un pilar fonamental, oferint-me consells, ànims i confiança en cada etapa del camí. Gràcies per ser sempre al meu costat i per recordar-me, fins i tot en els moments més complicats, que no s'ha de deixar de lluitar.

A continuació, agraeixo al meu tutor, Antoni Bardera, per la seva orientació, disponibilitat i assessorament durant el desenvolupament d'aquest Treball de Fi de Grau. Les seves aportacions, suggeriments i esperit crític han contribuït de manera decisiva a millorar la qualitat d'aquest projecte i a enriquir el meu aprenentatge.

També vull reconèixer el suport dels meus amics, que m'han ajudat a mantenir l'equilibri emocional en els moments més exigents. La seva companyia i capacitat per ajudar-me a desconnectar han estat essencials per afrontar les dificultats amb una actitud més positiva.

Finalment, agraeixo a totes aquelles persones que, d'una manera o altra, han contribuït a la realització d'aquest treball, ja sigui amb paraules d'ànim, consells o suport puntual. A tots ells, moltes gràcies per formar part d'aquest camí.

Índex

Agraïments	iii
Índex de figures	viii
Índex de taules	x
1 Introducció, motivacions, propòsit i objectius del projecte	1
1.1 Introducció	1
1.2 Motivacions	2
1.3 Propòsit	2
1.4 Objectius	2
1.5 Estructura del document	3
2 Estudi de viabilitat	4
2.1 Viabilitat tecnològica	4
2.2 Recursos necessaris	4
2.2.1 Recursos tècnics	4
2.2.2 Recursos humans	5
2.3 Viabilitat econòmica	5
2.3.1 Estimació de costos operatius	5
2.3.2 Model de negoci	5
2.4 Viabilitat legal i ètica	6
2.4.1 Protecció de dades (RGPD)	6
2.4.2 Ètica de la IA	6
2.5 Riscos i ciberseguretat	6
2.6 Conclusions de viabilitat	6
3 Metodologia	7
3.1 Ús d'eines d'intel·ligència artificial	7
4 Planificació	9
5 Marc de treball i conceptes previs	13
5.1 Automatització de processos	13
5.2 Intel·ligència artificial	13
5.3 Integració de serveis	13
5.4 Autenticació i gestió de credencials	14
5.5 Tecnologies de contenidors	14
5.6 Webhooks i comunicació basada en esdeveniments	14
5.7 Funcionalitats d'automatització de Chatwoot	15
5.7.1 Atributs personalitzats	15
5.7.2 Etiquetes	15

5.7.3	Regles d'automatització	15
5.8	Sistemes RAG (Retrieval-Augmented Generation)	16
5.9	Exposició de serveis mitjançant Ngrok	16
6	Requisits del sistema	17
6.1	Requisits funcionals	17
6.2	Requisits no funcionals	18
6.3	Requisits de <i>hardware</i>	18
7	Estudis i decisions	19
7.1	Maquinari	19
7.2	Programari i eines principals	19
7.3	Llibreries i tecnologies utilitzades	20
7.4	Justificació de les decisions	20
8	Anàlisi i disseny del sistema	23
8.1	Anàlisi del sistema	23
8.1.1	Tipus d'usuari	23
8.1.2	Anàlisi de requisits	23
8.1.3	Casos d'ús	26
8.1.4	Model conceptual de dades	27
8.1.5	Disseny de la base de dades	27
8.1.6	Disseny de les interfícies d'usuaris	28
8.1.7	Patrons de disseny aplicats	28
9	Implementació i proves	29
9.1	Cas d'ús 1: Atenció al client automatitzada i omnicanal	29
9.1.1	Arquitectura del sistema	29
9.1.2	Eines de Chatwoot	30
9.1.3	Gestió de cites	30
9.1.4	Etiqueta respon 1 vegada	31
9.1.5	Funcionament del sistema fora d'hores	31
9.1.6	Etiqueta respon X vegades	31
	Limitació detectada i adaptació del sistema	32
	Adaptació als diferents formats d'esdeveniment del webhook	32
9.1.7	Filtratge de missatges i prevenció de bucles	33
9.1.8	Integració amb Gmail	34
9.1.9	Integració amb Flowise	34
	Lògica del flux a Flowise	34
9.2	Cas d'ús 2: Assistència cognitiva	35
9.2.1	Limitacions en la configuració del webhook de Telegram	35
9.2.2	Airtable com a base de dades	36
	Preparació de l'entorn	36
9.2.3	Registre inicial de l'usuari	36
9.2.4	Captura de la ubicació en temps real	37
9.2.5	Enregistrament de l'historial de missatges	37
9.2.6	Interpretació de text lliure	38
9.2.7	Gestió de comandes del bot	38
9.2.8	Vídeo amb el codi desenvolupat	40
10	Implantació i resultats	41
10.1	Procés de desplegament	41

10.1.1	Infraestructura i creació de la màquina virtual	41
10.1.2	Configuració de xarxa i dominis	41
10.1.3	Preparació del repositori i seguretat de dades	41
10.1.4	Preparació del servidor i clonatge del repositori	42
10.1.5	Configuració de <i>docker-compose</i> i variables d'entorn	42
10.1.6	Gestió de xarxa: De Nginx a Traefik	42
10.1.7	Regles d'accés i seguretat de xarxa	42
10.1.8	Preparació de certificats i desplegament final	43
10.1.9	Restauració de la configuració del sistema	43
10.1.10	Interfície de benvinguda (<i>Landing page</i>)	43
10.2	Resultats	43
10.2.1	Cas d'ús 1 - Atenció al client automatitzada	44
10.2.2	Cas d'ús 2 - Assistència cognitiva	45
10.3	Objectius i requisits assolits	46
10.3.1	Avaluació dels objectius	46
10.3.2	Compliment dels requisits	46
11	Conclusions	47
12	Treball futur	49
A	Ètica de la IA i responsabilitat en el tractament de dades	50
B	Integració de serveis externs amb n8n	52
B.1	OpenAI	52
B.2	Compte de Google	53
B.3	Chatwoot	53
B.4	Telegram	53
B.5	Airtable	54
C	Fitxes de casos d'ús	55
D	Estructura de les dades en AirTable	57
D.0.1	Taula 1: Historial ubicacions	57
D.0.2	Taula 2: Historial de converses	57
D.0.3	Taula 3: Historial missatges	58
D.0.4	Taula 4: Medicació	58
D.0.5	Taula 5: Agenda	59
D.0.6	Taula 6: Exercicis	60
D.0.7	Taula 7: Usuari	60
E	Preparació de l'entorn del cas d'ús 1	62
F	Coneixements bàsics d'n8n	63
F.1	Recepció de missatges mitjançant webhook	63
F.2	Respondre un missatge	64
F.3	Gestió dels diferents tipus de missatge	64
G	Integració de WhatsApp mitjançant Evolution API	66
H	Configuració de l'atribut, etiquetes i regles d'automatització	68
H.1	Configuració dels atributs personalitzats	68
H.2	Configuració de les etiquetes	69

H.3	Configuració de les regles d'automatització	70
I	Configuració de cites mitjançant Google Calendar	74
J	Configuració de Flowise	77
J.1	Configuració de la sessió i gestió d'estats	77
J.2	Classificació i lògica de resposta	77
K	Configuració del sistema RAG	80
K.1	Preparació de la base de coneixement	80
K.2	Fonaments de la divisió en fragments (<i>Chunking</i>)	80
K.3	Procés d'Embedding	81
K.4	Funcionament del sistema RAG	82
K.5	Validació i Proves de Recuperació	82
L	Descripció del flux del primer cas d'ús	84
M	Creació del bot de Telegram	89
N	Descripció del flux del segon cas d'ús	90
O	Descripció del flux del segon cas d'ús: Neteja de dades	98
O.1	Implementació del flux	98
P	Descripció del flux del segon cas d'ús: Recordatoris automàtics	101
P.1	Implementació del flux	101
Q	Preparació de l'entorn del servidor	105
R	Clonatge del repositori	106
S	Configuració dels fitxers de producció	107
S.1	Variables d'entorn (.env)	107
S.1.1	Ajustos a Chatwoot	107
S.1.2	Ajustos a Evolution API	107
S.1.3	Configuració de Flowise	108
S.2	Arquitectura del Docker Compose	108
S.2.1	Professionalització d'n8n	108
S.2.2	Implementació tècnica de Traefik al Docker Compose	108
T	Resultats demostració del cas d'ús 1: Atenció al client	110
T.1	Canal WhatsApp	110
T.2	Canal Gmail	112
U	Resultats demostració del cas d'ús 2: Assistència cognitiva	113
U.1	Comanda <i>/les_meves_ubicacions:</i>	113
U.2	Comanda <i>/historial_ubicacions:</i>	114
U.3	Comanda <i>/resum_matinal:</i>	115
U.4	Text en llenguatge natural: <i>/medicacio:</i>	116
U.5	Comandes <i>/agenda_avui</i> i <i>/resum_ahir:</i>	117
U.6	Comanda <i>/exercici_cognitiu:</i>	117
	Bibliografia	118

Índex de figures

5.1	Configuració de credencials a n8n	14
8.1	Diagrama de casos d'ús d'atenció al client	26
8.2	Diagrama de casos d'ús d'assistència cognitiva	26
8.3	Diagrama Entitat-Relació del model de dades	27
9.1	Flux per gestionar cites	30
9.2	Flux actualitzat per gestionar respostes puntuals	31
9.3	Node <i>Code</i> actualitzat	32
9.4	Control d'etiquetes per respostes múltiples	33
9.5	Reinici de l'estat de l'assistent per respostes múltiples	33
9.6	Integració del flux amb Flowise	35
9.7	Error en la url de Telegram	36
9.8	Execució del Telegram Trigger amb /start.	37
9.9	Processament de text lliure	38
9.10	Afegir i consultar ubicacions d'interès	39
10.1	Ús de recursos de la MV	46
B.1	Connexió de credencials d'OpenAI a n8n	52
D.1	Taula 1: Historial ubicacions	57
D.2	Taula 2: Historial de converses	58
D.3	Taula 3: Historial missatges	58
D.4	Taula 4: Medicació	59
D.5	Taula 5: Agenda	60
D.6	Taula 6: Exercicis	60
D.7	Taula 7: Usuari	61
F.1	Esquema de comunicació entre serveis	63
F.2	Flux bàsic per generar respostes automàtiques	64
F.3	Node <i>set</i> per gestionar missatges	65
F.4	Gestió de contingut multimèdia	65
G.1	Connexió serveis Meta o Evolution API	67
H.1	Configuració atribut <i>Assistent Virtual</i>	68
H.2	Atributs personalitzats	69
H.3	Definició de les etiquetes	69
H.4	Regla automatització <i>Sense assignar</i>	70
H.5	Definició de les etiquetes	71
H.6	Definició de les etiquetes	72
H.7	Definició de les etiquetes	73

I.1	Horari cites Google Calendar	75
I.2	Vista de Google Calendar cites	76
J.1	Node <i>Retriever</i> de Flowise	78
J.2	Flux final de flowise	79
K.1	Configuració de l'embedding	82
K.2	Prova del Retrieval	83
L.1	Flux resultant del primer cas d'ús1	87
L.2	Flux resultant del primer cas d'ús2	87
M.1	Configuració del bot de Telegram	89
O.1	Flux final de la neteja de taules	100
P.1	Flux recordatoris automàtics: <i>Schedule Trigger</i>	102
P.2	Flux de recordatoris automàtics	104
T.1	Resposta agendar cita	110
T.2	Resposta respondre 1 missatge	110
T.3	Resposta respondre X missatges1	111
T.4	Resposta respondre X missatges2	111
T.5	Resposta respondre gmail	112
U.1	Resposta llistat ubicacions	113
U.2	Resposta afegir ubicació	114
U.3	Resposta historial ubicacions	114
U.4	Resposta resum matinal	115
U.5	Resposta medicacio	116
U.6	Resposta agenda	117
U.7	Resposta exercici cognitiu	117

Índex de taules

2.1	Estimació de costos operatius mensuals del sistema	5
4.1	Cronograma de planificació del projecte.	12
6.1	Característiques de <i>hardware</i> utilitzat pel projecte.	18

Capítol 1

Introducció, motivacions, propòsit i objectius del projecte

1.1 Introducció

En l'actualitat, la digitalització ha transformat profundament la manera com es duen a terme processos tant en l'àmbit empresarial com en el personal. Cada vegada més activitats es realitzen de forma digital, des de compres en línia fins a la gestió de serveis o la comunicació entre usuaris. Aquest context ha provocat un augment notable de processos repetitius i de tasques que requereixen una gestió constant.

Davant aquesta realitat, l'automatització de processos s'ha convertit en una eina clau per millorar l'eficiència i reduir la intervenció manual. Automatitzar tasques permet disminuir errors humans, agilitzar operacions i optimitzar recursos, fet que es tradueix en una reducció de costos i una millora en la productivitat. Per aquest motiu, cada vegada més organitzacions incorporen sistemes automatitzats en el seu funcionament habitual.

En els darrers anys, la irrupció de la intel·ligència artificial ha suposat un pas més enllà en aquest àmbit. A diferència de les automatitzacions tradicionals, basades en regles fixes, els sistemes amb IA són capaços d'interpretar informació, adaptar-se a diferents situacions i prendre decisions de manera autònoma. Això amplia considerablement les possibilitats d'aplicació, permetent automatitzar processos més complexos i propers al comportament humà.

En aquest context, aquest projecte se centra en el disseny i implementació d'un sistema d'automatització intel·ligent orientat tant a l'atenció al client com a l'assistència personal. Concretament, es desenvolupa una solució basada en fluxos automatitzats i integració amb serveis externs que permet gestionar comunicacions, interpretar missatges en llenguatge natural i oferir funcionalitats d'assistència cognitiva.

Aquest treball s'emmarca en l'exploració del potencial d'aquestes tecnologies en escenaris reals, analitzant com la combinació d'automatització i intel·ligència artificial pot aportar solucions eficients, escalables i adaptables tant en entorns professionals com en la vida quotidiana.

1.2 Motivacions

L'interès per desenvolupar aquest projecte neix, en primer lloc, de l'experiència personal en la realització de tasques repetitives del dia a dia, tant en l'àmbit acadèmic com en situacions quotidianes. Accions com respondre missatges similars, gestionar informació o fer seguiment de recordatoris són processos habituals que, tot i ser necessaris, no aporten un valor afegit significatiu i requereixen temps i atenció que es podrien destinar a activitats més rellevants. Aquesta situació no només es dona a nivell personal, sinó també en entorns professionals, on sovint es destinen recursos a tasques rutinàries que limiten l'eficiència global.

D'altra banda, el projecte sorgeix també de l'interès per l'evolució recent de les eines tecnològiques que faciliten l'automatització d'aquest tipus de processos. L'aparició de solucions cada vegada més accessibles ha posat de manifest el seu potencial per simplificar tasques quotidianes i millorar la gestió del temps. En aquest sentit, aquest treball es planteja com una oportunitat per aprofundir en un camp amb una clara projecció de futur, amb l'objectiu de transformar aquesta inquietud en una solució pràctica que contribueixi a optimitzar processos habituals i aporti valor en escenaris reals.

1.3 Propòsit

El propòsit d'aquest Treball de Fi de Grau és demostrar la viabilitat de substituir la intervenció manual en tasques bàsiques mitjançant agents intel·ligents. No es tracta només d'executar ordres, sinó d'analitzar com aquestes tecnologies poden simplificar l'estructura de treball, optimitzar els fluxos d'informació i oferir una resposta més àgil i personalitzada en diferents contextos.

A través del desenvolupament d'una plataforma funcional, es pretén validar que la combinació d'eines d'automatització sense codi o de baix codi amb models de llenguatge avançats permet resoldre problemes reals que fins fa poc requerien sistemes extremadament costosos o supervisió humana constant.

1.4 Objectius

Aquest projecte té com a objectiu principal dissenyar, implementar i avaluar un sistema d'automatització de processos basat en intel·ligència artificial mitjançant el desenvolupament de dos casos d'ús diferenciats, que permeten analitzar-ne l'efectivitat en contextos diversos:

- **Atenció al client omnicanal:** Per optimitzar la comunicació en entorns empresarials.
- **Assistència cognitiva:** Per donar suport a persones amb problemes de memòria en les seves activitats quotidianes.

A partir d'aquests dos escenaris, es defineixen els següents objectius:

- Definir una arquitectura modular i escalable que permeti la integració i comunicació entre diferents serveis i fonts d'informació.

- Desenvolupar fluxos d'automatització capaços d'executar tasques de manera autònoma, eficient i coordinada.
- Implementar agents d'intel·ligència artificial capaços d'interpretar entrades en llenguatge natural i generar respostes contextualitzades.
- Avaluar el comportament del sistema en els dos casos d'ús plantejats, analitzant-ne la utilitat i les limitacions.
- Desplegar el sistema en un entorn de producció i verificar-ne el funcionament estable.

1.5 Estructura del document

Aquesta memòria s'ha estructurat en diversos capítols amb l'objectiu de presentar de manera clara i ordenada el desenvolupament del projecte.

En aquest primer capítol, s'ha introduït el context actual de la digitalització, la problemàtica associada a les tasques repetitives i la necessitat d'automatitzar processos, així com les motivacions personals i els objectius que defineixen el rumb del projecte.

Els següents tres capítols (capítols 2, 3 i 4) serveixen per situar el lector en el marc de gestió del treball. S'hi analitza si la proposta és realitzable, quina estratègia s'ha seguit per dur a terme el desenvolupament del treball i com s'ha organitzat temporalment l'execució del projecte.

A continuació, s'exposa el bloc de fonamentació i disseny (5, 6 i 7). En aquestes seccions es defineixen els conceptes tècnics clau i les eines necessàries per entendre el sistema, s'especifiquen els requisits que ha de complir la solució i es justifiquen les decisions preses pel que fa a l'elecció de programari i tecnologies.

El nucli tècnic del TFG es troba en els capítols 8 i 9, on es detalla l'anàlisi, el disseny conceptual de les dades i, sobretot, la implementació pràctica. Aquí es descriu pas a pas la construcció dels dos casos d'ús.

Finalment, els últims capítols aborden la posada en marxa i els resultats obtinguts en entorns reals. El document clou amb les conclusions, on es valora el grau d'assoliment dels objectius, es reflexiona sobre les lliçons apreses i s'identifiquen possibles millores i ampliacions del sistema.

Capítol 2

Estudi de viabilitat

L'objectiu d'aquest capítol és analitzar la viabilitat del projecte de manera global, considerant-ne els aspectes tecnològics, econòmics, legals i ètics. Aquest estudi permet determinar si el sistema proposat és realitzable en condicions reals, avaluant els recursos necessaris per a la seva implementació i la seva possible explotació comercial futura.

2.1 Viabilitat tecnològica

El projecte es fonamenta en la integració de tecnologies consolidades en entorns professionals: n8n com a eina d'automatització de processos, el sistema de persistència Airtable i models de llenguatge basats en intel·ligència artificial (IA).

L'ús d'n8n com a motor d'orquestració permet implementar automatitzacions flexibles i escalables mitjançant una arquitectura de microserveis contenitzats amb Docker. La seva compatibilitat nativa amb webhooks facilita la recepció de dades en temps real i la integració amb serveis externs de missatgeria. Per altra banda, la utilització d'Airtable permet una gestió eficient de la informació estructurada gràcies a una API robusta i una interfície accessible.

Pel que fa a la IA, el consum de models a través d'APIs simplifica la integració de capacitats cognitives avançades. En conjunt, la viabilitat tecnològica és alta, ja que el sistema es basa en tecnologies consolidades i compatibles entre si. No obstant això, cal considerar una dependència crítica de la disponibilitat operativa dels proveïdors de serveis externs, fet que podria condicionar la continuïtat del servei en cas d'interrupcions alienes al sistema.

2.2 Recursos necessaris

Per al correcte desenvolupament i manteniment del sistema, s'han identificat els següents recursos:

2.2.1 Recursos tècnics

- **Infraestructura de computació:** Servidor per executar les instàncies de Docker (n8n, base de dades, etc.) en un proveïdor cloud com Azure o AWS.
- **Connectivitat:** Accés a APIs d'IA i sistemes de persistència externs.

- **Seguretat:** Certificats SSL/TLS per a comunicacions segures i sistemes de control d'accessos.

2.2.2 Recursos humans

- **Desenvolupador/a de programari:** Amb coneixements en integració d'APIs, automatització de processos, i coneixements en IA.
- **Administrador/a de sistemes:** Per a la gestió de la infraestructura contenitzada i el monitoratge del rendiment.
- **Consultoria legal/seguretat:** Per assegurar el compliment normatiu del tractament de dades sensibles.

2.3 Viabilitat econòmica

S'ha realitzat una estimació de costos per avaluar la sostenibilitat del projecte en un escenari de comercialització real.

2.3.1 Estimació de costos operatius

A continuació, es detallen els costos aproximats mensuals per mantenir el sistema en producció. Cal destacar que aquests costos poden incrementar-se significativament en funció del volum d'usuaris i de la intensitat en el consum de tokens d'IA. El concepte de *Desenvolupament* correspon a una despesa única inicial i no es computa en el total recurrent mensual.

Concepte	Descripció	Cost estimat
Desenvolupament	Hores de disseny i implementació (cost únic)	300€ – 800€
Infraestructura cloud	Servidor virtual (VPS) per a Docker	10€ – 50€
Plataforma de dades	Subscripció Airtable (pla mensual per usuari)	10€ – 20€
Tokens d'IA	Consum de l'API segons volum d'ús	20€ – 100€
Altres serveis	Domini, seguretat i manteniment	5€ – 15€
TOTAL ESTIMAT		45€ – 185€

TAULA 2.1: Estimació de costos operatius mensuals del sistema

2.3.2 Model de negoci

Per garantir la rendibilitat, es plantegen diversos models de generació d'ingressos:

- **Model SaaS (*Software as a Service*¹):** Mitjançant una quota de subscripció mensual per l'ús de la plataforma. Aquest model es considera el més adequat, ja que permet assegurar ingressos recurrents i facilita l'escalabilitat del servei.
- **Pagament per ús:** Facturació basada en el volum de missatges processats o peticions a la IA.
- **Implementació personalitzada:** Un pagament únic inicial per la configuració i adaptació del sistema a les necessitats d'un client.

¹Model en què el programari s'ofereix com un servei accessible a través d'internet, sense necessitat d'instal·lació ni gestió d'infraestructura per part de l'usuari.

2.4 Viabilitat legal i ètica

2.4.1 Protecció de dades (RGPD)

El tractament de dades personals a través de webhooks obliga al compliment del Reglament General de Protecció de Dades (RGPD). Per mitigar riscos legals, el projecte incorpora:

- **Consentiment explícit:** Implementació de clàusules informatives abans que l'usuari interaccioni amb l'assistent.
- **Principi de minimització:** Només es recullen i processen les dades estrictament necessàries per a l'execució del servei.
- **Dret a l'oblit:** S'han dissenyat fluxos de neteja per eliminar registres antics d'Airtable i historials de converses de manera periòdica.

En un escenari real, el responsable del tractament de les dades hauria de definir clarament les polítiques de privacitat i assegurar el compliment normatiu en tots els punts d'integració amb serveis de tercers.

2.4.2 Ètica de la IA

L'ús de la IA es gestiona sota principis de transparència. Mitjançant la tècnica RAG (*Retrieval-Augmented Generation*), les respostes es limiten al context de la base de coneixements, reduint el nombre de respostes inapropiades. Tot i això, es reconeix la necessitat de supervisió periòdica per evitar biaixos algorítmics (vegeu Annex A).

2.5 Riscos i ciberseguretat

S'han valorat els aspectes de risc tecnològic i s'han proposat mesures de mitigació:

1. **Exposició de dades:** S'utilitza xifrat HTTPS i claus API per restringir l'accés als webhooks.
2. **Seguretat de la infraestructura:** L'ús de contenidors Docker aïlla els serveis, tot i que requereix una política estricta d'actualitzacions de seguretat de les imatges utilitzades.
3. **Dependència de tercers:** Risc d'interrupció per caigudes de les APIs. Per mitigar-ho, es planteja la implementació de mecanismes de *fallback* (respostes predefinides) com a Pla B per gestionar aquestes caigudes i evitar la degradació total del servei.

2.6 Conclusions de viabilitat

Un cop analitzats els factors, es conclou que el projecte és viable. Tecnològicament, se sustenta sobre eines robustes; econòmicament presenta uns costos operatius controlats amb un model SaaS escalable; i legalment s'han establert les bases per complir amb la normativa vigent. Malgrat les dependències externes identificades, el sistema presenta un equilibri òptim entre innovació i realització tècnica, essent apte per a la seva implementació en entorns reals.

Capítol 3

Metodologia

Per al desenvolupament del projecte s'ha adoptat una metodologia de tipus *Agile*, amb un enfocament iteratiu i incremental adaptat a un entorn de treball individual. Tot i que les metodologies *Agile* estan habitualment orientades a equips de treball, en aquest cas s'ha aplicat una adaptació simplificada, sense distribució de tasques entre membres, però mantenint la idea de desenvolupament progressiu i validació contínua. En aquest context, s'han dut a terme seguiments periòdics amb el tutor del TFG per revisar l'estat del projecte i orientar les decisions tècniques.

L'elecció d'aquesta metodologia es justifica per la naturalesa del projecte, que depèn fortament de la integració de múltiples tecnologies i serveis externs que sovint presenten limitacions no previstes inicialment. Una aproximació àgil ha permès desenvolupar funcionalitats de manera gradual, detectar incidències en fases primerenques i adaptar l'arquitectura del sistema segons els resultats obtinguts en cada etapa.

Tot i que des de l'inici es disposava d'una visió general clara dels dos casos d'ús plantejats, alguns detalls tècnics i decisions d'implementació s'han anat ajustant durant el desenvolupament. Per aquest motiu, un model flexible resultava més adequat que una planificació estrictament seqüencial.

El projecte s'ha estructurat en diverses iteracions successives. En una primera fase s'ha dut a terme l'anàlisi inicial de requisits, la definició de l'arquitectura general i la selecció de les tecnologies. A partir d'aquí, en iteracions posteriors, s'han desenvolupat de manera progressiva els diferents casos d'ús, incorporant noves funcionalitats i millorant els components existents segons els resultats obtinguts.

Cada iteració ha inclòs, de manera transversal, tasques d'anàlisi, disseny, implementació i validació. Aquest enfocament ha permès verificar de forma contínua el funcionament dels diferents mòduls i assegurar una evolució progressiva del sistema.

Finalment, la implementació s'ha iniciat en un entorn de desenvolupament controlat, que ha permès construir i ajustar el sistema abans del seu desplegament en un entorn més proper a producció.

3.1 Ús d'eines d'intel·ligència artificial

Durant l'elaboració del projecte s'han utilitzat eines d'intel·ligència artificial generativa com a suport puntual en diferents fases del treball.

Aquestes eines s'han emprat principalment per resoldre dubtes tècnics relacionats amb les tecnologies utilitzades, obtenir explicacions conceptuals i agilitzar determinades tasques de desenvolupament. En concret, l'ús més rellevant s'ha produït en els aspectes següents:

- Definició de *prompts* i instruccions per agents d'IA utilitzats dins dels fluxos d'n8n i Flowise, amb l'objectiu de millorar la interpretació de consultes, la qualitat de les respostes generades i la presa de decisions automàtica.
- Assistència en codi personalitzat dins dels nodes *Code* d'n8n, especialment per estructurar millor fragments de JavaScript, simplificar lògica interna i detectar errors d'implementació.
- Suport en configuracions d'infraestructura relacionades amb Docker i Docker Compose, incloent revisió d'arxius *docker-compose.yaml*, configuració de serveis i resolució d'errors d'arrencada.
- Anàlisi de logs i incidències tècniques, mostrant missatges d'error generats pels contenidors o serveis per identificar possibles incompatibilitats de versions, dependències mancants o errors de configuració.
- Consulta sobre eines externes integrades al projecte, per entendre millor funcionalitats i opcions de plataformes com Chatwoot, ngrok, Airtable, Evolution API o Telegram Bot API.
- Ajuda en la redacció del contingut de l'arxiu *RAG*, el document de coneixement utilitzat pel primer cas d'ús. A partir d'idees generals proporcionades per l'autor, la IA va ajudar a estructurar i redactar contingut amb format coherent i orientat a un context empresarial.
- Assistència puntual en comandes de sistema, instal·lació de paquets, configuració de servidor Linux i execució de comandes necessàries per preparar l'entorn de desenvolupament i desplegament.
- Generació de la interfície de benvinguda (*Landing page*) del sistema, disponible a <https://automatitzacions.dev>. Aquesta pàgina web estàtica (fitxers *index.html*, *style.css* i *script.js*) ha estat generada íntegrament mitjançant eines d'intel·ligència artificial a partir d'instruccions proporcionades per l'autor. Aquesta interfície té una funció exclusivament visual i de navegació, i no forma part de la lògica ni de la implementació tècnica del sistema.

En cap cas aquestes eines han substituït el treball tècnic ni les decisions principals del projecte. L'anàlisi de requisits, l'arquitectura del sistema, la implementació final, la validació dels resultats i la redacció global de la memòria s'han desenvolupat dins del marc del projecte acadèmic.

Capítol 4

Planificació

La planificació del projecte s'ha estructurat en diferents fases que segueixen una evolució progressiva des de la preparació de l'entorn fins al desplegament final en producció. Aquesta planificació està alineada amb una metodologia incremental, en la qual cada etapa construeix sobre l'anterior, permetent validar funcionalitats de manera contínua.

1. Preparació de l'entorn

- Tasques:
 - **Configuració Docker i serveis:** Es defineix el fitxer *docker-compose.yaml* amb tots els serveis necessaris i es despleguen en contenidors.
 - **Configuració variables d'entorn:** Es preparen els fitxers *.env* amb les credencials i configuracions necessàries per a cada servei.
 - **Verificació funcionament serveis:** Es comprova que tots els serveis estan operatius i accessibles correctament.

2. Definició del projecte

- Tasques:
 - **Anàlisi de casos d'ús:** Es seleccionen els escenaris més representatius per demostrar el potencial de l'automatització amb IA.
 - **Definició requisits funcionals:** S'estableixen les funcionalitats que ha de complir el sistema en base als objectius del projecte.
 - **Selecció tecnologies a utilitzar:** Es trien les eines i tecnologies més adequades tenint en compte la seva integració, escalabilitat i les més destacades dins la comunitat.

3. Integració serveis externs

- Tasques:
 - **Connexió WhatsApp amb API:** Es vincula el compte de WhatsApp amb Evolution API per habilitar la comunicació amb el sistema.

- **Configuració Chatwoot:** Es prepara la plataforma per centralitzar els missatges provinents de diferents canals.
- **Integració correu electrònic:** Es configuren els protocols IMAP i SMTP per gestionar l'enviament i recepció de correus.
- **Validació enviament i recepció de missatges:** Es realitzen proves per assegurar que tots els canals funcionen correctament.

4. Cas d'ús 1, atenció al client

- Tasques:
 - **Investigació n8n i Chatwoot:** S'analitzen les funcionalitats disponibles per aprofitar-ne al màxim les capacitats.
 - **Connexió webhooks amb n8n:** Es configura la recepció d'esdeveniments des de Chatwoot per activar els fluxos d'automatitzacions.
 - **Implementació de les funcionalitats:** Es desenvolupen els fluxos per automatitzar la gestió de la comunicació amb clients.
 - **Optimització i millora flux:** Es millora l'eficiència del sistema reduint redundàncies i evitant comportaments no desitjats.
 - **Integració Flowise:** Es dissenya un flux a Flowise que actua com el nucli intel·ligent de l'agent, encarregat de processar la informació i generar respostes coherents i contextuals.
 - **Implementació sistema RAG:** Es desenvolupa un sistema de recuperació d'informació per generar respostes més precises i contextualitzades.

5. Cas d'ús 2, assistència cognitiva

- Tasques:
 - **Creació i configuració bot Telegram:** Es crea i configura un bot de Telegram que serà la interfície d'interacció amb l'usuari.
 - **Configuració webhooks amb ngrok:** Es permet la comunicació entre Telegram i l'entorn local mitjançant una URL pública.
 - **Disseny base de dades Airtable:** Es defineixen les estructures necessàries per emmagatzemar la informació de l'usuari.
 - **Implementació de les funcionalitats:** Es desenvolupen les diferents comandes i automatitzacions del bot.
 - **Neteja i manteniment de dades:** S'implementa un nou flux per evitar l'acumulació de dades innecessàries.

- **Sistema de recordatoris automàtics:** S'implementa un nou flux per enviar notificacions periòdiques relacionades amb tasques i medicacions pendents.

6. Desplegament en producció

- Tasques:
 - **Configuració màquina virtual (Azure):** Es prepara l'entorn de producció amb els recursos necessaris per executar el sistema.
 - **Desplegament serveis amb Docker:** Es posen en funcionament tots els serveis mitjançant *docker-compose*.
 - **Configuració domini i SSL:** Es configura el domini i el certificat SSL per poder accedir al sistema de manera segura.
 - **Importació fluxos n8n i Flowise:** S'importen els fluxos d'n8n i Flowise que es van desenvolupar prèviament en local.
 - **Reconfiguració integracions externes:** S'ajusten de nou les configuracions de Chatwoot i EvolutionAPI per l'entorn de producció.

7. Validació i proves

- Tasques:
 - **Proves funcionals del sistema:** Es verifica el correcte funcionament de totes les funcionalitats i integracions.

8. Redacció memòria

- Tasques:
 - **Redacció documentació tècnica:** Es documenta el procés de desenvolupament i les decisions preses.
 - **Revisió i correcció final:** Es realitza una revisió global per assegurar la coherència i qualitat del document.

	Tasques	Gener				Febrer				Març				Abril				Maig					
		S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4		
1. Preparació de l'entorn	Configuració Docker i serveis																						
	Configuració variables d'entorn																						
	Verificació funcionament serveis																						
	Anàlisi de casos d'ús																						
2. Definició del projecte	Definició requisits funcionals																						
	Selecció tecnologies a utilitzar																						
	Connexió WhatsApp amb API																						
	Configuració Chatwoot																						
3. Integració serveis externs	Integració correu electrònic																						
	Validació enviament / recepció missatges																						
	Investigació n8n i Chatwoot																						
	Connexió webhooks amb n8n																						
4. Cas d'ús 1, atenció al client	Implementació de les funcionalitats																						
	Optimització i millora flux																						
	Integració Flowise																						
	Implementació sistema RAG																						
5. Cas d'ús 2, assistència cognitiva	Creació i configuració bot Telegram																						
	Configuració webhooks amb ngrok																						
	Disseny base de dades Airtable																						
	Implementació de les funcionalitats																						
	Neteja i manteniment de dades																						
	Sistema de recordatoris automàtics																						
6. Desplegament en producció	Configuració màquina virtual (Azure)																						
	Desplegament serveis amb Docker																						
	Configuració domini i SSL																						
	Importació fluxos n8n i Flowise																						
7. Validació i proves	Reconfiguració integracions externes																						
	Proves funcionals del sistema																						
8. Redacció memòria	Redacció documentació tècnica																						
	Revisió i correcció final																						

TAULA 4.1: Cronograma de planificació del projecte.

Capítol 5

Marc de treball i conceptes previs

El present capítol descriu el marc de treball del projecte i introdueix els conceptes necessaris per entendre el sistema desenvolupat. S'hi presenten l'entorn utilitzat i els principis tecnològics que sustenten la implementació del sistema.

El desenvolupament s'ha realitzat en un entorn propi, sense vinculació amb cap empresa, utilitzant eines actuals d'automatització i integració de serveis. Inicialment, s'ha treballat en local per validar components i realitzar proves, i posteriorment el sistema s'ha desplegat en un servidor virtual per simular un entorn de producció. A continuació, es presenten els conceptes i tecnologies clau utilitzats durant el projecte.

5.1 Automatització de processos

L'automatització de processos fa referència a l'ús de sistemes tecnològics per executar tasques de manera automàtica, reduint la intervenció humana i millorant l'eficiència. Aquests sistemes permeten definir fluxos de treball activats per determinats esdeveniments, com ara la recepció d'un missatge o la generació d'una incidència. En aquest projecte, els fluxos d'automatització constitueixen la base del sistema i permeten coordinar les funcionalitats implementades en els dos casos d'ús.

5.2 Intel·ligència artificial

La intel·ligència artificial és un conjunt de tècniques que permeten als sistemes informàtics simular capacitats pròpies de la intel·ligència humana, com el processament del llenguatge natural o la presa de decisions [1]. En aquest projecte, la IA juga un paper clau en la interpretació de les comunicacions i en la generació de respostes automatitzades, aportant un nivell d'intel·ligència als processos automatitzats.

5.3 Integració de serveis

La integració de serveis consisteix en la connexió de diferents aplicacions i plataformes perquè puguin intercanviar informació de manera automàtica. Aquesta integració és essencial en sistemes d'automatització, ja que permet coordinar múltiples eines dins d'un mateix flux de treball. En aquest projecte, facilita la comunicació entre plataformes com Chatwoot, Telegram, WhatsApp o Airtable.

5.4 Autenticació i gestió de credencials

Per integrar serveis externs dins del sistema, és necessari configurar mecanismes d'autenticació que permetin establir comunicacions segures entre plataformes. Aquesta integració es realitza mitjançant credencials com *API keys*, tokens d'accés o protocols *OAuth2*.

En aquest projecte, la gestió de credencials s'ha centralitzat mitjançant el sistema proporcionat per n8n, que permet emmagatzemar i reutilitzar connexions de manera segura entre diferents serveis externs. Aquesta aproximació facilita una arquitectura modular i escalable, on cada servei es pot configurar de manera independent i reutilitzar en múltiples automatitzacions.

Per a més detall sobre el procés de configuració de les credencials utilitzades, es pot consultar l'Annex B.

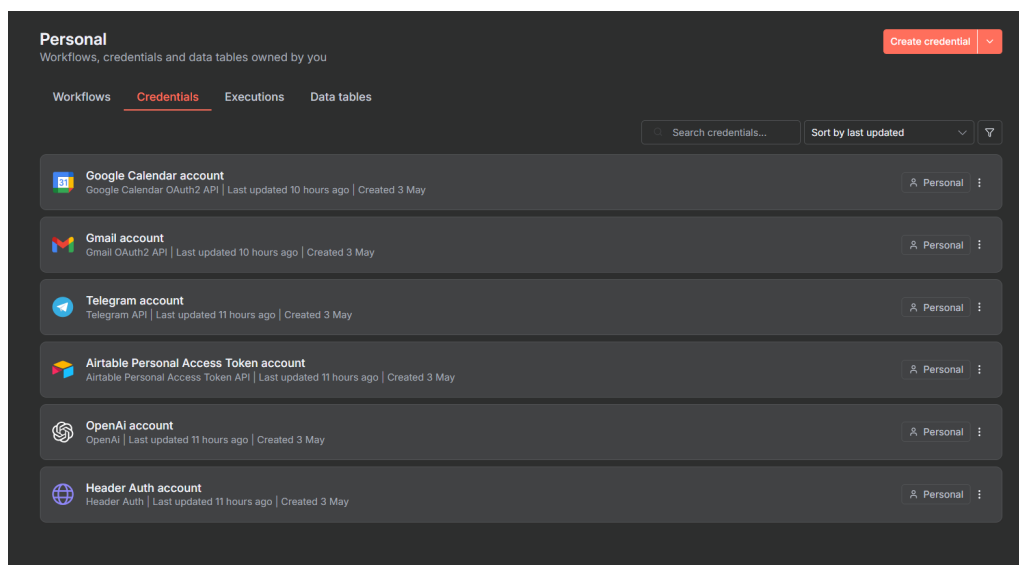


FIGURA 5.1: Vista de la interfície de configuració i xifratge de credencials d'usuari a la plataforma n8n.

5.5 Tecnologies de contenidors

Les tecnologies de contenidors permeten empaquetar aplicacions i dependències en entorns aïllats i reproduïbles [2]. Aquest enfocament facilita el desplegament i la gestió dels serveis mantenint la coherència entre diferents entorns. En aquest projecte s'ha utilitzat Docker per gestionar els diferents serveis del sistema.

5.6 Webhooks i comunicació basada en esdeveniments

Els webhooks són un mecanisme que permet que un sistema notifiqui automàticament un altre sistema quan es produeix un esdeveniment concret. A diferència de les APIs tradicionals basades en *polling* [3], els webhooks funcionen de manera reactiva: el sistema emissor envia la informació quan es produeix un esdeveniment.

Aquest mecanisme utilitza peticions HTTP, habitualment de tipus POST, que contenen informació estructurada, normalment en format JSON. El sistema emissor envia aquesta informació a una URL prèviament configurada, anomenada *endpoint* del webhook, i el sistema receptor processa les dades rebudes.

Aquestes notificacions poden correspondre a diferents tipus d'esdeveniments, com ara *conversation_updated*, *message_created* o *message_updated* en el cas de Chatwoot, fet que permet construir automatitzacions específiques segons el cas d'ús. Cada esdeveniment indica una situació diferent dins de la conversa: per exemple, *message_created* indica que s'ha registrat un nou missatge dins de la conversa, mentre que *message_updated* reflecteix que un missatge existent ha estat modificat. La selecció de l'esdeveniment adequat és clau, ja que determina el comportament i l'activació del flux d'automatització.

5.7 Funcionalitats d'automatització de Chatwoot

Chatwoot ofereix diverses eines que es poden combinar entre elles per construir automatitzacions avançades i personalitzades. En aquest projecte s'han utilitzat principalment tres mecanismes: els atributs personalitzats, les etiquetes i les regles d'automatització. A continuació, es descriu l'aplicació d'aquestes eines per desenvolupar funcionalitats progressivament més complexes dins del flux.

5.7.1 Atributs personalitzats

Una de les funcionalitats més rellevants de Chatwoot és la possibilitat de crear atributs personalitzats associats a les converses. Aquests atributs permeten emmagatzemar informació addicional i poden ser de diferents tipus, com ara textos, números o llistes de valors.

Aquests camps permeten enriquir la informació disponible dins del sistema i facilitar la presa de decisions en processos automatitzats, ja que poden ser utilitzats com a variables dins de les regles o com a condicions en fluxos externs.

5.7.2 Etiquetes

Les etiquetes són un mecanisme de classificació que permet agrupar i organitzar les converses segons diferents criteris. Chatwoot permet crear i assignar múltiples etiquetes a cada conversa, facilitant la seva identificació i gestió.

A més de la seva funció organitzativa, les etiquetes poden ser utilitzades com a element clau dins de sistemes d'automatització, ja que permeten indicar l'estat d'una conversa o activar determinades accions en funció del seu valor.

5.7.3 Regles d'automatització

Les regles d'automatització permeten definir comportaments automàtics dins de Chatwoot a partir de condicions específiques. El seu funcionament es basa en una estructura senzilla: quan es compleixen una o diverses condicions, s'executen una o més accions.

Aquestes accions poden incloure, entre d'altres, l'assignació o eliminació d'etiquetes, la modificació d'atributs personalitzats o altres operacions sobre la conversa. Aquest

mecanisme permet automatitzar tasques repetitives i establir lògiques de funcionament sense necessitat d'intervenció manual.

5.8 Sistemes RAG (Retrieval-Augmented Generation)

El *Retrieval-Augmented Generation* (RAG) és una tècnica de la intel·ligència artificial generativa que combina la recuperació d'informació amb la generació de text. Aquest enfocament permet que els models de llenguatge generin respostes basades no només en el seu entrenament previ, sinó també en informació externa específica, millorant així la precisió i la rellevància de les respostes.

El funcionament general d'un sistema RAG consisteix a consultar una base de coneixement per recuperar informació relacionada amb la pregunta de l'usuari i utilitzar aquest context per generar una resposta coherent i contextualitzada.

Per facilitar aquest procés, els documents es divideixen en fragments i es transformen en representacions numèriques anomenades *embeddings*, que permeten comparar semànticament les consultes amb la informació emmagatzemada.

En aquest projecte, aquesta tècnica s'ha utilitzat per proporcionar respostes basades exclusivament en informació corporativa definida prèviament, reduint el risc de respostes incorrectes o fora de context.

Per a més detall sobre la configuració i el funcionament tècnic del sistema RAG implementat, vegeu l'Annex K.

5.9 Exposició de serveis mitjançant Ngrok

En entorns de desenvolupament local, les aplicacions sovint s'executen en servidors accessibles únicament des de la pròpia màquina, utilitzant adreces com *localhost*. Aquesta limitació impedeix que serveis externs puguin establir comunicació directa amb l'aplicació, fet que dificulta la integració amb sistemes basats en esdeveniments, com ara els webhooks.

Per solucionar aquesta problemàtica, existeixen eines que permeten exposar serveis locals a Internet de manera segura. Una de les més utilitzades és Ngrok, que actua com a intermediari entre l'entorn local i la xarxa pública.

Ngrok crea un túnel segur entre un port local i una URL pública accessible des d'Internet [4]. D'aquesta manera, les peticions externes rebudes a través d'aquesta URL són redirigides automàticament cap al servei que s'està executant en local. Això permet que aplicacions en desenvolupament puguin interactuar amb serveis externs com si estiguessin desplegades en un entorn de producció.

Aquest mecanisme és especialment útil en escenaris on es requereix una URL pública per rebre notificacions, com és el cas dels webhooks de plataformes de missatgeria o APIs externes.

Capítol 6

Requisits del sistema

Aquests requisits s'han definit a partir dels dos casos d'ús plantejats i serveixen com a base per al disseny i implementació del sistema.

6.1 Requisits funcionals

- **Gestió de comunicacions multicanal:** El sistema ha de permetre la recepció i l'enviament de missatges a través de diferents canals, com WhatsApp i correu electrònic, mitjançant la integració amb Chatwoot. Prioritat: alta.
- **Automatització de respostes:** El sistema ha de generar respostes automàtiques als missatges rebuts, tenint en compte el context de la conversa i utilitzant tècniques d'intel·ligència artificial. Prioritat: alta.
- **Gestió de la informació de l'usuari:** El sistema ha de permetre emmagatzemar i consultar informació rellevant de l'usuari, com ubicacions, historial de missatges, medicació i tasques. Prioritat: alta.
- **Seguiment de la ubicació:** El sistema ha de registrar periòdicament la ubicació de l'usuari a partir de la compartició en temps real via Telegram, amb l'objectiu de permetre'n el seu posterior processament. Prioritat: alta.
- **Gestió d'ubicacions importants:** El sistema ha de permetre a l'usuari definir i consultar ubicacions rellevants per facilitar la generació d'informació contextual. Prioritat: mitjana.
- **Generació de resums automàtics:** El sistema ha de generar resums automàtics basats en diferents fonts d'informació, com correus electrònics, converses amb el bot i historial d'ubicacions. Prioritat: alta.
- **Gestió de tasques i esdeveniments:** El sistema ha de permetre consultar els esdeveniments del calendari, marcar-los com a completats i visualitzar-ne l'estat. Prioritat: alta.
- **Sistema de recordatoris:** El sistema ha d'enviar recordatoris automàtics a l'usuari, incloent-hi resums diaris i recordatoris de medicació segons la seva freqüència. Prioritat: alta.
- **Exercicis cognitius:** El sistema ha de proporcionar exercicis de tipus trivial i registrar la puntuació obtinguda per l'usuari. Prioritat: mitjana.

6.2 Requisits no funcionals

- **Escalabilitat:** El sistema ha de poder créixer fàcilment afegint nous serveis o funcionalitats gràcies a una arquitectura basada en microserveis i contenidors.
- **Modularitat:** Els diferents components (n8n, Chatwoot, Flowise, etc.) han d'estar desacoblats, permetent modificar-los o substituir-los sense afectar la resta del sistema.
- **Rendiment:** El sistema ha de respondre de manera eficient als missatges i execucions de fluxos, evitant retards significatius en la resposta a l'usuari.
- **Seguretat:** El sistema ha de garantir la protecció de les comunicacions mitjançant l'ús de protocols segurs (HTTPS). A més, tots els serveis desplegats han d'estar aïllats en contenidors per reduir la superfície d'atac, i l'accés a la màquina virtual ha d'estar restringit mitjançant claus SSH.
- **Mantenibilitat:** El sistema ha de ser fàcil de mantenir i actualitzar gràcies a l'ús de Docker i a una estructura clara dels fluxos de treball.
- **Portabilitat:** El sistema ha de poder desplegar-se tant en entorns locals com en entorns al núvol sense modificacions significatives.
- **Usabilitat:** La interacció amb l'usuari ha de ser senzilla i intuïtiva.
- **Disponibilitat:** El sistema ha de garantir un funcionament continu i estable un cop desplegat en producció.

6.3 Requisits de *hardware*

Per garantir el funcionament correcte del sistema, s'han definit uns requisits mínims de maquinari basats en la càrrega dels serveis conteniritzats.

El requisit més crític identificat és la memòria RAM. Tot i que una configuració de 2 GiB podria semblar suficient per a processos aïllats, l'execució simultània de l'orquestració definida al fitxer *docker-compose* (que inclou instàncies d'n8n, bases de dades, i motors d'IA) requereix un mínim de 4 GiB de RAM. Durant les proves tècniques, s'ha constatat que valors inferiors provoquen el desbordament de la memòria i la interrupció crítica dels serveis en el moment de l'arrencada.

A la taula següent es mostren les característiques del maquinari utilitzat tant en l'entorn de desenvolupament com en el de producció.

Característiques	Local	Producció
Processador	Intel Core i7-1165G7	Standard B2als v2 (2 vCPU)
Targeta gràfica	Intel Iris Xe Graphics	-
Memòria RAM	16 GB	4 GiB
Disc dur	512 GB	64 GB
Sistema operatiu	Windows 11	Ubuntu 24.04 LTS Server (Linux)

TAULA 6.1: Característiques de *hardware* utilitzat pel projecte.

Capítol 7

Estudis i decisions

7.1 Maquinari

El desenvolupament del projecte s'ha realitzat inicialment en un entorn local mitjançant un ordinador personal, suficient per a la configuració i execució dels serveis necessaris gràcies a l'ús de contenidors Docker. Posteriorment, el sistema s'ha desplegat en una màquina virtual al núvol (Azure), aprofitant els avantatges d'escalabilitat, accessibilitat remota i disponibilitat contínua.

Aquesta combinació d'entorn local i entorn de producció ha permès separar les fases de desenvolupament i desplegament, facilitant la validació del sistema abans de la seva posada en funcionament.

7.2 Programari i eines principals

El projecte es basa en una arquitectura modular composta per diverses eines integrades entre si:

- **Docker:** Plataforma de contenització que permet executar aplicacions en entorns aïllats. Aquesta decisió garanteix la portabilitat i la coherència entre entorns.
- **n8n:** Plataforma d'automatització de fluxos que permet definir processos mitjançant una interfície visual basada en nodes.
- **Chatwoot:** Sistema de gestió de comunicacions que permet centralitzar interaccions provinents de múltiples canals.
- **EvolutionAPI:** Eina que permet connectar WhatsApp amb el sistema, actuant com a intermediari per a l'enviament i recepció de missatges.
- **Flowise:** Eina per a la creació de fluxos d'intel·ligència artificial amb suport per a models de llenguatge i sistemes RAG.
- **PostgreSQL:** Sistema de gestió de bases de dades utilitzat per Chatwoot, que el fa servir per emmagatzemar la informació de converses i usuaris.
- **Redis:** Sistema de memòria cau utilitzat internament per alguns serveis com Chatwoot per gestionar cues de processos i optimitzar el rendiment. És un component necessari per al correcte funcionament del sistema.

- **AirTable:** Base de dades utilitzada per emmagatzemar informació estructurada (ubicacions, medicació, historial de missatges, etc.). La seva facilitat d'ús i API integrada la fan adequada per a prototips i aplicacions amb requisits moderats.
- **Telegram Bot API:** Utilitzada per implementar el bot d'assistència cognitiva. Permet la comunicació bidireccional amb l'usuari mitjançant webhooks.
- **Ngrok:** Eina utilitzada durant el desenvolupament per exposar serveis locals a internet, permetent rebre webhooks de Telegram en entorn local.
- **Nginx i Certbot:** Utilitzats en la fase de desplegament per gestionar el servidor web i configurar connexions segures mitjançant HTTPS.

7.3 Llibreries i tecnologies utilitzades

A nivell de desenvolupament i integració del sistema, s'han utilitzat principalment:

- **JavaScript:** Per implementar la lògica personalitzada dins dels nodes *Code*. Aquesta elecció permet una gran flexibilitat en el processament de dades i en la implementació de condicions complexes.
- **Luxon:** Llibreria nativa en n8n per a la gestió de dates i hores. S'ha utilitzat per calcular intervals temporals, gestionar zones horàries i comparar dates de manera fiable en els fluxos d'automatització.
- **Baileys:** Llibreria de Node.js que actua com a motor intern d'Evolution API. S'utilitza per connectar i automatitzar la interacció amb els servidors de WhatsApp Web, fent possible l'enviament i la recepció de missatges des de números estàndard o personals sense requerir la infraestructura oficial de Meta.

Aquestes tecnologies han estat escollides per la seva simplicitat, robustesa i integració amb l'arquitectura del projecte.

7.4 Justificació de les decisions

Elecció d'un entorn de desenvolupament local Atesa la naturalesa del projecte, basada en la integració de múltiples serveis (plataformes de missatgeria, eines d'automatització, serveis d'intel·ligència artificial i bases de dades), es va considerar necessari disposar d'un entorn de proves estable abans de desplegar el sistema en producció. Durant la fase inicial, es va optar per treballar en un entorn local de desenvolupament amb els següents objectius:

- Realitzar proves iteratives sense afectar un entorn públic
- Validar la interacció entre serveis i resoldre errors d'integració
- Experimentar amb configuracions sense risc operatiu
- Reduir costos durant la fase experimental

Aquest enfocament va permetre aplicar una metodologia incremental, validant progressivament cada component abans del desplegament final. A més, es va decidir utilitzar com a punt de partida un repositori públic basat en Docker i Docker Compose, amb l'objectiu d'accelerar la configuració inicial del sistema.

Arquitectura modular amb Docker Pel que fa a l'arquitectura, s'ha decidit contenir tots els serveis mitjançant Docker. Aquesta decisió respon a la necessitat de construir un sistema modular, escalable i fàcilment mantenible. La virtualització mitjançant contenidors permet encapsular cada servei amb totes les seves dependències, assegurant un comportament consistent independentment de l'entorn d'execució. L'ús de Docker Compose permet gestionar tots els serveis de manera centralitzada, simplificant tant la posada en marxa com l'administració global de l'arquitectura.

En comparació amb altres alternatives com màquines virtuals tradicionals, aquest enfocament redueix el consum de recursos i facilita la portabilitat del sistema.

Selecció d'n8n com a motor d'automatització En aquest context, l'elecció d'una arquitectura basada en eines com n8n reforça aquesta filosofia modular. n8n permet desenvolupar fluxos complexos sense necessitat de programació intensiva, reduint el temps de desenvolupament i facilitant la seva modificació i escalabilitat futura.

Durant la fase d'anàlisi es van considerar alternatives com Zapier o Make, que ofereixen una gran facilitat d'ús i integracions predefinides. No obstant això, aquestes plataformes presenten limitacions en entorns autoal·lotjats i menor flexibilitat en la personalització dels fluxos. En canvi, n8n permet desplegament propi (*self-hosted*¹) i la incorporació de codi personalitzat, oferint un major control sobre el sistema.

Per aquest motiu, es considera una eina més adequada per a un entorn d'automatització complex i integrat.

Integració de serveis externs Pel que fa a la integració de canals de comunicació, l'ús de serveis com Chatwoot i Evolution API permet incorporar canals reals com WhatsApp i correu electrònic dins del sistema, aportant un alt valor pràctic i apropant la solució a un entorn real d'aplicació.

En aquest àmbit, es van valorar alternatives com Zendesk o Freshchat, que ofereixen funcionalitats avançades orientades a entorns empresarials. Tot i això, aquestes eines solen requerir subscripcions de pagament i presenten més limitacions en la seva personalització i integració amb sistemes propis. Chatwoot, en canvi, és una solució *open-source* que permet també desplegament propi i accés directe a la seva API, facilitant la seva integració dins d'una arquitectura modular.

Pel que fa a la integració amb WhatsApp, també existeixen opcions oficials com *WhatsApp Business API* (a través de proveïdors com Twilio). No obstant això, aquestes impliquen processos de validació més estrictes. Evolution API s'ha escollit per la seva simplicitat d'ús, facilitat d'integració i l'absència de costos per volum de missatgeria en comparació amb els proveïdors oficials.

¹L'eina s'executa en servidors propis, sota control total de l'organització, en lloc de dependre d'una plataforma allotjada per tercers.

D'altra banda, Airtable simplifica la gestió de dades, evitant la necessitat de desplegar i mantenir una base de dades tradicional, fet que redueix la complexitat del sistema.

Incorporació de Flowise i tècniques RAG Tot i que inicialment es va utilitzar la integració directa amb serveis d'intel·ligència artificial mitjançant nodes d'OpenAI dins d'n8n, aquesta solució presentava limitacions en escenaris més avançats, especialment en la gestió del context conversacional entre interaccions.

Es van considerar alternatives com l'ús directe d'APIs de models de llenguatge o *frameworks*² com LangChain, que tot i ser la base de moltes d'aquestes eines, requereixen un desenvolupament basat estrictament en codi, fet que hauria incrementat el temps d'implementació. Per aquest motiu, es va incorporar Flowise com a capa intermèdia especialitzada en la construcció d'agents d'intel·ligència artificial.

Aquesta eina permet gestionar de manera unificada el context, l'estat de la conversa i les sessions d'usuari, facilitant interaccions més coherents i naturals. A més, incorpora tècniques RAG (*Retrieval-Augmented Generation*), que permeten enriquir les respostes amb informació recuperada de fonts internes, incrementant la precisió i rellevància del sistema.

Selecció de Telegram com a plataforma de comunicació En relació amb la interacció amb l'usuari, s'ha escollit Telegram com a plataforma per implementar el bot d'assistència cognitiva. Aquesta decisió es basa en la seva àmplia popularitat i en la facilitat d'ús de la seva API.

En comparació amb altres plataformes de missatgeria, Telegram ofereix una configuració de bots més senzilla i una documentació clara, fet que permet una integració ràpida amb altres serveis com n8n.

Desplegament en producció Finalment, pel que fa al desplegament en producció, s'ha seleccionat una màquina virtual amb 4 GiB de memòria RAM, 2 vCPU i 64 GB d'emmagatzematge. Inicialment, es va utilitzar una configuració amb 2 GiB de RAM, però es va demostrar insuficient per executar simultàniament tots els serveis definits en el docker-compose. Aquesta limitació provocava bloquejos freqüents del sistema i la necessitat de reiniciar la màquina.

Per aquest motiu, es va optar per augmentar els recursos disponibles, garantint així un funcionament estable i fluid del sistema en producció.

Conclusions En conjunt, les decisions preses permeten construir un sistema complet, funcional i escalable, amb una arquitectura moderna basada en serveis desacoblats i fàcilment integrables.

²Conjunt d'eines i estructures predefinides que faciliten el desenvolupament d'aplicacions aportant una base organitzada sobre la qual desenvolupar-les.

Capítol 8

Anàlisi i disseny del sistema

8.1 Anàlisi del sistema

8.1.1 Tipus d'usuari

El sistema desenvolupat contempla dos casos d'ús diferenciats, cadascun amb els seus propis tipus d'usuari i rols associats.

Cas d'ús 1 – Atenció al client automatitzada:

- **Treballador de l'empresa:** Estableix les regles de resposta que ha de donar l'assistent virtual i supervisa les converses amb els clients. També pot intervenir manualment en cas necessari.
- **Client:** És l'usuari final que interactua amb el sistema a través de canals com WhatsApp o correu electrònic. Realitza consultes i rep respostes generades de forma automàtica o assistida. No té control sobre el sistema, només interactua amb ell.

Cas d'ús 2 – Assistència cognitiva:

- **Usuari:** Persona amb possibles dificultats de memòria que utilitza el bot de Telegram com a assistent personal. Pot registrar informació (tasques, ubicacions, medicació) i rebre recordatoris i resums automàtics. Interactua exclusivament mitjançant text, comandes o botons dins de Telegram.

8.1.2 Anàlisi de requisits

- **Gestió de comunicacions multicanal:** Aquest requisit defineix la capacitat del sistema per gestionar comunicacions a través de múltiples canals, principalment WhatsApp i correu electrònic, centralitzant tota la informació en una única plataforma.

Per a la seva implementació, s'utilitza Chatwoot com a gestor central de converses, mentre que Evolution API permet la integració amb WhatsApp. El sistema rep els missatges entrants mitjançant webhooks i envia les respostes utilitzant els mateixos canals.

Les dades implicades inclouen el contingut dels missatges, els identificadors

de conversa i informació de l'usuari (identificador del xat de chatwoot, identificador del contacte, nom de l'usuari, número de telèfon o correu electrònic), les marques temporals (*timestamps*), el canal de comunicació (WhatsApp o correu electrònic), el tipus de missatge rebut (text, imatge, audio), i el tipus d'esdeveniment (per exemple, *message_created* o *conversation_updated*).

- **Automatització de respostes:** Aquest requisit estableix que el sistema ha de generar respostes automàtiques als missatges rebuts, interpretant-ne el contingut i el context.

Per fer-ho, s'integra amb eines d'intel·ligència artificial com Flowise, que permeten processar el llenguatge natural i generar respostes coherents. El sistema utilitza el contingut del missatge i l'historial de la conversa per mantenir la coherència en les respostes.

Les dades implicades inclouen el text del missatge, l'historial conversacional i informació bàsica de l'usuari (com el nom o l'idioma). També es contempla la possibilitat d'intervenció manual en cas que la resposta automàtica no sigui suficient.

- **Gestió de la informació de l'usuari:** Aquest requisit implica la capacitat del sistema per emmagatzemar i gestionar informació rellevant de cada usuari, com l'historial de missatges, ubicacions, tasques o medicació.

Per a la seva implementació, s'utilitza Airtable com a base de dades no relacional, que permet una gestió flexible de la informació. Cada usuari està associat a múltiples registres relacionats amb la seva activitat.

Les dades inclouen identificadors d'usuari, informació bàsica i registres associats (missatges, ubicacions, tasques, etc.). Aquest requisit garanteix la persistència de la informació i la seva disponibilitat per als diferents fluxos d'automatització.

- **Seguiment de la ubicació:** Aquest requisit estableix que el sistema ha de registrar la ubicació de l'usuari en temps real a través de Telegram.

El sistema rep coordenades geogràfiques mitjançant webhooks i les emmagatzema de manera periòdica per al seu posterior processament. Això permet generar funcionalitats com el seguiment d'hàbits o la creació de resums d'activitat.

Les dades implicades inclouen latitud, longitud, data i hora. A més, es pot associar informació contextual com el carrer o la ciutat. També es controla la freqüència d'emmagatzematge per evitar un volum excessiu de dades, ja que la ubicació es pot actualitzar cada 30-60 segons.

- **Gestió d'ubicacions importants:** Aquest requisit permet a l'usuari definir ubicacions rellevants, com ara casa o llocs habituals, per aportar context al sistema.

El sistema permet afegir, consultar i utilitzar aquestes ubicacions dins dels fluxos d'automatització. Això implica associar noms descriptius a coordenades geogràfiques i relacionar-los amb l'usuari.

Les dades inclouen el nom de la ubicació, coordenades i informació associada (carrer i ciutat). Aquest requisit millora altres funcionalitats, com la generació de resums o respostes contextualitzades. També permet mostrar enllaços a Google Maps per facilitar la visualització.

- **Generació de resums automàtics:** Aquest requisit defineix la capacitat del sistema per generar resums a partir de diverses fonts d'informació, com correus electrònics, converses i dades d'ubicació.

El sistema recopila la informació rellevant, la processa i genera un resum coherent mitjançant models d'intel·ligència artificial. Per fer-ho, cal definir criteris de selecció per evitar incloure informació irrellevant.

Les dades implicades inclouen contingut de missatges, registres d'activitat i informació contextual. L'objectiu és oferir un resum clar i útil sense sobrecarregar l'usuari.

- **Gestió de tasques i esdeveniments:** Aquest requisit estableix que el sistema ha de permetre gestionar tasques vinculades al calendari de Google de l'usuari.

El sistema permet consultar esdeveniments, mostrar-ne la informació i marcar-los com a completats mitjançant interacció directa (botons). També permet fer seguiment de les tasques del dia actual i consultar les del dia anterior.

Les dades inclouen identificador de la tasca, descripció, data i estat (pendent o completada). Aquest requisit facilita el control de l'activitat diària de l'usuari.

- **Sistema de recordatoris:** Aquest requisit defineix la capacitat del sistema per enviar notificacions automàtiques en funció de configuracions prèvies.

El sistema utilitza mecanismes de planificació (*schedule triggers*¹) per enviar recordatoris, com el resum diari o avisos de medicació segons una freqüència definida.

Les dades implicades inclouen horaris, freqüències, tipus de recordatori i informació de l'usuari. Aquest requisit permet automatitzar tasques recurrents sense intervenció manual.

- **Exercicis cognitius:** Aquest requisit estableix que el sistema ha de proporcionar activitats orientades a estimular la memòria de l'usuari.

El sistema genera preguntes de tipus trivial, recull les respostes i calcula la puntuació. Posteriorment, emmagatzema els resultats per permetre'n el seguiment.

Les dades inclouen preguntes, respostes, puntuacions, data i hora. Aquest requisit permet analitzar l'evolució de l'usuari al llarg del temps.

¹Node que activa automàticament un flux cada cert interval de temps, funcionant com un temporitzador programat que no depèn de cap esdeveniment extern.

8.1.3 Casos d'ús

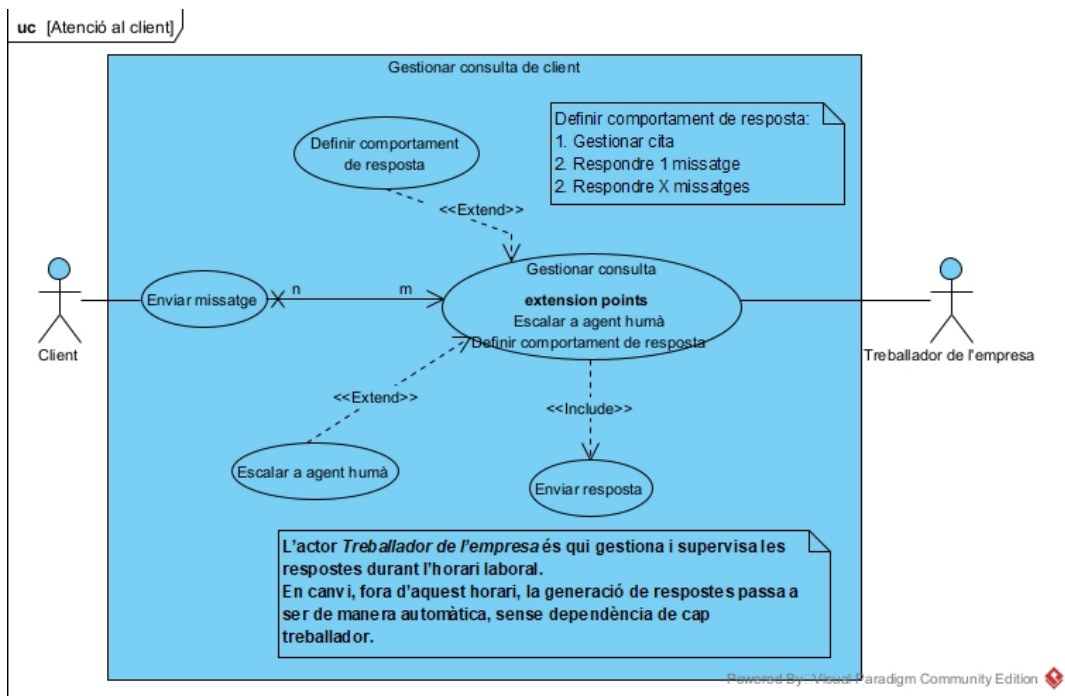


FIGURA 8.1: Diagrama de casos d'ús per al sistema d'atenció al client.

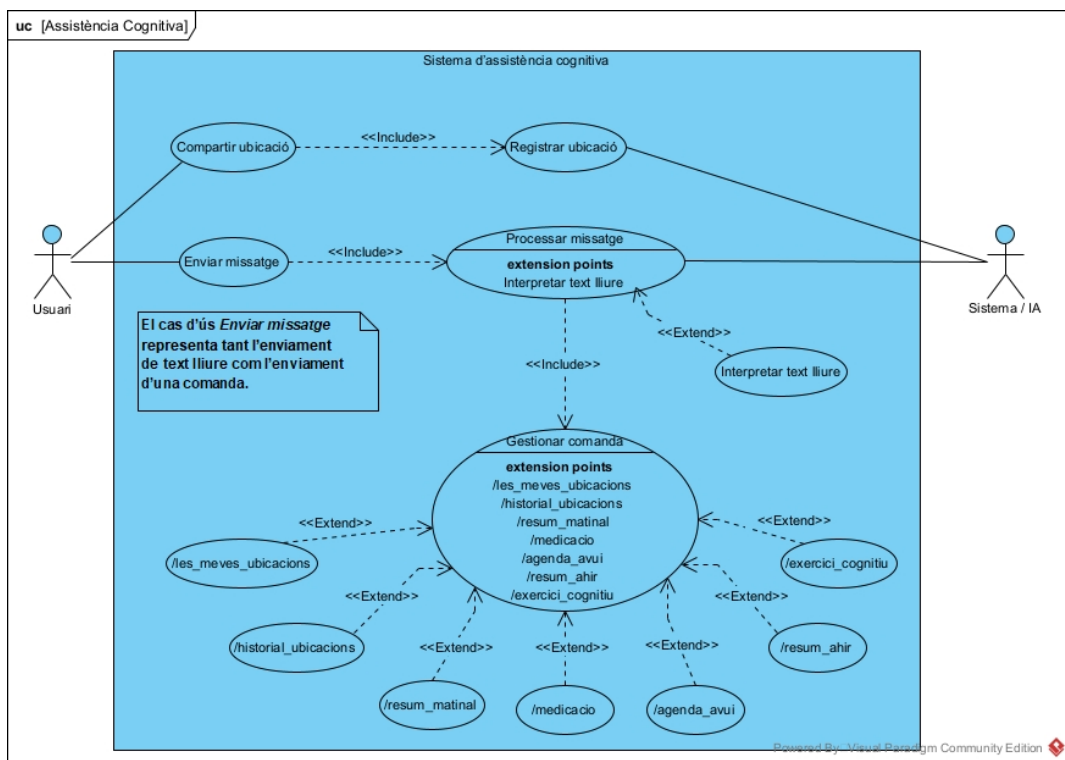


FIGURA 8.2: Diagrama de casos d'ús per a l'assistència cognitiva.

Per veure les fitxes de casos d'ús, vegeu l'Annex C.

8.1.4 Model conceptual de dades

El model de dades del sistema se centra principalment en el cas d'ús d'assistència cognitiva, ja que és l'únic que requereix persistència estructurada de dades.

El model segueix una estructura entitat-relació (ER), on l'entitat principal és *Usuari*. A partir d'aquesta entitat es relacionen la resta de dades del sistema, com l'historial d'ubicacions, missatges, medicació, agenda i exercicis cognitius.

Les relacions definides són de tipus 1:N, permetent associar múltiples registres a un mateix usuari. Aquest model conceptual serveix com a base per al disseny posterior de la persistència de dades.

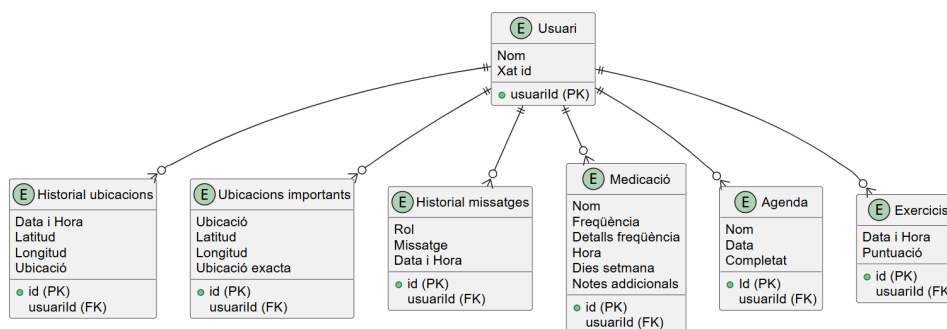


FIGURA 8.3: Diagrama Entitat-Relació que representa l'estructura de la base de dades.

8.1.5 Disseny de la base de dades

La persistència de dades del sistema es basa en un model no relacional implementat mitjançant *Airtable*.

A diferència del model conceptual presentat anteriorment, en aquest cas es defineix l'estructura real amb què s'emmagatzemen les dades i com són utilitzades pels fluxos d'automatització.

Airtable organitza la informació en forma de taules independents, cadascuna orientada a una funcionalitat concreta del sistema (ubicacions, missatges, medicació, etc.). Cada registre representa una unitat d'informació específica dins del context del sistema.

Tot i no disposar de claus foranes de manera nativa, el disseny conceptual preveu la utilització d'un camp identificador d'usuari (*usuariId*) per associar els registres amb el seu usuari corresponent.

No obstant això, com que la implementació actual del sistema s'ha desenvolupat com un prototip per a un únic usuari, en el disseny final d'*Airtable* s'ha omès aquest identificador per simplificar l'estructura. Aquesta decisió redueix la complexitat dels fluxos actuals, deixant la incorporació del camp (*usuariId*) com una millora pendent per a una futura evolució cap a un escenari multiusuari.

Cada taula conté camps específics en funció de la seva funcionalitat (per exemple, coordenades geogràfiques en el cas de les ubicacions o informació temporal en els registres d'activitat).

La descripció detallada de les taules i dels seus camps es presenta a l'Annex D.

8.1.6 Disseny de les interfícies d'usuari

El sistema no disposa d'una interfície gràfica pròpia, sinó que reutilitza plataformes externes ja consolidades.

Aquesta decisió respon a un enfocament orientat a la simplicitat i a la reutilització d'eines ja conegudes, evitant la necessitat d'aprenentatge d'una nova interfície.

En el cas de l'atenció al client, la interacció es realitza mitjançant Chatwoot, que centralitza la gestió de converses. Pel que fa a l'assistència cognitiva, la comunicació amb l'usuari es duu a terme a través de Telegram.

8.1.7 Patrons de disseny aplicats

En el desenvolupament del sistema no s'han implementat patrons de disseny clàssics de manera explícita a nivell de codi orientat a objectes, ja que la solució es basa principalment en fluxos d'automatització mitjançant n8n i en la integració de serveis externs. Malgrat això, sí que es poden identificar diversos patrons de disseny aplicats de manera implícita a nivell arquitectònic.

Patró Facade: El sistema fa ús d'un patró de tipus Facade mitjançant n8n, que actua com a capa d'abstracció entre els diferents serveis externs (Telegram, Airtable, Chatwoot, Evolution API, etc.). En lloc que cada component interactuï directament amb tots aquests serveis, n8n centralitza la lògica i ofereix una interfície simplificada basada en fluxos. Això permet reduir l'acoblament entre components i simplificar la gestió de les integracions [5].

Patró Adapter: També es pot interpretar una aplicació del patró Adapter en la manera com n8n transforma les dades entre diferents sistemes. Cada servei extern (per exemple, Telegram o Airtable) té el seu propi format de dades, i els nodes d'n8n s'encarreguen d'adaptar aquestes estructures a un format comú utilitzat dins dels fluxos. Aquesta adaptació permet que sistemes amb interfícies incompatibles puguin treballar conjuntament sense necessitat de modificar-los [6].

Patró Observer: El funcionament basat en webhooks segueix el patró Observer. En aquest cas, serveis com Telegram o Chatwoot actuen com a "subjectes" que notifiquen esdeveniments (per exemple, recepció d'un missatge), mentre que n8n actua com a observador que reacciona automàticament a aquests canvis d'estat. Aquest enfocament permet una arquitectura reactiva i desacoblada, on el sistema respon a esdeveniments en temps real [7].

Patró Command: Els fluxos d'n8n també presenten similituds amb el patró Command, ja que cada acció (consultar dades, enviar un missatge, registrar informació, etc.) es defineix com una seqüència d'operacions encapsulades. Aquestes operacions es poden executar en resposta a diferents triggers (webhooks, horaris, comandes de l'usuari), permetent desacoblar qui invoca l'acció de qui l'executa [8].

Capítol 9

Implementació i proves

El sistema ha estat desenvolupat seguint un enfocament progressiu, iniciant-se en un entorn local de proves i finalitzant amb el seu desplegament en un servidor virtual (VPS), amb l'objectiu de simular un escenari real d'utilització.

9.1 Cas d'ús 1: Atenció al client automatitzada i omnicanal

L'objectiu d'aquest cas d'ús és automatitzar l'atenció al client mitjançant canals com WhatsApp i Gmail, permetent gestionar consultes, cites i respostes automàtiques.

Un aspecte clau del disseny és la incorporació d'un mecanisme de derivació i control humà. No s'ha optat per una automatització completament autònoma, ja que aquest enfocament resultaria excessivament simple i no permetria explotar tot el potencial de la plataforma. La intervenció humana permet verificar, activar o desactivar determinades funcionalitats de l'assistent virtual, aportant més control i seguretat.

9.1.1 Arquitectura del sistema

L'arquitectura del sistema es basa en una infraestructura contenitzada mitjançant Docker, que permet integrar diferents serveis de manera modular. Per a la seva implementació, es va utilitzar com a punt de partida un repositori públic basat en Docker i Docker Compose [9], que proporcionava una configuració inicial funcional amb els següents components: Chatwoot, Evolution API, PostgreSQL, Redis, Nginx i PgAdmin.

No obstant això, per completar la funcionalitat del projecte, va ser necessari incorporar serveis addicionals que no estaven inclosos en el repositori original, com n8n i Flowise.

La coordinació entre serveis es gestiona mitjançant un fitxer *docker-compose.yaml*, encarregat de definir els contenidors, dependències i xarxa interna del sistema.

Per conèixer els passos necessaris per preparar l'entorn, vegeu l'Annex E.

El procés iteratiu de configuració, que inclou les proves inicials de recepció de dades, la validació de la connexió amb la intel·ligència artificial i la gestió de formats multimèdia (àudio i imatge), es detalla a l'Annex F.

9.1.2 Eines de Chatwoot

Per coordinar el comportament del sistema amb n8n, s'utilitzen tres funcionalitats clau de Chatwoot com a mecanisme de control:

- **Atributs personalitzats:** S'ha definit l'atribut *Assistent virtual* com una llista de valors. Aquest permet seleccionar manualment la funcionalitat a executar, garantint que la intervenció de l'IA no sigui sempre automàtica, sinó supervisada.
- **Etiquetes:** Funcionen com a indicadors d'estat de la conversa. Actuen com el nexce d'unió amb n8n, ja que el flux les utilitza com a condicions lògiques per bifurcar el camí a seguir.
- **Regles d'automatització:** Vinculen els atributs amb les etiquetes. En modificar el valor de l'atribut *Assistent virtual*, una regla interna assigna automàticament l'etiqueta corresponent.

Aquest mecanisme permet activar fluxos específics d'n8n a partir dels canvis d'estat de la conversa amb l'esdeveniment *conversation_updated*.

9.1.3 Gestió de cites

Per implementar aquesta primera funcionalitat, es van crear els valors *Agendar cita* i *Sense assignar* dins de l'atribut *Assistent virtual*, vinculats a l'etiqueta *agendar_cita* mitjançant regles d'automatització (vegeu Annex H).

Quan el treballador selecciona *Agendar cita*, el sistema assigna l'etiqueta corresponent, disparant l'esdeveniment que n8n captura. El flux d'automatització verifica la presència d'aquesta etiqueta i, en cas afirmatiu, envia al client un missatge estàtic amb un enllaç de Google Calendar per a la reserva (vegeu Annex I).

Un cop enviada la resposta, el flux realitza una segona petició HTTP per retornar l'atribut a *Sense assignar*. Aquest canvi activa la regla de neteja a Chatwoot, que elimina l'etiqueta *agendar_cita* i deixa la conversa en estat neutral, garantint que el sistema quedi preparat per a futures interaccions sense intervenció manual addicional.

Flux resultant:

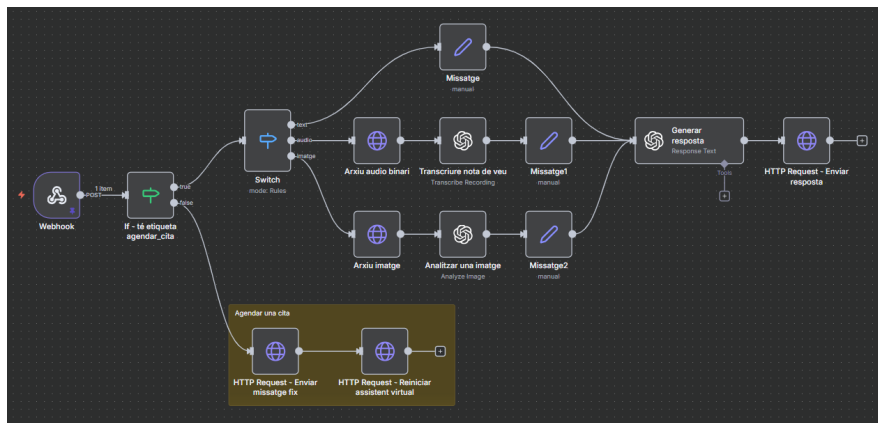


FIGURA 9.1: Flux per gestionar la funcionalitat d'agendar cites.

9.1.4 Etiqueta respon 1 vegada

Aquesta funcionalitat permet respondre puntualment un únic missatge del client. Per implementar-la, es va crear un nou valor *Activat – respon 1 missatge* per l'atribut de l'assistent i la seva etiqueta associada *assistent_activat_respon_1*. Es va afegir una nova regla d'automatització per garantir l'exclusivitat, la qual en assignar-se aquesta tasca s'elimina qualsevol altra etiqueta activa per evitar conflictes (vegeu Annex H).

El flux s'ha estructurat mitjançant una seqüència de condicions que comença amb un filtre per aturar el procés si no hi ha cap etiqueta activa, seguit d'una bifurcació lògica que deriva el missatge cap a la generació de resposta per IA o cap a l'enviament de l'enllaç de cites. Finalment, una petició HTTP retorna l'atribut de l'assistent a *Sense assignar* per deixar l'estat net un cop enviada la resposta.

Flux resultant:

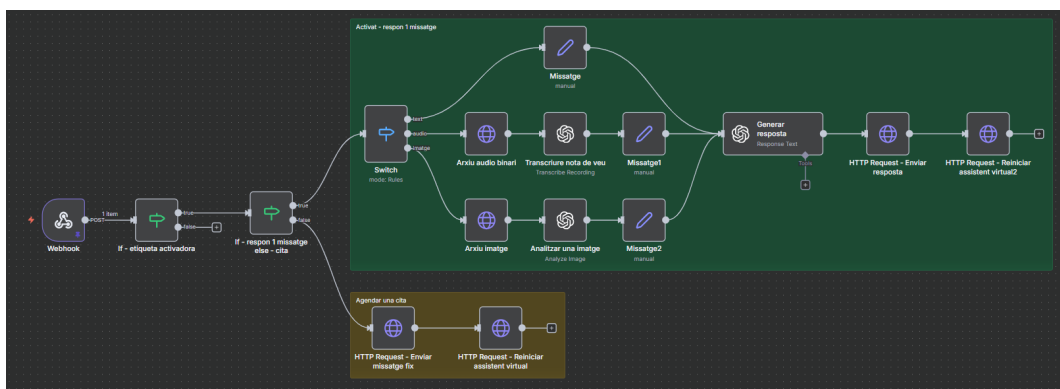


FIGURA 9.2: Flux per gestionar la resposta puntual a un missatge.

9.1.5 Funcionament del sistema fora d'hores

Fins ara, el sistema només actuava sota demanda manual mitjançant etiquetes. Per garantir l'atenció quan no hi ha operadors disponibles, s'ha implementat una lògica que automatitza les respostes fora de la jornada laboral. Per aconseguir-ho, s'han afegit dos passos clau després del webhook:

- **Càlcul temporal:** Un node de codi JavaScript processa la marca de temps (*timestamp*) del missatge rebut per determinar si es troba fora de l'horari.
- **Bifurcació del flux:** Si el missatge és fora d'hores, el flux ignora les etiquetes i genera la resposta automàtica. Durant l'horari laboral, el sistema queda supeditat a l'activació manual per part d'un operador.

Aquesta lògica assegura una resposta immediata durant la nit o els festius sense perdre el control humà durant el dia.

9.1.6 Etiqueta respon X vegades

Com a evolució de la funcionalitat *Activat - respon 1 missatge*, s'ha implementat una funcionalitat que permet definir un nombre determinat de respostes automàtiques consecutives mitjançant l'etiqueta *assistent_activat_respon_x*. El sistema utilitza dos

atributs addicionals a Chatwoot per gestionar aquest cicle: un límit de respostes configurable pel treballador i un comptador intern de les respostes enviades.

Limitació detectada i adaptació del sistema

El funcionament d'aquest cicle presenta un repte d'activació mecànica. En rebre el primer missatge, el treballador assigna l'etiqueta, fet que dispara l'esdeveniment *conversation_updated* i executa correctament la primera resposta automàtica. No obstant això, quan el client envia els missatges posteriors, el flux no es reactivaria; com que l'etiqueta ja està assignada, Chatwoot no detecta cap canvi d'estat a la conversa i no envia cap notificació a n8n.

Per resoldre aquesta limitació, s'incorpora l'esdeveniment *message_created* al webhook. El sistema està dissenyat perquè, en arribar el primer missatge, aquesta notificació s'ignori (ja que la resposta es gestiona a través de l'actualització de la conversa), però s'utilitzi de manera activa per capturar i respondre totes les rèpliques successives del client mentre el comptador resti actiu.

Adaptació als diferents formats d'esdeveniment del webhook

Aquesta ampliació resol la limitació detectada en l'apartat anterior, però introdueix una nova complexitat tècnica: l'estructura del JSON rebut varia segons l'esdeveniment. Per evitar errors d'execució, es van haver d'adaptar nodes ja configurats prèviament utilitzant l'operador d'encadenament opcional (?.), permetent extreure la informació correctament independentment del format de l'esdeveniment rebut.

```

JavaScript
1 // Missatge amb l'event message_created
2 const msg1 = $json.body.conversation?.messages?.[0].created_at;
3
4 // Missatge amb l'event conversation_updated
5 const msg2 = $json.body.messages?.[0].created_at;
6
7 // Triem el que existeixi
8 const msg = msg1 || msg2;
9
10 const hora = DateTime
11   .fromSeconds(msg)
12   .setZone('Europe/Madrid')
13   .toFormat('HH:mm');
14
15 const foraHorari = (hora > '18:00' || hora < '10:00');
16 return { foraHorari };

```

FIGURA 9.3: Exemple: Node Code adaptat pel nou esdeveniment.

Filtrat d'etiquetes activadores segons el tipus d'esdeveniment L'ampliació del webhook també va requerir ajustar la detecció d'etiquetes activadores. En cas d'un esdeveniment *conversation_updated*, el sistema inspecciona el camp *changed_attributes* per detectar si l'etiqueta s'acaba d'afegir. En canvi, si l'esdeveniment és *message_created*, el flux simplement verifica si l'etiqueta de resposta múltiple ja és present a la conversa.

Un cop el flux d'n8n confirma l'activació, s'aplica una nova jerarquia de decisions:

1. **Validació del límit:** Un node condicional compara si el comptador de respostes és inferior al límit establert. Si s'ha assolit el màxim, el flux reinicia els atributs i s'atura.

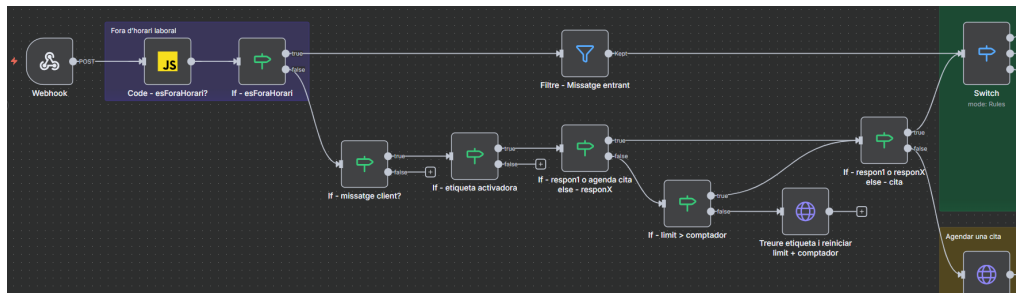


FIGURA 9.4: Nova estructura lògica per tractar diversos missatges en cadena.

2. **Execució i actualització:** Si encara no s'ha arribat al límit, l'agent d'IA genera la resposta i, un cop enviada, una petició HTTP incrementa el comptador a Chatwoot.
3. **Finalització del cicle:** Després de cada enviament, el sistema comprova si el nou valor del comptador iguala el límit. En cas afirmatiu, el sistema restableix automàticament l'estat de l'assistent a *Sense assignar* i posa el comptador a zero.

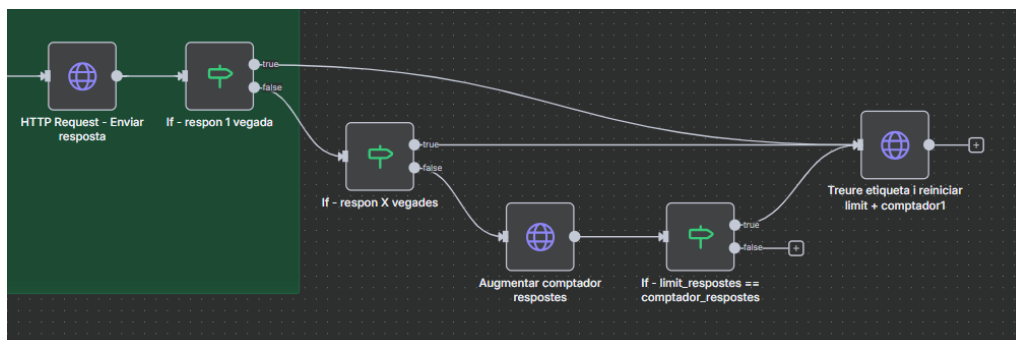


FIGURA 9.5: Actualització del flux per reiniciar l'estat de l'assistent.

Aquesta arquitectura permet delegar converses senceres a l'IA de forma controlada, assegurant que el sistema retorni la gestió a l'operador humà un cop finalitzat el cicle programat.

9.1.7 Filtratge de missatges i prevenció de bucles

L'ús de l'esdeveniment *message_created* implica que el sistema rep notifiacions de qualsevol missatge, incloent-hi els enviats pel propi sistema. Per evitar respostes duplicades o bucles infinits, s'ha implementat un node condicional que filtra l'origen de la comunicació.

Aquest mecanisme de control assegura que el flux només s'executi quan el missatge prové del client extern, descartant qualsevol resposta generada internament per la plataforma o pels treballadors.

9.1.8 Integració amb Gmail

L'ampliació del sistema cap al correu electrònic s'ha dissenyat per reutilitzar la infraestructura d'intel·ligència artificial existent, desacoblant el processament del canal de comunicació utilitzat. Aquesta integració s'ha resolt mitjançant dos mecanismes:

- **Identificació del canal:** Un node condicional detecta si l'origen del missatge prové de WhatsApp o del correu electrònic per processar les dades segons correspongui.
- **Normalització del contingut:** Atès que l'estructura de les dades d'un correu a Chatwoot difereix de la de WhatsApp, s'utilitza un node *Set* per extreure el text de l'email (ubicat generalment a *content_attributes* o dins del text citat).

Gràcies a aquesta centralització, el node d'IA processa la consulta independentment de l'origen. Aquesta arquitectura modular evita la duplicació de fluxos i facilita l'escalabilitat del sistema davant la futura incorporació de nous canals de comunicació.

9.1.9 Integració amb Flowise

S'ha substituït el node d'IA natiu d'n8n per una integració amb Flowise per implementar una arquitectura de Generació Augmentada per Recuperació (RAG). Aquest canvi garanteix que l'assistent basi les seves respostes exclusivament en la documentació oficial de l'empresa, reduint significativament el risc d'al·lucinacions.

Lògica del flux a Flowise

El flux de Flowise actua com un motor de decisió que manté la persistència de la conversa mitjançant identificadors de sessió. La seva arquitectura es basa en:

1. **Classificació d'intencions:** Un agent categoritza el missatge entrant (salutació, pregunta genèrica o agraïment) per dirigir-lo a la branca de resposta adequada.
2. **Motor RAG:** Davant consultes de negoci, el sistema recupera la informació precisa d'una base de dades vectorial (Pinecone) on resideix la documentació fragmentada (*chunks*) i vectoritzada (*embeddings*).
3. **Generació contextualitzada:** El model combina la pregunta de l'usuari amb el context extret del RAG per generar una resposta fidel a la realitat corporativa.

Per la descripció exhaustiva de la configuració dels nodes, els *prompts* utilitzats i el procés tècnic de *chunking* i *embedding*, vegeu els Annexos J i K.

Integració amb el flux d'n8n La connexió es realitza mitjançant un node HTTP Request que invoca l'API de Flowise enviant la consulta i un *sessionId*. L'estratègia per mantenir la traçabilitat varia segons el canal:

- **WhatsApp:** S'utilitza el *remoteJid*¹ del client per vincular el número de telèfon a una sessió única.

- **Gmail:** Es genera un identificador combinant el correu del remitent amb l'assumpte per simular fils de conversa coherents.

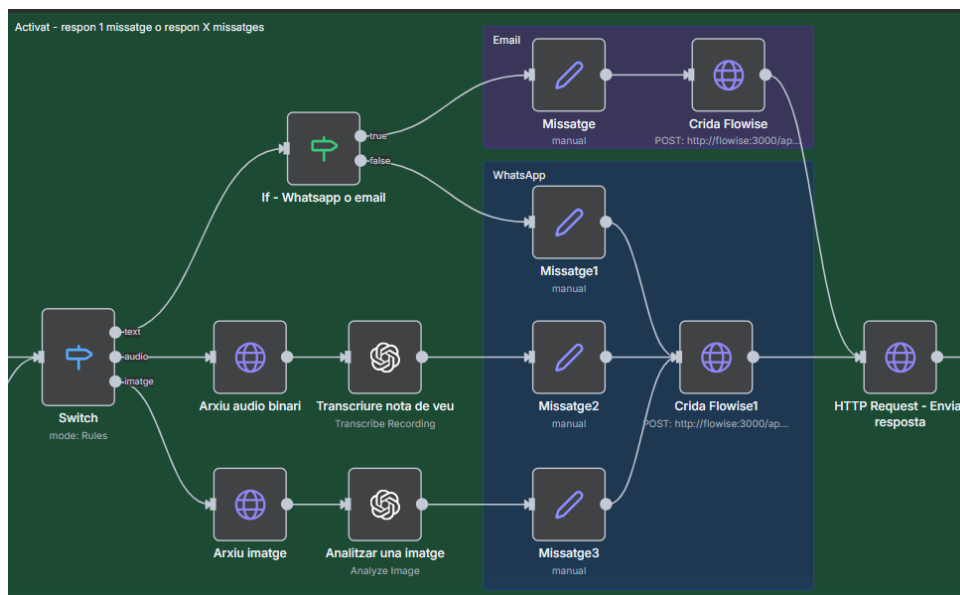


FIGURA 9.6: Integració del flux d'n8n amb el sistema de Flowise.

Per veure la descripció completa del flux resultant final del cas d'ús d'atenció al client, vegeu l'Annex L.

9.2 Cas d'ús 2: Assistència cognitiva

L'objectiu d'aquest segon cas d'ús és desenvolupar un assistent personal intel·ligent per optimitzar la informació quotidiana de l'usuari. A diferència de l'enfocament empresarial del primer cas, l'assistència cognitiva se centra en el suport personal mitjançant la integració de dades com el correu electrònic, calendaris i registres d'ubicació per oferir respostes contextualitzades.

Mitjançant aquesta automatització, es pretén reduir la càrrega cognitiva de l'usuari, centralitzant la gestió de les seves tasques i dades rellevants en una única interfície conversacional.

Per consultar el procés de creació del bot de Telegram, vegeu l'Annex M.

9.2.1 Limitacions en la configuració del webhook de Telegram

Des de l'inici de la integració del bot de Telegram amb n8n es va detectar una limitació important: Telegram exigeix que el webhook apunti a una URL pública i segura (HTTPS). En cas contrari, es produeix un error que impedeix la recepció de missatges, fet que impossibilita l'ús directe d'un entorn local basat en *localhost*.

¹Identificador únic del client a WhatsApp, utilitzat per saber de quin número prové cada missatge.

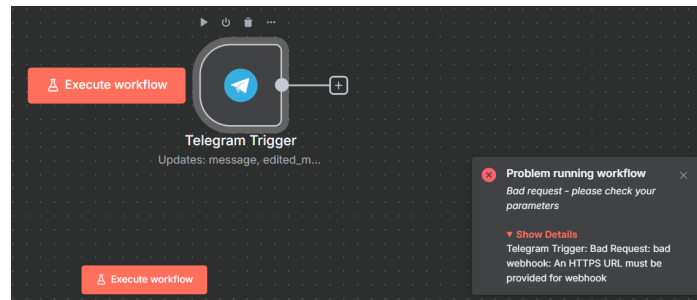


FIGURA 9.7: Error produït en la URL del webhook de Telegram.

Per resoldre-ho, es va utilitzar Ngrok per exposar el servei local a Internet. La implementació va requerir la modificació de les variables d'entorn al fitxer *docker-compose.yaml*, substituint les adreces locals per la URL pública generada:

- `WEBHOOK_URL=http://n8n:5678/`
-> `WEBHOOK_URL=https://<ngrok-url>.ngrok-free.dev`
- `N8N_EDITOR_BASE_URL=http://localhost:8083`
-> `N8N_EDITOR_BASE_URL=https://<ngrok-url>.ngrok-free.dev`

Aquesta configuració permet que Telegram validi el webhook i envii les notificacions en temps real al sistema local, fent viable la fase de proves abans del desplegament definitiu al VPS.

9.2.2 Airtable com a base de dades

Preparació de l'entorn

Per a la gestió de les dades generades pel sistema, es va optar per utilitzar Airtable com a solució de base de dades. Per habilitar la seva integració, va ser necessari crear un compte i obtenir un token d'accés, necessari per autenticar les peticions realitzades des d'n8n.

Aquest token es va configurar posteriorment dins del sistema de credencials d'n8n, permetent així la connexió entre els fluxos d'automatització i la base de dades.

Per consultar l'estructura de les dades emmagatzemades a Airtable, vegeu l'Annex D.

9.2.3 Registre inicial de l'usuari

El flux s'inicia mitjançant un webhook de Telegram configurat amb l'esdeveniment de tipus *message*, permetent rebre els missatges enviats pels usuaris.

Quan un usuari comença una conversa amb el bot mitjançant la comanda */start*, el sistema executa un node d'Airtable encarregat de crear un nou registre a la taula *Usuari*. En aquest procés s'emmagatzemen dades bàsiques com el nom d'usuari i l'identificador del xat de Telegram.

Aquest registre inicial permet identificar l'usuari dins del sistema i facilita futures interaccions i notificacions personalitzades.

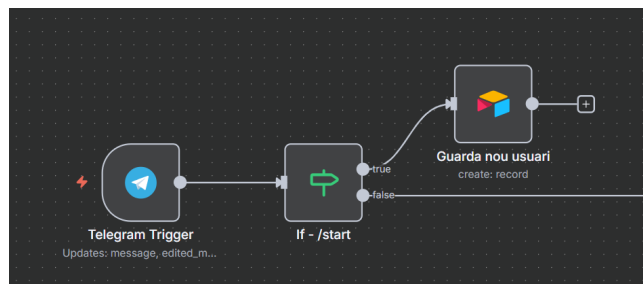


FIGURA 9.8: Flux d'informació per dur a terme el registre inicial de l'usuari.

9.2.4 Captura de la ubicació en temps real

Aquesta funcionalitat permet registrar la ubicació de l'usuari en temps real mitjançant la compartició de localització de Telegram.

Per implementar-la, el *Telegram Trigger* configurat prèviament es complementa amb un nou tipus d'esdeveniment: *edited_message*. Això permet que el sistema no només detecti l'inici de la compartició d'ubicació, sinó també les actualitzacions successives de posició enviades per Telegram.

Aquestes actualitzacions no es generen de manera estrictament periòdica, sinó únicament quan Telegram detecta un canvi rellevant en la ubicació de l'usuari. Això implica que, mentre l'usuari es desplaça, Telegram envia noves coordenades automàticament cada pocs segons.

El sistema diferencia dos tipus d'esdeveniments: *message*, que indica l'inici de la compartició de la ubicació, mentre que *edited_message* correspon a les actualitzacions successives de la mateixa ubicació que l'usuari continua compartint.

És important destacar que aquestes actualitzacions no es reben com a missatges independents, sinó com a modificacions del missatge original. Aquesta diferenciació és essencial per gestionar correctament el flux de les dades.

Amb l'objectiu d'evitar l'emmagatzematge excessiu d'informació, quan es rep un *edited_message* el sistema consulta l'últim registre guardat a la taula *Historial ubicacions* i només desa la nova posició si han passat 10 minuts des de l'últim registre.

Finalment, les coordenades rebudes es processen mitjançant un servei extern de geocodificació (Geoapify¹) per obtenir una descripció textual de la ubicació, que s'emmagatzema a la base de dades juntament amb la data i hora del registre.

9.2.5 Enregistrament de l'historial de missatges

Per garantir la traçabilitat i permetre anàlisis posteriors, el sistema emmagatzema les interaccions a la taula *Historial missatges*. Mentre que tots els missatges de l'usuari es registren automàticament, en el cas del bot només es desen aquells amb rellevància informativa, com les respostes de text lliure i els resultats de les comandes

¹Servei de geocodificació que permet tant convertir coordenades geogràfiques en adreces comprensibles (geocodificació inversa) com transformar una adreça en coordenades de latitud i longitud (geocodificació directa).

d'historial d'ubicacions, resum matinal i agenda del dia. Aquesta selecció optimitza l'emmagatzematge centrant-se en el contingut que aporta valor contextual.

9.2.6 Interpretació de text lliure

Aquesta funcionalitat permet que el sistema pugui interpretar missatges en llenguatge natural introduïts per l'usuari, sense necessitat d'utilitzar explícitament una comanda. El flux aplica una lògica de discriminació inicial:

- **Detecció de comandes:** Si el missatge comença amb el caràcter "/", s'executa directament la instrucció vinculada.
- **Processament amb IA:** En cas contrari, un node d'IA analitza la intenció de l'usuari i la classifica automàticament en una de les set comandes disponibles.

Aquest mecanisme de "traducció" millora la usabilitat fent que la interacció sigui més fluida i humana.

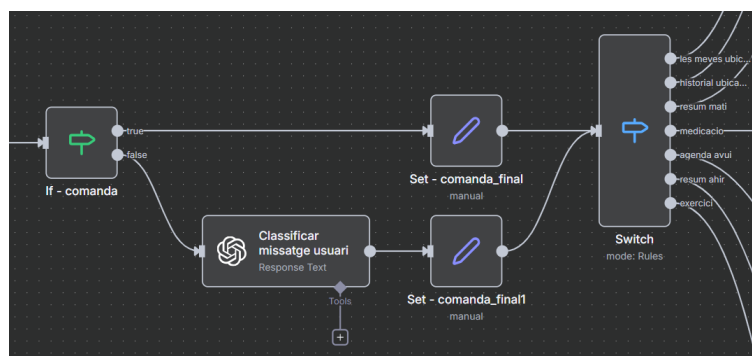


FIGURA 9.9: Processament de text lliure mitjançant IA.

9.2.7 Gestió de comandes del bot

/les_meves_ubicacions Per implementar aquesta funcionalitat, s'ha afegit al web-hook de Telegram l'esdeveniment de tipus *callback_query*, que permet gestionar les accions dels botons interactius (*inline keyboard*).

Quan l'usuari executa la comanda */les_meves_ubicacions*, el sistema mostra dues opcions:

- Afegir una nova ubicació
- Obtenir el llistat d'ubicacions importants

En seleccionar un d'aquests botons, es genera un esdeveniment de tipus *callback_query*, que inclou un identificador únic associat a cada acció (*afegir_ubi* o *llistat_ubis*).

Afegir una nova ubicació El sistema inicia un petit procés formulari on es demana el nom i l'adreça de la ubicació a enregistrar. Posteriorment, mitjançant Geoapify, l'adreça es transforma en coordenades geogràfiques i la informació s'emmagatzema a la base de dades. Finalment, s'envia un missatge de confirmació a l'usuari.

Consulta del llistat d'ubicacions El flux recupera les ubicacions d'interès guardades en la taula *Ubicacions importants* i l'última ubicació coneguda de l'usuari. Per a cadascuna d'aquestes ubicacions, s'inclou el seu nom i un enllaç de Google Maps amb indicacions des de la posició actual de l'usuari (última ubicació guardada en la taula *Historial ubicacions*) fins al destí.

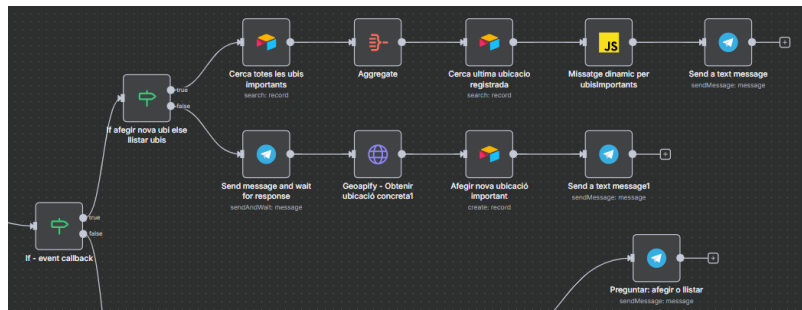


FIGURA 9.10: Flux per gestionar les ubicacions d'interès de l'usuari.

/historial_ubicacions Quan l'usuari executa aquesta comanda, el sistema consulta les ubicacions registrades durant les últimes 24 hores a la taula *Historial ubicacions*. Posteriorment, els resultats s'agrupen i s'ordenen cronològicament per construir una línia temporal dels desplaçaments de l'usuari.

A continuació, aquesta informació es combina amb les ubicacions importants definides per l'usuari. Aquest encreuament de dades és fonamental per contextualitzar el recorregut: en lloc de mostrar només una llista de carrers de la taula d'historial, el sistema permet que el model d'intel·ligència artificial associi aquestes adreces amb els noms familiars registrats per l'usuari (com ara "casa", "feina" o "escola"). A partir d'aquesta relació i mitjançant un *prompt* específic, es genera un resum textual de l'activitat del dia amb un llenguatge natural i fàcilment interpretable.

Finalment, el sistema envia el resum a Telegram juntament amb un mapa del recorregut de les darreres 24 hores, generat a partir de les coordenades registrades.

/resum_matinal Aquesta funcionalitat genera un resum global de l'activitat recent de l'usuari combinant informació de diferents fonts.

En primer lloc, el sistema consulta els correus electrònics enviats i rebuts durant les últimes 24 hores mitjançant Gmail. Les dades obtingudes s'agrupen i s'envien a un node d'OpenAI, que genera un resum estructurat en llenguatge natural, destacant els correus més rellevants i filtrant contingut poc important o publicitari.

Posteriorment, el sistema recupera els missatges intercanviats amb el bot de Telegram durant el mateix període. Igual que en el cas anterior, els registres es processen amb OpenAI per generar un resum interpretatiu de l'activitat i les converses mantingudes amb l'assistent.

Finalment, el sistema envia a l'usuari un resum consolidat amb la informació més rellevant de les darreres 24 hores.

/medicacio Aquesta funcionalitat permet consultar la medicació que l'usuari té configurada, així com identificar quines preses resten pendents durant el dia actual.

En primer lloc, el sistema consulta la taula *Medicació* de la base de dades i processa els registres mitjançant nodes de tipus *Code*, generant un resum amb el nom dels medicaments, l'hora i la descripció associada a cada presa.

Posteriorment, el flux determina quins medicaments encara no s'han pres segons l'hora actual i la freqüència configurada, i calcula la pròxima presa pendent per a cada medicament.

/agenda_avui Aquesta funcionalitat recupera els esdeveniments programats a Google Calendar per al dia actual. En primer lloc, el sistema realitza una consulta a l'API de Google Calendar per sincronitzar i actualitzar les tasques planificades a la taula *Agenda* d'Airtable, filtrant posteriorment aquelles que resten no completades per enviar-les a l'usuari.

A continuació, s'emet un segon missatge amb botons interactius (*inline keyboard*) que mostren les tasques pendents. Quan l'usuari clica una opció, es genera un esdeveniment de tipus *callback_query* que registra automàticament la tasca com completada.

/resum_ahir Aquesta funcionalitat avalua el grau de compliment de les tasques planificades el dia anterior. El sistema consulta la taula *Agenda* per recuperar només els esdeveniments del dia passat i genera un llistat concís amb el nom, l'horari i l'estat de completitud de cada activitat per facilitar-ne el seguiment.

/exercici_cognitiu Aquesta funcionalitat implementa un sistema d'estimulació cognitiva basat en preguntes de cultura general de tipus veritable o fals.

El sistema obté les preguntes des d'una API externa i les presenta a l'usuari mitjançant un formulari interactiu de Telegram. Un cop completat el qüestionari, les respostes es processen i es comparen amb les respostes correctes per calcular la puntuació obtinguda.

Posteriorment, els resultats s'emmagatzemen a la base de dades juntament amb la data i hora de realització. Finalment, el sistema envia la puntuació a l'usuari i mostra un historial amb els exercicis realitzats anteriorment, permetent fer un seguiment de l'evolució al llarg del temps.

Per veure la descripció completa del flux resultant final del cas d'ús d'assistència cognitiva, vegeu l'Annex N.

Per veure la implementació del segon flux, encarregat de la neteja periòdica de dades a Airtable per optimitzar el rendiment del sistema, vegeu l'Annex O.

Per veure la implementació del tercer flux, que gestiona l'enviament automàtic del resum matinal i els recordatoris de medicació de l'usuari, vegeu l'Annex P.

9.2.8 Vídeo amb el codi desenvolupat

En el següent enllaç es pot visualitzar un vídeo amb el codi desenvolupat per dur a terme les automatitzacions: <https://youtu.be/Lt2XhQCMI-A>

Capítol 10

Implantació i resultats

En aquest capítol es descriu el procés seguit per desplegar el sistema en un entorn de producció, així com els resultats obtinguts durant la validació dels dos casos d'ús plantejats.

10.1 Procés de desplegament

Per traslladar el projecte de l'entorn local al núvol, s'ha seguit una estratègia de contenització que garanteix la coherència entre entorns i la seguretat de les dades.

10.1.1 Infraestructura i creació de la màquina virtual

S'ha utilitzat la plataforma Microsoft Azure per desplegar una màquina virtual (Ubuntu Server, 4 GiB de RAM). L'accés al servidor està restringit exclusivament al protocol SSH (*Secure Shell*¹) amb autenticació per clau pública, eliminant la vulnerabilitat de les contrasenyes convencionals.

10.1.2 Configuració de xarxa i dominis

S'ha registrat el domini *automatitzacions.dev* per exposar els serveis de l'arquitectura de forma individualitzada mitjançant subdominis:

- **n8n.automatitzacions.dev**: Orquestració de fluxos.
- **chatwoot.automatitzacions.dev**: Gestió omnicanal de clients.
- **evolution.automatitzacions.dev**: API de missatgeria WhatsApp.
- **flowise.automatitzacions.dev**: Motor d'IA i RAG.

10.1.3 Preparació del repositori i seguretat de dades

Abans de procedir al desplegament, s'ha dut a terme una tasca de depuració dels fitxers per simular un entorn de producció professional. S'ha creat un repositori privat a GitHub on només s'han inclòs els fitxers essencials per a l'arquitectura, evitant la pujada de dades sensibles o temporals:

¹Protocol de xarxa que permet accedir i gestionar de manera segura un servidor remot mitjançant connexions xifrades.

- **Exclusions:** S'han exclòs les carpetes de bases de dades locals (*database/*), configuracions internes d'instàncies i sessions temporals de WhatsApp (*evolution_instances*), ja que aquests volums s'han de generar de nou en l'entorn de destí per evitar conflictes de compatibilitat entre sistemes operatius.
- **Gestió de variables d'entorn:** S'han transformat els fitxers *.env* (que contenen claus API i contrasenyes) en fitxers de plantilla *.env.example*. Aquests fitxers serveixen de guia per a la configuració manual al servidor, assegurant que cap dada crítica quedi exposada.

10.1.4 Preparació del servidor i clonatge del repositori

Després d'establir la connexió SSH, s'ha preparat el servidor mitjançant la instal·lació de Docker i les seves dependències bàsiques per permetre l'execució de l'arquitectura contenitzada. Per veure les comandes exactes de la preparació de l'entorn del servidor, vegeu l'Annex Q.

Per veure el procediment de clonatge del repositori i la configuració de les variables d'entorn al servidor, vegeu l'Annex R.

10.1.5 Configuració de *docker-compose* i variables d'entorn

Els detalls específics sobre la configuració dels fitxers *.env* i *docker-compose.yaml*, així com l'anàlisi de les diferències respecte a l'entorn de desenvolupament local, es poden consultar a l'Annex S.

10.1.6 Gestió de xarxa: De Nginx a Traefik

Un dels canvis d'arquitectura més rellevants en el pas a producció ha estat la substitució del servidor Nginx per Traefik. Mentre que en l'entorn de desenvolupament local Nginx actuava com un servidor intermediari configurat manualment per redirigir ports, en un entorn real aquesta solució presentava limitacions d'escalabilitat i seguretat.

A diferència de Nginx, que requeriria la instal·lació i configuració manual de Certbot o *scripts* externs, Traefik integra de forma nativa el protocol ACME. Això permet que el sistema sol·liciti, configuri i renovi automàticament els certificats TLS gratuïts de *Let's Encrypt* per a tots els subdominis del projecte.

A més, Traefik no requereix cap fitxer de configuració estàtic (*nginx.conf*) que s'hagi de reiniciar cada cop que s'afegeix un servei. En lloc d'això, "escolta" el *socket* de Docker i detecta els nous contenidors mitjançant etiquetes (*labels*). Això simplifica significativament el manteniment i l'escalabilitat de l'arquitectura.

10.1.7 Regles d'accés i seguretat de xarxa

Perquè Traefik pugui gestionar les peticions i certificar els dominis, cal configurar les regles d'accés a la infraestructura de Microsoft Azure. Tot i que els contenidors estan configurats internament, el *Network Security Group* (NSG) d'Azure actua com un tallafoc que bloqueja tot el trànsit per defecte.

S'ha procedit a l'obertura de dos ports específics mitjançant regles d'entrada (*Inbound Port Rules*):

- **HTTP (Port 80):** Aquest port és essencial per a la validació del protocol ACME per part de *Let's Encrypt*. Permet que l'entitat certificadora verifiqui la propietat del domini i facilita la redirecció automàtica del trànsit cap a la connexió segura.
- **HTTPS (Port 443):** És el port principal per on circularà tot el trànsit xifrat del sistema. A través d'aquest port es garanteix la privadesa de les dades en els serveis de Chatwoot, n8n, Flowise i Evolution API.

10.1.8 Preparació de certificats i desplegament final

Abans d'aixecar l'arquitectura, s'ha creat el fitxer *acme.json* amb permisos de lectura i escriptura exclusius per a l'usuari (`chmod 600`), garantint així l'emmagatzematge segur dels certificats SSL per part de Traefik.

```
mkdir -p traefik/letsencrypt & touch traefik/letsencrypt/acme.json
```

```
chmod 600 traefik/letsencrypt/acme.json
```

Un cop validada la configuració del fitxer d'entorn i del desplegament contenitzat, s'ha executat l'arquitectura en segon pla, verificant-ne el correcte funcionament de tots els serveis mitjançant la llista de contenidors actius.

```
docker compose up -d & docker ps
```

10.1.9 Restauració de la configuració del sistema

Atès que les bases de dades locals no es van incloure en el repositori per raons d'eficiència i seguretat, un cop operatius els serveis al núvol, s'ha procedit a la restauració de la lògica del negoci:

- **Serveis de dades (Chatwoot i Evolution API):** S'han replicat les configuracions manuals realitzades inicialment en l'entorn local (creació d'agents, safates d'entrada i connexió d'instàncies de WhatsApp).
- **Fluxos d'automatització (n8n i Flowise):** S'ha realitzat una migració directa mitjançant la importació dels fitxers JSON exportats prèviament de l'entorn de desenvolupament. Aquest procediment garanteix que tota la intel·ligència de negoci i les integracions dissenyades es mantinguin intactes en el nou entorn.

10.1.10 Interfície de benvinguda (*Landing page*)

Per facilitar l'accés als diferents serveis, s'ha implementat una pàgina web estàtica que serveix com a punt d'entrada en <https://automatitzacions.dev>, desenvolupada amb HTML, CSS i JavaScript. El seu objectiu és oferir una manera senzilla d'accedir als serveis desplegats (n8n, Chatwoot, etc.). Es tracta d'una pàgina auxiliar generada íntegrament amb IA que no forma part de la implementació tècnica del TFG.

10.2 Resultats

Un cop finalitzat el desplegament, s'ha verificat que tots els subdominis configurats són accessibles i operatius sota el protocol HTTPS. La solució és plenament funcional

i respon de manera integrada a les peticions dels usuaris.

A continuació, per a la validació pràctica dels dos casos d'ús, s'ha enregistrat un vídeo demostratiu on es mostra el funcionament complet del sistema:

https://www.youtube.com/watch?v=FgOgm_jKWO8

A continuació, es descriuen els resultats mostrats al vídeo, simulant interaccions reals per validar la lògica de negoci implementada.

10.2.1 Cas d'ús 1 - Atenció al client automatitzada

En aquest escenari, el sistema actua com un agent virtual intel·ligent per a una per-ruqueria, gestionant consultes, cites i resolució de dubtes freqüents de manera omnicanal.

Per validar el correcte funcionament de la solució, s'han realitzat diferents proves d'execució simulant interaccions reals amb clients:

1. **Gestió de cites:** La primera prova s'ha orientat a validar el procés de gestió de cites a través del canal de WhatsApp. En rebre el missatge del client ("*Hola, bones! Voldria demanar una cita...*"), s'ha procedit a l'assignació manual de la tasca *Agendar cita* a l'agent a través de la interfície de Chatwoot.

A nivell tècnic, s'ha verificat que aquesta acció activa correctament el flux corresponent, seguint la branca prevista per enviar enllaços de reserva. També s'ha comprovat que no es crea cap sessió a Flowise, ja que aquest tipus d'interacció no requereix intervenció del model d'intel·ligència artificial.

2. **Resolució de consultes puntuals:** A continuació, s'ha validat la funcionalitat *Respondre a 1 missatge* davant una consulta puntual. Un cop assignada la tasca a l'assistent virtual, el sistema ha executat el flux de manera satisfactòria, connectant amb el model de llenguatge allotjat a Flowise.

S'ha comprovat que el sistema ha creat correctament una nova sessió a Flowise, utilitzant el número de telèfon del client com a identificador únic (*session_id*). La resposta generada per l'IA s'ha enviat al client de forma integrada a través de WhatsApp, validant la correcta recuperació d'informació del motor RAG.

3. **Resolució de consultes múltiples:** Una de les proves més rellevants ha consistit a validar el comportament del sistema davant d'una seqüència de missatges, incloent-hi diferents formats d'entrada. S'ha seguit el següent procediment:
 - **Recepció d'àudio i gestió de variables:** Per posar a prova el processament de missatges no textuais, s'ha enviat una nota de veu. Un cop rebut, s'ha establert manualment un límit de 4 respostes a la variable de control del client. En assignar la tasca "Respondre x missatges", el sistema ha generat la resposta incrementant el comptador intern a 1. S'ha verificat que el flux ha processat l'àudio correctament mitjançant els nodes d'OpenAI.
 - **Automatització de les respostes:** En els missatges successius (2n i 3r), l'assistent ha detectat l'etiqueta activa de resposta múltiple i ha intervingut automàticament sense requerir acció manual, actualitzant el comptador en cada iteració.

- **Reinici del comptador:** Finalment, en arribar al 4t missatge, el flux ha seguit la branca de finalització, reiniciant automàticament tant el comptador com el límit de respostes a valor 0, deixant el sistema en l'estat inicial.
4. **Canal de Gmail i control d'horari laboral:** En darrer lloc, s'ha posat a prova la integració amb el canal de correu electrònic (Gmail) i la lògica de control temporal. S'ha simulat la recepció d'un correu durant un període definit com a "fora de l'horari laboral".

En aquest cas, s'ha verificat que el node de control de temps d'n8n ha desviat l'execució cap a la branca de resposta automàtica d'absència. L'assistent ha actuat de forma totalment autònoma, sense necessitat d'assignació manual de tasques, enviant una resposta al correu del client.

Per consultar el detall complet dels missatges intercanviats, podeu dirigir-vos a l'Annex T.

10.2.2 Cas d'ús 2 - Assistència cognitiva

En aquest segon escenari, es valida el funcionament d'un assistent personal basat en Telegram dissenyat per a l'ajuda cognitiva, actuant com un gestor proactiu de la informació de l'usuari. Les proves s'han centrat en els següents punts clau:

1. **Registre de l'usuari:** Per validar l'entrada de nous usuaris al sistema, s'ha iniciat la interacció amb el bot mitjançant la comanda `/start`.

S'ha verificat que el flux d'n8n captura correctament les metadades de Telegram i realitza una escriptura a la base de dades d'Airtable, desant de forma automatitzada el nom d'usuari i el seu `xat id` corresponent.

2. **Gestió intel·ligent de la ubicació:** S'ha posat a prova la funcionalitat de seguiment mitjançant la compartició de la ubicació en temps real.

S'ha validat la recepció constant de notificacions via webhook a mesura que l'usuari es desplaçava. Per optimitzar l'ús de recursos i evitar redundàncies a la base de dades, s'ha validat la lògica de control temporal: el sistema rep totes les actualitzacions, però només executa l'escriptura a Airtable si han transcorregut, com a mínim, 10 minuts des de l'últim registre. El flux ha demostrat gestionar aquest filtre de manera precisa, assegurant un historial de moviments net i eficient.

3. **Processament del llenguatge natural:** S'ha testejat la capacitat del sistema per interpretar la intencionalitat de l'usuari sense necessitat d'utilitzar sintaxi rígida.

Mitjançant un node d'IA configurat per a la classificació d'intencions, s'ha verificat que les entrades en text lliure es transformen correctament en comandes operatives.

4. **Integració amb serveis externs:** S'ha corroborat que el sistema s'integra correctament amb les eines de Google per oferir a l'usuari un resum del seu dia.

Ha quedat demostrat que les crides a les API de Gmail i Google Calendar retornen la informació pertinent de les últimes 24 hores (correus rebuts i esdeveniments programats). El sistema processa aquestes dades i les presenta a l'usuari de manera simplificada, complint així amb l'objectiu d'assistència cognitiva i reducció de la càrrega informativa.

Per consultar el detall complet dels missatges intercanviats, vegeu l'Annex U.

10.3 Objectius i requisits assolits

Un cop finalitzada la fase d'implantació en l'entorn de producció i verificat el comportament del sistema, es procedeix a analitzar el grau de compliment del projecte. Aquesta avaluació permet confirmar que la solució és plenament operativa.

10.3.1 Avaluació dels objectius

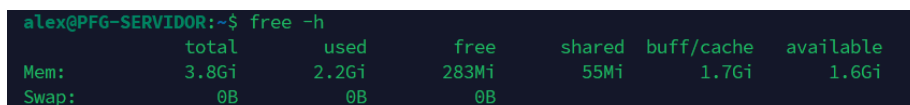
El projecte ha assolit els objectius definits al Capítol 1. L'arquitectura basada en microserveis ha permès construir un sistema modular i escalable, integrant eines d'automatització, serveis externs i models d'intel·ligència artificial dins d'un únic entorn.

Els fluxos desenvolupats permeten automatitzar tasques de manera coordinada i interpretar instruccions en llenguatge natural. A més, la validació dels dos casos d'ús ha demostrat que el sistema funciona correctament en un entorn real desplegat al núvol amb Azure.

10.3.2 Compliment dels requisits

Tots els requisits funcionals s'han implementat i validat amb èxit. La gestió omnicanal, l'automatització de respostes i el seguiment de dades com la ubicació o el calendari funcionen correctament, integrant-se de manera coherent en la lògica de negoci.

Pel que fa als requisits no funcionals, la seguretat ha quedat garantida mitjançant el xifratge TLS de Traefik i l'accés restringit per SSH. El rendiment a la màquina virtual d'Azure és estable, amb un ús de memòria RAM d'aproximadament el 58% (2,2 GiB utilitzats d'un total de 3,8 GiB) i temps de resposta d'entre 2 i 5 segons.



```
alex@PFG-SERVIDOR:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           3.8Gi         2.2Gi         283Mi         55Mi         1.7Gi         1.6Gi
Swap:          0B           0B           0B
```

FIGURA 10.1: Ús de recursos de la MV amb els serveis aixecats.

La portabilitat i la mantenibilitat s'han demostrat durant la migració de l'entorn local al núvol. L'estructura modular permet incorporar nous canals o serveis amb una configuració mínima.

Capítol 11

Conclusions

Aquest Treball de Fi de Grau ha permès demostrar el potencial de les automatitzacions intel·ligents basades en intel·ligència artificial aplicades tant a entorns empresarials com a situacions de la vida quotidiana. Mitjançant el desenvolupament de dos casos d'ús diferenciats, s'ha pogut validar que la combinació d'eines d'automatització, integració de serveis i models de llenguatge avançats permet construir sistemes funcionals, flexibles i capaços de donar resposta a necessitats reals.

Quant a l'assoliment dels objectius del PFG, s'ha complert amb la finalitat acadèmica de dissenyar una arquitectura modular, escalable i desplegada en un entorn de producció real (Azure). En l'àmbit tècnic, s'han satisfet tots els requisits funcionals definits: la gestió multicanal a través de Chatwoot, la integració del sistema RAG per a respostes contextualitzades i la implementació de funcionalitats d'assistència cognitiva mitjançant Telegram i Airtable. A diferència d'altres solucions comercials tancades, aquesta arquitectura basada en microserveis ofereix una flexibilitat total i un control absolut sobre el flux de les dades, factor crític en entorns que gestionen informació sensible.

Pel que fa als resultats obtinguts, el primer cas d'ús ha demostrat la capacitat del sistema per automatitzar processos d'atenció al client de manera eficient, integrant múltiples canals de comunicació i generant respostes contextualitzades mitjançant IA. La incorporació del sistema RAG ha permès limitar les respostes al context disponible, millorant la coherència i reduint respostes incorrectes o fora de context. D'altra banda, el segon cas d'ús ha evidenciat que aquest tipus de tecnologies també poden tenir aplicacions amb impacte social i assistencial, oferint suport cognitiu mitjançant recordatoris, seguiment d'activitats i interpretació de llenguatge natural.

En comparació amb altres solucions existents, moltes plataformes comercials ofereixen automatitzacions avançades, però acostumen a estar orientades a entorns molt específics o depenen de serveis tancats amb poca flexibilitat de personalització. En canvi, el sistema desenvolupat aposta per una arquitectura modular basada en serveis desacoblats i tecnologies contenitzades, fet que facilita l'escalabilitat, la reutilització de components i l'adaptació a diferents escenaris. A més, l'ús d'eines *low-code* com n8n ha permès accelerar considerablement el desenvolupament sense renunciar a la personalització tècnica mitjançant codi quan ha estat necessari.

Cal destacar, però, que aquesta flexibilitat també comporta una major complexitat de configuració i manteniment en comparació amb plataformes comercials totalment gestionades. Solucions propietàries com Zapier, Make o plataformes d'assistents

virtuals integrades ofereixen una posada en marxa més immediata i simplifiquen molts aspectes d'infraestructura, però sovint limiten la personalització, l'accés intern als fluxos i la capacitat d'adaptar el sistema a necessitats molt específiques. En aquest projecte, en canvi, s'ha prioritzat un enfocament orientat al control tècnic, la modularitat i la independència respecte a proveïdors externs, assumint al mateix temps la complexitat addicional que implica gestionar directament la infraestructura i les integracions.

Durant el desenvolupament del projecte també s'han produït diverses desviacions respecte a la planificació inicial. Tot i que les fases generals s'han mantingut, diverses tasques han requerit més temps del previst a causa de limitacions tècniques no detectades inicialment, especialment en la integració entre serveis externs, la configuració de webhooks, la gestió dels diferents esdeveniments generats per les APIs i el desplegament en producció. Aquestes dificultats han obligat a replantejar determinades decisions tècniques i a adaptar progressivament l'arquitectura del sistema. Tot i això, l'enfocament iteratiu adoptat ha permès absorbir aquests canvis de manera natural i continuar evolucionant el projecte sense comprometre els objectius finals.

Pel que fa a les aportacions personals, aquest projecte ha suposat una immersió pràctica en tecnologies actuals relacionades amb automatització, intel·ligència artificial, desplegament de serveis i integració de plataformes. Més enllà de la implementació tècnica, una de les aportacions principals ha estat la capacitat de construir un sistema complet des de zero, passant per totes les fases del desenvolupament: anàlisi, disseny, implementació, validació i desplegament real. També ha representat un exercici important de resolució de problemes, investigació autònoma i presa de decisions tècniques davant incidències reals.

Finalment, el principal aprenentatge extret d'aquest projecte no ha estat únicament el coneixement tècnic adquirit sobre eines com n8n, Docker, Flowise, Airtable o serveis basats en IA, sinó també l'experiència d'haver treballat durant mesos en un projecte de gran dimensió, evolucionant progressivament des d'un entorn local fins a un desplegament funcional en producció mitjançant màquines virtuals, dominis propis i serveis accessibles des d'internet. Aquest procés ha permès entendre de manera molt més pràctica la importància de l'arquitectura, la seguretat, la mantenibilitat i la capacitat d'adaptació dels sistemes reals.

En conjunt, el projecte no només ha servit per aprofundir en coneixements tècnics, sinó també per consolidar una metodologia de treball basada en la constància, l'experimentació i l'aprenentatge continu, aspectes fonamentals en el desenvolupament de sistemes tecnològics reals i escalables.

Capítol 12

Treball futur

Tot i que el sistema desenvolupat compleix els objectius plantejats inicialment i demostra la viabilitat de la solució proposada, existeixen diverses línies de millora i ampliació que podrien desenvolupar-se en futurs treballs.

En primer lloc, pel que fa al cas d'ús d'atenció al client, una possible evolució consistiria a ampliar els canals de comunicació suportats. A més de WhatsApp i correu electrònic, es podria integrar el sistema amb altres plataformes habituals en l'àmbit empresarial, com Telegram, Instagram o Facebook Messenger.

També seria interessant incorporar la gestió automatitzada de trucades telefòniques mitjançant sistemes de veu interactiva. D'aquesta manera, quan un client truqués a l'empresa, un assistent virtual podria atendre la trucada, identificar la seva necessitat i gestionar accions senzilles, com ara reservar una cita, consultar horaris, confirmar visites o redirigir la trucada al departament corresponent.

Pel que fa al segon cas d'ús, orientat a l'assistència cognitiva, el sistema podria evolucionar cap a una plataforma més completa de suport personal. Entre les possibles millores destaca la incorporació de reconeixement i síntesi de veu, permetent que l'usuari interactuï verbalment amb l'assistent sense necessitat d'escriure missatges.

També es podria ampliar la part de seguiment cognitiu mitjançant nous exercicis adaptatius, ajustats automàticament al nivell i evolució de l'usuari. Igualment, es podria incorporar un sistema d'estadístiques per analitzar tendències en la memòria, compliment de tasques, rutines diàries o evolució general de l'usuari al llarg del temps.

Una altra millora de gran utilitat seria la integració amb dispositius intel·ligents o *wearables*, com rellotges intel·ligents, polseres d'activitat o sensors domèstics. Això permetria detectar inactivitat prolongada, possibles caigudes, alteracions de rutines o enviar avisos automàtics a familiars o cuidadors.

Des del punt de vista tècnic, una futura evolució del sistema seria transformar l'arquitectura actual en una plataforma multiusuari completa, amb autenticació, gestió de perfils independents i separació total de dades entre usuaris. Aquesta ampliació facilitaria la seva adopció real per part d'empreses o entorns assistencials.

Finalment, també es podria plantejar la incorporació de nous casos d'ús aprofitant la mateixa base tecnològica desenvolupada en aquest projecte.

Apèndix A

Ètica de la IA i responsabilitat en el tractament de dades

La intel·ligència artificial actua com el motor cognitiu del sistema desenvolupat, encarregant-se de la interpretació de missatges i la generació de respostes naturalitzades. A causa de la rellevància d'aquest component, aquest annex amplia l'anàlisi dels principis ètics que s'haurien d'aplicar en un escenari de desplegament comercial real per garantir un disseny responsable i alineat amb les bones pràctiques.

Confidencialitat i protecció de les dades El sistema envia el text a serveis externs d'intel·ligència artificial mitjançant connexions segures de tipus API. Com que les dades enviades poden incloure informació altament sensible (com ara historials de converses, correus electrònics, esdeveniments programats en el calendari o dades sobre medicació i salut), s'han definit diverses mesures de protecció arquitectònica:

- **Absència d'entrenament extern:** S'utilitzen canals d'API empresarials on els proveïdors (com OpenAI en el cas d'aquest projecte) garanteixen contractualment que les dades enviades no es faran servir per entrenar futurs models públics.
- **Desacoblament de les dades persistents:** La lògica de l'orquestrador (n8n) segmenta la informació. D'aquesta manera, el model d'IA només rep el contingut estrictament necessari per executar la tasca actual, evitant bolcar tota la base de dades d'Airtable en una sola petició.
- **Filtratge de contingut sensible:** El sistema pot incorporar mecanismes de detecció de dades especialment sensibles, com contrasenyes o informació mèdica crítica, amb l'objectiu d'advertir l'usuari abans que aquestes dades siguin processades per la IA.

Mitigació d'al·lucinacions dels models generatius Un dels principals riscos ètics de l'ús de models generatius és l'aparició d'al·lucinacions (és a dir, la generació d'informació incorrecta o inventada amb aparença de credibilitat). En un entorn d'automatització de processos, una al·lucinació pot derivar en accions inadequades (com programar una cita en una data incorrecta o proporcionar informació inexacta als usuaris).

Per mitigar aquest risc, s'han adoptat les següents estratègies:

- **Restricció mitjançant RAG:** S'elimina la llibertat creativa del model forçant-lo, mitjançant *system prompts*¹, a respondre exclusivament a partir dels vectors de coneixement recuperats del sistema. En cas que la informació no estigui disponible, el model està configurat per indicar explícitament que no disposa de la resposta.
- **Parametrització de la temperatura:** Es configura la "temperatura"² del model a valors propers a zero (0.1 - 0.2) per prioritzar respostes més deterministes, coherents i precises, reduint així la creativitat lingüística innecessària.

Transparència, biaixos i supervisió humana L'ús responsable de la intel·ligència artificial requereix garantir la transparència en la interacció amb l'usuari.

- **Principi de transparència:** L'usuari ha de ser informat en tot moment que està interactuant amb un sistema assistit per intel·ligència artificial. Per aquest motiu, qualsevol interacció generada per l'assistent inclou una marca textual clara que indica que la resposta ha estat assistida per IA.
- **Control del biaix algorítmic:** Els models de llenguatge basats en IA estan entrenats amb grans conjunts de dades provinents de múltiples contextos socials i culturals, fet que pot introduir biaixos en les respostes generades. Per minimitzar aquest risc, es preveu la realització d'auditories periòdiques sobre les converses, amb l'objectiu de supervisar el comportament del sistema i garantir criteris de neutralitat, equitat i coherència en les respostes.
- **Supervisió i intervenció humana:** En situacions crítiques o quan la IA no sigui capaç de proporcionar una resposta fiable, el sistema permet la intervenció manual per part del personal responsable de l'empresa. D'aquesta manera, un operador humà pot revisar la consulta i oferir una resposta adequada sense delegar responsabilitats a la intel·ligència artificial.

¹Instrucció de configuració inicial i prioritària que s'envia al model d'IA per definir el seu rol, comportament i limitacions, servint precisament per obligar-lo a respondre de manera lògica i determinada.

²Paràmetre de configuració que controla el grau d'aleatorietat de la intel·ligència artificial, entre més alt, més creativa serà la resposta, entre més baix, més precisa i determinista serà la resposta.

Apèndix B

Integració de serveis externs amb n8n

B.1 OpenAI

Per utilitzar serveis d'intel·ligència artificial dins del sistema, és necessari disposar de credencials d'accés als proveïdors corresponents. En aquest projecte, s'ha utilitzat la plataforma d'OpenAI per a la generació de respostes en llenguatge natural.

La configuració es basa en la generació d'una clau d'API (*API key*), que permet autenticar les peticions realitzades des de les diferents eines del sistema. Aquesta clau es va obtenir a través del panell de configuració d'OpenAI, on es va crear una nova credencial associada al compte del projecte.

<https://platform.openai.com/api-keys>

Un cop generada, la clau es va integrar en eines com n8n i Flowise mitjançant els seus sistemes de gestió de credencials. Això permet que els diferents components que utilitzen models d'intel·ligència artificial es comuniquin amb el servei de manera segura i autenticada.

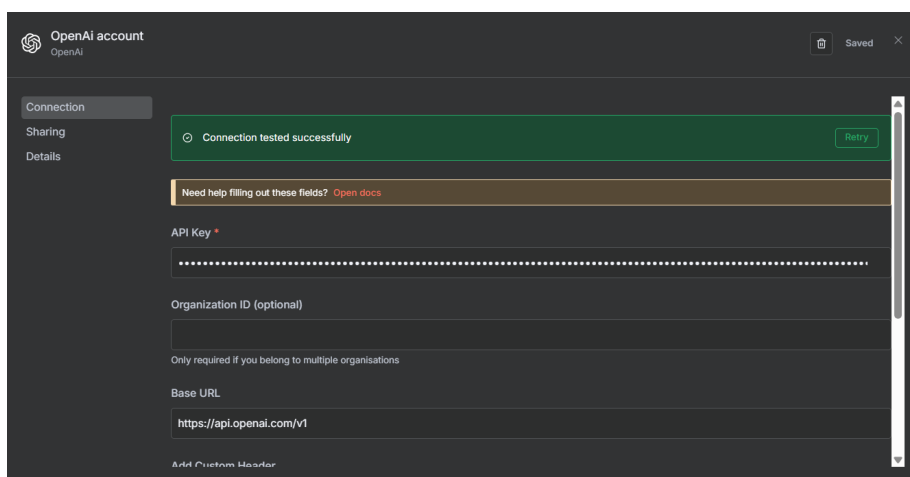


FIGURA B.1: Credencials d'OpenAI connectades a n8n.

B.2 Compte de Google

La integració amb Google Calendar i Gmail es realitza mitjançant el sistema d'autenticació OAuth 2.0 proporcionat per Google Cloud. Aquest mecanisme permet autoritzar l'accés a dades de l'usuari de forma segura sense exposar directament les seves credencials.

<https://console.cloud.google.com/projectcreate>

El primer pas consisteix en la creació d'un projecte a Google Cloud, des d'on es gestionen tots els serveis i configuracions associades. Un cop creat, s'activen les APIs necessàries per al funcionament del sistema, concretament Google Calendar i Gmail.

A continuació, es configura la pantalla de consentiment OAuth, on es defineixen les dades bàsiques de l'aplicació i els permisos que se sol·licitaran a l'usuari. Posteriorment, es crea un client OAuth, en el qual cal especificar les URI de redirecció autoritzades. En aquest cas, s'utilitza l'URL proporcionada per n8n per completar el procés d'autenticació.

Finalment, Google genera un identificador de client (*client ID*) i una clau secreta (*client secret*), que s'introdueixen a n8n per establir la connexió amb els serveis. Aquesta configuració permet que els fluxos d'automatització puguin interactuar amb Google Calendar i Gmail de manera segura i controlada.

B.3 Chatwoot

La integració amb Chatwoot es realitza mitjançant peticions HTTP des d'n8n, utilitzant el node *HTTP Request*. Aquest enfocament permet interactuar directament amb l'API de Chatwoot per gestionar converses, contactes i altres funcionalitats del sistema.

L'autenticació es configura a través del tipus de credencial genèrica (*Generic Credential Type*), seleccionant l'opció *Header Auth*. En aquest cas, és necessari crear una credencial específica on s'inclou el token d'accés proporcionat per Chatwoot.

Aquest token es pot obtenir des del panell de configuració del perfil d'usuari de Chatwoot, i és el que permet autenticar les peticions realitzades contra la seva API.

B.4 Telegram

La integració amb Telegram es basa en la creació d'un bot mitjançant l'eina oficial *BotFather*. Aquest procés s'inicia amb la comanda */newbot*, a partir de la qual es defineixen el nom visible del bot i el seu nom d'usuari únic dins de la plataforma.

Un cop completada la configuració inicial, *BotFather* proporciona un access token que permet interactuar amb l'API HTTP de Telegram. Aquest token és l'element clau per autenticar les peticions i gestionar l'enviament i recepció de missatges.

Per establir la connexió amb n8n, només cal introduir-lo dins del sistema de credencials, al camp corresponent. A partir d'aquest moment, els fluxos d'automatització poden comunicar-se amb Telegram de manera directa i segura.

B.5 Airtable

La connexió amb Airtable es realitza mitjançant l'ús de *Personal Access Tokens*, que permeten definir de manera precisa els permisos d'accés a les dades. Aquests tokens es generen des del panell de desenvolupadors dins del centre de creadors d'Airtable.

<https://airtable.com/create/tokens>

Durant la seva configuració, es poden especificar els diferents àmbits d'accés (*scopes*), com ara la lectura i escriptura de registres, la gestió de comentaris o la consulta de l'estructura de les bases de dades. Aquesta flexibilitat permet adaptar el nivell de permisos a les necessitats concretes del sistema.

A més, és necessari concedir accés explícit al token sobre la base de dades o espai de treball on es troben les taules utilitzades.

Apèndix C

Fitxes de casos d'ús

CAS D'ÚS:	Gestionar consulta d'un client
Versió	Visió usuari
Descripció	Un treballador de l'empresa gestiona una consulta enviada per un client, definint el tipus de resposta i supervisant el procés, que pot ser automàtic o manual.
Actors	Client, Treballador de l'empresa
Precondició	Escenari principal: <ul style="list-style-type: none"> • El client ha enviat un missatge a través d'un canal disponible (WhatsApp o correu electrònic).
Flux principal	<ol style="list-style-type: none"> 1. El client envia un missatge a través d'un canal. 2. El sistema registra la consulta a Chatwoot. 3. El treballador accedeix a la conversa. 4. El treballador defineix el tipus de resposta (automàtica o assistida). 5. El sistema processa la consulta segons la configuració establerta. 6. El sistema genera una resposta. 7. El sistema envia la resposta al client.
Flux alternatiu	<p>4a. Escalar a agent humà</p> <ul style="list-style-type: none"> • El treballador decideix no utilitzar el sistema automàtic. • El treballador redacta la resposta manualment. • El treballador envia la resposta al client.
Postcondició	El client rep una resposta a la seva consulta.

CAS D'ÚS:	Gestionar assistència cognitiva
Versió	Visió usuari
Descripció	Un usuari interactua amb el sistema d'assistència cognitiva mitjançant un bot de Telegram per consultar informació, registrar dades i rebre assistència automatitzada (resums, recordatoris o exercicis).
Actors	Usuari, Bot de Telegram
Precondició	Escenari principal: <ul style="list-style-type: none"> • L'usuari ha iniciat una conversa amb el bot. • L'usuari està registrat al sistema.
Flux principal	<ol style="list-style-type: none"> 1. L'usuari envia un missatge o comanda al bot de Telegram. 2. El sistema rep i interpreta la comanda o missatge en llenguatge natural. 3. El sistema identifica la funcionalitat a executar. 4. El sistema consulta o registra la informació necessària. 5. El sistema genera una resposta. 6. El sistema envia la resposta al usuari.
Flux alternatiu	<p>2a. Missatge en llenguatge natural</p> <ul style="list-style-type: none"> • El sistema interpreta el contingut del missatge. • El sistema determina la comanda corresponent. <p>3a. Execució de funcionalitats específiques: Segons la comanda detectada, el sistema pot executar:</p> <ul style="list-style-type: none"> • Consultar ubicacions (<i>/les_meves_ubicacions</i>) • Consultar historial d'ubicacions (<i>/historial_ubicacions</i>) • Generar resum matinal (<i>/resum_matinal</i>) • Consultar medicació (<i>/medicacio</i>) • Consultar agenda del dia (<i>/agenda_avui</i>) • Generar resum del dia anterior (<i>/resum_ahir</i>) • Realitzar exercicis cognitius (<i>/exercici_cognitiu</i>)
Postcondició	L'usuari rep una resposta adequada a la seva petició.

Apèndix D

Estructura de les dades en AirTable

D.0.1 Taula 1: Historial ubicacions

Aquesta taula té com a objectiu registrar periòdicament la ubicació geogràfica de l'usuari. El sistema genera automàticament un nou registre cada deu minuts, permetent construir un historial detallat dels seus moviments al llarg del temps.

Els camps definits en aquesta taula són els següents:

- **Data i Hora:** moment en què s'ha registrat la ubicació.
- **Latitud:** coordenada de latitud.
- **Longitud:** coordenada de longitud.
- **Ubicació:** descripció textual de la ubicació (adreça o lloc).



	Data i Hora	# Latitud	# Longitud	# Ubicació
1	11/3/2026 13:34	41,982843	2,823977	Pl. de Catalunya, 3, Girona
2	11/3/2026 13:44	41,982843	2,823977	Pl. de Catalunya, 3, Girona
3	11/3/2026 13:54	41,976994	2,816924	Maxi Dia, Carrer de Barcelona
4	11/3/2026 14:04	41,976994	2,816924	Maxi Dia, Carrer de Barcelona
5	11/3/2026 14:14	41,982843	2,823977	Pl. de Catalunya, 3, Girona
6	11/3/2026 14:24	41,963520	2,829845	Carrer de Maria Aurèlia Capmany i Farnés
7	11/3/2026 14:34	41,963520	2,829845	Carrer de Maria Aurèlia Capmany i Farnés
8	11/3/2026 14:44	41,963520	2,829845	Carrer de Maria Aurèlia Capmany i Farnés
9	11/3/2026 14:54	41,963520	2,829845	Carrer de Maria Aurèlia Capmany i Farnés

FIGURA D.1: Taula 1: Historial ubicacions.

D.0.2 Taula 2: Historial de converses

Aquesta taula emmagatzema ubicacions rellevants per a l'usuari, com ara casa, feina o altres llocs habituals. Aquestes dades permeten enriquir la interpretació de la informació geogràfica i facilitar tant la contextualització dels moviments com la generació d'indicacions per arribar a aquests punts d'interès.

Camps definits:

- **Ubicació:** nom assignat per l'usuari (p. ex., "Casa", "Gimnàs").

- **Latitud:** coordenada de latitud.
- **Longitud:** coordenada de longitud.
- **Ubicació exacta:** descripció textual de la ubicació (adreça o lloc).

	A Ubicació	# Latitud	# Longitud	≡ Ubicació exacta
1	Casa 1	41,982843	2,823977	Pl. de Catalunya, 3, Girona
2	Casa 2	41,920829	3,157528	Calle De Francisco De Quevedo
3	Universitat	41,963591	2,829695	Carrer de Maria Aurèlia Capmany i Farnés
4	Metge	41,972346	2,827745	Carrer d'Andreu Tuset i Santamaria
5	Gimnàs	41,969348	2,815124	Carrer del Riu Freser
6	Estació	41,979367	2,816777	Girona l'Estació
7	Supermercat 1	41,976994	2,816928	Maxi Dia, Carrer de Barcelona
8	Supermercat 2	41,964663	2,808363	Lidl, Carrer de Salt
9	Futbol	41,961055	2,827256	Avinguda de Montilivi, 141, Girona
+				

FIGURA D.2: Taula 2: Historial de converses.

D.0.3 Taula 3: Historial missatges

Aquesta taula registra totes les interaccions entre l'usuari i el bot de Telegram, permetent analitzar converses passades i generar resums interpretatius de l'activitat recent.

Camps definits:

- **Rol:** indica l'origen del missatge ("Client" o "Bot").
- **Missatge:** contingut textual del missatge.
- **Data i Hora:** moment en què s'ha enviat o rebut el missatge.

	A Rol	≡ Missatge	📅 Data i Hora	
1	Client	hola	11/3/2026	14:34
2	Bot	Hola, bones! En què et puc ajudar?	11/3/2026	14:34
3	Client	/medicacio	16/3/2026	19:30
4	Client	/resum_ahir	23/3/2026	15:50
+				

FIGURA D.3: Taula 3: Historial missatges.

D.0.4 Taula 4: Medicació

Aquesta taula conté la informació relacionada amb la medicació de l'usuari, permetent gestionar recordatoris, planificació de preses i seguiment del tractament.

Camps definits:

- **Nom:** nom del medicament.
- **Freqüència:** defineix la periodicitat de la presa. Pot prendre els següents valors:
 - Diari: El medicament es pren cada dia.
 - Dies específics: El medicament només es pren dies concrets de la setmana.
 - Cada X hores: El medicament es pren cada un cert interval d’hores.
- **Detalls freqüència:** interval d’hores entre preses. Només es posa valor en cas de que la freqüència sigui *Cada X hores*.
- **Hora:** hora exacta en què s’ha de prendre el medicament.
- **Dies setmana:** Segons la freqüència:
 - Diària o Cada X hores: S’indica que s’ha de prendre “Tots els dies”.
 - Dies específics: S’indiquen els dies de la setmana concrets a prendre la medicació.
- **Notes addicionals:** informació rellevant per a la presa del medicament (per exemple, indicacions mèdiques o recomanacions).

	Nom	Freqüència	Detalls freqüència	Hora	Dies setmana	Notes addicionals
1	Omeprazol	Diari		7:30	Tots els dies	Prendre abans d'esmorzar per protegir l'estómac.
2	Levotiroxina	Diari		8:00	Tots els dies	Sempre amb estómac buit i esperar 30 min abans de menjar.
3	Metformina	Dies específics		12:00	Dilluns Dimercres Divendres	Ajuda a controlar la glucosa, beure aigua abans.
4	Alopurinol	Dies específics		20:00	Dimarts Dijous Dissabte	Prendre amb sopar i evitar alcohol.
5	Paracetamol	Cada X hores	6	14:30	Tots els dies	Només quan cal alleujar febre o dolor.
6	Ibuprofèn	Cada X hores	8	9:00	Tots els dies	No prendre amb estómac buit.

FIGURA D.4: Taula 4: Medicació.

D.0.5 Taula 5: Agenda

Aquesta taula representa els esdeveniments provinents de Google Calendar, sincronitzats amb el sistema per a la seva gestió i seguiment. Permet tenir una visió estructurada de les activitats programades i del seu estat de completitud.

Camps definits:

- **Id:** identificador únic de l'esdeveniment.
- **Nom:** nom de l'esdeveniment.
- **Data:** dia en què té lloc l'esdeveniment.
- **Completat:** estat de la tasca:

- ✗: Tasca pendent
- ✓: Tasca completada

	A Id	Nom	Data	Completat
1	57t1klqo6dt0n121ph3oupebqr	Compra	14/3/2026	✗
2	13j7u8bcmgt7n7rarbrjfq02hp	Partit Girona Montilivi	14/3/2026	✓
3	0anu1vq2o1qq71ksjk4sropuk	Película	14/3/2026	✓
4	1i85t0ge4thlv31psk4mkjtjt	angles	23/3/2026	✗
5	49hje2enbqbhthttib94a11krao	examen	23/3/2026	✓
6	3iqjcillb20a241absolv8oaj	sopar	23/3/2026	✓
7	6k8b138io5s7q62no8birgnur6	Cine	24/3/2026	✗
8	2tim7svcl476vriieff679f6c2	partit futbol	24/3/2026	✓

FIGURA D.5: Taula 5: Agenda.

D.0.6 Taula 6: Exercicis

Aquesta taula registra els resultats dels exercicis cognitius realitzats per l'usuari, permetent fer un seguiment del seu rendiment i evolució al llarg del temps.

Camps definits:

- **Data i Hora:** moment en què s'ha realitzat l'exercici.
- **Puntuació:** resultat obtingut en l'exercici.

	Data i Hora	Puntuació
1	14/3/2026 13:29	2
2	24/3/2026 19:22	3

FIGURA D.6: Taula 6: Exercicis.

D.0.7 Taula 7: Usuari

Aquesta taula emmagatzema informació important de l'usuari.

Camps definits:

- **Nom:** nom de l'usuari.
- **Xat id:** identificador únic de la conversa.

	A Nom	# Xat id
1	Alex	5210720889
+		

FIGURA D.7: Taula 7: Usuari.

Apèndix E

Preparació de l'entorn del cas d'ús 1

El procés de preparació de l'entorn local va consistir en:

- Clonació del repositori mitjançant Git.

```
git clone https://github.com/willph/evolutionApi_chatwoot_docker server
```

- Configuració dels fitxers d'entorn (.env), utilitzats per definir variables de configuració del sistema; no es van modificar les contrasenyes per tractar-se d'un entorn de proves i no d'un producte final.

- Construcció dels contenidors amb Docker Compose.

```
docker-compose up --build --no-start
```

- Inicialització de la base de dades.

```
docker-compose run --rm rails bundle exec rails db:chatwoot_prepare
```

- Execució dels serveis en mode local.

```
docker-compose up -d
```

Un cop finalitzat aquest procés, es disposava d'un entorn completament funcional per iniciar les proves d'integració.

Apèndix F

Coneixements bàsics d'n8n

F.1 Recepció de missatges mitjançant webhook

El primer element del flux d'automatització implementat a n8n és el node webhook, que actua com a punt d'entrada per a la recepció d'esdeveniments externs. En aquest projecte, aquest mecanisme permet rebre en temps real les notificacions generades per Chatwoot quan es produeix una interacció amb un client.

Per a la integració amb Chatwoot, es van configurar dos *endpoints* de webhook: un per a l'entorn de proves i un altre per a l'entorn de producció. El webhook de proves s'utilitza durant el desenvolupament per validar el comportament del flux, mentre que el de producció es manté actiu de manera contínua per gestionar les interaccions reals.

- http://n8n:5678/webhook-test/assistent_reactivat ⇒ Eventos suscritos: Conversación actualizada
- http://n8n:5678/webhook/assistent_reactivat ⇒ Eventos suscritos: Conversación actualizada

Ambdós *endpoints* es van configurar per escoltar l'esdeveniment *conversation_updated*, que s'activa cada vegada que es produeix una modificació en una conversa (canvis d'etiquetes, atributs, etc.). Aquesta configuració permet capturar qualsevol interacció rellevant realitzada per l'usuari.

El flux de comunicació és el següent: el client envia un missatge a través de WhatsApp, aquest és processat per Evolution API (vegeu Annex G), posteriorment gestionat per Chatwoot i, finalment, Chatwoot envia una notificació al webhook d'n8n, activant així el flux d'automatització.



FIGURA F.1: Esquema de comunicació entre els diferents serveis.

La informació rebuda pel webhook s'envia en format JSON i conté múltiples camps rellevants, entre els quals destaquen l'identificador de la conversa, el contingut del missatge, el número de telèfon d'origen, el tipus de missatge (text, àudio o imatge), el nom del contacte i altres metadades addicionals. Aquestes dades constitueixen la base per al processament posterior dins del flux.

F.2 Respondre un missatge

Un cop es va validar el correcte funcionament del webhook com a punt d'entrada, el següent pas va consistir a afegir un node d'OpenAI per provar la connexió directa entre n8n i la intel·ligència artificial. L'objectiu d'aquesta fase era confirmar que la comunicació entre ambdós sistemes era fluida, per la qual cosa es va configurar un *prompt* senzill amb una tasca genèrica on l'agent simulava ser un representant de l'empresa per respondre a les preguntes de l'usuari.

Després d'aquest node de processament, es va implementar una petició HTTP Request de tipus POST dirigida a l'API de Chatwoot. Aquesta configuració permet enviar el missatge generat per la IA de tornada a la plataforma de gestió de xats, tancant així el cicle d'automatització: des de la recepció de la consulta original fins a l'enviament de la resposta automàtica al client. Aquesta prova inicial va servir per demostrar la viabilitat tècnica de la solució abans d'implementar lògiques més complexes.

Flux resultant:

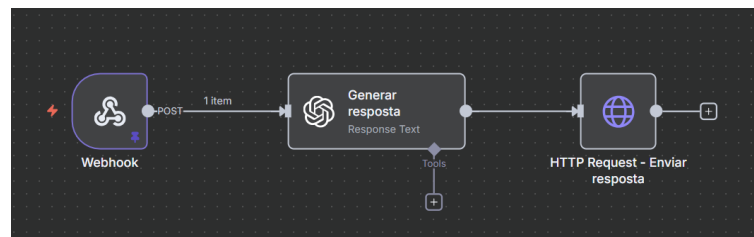


FIGURA F.2: Generar i enviar una resposta automàtica al client.

F.3 Gestió dels diferents tipus de missatge

Un cop implementada la primera versió funcional del flux, capaç de rebre un missatge i generar una resposta automàtica, el següent pas en el procés d'aprenentatge va ser gestionar correctament els diferents tipus de contingut que pot enviar un client. En els canals de missatgeria instantània no tots els missatges són de text, ja que els usuaris també poden enviar notes de veu o imatges, fet que requereix un tractament específic en cada cas.

Per abordar aquesta problemàtica, es va introduir un node de tipus *switch* immediatament després del webhook. Aquest node permet ramificar el flux en funció d'una condició, en aquest cas segons el tipus d'arxiu associat al missatge rebut. La classificació es realitza a partir del camp que indica el tipus de fitxer de l'adjunt rebut.

En el cas dels missatges de text, el flux segueix el camí més directe. En absència de fitxers adjunts, el contingut textual del missatge es recupera directament del JSON rebut pel webhook i s'emmagatzema en un node de tipus *set* dins d'un camp comú anomenat *missatge*. Aquesta primera branca permet mantenir el comportament bàsic del flux ja implementat en l'apartat anterior.

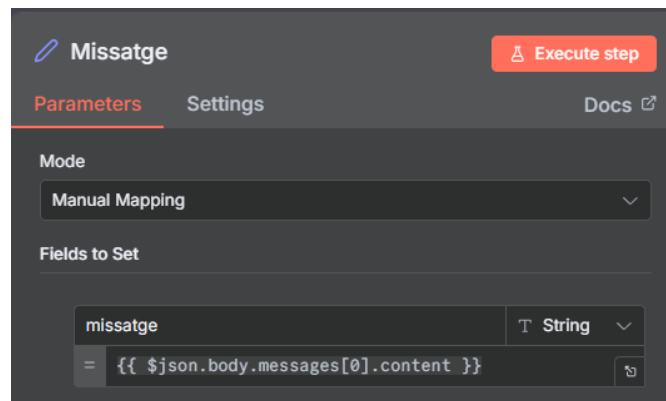


FIGURA F.3: Node *set* utilitzat per unificar el contingut del missatge en un camp comú independentment del canal d'origen.

Quan el missatge rebut és una nota de veu, el tractament és diferent. En aquest cas, el flux accedeix primer a l'arxiu d'àudio mitjançant una petició HTTP de tipus GET, utilitzant l'URL proporcionada per Chatwoot provinent del webhook. Un cop recuperat l'arxiu, aquest es processa mitjançant un agent d'intel·ligència artificial d'OpenAI amb capacitats de transcripció d'àudio. El resultat d'aquesta transcripció és un text que representa el contingut del missatge original, el qual s'emmagatzema novament en un node *set* sota el mateix camp *missatge*.

De manera anàloga, per als missatges que contenen imatges, el flux recupera primer l'arxiu d'imatge a partir de l'URL proporcionada al webhook. Aquest arxiu es processa posteriorment amb l'agent d'IA, al qual se li indica mitjançant un *prompt* específic que descriu el contingut de la imatge. El text generat com a descripció s'emmagatzema també en el camp *missatge* mitjançant un node *set*.

Flux resultant:

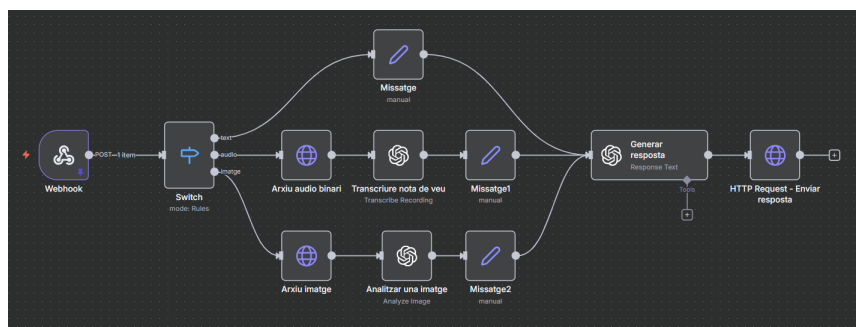


FIGURA F.4: Adaptació del flux de treball per incloure el tractament de notes de veu i imatges.

Apèndix G

Integració de WhatsApp mitjançant Evolution API

Per tal d'integrar el canal de WhatsApp amb Chatwoot, en aquest projecte s'ha utilitzat Evolution API com a servei intermediari, en comptes de la solució oficial de Meta.

La principal motivació d'aquesta elecció és que la integració oficial, anomenada *WhatsApp Cloud API*, no està orientada a comptes personals ni a entorns de prova senzills. Aquesta API només permet la connexió de números registrats específicament com a *WhatsApp Business API*, un model pensat principalment per empreses.

L'ús de la solució oficial implica diversos requisits addicionals, com ara la creació d'un compte empresarial de Meta, processos de verificació, configuració avançada i la dedicació exclusiva del número de telèfon a aquest servei. A més, un cop un número és migrat a aquest model, deixa d'estar disponible per al seu ús habitual des de l'aplicació estàndard de WhatsApp.

Per aquest motiu, en entorns d'aprenentatge, desenvolupament o prototipatge, és habitual utilitzar alternatives com Evolution API. Aquesta eina es basa internament en la llibreria *Baileys*, que permet establir connexió amb WhatsApp Web de forma automatitzada.

Aquesta aproximació presenta diversos avantatges pràctics:

- Permet utilitzar números personals o comptes estàndard de WhatsApp.
- No requereix verificació empresarial per part de Meta.
- Facilita una posada en marxa ràpida.
- Simplifica la integració amb plataformes com Chatwoot.
- Resulta adequada per a entorns locals, proves i desenvolupament.

Per aquests motius, Evolution API s'ha considerat la solució més adequada per al present projecte, ja que permet implementar funcionalitats reals de missatgeria amb una configuració molt més accessible que la solució oficial.

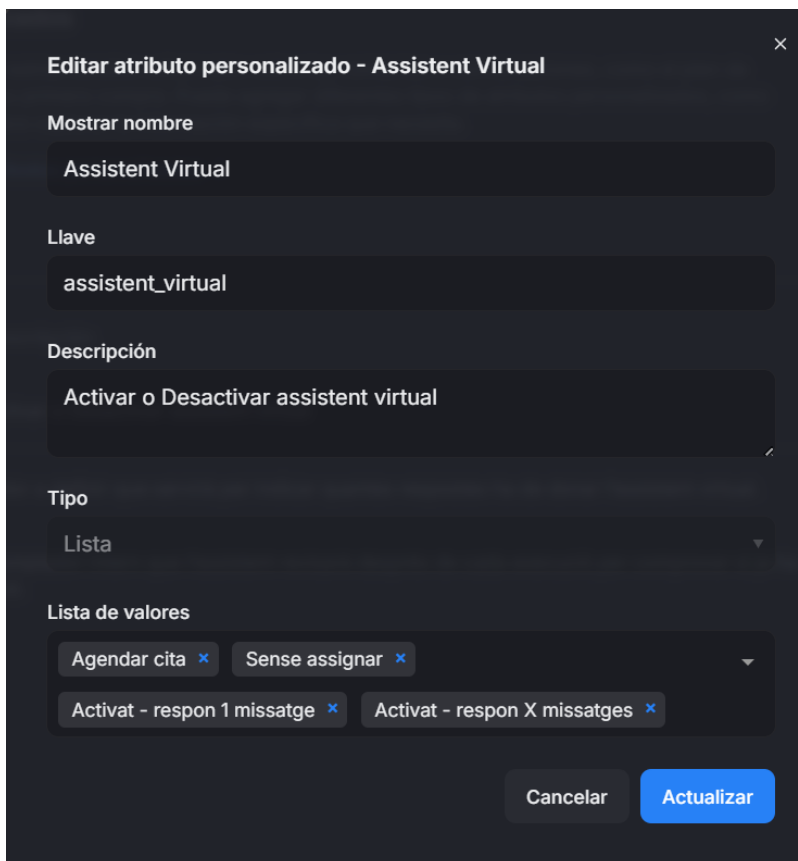


FIGURA G.1: Esquema de comunicació entre Evolution API i Meta.

Apèndix H

Configuració de l'atribut, etiquetes i regles d'automatització

H.1 Configuració dels atributs personalitzats



Editar atributo personalizado - Asistente Virtual

Mostrar nombre
Assistent Virtual

Llave
assistent_virtual

Descripción
Activar o Desactivar asistente virtual

Tipo
Lista

Lista de valores
Agendar cita × Sense assignar ×
Activat - respon 1 missatge × Activat - respon X missatges ×

Cancelar Actualizar

FIGURA H.1: Configuració de l'atribut personalitzat *Assistent Virtual* de Chatwoot.



FIGURA H.2: Configuració dels tres atributs personalitzats definits: *Assistent Virtual*, *Limit respostes* i *Comptador respostes*.

H.2 Configuració de les etiquetes

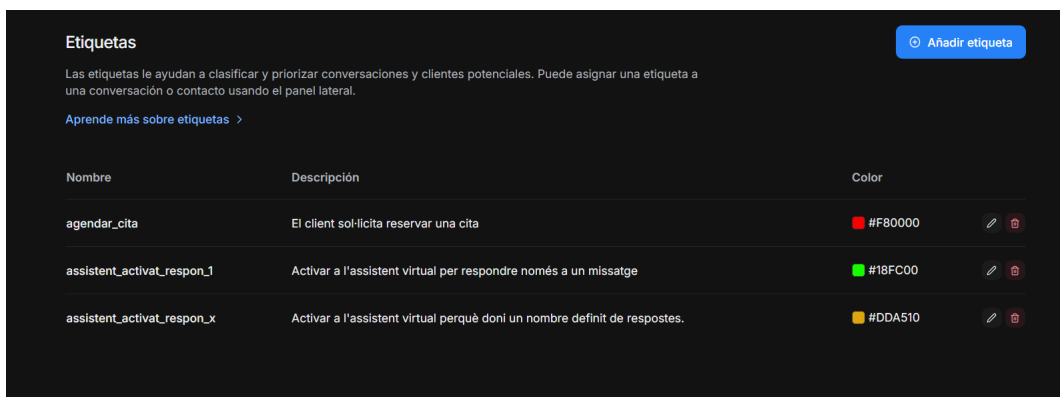


FIGURA H.3: Etiquetes definides per a l'*Assistent Virtual*: *agendar_cita*, *assistent_activat_respon_1* i *assistent_activat_respon_x*.

H.3 Configuració de les regles d'automatització

Editar regla de automatización

Nombre de regla
Assistent sense assignar

Descripción
Es desactiven totes les etiquetes i tasques assignades a l'assistent (reinici).

Evento
Conversación actualizada

Condiciones
Assistent Virtual Igual a Sense assignar

+ Añadir condición

Acciones
Eliminar etiqueta

- agendar_cita
- assistent_activat_respon_1
- assistent_activat_respon_x

+ Añadir Acción

Cancelar Actualizar

FIGURA H.4: Regla d'automatització que en assignar-se el valor *Sense assignar* a l'atribut *Assistent virtual*, elimina qualsevol etiqueta que hi hagi activa.

Editar regla de automatización ×

Nombre de regla
Agendar cita

Descripción
Agendar una cita

Evento
Conversación actualizada

Condiciones

Assistent Virtual Igual a Agendar cita ×

+ Añadir condición

Acciones

Añadir etiqueta agendar_cita ×

Eliminar etiqueta asistent_activat_respon_1 ×
asistent_activat_respon_x ×

+ Añadir Acción

Cancelar Actualizar

FIGURA H.5: Regla d'automatització que en assignar-se el valor *Agendar cita* a l'atribut *Assistent virtual*, s'afegeix l'etiqueta *agendar_cita* i s'elimina qualsevol altra etiqueta activa.

Editar regla de automatización

Nombre de regla
Asistent activat respon 1

Descripción
Asistent activat, però respon només 1 missatge

Evento
Conversación actualizada

Condiciones

Asistent Virtual Igual a Activat - respon 1 missatge

+ Añadir condición

Acciones

Añadir etiqueta asistent_activat_respon_1

Eliminar etiqueta agendar_cita asistent_activat_respon_x

+ Añadir Acción

Cancelar Actualizar

FIGURA H.6: Regla d'automatització que en assignar-se el valor *Activat - respon 1 missatge* a l'atribut *Asistent virtual*, s'afegeix l'etiqueta *asistent_activat_respon_1* i s'elimina qualsevol altra etiqueta activa.

Editar regla de automatización

Nombre de regla
Asistent activat respon X

Descripción
L'assistent dona un nombre X de respostes segons el que hàgim definit.

Evento
Conversación actualizada

Condiciones

Asistent Virtual Igual a Activat - respon X missatges

+ Añadir condición

Acciones

Añadir etiqueta asistent_activat_respon_x

Eliminar etiqueta agendar_cita asistent_activat_respon_1

+ Añadir Acción

Cancelar Actualizar

FIGURA H.7: Regla d'automatització que en assignar-se el valor *Activat - respon X missatges* a l'atribut *Asistent virtual*, s'afegeix l'etiqueta *asistent_activat_respon_x* i s'elimina qualsevol altra etiqueta activa.

Apèndix I

Configuració de cites mitjançant Google Calendar

Per facilitar la gestió de cites amb clients dins del primer cas d'ús, es va utilitzar la funcionalitat *Appointment Schedule* de Google Calendar. Aquesta eina permet definir franges horàries disponibles i generar una pàgina pública de reserves, des de la qual els clients poden seleccionar directament el dia i l'hora que millor s'adapten a les seves necessitats.

L'objectiu d'aquesta integració és automatitzar el procés de reserva de cites, evitant intercanvis manuals de missatges per acordar disponibilitats i reduint el temps dedicat a tasques administratives.

Per a la simulació del sistema es va crear un calendari de reserves anomenat *TFG cites perruqueria*, amb una durada estàndard de 30 minuts per cita.

La disponibilitat configurada fou la següent, reproduint un horari típic de jornada laboral:

- De dilluns a divendres:
 - de 9:00 h a 13:00 h
 - de 14:00 h a 16:00 h
- Dissabte i diumenge: no disponible

També es va establir la zona horària corresponent a Madrid (GMT+02:00), garantint que les hores mostrades als usuaris coincidissin amb l'horari local del negoci.

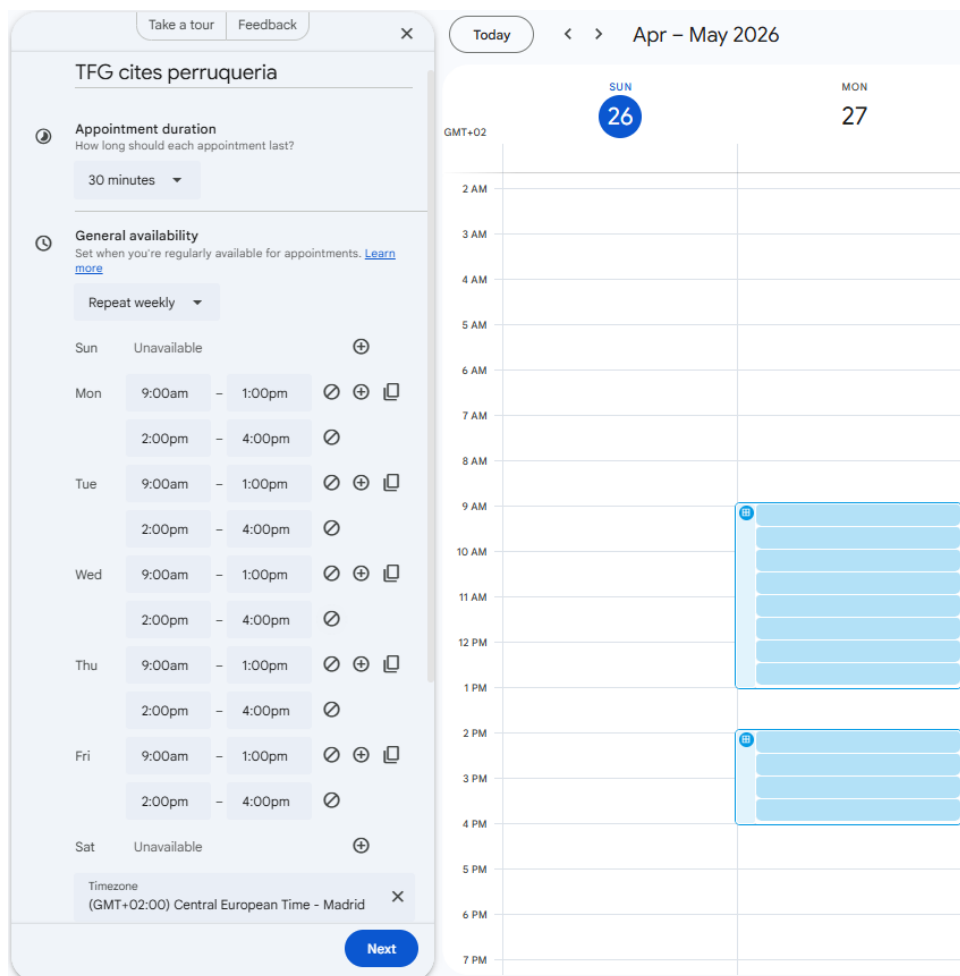


FIGURA I.1: Configuració de l'horari de cites a Google Calendar.

A més, la pàgina de reserves inclou una descripció informativa visible pels clients:

Benvolgut, aquesta agenda de cites serveix per simular un calendari intern que ens permet gestionar i reservar hores amb els clients de manera organitzada i eficient.

Un cop configurat el calendari, Google genera automàticament un enllaç públic de reserva. Aquest enllaç és el que el sistema envia automàticament als clients quan, dins del flux d'n8n, es detecta que la conversa ha estat etiquetada amb *agendar_cita*.

<https://calendar.app.google/KRiUyr69heiwucyq7>

D'aquesta manera, quan el client accedeix a l'enllaç, pot visualitzar únicament les franges disponibles i confirmar la reserva de forma autònoma, sense intervenció manual per part de l'empresa.

Aquest mecanisme aporta diversos avantatges:

- Reducció del temps dedicat a la coordinació de cites.
- Disponibilitat permanent del sistema de reserves.

- Eliminació d'errors derivats de reserves manuals.
- Millora de l'experiència de l'usuari, en oferir una reserva ràpida i senzilla.
- Integració directa amb l'ecosistema de Google Calendar.

En un entorn real, aquesta configuració podria ampliar-se incorporant diferents tipus de serveis, durades variables de cita, marges de seguretat entre reserves o múltiples calendaris segons professionals disponibles.

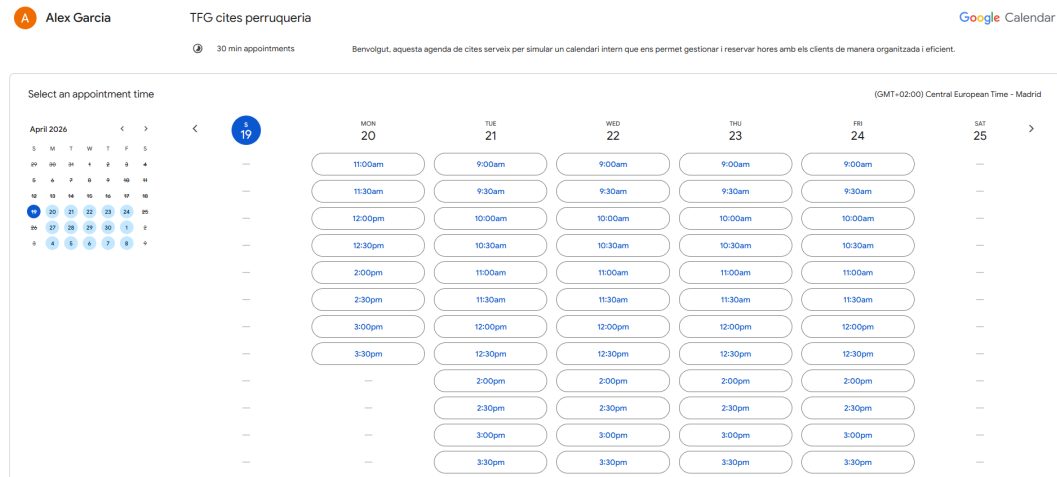


FIGURA I.2: Vista del client per la gestió de cites a Google Calendar.

Apèndix J

Configuració de Flowise

En aquest apartat es detallen els passos d'implementació del flux realitzat a Flowise.

J.1 Configuració de la sessió i gestió d'estats

El flux s'inicia amb un node *Start* configurat per gestionar la persistència de les dades, assegurant que cada interacció mantingui la continuïtat necessària:

- *Input Type*: Chat Input.
- *Ephemeral Memory*: Desactivat (per evitar la pèrdua de context en tancar la sessió).
- *Persist State*: Activat (permet l'emmagatzematge de l'estat mitjançant un identificador únic).

J.2 Classificació i lògica de resposta


S'utilitza un *Condition Agent* com a motor de decisió per categoritzar l'entrada de l'usuari en tres escenaris diferenciats:


- Escenari 1: *Salutació*: Dissenyat per a interaccions inicials on l'usuari no realitza cap consulta específica.


Prompt: Respon a la salutació de l'usuari en el seu mateix idioma amb una forma amable i propera, donant-li la benvinguda i demanant-li de forma oberta com el pots ajudar per fer-lo sentir ben atès.

- Escenari 2: *Pregunta genèrica*: Aquest escenari gestiona les preguntes sobre el negoci mitjançant una arquitectura de recuperació d'informació. Consta de dues fases:
 - **Recuperació (*Retriever*)**: El node analitza el document RAG per identificar els fragments (*chunks*) amb major similitud semàntica a la consulta de l'usuari.
 - **Generació de la resposta**: L'agent processa la pregunta i els resultats del retriever sota les següents directrius:

Prompt: Actua com si fossis un representant oficial de l'empresa. Respon de manera natural i professional, sempre en primera persona del plural ("nosaltres"). No parlis de l'empresa en tercera persona. Genera la resposta final a la PREGUNTA DE L'USUARI només si trobes la resposta als RESULTATS. En cas contrari respón "No puc donar resposta a la teva pregunta" en el mateix idioma. Dona explicacions llargues, clares i detallades. Si cal informar del email de "perruqueriaAlexTFG@mustramit.com" fes-ho només una vegada (no ho repeteixis varis cops).



Retriever 

Knowledge (Document Stores) * 

Document Store *  0

Simulació Perruqueria Alex TFG


+ Add Knowledge (Document Stores)

Retriever Query *  

{{ question }}

Output Format *

Text

Update Flow State 

+ Add Update Flow State

FIGURA J.1: Configuració del node *Retriever* de Flowise amb el document RAG.

- Escenari 3: *Agrair l'agraïment*: Escenari de tancament per finalitzar la interacció de manera cordial.

Prompt: Respon a l'agraïment de l'usuari en el seu idioma de manera natural, agraint-li el gest i indicant disponibilitat per a futures consultes. La resposta ha de ser breu i mancada d'informació comercial, prioritzant un comiat atent i obert.

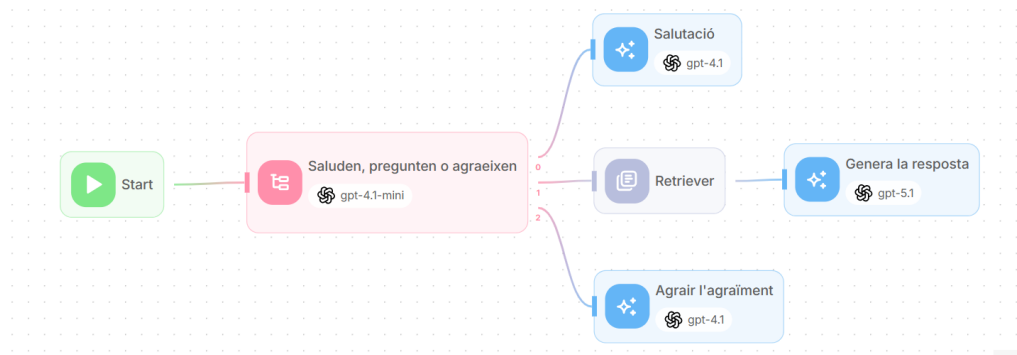


FIGURA J.2: Flux resultant de Flowise.

Apèndix K

Configuració del sistema RAG

En aquest apartat es descriu el procés tècnic de preparació, processament i emmagatzematge de la base de coneixement utilitzada per a l'atenció al client.

K.1 Preparació de la base de coneixement

Per a la implementació d'aquest sistema, s'ha creat un repositori documental dins de la secció *Document Stores* de Flowise que conté tota la informació operativa del negoci. Per a la redacció d'aquest document, s'ha comptat amb el suport d'agents d'IA, fet que ha permès estructurar de manera clara i exhaustiva les dades de l'empresa.

El procés de preparació d'aquesta informació per al sistema consta de tres fases tècniques:

1. **Divisió en fragments (*Chunking*):** El document original es fragmenta en trosos més petits anomenats *chunks*. Això és essencial perquè l'IA pugui localitzar la dada exacta sense haver de processar tot el document sencer en cada consulta.
2. **Generació d'*embeddings*:** Mitjançant el model d'OpenAI, cada fragment de text es converteix en un vector numèric (*embedding*). Aquest vector representa el significat semàntic del text.
3. **Emmagatzematge vectorial (*Vector Store & Record Manager*):** Aquests vectors es guarden en una base de dades especialitzada. El *Record Manager* s'encarrega de gestionar la sincronització i l'actualització d'aquesta informació per evitar duplicats i optimitzar la recuperació.

K.2 Fonaments de la divisió en fragments (*Chunking*)

Perquè el sistema pugui recuperar informació de manera eficient, el document es fragmenta en unitats menors anomenades *chunks*. Aquesta divisió és crítica per a la precisió del sistema:

- **Mida dels fragments:** S'ha optat per una granularitat mitjana per equilibrar la precisió semàntica i el cost computacional. Fragments massa grans podrien introduir soroll (informació irrellevant), mentre que fragments massa petits podrien perdre el context necessari per a una resposta coherent.

Concretament, el RAG utilitzat pel projecte, té una extensió total de 12.200 caràcters, els quals s'han segmentat en un total de 23 chunks amb un límit màxim de 730 caràcters cadascun. És important destacar que la mida dels fragments no és uniforme, ja que el sistema s'ha configurat per respectar els salts de paràgraf originals (generalment d'entre 500 i 600 caràcters). D'aquesta manera, cada chunk conté un paràgraf complet sense sobrepassar el límit establert, evitant talls arbitraris que podrien fer perdre el context necessari per a una resposta coherent.

- **Estratègia de cerca:** El sistema transforma cada fragment i cada consulta de l'usuari en un vector numèric (*embedding*). Mitjançant el càlcul de la distància vectorial, el sistema identifica quins fragments de la base de dades s'assemblen més a la pregunta formulada.

K.3 Procés d'Embedding

El procés d'*upsert* (pujada i indexació) dels fragments s'ha realitzat seguint tres passos configurats a Flowise:

1. **Pas 1: Selecció del Model d'Embedding** S'ha seleccionat el model *text-embedding-3-small* d'OpenAI, una elecció fonamentada en la seva excel·lent viabilitat econòmica i la seva robustesa multilingüe, que permet vincular conceptes en català i castellà amb alta precisió semàntica. Tècnicament, s'ha configurat mantenint les 1536 dimensions natives per maximitzar la intel·ligència de la cerca i s'ha desactivat l'opció *Strip New Lines* per preservar l'estructura de paràgrafs original del document.
2. **Pas 2: Emmagatzematge Vectorial (Pinecone)** Com a base de dades de vectors (*Vector Store*), s'ha utilitzat Pinecone, configurant un índex amb les següents especificacions tècniques:
 - **Mètrica de similitud:** *Cosine* (Cosina), ideal per comparar la proximitat entre vectors de text.
 - **Infraestructura:** Desplegat sobre AWS (regió us-east-1).
 - **Top K: 4.** El sistema recupera exactament els 4 fragments més rellevants per a cada consulta per nodrir de context el model sense saturar la seva finestra de context.
 - **Search Type:** *Similarity*, és a dir, la cerca es basa en la similitud entre vectors.
3. **Pas 3: Gestió de Registres (Record Manager)** Actua com una capa de control intel·ligent que supervisa la sincronització entre el document original i la base de dades vectorial. La seva funció principal és evitar la duplictat de dades i optimitzar el procés d'actualització: en lloc de tornar a processar i pujar tot el document cada vegada que es fa un canvi, el *Record Manager* detecta quins fragments han estat modificats, afegits o eliminats.

Simulació Perruqueria Alex TFG / Document
Perruqueria TFG (None)

Configure Embeddings, Vector Store and Record Manager

Reset Save Config Upsert

Embeddings Vector Store Record Manager

OpenAI Embeddings

Connect Credential *
n8n-tfg

Model Name *
text-embedding-3-small

Strip New Lines

Batch Size

Timeout

BasePath

Dimensions

Pinecone

Connect Credential *
n8n-flowise

Pinecone Index *
simulacio-perruqueria-tfg

Pinecone Namespace
my-first-namespace

File Upload

Pinecone Text Key
text

Pinecone Metadata Filter
@ items

Top K
4

Search Type
Similarity

Fetch K (for MMR Search)
20

Lambda (for MMR Search)
0.5

Postgres Record Manager

Connect Credential *
rag-flowise

Host *
postgres-chatwoot

Database *
chatwoot

Port
5432

SSL

Additional Connection Configuration
@ items

Table Name
upsertion_records

Namespace

Cleanup *
None

SourceId Key
source

FIGURA K.1: Configuració de l'embedding pel document RAG.

K.4 Funcionament del sistema RAG

Quan un client fa una pregunta sobre l'empresa, el node *Retriever* analitza el missatge, cerca els fragments (*chunks*) més similars en la base de dades vectorial i els lliura a l'agent d'IA.

Finalment, el node de generació rep dues entrades: la pregunta de l'usuari i el context extret de la documentació. Amb tot plegat, el model genera una resposta que no es basa en el seu coneixement general d'Internet, sinó exclusivament en la informació corporativa definida, garantint així la màxima precisió i fiabilitat.

K.5 Validació i Proves de Recuperació

Per verificar el funcionament del sistema, es va realitzar un test de recuperació (*Retrieval Test*) amb la pregunta: "Com em puc donar d'alta al sistema?".

1. El sistema va retornar com a primera opció el fragment específic que descriu el procés d'alta automàtica.

2. L'ordre de prioritats dels fragments recuperats confirma que la mètrica *Cosine* i el model d'OpenAI funcionen correctament, prioritzant la coincidència semàntica (significat) per sobre de la simple coincidència de paraules clau.

The screenshot shows the Pinecone Retrieval Playground interface. At the top, the title is "Simulació Perruqueria Alex TFG" with a "Save Config" button. Below the title, there is a search bar with the query "Com em puc donar d'alta al sistema?". The interface is divided into two main sections: "Pinecone" on the left and "Retrieved Documents" on the right.

Pinecone Configuration:

- Connect Credential: n8n-flowise
- Pinecone Index: simulacio-perruqueria-tfg
- Pinecone Namespace: my-first-namespace
- File Upload: Disabled
- Pinecone Text Key: (empty)

Retrieved Documents: Count: 4, Time taken: 2677 millis.

The retrieved documents are displayed in a grid:

- #4. Characters: 543**
Disposen d'un sistema d'alta i baixa de clients. L'alta es produeix automàticament en completar el registre. Si un client vol donar-se de baixa, pot fer-ho des del seu perfil a l'apartat "Configuració del compte" o bé sol·licitant-ho per correu electrònic a baixes@perruqueriaalexftg.com.
Un cop tramitada la baixa, les seves dades quedaran bloquejades i no podrà accedir a promocions ni al sistema de reserves. Hem creat un videotutorial
- #23. Characters: 677**
Puc contractar un servei per a una altra persona (regal)? Si, pots comprar una targeta regal o un bo de tractament per a una altra persona des de la nostra web, indicant l'import o el servei concret. La persona que el rep haurà de registrar-se (si no ho està) i podrà bescanviar-lo des del seu perfil o presentant el codi del bo al saló.
Com puc saber si un producte o tractament és adequat per al meu tipus de cabell? Pots consultar-
- #6. Characters: 510**
Complementant la nostra secció de Clients VIP, oferim el programa de fidelització "Punts Alex", accessible per a tots els clients registrats. Per cada 10 € gastats en serveis o productes, el client acumula 1 punt. Cada punt acumulat equival a un
- #21. Characters: 349**
Puc portar el meu propi producte per a un tractament? Si, és possible. En aquest cas, el servei es facturarà com a "aplicació de producte extern" i no es podrà garantir el resultat final, atès que desconeixem la composició i l'estat del producte. No obstant això

FIGURA K.2: Prova de funcionament pel Retrieval en Flowise.

Apèndix L

Descripció del flux del primer cas d'ús

Webhook Aquest node inicia el flux quan es rep un esdeveniment des de Chatwoot, tant per missatges nous com per actualitzacions de converses.

Code - esForaHorari? Aquest node comprova si el missatge s'ha rebut fora de l'horari laboral configurat.

If - esForaHorari Aquest node divideix el flux segons si el missatge arriba fora o dins de l'horari laboral:

- **True:** execució del flux automàtic fora d'horari (sense revisió d'etiquetes), continua cap a *Filtre - Missatge entrant*.
- **False:** el flux continua cap a la revisió de les etiquetes, concretament cap al node *If - missatge client?*.

Filtre - Missatge entrant Aquest node filtra únicament els missatges entrants enviats pel client, tant des de WhatsApp com des de correu electrònic. Els missatges vàlids continuen cap al node *Switch* per identificar el tipus de contingut rebut.

If - missatge client? Aquest node comprova que l'esdeveniment rebut correspon realment a un missatge enviat pel client i no a una acció interna del sistema o d'un agent.

- **True:** continua el flux cap a *If - etiqueta activadora*.
- **False:** el flux es deté.

If - etiqueta activadora Aquest node verifica si la conversa ha rebut alguna de les etiquetes que activen l'assistent (*assistent_activat_respon_1*, *agendar_cita* o *assistent_activat_respon_x*).

- **True:** continua el flux cap a *If - respon1* o *agenda cita*; *else - responX*.
- **False:** el flux es deté.

If - respon1 o agenda cita; else - responX Aquest node comprova si la conversa conté les etiquetes *assistent_activat_respon_1* o *agendar_cita*.

- **True:** continua el flux cap a *If - respon1 o responX; else - cita*.
- **False:** continua el flux cap a *If - limit > comptador*.

If - limit > comptador Aquest node valida que el nombre de respostes automàtiques enviades no hagi superat el límit configurat a la conversa.

- **True:** continua cap a *If - respon1 o responX; else - cita*.
- **False:** continua el flux cap a *HTTP Post - Treure etiqueta i reiniciar limit + comptador*.

HTTP Post - Treure etiqueta i reiniciar limit + comptador Aquest node elimina l'etiqueta assignada i reinicia el comptador i el límit de respostes a 0. El flux es deté.

If - respon1 o responX else - cita Aquest node comprova si la conversa està configurada en mode de resposta automàtica (*assistent_activat_respon_1* o *assistent_activat_respon_x*) o si s'ha de derivar a l'agendament d'una cita.

- **True:** continua el flux cap a *Switch*.
- **False:** continua el flux cap a *HTTP Request - Enviar missatge fix* (agendament de cita).

HTTP Request - Enviar missatge fix Aquest node envia un missatge fix al client amb l'enllaç de Google Calendar perquè pugui seleccionar l'hora de la reunió. Després continua cap al node *HTTP Request - Reiniciar assistent virtual*.

HTTP Request - Reiniciar assistent virtual Aquest node reinicia el valor de l'assistent virtual a *Sense assignar*, eliminant així qualsevol etiqueta activa.

Switch Aquest node divideix el flux segons el tipus de missatge rebut:

- **Text:** continua cap a *If - Whatsapp o email*.
- **Audio:** continua cap a *HTTP Get - Arxiu audio binari*.
- **Image:** continua cap a *HTTP Get - Arxiu imatge*.

If - Whatsapp o email Aquest node identifica el canal des d'on s'ha rebut el missatge.

- **Whatsapp:** continua cap a *Set - missatge*.
- **Email:** continua cap a *Set - missatge1*.

Set - missatge Aquest node normalitza el contingut del missatge rebut en un únic camp anomenat *missatge*, independentment de si l'esdeveniment prové de *message_created* o *conversation_updated*. Aquest node va cap a *HTTP Post - Crida Flowise*.

HTTP Post - Crida Flowise Aquest node envia la consulta de l'usuari al flux de Flowise per generar una resposta automàtica. Es passen com a paràmetres el missatge, el *sessionId* i el *chatId*. En el cas del correu electrònic, aquests identificadors es construeixen a partir del correu de l'usuari i l'assumpte del missatge. Després continua cap a *HTTP Request - Enviar resposta*.

Set - missatge1 Aquest node realitza la mateixa normalització del missatge que el node anterior, guardant el contingut al camp *missatge* independentment del tipus d'esdeveniment rebut. Aquest node va cap a *HTTP Post - Crida Flowise1*.

HTTP Get - Arxiu audio binari Aquest node recupera l'arxiu binari corresponent a la nota de veu enviada pel client. Continua cap a *Transcriure nota de veu*.

Transcriure nota de veu Aquest node utilitza OpenAI per transcriure la nota de veu del missatge rebut. Continua cap a *Set - missatge2*.

Set - missatge2 Aquest node desa la transcripció generada al camp comú *missatge* per tal de poder reutilitzar el mateix procés de resposta automàtica. Aquest node va cap a *HTTP Post - Crida Flowise1*.

El mateix procediment del camí de la nota de veu també es segueix per quan es rep una imatge: Primer es recupera l'arxiu *HTTP Get - Arxiu imatge* i es passa al node *Analitzar una imatge* d'OpenAI. Continua cap a *Set - Missatge3*. Finalment, Aquest node va cap a *HTTP Post - Crida Flowise1*.

HTTP Post - Crida Flowise1 Aquest node envia el contingut del camp *missatge* al flux de Flowise independentment del tipus de missatge original rebut. En aquest cas, el *sessionId* i el *chatId* corresponen al número de telèfon del client. El flux continua cap a *HTTP Request - Enviar resposta*.

HTTP Request - Enviar resposta Aquest node envia al client la resposta generada automàticament per Flowise a partir de la consulta realitzada.

If - respon 1 vegada Aquest node comprova si la conversa ha estat marcada amb l'etiqueta *assistent_activat_respon_1*, indicant que l'assistent només ha de respondre una única vegada.

- **True:** continua el flux cap a *HTTP Post - Treure etiqueta i reiniciar limit + comptador1*.
- **False:** continua cap a *If - respon X vegades*.

If - respon X vegades Aquest node comprova si la conversa té activa l'etiqueta *assistent_activat_respon_x*, corresponent al mode de múltiples respostes automàtiques.

- **True:** continua cap a *HTTP Post - Augmentar comptador respostes*.
- **False:** continua el flux cap a *HTTP Post - Treure etiqueta i reiniciar limit + comptador1*.

HTTP Post - Augmentar comptador respostes Aquest node incrementa en una unitat el comptador de respostes automàtiques enviades durant la conversa. A continuació, el flux segueix cap a *If - limit_respostes == comptador_respostes*.

If - limit_respostes == comptador_respostes Aquest node comprova si el nombre de respostes enviades ha arribat al límit màxim configurat per a la conversa.

- **True:** continua el flux cap a *HTTP Post - Treure etiqueta i reiniciar limit + comptador1*.
- **False:** el flux finalitza i l'assistent continua actiu per futures interaccions.

HTTP Post - Treure etiqueta i reiniciar limit + comptador1 Aquest node elimina l'activació de l'assistent virtual assignant el valor *Sense assignar* i reinicia tant el límit com el comptador de respostes automàtiques a 0. Finalment, el flux es dona per finalitzat.

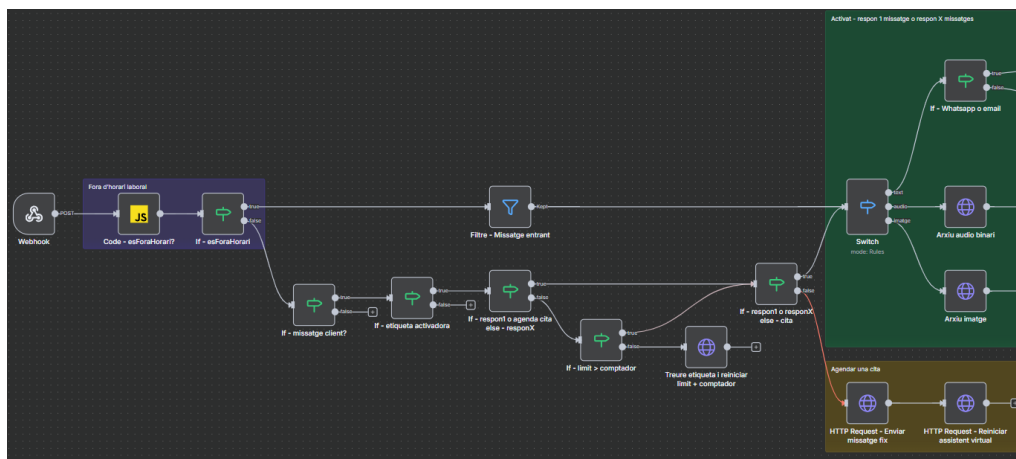


FIGURA L.1: Primera part del flux resultant del primer cas d'ús: Atenció al client

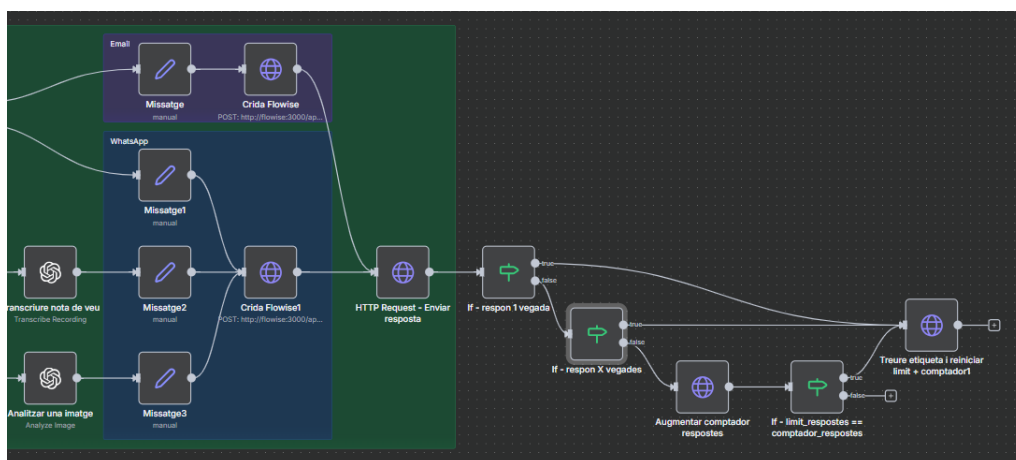


FIGURA L.2: Segona part del flux resultant del primer cas d'ús: Atenció al client

Per consultar l'estructura completa del flux en format imatge, podeu accedir al repositori del projecte a GitHub (Per veure en la màxima qualitat, cal descarregar l'arxiu):

https://github.com/alexgcl/automatitzacions-tfg/tree/main/fluxos/imatges_fluxos_n8n/flux_cas_us_1_atencio_client.pdf.

Apèndix M

Creació del bot de Telegram

El primer pas en el desenvolupament del sistema d'assistència cognitiva va consistir en la creació d'un bot de Telegram, que actua com a interfície principal de comunicació amb l'usuari.

Per a la seva configuració, es va utilitzar el bot oficial *BotFather*, proporcionat per la pròpia plataforma de Telegram. El procés s'inicia mitjançant la comanda `/start`, que permet establir la comunicació amb el servei. A continuació, s'executa la comanda `/newbot`, a través de la qual es crea un nou bot especificant-ne el nom i el nom d'usuari.

Un cop completat aquest procés, *BotFather* genera un token d'accés únic, que actua com a credencial per autenticar les peticions a l'API de Telegram. Aquest token és essencial per permetre la integració del bot amb serveis externs. Finalment, aquestes credencials es van configurar a n8n mitjançant el sistema de gestió de credencials de la plataforma.

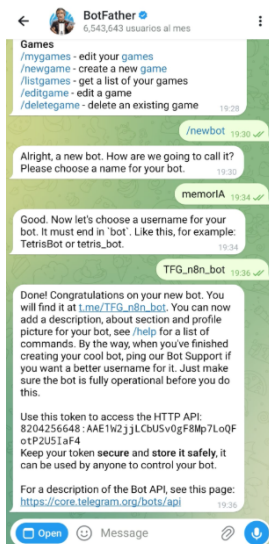


FIGURA M.1: Procés de configuració del bot de Telegram.

Apèndix N

Descripció del flux del segon cas d'ús

Telegram Trigger Aquest node inicia automàticament el flux quan es rep qualsevol esdeveniment des de Telegram, ja sigui un missatge nou (*Message*), un missatge editat (*Edited Message*) o una interacció amb un botó (*Callback Query*).

If - /start Aquest node comprova si el missatge rebut correspon a la comanda */start*, indicant que l'usuari ha iniciat una conversa amb el bot.

- **True:** el flux continua cap a *AirTable Create Record - Guardar nou usuari*.
- **False:** el flux continua cap a *Set - Edit Fields*.

AirTable Create Record - Guardar nou usuari Aquest node crea un nou registre a la taula *Usuari* amb la informació bàsica de l'usuari que ha iniciat la conversa.

Set - Edit Fields Aquest node reorganitza i renombra els camps rebuts des del *Telegram Trigger* per facilitar-ne l'ús durant la resta del flux.

If - location Aquest node comprova si l'esdeveniment rebut correspon a una compartició d'ubicació o bé a un missatge convencional.

- **True:** el flux continua cap a *If - edited message*.
- **False:** el flux continua cap a *If - event callback*.

If - edited message Aquest node verifica si la ubicació rebuda correspon a una actualització d'una ubicació en temps real o si és la primera ubicació compartida.

- **True:** el flux continua cap a *AirTable Search Record - Cerca ultima ubicació guardada*.
- **False:** el flux continua cap a *HTTP Get - Geoapify - Obtenir ubicació concreta*.

AirTable Search Record - Cerca ultima ubicació guardada Aquest node recupera l'última ubicació guardada a la taula *Historial ubicacions*. Segueix cap a *Code - 10min?*.

Code - 10min? Aquest node calcula si han transcorregut almenys 10 minuts des de l'última ubicació guardada respecte a la nova ubicació rebuda.

If - han passat 10min Aquest node valida el resultat del càlcul temporal realitzat anteriorment.

- **True:** el flux continua cap a *HTTP Get - Geoapify - Obtenir ubicació concreta*.
- **False:** el flux es deté per evitar emmagatzemar ubicacions massa freqüents.

HTTP Get - Geoapify - Obtenir ubicació concreta Aquest node utilitza el servei Geoapify per convertir les coordenades geogràfiques rebudes (*lat* i *lon*) en una ubicació interpretable i detallada. Segueix cap a *AirTable Create Record - Afegir nova ubicació*.

AirTable Create Record - Afegir nova ubicació Aquest node desa la nova ubicació a la taula *Historial ubicacions* juntament amb la informació obtinguda anteriorment. Finalment, el flux es dona per finalitzat.

If - event callback Aquest node comprova si l'esdeveniment rebut correspon a una interacció amb un botó interactiu (*callback query*).

- **True:** el flux continua cap a *If /agenda_avui; else /les_meves_ubicacions*.
- **False:** el flux continua cap a *Airtable Create Record - Guardar missatge historial (client)*.

Airtable Create Record - Guardar missatge historial (client) Aquest node desa el missatge enviat pel client a la taula *Historial missatges* per mantenir un registre de la conversa. A continuació, el flux continua cap a *If - comanda*.

If - comanda Aquest node comprova si el missatge rebut correspon a una comanda, és a dir, si comença amb el caràcter `"/`.

- **True:** el flux continua cap a *Set - comanda_final*.
- **False:** el flux continua cap a *OpenAI - Classificar missatge usuari*.

Set - comanda_final Aquest node desa la comanda rebuda en un camp comú anomenat *comanda_final*, independentment de l'origen del missatge. El flux continua cap al node *Switch*.

OpenAI - Classificar missatge usuari Aquest node utilitza OpenAI per analitzar el contingut del missatge i identificar la intenció de l'usuari. El model classifica el missatge en una de les comandes disponibles del sistema. El flux continua cap a *Set - comanda_final1*.

Set - comanda_final1 Aquest node desa el resultat de la classificació automàtica en el camp comú *comanda_final*. Posteriorment, el flux continua cap al node *Switch*.

Switch Aquest node rep el valor de *comanda_final* i divideix el flux segons la funcionalitat seleccionada:

- **/les_meves_ubicacions:** continua cap a *Preguntar: afegir o llistar*.
- **/historial_ubicacions:** continua cap a *Airtable Search Records - Cerca historial ubicacions 24h*.
- **/resum_matinal:** continua cap a *Gmail - Get many messages*.
- **/medicació:** continua cap a *AirTable Search Records - Cerca llistat medicacio*.
- **/agenda_avui:** continua cap a *Google Calendar - Obtenir events d'avui*.
- **/resum_ahir:** continua cap a *Airtable Search Records - Cercar agenda ahir*.
- **/exercici_cognitiu:** continua cap a *HTTP Get - Api retorna preguntes*.

Preguntar: afegir o llistar Aquest node envia un missatge interactiu amb botons (*inline keyboard*) perquè l'usuari pugui escollir entre afegir una nova ubicació o consultar les ubicacions existents. El flux queda en espera de la resposta de l'usuari.

If agenda_avui; else les_meves_ubicacions Aquest node comprova si el missatge rebut correspon a la comanda */agenda_avui* o */les_meves_ubicacions*.

- **/agenda_avui:** continua cap a *AirTable Update Record - Actualitzar tasca completada*.
- **/les_meves_ubicacions:** continua cap a *If afegir nova ubi; else llistar ubis*.

If afegir nova ubi; else llistar ubis Aquest node identifica l'acció seleccionada per l'usuari a partir del valor rebut al *callback query*.

- **Afegir nova ubi:** continua cap a *AirTable Search Record - Cerca totes les ubis importants*.
- **Llistar ubis:** continua cap a *Telegram - Send message and wait for response*.

AirTable Search Record - Cerca totes les ubis importants Aquest node recupera totes les ubicacions importants associades a l'usuari des de la taula *Ubicacions importants*. Posteriorment, el flux continua cap al node *Aggregate*.

Aggregate Aquest node agrupa totes les ubicacions recuperades en un únic element per facilitar-ne el tractament posterior.

Airtable Search Records - Cerca ultima ubicacio registrada Aquest node recupera l'última ubicació registrada de l'usuari des de la taula *Historial ubicacions*.

Code - Missatge dinamic per ubisImportants Aquest node genera dinàmicament un missatge amb les ubicacions importants de l'usuari i crea, per a cadascuna, un enllaç de Google Maps amb la ruta des de la ubicació actual (última ubicació registrada) fins a la destinació seleccionada.

Telegram - Send a text message Aquest node envia al client el missatge generat amb les ubicacions importants i els enllaços de navegació corresponents. El flux finalitza.

Telegram - Send message and wait for response Aquest node envia un missatge amb un enllaç a un formulari perquè l'usuari pugui introduir una nova ubicació d'interès indicant-ne el nom i l'adreça. El flux queda en espera fins que el formulari és completat. Segueix cap a *HTTP Get - Geoapify - Obtenir ubicació concreta1*.

HTTP Get - Geoapify - Obtenir ubicació concreta1 Aquest node utilitza el servei Geoapify per convertir l'adreça introduïda al formulari en coordenades geogràfiques (lat i lon). Segueix cap a *AirTable Create Record - Afegir nova ubicació important*.

AirTable Create Record - Afegir nova ubicació important Aquest node desa la nova ubicació important a la taula *Ubicacions importants* juntament amb les coordenades obtingudes anteriorment. El flux segueix cap a *Telegram - Send a text message1*.

Telegram - Send a text message1 Aquest node envia un missatge de confirmació indicant que la nova ubicació s'ha afegit correctament. Finalment, el flux es dona per finalitzat.

Airtable Search Records - Cerca historial ubicacions 24h Aquest node recupera totes les ubicacions registrades durant les últimes 24 hores des de la taula *Historial ubicacions*. El flux continua cap a *Aggregate1*.

Aggregate1 Aquest node agrupa totes les ubicacions recuperades en un únic element conservant únicament els camps *Data i Hora* i *Ubicació*, facilitant així el seu tractament posterior.

Code - Netejar input Aquest node processa i ordena cronològicament les ubicacions recuperades. Posteriorment, agrupa ubicacions consecutives iguals per generar una línia temporal simplificada amb l'hora d'inici i final de cada ubicació visitada.

Airtable Search Records - Cerca totes les ubis importants1 Aquest node recupera totes les ubicacions importants definides per l'usuari a la taula *Ubicacions importants*. El flux continua cap a *Aggregate2*.

Aggregate2 Aquest node agrupa totes les ubicacions importants recuperades en un únic element per facilitar-ne l'ús durant la generació del resum.

OpenAI - Interpretar recorregut Aquest node utilitza OpenAI per interpretar la línia temporal de les ubicacions registrades i generar un resum natural del recorregut de l'usuari durant el dia. El model té en compte tant els intervals temporals com les ubicacions importants definides per l'usuari per generar descripcions més naturals i entenedores.

Telegram - Send a text message2 Aquest node envia al client el resum textual del recorregut generat pel model d'OpenAI.

Airtable Create Record - Guardar missatge historial (BOT) Aquest node desa el resum enviat pel bot a la taula *Historial missatges* per mantenir un registre de les respostes generades automàticament.

HTTP Get - Generar imatge mapa recorregut Aquest node genera dinàmicament una imatge del recorregut realitzat per l'usuari durant el dia utilitzant el servei de mapes estàtics de Yandex. La ruta es construeix a partir de totes les coordenades registrades a l'historial d'ubicacions.

Telegram - Send a photo message Aquest node envia al client la imatge del mapa amb el recorregut generat anteriorment. Finalment, el flux es dona per finalitzat.

Gmail - Get many messages Aquest node recupera tots els correus electrònics rebuts durant les últimes 24 hores del compte de l'usuari. El flux continua cap a *Aggregate3*.

Aggregate3 Aquest node agrupa tots els correus recuperats en un únic element per facilitar-ne el tractament posterior.

OpenAI - Interpretar llistat correus Aquest node utilitza OpenAI per interpretar el contingut dels correus electrònics i generar un resum natural i entenedor de l'activitat recent de l'usuari. El model analitza l'assumpte, el remitent i el contingut resumit de cada correu per destacar la informació més rellevant i ignorar notificacions poc importants o publicitat.

Telegram - Send a text message3 Aquest node envia al client el resum textual dels correus electrònics generat pel model d'OpenAI.

Airtable Create Record - Guardar missatge historial (BOT)1 Aquest node desa el resum generat pel bot a la taula *Historial missatges* per mantenir un registre de les respostes enviades.

AirTable Search Records - Cerca historial missatges Aquest node recupera totes les converses mantingudes amb el bot de Telegram durant les últimes 24 hores des de la taula *Historial missatges*. El flux continua cap a *Aggregate4*.

Aggregate4 Aquest node agrupa tots els missatges recuperats en un únic element per facilitar-ne el tractament posterior.

OpenAI - Interpretar historial missatges Aquest node utilitza OpenAI per interpretar l'historial de converses entre el client i el bot de Telegram. El model genera un resum cronològic i natural de les accions i consultes realitzades per l'usuari durant les últimes 24 hores, interpretant tant missatges normals com les diferents comandes executades.

Telegram - Send a text message4 Aquest node envia al client el resum textual de la conversa generat pel model d'OpenAI. Finalment, el flux es dona per finalitzat.

Airtable Search Record - Cerca llistat medicació Aquest node recupera totes les medicacions registrades de l'usuari des de la taula *Medicació*. El flux continua cap a *Code - Resum llistat medicació*.

Code - Resum llistat medicació Aquest node processa totes les medicacions recuperades i genera un resum estructurat amb el nom del medicament, la freqüència configurada, l'hora de presa i les notes addicionals. El resultat es prepara en format compatible amb Markdown per poder-se enviar correctament a Telegram.

Telegram - Send a text message5 Aquest node envia al client el resum complet de la medicació configurada. Posteriorment, el flux continua cap a *Code - Calcul hora proxima medicació*.

Code - Calcul hora proxima medicació Aquest node calcula quines medicacions encara resten pendents de prendre durant el dia actual. El sistema té en compte la freqüència configurada per cada medicament, ja sigui diària, en dies específics o cada determinat nombre d'hores, i calcula la pròxima hora de presa corresponent.

Telegram - Send a text message6 Aquest node envia al client el resum de les pròximes medicacions pendents amb l'hora corresponent de cada presa. Finalment, el flux es dona per finalitzat.

Google Calendar - Obtenir events d'avui Aquest node recupera tots els esdeveniments programats al Google Calendar de l'usuari per al dia actual. El flux continua cap a *Airtable Create or Update Record - Guardem / actualitzem els events d'avui*.

Airtable Create or Update Record - Guardem / actualitzem els events d'avui Aquest node crea o actualitza automàticament els registres de la taula *Agenda* a partir dels esdeveniments recuperats de Google Calendar, mantenint sincronitzada la informació del calendari amb la base de dades del sistema.

Set - Tasques creades Aquest node reorganitza i prepara els camps principals dels esdeveniments recuperats, com ara l'identificador, el nom de la tasca, la descripció i l'interval horari, per facilitar-ne el tractament posterior.

Aggregate5 Aquest node agrupa tots els esdeveniments del calendari en un únic element per simplificar-ne la gestió posterior.

Airtable Search Record - Cercar tasques NO completades Aquest node recupera de la taula *Agenda* totes les tasques del dia actual que encara no han estat marcades com a completades.

Aggregate6 Aquest node agrupa totes les tasques pendents recuperades en un únic element per facilitar-ne el tractament.

Telegram - Send a text message7 Aquest node envia al client un resum dels esdeveniments programats per al dia actual que encara no han estat completats, indicant per a cadascun el nom, l'horari i la descripció.

Airtable Create Record - Guardar missatge historial (BOT)3 Aquest node desa el missatge enviat pel bot a la taula *Historial missatges* per mantenir un registre de les notificacions generades.

HTTP Get - Llistat tasques NO completades Aquest node envia un segon missatge interactiu amb un botó (*inline keyboard*) per a cada tasca pendent. Quan l'usuari selecciona un botó, es genera un esdeveniment *callback_query* que permet identificar la tasca completada.

AirTable Update Record - Actualitzar tasca completada Aquest node actualitza l'estat de la tasca seleccionada a la taula *Agenda*, canviant-la de no completada a completada. Posteriorment, el flux continua cap a *Airtable Search Record - Tasques NO completades1*.

Airtable Search Record - Tasques NO completades1 Aquest node torna a recuperar les tasques pendents del dia actual després de l'actualització, excloent la tasca que s'acaba de marcar com a completada.

Aggregate7 Aquest node agrupa novament totes les tasques pendents actualitzades en un únic element.

HTTP Get - Tasques no completades actualitzades Aquest node envia un nou missatge interactiu amb el llistat actualitzat de tasques encara pendents, mostrant únicament les que continuen sense completar-se. Finalment, el flux es dona per finalitzat.

Airtable Search Record - Cercar agenda ahir Aquest node recupera totes les tasques i esdeveniments registrats a la taula *Agenda* corresponents al dia anterior. El flux continua cap a *Aggregate8*.

Aggregate8 Aquest node agrupa tots els registres recuperats en un únic element per facilitar-ne el tractament posterior.

Telegram - Send a text message8 Aquest node envia al client un resum de les tasques programades el dia anterior, mostrant el nom de cada esdeveniment i el seu estat de completitud.

AirTable Search Records - Cerca historial missatges1 Aquest node recupera tots els missatges intercanviats amb el bot durant el dia anterior des de la taula *Historial missatges*. El flux continua cap a *Aggregate9*.

Aggregate9 Aquest node agrupa tots els missatges recuperats en un únic element per facilitar-ne el processament posterior.

OpenAI - Interpretar historial missatges1 Aquest node utilitza OpenAI per interpretar l'historial de missatges del dia anterior i generar un resum cronològic i natural de les interaccions de l'usuari amb el bot. El model interpreta tant missatges convencionals com les diferents comandes executades durant el dia.

Telegram - Send a text message⁹ Aquest node envia al client el resum textual de les interaccions mantingudes amb el bot durant el dia anterior. Finalment, el flux es dona per finalitzat.

HTTP Get - Api retorna preguntes Aquest node consulta l'API *Open Trivia Database* per recuperar tres preguntes de tipus vertader o fals relacionades amb coneixement general.

Telegram - Envia missatge i espera resposta formulari Aquest node envia al client un formulari interactiu amb les preguntes recuperades de l'API i espera les respostes de l'usuari. Cada pregunta només permet seleccionar una resposta entre *True* o *False*. El flux queda en espera fins que l'usuari respon o bé fins que transcorren 10 minuts.

Code - Renombrar preguntes Aquest node reorganitza les respostes rebudes del formulari i renombra automàticament els camps amb el format *Pregunta 1*, *Pregunta 2* i *Pregunta 3* per facilitar-ne el tractament posterior.

Code - Recompte puntuació Aquest node compara les respostes proporcionades pel client amb les respostes correctes retornades per l'API i calcula la puntuació total obtinguda en l'exercici cognitiu.

Airtable Create Record - Guardar puntuació assolida Aquest node desa la puntuació obtinguda pel client a la taula *Exercicis* per mantenir un historial dels resultats assolits.

Telegram - Send a text message¹⁰ Aquest node envia al client la puntuació aconseguida en l'exercici cognitiu.

AirTable Search Records - Cercar historial de totes les puntuacions Aquest node recupera totes les puntuacions històriques registrades del client a la taula *Exercicis*. El flux continua cap a *Aggregate¹⁰*.

Aggregate¹⁰ Aquest node agrupa totes les puntuacions recuperades en un únic element per facilitar-ne la presentació final.

Telegram - Send a text message¹¹ Aquest node envia al client el llistat complet de puntuacions obtingudes anteriorment perquè pugui consultar l'evolució i el progrés històric dels exercicis cognitius realitzats.

Per consultar l'estructura completa del flux en format imatge, podeu accedir al repositori del projecte a GitHub (Per veure en la màxima qualitat, cal descarregar l'arxiu):

https://github.com/alexgcl/automatitzacions-tfg/tree/main/fluxos/imatges_fluxos_n8n/flux_cas_us_2_assistencia_cognitiva.pdf.

Apèndix O

Descripció del flux del segon cas d'ús: Neteja de dades

Per evitar l'acumulació d'informació innecessària i mantenir el rendiment del sistema, s'ha implementat un flux automàtic de neteja de dades executat periòdicament cada 12 hores mitjançant un *Schedule Trigger*.

En primer lloc, el sistema elimina els registres antics de la taula *Historial ubicacions*, conservant únicament les ubicacions recents de les últimes 24 hores. De manera similar, també es realitza la neteja periòdica de la taula *Historial missatges*, eliminant les converses antigues per reduir el volum de dades emmagatzemades.

Pel que fa a la taula *Agenda*, s'aplica una política diferent: només s'eliminen els esdeveniments anteriors al dia d'ahir. Això permet mantenir disponibles tant les tasques del dia actual com les del dia anterior, necessàries per al correcte funcionament de funcionalitats com */agenda_avui* i */resum_ahir*.

La resta de taules no requereixen processos de neteja específics, ja que contenen dades persistents o un volum reduït d'informació.

O.1 Implementació del flux

A continuació, es detalla la configuració de cadascun dels nodes que formen el flux de neteja automàtica de dades.

Schedule Trigger Aquest node és l'encarregat d'iniciar automàticament el flux de manera periòdica. La seva configuració es basa en una regla (*Trigger rules*):

- **Trigger Interval: Hours** Defineix que l'execució del flux es realitza segons un interval d'hores determinat.
- **Hours between Triggers: 12** Estableix que el flux s'executa automàticament cada 12 hores.
- **Trigger at minute: 0** Indica que l'execució es produeix exactament al minut 0 de l'hora corresponent.

AirTable Search Records - Cerca ubicacions amb més de 24 h Aquest node consulta la taula *Historial ubicacions* per identificar aquells registres amb una antiguitat superior a 24 hores. Per fer-ho, s'utilitza la següent fórmula de filtratge:

```
IS_BEFORE({Data i Hora}, DATEADD(NOW(), -24, 'hours'))
```

D'aquesta manera, només es recuperen les ubicacions antigues que han de ser eliminades.

AirTable Delete Records - Eliminar ubicacions trobades Aquest node elimina els registres obtinguts en el node anterior de la taula *Historial ubicacions*. L'eliminació es realitza individualment per a cada registre utilitzant el seu identificador únic:

```
{{ $json.id }}
```

Code - Netejar input1 Després del procés d'eliminació, el node anterior retorna tants elements com registres s'han eliminat. Això provocaria que el següent node del flux s'executés múltiples vegades de forma innecessària.

Per evitar aquest comportament, s'utilitza un node de codi amb la següent instrucció:

```
return [{}];
```

Aquest codi força la sortida d'un únic element buit, garantint que el següent node només s'executi una sola vegada.

AirTable Search Records - Cerca missatges amb més de 24 h Aquest node consulta la taula *Historial missatges* per localitzar els missatges amb una antiguitat superior a 24 hores. La cerca es realitza mitjançant la fórmula següent:

```
IS_BEFORE({Data i Hora}, DATEADD(NOW(), -24, 'hours'))
```

AirTable Delete Records - Eliminar missatges trobats Aquest node elimina els registres recuperats en el node anterior de la taula *Historial missatges*. Cada registre s'elimina utilitzant el seu identificador:

```
{{ $json.id }}
```

Code - Netejar input Aquest node aplica exactament la mateixa lògica descrita anteriorment per evitar execucions múltiples innecessàries del següent node del flux. El codi utilitzat és:

```
return [{}];
```

AirTable Search Records - Cerca esdeveniments anteriors al dia d'ahir Aquest node consulta la taula *Agenda* per identificar aquells esdeveniments anteriors al dia d'ahir. La condició de filtratge utilitzada és la següent:

```
IS_BEFORE({Data}, DATEADD(TODAY(), -1, 'days'))
```

Aquesta configuració permet conservar els esdeveniments del dia actual i del dia anterior, necessaris per al correcte funcionament de determinades funcionalitats del sistema.

AirTable Delete Records - Eliminar esdeveniments trobats Finalment, aquest node elimina de la taula *Agenda* tots els esdeveniments recuperats en el node anterior, utilitzant l'identificador de cada registre:

```
{{ $json.id }}
```

Flux resultant:



FIGURA O.1: Flux resultant per la neteja de dades de les taules.

Apèndix P

Descripció del flux del segon cas d'ús: Recordatoris automàtics

A més de les funcionalitats activades manualment mitjançant comandes, el sistema incorpora un tercer flux automatitzat de recordatoris que permet executar accions de manera periòdica sense intervenció directa de l'usuari. Aquest flux es basa en un node de tipus *Schedule Trigger*, configurat amb dues regles principals.

La primera regla indica que el flux s'executa diàriament a les 8:30 del matí. En aquest cas, consisteix en l'enviament automàtic d'un resum matinal cada dia. Aquest resum segueix la mateixa lògica de la comanda */resum_matinal*, recopilant informació sobre els correus electrònics i la interacció recent amb el bot.

Per poder enviar aquestes notificacions de manera proactiva, el sistema utilitza el *xat id* de Telegram emmagatzemat prèviament a la taula *Usuari*.

La segona funcionalitat correspon als recordatoris de medicació. El flux s'executa periòdicament cada 30 minuts i consulta la taula *Medicació* per comprovar quines preses s'han de notificar segons la configuració definida per cada medicament.

Quan es detecta una coincidència amb l'horari configurat, el sistema envia automàticament una notificació a l'usuari amb la informació corresponent del medicament i la presa pendent.

P.1 Implementació del flux

A continuació, es detalla la configuració dels nodes que formen el flux de recordatoris automàtics.

Schedule Trigger Aquest node inicia automàticament el flux segons dues regles diferents:

- Una execució diària a les 8:30 del matí mitjançant una expressió *Cron*.
- Una execució periòdica cada 30 minuts per comprovar possibles recordatoris de medicació.

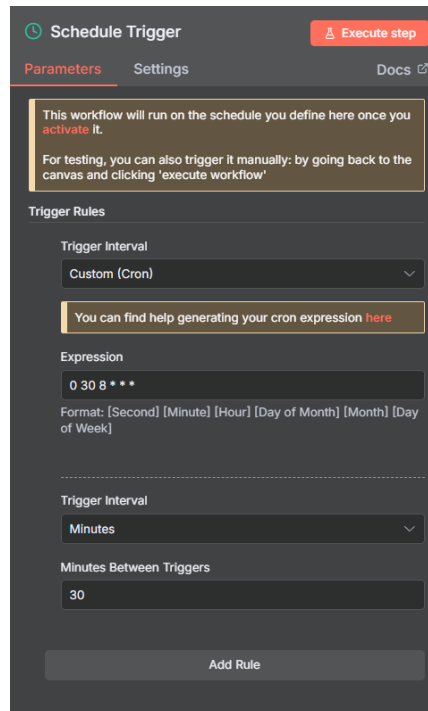


FIGURA P.1: Configuració del node *Schedule Trigger* per a l'execució del flux de recordatoris automàtics.

If - 8:30am Aquest node comprova si l'hora actual correspon exactament a les 8:30 del matí. En funció del resultat, el flux segueix dues branques diferents:

- **True:** execució del resum matinal.

Gmail - Get many messages Aquest node recupera els correus electrònics rebuts o enviats durant les últimes 24 hores. Només s'executa en la branca del resum matinal.

Aggregate El node de Gmail retorna múltiples elements independents. Mitjançant aquest node, tots els correus es combinen en un únic element agrupant únicament els camps *From*, *Subject* i *snippet* dins d'una llista.

OpenAI - Interpretar llistat correus Aquest node utilitza OpenAI per generar un resum natural i entenedor dels correus electrònics recuperats. El resum es genera en català a partir de la informació agregada al node anterior.

AirTable Search Records - Cerca xat id Aquest node consulta la taula *Usuari* per recuperar el *xat id* de Telegram necessari per enviar el resum automàticament a l'usuari.

Telegram - Send a text message Aquest node envia al Telegram de l'usuari el resum dels correus generat prèviament amb OpenAI.

AirTable Create Record - Guardar missatge historial (BOT) Després d'enviar el resum, aquest node desa el missatge generat pel bot dins la taula *Historial missatges* amb el rol *Bot*.

AirTable Search Records - Cerca historial missatges Aquest node recupera tots els missatges intercanviats amb el bot durant les últimes 24 hores.

Aggregate1 Aquest node agrupa els missatges recuperats en un únic element, conservant únicament els camps *Rol*, *Missatge* i *Data i Hora*.

OpenAI - Interpretar historial missatges Aquest node utilitza OpenAI per generar un resum natural de la interacció de l'usuari amb el bot durant les últimes 24 hores, interpretant tant missatges normals com les diferents comandes executades.

Telegram - Send a text message1 Aquest node envia al Telegram de l'usuari el resum de l'activitat i interacció amb el bot generat en el node anterior.

- **False:** comprovació dels recordatoris de medicació.

AirTable Search Records - Cerca llistat medicació En la branca corresponent als recordatoris de medicació, aquest node recupera tots els medicaments registrats a la taula *Medicació*.

Code - Calcula si toca enviar recordatori Aquest node comprova si cal enviar algun recordatori de medicació segons la configuració de cada registre. El sistema contempla tres tipus de freqüència diferents:

- **Diari:** el recordatori s'envia cada dia a una hora concreta.
- **Dies específics:** el recordatori només s'envia en determinats dies de la setmana configurats per l'usuari.
- **Cada X hores:** el sistema calcula periòdicament si ha transcorregut l'interval d'hores definit des de l'hora inicial configurada.

A més, s'aplica un marge de tolerància de cinc minuts per evitar problemes derivats de petites diferències temporals en l'execució del flux.

Quan una medicació compleix les condicions establertes, el node prepara la informació necessària per generar el recordatori corresponent.

AirTable Search Records - Cerca xat id1 Aquest node recupera el *xat id* de Telegram de l'usuari per poder enviar la notificació de medicació.

Telegram - Send a text message2 Finalment, aquest node envia automàticament el recordatori de medicació a l'usuari mitjançant Telegram, indicant el nom del medicament, l'hora corresponent i les notes addicionals configurades.

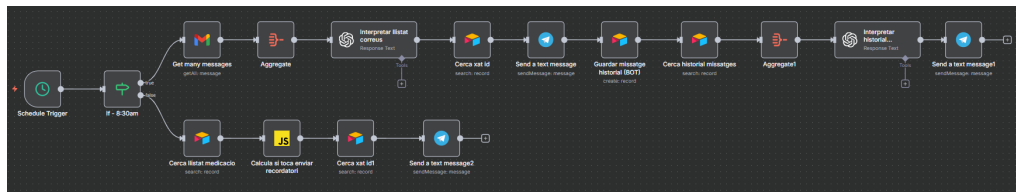


FIGURA P.2: Flux de recordatoris automàtics sobre la medicació i de resum de correus i interaccions amb el bot.

Apèndix Q

Preparació de l'entorn del servidor

Un cop establerta la connexió SSH amb la màquina virtual, s'ha procedit a la instal·lació i configuració de Docker i Docker Compose mitjançant els següents passos:

1. **Instal·lació de dependències:** S'ha realitzat una actualització de la llista de paquets del sistema per garantir la compatibilitat i s'han instal·lat les eines bàsiques necessàries (*curl* i *git*) per a la descàrrega de l'script i el posterior clonatge del projecte.

```
sudo apt update & sudo apt install -y curl git
```

2. **Instal·lació automatitzada de Docker:** Per simplificar el procés i assegurar la compatibilitat amb la distribució d'Ubuntu de la màquina virtual, s'ha utilitzat l'*script* d'instal·lació oficial de Docker. Aquest mètode automatitza la configuració dels repositoris i la instal·lació de l'última versió estable del motor de contenització.

```
curl -fsSL https://get.docker.com -o get-docker.sh
```

```
sudo sh get-docker.sh
```

3. **Activació i verificació del servei:** S'ha iniciat el servei de Docker i s'ha configurat perquè el sistema l'arrenqui automàticament cada vegada que s'encengui la màquina virtual. Addicionalment, s'ha comprovat que el servei estigui actiu i funcionant correctament.

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo systemctl status docker
```

4. **Gestió de permisos de l'usuari:** Per motius de seguretat i per agilitzar l'administració, s'ha afegit l'usuari del sistema al grup Docker. Aquesta configuració és fonamental per poder gestionar els contenidors i executar les comandes de Docker sense haver d'utilitzar constantment privilegis de superusuari (*sudo*).

```
sudo usermod -aG docker ${USER}
```

Apèndix R

Clonatge del repositori

Un cop l'entorn de la màquina virtual ha estat preparat amb Docker, s'ha procedit a la transferència del codi i a l'activació de tota l'arquitectura del sistema.

1. **Clonatge del repositori i accés al directori:** S'ha descarregat el codi font del projecte des del repositori privat de GitHub.

```
git clone https://github.com/alexgcl/automatitzacions-tfg.git
```

```
cd automatitzacions-tfg/ # Accés al directori del projecte
```

2. **Configuració de les variables d'entorn reals:** Dins del servidor, s'han creat els fitxers `.env` definitius a partir de les plantilles d'exemple (`.env.example`). En aquest punt, s'han introduït les contrasenyes de base de dades que s'utilitzaran en l'entorn de producció, garantint que aquesta informació sensible mai surti del servidor.

```
cp .env.example .env # Creació del fitxer d'entorn a partir de la plantilla
```

Apèndix S

Configuració dels fitxers de producció

En aquest apartat es detallen les modificacions realitzades en els fitxers de configuració per fer el salt de l'entorn de desenvolupament local a l'entorn de producció a Azure. Aquest procés ha implicat no només l'actualització de credencials, sinó una reestructuració de l'arquitectura per garantir la seguretat i l'estabilitat del sistema.

S.1 Variables d'entorn (.env)

La transició a producció ha requerit una revisió exhaustiva de les variables d'entorn per adaptar-les a dominis reals i xarxes xifrades.

S.1.1 Ajustos a Chatwoot

- **SECRET_KEY_BASE:** S'ha generat un nou codi hexadecimal aleatori per garantir la integritat de les sessions.
- **FRONTEND_URL:** S'ha substituït l'adreça local (<http://nginx:8081>) pel domini final amb xifratge (<https://chatwoot.automatitzacions.dev>).
- **Seguretat a Redis:** En local, la connexió es feia sense credencials. En producció, s'ha implementat l'ús d'una contrasenya (`REDIS_PASSWORD`) i s'ha ajustat la URI de connexió per incloure l'autenticació: `redis://:contrasenya@redis-chatwoot:6379`.
- **Segregació de n8n:** S'han eliminat totes les variables referents a n8n del fitxer `.env` principal per evitar redundàncies i centralitzar la seva configuració en el seu propi bloc de serveis.

S.1.2 Ajustos a Evolution API

Evolution API actua com a pont crític entre WhatsApp i la resta del sistema, per la qual cosa la seva configuració ha estat fortament blindada:

- **SERVER_URL:** S'ha passat d'una adreça interna (<http://evolution-api:8080>) a la URL pública xifrada <https://evolutionapi.automatitzacions.dev>. Aquest canvi és vital per a la recepció correcta de webhooks.

- **AUTHENTICATION_API_KEY:** S'ha substituït el valor per defecte de l'exemple per una clau única i segura, restringint l'accés a l'API només a serveis autoritzats. (API KEY generada en <https://skybootstrap.com/tools/es/uuid-generator>).
- **Connexions a Bases de Dades:**
 - Per a la seva pròpia persistència, s'ha configurat la URI postgresql://postgres:evolution@postgres-evolution:5432/evolution.
 - Per a la integració amb Chatwoot, s'ha ajustat la URI d'importació (CHATWOOT_IMPORT_DATABASE_CONNECTION_URI), passant d'apuntar a *chatwoot_production* (local) a la base de dades Chatwoot de producció, utilitzant les credencials definides en el servidor.

S.1.3 Configuració de Flowise

En el cas de Flowise, s'ha mantingut la mateixa configuració que en l'entorn local pel que fa a variables d'entorn internes (ports i rutes de dades). Això és possible gràcies al fet que l'exposició a l'exterior i el xifratge HTTPS no es gestionen des del propi servei, sinó que es deleguen completament a la capa de *Reverse Proxy* (Traefik) mitjançant etiquetes en el fitxer *docker-compose.yaml*.

S.2 Arquitectura del Docker Compose

El fitxer *docker-compose.yaml* ha patit la transformació més significativa, passant d'un model de proves a un model professional d'alta disponibilitat.

S.2.1 Professionalització d'n8n

La configuració d'n8n ha evolucionat per suportar càrregues de treball reals:

- **Seguretat de sessió:** S'ha activat `N8N_SECURE_COOKIE=true` i s'ha canviat el protocol a HTTPS. D'aquesta manera les *cookies* de sessió només es transmeten a través de connexions xifrades, evitant que puguin ser interceptades o manipulades.
- **Estabilitat (Mode Cua):** S'ha incorporat un nou servei de Redis (*redis-n8n*) amb contrasenya. Això permet gestionar les cues d'execució de forma asíncrona, evitant col·lapses davant de volums elevats de dades.
- **Control d'arrencada:** S'ha implementat un *healthcheck* al servei de PostgreSQL, assegurant que n8n no s'iniciï fins que la base de dades estigui llesta per rebre connexions.

S.2.2 Implementació tècnica de Traefik al Docker Compose

En l'entorn de producció, la configuració de xarxa s'ha descentralitzat. S'ha eliminat el servei nginx del fitxer *docker-compose.yaml* i s'ha integrat el servei Traefik, configurant cada aplicació mitjançant etiquetes de Docker.

Per a cada servei exposat a l'exterior (Chatwoot, n8n, Evolution API, Flowise i *Landing page*), s'han afegit les següents línies de configuració:

- **Activació:** `"traefik.enable=true"` per indicar al *proxy* que gestioni el contenidor.
- **Regla d'encaminament:** `"traefik.http.routers.[servei].rule=Host(domini.com)"` per assignar el subdomini corresponent.
- **Seguretat TLS:** `"traefik.http.routers.[servei].tls.certresolver=letsencrypt"` per activar la generació del certificat de seguretat.
- **Port Intern:** `"traefik.http.services.[servei].loadbalancer.server.port=[port_intern]"`, indicant a quin port del contenidor ha de dirigir Traefik el trànsit, evitant haver d'exposar ports manualment a la màquina virtual.

Apèndix T

Resultats demostració del cas d'ús 1: Atenció al client

T.1 Canal WhatsApp

Gestió de cites: *Hola, bones! Voldria demanar una cita per tallar-me les puntes i fer-me metxes la setmana que ve. Quan tindrieu un forat lliure?*

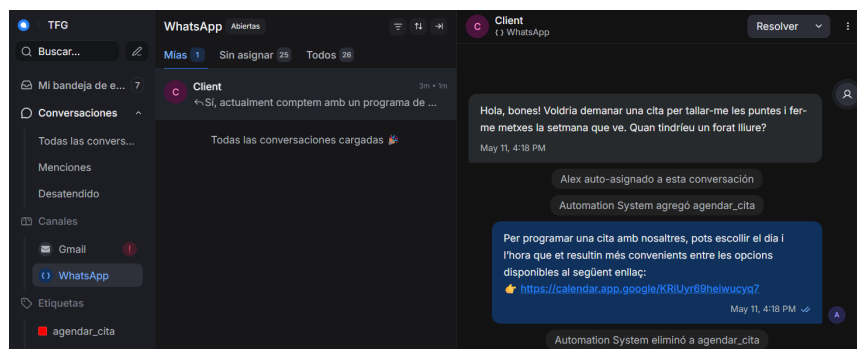


FIGURA T.1: Resultat d'execució de la tasca "Agendar cita".

Respondre a un missatge: *Teniu algun tipus de descompte o promoció vigent per a nous clients?*

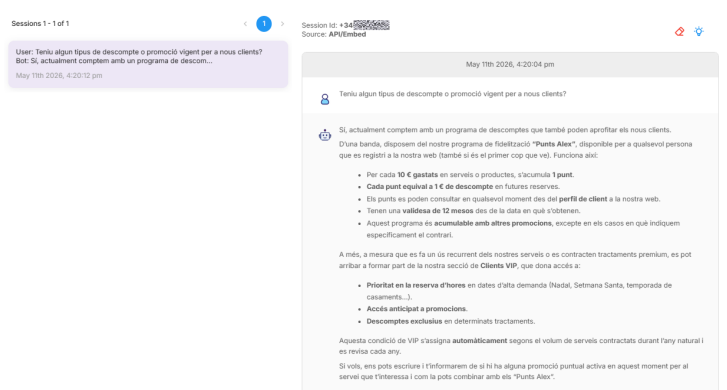


FIGURA T.2: Resultat d'execució de la tasca "Respondre 1 missatge".

Respondre a diversos missatges:

- (Àudio): *Hola, bones. Tinc algunes consultes. Em podeu ajudar?*
- *Quins són els vostres horaris d'obertura dissabte a la tarda?*

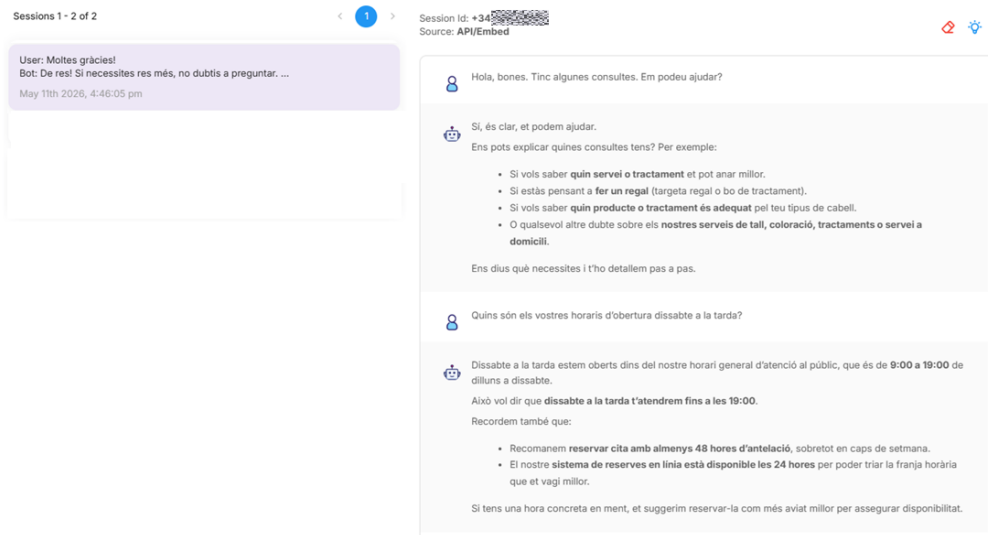


FIGURA T.3: Resultat d'execució de la tasca "Respondre X missatges".

- *I com puc posar-me en contacte amb vosaltres si m'ho vull repensar?*
- *Moltes gràcies!*

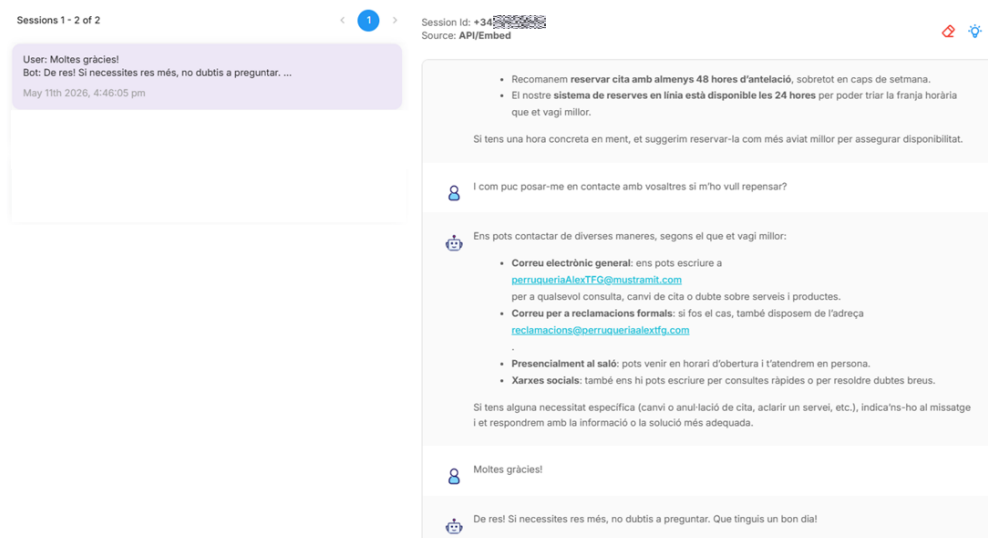


FIGURA T.4: Resultat d'execució de la tasca "Respondre X missatges".

T.2 Canal Gmail

Fora de l'horari laboral:

- *Assumpte: Consulta sobre producte capil·lar personal.*
- *Cos: Hola, em poso en contacte amb vosaltres perquè tinc un problema al cuir cabellut. El metge m'ha receptat un xampú especial i m'ha dit que no n'utilitzi cap altre. Hi hauria algun inconvenient si el porto jo el dia de la cita i feu servir aquest en comptes dels vostres?*

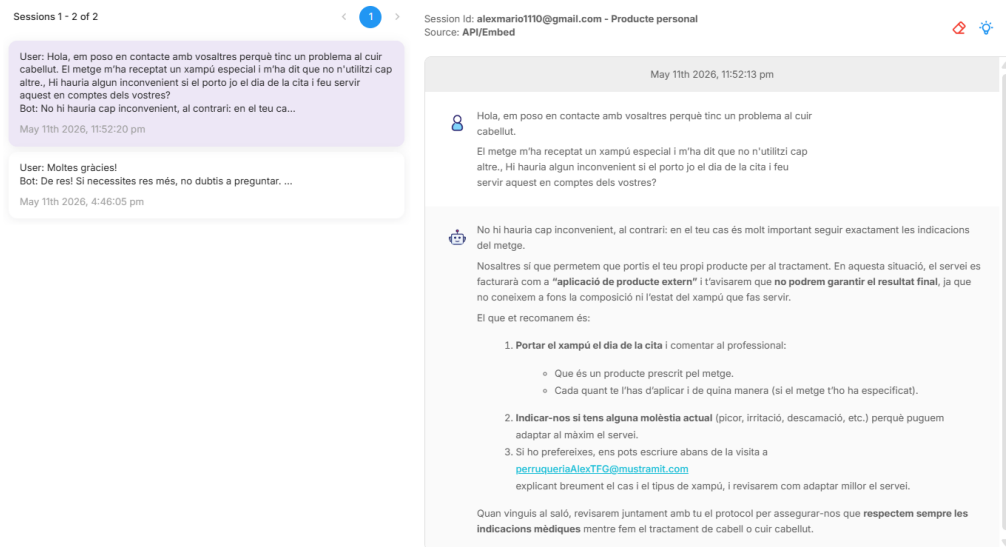


FIGURA T.5: Resultat d'execució responenent a través de Gmail fora de l'horari laboral.

Apèndix U

Resultats demostració del cas d'ús 2: Assistència cognitiva

U.1 Comanda */les_meves_ubicacions*:

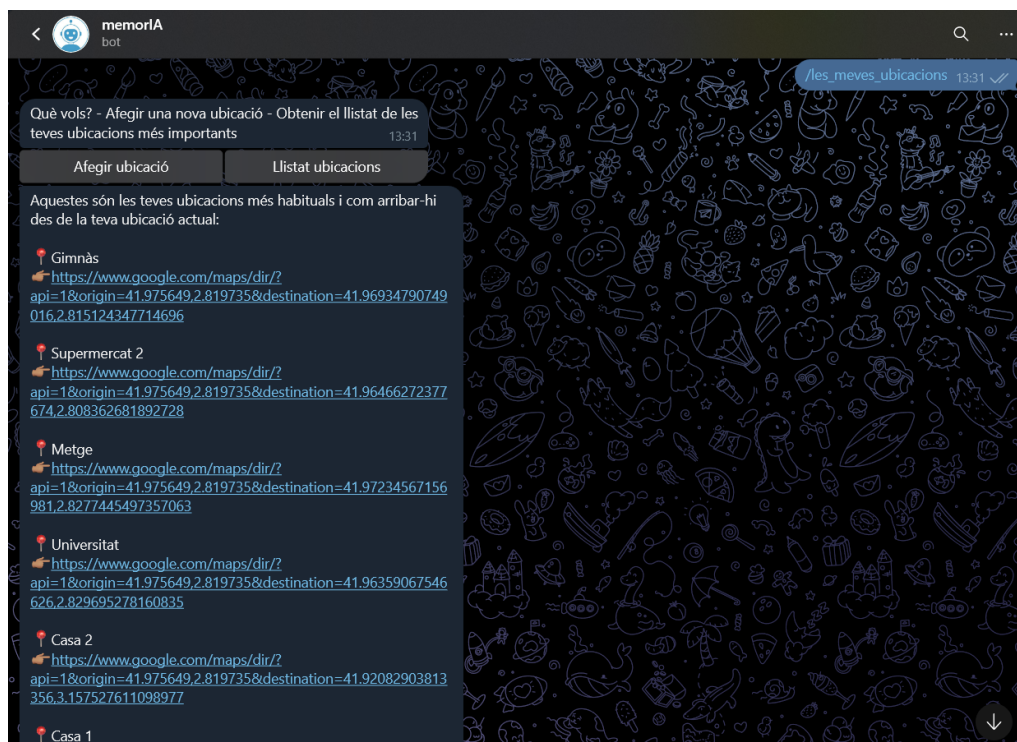


FIGURA U.1: Resposta llistant les ubicacions importants de l'usuari.

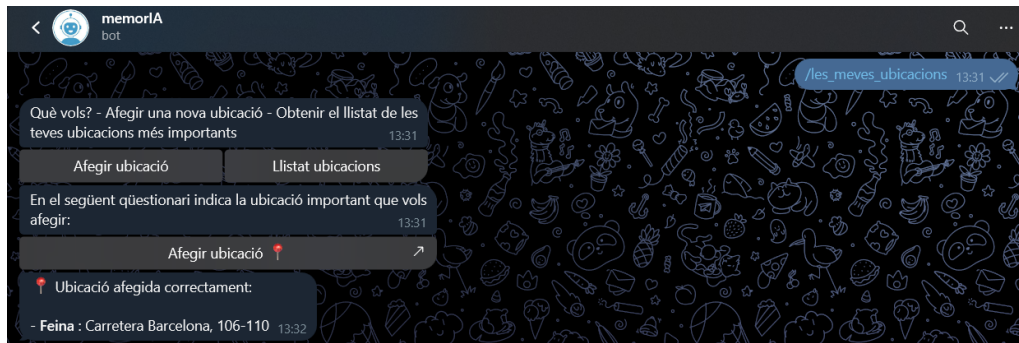


FIGURA U.2: Nova ubicació afegida pel llistat d'ubicacions d'interès.

U.2 Comanda */historial_ubicacions*:

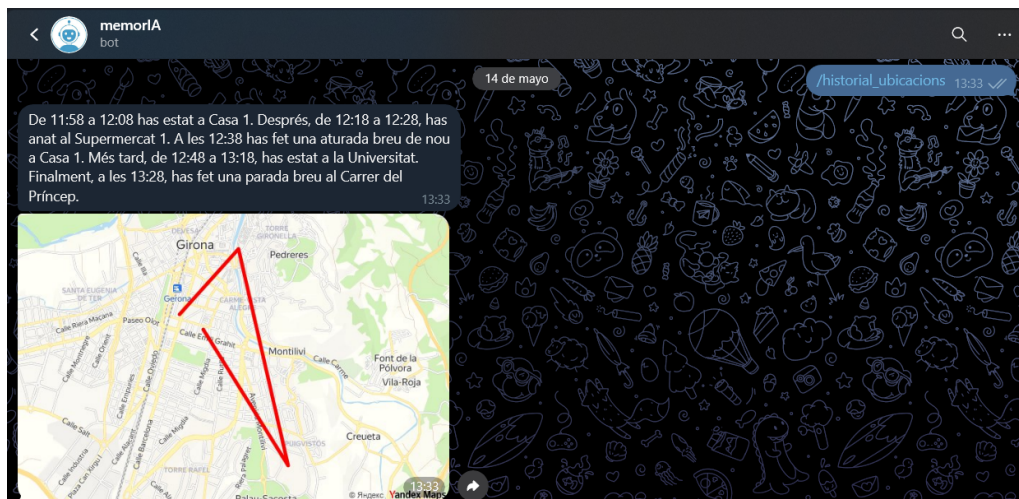


FIGURA U.3: Resum de les ubicacions registrades de l'usuari.

U.3 Comanda /resum_matinal:

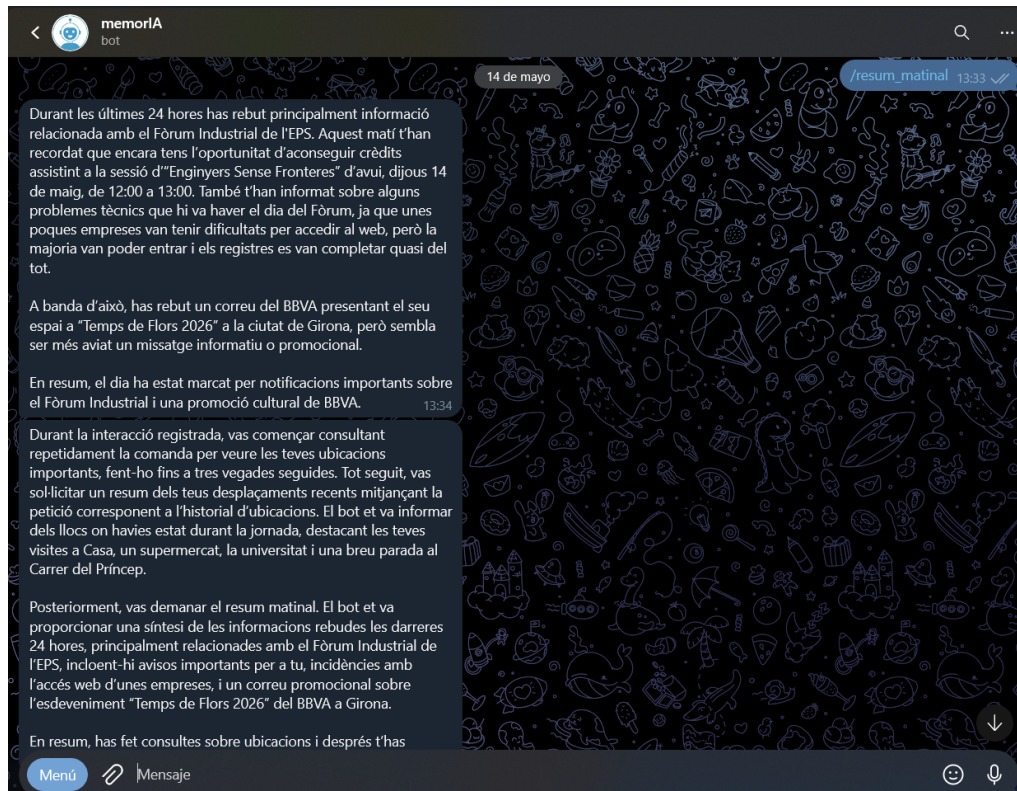


FIGURA U.4: Resum dels correus electrònics i conversa amb el bot.

U.4 Text en llenguatge natural: /medicacio:

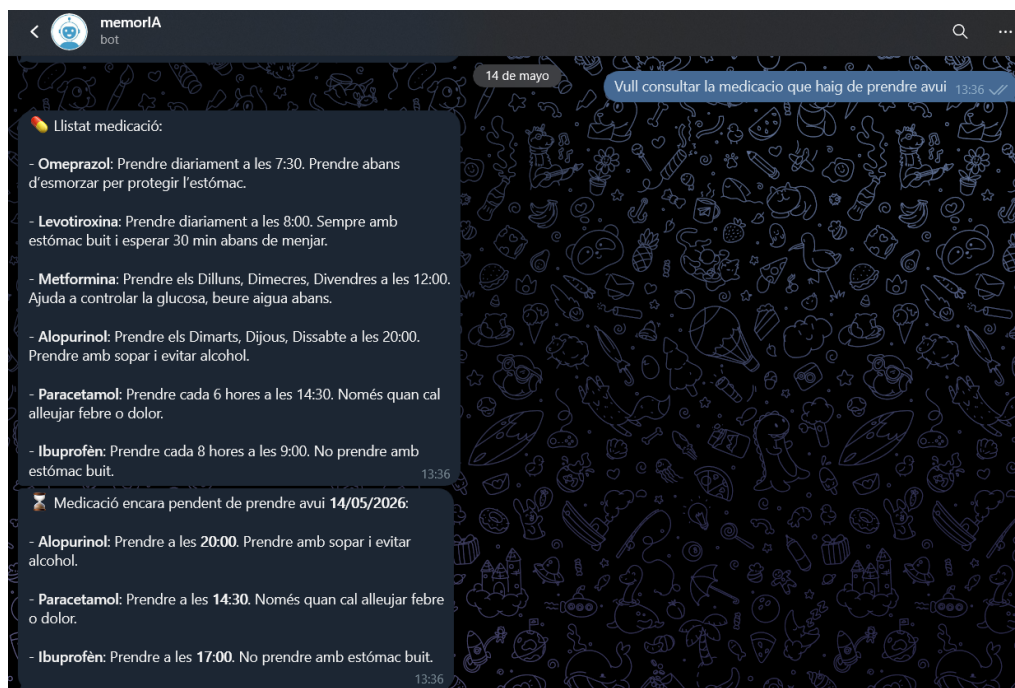


FIGURA U.5: Resposta a la comanda de medicació amb llenguatge natural.

U.5 Comandes /agenda_avui i /resum_ahir:

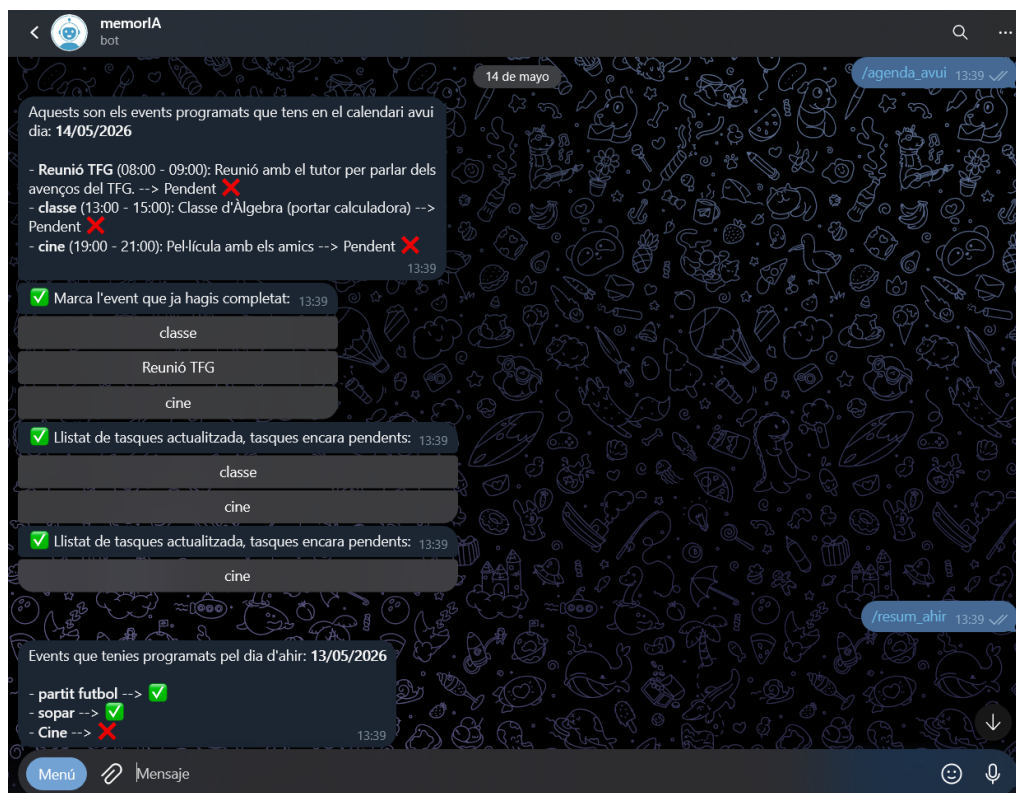


FIGURA U.6: Respostes a les comandes relacionades amb esdeveniments de calendari.

U.6 Comanda /exercici_cognitiu:

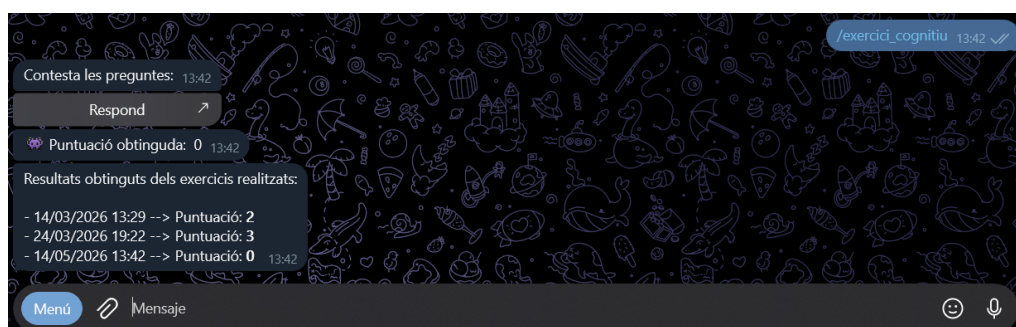


FIGURA U.7: Resultat d'execució de la comanda d'exercici cognitiu.

Bibliografía

- [1] Cole Stryker; Eda Kavlakoglu. *What is artificial intelligence (AI)?* <https://www.ibm.com/think/topics/artificial-intelligence>. Accessed: 06-04-2026.
- [2] Docker. *Use containers to Build, Share and Run your applications*. <https://www.docker.com/resources/what-container/>. Accessed: 06-04-2026.
- [3] Red Hat. *What is a webhook?* <https://www.redhat.com/en/topics/automation/what-is-a-webhook>. Accessed: 06-04-2026.
- [4] Ngrok. *What is Ngrok?* <https://ngrok.com/docs/what-is-ngrok>. Accessed: 07-04-2026.
- [5] Refactoring.guru. *Facade*. <https://refactoring.guru/design-patterns/facade>. Accessed: 02-05-2026.
- [6] Refactoring.guru. *Adapter*. <https://refactoring.guru/design-patterns/adapter>. Accessed: 02-05-2026.
- [7] Refactoring.guru. *Observer*. <https://refactoring.guru/design-patterns/observer>. Accessed: 02-05-2026.
- [8] Refactoring.guru. *Command*. <https://refactoring.guru/design-patterns/command>. Accessed: 02-05-2026.
- [9] willph Wilmerson. *evolutionApi_chatwoot_docker*. *GitHub repository*. https://github.com/willph/evolutionApi_chatwoot_docker. Accessed: 18-02-2026.
- [10] Josema Fernandez | IA. 🔥 *La Guía Definitiva para Automatizar WhatsApp con Chatwoot y n8n GRATIS*. <https://www.youtube.com/watch?v=soNA-A7JzII>. [Video]. Accessed: 20-02-2026.
- [11] n8n. *Google OAuth Authentication in n8n - From Setup to Connection*. <https://www.youtube.com/watch?v=FBGtpWMTppw>. [Video]. Accessed: 20-02-2026.
- [12] Chatwoot. *How to create an API channel inbox?* <https://www.chatwoot.com/hc/user-guide/articles/1677839703-how-to-create-an-api-channel-inbox>. Accessed: 20-02-2026.
- [13] Diego Vásquez | AI. *Agentes IA con Intervención Humana: Tutorial Chatwoot + N8N (Plantilla Gratis)*. <https://www.youtube.com/watch?v=8PyS6YW30R8>. [Video]. Accessed: 05-03-2026.
- [14] evolutionapi. *Get Started Running EvolutionAPI on Your Own Server*. <https://docs.evoapicloud.com/servers/newserver>. Accessed: 10-03-2026.
- [15] Flowise. *Agentflow V2*. <https://docs.flowiseai.com/using-flowise/agentflow-v2>. Accessed: 16-03-2026.
- [16] Chatwoot. *Chatwoot User Guide*. <https://www.chatwoot.com/hc/user-guide/articles/1677693021-how-to-use-webhooks>. Accessed: 16-03-2026.
- [17] enguillem. *Cómo Conectar N8N con Telegram Usando Ngrok en Local | Automatizatus Workflows con HTTPS*. <https://www.youtube.com/watch?v=hl2vNwCwxvk>. Accessed: 24-03-2026.
- [18] n8n. *Welcome to n8n Docs*. <https://docs.n8n.io/>. Accessed: 20-04-2026.