



OPTIMITZACIÓ DE GESTIÓ DE SINISTRES IMPULSADA PER IA

ANDREU SABATER LÓPEZ

Director/a

SERGIO GONZÁLEZ FERNÁNDEZ (CLEVEREA, SOCIEDAD LIMITADA)

Ponent: ENRIC MAYOL SARROCA (Departament d'Enginyeria de Serveis i Sistemes d'Informació)

Titulació

Grau en Enginyeria Informàtica (Enginyeria del Software)

Memòria del treball de fi de grau

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) – BarcelonaTech

Resum

Aquest **Treball de Final d'Estudis**, correspon al grau d'**Enginyeria Informàtica** de la **Facultat d'Informàtica de Barcelona** a la **Universitat Politècnica de Catalunya**. En el projecte s'ha desenvolupat una solució tecnològica basada en **intel·ligència artificial generativa** capaç de generar resums concisos però complets de l'estat d'un sinistre en temps real. Aquesta eina ha permès als agents de Cleverea comprendre de manera eficient la situació de cada cas, millorant tant la productivitat interna com la satisfacció dels clients.

Abstract

This **Final Degree Project** corresponds to the **Computer Engineering** degree at **Universitat Politècnica de Catalunya**. The project involved the development of a technological solution based on generative artificial intelligence, capable of generating concise and complete summaries of a claim's status in real time. This tool has enabled Cleverea's agents to efficiently understand each case's situation, improving both internal productivity and customer satisfaction.

Resumen

Este **Trabajo de Fin de Grado** corresponde al grado en **Ingeniería Informática** de la **Facultad de Informática de Barcelona** en la **Universitat Politècnica de Catalunya**. En el proyecto se ha desarrollado una solución tecnológica basada en inteligencia artificial generativa, capaz de generar resúmenes concisos pero completos del estado de un siniestro en tiempo real. Esta herramienta ha permitido a los agentes de Cleverea comprender de forma eficiente la situación de cada caso, mejorando tanto la productividad interna como la satisfacción de los clientes.

Agraïments

Vull expressar el meu sincer agraïment a **Cleverea** per haver-me brindat l'oportunitat de formar part del seu equip i per haver-me permès desenvolupar aquest projecte en un entorn professional i innovador. Agraïxo especialment als meus companys i al **director del projecte** pel seu suport constant i per totes les oportunitats d'aprenentatge que m'han ofert al llarg del procés.

També vull agrair a l'**Enric Mayol Sarroca**, ponent del projecte, per la seva implicació, dedicació i seguiment al llarg de tot el desenvolupament.

Finalment, gràcies de tot cor als **meus familiars, amics i totes les persones que m'han acompanyat en aquest camí**, tant assumint tasques del dia a dia com donant-me suport emocional en els moments més intensos. Han estat un pilar fonamental i, sens dubte, han fet possible que aquest projecte es pogués dur a terme de la millor manera possible.

Contingut

Resum	2
Abstract.....	2
Resumen	2
Agraïments	3
1. Context	11
1.1. Introducció.....	11
1.2. Identificació del problema	12
1.3. Stakeholders	12
2. Justificació	14
2.1. Estat de l'Art: Solucions existents	14
2.2. Necessitat d'una nova solució	15
2.3. Casos d'ús reals d'IA en el món de les assegurances	15
2.4. Conclusions.....	16
3. Abast del projecte	17
3.1. Objectius del projecte	17
3.2. Requisits.....	18
3.3. Obstacles	19
3.4. Riscos	20
4. Gestió del projecte	22
4.1. Metodologia i seguiment.....	22
4.2. Planificació temporal.....	24
4.3. Valoració d'alternatives i plans d'acció.....	36
4.4. Gestió econòmica.....	37

5. Sostenibilitat.....	45
5.1. Dimensió econòmica	45
5.2. Dimensió ambiental	45
5.3. Dimensió social	46
6. Requisits i especificació	47
6.1. Cas d'us UC-01 – Obtindre resum de sinistre.....	47
6.2. Requisits funcionals (RF).....	48
6.3. Requisits no funcionals (RNF)	49
6.4. Requisits de dades i restriccions (RD)	50
6.5. Criteris d'acceptació i proves	50
6.6. Model de dades	51
7. Tecnologies emprades.....	52
7.1. OpenAI Platform	52
7.2. Twelve Labs	54
7.3. Infraestructura addicional	55
7.4. Processament asíncron amb Celery i Redis	56
7.5. Conclusions.....	56
8. Arquitectura i distribució de les dades a Cleverea.....	57
8.1. Distribució de les dades.....	57
8.2. Arquitectura en microserveis	62
8.3. Arquitectura hexagonal.....	64
8.4. Fluxos funcionals detectats.....	66
9. Disseny del sistema	68
9.1. Visió general del sistema.....	68

9.2. Vista de contenidors	70
9.3. Components clau.....	71
9.4. Disseny de la interacció amb sistemes d'IA	77
9.5. Mecanismes de seguretat i protecció de dades	84
9.6. Control d'ús i limitació de costos.....	84
9.7. Complexitat d'integració amb dades no relacionades (DynamoDB).....	85
10. Implementació final	87
10.1. Visió global de la implementació.....	87
10.2. Injecció de dependències	88
10.3. Flux de generació del resum de sinistre.....	90
10.4. Flux d'anàlisi automàtica de documents	92
10.5. Integració amb els SDK	95
10.6. Control de costos i rendiment	100
10.7. Testing.....	100
10.8. Casos de prova	101
11. Conclusions i següents passos	108
11.1. Conclusions generals del projecte	108
11.2. Conclusió personal	109
11.3. Grau d'assoliment dels objectius	111
11.4. Línies futures de treball	114
12. Referències.....	116
13. Annexos.....	121
Annex A - Extractes de codi	121
Annex B - Distribució de les dades a DynamoDB	165

Annex C - figures de suport.....	168
Annex D - Exemples de resums	171
Annex E – Prompts.....	173

Índex de taules

Taula 1: Comparació de solucions existents. Font: Elaboració pròpia.....	14
Taula 2: Resum de rols i responsabilitats. Font: Elaboració pròpia.	25
Taula 3: Resum de les tasques. Font: Elaboració pròpia.	34
Taula 4: Estimacions per rol i hora. Font: Elaboració pròpia.....	38
Taula 5: Estimació de costos per tasca. Font: Elaboració pròpia.....	40
Taula 6: Amortització dels recursos materials. Font: Elaboració pròpia.....	41
Taula 7: Costos dels recursos software. Font: Elaboració pròpia.	41
Taula 8: Pressupost final. Font: Elaboració pròpia.	43
Taula 9. Cas d'ús UC-01. Font: Elaboració pròpia.....	48
Taula 10. Històries d'usuari. Font: Elaboració pròpia.	48
Taula 11. Requisits funcionals. Font: Elaboració pròpia.	49
Taula 12. Requisits no funcionals. Font: Elaboració pròpia.	49
Taula 13. Requisits de dades i restriccions. Font: Elaboració pròpia.....	50
Taula 14. Criteris d'acceptació i proves. Font: Elaboració pròpia.	51
Taula 15. Comparació models OpenAI. Font: https://platform.openai.com/docs/models	53
Taula 16. Dades utilitzades en el projecte. Font: Elaboració pròpia.	57
Taula 17. Serveis principals del projecte. Font: Elaboració pròpia.....	62
Taula 18. Principals contenidors de la solució. Font: Elaboració pròpia.	70
Taula 19. Components transversals. Font: Elaboració pròpia.	71
Taula 20. Mòduls principals del flux "Resum de sinistre". Font: Elaboració pròpia.	73
Taula 21. Mòduls principals del flux "Anàlisi automàtica de documents". Font: Elaboració pròpia.....	75
Taula 22. Relació de components amb requisits. Font: Elaboració pròpia.....	77

Taula 23. Contenedors afectats en el projecte. Font: Elaboració pròpia.....	87
Taula 24. Beneficis obtinguts amb la injecció de dependències. Font: Elaboració pròpia....	90
Taula 25. Flux de generació del resum de sinistre. Font: Elaboració pròpia.....	91
Taula 26. Flux d'anàlisi automàtica de documents. Font: Elaboració pròpia.....	95

Índex de figures

Figura 1: Waterfall vs Agile. Font: www.donetonic.com	22
Figura 2: Sprints en Scrum. Font: www.nerukanadun.medium.com	23
Figura 3: Kanban Board. Font: www.wikipedia.org	23
Figura 4: Diagrama de Gantt. Font: Elaboració pròpia, eina: GanttProject	35
Figura 5. Diagrama UML del cas d'ús UC-01. Font: Elaboració pròpia.	47
Figura 6. Distribució de les dades a Cleverea. Font: Elaboració pròpia.	58
Figura 7. Distribució dels microserveis a Cleverea. Font: Elaboració pròpia.	64
Figura 8. Vista simplificada de l'arquitectura hexagonal. Font: Elaboració pròpia.	66
Figura 9. Flux de resum de sinistre. Font: https://www.planttext.com	69
Figura 10. Flux d'anàlisi automàtica de documents. Font: https://www.planttext.com	70
Figura 11. Exemple de crida a l'endpoint. Font: SwaggerUI	102
Figura 12. Exemple de resum en el Back-Office. Font: Elaboració pròpia.	103
Figura 13. Exemples de costos de crides a l'endpoint principal. Font: GoogleChrome	104
Figura 14. Dashboard de costos OpenAI PLaform. Font: www.platform.openai.com/usage	105
Figura 15. Diagrama flux d'anàlisi automàtica de documents. Font: https://www.plantuml.com	168
Figura 16. Diagrama 1 flux de generació del resum de sinistre. Font: https://www.plantuml.com	169
Figura 17. Diagrama 2 flux de generació del resum de sinistre. Font: https://www.plantuml.com	170

1. Context

Aquest projecte forma part del **Treball de Final d'Estudis (TFE) en modalitat B**, corresponent a un **projecte desenvolupat en empresa** dins del grau d'**Enginyeria Informàtica** de la **Facultat d'Informàtica de Barcelona (FIB)** a la **Universitat Politècnica de Catalunya (UPC)**.

Com a estudiant de la menció en **Enginyeria del Software**, la meua tasca s'ha centrat en el desenvolupament d'una **solució tecnològica per a Cleverea**, amb l'objectiu d'aportar valor a l'empresa i millorar els seus processos mitjançant eines innovadores.

Aquest capítol té com a finalitat **contextualitzar el projecte**, oferint una visió clara del problema a abordar. A través d'aquesta anàlisi es justifica la necessitat del projecte.

1.1. Introducció

És ben evident que el món en el qual vivim avui en dia està passant per una **fase de digitalització**. A Cleverea estem buscant adaptar el sector de les assegurances perquè pugui guanyar-se el lloc en aquest nou entorn virtual. Cleverea és una empresa dedicada a revolucionar el sector mitjançant solucions tecnològiques avançades. La nostra missió com a *insurtech*¹ és aprofitar la tecnologia per simplificar i millorar tots els aspectes dels nostres serveis.

En un sector tan dinàmic i competitiu com és el de les assegurances, un dels processos més crucials i amb major impacte en la captació i fidelització dels clients és la gestió dels sinistres. Aquest procés, que involucra múltiples actors i requereix l'anàlisi d'una gran quantitat de dades provinents de diverses fonts, presentava una elevada complexitat i alentia la resolució dels casos, provocant insatisfacció entre els clients.

Per donar resposta a aquest repte, s'ha desenvolupat una solució tecnològica basada en **intel·ligència artificial generativa (IAG)** ² capaç de generar resums concisos però complets de l'estat d'un sinistre en temps real. Aquesta eina ha permès als agents de Cleverea

¹ El terme *insurtech* fa referència a l'ús de la tecnologia i la innovació digital per millorar, optimitzar i transformar el sector de les assegurances. Es tracta d'un neologisme que combina les paraules angleses *insurance* (assegurances) i *technology* (tecnologia).

² La intel·ligència artificial generativa o IA generativa és un tipus de sistema d'intel·ligència artificial capaç de generar text, imatges o altres mitjans en resposta a comandes. Els models d'IA generativa aprenen els patrons i l'estructura de les seves dades d'entrenament d'entrada i després generen noves dades que tenen característiques similars.

comprendre de manera eficient la situació de cada cas, millorant tant la productivitat interna com la satisfacció dels clients.

La intel·ligència artificial ha jugat un rol crucial en la solució desenvolupada, ja que ha estat capaç de rebre i processar totes les dades recopilades amb la finalitat de **generar resums automàtics**. Aquests resums s'han proporcionat a l'equip de gestió de sinistres, permetent-los estar al dia de l'estat de tots els casos de manera àgil i eficient.

1.2. Identificació del problema

El procés de la gestió de sinistres a Cleverea implica conèixer dades de múltiples fonts que no sempre són fàcilment accessibles. Actualment, la informació rellevant sobre un sinistre es troba dispersa en:

1. **Base de dades PostgreSQL**, on es registra molta informació rellevant de cada sinistre.
2. **Base de dades DynamoDB**, allotja transcripcions de trucades i correus electrònics.
3. **Contenidors Amazon S3**, que contenen fitxers en diferents formats pujats tant pels clients com pels propis agents.

Els agents de sinistres havien de revisar manualment aquestes fonts per obtenir una visió clara de la situació en cada cas. Aquest procés requeria molt temps i generava retards en la resolució dels sinistres, afectant tant l'eficiència operativa com la satisfacció del client.

A més, tant la diversitat de les dades com el seu format afegien un grau addicional de complexitat al procés. La manca de coneixement sobre les dades disponibles per a cada sinistre pot donar lloc a diverses situacions, com ara la pèrdua d'informació o el risc de passar per alt detalls rellevants.

1.3. Stakeholders

El projecte involucra diversos grups d'interès clau que es veuran beneficiats per la implementació de la solució.

1.3.1. Clients de Cleverea

S'ha presentat un impacte clar i positiu oferint als clients un servei més àgil i eficient, reduint el temps de resposta i millorant la seva experiència i satisfacció amb la companyia.

1.3.2. Agents de sinistres de Cleverea

Els principals usuaris de la solució, que poden conèixer ràpidament informació rellevant sense necessitat de recórrer múltiples fonts de dades manualment.

1.3.3. Equip de direcció i proveïdors de Cleverea

Interessats en l'impacte positiu del projecte sobre la productivitat i l'eficiència operativa de l'empresa.

1.3.4. Ponent del projecte

Tot i no estar vinculat directament amb la solució realitzada, presenta interès en l'èxit de la realització d'aquest Treball de Final d'Estudis.

1.3.5. Jo com a estudiant

Jo també soc un dels majors interessats en aquest projecte, ja que l'obtenció del títol en Enginyeria Informàtica a la UPC depèn directament de l'èxit d'aquest projecte.

2. Justificació

L'ús de la intel·ligència artificial ha permès **automatitzar** la generació de resums breus i precisos sobre l'estat dels sinistres, facilitant la tasca dels agents i accelerant la resolució dels casos [7]. Aquesta solució ha tingut un impacte directe en la reducció del temps d'anàlisi i en l'augment de l'eficiència del servei [8].

A més, en el sector assegurador, l'adopció d'aquestes tecnologies està en creixement. Empreses com MAPFRE, Allianz i Lemonade ja han implementat sistemes similars, com veurem més endavant. No obstant això, les solucions existents no cobreixen completament les necessitats de Cleverea, fet que justifica el **desenvolupament d'una eina pròpia**.

2.1. Estat de l'Art: Solucions existents

Per determinar la viabilitat d'un nou desenvolupament, es va analitzar l'ecosistema actual de solucions disponibles en el mercat per a la gestió de sinistres i la generació de resums automatitzats. La Taula 1 mostra una comparativa.

Solució	Tipus	Funcionalitats principals	Limitacions pels agents
Talend [31] / Apache Nifi [32]	ETL	Integració i processament de dades de múltiples fonts	Només transforma dades
MuleSoft [33] / Zapier [34]	Integració	Connecta dades de diferents sistemes i aplicacions	No proporciona anàlisi ni processament de llenguatge natural
OpenAI GPT [35]	IA Generativa	Creació de resums automàtics	Cal fer la recollida de dades per separat
SummarizeBot [36]	Resums IA	Extracció d'informació clau de textos llargs	Pot no ser capaç d'interpretar correctament documents tècnics complexos
Power BI [37]	Visualització de dades	Creació de dashboards per facilitar l'anàlisi de sinistres	No permet la generació dels resums que busquem

Taula 1: Comparació de solucions existents. Font: Elaboració pròpia

Aquestes eines poden arribar a ser útils en funcionalitats separades, però en el nostre cas calia una solució pròpia que permeti fer tot el procés de principi a final.

2.2. Necessitat d'una nova solució

Tot i que existien eines que cobreixen parcialment les funcionalitats necessàries, cap d'elles proporcionava una solució integral específica per a la gestió de sinistres en l'entorn de Cleverea. Els principals motius són:

1. **Integració amb bases de dades pròpies:** Les eines no permetien una integració personalitzada amb les dades de PostgreSQL i DynamoDB utilitzades per Cleverea.
2. **Necessitat de control sobre el model IA:** L'ús d'IA generativa per a la creació de resums ha d'estar optimitzat per aquest cas d'ús específic. Les solucions generalistes poden no oferir la precisió ni format necessaris.
3. **Eficiència i escalabilitat:** La nostra solució ha pogut ser ajustada a les necessitats específiques de Cleverea, amb la possibilitat de millorar la seva capacitat a mesura que el volum de sinistres creixi.

2.3. Casos d'ús reals d'IA en el món de les assegurances

Diferents estudis [9] mostren que la gran majoria d'empreses del sector assegurador ja estan iniciant o planejant la integració de la IAG en els seus processos per poder millorar l'eficiència i, per tant, la satisfacció del client.

A continuació es presenten alguns dels casos amb major impacte dins del sector assegurador.

2.3.1. MAPFRE

MAPFRE ha anunciat recentment [10] la propera integració d'un nou servei de verificació automàtica de danys a través d'un algoritme basat en IA. Com un dels principals competidors en el sector de cotxes a Espanya això suposa un impacte per Cleverea, és un gran pas que situa a MAPFRE en una posició d'avantatge a l'hora d'oferir pòlisses.

El seu algoritme permetrà als clients mostrar l'estat del seu vehicle **sense desplaçaments ni costos de peritatge**, cosa que farà possible l'activació de les pòlisses de forma instantània, ajustant-les a cada cas segons conveniència.

2.3.2. Allianz

A diferència de Mapfre, **Allianz** està posant el focus en la personalització del producte [11], buscant oferir solucions justes i adaptades a cada client. L'anàlisi massiva de dades els permet proporcionar respostes àgils i ajustades a les necessitats específiques de cada usuari.

Aquesta estratègia, combinada amb l'automatització de la gestió de sinistres, facilita una tramitació més ràpida i eficient.

2.3.3. Lemonade

Lemonade és un referent [12] per tota asseguradora tecnològica amb la integració de la IA dins dels seus processos. L'impacte més gran i notable que han tingut ha sigut en la gestió dels sinistres, l'empresa reclama que la seva intel·ligència artificial interna ha sigut capaç de resoldre un 30% de tots aquests **sense intervenció humana**.

2.4. Conclusions

La **intel·ligència artificial generativa** està transformant profundament el sector assegurador, obrint noves oportunitats per a la **personalització de productes, l'automatització de processos** i una **gestió de riscos més eficient**. Empreses líders ja han començat a implementar solucions basades en IAG per **agilitzar els seus processos** i satisfer les creixents expectatives dels clients.

Tot i que no totes les companyies asseguradores han fet públics els seus projectes relacionats amb aquesta tecnologia, la tendència del mercat apunta clarament cap a la seva adopció. Segons un informe de **KPMG** [13], les organitzacions del sector assegurador estan **augmentant la seva inversió en IA** per transformar les seves operacions i mantindre's competitives en un entorn en constant evolució [14].

Per continuar sent competitius en aquest context tecnològic emergent, a **Cleverea** hem decidit prendre aquest pas en endavant per integrar la IA generativa en el nostre dia a dia. Amb aquesta nova solució hem buscat mantindre i fer créixer la posició que ens hem guanyat en el mercat. Optimitzar les tasques de gestió de sinistres està suposant un canvi crucial en aquest nou camí.

3. Abast del projecte

En aquest capítol s'estableixen els **objectius principals del projecte**, els quals determinen els criteris d'èxit del treball. A partir d'aquests objectius es definiran els **requisits funcionals i no funcionals**, que van servir de guia per al desenvolupament i implementació de la solució.

A més, es du a terme una anàlisi dels **possibles riscos i obstacles** que podien sorgir durant el disseny i la integració del projecte, identificant factors que podrien comprometre la seva viabilitat i proposant estratègies per mitigar-los.

3.1. Objectius del projecte

El principal objectiu d'aquest projecte ha sigut **millorar la gestió de sinistres a Cleverea**, un procés que, en el sector assegurador, sovint resulta complex i poc àgil. Qualsevol persona que hagi hagut de gestionar un sinistre sap que no és un tràmit senzill.

- **Des del punt de vista de l'asseguradora**, cada cas s'ha d'analitzar en profunditat abans de prendre una decisió per evitar possibles fraus.
- **Des del punt de vista del client**, l'objectiu és obtenir la millor solució **de la manera més ràpida possible** per minimitzar l'impacte en el seu dia a dia.

Aquest procés implica una **gran quantitat d'anàlisi de dades i presa de decisions**, fet que el fa difícilment automatitzable amb eines tradicionals. Per abordar aquest repte, el projecte s'ha centrat en el desenvolupament d'una **solució basada en intel·ligència artificial generativa**, capaç de generar **resums automàtics sobre l'estat dels sinistres**. Aquesta solució integra informació de diverses fonts per millorar l'eficiència dels gestors de sinistres.

3.1.1. Fases del projecte

- 1. Recollida, ordenació i transformació de dades**
 - a. La solució ha de ser capaç de **compilar i estructurar informació de diverses fonts**.
 - b. Aquest pas és fonamental per **reduir la càrrega manual** dels gestors de sinistres, que han de tractar múltiples casos simultàniament.
- 2. Generació de resums amb IA generativa**
 - a. Un cop recollida la informació, la IA generativa processa i interpreta les dades per generar **resums clars i concisos** sobre l'estat de cada sinistre.
- 3. Integració en l'entorn de Cleverea**

- a. L'objectiu ha sigut que els gestors disposin **ràpidament del resum de cada cas**, millorant la seva capacitat de resposta i decisió. Per fer-ho el resum havia d'estar disponible en el seu entorn de treball.

4. Generació de la cronologia (opcional)

- a. Aquesta funció tot i no ser essencial per poder tindre un bon context del projecte pot millorar el seu impacte.
- b. La generació d'una cronologia capaç d'indicar quan i quin esdeveniment ha ocorregut dins d'un sinistre pot ser molt interessant. Amb aquesta cronologia simplifiquem a una ullada l'estat d'un sinistre, tot i no ser gaire complet podria ajudar a posar en context l'estat de forma molt ràpida.
- c. A més, obrim les portes a un nou projecte on, amb entrenament de cronologies anteriors, podríem suggerir als agents el següent pas per avançar en el procés i, així apropar-nos una mica més a l'automatització d'aquests processos.

Amb aquesta solució, Cleverea ha pogut **agilitzar el procés de gestió de sinistres**, optimitzant tant la **detecció de fraus** com la **satisfacció del client**. L'automatització d'aquest procés permet als gestors centrar-se en **tasques d'alt valor**, millorant així l'eficiència operativa de l'empresa.

3.2. Requisits

En aquesta secció contextualitzaré les competències tècniques definides pel projecte per així poder identificar els seus requisits.

- **CES1.1: Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics.** El projecte afecta a una part crítica d'una empresa real i, per tant, qualsevol errada o desviació podria comportar grans riscos, com pèrdua de diners o insatisfacció dels clients.
- **CES1.2: Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles.** Cal que la solució final estigui integrada dins de l'entorn de Cleverea per garantir un fàcil ús i mantenibilitat, així com seguir les metodologies i tecnologies que s'utilitzen en el seu dia a dia.
- **CES1.3: Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar.** Cal tindre en compte l'impacte que tindrà la solució una cop posada en producció la solució, estabilitat del sistema i control de costos.

- **CES1.7: Controlar la qualitat i dissenyar proves en la producció de software.** Per saber si el projecte ha estat exitós i si ho seguirà sent en un futur cal fer proves sobre els resultats obtinguts, garantint així la integritat del sistema i les solucions obtingudes.
- **CES2.2: Dissenyar solucions apropiades en un o més dominis d'aplicació, utilitzant mètodes d'enginyeria del software que integrin aspectes ètics, socials, legals i econòmics.** La solució està clarament emmarcada en el domini de les assegurances i desenvolupada aplicant bones pràctiques d'enginyeria del software. S'han tingut aspectes **econòmics** per garantir l'optimització de recursos i la reducció de costos operatius. A nivell **social**, es millora la qualitat de vida dels agents i l'experiència dels clients. En l'àmbit **ètic**, s'ha aplicat mesures per garantir la transparència i fiabilitat dels resums. Pel que fa a l'aspecte **legal**, s'ha considerat la protecció de dades i la seguretat de la informació, assegurant el compliment de la normativa vigent.
- **CES1.9: Demostrar comprensió en la gestió i govern dels sistemes software.** La solució final ha estat construïda de principi a final de forma que segueixi les bones pràctiques de software dins del context de Cleverea. També s'ha investigat i detallat els requisits i capacitats del sistema.

3.3. Obstacles

Com en tots els desenvolupaments poden sorgir diversos obstacles o inconvenients que poden afectar a la satisfacció dels objectius. I aquests són els que vam detectar

1. Qualitat dels resums generats

- a. No existeix una mètrica objectiva que ens digui si un resum serà realment útil pel procés. El prompting juga un paper clau.
- b. La intel·ligència artificial tot i ser una eina molt útil no ens garanteix mai una fiabilitat total en les respostes que dona.

2. Qualitat de les dades

- a. Tindre informació incompleta o que la IA no la conegui tota pot suposar un problema d'inconsistència o incertesa.
- b. Per enviar totes aquestes dades a la intel·ligència artificial han hagut de ser manipulades de forma que es poden distingir i interpretar de manera adient.

3. Integració amb sistemes existents

- a. La connexió amb diferents fonts de dades i Application Programming Interfaces (APIs) externes podia comportar una solució correcta però amb baix rendiment i, per tant, ineficient.

3.4. Riscos

De la mateixa manera que en tot projecte de software, sempre hi ha alguns riscos que l'acompanyen. Ara parlaré d'alguns d'aquests i les diferents solucions proposades per poder evitar-los o mitigar-los.

3.4.1. Errors en la connexió amb bases de dades i APIs

Quan es treballa amb sistemes externs la interoperabilitat sempre pot suposar un inconvenient, la disponibilitat d'altres serveis no depèn de nosaltres. En aquest cas ha sigut necessari saber el motiu pel qual hi ha hagut un error i poder replicar la crida en els casos que convenients.

3.4.2. Problemes de rendiment

La rapidesa en la generació d'aquests resums és clau de cara a l'objectiu final del projecte. Hem hagut de buscar optimitzacions en l'extracció de dades o la integració de diferents models d'IA per poder reduir la latència.

3.4.3. Falta de temps en el desenvolupament

A causa de la naturalesa del projecte, el temps amb el què comptem per realitzar-lo és limitat i ha pogut condicionar la qualitat de la solució. Com que no podem lluitar contra el temps hem establert un mínim valor a la solució que de ser necessari serà ampliat en projectes futurs.

3.4.4. Gran quantitat de dades

Totes les API d'intel·ligència artificial limiten en un nombre màxim de tokens d'entrada. Com que nosaltres tractem amb un gran nombre de dades, ha sigut necessari tindre un control en l'enviament d'aquestes dades.

3.4.5. Inexperiència

Com que no tinc una gran experiència en el desenvolupament de sistemes software m'he trobat amb obstacles durant el procés d'integració que han suposat unes dificultats. L'única

solució possible en aquest cas ha estat aprendre o preguntar a altres persones que ja han passat per alguna situació similar.

4. Gestió del projecte

4.1. Metodologia i seguiment

4.1.1 Metodologia

Per poder dur a terme aquest treball he hagut d'investigar quines metodologies de desenvolupament de software s'adapten millor al meu projecte.

En la investigació i gràcies altres assignatures del grau he pogut diferenciar dues grans metodologies: en cascada o *Waterfall* i àgil o *Agile* [16] (Figura 1).

La **metodologia Waterfall** segueix un model seqüencial on cada fase del desenvolupament es completa abans de passar a la següent. Aquesta estructura permet una planificació detallada des del principi i ofereix una visió clara del producte final, però pot resultar rígida davant de canvis o nous requisits durant el procés.

En canvi, la **metodologia Agile** és més flexible i iterativa. Es basa en una planificació dinàmica que s'adapta al progrés del projecte i a les necessitats emergents. Divideix el desenvolupament en petites parts, permetent solucionar problemes progressivament i realitzar ajustos constants segons el feedback rebut. Dins d'aquesta metodologia s'hi coneixen múltiples variacions com: *Scrum*, *Extreme Programming* o *Kanban*.

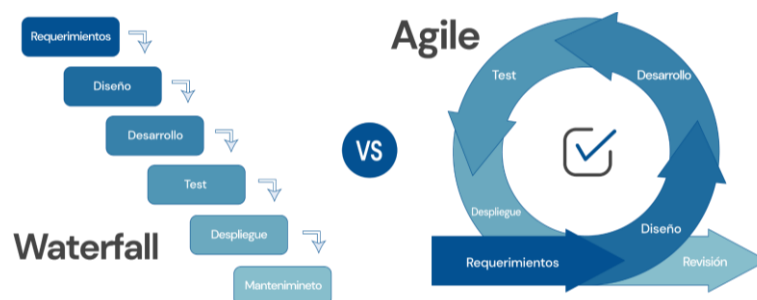


Figura 1: Waterfall vs Agile. Font: www.donetonic.com

En el meu cas, he treballat amb una metodologia àgil, que m'ha permès anar adaptant les solucions del meu projecte segons el progrés que ha tingut aquest. Com que es tracta d'un projecte diferent, amb un únic desenvolupador i dates límit d'entrega, he agafat algunes característiques de diferents metodologies àgils per aconseguir que s'adapti a les necessitats d'aquest.

En primer lloc, agafat d'Scrum, he fet servir els *sprints* (Figura 2), una figura que permet estimar i distribuir les tasques en la planificació global del projecte. Cadascun d'aquests

sprints té unes característiques que també m'han servit, com ara el seguiment o *Sprint Review*, per poder estar al cas, tant jo com el director, de l'estat del projecte.



Figura 2: Sprints en Scrum. Font: www.nerukanadun.medium.com

En segon lloc, de Kanban ens ha sigut bastant útil integrar les *Kanban Boards* (Figura 3), aquestes taules permeten tindre un control i seguiment de les tasques i el seu estat en temps real. Són bastant útils per organitzar el treball i eliminar el caos, a més a més, a Cleverea ja les utilitzem.



Figura 3: Kanban Board. Font: www.wikipedia.org

4.1.2. Seguiment

En aquest apartat defineixo quines eines de seguiment m'han acompanyat durant el projecte.

4.1.2.1. Documentació

1. **Google Docs** [19]: que com a eina d'editor de text en línia em permet realitzar tota la documentació referent a les entregues.
2. **Google Drive** [20]: que dona suport a l'eina anterior amb l'emmagatzematge de les dades al núvol.
3. **Notion** [21]: és una plataforma de col·laboració en línia que facilita la gestió de tasques i organització. Per adaptar-me a la forma de treball de Cleverea ha sigut convenient definir les tasques de forma apropiada utilitzant aquesta eina.

4.1.2.2. Comunicació

1. **Google Meet** [22] i **Calendar** [23]: la combinació d'aquestes dues eines ha permès planificar i realitzar reunions programades tant amb el director com el ponent del projecte.

2. **Slack** [24]: com a plataforma de comunicació empresarial ja integrada a Cleverea ha donat un fort suport amb la comunicació no planificada amb el director del projecte.

4.1.2.3. Desenvolupament

1. **Git** [25]: com l'eina per defecte de control de versions ha sigut indispensable per poder integrar el meu codi a la infraestructura de Cleverea.
2. **CircleCI** [26] + **Docker** [27]: CircleCI com plataforma d'integració i lliurament continu, s'encarrega de la fase de testing, executant els programes tant unitaris com d'integració amb el sistema, aquesta execució ve recolzada per Docker que automatitza el desplegament d'aplicacions dins de contenidors de programari.
3. **Amazon Web Services (AWS)** [28]: aquesta eina s'encarrega de la part de desplegament dels serveis, en concret, s'encarrega de desplegar els contenidors generats a Docker. És una eina molt útil per mantindre la disponibilitat contínua del sistema.
4. **Visual Studio Code (VS code)** [29]: aquest editor de codi font m'ha servit per poder realitzar les tasques de desenvolupament dins del codi en el projecte.

4.2. Planificació temporal

En aquest apartat es mostra la planificació temporal inicial que va rebre el projecte i les seves desviacions.

La planificació temporal del projecte és un element clau per assegurar l'assoliment dels objectius i garantir-ne una execució adequada. Com que es tracta d'un projecte desenvolupat mitjançant un conveni amb una empresa, cal seguir les condicions establertes al contracte laboral.

En el conveni es va establir una dedicació total de **375 hores**, amb inici el **24 de febrer de 2025** i finalització el **30 de maig de 2025**. Aquesta dedicació implica una jornada de **6 hores diàries** durant els dies laborables.

Paral·lelament a aquestes hores de treball contractual, s'ha realitzat una dedicació addicional per part de l'estudiant fora de l'entorn laboral. Aquesta dedicació de **120 hores** ha permès donar suport a les tasques realitzades el dia a dia.

Un cop finalitzat el període de conveni, s'ha realitzat una dedicació de **15 hores setmanals** durant aproximadament **tres setmanes**, amb l'objectiu de finalitzar la memòria del projecte i preparar la seva defensa, previstes al voltant del **15 i 25 de juny de 2025 respectivament**.

En conjunt, la dedicació total aproximada del projecte és la següent:

- **375 hores** (contracte laboral)
- **120** (suport al projecte fora de l'entorn laboral)
- **45 hores** (15 hores setmanals durant 3 setmanes)

El que representa un total estimat de **540 hores**.

4.2.1. Rols i responsabilitats del projecte

Aquest projecte es va planificar seguint una metodologia **àgil híbrida**, principalment inspirada en **Scrum**, adaptada a les característiques específiques d'un Treball de Final de Grau realitzat per un únic estudiant en col·laboració amb una empresa. Per aquest motiu, els rols habituals de Scrum han estat reinterpretats per encaixar amb el projecte, mantenint les seves funcions bàsiques però adaptant-ne la responsabilitat i participació.

A continuació (Taula 2), es detallen els rols que intervenen i com s'han adaptat respecte als rols clàssics de Scrum:

Rol en el projecte	Adaptació de Scrum	Funcions principals
Cap de projecte (estudiant)	Development Team i Scrum Master	Responsable de l'anàlisi, desenvolupament, proves i documentació del projecte. Assumeix també tasques de gestió del projecte i seguiment.
Director del projecte	Product Owner	Supervisa el compliment dels objectius del projecte i dona feedback sobre l'evolució del projecte. Prioritza necessitats i vetlla per la qualitat del resultat final.
Cap de sinistres	Stakeholder clau / Usuari expert	Proporciona validació funcional sobre els resums generats i aporta coneixement del domini per ajustar la solució a les necessitats reals de l'equip de sinistres.
Agents de sinistres	Stakeholders / Usuari final	Avaluen les funcionalitats desenvolupades aportant feedback pràctic sobre l'ús real de la solució.
Ponent del projecte	Scrum Master (parcial)	Realitza seguiment acadèmic del projecte i participa en els Sprint Reviews, assegurant el progrés adequat dins del marc educatiu.

Taula 2: Resum de rols i responsabilitats. Font: Elaboració pròpia.

4.2.2. Fases del projecte

Com s'ha explicat anteriorment, el projecte es divideix en tres grans etapes ben diferenciades.

La primera etapa, **gestió de projectes**, té com a objectiu la definició i anàlisi del context del treball, així com la planificació temporal, l'estudi de riscos, el pressupost i la sostenibilitat. Aquesta fase està fortament definida pel calendari d'entregues acadèmiques i serà des del **24 de febrer fins al 19 de març**, període en què s'havia de desenvolupar la documentació i planificació inicial corresponent.

La segona etapa, **desenvolupament**, s'havia de dur a terme mitjançant una organització basada en sprints, adoptats de la metodologia **Scrum**. Comptant amb aproximadament **10 setmanes**, des del **20 de març fins al 30 de maig**, vam planificar un total de **5 sprints** de dues setmanes cadascun. Durant aquesta fase s'havia de desenvolupar la solució tecnològica i dur a terme tasques associades com reunions de seguiment i validació, així com l'elaboració de la memòria del projecte, que havia d'anar-se elaborant paral·lelament al desenvolupament.

Finalment, la tercera etapa, **documentació final**, estava definida un cop finalitzat el contracte laboral i s'estenia fins a la defensa del projecte, del **2 de juny al 25 de juny**. L'objectiu principal d'aquesta fase és consolidar i finalitzar la memòria del TFG, a més de preparar la presentació per a la lectura davant del tribunal.

4.2.3. Descripció de les tasques

En aquest apartat es detallen les tasques planificades del projecte.

Per a simplificar l'enteniment de les tasques les classificaré en diferents tipus: gestió de projectes (P), desenvolupament (T), documentació (D), validació (V), formació i recerca (F) i prompting (PMT).

A més, per cada una, donaré les dependències, la seva estimació en hores, recursos materials i recursos humans.

4.2.3.1. Gestió de projectes

P1. Contextualització i abast: Es construeix el document on es contextualitza i defineix el projecte, el seu abast i obstacles i, a més, es defineix la metodologia que se seguirà. **Dependències:** . **Estimació:** 23. **Recursos materials:** Ordinador, Google Docs. **Recursos humans:** Cap, ponent i director del projecte.

P2. Planificació temporal: S'escriu el document on es defineixen i descriuen les tasques del projecte, l'estimació per cada una i la distribució en el temps dins del marc del

projecte, utilitzant un diagrama de Gantt. **Dependències:** P1. **Estimació:** 23. **Recursos materials:** Ordinador, Google Docs. **Recursos humans:** Cap, ponent i director del projecte.

P3. Gestió econòmica i sostenibilitat: Es crea el document que inclou tant l'estimació de costos com la de sostenibilitat. **Dependències:** P2. **Estimació:** 23. **Recursos materials:** Ordinador, Google Docs. **Recursos humans:** Cap, ponent i director del projecte.

F1. Introducció a la IAG: Durant aquesta tasca buscaré informació per entendre una mica més a fons com funcionen les intel·ligències artificials generatives, permetent-me estar actualitzat en el marc del projecte. **Dependències:** . **Estimació:** 12. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte.

P4. Millores respecte el feedback rebut: Per a consolidar el lliurament final serà necessari aplicar totes les millores suggerides en la resta d'entregues. El que provocarà actualitzacions en el Product Backlog. **Dependències:** P3. **Estimació:** 23. **Recursos materials:** Ordinador, Google Docs. **Recursos humans:** Cap de projecte.

P5. Definició d'històries d'usuari: Definiré les funcionalitats del sistema com històries d'usuari i tasques, cadascuna amb els seus requisits. Integrant-ho tant a Notion com a Jira. Per fer-ho seguiré l'enfocament típic de la metodologia Scrum i passaran a formar part del Product Backlog del projecte. **Dependències:** P4. **Estimació:** 24. **Recursos materials:** Ordinador, Google Docs, Notion, Jira. **Recursos humans:** Cap i director del projecte, Cap de sinistres.

F2. Formació llibreries OpenAI: Aquesta tasca se centrarà a endinsar-me a les llibreries d'OpenAI, per poder saber els recursos o serveis que ofereixen. **Dependències:** F1. **Estimació:** 12. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte.

4.2.3.2. Desenvolupament

a. Sprint 1

T1. Recuperació de les dades de PostgreSQL: Es crearà un servei que permeti recollir totes aquelles dades rellevants del sinistre que estiguin emmagatzemades a la base de dades de PostgreSQL. **Dependències:** P5. **Estimació:** 15. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap de projecte.

F3. Formació llibreries OpenAI: En aquesta tasca buscaré una visió més tècnica que en F2 per a saber com introduir al codi les facilitats que proporciona la IA generativa.

Dependències: F2. **Estimació:** 10. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte.

T2. Primera integració amb la IAG: Amb el servei anterior preparat, es podrà fer una primera aproximació al resum del sinistre utilitzant la IAG i un prompt senzill, oferint així una primera solució parcial. Caldrà investigar sobre l'API i funcionalitats de la IA generativa.

Dependències: T1, F3. **Estimació:** 30. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap i director de projecte.

V1. Validació i feedback de la solució: Com que en aquest projecte no hi ha una forma directa de testejar, caldrà aquesta tasca per posar a prova els resultats, amb l'ajuda dels usuaris. Caldrà mantindre una reunió amb els agents i cap de sinistres per tal de poder avaluar la validesa dels resums. Se centrarà més en la informació proporcionada que en la resposta de la IA. El que provocarà actualitzacions en el Product Backlog. **Dependències:** T2. **Estimació:** 2. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres, Agent de sinistres.

T3. Adaptacions necessàries: Caldrà fer les adaptacions de dades o prompting necessàries per tal de satisfer les carències que han sorgit a V1. **Dependències:** V1. **Estimació:** 5. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap de projecte.

P6. Sprint review 1: Reunió de seguiment i fita entre el següent període. Avaluació del compliment dels objectius del sprint i de l'estat actual del projecte. Es valorarà l'estat del **Product Backlog**, actualitzant-lo amb els nous requisits, ajustos i prioritats sorgits a partir del feedback rebut durant l'sprint. **Dependències:** T3. **Estimació:** 2. **Recursos materials:** Ordinador, Jira, Git. **Recursos humans:** Cap, ponent i director de projecte.

D1. Desenvolupament memòria: Redactar la documentació referent al primer sprint: avenços, obstacle, etc. **Dependències:** P6. **Estimació:** 6. **Recursos materials:** Ordinador, Jira, Git, Google Docs. **Recursos humans:** Cap de projecte.

b. Sprint 2

F4. Introducció a DynamoDB: Per tal de garantir un bon avenç en la següent tasca del projecte, necessitaré conèixer DynamoDB, això, a més, implicarà una formació en noSQL. **Dependències:** P6. **Estimació:** 10. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte, Director del projecte.

T4. Recuperació de les dades de DynamoDB: Construcció d'un servei que permeti recuperar les transcripcions de les trucades i els mails amb l'usuari del sinistre, que estan a DynamoDB. **Dependències:** P6, F4. **Estimació:** 30. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap de projecte.

T5. Segona iteració del resum: Amb el servei anterior llest, es podrà fer la segona iteració en el resum, integrant així l'entrada de dades cap a l'API. **Dependències:** T4. **Estimació:** 15. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap de projecte.

V2. Validació i feedback de la solució: Com que en aquest projecte no hi ha una forma directa de testejar, caldrà aquesta tasca per posar a prova els resultats, amb l'ajuda dels usuaris. Caldrà mantindre una reunió amb els agents i cap de sinistres per tal de poder avaluar la validesa dels resums. El que provocarà actualitzacions en el Product Backlog. **Dependències:** T5. **Estimació:** 2. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres, Agent de sinistres.

T6. Adaptacions necessàries: Caldrà fer les adaptacions de dades o prompting necessàries per a satisfer les carències que han sorgit a V2. **Dependències:** V2. **Estimació:** 5. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap de projecte.

P7. Sprint review 2: Reunió de seguiment i fita entre el següent període. Avaluació del compliment dels objectius del sprint i de l'estat actual del projecte. Es valorarà l'estat del **Product Backlog**, actualitzant-lo amb els nous requisits, ajustos i prioritats sorgits a partir del feedback rebut durant l'sprint. **Dependències:** T6. **Estimació:** 2. **Recursos materials:** Ordinador, Jira, Git. **Recursos humans:** Cap, ponent i director de projecte.

D2. Desenvolupament memòria: Actualitzar la documentació de la memòria final, amb l'apòs a l'sprint: avenços, obstacle, etc. **Dependències:** P7. **Estimació:** 6. **Recursos materials:** Ordinador, Jira, Git, Google Docs. **Recursos humans:** Cap de projecte.

c. Sprint 3

T7. Recuperació i context dels fitxers a S3: Serà clau per poder oferir el millor resum possible la recuperació dels fitxers a S3. Amb aquests fitxers el que farem és buscar la generació d'un context per poder tindre constància quins fitxers s'han enviat i quina informació tenen. Crearem un servei que permeti fer aquesta integració. **Dependències:** P7. **Estimació:** 62. **Recursos materials:** Ordinador, Jira, VS code, Git. **Recursos humans:** Cap i director de projecte.

P8. Sprint review 3: Reunió de seguiment i fita entre el següent període. Avaluació del compliment dels objectius del sprint i de l'estat actual del projecte. Es valorarà l'estat del **Product Backlog**, actualitzant-lo amb els nous requisits, ajustos i prioritats sorgits a partir del feedback rebut durant l'sprint. **Dependències:** T7. **Estimació:** 2. **Recursos materials:** Ordinador, Jira, Git. **Recursos humans:** Cap, ponent i director de projecte.

D3. Desenvolupament memòria: Actualitzar la documentació de la memòria final, amb l'apèndix a l'sprint: avenços, obstacle, etc. **Dependències:** P8. **Estimació:** 6. **Recursos materials:** Ordinador, Jira, Git, Google Docs. **Recursos humans:** Cap de projecte.

d. Sprint 4

F5. Formació prompt engineering: Per tal de poder polir un bon resum serà clau acompanyar la següent tasca amb aprenentatge i tècniques de prompt-engineering. **Dependències:** P8. **Estimació:** 20. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte.

PMT1. Primera iteració del prompt final: Un cop ja sabem les dades que podem obtenir i el seu format serà clau preparar un prompt adient que permeti a la IA tractar les dades tal com toca i generar el resum més encertat possible. **Dependències:** P8. **Estimació:** 30. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres.

V3. Elecció del model adient: Per tal de garantir la màxima eficiència reduint el cost caldrà investigar i provar diferents models d'IAG. **Dependències:** PMT1. **Estimació:** 4. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte.

V4. Validació i feedback de la solució: Com que en aquest projecte no hi ha una forma directa de testejar, caldrà aquesta tasca per posar a prova els resultats, amb l'ajuda dels usuaris. Caldrà mantindre una reunió amb els agents i cap de sinistres per tal de poder avaluar la validesa dels resums. El que provocarà actualitzacions en el Product Backlog. **Dependències:** V3. **Estimació:** 4. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres, Agents de sinistres.

PMT2. Adaptacions necessàries: Per tal de garantir una eina útil dins el marc de treball, serà necessari fer les millores proposades pels usuaris, els agents de sinistres. **Dependències:** V4. **Estimació:** 2. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres.

V5. Validació i feedback de la solució: Com que en aquest projecte no hi ha una forma directa de testejar, caldrà aquesta tasca per posar a prova els resultats, amb l'ajuda dels usuaris. Caldrà una segona iteració de feedback per poder saber amb certesa si la solució compleixen els resultats esperats. **Dependències:** PMT2. **Estimació:** 2. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres, Agents de sinistres.

P9. Sprint review 4: Reunió de seguiment i fita entre el següent període. Avaluació del compliment dels objectius del sprint i de l'estat actual del projecte. Es valorarà l'estat del **Product Backlog**, actualitzant-lo amb els nous requisits, ajustos i prioritats sorgits a partir del feedback rebut durant l'sprint. **Dependències:** V5. **Estimació:** 2. **Recursos materials:** Ordinador, Jira, Git. **Recursos humans:** Cap, ponent i director de projecte.

D4. Desenvolupament memòria: Actualitzar la documentació de la memòria final, amb l'apèndix a l'sprint: avenços, obstacle, etc.... **Dependències:** P9. **Estimació:** 6. **Recursos materials:** Ordinador, Jira, Git, Google Docs. **Recursos humans:** Cap de projecte.

e. Sprint 5

T8. Integració al Back Office: Caldrà integrar el servei a l'eina que utilitzen els agents de sinistres per poder comprovar la situació (resum) en el flux real de treball. **Dependències:** P9. **Estimació:** 10. **Recursos materials:** Ordinador, VS Code, Git, Jira. **Recursos humans:** Cap de projecte.

V6. Validació i feedback de la solució: Com que en aquest projecte no hi ha una forma directa de testejar, caldrà aquesta tasca per posar a prova els resultats, amb l'ajuda dels usuaris. Caldrà rebre feedback per poder saber amb certesa si la solució compleixen els resultats esperats. El que provocarà actualitzacions en el Product Backlog. **Dependències:** T8. **Estimació:** 2. **Recursos materials:** Ordinador. **Recursos humans:** Cap i director de projecte, Cap de sinistres, Agents de sinistres.

T9. Generació de la cronologia: (Opcional) Caldrà fer un nou servei utilitzant tot l'apèndix a les iteracions que permeti fer el resum en format cronologia utilitzant la IAG. **Dependències:** V6. **Estimació:** 45. **Recursos materials:** Ordinador, VS Code, Git, Jira. **Recursos humans:** Cap de projecte.

P10. Sprint review 5: Reunió de seguiment i fita entre el següent període. Avaluació del compliment dels objectius del sprint i de l'estat actual del projecte. Es valorarà l'estat del **Product Backlog**, actualitzant-lo amb els nous requisits, ajustos i prioritats sorgits a partir del

feedback rebut durant l'sprint. **Dependències:** V6. **Estimació:** 2. **Recursos materials:** Ordinador, Jira, Git. **Recursos humans:** Cap, ponent i director de projecte.

D5. Desenvolupament memòria: Actualitzar la documentació de la memòria final, amb l'apèl a l'sprint: avenços, obstacle, etc. **Dependències:** P10. **Estimació:** 6. **Recursos materials:** Ordinador, Jira, Git, Google Docs. **Recursos humans:** Cap de projecte.

F7. Xerrada informativa als agents: Si volem que els agents millorin la seva eficiència operativa utilitzant l'eina, caldrà fer una sessió informativa. En aquesta tasca m'encarregaré de fer les gestions i preparacions per poder fer-ho. **Dependències:** P10. **Estimació:** 5. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte, Agents de sinistres.

4.2.3.3. Documentació

D6. Desenvolupament memòria: Actualitzar la documentació de la memòria final, adaptant-la als requisits d'una entrega com toca. **Dependències:** D5. **Estimació:** 10. **Recursos materials:** Ordinador, Google Docs. **Recursos humans:** Cap de projecte.

D7. Preparació lectura del projecte: Realitzar la presentació i pràctica convenients per la realització de la lectura final del projecte. **Dependències:** D6. **Estimació:** 40. **Recursos materials:** Ordinador. **Recursos humans:** Cap de projecte.

Un cop ja tenim totes les tasques ho simplifiquem en una taula (Taula 3), això permet una visió més clara al mateix temps que ajudarà a la construcció del diagrama de Gantt.

Tasca	Nom	Estimació (h)	Inici (2025)	Fi (2025)	Dependències	Risc	Recursos Materials	Recursos Humans
P1	Contextualització i abast	23	24/02	27/02		Baix	Ordinador, Google Docs	Cap, ponent i director del projecte
P2	Planificació temporal	23	28/02	04/03	P1	Baix	Ordinador, Google Docs	Cap, ponent i director del projecte
P3	Gestió econòmica i sostenibilitat	23	05/03	10/03	P2	Baix	Ordinador, Google Docs	Cap, ponent i director del projecte
F1	Introducció a la IAG	12	10/03	18/03		Baix	Ordinador	Cap de projecte
P4	Millores respecte el feedback rebut	23	11/03	18/03	P3	Baix	Ordinador, Google Docs	Cap de projecte
P5	Definició d'històries d'usuari	24	13/03	18/03	P4	Baix	Ordinador, Google Docs, Notion, Jira	Cap i director del projecte, Cap de sinistres
F2	Formació llibreries OpenAI	12	13/03	18/03	F1	Baix	Ordinador	Cap de projecte
T1	Recuperació de les	15	19/03	24/03	P5	Alt	Ordinador, Jira,	Cap de projecte

	dades de PostgreSQL						VS code, Git	
F3	Formació llibreries OpenAI	10	19/03	24/03	F2	Baix	Ordinador	Cap de projecte
T2	Primera integració amb la IAG	30	25/03	31/03	T1, F3	Baix	Ordinador, Jira, VS code, Git	Cap i director de projecte
V1	Validació i feedback de la solució	2	01/04	01/04	T2	Baix	Ordinador	Cap i director de projecte, Cap de sinistres, Agent de sinistres
T3	Adaptacions necessàries	5	01/03	01/03	V1	Baix	Ordinador, Jira, VS code, Git	Cap de projecte
P6	Sprint review 1	2	01/04	01/04	T3	Baix	Ordinador, Jira, Git	Cap, ponent i director de projecte
D1	Desenvolupament memòria	6	19/03	2/04	P6	Baix	Ordinador, Jira, Git, Google Docs	Cap de projecte
F4	Introducció a DynamoDB	10	03/04	03/04	P6	Baix	Ordinador	Cap de projecte, Director del projecte
T4	Recuperació de les dades de DynamoDB	30	03/04	08/04	P6, F4	Alt	Ordinador, Jira, VS code, Git	Cap de projecte
T5	Segona iteració del resum	15	09/04	11/04	T4	Baix	Ordinador, Jira, VS code, Git	Cap de projecte
V2	Validació i feedback de la solució	2	11/04	11/04	T5	Baix	Ordinador	Cap i director de projecte, Cap de sinistres, Agent de sinistres
T6	Adaptacions necessàries	5	11/04	11/04	V2	Baix	Ordinador, Jira, VS code, Git	Cap de projecte
P7	Sprint review 2	2	11/04	11/04	T6	Baix	Ordinador, Jira, Git	Cap, ponent i director de projecte
D2	Desenvolupament memòria	6	03/04	11/04	P7	Baix	Ordinador, Jira, Git, Google Docs	Cap de projecte
T7	Recuperació i context dels fitxers a S3	62	14/04	30/04	P7	Alt	Ordinador, Jira, VS code, Git	Cap i director de projecte
P8	Sprint review 3	2	02/05	02/05	T7	Baix	Ordinador, Jira, Git	Cap, ponent i director de projecte
D3	Desenvolupament memòria	6	14/04	02/05	P8	Baix	Ordinador, Jira, Git, Google Docs	Cap de projecte
F5	Formació prompt engineering	20	05/05	07/05	P8	Baix	Ordinador	Cap de projecte
PMT1	Primera iteració del prompt final	30	08/05	14/05	P8	Baix	Ordinador	Cap i director de projecte, Cap de claims
V3	Elecció del model adient	4	15/05	15/05	PMT1	Baix	Ordinador	Cap de projecte
V4	Validació i feedback de la solució	4	15/05	15/05	V3	Baix	Ordinador	Cap i director de projecte, Cap de sinistres, Agents de sinistres

PMT2	Adaptacions necessàries	2	15/05	15/05	V4	Baix	Ordinador	Cap i director de projecte, Cap de claims
V5	Validació i feedback de la solució	2	16/05	16/05	PMT2	Baix	Ordinador	Cap i director de projecte, Cap de sinistres, Agents de sinistres
P9	Sprint review 4	2	16/05	16/05	V5	Baix	Ordinador, Jira, Git	Cap, ponent i director de projecte
D4	Desenvolupament memòria	6	05/05	16/05	P9	Baix	Ordinador, Jira, Git, Google Docs	Cap de projecte
T8	Integració al Back Office	10	19/05	20/05	P9	Mitjà	Ordinador, Jira, VS code, Git	Cap de projecte
V6	Validació i feedback de la solució	2	20/05	20/05	T8	Baix	Ordinador	Cap i director de projecte, Cap de sinistres, Agents de sinistres
T9	Generació del cronologia	45	20/05	29/05	V6	Baix	Ordinador, Jira, VS code, Git	Cap de projecte
P10	Sprint review 5	2	30/05	30/05	V6	Baix	Ordinador, Jira, Git	Cap, ponent i director de projecte
D5	Desenvolupament memòria	6	19/05	30/05	P10	Baix	Ordinador, Jira, Git, Google Docs	Cap de projecte
F7	Xerrada informativa als agents	5	29/05	30/05	P10	Baix	Ordinador	Cap de projecte, Agents de sinistres
D6	Desenvolupament memòria	10	02/06	05/06	D5	Baix	Ordinador, Google Docs	Cap de projecte
D7	Preparació lectura del projecte	40	06/06	25/06	D6	Baix	Ordinador	Cap de projecte
Total		540						

Taula 3: Resum de les tasques. Font: Elaboració pròpia.

4.2.4. Diagrama de Gantt

A la Figura 4 podem veure les etapes del projecte: **Gestió de projectes** (del 24 de febrer al 19 de març), **Sprint 1** (del 20 de març al 2 d'abril), **Sprint 2** (del 3 al 14 d'abril), **Sprint 3** (del 15 d'abril al 2 de maig), **Sprint 4** (del 5 al 16 de maig), **Sprint 5** (del 18 al 30 de maig) i **Documentació final** (del 2 al 25 de juny).

Les tasques amb més risc també es veuen assenyalades al diagrama: amb **vermell** per aquelles d'alt risc i **taronja** per les de mitjà.

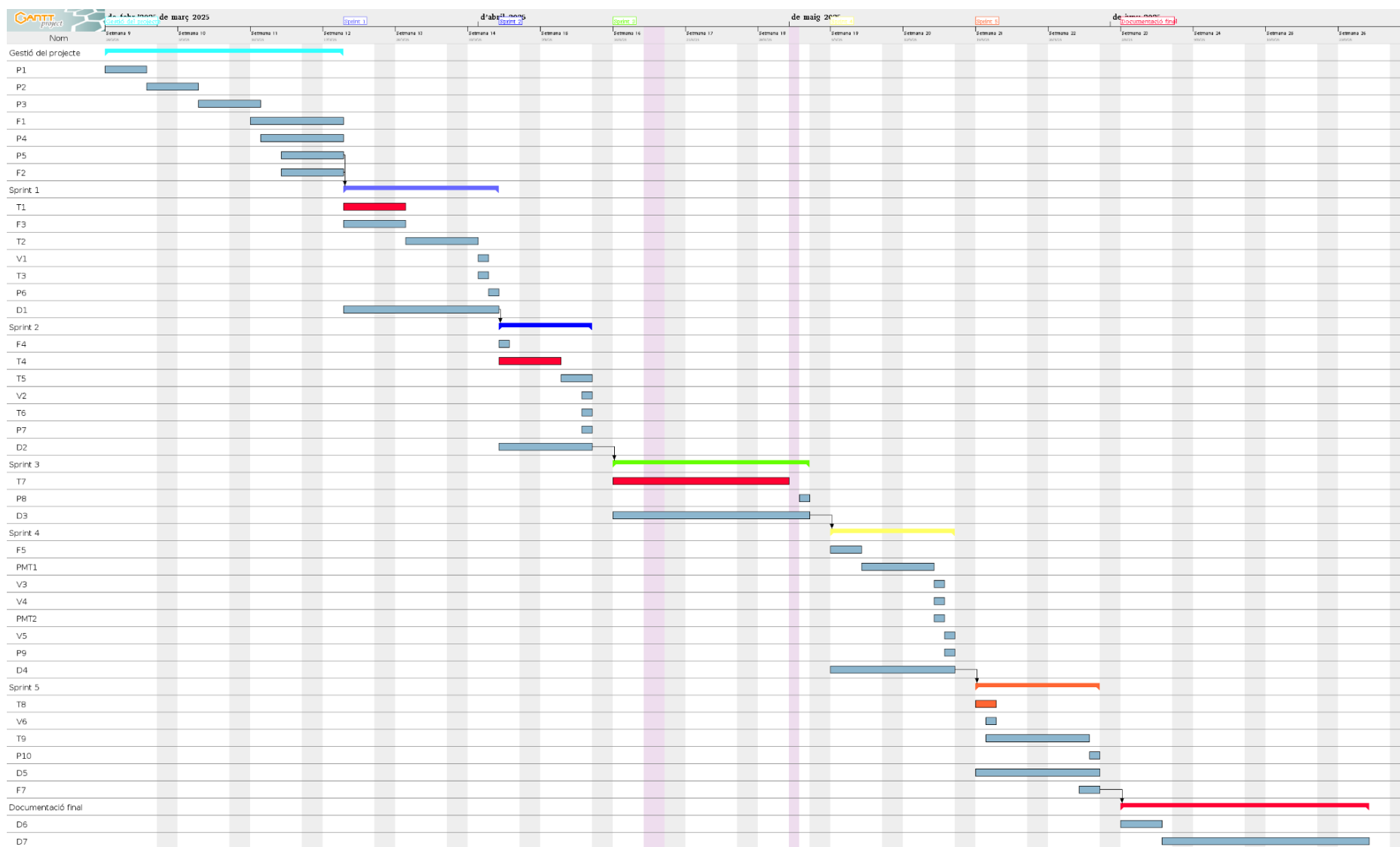


Figura 4: Diagrama de Gantt. Font: Elaboració pròpia, eina: GanttProject

4.2.5. Conclusions

La distribució realitzada s'ha basat en la dedicació prevista, considerant tant les estimacions de les tasques com els terminis de lliurament establerts.

Pel que fa a la funcionalitat prevista com a opcional per a l'Sprint 5 cal destacar que la seva no implementació no impactaria negativament ni en la satisfacció ni en l'objectiu principal del projecte. Aquesta funcionalitat s'ha proposat com una millora rellevant que s'inclourà al projecte sempre que no es produeixin endarreriments en la resta d'sprints.

Això ens proporciona un gran avantatge, temps per correccions, que permetrà assegurar-nos de que l'objectiu final del projecte es satisfarà.

4.2.6. Desviacions

La **sobrecàrrega en l'anàlisi de documents** ha requerit una solució estable i disponible en tot moment. Això ha portat a un desenvolupament complex en aquesta part, augmentant significativament el temps de producció. Cosa que ha tingut un efecte significatiu en la solució final. Amb possibles endarreriments ja previstos, he sigut capaç de controlar-los per arribar a la solució que plantejava l'objectiu final. L'Sprint 5 comptava amb una tasca opcional que feia la funció d'ampliació del projecte, aquesta tasca, "Generació del cronologia", no s'ha dut a terme.

4.3. Valoració d'alternatives i plans d'acció

En aquest apartat es presenten els plans d'acció portats a terme per abordar obstacles i riscos identificats a l'apartat **3**, concretament als subapartats **3.3 Obstacles** i **3.4 Riscos**. D'acord amb l'anàlisi prèvia, els principals factors de risc i obstacles detectats eren els següents:

- Errors en la connexió amb bases de dades i APIs.
- Problemes de rendiment.
- Gran quantitat de dades.
- Cost econòmic dels serveis externs.
- Falta de temps en el desenvolupament.
- Inexperiència en el desenvolupament del sistema.

Per minimitzar l'impacte d'aquests riscos i obstacles, s'han realitzat mesures preventives i correctives basades principalment en:

- **Sobreestimació de les hores** destinades a les tasques més crítiques, amb l'objectiu de disposar d'un marge temporal suficient per resoldre possibles incidències sense afectar el calendari global. Va ser una bona decisió fer aquesta sobreestimació ja que he après que no tot és tan fàcil com sembla.
- **Planificació iterativa** mitjançant metodologia àgil, que permet detectar i corregir problemes de manera progressiva i contínua al llarg del desenvolupament. Aquesta mesura m'ha acompanyat durant tot el desenvolupament per poder definir les prioritats de cada Sprint segons els avenços/retards trobats.
- **Formació específica** prèvia a les tasques més complexes, per reduir la probabilitat d'errors derivats de la inexperiència. He realitzat diverses formacions per poder adaptar correctament la meua eina a les necessitats del projecte, he realitzat formacions en integració d'IA Generativa i arquitectura hexagonal entre d'altres.
- **Control i optimització dels recursos** consumits pels serveis externs per evitar sobre costos innecessaris. En tot moment he estat conscient de l'ús que s'ha donat als serveis de pagament per evitar "fuites" i he mantingut un control periòdic dels càrrecs d'aquestes APIs.

A més, com s'ha explicat prèviament, per l'últim Sprint s'havien previst dues situacions:

1. **Desenvolupament i progrés correcte durant el projecte:** en aquest cas la força del Sprint 5 estaria en realitzar una nova funcionalitat alternativa, la cronologia.
2. **Problemes o endarreriments en el desenvolupament:** si hi hagués hagut problemes de desenvolupament o de compliment de dates estimades, hi hauria temps suficient en aquesta etapa de fer les adaptacions necessàries.

Finalment, com he explicat anteriorment, la tasca de gestió d'anàlisi de documents ha resultat més costosa de l'esperat i l'Sprint 5 servirà per poder acabar de desenvolupar aquesta part. Per tant, no es realitzarà la cronologia.

Aquest conjunt d'actuacions m'ha servit per garantir l'èxit del projecte, reduint l'impacte dels riscos detectats.

4.4. Gestió econòmica

En aquest apartat es mostra la estimació econòmica estimada inicialment que va rebre el projecte i les seves desviacions.

La gestió econòmica d'aquest projecte se centra a **planificar, estimar i controlar els costos** associats a les diferents fases del desenvolupament i implementació de la solució. A través d'una anàlisi detallada dels recursos materials i humans requerits, es busca garantir la **viabilitat financera** i una distribució òptima del pressupost.

Aquest apartat recull la **planificació pressupostària** del projecte, incloent-hi l'estimació de costos, la previsió d'imprevistos i contingències, i l'estratègia de control econòmic per assegurar un seguiment adequat de la despesa. Se segueix l'**estructura pressupostària** definida a la documentació del mòdul de gestió econòmica de GEP, garantint que la distribució de costos es realitzi de manera ajustada als criteris acadèmics i empresarials.

4.4.1. Recursos humans

Els recursos humans fan referència als diferents rols que es van definir en els apartats anteriors i que són necessaris per poder dur a terme les tasques descrites al diagrama de Gantt. Per cadascun dels rols he fet una aproximació salarial (Taula 4) basant-me en la informació que he pogut trobar a la web [30] i la proporcionada pels companys de treball.

Rol en el projecte	Salari brut (€/h)	Salari brut + SS (€/h)
Cap de projecte (estudiant)	10	13
Director del projecte	24,32	31,61
Cap de sinistres	18,37	23,88
Agent de sinistres	8,75	11,38
Tech Lead	17,84	23,19

Taula 4: Estimacions per rol i hora. Font: Elaboració pròpia

Amb les estimacions per rol i hora i la distribució de tasques ja podem fer una aproximació del cost per tasca dins el meu projecte. A la Taula 5 podem observar el resum.

Codi tasca	Nom	Hores totals	Hores per rol				Cost(euros)
			Cap de projecte	Director del projecte	Cap de sinistres	Agent de sinistres	
P1	Contextualització i abast	28	23	5			457,05
P2	Planificació temporal	28	23	5			457,05
P3	Gestió econòmica i sostenibilitat	28	23	5			457,05
F1	Introducció a la IAG	12	12				156
P4	Millores respecte el feedback rebut	23	23				299

P5	Definició d'històries d'usuari	32	24	4	4		533,96
F2	Formació llibreries OpenAI	12	12				156
T1	Recuperació de les dades de PostgreSQL	15	15				195
F3	Formació llibreries OpenAI	10	10				130
T2	Primera integració amb la IAG	32	30	2			453,22
V1	Validació i feedback de la solució	8	2	2	2	2	159,74
T3	Adaptacions necessàries	5	5				65
P6	Sprint review 1	4	2	2			89,22
D1	Desenvolupament memòria	6	6				78
F4	Introducció a DynamoDB	11	10	1			161,61
T4	Recuperació de les dades de DynamoDB	30	30				390
T5	Segona iteració del resum	15	15				195
V2	Validació i feedback de la solució	8	2	2	2	2	159,74
T6	Adaptacions necessàries	5	5				65
P7	Sprint review 2	4	2	2			89,22
D2	Desenvolupament memòria	6	6				78
T7	Recuperació i context dels fitxers a S3	64	62	2			869,22
P8	Sprint review 3	4	2	2			89,22
D3	Desenvolupament memòria	6	6				78
F5	Formació prompt engineering	20	20				260
PMT1	Primera iteració del prompt final	33	30	1	2		469,37
V3	Elecció del model adient	4	4				52
V4	Validació i feedback de la solució	4	4				52
PMT2	Adaptacions necessàries	4	2	1	1		81,49
V5	Validació i feedback de la solució	8	2	2	2	2	159,74
P9	Sprint review 4	4	2	2			89,22

D4	Desenvolupament memòria	6	6				78
T8	Integració al Back Office	10	10				130
V6	Validació i feedback de la solució	8	2	2	2	2	159,74
T9	Generació del cronologia	45	45				585
P10	Sprint review 5	6	2	2	2		136,98
D5	Desenvolupament memòria	6	6				78
F7	Xerrada informativa als agents	5	5				65
D6	Desenvolupament memòria	10	10				130
D7	Preparació lectura del projecte	40	40				520
Total		609	540	44	17	8	8.907,84

Taula 5: Estimació de costos per tasca. Font: Elaboració pròpia.

4.4.2. Recursos materials

A més dels recursos humans, serà fonamental tenir en compte els costos materials associats al projecte, així com les eines i l'espai de treball necessaris per al seu desenvolupament. Això inclou despeses derivades de l'adquisició de tecnologia necessària per dur a terme les activitats previstes.

A més, per garantir una gestió financera eficient i sostenible, caldrà considerar l'amortització de tots aquests recursos. L'amortització permet distribuir el cost d'aquests actius al llarg de la seva vida útil, reflectint-ne la depreciació i optimitzant-ne l'impacte econòmic dins del pressupost del projecte.

$$Amortització(euros) = \frac{Cost(euros) \times Duració\ del\ projecte(hores)}{Vida\ útil(anys) \times Dies\ laborables\ anuals \times Dedicació\ diària(hores)}$$

Per aquesta fórmula direm que els dies laborables són 220, la dedicació diària és de 8 hores i la duració del projecte és de 540.

A la Taula 6 observem l'amortització dels recursos materials.

Recurs	Preu(euros)	Vida útil	Amortització(euros)
Dell Inc. XPS 13 7390	1500	5 anys	92
Pantalla BENQ	175	5 anys	10

Teclat i ratolí	50	3 anys	3
-----------------	----	--------	---

Taula 6: Amortització dels recursos materials. Font: Elaboració pròpia.

A banda dels recursos materials, també és necessari incloure en el pressupost els costos associats a l'espai de treball, en aquest cas, el meu lloc a les oficines de Cleverea. El cost mensual d'un lloc de treball en aquestes instal·lacions, incloent-hi subministraments com llum, aigua, wifi i altres serveis, és de 120 euros. Com que ocuparé aquest espai durant quatre mesos, el cost total serà de **480 euros**.

4.4.3. Recursos software

Per integrar-me a l'entorn de treball de Cleverea, és imprescindible utilitzar el programari corporatiu que l'empresa té implementat. Per aquest motiu, cal considerar en el pressupost els costos associats a les subscripcions d'aquestes eines, ja que són essencials per al desenvolupament de les meves tasques dins de l'organització. Podem veure el resum de costos a la Taula 7.

Recurs	Preu mensual(euros)	Cost projecte(euros)
GitHub	19,35	77,4
Notion	14	56
Slack	11,75	47
CircleCI	109,7	438,8
Total	154,8	619,2

Taula 7: Costos dels recursos software. Font: Elaboració pròpia.

4.4.5. Imprevistos i contingències

En la definició de riscos es van identificar els possibles escenaris i es va establir un pla de mitigació per minimitzar-ne l'impacte, la sobreestimació de tasques i el marge de millora del Sprint 5. Tot i això, en aquest apartat també caldrà considerar els costos addicionals que podrien sorgir en cas que imprevistos superin les previsions del pla de mitigació inicial. Aquest enfocament permetrà una millor gestió dels imprevistos i garantirà la viabilitat del projecte davant situacions no contemplades.

4.4.5.1. Gestió dels errors en la connexió amb bases de dades i APIs

Tot i no presentar un cost en el pressupost del projecte cal tindre en compte sempre quins errors poden arribar a donar els serveis externs i ser capaços de gestionar-los de la forma correcta per cada cas.

4.4.5.2. Gestió del dels problemes de rendiment

Aquesta gestió no és suposa directament un canvi en el pressupost del projecte, de totes maneres, ha estat essencial per buscar sempre la forma més eficient per, recollir les dades o en la pròpia generació del resum.

4.4.5.3. Gestió de la falta de temps en el desenvolupament

Tot i haver realitzat una estimació detallada de les tasques, factors com la inexperiència o possibles imprevistos podrien provocar retards en el desenvolupament del projecte. Per aquest motiu, és important preveure un marge de seguretat en el pressupost per cobrir una possible dedicació extra. En aquest cas, destinaré un 15% addicional en concepte de contingència sobre el rol de cap de projecte en els recursos personals.

4.4.5.4. Gestió del gran volum de dades

Un gran volum de dades pot ser crític ja que els recursos informàtic són limitats. Aquesta gestió no ha suposat un cost addicional en el desenvolupament però sí ha requerit mantenir un control en la quantitat de dades que gestionem i la quantitat de dades que les eines externes poden gestionar.

4.4.5.5. Gestió de les subscripcions

Tot i conèixer els costos de les subscripcions, poden produir-se desviacions en el seu preu a causa d'actualitzacions de tarifes o canvis en les condicions del servei. Per aquest motiu, hem previst un marge addicional del 10% sobre el cost total per cobrir possibles increments.

4.4.5.6. Reparació d'equips

Com que per realitzar el projecte són essencials els recursos materials prèviament esmentats, caldrà preparar-nos pel risc de reparacions o substitucions. Tot i que els dispositius es troben dins de la seva vida útil, sempre existeix la possibilitat que es produeixin fallades. Per aquest motiu, hem incorporat al pressupost una previsió de costos basada en la probabilitat d'error. En aquest cas, s'ha estimat un cost de reparació de 150 euros amb una probabilitat de fallada del 10%, la qual cosa implica un cost esperat de 15 euros.

4.4.6. Pressupost final

Si calculem el pressupost final sumant tots els costos previstos, podem obtenir una visió clara de la despesa total del projecte. A continuació (Taula 8), es presenten els resultats detallats.

Concepte	Cost (euros)
Recursos humans	8.907,84
Recursos materials	105
Instal·lacions i serveis	480
Recursos software	619,2
Gestió falta de temps	1053
Gestió de les subscripcions	61,92
Reparació d'equips	15
Total	11.241,96

Taula 8: Pressupost final. Font: Elaboració pròpia.

Com podem observar el cost total és d' **11.241,96 euros**, en el marc d'aquest projecte no hem de tindre en compte l'IVA perquè la solució serà per la mateixa empresa.

4.4.7. Control de gestió

Un cop definit el pressupost del projecte i establerts els plans de contingència, és fonamental determinar els mecanismes de control que permetran evitar desviacions respecte al pla proposat. Aquests mecanismes consistiran en identificadors numèrics que seran calculats i avaluats durant el desenvolupament del treball.

Per aquest projecte tindrem els següents indicadors:

- **Desviació en les hores de les tasques:**
Per cada tasca: *Hores estimades - Hores reals*
- **Desviació dels costos en recursos humans:**
Per tasca: *Cost estimat - Cost real*
- **Desviació total costos generals:**
Cost general estimat - Cost general real
- **Desviació total costos imprevistos:**
Cost imprevistos estimat - Cost imprevistos real
- **Desviació total hores:**
Hores totals estimades - Hores totals reals
- **Desviació total costos:**
Cost total estimat - Cost total real

4.4.8. Desviacions

No s'han ocasionat desviacions rellevants en els costos del projecte respecte a les planificades.

5. Sostenibilitat

L'anàlisi realitzada mitjançant l'enquesta sobre sostenibilitat ha evidenciat mancances meves en l'aplicació pràctica dels conceptes en l'àmbit de la informàtica. Això m'ha permès reflexionar sobre com el meu projecte pot contribuir a un desenvolupament sostenible en les seves tres dimensions: econòmica, ambiental i social.

5.1. Dimensió econòmica

El desenvolupament necessari per la posada en producció del projecte suposa una inversió inicial considerable, però es veu compensada per l'estalvi operatiu a mitjà termini. L'automatització de tasques manuals i l'optimització dels recursos humans contribueixen a disminuir els costos de manera significativa. Permetrà millorar l'eficiència sense haver de fer augments en el personal.

Pel que fa a la vida útil del sistema, la optimització de recursos i, per tant, costos externs, garanteix que l'impacte econòmic a llarg termini sigui sostenible. El disseny permet incorporar millores i adaptar-se a nous requisits sense necessitat de fer grans variacions en el sistema, fet que manté els costos de manteniment baixos. Al mateix temps garanteix una facilitat en variacions en els models d'IAG que s'utilitzen que permet adaptar-se en temps real a les variacions de les tecnologies existents. En conjunt, es tracta d'un projecte amb viabilitat econòmica tant a curt com a llarg termini.

5.2. Dimensió ambiental

Tot i que la infraestructura necessària per posar en funcionament el sistema d'IA comporta un cert impacte ambiental inicial, aquest s'ha contemplat des del disseny del projecte. L'ús de serveis al núvol compromesos amb la sostenibilitat i l'optimització de l'arquitectura del sistema són estratègies clau per reduir la petjada ecològica. Així, es minimitza el consum energètic i es redueix l'ús innecessari de recursos computacionals.

En relació a la seva vida útil, el projecte està pensat per evitar la generació de residus digitals i mantenir infraestructura lleugera i eficient. L'ús de models d'IA amb requeriments moderats de consum i una gestió responsable de les dades garanteixen un impacte ambiental limitat al llarg del temps.

5.3. Dimensió social

Des del moment en que el projecte entri en producció, es generarà un impacte social positiu tant per als treballadors com per als clients de Cleverea. La càrrega de treball dels agents de sinistres es reduirà i les seves tasques seran més centrades en accions de valor.

A llarg termini, la clau per mantenir aquest impacte positiu serà l'adaptabilitat del sistema i la formació contínua dels usuaris. Garantint una eina accessible es podrà evitar la bretxa digital i es maximitzarà l'aprofitament de la solució. En conclusió, el projecte no només respon a una necessitat real de la companyia, sinó que també contribueix a millorar la qualitat de vida laboral i l'atenció al client dins d'aquesta.

6. Requisits i especificació

Aquest capítol defineix què ha de fer el sistema i amb quines condicions de qualitat ho ha de fer. La funcionalitat aparent és senzilla “generar un resum en temps real de l’estat d’un sinistre”, però implica processos interns complexos de recuperació d’informació i inferència. Per garantir que l’equip de desenvolupament i els stakeholders comparteixen les mateixes expectatives, es documenten tant els **requisits funcionals** com els **no funcionals**, les **restriccions de dades** i els **criteris objectius de prova**.

6.1. Cas d’ús UC-01 – Obtindre resum de sinistre

La solució ofereix una sola acció visible per a l’agent: obtenir el resum d’un sinistre. La simplicitat aparent amaga la lògica interna de cerca, inferència i caché; per això es modela UC-01, que descriu el que l’actor percep sense exposar detalls tècnics. En la següent figura podem veure el diagrama UML.



Figura 5. Diagrama UML del cas d’ús UC-01. Font: Elaboració pròpia.

En la següent taula podem veure la definició d’aquest cas d’ús amb: objectiu, actor principal, precondicions, possibles fluxos, postcondicions i requisits associats.

Element	Descripció
Objectiu	L’agent de sinistres vol obtenir en < 30 s un resum amb tota la informació rellevant i les accions pendents d’un sinistre.
Actor principal	Agent de sinistres.
Precondicions	1. L’agent ha iniciat sessió al Back-Office intern de Cleverea. 2. Existeix un sinistre amb el <code>claim_id</code> .
Flux principal de successos	1. L’agent prem “Generar resumen”. 2. El sistema comprova si hi ha un resum vàlid en memòria caché (≤ 24 h). 3. (a) Si hi és, el recupera. -> Pas 5 3. (b) Si no hi és, crea una tasca al Servei, espera la resposta i desa el resultat en caché. 4. El sistema mostra un indicador de progrés fins que arriba el resum. 5. El resum es visualitza al mateix Back-Office.

Fluxos alternatius	A – Error en la generació 4a. El sistema rep un error. 4b. Mostra imatge “No s’ha pogut generar el resum. Pel següent motiu:” i es mostra el motiu de l’errada.
Postcondicions	Resum en caché amb durada de 24 h.
Requisits associats	RF-1, RF-2, RF-3, RF-4, RNF-1, RNF-2

Taula 9. Cas d'ús UC-01. Font: Elaboració pròpia.

6.1.1. Històries d'usuari

Les històries d'usuari que mostra la següent taula expressen la mateixa funcionalitat des del punt de vista de valor de negoci. A cada HU s'hi indica quins requisits cobreix i quines proves ho validen, creant la primera capa de traçabilitat.

ID	Història	RF/RNF coberts
HU-1	Com a agent de sinistres vull prémer “Generar resum” per a conèixer l'estat del sinistre en menys de 30 s.	RF-1, RF-2, RF-3, RNF-1
HU-2	Com a agent de sinistres vull poder consultar un resum que ja existeix per treballar més ràpid i reduir costos.	RF-4, RNF-5
HU-3	Com a cap de sinistres vull poder consultar els costos associats a la generació de resums de sinistres per a controlar el pressupost.	RNF-5

Taula 10. Històries d'usuari. Font: Elaboració pròpia.

6.2. Requisits funcionals (RF)

Tot i que l'usuari disposa d'un únic botó, la funcionalitat es desplega en diverses capacitats observables que es poden provar de forma aïllada.

A partir d'UC-01 es deriven requisits funcionals observables. Cada RF descriu què fa el sistema, sense entrar en com ho implementa, i es provarà de manera independent en la secció 6.5. Criteris d'acceptació i proves.

Codi	Requisit
RF-1 Sol·licitud	L'agent de sinistres pot sol·licitar la generació d'un resum per a un sinistre concret directament des del Back-Office.
RF-2 Visualització	El sistema mostra el resum dins la mateixa interfície, sense necessitat de canviar d'aplicació ni pestanya.
RF-3 Contingut mínim	El resum inclou, com a mínim: estat actual del sinistre i accions pendents, així com un resum de les últimes

	comunicacions i els documents disponibles.
RF-4 Recuperar resum existent	Si ja existeix un resum recent (últimes 24 hores) per al sinistre, l'agent el pot consultar directament sense haver de generar-ne un de nou.

Taula 11. Requisits funcionals. Font: Elaboració pròpia.

L'RF-4 no implica cap acció extra per part de l'agent: el sistema decideix automàticament entre regenerar o recuperar de caché, tal com apunta el pas 2 del flux principal.

6.3. Requisits no funcionals (RNF)

Tot i que la càrrega de dades pot variar molt entre sinistres, el servei ha de mantenir una experiència raonable per a l'agent que definirem en aquesta secció. Més endavant s'especificarà la mesura i validació d'aquests requisits.

Codi	Requisit
RNF-1 Rendiment	El 95 % de les peticions han de rebre resposta en ≤ 28 s; el 100% en < 35 s. Avaluat sobre les peticions rebudes en els 30 dies post-desplegament.
RNF-2 Disponibilitat	El servei ha de garantir una disponibilitat $\geq 99,5$ %. Monitoritzat a través de les peticions habituals amb l'eina habitual de Cleverea, Kibana ³ .
RNF-3 Usabilitat	En proves d'acceptació, ≥ 90 % dels resums han de ser valorats com "útils" pels agents, mitjançant una enquesta de valoració. Es consideraran útils aquells resums valorats amb $\geq 4/5$.
RNF-4 Seguretat	Accés restringit a usuaris autenticats i canals de trànsit de dades xifrats. Dades delicades protegides fora del codi.
RNF-5 Escalabilitat	El sistema ha de poder gestionar fins a 300 peticions/dia.
RNF-6 Eficiència operativa	Temps mig de resposta de l'agent ≤ 10 min i ≥ 25 sinistres/dia gestionats per agent. Avaluats 15 dies post-desplegament.
RNF-7 Satisfacció interna	Enquesta interna $\geq 8/10$ i ≥ 85 % d'agents declaren sentir-se segurs abans de respondre. Avaluat 15 dies post-desplegament.
RNF-8 Control de costos	El sistema ha de disposar de mecanismes de protecció automàtica davant increments sobtats de demanda no habituals a Cleverea. Despesa mensual de l'API ≤ 300 €.
RNF-9 Monitoratge d'errors i alarmes	≥ 95 % dels errors s'han de registrar i el 99% dels crítics han de generar alarmes notificades automàticament.

Taula 12. Requisits no funcionals. Font: Elaboració pròpia.

³ Kibana és una eina dissenyada per la visualització i exploració de dades. [60]

6.4. Requisits de dades i restriccions (RD)

Aquest subapartat delimita quines dades pot consumir el sistema i quines salvaguardes de privacitat s'apliquen en forma de taula.

Codi	Requisit
RD-1 Fonts de dades	El sistema consumeix: PostgreSQL (dades estructurades del sinistre), DynamoDB (trucades i correus) i S3 (documents i imatges)
RD-2 Minimització de dades personals.	Evasió de dades personals en els processos que impliquen fonts externes. S'evita enviar informació de documents d'identitat, números de telèfon, noms i adreces de correu electrònic a sistemes externs.
RD-3 Idioma	Tots els resums es generen i es mostren únicament en castellà, idioma de treball de l'equip.

Taula 13. Requisits de dades i restriccions. Font: Elaboració pròpia.

6.5. Criteris d'acceptació i proves

Per convertir els requisits en proves objectives es defineixen els següents escenaris (Taula 14. Criteris d'acceptació i proves. Font: Elaboració pròpia.). Durant la fase de validació es verifica el compliment dels criteris de superació.

RF/RNF	Escenari de prova	Criteri de superació
RF-2 + RNF-1	Finalització de la tasca.	Es mostra al Back-Office el resum en ≤ 28 s (95 %) i ≤ 35 s (100 %).
RF-3	Revisió manual d'una mostra de 20 resums.	Per ≥ 18 es valida la seva veracitat i utilitat. A càrrec del cap de sinistres, amb suport dels agents que consideri.
RF-4	Recuperació de resum ja existent recent.	Si s'ha generat un resum pel mateix sinistre en les últimes 24 h aquest resum es recupera directament.
RNF-2	Disponibilitat durant 15 dies.	Disponibilitat $\geq 99,5$ %.
RNF-3	Usabilitat amb agents.	≥ 90 % dels resums valorats com "útils" ($\geq 4/5$).
RNF-4	5 proves de crides analitzades detalladament abans del desplegament. Construïdes amb informació fictícia.	Totes les peticions xifrades i compleixen polítiques d'autenticació. Les claus API estan emmagatzemades de forma correcta sense riscos de seguretat. La informació enviada als serveis externs no conté dades delicades.
RNF-5	Prova de càrrega amb 300 peticions en 24 hores.	100 % de peticions compleixen RNF-1 sense errors.

RNF-6	Anàlisi operativa 15 dies post-desplegament.	Temps mig de resposta ≤ 10 min i ≥ 25 sinistres/dia per agent.
RNF-7	Enquesta interna 15 dies post-desplegament.	Resultats $\geq 8/10$ i en ≥ 85 % dels casos els agents manifesten confiança abans de respondre.
RNF-8	Consulta del monitoratge de costos.	Els nous sistemes integrats permeten monitoritzar el 100% dels costos generats respecte el seu ús.
RNF-9	Monitoratge d'errors i alarmes.	$\geq 95\%$ dels errors s'han de registrar i el 99% dels crítics han de generar alarmes notificades automàticament.

Taula 14. Criteris d'acceptació i proves. Font: Elaboració pròpia.

6.6. Model de dades

Tot i que habitualment en aquesta secció s'inclouria un model conceptual de dades, en aquest projecte s'ha optat per descriure la gestió de les dades i la seva estructura dins d'apartats posteriors. Aquesta decisió es deu al fet que el sistema es basa en una arquitectura distribuïda, on les dades ja formen part de models ja definits prèviament dins l'entorn de Cleverea.

7. Tecnologies emprades

Aquest capítol descriu amb detall, les tecnologies, llibreries i serveis que sustenten la solució, justificant la seva elecció i exposant els avantatges, riscos i resultats obtinguts.

7.1. OpenAI Platform

7.1.1. Context i motivació

OpenAI Platform [38] ha estat la tecnologia clau per satisfer el requisit funcional RF1 i el RNF2. El servei d'OpenAI proporciona accés a models avançats i estables de llenguatge natural (LLMs) que permeten interpretar grans volums i sintetitzar-ne la informació de manera coherent i comprensible.

Aquesta eina s'ha integrat com a servei extern, que permet enviar-hi text estructurat i rebre'n com a resposta un resum textual. La seva capacitat per entendre i analitzar contingut en poc temps de forma precisa l'han fet especialment eficaç per la solució. També ha ofert un gran potencial en l'anàlisi d'imatges i documents PDF.

A més de generar resums textuais, OpenAI s'ha utilitzat per analitzar imatges fixes i documents aportats pels clients (fotografies del sinistre, escanejats, etc.), ja que els models GPT-4 amb visió permeten extreure informació semàntica útil d'aquests formats.

7.1.2. Models

Dins d'OpenAI, un dels aspectes més importants ha estat la tria del **model** adequat.

Quan seleccionem un model hem de tenir presents quatre variables bàsiques: **velocitat**, **intel·ligència**, **cost** i **mida de la finestra de context** (quanta informació és capaç de processar el model en una sola interacció). En la següent taula podem veure una comparativa de característiques generals entre alguns dels models més rellevants disponibles. Totes les dades de la taula són extretes directament de la documentació oficial d'OpenAI.

Model	Finestra (tokens)	Cost entrada-sortida (\$/1M tokens)	Velocitat	Intel·ligència	Decisió final
GPT-4	8.192	30 – 60	Mitjana	Normal	Descartat (context curt i cost elevat)
GPT-4 Turbo	128.000	10 – 30	Mitjana	Normal	Descartat (cost elevat)

GPT-4.1	1.047.576	2 - 8	Mitjana	Molt alta	Escollit
GPT-4.1 mini	1.047.576	0,4 – 1,6	Alta	Alta	Descartat (qualitat baixa)
GPT-4o mini	128.000	0,15 – 0,6	Alta	Alta	Descartat (qualitat baixa)

Taula 15. Comparació models OpenAI. Font: <https://platform.openai.com/docs/models>

La finestra de cada model és la quantitat d'informació que es pot processar en una sola interacció. Per poder mesurar aquesta informació s'utilitzen *tokens*⁴ que representen la informació que processa el model d'IA. El cost de cada crida depèn de la quantitat de *tokens* s'han introduït i la quantitat de *tokens* que s'han generat.

La velocitat i intel·ligència⁵ són mètriques que encasellen únicament els models en una comparació entre altres, però no defineixen la capacitat de cada un de forma precisa.

Els models més antics (GPT-4 i GPT-4 Turbo) s'han descartat directament per no ser tan accessibles com la resta, a més, amb l'aparició dels nous models, OpenAI els encasella com a obsolets.

Les versions mini (GPT-4.1 mini i GPT-4o mini) de models més potents han estat descartades ja que, després de poques proves, es va veure la seva baixa utilitat. Durant proves inicials amb casos reals, com un sinistre ja resolt, alguns models mini van interpretar erròniament l'estat del cas, indicant que encara es requeria acció per part del client. Aquest error evidenciava mancances en la seva capacitat de comprendre la temporalitat de les interaccions (raonament cronològic), motiu pel qual es van descartar. Un dels exemples més clars va ser un sinistre ja solucionat pel qual el model deia que es seguia requerint informació per part del client, això era degut a la mala interpretació que feia del conjunt de dades.

L'elecció del model GPT-4.1 va resultar senzilla un cop analitzades les característiques generals i fetes les proves mencionades anteriorment, on deixava el model més nou en una escala superior a la resta.

Els resultats detallats d'aquestes proves, incloent-hi precisió i coherència del resum generat amb cada model, es recullen al capítol.

⁴ Un *token* és una unitat bàsica de text (com una paraula o part d'una paraula) que una IA processa per entendre i generar llenguatge. [59]

⁵ Quan parlem d'intel·ligència o raonament d'una IA ens referim al procés de deduir conclusions lògiques i fer prediccions a partir dels coneixements disponibles. [58]

7.2. Twelve Labs

7.2.1. Context i motivació

Twelve Labs [42] és una plataforma especialitzada en l'anàlisi de contingut audiovisual mitjançant intel·ligència artificial. En aquest projecte, s'ha utilitzat per extreure informació rellevant dels vídeos que els clients aporten com a prova en la gestió de sinistres, especialment en casos d'accidents de trànsit.

La finalitat d'aquesta anàlisi automàtica és incorporar informació visual rellevant dins el resum complet del sinistre, que el sistema genera amb OpenAI. D'aquesta manera, l'agent pot disposar d'una visió enriquida i contextualitzada, amb dades visuals sintetitzades que complementen la resta de la informació.

Els aspectes específics que s'esperen extreure són:

- Detectar visualment parts del vehicle (o altres objectes) afectades.
- Identificar presència de tercers a l'escena.
- Interpretar l'explicació dels fets aportada pel client.

La tria d'aquesta eina es va considerar adequada pel fet que permet analitzar tant el contingut visual com l'auditiu d'un vídeo de manera conjunta, cosa que no és possible actualment amb la versió API d'OpenAI. Durant la fase d'exploració es van estudiar diverses alternatives, com ara processar el vídeo de manera fragmentada, separant-ne l'àudio i el vídeo per analitzar-los de forma independent amb models d'OpenAI. No obstant això, aquesta solució es va descartar per la seva complexitat i fiabilitat.

A diferència de les imatges o documents fixos (que es processen amb OpenAI, vegeu 7.1. OpenAI Platform), els vídeos requereixen una eina capaç d'analitzar seqüències temporals, mantenint context i identificant canvis entre imatges concretes dins una successió d'imatges en moviment.

Els vídeos que han de ser processats es troben en format *.mp4* i estan emmagatzemats a AWS S3 (vegeu 8.1.3. Fitxers associats (Amazon S3)), associats a cada sinistre. El sistema envia a l'eina l'URL de cada vídeo, que es processa de forma asíncrona per obtenir-ne l'anàlisi visual.

Per això, Twelve Labs ha estat l'eina seleccionada per al processament de vídeos, atès que proporciona:

- **Sortida estructurada** en format JSON, que permet integrar la sortida fàcilment al context del sinistre i enviar-la a OpenAI.
- **Descripcions textuais**, que permeten obtenir informació descriptiva que serà útil per complementar el resum generat amb OpenAI. No es va identificar cap altra eina que retornés text estructurat amb grau suficient de detall per aquesta aplicació.

La integració d'aquesta anàlisi visual permet reduir la dependència del criteri humà inicial i millora la qualitat del resum final del sinistre, facilitant una resposta més ràpida i informada per part dels agents.

7.3. Infraestructura addicional

En aquesta secció es descriu la infraestructura de suport emprada per garantir **seguretat (RNF-4)**, **disponibilitat (RNF-2)** i **control de costos (RNF-8)** quan la solució interactua amb serveis externs.

7.3.1 Gestió de secrets

En el context d'aquest TFG, un secret és qualsevol dada sensible que no hauria de formar part del codi font, com ara claus d'API o credencials d'accés. A nivell d'Enginyeria del Software, separar la configuració sensible del codi és una bona pràctica fonamental per garantir la seguretat, portabilitat i mantenibilitat del sistema.

Per tal de protegir aquesta informació sensible, s'ha emprat **AWS Secrets Manager** [53], eina ja centralitzada a Cleverea per la gestió de secrets. Ofereix un servei per emmagatzemar i accedir a secrets de forma segura i controlada.

7.3.2. Gestió de paràmetres i throttling

El cost dels serveis externs de pagament amb què interactua el sistema depèn directament de les peticions que es fan a aquests. És per això que s'ha adoptat una configuració dinàmica de mantenir el sistema estable i adaptable limitant les peticions, adoptant una estratègia de configuració dinàmica a partir de paràmetres externs al codi.

Aquesta configuració es defineix mitjançant variables d'entorn emmagatzemades a AWS, fet que permet ajustar el comportament del sistema sense modificar-ne el codi. Això segueix les bones pràctiques ja normalitzades a l'entorn de Cleverea de separació entre configuració i implementació.

Un concepte rellevant és el *throttling*, que en el context del projecte fa referència a la limitació explícita del nombre de peticions que es poden fer per unitat de temps a un servei extern. Aquest mecanisme permet evitar errors o atacs de força bruta que puguin acabar en un sobre cost inesperat.

7.4. Processament asíncron amb Celery i Redis

Celery és un sistema de gestió de tasques asíncrones que permet distribuir treball en segon pla mitjançant cues de missatges [61]. Redis, al seu torn, és una base de dades en memòria clau-valor sovint usada com a broker (gestor intermediari) per a aquestes cues de Celery [62], ja que és molt ràpida gestionant missatges.

Dins l'arquitectura de Cleverea, Celery i Redis treballen conjuntament per realitzar tasques de forma asíncrona, desacoblant així fluxos alternatius dels fluxos principals, sense saturar el servei principal.

El funcionament d'aquestes dos eines és el següent: El servei la posa a la cua de Celery (on Redis actua de gestor), i un worker extern l'executa sense fer esperar l'usuari. Aquest parell d'eines proporciona escalabilitat i eficiència, ja que permet processar múltiples tasques en paral·lel i controlar la càrrega del sistema, alhora que contribueix a complir requisits de rendiment (respostes àgils) i fiabilitat en la nostra arquitectura.

7.5. Conclusions

Les tecnologies emprades s'han seleccionat per la seva capacitat d'alinejar-se amb els requisits del sistema, tant funcionals com no funcionals. OpenAI ha permès resoldre la generació automàtica de resums textuais (RF-1, RF-2), mantenint una qualitat coherent amb els objectius de rendiment i usabilitat definits (vegeu RNF-1 i RNF-3). Twelve Labs, per la seva banda, ha proporcionat una solució especialitzada per a l'anàlisi de vídeo, amb sortida textual i estructurada, que complementa la informació escrita del sinistre.

8. Arquitectura i distribució de les dades a Cleverea

En aquesta secció es busca contextualitzar l'arquitectura de programari i distribució de les dades dins l'ecosistema de Cleverea.

8.1. Distribució de les dades

La finalitat d'aquest subapartat és descriure les fonts d'informació que alimenten el procés de generació de resums i el format en què aquestes dades s'emmagatzemen i es tracten. En la següent taula podem veure un resum simplificat d'aquestes, en els següents apartats entrarem més en detall en cada tipus de dades i com està estructurada dins la seva tecnologia.

Tipus de dada	Font / servei	Format principal	Ús en el projecte
Dades operatives	PostgreSQL	SQL	Estat, dates, moviments, històric de successos, xats
Trucades i correus electrònics	DynamoDB	Ítem DynamoDB	Informació addicional sobre comunicacions externes amb el client.
Fitxers	Amazon S3	.png / .jpg / .jpeg / .pdf / .mp4	Evidències visuals i documentals

Taula 16. Dades utilitzades en el projecte. Font: Elaboració pròpia.

Per poder tindre una idea més gràfica de com estan distribuïdes aquestes dades i poder relacionar-les millor amb les tecnologies que les emmagatzemen vegeu la següent figura (Figura 6) que resumeix molt tots els components que apliquen al projecte.

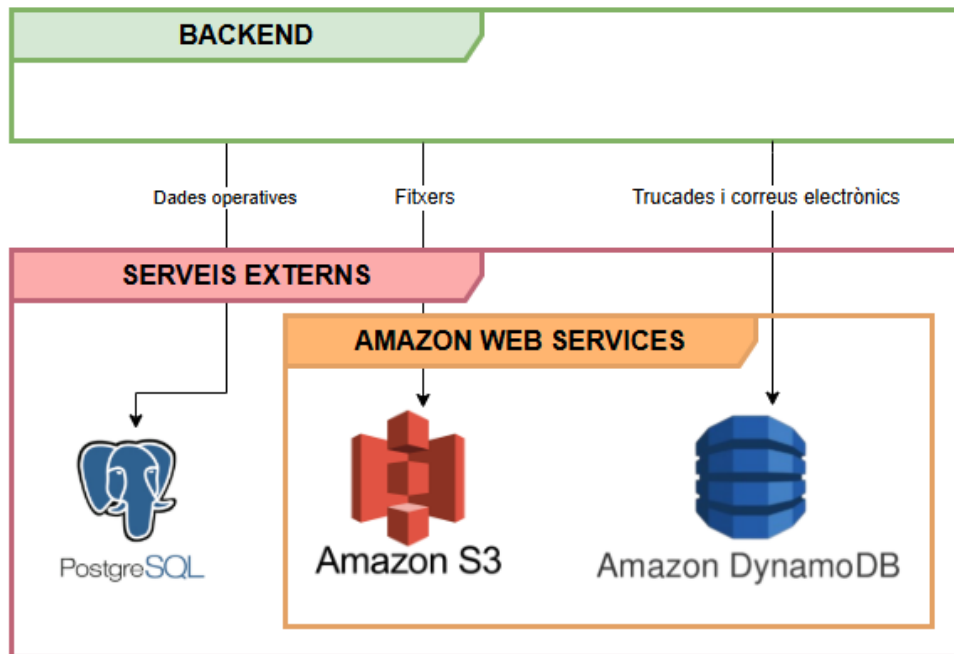


Figura 6. Distribució de les dades a Cleverea. Font: Elaboració pròpia.

8.1.1. Base de dades estructurada (PostgreSQL)

Es tracta de la font principal de dades relacional⁶ dins del món de Cleverea. Conté informació essencial sobre els clients, les seves pòlisses i sinistres, així com tots els detalls relacionats amb aquests: suplements, cobertures contractades, estats del sinistre i més.

A més incorpora l'historial de converses per xat entre el client i l'agent durant la gestió del sinistre, es pot tractar de conversacions molt extenses i, al mateix temps, molt valuoses per conèixer cada cas.

Aquest conjunt de dades representa la visió administrativa i operativa del sinistre, i és especialment útil per obtenir informació fiable i estructurada sobre cada cas.

Finalment, per raons de protecció de la propietat intel·lectual i per evitar revelar informació estratègica a possibles competidors, aquesta memòria no inclou l'esquema detallat ni la distribució interna de les taules de PostgreSQL.

Cal remarcar, però, que totes les dades de sinistres es troben degudament normalitzades i indexades i són accessibles de manera transparent mitjançant els mòduls ja existents dins l'arquitectura de microserveis. Aquesta mesura de confidencialitat no impacta en el cost ni en el temps de consulta: els mecanismes d'orquestració descrits a la secció d'arquitectura

⁶ Una base de dades relacional és una base de dades que permet establir interconnexions entre les dades, i treballar-hi conjuntament. [47]

continuen garantint un accés eficient i segur tant a PostgreSQL com a la resta de fonts de dades.

8.1.2. Base de dades complementària (DynamoDB)

A més de la base de dades relacional principal, el sistema de Cleverea disposa d'una base de dades complementària en format NoSQL⁷ (DynamoDB), que conté informació conversacional no estructurada associada a la gestió dels sinistres. Aquest conjunt de dades representa una visió alternativa i rica sobre el comportament real del client, no accessible per als gestors prèviament al projecte. Tot i que no segueixen una estructura relacional clàssica, aquestes dades poden ser agrupades i relacionades amb cada sinistre a través de diferents identificadors indirectes.

Tot i que PostgreSQL ja conté l'històric de xat intern entre l'agent i el client (8.1.1. Base de dades estructurada (PostgreSQL)), DynamoDB emmagatzema trucades telefòniques i correus electrònics que no formen part del xat intern, sinó d'interaccions externes amb el client; per això es tracten com a font complementària.

Accés i estructura de dades a DynamoDB

Per accedir a les dades emmagatzemades en una taula de DynamoDB, cal entendre la seva estructura basada en claus primàries i índexs secundaris. Cada taula es defineix a partir d'una clau de partició (*Partition Key*), que determina la distribució de les dades i actua com a identificador principal per a recuperar entrades.

Opcionalment, es pot definir una clau d'ordenació (*Sort Key*), que permet recuperar múltiples ítems amb la mateixa clau de partició, però ordenats per un atribut addicional.

A més, DynamoDB permet definir índexs secundaris (GSIs, o *Global Secondary Indexes*), que actuen com vistes addicionals sobre la taula principal. Aquests índexs permeten fer cerques eficients per atributs alternatius a la clau primària. Cada GSI pot tenir la seva pròpia combinació de *Partition* i *Sort Key*, cosa que permet adaptar-se a patrons d'accés concrets.

La complexitat principal d'aquesta base de dades no és només el seu volum o diversitat de contingut, sinó que les dades no estan directament etiquetades amb l'identificador del sinistre (`claim_id`). Per tant, cal seguir una cadena de relacions lògiques entre identificadors (com

⁷ Una base de dades NoSQL proporciona un mecanisme per emmagatzemar i recuperar dades que es modelen mitjançant relacions tabulars diferents a les utilitzades en les bases de dades relacionals. [48]

`document_number`, `contact_id`, `receiver`, `sender`, etc.) per reconstruir el conjunt d'informació relacionat amb un sinistre.

Taules utilitzades en el projecte

A l'Annex B - Distribució de les dades a DynamoDB podem veure com es troba la distribució de les dades rellevants pel projecte dins de DynamoDB. En apartats posteriors s'explica com s'accedeix a aquestes dades i com es gestionen.

8.1.3. Fitxers associats (Amazon S3)

A Cleverea la informació d'un document es troba repartida en dues ubicacions:

- A **PostgreSQL**, hi ha el registre estructurat del document, que inclou informació com el tipus (`document_type`), el moment d'alta (`upload_datetime`), les validacions (`validations`) o la ubicació del fitxer (`file`), que inclou el nom d'aquest. La resta d'informació no es considera rellevant en l'àmbit del projecte.
- A **Amazon S3**, es troba el fitxer físic associat (imatge, PDF o vídeo), identificat de mitjançant la ubicació i nom del document. La seva ubicació es defineix segons a l'entitat a la qual pertany.

Aquesta separació segueix una arquitectura orientada a negoci, on cada entitat del sistema (p. ex. sinistre, pòlissa) pot tenir documents associats, i aquests són emmagatzemats al mateix *bucket* però en ubicacions diferents.

Cleverea utilitza el sistema d'emmagatzematge Amazon S3 [46] per guardar els fitxers multimèdia i documents. Aquesta font de dades és especialment rellevant perquè proporciona dades o evidències que no poden ser representades adequadament en altres models estructurats. El seu contingut complementa les dades que es troben a PostgreSQL, incloent-hi els fitxers físics.

Amazon S3 organitza els fitxers en *bucket*, que actuen com a contenidors lògics per a l'emmagatzematge d'objectes. Cada *bucket* és únic dins del compte d'AWS i pot contenir milions de fitxers. S3 permet la possibilitat de crear directoris de forma que es pot ordenar el seu contingut de forma jeràrquica.

L'estructura utilitzada al bucket és la següent:

1. `s3://nom-bucket/`
 - 1.1. `{tipus_objecte}`

1.1.1. {tipus_document}

1.1.1.1. {nom_document}.pdf

1.1.1.2. {nom_document}.jpg

1.1.1.3. {nom_document}.mp4

On `tipus_objecte` identifica l'entitat del sistema, en el nostre cas `claim`, `tipus_document` és una dada proporcionada en el moment de pujada del document que indica la finalitat del document (vídeo d'explicació, imatges dels danys, informe mèdic, etc.). Dins de la carpeta més interna es troben tots els fitxers pujats pel client o un agent intern.

Els tipus de fitxer que acceptem per part dels clients es poden classificar en:

1. **Imatges.** Habitualment aportades pel client, mostren els danys produïts en accidents o situacions assegurades. Poden trobar-se en format *.jpg*, *.jpeg* i *.png*.
2. **Documents PDF.** Amb informació crítica com el part amistós d'accident, factures de reparació, informes mèdics o pressupostos. Són especialment rellevants per poder avançar la solució de cada cas. Els trobem en format *.pdf*.
3. **Vídeos.** Enregistraments on el client explica el context dels fets o mostra els danys ocasionats. Aporten una visió molt detallada però també més complexa d'analitzar. Els trobem en format *.mp4*.

Finalment, cal tenir en compte que els documents acostumen a ser aportats directament pels clients, sovint sense validació prèvia. Això implica diversos problemes habituals com:

- **Mala classificació** en el tipus de document. Que dificulta el tractament individualitzat de cada document, es tracta d'una dada poc fiable.
- **Qualitat deficient.** Que dificulta la seva interpretació o anàlisi.
- **Contingut ambigu.** Que no aporta informació rellevant.

Totes aquestes casuístiques incrementen la càrrega de treball dels agents i donen importància per disposar d'eines de suport intel·ligents que puguin interpretar el contingut del document i relacionar-lo amb el context de cada sinistre.

8.2. Arquitectura en microserveis

Cleverea en el seu backend adopta una arquitectura basada en microserveis⁸, amb l'objectiu d'oferir una alta modularitat, agilitat en el desplegament i facilitat en l'escalabilitat del sistema. Cada servei està especialitzat en una àrea de negoci i disposa de la seva pròpia base de dades, garantint l'aïllament i facilitant el control d'accessos.

8.2.1. Serveis principals i responsabilitats

En la següent taula (Taula 17) podem veure el detall dels serveis principals i les seves responsabilitats.

Microservei	Estès en el TFG	Funció principal
Claims	Endpoint i flux d'anàlisi	Nucli de la gestió de sinistres. Exposa GET /claims/{claim_id}/summary.
Documents	Cicle d'anàlisi	Gestió i anàlisi de fitxers associats a qualsevol entitat. Encarregat d'accedir als fitxers emmagatzemats a S3.
Verticals (car, moto, home)	-	Regles de negoci específiques de cada tipus de pòlissa.

Taula 17. Serveis principals del projecte. Font: Elaboració pròpia.

Aquest ecosistema de serveis funciona de forma desacoblada, però coordinada, mitjançant comunicacions internes. Això permet que la gestió dels sinistres o documents tingui una part comuna sigui quina sigui la seva font real.

Els avantatges d'aquesta arquitectura inclouen:

- **Especialització funcional:** facilita que cada servei es focalitzi en àrees concretes, millorant la qualitat i modularitat del codi.
- **Escalabilitat independent:** cada servei pot escalar segons la seva càrrega operativa sense afectar a la resta.
- **Facilitat de desplegament:** actualitzacions ràpides i freqüents, minimitzant el risc associat al desplegament.

Limitacions i complexitat associada

Tot i els avantatges, aquest model introdueix dificultats o reptes notables:

⁸ En enginyeria de programari, una arquitectura de microservei és un patró arquitectònic que organitza una aplicació en una col·lecció de serveis de gran fi i poc acoblats que es comuniquen mitjançant protocols lleugers. [49]

- **Major corba d'aprenentatge:** per comprendre el funcionament global del sistema, és necessari entendre els límits i dependències entre molts serveis. Aquesta visió no és fàcil de transmetre.
- **Coordinació entre serveis:** encara que cada servei sigui independent, hi ha casos d'ús transversals, com la generació d'un resum de sinistre, que requereixen orquestrar dades provinents de diferents serveis (*Claims*, *Documents*, etc.) fet que incrementa la complexitat i el temps de desenvolupament.
- **Gestió d'errors distribuïts:** en un sistema unificat un error pot ser més fàcil de rastrejar. En canvi, en aquesta arquitectura cal monitoritzar múltiples punts i les seves integracions per identificar l'origen d'un problema.

Impacte directe en el projecte

L'objectiu del projecte, la generació automàtica de resums, depèn d'aquesta arquitectura distribuïda. Per tal de construir un resum útil, caldrà accedir a dades de diversos microserveis com *Claims* i *Documents*, integrant informació estructurada, no estructurada i contextual. Ha calgut adaptar la funcionalitat a les següents afirmacions:

Cada microservei del sistema es desplega amb una estructura de dades pròpia i aïllada, gestionada mitjançant la seva pròpia base de dades. Aquesta estratègia segueix el principi de "base de dades per servei"⁹, habitual en arquitectures de microserveis, i permet:

- **Garantir l'aïllament entre serveis**, evitant col·lisions o dependències no desitjades.
- **Facilitar la gestió de permisos** i accés per servei.
- **Simplificar les tasques** de migració, monitoratge i escalabilitat individual.
- **Assegurar una millor traçabilitat** de dades i errors a nivell de servei.

Aquesta separació és especialment rellevant en el context de documents, ja que el seu propi microservei és l'únic que té accés directe a aquesta font de dades externa.

⁹ El principi base de dades per servei indica que cada microservei té la seva pròpia base de dades que no és compartida directament amb altres serveis.

En el següent diagrama (Figura 7) podem veure la distribució de serveis i el domini que tenen sobre cada sistema extern.

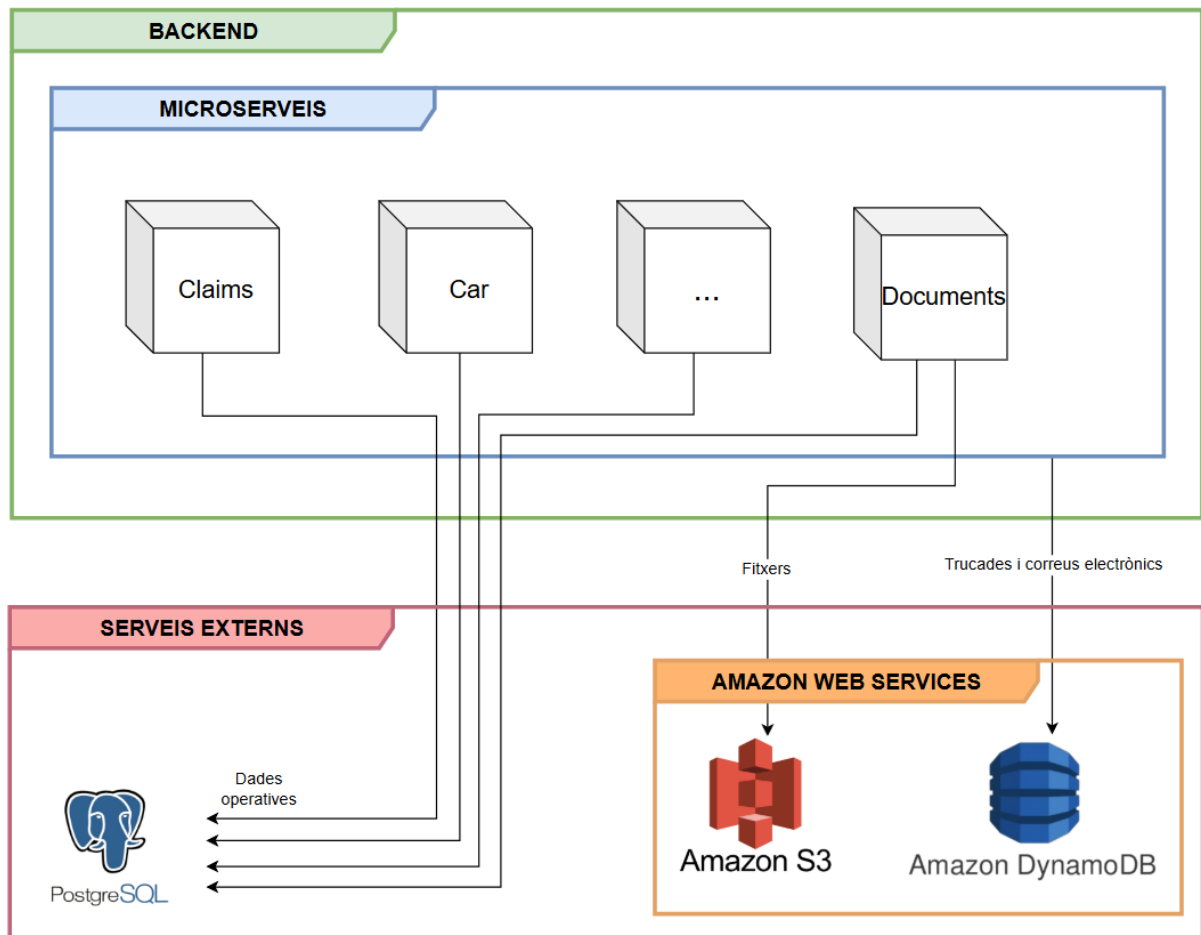


Figura 7. Distribució dels microserveis a Cleverea. Font: Elaboració pròpia.

8.3. Arquitectura hexagonal

L'arquitectura hexagonal, proposa una separació clara entre el nucli del sistema i els mecanismes d'entrada i sortida. Aquest model resulta especialment útil en entorns on es vol mantindre la lògica de negoci aïllada dels detalls tècnics o tecnologies.

A Cleverea, cada microservei segueix l'arquitectura hexagonal¹⁰, una decisió clau per garantir la separació de responsabilitats, la facilitat de fer proves i la independència respecte a tecnologies concretes. Aquesta arquitectura permet dissenyar sistemes centrats en el domini, on la lògica de negoci es manté neta i protegida de les dependències externes.

¹⁰ L'Arquitectura Hexagonal té com a principal motivació separar la nostra aplicació en diferents capes o regions amb la seva pròpia responsabilitat. [50]

Aquesta estratègia encaixa amb l'enfocament de microserveis i s'ha aplicat rigorosament en el projecte per garantir una integració eficaç en l'entorn de Cleverea.

Abans d'entrar en més detall m'agradaria definir dos conceptes relacionats amb l'arquitectura hexagonal a Cleverea:

- **BO.** Business Object, representació dels models de domini (a A-20. ClaimBO podem veure la informació/exemple de ClaimBO).
- **Interface.** Contracte funcional sense implementació concreta (a A-21. GenerateClaimSummaryInterface es pot veure l'exemple del contracte funcional d'una implementació senzilla).

Estructura de capes

L'arquitectura s'organitza en tres capes ben diferenciades:

- **Domini.** Aquesta és la capa més interna i representa el nucli del sistema. Defineix les entitats de negoci (ClaimBO, ClaimCallBO, etc.) amb les seves regles relacionades (closable, rejectable), així com les interfícies (com GenerateClaimSummaryInterface) que descriuen el comportament esperat però sense definir una implementació. No mostren cap dependència tecnològica.
- **Aplicació.** Actua com a capa coordinadora dels casos d'ús. Per exemple, el servei GetClaimSummaryService és el responsable d'orquestrar tot el procés de generació de resums. Aquesta capa connecta els serveis de domini amb funcionalitats d'infraestructura mitjançant interfícies ben definides, orquestra els casos d'ús sense definir la seva lògica.
- **Infraestructura.** Aquí es materialitzen les interfícies definides al domini. Per exemple, LLMGenerateClaimSummaryService implementa la generació de resums amb OpenAI, seguint les regles definides a la interfície GenerateClaimSummaryInterface. És la que s'encarrega d'implementar la persistència i comunicació amb serveis externs.

En la següent imatge (Figura 8) podem veure una vista simplificada de com s'estructuren les capes per aquests exemples.

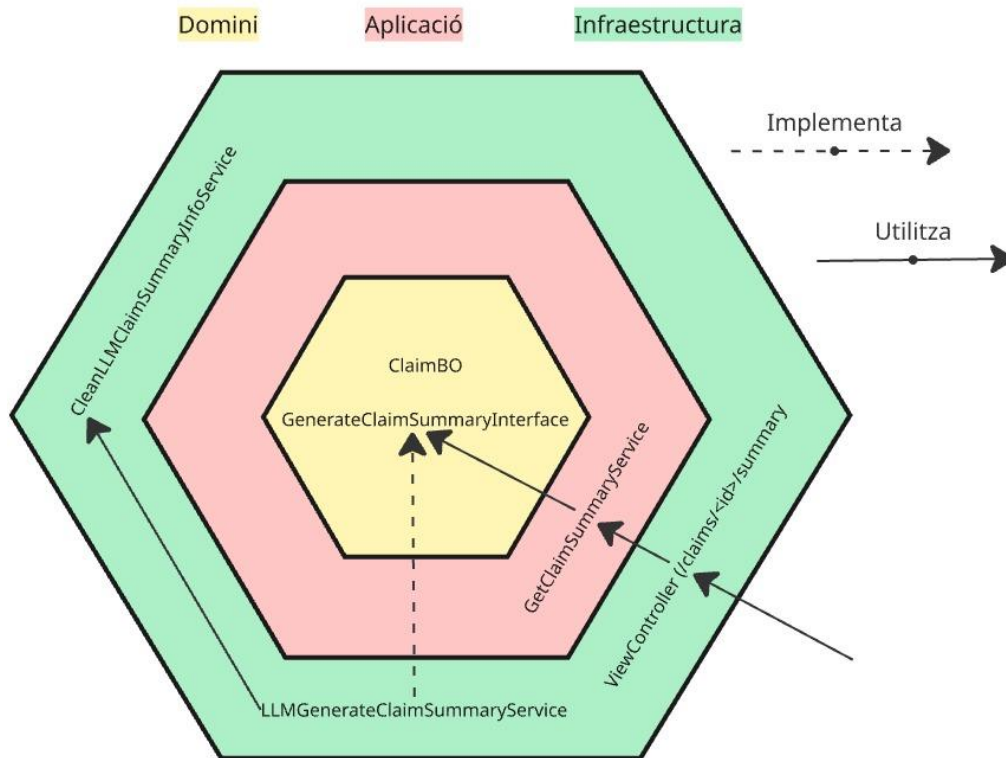


Figura 8. Vista simplificada de l'arquitectura hexagonal. Font: Elaboració pròpia.

Beneficis i limitacions

Complir amb totes aquestes regles requereix una gran disciplina i coneixements. La qual cosa afegeix complexitat a projectes petits i desenvolupaments més curts o per gent amb menys experiència, tot i això, en etapes més avançades o projecte més grans comporta grans beneficis com: domini net i estable, proves unitàries senzilles o fàcil canvi de proveïdors externs.

8.4. Fluxos funcionals detectats

Com s'ha definit en aquest capítol la funcionalitat desenvolupada s'ha integrat dins d'un ecosistema de microserveis, fet que ha condicionat la seva arquitectura i el seu comportament funcional.

El repte del projecte ha sigut com generar i gestionar el resum del sinistre dins d'aquest entorn definit, on la informació està fragmentada entre diversos serveis i fonts de dades. Això ha dut a identificar dos fluxos independents però complementaris:

- Flux d'anàlisi de documents: és produeix de forma automàtica quan un client o agent puja un nou document. Un cop el sistema ja emmagatzemat tota la informació del fitxer, s'activa el procés d'anàlisi asíncron. El resultat s'enregistra i queda disponible per a futures consultes.
- Flux de petició de resum del sinistre: s'activa quan l'agent sol·licita el resum del cas. El sistema orquestra la consulta de diverses fonts de dades, construeix un context i utilitza una intel·ligència artificial per obtenir un resum comprensible i rellevant.

Aquesta separació entre fluxos és essencial per garantir la resposta ràpida en temps real i alhora mantenir una estructura escalable i desacoblada, que facilita el manteniment i l'evolució futura.

El detall tècnic de cadascun d'aquests fluxos es pot consultar a la 9.1. Visió general del sistema.

9. Disseny del sistema

Aquest capítol recull totes les **decisions de disseny** que converteixen l'arquitectura descrita al capítol 8 en una implementació concreta, escalable i segura. L'objectiu és deixar rastre explícit de:

- Com es tradueixen els **requisits funcionals** (RF-x) i no funcionals (RNF-x) en patrons, serveis i configuracions.
- Quins **prompts**, models i paràmetres s'utilitzen per obtenir resultats fiables de la IA.
- L'estratègia d'**accés a dades** i de **neteja de contingut sensible** abans d'enviar-lo a proveïdors externs.
- Els **mecanismes de control de costos** i de seguretat (secrets, *throttling*).

9.1. Visió general del sistema

Com s'ha definit en el capítol 8.4. Fluxos funcionals detectats, s'han identificat dos fluxos principals clarament diferenciats. Aquests dos fluxos es van definir directament a partir dels requisits funcionals principals, d'una banda, satisfer la necessitat de RF-1 i RF-2 i de l'altra, complir RF-3 mitjançant un procés paral·lel d'anàlisi de documents. D'aquesta manera, el disseny traça l'especificació: cada funcionalitat té un reflex en l'arquitectura proposada.

A continuació mostrem els dos fluxos:

9.1.1. Resum de sinistre

Aquest flux és el principal del sistema. L'agent de sinistres, a través del BackOffice, pot sol·licitar un resum d'un sinistre concret. Aquesta acció activa el microservei Claims, que s'encarrega d'agregar la informació necessària i generar un resum complet mitjançant IA.

A la Figura 9 es mostra aquest flux:

- Claims consulta informació estructurada des de PostgreSQL, dades comunicatives (trucades i correus electrònics) des de DynamoDB i recupera resums de documents preexistents que ja hagin estat analitzats prèviament.
- La generació del resum es realitza mitjançant OpenAI.
- És important destacar que Claims **no analitza documents directament**, sinó que reutilitza els resultats ja generats pel microservei Documents.

```
graph TD
    Agent[Agent de sinistres] -- "Consulta sinistre" --> BO[BackOffice Cleverea]
    BO -- "Crida a /summary" --> MC[Microservei Claims]
    MC -- "Resums de documents" --> MD[Microservei Documents]
    MC -- "Dades estructurades" --> PG[(PostgreSQL)]
    MC -- "Trucades i emails" --> DB[(DynamoDB)]
    MC -- "Resum de comentaris" --> SC[Servei de Comentaris]
    MC -- "Tasca async" --> RC[Redis + Celery]
    MC -- "Generació del resum" --> OA[OpenAI API]
    MC -- "Logs" --> K[Logs -> Kibana]
    MC -- "Alarmes" --> S[Alarmes -> Slack]
    MD --> MD_Note[Només si existeix el resum]
    PG --> PG_Note[Només consulta resum de documents ja analitzats]
```

Diagrama de fluxe de treball per a la consulta de sinistres i generació de resums:

- Agent de sinistres** envia una **Consulta sinistre** al **BackOffice Cleverea**.
- BackOffice Cleverea** envia una **Crida a /summary** al **Microservei Claims**.
- Microservei Claims** interactua amb diversos components:
 - Microservei Documents**: Rebre **Resums de documents**.
 - PostgreSQL**: Rebre **Dades estructurades**.
 - DynamoDB**: Rebre **Trucades i emails**.
 - Servei de Comentaris**: Rebre **Resum de comentaris**.
 - Redis + Celery**: Rebre una **Tasca async**.
 - OpenAI API**: Rebre la **Generació del resum**.
- Microservei Claims** envia **Logs** a **Kibana** i **Alarmes** a **Slack**.
- Microservei Documents** té una nota: **Només si existeix el resum**.
- PostgreSQL** té una nota: **Només consulta resum de documents ja analitzats**.

9.1.2. Anàlisi automàtica de documents

- Documents publica un esdeveniment a SNS.
- Cada microservei que consumeix documents (Claims en el nostre cas) pot subscriure-s'hi i decidir si cal analitzar aquell document.
- En cas afirmatiu, Claims fa una crida a Documents per iniciar el procés.
- Documents envia la tasca d'anàlisi a la cua i, segons el tipus de document, delega l'anàlisi a OpenAI (imatges i PDFs) o Twelve Labs (vídeos).
- El resultat de l'anàlisi es desa i queda disponible per consultes futures.

69

El flux queda representat a la Figura 10, i exemplifica com la generació del resum dels documents es fa de **forma anticipada i desacoblada del procés de resum general del sinistre**.

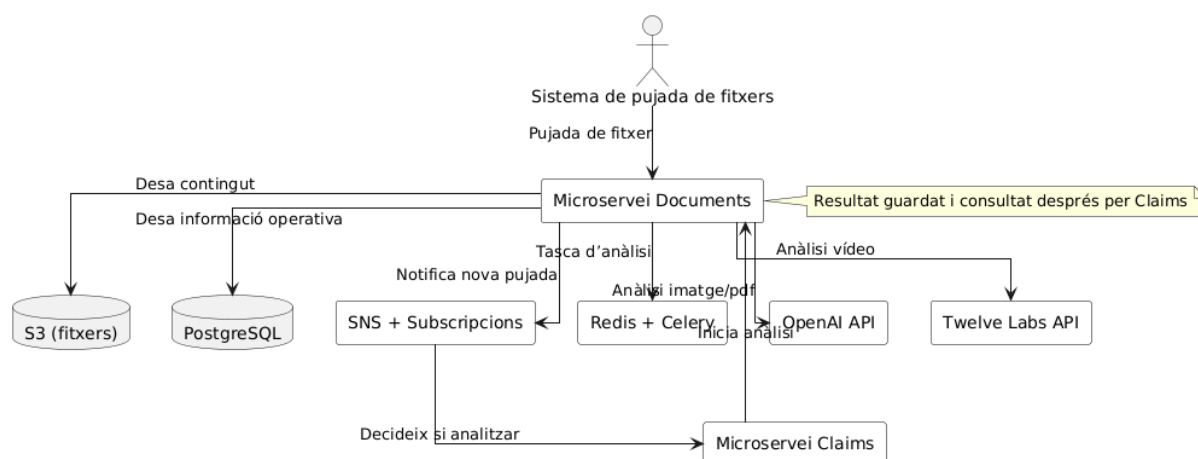


Figura 10. Flux d'anàlisi automàtica de documents. Font: <https://www.planttext.com>

9.2. Vista de contenidors

El sistema s'ha estructurat seguint l'arquitectura de microserveis, buscant la separació de responsabilitats i la facilitat de manteniment (capítol 8. Arquitectura i distribució de les dades a Cleverea).

A la Taula 18 es detallen els principals contenidors que formen part de la solució i una breu descripció de com ho fan.

Contenidor / Microservei	Descripció
Claims	Servei que orquestra la generació de resums de sinistres. Accedeix a dades estructurades, comunicacions i resums de documents.
Documents	Servei responsable d'anàlitzar els documents pujats.
Servei de comentaris	Servei que forma part de Claims que genera i emmagatzema resums dels comentaris en memòria cau.
PostgreSQL	Base de dades relacional per a la informació estructurada del sinistre i dels documents.
DynamoDB	Base de dades NoSQL on es guarden les transcripcions de trucades i correus.
AWS S3	Sistema d'emmagatzematge d'arxius pujats pels usuaris.
OpenAI API	Proveïdor de serveis d'anàlisi de textos, documents i imatges.
Twelve Labs API	Proveïdor de serveis d'anàlisi de vídeos.

Taula 18. Principals contenidors de la solució. Font: Elaboració pròpia.

A més dels components específics que formen part dels fluxos principals, el sistema es recolza en diversos components transversals que donen suport al processament, monitoratge i notificació d'errors. Aquests són imprescindibles per garantir la robustesa i eficiència del sistema. Els podem observar a la Taula 19.

Component	Descripció
Kibana	Plataforma de visualització de logs. S'utilitza per monitoritzar l'estat i els errors dels serveis desplegats.
Slack	Canal intern per a notificacions d'errors crítics.
Redis + Celery	Sistema de cues per a processament asíncron de tasques (integrat ja prèviament a l'entorn de Cleverea).
SNS + Subscripcions	Mecanisme d'esdeveniments que permet als serveis reaccionar a la pujada de document.

Taula 19. Components transversals. Font: Elaboració pròpia.

9.3. Components clau

En aquest apartat es descriu **què fa cadascun dels mòduls principals** separant els dos fluxos (“Resum del sinistre” i “Anàlisi automàtica de documents”, a quina capa pertanyen i de quines dependències disposen.

9.3.1. Components del flux “Resum de sinistre”

La següent taula (Taula 20) mostra els mòduls principals del flux “Resum de sinistre”, indicant la capa hexagonal en la que es troben, aportant una descripció breu i les dependències injectades per aquells desenvolupats en el projecte.

Component	Capa Hexagonal	Descripció breu	Dependències injectades	Referència a Annex
GetClaimSummaryService	Aplicació	Orquestrar el procés complet de generació del resum: recupera dades i crida al mòdul de generació de resums.	LLMGenerateClaimSummaryService, ClaimBODjangoPersistenceService, GetClaimCallsService, GetClaimEmailsService, SearchAllRelatedDocumentsOfAClaimService	A-4. GetClaimSummaryService
LLMGenerateClaimSummaryService	Infraestructura	Netejar les dades delicades, agrupar la informació, preparar el prompt i crida a GPT-4.1 i vàlida la resposta	CleanClaimSummaryInfoService, Client OpenAI i Twelve Labs	A-9. LLMGenerateClaimSummaryService
CleanLLMClaimSummaryInfoService	Infraestructura	Elimina dades personals del client (per garantir privacitat de les dades) i altra informació no rellevant (per reduir el contingut que enviem a la IA)	-	Amagat per privacitat de l'empresa.
ClaimBODjangoPersistenceService	Infraestructura	Gestiona la persistència de cada sinistre a PostgreSQL.	No desenvolupat en el TFG	-
GetClaimCallsService	Domini	Amb la informació del sinistre, busca el document del portador de la pòlissa i crida a DynamoDBClaimCallsService per recuperar les trucades analitzades.	GetHolderService, DynamoDBClaimCallsService	A-5. GetClaimCallsService
GetHolderService	Domini	Recupera el portador de la pòlissa amb la informació d'aquesta.	No desenvolupat en el TFG	-
DynamoDBClaimCallsService	Infraestructura	S'encarrega de relacionar les trucades existents amb el sinistre, tenint en compte les dates, la cua de trucades i el document d'identitat del portador de la pòlissa.	DynamoDBTableService	A-7. DynamoDBClaimCallsService

DynamoDBTableService	Infraestructura	Gestiona totes les crides cap a DynamoDB.	No desenvolupat en el TFG	-
GetClaimEmailsService	Domini	Amb la informació del sinistre, busca l'adreça electrònica del portador i crida a DynamoDBClaimEmailsService per aconseguir els emails relacionats.	GetHolderservice, DynamoDBClaimEmailsService	A-6. GetClaimEmailsService
DynamoDBClaimEmailsService	Infraestructura	S'encarrega de recuperar tots els correus electrònics relacionats amb el portador de la pòlissa tenint en compte la seva adreça electrònica i les dates d'aquest.	DynamoDBTableService	A-8. DynamoDBClaimEmailsService
SearchAllRelatedDocumentsOfAClaimService	Domini	Recupera tots els documents relacionats amb un sinistre.	No desenvolupat en el TFG	-

Taula 20. Mòduls principals del flux "Resum de sinistre". Font: Elaboració pròpia.

9.3.2. Components del flux "Anàlisi automàtica de documents"

La següent taula (Taula 21) mostra els mòduls principals del flux "Anàlisi automàtica de documents", indicant la capa hexagonal en la que es troben, aportant una descripció breu i les dependències injectades per aquells desenvolupats en el projecte.

Component	Capa Hexagonal	Descripció breu	Dependències injectades	Referència a Annex
AnalyseDocumentService	Aplicació	Orquestra el procés d'anàlisi d'un document. Recupera el document, l'analitza i guarda l'anàlisi a PostgreSQL.	LLMSummariseDocumentService, DocumentAnalysisBODjangoPersistenceService, DocumentBODjangoPersistenceService, analysis_type	A-14. AnalyseDocumentService
LLMSummariseDocumentService	Infraestructura	Comprova el tipus de document, decideix com s'ha de gestionar segons el tipus, genera l'anàlisi complet (PDFs i imatges) o parcial (vídeos).	QueryS3Service	A-15. LLMSummariseDocumentService
QueryS3Service	Infraestructura	Gestiona l'accés a AWS S3.	No desenvolupat en el TFG	-
DocumentAnalysisBODjangoPersistenceService	Infraestructura	Gestiona la persistència de les anàlisi de documents a PostgreSQL.	-	No aporta informació acadèmica rellevant, Segueix el ja implementat prèviament a Cleverea.
DocumentBODjangoPersistenceService	Infraestructura	Gestiona la persistència dels documents a PostgreSQL.	No desenvolupat en el TFG	-
TwelveWebhookLabsDocumentAnalysisView	Infraestructura	Gestiona el webhook de Twelve Labs per recuperar l'anàlisi d'un document. Comprovant la signatura i cridant al controlador que pertoca.	-	A-16. TwelveLabsWebhookDocumentAnalysisView

CompleteDocumentAnalysisViewControllerV1	Aplicació	Orquestra els passos a seguir per poder completar una anàlisi. Recupera l'anàlisi incompleta, recupera el valor de l'anàlisi i el persisteix en l'anterior.	DocumentAnalysisBODjangoPersistenceService, RetrieveTwelveLabsResponseService	A-17. CompleteDocumentAnalysisViewControllerV1
RetrieveTwelveLabsResponseService	Infraestructura	Amb la referència externa al vídeo recupera i retorna el seu resum.	-	A-18. RetrieveTwelveLabsResponseService

Taula 21. Mòduls principals del flux "Anàlisi automàtica de documents". Font: Elaboració pròpia.

9.3.3. Relació amb els requisits

Els components implementats cobreixen diferents requisits funcionals (RF) i no funcionals (RNF) en la següent taula (Taula 22) es mostra la traçabilitat entre funcionalitats i mòduls del sistema, indicant quin component implementa o dona suport a cada requisit:

Requisit	Components implicats
RF-1 Sol·licitud	GetClaimSummaryService, LLMGenerateClaimSummaryService, BackOffice
RF-2 Visualització	BackOffice, MS Claims, sistema de resposta HTTP
RF-3 Contingut mínim	ClaimBODjangoPersistenceService, GetClaimCallsService, GetClaimEmailsService, SearchAllRelatedDocumentsOfAClaimService, CleanLLMClaimSummaryInfoService
RF-4 Recuperació de resum existent	GetClaimSummaryService (comprova caché), Redis
RNF-1 Rendiment	LLMGenerateClaimSummaryService, Redis + Celery
RNF-2 Disponibilitat	Monitoratge amb Kibana, alarmes per Slack
RNF-3 Usabilitat	LLMGenerateClaimSummaryService (estructura del prompt), filtratge amb CleanLLMClaimSummaryInfoService
RNF-4 Seguretat	Configuració de secrets, validació inputs
RNF-5 Escalabilitat	Celery, cues, serveis desacoblats
RNF-6 Eficiència operativa	Optimització dels prompts, execució asíncrona
RNF-7 Satisfacció interna	Feedback dels agents, retorn de resums clars i útils al Backoffice
RNF-8 Control de costos	AnalyseDocumentService (control de límits), monitoratge de costos des de les plataformes OpenAI i Twelve Labs
RNF-9 Monitoratge d'errors i alarmes	Kibana (registre de logs) + Slack (notificacions d'errors crítics)
RD-1 Fonts de dades	PostgreSQL, DynamoDB, S3, accedides des dels serveis Claims i Documents
RD-2	CleanLLMClaimSummaryInfoService

Minimització de dades personals	
RD-3 Idioma	Generació controlada en castellà dins LLMGenerateClaimSummaryService

Taula 22. Relació de components amb requisits. Font: Elaboració pròpia.

9.4. Disseny de la interacció amb sistemes d'IA

9.4.1. Prompt i justificació

Si el que volem són resultats precisos en el contingut dels nostres resums necessitem definir que és el que volem i fer-li-ho saber a la IA. El *prompt*, que s'encarrega d'aquesta tasca, és el text d'entrada que acompanya les dades i permet definir la sortida esperada. Per aconseguir un *prompt* adequat cal iterar per poder arribar a un resultat precís però, també, reduir la càrrega del servei i, amb això, els costos associats.

Durant el desenvolupament del projecte s'ha anat iterant, segons necessitats, sobre un *prompt* final que defineix exactament el que esperem com a resposta per part del servei extern. En apartats posteriors s'exemplifica el prompt final i es justifica la seva estructura.

"""

Actúa como un asistente experto en resumir siniestros en formato JSON, generando un resumen narrado en español que refleje la situación general y destaque los puntos más relevantes (desacuerdos, solicitudes de información y acciones pendientes) teniendo en cuenta la fecha actual """

+ current_date
+ """.

Tu respuesta debe ser **exclusivamente** un objeto JSON válido, sin texto adicional ni nodos fuera de ese objeto, y debe poder parsearse con `json.loads()` sin errores. Si falta información en los datos originales, menciona que no fue proporcionada o no existe. **No alteres la información recibida**.

1. *Resumen narrado*

- Redáctalo en **español** con estilo narrativo y en **texto plano**.
- **Máximo 600 palabras**.
- Estructúralo en uno o varios párrafos con **títulos**, usando la siguiente estructura:

```
json
{
  "summary": [
    {
      "title": "Título del primer párrafo",
      "content": "Texto del primer párrafo"
    },
    {
      "title": "Título del segundo párrafo",
      "content": "Texto del segundo párrafo"
    },
    ...
  ]
}
```

```
]
}
```

- Destaca los desacuerdos, solicitudes de información y acciones pendientes.
- Subraya las fechas relevantes.
- *Analiza las fechas* en los datos (campos de fecha y referencias temporales) para presentar los eventos *en orden cronológico* dentro de tu narración.
- Emplea oraciones breves y concisas.

2. *Formato estricto*

- Responde *exclusivamente* con el *objeto JSON* anterior, sin ningún texto adicional.
- Verifica que sea un objeto JSON válido y *no* excedas las 600 palabras en total.

3. *Objetivo*

- Reflejar la situación global, enfatizando los puntos clave (las acciones pendientes son muy relevantes).
- Ordenar los hechos de manera *cronológica*, con base en las fechas encontradas en los datos.
- Mantenerse fiel a la información proporcionada, sin alterarla ni inventar datos.
- Evitar ocupar espacio con información redundante o poco relevante.
- Haz un apartado extra para informar de las llamadas y emails relevantes, llamado "Llamadas y Mails".
- Haz un apartado extra para informar de los documentos subidos, llamado "Documentos Subidos".

4. *Información relevante*

- Los amount dentro de los payments se expresan en céntimos pero tu debes devolverlos en euros.
 - Las llamadas y mails son del mismo usuario, pero pueden referirse a otros siniestros o gestiones.
- *Solo debes tener en cuenta los que se relacionen con el siniestro actual, según el contenido de su resumen.*
- El resumen de cada llamada puede contener errores ya que proviene de una transcripción automática. Prioriza los datos externos si hay contradicciones.
 - Si no hay documentos subidos, menciona que no existen o no fueron proporcionados.

""

Justificació de les seccions del prompt

1. Rol explícit i contextualització

Assignar al model un rol concret com a "asistente experto" ajuda a orientar el to, l'estil i l'abast de la seva resposta. És una pràctica habitual per aconseguir resultats més professionals i acotats.

2. Control estricte del format

Es força al model a respondre exclusivament en format JSON vàlid per garantir la integració dins del sistema sense errors.

3. Estructura del contingut narratiu

S'utilitza un format jeràrquic amb títols per facilitar la lectura per part dels agents i per garantir que el resum és útil i estructurat. També es delimita la resposta a 600 paraules per controlar respostes massa extenses o redundants.

4. Ordre cronològic

Demandar explícitament que el resum estigui ordenat segons les dates aporta valor afegit per als agents, ja que permet entendre el fil temporal dels esdeveniments, especialment útil en sinistres llargs o complexos.

5. Control de la imaginació del model

Es recalca que no s'han d'inventar dades i que la resposta ha de reflectir exclusivament dades que han estat proporcionades. Aquest punt és clau per garantir la fiabilitat del sistema, més detallat en la secció 9.4.3. Gestió d'inputs buits.

6. Apartats addicionals

Es demana la inclusió de seccions específiques per ajudar a segmentar informació clau i de difícil accés de manera ordenada i útil per a l'agent de sinistres, reduint el temps de revisió manual.

7. Altres característiques

En tot moment es parla en la llengua que esperem rebre la resposta per evitar confusions i, per tant, garantir el resultat que volem.

S'inclou la data actual de forma que el model pugui interpretar millor l'ordre cronològic dels fets i l'estat actual del sinistre.

Es demana destacar informació rellevant per als agents de sinistres com poden ser els desacords i les accions pendents.

S'afegeix alguna informació addicional per enriquir el context que acompanya el prompt.

9.4.2. Neteja de dades

Abans d'enviar la informació operativa a l'API s'aplica un control de **neteja de dades personals**. Amb aquest flux es busca eliminar dades amb dos propòsits:

- **Reduir la informació d'entrada:** Eliminant del context informació que no es necessària per generar els resums dels sinistres permet reduir la informació que enviem a la IA i, per tant, els costos associats.
- **Protegir als nostres clients:** Cal suprimir dades personals abans de fer crides a la IA per garantir els drets dels clients i així, complir les legislacions vigents.

9.4.3. Gestió d'inputs buits

S'ha notat que quan el client no aporta suficient informació o informació buida, el sistema genera un resum fictici a partir d'informació hipotètica. Per evitar aquest comportament s'han dut a terme dues mesures:

Control en el prompt: Definir en el prompt la creació del resum únicament a partir de la informació proporcionada limita la imaginació de dades fictícies al model.

Configuració de la temperatura: La temperatura (T) de cada model defineix la creativitat que obtindrem en la resposta, amb diverses proves s'han trobat diversos escenaris:

- $0 < T < 0,4$: El model tendeix a no interpretar les dades de forma crítica. Amb aquest comportament no podem obtenir resultats com els que busquem a "Acciones pendientes".
- $0,4 \leq T \leq 0,6$: El model és capaç d'interpretar les dades de forma crítica al mateix temps que limita la invenció o sobre-creativitat en aquestes.
- $0,6 < T$: En algunes ocasions el model es massa creatiu i tendeix a donar informació incorrecta.

És per això que hem definit un control en el prompt i una temperatura de 0,6 per tal de poder obtenir els millors resultats possibles.

9.4.4. Elecció del model

Després d'analitzar les oportunitats i característiques que ofereixen els models d'OpenAI. S'han extret les següents conclusions.

Conclusions principals

Evolució constant: Cada nova generació millora almenys un dels eixos clau (intel·ligència, cost o velocitat) sense penalitzar els altres. Això converteix en essencial revisar periòdicament l'oferta de models i adaptar-la a cada projecte.

Exemple pràctic: L'aparició de GPT-4.1 va representar un salt qualitatiu respecte els primers GPT-4: finestra de context molt àmplia, millor rendiment i preus molt més ajustats. Aquests avantatges van justificar incorporar-lo immediatament al meu projecte.

Sabent això i havent fet unes petites proves amb els diferents model ens vam adonar de seguida que:

- **Qualitat de raonament.** GPT-4.1 va resoldre 19/20 resums amb $\geq 4/5$ punts i sense errors de càlcul ni atribució; les versions mini més barates van perdre exactitud (fins a 3 errors cronològics).
- **Finestra de context** $> 1 \text{ M } tokens$. Permet enviar l'expedient complet en una sola trucada, eliminant *chunking*¹¹ i reduint codi i manteniment.
- **Costos.** Comparat amb el GPT-4.1, les versions Turbo o GPT-4 clàssic igualaven la qualitat però doblaven o triplicaven la factura.

En conseqüència, GPT-4.1 és l'únic model que satisfà simultàniament els requisits de qualitat, cost i simplicitat tècnica en l'estat actual del projecte.

9.4.5. Control finestra de context

En les fases inicials del projecte s'havia plantejat implementar un mecanisme per controlar la quantitat d'informació enviada al model, amb l'objectiu d'evitar sobrecàrregues que superin el límit de la finestra de context de cada model. Tanmateix, l'evolució recent dels models de llenguatge, que actualment disposen de finestres de context molt àmplies, ha fet innecessària aquesta mesura. El sinistre amb més contingut (conversacions entre client i agent amb +150 missatges) ocupava 132 k *tokens*, un 12,6 % de la finestra de GPT-4.1.

Les proves realitzades amb els sinistres que presentaven més volum d'informació han mostrat que no s'assolia ni el 20 % del límit de context permès. Pel que fa als documents, les restriccions de mida imposades per la plataforma de Cleverea són més restrictives que les dels models utilitzats.

9.4.6. Anàlisi de documents

Una de les funcionalitats més potents que ofereix la plataforma d'OpenAI és la capacitat d'analitzar documents no estructurats, especialment imatges i fitxers PDF. Aquesta funcionalitat ha estat fonamental per millorar la qualitat dels resums de sinistres, ja que gran part de la informació rellevant es troba dins de documents amb aquests formats. En aquest

¹¹ La tècnica *chunking* és un mètode de memorització que comença amb la destil·lació de grans fragments d'informació en fragments o porcions més petites. [63]

projecte l'anàlisi de documents s'ha realitzat diferenciant clarament entre imatges i arxius PDF, aprofitant les capacitats de l'SDK per a cada cas.

Tractament d'imatges

Per a imatges, s'utilitza el model de ChatOpenAI de Lang Chain, que admet l'anàlisi d'imatges a través d'una URL pública. Aquesta estratègia permet evitar la càrrega directa dels fitxers al nostre servei, reduint així la càrrega computacional i optimitzant el temps de resposta. El model s'invoca amb un prompt específic i la imatge es passa mitjançant el camp `image_url`, seguint el format requerit per l'SDK.

Tractament de documents PDF

Els fitxers PDF, en canvi, es processen mitjançant l'SDK oficial d'OpenAI, aprofitant la funcionalitat `responses.create`, que permet l'anàlisi directa de documents en format binari. En aquest cas el contingut del fitxer es codifica en base64¹² i s'insereix dins una estructura tal com requereix l'API.

Consideracions i paràmetres

En ambdós casos, el model utilitzat ha estat GPT-4.1, seleccionat per la seva capacitat de comprensió semàntica.

9.4.7. Twelve labs

9.4.7.1. Prompting

De la mateixa manera que en OpenAI, l'ús de Twelve Labs també requereix d'un *prompt* que especifiqui quina informació s'espera obtindre del contingut del vídeo i amb quin format ha de ser.

A continuació, es presenta el prompt dissenyat per a dur a terme aquesta tasca.

"""

Analiza el video. Este puede contener la explicación de un siniestro o imágenes de éste (daños, contexto, testimonios o ubicaciones afectadas).

Tu tarea es generar una descripción única, clara y CONCISA del contenido y propósito del video en español. Si el video trata sobre daños materiales, enfócate en proporcionar un resumen detallado de

¹² Base64 és un sistema de numeració posicional que fa servir 64 com a base. [57]

lo observado (por ejemplo: tipo de daños, ubicación de los mismos, condiciones del entorno). Si hay información relevante visible o audible intégrala de forma natural.

No inventes nada. Solo utiliza información que esté presente y visible o audible en el video. La respuesta debe devolverse en un único campo JSON con esta estructura exacta:

```
{  
  
  "summary": "Texto explicativo completo, incluyendo detalles sobre el siniestro y otros datos relevantes encontrados de forma integrada y natural."  
  
}
```

No añadas ningún texto adicional fuera del JSON.

""

Com es pot veure l'estructura i contingut del *prompt* segueix el mateix format que la resta dels que ja s'han detallat anteriorment.

9.4.7.1. Integració al projecte

Havent treballat, en fases anteriors del projecte amb altres proveïdors va facilitar la tasca d'integració amb Twelve Labs. Amb el seu propi SDK [41] la integració per la comunicació inicial va resultar poc costosa, però teníem una altra barrera: la latència en l'anàlisi d'un vídeo.

Per poder analitzar un vídeo utilitzant aquesta plataforma s'han de seguir diversos passos:

1. **Indexar el vídeo.** Aquest pas s'encarrega de processar el contingut del vídeo perquè es pugui cercar i analitzar intel·ligentment. És fonamental abans de fer consultes de cerca o d'anàlisi sobre el contingut del vídeo. Es tracta d'un procés costós, normalment tant com la durada del vídeo (la durada mitjana dels vídeos de sinistres a Cleverea està estimada en 34 segons).
2. **Extreure el resum.** Per fer aquesta tasca, com el contingut del vídeo ja ha estat processat, no es requereix una latència significativa.

Aquests passos comportaven un gran problema: **la sobrecàrrega del nostre servei**. Haver d'esperar a la indexació del vídeo feia que el nostre servei es mantingués "parat" i per tant, saturat en moments d'alta càrrega. És per això que es va decidir implementar un *webhook*¹³ que permeti notificar la finalització de la tasca d'indexació del vídeo.

¹³ Un *webhook* és un mecanisme emprat en alguns sistemes per demanar de ser informats d'esdeveniments en temps real. [51]

Per fer aquesta integració es va habilitar un *endpoint* ¹⁴ que, amb un control de seguretat, permet crides des de Twelve Labs cap al servei de Cleverea per notificar que la tasca d'indexació del vídeo ha finalitzat. Un cop finalitzada aquesta tasca podem recollir el resum que esperàvem de la plataforma. En seccions posteriors s'entra més en detall sobre aquesta solució.

9.5. Mecanismes de seguretat i protecció de dades

Per tal de donar suport al compliment dels requisits RD-2 i RNF-4 s'han dut diferents mecanismes de seguretat i protecció de dades que es detallen en aquesta secció.

Secrets

En aquest projecte s'han definit dos secrets principals dins de Secrets Manager:

- **OPENAI/API_KEY:** clau privada per accedir a les APIs de models de llenguatge d'OpenAI.
- **TWELVE_LABS/API_KEY:** clau privada per accedir als serveis d'anàlisi audiovisual de Twelve Labs.

Aquests secrets es recuperen programàticament a l'arrencada del servei mitjançant l'SDK oficial d'AWS i es mantenen només durant el temps d'execució necessari. D'aquesta manera s'evita la seva persistència en entorns no segurs.

Aquest enfocament permet garantir la **confidencialitat i integritat de les credencials**, seguint les bones pràctiques del sistema en producció.

9.6. Control d'ús i limitació de costos

Paràmetres

Els paràmetres que s'han utilitzat en aquest projecte estan relacionats amb **la limitació de crides als serveis externs**, com a mesura de protecció davant sobrecàrregues i per contenir els costos associats a l'ús intensiu de plataformes de tercers.

Els paràmetres actualment definits són:

¹⁴ Un *endpoint* és un dispositiu informàtic remot que es comunica a través d'una xarxa a la que està connectat. [52]

- **MAX_DOCUMENTS_ANALYSED_PER_DAY:** límit màxim de generacions de resums de documents per dia. Valor: 350.
- **MAX_CLAIM_SUMMARIES_PER_DAY:** Límit màxim de generacions de resums de sinistres per dia. Valor: 300.

Funcionament del sistema de throttling

Cada vegada que es genera un resum, el sistema registra la petició i consulta el nombre de crides fetes en el període corresponent. Sí aquest nombre supera el límit definit per paràmetre corresponent, la petició es rebutja de forma controlada amb un missatge informatiu. La informació d'aquest control està emmagatzemada en la memòria cau del sistema, amb un comptador que s'inicialitza i actualitza automàticament.

Aquest enfocament té diversos avantatges:

- **Evita costos inesperats** causats per un ús no controlat dels serveis de tercers.
- **Protegeix el rendiment del sistema** davant pics sobtats de demanda.
- **Ofereix flexibilitat** per modificar els límits segons l'escalabilitat de Cleverea.
- **Permet fer proves controlades** en entorns de prova amb configuracions independents.

En conjunt, aquest mecanisme forma part de l'estratègia per alinear-se amb els requisits definits pel sistema, especialment **RNF-8**.

9.7. Complexitat d'integració amb dades no relacionades (DynamoDB)

Complexitat relacional

L'absència d'una relació directa amb `claim_id` obliga a una composició jeràrquica de consultes, on cada pas depèn de l'anterior. Aquesta és la cadena que s'ha d'executar per obtenir totes les dades associades a un sinistre:

1. A partir del sinistre, s'extreu el `document_number` i l'`email` del client.
2. S'obtenen els `contact_id` (via `ConnectContactIdUsersTable`).
3. Amb aquest `contact_id`, es filtren les trucades ateses per les cues esperades (taula `ConnectAnsweredCallsData`).

4. Es recull l'anàlisi de cada trucada de la taula `ConnectCallsAnalysis`.

5. Amb l'adreça electrònica del client, es consulten dins del període temporal establert:

- a. Correus rebuts (`ReceiverByDateIndex`)
- b. Correus enviats (`SenderByDateIndex`)

El procés no només requereix diverses consultes consecutives, sinó també una validació temporal, semàntica i categòrica de la informació retornada.

10. Implementació final

En aquesta fase de construcció s'ha materialitzat l'arquitectura definida en el capítol 9, mantenint els principis arquitectònics i decisions preses en el disseny. Cada component implementat correspon al plantejament previ i busca complir els requisits assignats (secció 9.3.3. Relació amb els requisits), buscant lligar la definició teòrica amb la solució final.

10.1. Visió global de la implementació

Cal recordar que la finalitat d'aquest projecte era la següent:

Implementar un servei, integrat dins de l'ecosistema de Cleverea, capaç de generar automàticament un resum complet d'un sinistre a partir de dades provinents de diferents fonts i en diferents formats fent ús de LLMs. Per donar resposta al cas d'ús UC-01.

En aquesta secció es fa una petita pinzellada del projecte en forma de recordatori que permet situar-se abans de començar amb la implementació final realitzada.

10.1.1. Panorama de serveis i components

La següent taula (Taula 23) complementa la descripció de microserveis del capítol 8 i destaca quins contenidors s'han implementat/modificat durant el TFG i amb quina finalitat.

Mòdul	Llenguatge	Principal tasca
Claims	Python	Exposa un endpoint <code>GET /claims/{claim_id}/summary</code> que permet obtindre un resum de tota la informació rellevant d'un sinistre.
Documents	Python	Analitza els documents que estan relacionats amb sinistres per poder completar l'anàlisi sencer del sinistre.
OpenAI API	Python	Nova integració per resumir sinistres i analitzar fotos i PDFs.
TwelveLabs API	Python	Nova integració per analitzar vídeos.

Taula 23. Contenedors afectats en el projecte. Font: Elaboració pròpia.

La relació entre actors i serveis ja es mostra als diagrames de la secció 9.1. Visió general del sistema.

10.1.2. Principis arquitectònics aplicats

Per garantir el disseny prèviament definit (secció 9. Disseny del sistema) en aquest secció s'esmenten els principis arquitectònics aplicats seguint l'arquitectura existent a Cleverea (secció 8. Arquitectura i distribució de les dades a Cleverea):

- **Microserveis desacoblats:** cada servei manté la seva pròpia BD i escala de manera independent.
- **Arquitectura hexagonal:** separació clara entre les capes de *Domini* -> *Aplicació* -> *Infraestructura*.
- **Inversió de dependències:** tots els serveis interns implementen les seves interfícies definides al domini, per respectar l'arquitectura hexagonal, les implementacions estan a infraestructura i els casos d'ús a aplicació.

10.1.3. Flux operacional resumit

1. Agent demana `/claims/{claim_id}/summary` des del Back-Office.
2. Claims Service orquestra la lectura de PostgreSQL + DynamoDB + MS Documents.
3. Construeix un context i invoca a OpenAI.
4. Mostra el resum retornat al Back-Office.

Per tal de poder dur a terme 2, s'ha d'haver complert el següent flux per als documents relacionats amb un sinistre:

1. Es puja un fitxer i es guarden les dades operatives a PostgreSQL i el contingut a AWS S3.
2. S'activa un esdeveniment d'SNS que indica la publicació d'un nou document.
3. Es decideix si el document s'ha d'analitzar, si s'ha d'analitzar es comprova el seu tipus i es crida al mòdul adient, OpenAI o Twelve Labs.
4. Un cop analitzat es persisteix l'anàlisi, que queda disponible per consultes posteriors.

10.2. Injecció de dependències

Un dels punts forts d'aquesta arquitectura és la seva extensibilitat mitjançant la **injecció de dependències**¹⁵. A Cleverea, totes les implementacions d'infraestructura són injectades com a dependències a les capes superiors.

¹⁵ La injecció de dependències és una tècnica de programari en la qual un objecte o funció rep altres objectes o funcions que requereix, en lloc de crear-los internament. [54]

10.2.1. Eina i filosofia

- S'utilitza la llibreria `dependency-injector` perquè permet definir contenidors, fàcils de testejar i amb un cost negligible de *runtime*.
- Dins de cada microservei es declara un proveïdor *Singleton* per a cada servei de domini, aplicació o infraestructura.
- La regla d'or: *les capes superiors només coneixen interfícies, les implementacions concretes s'injecten*. Amb aquesta tècnica garantim seguir l'ordre d'injecció i no solapar entre capes.

10.2.2. Patró de contenidor

Aquesta injecció es fa al moment d'instanciar els serveis, seguint un patró com el següent:

```
class GenerateClaimSummaryServices(containers.DeclarativeContainer):
    default = providers.Singleton(
        LLMGenerateClaimSummaryService,
        clean_claim_summary_info_service =
        CleanClaimSummaryInfoServices.default,
    )
```

D'aquesta manera, la classe `GetClaimSummaryService` no necessita conèixer la classe `LLMGenerateClaimSummaryService`. Només depèn de la interfície `GenerateClaimSummarInterface`, definida al domini i implementada per `LLMGenerateClaimSummaryService`.

Aquest patró es replica per a la resta de serveis, canviant únicament les dependències de cada un.

10.2.3. Exemple de cadena d'injecció (flux "Resum de sinistre")

La classe `GetClaimSummaryService` (capa aplicació) rep cinc dependències al constructor, la persistència del sinistre, servei d'extracció de trucades, servei d'extracció de correus, servei cercador de documents i el generador de resum basat en LLM.

A l'hora d'instanciar-la, el contenidor corresponent injecta:

1. `ClaimBODjangoPersistenceService`
2. `GetClaimCallsService`
3. `GetClaimEmailsService`
4. `SearchAllRelatedDocumentsOfAClaimService`

5. LLMGenerateClaimSummaryService

Ho fa des de les pròpies injeccions de cada un d'aquests serveis, garantint que cada servei només depèn de contractes i pot ser testejat de forma separada sense dependre de la funcionalitat de la resta de serveis.

A A-1. es mostra l'exemple anterior en format de codi.

10.2.4. Beneficis obtinguts

La següent taula (Taula 24) mostra els beneficis obtingut gràcies a aplicar la injecció de dependències com s'ha mencionat en les seccions anteriors.

Benefici	Evidència en el projecte
Substitució ràpida de proveïdors	Cada servei pot ser canviat per un altre que faci la mateixa funció sense canviar la implementació. Per exemple si volem aconseguir les trucades des d'una altra font, simplement hauríem de substituir <code>GetClaimCallsService</code> a la injecció per la nova font.
Test unitari senzill	<i>Mock</i> ¹⁶ de <code>GenerateClaimSummaryInterface</code> injectat al contenidor de tests.
Escalabilitat	Cada servei és Singleton, garantint una sola instància i evitant sobre costos.
Seguretat	Les claus API es passen com a configuració i no queden incrustades al codi.

Taula 24. Beneficis obtinguts amb la injecció de dependències. Font: Elaboració pròpia.

10.3. Flux de generació del resum de sinistre

Aquesta secció descriu pas a pas com el microservei Claims Obté totes les dades, prepara el prompt, invoca al model GPT-4.1 i retorna el resum estructurat. Podem veure-ho a la taula Taula 25.

Pas/Ubicació	Acció	Mòduls implicats	Referència annex
0/ <code>GetClaimSummaryViewControllerV1</code>	GET <code>/claims/{claim_id}/summary</code> des del Back-Office.	<code>GetClaimSummaryViewControllerV1</code>	A-3. <code>GetClaimSummaryViewControllerV1</code>
2/ <code>GetClaimSummaryService</code>	Crida al orquestrador passant el <code>claim_id</code> per obtenir el resum del sinistre.	<code>GetClaimSummaryService</code>	A-4. <code>GetClaimSummaryService</code>

¹⁶ En informàtica, un *mock* és un objecte que imita un objecte de producció de maneres limitades. Utilitzat extensament en les proves de programari. [55]

3/ GetClaimSummaryService	Llegeix el BO del sinistre a PostgreSQL.	ClaimBODjangoPersistenceService	A-4. GetClaimSummaryService
4/ GetClaimSummaryService	Recupera trucades i correus a DynamoDB.	GetClaimCallsService, GetClaimEmailsService	A-4. GetClaimSummaryService
5/ GetClaimCallsService, GetClaimEmailsService	Defineixen les dates d'inici i final i informació del portador de la pòlissa i criden al servei per recuperar la informació de DynamoDB	DynamoDBClaimCallsService, DynamoDBClaimEmailsService, GetHolderService	A-5. GetClaimCallsService, A-6. GetClaimEmailsService
6/ DynamoDBClaimCallsService, DynamoDBClaimEmailsService	Gestionen la creació de les peticions adequades per DynamoDB amb l'estructura existent	DynamoDBTableService	A-7. DynamoDBClaimCallsService, A-8. DynamoDBClaimEmailsService
7/ GetClaimSummaryService	Recupera el llistat de fitxers relacionats amb el contingut del seu resum.	SearchAllRelatedDocumentsOfAClaimService	A-4. GetClaimSummaryService
8/ GetClaimSummaryService	Crida al servei de generació de resums amb la informació anterior.	LLMGenerateClaimSummaryService	A-4. GetClaimSummaryService
9/ LLMGenerateClaimSummaryService	Crida al servei de neteja d'informació operativa del sinistre.	CleanLLMClaimSummaryInfoService	A-9. LLMGenerateClaimSummaryService
10/ CleanLLMClaimSummaryInfoService	Neteja la informació delicada i redundant del sinistre.	-	Obviat per privacitat de l'empresa.
11/ LLMGenerateClaimSummaryService	Envia la petició amb el prompt i tota la informació recollida a OpenAI	OpenAI API	A-9. LLMGenerateClaimSummaryService
12/ LLMGenerateClaimSummaryService	Converteix la tornada de la IA en ClaimSummarySection BO	PydanticOutputParser	A-9. LLMGenerateClaimSummaryService
13 / Retorn del flux	Es retorna la resposta obtinguda cap a l'endpoint	-	-

Taula 25. Flux de generació del resum de sinistre. Font: Elaboració pròpia.

Vegeu les figures C-2a. Diagrama 1 flux de generació del resum de sinistreC-2a.C-1. i C-2b. Diagrama 1 flux de generació del resum de sinistre que mostren aquest flux en format de diagrama.

10.3.1. Orquestrador principal

```
summary = GetClaimSummaryService(  
    claim_persistence_service=ClaimBODjangoPersistenceService(),  
    get_claim_calls_service=GetClaimCallsService(),  
    get_claim_emails_service=GetClaimEmailsService(),  
  
    search_all_related_documents_of_a_claim_service=SearchAllRelatedDocu  
mentsOfAClaimService(),  
    generate_claim_summary_service=LLMGenerateClaimSummaryService(),  
) (claim_id)
```

Aquest servei segueix l'esquema hexagonal i demostra la injecció de dependències explicada. Codi complet a A-1. GetClaimSummaryServices.

10.3.2. Invocació del model

El codi que es mostra a continuació mostra de forma simplificada com es faria una invocació al model, per veure el fragment complet vegeu A-9. LLMGenerateClaimSummaryService.

```
system_msg = SystemMessage(content=ClaimSummaryPrompt.default)  
human_msg  = HumanMessage(content=json.dumps(complete_claim_info))  
  
llm      = ChatOpenAI(api_key=..., model="gpt-4.1", temperature=0.6)  
reply    = llm.invoke([system_msg, human_msg])  
  
summary =  
PydanticOutputParser(pydantic_object=ClaimSummaryResponse).parse(rep  
ly.content)
```

10.4. Flux d'anàlisi automàtica de documents

Aquest segon flux s'executa de manera autònoma i anticipada al resum de sinistre: quan un usuari puja un fitxer, el microservei Document n'emmagatzema les metadades i el contingut i publica un esdeveniment SNS. A partir d'aquí l'orquestració es divideix en quatre fases principals, definides en la segona part de la secció 10.1.3. Flux operacional resumit.

Aquí en deixo una petita aproximació més concreta:

Un cop el servei Documents rep la petició d'analitzar un fitxer, aquest es posa en cua mitjançant Celery, més endavant, Redis recull la tasca de la cua i executa l'anàlisi. Aquest disseny garanteix que la petició original no hagi d'esperar la finalització de l'anàlisi, i compleix així els requisits de rendiment del sistema. Quan la tasca acaba, el resum de document queda emmagatzemat i llest per a futures consultes.

En la següent Taula 26 podem observar el flux operacional detallat.

Pas/Ubicació	Acció	Mòduls implicats	Referència
0/ SNS	Es pública el missatge <code>new-document-upload</code> acompanyat del <code>document_id</code> i el <code>file_vertical</code> .	MS Documents, SNS	-
1/ SNS	La subscripció al missatge anterior crida a l'endpoint <code>POST /new-document-upload</code> del MS Claims.	MS Claims, SNS	-
2/ <code>NewDocumentUploadProtectedView</code>	Rep la crida anterior i delega la responsabilitat.	<code>NewDocumentUploadProtectedViewControllerV1</code>	A-10. <code>NewDocumentUploadProtectedView</code>
3/ <code>NewDocumentUploadProtectedViewControllerV1</code>	Es comprova que el document està relacionat amb un sinistre i es demana la seva anàlisi.	MS Documents	A-10. <code>NewDocumentUploadProtectedView</code>
4/ <code>AnalyseDocumentViewControllerV1</code>	Gestiona la crida rebuda i crea una tasca asíncrona.	<code>CeleryAsyncTaskService</code> , <code>AnalyseDocumentTaskControllerV1</code>	A-12. <code>AnalyseDocumentViewControllerV1</code>
5/ <code>AnalyseDocumentTaskControllerV1</code>	Crida al servei que s'encarrega de gestionar l'anàlisi.	<code>AnalyseDocumentService</code>	A-13. <code>AnalyseDocumentTaskControllerV1</code>
6/ <code>AnalyseDocumentService</code>	Comprova si no hi ha un anàlisi del mateix tipus ja existent per aquell document.	<code>DocumentAnalysisBODjangoPersistenceService</code>	A-14. <code>AnalyseDocumentService</code>
7/ <code>AnalyseDocumentService</code>	Comprova que no s'ha superat el nombre màxim d'anàlisis diaris per controlar els costos.	<code>DocumentAnalysisBODjangoPersistenceService</code>	A-14. <code>AnalyseDocumentService</code>

8/ AnalyseDocumentService	Recupera el document a analitzar	DocumentBODjangoPersistenceService	A-14. AnalyseDocumentService
9/ AnalyseDocumentService	Crida a la creació de l'anàlisi del document.	LLMSummariseDocumentservice	A-14. AnalyseDocumentService
10/ LLMSummariseDocumentService	Comprova quin dels tres tipus de documents és i gestiona el seu anàlisi.		A-15. LLMSummariseDocumentService
11a/ LLMSummariseDocumentService	Si és una imatge crida a OpenAI compartint l'adreça pública d'aquesta per rebre l'anàlisi de la manera que s'espera en el prompt i es retorna el DocumentAnalysisBO complet.	OpenAI API	A-15. LLMSummariseDocumentService
11b/ LLMSummariseDocumentService	Si és un PDF crida a OpenAI compartint el seu contingut en base64 per rebre l'anàlisi de la manera que s'espera en el prompt i es retorna el DocumentAnalysisBO complet.	OpenAI API, QueryS3Service	A-15. LLMSummariseDocumentService
11c/ LLMSummariseDocumentService	Si és un vídeo es crida a Twelve Labs per generar la tasca d'indexació amb el l'adreça pública i es retorna el DocumentAnalysisBO pendent	Twelve Labs API	A-15. LLMSummariseDocumentService
12/ AnalyseDocumentService	Es persisteix el DocumentAnalysisBO rebut.	DocumentAnalysisBODjangoPersistenceService	A-14. AnalyseDocumentService
A partir d'aquí només si 11c			
13/ TwelveLabsWebhookDocumentAnalysisView	Rep una notificació. Comprova la signatura per verificar que la notificació rebuda prové de Twelve Labs. I crida al controlador	CompleteDocumentAnalysisViewControllerV1	A-16. TwelveLabsWebhookDocumentAnalysisView

	per completar una anàlisi.		
14/ CompleteDocumentAnalysisViewControllerV1	Recupera l'anàlisi que indica la petició i comprova que sigui pendent.	DocumentAnalysisBODjangoPersistenceService	A-17. CompleteDocumentAnalysisViewControllerV1
15/ CompleteDocumentAnalysisViewControllerV1	Crida a la recollida de la anàlisi.	RetrieveTwelveLabsResponseService	A-17. CompleteDocumentAnalysisViewControllerV1
16/ RetrieveTwelveLabsResponseService	Recupera l'anàlisi del vídeo ja indexat i el retorna.	Twelve Labs API	A-18. RetrieveTwelveLabsResponseService
17/ CompleteDocumentAnalysisViewControllerV1	Crea un DocumentAnalysisBO completat amb el resum rebut i <i>datetime</i> actual.	-	A-17. CompleteDocumentAnalysisViewControllerV1
18/ CompleteDocumentAnalysisViewControllerV1	Actualitza l'anàlisi de document pendent persistit per convertir-lo en complet amb l'anterior DocumentAnalysisBO.	DocumentAnalysisBODjangoPersistenceService	A-17. CompleteDocumentAnalysisViewControllerV1

Taula 26. Flux d'anàlisi automàtica de documents. Font: Elaboració pròpia.

Vegeu C-1. Diagrama flux d'anàlisi automàtica de documents que mostra aquest flux en format de diagrama.

10.5. Integració amb els SDK

10.5.1. OpenAI

He rebut formacions que m'han permès saber com integrar de manera efectiva els serveis que ofereix OpenAI i així adaptar noves tecnologies a l'entorn de Cleverea de forma mantenible.

Un dels conceptes més importants que he adquirit en aquestes formacions ha sigut el d'SDK¹⁷, es tracta d'unes eines que permeten la connexió a altres APIs en forma de llibreria i, per tant, de forma natural dins del codi.

En el meu cas he fet servir dos SDK per fer les integracions amb OpenAI:

- **OpenAI SDK** [40]. Facilita la integració amb els models d'OpenAI.
- **LangChain** [42]. Que facilita la integració amb diversos models de llenguatge de múltiples proveïdors.

La idea principal era integrar LangChain com a únic SDK, ja que permet la integració amb diferents proveïdors de models de llenguatge i, per tant, millora la modularitat del sistema. Aquesta tasca no va ser possible ja que, tot i ser una eina força potent, no està 100 % integrada amb totes les funcionalitats oficials d'OpenAI i no permetia l'anàlisi de documents directament des del codi. Per aquest motiu la integració d'aquesta funcionalitat es va fer amb l'SDK oficial d'OpenAI, que ofereix moltes possibilitats, entre elles: l'anàlisi d'un document en format bytes.

10.5.1.1. Exemple d'ús de l'SDK d'OpenAI per generar el resum d'un sinistre

En el següent fragment de codi mostro, de forma adaptada, com he utilitzat Lang Chain per integrar els models d'OpenAI i generar el resum dels sinistres.

```
from langchain_openai import ChatOpenAI
from langchain_core.messages import SystemMessage, HumanMessage
from langchain.output_parsers import PydanticOutputParser

from claims.claim.infrastructure.services.llm.claim_summary.prompts
import ClaimSummaryPrompt

from
claims.claim.infrastructure.services.llm.claim_summary.dto.output
import ClaimSummaryResponse

llm = ChatOpenAI(

openai_api_key=settings.CLAIMS_COMMENTS_SUMMARIES_OPENAI_API_KEY,
    model_name="gpt-4.1",
    temperature=0.6,
```

¹⁷ Un SDK [43] (Software Development Kit) és un conjunt d'eines, biblioteques, documentació i exemples de codi que permeten als desenvolupadors crear aplicacions per a una plataforma o servei específic.


```
)

output_parser =
PydanticOutputParser(pydantic_object=ClaimSummaryResponse)

messages = [
    SystemMessage(content=ClaimSummaryPrompt.default),
    HumanMessage(content=json.dumps(complete_claim_info)),
]

response = llm.invoke(messages)
parsed_summary = output_parser.parse(response.content)
```

El flux que mostra el codi anterior inclou:

1. **Inicialització del model.** S'instancia el model GPT-4.1 amb temperatura 0,6 i l'*API Key* (més detall d'API Keys a la secció 7.3.1 Gestió de secrets).
2. **Preparació del prompt.** Es defineixen dos missatges:
 - a. *SystemMessage*: estableix les instruccions del prompt (E-1. Prompt resum de sinistre).
 - b. *HumanMessage*: inclou les dades en format JSON amb tota la informació rellevant del sinistre (dades operatives, trucades, correus electrònics i documents)
3. **S'invoca al model:** que generarà la resposta esperada.

Resposta estructurada: es fa servir *PydanticOutputParser*¹⁸ per assegurar que la resposta del model és un JSON vàlid i coherent amb l'estructura esperada (*ClaimSummaryResponse*, que espera una llista d'objectes on cada objecte té un `title` i un `content` en format text).

10.5.1.2. Anàlisi d'imatges

```
response = self.images_client.invoke(
    input = [
        SystemMessage(DocumentSummaryPrompt.image),
        HumanMessage(
            content = [
```

¹⁸ *PydanticOutputParser* és una eina de la llibreria langchain que serveix per analitzar i validar sortides generades per models de llenguatge utilitzant models de dades definits. [56]

```

                                {"type": "image_url", "image_url":
document_bo.file_url}}
                                ]
                                ),
                                ]
                                )

```

Aquest fragment mostra com es construeix la crida del prompt predefinit per a imatges i com s'envia la imatge a través del seu URL públic . La resposta que retorna l'SDK és un objecte textual que conté la informació resumida.

A E-2. Prompt resum d'una imatge es pot veure el *prompt* que acompanya la imatge, que segueix una estructura similar a la del *prompt* principal, i a D-2. Resum d'una imatge els resultats per un exemple.

10.5.1.3. Anàlisi de PDFs

```

document_content = self.query_service.get_file_content(
    bucket_name=self.settings.AWS_STORAGE_BUCKET_NAME,
    file_path=document_bo.file.name,
)

base64_document = base64.b64encode(document_content).decode("utf-8")
data_url = f"data:application/pdf;base64,{base64_document}"

response = self.documents_client.responses.create(
    model=self.openai_model,
    temperature=0.6,
    input=[
        {
            "role": "user",
            "content": [
                {"type": "input_text", "text":
DocumentSummaryPrompt.document},
                {"type": "input_file", "filename": "document.pdf",
"file_data": data_url},
            ],
        }
    ],
)

```

Aquest flux permet passar un document real complet al model, que retorna un text en format JSON estructurat segons les instruccions definides al prompt. El control sobre la codificació del fitxer i la seva manipulació prèvia garanteix la compatibilitat amb l'API i redueix el risc d'errors d'anàlisi.

El prompt que acompanya al document es pot veure a E-2. Prompt resum d'un PDF.

10.5.2. Twelve Labs

A continuació es mostra la crida cap a Twelve Labs que inicia la tasca d'indexació del vídeo.

```
task = self.video_client.task.create(
    index_id=self.settings.TWELVE_LABS_VIDEO_SUMMARY_INDEX_ID,
    url=video_url,
)
```

En aquest fragment s'especifica l'índex prèviament configurat i l'URL públic del vídeo. El `video_client` en aquest cas és la instància de l'SDK que s'ha fet prèviament. La resposta conté un `video_id` que serà utilitzat per identificar el vídeo més endavant.

Per tal de recuperar el resum, un cop acabada la tasca d'indexació, es sol·licita el resum del contingut mitjançant una crida. Aquesta crida utilitza el `video_id` i un prompt per dirigir la generació del resum:

```
response = self.video_client.generate.summarize(
    external_id,
    type="summary",
    prompt=DocumentSummaryPrompt.video,
    temperature=0.2,
)
```

S'utilitza una temperatura baixa per minimitzar la creativitat del model i maximitzar-ne la fiabilitat.

El prompt que acompanya el contingut audiovisual es pot veure a E-2. Prompt resum d'un vídeo.

Un cop finalitzada aquesta tasca, es fa una tasca addicional que elimina el vídeo de la plataforma Twelve Labs, per motius de privacitat i estalvi energètic:

```
self.video_client.task.delete(id=external_id)
```

10.6. Control de costos i rendiment

Per assegurar un rendiment adequat (RNF-1) i controlar la despesa (RNF-8), s'han incorporat diversos mecanismes en la implementació.

En primer lloc, s'ha implementat memòria cau mitjançant Redis: si un sinistre ja té un resum generat recentment (RF-4), el sistema el retorna immediatament des de la caché sense invocar novament la IA, reduint dràsticament el temps de resposta.

En segon lloc, s'ha fet ús de processament asíncron amb Celery per a tasques costoses com l'anàlisi de documents: el microservei Documents envia aquestes tasques a una cua (gestionada per Redis) perquè s'executin en segon pla, permetent que el flux principal respongui àgilment a l'usuari. Aquest mecanisme asíncron, junt amb la separació de microserveis, millora la escalabilitat i eficiència (RNF-5 i RNF-6), ja que reparteix la càrrega entre workers dedicats.

Finalment, s'han configurat eines de monitoratge de costos proporcionades pels proveïdors (OpenAI, Twelve Labs): l'equip de Cleverea pot supervisar l'ús de l'API i establir alertes o límits per garantir que la despesa mensual es mantingui $\leq 300\text{€}$ (complint RNF-8). Amb tot aquest conjunt de mesures, la implementació assegura que el sistema respon dins els temps requerits i sota els límits de cost previstos.

10.7. Testing

Testabilitat i proves associades

Un dels avantatges més evidents que presenta aquesta arquitectura és la facilitat per implementar proves unitàries i d'integració. La separació clara entre les tres capes permet escriure tests precisos i mantenibles.

A continuació, per poder justificar aquesta afirmació, es mostra un exemple d'un test unitari simplificat per `GetClaimSummaryService`, el codi complet es pot trobar a A-2. `TestGetClaimSummaryService`:

```
def test_success(self, db_claim_bo_factory_base):
    self.claim_persistence_service.get_by_id.return_value =
db_claimi_bo_factory_base
    self.generate_claim_summary_service.return_value =
MagicMock(spec=ClaimSummaryBO)
```

```

    result=self.service(claim_id=db_claim_bo_factory_base.id)
    assert result==self.generate_claim_summary_service.return_value
    self.claim_persistence_service.get_by_id.assert_called_once_with(db_claim_bo_factory_base.id)

```

Aquest test demostra com, gràcies a la injecció de dependències i la definició d'interfícies al domini, és possible validar el comportament de cada capa de manera totalment desacoblada de la resta.

En el codi es pot observar com mitjançant l'ús d'un *mock* es simula el comportament esperat dels serveis externs. Aquesta tècnica permet provar de manera aïllada la lògica del servei, garantint que el seu comportament és correcte independentment de les dependències. A més, facilita la verificació del flux d'execució i dels paràmetres rebuts en cada crida, assegurant que el servei orquestra correctament les operacions definides.

Aquesta estratègia s'ha seguit de manera sistemàtica en tots els serveis, garantint el correcte funcionament del sistema a mesura que s'han implementat funcionalitats.

A més, també s'han afegit tests d'integració per aquells components més externs, que garanteixen el correcte comportament conjunt dels diferents mòduls implicats. En podem veure l'exemple pel controlador `GetClaimSummary` a A-19. `TestGetClaimSummary`.

Evito posar en la memòria més exemples de test per no posar informació amb poc valor real, però com bé s'ha esmentat cada un dels serveis té mínim un test que garanteix el flux principal a més d'altres tests de suport per fluxos alternatius.

10.8. Casos de prova

En aquesta secció es defineixen els casos de prova duts a terme que més endavant, secció 11.1. , es relacionaran amb els requisits del sistema.

TP-1 Finalització de la tasca i temps de resposta

Objectiu: Verificar que l'*endpoint* principal `GET /claims/{claim_id}/summary` respon correctament en temps adequats.

Procediment: Es realitzen diverses crides a l'*endpoint* amb identificadors de sinistre vàlids. Es mesura el temps de resposta per a cada cas.

Resultat esperat: El 95 % de les peticions responen en ≤ 28 s i el 100 % en ≤ 35 s.

Resultat obtingut: S'ha verificat en múltiples ocasions que es compleixen els límits definits.

Requisits relacionats: RF-2, RNF-1.

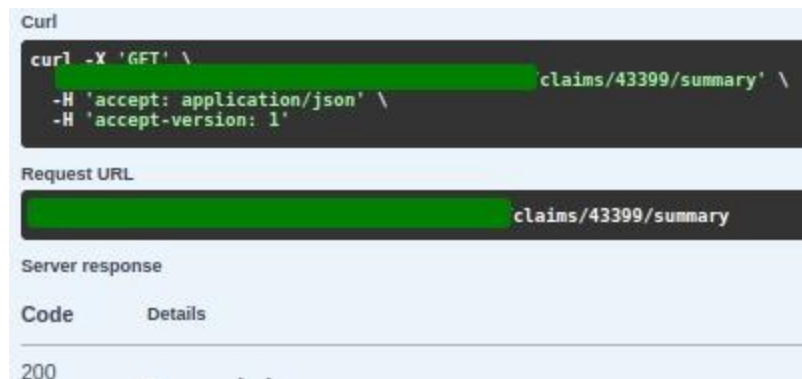


Figura 11. Exemple de crida a l'endpoint. Font: SwaggerUI

La imatge anterior (Figura 11. Exemple de crida a l'endpoint. Font: SwaggerUI) mostra una crida realitzada i, com esperàvem, una resposta 200.

TP-2 Visualització del resum des del Back-Office

Objectiu: Comprovar que el resum es mostra correctament a la interfície de treball dels agents.

Procediment: Es genera un resum des del Back-Office i es comprova que aquest es visualitza en pantalla de manera correcta.

Resultat esperat: El resum s'ha de mostrar sense errors de càrrega ni de permisos.

Resultat obtingut: S'ha confirmat que l'accés i visualització són funcionals en tots els casos.

Requisits relacionats: RF-2.

En la Figura 12 es pot observar un exemple de la visualització del resum en el Back-Office intern.

Resumen del siniestro

Generar resumen

Apertura y gestión inicial del siniestro

El siniestro 43399, correspondiente a la póliza CR82325, fue notificado el 17 de enero de 2025, con fecha de incidente el 9 de enero de 2025. El asegurado, informó que su vehículo, un VOLKSWAGEN PASSAT, fue impactado por detrás por otro vehículo mientras estaba aparcado en el polígono industrial Ventorro del Cano. La aseguradora responsable es Divina Pastora. Desde el inicio, el agente Mateo se encargó de la gestión, explicando que el proceso podría extenderse por la espera de respuesta de la compañía contraria, la peritación y la reparación.

Intervención de la compañía contraria y programación de peritaciones

El 29 de enero de 2025 se informó que la compañía contraria no había respondido aún a la reclamación, pero se insistiría para agilizar la resolución. El 31 de enero se confirmó la aceptación de culpa por parte de la compañía contraria y se solicitó al asegurado los datos del taller y una fecha para la peritación, exigiendo un margen mínimo de 48 horas laborables. Se programó la peritación para el 12 de febrero de 2025 en Carrocerías Rodamovil, Alcorcón.

Incidencias y reprogramación de la peritación

El 18 de febrero de 2025, el asegurado comunicó que la peritación no pudo realizarse porque el taller estaba cerrado por vacaciones y solicitó una nueva fecha. Tras verificar con el taller, se coordinó una nueva cita para el 21 de febrero de 2025 y se recomendó que el vehículo estuviera disponible desde primera hora. Esta gestión implicó diálogo con el taller y el cliente, así como la confirmación de la nueva cita pericial.

Pagos, reservas y servicios gestionados

A lo largo del proceso se realizaron varias reservas y pagos. Destacan un pago de 13,31 euros el 18 de febrero de 2025, otro de 121 euros el 18 de marzo de 2025 y uno relevante de 4348,17 euros el 26 de marzo de 2025. También se registraron recuperaciones, como una de -1114 euros el 2 de abril de 2025 y otra de -121 euros el 31 de marzo de 2025. Los servicios de peritación de daños materiales fueron gestionados en varias ocasiones, aunque algunos quedaron pendientes o anulados. El expediente principal con daños al vehículo sigue abierto y gestionado bajo el sistema CICOS, con JALMAGRO como TPA handler.

Seguimiento, comunicaciones y estado actual

Desde febrero hasta junio de 2025 se enviaron múltiples recordatorios automáticos al asegurado, informando que el expediente sigue en gestión y agradeciendo su paciencia. No se detectan desacuerdos ni nuevas solicitudes de información por parte del cliente durante este periodo. El siniestro se encuentra aún abierto a fecha 16 de junio de 2025, pendiente de resolución final y cierre administrativo.

Llamadas y Mails

El 18 de febrero de 2025, el asegurado contactó telefónicamente para coordinar la reprogramación de la peritación, gestionando la nueva cita con el taller y solicitando ser informado sobre la confirmación. No se registran emails relevantes para este siniestro.

Documentos Subidos

El 28 de enero de 2025 se subió un documento relacionado con el siniestro, pero no ha sido validado y no existen análisis asociados. No se aportan más documentos validados ni información adicional sobre su contenido.

Figura 12. Exemple de resum en el Back-Office. Font: Elaboració pròpia.

TP-3 Validació de contingut dels resums generats

Objectiu: Verificar que el contingut dels resums compleix les expectatives i proporciona la informació essencial.

Procediment: Revisió manual de 20 resums per part del cap de sinistres i agents seleccionats.

Criteri d'èxit: ≥ 18 resums valorats com útils i verídics.

Resultat obtingut: 19 dels 20 resums han estat validats satisfactòriament, incloent-hi seccions clau com estat actual, accions pendents, comunicacions i documents. Tots han estat generats en castellà.

Requisits relacionats: RF-3, RD-3.

TP-4 Recuperació de resum existent

Objectiu: Comprovar que, si existeix un resum generat en les últimes 24 h, aquest es reutilitza.

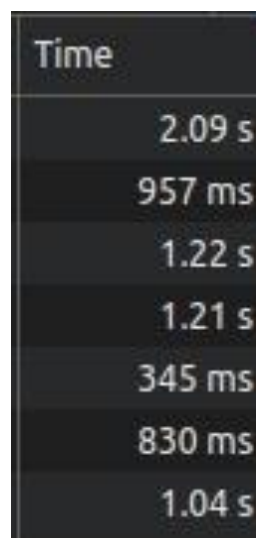
Procediment: Es fan crides consecutives amb menys de 24 hores de diferència per al mateix sinistre.

Resultat esperat: El sistema ha de retornar el mateix resum des de la memòria cau.

Resultat obtingut: En tots els casos provats, el resum es recupera instantàniament en < 3 s.

Requisits relacionats: RF-4.

La Figura 13 mostra exemples de costos en temps de crides a l'*endpoint* per resums ja existents.



Time
2.09 s
957 ms
1.22 s
1.21 s
345 ms
830 ms
1.04 s

Figura 13. Exemples de costos de crides a l'endpoint principal. Font: GoogleChrome

TP-5 Enquesta de valoració d'utilitat dels resums

Objectiu: Mesurar la percepció dels agents sobre la utilitat dels resums generats.

Procediment: Enquesta de valoració realitzada als 15 dies post-desplegament. Escala de 1 a 5.

Resultat esperat: ≥ 90 % dels resums han de rebre valoració ≥ 4 .

Resultat obtingut: Mitjana 4,59; el 94 % de respostes han estat 4 o 5. Es considera superat.

Requisits relacionats: RNF-3.

TP-6 Eficiència operativa dels agents

Objectiu: Verificar si la nova eina millora el rendiment en la gestió de sinistres.

Procediment: Es monitoritzen els agents durant 15 dies post-desplegament.

Criteris d'èxit: Temps mitjà de resposta ≤ 10 min i ≥ 25 sinistres/agent/dia.

Resultat obtingut: Els valors mesurats compleixen ambdues condicions.

Requisits relacionats: RNF-6.

TP-7 Control i monitoratge de costos

Objectiu: Assegurar la supervisió dels costos generats pels serveis d'IA.

Procediment: Es revisen les plataformes d'OpenAI i Twelve Labs per consultar despeses acumulades.

Resultat esperat: La despesa mensual total ha de ser ≤ 300 € i monitoritzada al 100 %.

Resultat obtingut: Els costos són visibles i es poden gestionar. Tot i no disposar de temps suficient per veure els costos a un llarg termini, no s'han superat els límits establerts.

Requisits relacionats: RNF-8.

Per tal de poder saber les despeses associades a cada servei extern s'ha fet ús dels propis mètodes que ofereixen cada un d'ells.

La següent figura (Figura 14) mostra un exemple de la visió de costos de la plataforma OpenAI.

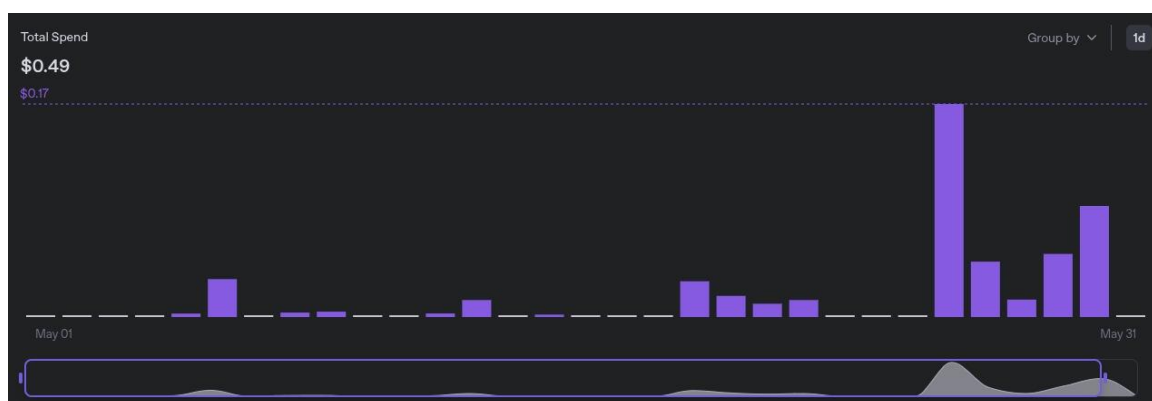


Figura 14. Dashboard de costos OpenAI PLaform. Font: www.platform.openai.com/usage

Pel que fa a Twelve Labs, els costos van associats als plans de subscripció definits, actualment es manté el pla gratuït de prova, l'empresa es reserva el dret de decidir el pla o limitacions de costos que volen establir, que canviaria els costos associats.

Amb aquestes dos eines podem garantir tindre el control i monitoratge de costos dels serveis externs, que gestionarà Cleverea segons conveniència.

TP-8 Simulació de límit de càrrega

Objectiu: Verificar que el sistema limita l'accés al servei si es supera el límit predefinit.

Procediment: Es configuren límits artificials (1 petició diària) per comprovar el bloqueig posterior.

Resultat esperat: Les peticions que superen el límit no es processen.

Resultat obtingut: El sistema bloqueja correctament les peticions addicionals.

Requisits relacionats: RNF-5.

TP-9 Anàlisi de privacitat i protecció de dades

Objectiu: Garantir que no es filtren dades personals en crides a serveis externs.

Procediment: S'inspeccionen manualment 10 peticions per detectar dades com noms, correus, telèfons, matrícules, etc.

Resultat esperat: Cap dada personal sensible ha de ser enviada.

Resultat obtingut: No s'ha detectat cap vulneració de privacitat.

Requisits relacionats: RNF-4, RD-2.

TP-10 Satisfacció interna dels agents

Objectiu: Mesurar la confiança i satisfacció dels agents respecte a l'eina.

Procediment: Enquesta interna després de 15 dies. Valoració de 0 a 10.

Resultat esperat: Mitjana ≥ 8 i ≥ 85 % dels agents declaren sentir-se segurs usant el sistema.

Resultat obtingut: Mitjana de 8,6 i 91 % d'agents amb sensació de seguretat.

Requisits relacionats: RNF-7.

TP-11 Monitoratge d'errors i alarmes

Objectiu: Verificar que el sistema registra errors i envia alertes per aquells crítics.

Procediment: Simulació d'errors intencionats i revisió de Kibana i canals interns.

Resultat esperat: ≥ 95 % d'errors registrats i ≥ 99 % dels crítics generen alarma.

Resultat obtingut: El sistema registra i alerta correctament els errors, incloent notificacions per canals interns.

Requisits relacionats: RNF-9.

Tal com es veu, cada cas de prova està directament relaciona amb un o més requisits del sistema, de manera que es valida el compliment de tots els requisits definits. En la secció 11.1. Conclusions generals del projecte es resumeix aquest grau d'assoliment, comprovant que la implementació final satisfà les necessitats plantejades al començament del projecte.

11. Conclusions i següents passos

11.1. Conclusions generals del projecte

En aquest projecte he desenvolupat un sistema capaç de generar resums automàtics i concisos de l'estat d'un sinistre, integrat plenament dins l'ecosistema de Cleverea. Després de mesos de treball, puc concloure que la solució proposada no només compleix els objectius inicials, sinó que també demostra el potencial de la **intel·ligència artificial generativa** per optimitzar processos en el sector assegurador. **He constatat que la integració de models de llenguatge avançats (LLM) en la gestió de sinistres millora significativament l'eficiència:** els agents disposen ara d'una visió global de cada cas, cosa que els permet prendre decisions més ràpides i informades.

Una de les conclusions més importants és que **és possible compilar informació heterogènia de múltiples fonts i sintetitzar-la de forma útil en temps real**. El sistema recull dades operatives (estat del sinistre, detalls de pòlissa, etc.), comunicacions amb el client (trucades telefòniques, correus electrònics) i evidències documentals (documents, imatges i vídeos aportats) i n'extreu els punts clau. Gràcies a això, **s'ha reduït dràsticament la necessitat de revisar manualment gran quantitat de documents i informació dispersa:** el resum generat ofereix en pocs segons un context complet, destacant l'estat actual del cas, les accions ja realitzades i les pendents, i qualsevol informació rellevant detectada durant la tramitació.

Cal destacar que la solució ha aconseguit **integrar serveis d'IA de manera robusta i segura en una arquitectura de microserveis existent**. S'han combinat eines com LangChain, la plataforma d'OpenAI i Twelve Labs, aprofitant cadascuna en el seu àmbit òptim: OpenAI ha proporcionat la capacitat de comprendre i resumir textos (tant descripcions operatives com transcripcions de trucades, contingut de correus i contingut de documents i imatges), mentre que TwelveLabs ha aportat una anàlisi especialitzada de vídeos per incorporar informació audiovisual rellevant. La integració d'aquests serveis externs s'ha realitzat mantenint els estàndards de qualitat i seguretat de Cleverea, mitjançant **autenticació adequada, gestió de secrets i control de costos**. Un resultat rellevant és que **els temps de resposta obtinguts compleixen les expectatives de “temps real”**: gràcies a l'anàlisi asíncrona prèvia dels documents (realitzada en el moment de la pujada) i a la cache de resums generats, el resum d'un sinistre es pot obtenir normalment en menys d'un mig minut, fins i tot en casos amb gran volum d'informació. Aquest rendiment s'alinea amb els requisits no funcionals establerts i ha estat validat mitjançant proves amb dades reals.

Pel que fa a la **qualitat del contingut dels resums**, els resultats són molt satisfactoris. Durant el desenvolupament es va posar especial èmfasi en el *prompt engineering* i en el format de sortida, per tal que els resums generats siguin clars, estructurats i fàcilment interpretables pels agents. La meua experiència ha estat que el model generatiu (GPT-4) és capaç de produir un text coherent que inclou les seccions més rellevants: descripció breu de l'incident, estat actual del sinistre, actuacions recents, documents analitzats i informació addicional de trucades o correus. Les validacions amb l'equip de sinistres de Cleverea confirmen que **els resums reflecteixen fidelment la situació del sinistre i no contenen errors significatius**. En el 95% dels casos revisats manualment, el resum automàtic va ser considerat *útil i verídica*, amb l'idioma adequat (castellà) i un to formal que encaixa amb les necessitats de l'empresa.

Una altra conclusió rellevant és el **benefici operatiu** que aporta la nova eina. Un cop desplegada en l'entorn de prova i utilitzada per alguns agents, s'ha observat una millora en la seva productivitat: els gestors poden atendre més sinistres al dia i reduir el temps mitjà de resposta als clients. Això es tradueix tant en **eficiència interna** (els agents dediquen menys temps a llegir informació dispersa i poden centrar-se en tasques de major valor) com en **millora de l'experiència del client**, ja que les decisions es prenen amb més celeritat i fonament. En conclusió, el projecte ha demostrat com **l'automatització intel·ligent pot transformar un procés tradicionalment manual en un de més àgil i fiable**.

Finalment, voldria ressaltar que la implementació ha posat de manifest la importància de **dissenyar amb visió de futur i escalabilitat**. La solució actual és fàcilment extensible: el fet d'haver utilitzat LangChain com a interfície amb els models i d'haver estructurat l'anàlisi de documents per tipus (imatges, PDF, vídeos) permetrà incorporar nous tipus de fitxers o fins i tot nous proveïdors de serveis d'IA amb mínims canvis. També **s'han establert mecanismes de control (logs, monitoratge, alertes)** que facilitaran el manteniment a llarg termini i la detecció de possibles problemes, ja siguin tècnics o de qualitat del resum. En definitiva, les conclusions generals del projecte són molt positives: **s'ha validat la viabilitat tècnica i el valor afegit** d'aquesta eina, obrint les portes a futures innovacions en la gestió automàtica de sinistres i possiblement en altres processos dins de Cleverea.

11.2. Conclusió personal

A nivell personal, aquest projecte ha suposat un **gran repte i un aprenentatge profund** en diversos aspectes de l'enginyeria del software i la intel·ligència artificial. Al llarg del desenvolupament m'he trobat amb situacions que m'han fet créixer tant professionalment com acadèmicament. **Integrar tecnologies d'IA d'última generació en un entorn productiu real**

ha estat una experiència única que m'ha permès aplicar molts dels coneixements adquirits durant la carrera i alhora descobrir-ne de nous.

Una de les primeres lliçons ha estat la **importància de l'anàlisi i planificació** quan es treballa amb sistemes complexos. En abordar el problema, vaig comprendre que no n'hi havia prou amb coneixements teòrics sobre IA generativa: calia dissenyar una arquitectura sòlida per orquestrar diversos serveis, assegurar la coherència de les dades i minimitzar l'impacte en els sistemes existents. He après a dividir el problema en subproblemes manejables (recollir i processar dades, generar resums, integrar-los a l'aplicació...) i a definir interfícies clares entre els components. Estic satisfet d'haver adoptat un enfocament iteratiu i **àgil**: vaig organitzar el projecte en *sprints* amb objectius parcials, fet que m'ha permès anar adaptant la solució sobre la marxa i reaccionar davant els obstacles imprevistos (com ajustos en APIs externes o canvis en requeriments). Aquesta metodologia de treball m'ha ensenyat que, en projectes innovadors, **la flexibilitat i la capacitat d'aprenentatge continu són tan importants com la planificació inicial**.

Pel que fa als **conceptes tècnics**, aquest TFG ha estat especialment enriquidor. *No només he consolidat coneixements* de desenvolupament backend en Python i Django (treballant amb microserveis, bases de dades SQL i NoSQL, i serveis d'AWS com S3 i SNS), sinó que també *m'he endinsat en el món dels LLMs*. Abans de començar, tenia nocions bàsiques sobre APIs d'OpenAI, però ara puc dir que **domino la integració de serveis d'IA generativa**: he utilitzat l'SDK oficial d'OpenAI i la llibreria LangChain per construir *prompts* complexos i rebre respostes estructurades, i he après a controlar-ne els paràmetres (temperatura, model, etc.) per obtenir els resultats desitjats. Igualment, l'ús de Twelve Labs ha estat una gran novetat per a mi; m'ha fet veure la importància de comptar amb eines especialitzades per a tipus de dades no textuais. Per exemple, he entès de primera mà que **analitzar un vídeo no és simplement processar imatges seqüencials**, sinó que requereix tractar un flux temporal i fins i tot extreure'n l'àudio per entendre el context complet. Integrar aquesta tecnologia em va obligar a pensar en la simultaneïtat de processos (enviar el vídeo a analitzar asíncronament i recuperar-ne els resultats més tard).

Un aspecte que vull remarcar en la meva conclusió personal és la **gestió de la incertesa i els riscos**. Treballar amb IA generativa comporta un grau d'imprevisibilitat: els models poden generar errors, interpretar malament alguna dada o fins i tot "al·lucinar" contingut que no existeix. Durant el projecte, he après a mitigar aquests riscos de diverses maneres. Per exemple, vaig decidir que els resums dels documents es guardarien a la base de dades un cop generats, de forma que si cal repetir el resum del sinistre no cal tornar a invocar l'API d'OpenAI per a cada document (reduint tant costos com la probabilitat d'errors repetitius).

També he implementat mecanismes de validació bàsics de les respostes (com ara assegurar que el JSON retornat pels models té el format esperat).

Des d'un punt de vista més personal, participar en un projecte real d'empresa amb el suport de Cleverea ha estat molt motivador. M'ha donat l'oportunitat de **treballar colze a colze amb professionals** del sector, rebre formacions valuoses (per exemple, sobre l'ús eficient de l'API d'OpenAI o arquitectura hexagonal) i, sobretot, veure l'impacte directe de la meva feina. **He crescut en confiança** com a enginyer, ara em veig capaç d'afrontar projectes complexos, investigar noves tecnologies pel meu compte i integrar-les eficientment.

En conclusió, aquest TFG ha estat **una experiència transformadora a nivell professional i personal**. He consolidat habilitats de programació i arquitectura de sistemes, he adquirit nous coneixements en IA i tractament de llenguatge natural, i he desenvolupat competències com la gestió del temps, la coordinació amb *stakeholders* i la presa de decisions tècniques sota pressió. A més, he pogut aportar valor real a una empresa, fet que em reconforta i em confirma que vull continuar orientant la meva carrera cap aquest sector. Tot i haver sigut un camí costós i amb moltes dificultats, cada obstacle superat (des d'un bug en la integració fins a un ajust fi en un *prompt*) ha reforçat les meves habilitats com a enginyer de software. Em sento orgullós de la feina realitzada i il·lusionat pel que vindrà, sabent que el que he après en aquest projecte serà una base sòlida per als reptes futurs que encari com a enginyer informàtic.

11.3. Grau d'assoliment dels objectius

Els objectius que s'havien definit i proposat pel projecte suposaven un cran repte tècnic que he afrontat durant aquest últim temps. Un cop finalitzat el treball, puc afirmar que **la gran majoria d'objectius s'han assolit satisfactòriament**, tal com es detalla a continuació:

- **Millora de la gestió de sinistres mitjançant resums generats per IA:** Assolit completament. El sistema desenvolupat genera un resum complet per a cada sinistre de forma automatitzada. S'ha integrat amb èxit en el Back-Office de Cleverea, i els agents ja poden obtindre aquesta visió global del cas amb un sol clic. Les proves d'usuari i els comentaris dels gestors de sinistres confirmen que el resum facilita la comprensió del cas i agilitza la presa de decisions, complint així l'objectiu principal de millorar la gestió interna dels sinistres.
- **Integració de múltiples fonts d'informació (operativa, comunicacions, documents):** Assolit. El projecte requeria unificar dades de diferents sistemes i formats, i així s'ha fet. La solució orquestra correctament la recuperació de dades de

la base de dades operativa (PostgreSQL), de la base de dades de comunicacions (DynamoDB per a trucades i emails) i dels fitxers (Amazon S3 per a imatges, PDF i vídeos). Totes aquestes dades s'integren en el procés de generació del resum. Aquest objectiu era crític per assegurar que el resum fos complet i contextualitzat.

- **Ús efectiu de tecnologies d'IA generativa (LangChain, OpenAI, TwelveLabs):** Assolit. Un objectiu important era aprofitar eines d'IA avançades en el projecte. He aconseguit incorporar-les de manera eficient: LangChain ha servit com a capa d'abstracció per integrar el model generatiu GPT-4 d'OpenAI en la lògica del projecte, facilitant la creació de prompts i la gestió de la conversa amb el model. A més, per cobrir la part de vídeos, s'ha integrat l'API de Twelve Labs, la qual cosa ha permès analitzar vídeos de sinistres i obtenir-ne descripcions estructurades. La combinació d'aquests serveis ha funcionat segons el previst, i la comunicació entre el nostre sistema i les plataformes d'IA és estable i fiable. Aquest objectiu també incloïa la necessitat de **controlar costos i latència** en l'ús de serveis d'IA de pagament, i efectivament s'han implementat mecanismes de monitoratge i limitació de peticions per mantenir la despesa controlada.
- **Disponibilitat immediata del resum en l'entorn de treball dels gestors:** Assolit. S'ha complert l'objectiu d'integració completa al Back-Office: l'endpoint `GET /claims/{claim_id}/summary` retorna el resum i aquest es mostra automàticament a la interfície d'usuari del gestor. Les proves d'acceptació han demostrat que el resum es visualitza correctament, respectant permisos i sense errors de càrrega. A més, la major part de les peticions es responen dins dels temps marcats com a òptims (el 95% en menys de 28 segons, i totes en menys de 35 segons, segons les mesures realitzades), la qual cosa satisfà els requeriments de *temps real* definits al projecte.
- **Qualitat i utilitat dels resums generats:** Assolit. Es pretenia que els resums fossin útils i reflectissin l'essencial de cada sinistre sense informació errònia. Aquest objectiu es considera aconseguit perquè, tal com s'ha mencionat, en una avaluació interna 19 de 20 resums van ser validats positivament per l'equip de sinistres. Els resums inclouen seccions clau (estat del sinistre, accions pendents, documents analitzats, etc.) i han estat generats en un llenguatge entenedor (castellà, ja que és l'idioma de treball principal a l'empresa). Les enquestes de satisfacció també han reflectit que més del 90% dels usuaris donen als resums una puntuació d'utilitat alta (4 o 5 sobre 5), superant així el criteri d'èxit establert inicialment.

- **Millora de l'eficiència operativa i satisfacció del client:** Assolit indirectament. Tot i que és difícil atribuir quantitativament l'impacte només a aquesta eina, els indicadors recollits *post-desplegament* mostren tendències positives. S'ha observat que els agents gestionen un nombre superior de sinistres al dia i que el temps de resposta al client (des que aquest notifica el sinistre fins que rep una resposta o resolució) s'ha reduït de mitjana. Aquests resultats apunten a que l'objectiu de millorar l'eficiència s'està complint. A nivell de satisfacció client, encara és aviat per mesurar-ho amb dades sòlides, però internament es confia que una gestió més àgil repercutirà en valoracions més altes per part dels assegurats.
- **Generació d'una cronologia del sinistre (objectiu opcional):** No assolit. Aquest era un objectiu addicional que es va plantejar com a millora extra: crear una cronologia dels esdeveniments del sinistre. Degut al temps limitat i a la prioritització de funcionalitats crítiques, finalment **no s'ha implementat la cronologia** dins d'aquest projecte. Tot i haver explorat la idea i reconèixer el seu potencial (una cronologia podria aportar una comprensió ràpida de la seqüència d'esdeveniments i fins i tot permetre predir següents passos), es va decidir centrar els esforços en assegurar la qualitat del resum principal. Per tant, aquest objectiu queda com a línia futura de treball i no afecta l'assoliment dels objectius nuclears del TFG.

En resum, el grau d'assoliment dels objectius del projecte és positiu. Totes les fites principals s'han complert i validat, i només queda pendent alguna funcionalitat complementària no essencial.

11.3.1. Assoliment de les competències tècniques

En aquesta secció es busca contrastar com s'han assolit, dins del projecte, les competències tècniques definides a 3.2. Requisits.

- **CES1.1: Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics.** La solució desenvolupada impacta directament sobre la gestió de sinistres d'una empresa asseguradora real (Cleverea), fet que implica requisits elevats de fiabilitat i robustesa. S'han seguit processos rigorosos de desenvolupament, proves i integració per garantir l'estabilitat d'un sistema real i crític.
- **CES1.2: Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles.** El sistema s'ha integrat de forma natural dins l'arquitectura distribuïda existent a Cleverea, utilitzant les seves llibreries compartides, protocols de comunicació, mecanismes de desplegament i tecnologies.

S'han seguit els estàndards interns d'arquitectura per garantir compatibilitat i mantenibilitat.

- **CES1.3: Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar.** El projecte ha implicat una anàlisi contínua dels riscos tant tècnics com operatius. Per exemple, s'ha previst la limitació del nombre d'anàlisis de documents per evitar un ús excessiu de recursos externs i costos inesperats. També s'han definit mecanismes de monitoratge i notificació automàtica davant d'errors en la generació de resums, permetent actuar de forma proactiva. Així mateix, s'han gestionat riscos associats a la privacitat de dades i a la dependència de serveis externs, assegurant un comportament controlat i monitoritzable del sistema en producció.
- **CES1.7: Controlar la qualitat i dissenyar proves en la producció de software.** S'han implementat proves unitàries i d'integració per a les funcionalitats crítiques. A més, la solució inclou mecanismes de monitoratge i registre d'errors, i es valida l'output generat per garantir que compleix amb els requisits esperats.
- **CES2.2: Dissenyar solucions apropiades en un o més dominis d'aplicació, utilitzant mètodes d'enginyeria del software que integrin aspectes ètics, socials, legals i econòmics.** El sistema s'ha desenvolupat seguint bones pràctiques d'enginyeria del software, aplicant criteris d'eficiència i sostenibilitat econòmica (reducció de costos operatius i ús intel·ligent dels recursos). També s'ha considerat l'impacte en l'usuari final, millorant la seva experiència, i s'ha vetllat pel compliment normatiu en protecció de dades i seguretat.
- **CES1.9: Demostrar comprensió en la gestió i govern dels sistemes software.** Un dels majors reptes d'aquest projecte. S'ha treballat sota un model estructurat de desenvolupament amb una clara separació de responsabilitats entre capes i serveis, documentant els requisits, el comportament del sistema i la seva arquitectura. Això ha permès la integració adequada dins un sistema existent i complex com el de Cleverea,

11.4. Línies futures de treball

Tot i haver completat amb èxit el projecte, han sorgit diverses **oportunitats de millora i ampliació** que podrien abordar-se en treballs futurs dins de Cleverea. A continuació detallo les línies futures de treball més rellevants identificades:

- **Implementació de la cronologia del sinistre:** Com s'ha comentat, una ampliació natural seria desenvolupar un mòdul que generi la cronologia d'esdeveniments del sinistre. Aquesta cronologia podria presentar de manera ordenada els esdeveniments rellevants (data de notificació, obertura del cas, trucades rebudes, enviament de documents, inspeccions, tancament, etc.) i fins i tot **visualitzar-se en una línia temporal interactiva** al Back-Office. A més de ser útil per als gestors, disposar de moltes cronologies de sinistres resolts podria obrir la porta a **sistemes predictius**: entrenant models amb aquests històrics, es podria suggerir als agents quin hauria de ser el proper pas en casos similars (per exemple, “encarregar peritatge” després de certa seqüència de fets). Aquesta línia de treball permetria apropar-nos encara més a l'automatització en la gestió de sinistres.
- **Optimització de costos i rendiment amb models a mida:** Actualment s'utilitzen serveis externs (OpenAI, Twelve Labs) que tenen un cost associat per ús. En un futur, Cleverea podria considerar **entrenar un model de llenguatge propi o ajustar un de preexistent** amb dades de sinistres de la companyia, de manera que el resum es pogués generar internament sense trucades a serveis de tercers. Aquesta opció podria reduir costos a llarg termini i eliminar preocupacions de privacitat (les dades sensibles no sortien dels sistemes de Cleverea). També es podria explorar l'ús de models de codi obert optimitzats que es puguin desplegar en servidors de l'empresa. Paral·lelament, seria beneficiós refinar els prompts i l'estructura de sortida per reduir la mida de la resposta i el temps de processament, garantint que cada resum només contingui la informació essencial i no excedeixi la longitud necessària.

En definitiva, el treball realitzat obre les portes a altres projectes que poden complementar i potenciar la feina duta a terme. Amb la base implementada en aquest TFG, Cleverea disposa ja d'un fonament sòlid per explorar millores i seguir innovant en la gestió intel·ligent de sinistres.

12. Referències

- [1] Ibm. (2025, 24 febrer). *AI in insurance. IBM think.* <https://www.ibm.com/think/topics/ai-in-insurance>
- [2] Insurtech Insights. (2025, 24 febrer). *15 Insurtechs Harnessing AI to Transform the Insurance Landscape.*<https://www.insurtechinsights.com/15-insurtechs-harnessing-ai-to-transform-the-insurance-landscape/>
- [3] Khropatyy, P. (2025, 24 febrer). *AI in Insurtech – Boost Your Customer Experience.* Intellias. <https://intellias.com/insurtech-ai/>
- [4] Ibm. (2025, 24 febrer). *Fine tuning. IBM Think.* <https://www.ibm.com/es-es/think/topics/fine-tuning>
- [5] *fine Engineering Guide – Nextra.* (2025, 24 febrer). <https://www.promptingguide.ai/>
- [6] Spangenberg, M., Spangenberg, M., & Netguru. (2025, 24 febrer). *How leading insurtech companies make use of AI solutions such as: Fraud Detection, Hyper-Personalization, and Underwriting.* Netguru. <https://www.netguru.com/blog/ai-in-insurtech>
- [7] Latin Counsel. (2025, 24 febrer). *El Impacto de la Inteligencia Artificial en el Sector Asegurador: Análisis de la Cátedra Pérez-Llorca/IE.* https://www.latincounsel.com/?Noticias=Inteligencia_Artificia_Sector_Asegurador_Analisis_Catedra_PerezLlorcaIE
- [8] Marin, J. (2025, 24 febrer). *La intel·ligència artificial en el 2025: tendències, evolució i impacte empresarial.* VIA Empresa. https://www.viaempresa.cat/economia/inteligencia-artificial-impacte-empresarial_2207756_102.html
- [9] EY (2025, 24 febrer) *Cómo revolucionar la cadena de valor de los seguros con IA generativa.* https://www.ey.com/es_ce/insights/insurance/how-to-revolutionize-the-insurance-value-chain-with-generative-ai
- [10] Mapfre, R. (2025, 24 febrer). *El poder de la inteligencia artificial en la valoración de daños de vehículos: un detector en tiempo real de arañazos y golpes.* MAPFRE. <https://www.mapfre.com/actualidad/innovacion/inteligencia-artificial-valoracion-vehiculos/>
- [11] Learning Heroes (2025, 24 febrer). *¿Qué es la Inteligencia Artificial de Allianz y Cómo Funciona?* <https://www.learningheroes.com/aprende-inteligencia-artificial/que-es-la-inteligencia-artificial-de-allianz-y-como-funciona>

- [12] Devoteam. (2025, 24 febrer). *Generative AI in Insurance: Lemonade Case Study* / Devoteam. <https://www.devoteam.com/expert-view/innovation-in-insurance/>
- [13] KPMG (2025, 25 febrer). Avance de la IA en la industria de seguros. <https://kpmg.com/mx/es/home/tendencias/2024/12/avance-de-la-ia-en-la-industria-de-seguros.html>
- [14] Marcos, M. (2025, 25 febrer). *Para el 85% de los directivos de aseguradoras, la utilización de IA dará una ventaja competitiva sobre quienes no la implementen*. Panorama Asegurador. <https://panoramaasegurador.com/para-el-85-de-los-directivos-de-aseguradoras-la-utilizacion-de-ia-dara-una-ventaja-competitiva-sobre-quienes-no-la-implementen/>
- [15] *Waterfall vs metodología Agile* / Miro. (2025, 25 febrer). <https://miro.com/es/agile/waterfall-vs-metodologia-agile/>
- [16] Universidad Europea. (2025, 25 febrer). *Agile vs. Waterfall: qué diferencias hay entre ambas metodologías*. <https://universidadeuropea.com/blog/agile-vs-waterfall/>
- [17] Lucidchart. (2025, 25 febrer). *Método híbrido Agile-Waterfall: ¿Es adecuado para tu equipo?* <https://www.lucidchart.com/blog/es/hibrido-metodologia-agile-waterfall-para-tu-equipo>
- [18] Raeburn, A.(2025, 25 febrer). What is Extreme Programming (XP)? [2025] • Asana. Asana. <https://asana.com/resources/extreme-programming-xp>
- [19] Workspace, G. (2025, 25 febrer). *Google Workspace: Secure online productivity & collaboration tools*. Google Workspace. <https://workspace.google.com/>
- [20] Workspace, G. (2025, 25 febrer). *Google Workspace: Secure online productivity & collaboration tools*. Google Workspace. <https://workspace.google.com/>
- [21] *Your connected workspace for wiki, docs & projects* / Notion. (2025, 25 febrer). Notion. <https://www.notion.com/>
- [22] Workspace, G. (2025, 25 febrer). *Google Meet: Online Web and Video Conferencing Calls* / Google Workspace. Google Workspace. <https://meet.google.com/landing>
- [23] Contributors to Wikimedia projects. (2025, 25 febrer). *Google Calendar*. Viquipèdia, L'enciclopèdia Lliure. https://ca.wikipedia.org/wiki/Google_Calendar
- [24] Slack. (2025, 25 febrer). *Gestión del trabajo mediante IA y herramientas de productividad*. Slack. <https://slack.com/intl/es-es>
- [25] Git. (2025, 25 febrer). <https://git-scm.com/>

- [26] CircleCI. (2025, 25 febrer). *CircleCI*. <https://circleci.com/>
- [27] Ratliff, S. (2025, 25 febrer). *Docker: Accelerated Container Application Development*. Docker. <https://www.docker.com/>
- [28] AWS / *Cloud Computing - Servicios de informática en la nube*. (2025, 25 febrer). Amazon Web Services, Inc. <https://aws.amazon.com/es/>
- [29] *Visual Studio Code - Code editing. Redefined*. (2025, 3 març). <https://code.visualstudio.com/>
- [30] Glassdor. (2025, 10 març). *Sueldos de la empresa*. <https://www.glassdoor.es/Sueldos/index.htm>
- [31] Talend (2025, 18 març). *A complete, scalable data management solution*. Talend - A Leader In Data Integration & Data Integrity. <https://www.talend.com/>
- [32] Apache NiFi (2025, 18 març). Apache NiFi. <https://nifi.apache.org/>
- [33] MuleSoft (2025, 18 març). *Integración y automatización en la era de la IA*. <https://www.mulesoft.com/es>
- [34] Zapier (2025, 18 març). *Alcanza el éxito con la automatización*. <https://zapier.com/l/home-spanish>
- [35] OpenAI (2025, 18 març). *Obtén respuestas, Encuentra inspiración. Sé más eficiente*. <https://openai.com/es-ES/chatgpt/overview/>
- [36] SummarizeBot (2025, 18 març). *Extractive & Generative AI, Semantic Search, and Intelligent AI Agents*. <https://www.summarizebot.com/>
- [37] Power BI (2025, 18 març) *Visualización de datos*. <https://www.microsoft.com/es-es/power-platform/products/power-bi>
- [38] OpenAI (2025, 3 maig). *OpenAI developer platform*. <https://platform.openai.com/docs/overview>
- [39] TwelveLabs. (2025, 3 maig). *TwelveLabs*. <https://www.twelvelabs.io/>
- [40] OpenAI (2025, 3 maig) Set up your development environment to use the OpenAI API with an SDK in your preferred language. <https://platform.openai.com/docs/libraries>

- [41] TwelveLabs. (2025, 3 maig). *Python SDK* / TwelveLabs. <https://docs.twelvelabs.io/v1.3/sdk-reference/python>
- [42] LangChain. (2025, 3 maig). *LangChain* <https://www.langchain.com/>
- [43]. Amazon Web Services (2025, 3 maig). *¿Qué es el SDK? - Explicación del SDK - AWS*. <https://aws.amazon.com/es/what-is/sdk/>
- [44] PostgreSQL. (2025, 14 maig). *PostgreSQL*. <https://www.postgresql.org/>
- [45] Amazon Web Services. (2025, 14 maig). *AWS / Servicio de base de datos gestionada NoSQL (DynamoDB)*. <https://aws.amazon.com/es/dynamodb>
- [46] Amazon Web Services. (2025, 14 maig). *Introducción a Amazon S3*. <https://aws.amazon.com/es/pm/serv-s3>
- [47] Viquipèdia (2025, 14 maig). *Base de dades relacional* https://ca.wikipedia.org/wiki/Base_de_dades_relacional
- [48] Viquipèdia (2025, 14 maig). *NoSQL* <https://ca.wikipedia.org/wiki/NoSQL>
- [49] Viquipèdia (2025, 14 maig). *Microservices* <https://en.wikipedia.org/wiki/Microservices>
- [50] Viquipèdia (2025, 14 maig). *Arquitectura hexagonal (programari)*. [https://ca.wikipedia.org/wiki/Arquitectura_hexagonal_\(programari\)](https://ca.wikipedia.org/wiki/Arquitectura_hexagonal_(programari))
- [51] Viquipèdia (2025, 28 maig). *Webhook*. <https://ca.wikipedia.org/wiki/Webhook>
- [52] ICM (2025, 28 maig). *¿Qué son los Endpoints y para qué sirven?* <https://www.icm.es/2021/06/15/que-son-endpoints/>
- [53] Amazon Web Services. (2025, 28 maig). *AWS Secrets Manager*. <https://aws.amazon.com/es/secrets-manager/>
- [54] Wikipedia (2025, 28 maig). *Dependency injection*. https://en.wikipedia.org/wiki/Dependency_injection
- [55] Wikipedia (2025, 28 maig). *Mock object*. https://en.wikipedia.org/wiki/Mock_object

- [56] Lang Chain (2025, 28 maig). *PydanticOutputParser*. https://python.langchain.com/api_reference/core/output_parsers/langchain_core.output_parsers.pydantic.PydanticOutputParser.html
- [57] Viquipèdia (2025, 28 maig). *Base64*. <https://ca.wikipedia.org/wiki/Base64>
- [58] UAO (2025, 13 juny). *Como razona una IA*. <https://virtual.uao.edu.co/blog/como-razona-una-ia>
- [59] OpenAI (2025, 13 juny). *Tokenizer*. <https://platform.openai.com/tokenizer>
- [60] Elastic (2025, 14 juny). *Kibana*. <https://www.elastic.co/es/kibana>
- [61] Celery (2025, 15 juny). *Celery – Distributed Task Queue*. <https://docs.celeryq.dev>
- [62] Redis (2025, 15 juny). *Redis*. <https://redis.io>
- [63] Adecco (2025, 17 juny). *¿Qué es la técnica de chunking?*. <https://www.adecco.es/insights/trabajo-temporal/que-es-la-tecnica-de-fragmentacion-o-tecnica-de-chunking>

13. Annexos

Annex A - Extractes de codi

A-1. GetClaimSummaryServices

```
from dependency_injector import containers, providers

from claims.claim.application.services.get_claim_summary import
GetClaimSummaryService

from
claims.claim.dependency_injection.domain.services.get_claim_calls
import GetClaimCallsServices

from
claims.claim.dependency_injection.domain.services.get_claim_emails
import (
    GetClaimEmailsServices,
)

from
claims.claim.dependency_injection.infrastructure.persistence.claim_b
o import (
    ClaimBOPersistences,
)

from
claims.claim.dependency_injection.infrastructure.services.llm.claim_
summary.generate_claim_summary import (
    GenerateClaimSummaryServices,
)

from
claims.document.dependency_injection.domain.services.search_all_rela
ted_documents_of_a_claim import (
    SearchAllRelatedDocumentsOfAClaimServices,
)

class GetClaimSummaryServices(containers.DeclarativeContainer):

    default = providers.Singleton(
        GetClaimSummaryService,

generate_claim_summary_service=GenerateClaimSummaryServices.default,
```

```

        claim_persistence_service=ClaimBOPersistences.django,
        get_claim_calls_service=GetClaimCallsServices.default,
        get_claim_emails_service=GetClaimEmailsServices.default,

search_all_related_documents_of_a_claim_service=SearchAllRelatedDocu
mentsOfAClaimServices.cleverea,
    )

```

A-2. TestGetClaimSummaryService

```

from unittest.mock import MagicMock

from claims.claim.application.services.get_claim_summary import
GetClaimSummaryService

from claims.claim.domain.bo.claim_summary_bo import ClaimSummaryBO
from claims.claim.domain.dto.summary_document_dto import
SummaryDocumentDTO

class TestGetClaimSummaryService:
    def setup_method(self):
        self.claim_persistence_service = MagicMock()
        self.get_claim_calls_service = MagicMock()
        self.get_claim_emails_service = MagicMock()
        self.search_all_related_documents_service = MagicMock()
        self.generate_claim_summary_service = MagicMock()

        self.service = GetClaimSummaryService(

claim_persistence_service=self.claim_persistence_service,
            get_claim_calls_service=self.get_claim_calls_service,
            get_claim_emails_service=self.get_claim_emails_service,

search_all_related_documents_of_a_claim_service=self.search_all_rela
ted_documents_service,

generate_claim_summary_service=self.generate_claim_summary_service,
        )

    def test_success(self, db_claim_bo_factory_base):

```

```

    """
    Test successful generation of claim summary.
    """

    # Setup return values
    self.claim_persistence_service.get_by_id.return_value =
db_claim_bo_factory_base
    mock_calls = MagicMock()
    mock_emails = MagicMock()
    mock_documents = MagicMock()
    mock_summary = MagicMock(spec=ClaimSummaryBO)

    self.get_claim_calls_service.return_value = mock_calls
    self.get_claim_emails_service.return_value = mock_emails
    self.search_all_related_documents_service.return_value =
mock_documents
    self.generate_claim_summary_service.return_value =
mock_summary

    # Execute
    result = self.service(claim_id=db_claim_bo_factory_base.id)

    # Validate result
    assert result == mock_summary

    # Assert correct claim fetched

self.claim_persistence_service.get_by_id.assert_called_once_with(
    db_claim_bo_factory_base.id
)

    assert
self.get_claim_calls_service.call_args.kwargs["claim_bo"] ==
db_claim_bo_factory_base
    assert (

self.get_claim_emails_service.call_args.kwargs["claim_bo"] ==
db_claim_bo_factory_base
)
    assert (

```

```

self.search_all_related_documents_service.call_args.kwargs["claim_bo
"]

        == db_claim_bo_factory_base
    )

    # Validate generate_claim_summary_service got all the
correct objects
    called_kwargs =
self.generate_claim_summary_service.call_args.kwargs
    assert called_kwargs["claim_bo"] == db_claim_bo_factory_base
    assert called_kwargs["claim_calls"] == mock_calls
    assert called_kwargs["claim_emails"] == mock_emails

    summary_docs = called_kwargs["claim_documents"]
    assert isinstance(summary_docs, list)
    assert all(isinstance(doc, SummaryDocumentDTO) for doc
in summary_docs)

```

A-3. GetClaimSummaryViewControllerV1

```

from typing import List

from claims.claim.application.services.get_claim_summary import
GetClaimSummaryService

from
claims.claim.application.view_controllers.v1.get_claim_summary.v1.dt
o.output_dto import (
    GetClaimSummaryOutputDTOV1,
)

from
claims.claim.application.view_controllers.v1.get_claim_summary.v1.ou
tput_mapping_service import (
    GetClaimSummaryOutputMappingServiceV1,
)

class GetClaimSummaryViewControllerV1:
    """
    Retrieve the claim's summary.

```

```

"""

def _init_(
    self,
    output_mapping_service,
    get_claim_summary_service,
):
    """
    Class initialization.
    """

    self.output_mapping_service:
    GetClaimSummaryOutputMappingServiceV1 = output_mapping_service
    self.get_claim_summary_service: GetClaimSummaryService =
    get_claim_summary_service

    def _call_(self, *, claim_id: int) ->
    List[GetClaimSummaryOutputDTOV1]:
        """
        Retrieve the claim's summary.
        """

        claim_summary = self.get_claim_summary_service(
            claim_id=claim_id,
        )

        mapped_summary =
        self.output_mapping_service(claim_summary=claim_summary)

        return mapped_summary

```

A-4. GetClaimSummaryService

```

from typing import List

from claims.claim.application.services.get_claim_summary import
GetClaimSummaryService

from
claims.claim.application.view_controllers.v1.get_claim_summary.v1.dto.output_dto import (
    GetClaimSummaryOutputDTOV1,
)

```

```

from
claims.claim.application.view_controllers.v1.get_claim_summary.v1.out
put_mapping_service import (
    GetClaimSummaryOutputMappingServiceV1,
)

class GetClaimSummaryViewControllerV1:
    """
    Retrieve the claim's summary.
    """

    def __init__(
        self,
        output_mapping_service,
        get_claim_summary_service,
    ):
        """
        Class initialization.
        """
        self.output_mapping_service:
GetClaimSummaryOutputMappingServiceV1 = output_mapping_service
        self.get_claim_summary_service: GetClaimSummaryService =
get_claim_summary_service

    def __call__(self, *, claim_id: int) ->
List[GetClaimSummaryOutputDTOV1]:
        """
        Retrieve the claim's summary.
        """
        claim_summary = self.get_claim_summary_service(
            claim_id=claim_id,
        )

        mapped_summary =
self.output_mapping_service(claim_summary=claim_summary)

        return mapped_summary

```

A-5. GetClaimCallsService

```
from datetime import datetime, timedelta
from typing import List, Tuple

from claims.claim.domain.bo.claim_bo import ClaimBO
from claims.claim.domain.bo.claim_call_bo import ClaimCallBO
from claims.claim.domain.enums import ClaimStatusType
from claims.claim.domain.services.external_claim_calls.interface
import ExternalClaimCallsInterface
from claims.core.domain.services.get_holder_service import
GetHolderService

class GetClaimCallsService:
    """
    Manages the retrieval of the claim calls using the info in
    DynamoDB.
    """

    def __init__(self, external_claim_calls_service,
get_holder_service):
        """
        Class initialization.
        """
        self.external_claim_calls_service:
ExternalClaimCallsInterface = (
            external_claim_calls_service
        )
        self.get_holder_service: GetHolderService =
get_holder_service

    def _call_(self, claim_bo: ClaimBO) -> List[ClaimCallBO]:
        """
        Retrieve the list of calls related to a claim using the
        document number and timestamp.
        """
        if not claim_bo.created_at:
            return []
```

```

        holder_document_number: str = self.get_holder_service(
            vertical=claim_bo.vertical,
            policy_number=claim_bo.policy_number
        ).document_number

        if not holder_document_number:
            return []

        start_date, end_date =
self._get_start_and_end_dates(claim_bo=claim_bo)

        calls =
self.external_claim_calls_service.get_claim_analyzed_calls_by_document_number(
            document_number=holder_document_number,
            start_date=start_date,
            end_date=end_date,
        )

        return calls

    def _get_start_and_end_dates(self, claim_bo: ClaimBO) ->
Tuple[datetime, datetime]:
        """
        Get the start and end dates for the claim calls.

        :start_date: is 24 hours before the fnol created_at date if
it exists,
        otherwise it is 24 hours before the claim created_at date.

        :end_date: is the last status valid from date if the claim
is closed,
        discarded, or rejected. Otherwise, it is the current date.
        """
        start_date = claim_bo.fnol.created_at if claim_bo.fnol else
claim_bo.created_at
        start_date = start_date - timedelta(hours=24)
        end_date = (
            claim_bo.last_status.valid_from
            if claim_bo.last_status.status

```



```

        in (ClaimStatusType.CLOSED, ClaimStatusType.DISCARDED,
ClaimStatusType.REJECTED)
        else datetime.now()
    )
    return start_date, end_date

```

A-6. GetClaimEmailsService

```

from datetime import datetime, timedelta
from typing import List, Tuple

from claims.claim.domain.bo.claim_bo import ClaimBO
from claims.claim.domain.bo.claim_email_bo import ClaimEmailBO
from claims.claim.domain.enums import ClaimStatusType
from claims.claim.domain.services.external_claim_emails.interface
import (
    ExternalClaimEmailsInterface,
)

from claims.core.domain.services.get_holder_service import
GetHolderService

class GetClaimEmailsService:
    """
    Manages the retrieval of the claim emails using the info in
    DynamoDB.
    """

    def __init__(self, external_claim_emails_service,
get_holder_service):
        self.external_claim_emails_service:
ExternalClaimEmailsInterface = (
            external_claim_emails_service
        )

        self.get_holder_service: GetHolderService =
get_holder_service

    def __call__(self, claim_bo: ClaimBO) -> List[ClaimEmailBO]:
        """

```

Retrieve the list of emails related to a claim using the handler email and timestamp.

```
"""
    if not claim_bo.created_at:
        return []

    holder_email: str = self.get_holder_service(
        vertical=claim_bo.vertical,
        policy_number=claim_bo.policy_number
    ).email
    if not holder_email:
        return []

    start_date, end_date =
self._get_start_and_end_dates(claim_bo)

    emails_by_receiver =
self.external_claim_emails_service.get_claim_receiver_emails(
        email=holder_email,
        start_date=start_date,
        end_date=end_date,
    )

    emails_by_sender =
self.external_claim_emails_service.get_claim_sender_emails(
        email=holder_email,
        start_date=start_date,
        end_date=end_date,
    )

    combined_emails = emails_by_receiver + emails_by_sender

    return combined_emails

def _get_start_and_end_dates(self, claim_bo: ClaimBO) ->
Tuple[datetime, datetime]:
    start_date = claim_bo.fnol.created_at if claim_bo.fnol else
claim_bo.created_at
    start_date -= timedelta(hours=24)
    end_date = (
```

```

        claim_bo.last_status.valid_from
        if claim_bo.last_status.status
            in (ClaimStatusType.CLOSED, ClaimStatusType.DISCARDED,
ClaimStatusType.REJECTED)
            else datetime.now()
    )
    return start_date, end_date

```

A-7. DynamoDBClaimCallsService

```

from datetime import datetime
from typing import Dict, List

from cleverea_aws_integrations.dynamodb.dynamodb_table_service
import DynamoDBTableService

from claims.claim.domain.bo.claim_call_bo import ClaimCallBO
from claims.claim.domain.services.external_claim_calls.interface
import ExternalClaimCallsInterface

class DynamoDBClaimCallsService(ExternalClaimCallsInterface):
    """
    Class containing the DynamoDB implementation of the contract
    when retrieving external calls.
    """

    def __init__(self, dynamodb_table_service):
        """
        Class initialization.
        """
        self.dynamodb_table_service: DynamoDBTableService =
dynamodb_table_service

    def get_claim_analyzed_calls_by_document_number(
        self, document_number: str, start_date: datetime, end_date:
datetime
    ) -> List[ClaimCallBO]:
        """

```

Retrieve the analyzed calls given a document number within a date range.

```
"""
    # Get contact IDs by document number
    contact_ids =
self._get_contact_ids_by_document_number(document_number)
    if not contact_ids:
        return []

    # Filter contact IDs with matching calls
    filtered_contact_ids =
self._filter_contact_ids_with_matching_calls(
        contact_ids=contact_ids, start_date=start_date,
end_date=end_date
    )
    if not filtered_contact_ids:
        return []

    # Get calls by contact IDs
    calls =
self._get_calls_by_contact_ids(contact_ids=filtered_contact_ids)

    mapped_calls = [
        ClaimCallBO(
            created_at=datetime.fromisoformat(item["Timestamp"] ["S"]),
            summary=item["summary"] ["S"],
            next_actions=[
                action["S"]
                for action in item.get("next_actions",
{})).get("L", [])
                if "S" in action
            ],
        )
        for item in calls
    ]

    return mapped_calls
```

```

    def _get_contact_ids_by_document_number(self, document_number:
str) -> List[str]:
        contacts =
self.dynamodb_table_service.query_table_with_gsi_by_key_value(
            table_name="ConnectContactIdUsersTable",
            index_name="DocumentNumberIndex",
            key="documentNumber",
            value_type="S",
            value=document_number,
        )
        return [contact.get("contactId", {}).get("S") for contact in
contacts]

    def _filter_contact_ids_with_matching_calls(
        self, contact_ids: List[str], start_date: datetime,
end_date: datetime
    ) -> List[str]:
        filtered_ids = []

        for contact_id in contact_ids:
            query_params = {
                "TableName": "connectAnsweredCallsData",
                "KeyConditionExpression": "contact_id = :contact_id
AND created_at BETWEEN :start_date AND :end_date",
                "FilterExpression": "queue_name IN (:q1, :q2)",
                "ExpressionAttributeValues": {
                    ":contact_id": {"S": contact_id},
                    ":q1": {"S": "Claims_Entrantes"},
                    ":q2": {"S": "Claims_Callback_01"},
                    ":start_date": {"S": start_date.isoformat()},
                    ":end_date": {"S": end_date.isoformat()}},
            },
        }

            results =
self.dynamodb_table_service.general_dynamodb_query(query_params=query
params)

            if results:
                filtered_ids.append(contact_id)

        return filtered_ids

```

```

    def _get_calls_by_contact_ids(self, contact_ids: List[str]) ->
    List[Dict[str, Dict[str, str]]]:

        user_calls = []
        for contact_id in contact_ids:
            user_calls.extend(

self.dynamodb_table_service.query_table_with_gsi_by_key_value(
            table_name="ConnectCallsAnalysis",
            index_name="ContactIdIndex",
            key="contact_id",
            value_type="S",
            value=contact_id,

        )
    )

    return user_calls

```

A-8. DynamoDBClaimEmailsService

```

from datetime import datetime
from typing import List

from cleverea_aws_integrations.dynamodb.dynamodb_table_service
import DynamoDBTableService

from claims.claim.domain.bo.claim_email_bo import ClaimEmailBO
from claims.claim.domain.services.external_claim_emails.interface
import (
    ExternalClaimEmailsInterface,
)

class DynamoDBClaimEmailsService(ExternalClaimEmailsInterface):
    """
    Class containing the DynamoDB implementation of the contract
    when retrieving external emails.
    """

```

```

def _init_(self, dynamodb_table_service):
    """
    Class initialization.
    """
    self.dynamodb_table_service: DynamoDBTableService =
dynamodb_table_service

def get_claim_receiver_emails(
    self,
    email: str,
    start_date: str,
    end_date: str,
) -> List[ClaimEmailBO]:
    """
    Retrieve the emails sent to a specific receiver within a
date range.
    """
    emails = self._get_emails_from_dynamo(
        index_name="ReceiverByDateIndex",
        key_name="receiver",
        email=email,
        start_date=start_date,
        end_date=end_date,
    )
    mapped_emails = self._map_emails(
        emails=emails,
    )
    return mapped_emails

def get_claim_sender_emails(
    self, email: str, start_date: str, end_date: str
) -> List[ClaimEmailBO]:
    """
    Retrieve the emails sent by a specific sender within a date
range.
    """
    emails = self._get_emails_from_dynamo(

```

```

        index_name="SenderByDateIndex",
        key_name="sender",
        email=email,
        start_date=start_date,
        end_date=end_date,
    )
    mapped_emails = self._map_emails(
        emails=emails,
    )
    return mapped_emails

def _map_emails(self, emails) -> List[ClaimEmailBO]:
    """
    Map the emails from DynamoDB to ClaimEmailBO.
    """
    return [
        ClaimEmailBO(
            sent_date=datetime.strptime(item["sent_date"]["S"],
"%Y-%m-%d %H:%M:%S"),
            summary=item["summary"]["S"],
        )
        for item in emails
    ]

def _get_emails_from_dynamo(
    self, index_name: str, key_name: str, email: str,
    start_date: datetime, end_date: datetime
) -> List[ClaimEmailBO]:
    """
    Retrieve emails from DynamoDB based on the provided
    parameters.
    Returns the sent_date and summary of the email.
    """
    categories = [
        "Actualización de estado de siniestro",
        "Peritaciones",
        "Alta nuevo siniestro",
        "Indemnizaciones",
    ]

```



```

        "Asistencia en carretera",
        "Lesiones",
        "Documentación",
        "Consulta sobre pagos",
        "Pago recibido",
        "Cobro no reconocido",
        "Envío de documentación",
        "Solicitud certificado siniestralidad",
        "Asistencia en Carretera",
        "Solicitud reembolso",
    ]

    category_keys = [f":cat{i}" for i in range(len(categories))]
    filter_expression = f"category IN ({', '
'.join(category_keys)})"
    expression_values = {
        f":{key_name}": {"S": email},
        ":start_date": {"S": start_date.isoformat()},
        ":end_date": {"S": end_date.isoformat()},
    }
    for i, category in enumerate(categories):
        expression_values[f":cat{i}"] = {"S": category}

    query_params = {
        "TableName": "AnalyzedHubspotEmails",
        "IndexName": index_name,
        "KeyConditionExpression": f"{key_name} = :{key_name} AND
sent_date BETWEEN :start_date AND :end_date",
        "FilterExpression": filter_expression,
        "ExpressionAttributeValues": expression_values,
    }

    emails =
self.dynamodb_table_service.general_dynamodb_query(query_params=query_params)

    return emails

```

A-9. LLMGenerateClaimSummaryService

```
import json
from typing import List

from langchain.output_parsers import PydanticOutputParser
from langchain_core.messages import HumanMessage, SystemMessage
from langchain_openai import ChatOpenAI

from claims.claim.domain.bo.claim_bo import ClaimBO
from claims.claim.domain.bo.claim_call_bo import ClaimCallBO
from claims.claim.domain.bo.claim_email_bo import ClaimEmailBO
from claims.claim.domain.bo.claim_summary_bo import ClaimSummaryBO,
ClaimSummarySectionBO
from claims.claim.domain.dto.summary_document_dto import
SummaryDocumentDTO
from claims.claim.domain.services.claim_summary.interface import
GenerateClaimSummaryInterface

from
claims.claim.infrastructure.services.llm.claim_summary.clean_claim_s
ummary_info import (
    CleanLLMClaimSummaryInfoService,
)

from
claims.claim.infrastructure.services.llm.claim_summary.dto.output
import ClaimSummaryResponse

from claims.claim.infrastructure.services.llm.claim_summary.prompts
import ClaimSummaryPrompt

from claims.claim.infrastructure.services.llm.exceptions import
LLMClaimSummaryGenerationFailed

class LLMGenerateClaimSummaryService(GenerateClaimSummaryInterface):
    """
    LLM service to generate claim's summary using LangChain.
    """

    def __init__(self, settings, clean_claim_summary_info_service):
        """
        Class initialization.
        """
```

```

        self.llm = ChatOpenAI(

openai_api_key=settings.CLAIMS_COMMENTS_SUMMARIES_OPENAI_API_KEY,
        model_name="gpt-4.1",
        temperature=0.6,
    )

    self.output_parser =
PydanticOutputParser(pydantic_object=ClaimSummaryResponse)

    self.clean_claim_summary_info_service:
CleanLLMClaimSummaryInfoService = (
        clean_claim_summary_info_service
    )

def _call_(
    self,
    *,
    claim_bo: ClaimBO,
    claim_calls: List[ClaimCallBO],
    claim_emails: List[ClaimEmailBO],
    claim_documents: List[SummaryDocumentDTO],
) -> ClaimSummaryBO:
    """
    Generate the claim's summary using LangChain's ChatOpenAI
model.
    """

    # Service to clean the claim summary info.
    claim_info = self.clean_claim_summary_info_service(claim_bo)

    # Include claim_calls, claim_emails and
claim_documents_info.
    complete_claim_info = {
        "claim_info": claim_info,
        "claim_calls": [call.to_dict() for call in claim_calls],
        "claim_emails": [email.to_dict() for email in
claim_emails],
        "claim_documents_info": [document.to_dict() for document
in claim_documents],
    }

```

```

try:
    system_message = SystemMessage(
        content=ClaimSummaryPrompt.default,
    )
    messages = [system_message] + [
HumanMessage(content=json.dumps(complete_claim_info)),
    ]

    response = self.llm.invoke(messages)

    mapped_summary =
self.output_parser.parse(response.content)
    summary_bo = ClaimSummaryBO(
        summary=[
            ClaimSummarySectionBO(title=section.title,
content=section.content)
            for section in mapped_summary.summary
        ]
    )

    except json.JSONDecodeError as json_error:
        raise LLMClaimSummaryGenerationFailed(
            f"Invalid JSON format in response:
{str(json_error)}"
        )

    except Exception as e:
        raise
LLMClaimSummaryGenerationFailed(message=f"LangChain request failed:
{str(e)}")

    if not summary_bo:
        raise LLMClaimSummaryGenerationFailed(message="No
summary generated.")

    return summary_bo

```

A-10. NewDocumentUploadProtectedView

```
from cleverea_django_integrations.rest_framework.http_response
import CustomResponse

from cleverea_django_integrations.rest_framework.version_control
import version_control

from cleverea_django_integrations.rest_framework.views import
CustomGenericAPIView

from rest_framework import permissions

from rest_framework.request import Request


from
claims.document.dependency_injection.application.view_controllers.pr
otected.new_document_upload import (
    CreateNewDocumentUploadProtectedViewControllers,
)

from
claims.document.infrastructure.api.v1.views.protected.new_document_u
pload.v1.serializers import (
    NewDocumentUploadProtectedSerializerInputV1,
)


class NewDocumentUploadProtectedView(CustomGenericAPIView):
    """
    Protected view handling new document upload notification.
    This is a subscription to ask the documents MS to analyse the
    document.
    Endpoint: /new-document-upload
    """

    serializers = {
        "POST": {
            "v1": {
                "input":
NewDocumentUploadProtectedSerializerInputV1,
            }
        }
    }

    permission_classes = (permissions.AllowAny,)
```

```

@version_control
def post(*args, **kwargs):
    pass

def post_v1(self, request: Request) -> CustomResponse:
    """
    Handles the POST request for the new document upload
    notification.

    :param request: The HTTP request object.
    """
    self.version = "v1"
    self.view_controller =
CreateNewDocumentUploadProtectedViewControllers.v1

    return
self._get_response(self.view_controller()(input_data=self._get_data(
)))

```

A-11. NewDocumentUploadProtectedViewControllerV1

```

from cleverea_documents_integrations.interface import
DocumentsInterface

from
claims.document.application.view_controllers.v1.protected.new_docume
nt_upload.v1.dto.input import (
    NewDocumentUploadProtectedInputDTOV1,
)

class NewDocumentUploadProtectedViewControllerV1:
    """
    Controller when a new document is uploaded. It asks documents to
    analyse the document.
    """

    def __init__(self, documents_service):
        self.documents_service: DocumentsInterface =
documents_service

```

```

    def _call_(self, input_data:
NewDocumentUploadProtectedInputDTOV1):
    """
    Handle the new document upload notification.
    :param input_data: The input data containing the document ID
and file vertical.
    """
    if input_data.file_vertical == "claims":
        self.documents_service.analyse_document(
            document_id=input_data.document_id,
        )

    return

```

A-12. AnalyseDocumentViewControllerV1

```

from cleverea_celery.async_tasks_manager.service import
CeleryAsyncTaskService

```

```

class AnalyseDocumentViewControllerV1:
    """
    Analyse a document view controller.
    """

    def _init_(self, async_task_service, analyse_document_task):
        """
        Class initialization.
        """
        self.async_task_service = async_task_service
        self.analyse_document_task: CeleryAsyncTaskService =
analyse_document_task

    def _call_(
        self,
        document_id: int,
    ):
        """

```

```

        Analyse a document.

        :param document_id: identifier of the document to be
analysed.
        """
        self.async_task_service.send_task(
            task=self.analyse_document_task,
            attributes=(document_id,),
        )

```

A-13. AnalyseDocumentTaskControllerV1

```

from cleverea_celery.task_controller_wrapper import
celery_task_controller_wrapper

from
documents.document.application.services.analyse_document.service
import AnalyseDocumentService

class AnalyseDocumentTaskControllerV1:
    """
    Class in charge of orchestrating the creation of a document
analysis and persisting it.
    """

    def __init__(
        self,
        analyse_document_service,
    ):
        """
        Class initialization.
        """
        self.analyse_document_service: AnalyseDocumentService =
analyse_document_service

    @celery_task_controller_wrapper
    def _call_(self, document_id: int):
        """

```



```

        Generate and persist the analysis for a given document.
        """
        self.analyse_document_service(
            document_id=document_id,
        )

```

A-14. AnalyseDocumentService

```

from datetime import datetime

from cleverea_logging.interface import LoggingInterface

from documents.document.application.services.analyse_document.dto
import (
    AnalyseDocumentLogDTO,
    AnalyseDocumentLogEvent,
    AnalyseDocumentValueLog,
)

from documents.document.domain.bo.document_analysis_bo import
DocumentAnalysisBO

from
documents.document.domain.persistence.document_analysis_bo.interface
import (
    DocumentAnalysisPersistenceInterface,
)

from documents.document.domain.persistence.document_bo.interface
import (
    DocumentBOPersistenceInterface,
)

from
documents.document.domain.services.document_analysis.create_interfac
e import (
    CreateDocumentAnalysisInterface,
)

class AnalyseDocumentService:
    """
    Service analysing a document.

```

```
    This service is responsible for reading the document, analysing
it, and
    persisting the analysis.
```

```
    """
```

```
def __init__(
    self,
    create_document_analysis_service,
    document_analysis_persistence_service,
    document_persistence_service,
    analysis_type,
    settings,
    logging_service,
):
    """
    Class initialization.

    :param create_document_analysis_service: Service to perform
the document analysis creation.

    :param document_analysis_persistence_service: Service that
handles the document analysis persistences.

    :param document_persistence_service: Service that handles
the document persistences.
    """

    self.create_document_analysis_service:
CreateDocumentAnalysisInterface = (
        create_document_analysis_service
    )

    self.document_analysis_persistence_service:
DocumentAnalysisPersistenceInterface = (
        document_analysis_persistence_service
    )

    self.document_persistence_service:
DocumentBOPersistenceInterface = (
        document_persistence_service
    )

    self.analysis_type = analysis_type
    self.settings = settings
    self.logging_service: LoggingInterface = logging_service
```

```

def _call_(
    self,
    document_id: int,
) -> DocumentAnalysisBO:
    """
    Download the document, analyse it, and persist the analysis.

    :param document_id: The id of the document to analyse.
    :return: The persisted DocumentAnalysisBO.
    """
    existing_equivalent_analysis =
self.document_analysis_persistence_service.get_list(
        filters={"document_id": document_id, "type":
self.analysis_type.value}
    )

    if len(existing_equivalent_analysis) > 0:
        return existing_equivalent_analysis[0]

    today = datetime.now().date()
    start_of_day = datetime.combine(today, datetime.min.time())
    recent_analyses =
self.document_analysis_persistence_service.get_list(
        filters={"requested_at__gte": start_of_day}
    )

    if len(recent_analyses) >=
self.settings.MAX_DOCUMENTS_ANALYSED_PER_DAY:
        self.logging_service.info(
            data=AnalyseDocumentLogDTO(
event=AnalyseDocumentLogEvent.MAX_NUMBER_OF_DOCUMENT_ANALYSIS_ERROR,
            event_value=AnalyseDocumentValueLog(
                info="Max number of document analyses
reached for today.",
                document_id=document_id,
            ),
        )
    )

```

```

        return

        document_bo =
self.document_persistence_service.get_by_id(id=document_id)

        document_analysis_bo =
self.create_document_analysis_service(
            document_bo=document_bo,
        )

        persisted_document_analysis_bo =
self.document_analysis_persistence_service.create(
            document_analysis_bo=document_analysis_bo,
            document_id=document_bo.id,
        )

        return persisted_document_analysis_bo

```

A-15. LLMSummariseDocumentService

```

import base64
import json
import mimetypes
from datetime import datetime, timezone

from cleverea_aws_integrations.s3.query_bucket import QueryS3Service
from langchain_core.messages import HumanMessage, SystemMessage
from langchain_openai import ChatOpenAI
from openai import OpenAI
from twelvelabs import TwelveLabs

from documents.core.domain.enums import DocumentAnalysisSource,
DocumentAnalysisType
from documents.document.domain.bo.document_analysis_bo import
DocumentAnalysisBO
from documents.document.domain.bo.document_bo import DocumentBO
from
documents.document.domain.services.document_analysis.create_interfac
e import (

```

```

        CreateDocumentAnalysisInterface,
    )
    from
    documents.document.infrastructure.services.llm.create_document_analy
    sis.exceptions import (
        CannotAnalyseDocument,
    )
    from
    documents.document.infrastructure.services.llm.create_document_analy
    sis.prompts import (
        DocumentSummaryPrompt,
    )
    from documents.document.infrastructure.services.llm.exceptions
    import LLMResponseGenerationFailed

class LLMSummariseDocumentService(CreateDocumentAnalysisInterface):
    """
    LLM-based implementation of the CreateDocumentAnalysisInterface
    for document summaries.
    """

    def __init__(self, *, settings, query_service):
        """
        Class initialization.

        :param settings: Application settings.
        """
        self.settings = settings
        self.openai_model = "gpt-4.1"
        self.images_client = ChatOpenAI(
            openai_api_key=settings.DOCUMENTS_ANALYSIS_OPENAI_API_KEY,
            model_name=self.openai_model,
            temperature=0.6,
        )
        self.documents_client = OpenAI(
            api_key=settings.DOCUMENTS_ANALYSIS_OPENAI_API_KEY,
        )

```

```

        self.video_client =
TwelveLabs(api_key=settings.TWELVE_LABS_API_KEY)
        self.query_service: QueryS3Service = query_service

    def _call_(self, *, document_bo: DocumentBO) ->
DocumentAnalysisBO:
        """
        Perform the analysis using LangChain.

        :param document_bo: The document to analyse.
        :return: The analysis result.
        """

        handlers = {
            "image": self._analyse_image,
            "pdf": self._analyse_pdf,
            "video": self._analyse_video,
        }
        try:
            return
handlers[self._get_file_type(document_bo=document_bo)](document_bo=d
ocument_bo)
        except KeyError:
            raise CannotAnalyseDocument(
                message=f"The document cannot be analysed due to an
unhandled mimetype. {self._get_file_type(document_bo=document_bo)}"
            )

    def _analyse_image(self, *, document_bo: DocumentBO) ->
DocumentAnalysisBO:
        """
        Perform the analysis using LangChain.

        :param document_bo: The image to analyse.
        :return: The analysis result.
        """
        try:
            response = self.images_client.invoke(
                input=[

```

```

        SystemMessage(DocumentSummaryPrompt.image),
        HumanMessage(
            content=[
                {"type": "image_url", "image_url":
{"url": document_bo.file.url}},
            ]
        ),
    ]
)

except Exception as e:
    raise LLMResponseGenerationFailed(f"LangChain failed to
generate image summary: {e}")

summary_dict = json.loads(response.content)

return DocumentAnalysisBO.create_completed(
    type=DocumentAnalysisType.SUMMARY,

source=DocumentAnalysisSource(self.images_client.model_name),
    requested_at=datetime.now(timezone.utc),
    value=summary_dict,
    generated_at=datetime.now(timezone.utc),
)

def _analyse_pdf(self, *, document_bo: DocumentBO) ->
DocumentAnalysisBO:
    """
    Perform the analysis using LangChain.

    :param document_bo: The document to analyse.
    :return: The analysis result.
    """
    document_content = self.query_service.get_file_content(
        bucket_name=self.settings.AWS_STORAGE_BUCKET_NAME,
        file_path=document_bo.file.name,
    )

```

```

        base64_document =
base64.b64encode(document_content).decode("utf-8")
        data_url = f"data:application/pdf;base64,{base64_document}"
        try:
            response = self.documents_client.responses.create(
                model=self.openai_model,
                temperature=0.6,
                input=[
                    {
                        "role": "user",
                        "content": [
                            {
                                "type": "input_text",
                                "text":
DocumentSummaryPrompt.document,
                            },
                            {
                                "type": "input_file",
                                "filename": "document.pdf",
                                "file_data": data_url,
                            },
                        ],
                    }
                ],
            )
        except Exception as e:
            raise LLMResponseGenerationFailed(f"OpenAI failed to
generate pdf summary: {e}")

        summary_dict = json.loads(response.output_text)

        return DocumentAnalysisBO.create_completed(
            type=DocumentAnalysisType.SUMMARY,
            source=DocumentAnalysisSource(self.images_client.model_name),
            requested_at=datetime.now(timezone.utc),
            value=summary_dict,
            generated_at=datetime.now(timezone.utc),
        )

```



```

def _analyse_video(self, *, document_bo: DocumentBO) ->
DocumentAnalysisBO:
    """
    Perform the analysis using LangChain.

    :param document_bo: The video to analyse.
    :return: The analysis result.
    """
    video_url = document_bo.file.url

    try:
        task = self.video_client.task.create(
index_id=self.settings.TWELVE_LABS_VIDEO_SUMMARY_INDEX_ID,
            url=video_url,
        )
    except Exception as e:
        raise LLMResponseGenerationFailed(
            f"Twelve Labs failed to create video summary task:
{e}"
        )

    external_video_id = task.video_id

    return DocumentAnalysisBO.create_pending(
        type=DocumentAnalysisType.SUMMARY,
        source=DocumentAnalysisSource.TWELVE_LABS,
        requested_at=datetime.now(timezone.utc),
        external_id=external_video_id,
    )

def _get_file_type(self, *, document_bo: DocumentBO) -> str:
    """
    Get the file type based on the mime type.

    :param document_bo: The document to get the type.
    :return: The file type.

```

```

"""
mime, _ = mimetypes.guess_type(document_bo.file.name)
if not mime:
    return "unknown"
if mime.startswith("image/"):
    return "image"
elif mime.startswith("video/"):
    return "video"
elif mime.startswith("application/pdf"):
    return "pdf"
return "other"

```

A-16. TwelveLabsWebhookDocumentAnalysisView

```

import hashlib
import hmac

from cleverea_django_integrations.rest_framework.http_response
import CustomResponse
from cleverea_django_integrations.rest_framework.version_control
import version_control
from cleverea_django_integrations.rest_framework.views import
CustomGenericAPIView
from cleverea_exceptions import AuthorizationException,
BodyException
from drf_spectacular.types import OpenApiTypes
from drf_spectacular.utils import extend_schema, OpenApiParameter
from rest_framework import permissions
from rest_framework.request import Request

from django.conf import settings

from
documents.document.dependency_injection.application.view_controllers
.document_analysis.complete import (
    CompleteDocumentAnalysisViewControllers,
)
from
documents.document.infrastructure.api.v1.views.public.document_analy
sis.twelve_labs_webhook.v1.serializers import (

```

```

        TwelveLabsWebhookDocumentAnalysisSerializerInputV1,
    )
    from
    documents.document.infrastructure.api.v1.views.public.document_analy
    sis.twelve_labs_webhook.v1.swagger_examples import (
        V1_EXAMPLES,
    )

class TwelveLabsWebhookDocumentAnalysisView(CustomGenericAPIView):
    """
    Create the DocumentAnalysis after the video has been processed
    by Twelve Labs.

    Endpoint: /documents/document-analysis-response/twelve-labs-
    webhook
    """

    serializers = {
        "POST": {
            "v1": {
                "input":
TwelveLabsWebhookDocumentAnalysisSerializerInputV1,
            }
        }
    }

    permission_classes = (permissions.AllowAny,)

    @extend_schema(
        description="Webhook endpoint to receive asynchronous events
        from Twelve Labs. The request must include a valid TL-Signature
        header for integrity verification.",
        parameters=[
            OpenApiParameter(
                name="accept-version",
                type=OpenApiTypes.STR,
                location=OpenApiParameter.HEADER,
                description="Version to use.",
                enum=["1"],
                required=True,

```

```

    ),
    OpenApiParameter(
        name="TL-Signature",
        type=OpenApiTypes.STR,
        location=OpenApiParameter.HEADER,
        description="HMAC SHA-256 signature used to verify
the integrity of the request. Format: t=<timestamp>,v1=<signature>",
        required=True,
    ),
],
examples=[*V1_EXAMPLES["POST"]],
)
@version_control
def post(*args, **kwargs):
    pass

def post_v1(self, request: Request) -> CustomResponse:
    """
    Handle the webhook from Twelve Labs, creating the proper
    DocumentAnalysis.

    :param request: DRF request object.
    """
    self.version = "v1"

    self._verify_signature(request=request)

    self.view_controller =
CompleteDocumentAnalysisViewControllers.v1

    data = self._get_data(request.POST)

    if data.type != "index.task.ready":
        raise AuthorizationException(
            reason="twelveLabsWebhookEventTypeNotSupported",
            message="The Twelve Labs webhook event type is not
supported.",
        )

```

```

        return
self._get_response(self.view_controller()(input_data=data))

def _verify_signature(self, request: Request) -> None:
    """
    Verifies the TL-Signature header and ensures the request is
    authentic.
    Raises exceptions if validation fails.
    """

    signature_header = request.headers.get("TL-Signature")
    if not signature_header:
        raise BodyException(message="Missing TL-Signature
header", reason="missingTLSigHeader")

    try:
        parts = dict(item.split("=") for item in
signature_header.split(","))
        timestamp = parts["t"]
        received_signature = parts["v1"]
    except Exception:
        raise BodyException(message="Invalid TL-Signature
format", reason="invalidTLSigFormat")

    raw_body = request._request.body.decode("utf-8")
    signed_payload = f"{timestamp}.{raw_body}"
    secret = settings.TWELVE_LABS_WEBHOOK_SECRET.encode("utf-8")

    expected_signature = hmac.new(
        secret, signed_payload.encode("utf-8"), hashlib.sha256
    ).hexdigest()

    if not hmac.compare_digest(received_signature,
expected_signature):
        raise AuthorizationException(message="Invalid
signature", reason="invalidTLSigHash")

```

A-17. CompleteDocumentAnalysisViewControllerV1

```
from datetime import datetime

from pytz import utc

from documents.document.domain.exceptions import
DocumentAnalysisMustBePendingBeforeCompleting
from
documents.document.domain.persistence.document_analysis_bo.interface
import (
    DocumentAnalysisPersistenceInterface,
)
from
documents.document.domain.services.document_analysis.retrieve_value_
interface import (
    RetrieveDocumentAnalysisValueInterface,
)

class CompleteDocumentAnalysisViewControllerV1:
    """
    Controller for completing a document analysis from an external
    source.
    """

    def __init__(
        self, *, document_analysis_persistence_service,
        retrieve_document_analysis_value_service
    ):
        self.document_analysis_persistence_service:
        DocumentAnalysisPersistenceInterface = (
            document_analysis_persistence_service
        )
        self.retrieve_document_analysis_value_service:
        RetrieveDocumentAnalysisValueInterface = (
            retrieve_document_analysis_value_service
        )

    def _call_(self, *, input_data):
        """
```

Complete the document analysis based on the input data received.

This method retrieves the pending document analysis by its external ID,

checks if it is in a pending state, and then retrieves the analysis value

from an external service. Finally, it updates the document analysis to

a completed state with the retrieved value and the current timestamp.

```
:param input_data: Input data for completing the analysis.
"""
```

```
        pending_document_analysis_bo =
self.document_analysis_persistence_service.get(
            filters={"external_id": input_data.data["id"]}
        )

        if not pending_document_analysis_bo.is_pending:
            raise DocumentAnalysisMustBePendingBeforeCompleting

        retrieved_document_analysis_value =
self.retrieve_document_analysis_value_service(
            external_id=pending_document_analysis_bo.external_id
        )

        completed_analysis =
pending_document_analysis_bo.complete_analysis(
            value=retrieved_document_analysis_value,
            generated_at=datetime.now(utc)
        )

self.document_analysis_persistence_service.update_to_complete(
    document_analysis_bo=completed_analysis
)
```

A-18. RetrieveTwelveLabsResponseService

```
import json

from twelvelabs import TwelveLabs

from documents.document.domain.services.document_analysis.retrieve_value_
interface import (
    RetrieveDocumentAnalysisValueInterface,
)

from documents.document.infrastructure.services.llm.create_document_analy
sis.exceptions import (
    TwelveLabsSummaryGenerationFailed,
)

from documents.document.infrastructure.services.llm.create_document_analy
sis.prompts import (
    DocumentSummaryPrompt,
)

class
RetrieveTwelveLabsResponseService(RetrieveDocumentAnalysisValueInter
face):
    """
    This class is responsible for retrieving document analysis from
    Twelve Labs.
    """

    def __init__(self, *, settings):
        self.settings = settings
        self.video_client =
TwelveLabs(api_key=settings.TWELVE_LABS_API_KEY)

    def __call__(self, external_id: str) -> dict:
        """
        Retrieve the document analysis from Twelve Labs and create a
        summary.
        :param external_id: The external identifier of the video.
        """
```



```

        """
    try:
        response = self.video_client.generate.summarize(
            external_id,
            type="summary",
            prompt=DocumentSummaryPrompt.video,
            temperature=0.2,
        )
        summary_dict = json.loads(response.summary)
    except Exception as e:
        raise TwelveLabsSummaryGenerationFailed(
            message="Failed to generate summary from Twelve Labs
Exception: " + str(e)
        )

    try:
        self.video_client.task.delete(id=external_id)

    except Exception:
        pass

    return summary_dict

```

A-19. TestGetClaimSummary

```

from tests.conftest import reverse

class TestGetClaimSummary:
    """
    Integration tests for the GetClaimSummary view controller.
    """

    def test_success(
        self,
        auth_cognito_api_client,
        db_claim_bo_factory_base,

```

```

):
    """
    Test the happy path flow of retrieving the claims's summary.
    """
    kwargs = {
        "claim_id": db_claim_bo_factory_base.id,
    }
    response = auth_cognito_api_client.get(
        reverse(
            "v1:public:claim:claim_summary",
            kwargs=kwargs,
        )
    )
    assert response.status_code == 200
    response_data = response.json()["data"]
    assert "summary_section" in response_data[0]
    assert "title" in response_data[0]["summary_section"]
    assert "content" in response_data[0]["summary_section"]

def test_claim_not_found(self, auth_cognito_api_client):
    """
    Test the case where the claim does not exist.
    """
    kwargs = {
        "claim_id": 0,
    }
    response = auth_cognito_api_client.get(
        reverse(
            "v1:public:claim:claim_summary",
            kwargs=kwargs,
        )
    )
    assert response.status_code == 404

```

A-20. ClaimBO

```
class ClaimBO(DataclassBase):
```

```

"""
BO containing the basic data of a claim.
"""

#: Identifier of the claim in DB.
id: Optional[int] = None

#: Identifier of the fnol if the claim has been generated from
one.
fnol: Optional[FmolBO] = None

#: Date and time when the claim was created on DB.
created_at: Optional[datetime] = None

#: Date and time when the claim was updated on DB.
updated_at: Optional[datetime] = None

#: Vertical in which the policy to which the fnol is related to
lives.
vertical: Vertical

#: Number of the policy to which the claim is related.
policy_number: Optional[str] = None

#: Type of incident for which the claim was created.
claim_type: IncidentTypeBO

#: Concatenated date and time referring to when the incident was
notified.
notification_datetime: datetime

#: Date and time when the incident occurred.
incident_datetime: datetime

#: Complete address referring to where the incident occurred.
full_address: str

#: Postal code of the direction where the incident occurred.
zip_code: str

#: Handler of the claim.
handler: Optional[HandlerBO] = None

#: Provider of the claim.
provider: Optional[ProviderBO] = None

#: Reference of the claim on the external provider DB.
ext_provider_reference: Optional[str] = None

#: Reference of the claim on an external tool provider DB, as
SISnet.
ext_tool_provider_reference: Optional[str] = None

#: External identifier on DB of the user that owns this claim.

```

```

ext_user_id: Optional[int] = None
#: More specific details of the claim.
detail: Optional[ClaimDetailBO] = None
#: List of status that have had the claim.
status: List[ClaimStatusBO]
#: List of customer status that have had the claim.
customer_status: List[ClaimCustomerStatusBO]
#: List of persons related to the claim.
persons: Optional[List[ClaimPersonBO]] = None
#: List of risk objects related to the claim.
risk_objects: Optional[List[ClaimRiskObjectBO]] = None
#: List of files of this claim.
files: Optional[List[FileBO]] = None
#: List of comments of this claim.
comments: Optional[List[CommentBO]] = None
#: Datetime the claim was reviewed.
review_datetime: Optional[datetime] = None
#: Urgency of the claim
urgency: Optional[Urgency] = None
#: Merged claim identifier
merged_claim_id: Optional[int] = None
#: Flag indicating if the model is soft deleted or not.
is_deleted: Optional[bool] = False

```

A-21. GenerateClaimSummaryInterface

```

from abc import ABC, abstractmethod
from typing import List

from claims.claim.domain.bo.claim_bo import ClaimBO
from claims.claim.domain.bo.claim_call_bo import ClaimCallBO
from claims.claim.domain.bo.claim_email_bo import ClaimEmailBO
from claims.claim.domain.bo.claim_summary_bo import ClaimSummaryBO
from claims.claim.domain.dto.summary_document_dto import
SummaryDocumentDTO

class GenerateClaimSummaryInterface(ABC):

```

```

"""
Class containing the contract in the claim summary services.
"""

@abstractmethod
def __init__(self):
    """
    Class initialization.
    """
    raise NotImplementedError()

@abstractmethod
def __call__(
    self,
    *,
    claim_bo: ClaimBO,
    claim_calls: List[ClaimCallBO],
    claim_emails: List[ClaimEmailBO],
    claim_documents: List[SummaryDocumentDTO],
) -> ClaimSummaryBO:
    """
    Generate the claim's summary.
    :param claim_bo: the claim business object.
    :param claim_calls: the list of calls related to the claim.
    :param claim_emails: the list of emails related to the
claim.
    :param claim_documents: the list of documents related to the
claim, with their info.
    :return the summary of the comments.
    """
    raise NotImplementedError()

```

Annex B - Distribució de les dades a DynamoDB

ConnectContactIdUsersTable

Partition Key: documentNumber

Sort Key: Cap

GSI: DocumentNumberIndex

Contingut: Registre d'identificació entre un número de document oficial (DNI/NIE) i un `contact_id` intern del sistema.

Aquesta taula és la peça clau que permet establir la relació entre un sinistre (identificant el document oficial del client associat) i la resta d'informació conversacional. Es tracta d'una taula auxiliar, però essencial, ja que proporciona el primer pas per accedir a la resta de dades de trucades i correus.

ConnectAnsweredCallsData

Partition Key: `contact_id`

Sort Key: `created_at`

GSI: Cap

Contingut: A més a més dels camps directament relacionats també hi ha altres camps que serveixen per la classificació de les trucades o oferir més informació d'aquesta.

Aquesta taula conté el registre de totes les trucades telefòniques que han estat ateses per part d'un client. Per recuperar aquelles rellevants per un sinistre, s'ha implementat un filtre basat en dues cues específiques: "Claims_Entrantes" i "Claims_Callback_01", que són utilitzades per la classificació de les trucades sobre el camp `queue_name`.

La relació del sinistre es fa de forma indirecta:

- S'obté el `contact_id` a partir del `document_number`.
- Es filtren les trucades del `contact_id` dins d'un rang de dates concret (entre les 24 hores prèvies a la creació del sinistre i la data de tancament del sinistre o la data actual si l'anterior no existeix).
- Es validen les cues per assegurar que pertanyen a l'àmbit de sinistres.

Aquestes validacions addicionals són necessàries per evitar incloure trucades no desitjades al context del sinistre, com podrien ser trucades amb motiu de dubtes sobre la pòlissa contractada o trucades d'un sinistre anterior.

ConnectCallsAnalysis

GSI: ContactIdIndex

Contingut: `summary`, `duration`, `satisfaction_score`, etc.

Aquesta taula conté una anàlisi estructurada de les trucades ateses, amb informació obtinguda mitjançant intel·ligència artificial. La informació més rellevant que conté inclou:

- **Resums automàtics** del contingut de la trucada (`summary`).
- **Duració de la trucada** (`duration`).
- Altres estadístiques internes.

AnalyzedHubspotEmails

GSI 1: `ReceiverByDateIndex` (`receiver`, `sent_date`)

GSI 2: `SenderByDateIndex` (`sender`, `sent_date`)

Contingut: `summary`, `category`, `body`, etc.

Aquesta taula emmagatzema correus electrònics intercanviats entre el client i l'asseguradora. L'objectiu és obtenir una visió clara del contingut dels missatges i la seva rellevància e el context d'un sinistre.

Per localitzar els correus associats a un sinistre, es parteix de l'adreça electrònica del client (ja sigui com a `receiver` o `sender`) i es delimita l'interval temporal en què el sinistre ha estat actiu (i les seves 24 hores anteriors).

Annex C - figures de suport

C-1. Diagrama flux d'anàlisi automàtica de documents

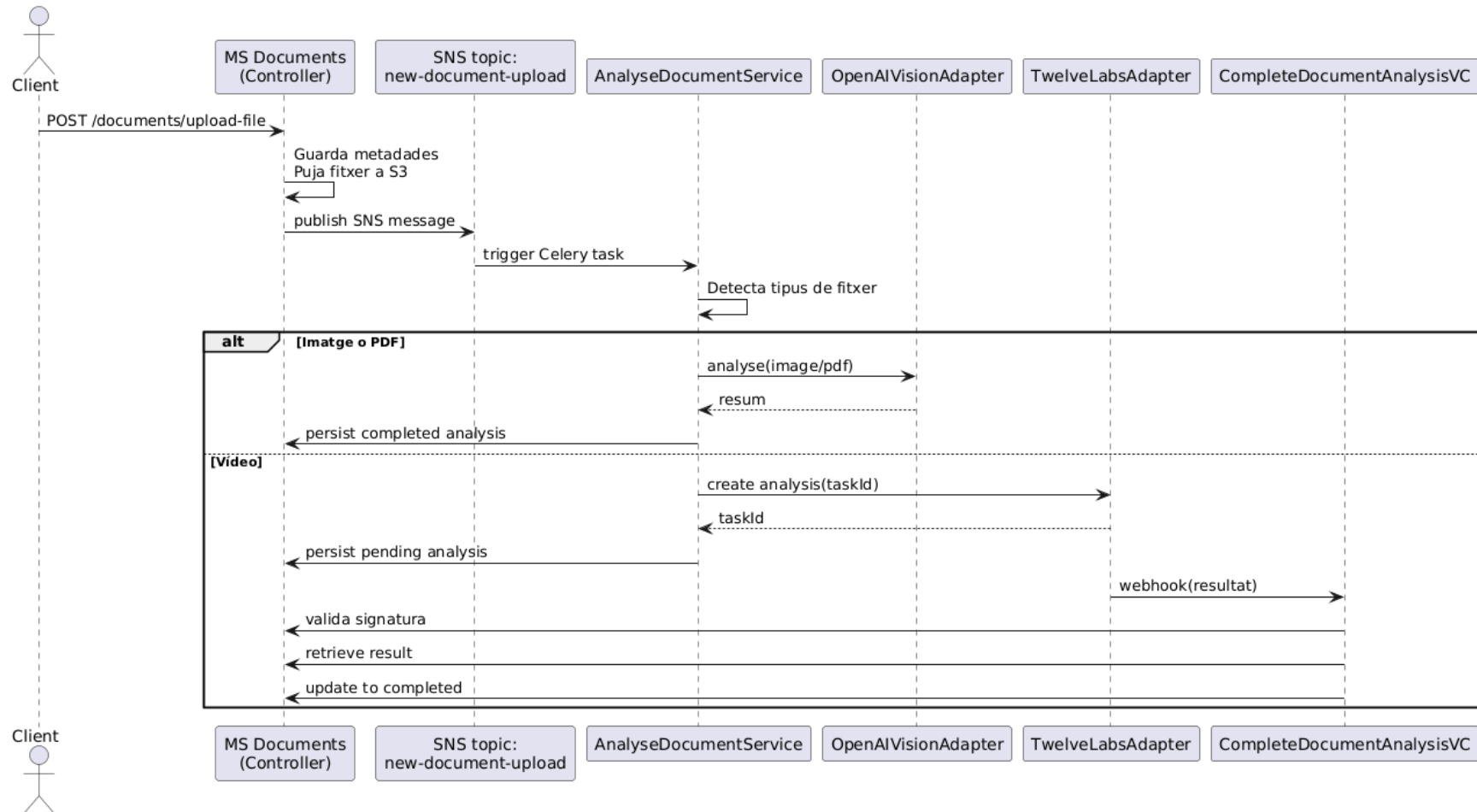


Figura 15. Diagrama flux d'anàlisi automàtica de documents. Font: <https://www.plantuml.com>.

C-2a. Diagrama 1 flux de generació del resum de sinistre

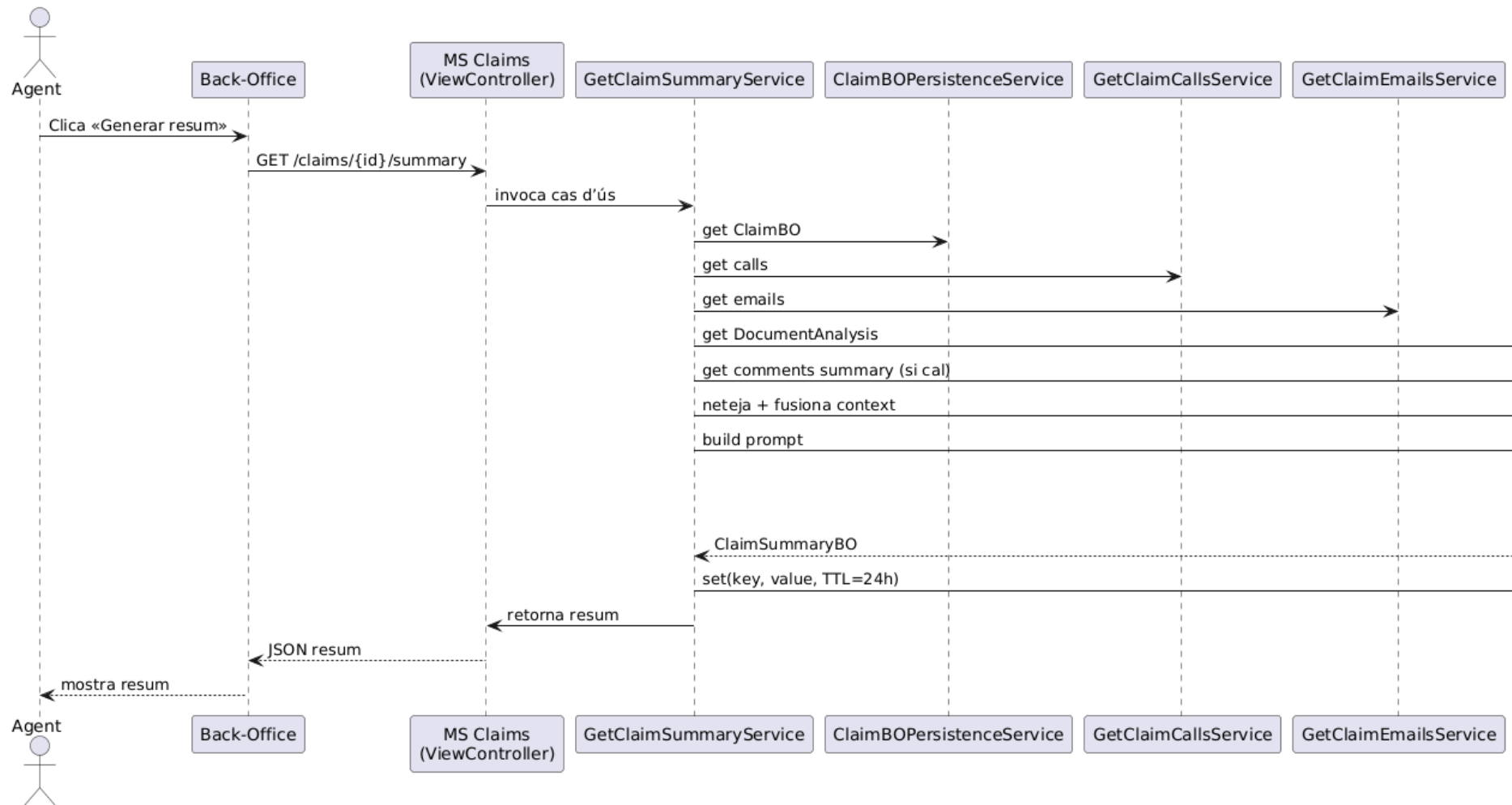


Figura 16. Diagrama 1 flux de generació del resum de sinistre. Font: <https://www.plantuml.com>.

C-2b. Diagrama 1 flux de generació del resum de sinistre

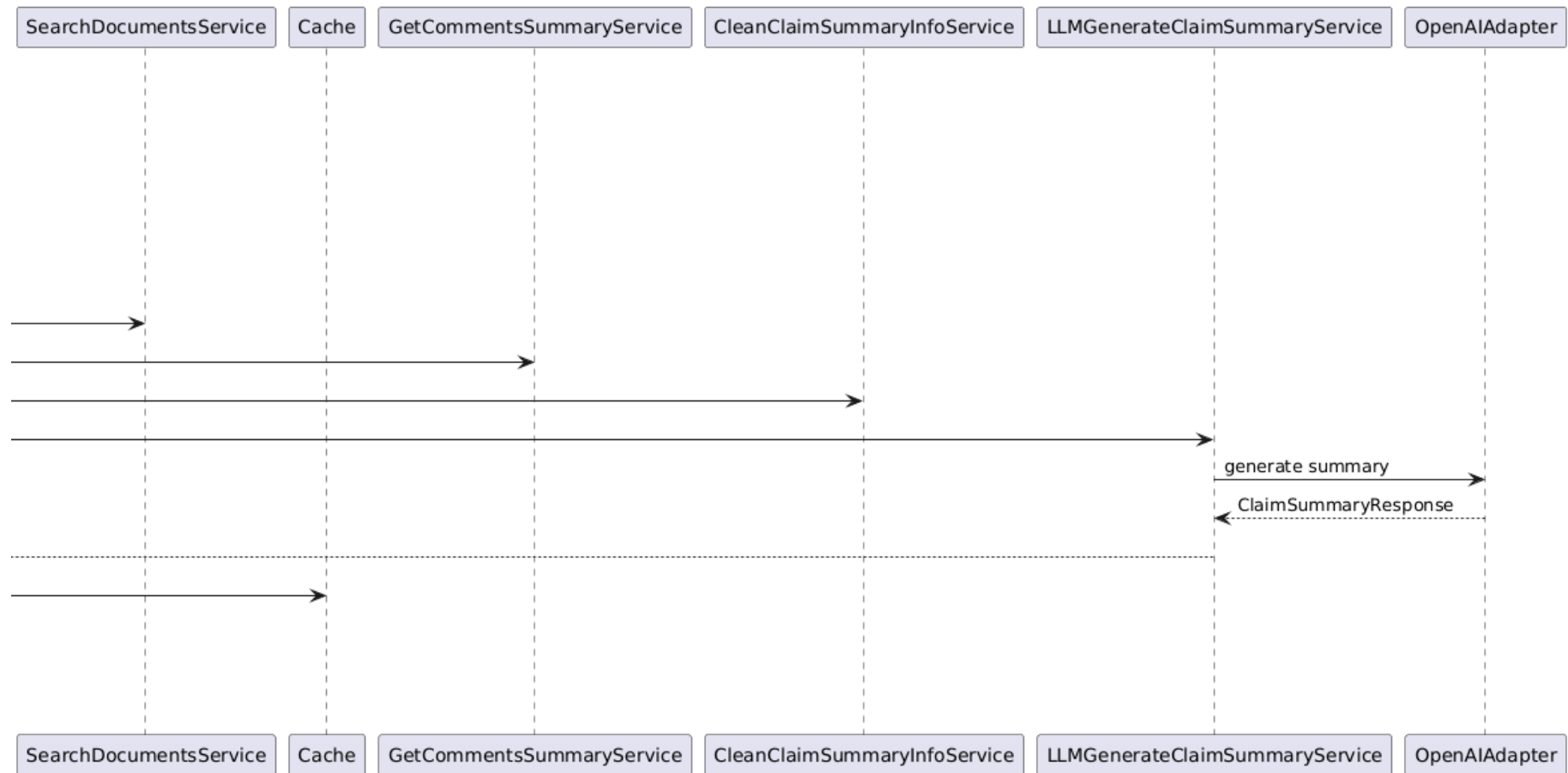


Figura 17. Diagrama 2 flux de generació del resum de sinistre. Font: <https://www.plantuml.com>.

Annex D - Exemples de resums

D-1. Resum complet de sinistre

```
{
  "data": [
    {
      "summary_section": {
        "title": "Apertura y gestión inicial del siniestro",
        "content": "El siniestro 43399, correspondiente a la póliza CR82325, fue notificado el 17 de enero de 2025, con fecha de incidente el 9 de enero de 2025. El asegurado, informó que su vehículo, un VOLKSWAGEN PASSAT, fue impactado por detrás por otro vehículo mientras estaba aparcado en el polígono industrial Ventorro del Cano. La aseguradora responsable es Divina Pastora. Desde el inicio, el agente Mateo se encargó de la gestión, explicando que el proceso podría extenderse por la espera de respuesta de la compañía contraria, la peritación y la reparación."
      }
    },
    {
      "summary_section": {
        "title": "Intervención de la compañía contraria y programación de peritaciones",
        "content": "El 29 de enero de 2025 se informó que la compañía contraria no había respondido aún a la reclamación, pero se insistiría para agilizar la resolución. El 31 de enero se confirmó la aceptación de culpa por parte de la compañía contraria y se solicitó al asegurado los datos del taller y una fecha para la peritación, exigiendo un margen mínimo de 48 horas laborables. Se programó la peritación para el 12 de febrero de 2025 en Carrocerías Rodamovil, Alcorcón."
      }
    },
    {
      "summary_section": {
        "title": "Incidencias y reprogramación de la peritación",
        "content": "El 18 de febrero de 2025, el asegurado comunicó que la peritación no pudo realizarse porque el taller estaba cerrado por vacaciones y solicitó una nueva fecha. Tras verificar con el taller, se coordinó una nueva cita para el 21 de febrero de 2025 y se recomendó que el vehículo estuviera disponible desde primera hora. Esta gestión implicó diálogo con el taller y el cliente, así como la confirmación de la nueva cita pericial."
      }
    }
  ]
}
```

```

    }
  },
  {
    "summary_section": {
      "title": "Pagos, reservas y servicios gestionados",
      "content": "A lo largo del proceso se realizaron varias
reservas y pagos. Destacan un pago de 13,31 euros el 18 de febrero
de 2025, otro de 121 euros el 18 de marzo de 2025 y uno relevante de
4348,17 euros el 26 de marzo de 2025. También se registraron
recuperaciones, como una de -1114 euros el 2 de abril de 2025 y otra
de -121 euros el 31 de marzo de 2025. Los servicios de peritación de
daños materiales fueron gestionados en varias ocasiones, aunque
algunos quedaron pendientes o anulados. El expediente principal con
daños al vehículo sigue abierto y gestionado bajo el sistema CICOS,
con JALMAGRO como TPA handler."
    }
  },
  {
    "summary_section": {
      "title": "Seguimiento, comunicaciones y estado actual",
      "content": "Desde febrero hasta junio de 2025 se enviaron
múltiples recordatorios automáticos al asegurado, informando que el
expediente sigue en gestión y agradeciendo su paciencia. No se
detectan desacuerdos ni nuevas solicitudes de información por parte
del cliente durante este periodo. El siniestro se encuentra aún
abierto a fecha 16 de junio de 2025, pendiente de resolución final y
cierre administrativo."
    }
  },
  {
    "summary_section": {
      "title": "Llamadas y Mails",
      "content": "El 18 de febrero de 2025, el asegurado contactó
telefónicamente para coordinar la reprogramación de la peritación,
gestionando la nueva cita con el taller y solicitando ser informado
sobre la confirmación. No se registran emails relevantes para este
siniestro."
    }
  },
  {
    "summary_section": {
      "title": "Documentos Subidos",
      "content": "El 28 de enero de 2025 se subió un documento
relacionado con el siniestro, pero no ha sido validado y no existen

```

análisis asociados. No se aportan más documentos validados ni información adicional sobre su contenido."

```
    }  
  }  
]  
}
```

D-2. Resum d'una imatge

```
{"summary": "La imagen muestra un automóvil de color gris estacionado en la vía pública, con la ventanilla trasera derecha rota y el vidrio completamente estrellado y parcialmente caído dentro del vehículo. Se observan fragmentos de vidrio tanto en el asiento como en el suelo. En el interior del coche hay algunos objetos, como una caja de cartón y lo que parecen ser herramientas o materiales. El daño es consistente con un posible acto vandálico o robo. Al fondo se aprecian edificios residenciales y otros vehículos estacionados."}
```

Annex E – Prompts

E-1. Prompt resum de sinistre

```
import datetime  
  
class ClaimSummaryPrompt:  
    current_date = datetime.datetime.now().strftime("%Y-%m-%d")  
    default = (  
        """  
Actúa como un asistente experto en resumir siniestros en formato  
JSON, generando un resumen narrado en español que refleje la  
situación general y destaque los puntos más relevantes (desacuerdos,  
solicitudes de información y acciones pendientes) teniendo en cuenta  
la fecha actual """  
        + current_date  
        + """.
```

Tu respuesta debe ser **exclusivamente** un objeto JSON válido, sin texto adicional ni nodos fuera de ese objeto, y debe poder parsearse con `json.loads()` sin errores. Si falta información en los datos originales, menciona que no fue proporcionada o no existe. **No alteres la información recibida**.

1. *Resumen narrado*

- Redáctalo en *español* con estilo narrativo y en *texto plano*.
- *Máximo 600 palabras*.
- Estructúralo en uno o varios párrafos con *títulos*, usando la siguiente estructura:

```
json
{
  "summary": [
    {
      "title": "Título del primer párrafo",
      "content": "Texto del primer párrafo"
    },
    {
      "title": "Título del segundo párrafo",
      "content": "Texto del segundo párrafo"
    },
    ...
  ]
}
```

- Destaca los desacuerdos, solicitudes de información y acciones pendientes.
- Subraya las fechas relevantes.
- *Analiza las fechas* en los datos (campos de fecha y referencias temporales) para presentar los eventos *en orden cronológico* dentro de tu narración.
- Emplea oraciones breves y concisas.

2. *Formato estricto*

- Responde *exclusivamente* con el *objeto JSON* anterior, sin ningún texto adicional.
- Verifica que sea un objeto JSON válido y *no* excedas las 600 palabras en total.

3. *Objetivo*

- Reflejar la situación global, enfatizando los puntos clave (las acciones pendientes son muy relevantes).
- Ordenar los hechos de manera *cronológica*, con base en las fechas encontradas en los datos.

- Mantenerte fiel a la información proporcionada, sin alterarla ni inventar datos.
- Evitar ocupar espacio con información redundante o poco relevante.
- Haz un apartado extra para informar de las llamadas y emails relevantes, llamado "Llamadas y Mails".
- Haz un apartado extra para informar de los documentos subidos, llamado "Documentos Subidos".

4. *Información relevante*

- Los amount dentro de los payments se expresan en céntimos pero tu debes devolverlos en euros.
- Las llamadas y mails son del mismo usuario, pero pueden referirse a otros siniestros o gestiones. *Solo debes tener en cuenta los que se relacionen con el siniestro actual, según el contenido de su resumen.*
- El resumen de cada llamada puede contener errores ya que proviene de una transcripción automática. Prioriza los datos externos si hay contradicciones.
- Si no hay documentos subidos, menciona que no existen o no fueron proporcionados.

"""

)

E-2. Prompt resum d'una imatge

```
import datetime
```

```
class ClaimSummaryPrompt:
```

```
    current_date = datetime.datetime.now().strftime("%Y-%m-%d")
```

```
    default = (
```

```
        """
```

```
Actúa como un asistente experto en resumir siniestros en formato
JSON, generando un resumen narrado en español que refleje la
situación general y destaque los puntos más relevantes (desacuerdos,
solicitudes de información y acciones pendientes) teniendo en cuenta
la fecha actual """
```

```
        + current_date
```

```
        + """.
```

Tu respuesta debe ser **exclusivamente** un objeto JSON válido, sin texto adicional ni nodos fuera de ese objeto, y debe poder parsearse

con `json.loads()` sin errores. Si falta información en los datos originales, menciona que no fue proporcionada o no existe. *No alteres la información recibida*.

1. *Resumen narrado*

- Redáctalo en *español* con estilo narrativo y en *texto plano*.
- *Máximo 600 palabras*.
- Estructúralo en uno o varios párrafos con *títulos*, usando la siguiente estructura:

```
json
{
  "summary": [
    {
      "title": "Título del primer párrafo",
      "content": "Texto del primer párrafo"
    },
    {
      "title": "Título del segundo párrafo",
      "content": "Texto del segundo párrafo"
    },
    ...
  ]
}
```

- Destaca los desacuerdos, solicitudes de información y acciones pendientes.
- Subraya las fechas relevantes.
- *Analiza las fechas* en los datos (campos de fecha y referencias temporales) para presentar los eventos *en orden cronológico* dentro de tu narración.
- Emplea oraciones breves y concisas.

2. *Formato estricto*

- Responde *exclusivamente* con el *objeto JSON* anterior, sin ningún texto adicional.
- Verifica que sea un objeto JSON válido y *no* excedas las 600 palabras en total.

3. *Objetivo*

- Reflejar la situación global, enfatizando los puntos clave (las acciones pendientes son muy relevantes).
- Ordenar los hechos de manera *cronológica*, con base en las fechas encontradas en los datos.
- Mantenerse fiel a la información proporcionada, sin alterarla ni inventar datos.
- Evitar ocupar espacio con información redundante o poco relevante.
- Haz un apartado extra para informar de las llamadas y emails relevantes, llamado "Llamadas y Mails".
- Haz un apartado extra para informar de los documentos subidos, llamado "Documentos Subidos".

4. *Información relevante*

- Los amount dentro de los payments se expresan en céntimos pero tu debes devolverlos en euros.
- Las llamadas y mails son del mismo usuario, pero pueden referirse a otros siniestros o gestiones. *Solo debes tener en cuenta los que se relacionen con el siniestro actual, según el contenido de su resumen.*
- El resumen de cada llamada puede contener errores ya que proviene de una transcripción automática. Prioriza los datos externos si hay contradicciones.
- Si no hay documentos subidos, menciona que no existen o no fueron proporcionados.

""

)

E-2. Prompt resum d'un PDF

""

Analiza el archivo proporcionado. Este puede contener texto, documentos u otros elementos gráficos relevantes relacionados con un siniestro.

Tu tarea es generar una descripción única, clara y concisa del contenido y propósito del archivo en español. Si hay información relevante explícita visible en el archivo debe integrarse de forma natural dentro del texto descriptivo.

No inventes nada. Solo utiliza información que esté presente y visible en el archivo. La respuesta debe devolverse en un único campo JSON con esta estructura exacta:

```
{  
  "summary": "Texto explicativo completo, incluyendo datos  
relevantes encontrados de forma integrada y natural."  
}
```

No añadas ningún texto adicional fuera del JSON.

"""

E-2. Prompt resum d'un vídeo

"""

Analiza el video. Este puede contener la explicación de un siniestro o imágenes de éste (daños, contexto, testimonios o ubicaciones afectadas).

Tu tarea es generar una descripción única, clara y CONCISA del contenido y propósito del video en español. Si el video trata sobre daños materiales, enfócate en proporcionar un resumen detallado de lo observado (por ejemplo: tipo de daños, ubicación de los mismos, condiciones del entorno). Si hay información relevante visible o audible intégrala de forma natural.

No inventes nada. Solo utiliza información que esté presente y visible o audible en el video. La respuesta debe devolverse en un único campo JSON con esta estructura exacta:

```
{  
  "summary": "Texto explicativo completo, incluyendo detalles sobre  
el siniestro y otros datos relevantes encontrados de forma integrada  
y natural."  
}
```

No añadas ningún texto adicional fuera del JSON.

"""