
This is the **published version** of the bachelor thesis:

Seró Alsina, Àlex. *Aplicació web de planificació de viatges amb execució d'accions mitjançant IA*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334368>

under the terms of the  license.

Aplicació web de planificació de viatges amb execució d'accions mitjançant IA

Àlex Seró Alsina

29 de juny de 2026

Resum— Aquest projecte té com a objectiu el desenvolupament d'una aplicació web per a la planificació i la gestió de viatges. L'aplicació permet el registre, l'autenticació d'usuaris, la gestió de viatges i l'organització de l'itinerari dia a dia amb les seves activitats. Com a element central, s'incorpora un agent d'intel·ligència artificial que executa accions reals sobre el sistema a partir d'instruccions en llenguatge natural. Es construeix seguint una metodologia àgil i una arquitectura en capes, amb Vue al frontend i Node.js amb PostgreSQL al backend, i s'hi integra l'agent mitjançant el Model Context Protocol (MCP) amb l'API d'Anthropic. Amb aquesta eina es busca centralitzar tota la informació del viatge i simplificar-ne la gestió.

Paraules clau— Planificació de viatges, intel·ligència artificial (IA), agent d'IA, Model Context Protocol (MCP), Claude, aplicació web, API REST, PostgreSQL.

Abstract— This project aims to develop a web application for planning and managing trips. The application allows user registration, authentication, trip management, and the day-by-day organization of the itinerary with its activities. As a central element, an artificial intelligence agent is incorporated, which performs real actions on the system from natural-language instructions. It is built following an agile methodology and a layered architecture, with Vue on the frontend and Node.js with PostgreSQL on the backend, and the agent is integrated through the Model Context Protocol (MCP) with Anthropic's API. This tool aims to centralize all the trip information and simplify its management.

Keywords— Trip planning, artificial intelligence (AI), AI agent, Model Context Protocol (MCP), Claude, web application, REST API, PostgreSQL.



1 INTRODUCCIÓ

EN els darrers anys, la intel·ligència artificial s'ha integrat de manera creixent en les aplicacions web, especialment en àmbits relacionats amb l'assistència personalitzada i l'automatització de tasques. L'aparició dels grans models de llenguatge, capaços d'interpretar les peticions dels usuaris expressades en llenguatge natural, ha impulsat aquest avenç i ha donat lloc a assistents intel·ligents cada cop més presents en el dia a dia. Malgrat això, la planificació de viatges continua sent una activitat que requereix temps i organització, ja que obliga a coordinar diferents elements com transports, allotjaments, itineraris i activitats.

Aquest projecte té com a objectiu el disseny i el desenvolupament d'una aplicació web orientada a la gestió i la planificació de viatges, que incorpora un agent d'intel·ligència

artificial capaç d'assistir l'usuari en la creació i la modificació dels seus itineraris. El nucli d'aquesta proposta és el Model Context Protocol (MCP) [1], [2], un protocol obert que estandarditza la connexió entre els models de llenguatge i les eines o fonts de dades externes, de manera que el model pot anar més enllà de la conversa i executar accions reals sobre el sistema. Sobre aquesta base, l'agent utilitza Claude [3] com a interfície d'alt nivell per interpretar les peticions de l'usuari expressades en llenguatge natural i traduir-les en accions concretes, com ara crear o modificar un viatge.

A més de la integració de la intel·ligència artificial, el projecte busca definir una arquitectura robusta i escalable, basada en una separació clara de responsabilitats entre el frontend, el backend i la base de dades, que en garanteix la mantenibilitat i la seguretat, i en facilita l'evolució futura del sistema.

Amb aquest projecte es pretén aportar una solució digital que faciliti la gestió de viatges a qualsevol persona que

- E-mail de contacte: alex.sero@autonoma.cat
- Menció realitzada: Tecnologies de la Informació
- Treball tutoritzat per: Oriol Cortés Comellas
- Curs 2025/26

n'estigui planificant un, oferint-li una interfície des de la qual pugui crear i consultar el seu viatge de la manera més còmoda possible.

Aquest informe recull tot el procés desenvolupat durant el projecte, començant per la definició dels objectius. A continuació, se situa el treball en el seu context mitjançant l'estat de l'art i es descriuen la metodologia i la planificació que han guiat l'execució del treball. Tot seguit, s'analitzen els requisits necessaris per garantir el correcte funcionament de l'eina. La part central del document es dedica al desenvolupament, on s'exposen les decisions tècniques, les tecnologies emprades i la seva implementació. Posteriorment, es presenten i es discuteixen els resultats obtinguts. Finalment, l'informe es tanca amb les conclusions, que apleguen les reflexions finals, els aprenentatges i possibles línies de millora futura.

2 OBJECTIUS

2.1 Objectiu general

La finalitat del projecte és dissenyar i desenvolupar una aplicació web que permeti planificar i gestionar viatges i els seus itineraris mitjançant un agent d'IA, basat en el Model Context Protocol i utilitzant Claude com a interfície d'alt nivell, per assistir l'usuari en la creació o modificació dels seus viatges.

2.2 Objectius específics

Amb la finalitat d'assolir l'objectiu principal del projecte, es plantegen els següents objectius específics:

- **Reduir el temps dedicat a la planificació de viatges**, proporcionant un espai on gestionar tota la informació relacionada amb el viatge.
- **Facilitar la creació, edició i eliminació d'itineraris**, permetent gestionar viatges, dies i activitats de manera senzilla i intuïtiva.
- **Facilitar la visualització de l'itinerari**, mostrant les activitats sobre un mapa interactiu amb marcadors i rutes que representin l'ordre del recorregut.
- **Minimitzar l'esforç necessari per gestionar viatges mitjançant intel·ligència artificial**, permetent realitzar modificacions a través de llenguatge natural perquè l'agent pugui recomanar i ajudar l'usuari en la creació del viatge.

2.3 Objectius tècnics

Per tal d'assolir l'objectiu principal es plantegen els següents objectius tècnics:

- **Dissenyar una arquitectura moderna amb separació clara de responsabilitats**, estructurant l'aplicació web en capes diferenciades (frontend, backend i base de dades), per garantir l'evolució de les diferents capes sense afectar la resta del sistema.
- **Dissenyar i implementar una base de dades relacional amb PostgreSQL**, definint les entitats principals del sistema i les seves relacions, assegurant la integritat i consistència de les dades.
- **Desenvolupar un backend** que implementi les operacions CRUD (Create, Read, Update, Delete) per gestionar viatges, itineraris i activitats, i que proporcionï una API REST que permeti a l'agent d'IA executar aquestes accions de manera segura.

- **Implementar autenticació mitjançant JSON Web Tokens (JWT)**, així com la validació de les dades d'entrada.
- **Implementar un frontend**, amb una interfície intuïtiva que permeti als usuaris gestionar els seus viatges i interactuar amb l'agent d'intel·ligència artificial de manera senzilla.
- **Integrar un model d'intel·ligència artificial Claude mitjançant el Model Context Protocol (MCP) per executar accions de domini amb registre d'accions**, assistint a l'usuari per executar accions concretes com crear o modificar els seus viatges invocant funcions definides al backend.

3 ESTAT DE L'ART

El creixement del turisme ha impulsat l'aparició de nombroses plataformes digitals per a la gestió i l'organització d'itineraris, que difereixen en funcionalitats, usabilitat i enfocament. Tot i això, moltes d'aquestes eines encara presenten mancances importants, com ara la necessitat que l'usuari introdueixi i organitzi manualment gran part de la informació, un baix nivell d'automatització de les tasques i un aprofitament limitat de les possibilitats que ofereix actualment la intel·ligència artificial.

3.1 Solucions existents

A l'hora de planificar un viatge, es poden distingir dues solucions principals: els planificadors d'itineraris tradicionals i els assistents conversacionals basats en intel·ligència artificial. Tot i que cadascuna resol de manera eficient una part del problema, cap d'elles ofereix una solució integral que combini la gestió estructurada de la informació amb la flexibilitat i la capacitat d'automatització que proporciona la IA.

D'una banda, els planificadors d'itineraris tradicionals, com TripIt [4] o Wanderlog [5], permeten desar els viatges, organitzar-los per dies i afegir-hi activitats, sovint amb visualització sobre mapa. El seu punt feble és que tota l'organització recau en l'usuari, que ha d'omplir formularis i col·locar cada activitat a mà. Algunes ofereixen recomanacions automàtiques, però cap no deixa que un assistent creï o modifiqui el viatge en lloc seu.

D'altra banda, els assistents conversacionals basats en intel·ligència artificial permeten obtenir recomanacions i propostes d'itineraris mitjançant l'ús de llenguatge natural. Tanmateix, aquestes interaccions solen limitar-se a l'entorn conversacional. Tot i que el model genera suggeriments útils, aquests no s'integren de manera automàtica en una estructura persistent, fet que obliga l'usuari a copiar-los i reorganitzar-los manualment en una altra eina. Com a conseqüència, la conversa i la planificació efectiva del viatge queden desconnectades.

3.2 Aportació del projecte

La plataforma proposada en aquest projecte busca combinar les dues aproximacions anteriors; un planificador estructurat i persistent dels viatges, dies i activitats amb visualització sobre mapa i un agent d'intel·ligència artificial que actua directament sobre aquestes dades. Així, a diferència dels planificadors tradicionals, l'usuari pot operar en llenguatge natural i, a diferència dels assistents conversacionals, les accions tenen un efecte real en l'aplicació. D'aquesta manera,

es busca crear una eina completa i accessible, adaptada a les necessitats reals de qualsevol persona que vulgui planificar o organitzar un viatge.

3.3 Agents d'intel·ligència artificial i execució d'accions

El nucli diferencial d'aquest projecte és la capacitat de l'agent d'executar accions reals sobre l'aplicació en lloc de limitar-se a generar text. A diferència d'un model de llenguatge convencional, que només produeix una resposta textual, un agent d'intel·ligència artificial decideix de manera autònoma quines accions cal cridar [6]. El mecanisme que ho fa possible és el function calling; es defineix un conjunt d'eines amb un nom, una descripció i un esquema de paràmetres que el model pot invocar amb una crida estructurada perquè el sistema les executi i en retorni el resultat. Aquesta connexió entre el model i les eines es pot estandaritzar mitjançant el Model Context Protocol (MCP), un protocol obert que separa la lògica de les eines del model d'intel·ligència artificial, facilitant una arquitectura més modular i escalable.

El fet que un agent pugui executar accions reals també amplia els riscos de seguretat. La principal amenaça és la injecció de prompt (prompt injection) [7], que consisteix a introduir instruccions malicioses a l'entrada del model, directament o a través de continguts externs que aquest processa, amb l'objectiu d'alterar-ne el comportament o provocar l'execució d'accions no autoritzades. Aquest risc esdevé especialment rellevant en agents que utilitzen function calling, ja que una injecció de prompt pot induir l'agent a invocar eines que modifiquin l'estat del sistema sense la intenció de l'usuari.

3.4 Tecnologies per al desenvolupament del projecte

A més de revisar les plataformes existents per a la planificació de viatges, també s'han valorat diverses tecnologies per al desenvolupament de la plataforma web.

3.4.1 Frontend

Per a la implementació de la interfície d'usuari, algunes de les opcions més habituals són:

- **Angular:** framework complet mantingut per Google, amb una estructura rígida i orientat a aplicacions empresarials grans; ofereix moltes funcionalitats integrades a canvi d'una corba d'aprenentatge més elevada.
- **React:** llibreria de codi obert molt estesa, amb un ecosistema molt ampli i una gran comunitat; sovint s'acompanya de frameworks com Next.js per afegir funcionalitats com el renderitzat al servidor.
- **Vue.js** [8]: framework progressiu, lleuger i de ràpid aprenentatge, pensat per construir interfícies mitjançant components reactius; s'acostuma a combinar amb Vite [9], una eina de construcció i servidor de desenvolupament.

3.4.2 Backend

Per al desenvolupament del servidor i la lògica de negoci destaquen tres alternatives:

- **Java amb Spring Boot:** solució robusta i molt utilitzada en entorns corporatius, amb tipatge fort i moltes eines integrades, però també més verbosa i pesada.

- **Python (amb Django o FastAPI):** opció molt popular pel seu ecosistema i la seva sintaxi senzilla, especialment estesa en projectes relacionats amb dades i intel·ligència artificial.
- **Node.js** [10]: entorn d'execució de JavaScript amb una arquitectura asíncrona i no bloquejant, adequat per gestionar moltes connexions simultànies i que permet compartir el mateix llenguatge entre el frontend i el backend.

3.4.3 Base de dades

Per a l'emmagatzematge i la gestió de les dades es poden considerar tres opcions:

- **MongoDB:** base de dades NoSQL orientada a documents, amb un esquema flexible que facilita els canvis, però amb menys garanties d'integritat relacional.
- **MySQL:** base de dades relacional molt estesa, fiable i senzilla d'utilitzar, tot i que amb menys funcionalitats avançades que altres alternatives.
- **PostgreSQL** [11]: base de dades relacional robusta, amb suport per a relacions i claus foranes i per a tipus de dades avançats com JSONB; per comunicar-s'hi des del backend es pot emprar node-postgres [12].

4 METODOLOGIA I PLANIFICACIÓ

El desenvolupament d'aquesta aplicació web ha seguit una metodologia àgil basada en Scrum [13], adaptada als terminis i a l'abast del treball. L'elecció de Scrum es deu a la necessitat de dividir el projecte en iteracions curtes que permetin desenvolupar funcionalitats de manera incremental i validar-ne el funcionament de forma progressiva. Aquesta aproximació és especialment adequada en aquest projecte, ja que integra components complexos com un agent d'intel·ligència artificial, que poden requerir ajustos al llarg del desenvolupament.

El desenvolupament s'ha estructurat en sprints, en els quals s'han treballat funcionalitats incrementals i verticals de l'aplicació web, tenint en compte la seva càrrega de treball. En total, s'han definit cinc sprints principals:

Sprint 1: Registre i gestió d'usuaris (del 9 al 22 de març de 2026). Durant aquest sprint s'ha desenvolupat el sistema d'autenticació d'usuaris: s'han dissenyat i implementat la base de dades d'usuaris, el registre, l'inici de sessió i la modificació de perfil, amb la interfície d'usuari corresponent i l'autenticació mitjançant JWT.

Sprint 2: Gestió de viatges i integració de l'agent d'IA (del 23 de març al 12 d'abril de 2026). En aquest sprint s'ha implementat la gestió de viatges, desenvolupant la base de dades i l'API per a les operacions CRUD, juntament amb la seva interfície d'usuari. També s'ha iniciat la integració de l'agent d'intel·ligència artificial.

Sprint 3: Itineraris i activitats (del 13 d'abril al 10 de maig de 2026). S'han desenvolupat els itineraris i les activitats, incloent-hi la base de dades, l'API i la interfície d'usuari per a les operacions CRUD, i s'ha ampliat l'agent perquè pogués gestionar aquests elements.

Sprint 4: Integració d'ajustos i millores del sistema (del 11 al 24 de maig de 2026). S'ha dut a terme una millora global del sistema, centrada especialment en el comportament de l'agent i en l'optimització del consum de tokens.

Sprint 5: Testing i validació del sistema (del 25 al 31 de maig de 2026). S'ha validat el sistema mitjançant proves

de funcionalitat i d'usabilitat, amb l'objectiu de garantir que l'aplicació complís els requisits definits.

El diagrama de Gantt de la planificació del projecte es pot consultar a l'apèndix A.1.

Per a la planificació i el seguiment del desenvolupament s'ha utilitzat Kanban [14] com a eina de suport per a la gestió de tasques, implementat mitjançant la plataforma Trello. Aquesta metodologia permet visualitzar l'estat de cada activitat mitjançant columnes com “per fer”, “en curs” i “fet”, facilitant el control del progrés. L'ús de Kanban es justifica per la seva senzillesa i flexibilitat, ja que no imposa una estructura rígida de treball, sinó que s'adapta al flux real de tasques del projecte. Això ha resultat especialment útil en un entorn com aquest, on les tasques poden evolucionar durant el desenvolupament i on és necessari mantenir una visió contínua i actualitzada de l'estat del projecte.

A l'inici de cada sprint es revisaven les tasques pendents i les realitzades en el sprint anterior, per comprovar si les funcionalitats afegides funcionaven correctament i valorar si calia reordenar les prioritats o afegir-ne de noves no previstes inicialment. Aquest enfocament ha permès adaptar-se als imprevistos sorgits durant el desenvolupament, mantenint alhora un esquema estructurat de les tasques de cada sprint.

5 DESENVOLUPAMENT

En aquest apartat es detalla el procés de desenvolupament del projecte, des de l'anàlisi inicial fins a la implementació final. En primer lloc, s'estableixen els requisits funcionals i no funcionals que ha de complir l'aplicació i es presenten les eines i les tecnologies emprades. A continuació, es descriu el disseny de la base de dades i la implementació de les funcionalitats del sistema, com ara la gestió d'usuaris, viatges, itineraris i activitats. Finalment, s'aprofundeix en l'arquitectura de l'agent d'intel·ligència artificial i el servidor MCP, que constitueix el nucli diferencial del treball, i en les mesures de seguretat adoptades.

5.1 Anàlisi de requisits

En aquest apartat es defineixen els requisits funcionals i no funcionals de l'aplicació web.

5.1.1 Requisits funcionals

L'aplicació web oferirà als usuaris:

- El sistema haurà de permetre el registre i l'autenticació d'usuaris, garantint la seguretat del seu compte.
- L'usuari autenticat haurà de poder consultar i modificar la seva informació personal.
- L'aplicació haurà de permetre crear nous viatges per part dels usuaris.
- Cada viatge haurà de poder contenir itineraris que l'usuari podrà crear, modificar o eliminar.
- Dins de cada itinerari, l'usuari haurà de poder crear, modificar o eliminar activitats, incloent-hi informació relacionada amb el transport i l'allotjament.
- L'usuari haurà de poder consultar un llistat amb els seus viatges creats i visualitzar amb detall la informació de cadascun.
- L'usuari haurà de poder invocar un agent d'intel·ligència artificial que li permeti facilitar la creació i modificació dels seus viatges i itineraris.

- El sistema haurà de permetre confirmar o rebutjar les accions crítiques proposades per l'agent d'IA abans de fer-les efectives.
- El sistema haurà de registrar les accions realitzades per l'agent d'intel·ligència artificial, indicant la data, l'acció i les dades afectades, per garantir traçabilitat i control de les operacions.

5.1.2 Requisits no funcionals

L'aplicació web ha de garantir:

- Usabilitat: la interfície ha de ser clara i intuïtiva, facilitant-ne l'ús als usuaris.
- Rendiment: l'aplicació web ha de proporcionar temps de resposta àgils tant en la càrrega de pàgines com en la interacció amb l'agent d'intel·ligència artificial, garantint una experiència fluida.
- Seguretat: les dades dels usuaris han d'estar protegides mitjançant mecanismes d'autenticació segura (JWT) i xifrat de contrasenyes. El sistema ha de validar totes les dades d'entrada per prevenir injeccions o accessos no autoritzats, i les accions executades per l'agent d'IA han de ser verificades abans de persistir qualsevol modificació.
- Mantenibilitat: el codi s'estructurarà mitjançant bones pràctiques com la separació de responsabilitats per facilitar canvis o modificacions.

5.2 Selecció d'eines i tecnologies

D'entre les tecnologies analitzades en l'estat de l'art, s'han seleccionat les següents per al desenvolupament de l'aplicació, atesa la seva adequació als requisits del projecte:

Backend: s'ha utilitzat Node.js, per la seva arquitectura asíncrona i no bloquejant, que gestiona de manera eficient múltiples connexions simultànies, fet clau en un sistema que es comunica constantment amb serveis externs d'intel·ligència artificial i perquè permet emprar JavaScript tant al frontend com al backend. El backend és també l'encarregat de generar els tokens JWT que assegurin totes les peticions abans de manipular la informació.

Frontend: s'ha optat per Vue.js, davant d'alternatives com Next.js [15], per la familiaritat prèvia amb el framework i perquè permet un desenvolupament ràpid i lleuger, suficient per a la interfície del projecte; a més, els seus components reactius faciliten actualitzar la interfície en temps real segons les respostes de l'agent. S'ha completat amb Vite com a eina de construcció i servidor de desenvolupament.

Base de dades: s'ha triat PostgreSQL per la seva robustesa i per garantir la integritat i la consistència de les dades mitjançant un model relacional, a més del suport de tipus de dades avançats com JSONB. La comunicació amb el backend es fa amb node-postgres, que permet establir connexions segures i executar consultes SQL parametritzades.

Assistent d'intel·ligència artificial: la interacció amb l'usuari mitjançant IA s'ha implementat amb un agent assistent basat en l'MCP que utilitza el model Claude d'Anthropic.

5.3 Disseny de la base de dades

El disseny de la base de dades segueix un model relacional, per garantir la integritat i la coherència de les dades i faci-

litar l'escalabilitat del sistema. El diagrama complet es pot consultar a l'apèndix A.2.

El domini de l'aplicació s'estructura en quatre taules encadenades per claus foranes amb eliminació en cascada. La taula *users* guarda les dades de cada usuari, amb el correu electrònic únic i la contrasenya xifrada amb bcrypt [16]. La taula *trips* conté els viatges, associats a un usuari (*user_id*), amb el nom, la destinació i les dates d'inici i fi. La taula *itinerary_days* representa els dies de cada viatge (*trip_id*), amb la combinació de viatge i data única per evitar dies duplicats. Finalment, la taula *itinerary_activities* desa les activitats de cada dia (*day_id*), amb el nom, l'hora d'entrada, un enllaç opcional, les coordenades geogràfiques (*lat* i *lng*) per a la representació cartogràfica i un índex d'ordenació.

La part de l'agent es recolza en dues taules addicionals. La taula *subagents* defineix els subagents especialitzats: a més del nom, el camp *router_description* l'utilitza l'encaminador per decidir quins subagents atenen cada petició, i el *system_prompt* aporta instruccions específiques que només s'afegeixen al context quan el subagent és seleccionat. La taula *agent_tools* emmagatzema les eines que pot invocar el model, indicant per a cadascuna el subagent al qual pertany (*subagent_id*), el nom i la descripció, la referència al fitxer i a la funció que la implementen, els paràmetres en format JSONB i un indicador per activar-la o desactivar-la.

Finalment, dues taules aporten traçabilitat a les interaccions amb l'agent. La taula *chat_logs* registra cada intercanvi, agrupat per sessió (*session_id*): hi consten la pregunta de l'usuari, la resposta generada (la d'un subagent o la decisió presa per l'encaminador), un resum de les eines emprades i el recompte de tokens d'entrada i de sortida. A diferència de la resta de taules, no aplica eliminació en cascada, de manera que l'historial de converses es preserva. La taula *action_chat_logs*, per la seva banda, detalla cada execució d'eina associada a un intercanvi (*chat_log_id*), amb els arguments amb què s'ha cridat, el resultat, l'estat (èxit o error) i la durada. Conjuntament, ambdues taules permeten auditar i analitzar les accions de l'agent.

5.4 Implementació de les funcionalitats del sistema

En aquest apartat es descriu la implementació de les funcionalitats principals: la gestió d'usuaris, viatges i itineraris amb les seves activitats. Totes segueixen el mateix patró en capes: el frontend envia les dades al backend, on una ruta REST les delega a un controlador que les valida i un model les guarda a la base de dades.

5.4.1 Gestió d'usuaris

Per a la gestió d'usuaris, el backend disposa de tres rutes, gestionades pel controlador *UserController*:

- Registre d'usuari: *POST /api/register*.
- Inici de sessió: *POST /api/login*.
- Modificació de perfil: *POST /api/modifyProfile* (protegida pel middleware *tokenVerify.protect*).

El controlador valida les dades abans de modificar la base de dades: en el registre i la modificació comprova que no existeixi cap usuari amb el mateix correu o nom, i en l'inici de sessió verifica que les credencials coincideixin amb un compte existent. A més, en el registre la contrasenya ha de complir uns criteris mínims de seguretat (almenys sis

caràcters, amb una majúscula, un número i un símbol); si no els compleix, el servidor rebutja la petició i retorna un missatge d'error.

Durant l'inici de sessió, es genera un JWT (JSON Web Token) amb validesa d'una hora. Aquest token s'emmagatzema dins d'una cookie al client per permetre la transmissió segura de la informació privada de l'usuari. En les rutes on es requereix autenticació, es verifica el token mitjançant el middleware *tokenVerify.protect*, que comprova la validesa del token i extreu les dades de l'usuari per autoritzar l'accés a la funcionalitat corresponent.

5.4.2 Gestió de viatges

Per a la gestió dels viatges, els usuaris poden crear, consultar, modificar i eliminar els seus viatges. Per crear un viatge, l'usuari ha d'introduir les dades necessàries, com ara el nom del viatge, la destinació i el rang de dates, mitjançant formularis visuals que utilitzen el component *DatePicker* de *v-calendar* per seleccionar les dates d'inici i fi, com es pot veure en la figura 1.

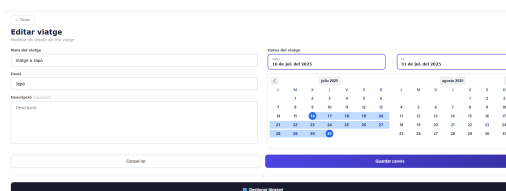


Fig. 1: Vista de la pàgina de creació de viatges.

Quan l'usuari envia el formulari de creació o edició, les dades es formaten automàticament al format *YYYY-MM-DD*, compatible amb la base de dades PostgreSQL, i s'envien al backend mitjançant crides *fetch*, juntament amb la cookie d'autenticació amb el token JWT.

El backend disposa de les rutes REST per gestionar els viatges, totes protegides pel middleware *tokenVerify.protect* i delegades al controlador *tripController*, que valida les dades i les passa al model *tripModel*:

- Consultar tots els viatges de l'usuari: *GET /api/getTrips*.
- Consultar un viatge per ID: *GET /api/getTripsById/:id*.
- Crear un viatge: *POST /api/createTrip*.
- Editar un viatge: *PUT /api/editTrip/:id*.
- Eliminar un viatge: *DELETE /api/deleteTrip/:id*.

Al frontend, la interacció amb aquestes rutes es realitza a través de formularis reactius. Quan l'usuari crea o edita un viatge, el component recull les dades, les valida i crida la ruta corresponent. En cas d'error, el missatge es mostra dins de la interfície; en cas d'èxit, l'usuari és redirigit a la pàgina principal on ja es poden veure els canvis.

5.4.3 Gestió d'itineraris i activitats

Per a la gestió dels itineraris i activitats, els usuaris poden crear, consultar, modificar i eliminar activitats en els diferents dies dins de cada viatge. La interfície s'organitza en tres zones: una barra lateral esquerra amb la navegació per setmanes i dies, una zona central amb un mapa interactiu i una barra lateral dreta amb el llistat d'activitats del dia seleccionat i el formulari per afegir-ne de noves, com es pot veure en la figura 2.

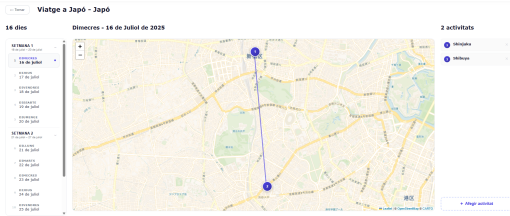


Fig. 2: Vista de la pàgina d'itinerari i activitats.

La navegació per dates es presenta agrupada per setmanes, mostrant tots els dies compresos entre les dates d'inici i fi del viatge. En seleccionar un dia, s'actualitzen tant el mapa com el llistat d'activitats corresponents a aquella data.

Per afegir una activitat, l'usuari obre un formulari modal on introdueix el nom de l'activitat, una descripció opcional, una hora d'entrada (*entry_time*), un enllaç opcional i una localització. Per obtenir les coordenades geogràfiques, el component realitza cerques a l'API de Nominatim d'OpenStreetMap, que retorna fins a cinc resultats per a la consulta introduïda. Un cop seleccionat el lloc, les coordenades de latitud i longitud s'emmagatzemen associades a l'activitat i, si el camp nom és buit, s'omple automàticament amb el nom del lloc.

El mapa interactiu s'implementa mitjançant Leaflet.js a través de la llibreria Vue-Leaflet, amb capes de mapa de CartoDB Voyager. Per a cada activitat del dia seleccionat es mostra un marcador numerat circular que indica l'ordre de l'activitat, i en clicar-lo apareix una finestra emergent amb el nom i la descripció. Si hi ha dues activitats o més, es dibuixa una línia que les connecta en ordre, facilitant la visualització del recorregut durant el dia.

El backend disposa de les rutes REST per gestionar els dies de l'itinerari, totes protegides pel middleware *tokenVerify.protect* i delegades al controlador *itineraryDaysController*:

- Consultar tots els dies d'un viatge: `GET /api/trips/:tripId/days`.
- Consultar un dia per ID: `GET /api/days/:id`.
- Crear un dia: `POST /api/trips/:tripId/days`.
- Editar un dia: `PUT /api/days/:id`.
- Eliminar un dia: `DELETE /api/days/:id`.

I les rutes per gestionar les activitats, delegades al controlador *itineraryActivitiesController*:

- Consultar totes les activitats d'un dia: `GET /api/days/:dayId/activities`.
- Consultar una activitat per ID: `GET /api/activities/:id`.
- Crear una activitat: `POST /api/days/:dayId/activities`.
- Editar una activitat: `PUT /api/activities/:id`.
- Eliminar una activitat: `DELETE /api/activities/:id`.

Totes les rutes actuen com a punts d'entrada que criden els seus controladors respectius, els quals validen les dades, mentre que els models s'encarreguen d'interactuar amb la base de dades.

5.5 Arquitectura de l'agent d'IA i el servidor MCP

El sistema d'intel·ligència artificial s'ha implementat seguint l'arquitectura del Model Context Protocol (MCP), que estableix una separació clara entre el model de llenguatge i

les eines que aquest pot invocar. S'articula al voltant de dos components principals: l'agent d'IA, que s'executa dins del backend i orquestra tota la conversa amb el model, i el servidor MCP, un procés independent que exposa i executa les eines. En termes del protocol, la relació entre tots dos és de tipus client-servidor: l'agent actua com a client MCP, que es connecta al servidor per descobrir i invocar les eines, mentre que el servidor MCP n'és el proveïdor. L'agent incorpora, a més, un encaminador que decideix quins subagents especialitzats han d'atendre cada petició. A l'apèndix A.3 es pot veure una visió general d'aquesta arquitectura.

5.5.1 Definició i organització de les eines

L'adopció del MCP respon a la voluntat de desacoblar la lògica de conversa de l'execució d'accions. En separar l'agent (client) del servidor d'eines (tools), es poden afegir, modificar o desactivar capacitats sense alterar el nucli de l'agent. Sobre aquesta base, s'han pres dues decisions de disseny rellevants: com es defineixen les eines i com s'organitzen en subagents.

La primera decisió afecta la manera de definir les eines que pot invocar el model Claude d'Anthropic. Es van valorar dues alternatives. La primera consistia a declarar-les directament al codi, amb la descripció i els paràmetres de cada funció. La segona, finalment escollida, consisteix a emmagatzemar-les a la taula *agent_tools* de la base de dades, on cada registre recull el nom de l'eina, la seva descripció, l'esquema de paràmetres i la referència al fitxer i a la funció que la implementen. Aquest enfocament permet afegir, modificar o desactivar eines sense tocar el nucli de l'agent i evita concentrar tota la informació en un únic fitxer; en conjunt, ofereix una gestió més escalable i facilita el manteniment i l'evolució del sistema.

La segona decisió consisteix a agrupar les eines en subagents especialitzats i introduir un encaminador. Cada subagent (per exemple, el de viatges, el d'itineraris o el de generació de PDF) reuneix un conjunt d'eines afins i es defineix a la taula *subagents*. L'encaminador és una capa lleugera que, per a cada missatge, selecciona únicament els subagents necessaris per atendre'l. Aquesta decisió aborda un problema pràctic dels agents amb moltes eines: el cost en tokens. Enviar al model les definicions de totes les eines a cada petició encareix i alenteix la interacció; en canvi, carregar només les eines dels subagents seleccionats redueix el nombre de tokens d'entrada per missatge sense renunciar a cap capacitat.

5.5.2 Agent d'IA i client MCP

El flux d'execució s'inicia quan l'usuari envia un missatge en llenguatge natural des del frontend. La interfície del xatbot es pot observar a la figura 3. El missatge arriba al backend mitjançant una crida `POST` a la ruta `/api/chat`, protegida pel middleware d'autenticació, i la resposta es retorna de manera incremental mitjançant Server-Sent Events (SSE), de manera que el text es mostra a l'usuari a mesura que es genera. El controlador *aiController* extreu la identitat de l'usuari a partir del token JWT, recupera el mateix token per autenticar les crides posteriors, i delega la petició a l'agent.

L'agent és l'orquestrador central del sistema. En primer lloc, carrega l'historial de la sessió actual des de la taula *chat_logs*, recuperant els darrers intercanvis per recons-

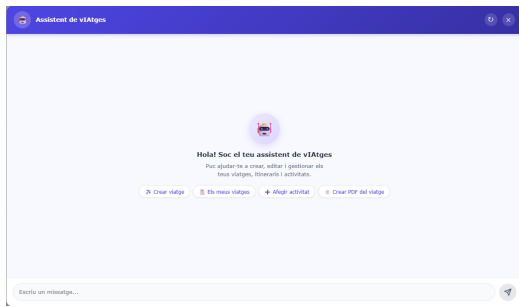


Fig. 3: Vista de la pàgina del xatbot.

truir el context de la conversa. A continuació, i de manera paral·lela, executa dues tasques. D'una banda, actuant com a client MCP, obre una sessió amb el servidor MCP mitjançant *createMcpSession*, que estableix una connexió via *StreamableHTTPClientTransport* i hi passa el token JWT de l'usuari a la capçalera d'autorització; un cop oberta, l'agent pot consultar les eines que ofereix el servidor i invocar-les. De l'altra, invoca l'encaminador descrit a l'apartat anterior, que retorna els subagents necessaris per a la petició. A partir d'aquests subagents, l'agent construeix el prompt de sistema (combinant un prompt base comú amb les instruccions específiques, el *system_prompt*, de cada subagent seleccionat) i filtra les eines que oferirà al model, convertint-les al format que requereix l'API d'Anthropic.

Tot seguit s'inicia el bucle de l'agent: el missatge de l'usuari, juntament amb l'historial, el prompt de sistema i la llista d'eines, s'envia a l'API d'Anthropic utilitzant el model *claude-haiku-4-5*. L'elecció d'aquest model es deu al fet que les tasques que ha de realitzar l'agent (interpretar accions en llenguatge natural i invocar les eines adequades per executar operacions CRUD ben definides) no requereixen les grans capacitats de raonament de models com Sonnet o Opus; per tant, és preferible utilitzar Haiku, que ofereix temps de resposta més ràpids i un cost per token significativament inferior. Tot i això, el model és configurable mitjançant una variable d'entorn, de manera que es pot substituir per un de més potent sense modificar el codi, cosa que facilita el manteniment i l'evolució futura del sistema. El model analitza el missatge i determina si pot generar una resposta directament o si cal invocar alguna eina. Si el *stop_reason* és *end_turn*, el model ha generat la resposta final i el bucle s'acaba. Si és *tool_use*, el model ha decidit invocar una o més eines: l'agent extreu els blocs de tipus *tool_use* de la resposta, crida cada eina al servidor MCP mitjançant *mcp.callTool*, registra el resultat i les dades de cada execució (arguments, estat i durada) i afegeix els resultats al context de la conversa com a missatges de tipus *tool_result*. El bucle continua, amb un límit màxim d'iteracions per evitar bucles infinits, fins que el model genera una resposta final en llenguatge natural. Per reduir encara més el cost, el context acumulat s'envia amb memòria cau (prompt caching), de manera que les parts que es repeteixen entre crides no es tornen a facturar com a tokens nous.

5.5.3 Servidor MCP

El servidor MCP és la contrapart del client: s'executa com un procés independent al port 3002. En arrencar, carrega totes les eines actives de la taula *agent_tools* i les re-

gistra dinàmicament, construint l'esquema de validació de paràmetres amb Zod a partir de la definició JSON emmagatzemada a la base de dades. Cada petició al servidor porta associat un identificador de sessió i el token JWT de l'usuari, que es desa per a aquella sessió i s'utilitza per fer les crides al backend amb les credencials correctes. Quan el model invoca una eina, el servidor MCP executa la funció corresponent, que fa una crida a l'API REST del backend, i en retorna el resultat al model.

5.5.4 Traçabilitat d'accions

Un cop finalitzat el bucle de l'agent, el controlador *aiController* desa l'intercanvi a la taula *chat_logs* (la pregunta de l'usuari, la resposta de l'agent, el resum de les eines invocades i el recompte de tokens d'entrada i sortida) i, a la taula *action_chat_logs*, el detall de cada acció executada, amb els arguments amb què s'ha cridat l'eina, el resultat retornat, l'estat (èxit o error) i la durada. Aquest doble registre permet la traçabilitat de les operacions i el monitoratge del cost en tokens. Finalment, la resposta es retorna al frontend, on es mostra a l'usuari dins la interfície del xatbot.

5.6 Seguretat

La seguretat ha estat una consideració present al llarg de tot el desenvolupament del sistema. Les mesures adoptades es poden agrupar en dos blocs. El primer és el de la seguretat habitual de qualsevol aplicació web, que cobreix l'autenticació, l'autorització i la defensa davant els atacs més comuns. El segon respon als riscos propis d'incorporar un agent d'IA capaç d'executar accions reals sobre el sistema.

5.6.1 Seguretat de l'aplicació web

L'autenticació es basa en JSON Web Tokens (JWT), que es guarden en cookies protegides amb tres flags. La flag *HttpOnly* evita que el token es pugui llegir des de JavaScript i, per tant, el protegeix davant els atacs de Cross-Site Scripting (XSS). La flag *Secure* fa que la cookie només viatgi per connexions HTTPS. I la flag *SameSite=Strict* impedeix que altres llocs web facin peticions en nom de l'usuari, una defensa contra els atacs de Cross-Site Request Forgery (CSRF). Més enllà de l'autenticació, s'han pres altres mesures:

- La configuració de CORS restringeix l'accés al backend a un únic origen autoritzat, de manera que cap altre domini no pot fer crides a l'API.
- Les consultes a la base de dades es fan amb sentències preparades i paràmetres, cosa que evita les injeccions SQL.
- Les contrasenyes no es desen mai en clar: es xifren amb *bcrypt* abans de guardar-les a la base de dades.
- Abans d'enviar qualsevol resposta al client, es filtren les dades sensibles, perquè no surtin mai del servidor.
- Al frontend, les rutes estan protegides amb guàrdies de navegació; quan algú intenta entrar en una ruta que requereixi autenticació, es comprova amb el backend si la sessió és vàlida i, si no ho és, se'l redirigeix a la pàgina d'inici de sessió. Així, un usuari no autenticat no pot accedir a les vistes restringides.
- L'autorització treballa a nivell d'objecte: totes les operacions del backend filtren les dades per l'identificador de l'usuari autenticat, que s'extreu del token JWT.

D'aquesta manera, ningú no pot consultar ni modificar viatges, dies o activitats que no siguin seus.

5.6.2 Seguretat de l'agent d'IA

Que l'agent d'IA pugui executar accions reals obre un front de seguretat que una aplicació convencional no té, i per controlar-lo s'han combinat diverses mesures. La principal, que aplica el principi de mínim privilegi, és que l'agent no disposa de privilegis propis; el servidor MCP rep el token JWT de l'usuari i el fa servir per a totes les crides al backend, de manera que l'agent actua amb els mateixos permisos que l'usuari i el backend torna a comprovar l'autorització a nivell d'objecte en cada operació. Això protegeix el sistema davant els atacs de prompt injection, ja que encara que algú aconseguís colar una instrucció maliciosa a través de les dades i l'agent l'obeís, no podria escalar privilegis ni accedir a les dades d'altres usuaris.

A més, l'agent no pot executar codi ni consultes arbitràries: només actua a través d'un conjunt tancat d'eines del servidor MCP, cadascuna lligada a una operació del backend que aplica les mateixes validacions que la resta de l'aplicació, i l'encaminador només n'exposa el subconjunt necessari a cada petició. Finalment, s'incorpora la confirmació humana (human-in-the-loop): davant d'operacions sensibles o irreversibles, l'agent no les executa directament, sinó que en proposa la realització i n'espera l'aprovació explícita de l'usuari, que manté així el control total sobre l'aplicació.

6 PRESENTACIÓ I DISCUSSIÓ DE RESULTATS

En aquest apartat es valora el grau d'assoliment dels objectius plantejats a l'inici del projecte.

6.1 Assoliment dels objectius

Els dos primers objectius específics, **reduir el temps de planificació** mitjançant un espai únic que centralitza tota la informació i **facilitar la creació, l'edició i l'eliminació d'itineraris**, s'han assolit de manera satisfactòria. Un cop registrat, l'usuari pot definir els seus viatges amb la destinació i les dates corresponents i organitzar les activitats de cada dia des d'una mateixa interfície, de manera que la gestió queda centralitzada en un únic entorn.

Aquest resultat se sustenta en els objectius tècnics que el fan possible: una arquitectura en capes (frontend, backend i base de dades), una base de dades relacional implementada en PostgreSQL i un backend amb una API REST que cobreix la totalitat de les operacions CRUD. L'autenticació basada en JWT i la verificació de la identitat en cada petició garanteixen que cada usuari accedeixi únicament a les seves dades, mentre que el frontend en proporciona una interfície d'ús intuïtiu.

El tercer objectiu, **millorar la visualització de l'itinerari diari**, s'ha resolt igualment des del frontend. Mitjançant la geocodificació proporcionada per l'API de Nominatim i la representació cartogràfica de Leaflet, integrada a través de Vue-Leaflet, l'usuari pot cercar qualsevol lloc pel seu nom i visualitzar les activitats de cada dia amb marcadors numerats i el recorregut connectat sobre el mapa.

6.2 Integració i avaluació de l'agent d'IA

El quart objectiu específic, **minimitzar l'esforç de gestió mitjançant intel·ligència artificial**, es correspon directament amb l'objectiu tècnic d'integrar Claude mitjançant el Model Context Protocol i constitueix l'aportació més rellevant del treball. L'usuari pot gestionar els seus viatges i activitats en llenguatge natural, sense haver de navegar pels formularis, i fins i tot generar un document PDF complet de l'itinerari. Tal com es preveia en els objectius tècnics, l'agent executa accions reals sobre el domini reutilitzant la mateixa API REST segura del backend i en deixa registre, cosa que en garanteix la traçabilitat. La part tècnica més complexa ha estat el disseny del bucle de l'agent, que carrega les eines dinàmicament des del servidor MCP, les passa al model i gestiona la crida de múltiples eines.

6.2.1 Validació i proves

Per validar el comportament de l'agent s'han dut a terme proves funcionals sobre les operacions principals, totes expressades en llenguatge natural: la creació, la consulta i la modificació de viatges, dies i activitats, la generació de PDF i el tractament de casos amb dades incorrectes. La taula 1 en recull alguns casos representatius, amb la petició de l'usuari, les eines que l'agent ha invocat per resoldre-la i el resultat obtingut.

TAULA 1: CASOS DE PROVA DE L'AGENT.

Petició	Eines invocades	Resultat
Mostrar viatges	getTrips	Llista correcta
Crear un viatge	createTrip	Viatge creat
Crear itinerari	createDay, createActivity	Itinerari creat
Generar un PDF	getTripByName, generate-TripPDF	PDF generat

Com a exemple real, la conversa següent mostra com l'agent gestiona una petició amb dades incoherents:

Usuari: Crea un viatge a Londres del 5 al 4 de maig de 2027

Agent: Crec que hi ha una confusió amb les dates: el 5 de maig és posterior al 4 de maig. Vols dir del 4 al 5 de maig?

Usuari: Si

Agent: Viatge a Londres creat correctament per als dies 4-5 de maig de 2027.

Les proves també confirmen que l'agent encadena correctament múltiples eines per resoldre una sola petició. Per exemple, davant la instrucció "crea la ruta de divuit dies pel viatge a Indonèsia", l'agent va generar de manera autònoma un itinerari complet de divuit dies per Indonèsia, invocant en una sola interacció les eines *createDay* i *createActivity*.

Durant les proves també s'han detectat algunes limitacions del comportament de l'agent. En alguns casos demanava dades que ja tenia disponibles o, ocasionalment, afirmava no disposar d'una eina que en realitat existia. Aquest comportament depèn en bona part de la qualitat del prompt i de les descripcions de les eines, i no està garantit de manera estricta en tots els casos.

6.2.2 Temps de resposta

Quant als temps de resposta, les operacions simples com mostrar els viatges o crear un viatge es resolen entre 3 i 5

segons; la generació d'un PDF triga entre 15 i 20 segons; i les operacions més intensives, com la creació d'un itinerari complet amb desenes d'activitats, poden allargar-se fins a un minut, ja que impliquen un gran nombre de crides a les eines.

També s'ha avaluat el rendiment de l'agent sota càrrega concurrent, simulant que diversos usuaris envien alhora la mateixa petició de creació d'un viatge. S'han provat nivells d'1 a 100 peticions simultànies, mesurant per a cadascun el temps mínim, el mitjà, el màxim i el percentil 95 (el valor per sota del qual queden el 95% de les peticions), tal com recull la taula 2. Tant el temps mínim (3,7–4,5 segons) com el mitjà (4–5 segons) es mantenen pràcticament constants en tots els nivells, i cap petició no falla ni amb 100 usuaris simultanis; només una part reduïda, reflectida en el percentil 95, s'allarga fins a prop de 10 segons amb la càrrega més alta. Dins del rang provat, doncs, el sistema absorbeix la concurrència sense degradar-se de manera apreciable, un comportament coherent amb l'arquitectura asíncrona i no bloquejant de Node.js, que permet atendre altres peticions mentre cadascuna espera la resposta de l'API d'Anthropic.

TAULA 2: TEMPS DE RESPOSTA DE L'AGENT SEGONS EL NOMBRE DE PETICIONS (EN SEGONS).

Usuaris	Mín	Mitjana	Màx	p95
1	4,03	4,03	4,03	4,03
5	4,49	4,83	5,83	5,83
10	3,89	4,36	5,30	5,30
20	3,71	4,78	8,00	5,45
50	4,03	4,88	7,98	7,30
100	3,95	5,12	9,75	6,68

Aquest comportament sota càrrega també depèn del servidor MCP, que és el punt pel qual passen totes les invocacions d'eines abans d'arribar a l'API REST del backend i a la base de dades. Quan s'atenen moltes converses alhora, totes les seves crides a eines conflueixen en aquest únic procés independent (al port 3002), que es converteix en un possible punt de saturació. Tanmateix, com el backend, el servidor MCP s'executa sobre Node.js amb una arquitectura asíncrona i no bloquejant; mentre una invocació espera la resposta del backend, en pot atendre moltes altres de manera concurrent. Per això, tot i concentrar totes les crides a eines de totes les peticions simultànies, el servidor MCP no es va convertir en un coll d'ampolla durant les proves de càrrega i no va degradar els temps de resposta.

6.2.3 Optimització i cost del consum de tokens

Un dels reptes principals d'aquest tipus de sistema és el cost en tokens. Com que el model no manté estat entre crides, cada petició ha d'incloure de nou el prompt de sistema, l'historial de la conversa i les definicions de totes les eines disponibles; aquestes últimes creixen amb el nombre de funcionalitats i poden arribar a dominar el cost d'entrada. Per contenir-ho s'ha introduït l'encaminador, que per a cada missatge selecciona únicament els subagents pertinents i carrega només les definicions de les seves eines, descartant la resta.

Per quantificar l'impacte d'aquesta estratègia, s'ha comparat el nombre de tokens d'entrada de peticions equivalents en dos escenaris: carregant les definicions de totes les eines disponibles i carregant només les del subagent seleccionat

per l'encaminador. La taula 3 en recull alguns exemples representatius. En una petició de creació d'un viatge, per exemple, carregar totes les eines suposa uns 10.906 tokens d'entrada, mentre que amb l'encaminador la xifra baixa fins als 7.329. La reducció es manté en la resta de casos i oscil·la entre un 33% en les peticions més simples i prop d'un 68% en la generació d'un itinerari complet, sense que el resultat de l'operació se'n vegi afectat. A més, el cost afegit per la mateixa decisió de l'encaminador és menyspreable, ja que es resol amb una crida lleugera d'uns pocs centenars de tokens. Finalment, a aquesta optimització s'hi suma la memòria cau del context (prompt caching), que evita tornar a facturar les parts del prompt que es repeteixen entre crides successives i en redueix encara més el cost efectiu.

TAULA 3: TOTAL DE TOKENS D'ENTRADA PER PETICIÓ.

Petició	Totes les eines	Subagents
Crear un viatge	10.906	7.329
Mostrar tots els viatges	7.078	4.224
Generar un PDF	13.900	6.121
Crear un itinerari complet	21.842	6.950

Pel que fa al cost econòmic, l'agent utilitza el model claude-haiku-4-5, la tarifa del qual és, aproximadament, de 0,92 € per milió de tokens d'entrada i 4,60 € per milió de tokens de sortida. Prenent els tokens d'entrada de la taula 3 (amb l'encaminador) i afegint-hi els de sortida de cada resposta, el cost per interacció es manté en uns pocs cèntims d'euro. Així, mostrar els viatges (4.224 tokens d'entrada i 417 de sortida) costa al voltant de 0,006 €, la generació d'un PDF (6.121 d'entrada i 2.541 de sortida) ronda els 0,017 €, i fins i tot la creació d'un itinerari complet de divuit dies (6.950 d'entrada i 10.933 de sortida) no supera els 0,06 €. En conjunt, l'encaminament d'eines i la memòria cau del context mantenen el cost del sistema baix, sense renunciar a la capacitat de l'agent d'executar accions reals.

6.3 Limitacions

El sistema també presenta algunes limitacions. El comportament de l'agent depèn en bona part de la qualitat del prompt de sistema i de la definició de les eines, i pot requerir ajustos per a casos d'ús poc habituals. A més, tant la generació de PDF com les respostes de l'agent depenen d'un servei extern (l'API d'Anthropic), cosa que introdueix una dependència de connectivitat i un cost per ús.

7 CONCLUSIONS

El projecte ha assolit els objectius principals plantejats a l'inici. S'ha dissenyat i implementat una aplicació web per a la planificació de viatges, estructurada en capes ben diferenciades i amb un agent d'intel·ligència artificial integrat mitjançant el Model Context Protocol. L'usuari disposa d'una única plataforma on gestionar tota la informació del seu viatge (dates, destinació i activitats de cada dia), amb una interfície senzilla i intuïtiva, la visualització del recorregut sobre un mapa interactiu i la possibilitat d'operar en llenguatge natural a través de l'agent, que executa accions reals sobre les seves dades.

Des del punt de vista tècnic, el treball demostra que és possible integrar un agent basat en MCP en una aplicació

web de domini específic de manera eficient; el repte del consum de tokens s'ha abordat amb l'encaminament d'eines i la memòria cau del context, i les capacitats de l'agent s'han ampliat fins a la generació autònoma de documents.

Pel que fa a la metodologia, l'enfocament àgil basat en Scrum, combinat amb un tauler Kanban a Trello, ha proporcionat un marc de treball flexible i adequat. La divisió del treball en sprints ha permès implementar les funcionalitats de manera incremental i detectar i corregir problemes de forma progressiva, alhora que ha facilitat adaptar-se als imprevistos: la integració de l'agent, per exemple, va requerir més temps del previst i es va haver d'allargar fins a l'inici del tercer sprint.

Malgrat aquests resultats, cal fer una valoració crítica de les limitacions del sistema. En primer lloc, l'agent depèn d'un servei extern, l'API d'Anthropic, fet que introdueix una dependència de connectivitat i de la disponibilitat i les condicions d'un proveïdor de tercers; tot i que el model és configurable mitjançant una variable d'entorn, cosa que en facilita la substitució, el funcionament del nucli d'intel·ligència artificial queda condicionat a un servei que no es controla. En segon lloc, aquesta dependència comporta un cost per ús: encara que l'encaminament d'eines i la memòria cau del context el mantenen baix, és un cost recurrent que creix amb el volum d'ús i amb la complexitat de les operacions, de manera que un desplegament a gran escala n'hauria de tenir en compte la sostenibilitat econòmica. Finalment, el comportament de l'agent no és del tot determinista: davant d'instruccions ambigües o incompletes pot interpretar malament la petició, demanar dades que ja té o, ocasionalment, afirmar que no disposa d'una eina existent. Aquestes limitacions són inherents als sistemes basats en models de llenguatge i, tot i que es mitiguen amb un bon disseny del prompt i amb la validació de les accions, no es poden eliminar del tot.

7.1 Línies de millora i treball futur

Tot i haver assolit els objectius, el sistema deixa diverses vies obertes de millora i de continuïtat:

- Refinar el prompt de sistema i les descripcions de les eines per millorar la precisió de l'agent davant peticions ambigües i reduir les crides innecessàries al model.
- Incorporar la compartició de viatges entre usuaris i la col·laboració en temps real sobre un mateix itinerari.
- Posar l'aplicació en producció en un entorn accessible públicament i amb connexió segura mitjançant HTTPS, perquè estigui disponible per al públic.
- Substituir l'API d'Anthropic per un model d'intel·ligència artificial executat localment, cosa que eliminaria la dependència d'un proveïdor extern i el seu cost per ús.

ÚS DE LA IA GENERATIVA

L'ús de la intel·ligència artificial generativa al llarg del projecte s'ha limitat a tasques de suport. En l'elaboració de l'informe, s'ha emprat per a la correcció i la millora de la redacció; en el desenvolupament de l'aplicació, per resoldre problemes puntuals de programació. En ambdós punts s'ha utilitzat com a eina d'ajuda, revisant i validant sempre els resultats, i mai per generar contingut de manera autònoma. Cal remarcar, a més, que la intel·ligència artificial consti-

tueix també el nucli funcional del producte desenvolupat, on s'integra com a agent d'execució d'accions mitjançant el Model Context Protocol.

AGRAÏMENTS

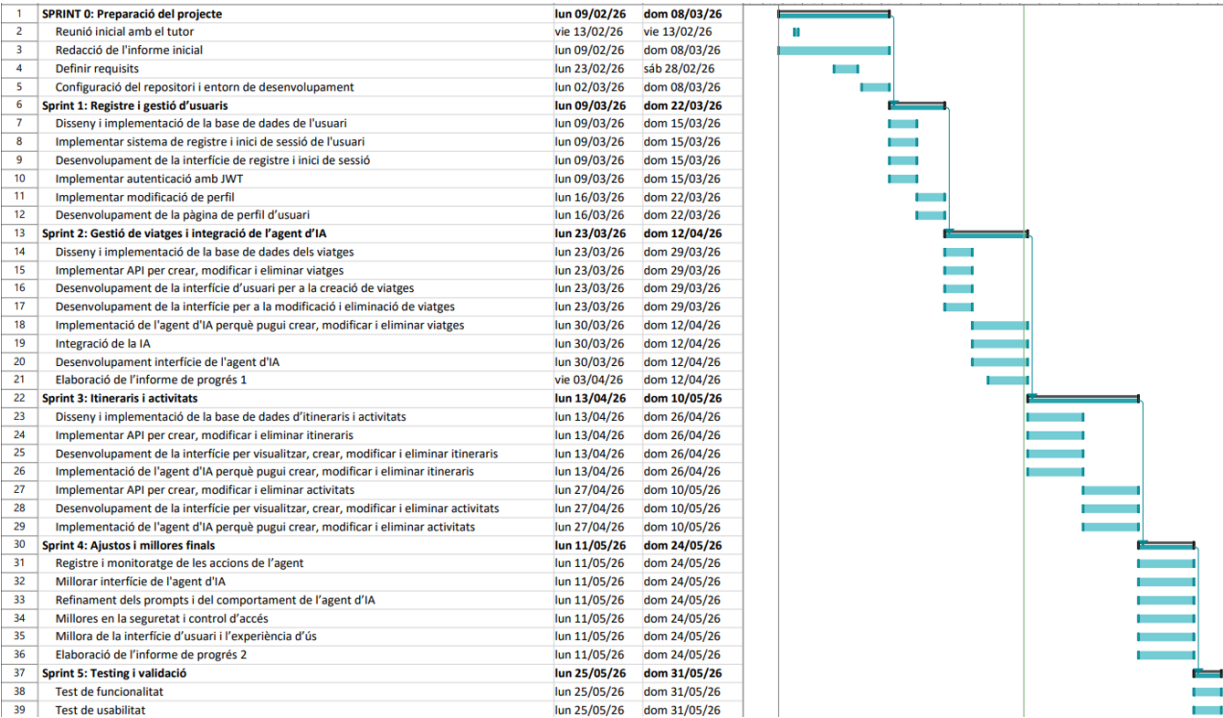
Vull agrair al tutor del treball, Oriol Cortés Comellas, el seguiment i l'orientació al llarg de tot el projecte, així com al Jordi Pons Aróztegui, que va assumir la tutoria durant el període en què l'Oriol no en va poder fer el seguiment.

REFERÈNCIES

- [1] A. Ehtesham, A. Singh, G. K. Gupta i S. Kumar, "A Survey of Agent Interoperability Protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)," arXiv preprint arXiv:2505.02279, 2025. [En línia]. Disponible a: <https://arxiv.org/abs/2505.02279>.
- [2] N. Krishnan, *Model Context Protocol for LLMs: Build Secure, Scalable, and Context-Aware AI Agents Using a Standardized Protocol*.
- [3] Anthropic, "Claude Developer Platform Documentation," Claude Docs. [En línia]. Disponible a: <https://platform.claude.com/docs/en/home>.
- [4] TripIt, "TripIt: The highest-rated trip planner and flight tracker," TripIt. [En línia]. Disponible a: <https://www.tripit.com/>.
- [5] Wanderlog, "Wanderlog: Travel itinerary and road trip planner," Wanderlog. [En línia]. Disponible a: <https://wanderlog.com/>.
- [6] L. Wang, C. Ma, X. Feng et al., "A Survey on Large Language Model based Autonomous Agents," *Frontiers of Computer Science*, vol. 18, núm. 6, 2024. [En línia]. Disponible a: <https://arxiv.org/abs/2308.11432>.
- [7] K. Greshake et al., "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," a Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023. [En línia]. Disponible a: <https://arxiv.org/abs/2302.12173>.
- [8] Vue.js, "Introduction," Vue.js. [En línia]. Disponible a: <https://vuejs.org/guide/introduction.html>.
- [9] Vite, "Introduction," Vite. [En línia]. Disponible a: <https://vite.dev/>.
- [10] OpenJS Foundation, "Node.js Documentation," Node.js. [En línia]. Disponible a: <https://nodejs.org/docs/latest/api/>.
- [11] PostgreSQL, PostgreSQL Documentation. [En línia]. Disponible a: <https://www.postgresql.org/docs/>.
- [12] Node-postgres, node-postgres Documentation. [En línia]. Disponible a: <https://node-postgres.com/>.
- [13] Atlassian, "¿Qué es scrum?" Atlassian. [En línia]. Disponible a: <https://www.atlassian.com/es/agile/scrum>.
- [14] Atlassian, "Kanban," Atlassian. [En línia]. Disponible a: <https://www.atlassian.com/es/agile/kanban>.
- [15] Next.js, "Introduction," Next.js. [En línia]. Disponible a: <https://nextjs.org/>.
- [16] bcrypt, npm package. [En línia]. Disponible a: <https://www.npmjs.com/package/bcrypt>.

APÈNDIX

A.1 Diagrama de Gantt de la planificació del projecte



A.2 Diagrama de base de dades



A.3 Arquitectura de l'assistent d'IA

