



---

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

Rosa Playà, Francesc Manel. *WhatStats*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

---

This version is available at <https://ddd.uab.cat/record/334403>

under the terms of the  license.

# WhatStats: anàlisi i visualització privada de xats de WhatsApp

Francesc Manel Rosa Playà

26 de juny de 2026

**Resum**– Aquest article presenta el disseny, la implementació i la preparació per al desplegament de WhatStats, una aplicació web i mòbil destinada a analitzar xats exportats de WhatsApp i generar estadístiques agregades de manera visual. El projecte parteix d'una necessitat clara: obtenir informació sobre patrons de comunicació sense haver de confiar els historials personals a serveis externs amb garanties de privadesa incertes. La solució proposada combina un *backend* amb FastAPI i Pandas, capaç de processar fitxers `.txt` i `.zip` íntegrament en memòria, amb un *frontend* desenvolupat amb Vue.js, Quasar, ECharts i Capacitor. Els resultats mostren que el sistema permet importar un xat, parsejar-lo, calcular mètriques generals, temporals, lèxiques i d'activitat, i representar-les mitjançant una interfície *responsive* amb opcions addicionals d'anonimització, mode fosc i *demo* local. També s'ha desplegat el *backend* de forma segura amb Docker i Cloudflare Tunnel, mantenint com a principi central la privadesa per disseny.

**Paraules clau**– WhatsApp, estadística de xats, privadesa per disseny, FastAPI, Vue.js, Quasar, Pandas, visualització de dades, Cloudflare Tunnel, Capacitor.

**Abstract**– This paper presents the design, implementation and production-oriented preparation of WhatStats, a web and mobile application for analysing exported WhatsApp chats and producing aggregated visual statistics. The project addresses a practical need: obtaining insights about communication patterns without uploading personal conversation histories to third-party services with uncertain privacy guarantees. The proposed system combines a FastAPI and Pandas backend, able to process `.txt` and `.zip` files entirely in memory, with a frontend developed with Vue.js, Quasar, ECharts and Capacitor. The results show that the system supports the main workflow: importing a chat, parsing it, computing general, temporal, lexical and activity metrics, and displaying them through a *responsive* interface with additional anonymisation, dark mode and local demo features. The backend has also been deployed safely using Docker and Cloudflare Tunnel, while preserving privacy by design as the main architectural principle.

**Keywords**– WhatsApp, chat analytics, privacy by design, FastAPI, Vue.js, Quasar, Pandas, data visualisation, Cloudflare Tunnel, Capacitor.

## 1 INTRODUCCIÓ I CONTEXT DEL TREBALL

WHATSTATS és un Treball de Fi de Grau orientat al desenvolupament d'una aplicació capaç d'analitzar xats exportats de WhatsApp i presentar-ne resultats estadístics de manera clara, interactiva i accessible. La motivació principal del projecte neix de l'ús quoti-

dià i extensiu de les aplicacions de missatgeria instantània. Aquestes eines acumulen una gran quantitat d'informació sobre hàbits comunicatius, participació, moments d'activitat, ús d'emojis, longitud dels missatges o ratxes de conversa. Tanmateix, aquesta informació no és fàcilment interpretable a partir del fitxer d'exportació que ofereix WhatsApp, ja que aquest es presenta com un document textual pla, pensat més per a la conservació o migració del contingut que no pas per a l'anàlisi.

El problema tècnic i ètic que aborda aquest treball és doble. D'una banda, cal transformar un fitxer textual heterogeni, provinent de diferents plataformes i amb formats d'exportació variables, en una estructura de dades consistent

- E-mail de contacte: manelrosaplaya@gmail.com
- Menció: Tecnologies de la Informació
- Treball tutoritzat per: Domènec Madrid i Jordi Herrera
- Curs 2025/26

i apta per al càlcul estadístic. De l'altra, cal fer-ho sense comprometre la privadesa de l'usuari. Els historials de conversa poden contenir dades personals, informació sensible, noms de tercers, ubicacions, opinions privades o elements emocionals que no haurien de persistir en un servidor ni ser tractats per serveis externs sense control. Per aquest motiu, el projecte adopta com a principi rector la privadesa per disseny, d'acord amb els criteris formulats per Cavoukian [6].

La proposta consisteix en una arquitectura client-servidor desacoblada. El *backend* actua com a motor de processament i anàlisi: rep el fitxer exportat, el llegeix en memòria, el parseja, en calcula mètriques agregades i retorna un objecte JSON al client. El *frontend*, per la seva banda, és responsable de la interacció amb l'usuari, la pujada del fitxer, la recepció de les dades i la seva representació visual. Aquesta separació permet que el motor analític evolucioni de manera independent respecte de la capa de visualització i facilita la incorporació de nous gràfics, nous filtres o noves plataformes sense haver de reescriure la lògica de *parsing*.

La Figura 1 resumeix l'arquitectura conceptual de WhatStats, basada en una aplicació web/mòbil que envia fitxers al *backend* i rep estadístiques agregades sense que el sistema incorpori una base de dades de converses.

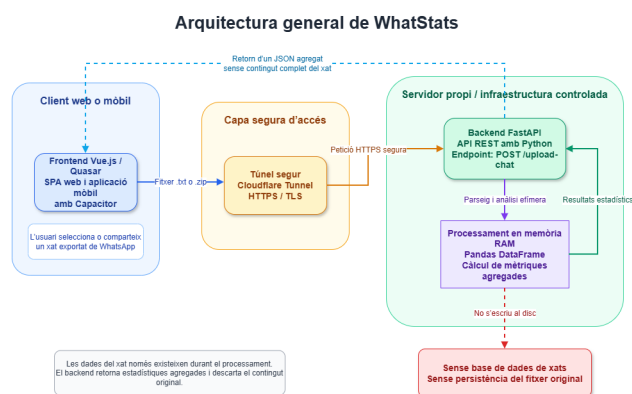


Fig. 1: Arquitectura general de WhatStats.

Aquest document s'organitza de la manera següent. A la secció 2 es defineixen els objectius generals i específics del treball. A la secció 3 es descriu l'estat de l'art i es justifica la contribució diferencial del projecte. A la secció 4 s'exposa la metodologia de desenvolupament. A les seccions 5 i 6 es presenten, respectivament, el disseny i la implementació del *backend* i del *frontend*. A la secció 7 es tracta la integració, l'aplicació mòbil i el desplegament en producció. A la secció 8 es presenten els resultats assolits i, a la secció 9, les conclusions. Finalment, s'inclouen els apartats dedicats a l'ús de la IA generativa, els agraïments, la bibliografia i els apèndixs de suport.

## 2 OBJECTIUS

L'objectiu general del projecte és dissenyar i implementar una aplicació multiplataforma, orientada a la privadesa, que permeti carregar un xat exportat de WhatsApp, calcular-ne estadístiques agregades i visualitzar-les mitjançant una interfície gràfica clara, *responsive* i utilitzable tant des d'un navegador com des d'un dispositiu mòbil.

Aquest objectiu general s'ha concretat en quatre objectius específics. El primer ha estat desenvolupar un motor de *parsing* en Python capaç de normalitzar fitxers `.txt` de WhatsApp provinents tant d'iOS com d'Android, incloent-hi la gestió de missatges multilínia i la transformació de les dades en un `DataFrame` de `Pandas`. Aquest objectiu és fonamental perquè tota la resta del sistema depèn de la qualitat de l'estructura de dades generada.

El segon objectiu ha estat construir una API *RESTful* amb `FastAPI` [3] que gestioni la pujada segura de fitxers, executi els càlculs estadístics en memòria i retorni les mètriques agregades en format JSON. Aquesta API s'ha dissenyat sense base de dades associada i amb mecanismes de control bàsic, com ara límit de mida de fitxer, *rate limiting* i configuració restrictiva per a producció.

El tercer objectiu ha estat implementar una interfície d'usuari interactiva amb `Vue.js` [4], `Quasar` [9] i `ECharts` [5]. La interfície havia de permetre la pujada de fitxers, la visualització de resultats en forma de targetes, gràfiques, rànquings i calendaris, i l'adaptació automàtica a pantalles d'escriptori i dispositius mòbils.

El quart objectiu ha estat preparar el sistema per a un desplegament en infraestructura pròpia, amb especial atenció a l'aïllament del *backend*, l'execució en contenidor `Docker` [19], la publicació només a través de serveis segurs i la possibilitat d'utilitzar `Cloudflare Tunnel` per evitar exposar ports directament a Internet [14].

A més dels objectius principals, es van definir objectius complementaris: anàlisi avançada amb IA, anàlisi de contingut multimèdia i gestió d'usuaris. Durant el desenvolupament, aquests objectius no s'han implementat com a funcionalitats principals per motius de temps i de coherència amb l'abast. En canvi, el temps reservat a millores s'ha orientat a funcionalitats més alineades amb el producte final: anonimització local de participants, mode clar/fosc/automàtic, configuració dinàmica de la URL del *backend*, importació mòbil de fitxers ZIP i una demo local de resultats.

TAULA 1: RELACIÓ ENTRE OBJECTIUS I RESULTATS ASSOLITS

| Objectiu               | Resultat                                                                                                           | Estat                               |
|------------------------|--------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| <i>Parsing</i> de xats | Motor amb <code>Regex</code> , gestió de ZIP en memòria i <code>DataFrame</code> de <code>Pandas</code>            | Assolit                             |
| API REST               | <i>Endpoint</i> de pujada, càlcul d'estadístiques i JSON de resposta                                               | Assolit                             |
| <i>Frontend</i> visual | <i>Dashboards</i> , gràfiques, rànquings, ratxes i disseny <i>responsive</i>                                       | Assolit                             |
| Desplegament           | <code>Dockerfile</code> , <code>.env.example</code> , <i>healthcheck</i> i guia amb <code>Cloudflare Tunnel</code> | Assolit                             |
| Millores opcionals     | Anonimització, demo local, mode fosc i integració mòbil                                                            | Assolit mitjançant una reorientació |

### 3 ESTAT DE L'ART

L'anàlisi de xats exportats no és una idea nova. Existeixen eines comercials, aplicacions mòbils i projectes acadèmics que permeten generar estadístiques a partir de fitxers de conversa. Entre les alternatives analitzades durant la fase inicial es van identificar eines com Chatlyzer [20], Chat Stats [21] i altres projectes. Aquestes solucions acostumen a oferir gràfiques, recompte de missatges, activitat per participants i altres mètriques de conversa. El seu valor rau en la capacitat de convertir un arxiu textual difícil de llegir en una representació visual entenedora.

Tanmateix, el principal inconvenient d'aquest tipus d'eines és el tractament de la privadesa. En molts casos, l'usuari ha de carregar l'historial complet de conversa en una plataforma externa. Encara que el servei declari que no conserva les dades, l'usuari perd control sobre el lloc on s'executa el processament, sobre els registres interns del servidor i sobre possibles còpies temporals. Aquest risc és especialment rellevant en converses personals, grups familiars, xats professionals o converses amb tercers que no han consentit explícitament la pujada del contingut a un servei d'anàlisi.

WhatStats es diferencia d'aquestes alternatives per una decisió arquitectònica explícita: el *backend* no fa servir base de dades per emmagatzemar xats i el processament s'ha concebut com una operació efímera. El fitxer es rep, es llegeix en memòria, es transforma, es calcula i es descarta. Els resultats retornats són principalment agregats estadístics. Aquesta aproximació redueix la superfície de risc i s'alinea amb els principis de minimització i limitació de la finalitat propis de la privadesa per disseny [6].

Des del punt de vista tecnològic, el projecte es basa en eines consolidades. Python i Pandas s'han utilitzat per al tractament i l'anàlisi de dades tabulars [1]. Les expressions regulars de Python han permès definir patrons de detecció per a línies d'Android i iOS [2]. FastAPI s'ha triat per construir una API clara, ràpida i amb suport natiu per a càrrega de fitxers [3]. Al *frontend*, Vue.js proporciona un model reactiu adequat per a interfícies dinàmiques [4], mentre que Quasar facilita la creació d'una aplicació *responsive* i multiplataforma [9]. Finalment, ECharts i vue-echarts han permès generar gràfiques interactives amb redimensionament automàtic [5].

La seguretat de l'API també ha estat considerada des de l'inici. Encara que l'aplicació no gestiona comptes d'usuari ni dades persistents, continua essent una API pública potencialment exposada a fitxers maliciosos, abús de peticions o intents de denegació de servei. Per aquest motiu, s'han incorporat mesures com límit de mida de pujada, *rate limiting*, configuració CORS restrictiva i *Trusted Host Middleware*. Aquestes decisions s'emmarquen en bones pràctiques generals de seguretat d'APIs, com les recollides per OWASP [7].

### 4 METODOLOGIA

La metodologia seguida ha estat seqüencial, basada en el model en cascada. Aquesta elecció respon a la naturalesa de les dependències tècniques del projecte. No era possible dissenyar correctament els *dashboards* sense saber abans quina estructura tindria el JSON de resposta; tampoc no era convenient iniciar el desplegament sense haver establit

zat la configuració del *backend* i del *frontend*. Així, el treball s'ha dividit en fases successives: planificació i disseny, *backend* i processament de dades, *frontend* i visualització, preparació del desplegament i tancament documental.

La primera fase va consistir a definir el problema, l'abast, els objectius i l'arquitectura general. També es va elaborar una planificació temporal que separava les tasques principals i identificava les dependències. Aquesta etapa va ser important per delimitar funcionalitats: el treball no havia d'esdevenir una plataforma social, ni una eina de mineria massiva de dades, sinó una aplicació centrada en l'anàlisi privada d'un fitxer exportat.

La segona fase es va dedicar al *backend*. En aquest punt es van implementar la lectura en memòria, la detecció del format del missatge, la conversió a `DataFrame` i el càlcul d'estadístiques. Durant aquesta fase es va detectar una desviació respecte a la planificació inicial: el desenvolupament del motor analític no havia estat estimat amb prou detall i va requerir més temps del previst. Aquesta desviació es va compensar posteriorment ajustant l'abast d'algunes millores opcionals (per a una comparació gràfica detallada entre la planificació temporal prevista i l'execució real de les fases, vegeu l'Apèndix: [Comparació de la planificació inicial i final](#)).

La tercera fase es va centrar en el *frontend*. Primer es va definir l'estructura visual amb Quasar i Vue.js; després es van crear els components de resultats, les gràfiques i els rànquings; finalment es va integrar la comunicació amb el *backend*. Aquesta fase va confirmar la utilitat de l'arquitectura desacoblada: una vegada establert el contracte JSON, la interfície podia consumir dades reals o de demostració sense modificar el motor analític.

La quarta fase ha consistit en la preparació per a producció i el desplegament final. En el *backend* s'han fixat dependències, s'ha creat un `Dockerfile`, s'ha afegit un *healthcheck*, s'ha separat la configuració per variables d'entorn [17] i s'ha preparat una guia de desplegament a l'arxiu `README.md` de la branca del *backend*. En el *frontend* s'han actualitzat dependències, s'ha reforçat la validació de la URL del *backend*, s'ha generat una *demo* local, s'han acabat de polir algunes vistes i s'ha servit a través de Nginx. Finalment, el sistema complet s'ha desplegat amb èxit en producció configurant el Cloudflare Tunnel, establint així un accés perimetral segur cap a la xarxa pública sense necessitat d'exposar els ports locals del servidor.

La metodologia ha incorporat verificació contínua al final de cada fase. En el *backend* s'han executat proves automatitzades amb resultat de 39 de 39 proves superades, comprovacions de dependències amb `pip check` i auditoria amb `pip-audit`. En el *frontend* s'han revisat vulnerabilitats amb `npm audit`, s'ha executat el *build* Android i s'ha generat un *APK release*. Aquestes verificacions no substitueixen una auditoria externa de seguretat, però aporten evidència tècnica sobre l'estat funcional del sistema.

### 5 DISSENY I IMPLEMENTACIÓ DEL BACKEND

El *backend* de WhatStats s'ha desenvolupat com una API FastAPI sense base de dades. El seu *endpoint* principal és `POST /upload-chat`, que rep un fitxer `.txt` o

.zip mitjançant multipart/form-data. També s'han definit GET /, com a *endpoint* bàsic d'estat, i GET /healthz, pensat per a monitorització i *healthcheck* del contenidor. La Taula 2 mostra els *endpoints* principals.

TAULA 2: *Endpoints* PRINCIPALS DEL BACKEND

| Mètode | Ruta         | Funció                                                                    |
|--------|--------------|---------------------------------------------------------------------------|
| GET    | /            | Comprovar l'estat bàsic de l'API amb <i>rate limit</i> .                  |
| GET    | /healthz     | <i>Healthcheck</i> sense <i>rate limit</i> per a Docker i monitorització. |
| POST   | /upload-chat | Rebre el xat, processar-lo en memòria i retornar estadístiques.           |

La primera decisió rellevant ha estat evitar l'escriptura del fitxer al disc. Per aconseguir-ho, la funció d'extracció rep els bytes del fitxer i, en cas que sigui un ZIP, utilitza `io.BytesIO` i `zipfile.ZipFile` per llegir el contingut directament des de memòria. Aquesta aproximació permet tractar el ZIP com si fos un fitxer físic sense crear-ne una còpia persistent. En cas que l'exportació contingui diversos fitxers de text, la lògica selecciona el fitxer que correspon a l'historial del xat.

El *parser* es basa en dues expressions regulars principals: una per al format d'Android i una altra per al format d'iOS. Cada patró extreu la data, l'hora, l'autor i el contingut del missatge. Quan una línia no coincideix amb cap patró, el sistema la interpreta com una continuació del missatge anterior. En el cas que això passi abans d'haver identificat cap missatge vàlid, la línia es descarta, ja que no es pot associar a cap entrada prèvia. Aquesta lògica és essencial, ja que els xats de WhatsApp poden incloure missatges amb salts de línia, llistes, fragments llargs o contingut enganxat des d'altres aplicacions.

Un cop extretes les parts de cada missatge, la funció principal de *parsing* construeix un `DataFrame` de Pandas. Aquesta estructura permet aplicar operacions d'agrupació, recompte, filtratge i ordenació amb eficiència. Les funcions analítiques implementades inclouen estadístiques bàsiques, missatges per usuari, hores més actives, activitat per calendari, paraules més freqüents, emojis més utilitzats, missatges més llargs i ratxes de dies consecutius d'activitat.

La Figura 2 mostra el flux intern del *backend*: recepció del fitxer, lectura en memòria, *parsing*, `DataFrame`, càlcul d'estadístiques i resposta JSON.

Des del punt de vista de la seguretat, s'han incorporat diverses mesures. La mida màxima de pujada s'ha fixat en 50 MB. El *rate limit* s'ha configurat a Cloudflare, establint un màxim de 6 peticions per minut i per IP. CORS s'ha restringit a mètodes i capçaleres necessàries, i s'ha definit una llista d'orígens permesa. També s'ha afegit `Trusted Host Middleware` per validar la capçalera `Host`. Aquestes mesures no converteixen l'API en un sistema invulnerable, però redueixen riscos bàsics d'abús.

També s'ha millorat la preparació per a producció. La configuració s'ha centralitzat a `config.py` mitjançant variables d'entorn `.env`. S'han introduït valors com `PORT`, `MAX_UPLOAD_SIZE` i llistes configurables d'orígens i hosts. El `Dockerfile` utilitza `python:3.11-slim`,

Flux intern del backend de WhatStats

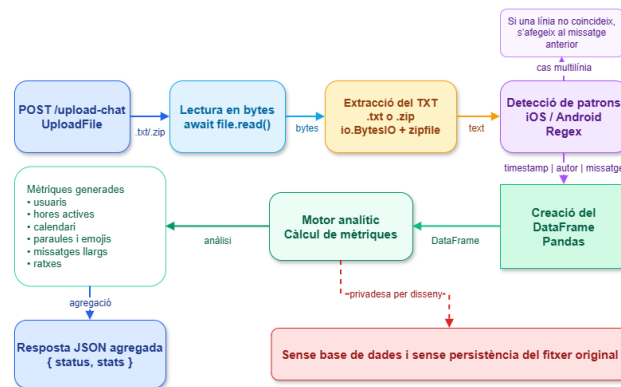


Fig. 2: Flux intern del *backend* de WhatStats.

descarrega les dependències necessàries (com ara les *stopwords* de NLTK [15]) durant el *build* i incorpora un *healthcheck* contra /healthz. Aquest disseny evita que el contenidor hagi de descarregar recursos en temps d'execució i facilita l'operació en un servidor propi.

## 6 DISSENY I IMPLEMENTACIÓ DEL FRONTEND

El *frontend* de WhatStats s'ha desenvolupat amb Vue.js i Quasar. La decisió d'incorporar Quasar va modificar lleugerament el plantejament inicial, basat només en Vue.js, però va aportar avantatges importants: components visuals preparats, disseny *responsive*, integració amb Capacitor i una estructura de projecte adequada per generar tant aplicació web com aplicació mòbil. Aquesta decisió ha permès accelerar el desenvolupament i mantenir una base de codi comuna.

La interfície s'ha dissenyat amb un enfocament minimalista. Les estadístiques es presenten en targetes, gràfiques i rànquings per evitar que l'usuari hagi d'interpretar taules excessivament denses. El sistema utilitza CSS Grid, Flexbox, *breakpoints* de Quasar i components com `q-drawer` per adaptar la navegació a pantalles grans i petites. Els gràfics implementats amb `vue-echarts` utilitzen redimensionament automàtic, cosa que garanteix que la visualització s'adapti a l'espai disponible.

Els resultats s'organitzen en diferents blocs: resum general, activitat temporal, contingut, missatges i ratxes. Les mètriques generals inclouen total de missatges, participants, mida del fitxer i data d'inici del xat. Les distribucions temporals mostren activitat per hora, dia, mes i any. Els rànquings permeten identificar usuaris més actius, paraules més freqüents, emojis més utilitzats i altres valors destacats. Les ratxes d'activitat es representen mitjançant targetes interactives i un calendari emergent.

Un element visual rellevant és el component de podi per als rànquings. El component separa automàticament els tres primers resultats i els presenta amb una estructura de podi, medalles d'or, plata i bronze i valors destacats. La resta de posicions es mostren com una llista compacta amb barres horitzontals. Aquest recurs no només millora l'estètica, sinó que facilita la comprensió immediata dels resultats principals.

La Figura 3 mostra el component de podi, el mes utilitzat al *frontend* de WhatStats.

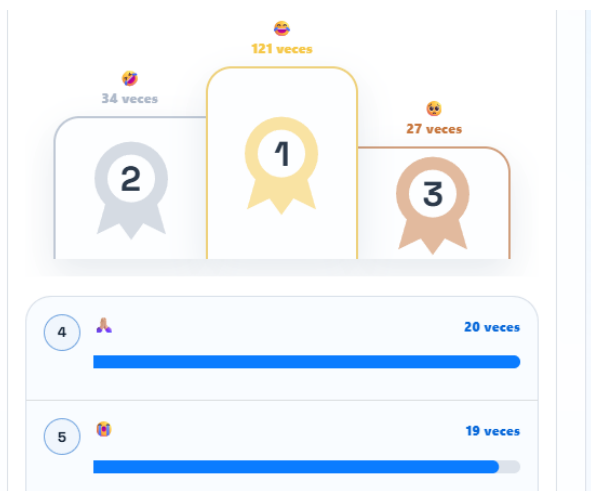


Fig. 3: Component de rànkung amb podi per als tres primers resultats.

Les ratxes d'activitat s'han tractat com un component específic perquè aporten una lectura temporal fàcil d'entendre. Cada ratxa indica data d'inici, data final i durada. A més, s'ha definit un sistema de nivells visuals segons la durada: gris, bronze, plata, or i diamant (per a més informació sobre les ratxes, vegeu l'Apèndix: [Lògica de nivells de ratxa](#)). En seleccionar una ratxa, s'obre un diàleg amb calendari mensual que ressalta els dies actius i mostra el nombre de missatges de cada data.

També s'han incorporat funcionalitats de millora de l'experiència d'usuari. Els modes clar, fosc i automàtic es gestionen amb Pinia [13] i el mòdul Dark de Quasar. L'opció d'anonimització substitueix noms reals per àlies del tipus Usuari 1, Usuari 2 i així successivament. Aquesta anonimització s'executa al *frontend* després de rebre la resposta del *backend*, i és especialment útil per generar captures de pantalla o mostrar resultats sense exposar noms de participants.

La configuració dinàmica de la URL del *backend* és una altra millora significativa. El *frontend* llegeix per defecte `VITE_WHATSTATS_API_BASE_URL`, però permet modificar l'adreça des del diàleg de configuració. Aquesta funcionalitat és útil per a proves locals, entorns mòbils amb IP LAN i usuaris que vulguin executar el *backend* en la seva pròpia màquina. Per reduir riscos, s'ha reforçat la validació de la URL: només s'accepten protocols `http://` o `https://`, es rebutgen credencials embegudes, paràmetres de consulta i fragments, i s'informa l'usuari que canviar el *backend* implica enviar els xats a aquella adreça. En l'entorn oficial de producció de WhatStats, l'accés HTTP es redirigeix automàticament a HTTPS mitjançant Cloudflare.

Finalment, s'ha incorporat una *demo* local de resultats. Aquesta *demo* carrega un *payload* compatible amb el contracte `ChatStatsPayload` sense necessitat de *backend* ni de fitxer real. Això permet ensenyar el funcionament de l'aplicació sense exposar dades privades, i facilita la defensa pública del projecte. La *demo* inclou participants, missatges per usuari, activitat temporal, paraules, emojis, missatges llargs i ratxes.

## 7 INTEGRACIÓ, APLICACIÓ MÒBIL I DESPLEGAMENT

La integració bidireccional entre la capa de presentació (*frontend*) i la lògica de negoci (*backend*) s'ha fonamentat en una arquitectura de serveis HTTP centralitzada, dissenyada per garantir una transferència de dades robusta. En l'entorn web d'escriptori, el *frontend* fa ús de la llibreria Axios [11] per a la gestió de les sol·licituds asíncrones. No obstant això, en l'entorn mòbil construït amb Capacitor, la càrrega de fitxers s'ha refactoritzat per utilitzar directament l'API nativa `XMLHttpRequest` [12]. Aquesta dualitat tècnica respon a la necessitat tècnica de monitoritzar amb precisió el progrés temporal de la pujada i garantir una compatibilitat robusta amb els motors de renderitzat WebView dels dispositius mòbils. En ambdós escenaris, el contracte de comunicació és estricte: el fitxer s'empaqueta i s'envia com un objecte de tipus `FormData` amb el nom de camp `file`, i el client resta a l'espera d'una resposta estructurada en format JSON que compleixi l'esquema `{ status: 'success', stats: ChatStatsPayload }`, garantint així la validesa estructural de les dades abans de visualitzar-les. Es pot trobar informació més detallada sobre el contracte JSON entre el *backend* i el *frontend* a l'Apèndix: [Contracte de dades entre backend i frontend](#).

Pel que fa al desplegament en dispositius mòbils, s'ha apostat per la tecnologia Capacitor [10] com a pont d'unió entre el món web i l'entorn natiu. Aquesta eina permet empaquetar l'aplicació construïda amb Vue.js i Quasar Framework directament dins dels projectes binaris de les plataformes mòbils. Per assegurar una navegació fluida i evitar errors de resolució de rutes en l'entorn de fitxers locals del dispositiu, l'enrutador del *frontend* s'ha adaptat específicament al mode *hash*. Més enllà de l'empaquetament bàsic, s'ha dut a terme una integració amb el sistema operatiu per permetre als usuaris obrir o compartir fitxers ZIP directament amb WhatStats des d'altres aplicacions. En la plataforma Android, aquesta captura d'esdeveniments s'ha configurat a través dels `intent-filters` del fitxer manifest, mentre que a iOS s'ha resolt mitjançant les definicions de tipus de document (`Document Types`). A nivell de codi, la ruta nativa del fitxer interceptat es transforma dinàmicament en un `Blob` binari i, posteriorment, es converteix en un objecte `File` estàndard que queda precarregat automàticament al formulari de pujada, millorant l'experiència d'usuari.

En el procés de preparació i compilació per a l'ecosistema Android, s'han aplicat estàndards actuals de la indústria mòbil. S'han actualitzat les dependències del nucli de Capacitor i s'han ajustat els paràmetres de compilació, situant el `compileSdkVersion` i el `targetSdkVersion` a la versió 35, d'acord amb les polítiques recents de Google. Alhora, s'ha mantingut el `minSdkVersion` a 22 per preservar la retrocompatibilitat, i s'ha actualitzat l'entorn a Java 21. Amb aquesta configuració, el procés de *build* s'ha completat correctament i ha generat un APK en mode *release*, signat criptogràficament amb una *keystore* privada. En cas d'una publicació oficial a Google Play, seria necessari generar un *Android App Bundle* (AAB) signat i incrementar seqüencialment el `versionCode` en cada nova versió.

En l'àmbit del desplegament web, s'ha dissenyat i documentat un flux de treball marcadament conservador. La compilació del projecte s'ha dut a terme en la màquina local de desenvolupament. Aquesta operació genera el directori `frontend/dist`, el qual conté codi purament estàtic. Només aquests fitxers resultants es transfereixen al servidor de producció per ser servits a través de Nginx [8]. Aquest paradigma elimina la necessitat d'instal·lar entorns d'execució com Node.js, gestors de paquets o dependències de desenvolupament (`node_modules`) en el servidor *host*. El servidor web Nginx es configura per escoltar peticions exclusivament en la interfície de *loopback* (`127.0.0.1:9000`). La sortida a Internet es resol mitjançant Cloudflare Tunnel, un servei que enruta de manera segura les peticions des del domini públic `www.whatstats.net` cap al servei local, fent innecessària l'obertura de cap port entrant en el tallafocs o en el router.

De manera anàloga, la posada en producció del *backend* s'ha executat seguint els mateixos principis de confinament. S'ha construït una imatge Docker directament al servidor físic per encapsular l'API *RESTful* i les seves dependències operatives. En executar el contenidor, aquest només exposa el seu servei a la xarxa interna, mapejant els ports únicament a `127.0.0.1:8000:8000`. Posteriorment, s'ha configurat el servei de Cloudflare Tunnel per encaminar de manera xifrada el trànsit provinent del subdomini `api.whatstats.net` directament cap a aquesta adreça local. Aquesta topologia de xarxa aïllada evita que el port 8000 quedi exposat directament a escanejos o atacs directes des de la xarxa pública d'Internet. A més, delegar l'entrada a la infraestructura de Cloudflare ha permès integrar de forma nativa capes avançades de protecció perimetral, com ara un tallafoc d'aplicacions web (WAF), regles estrictes de limitació de trànsit per ruta i polítiques específiques de *bypass* de memòria *cache*, assegurant que les respostes de l'API s'entreguin de manera fluida, segura i robusta. Finalment, cal destacar que tots aquests passos i configuracions de seguretat s'han documentat a l'arxiu `README.md` de la branca del *backend*, a manera de guia de desplegament, amb el propòsit de facilitar que qualsevol usuari pugui replicar aquesta mateixa arquitectura i executar un allotjament propi (*self-hosting*) en el seu servidor personal.

La Figura 4 mostra la proposta de desplegament amb *backend* i *frontend* separats, tots dos exposats mitjançant Cloudflare Tunnel.

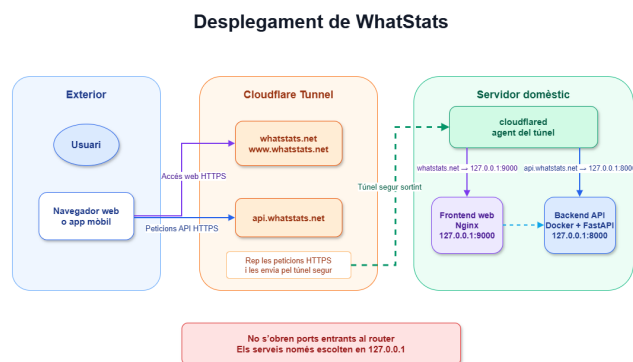


Fig. 4: Desplegament proposat de WhatStats amb Cloudflare Tunnel.

## 8 RESULTATS

Els resultats assolits mostren que WhatStats ha evolucionat des d'una proposta conceptual fins a una aplicació funcional capaç d'executar el flux principal d'ús: seleccionar o importar un xat de WhatsApp, validar el fitxer, enviar-lo al *backend*, processar-lo, rebre estadístiques agregades i visualitzar-les mitjançant una interfície *responsive*.

En el *backend*, el resultat més important és el motor de *parsing* i anàlisi. El sistema pot llegir fitxers `.txt` i `.zip`, detectar línies de missatge en formats d'iOS i Android, gestionar missatges multilínia i transformar el contingut en un `DataFrame`. A partir d'aquesta estructura, el motor calcula estadístiques bàsiques, participació per usuari, activitat per franges horàries, activitat per dies, mesos i anys, paraules freqüents, emojis, missatges llargs i ratxes d'activitat. Aquest conjunt cobreix el nucli funcional previst inicialment.

En el *frontend*, el resultat principal és una experiència visual coherent i multiplataforma. Els *dashboards* implementats permeten consultar les mètriques d'una manera entenedora, sense que l'usuari hagi de revisar dades crues. L'ús de targetes, gràfiques interactives, podis de rànquing i calendaris de ratxes facilita la lectura dels resultats. A més, el disseny *responsive* garanteix que la mateixa aplicació sigui utilitzable en ordinador i en dispositius mòbils.

També s'han assolit resultats concrets en matèria de privadesa. El *backend* no incorpora base de dades, el fitxer es processa en memòria i els *logs* no inclouen contingut del xat. El *frontend* no persisteix els resultats en una base de dades; si la pàgina es recarrega, les dades es perden. L'opció d'anonimització local redueix l'exposició de noms en pantalla, i la *demo* local permet mostrar el projecte sense utilitzar converses reals (per a una descripció detallada de com es reflecteixen visualment aquests mecanismes de transparència i advertències a la interfície, vegeu l'Apèndix: [Polítiques de privadesa visibles a la interfície](#)).

Pel que fa a la preparació per a producció, s'han aconseguit fites rellevants: dependències fixades, separació entre dependències *runtime* i desenvolupament, `Dockerfile` amb usuari no `root`, *healthcheck*, `.env.example` i `README` com a guia de desplegament. Les verificacions disponibles indiquen que el *backend* supera 39 proves automatitzades amb `pytest`. Aquestes proves cobreixen el *parser* d'exports de WhatsApp en format `.txt` i `.zip`, tant d'Android com d'iOS, incloent missatges multilínia, fitxers no compatibles i converses sense missatges vàlids. També validen el càlcul d'estadístiques del xat, com el nombre total de missatges, participants, missatges per usuari, hores d'activitat, agrupacions per dia/mes/any, paraules i emojis més freqüents, missatges més llargs i ratxes de conversa. A més, s'inclouen proves sobre l'API per comprovar el *healthcheck*, la pujada correcta de fitxers, el tractament d'errors, els límits de mida i el resultat complet retornat per l'*endpoint* principal. Finalment, `pip check` no detecta dependències trencades i `pip-audit` no reporta vulnerabilitats conegudes en dependències *runtime*. En el *frontend*, `npm audit` queda sense vulnerabilitats reportades i el *build* Android finalitza correctament.

La Taula 3 sintetitza les evidències de validació disponibles.

TAULA 3: EVIDÈNCIES DE VALIDACIÓ TÈCNICA

| Àmbit        | Verificació                     | Resultat                                          |
|--------------|---------------------------------|---------------------------------------------------|
| Backend      | Tests amb <code>pytest</code>   | 39 proves superades                               |
| Backend      | <code>pip check</code>          | Sense dependències trencades                      |
| Backend      | <code>pip-audit</code>          | Sense vulnerabilitats conegudes en <i>runtime</i> |
| Frontend     | <code>npm audit</code>          | Sense vulnerabilitats reportades                  |
| Android      | <i>Build</i> Capacitor          | Executat correctament                             |
| Desplegament | Docker <i>build</i> al servidor | Validat al servidor / Operatiu                    |

La discussió dels resultats evidencia tres aspectes. En primer lloc, l'arquitectura desacoblada ha estat adequada: el *backend* pot evolucionar afegint mètriques i el *frontend* pot modificar la presentació sense reescriure el *parser*. En segon lloc, el disseny *privacy-first* ha condicionat decisions tècniques rellevants, com l'absència d'una base de dades o la incorporació de la *demo* local. En tercer lloc, el desplegament del sistema s'ha culminat amb èxit en un entorn de producció.

## 9 CONCLUSIONS

El projecte WhatStats ha assolit el seu objectiu principal: desenvolupar una aplicació que permet analitzar xats exportats de WhatsApp i visualitzar-ne estadístiques de manera clara, mantenint la privadesa com a criteri de disseny. El sistema resultant integra un *backend* especialitzat en *parsing* i anàlisi amb un *frontend responsive* i orientat a l'experiència d'usuari.

Des del punt de vista tècnic, el *backend* resol el repte de convertir un fitxer textual heterogeni en una estructura tabular apta per al càlcul estadístic. La combinació d'expressions regulars, Pandas i FastAPI ha estat adequada per construir un motor eficient i modular. El *frontend*, construït amb Vue.js i Quasar, ha permès crear una interfície moderna, usable i adaptable a web i mòbil. L'ús d'ECharts ha facilitat la generació de gràfiques interactives, mentre que Capacitor ha obert la possibilitat de distribuir l'aplicació com a app mòbil.

Des del punt de vista de privadesa, la contribució del projecte és especialment rellevant. WhatStats no pretén competir només per la quantitat de gràfics, sinó per la manera com tracta les dades. El fitxer no es conserva en una base de dades, el processament és efímer i els resultats són agregats. Aquest enfocament no elimina tots els riscos, però redueix de manera substancial l'exposició respecte a alternatives que exigeixen pujar converses completes a serveis de tercers.

També s'han obtingut aprenentatges en planificació. El model en cascada ha resultat adequat per a les dependències

del projecte, però la fase de *backend* va revelar una subestimació inicial del motor analític. Aquesta desviació es va compensar reorientant part dels objectius secundaris cap a millores més assolibles i útils per al producte final. El resultat és una aplicació més coherent, encara que no inclogui algunes funcionalitats avançades plantejades inicialment, com l'anàlisi amb IA o la gestió completa d'usuaris.

Com a línies futures, es proposa reforçar la seguretat del *parser*, implementar una validació de mida durant la lectura real, revisar la protecció contra *ZIP bombs*, ampliar la compatibilitat amb variants d'exportació i afegir proves de càrrega. En l'àmbit de la privadesa i el desplegament, per mitigar l'exposició temporal de dades que suposa la terminació TLS als servidors perimetral de Cloudflare Tunnel, es planteja implementar un xifratge a nivell d'aplicació abans d'enviar el fitxer. També seria interessant estudiar un mode completament local, on el processament s'executés al dispositiu de l'usuari, o un mode avançat amb IA sempre que s'apliqués una anonimització estricta i transparent (per a més informació sobre el detall d'aquestes millores futures, vegeu l'Apèndix: [Línies de Treball Futur i Evolució de la Plataforma](#)).

## ÚS DE LA IA GENERATIVA

Durant el desenvolupament del projecte s'ha utilitzat IA generativa com a eina de suport, no com a substitut de l'anàlisi tècnica ni de la presa de decisions. Les eines emprades han estat principalment ChatGPT, Gemini i un agent de desenvolupament integrat a Visual Studio Code mitjançant Open-Code [18]. El seu ús s'ha documentat per criteris de transparència acadèmica.

En la fase inicial i de *backend*, la IA va donar suport en la depuració d'expressions regulars, la comprensió de llibreries com `io`, la consulta de patrons d'ús de Pandas i la generació de xats ficticis per fer proves sense exposar dades reals. També va ajudar a elaborar bateries inicials de tests. En tots els casos, el codi suggerit va ser revisat manualment i adaptat a l'arquitectura del projecte.

En la fase de *frontend*, Gemini es va utilitzar per obtenir propostes inicials de maquetació i estructura visual. Posteriorment, un agent de desenvolupament amb GPT-5.5 en mode de raonament avançat va ajudar a iterar components, refinar estils, crear variants *responsive* i resoldre detalls d'integració. Aquest ús va ser especialment útil per accelerar tasques repetitives i explorar alternatives de disseny.

La IA també s'ha utilitzat com a assistent de redacció en els informes de seguiment i en la preparació del present article. No obstant això, no s'ha emprat per inventar resultats, generar evidències no verificades ni substituir la verificació tècnica. Les decisions d'arquitectura, privadesa, desplegament i abast han estat definides i validades manualment per l'autor.

## AGRAÏMENTS

L'elaboració d'aquest Treball de Fi de Grau no hauria estat possible sense el suport, la guia i l'acompanyament de diverses persones a les quals vull expressar la meua més sincera gratitud.

En primer lloc, vull agrair als meus tutors, Domènec Madrid i Jordi Herrera, per la seva dedicació i consell al llarg de tot el projecte. La seva orientació durant les fases de definició del problema, el seguiment de la planificació i la revisió de l'abast ha estat fonamental per donar forma a WhatStats.

També vull donar les gràcies de manera especial als meus companys Gabriel Rios, Fiorella Queirolo i Sebastian Malbaceda, que sempre han estat disposats a ajudar-me amb problemes tècnics complexos i decisions de disseny. En la mateixa línia, agraeixo a totes les persones que han facilitat proves amb converses fictícies o entorns controlats, la qual cosa ha permès validar el funcionament del sistema de manera segura i sense exposar dades personals.

Així mateix, vull reconèixer el valor de la comunitat de programari lliure; sense les eines obertes i la documentació accessible, la implementació tècnica d'aquest projecte no hauria estat possible.

Finalment, el meu agraïment més profund és per als meus pares i la meva germana. Gràcies pel vostre suport incondicional, la vostra paciència i per haver estat sempre al meu costat al llarg de tota la meua carrera acadèmica. Aquest assoliment també és vostre.

## REFERÈNCIES

- [1] Pandas Development Team, “Working with text data,” pandas documentation. [En línia]. Disponible: [https://pandas.pydata.org/docs/user\\_guide/text.html](https://pandas.pydata.org/docs/user_guide/text.html)
- [2] Python Software Foundation, “re — Regular expression operations,” Python 3 Documentation. [En línia]. Disponible: <https://docs.python.org/3/library/re.html>
- [3] S. Ramírez, “Request Files,” FastAPI Documentation. [En línia]. Disponible: <https://fastapi.tiangolo.com/tutorial/request-files/>
- [4] Vue.js Core Team, “Introduction — The Progressive JavaScript Framework,” Vue.js Guide. [En línia]. Disponible: <https://vuejs.org/guide/introduction.html>
- [5] Apache Software Foundation, “Apache ECharts,” ECharts Official Website. [En línia]. Disponible: <https://echarts.apache.org/>
- [6] A. Cavoukian, “Privacy by Design: The 7 Foundational Principles,” Information and Privacy Commissioner of Ontario, 2009. [En línia]. Disponible: [https://iapp.org/media/pdf/resource\\_center/pbd\\_implement\\_7found\\_principles.pdf](https://iapp.org/media/pdf/resource_center/pbd_implement_7found_principles.pdf)
- [7] OWASP Foundation, “OWASP API Security Top 10,” OWASP Project, 2023. [En línia]. Disponible: <https://owasp.org/www-project-api-security/>
- [8] F5, Inc., “NGINX Reverse Proxy,” NGINX Docs. [En línia]. Disponible: <https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>
- [9] Quasar Team, “Introduction to Quasar,” Quasar Framework. [En línia]. Disponible: <https://quasar.dev/introduction-to-quasar>
- [10] Ionic Team, “Capacitor Documentation,” Capacitor. [En línia]. Disponible: <https://capacitorjs.com/docs>
- [11] Axios Team, “Getting Started,” Axios Documentation. [En línia]. Disponible: <https://axios-http.com/docs/intro>
- [12] Mozilla Contributors, “XMLHttpRequest,” MDN Web Docs. [En línia]. Disponible: <https://developer.mozilla.org/docs/Web/API/XMLHttpRequest>
- [13] Pinia Team, “Introduction,” Pinia Documentation. [En línia]. Disponible: <https://pinia.vuejs.org/introduction.html>
- [14] Cloudflare Team, “Cloudflare Tunnel,” Cloudflare Docs. [En línia]. Disponible: <https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/>
- [15] NLTK Project, “NLTK Documentation,” Natural Language Toolkit. [En línia]. Disponible: <https://www.nltk.org/>
- [16] T. Kimura et al., “emoji,” Python Package Index. [En línia]. Disponible: <https://pypi.org/project/emoji/>
- [17] Pydantic Team, “Settings Management,” Pydantic Documentation. [En línia]. Disponible: [https://docs.pydantic.dev/latest/concepts/pydantic\\_settings/](https://docs.pydantic.dev/latest/concepts/pydantic_settings/)
- [18] OpenCode Team, “OpenCode Documentation,” OpenCode. [En línia]. Disponible: <https://opencode.ai/docs/>
- [19] Docker Inc., “Docker Documentation,” Docker Docs. [En línia]. Disponible: <https://docs.docker.com/>
- [20] Chatilyzer, “Chatilyzer: Your WhatsApp chat analyzer.” [En línia]. Disponible: <https://chatilyzer.com>
- [21] José Luis Galán, “Chat Stats for WhatsApp,” Google Play. [En línia]. Disponible: [https://play.google.com/store/apps/details?id=com.joseluisgalan.android.chatstats&hl=es\\_419](https://play.google.com/store/apps/details?id=com.joseluisgalan.android.chatstats&hl=es_419)
- [22] Cyborgman1778, “WhatStats Repository,” GitHub. [En línia]. Disponible: <https://github.com/Cyborgman1778/what-stats>
- [23] F. M. Rosa Playà, “WhatStats,” pàgina oficial del projecte. [En línia]. Disponible: <https://www.whatstats.net/#/>

# Apèndixs

## A.1 Contracte de dades entre *backend* i *frontend*

El contracte de dades entre el *backend* i el *frontend* de WhatStats es basa en una resposta JSON validada mitjançant el model `ChatStatsPayload`. Aquesta estructura defineix de manera explícita tots els camps que el *backend* retorna després d'analitzar un xat i que el *frontend* utilitza per generar les targetes, gràfiques, rànquings i panells de resultats.

| Camp                              | Descripció                                                                    |
|-----------------------------------|-------------------------------------------------------------------------------|
| <code>status</code>               | Estat de l'anàlisi, amb valor <code>success</code> o <code>failed</code> .    |
| <code>message</code>              | Missatge informatiu retornat pel <i>backend</i> sobre el resultat del procés. |
| <code>total_messages</code>       | Nombre total de missatges detectats al xat.                                   |
| <code>participants</code>         | Llista amb els noms dels participants identificats.                           |
| <code>total_users</code>          | Nombre total de participants del xat.                                         |
| <code>n_messages_per_user</code>  | Diccionari amb el nombre de missatges enviats per cada participant.           |
| <code>hot_hours</code>            | Distribució de missatges segons l'hora del dia.                               |
| <code>messages_per_day</code>     | Nombre de missatges agrupats per dia.                                         |
| <code>messages_per_month</code>   | Nombre de missatges agrupats per mes.                                         |
| <code>messages_per_year</code>    | Nombre de missatges agrupats per any.                                         |
| <code>top_messages_per_day</code> | Dies amb més activitat dins del xat.                                          |
| <code>top_words</code>            | Paraules més freqüents després d'aplicar els filtres corresponents.           |
| <code>top_emojis</code>           | Emojis més utilitzats pels participants.                                      |
| <code>longest_messages</code>     | Llista amb els missatges més llargs i la informació associada.                |
| <code>top_streaks</code>          | Llista amb les ratxes d'activitat més destacades.                             |

Aquesta estructura permet mantenir una separació clara entre la lògica de càlcul del *backend* i la lògica de visualització del *frontend*. El *backend* és l'encarregat de processar el fitxer, generar les mètriques i retornar-les en un format estable, mentre que el *frontend* només ha d'interpretar aquests camps per representar-los visualment. Això facilita l'evolució del sistema, ja que es poden afegir nous components visuals sense modificar el *parser*, sempre que es mantingui el contracte de dades.

## A.2 Captures de pantalla dels components

Aquest apèndix recull diverses captures de pantalla dels components principals de la interfície de WhatStats. Les imatges mostren l'aparença final de l'aplicació i complementen l'explicació del desenvolupament del *frontend*.

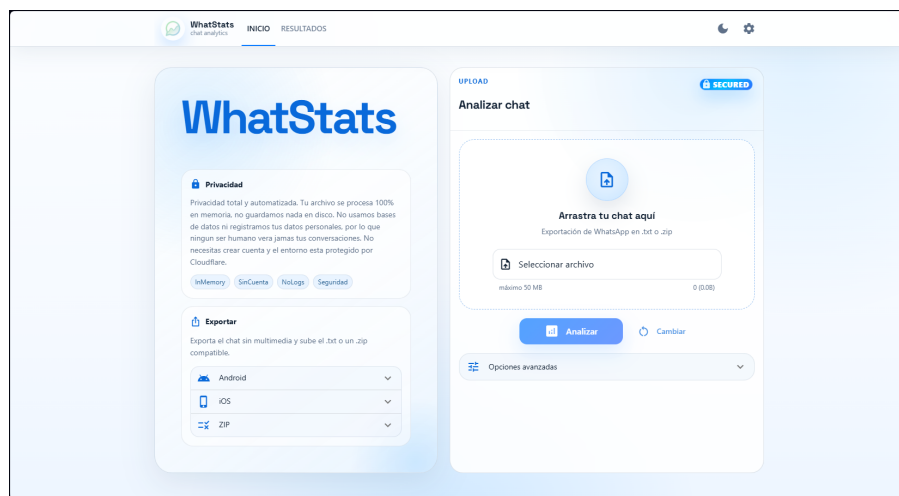


Fig. 5: Vista web de la pantalla inicial de WhatStats.

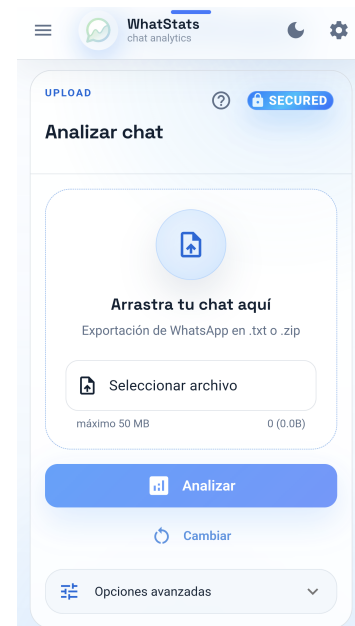


Fig. 6: Vista mòbil de la pantalla inicial de WhatStats.

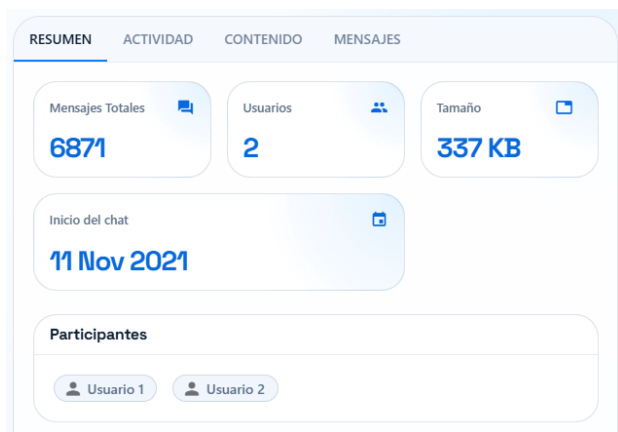


Fig. 7: Targetes de resum dels resultats.

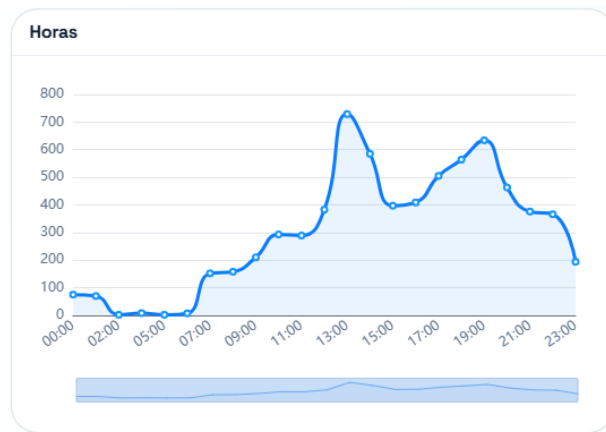


Fig. 8: Gràfica temporal d'activitat.

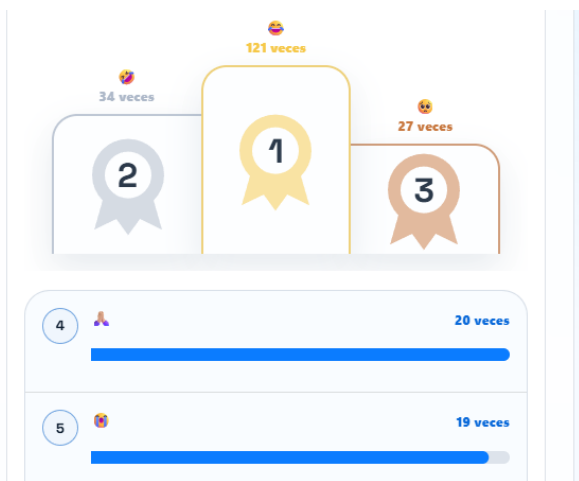


Fig. 9: Rànquing amb podi visual.

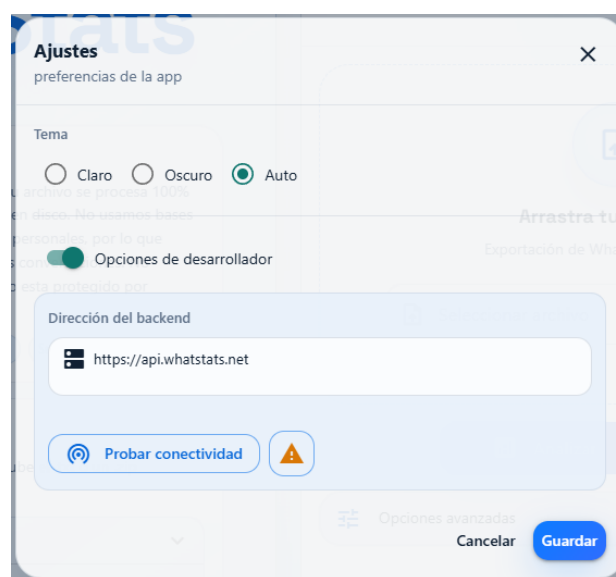


Fig. 10: Configuració de l'API del backend.

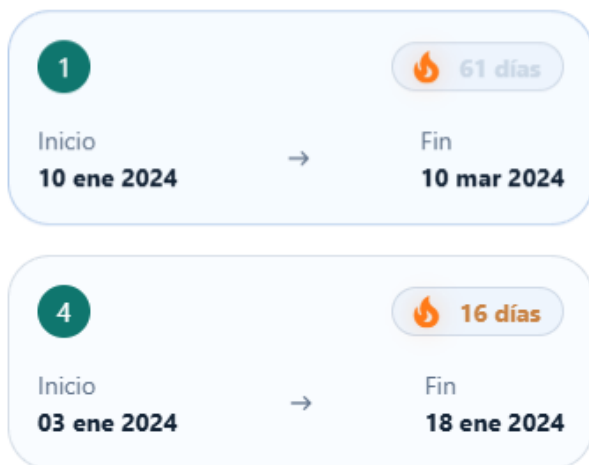


Fig. 11: Panell de ratxes d'activitat.

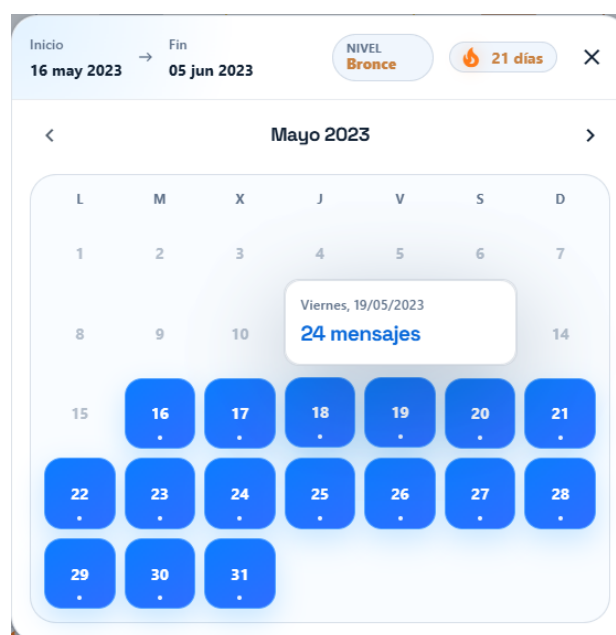


Fig. 12: Calendari detallat d'una ratxa.

### A.3 Lògica de nivells de ratxa

El sistema de ratxes classifica la durada de l'activitat continuada en cinc nivells visuals. Aquesta classificació és útil per convertir una mètrica numèrica en un element fàcilment interpretable per l'usuari.

| Nivell  | Durada            | Funció visual                |
|---------|-------------------|------------------------------|
| Gris    | Menys de 10 dies  | Ratxes curtes o inicials.    |
| Bronze  | De 10 a 49 dies   | Activitat sostinguda bàsica. |
| Plata   | De 50 a 149 dies  | Activitat prolongada.        |
| Or      | De 150 a 364 dies | Activitat molt destacada.    |
| Diamant | 365 dies o més    | Activitat excepcional.       |

### A.4 Polítiques de privadesa visibles a la interfície

La interfície de WhatStats incorpora missatges específics per reforçar la transparència en el tractament de dades. D'una banda, el banner de privadesa informa l'usuari que el fitxer del xat es processa en memòria, que no es desa en cap base de dades i que no es requereix cap compte d'usuari. D'altra banda, quan es modifica la URL del *backend* des de la configuració avançada, l'aplicació mostra una advertència indicant que els xats pujats s'enviaran a aquella adreça i que, per tant, l'usuari ha de confiar en el servidor seleccionat i ha d'assumir la responsabilitat de les conseqüències que es puguin derivar d'utilitzar un *backend* diferent del proporcionat per WhatStats.

### A.5 Línies de Treball Futur i Evolució de la Plataforma

Tot i que WhatStats disposa d'una base funcional i operativa sòlida en el seu entorn de producció, l'arquitectura modular i desacoblada del projecte permet projectar diverses línies de creixement i millora contínua per a futures iteracions:

- **Expansió de funcionalitats analítiques:** Es planteja implementar paràmetres dinàmics a l'API per permetre a l'usuari personalitzar el volum de dades processades i retornades al *frontend* (per exemple, permetre la selecció d'un rànquing *Top N* personalitzat en lloc del límit actual de *Top 10*). Així mateix, s'incorporaran filtres interactius per aïllar les dades d'un usuari específic i es dissenyaran gràfiques per comparar visualment l'activitat entre diferents participants de la conversa.
- **Evolució arquitectònica cap al processament local i IA:** A mitjà termini, resulta de gran interès explorar un model d'execució completament local (*on-device*), on el processament i l'algorisme de *parsing* s'executin íntegrament al dispositiu de l'usuari (sense necessitat de *backend*). Addicionalment, s'estudiarà un mode analític avançat basat en intel·ligència artificial generativa, sempre que s'apliqui un procés d'anonimització previ, estricte i transparent per a l'usuari.
- **Xifratge client-servidor a nivell d'aplicació i mitigació de la terminació TLS:** Durant l'anàlisi de la topologia de xarxa s'ha determinat que els túnels de Cloudflare orientats a protocols HTTP/HTTPS operen sota un model de *proxy* invers amb terminació TLS en els servidors perimetrals (*Edge*) de la companyia per poder aplicar polítiques de Capa 7 (com el WAF). Això significa que el fitxer del xat es troba breument en text clar dins dels nodes de Cloudflare. Per blindar completament la privadesa sense perdre les proteccions contra atacs de denegació de servei, es projecta implementar un *Xifratge a Nivell d'Aplicació*. El client xifrarà el fitxer `.zip` localment (p. ex., mitjançant AES-256) abans de realitzar la petició POST, de manera que viatgi a través del túnel com un flux de dades xifrades opac per a l'intermediari. Com que la clau criptogràfica es gestionaria fora del canal de Cloudflare i només seria coneguda pels extrems autoritzats, Cloudflare no podria accedir al contingut en clar de les converses.
- **Enduriment del rendiment i protecció perimetral:** Es proposa realitzar proves de càrrega i d'estrès per avaluar l'escalabilitat del *backend* en memòria sota escenaris de concurrència elevada. A nivell de xarxa, es refinirà el mecanisme de *rate limiting* mitjançant la configuració correcta de capçaleres de *proxy* (com `CF-Connecting-IP`) per garantir la traçabilitat de la IP real del client darrere del túnel. Addicionalment, s'implementarà una validació dinàmica del flux de lectura (*stream parsing*) per avaluar els ràtios de descompressió en temps real, protegint l'entorn contra atacs de denegació de servei de tipus *ZIP bombs* sense dependre únicament de la capçalera `Content-Length`.
- **Distribució oficial i consolidació del desplegament:** Per finalitzar el cicle d'integració en entorns mòbils, la propera fita consistirà en la generació de binaris en format AAB (*Android App Bundle*) signats criptogràficament mitjançant la *keystore* pròpia, habilitant així la distribució de l'aplicació a la Google Play Store. A nivell de *backend*, s'automatitzarà la validació de les imatges de Docker al servidor de producció assegurant una monitorització contínua a través de l'*endpoint* públic `/healthz`.

## A.6 Comparació de la planificació inicial i final

Aquest apèndix mostra la comparació entre el diagrama de Gantt proposat a l'inici del projecte i el diagrama resultant després del desenvolupament de WhatStats. Aquesta comparació permet observar les desviacions temporals, els reajustaments realitzats i l'evolució real de les diferents fases del treball.

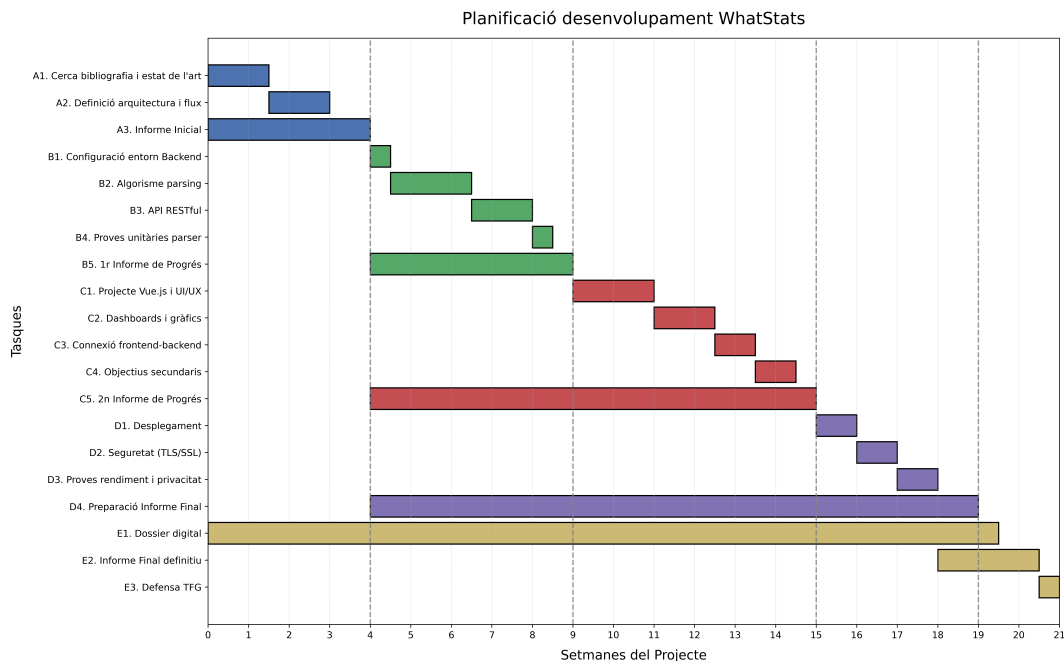


Fig. 13: Diagrama de Gantt inicial del projecte.

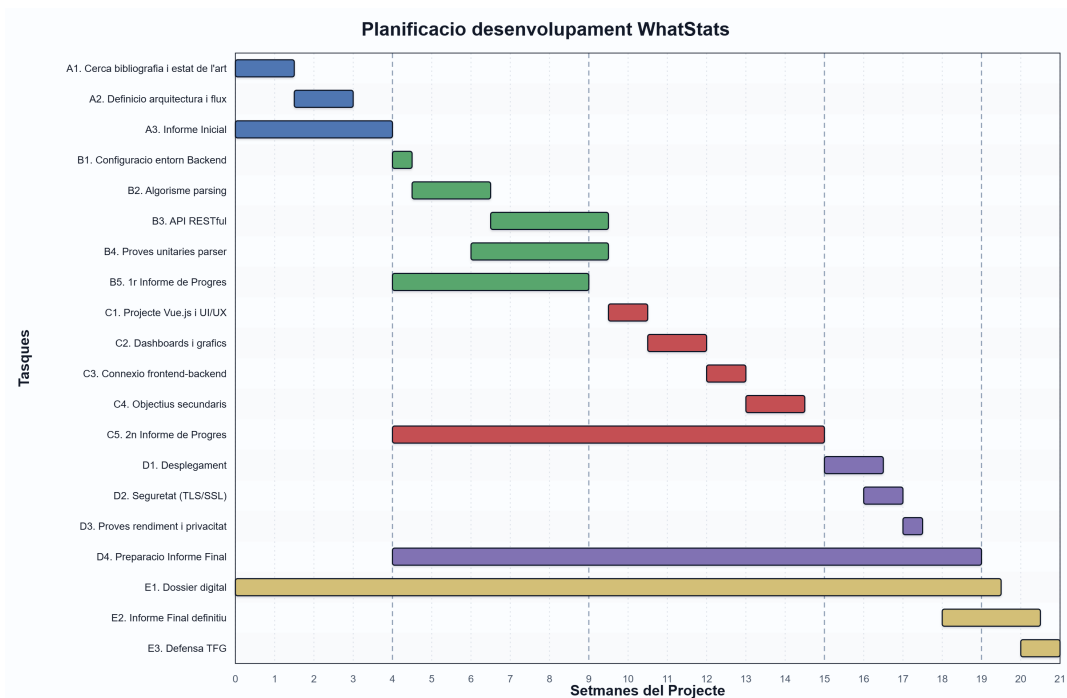


Fig. 14: Diagrama de Gantt final del projecte.

El diagrama inicial, mostrat a la Figura 13, estableix una planificació seqüencial en cinc fases principals, mentre que el diagrama final, representat a la Figura 14, reflecteix l'execució real del projecte. La comparació mostra que la planificació general es va mantenir coherent, tot i que algunes tasques van requerir reajustaments, especialment el desenvolupament del *backend* i del motor analític, que van requerir més dedicació de la prevista inicialment. Aquestes adaptacions no van alterar l'objectiu principal del treball, sinó que evidencien la necessitat d'ajustar la planificació a les necessitats reals detectades durant el desenvolupament.