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

Selva Reche, Xavier. *Portal del Pacient : Disseny i integració d'un portal web i app mòbil amb sistemes d'informació de salut*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334467>

under the terms of the  license.

Portal del Pacient: Disseny i integració d'un portal web i app mòbil amb sistemes d'informació de salut

Xavier Selva Reche

Resum—La digitalització del sector sanitari requereix solucions que permetin als pacients accedir de manera segura a la seva informació clínica i gestionar les seves cites mèdiques. Aquest treball presenta el disseny i la implementació d'un Portal del Pacient, una aplicació web i mòbil que actua com a capa d'accés entre el pacient i el sistema d'informació de salut (HIS). El sistema s'ha desenvolupat amb una arquitectura desacoblada basada en el patró adaptador, que permet integrar-se amb el HIS mitjançant diferents protocols estàndard del sector sanitari sense modificar la lògica de negoci. S'han implementat dos adaptadors d'integració: un basat en l'estàndard HL7 v2.5 a través de Mirth Connect i un altre basat en FHIR R4 a través d'un servidor HAPI FHIR independent. La seguretat s'ha tractat com a requisit transversal, amb xifratge de dades personals, autenticació basada en tokens JWT i registre d'auditoria. El resultat és un sistema funcional, validat amb una bateria de proves automatitzades, que demostra la viabilitat d'una integració multiprotocol amb sistemes sanitaris.

Paraules clau—Portal del pacient, sistemes d'informació de salut (HIS), HL7, FHIR R4, interoperabilitat, Mirth Connect, patró adaptador, seguretat de dades sanitàries.

Abstract—The digitalisation of the healthcare sector requires solutions that allow patients to securely access their clinical information and manage their medical appointments. This work presents the design and implementation of a Patient Portal, a mobile and web application that acts as an access layer between the patient and the Health Information System (HIS). The system has been developed with a decoupled architecture based on the adapter pattern, which allows it to integrate with the HIS through different standard healthcare protocols without modifying the business logic. Two integration adapters have been implemented: one based on the HL7 v2.5 standard through Mirth Connect, and another based on FHIR R4 through an independent HAPI FHIR server. Security has been treated as a cross-cutting requirement, with encryption of personal data, JWT token-based authentication and audit logging. The result is a functional system, validated with a suite of automated tests, that demonstrates the feasibility of a multi-protocol integration with healthcare systems.

Index Terms—Patient portal, Health Information Systems (HIS), HL7, FHIR R4, interoperability, Mirth Connect, adapter pattern, healthcare data security.

1 INTRODUCCIÓ I CONTEXT DEL TREBALL

La transformació digital del sector sanitari ha situat el pacient al centre del seu propi procés assistencial. Iniciatives com l'Espai Europeu de Dades de Salut (EHDS) o la Història Clínica Digital del Sistema Nacional de Salut (HCDSNS) a Espanya impulsen la creació de portals web que permetin als pacients consultar el seu historial, gestionar cites i respondre qüestionaris clínics sense haver de desplaçar-se físicament [13,14]. Aquests portals, però, no emmagatzemen les dades clíniques: actuen com una capa d'accés que s'ha d'integrar amb el sistema d'informació de salut (HIS), que és on resideix la informació real del pacient.

El principal repte d'aquesta integració és la interoperabilitat. Els sistemes sanitaris no exposen les seves dades de manera homogènia: alguns utilitzen l'estàndard HL7, àmpliament implantat des de fa dècades [1], mentre que d'altres adopten estàndards més moderns basats en serveis web com FHIR [2]. Autors com Bender i Sartipi [7] han demostrat que FHIR representa un canvi de paradigma significatiu en la manera d'entendre la interoperabilitat sanitària, però que la convivència amb HL7 v2.x serà inevitable durant molts anys a causa de la ingent base instal·lada existent.

Aquest treball, desenvolupat en el marc d'unes pràctiques a l'empresa PKF Attest Health, té com a objectiu dissenyar i implementar un Portal del Pacient com a prova de concepte (PoC) que mostri la viabilitat d'una integració multiprotocol amb dades de test.

1.1 Objectius

L'objectiu general és dissenyar i implementar un portal web que permeti al pacient accedir a la seva informació clínica i gestionar cites i qüestionaris, integrant-se amb un HIS mitjançant estàndards reals del sector. Els objectius específics són:

- Dissenyar una arquitectura desacoblada basada en el patró adaptador que aïlli la lògica de negoci del portal del protocol de comunicació amb el HIS.
- Implementar una capa de seguretat robusta, amb xifratge AES-256-GCM de dades sensibles, autenticació JWT de doble token i registre d'auditoria complet.
- Desenvolupar dos adaptadors: un basat en HL7 v2.5 via Mirth Connect i un basat en FHIR R4 via HAPI FHIR.
- Construir una interfície d'usuari que permeti gestionar cites, consultar visites i qüestionaris clínics i descarregar informes mèdics.
- Validar el sistema mitjançant 41 proves automatitzades (Jest + Supertest) que cobreixin tots els fluxos de l'API

amb cadascun dels adaptadors.

- Dissenyar i implementar una aplicació mòbil nativa amb React Native que consumeixi la mateixa API REST del backend, oferint als pacients accés mòbil a les funcionalitats principals del portal.

La motivació d'establir dos adaptadors HL7 i FHIR com a objectiu diferencial prové del fet que la coexistència d'ambdues tecnologies és la realitat del sector sanitari privat a Espanya. Mentre els hospitals públics tendeixen a adoptar FHIR per les exigències del EHDS [12], els centres privats d'una mida petita o mitja solen mantenir HIS antics amb HL7. Un portal del pacient que depengui d'un únic protocol no es pot desplegar en tots aquests centres, la qual cosa representa un obstacle comercial significatiu. L'arquitectura multiprotocolar cobreix tots els escenaris: centres amb HL7 actiu, centres en transició a FHIR i centres nous que adoptarien FHIR directament.

2 CONTEXT I ESTAT DE L'ART

La interoperabilitat entre sistemes sanitaris és un camp amb dècades de recerca i estandardització. En l'àmbit europeu, el RGPD [9] i l'EHDS estableixen un marc normatiu que obliga els sistemes a ser transparents, segurs i accessibles per al pacient. A continuació es revisen els conceptes, estàndards i solucions existents més rellevants.

2.1 Portals del pacient: estat actual

Els portals del pacient són aplicacions web o mòbils que ofereixen als pacients accés digital als seus registres mèdics i als serveis del centre sanitari. La literatura acadèmica distingeix entre portals tethered (vinculats a un HIS concret, com Epic MyChart o Cerner Patient Portal) i untethered (independents del HIS, que s'integren via APIs). Estudis recents demostren que l'ús de portals del pacient millora l'adhesió al tractament, augmenta la satisfacció i redueix les visites innecessàries, però que la barrera tècnica d'integració amb sistemes heretats continua sent el principal obstacle per a la seva adopció massiva.

En l'àmbit espanyol, la Estrategia de Digitalización del SNS i iniciatives autonòmiques com Clínica Digital Andalusia o la Carpeta Personal de Salut de Catalunya han impulsat el desplegament de portals públics. En el sector privat, on s'emmarca aquest treball, la integració multiprotocolar és particularment rellevant perquè les clíniques i hospitals privats utilitzen HIS de fabricants molt diversos (Hígea, Imed, Doctolink, entre d'altres) amb protocols d'integració heterogenis.

2.2 L'estàndard HL7

HL7 (Health Level Seven) v2.x és l'estàndard de missatgeria més estès en l'àmbit hospitalari [1]. Defineix missatges basats en text pla amb una estructura jeràrquica de segments separats pel caràcter pipe (|). Cada tipus de missatge té un propòsit concret: SIU (cites), ADT (altes i baixes), ORU (resultats), MDM (documents mèdics) i ORM/ORR (ordres). El transport estàndard és MLLP (Minimal Lower Layer Protocol), un protocol TCP molt simple que embolcalla el missatge HL7 amb byte d'inici (0x0B) i byte de fi (0x1C + 0x0D).

Tot i la seva antiguitat, HL7 v2.x domina el panorama

actual per la seva enorme base instal·lada: s'estima que més del 95% de les institucions sanitàries nord-americanes i una proporció similar a Europa encara utilitzen HL7 v2.x per a almenys una part de les seves integracions [7]. El principal inconvenient és la seva baixa llegibilitat, la rigidesa del format i la necessitat d'un motor d'integració especialitzat per interpretar i encaminar els missatges.

2.3 L'estàndard FHIR R4

FHIR (Fast Healthcare Interoperability Resources) és l'estàndard modern impulsat per HL7 International [2]. A diferència de HL7 v2.x, FHIR es basa en una API REST i representa la informació mitjançant recursos tipats (Patient, Appointment, Encounter, Questionnaire, QuestionnaireResponse, DocumentReference) codificats en JSON o XML. M. Ayaz et al. [8] analitzen els avantatges de FHIR per a la interoperabilitat mòbil i web, conclouent que la seva orientació a recursos i el suport a OAuth 2.0 el fan especialment adequat per a aplicacions centrades en el pacient.

El model de recursos FHIR s'organitza en categories funcionals. Els recursos administratius (Patient, Practitioner, Appointment, Slot) gestionen la programació i la demografia. Els recursos clínics (Encounter i Condition) recullen l'activitat assistencial. Els recursos de documents (DocumentReference, Questionnaire, QuestionnaireResponse) gestionen formularis i informes. Cada recurs té un conjunt de camps obligatoris i opcionals, i pot referenciar altres recursos, construint un graf d'informació clínica interconnectat. FHIR R4 és la primera versió que ha rebut el nivell de maduresa normatiu (Normative) per part d'HL7.

2.4 Motors d'integració: Mirth Connect

Els motors d'integració (Integration Engines) actuen com a intermediaris entre sistemes que parlen protocols diferents. Mirth Connect és un dels motors de codi obert més utilitzats en l'àmbit sanitari [3]. Permet definir canals que reben missatges per un protocol (HL7 v2.x via MLLP), els transformen mitjançant scripts JavaScript i els encaminen cap al destí (una BD, un servei web o un altre sistema). En aquest treball, Mirth Connect actua com a pont entre el portal i la BD del HIS simulat quan s'utilitza el protocol HL7: rep els missatges QBP i SIU generats per l'adaptador, extreu els camps rellevants de cada segment mitjançant un transformer JavaScript (motor Rhino) i executa les operacions SQL corresponents a la BD via JDBC. El transformer implementa set tipus de consulta QBP (dades del pacient, visites, informes, qüestionaris, i actualització d'estat de qüestionaris) i tres subtipus SIU per a la gestió de cites (S26 per llistar cites i slots disponibles, S12 per reservar i S15 per cancel·lar).

2.5 Seguretat en sistemes de salut digitals

La seguretat és un requisit crític en sistemes que gestionen dades de salut, regulades pel RGPD [9] i, a Espanya, per la Llei Orgànica de Protecció de Dades i Garantia dels Drets Digitals (LOPDGDD). Les recomanacions de l'OWASP [10] per a aplicacions web inclouen protecció contra injeccions SQL, gestió segura de sessions, xifratge de dades en repòs i en trànsit, i registre d'auditoria. Per a l'autenticació sense estat en APIs REST,

L'estàndard de facto és JSON Web Token (JWT) [11], que permet emetre tokens signats digitalment que el servidor pot verificar sense consultar una BD de sessions.

El xifratge de dades en repòs és especialment rellevant per a dades de salut. AES-256-GCM (Advanced Encryption Standard, mode Galois/Counter) és un algorisme de xifratge autènticat que proporciona tant confidencialitat com integritat: a més de xifrar les dades, genera un tag d'autenticació que detecta qualsevol manipulació. L'ús de hash deterministes (SHA-256) per a camps de cerca permet fer cerques eficients sobre camps xifrats sense exposar el valor en clar.

2.6 Solucions existents i posicionament

Existeixen portals del pacient comercials consolidats com Epic MyChart o Cerner Patient Portal, fortament acoblats al seu propi HIS. En l'àmbit dels sistemes d'integració oberts, projectes com HAPI FHIR [4] proporcionen servidors FHIR de referència. El valor diferencial d'aquest treball és demostrar, amb una implementació real i validada, com el patró adaptador permet a un portal desplegar-se en centres amb HIS heterogenis (HL7 o FHIR) amb un cost d'adaptació mínim.

Cal contextualitzar el treball en el marc normatiu i tecnològic europeu. La proposta de Reglament de l'Espai Europeu de Dades de Salut (EHDS) [12], presentada per la Comissió Europea el 2022, estableix que els pacients de la UE han de tenir dret a accedir als seus registres de salut electrònics en format digital i compartir-los entre estats membres. El EHDS preveu el format europeu d'intercanvi de dades de salut (EEHRxF), basat en FHIR, com a estàndard de facto per a aquest intercanvi. Això reforça la rellevància d'un portal del pacient que implementi FHIR com a protocol d'integració.

A Espanya, l'Estrategia de Digitalització del SNS 2022-2026 [13] inclou entre els seus eixos l'empoderament digital del pacient mitjançant portals i apps de salut. En el sector privat, la regulació del Ministeri de Sanitat exigeix que els centres sanitaris privats permetin als pacients accedir als seus informes clínics en format digital. Tot això crea un context molt favorable per a solucions com la presentada en aquest treball, que permeten al sector privat complir amb aquests requisits normatius sense haver de migrar el seu HIS existent.

3 METODOLOGIA I ARQUITECTURA

3.1 Metodologia de desenvolupament

El desenvolupament s'ha dut a terme seguint una metodologia àgil i iterativa, amb cicles curts centrats en funcionalitats concretes. La primera fase (març-abril) es va dedicar al disseny de l'arquitectura, la configuració de l'entorn i la implementació del mòdul d'autenticació amb el sistema de tokens JWT. El resultat va quedar documentat a l'Informe de Progrés (I). La segona fase (abril-maig) va cobrir la implementació dels adaptadors d'integració i la bateria de proves, documentada a l'Informe de Progrés (II). La tercera fase (maig) es va centrar en la implementació completa del backend: endpoints clínics, controladors, middleware i integració amb els dos adaptadors. La quarta fase (juny, setmanes

inicials i centrals) va cobrir tot el frontend: el portal web amb React i totes les pantalles, i l'aplicació mòbil amb React Native, incloent la generació de la APK via EAS Build. La cinquena i última fase (juny final) s'ha dedicat al poliment, les proves finals i la documentació.

El control de versions s'ha gestionat amb Git, seguint l'estratègia Gitflow: branques feature/* per a cada funcionalitat (feature/auth, feature/adapters, feature/frontend), integrades a develop i publicades a main quan el sistema és estable. Cada integració requeria que les proves automatitzades passessin. En total, s'han completat 38 commits repartits a diferents branques. El seguiment del projecte s'ha dut a terme mitjançant reunions periòdiques amb els tutors d'empresa i acadèmic.

El principal risc identificat era la complexitat dels protocols HL7 i FHIR, que es va mitigar definint primer la interfície abstracta HISAdapter i desenvolupant tots els components del portal contra aquesta abstracció. Un segon risc era la desviació temporal en el desenvolupament mòbil, que s'havia elevat a probabilitat Alta a l'Informe de Progrés (II). El Pla B previst —prioritzar Android i deixar iOS com a extensió futura— va resultar suficient per cobrir l'objectiu, i l'aplicació mòbil Android s'ha lliurat dins del termini.

3.2 Arquitectura general del sistema

El sistema segueix una arquitectura client-servidor de tres capes, representada a la Fig. 1. La capa de presentació inclou un portal web desenvolupat amb React i una aplicació mòbil amb React Native, ambdós comunicant-se amb el backend mitjançant peticions HTTP autenticades amb JWT. La capa de negoci és un servidor Node.js amb Express que exposa una API REST, gestiona l'autenticació i la seguretat, i conté la lògica de negoci. La capa d'accés al HIS és la capa d'integració intercanviable.

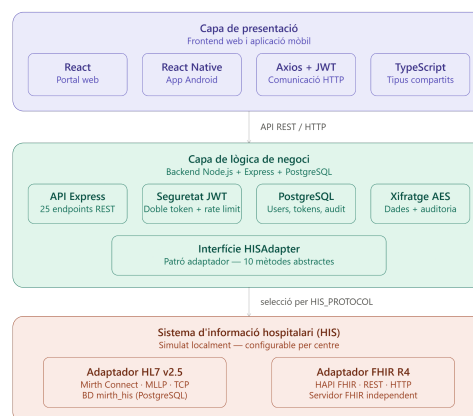


Fig. 1. Arquitectura de tres capes del sistema.

La capa d'integració està dissenyada seguint el patró adaptador (Fig. 2). El backend defineix la interfície abstracta HISAdapter amb mètodes per a totes les operacions del portal: verifyPatient, getSlots, createAppointment, cancelAppointment, getAppointments, getVisits, getReports, getQuestionnaires i completeQuestionnaire. Cadascun dels dos adaptadors de producció (HL7Adapter i FHIRAdapter) implementa aquesta interfície a la seva manera. La selecció de l'adaptador actiu es fa en temps

d'execució a través de la variable d'entorn HIS_PROTOCOL, sense cap canvi de codi.

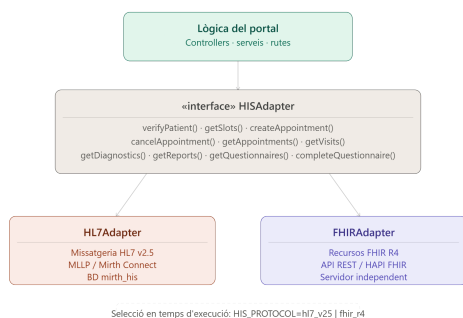


Fig. 2. Patró adaptador: interfície HISAdapter i les dues implementacions.

3.3 Model de dades

El portal manté la seva pròpia BD PostgreSQL [5] amb tres taules. La taula users emmagatzema les dades d'autenticació i el perfil del pacient: el document d'identitat i l'email es guarden tant en forma xifrada (AES-256-GCM, per a la visualització) com en hash determinista SHA-256 (per a cerques eficients sense exposar el valor en clar). Els camps personals com el nom, cognoms, telèfon i data de naixement es xifren amb AES-256-GCM. La taula inclou també camps de configuració (locale, notify_email, notify_push), control d'accés (role, verified) i metadades temporals (created_at, last_login). La taula refresh_tokens emmagatzema els tokens de renovació actius, amb el hash del token, la data d'expiració i un flag de revocació, permetent la invalidació selectiva de sessions sense necessitat de llista negra. La taula audit_log registra totes les operacions del sistema amb l'identificador d'usuari, l'acció executada (GET_VISITS, GET_APPOINTMENTS, CREATE_APPOINTMENT, etc.), la referència al recurs afectat i l'adreça IP. Les dades clíniques no es dupliquen al portal: es consulten en temps real al HIS via l'adaptador.

Aquesta decisió de no duplicar les dades clíniques té implicacions de rendiment. Cada vegada que l'usuari navega a la pantalla de visites o qüestionaris, el portal realitza una petició al HIS en temps real. En l'entorn de proves local la latència és mínima, però en producció podria ser més elevada. Per mitigar-ho, es podria implementar una caché de curt termini per a les dades que canvien poc (llista de professionals, especialitats disponibles), reduint el nombre de peticions al HIS. Això està identificat com una millora per a futures versions.

Per simular el HIS s'han creat dues BDs independents amb models de dades diferenciats. Aquesta separació és una decisió de disseny conscient: una alternativa hauria estat implementar una façana FHIR sobre la BD relacional existent (mirth_his), traduint les consultes SQL a recursos FHIR. Aquesta aproximació, però, no utilitza el protocol de comunicació FHIR com a tal — simplement exposa dades relacionals amb una capa de traducció — i no valida la interoperabilitat real amb sistemes FHIR natius. En canvi, disposar d'un servidor HAPI FHIR independent amb la seva pròpia BD estàndard permet demostrar que l'adaptador FHIR interactua correctament amb un servidor FHIR real, tal com ho faria en un entorn de producció amb un HIS modern.

La BD mirth_his, usada per l'adaptador HL7, organitza la informació en nou taules relacionals: pacients (document d'identitat, nom, data de naixement i gènere), professionals (nom, especialitat i estat actiu), slots (franja horària amb data, hora, especialitat, tipus de visita i estat lliure/reservat), cites (vinculades a slot i pacient, amb estat confirmada/cancel·lada), visites (historial assistencial amb motiu, diagnòstic i notes), informes (document amb títol, data, professional, especialitat i nombre de pàgines), qüestionaris (vinculats a visita, amb títol, URL del formulari extern i estat pendent/completat) i plantilles_qüestionaris (configuració de qüestionaris per tipus de visita i especialitat). La BD del servidor HAPI FHIR, usada per l'adaptador FHIR R4, segueix l'estructura estàndard de recursos FHIR: Patient, Practitioner, Schedule, Slot, Appointment, Encounter, DocumentReference, Questionnaire i QuestionnaireResponse. Tots dos models exposen exactament la mateixa informació clínica al portal, demostrant que el patró adaptador absorbeix completament les diferències estructurals entre els dos estàndards.

3.4 Tecnologies

El backend s'ha implementat amb Node.js v24 [6] i TypeScript, el framework Express 5 per a l'API REST, i la biblioteca pg per a la connexió a PostgreSQL 17 [5]. La gestió de Cross-Origin Resource Sharing (CORS) s'ha configurat explícitament per permetre peticions des del frontend web (localhost:3000) i des de xarxes locals (rang 192.168.x.x), exposant la capçalera X-New-Access-Token necessària per al mecanisme de sliding expiration del JWT. Les proves automatitzades utilitzen Jest com a framework de test i Supertest per a les proves d'integració HTTP. El frontend web s'ha construït amb React i TypeScript, React Router per a la navegació i Axios per a la comunicació HTTP. L'aplicació mòbil s'ha desenvolupat amb React Native i Expo SDK 54, utilitzant @react-navigation/native per a la navegació, Axios amb interceptors JWT per a la comunicació HTTP, @react-native-async-storage/async-storage per a la persistència local dels tokens. La generació de la APK s'ha fet via EAS Build (Expo Application Services). Per al protocol HL7 s'ha emprat Mirth Connect [3] en Docker (port MLLP 2575), i per al protocol FHIR R4 s'ha desplegat un servidor HAPI FHIR en Docker (port 8080), amb la seva pròpia BD interna seguint l'estructura estàndard de recursos FHIR. Tot l'entorn de HIS simulat corre en Docker.

L'elecció de Node.js i TypeScript en lloc d'alternatives com Java Spring Boot o Python Django respon a diverses consideracions. Node.js és un runtime orientat a operacions d'E/S (I/O-bound), que és exactament el cas d'un servidor d'API que fa moltes crides a BD i a serveis externs (Mirth, HIS). El model asíncron (async/await) permet gestionar moltes connexions concurrents amb pocs recursos. A més, l'ús de TypeScript tant al frontend web com al backend i l'app mòbil elimina el canvi de context entre les dues parts i permet reutilitzar les definicions de tipus. L'ecosistema npm disposa de totes les biblioteques necessàries (pg, bcrypt, jsonwebtoken, express) amb suport TypeScript de primera classe.

4 DESENVOLUPAMENT DEL SISTEMA

4.1 Backend: API REST i seguretat

El backend exposa una API REST organitzada per recursos. Les rutes `/v1/auth` gestionen el registre, l'inici de sessió, el refresh de tokens i el logout. Les rutes `/v1/users/me` gestionen el perfil i el canvi de contrasenya. Les rutes clíniques exposen totes les operacions sobre el HIS: GET `/v1/appointments/slots` (franges horàries disponibles, filtrable per especialitat), POST `/v1/appointments` (reservar cita), DELETE `/v1/appointments/:id` (cancel·lar), GET `/v1/appointments` (llistar cites), GET `/v1/visits` (historial de visites), GET `/v1/questionnaires` (qüestionaris clínics), PATCH `/v1/questionnaires/:id/complete` (marcar com a completat) i GET `/v1/reports` (informes mèdics). Cada petició clínica passa pel middleware d'autenticació JWT i registre d'auditoria.

La seguretat s'ha dissenyat seguint les recomanacions de l'OWASP [10] i l'estàndard JWT [11]. L'autenticació utilitza un esquema de doble token: access token (vigència 15 minuts) per a les peticions normals, i refresh token (vigència 30 dies, amb rotació en cada ús) per a la renovació transparent de sessió.

Les contrasenyes s'emmagatzemen amb bcrypt (cost 12). Les dades personals (nom, cognoms, telèfon, document d'identitat) es xifren amb AES-256-GCM, de manera que un eventual accés no autoritzat a la BD no exposi dades en clar. Per fer cerques eficients sobre camps xifrats s'utilitzen hash SHA-256 deterministes (email_hash, id_document_hash). Totes les operacions rellevants (login, registre, consulta de cites, consulta de qüestionaris...) es registren a audit_log amb l'ID d'usuari, l'acció, la IP i la marca temporal.

S'ha implementat també protecció contra atacs de força bruta en l'endpoint de login. Després de 5 intents fallits consecutius des de la mateixa adreça IP en una finestra de 15 minuts, l'endpoint retorna un error 429 (Too Many Requests) i no processa més intents fins que la finestra expiri. Això es gestiona mitjançant un middleware de rate limiting que utilitza una taula en memòria (o Redis en producció) per comptabilitzar els intents per IP. Addicionalment, tots els missatges d'error d'autenticació són intencionalment ambigus ("Credencials incorrectes", sense especificar si l'email o la contrasenya és incorrecte) per evitar l'enumeració d'usuaris.

La protecció contra injeccions SQL s'assegura mitjançant l'ús exclusiu de consultes parametritzades (prepared statements) a través de la biblioteca pg. Cap valor provinent de l'usuari s'interpol·la directament en cadenes SQL; tot s'envia com a paràmetre separat que el driver de PostgreSQL escapeja correctament. Les capçaleres de seguretat HTTP (Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security) s'afegeixen a totes les respostes mitjançant el middleware helmet, seguint les recomanacions de l'OWASP [10].

4.2 Adaptador HL7 v2.5

L'HL7Adapter es comunica amb Mirth Connect via MLLP (port 2575). Per a cada operació, construeix el missatge HL7 corresponent amb els segments adequats. Per a les consultes clíniques (visites, informes,

qüestionaris i dades del pacient), genera missatges QBP (Query By Parameter) amb els segments MSH i QPD, on el camp QPD-2 indica el tipus de consulta i QPD-3 conté el document d'identitat del pacient. Per a la gestió de cites, genera missatges SIU amb els segments MSH, SCH i PID: SIU^S12 per reservar (inclou el slot_id i el document del pacient), SIU^S15 per cancel·lar i SIU^S26 per llistar. El SIU^S26 té un comportament dual: si el missatge inclou un segment PID amb el document del pacient, retorna les cites existents d'aquell pacient; si no n'inclou, retorna els slots disponibles filtrats per especialitat i tipus de visita. Això permet reutilitzar el mateix tipus de missatge per a dos casos d'ús relacionats. El missatge s'envia via TCP amb els delimitadors MLLP estàndard (byte d'inici 0x0B, bytes de fi 0x1C+0x0D). Mirth Connect rep el missatge, extreu els segments al transformer JavaScript (motor Rhino), executa les operacions SQL sobre la BD mirth_his i retorna una resposta amb MSA|AA en cas d'èxit o MSA|AE en cas d'error.

La Fig. 3 il·lustra el flux complet d'una reserva de cita via HL7: el frontend envia una petició POST al backend; l'HL7Adapter construeix el missatge SIU^S12; Mirth Connect comprova la disponibilitat del slot, insereix la nova cita a la taula cites i marca el slot com a reservat (status = 'booked'); l'adaptador interpreta la resposta MSA|AA i el backend retorna la confirmació al frontend.

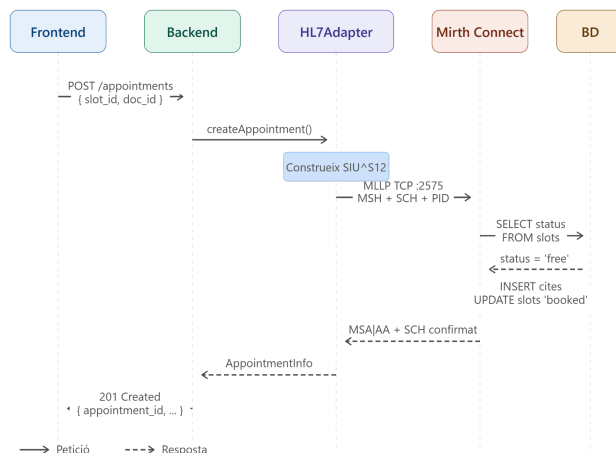


Fig. 3. Flux de reserva d'una cita amb el protocol HL7 v2.5.

4.3 Adaptador FHIR R4

El FHIRAdapter es comunica amb un servidor HAPI FHIR independent desplegat en Docker (port 8080), que actua com a HIS simulat FHIR natiu. A diferència de l'adaptador HL7, el FHIRAdapter fa crides HTTP estàndard a l'API REST FHIR R4, consumint i creant recursos en format JSON. Els recursos usats són: Patient (verificació del pacient), Slot i Schedule (franges horàries), Appointment (cites), Encounter (visites), DocumentReference (informes) i Questionnaire i QuestionnaireResponse (qüestionaris). La URL del formulari extern es gestiona via una extensió FHIR personalitzada al recurs Questionnaire.

4.4 Simulació del HIS

Atesa la impossibilitat de connectar-se a un HIS real en producció, s'han creat dues simulacions independents: la

BD mirth_his per a l'adaptador HL7 i un servidor HAPI FHIR poblat amb recursos FHIR R4 per a l'adaptador FHIR. Les dades de prova s'han dissenyat per ser clínicament coherents i equivalents en tots dos sistemes: els qüestionaris enviats corresponen a l'especialitat de la visita prèvia, les visites contenen motius i diagnòstics mèdicament plausibles, i els informes inclouen el contingut esperat de cada especialitat. Això permet fer demostracions del portal que resultin creïbles i detectar possibles errors en la visualització de dades, independentment del protocol actiu.

4.5 Frontend

La interfície d'usuari s'ha dissenyat amb una paleta corporativa basada en blaus i tipografia Manrope, amb animacions d'entrada (fade-in-up) que aporten naturalitat. Inclou sis pantalles principals: (1) inici de sessió i registre amb validació de DNI/NIE/passaport i indicador de força de contrasenya; (2) tauler principal amb resum de cites, visites, qüestionaris i informes; (3) gestió de cites amb circuit guiat de quatre passos i calendari visual interactiu; (4) historial de visites amb qüestionaris inclosos al modal de detall; (5) informes mèdics; i (6) perfil d'usuari amb gestió de notificacions i canvi de contrasenya.

El circuit de reserva de cites està estructurat en quatre passos: (1) selecció d'especialitat, mostrant les especialitats amb hores disponibles; (2) selecció del tipus de visita; (3) selecció de professional; (4) selecció de data i hora. En el quart pas, un calendari ad hoc mostra el mes actual (navegable) amb els dies que disposen d'hores marcats amb un punt blau i en negreta; el clic en un dia mostra els horaris disponibles. Un cop seleccionat un horari, un modal de confirmació mostra les dades resum i permet confirmar o cancel·lar. La interfície aplica actualització optimista: el canvi es reflecteix immediatament al frontend mentre la petició es processa al backend, millorant la percepció de velocitat.

Els qüestionaris clínics no tenen una pantalla pròpia independent — s'integren en dos punts de l'experiència d'usuari. Al tauler principal, el pacient veu un resum dels qüestionaris pendents amb accés directe. A la pantalla de visites, el modal de detall de cada visita mostra els qüestionaris associats a aquella visita concreta, amb el seu estat (pendent o completat). Cada qüestionari pendent ofereix dues accions: un botó "Emplenar" que obre el formulari extern en una nova pestanya, i un botó "Ja emplenat" que marca el qüestionari com a completat al HIS via l'endpoint PATCH /questionnaires/:id/complete.

La comunicació amb el backend es gestiona mitjançant interceptors d'Axios que afegeixen automàticament el token JWT a cada petició i gestionen la renovació transparent de sessions via el refresh token. L'estat d'autenticació es manté en un AuthContext de React, accessible des de qualsevol component. Les rutes protegides utilitzen un component PrivateRoute que redirigeix a la pantalla de login si l'usuari no està autenticat, i la ruta inversa redirigeix a /dashboard si l'usuari ja té sessió activa.

Els modals del portal s'implementen amb ReactDOM.createPortal, que renderitza el contingut fora del flux del DOM habitual i el munta directament al document.body. Això evita problemes de z-index i

overflow-hidden que poden aparèixer quan els modals s'intenten renderitzar dins d'elements amb posicionament relatiu. El modal té un overlay de fons semitransparent que bloqueja les interaccions amb el contingut de sota i es tanca al clic extern.

La gestió del servei clínic al frontend s'ha organitzat en tres capes. La capa més baixa és el mòdul api.ts, que exporta una instància d'Axios configurada amb la URL base del backend i els interceptors de token. Per damunt hi ha els serveis específics (clinical.service.ts, auth.service.ts, user.service.ts) que encapsulen les crides HTTP i les transformen a objectes TypeScript tipats. Finalment, les pàgines consumeixen directament els serveis, mantenint la lògica de la vista separada de la lògica d'accés a dades.

Des del punt de vista de l'accessibilitat, s'han aplicat bones pràctiques generals: ús d'atributs alt en les imatges, relació de contrast suficient entre el text i el fons (les combinacions blau fosc #1F3864 sobre blanc superen el ràtio de contrast 4.5:1 del nivell AA de les WCAG), i estructura HTML semàntica amb elements article, section i nav. El portal utilitza la propietat lang="ca-ES" al document HTML per facilitar la lectura per part dels lectors de pantalla.

La responsivitat del portal s'ha limitat al rang d'escriptoris i portàtils (amplades a partir de 1024 px), que és l'ús previst per a un portal clínic professional. L'accés des de dispositius mòbils es cobreix mitjançant l'aplicació mòbil nativa descrita a la secció 4.6.

L'estructura de tipus TypeScript del frontend defineix les interfaces per a tots els objectes de domini: Slot, Appointment, Visit, Report, Questionnaire i User, a més dels tipus d'entrada per a autenticació (LoginInput, RegisterInput, AuthTokens). Aquestes interfaces actuen com a contracte entre el frontend i el backend: si el backend canvia el format d'una resposta, el compilador TypeScript detectarà la incompatibilitat en temps de compilació. Això ha evitat diverses regressions durant el desenvolupament, especialment en els canvis de format de data entre l'adaptador HL7 (YYYYMMDD) i FHIR (YYYY-MM-DD).

4.6 Aplicació mòbil (React Native)

El Portal del Pacient inclou, com a component addicional al web, una aplicació mòbil nativa desenvolupada amb React Native i Expo SDK 54, orientada a la plataforma Android. L'objectiu és proporcionar als pacients accés còmode a les funcionalitats principals del portal des del seu dispositiu mòbil, complementant el portal web sense duplicar la lògica de negoci.

La decisió d'usar React Native respon a diverses consideracions. Primer, permet compartir una part significativa del codi entre plataformes (serveis, tipus, lògica d'autenticació), reduint el cost de desenvolupament i manteniment. Segon, l'ús de TypeScript, ja emprat al frontend web i al backend, elimina el canvi de tecnologia. Tercer, React Native consumeix la mateixa API REST del backend sense cap modificació. Expo, com a framework sobre React Native, simplifica la configuració de l'entorn i elimina la necessitat d'Android Studio durant el desenvolupament, permetent provar l'app al dispositiu físic via Expo Go. La generació de la APK es fa via EAS

Build (Expo Application Services), que compila el projecte al núvol.

L'aplicació mòbil comparteix la mateixa lògica d'autenticació JWT que el portal web, amb doble token (access de 15 minuts i refresh de 30 dies) i sliding expiration. Els tokens s'emmagatzemen via AsyncStorage i els interceptors d'Axios gestionen automàticament la renovació i el tancament de sessió. El flux d'autenticació és idèntic al del portal web, consumint els mateixos endpoints /auth del backend.

Les pantalles implementades a la versió final de l'app mòbil són sis: Login i Registre (amb validació de document d'identitat i indicador de força de contrasenya), Dashboard (resum de cites, visites, qüestionaris i informes), Cites (l·listat i circuit de reserva en 4 passos amb calendari natiu i stepper), Visites (historial amb filtres de dates via DateTimePicker natiu i modal amb gesture drag-to-close), Informes (l·listat amb els mateixos filtres i modal de detall), i Perfil (dades personals, preferències de notificació i canvi de contrasenya).

La navegació es resol amb un Drawer lateral natiu implementat amb l'API Animated de React Native, sense dependències externes, degut a incompatibilitats de react-native-reanimated amb el motor Hermes en SDK 54. El drawer reproduïx visualment el sidebar del portal web i s'obre amb el botó  de la capçalera. Totes les pantalles incorporen SafeAreaView per evitar solapaments amb els botons de sistema d'Android.

Arquitecturament, l'app mòbil es beneficia directament de l'arquitectura multiprotocolar del backend. Com que tots els adaptadors (HL7 i FHIR) exposen exactament la mateixa API REST, l'aplicació mòbil consumeix les dades sense cap adaptació específica per a cada protocol, igual que el portal web. Un centre amb HL7 i un altre amb FHIR serveixen les dades al mòbil de manera transparent, demostrant el valor del patró adaptador: el mateix frontend (web i mòbil) funciona per a qualsevol configuració de backend.

5 AVALUACIÓ I RESULTATS

5.1 Mètode d'avaluació

La validació del sistema s'ha dut a terme a tres nivells complementaris. El primer nivell comprèn proves unitàries que validen components aïllats: la correctesa del xifratge i desxifratge AES-256-GCM, la generació i verificació dels tokens JWT, el hashing determinista SHA-256 i la validació dels formats de document d'identitat (DNI, NIE, passaport). El segon nivell comprèn proves d'integració que validen els fluxos complets de l'API HTTP amb Supertest, des de la petició fins a la resposta, incloent la interacció amb la BD. El tercer nivell és una validació manual de consistència creuada: les mateixes operacions executades amb l'adaptador HL7 i amb el FHIR retornen resultats equivalents per als mateixos pacients, verificat manualment durant el desenvolupament canviant la variable d'entorn HIS_PROTOCOL entre hl7_v25 i fhir_r4.

5.2 Resultats de les proves automatitzades

El pla de proves del projecte defineix 95 casos de prova distribuïts en 9 mòduls, cobrints des de l'autenticació fins

a la seguretat avançada, amb proves unitàries, d'integració i funcionals. D'aquests 95 casos, s'han implementat i automatitzat 41 proves en 7 suïtes, que cobreixen els fluxos principals de cada mòdul. La resta de casos — principalment les proves de l'HL7Adapter aïllat amb mock MLLP, les proves E2E amb mocking complet del HIS, i les proves de seguretat avançada (rate limiting, timeout del HIS, HTTPS obligatori) — queden identificats com a treball futur per a una versió de producció.

Les suïtes d'integració (auth, appointments, clinical, users) validen els fluxos complets de l'API HTTP des de la petició fins a la resposta, incloent la interacció real amb la BD PostgreSQL. Les suïtes unitàries (middleware, adapterFactory, encryption/jwt) validen components aïllats: la correctesa del xifratge i desxifratge AES-256-GCM, la generació i verificació de tokens JWT, el hashing determinista SHA-256 i la selecció de l'adaptador per variable d'entorn.

Les 41 proves passen amb èxit amb l'adaptador FHIR actiu (HIS_PROTOCOL=fhir_r4). La validació amb l'adaptador HL7 s'ha realitzat manualment canviant la variable d'entorn HIS_PROTOCOL a hl7_v25 amb Mirth Connect actiu, confirmant que la lògica del portal és independent del protocol d'integració i que tots dos adaptadors retornen resultats funcionalment equivalents.

Pel que fa a la cobertura de codi, jest --coverage reporta una cobertura del 67% de línies, 53% de branques i 70% de funcions, excloent els adaptadors i fitxers auxiliars (seed, mock) que depenen de serveis externs no disponibles en l'entorn de test. L'AdapterFactory i totes les rutes assoleixen el 100% de cobertura. Les àrees menys cobertes corresponen als controladors de gestió d'errors (casos d'error de BD) i als fluxos de refresh token i logout, que requereixen condicions de test més complexes. La cobertura és inferior al llindar del 80% definit al pla de proves, cosa que reflecteix que una part significativa dels casos planificats no s'ha automatitzat en aquesta versió.

Des d'una perspectiva més àmplia, els resultats confirmen que l'arquitectura és sòlida i que els fluxos principals estan validats. L'ús de TypeScript, les proves automatitzades i la separació clara de responsabilitats garanteixen que qualsevol modificació futura — com afegir un nou endpoint o un nou mètode a la interfície HISAdapter — sigui detectada immediatament per les proves existents si trenca quelcom. Per a un desplegament real, caldria completar la bateria de proves amb mocking dels adaptadors, proves de càrrega i una auditoria de seguretat externa, però la base tecnològica i l'arquitectura de proves estan llestes per ser escalades.

5.3 Resultats funcionals

Des del punt de vista funcional, s'ha verificat el cicle complet d'ús del portal: registre amb document d'identitat, inici de sessió, consulta de l'historial clínic (visites, qüestionaris pendents i completats, informes), gestió de cites (reserva i cancel·lació). En reservar una cita, el slot passa a estat reservat a la BD del HIS (status = 'booked' a mirth_his i 'busy' al servidor HAPI FHIR) i el canvi es reflecteix immediatament al frontend. La prova de consistència creuada, realitzada manualment canviant la variable HIS_PROTOCOL, confirma que tots dos adaptadors retornen les mateixes dades normalitzades per al mateix pacient.

5.4 Discussió

Els resultats confirmen la hipòtesi de partida: és viable construir un portal del pacient que s'integri amb protocols sanitaris molt diferents sense modificar la lògica de negoci. El patró adaptador ha demostrat ser una decisió arquitectònica encertada, consistent amb les conclusions de treballs com el de Bender i Sartipi [7], que identifiquen l'abstracció de l'estàndard com una de les claus per a la interoperabilitat sostenible.

La principal limitació tècnica identificada és la latència del servidor HAPI FHIR en entorn local, que introdueix més overhead que la comunicació HL7 via MLLP. En producció, on el HIS exposi una API FHIR nativa optimitzada, aquesta latència seria comparable. Una segona limitació és que els qüestionaris vinculats a visites només sincronitzen l'estat de completat de manera asíncrona: el pacient marca el qüestionari com a completat des del portal, però la verificació de les respostes concretes es fa externament a la plataforma del formulari.

Una tercera limitació identificada té a veure amb la verificació d'identitat en el procés d'alta. El sistema verifica que el document d'identitat existeix al HIS via `verifyPatient()`, però no verifica que qui es registra és realment el titular del document. En un entorn de producció, caldria implementar un pas addicional de verificació, possiblement amb un codi enviat per SMS o correu electrònic, o integrant-se amb un servei d'identitat digital com Cl@ve o idCAT. Aquesta limitació es documenta explícitament com a requisit per al pas a producció.

Més enllà de les limitacions identificades, els resultats s'alineen amb les conclusions de la literatura: la interoperabilitat sanitària requereix decisions arquitectòniques meditada [8,9]. L'ús del patró adaptador, la separació estricta entre la lògica de negoci i els protocols d'integració, i la validació del comportament equivalent dels dos adaptadors, són les aportacions principals d'enginyeria que es poden extrapolar a projectes reals d'integració sanitària.

Des d'un punt de vista d'enginyeria del software, el principi d'inversió de dependències s'ha demostrat particularment adequat per a la integració amb sistemes heterogenis. Els mòduls d'alt nivell (controladors, serveis) no depenen dels mòduls de baix nivell (adaptadors concrets), sinó d'una abstracció comuna. En un escenari real, un portal així dissenyat podria desplegar-se a un centre amb HL7 i, en el futur, migrar a FHIR canviant la variable de configuració i ajustant el mapping de dades de l'adaptador al model concret del HIS del centre, sense afectar la resta del sistema.

6 CONCLUSIONS

Aquest treball ha assolit l'objectiu de dissenyar i implementar un Portal del Pacient funcional i segur, capaç d'integrar-se amb un HIS mitjançant HL7 v2.5 i FHIR R4. S'han complert tots els objectius: arquitectura desacoblada amb el patró adaptador; seguretat amb AES-256-GCM, JWT de doble token i auditoria; adaptadors HL7 via Mirth Connect i FHIR R4 via servidor HAPI FHIR independent; interfície web de set pantalles i aplicació mòbil Android de sis pantalles; i 41 proves automatitzades que validen els fluxos principals de l'API.

La principal aportació del treball és demostrar en codi real que el patró adaptador permet a un portal comunicar-se de manera transparent amb HIS que parlen protocols sanitaris molt diferents. Això té valor pràctic directe en un context, com el de PKF Attest, on els centres clients poden tenir HIS molt diversos. El portal podria desplegar-se en cada centre configurant l'adaptador adequat i ajustant el mapping de dades al model concret del HIS.

Com a línies de continuïtat es proposen: la descàrrega d'informes en PDF — funcionalitat que s'ha desescalat en aquesta versió perquè el procés de gestió documental varia significativament entre centres (alguns gestionen els PDFs al propi HIS, d'altres via sistemes documentals externs) i no era possible concretar un model genèric sense conèixer la infraestructura real del centre client; notifikacions push per a cites pròximes i qüestionaris pendents; la internacionalització (i18n) al castellà i anglès; i proves d'integració contra un HIS real en un entorn de producció controlat, que permetrien validar el comportament del sistema amb volums de dades i latències reals. També seria interessant ampliar la cobertura FHIR per donar suport a més recursos: al·lèrgies, medicació activa i vacunes. Finalment, atès que el present treball és una prova de concepte (PoC), tot el sistema s'ha executat en local — backend, BD, Mirth Connect i servidor HAPI FHIR — sense necessitat d'infraestructura cloud. Per al pas a producció, caldria definir una estratègia de desplegament: el backend migraria a un servei cloud (Railway, Render o AWS), amb la tria de proveïdor condicionada als requisits de residència de dades a la UE i al compliment de l'Esquema Nacional de Seguretat (ENS) per a dades de salut; el frontend web a un servei d'estàtics (Netlify o Vercel), i l'app mòbil a Google Play Store o a un MDM corporatiu, cadascun com a unitat independent apuntant al mateix backend via HTTPS. El HIS (Mirth Connect i servidor HAPI FHIR) quedaria sota responsabilitat del departament IT del centre mèdic client, i la comunicació entre el backend al cloud i el HIS on-prem es faria mitjançant un enllaç privat o VPN, embolcallant el trànsit MLLP sobre TLS tal com s'identifica a les limitacions (secció 6.3).

6.1 Implicacions per a la indústria i la recerca

Més enllà dels resultats concrets del projecte, el treball té implicacions més àmplies per a la indústria sanitària i per a la recerca en interoperabilitat. Des del punt de vista industrial, demostra que és factible comercialitzar un portal del pacient com a producte independent del HIS del centre, la qual cosa obriria la porta a un model de negoci on una empresa com PKF Attest pot oferir el portal als seus clients independentment del sistema de gestió clínica que tinguin instal·lat, reduint la dependència d'integradors especialitzats per a cada HIS.

Des del punt de vista de la recerca, el projecte contribueix al cos de coneixement sobre patrons d'integració en sistemes d'informació sanitària. En particular, documenta la decisió de descartar el patró FHIR Façana — que consistiria a posar una capa FHIR per damunt d'una BD relacional existent sense usar el protocol FHIR com a tal — en favor d'un servidor HAPI

FHIR independent que sí implementa el protocol de manera estàndard. Aquesta distinció és rellevant per a organitzacions que avaluin estratègies d'integració: la façana pot ser més ràpida d'implementar, però no valida la interoperabilitat real amb clients FHIR estàndard.

6.2 Valoració del compliment dels objectius

A continuació es valora el grau de compliment de cadascun dels nou objectius específics definits a la planificació inicial:

1. Anàlisi dels sistemes existents: Assolit. S'ha estudiat l'arquitectura dels principals estàndards d'integració sanitària (HL7 v2.5 i FHIR R4) i s'ha definit un contracte d'API genèric i reutilitzable mitjançant la interfície abstracta HISAdapter amb nou mètodes que cobreixen totes les operacions del portal.

2. Arquitectura oberta, escalable i desacoblada: Assolit. S'ha implementat una arquitectura basada en el patró adaptador que aïlla completament la lògica de negoci del protocol d'integració. La configuració del centre (protocol HIS, endpoint) es gestiona via variables d'entorn, sense cap canvi de codi.

3. Integració multi-sistema i multi-centre: Assolit amb concreció. L'objectiu s'ha concretat en un model white-label per centre amb instàncies independents: cada centre té el seu propi desplegament de backend i BD. No s'ha implementat routing multi-tenant, decisió que simplifica el codi i reforça l'aïllament de dades entre centres.

4. Implementació del backend: Assolit. El backend Node.js + TypeScript + Express exposa 25 endpoints REST, gestiona l'autenticació JWT de doble token, el xifratge AES-256-GCM de dades sensibles, el registre d'auditoria i la comunicació amb els dos adaptadors HIS.

5. Implementació del frontend web: Assolit. El portal web React inclou set pantalles funcionals amb circuit de reserva de cites en quatre passos, filtres de dates, modals de detall i gestió de qüestionaris i informes. Les bones pràctiques d'accessibilitat s'han aplicat parcialment; una validació formal WCAG 2.1 queda com a treball futur.

6. Implementació de les aplicacions mòbils: Assolit parcialment. S'ha desenvolupat l'aplicació Android completa amb React Native i Expo SDK 54, amb sis pantalles i una APK funcional generada via EAS Build. La versió iOS queda pendent com a extensió futura.

7. Seguretat i compliment normatiu: Assolit. S'han implementat autenticació JWT de doble token amb sliding expiration, xifratge AES-256-GCM de dades personals, hash SHA-256 per a cerques, bcrypt (cost 12) per a contrasenyes, registre d'auditoria complet, rate limiting a l'endpoint de login i capçaleres de seguretat HTTP via helmet. La verificació d'identitat del pacient (més enllà de la comprovació del document al HIS) queda com a requisit per al pas a producció.

8. Experiència d'usuari moderna i accessible: Assolit. El portal web i l'app mòbil segueixen un disseny modern i consistent, amb paleta corporativa, tipografia Manrope, animacions d'entrada i components reutilitzables. Les proves d'usabilitat amb usuaris reals i la validació formal WCAG 2.1 queden com a treball futur.

9. Verificació i validació: Assolit parcialment. S'han dissenyat 95 casos de prova al pla de proves i s'han

automatitzat 41 proves en 7 suïtes (Jest + Supertest), amb una cobertura del 67% de línies del codi backend exclosos els adaptadors externs. La validació amb l'adaptador HL7 s'ha realitzat manualment. Els casos de prova restants (mocking dels adaptadors, proves E2E i de seguretat avançada) queden identificats com a treball futur.

Des d'una perspectiva formativa, el projecte ha permès aplicar en un context professional real conceptes clau de la carrera: disseny d'arquitectures distribuïdes, patrons de disseny de software, seguretat d'aplicacions web, proves automatitzades i disseny d'interfícies d'usuari. La immersió en els estàndards HL7 i FHIR, i el treball conjunt amb professionals d'una empresa del sector sanitari, han representat una experiència formativa de gran valor per a la incorporació professional en l'àmbit de la salut digital.

6.3 Reflexió crítica sobre les limitacions del prototip

Tot i els resultats positius, cal fer una valoració honesta de les limitacions del prototip. La més significativa és l'absència de proves de càrrega. Les 41 proves implementades validen el comportament funcional amb un sol usuari concurrent, però no avaluen com es comporta el sistema amb centenars d'usuaris simultanis. En un entorn productiu, caldria una bateria de proves de rendiment (load testing) mitjançant eines com k6 o JMeter per identificar els colls d'ampolla i dimensionar la infraestructura.

Una segona limitació és la manca d'un sistema de notifikacions. En un portal clínic real, el pacient hauria de rebre recordatoris de les seves cites (correu electrònic o SMS), notifikacions quan el professional li enviés un qüestionari nou, i avisos quan un informe nou estigués disponible. Implementar-ho requeriria una infraestructura de missatgeria asíncrona (RabbitMQ, SQS) i integració amb serveis d'enviament com SendGrid o Twilio. Queda identificat com a tasca prioritària de continuïtat.

En tercer lloc, l'adaptador HL7 no implementa TLS sobre MLLP. En l'entorn de proves, backend i Mirth Connect s'executen a la mateixa xarxa local, de manera que la manca de xifratge no representa un risc real. Però en un desplegament productiu on estiguessin en servidors separats, caldria embolcallar MLLP sobre TLS per garantir la confidencialitat dels missatges HL7 en trànsit. Mirth Connect suporta TLS nativament i la modificació al costat del backend seria mínima, però excedia l'abast del TFG.

Una quarta limitació identificada afecta la consistència eventual en la reserva de cites amb l'adaptador FHIR. Quan es crea un nou recurs Appointment al servidor HAPI FHIR, el servidor retorna la confirmació immediatament però la indexació del recurs per a les cerques per paràmetre (GET /Appointment?actor=Patient/X) es fa de manera asíncrona en segon pla. Això provoca que, si el frontend consulta les cites immediatament després de crear-ne una, el servidor encara no retorna el nou recurs. Per mitigar-ho, el portal afegeix la nova cita directament a l'estat local del frontend en el moment de la confirmació, sense esperar la resposta del servidor, de manera que l'usuari la veu immediatament. Aquesta limitació és

inherent al comportament de HAPI FHIR en entorn local amb recursos limitats; en un servidor FHIR de producció amb PostgreSQL com a backend i recursos dedicats, la indexació seria pràcticament instantània.

Més enllà de les limitacions descrites, al llarg del document s'han anat identificant altres aspectes que caldria abordar en una versió de producció: l'absència de HTTPS en l'entorn local (les comunicacions entre frontend i backend no estan xifrades en trànsit, a diferència d'un desplegament real on tot el tràfic aniria per HTTPS); la manca de verificació d'identitat del pacient més enllà de la comprovació del document al HIS i la validació del sistema amb un volum de dades real i en un entorn de producció amb un HIS real, que podria revelar comportaments no anticipats en la latència, la paginació de resultats o la gestió d'errors del protocol. En conjunt, aquestes limitacions són inherents a la naturalesa del treball com a prova de concepte i no invaliden les conclusions arquitectòniques i tècniques del projecte.

ÚS DE LA IA GENERATIVA

Durant la realització d'aquest treball s'han emprat dues eines d'IA generativa. Claude s'ha usat com a eina de suport al desenvolupament: resolució de dubtes tècnics sobre TypeScript, Node.js, Express i React Native; depuració d'errors de compilació i d'execució; comprensió i interpretació dels estàndards HL7 v2.5 i FHIR R4 i els seus recursos; generació de dades simulades clínicament coherents per a les BDs de prova (mirth_his i seed FHIR); suport en la redacció i revisió d'apartats tècnics d'aquest informe i dels informes de progrés anteriors; i generació de recursos gràfics com diagrames de flux i figures il·lustratives. Figma Make s'ha emprat per generar un prototip visual de la interfície d'usuari durant la fase de disseny, abans de la implementació real amb React.

L'ús de la IA generativa s'ha limitat sempre a un rol de suport i acceleració, equivalent al d'un company tècnic amb qui contrastar idees o resoldre dubtes puntuals. En cap moment s'ha delegat a la IA la presa de decisions de disseny o arquitectura: totes les decisions rellevants — l'elecció del patró adaptador, el model de seguretat, l'estructura de la BD, la selecció de tecnologies — han estat preses i justificades per l'autor. Tot el codi generat o suggerit per la IA ha estat revisat, comprès, adaptat i validat críticament abans d'incorporar-se al projecte. Les decisions de disseny, l'arquitectura, la implementació i la validació del sistema són responsabilitat íntegra de l'autor.

- E-mail de contacte: xavisrd@gmail.com
- Menció realitzada: Enginyeria del Software
- Treball tutoritzat per: David Gutiérrez Yana (Departament de ciències de la computació)
- Curs 2025/26

AGRAÏMENTS

Vull agrair a l'empresa PKF Attest Health l'oportunitat de desenvolupar aquest projecte en un entorn professional real, posant a disposició els recursos tècnics i el context sanitari necessaris per fer d'aquest TFG una

experiència genuïnament professional. Molt especialment al tutor d'empresa, Oscar Palao Tomás, pel suport tècnic i l'acompanyament constants al llarg del semestre, per la disponibilitat per resoldre dubtes tècnics i per la confiança dipositada en l'autor durant la realització del treball.

Vull agrair també al tutor acadèmic, David Gutiérrez Yana, l'orientació, la supervisió i el feedback rebuts al llarg de tot el procés. La seva predisposició a realitzar reunions de seguiment setmanals, molt més sovint del que la normativa del TFG exigeix, ha estat fonamental per mantenir el focus, detectar desviacions a temps i garantir la qualitat acadèmica del treball.

Juny de 2026, Escola d'Enginyeria (UAB)

BIBLIOGRAFIA

- [1] Caristix, "HL7 Version 2.5.1 — Message and Segment Definitions." [En línia]. Disponible: <https://hl7-definition.caristix.com/v2/HL7v2.5.1>
- [2] HL7 International, "FHIR Release 4 (R4) Specification." [En línia]. Disponible: <https://hl7.org/fhir/R4/>
- [3] NextGen Healthcare, "Mirth Connect — Open Source Integration Engine." GitHub. [En línia]. Disponible: <https://github.com/nextgenhealthcare/connect>
- [4] Smile CDR Inc., "HAPI FHIR – Open Source FHIR API for Java." [En línia]. Disponible: <https://hapifhir.io>
- [5] PostgreSQL Global Development Group, "PostgreSQL 17 Documentation." [En línia]. Disponible: <https://www.postgresql.org/docs/17/>
- [6] OpenJS Foundation, "Node.js Documentation." [En línia]. Disponible: <https://nodejs.org>
- [7] D. Bender i K. Sartipi, "HL7 FHIR: An Agile and RESTful approach to healthcare information exchange," [En línia]. Disponible: <https://sartipi.azurewebsites.net/papers/CBMS2013.pdf>
- [8] M. Ayaz, M. Pasha, M. T. Alzahrani, R. Budiarto, D. Stiawan, "The Fast Health Interoperability Resources (FHIR) Standard: Systematic Literature Review of Implementations and Tools," JMIR Med. Inform., vol. 9, n.º 7, e21929, jul. 2021. [En línia]. Disponible: <https://medinform.jmir.org/2021/7/e21929>
- [9] Reglament (UE) 2016/679 del Parlament Europeu i del Consell, de 27 d'abril de 2016, relatiu a la protecció de les persones físiques en allò que respecta al tractament de dades personals (RGPD). DOUE L 119, 4.5.2016. [En línia]. Disponible: <https://eur-lex.europa.eu/legal-content/CA/TXT/?uri=CELEX%3A32016R0679>
- [10] OWASP Foundation, "OWASP Top Ten Web Application Security Risks, 2025 Edition." [En línia]. Disponible: <https://owasp.org/Top10/2025/>
- [11] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)," RFC 7519, Internet Engineering Task Force (IETF), maig 2015. Disponible: <https://datatracker.ietf.org/doc/html/rfc7519>
- [12] Comissió Europea, "Proposal for a Regulation on the European Health Data Space (EHDS)," COM(2022) 197 final, Brussel·les, maig 2022. [En línia]. Disponible: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52022PC0197>
- [13] Ministeri de Sanitat d'Espanya, Estrategia de Salud

Digital del Sistema Nacional de Salud 2021-2026. Madrid: Secretaría General de Salud Digital, Información e Innovación del SNS, 2022. [En línia]. Disponible: https://www.sanidad.gob.es/areas/saludDigital/doc/Estrategia_de_Salud_Digital_del_SNS.pdf

APÈNDIX

A1. COMPARATIVA D'IMPLEMENTACIÓ: CIRCUIT HL7 I FHIR

Aquest apèndix presenta una comparativa de les dues implementacions del patró adaptador prenent com a exemple l'operació de reserva de cita, que és el flux més representatiu del sistema. L'objectiu és il·lustrar com la mateixa operació s'implementa de manera radicalment diferent en cadascun dels dos protocols — missatgeria MLLP amb segments HL7 en un cas, API REST amb recursos JSON FHIR en l'altre — i com la interfície abstracta HISAdapter absorbeix completament aquesta diferència, retornant en tots dos casos el mateix tipus intern AppointmentInfo al backend.

El circuit HL7 implica tres actors: l'HL7Adapter (TypeScript), el transformer JavaScript de Mirth Connect (motor Rhino/JVM) i la BD mirth_his via JDBC. El circuit FHIR implica dos actors: el FHIRAdapter (TypeScript) i el servidor HAPI FHIR desplegat en Docker, que exposa una API REST FHIR R4 estàndard.

A1.1 SIU^S12 — Circuit complet de reserva de cita via HL7

A continuació es mostra el circuit complet de la creació d'una cita amb HL7.

```
async createAppointment(docId: string, slotId: string)
: Promise<AppointmentInfo> {

  // Construeix el missatge SIU^S12 amb els segments MSH, SCH i PID
  const msh = this.buildMSH('SIU^S12');
  const sch = `SCH|${slotId}|||||${slotId}`;
  const pid = `PID|||${docId}`;
  const message = [msh, sch, pid].join(this.SEG); // SEG = \r (CR)

  // Envia via MLLP i espera la resposta
  const response = await this.sendMessage(message);
  const lines = response.split('\r').filter(Boolean);

  // Comprova el segment MSA: AA = èxit, AE = error
  const msaLine = lines.find(l => l.startsWith('MSA'));
  const msa = msaLine ? this.parseSegment(msaLine) : [];
  if (msa[1] === 'AE') throw new Error('SLOT_NOT_FOUND');

  // Parseja el segment SCH de la resposta
  const schResp = lines.find(l => l.startsWith('SCH'));
  const schParts = schResp ? this.parseSegment(schResp) : [];
  const dateTime = schParts[11] || '';
  const dateRaw = dateTime.slice(0, 8);
  const timeRaw = dateTime.slice(8, 12);

  // Normalitza la resposta al tipus intern AppointmentInfo
  return {
    appointment_id: schParts[1] || slotId,
    slot_id: slotId,
    date: dateRaw ? `${dateRaw.slice(0,4)}-${dateRaw.slice(4,6)}-${dateRaw.slice(6,8)}` : '',
    time: timeRaw ? `${timeRaw.slice(0,2)}:${timeRaw.slice(2,4)}:`,
    specialty: schParts[9] || '',
    professional: schParts[14] || '',
    visit_type: schParts[6] || '',
    status: 'confirmed',
  };
}
```

Fig. 4. Funció createAppointment del HL7Adapter.

```
MSH|^~\&|Portal|PKF|HIS|Centre|20260628214705||SIU^S12|MSG1782683225136|P|2.5
SCH|slot-0025|||||slot-0025
PID|||12345678A
```

Fig. 5. Exemple de missatge HL7 generat per la funció createAppointment, enviat des del portal al transformer (via MLLP)

A1.2 Transformer: parsatge del MSH i enrutament

El primer bloc extreu el tipus de missatge del segment MSH i obre la connexió JDBC:

```
// Extreu el segment MSH per determinar el tipus de missatge
var mshFields = mshLine.split('|');
var messageType = (mshFields[8] || '').split('^')[0]; // QBP o SIU
var msgSub = (mshFields[8] || '').split('^')[1] || ''; // S12, S15, S26...
var msgId = mshFields[9] || '';

// Connexió JDBC a la BD mirth_his (PostgreSQL)
// Rhino permet instanciar classes Java directament des de JavaScript
java.lang.Class.forName('org.postgresql.Driver');
var dbConn = java.sql.DriverManager.getConnection(
  'jdbc:postgresql://host.docker.internal:5432/mirth_his',
  'postgres', ' '
);
```

Fig. 6. Fragment de codi del transformer que extreu la capçalera del missatge.

L'enrutament es fa a partir de messageType i msgSub: els missatges QBP cobreixen les consultes clíniques (pacient, visites, informes, qüestionaris), mentre que els missatges SIU gestionen les cites (S26 per llistar/consultar, S12 per reservar, S15 per cancel·lar).

A1.3 Transformer: lògica de reserva i resposta

Aquest és el flux més complex del transformer. Comprova primer si el slot és disponible, insereix la cita i marca el slot com a reservat en dues operacions consecutives:

```
} else if (msgSub === 'S12') {
  var slotId = (schFields[1] || '').trim(); // slot_id del segment SCH
  var docId = (pidFields[3] || '').trim(); // DNI/NIE del segment PID

  // 1. Comprova l'estat actual del slot
  var checkResult = dbConn.createStatement().executeQuery(
    "SELECT status FROM slots WHERE slot_id = '" + slotId + "'");
  var slotStatus = checkResult.next()
    ? ' ' + checkResult.getString('status') : '';

  if (slotStatus === 'booked') {
    // Slot ja reservat: retorna error MSA/AE
    response = 'MSH|^~\&|HIS|Centre|Portal|PKF|' + now +
      '||ACK|' + msgId + '|P|2.5\r';
    response += 'MSA|AE|' + msgId + '|Slot already booked\r';
  } else {
    // 2. Insereix la nova cita a la taula cites
    var insResult = dbConn.createStatement().executeQuery(
      "INSERT INTO cites (id_document, slot_id, date, time, " +
        "specialty, visit_type, professional_id, status) " +
        "VALUES ('" + docId + "', '" + slotId + "', '" + sDate +
        "', '" + sTime + "', '" + sSpec + "', '" + sVType +
        "', '" + sProfId + "', 'confirmed') RETURNING id");
    var newId = insResult.next() ? ' ' + insResult.getInt('id') : slotId;

    // 3. Marca el slot com a reservat
    dbConn.createStatement().executeUpdate(
      "UPDATE slots SET status = 'booked' WHERE slot_id = '" + slotId + "'");

    // 4. Retorna confirmació MSA/AA amb el segment SCH complet
    response = 'MSH|^~\&|HIS|Centre|Portal|PKF|' + now +
      '||SIU^S12|' + msgId + '|P|2.5\r';
    response += 'MSA|AA|' + msgId + '\r';
    response += 'SCH|' + newId + '|' + slotId + '||||' + sVType +
      '|||' + sSpec + '|||' + aDate + aTime +
      '||confirmed|' + sProf + '\r';
  }
}
```

Fig. 7. Fragment de codi del transformer que genera el missatge de resposta.

En cas d'èxit, el transformer retorna un segment SCH amb totes les dades de la cita creada (data, hora, especialitat, professional i estat confirmed), que l'HL7Adapter parseja i normalitza al tipus intern AppointmentInfo.

```
MSH|^~\&|HIS|Centre|Portal|PKF|20260628214705||$IU^S12|MSG1782683225136|P|2.5
MSA|AA|MSG1782683225136
SCH|55|slot-0025|||Ecocardiograma||Cardiologia||20260720900||confirmed|Dr. Ant
Serra
```

Fig. 8. Exemple de missatge de resposta que crea el transformer.

A1.4 Contrast FHIR — Reserva de cita via HTTP REST

La mateixa operació de reserva de cita que al transformer HL7 s'implementa al FHIRAdapter mitjançant crides HTTP estàndard a l'API REST FHIR R4. La diferència estructural és significativa: en lloc de construir un missatge de text pla i parsejat de segments, el FHIRAdapter crea un recurs JSON tipat i el publica al servidor HAPI FHIR:

```
async createAppointment(docId: string, slotId: string)
: Promise<AppointmentInfo> {

  // 1. Obté l'ID intern FHIR del pacient per document d'identitat
  const patientBundle = await this.request<any>(<any>{
    '/Patient?identifier=${docId}'
  });
  const patientId = patientBundle.entry[0].resource.id;

  // 2. Comprova que el slot és free (equivalent a SELECT status FROM slots)
  const slotCheck = await this.request<any>(<any>{ '/Slot/${slotId}'});
  if (slotCheck.status !== 'free') throw new Error('SLOT_NOT_AVAILABLE');

  // 3. Crea el recurs Appointment FHIR R4 via POST
  const appt = await this.request<any>(<any>{ '/Appointment', {
    method: 'POST',
    body: JSON.stringify({
      resourceType: 'Appointment',
      status: 'booked',
      start: `${date}T${time}:00:00:00`,
      end: `${date}T${time}:30:00:00`,
      appointmentType: visit_type ? { text: visit_type } : undefined,
      serviceType: specialty ? [{ text: specialty }] : undefined,
      slot: [{ reference: 'Slot/${slotId}'}],
      participant: [{
        actor: { reference: 'Patient/${patientId}'} ,
        status: 'accepted'
      }],
    }
  )}, {
  });

  // 4. Marca el Slot com a busy (equivalent a UPDATE slots SET status='booked')
  const slot = await this.request<any>(<any>{ '/Slot/${slotId}'});
  await this.request<any>(<any>{ '/Slot/${slotId}', {
    method: 'PUT',
    body: JSON.stringify({ ...slot, status: 'busy' } ),
  });

  return {
    appointment_id: appt.id, slot_id: slotId,
    date, time, specialty, professional, visit_type,
    status: 'confirmed'
  };
}
```

Fig. 9. Funció createAppointment del FHIRAdapter.

Tot i que la lògica és equivalent a la del transformer HL7 — comprova disponibilitat, crea la reserva, marca el recurs com a ocupat — la implementació és radicalment diferent: JSON estructurat en lloc de segments pipe-delimited, HTTP en lloc de MLLP, i recursos tipats FHIR en lloc de taules SQL. El patró adaptador absorbeix completament aquesta diferència: la resta del portal rep exactament el mateix AppointmentInfo independentment del protocol actiu.

A continuació es mostra la petició HTTP real que el FHIRAdapter envia al servidor HAPI FHIR en crear una cita, capturada mitjançant un console.log durant l'execució del sistema en entorn local, seguida de la

resposta normalitzada que el FHIRAdapter construeix a partir del recurs FHIR retornat pel servidor i que el backend retorna al frontend:

```
{
  "resourceType": "Appointment",
  "status": "booked",
  "start": "2026-07-24T09:00:00+00:00",
  "slot": [
    { "reference": "Slot/1625" }
  ],
  "participant": [
    {
      "actor": { "reference": "Patient/1000" },
      "status": "accepted"
    }
  ]
}
```

Fig. 10. Exemple del JSON que s'envia a la petició POST.

```
{
  "appointment_id": "1788",
  "slot_id": "1625",
  "date": "2026-07-24",
  "time": "09:00",
  "specialty": "Pediatría",
  "professional": "Dra. Núria Costa",
  "visit_type": "Primera visita",
  "status": "confirmed"
}
```

Fig. 11. Exemple del JSON rebut com a resposta, creant un Appointment nou.

A2. IMATGES DEL PORTAL WEB I L'APLICACIÓ MÒBIL

A continuació es mostren captures de pantalla del Portal del Pacient en funcionament, tant en la versió web com en l'aplicació mòbil Android. Les captures il·lustren les principals pantalles del sistema i demostren la coherència visual entre les dues plataformes, que comparteixen la mateixa paleta corporativa i la mateixa estructura de navegació.

En primer lloc, es presenten les imatges del portal web i, a continuació, les de l'aplicació mòbil, mostrant el mateix contingut per al mateix pacient de prova:

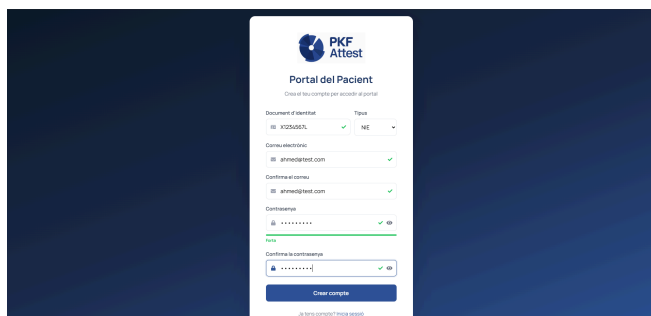


Fig. 12. Pantalla web registre.

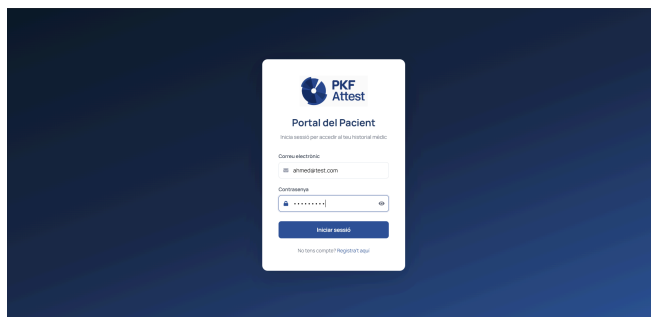


Fig. 13. Pantalla web login.

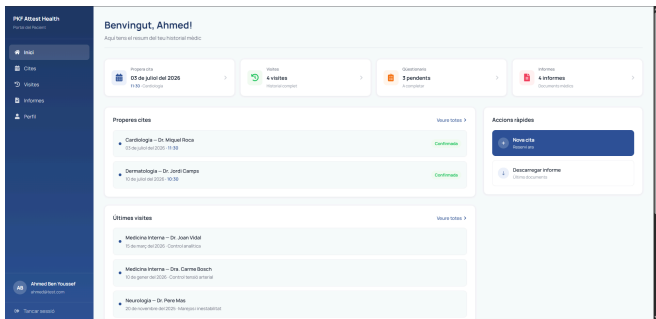


Fig. 14. Pantalla web dashboard.

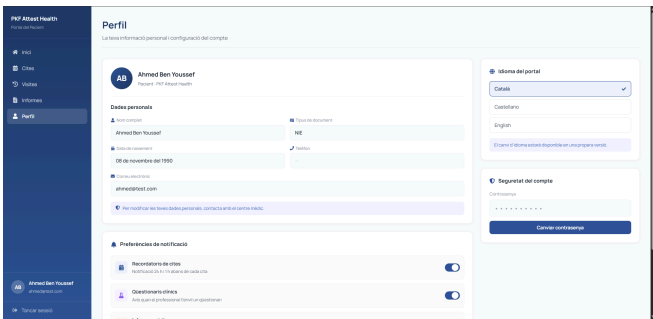


Fig. 19. Pantalla web perfil.

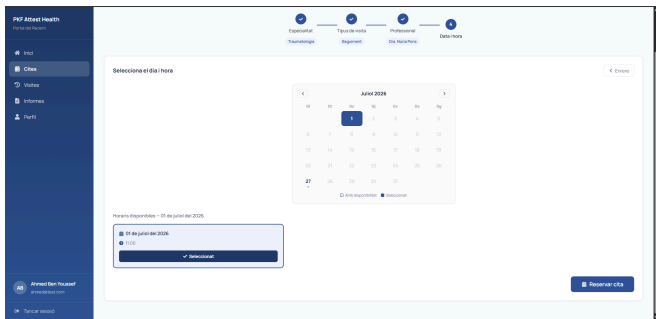


Fig. 15. Pantalla web de l'últim pas del circuit de demanar cita.

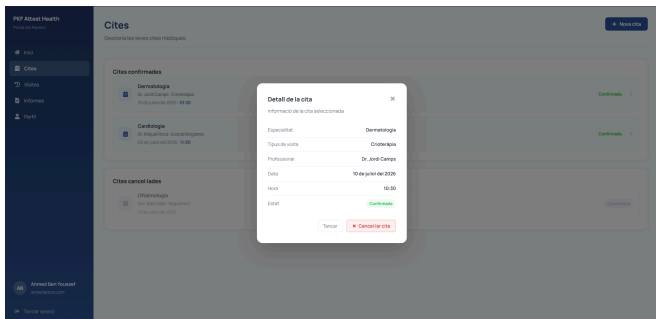


Fig. 16. Pantalla web del detall d'una cita.

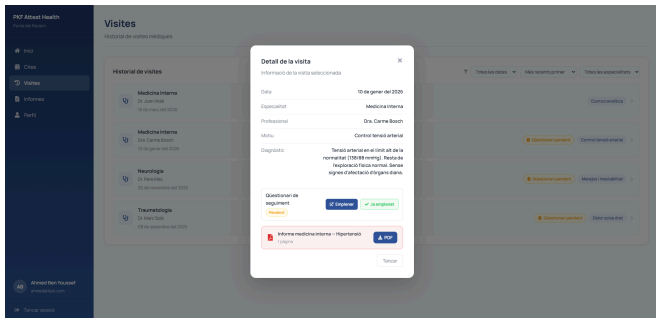


Fig. 17. Pantalla web del detall d'una visita.

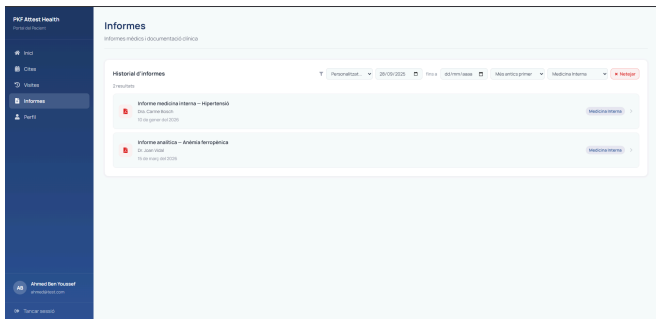


Fig. 18. Pantalla web del llistat d'informes, mostrant també l'opció dels filtres.

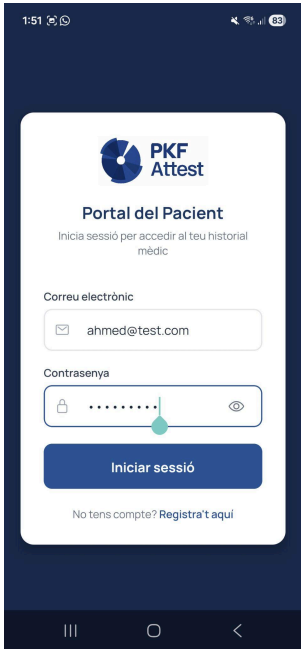
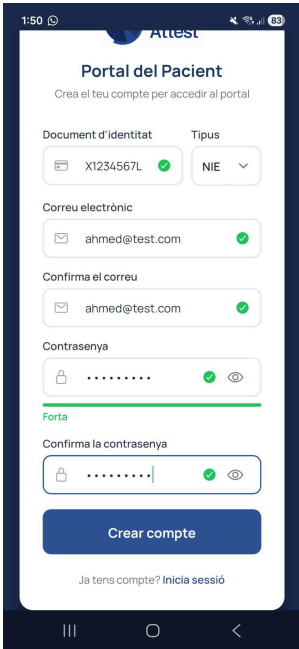


Fig. 20. Pantalles app Android, a l'esquerra el registre i a la dreta el login.

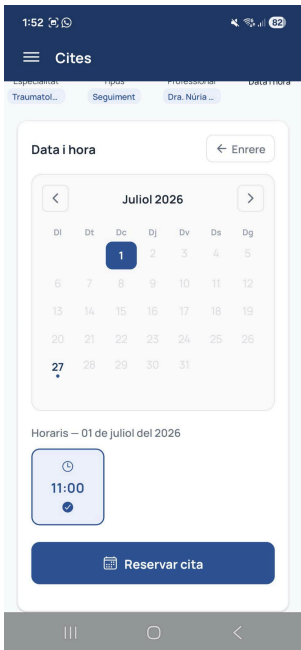


Fig. 21. Pantalles app Android, dashboard i demanar cita.

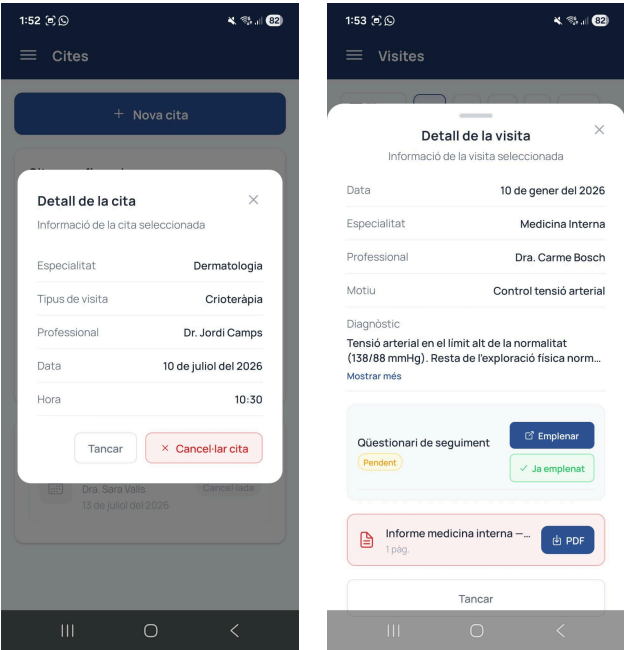


Fig. 22. Pantalles app Android, detall cita i detall visita.

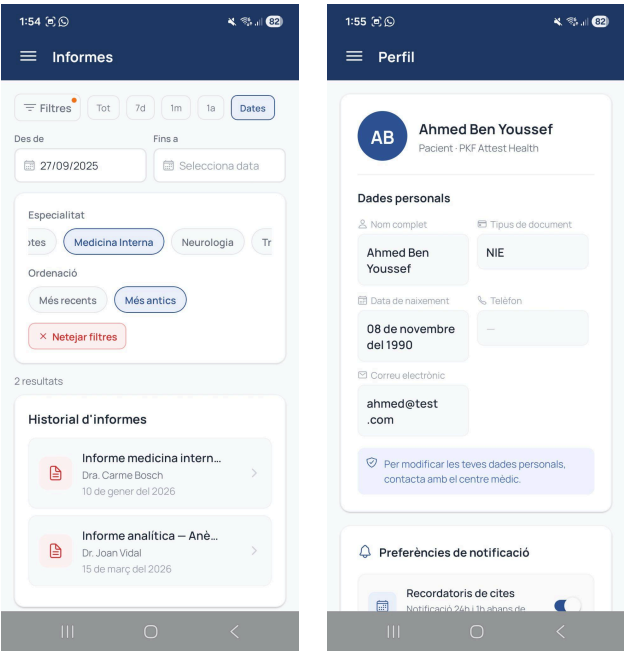


Fig. 23. Pantalles app Android, llistat informes amb filtres i pantalla de perfil.

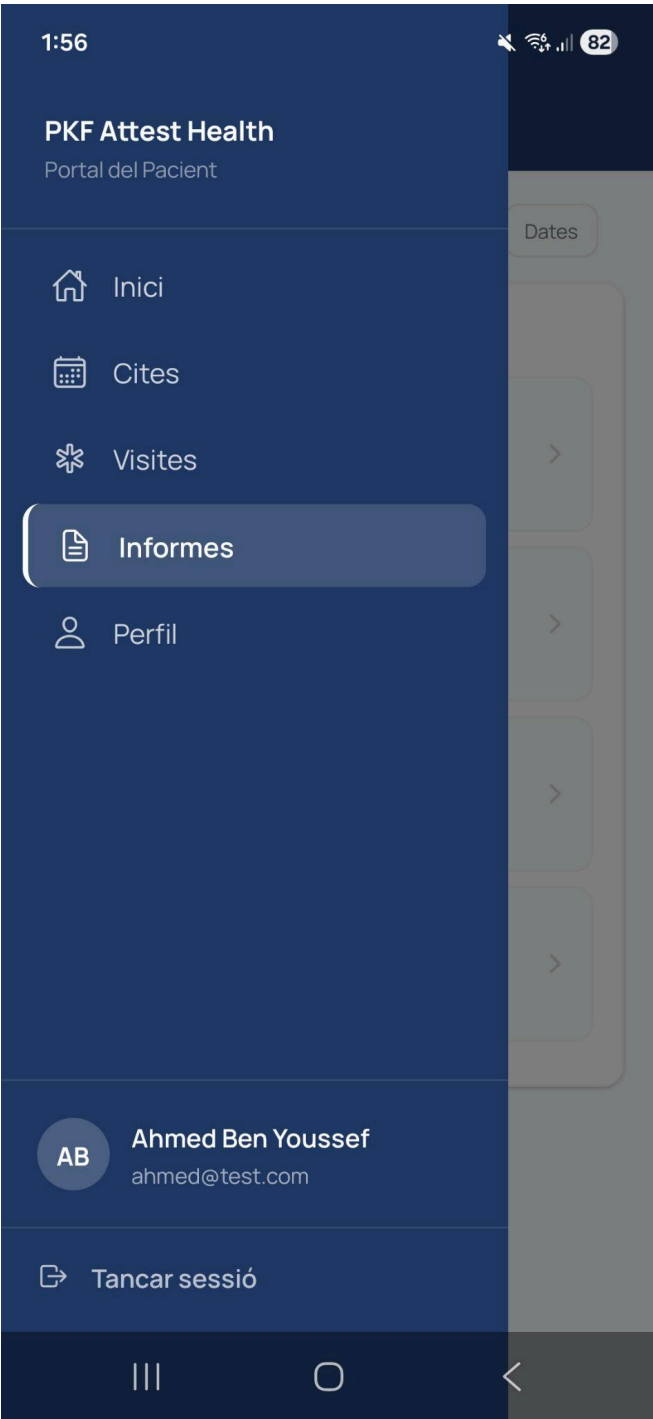


Fig. 24. Pantalla app Android, mostrant com és el drawer lateral.