Institut Puig Castellar

Cicle Formatiu de Grau Superior

Desenvolupament d'Aplicacions Multiplataforma (DAM)

Projecte Intermodular

Sistema de gestió de fitxatge laboral amb NFC

Autors: Manuel Navarro Cortés i Carlos Patricio Paredes

Tutor: Luís Elía

Curs: 2025-2026

Data d'entrega: 15/05/2026

Resum del projecte

EntroYa es un sistema de gestió de fitxatge laboral dissenyat per a petites i mitjanes empreses. El projecte permet als treballadors fitxar l'entrada i la sortida mitjançant una targeta NFC llegida per una aplicació Android instal·lada de forma fixa a l'empresa. Els registres es transmeten en temps real a un servidor backend desenvolupat amb Spring Boot, que els emmagatzema en una base de dades PostgreSQL allotjada al cloud. A mes del fitxatge, el sistema incorpora un panell de control web, desenvolupat amb React i Vite, que permet als administradors gestionar usuaris, horaris, nòmines i justificants d'absència. Els treballadors, per la seva banda, poden consultar el seu historial de fitxatges, descarregar les seves nòmines i sol·licitar justificants amb documents adjunts. El projecte s'ha desenvolupat seguint una metodologia iterativa i incremental, repartint les tasques entre els dos membres de l'equip al llarg de tot el curs.

Paraules clau

Fitxatge NFC, Spring Boot, React, PostgreSQL, Android, gestio laboral, API REST, autenticació

Abstract

EntroYa is a workforce attendance management system designed for small and medium-sized businesses. Employees clock in and out using an NFC card read by an Android application installed permanently on the company premises. Records are sent in real time to a Spring Boot backend, which stores them in a cloud-hosted PostgreSQL database. In addition to attendance tracking, the system includes a React-based web control panel that allows administrators to manage users, schedules, payroll documents and absence justifications. Workers can view their clock history, download their payslips and submit justification requests with file attachments. The project was developed using an iterative and incremental methodology, with tasks distributed between the two team members throughout the academic year.

Keywords

NFC attendance, Spring Boot, React, PostgreSQL, Android, workforce management, REST API, authentication, BCrypt, Kotlin

Index

1. Presentació del projecte

1.1 Introducció

EntroYa es un sistema integral de gestió de fitxatge laboral pensat per a empreses que necessiten controlar l'assistència dels seus treballadors de forma automática, fiable i senzilla. El sistema està format per tres components principals: una aplicació Android de fitxatge amb NFC que es queda instal·lada a l'empresa, un backend REST desenvolupat amb Spring Boot que centralitza tota la lògica de negoci i la persistencia de dades, i un panell de control web desenvolupat amb React que permet tant als administradors com als treballadors gestionar la seva informació laboral des de qualsevol dispositiu.

L'aplicació Android permet als treballadors fitxar l'entrada i la sortida simplement acostant la seva targeta NFC al dispositiu. La pantalla nomes mostra un missatge d'espera i, en detectar la targeta, confirma el fitxatge. Tota la logica de processament es realitza al backend, que determina si el fitxatge es una entrada o una sortida en funció de l'últim registre del treballador.

1.2 Context

El control d'assistència laboral es un procés crític en qualsevol empresa. Tradicionalment s'ha gestionat amb fitxes manuals, sistemes de codi de barres o targetes de proximitat connectades a maquinari propietari i molt coster. En el context actual, on la tecnologia mòbil i els serveis cloud son accessibles per a qualsevol empresa, existeix una oportunitat clara per desenvolupar solucions més econòmiques, flexibles i connectades.

El projecte sorgeix en el marc del cicle formatiu de Grau Superior de Desenvolupament d'Aplicacions Multiplataforma (DAM) de l'Institut Puig Castellar, com a projecte intermodular del curs 2025-2026. L'objectiu es aplicar els coneixements adquirits al llarg del cicle en un projecte real i complet, que integri tecnologies de backend, frontend web i desenvolupament d'aplicacions Android.

1.3 Justificació

La gestió manual del fitxatge laboral comporta errors humans, pèrdua de temps i dificultat per obtenir informes. Les solucions comercials existents solen ser cares, tancades i poc adaptables a les necessitats específiques de cada empresa. EntroYa neix com una alternativa de codi obert, modular i extensible, que permet a una empresa tenir el control total sobre les seves dades sense dependre de tercers.

A més, el projecte representa un repte tècnic complet: cal dissenyar una base de dades relacional, implementar una API REST, desenvolupar una interfície web moderna i crear una aplicació Android nativa que interaccioni amb hardware NFC real. Aquesta amplitud tecnologica el converteix en un projecte ideal per demostrar les competències adquirides durant el cicle formatiu.

1.4 Objectius

Els objectius generals del projecte son:

- Desenvolupar un sistema complet de fitxatge laboral que integri hardware NFC, backend i frontend web.
- Garantir que els registres de fitxatge siguin automatics, fiables i accessibles en temps real.
- Oferir un panell de gestió intuitu per a administradors i treballadors.

Els objectius específics son:

- Implementar una API REST amb Spring Boot que gestioni usuaris, fitxatges, horaris, nòmines i justificants.
- Desenvolupar una aplicació Android nativa amb Kotlin que llegeixi targetes NFC i comuniqui els fitxatges al backend.
- Crear un panell web amb React que permeti als administradors gestionar tota la informació laboral.
- Permetre als treballadors consultar el seu historial, descarregar nòmines i sol·licitar justificants amb arxius adjunts.
- Emmagatzemar els documents (nòmines i justificants) directament a la base de dades de forma segura.
- Desplegar el sistema en un entorn cloud accessible des de qualsevol lloc.

2. Estratègia i planificació

2.1 Estrategia de desenvolupament i viabilitat

El projecte s'ha plantejat des d'un inici com un sistema modular, on cada component (backend, frontend i app Android) pot desenvolupar-se i provar-se de forma independent. Aquesta decisió ha permès que els dos membres de l'equip treballessin en paral·lel sense bloquejar-se mútuament, integrant els components a mesura que cadascun estava funcional.

Des del punt de vista tècnic, el projecte és viable perquè totes les tecnologies utilitzades son de codi obert, gratuites i amb una comunitat activa. Spring Boot proporciona un framework robust per al backend, React permet construir interficies modernes, i Android Studio ofereix les eines necessàries per al desenvolupament natiu amb NFC. La base de dades PostgreSQL s'allotja a Supabase, que ofereix un nivell gratuit suficient per al volum de dades d'aquest projecte.

Des del punt de vista temporal, el projecte ha estat planificat per encabir-se en el període lectiu comprès entre l'octubre del 2025 i el maig del 2026, amb una entrega final prevista per al 18 de maig del 2026. La càrrega de treball s'ha repartit de forma equilibrada entre els dos membres de l'equip.

2.2 Metodologia de treball

S'ha utilitzat una metodologia iterativa i incremental, inspirada en els principis àgils. En lloc de planificar tot el sistema des del principi i desenvolupar-lo de forma seqüencial, s'ha optat per anar construint i validant funcionalitats de forma progressiva, prioritzant les mes crítiques i afegint complexitat en cada iteració.

Les característiques principals de la metodologia aplicada han estat:

- Iteracions curtes: cada funcionalitat és dissenyava, implementava i provava de forma independent abans de passar a la següent.
- Divisió de tasques: Manuel s'ha encarregat principalment del backend (Spring Boot, base de dades, API REST) i de l'aplicació Android, mentre que Carlos ha treballat principalment en el frontend web (React, Vite, Bootstrap).
- Integració continua: els dos components es provaven conjuntament de forma regular per detectar problemes d'integració el més aviat possible.
- Revisió i refactoritzacio: a mesura que el sistema creixia, s'han revisat i millorat parts anteriors per garantir la coherència i la qualitat del codi.

La comunicació entre els membres de l'equip s'ha fet principalment de forma presencial durant les hores de classe, complementada amb eines de missatgeria per coordinar tasques fora de l'horari escolar.

2.3 Planificació

El projecte s'ha dividit en les següents fases:

Fase 1 - Anàlisi i disseny (octubre - novembre 2025)

- Definició dels requisits funcionals i no funcionals.
- Disseny del model de dades i les relacions entre entitats.
- Selecció de tecnologies i configuració dels entorns de desenvolupament.
- Disseny inicial de la interfície d'usuari (wireframes).

Fase 2 - Backend i base de dades (novembre - desembre 2025)

- Implementació de les entitats JPA i la base de dades PostgreSQL a Supabase.
- Desenvolupament dels controladors REST: autenticació, usuaris, fitxatges, horaris, nòmines i justificants.
- Configuració de Spring Security amb BCrypt per a la gestió de contrasenyes.
- Proves de l'API amb Postman.

Fase 3 - Aplicació Android NFC (desembre 2025 - gener 2026)

- Desenvolupament de l'aplicació Android amb Android Studio i Kotlin.
- Implementació de la lectura de targetes NFC i comunicació amb Supabase via Postgrest.
- Implementació del cooldown anti-duplicats de 5 minuts.
- Proves amb targetes NFC reals i ajust del flux d'entrada/sortida.

Fase 4 - Frontend web (gener - març 2026)

- Desenvolupament del panell de control amb React i Vite.
- Implementació de les vistes d'administrador: gestió d'usuaris, horaris, nòmines, justificants.
- Implementació de les vistes de treballador: historial, nòmines, justificants i horaris.
- Integració amb el backend via Axios i configuració del proxy de Vite.
- Implementació de la gestió de fitxers (pujada i descarga de PDFs i imatges).
- Redisseny visual complet amb sistema de disseny propi (DM Sans, variables CSS).

Fase 5 - Proves, ajustos i documentació (abril - maig 2026)

- Proves integrals del sistema complet.
- Correcció d'errors i millores de la interfície.

- Implementacio de la pagina d'inici (landing page) publica.
- Desplegament a Railway com a monorepository (backend + frontend compilat).
- Redacció de la memoria del projecte.
- Preparació de la presentació final.

3. Anàlisi

3.1 Casos d'ús

El sistema té dos tipus d'usuari principals: l'Administrador i el Treballador.

Administrador:

- Iniciar sessió al panell web.
- Crear, editar i eliminar usuaris.
- Assignar i desactivar horaris laborals.
- Pujar nòmines en PDF i gestionar-les per treballador.
- Revisar justificants d'absència pendents i aprovar-los o rebutjar-los.
- Consultar el historial de fitxatges de qualsevol treballador.
- Consultar tots els justificants d'un treballador concret.
- Gestionar les targetes NFC vinculades als treballadors (Android).

Treballador:

- Fitxar entrada i sortida acostant la targeta NFC al dispositiu Android de l'empresa.
- Iniciar sessió al panell web.
- Consultar el seu historial complet de fitxatges agrupat per dies.
- Veure el resum setmanal de fitxatges des del dashboard.
- Sol·licitar justificants d'absència amb descripció i document adjunt opcional.
- Consultar l'estat dels seus justificants (pendent, aprovat, rebutjat).
- Descarregar les seves nòmines en PDF.
- Consultar el seu horari laboral actiu.

3.2 Requisits funcionals

Els requisits funcionals descriuen que ha de fer el sistema:

Autenticació i control d'accés:

- El sistema ha de permetre l'inici de sessió amb email i contrasenya.
- Els usuaris han de tenir un rol (ADMIN o TRABAJADOR) que determina les funcionalitats accessibles.
- Les rutes del frontend han d'estar protegides segons el rol de l'usuari.

Fitxatge NFC:

- L'aplicació Android ha de detectar automáticamente targetes NFC i registrar el fitxatge.
- El sistema ha de determinar si el fitxatge es una entrada o sortida en funció de l'últim registre del treballador.
- El sistema ha d'implementar un cooldown de 5 minuts per evitar fitxatges duplicats o entrada i sortida instantània.
- El sistema ha de registrar data i hora exacta de cada fitxatge.

Gestió de documents:

- El sistema ha de permetre pujar arxius PDF, JPG i PNG com a justificants i nòmines.
- Els arxius s'han d'emmagatzemar a la base de dades i ser descarregables posteriorment.
- Les respostes JSON de l'API no han d'incloure el contingut binari dels arxius per evitar problemes de rendiment.

Gestió administrativa:

- L'administrador ha de poder crear, editar i eliminar usuaris.
- L'administrador ha de poder assignar horaris i desactivar-los.
- L'administrador ha de poder aprovar o rebutjar justificants amb comentaris.
- L'administrador ha de poder consultar historials de fitxatge i justificants de qualsevol treballador.

3.3 Requisits no funcionals

- Usabilitat: la interfície web ha de ser intuitiva, responsive i accessible des de qualsevol navegador modern.
- Rendiment: les respostes de l'API han de ser ràpides, evitant la serialitzacio de dades binàries grans en els llistats.
- Seguretat: les contrasenyes s'han d'emmagatzemar de forma xifrada (BCrypt). Les rutes de l'API han d'estar protegides en entorn de producció.

- Escalabilitat: l'arquitectura modular ha de permetre afegir nous mòduls (vacances, informes, etc.) sense modificar els existents.
- Disponibilitat: el backend s'ha de poder desplegar en un entorn cloud accessible des de qualsevol lloc.
- Mantenibilitat: el codi ha d'estar organitzat en capes (controladors, repositoris, models, DTOs) per facilitar el manteniment i l'extensió.

3.4 Anàlisi d'alternatives tecnològiques

Durant la fase de planificació es van analitzar diverses alternatives tecnològiques per a cada component del sistema:

Backend:

Es va considerar utilitzar Node.js amb Express com a alternativa a Spring Boot. Tot i que Node.js ofereix una corba d'aprenentatge més suau i és molt adequat per a APIs senzilles, Spring Boot ofereix un ecosistema més madur per a aplicacions empresarials, una integració nativa amb JPA per a la persistència de dades, i es la tecnologia principal del cicle formatiu DAM. Per aquests motius es va optar per Spring Boot.

Frontend web:

Es van considerar Angular i Vue.js com a alternatives a React. Angular és molt complet pero te una corba d'aprenentatge elevada i genera un codi mes verbose. Vue.js és més senzill pero te un ecosistema més petit. React es va escollir per la seva popularitat, la gran quantitat de recursos disponibles i la seva flexibilitat per construir interficies complexes de forma modular.

Base de dades:

Es va considerar MySQL com a alternativa a PostgreSQL. Ambdues son bases de dades relacionals madures i compatibles amb Spring Data JPA. Es va optar per PostgreSQL per la seva millor gestió de tipus de dades complexos (com els camps LOB per emmagatzemar arxius binaris) i per què Supabase, la plataforma cloud escollida, esta construida sobre PostgreSQL i ofereix un nivell gratuit molt generosa.

Aplicació de fitxatge:

Es va valorar la possibilitat d'utilitzar Flutter per al desenvolupament de l'aplicació Android, cosa que hauria permès tenir una única base de codi per a Android i iOS. No obstant, l'accés al hardware NFC en Flutter requereix plugins de tercers menys estables. Es va optar per Android natiu amb Kotlin, que ofereix accés directe i fiable a les APIs NFC d'Android, garantint un funcionament correcte amb el hardware real.

4. Disseny

4.1 Arquitectura del sistema

EntroYa segueix una arquitectura de tres capes clarament separades, on cada component té una responsabilitat única i es comunica amb els altres a traves d'interficies ben definides.

Capa de presentació

Formada per dues aplicacions independents: el panell web desenvolupat amb React i Vite, que s'executa al navegador del client, i l'aplicació Android de fitxatge, que s'executa en un dispositiu fix a l'empresa. Ambdues es comuniquen amb el backend exclusivament a traves de l'API REST via protocol HTTP/JSON.

Capa de negoci (Backend)

Desenvolupada amb Spring Boot (Java 21), conte tota la lògica de negoci del sistema: autenticació d'usuaris, gestió de fitxatges, càlcul de resums setmanals, processament de documents i control de permisos. S'organitza en controladors REST, repositoris JPA i models d'entitat. El backend s'executa al port 8080 i el desplegament final es realitza a Railway com a monorepository, servint tant l'API REST com el frontend compilat des del mateix servei. Un WebConfig redirigeix les rutes de la SPA a index.html, i un CustomErrorController gestiona els errors 404 reenviant-los al frontend.

Capa de persistencia

PostgreSQL allotjat a Supabase (cloud). L'accés a la base de dades es gestiona a través de Spring Data JPA amb Hibernate com a implementació ORM. Els arxius binaris (nòmines i justificants) s'emmagatzemen directament a la base de dades com a camps LOB, evitant la necessitat d'un sistema d'arxius extern.

El flux d'una petició típica es el següent:

- L'usuari interacciona amb la interfície (web o Android).
- La capa de presentació envia una petició HTTP a l'API REST del backend.
- El controlador Spring Boot rep la petició, executa la lògica de negoci i consulta la BD via JPA.
- El backend retorna una resposta JSON (o bytes en cas de descarga d'arxius).
- La capa de presentació mostra el resultat a l'usuari.

4.2 Model de dades

La base de dades està formada per 6 taules principals amb les següents relacions:

Taula	Descripció i camps principals
usuarios	id, email (unique), password (BCrypt), nombre, rol (ADMIN/TRABAJADOR), departamento
fichajes	id, usuario_id (FK), fechaHora, tipo (ENTRADA/SALIDA)
justificantes	id, usuario_id (FK), tipo, fecha, descripcion, estado (PENDIENTE/APROBADO/RECHAZADO), comentariosAdmin, archivoNombre, archivoTipo, archivo (LOB)
nominas	id, usuario_id (FK), mes, año, archivoNombre, archivoTipo, archivoTamanio, archivoContenido (LOB), comentarios
horarios	id, usuario_id (FK), nombre, horaEntrada, horaSalida, diasSemana, activo
tarjetas_nfc	id, usuario_id (FK), cardUid (unique), activa

Totes les taules tenen una relació Many-to-One amb la taula usuarios (un usuari pot tenir molts fitxatges, molts justificants, moltes nomines, etc.). Els camps d'arxiu (byte[]) estan marcats com @Lob i @Basic(fetch = LAZY) per evitar que Hibernate carregui el contingut binari en les consultes de llistat, prevenint l'error 'Unable to access lob stream' i millorant el rendiment.

4.3 Disseny de la interfície

La interfície web esta dissenyada amb React, Bootstrap 5 i React Router DOM i un sistema de disseny propi basat en variables CSS i la tipografía DM Sans. S'ha prioritzat la usabilitat i la coherencia visual, amb un esquema de colors blau corporatiu consistent amb la identitat de l'aplicació Android.

Estructura de navegació

La navbar superior cambia dinámicamente segons el rol de l'usuari autenticat. Els administradors veuen: Dashboard, Usuaris, Justificants, Horaris i Nòmines. Els treballadors veuen: Mi Panel, Historial, Justificants, Horaris i Nòmines. Les rutes estan protegides amb un component ProtectedRoute que redirigeix al login si l'usuari no està autenticat o no te el rol adequat.

Panell d'administrador

- Dashboard: estadístiques en temps real (total usuaris, treballadors, fitxatges del dia, justificants pendents) i botons d'accio rapida.
- Gestió d'usuaris: taula amb tots els usuaris, modal per crear/editar, eliminació amb confirmació, accés al historial de fitxatge de cada treballador.
- Justificants: llistat de pendents amb opcions d'aprovar/rebutjar, selector per veure l'historial complet d'un treballador.
- Horaris: llistat de tots els horaris assignats, modal per assignar nous horaris, desactivació.
- Nòmines: llistat de totes les nòmines, modal per pujar nous PDFs, descarga i eliminacio.

Panell de treballador

- Dashboard: resum setmanal, últim fitxatge i registres del dia actual.
- Historial: tots els fitxatges agrupats per dies amb resum de totals.
- Justificants: dues pestanyes internes, sol·licitar (formulari amb arxiu adjunt opcional) i historial (estat de cada justificant).
- Horaris: visualització de l'horari actiu assignat.
- Nòmines: llistat de nòmines disponibles amb descarga directa en PDF.

Aplicació Android

Dissenyada com una pantalla unica minimalista amb arquitectura MVVM i desenvolupada en Kotlin. Mostra un missatge d'espera ('Acosta la targeta NFC...') i, en detectar una targeta valida, mostra el nom del treballador i el tipus de fitxatge (ENTRADA o SORTIDA) mitjancant un overlay visual de gran mida durant uns segons. Incorpora un sistema de cooldown de 5 minuts per evitar fitxatges duplicats accidentals i un BroadcastReceiver per detectar canvis d'estat del NFC en temps real.

Landing page

El sistema inclou una pagina d'inici publica (landing.html) accessible a la ruta /home, amb animacions CSS i disseny dark mode, que presenta el projecte i permet acces a la webapp i a la descarga de l'APK.

5. Desenvolupament

5.1 Estructura del projecte

El projecte està organitzat en tres repositoris/carpetes independents:

Backend (Spring Boot)

```
entroya-backend/

src/main/java/com/entroya/

controller/ -> AuthController, AdminController, TrabajadorController,

JustificanteController, NominaController,

HorarioController, NfcController, CustomErrorController

model/ -> Usuario, Fichaje, Justificante, Nomina, Horario, TarjetaNfc

repository/ -> Interficies JPA per a cada entitat

dto/ -> JustificanteDTO, NominaRequest, HorarioRequest

config/ -> SecurityConfig (BCrypt), WebConfig

src/main/resources/

application.properties -> Connexio Supabase amb variables, config JPA,

static/ -> Frontend compilat (npm run build), landing.html+captures+apk
```

Frontend web (React + Vite)

```
entroya-frontend/

src/

components/

admin/ -> Dashboard, UserManagement, JustificationsReview,

ScheduleManagement, PayrollManagement, NFCMgmt,

WorkerHistorialAdmin, WorkerJustificantesAdmin

worker/ -> Dashboard, ClockHistory, RequestJustification,
```

```
ScheduleView, PayrollView

shared/ -> Layout, Navbar, ProtectedRoute, Modal

context/ -> AuthContext (gestio de sessió)

services/ -> api.js (tots els serveis Axios)

 utils/ -> toast.jsx (notificacions amb react-hot-toast)

vite.config.js -> Proxy /api -> http://localhost:8080
```

Aplicació Android (Android Studio)

```
entroya-android/

app/src/main/java/com/entroya/

MainActivity.java -> Lectura NFC i enviament al backend

AndroidManifest.xml -> Permisos NFC declarats
```

5.2 Implementació de funcionalitats

Autenticació i gestió de sessions

El login envia email i contrasenya al endpoint POST /api/auth/login. El backend cerca l'usuari per email i compara la contrasenya amb BCryptPasswordEncoder.matches(). Si es correcte, retorna un objecte usuari (id, nom, email, rol) que el frontend emmagatzema al localStorage i gestiona a traves d'un AuthContext de React. El ProtectedRoute comprova el rol abans de renderitzar cada ruta.

Fitxatge NFC

L'aplicacio Android detecta targetes NFC via NfcAdapter i escolta events NFC_DISCOVERED. En detectar una targeta, consulta l'ultim registre de l'usuari a Supabase. Si l'ultim estat fou ENTRADA, registra una SORTIDA automaticament, i viceversa. Un cooldown de 5 minuts impedeix fitxatges duplicats accidentals. L'hora s'envia formatada com yyyy-MM-dd HH:mm:ss per evitar desfasaments de zones horaries. Un BroadcastReceiver detecta canvis d'estat del NFC en temps real sense reiniciar l'app.

Gestió de documents (PDFs i imatges)

Tant les nòmines com els justificants suporten arxius adjunts. El frontend envia l'arxiu com multipart/form-data. El backend el rep com MultipartFile, n'extreu el nom, el tipus MIME i el contingut com byte[], i ho emmagatzema a la base de dades en un camp @Lob marcat com @Basic(fetch = LAZY). Per evitar l'error 'Unable to access lob stream' de PostgreSQL, els llistats utilitzen DTOs que no accedeixen al camp binari, usant el camp archivoNombre com indicador de si existeix arxiu. La descarga es fa via un endpoint dedicat amb @Transactional

que retorna ResponseEntity<byte[]> amb els headers Content-Type i Content-Disposition correctes. Al frontend, la descarga es gestiona creant un Blob en memoria i simulant un click en un element <a> invisible.

Seguretat de contrasenyes

Les contrasenyes s'encripten amb BCrypt (factor de cost per defecte: 10) usant BCryptPasswordEncoder de Spring Security. En crear o actualitzar un usuari, s'aplica passwordEncoder.encode(). En el login, s'utilitza passwordEncoder.matches() per comparar la contrasenya introduïda amb el hash emmagatzemat.

Notificacions i UX

Totes les alertes del sistema (confirmacions, errors, exits) utilitzen react-hot-toast amb estil propi (fons blanc, vores de color semantic). Les confirmacions destructives (eliminar usuari, eliminar nomina) mostren un toast personalitzat amb botons Confirmar/Cancelar en lloc del window.confirm natiu del navegador.

Patrons tècnics destacats

- DTO pattern: les entitats amb camps LOB (Justificante, Nomina) s'exposen sempre a través de DTOs que exclouen el contingut binari, evitant problemes de serialitzacio i millorant el rendiment.
- Proxy Vite: en desenvolupament, el proxy de Vite redirigeix totes les peticions /api al backend en el port 8080.
- Fetch API per descàrregues: les descàrregues d'arxius utilitzen l'API fetch nativa del navegador en lloc d'Axios, per què permet treballar amb respostes binaries (blob) de forma nativa.
- Modal portal: el component Modal utilitza ReactDOM.createPortal per renderitzar sempre sobre el DOM principal, evitant problemes de z-index.
- AuthContext: el context de React gestiona l'estat d'autenticació de forma global, exposant isAuthenticated, isAdmin, isWorker i l'objecte user a tots els components.

5.3 Proves

Les proves s'han realitzat de forma manual i incremental al llarg de tot el desenvolupament, seguint l'estratègia de provar cada funcionalitat en el moment de la seva implementació.

Proves de l'API REST

Tots els endpoints s'han provat amb Postman abans d'integrar-los al frontend. S'han verificat els casos d'exit, els casos d'error (usuari no trobat, email duplicat, període de nomina ja existent) i els casos límit (arxius grans, camps buits, formats incorrectes).

Proves d'integració frontend-backend

S'ha provat el cicle complet de cada funcionalitat: login, creació d'usuaris, assignació d'horaris, pujada de nòmines en PDF, sol·licitud de justificants amb arxius adjunts, aprovació/rebuig, descarga d'arxius i eliminació de registres.

Proves de l'aplicació Android

S'ha provat la lectura de targetes NFC reals amb el dispositiu físic. S'han verificat els casos de targeta vinculada (fitxatge correcte) i targeta no reconeguda (missatge d'error). S'ha comprovat el flux complet d'entrada i sortida del mateix treballador en dies consecutius.

Problemes detectats i resolts

- Error 'Unable to access lob stream': causat per la serialitzacio de camps @Lob fora de transacció. Resolt amb @Basic(fetch = LAZY), @Transactional als endpoints de lectura i ús de DTOs.
- Descàrregues en blanc: les URLs absolutes (http://localhost:8080) no funcionen des del navegador per CORS. Resolt usant URLs relatives (/api/...) i la API fetch amb createObjectURL.
- Contrasenyes en text pla: detectat en la fase de seguretat. Resolt activant BCrypt i migrant les contrasenyes existents amb un endpoint temporal.
- Conflicte de rutes Spring: la ruta DELETE /{id} interceptava GET /descargar/{id}. Resolt canviant el path d'eliminació a DELETE /eliminar/{id}.
- Error 404 en recarregar la SPA: resolt amb CustomErrorController que reenviva els 404 a index.html.
- Cuadrats blancs al dashboard: els textos blancs de les stat cards eren invisibles per conflicte amb Bootstrap. Resolt amb estilos inline i !important al CSS.

6. Conclusions

6.1 Conclusions generals

El projecte EntroYa ha assolit els seus objectius principals: s'ha desenvolupat un sistema complet i funcional de gestió de fitxatge laboral que integra tres tecnologies molt diferentes (backend Java, frontend React i app Android Kotlin) en una solucio coherent i desplegada en producció.

El resultat és un sistema que qualsevol empresa podria utilitzar de forma real: els treballadors fitxen de forma instantània amb la targeta NFC, els administradors gestionen tota la informació des d'un panell web accessible des de qualsevol dispositiu, i les dades es persisteixen de forma segura al cloud.

El projecte ha suposat un repte tècnic significatiu, especialment en la gestió d'arxius binaris a PostgreSQL, la sincronització de zones horàries entre l'app Android i el servidor, i el desplegament com a monorepository a Railway.

6.2 Assoliment dels objectius

- API REST amb Spring Boot: completament implementada amb tots els mòduls previstos.
- Aplicació Android NFC: funcional amb lectura NFC, logica d'entrada/sortida intelligent i cooldown anti-duplicats.
- Panell web React: implementat amb dos rols diferenciats i totes les funcionalitats previstes.
- Gestió de documents: implementada amb emmagatzematge LOB i descarga via blob.
- Seguretat BCrypt: implementada i migrades les contrasenyes existents.
- Desplegament cloud: completat a Railway, accessible des de qualsevol lloc.

L'únic objectiu que ha quedat parcialment pendent es la implementació de JWT per a l'autenticacio, que s'ha deixat com a millora futura per prioritzar la estabilitat del sistema en el temps disponible.

6.3 Valoració de la metodologia i la planificació

La metodologia iterativa i incremental ha funcionat molt bé per a aquest tipus de projecte. El fet de poder provar cada funcionalitat de forma independent ha permès detectar i corregir errors ràpidament, sense bloquejar el desenvolupament d'altres parts del sistema.

La divisió de tasques entre els dos membres de l'equip ha estat encertada: treballar en components independents (backend i frontend) ha permès avançar en parallel sense conflictes. Les integracions periòdiques han garantit la coherencia del conjunt.

La planificació inicial ha estat bastant ajustada a la realitat, tot i que algunes fases han requerit més temps del previst, especialment la gestió dels camps LOB de PostgreSQL i el redisseny visual del frontend.

6.4 Visió de futur

El sistema te diverses possibilitats d'ampliació que no han pogut implementar-se per limitacions de temps:

- JWT i autenticació stateless: implementar tokens JWT per a una autenticació més segura i escalable, eliminant la dependencia del localStorage.
- Notificacions push: alertar el treballador quan un justificant es aprovat o rebutjat.
- Càlcul automàtic de vacances i absències: generar informes mensuals automatics.
- Exportació de dades: permetre exportar historials de fitxatge en format CSV o Excel.
- Autenticació biometrica: complementar el fitxatge NFC amb reconeixement facial o empremta dactilar.

7. Glossari

Terme	Definicio
API REST	Interfície de programació d'aplicacions que segueix l'estil arquitectònic REST (Representational State Transfer), basada en peticions HTTP.
BCrypt	Algoritme de hash de contrasenyes dissenyat per ser computacionalment coster i resistent a atacs de força bruta.
DTO (Data Transfer Object)	Patró de disseny que consisteix a crear objectes simplificats per transferir dades entre capes, evitant exposar l'entitat completa.
JWT (JSON Web Token)	Estàndard obert per a la transmissió segura d'informació entre parts com un objecte JSON signat.
LOB (Large Object)	Tipus de dada de PostgreSQL per emmagatzemar objectes grans com fitxers binaris (PDF, imatges).
MVVM	Patró d'arquitectura Model-View-ViewModel utilitzat a l'app Android per separar la lògica de negoci de la interfície.
NFC (Near Field Communication)	Tecnologia de comunicació sense contacte d'abast curt (menys de 10 cm) utilitzada per a la identificació de targetes.
ORM (Object-Relational Mapping)	Tècnica que permet interactuar amb una base de dades relacional usant objectes del llenguatge de programació.
PostgREST	API REST automàtica generada sobre una base de dades PostgreSQL, utilitzada per l'app Android per comunicar-se amb Supabase.

Railway	Plataforma cloud PaaS (Platform as a Service) on s'ha desplegat el backend i el frontend del projecte.
React Context	Mecanisme de React per compartir estat global entre components sense necessitat de passar props manualment.
SPA (Single Page Application)	Aplicació web que carrega una sola pàgina HTML i actualitza el contingut dinàmicament sense recarregar la pàgina completa.
Spring Boot	Framework de Java que facilita la creació d'aplicacions web i APIs REST amb configuració mínima.
Supabase	Plataforma Backend-as-a-Service basada en PostgreSQL que proporciona base de dades, autenticació i APIs automàtiques.
UID (Unique Identifier)	Identificador únic de cada targeta NFC, llegit per l'app Android per identificar el treballador.
Vite	Eina de construcció ràpida per a projectes frontend moderns, utilitzada com a bundler i servidor de desenvolupament per al projecte React.

8. Bibliografia

Documentació oficial

- Spring Boot Reference Documentation. Pivotal Software. https://docs.spring.io/spring-boot/docs/current/reference/html/
- Spring Data JPA Documentation. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
- Spring Security Reference. https://docs.spring.io/spring-security/reference/
- React Documentation. Meta Platforms. https://react.dev/
- Vite Documentation. https://vitejs.dev/
- Android Developer Guides - NFC. Google. https://developer.android.com/guide/topics/connectivity/nfc
- Kotlin Documentation. JetBrains. https://kotlinlang.org/
- Supabase Documentation. https://supabase.com/docs
- Railway Documentation. https://docs.railway.app/

Recursos tecnics consultats

- Baeldung - Spring tutorials. https://www.baeldung.com/
- MDN Web Docs - Fetch API, Blob, createObjectURL. Mozilla. https://developer.mozilla.org/
- Bootstrap 5 Documentation. https://getbootstrap.com/docs/5.0/
- react-hot-toast Documentation. https://react-hot-toast.com/
- Stack Overflow - Resolució de problemes tècnics específics. https://stackoverflow.com/

Eines d'intel·ligència artificial

- Claude - Asistencia en depuració de codi, resolucio de problemes tecnics i generació de documentació. https://claude.ai/

9. Annexos

Annex A - Endpoints de l'API REST

Endpoint	Descripció
POST /api/auth/login	Autenticació d'usuari. Retorna id, nom, email i rol.
GET /api/admin/dashboard	Estadístiques generals: usuaris, fitxatges del dia, justificants pendents.
GET /api/admin/usuarios	Llistat de tots els usuaris.
POST /api/admin/usuarios	Crear un nou usuari.
PUT /api/admin/usuarios/{id}	Actualitzar un usuari existent.
DELETE /api/admin/usuarios/{id}	Eliminar un usuari.
GET /api/trabajador/historial/completo/{id}	Historial complet de fitxatges d'un treballador.

GET /api/trabajador/resumen/{id}	Resum setmanal: hores, dies treballats, promedi diari.
POST /api/nfc/fichaje	Registrar fitxatge per UID de targeta NFC.
POST /api/nfc/asignar	Assignar una targeta NFC a un usuari.
POST /api/justificantes/solicitar	Sol·licitar un justificant amb arxiu adjunt opcional.
GET /api/justificantes/usuario/{id}	Llistat de justificants d'un treballador.
GET /api/justificantes/admin/pendientes	Llistat de justificants pendents de revisió.
PUT /api/justificantes/admin/{id}	Aprovar o rebutjar un justificant.
GET /api/justificantes/descargar/{id}	Descarregar l'arxiu d'un justificant.
POST /api/nominas/subir	Pujar una nòmina en PDF per a un treballador.
GET /api/nominas/usuario/{id}	Llistat de nòmines d'un treballador.
GET /api/nominas/descargar/{id}	Descarregar el PDF d'una nòmina.
DELETE /api/nominas/eliminar/{id}	Eliminar una nòmina.
POST /api/horarios/asignar	Assignar un horari a un treballador.
GET /api/horarios/usuario/{id}	Llistat d'horaris d'un treballador.
PUT /api/horarios/{id}/desactivar	Desactivar un horari.

Annex B - Tecnologies i versions

Tecnologia	Versió i us
Java	21 - Llenguatge del backend
Spring Boot	3.x - Framework backend
Spring Security	6.x - Seguretat i BCrypt
Spring Data JPA	3.x - Persistencia de dades
PostgreSQL	15 - Base de dades relacional
Supabase	Cloud - Hosting de la base de dades
Railway	Cloud - Desplegament del sistema complet
Kotlin	1.9 - Llenguatge de l'app Android
Android Studio	Hedgehog - IDE per a l'app Android
React	18 - Framework frontend
Vite	5.x - Bundler i servidor de desenvolupament
Axios	1.x - Client HTTP per al frontend
react-hot-toast	2.x - Notificacions
Bootstrap	5.x - Framework CSS base
DM Sans	Variable - Tipografia principal

Annex C - Estructura de desplegament a Railway

El desplegament s'ha realitzat com a monorepository a Railway seguint aquest flux:

- El frontend es compila amb npm run build, generant la carpeta dist/.
- Els fitxers de dist/ es copien a src/main/resources/static/ del backend.
- Spring Boot serveix els fitxers estàtics directament, incloent la SPA React i la landing page.
- Un WebConfig assegura que totes les rutes desconegudes es redirigeixen a index.html.
- Un CustomErrorController reenvia els errors 404 al frontend per evitar la pantalla d'error de Whitelabel.
- Les variables d'entorn (DB_URL, DB_USER, DB_PASSWORD) es configuren a Railway sense exposar-les al codi.

Nota sobre l'ús de la intel·ligència artificial

Durant el desenvolupament d'aquest projecte s'han utilitzat eines d'intelligencia artificial, concretament Claude (Anthropic), com a assistent tècnic en les següents situacions:

- Resolució de problemes tècnics específics: errors de compilació, problemes de serialització JSON amb camps LOB de PostgreSQL, conflictes de rutes a Spring Boot i configuració del desplegament a Railway.
- Depuració de codi: identificació de causes d'errors i proposta de solucions, especialment en la gestió de transaccions Hibernate i el sistema de descàrrega de fitxers binaris.
- Generació de codi boilerplate: components React amb estils inline, configuració de Spring Security, estructura de DTOs i missatgeria de notificacions.
- Assistència en la redacció d'aquesta memòria: estructuració del contingut, traducció de termes tècnics al català i revisió de la coherència del document.

En tots els casos, el codi generat ha estat revisat, comprovat i adaptat pels autors del projecte abans d'incorporar-lo. La comprensió de totes les solucions implementades ha estat verificada per ambdós membres de l'equip, que son responsables del contingut final del projecte.

L'ús de la IA s'ha fet de forma conscient i responsable, com una eina d'assistència equivalent a consultar documentació o recursos en línia, sense substituir el procés d'aprenentatge i comprensió propi del projecte.

Llicència