INSTITUT PUIG CASTELLAR

FCTNow

Proyecto de desarrollo

Ciclo Formativo de Grado Superior (CFGS)
Desarrollo de Aplicaciones Web

Autores:

◇ David Barbosa Olayo
◇ Marc Mancilla Pontejo

Grupo: DAW2
Curso académico: 2024-2025
Centro: IES Puig Castellar

Versión del proyecto: v1.0.0 (Entrega Final)

Índice

Resumen del proyecto

FCTNow es una plataforma web concebida para centralizar el ciclo completo de la Formación en Centros de Trabajo (FCT) de un centro de Formación Profesional, sustituyendo el habitual mosaico de hojas de cálculo, correos electrónicos y carpetas compartidas por un entorno único, accesible y trazable para todos los actores implicados: alumnado, empresas colaboradoras, tutores de centro y coordinadores de FCT. La aplicación se ha desarrollado como un proyecto integrador de las competencias del ciclo, con un alcance funcional ambicioso y un stack tecnológico alineado con los entornos profesionales actuales.

En el lado del servidor, un backend en Java 21 con Spring Boot 3.5 expone una API REST completa que gestiona usuarios y roles, empresas y ofertas internas de FCT, solicitudes y asignaciones, mensajería interna, notificaciones, alta masiva de alumnos por parte de los tutores y consumo de ofertas reales del mercado a través de la API pública de Adzuna. La capa de seguridad se construye sobre Spring Security y JWT (HS256), con autorización por roles diferenciados (`ALUMNO`, `EMPRESA`, `TUTOR_CENTRO`, `COORDINADOR`, `ADMIN`) y contraseñas hasheadas con BCrypt. La persistencia se apoya en PostgreSQL y Spring Data JPA, con un esquema versionado a través de Flyway que evoluciona del bootstrap inicial hasta el seguimiento de horas y retribución de las asignaciones.

En el lado del cliente, una aplicación Angular 20 con renderizado del lado del servidor (SSR) proporciona una experiencia responsive, con paneles diferenciados por rol y vistas específicas para el flujo de FCT: catálogo de ofertas con filtros y sincronización con ofertas externas de Adzuna, gestión de candidaturas para alumnos, panel de empresa con creación y mantenimiento de ofertas, panel de tutor para alta y seguimiento del alumnado, panel de coordinador para la asignación final, mensajería interna entre los participantes y notificaciones in-app. El frontend implementa además una capa de caché *stale-while-revalidate* para reducir la latencia percibida y guards de ruta por rol para proteger las áreas restringidas.

El despliegue de producción opera sobre una arquitectura cloud heterogénea pero acoplada por configuración: el backend se publica en Render como contenedor Docker (Java 21 + Spring Boot empaquetado en JAR), el frontend en Vercel como aplicación Angular estática con *rewrite* hacia la API, y la base de datos PostgreSQL se mantiene gestionada en Supabase. El correo transaccional para notificaciones de alta y restablecimiento se entrega a través de SMTP de Gmail mediante Spring Mail.

Más allá de la digitalización de un proceso administrativo, FCTNow aspira a aportar transparencia, trazabilidad y datos objetivos a una tarea que tradicionalmente queda dispersa entre múltiples herramientas y memorias informales del profesorado, ofreciendo una base sólida sobre la que un centro educativo puede operar de forma sostenible curso tras curso.

Palabras clave

- Formación en Centros de Trabajo
- Gestión integral de FCT
- Aplicación web full-stack
- Spring Boot 3
- Angular 20
- REST API
- JWT (HS256)
- Spring Security
- BCrypt
- PostgreSQL
- Flyway
- Adzuna API
- Render
- Vercel
- Supabase
- Asignación de alumnos
- Mensajería interna
- Notificaciones in-app
- Digitalización educativa

Abstract

FCTNow is a web platform designed to centralise the full lifecycle of in-company vocational training (FCT) at a Spanish Vocational Education and Training (VET) school, replacing the usual patchwork of spreadsheets, emails and shared folders with a single, accessible and traceable environment for every stakeholder involved: students, partner companies, school tutors and FCT coordinators. The application has been built as a capstone project for the higher-level VET cycle in Web Application Development, with an ambitious functional scope and a technology stack aligned with current professional environments.

On the server side, a Java 21 backend built with Spring Boot 3.5 exposes a complete REST API that manages users and roles, companies and internal FCT offers, applications and assignments, internal messaging, notifications, bulk student enrolment by tutors, and consumption of real labour-market offers through the public Adzuna API. The security layer is built on Spring Security and JWT (HS256), with role-based authorisation (`ALUMNO`, `EMPRESA`, `TUTOR_CENTRO`, `COORDINADOR`, `ADMIN`) and BCrypt password hashing. Persistence relies on PostgreSQL with Spring Data JPA, governed by a Flyway-versioned schema that evolves from the initial bootstrap up to assignment hour tracking and payment terms.

On the client side, an Angular 20 application with Server-Side Rendering (SSR) delivers a responsive experience with role-specific dashboards and views tailored to the FCT workflow: offer catalogue with filters and Adzuna-backed external listings, application management for students, company panel for offer maintenance, tutor panel for student onboarding and follow-up, coordinator panel for the final assignment, internal messaging between actors and in-app notifications. The frontend additionally implements a *stale-while-revalidate* caching layer to reduce perceived latency, and route guards that protect restricted areas based on the authenticated user role.

The production deployment runs on a heterogeneous cloud architecture coupled by configuration: the backend is published on Render as a Docker container (Java 21 + Spring Boot fat JAR), the frontend is hosted on Vercel as a static Angular application with API *rewrites*, and the PostgreSQL database is managed by Supabase. Transactional email for account bootstrapping and password recovery is delivered through Gmail SMTP via Spring Mail.

Beyond digitising an administrative process, FCTNow aims to bring transparency, traceability and objective data to a task traditionally scattered across multiple tools and the informal memory of teaching staff, providing a solid foundation on which an educational centre can operate sustainably year after year.

Keywords

- Vocational Training Internships
- End-to-end Internship Management

- Full-stack Web Application
- Spring Boot 3
- Angular 20
- REST API
- JWT (HS256)
- Spring Security
- BCrypt
- PostgreSQL
- Flyway
- Adzuna API
- Render
- Vercel
- Supabase
- Student Assignment Workflow
- In-app Messaging
- In-app Notifications
- Educational Process Digitalisation

1. Introducción

La Formación en Centros de Trabajo (FCT) es uno de los módulos más relevantes y, paradójicamente, peor instrumentados de toda la Formación Profesional española. Es el momento en el que el alumnado deja por primera vez el aula y se incorpora a una empresa real para aplicar, durante varias semanas, las competencias adquiridas a lo largo del ciclo. De su correcta gestión depende no solo la titulación del estudiante, sino también su primer contacto serio con el mundo laboral, la calidad percibida del centro educativo y la relación a largo plazo con el tejido productivo del entorno.

A pesar de su importancia, en la práctica la coordinación de la FCT sigue apoyándose mayoritariamente en hojas de cálculo compartidas, cadenas de correos electrónicos, formularios en papel y la memoria informal del profesorado. El resultado es un proceso frágil, intensivo en tiempo administrativo, difícil de auditar y poco transparente para el propio alumnado, que en muchos casos descubre las ofertas disponibles, su asignación o el estado de su candidatura a través de canales informales y desfasados.

FCTNow nace para revertir esta situación construyendo, desde cero, una plataforma web específicamente diseñada para el flujo de trabajo real de un centro de FP. No se trata de un gestor genérico adaptado a posteriori, sino de un sistema en el que cada pantalla, cada rol y cada endpoint responden a una etapa concreta del ciclo de la FCT: alta de empresas colaboradoras, publicación de ofertas, candidaturas del alumnado, validación por parte del centro, asignación oficial, seguimiento durante la estancia y cierre con evaluación final. El proyecto se ha desarrollado como TFG del ciclo de DAW del IES Puig Castellar, combinando un backend en Spring Boot, un frontend en Angular, una base de datos PostgreSQL y un despliegue real en la nube que ha permitido validar la solución de extremo a extremo.

Esta memoria documenta el recorrido completo del proyecto: el contexto y la justificación de la propuesta, los objetivos planteados al inicio, la planificación adoptada, los requisitos funcionales y no funcionales identificados, las tecnologías valoradas y finalmente escogidas, la arquitectura técnica resultante, las pruebas realizadas y las conclusiones extraídas a posteriori. Más allá de su valor académico, FCTNow se presenta como una propuesta operativa: un sistema completo, desplegado, accesible vía web y preparado para evolucionar al ritmo que un centro real pueda demandar.

1.1 Contexto

La Formación en Centros de Trabajo es un módulo obligatorio en la mayoría de ciclos formativos de FP de grado medio y superior, incluido el CFGS de Desarrollo de Aplicaciones Web. La normativa establece, en líneas generales, los siguientes elementos:

- Un número mínimo de horas que el alumnado debe completar en una empresa o entidad colaboradora.
- La firma de un convenio de colaboración entre el centro educativo y la empresa.
- La existencia de dos figuras tutoras: un **tutor de centro** que acompaña al alumno desde el instituto y un **tutor de empresa** que ejerce de referente durante la estancia.
- Un proceso de evaluación final con doble visión, centro y empresa, que determina la superación o no del módulo.

En el día a día de la mayoría de centros, ese marco normativo se materializa a través de un conjunto fragmentado de herramientas y rutinas no específicas:

- **Hojas de cálculo** para listar alumnado por ciclo y grupo, empresas colaboradoras, plazas ofertadas y asignaciones realizadas.
- **Formularios en papel** para convenios, anexos, partes de horas y registros de seguimiento, que después se digitalizan parcialmente al escanearse y archivarse en carpetas compartidas.
- **Correo electrónico y llamadas telefónicas** para coordinar fechas, horarios, incidencias y la comunicación con el tutor de empresa.
- **Carpetas físicas o unidades compartidas** que acumulan, a lo largo del curso, documentación de calidad variable y difícil trazabilidad.

Este enfoque presenta varios problemas estructurales:

- **Falta de centralización.** La información de un mismo alumno se reparte entre la hoja de cálculo del coordinador, el correo del tutor de centro, los partes que rellena la empresa y el sistema académico oficial. Conciliar todas esas fuentes requiere tiempo y aumenta el riesgo de error.
- **Falta de trazabilidad.** No queda registro estructurado de quién aceptó o rechazó una candidatura, cuándo se asignó al alumno o qué incidencias surgieron. El histórico depende de la memoria de las personas implicadas y desaparece al final del curso o cuando rota el profesorado.
- **Falta de visibilidad para el alumnado.** Los estudiantes a menudo desconocen qué empresas colaboran habitualmente con el centro, qué tareas reales se realizan en cada oferta o qué feedback dejaron antiguos alumnos. La información de calidad se queda en la cabeza del profesorado.
- **Carga administrativa creciente.** Cada nuevo ciclo, grupo o convenio amplía el número de cruces de datos y de comunicaciones a coordinar. Sin una herramienta específica, el coordinador acaba dedicando una parte desproporcionada de su jornada a tareas administrativas de bajo valor pedagógico.

Frente a este panorama, una herramienta diseñada específicamente para la gestión de FCT permite normalizar el proceso de alta de empresas y ofertas, registrar de forma estructurada las preferencias y el perfil del alumnado, mantener un seguimiento periódico y consultable durante la estancia y generar listados e informes con menos errores y menos tiempo invertido. FCTNow se inserta en este contexto como una solución pensada para el flujo real de un centro de FP, con roles, pantallas y procesos diseñados a medida del problema y no como una adaptación forzada de un CRM genérico.

1.2 Justificación

La justificación del proyecto se sostiene sobre cuatro ejes complementarios, que combinan necesidad real, oportunidad formativa, encaje tecnológico y viabilidad de despliegue.

1. Existe un problema real, recurrente y compartido por la práctica totalidad de los centros de FP. La gestión de la FCT no es un proceso opcional ni puntual: cada curso, todos los ciclos del centro pasan por el mismo flujo, con plazos compartidos y dependencias entre actores. Un error en la asignación, un incumplimiento de horarios o un problema con la documentación puede suponer que un alumno no titule en convocatoria ordinaria o que una empresa colaboradora se retire en el curso siguiente. Hoy, gran parte de esa criticidad se gestiona con herramientas genéricas que no han sido pensadas para este caso de uso.

2. La calidad de las prácticas no se puede mejorar sin estructurar los datos. Identificar "buenas empresas", detectar plazas que no cumplen los estándares esperados o ajustar la oferta a las expectativas del alumnado solo es posible si esa información existe en algún sitio más fiable que la memoria del equipo docente. Una plataforma que agregue ofertas, candidaturas, asignaciones, seguimientos y evaluaciones a lo largo de varios cursos abre la puerta a tomar decisiones informadas: con qué empresas reforzar la relación, qué ofertas tienen tasas de aceptación más bajas, qué perfiles encajan mejor con qué tipo de empresa.

3. El proyecto encaja de forma natural con las competencias del ciclo de DAW. El desarrollo de FCTNow ha exigido aplicar prácticamente todas las áreas troncales del CFGS de Desarrollo de Aplicaciones Web: diseño y modelado de bases de datos relacionales, desarrollo de servicios REST en un framework empresarial, autenticación y autorización basadas en tokens, desarrollo de aplicaciones de página única en un framework frontend moderno, pruebas unitarias y de integración, despliegue en la nube e integración con APIs externas. Frente a un proyecto pequeño y autocontenido, FCTNow ha permitido al equipo enfrentarse a problemas reales de escala intermedia (versionado de esquema, gestión de roles, integración con un proveedor externo inestable, despliegue heterogéneo) que difícilmente aparecen en un ejercicio académico tradicional.

4. La solución es desplegable y mantenible con costes prácticamente nulos. Una de las decisiones de diseño ha sido apoyarse exclusivamente en tecnologías de código abierto o con planes gratuitos suficientes para un piloto: Spring Boot, Angular, PostgreSQL, Render, Vercel y Supabase. Esta combinación permite que un centro educativo pueda evaluar la herramienta sin compromiso económico, y abre la puerta a un mantenimiento evolutivo razonable a medio plazo sin depender de licencias propietarias.

FCTNow pretende, por tanto, responder a una necesidad real del contexto educativo mientras sirve como proyecto integrador de las competencias del ciclo. No se trata de construir un "gestor genérico" más, sino de demostrar que un equipo de alumnos de DAW puede afrontar el desarrollo completo de una solución específica, alineada con un flujo de trabajo concreto, y entregarla operativa.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar, desarrollar, desplegar y validar una aplicación web *full-stack* para la gestión integral de la Formación en Centros de Trabajo, que centralice la información de alumnos, empresas, ofertas, candidaturas, asignaciones y seguimientos, soporte roles diferenciados para todos los actores implicados, integre fuentes externas de ofertas reales y opere en un entorno de despliegue cloud accesible vía navegador.

1.3.2 Objetivos específicos

- **Modelado completo del dominio FCT**
  - Diseñar un esquema relacional que represente alumnos, empresas, ofertas, solicitudes, asignaciones, seguimientos, conversaciones, mensajes y notificaciones.
  - Versionar el esquema con migraciones reversibles aplicadas automáticamente al arranque.
  - Incluir desde el primer momento atributos pensados para el seguimiento real (horas totales, fechas, retribución, observaciones).
- **Autenticación y autorización robustas**
  - Implementar autenticación con tokens JWT firmados con HS256 y vida configurable.
  - Definir cinco roles funcionales (`ALUMNO`, `EMPRESA`, `TUTOR_CENTRO`, `COORDINADOR`, `ADMIN`) con permisos diferenciados.
  - Hashear contraseñas con BCrypt y aplicar autorización tanto a nivel de endpoint como a nivel de servicio para impedir accesos cruzados entre roles.
- **Catálogo de empresas y ofertas internas**
  - Permitir el alta y mantenimiento de empresas colaboradoras con sus datos fiscales, de contacto y de ubicación.
  - Permitir a las empresas registrar y mantener sus propias ofertas de FCT, con familias profesionales, ciclos formativos, modalidad, fechas, plazas, requisitos y descripción de tareas.
  - Exponer un catálogo público filtrable por familia profesional, localidad, modalidad y palabra clave.
- **Integración con fuentes externas de ofertas**
  - Consumir la API pública de Adzuna para enriquecer el catálogo con ofertas reales del mercado.
  - Mapear las ofertas externas a un DTO interno coherente y aislar al frontend de la API ajena.
  - Soportar la falta de credenciales sin romper el catálogo interno (fallback degradado a `503` controlado).
- **Flujo completo de candidaturas y asignación**
  - Permitir que el alumnado se postule a ofertas internas y registre manualmente solicitudes externas.

  - Proporcionar a tutor y coordinador un panel donde validar, rechazar o reasignar candidaturas.
  - Generar asignaciones formales a partir de candidaturas aceptadas, con horas totales, fecha de inicio, retribución y observaciones.
- **Gestión del alumnado por parte del tutor**
  - Permitir al tutor crear alumnos individualmente o en bloque.
  - Generar credenciales temporales y enviar la bienvenida por correo electrónico al alumno.
  - Mantener actualizado el perfil del alumno, sus preferencias y su CV.
- **Comunicación interna entre actores**
  - Implementar mensajería interna entre alumnado, empresas y personal de centro.
  - Proporcionar notificaciones in-app para los eventos relevantes del flujo (candidatura aceptada, asignación creada, recomendación de oferta, etc.).
  - Soportar conversaciones uno a uno con histórico persistente.
- **Frontend moderno, responsive y consciente del rol**
  - Construir una SPA Angular con SSR para mejorar el tiempo a primer pintado.
  - Implementar guards de ruta por rol que impidan el acceso a áreas no autorizadas.
  - Aplicar caché *stale-while-revalidate* para reducir la latencia percibida en pantallas frecuentes.
- **Despliegue en producción**
  - Empaquetar el backend como contenedor Docker y desplegarlo en Render.
  - Desplegar el frontend Angular en Vercel con *rewrite* hacia la API.
  - Mantener la base de datos PostgreSQL en Supabase, gestionada y con copias de seguridad automáticas.
- **Calidad del software**
  - Implementar pruebas unitarias y de controlador con JUnit, Mockito y `MockMvc` para los endpoints críticos.
  - Implementar pruebas de componentes y servicios en el frontend con Jasmine y Karma.
  - Documentar la API con OpenAPI/Swagger y mantener un `README` operativo en el repositorio.

1.4 Estrategia y planificación del proyecto

El proyecto se ha abordado como un desarrollo *end-to-end* construido desde cero, sin partir de plantillas, *boilerplate* genéricos ni soluciones SaaS preexistentes que se hubieran limitado a ser configuradas. La estrategia elegida ha sido **iterativa, vertical y centrada en entregar incrementos utilizables**: cada fase debía cerrar un trozo del flujo de FCT que el resto del equipo pudiera ya tocar (aunque fuera mínimamente) en el frontend, en lugar de construir todo el backend primero y dejar el frontend para el final.

Esta decisión, lejos de ser anecdótica, ha condicionado tres aspectos importantes del proyecto:

- **Control sobre el modelo de datos.** Al modelar el dominio en pequeños bloques (autenticación → catálogo de ofertas → solicitudes → asignaciones → mensajería → notificaciones), cada migración Flyway ha podido validarse contra el código real que la utilizaba, evitando rediseños masivos posteriores. Las quince migraciones del esquema (`V1` a `V15`) reflejan ese crecimiento orgánico.
- **Independencia de soluciones "mágicas".** No se ha empleado ningún framework de tipo *low-code* ni se ha integrado un proveedor de identidad externo: la autenticación, la emisión de JWT, el control de roles y la gestión del ciclo de vida del usuario se han implementado de forma explícita con Spring Security, lo que ha exigido entender de verdad cada pieza del flujo.
- **Adaptación precisa al flujo real.** Todos los endpoints, pantallas y reglas de negocio se han ajustado al proceso concreto de la FCT en un centro como el IES Puig Castellar. Por ejemplo, la generación automática de un correo `nombreApellido@elpuig.xeill.net` para los alumnos creados por el tutor (migración `V13`) responde a una convención real del centro, no a un esquema genérico.

1.4.1 Fases del desarrollo

El proyecto se ha estructurado en siete fases principales. Las primeras se centraron en levantar la base técnica (backend, base de datos, autenticación, frontend mínimo), las intermedias en construir los flujos funcionales (ofertas, candidaturas, asignaciones, mensajería) y las últimas en pulir la experiencia y desplegar la solución.

Fase 0: Preparación del entorno (1 semana)

- Creación del monorepo con dos subdirectorios `backend/` y `frontend/`, junto con un `docker-compose.yml` para PostgreSQL local.
- Bootstrap del backend con Spring Initializr (Maven, Java 21, Spring Boot 3.5, *starters* Web, Data JPA, Security, Validation, Mail, OAuth2 Resource Server).
- Bootstrap del frontend con Angular CLI (Angular 20, TypeScript 5.9, RxJS, SSR).
- Convenciones de commits (Conventional Commits), ramas (`main` + `issue/*` o `issue-*`) y *pull requests* obligatorios para integrar cambios.
- Primer arranque de PostgreSQL en Docker y verificación de la conexión JDBC desde Spring Boot.

Fase 1: Núcleo técnico — autenticación, roles y base de Flyway (2 semanas)

- Migración `V1` con la tabla técnica `backend_metadata` para validar el pipeline de migraciones sin introducir todavía dominio.
- Migración `V2` con las tablas `app_users` y `user_roles`, incluyendo restricciones y un índice por rol.
- Configuración de Spring Security en modo *resource server* OAuth2 con JWT HS256.
- Implementación de `AuthService` y `JwtTokenService`: login por email/contraseña, emisión de token con los roles del usuario, *user_id*, *display_name* y, cuando aplica, el `centro_email`.
- Endpoints `POST /api/auth/login`, `GET /api/auth/me`, `GET /api/health`.
- Documentación OpenAPI/Swagger UI publicada en `/api/openapi` y `/api/swagger-ui.html`.

Fase 2: Catálogo de empresas y ofertas internas (2 semanas)

- Migración `V3` con las tablas `empresas` y `ofertas_fct` y sus índices por estado, familia profesional, localidad y modalidad.
- Migración `V5` para enlazar `app_users` con `empresas` mediante una clave foránea opcional.
- Servicios `EmpresaService`, `EmpresaOfertaService` y `OfertaFctService` con reglas de negocio (filtros, validaciones, estados).
- Endpoints REST de empresa y ofertas: alta, edición, baja lógica, listado público filtrable, listado privado por empresa.
- Pantallas Angular del rol empresa: panel general, listado de ofertas propias, formulario de creación/edición.
- Pantalla pública de **prácticas** con filtros por familia profesional, localidad y modalidad.

Fase 3: Alumnado, candidaturas y preferencias (2 semanas)

- Migración `V4` con la tabla `solicitudes_fct`, restricción única por alumno y oferta e índices por alumno, oferta y estado.
- Migración `V6` con la tabla `alumno_preferencias` para datos de orientación y subida de CV (incluido el binario como `BYTEA`).
- Servicio `AlumnoPreferenciasService` y endpoints de perfil del alumno.
- Pantallas Angular del rol alumno: panel, perfil con preferencias, subida de CV y bandeja de “mis solicitudes”.
- Pantallas Angular del rol empresa: bandeja de solicitudes recibidas, validación o rechazo.

Fase 4: Asignaciones y ofertas externas con Adzuna (2 semanas)

- Migración `V7` con la tabla `asignaciones_fct`, vinculada a la candidatura aceptada y con `uk_asignaciones_fct_solicitud` para impedir asignaciones duplicadas.

- Implementación del cliente Adzuna (`AdzunaClient`, `AdzunaService`, `AdzunaProperties`) con timeout configurable y manejo controlado de errores (`AdzunaUnavailableException`).
- Migración `V8` con las tablas `solicitudes_externas` y `asignaciones_fct_externas` para registrar el ciclo de vida de candidaturas que no se originan en el catálogo interno.
- Pantalla Angular de detalle de oferta externa, panel de coordinador y panel de tutor con sus candidaturas asignables.
- Endpoint `POST /api/asignaciones` y `POST /api/asignaciones/externas` para formalizar la asignación final.

Fase 5: Mensajería, notificaciones y onboarding por el tutor (3 semanas)

- Migración `V9` con la tabla `notificaciones` (campos para ofertas internas y externas, `leida`, `action_url`, `action_label`).
- Migración `V11` con `conversaciones` y `mensajes`, con índices optimizados para el listado por participante y la lectura cronológica de una conversación concreta.
- Migración `V12` para generalizar el destinatario de las notificaciones (`destinatario_id`), permitiendo notificar también a roles que no sean alumnado.
- Migración `V13` para añadir `centro_email` al usuario y rellenarlo automáticamente para los alumnos ya existentes a partir de su nombre.
- Migración `V14` para almacenar la foto de perfil de cualquier usuario; `V10` ya había hecho lo equivalente para el alumno.
- Implementación de `TutorAlumnoController` y servicios asociados para crear alumnos individualmente o por importación de fichero, generando contraseña temporal y notificando por correo (`AlumnoWelcomeMailer`).
- Pantallas Angular de mensajería, notificaciones, panel de tutor con onboarding del alumnado y panel de coordinador.

Fase 6: Seguimiento, retribución, pulido y caché en cliente (2 semanas)

- Migración `V15` con horas totales, horas diarias estimadas, fecha de inicio y datos de retribución sobre las dos tablas de asignaciones (internas y externas), con restricciones `CHECK` para mantener la coherencia (rango de horas y dependencia entre `remunerada` e `importe_mensual`).
- Pantallas Angular de detalle de asignación con barra de progreso de horas.
- Servicios de caché en cliente (`PracticasCacheService`, `AsignacionesCacheService`, `TutorCacheService`) con estrategia *stale-while-revalidate* para que el usuario vea inmediatamente los datos de la última visita mientras se refrescan en segundo plano.
- Pulido visual y de copys en todas las pantallas, refinamiento de cards, estados vacíos, gestión consistente de errores HTTP en la interfaz.

Fase 7: Despliegue en producción y bootstrap del tutor (1 semana)

- Construcción del `Dockerfile` multi-stage para el backend (Maven + Eclipse Temurin 21 JRE) y despliegue en Render como servicio web sobre el puerto provisto por la plataforma.

- Configuración del frontend en Vercel: `outputDirectory` apuntando a `dist/FCTManager/browser`, *rewrites* hacia `https://fctnow-backend.onrender.com/api/...` para que el navegador pueda mantener URLs relativas `/api/...`.
- Migración del esquema desde el contenedor local de desarrollo hacia PostgreSQL gestionado en Supabase, mediante Flyway al arrancar el backend en Render.
- Creación del perfil `application-prod.yml` y de un seed Flyway *repeatable* (`R__bootstrap_tutor.sql`) que crea o actualiza un tutor inicial leyendo variables de entorno (`FCTNOW_BOOTSTRAP_TUTOR_EMAIL`, `FCTNOW_BOOTSTRAP_TUTOR_PASSWORD_HASH`, `FCTNOW_BOOTSTRAP_TUTOR_DISPLAY_NAME`), evitando hardcodear credenciales en el repositorio.
- Verificación funcional de extremo a extremo: login real en producción, alta de alumnos, búsqueda de ofertas externas en Adzuna, creación de candidatura y asignación, mensajería entre roles.

Esta hoja de ruta ha permitido avanzar de forma iterativa y controlada, dejando al final de cada fase un sistema utilizable y un esquema de base de datos coherente con el código que lo usa, sin las migraciones “grandes” típicas que aparecen cuando se planifica todo el modelo de antemano.

1.5 Metodología de trabajo

Se ha adoptado una metodología ágil inspirada en Scrum, adaptada a la escala de un TFG y al tamaño del equipo. Los principios operativos han sido:

- **Iteraciones cortas centradas en una *issue* de GitHub.** Cada rama nueva (`issue/<n>-<slug>` o `issue-<n>-<slug>`) responde a una *issue* concreta, con criterios de aceptación claros, y se cierra mediante un *pull request* a `main`. El histórico de PRs (más de 80 fusionados) es a la práctica el log detallado del proyecto.
- **`main` siempre estable y desplegable.** La rama principal se ha protegido para que solo entre código revisado en PR. El despliegue de producción (Render y Vercel) se dispara automáticamente con cada *merge* a `main`, así que cualquier regresión llega muy rápido al entorno público y obliga a corregirla.
- **Conventional Commits.** Todos los commits siguen el patrón `tipo(ámbito): descripción` (`feat`, `fix`, `build`, `refactor`, ...). Esto facilita la lectura del historial y permite un cambio futuro a *changelogs* automáticos sin tener que reescribir nada.
- **Validación local antes de PR.** El `README` raíz documenta el comando estándar que se ejecuta antes de abrir cualquier *pull request*: `npm run build` y `npm test` en frontend, `mvn test` en backend. Esto detecta errores tipográficos, regresiones y problemas de compilación antes de que el revisor pierda tiempo con ellos.
- **Revisión cruzada.** Aunque el equipo es pequeño, los PRs se revisan siempre por una segunda persona antes del *merge*. En los hitos críticos (cambios en seguridad, migraciones, integración Adzuna) se han realizado revisiones más exhaustivas, con pruebas manuales explícitas en local antes del despliegue.

Las herramientas que han soportado este flujo son:

- **GitHub** para el repositorio, las *issues*, los *pull requests* y la revisión de código. También para almacenar la documentación de soporte (`README.md` raíz y por subproyecto).
- **GitHub Actions / CI básico** se ha mantenido a nivel de validación local (los tests `mvn test` y `npm test` se ejecutan antes del PR), dejando la integración continua automática como una mejora prevista de cara a un mantenimiento del proyecto a largo plazo.
- **IntelliJ IDEA** y **Visual Studio Code** como IDEs principales para backend y frontend respectivamente.
- **DBeaver / cliente SQL de Supabase** para inspeccionar el estado real de la base de datos durante el desarrollo y resolver incidencias puntuales.
- **Postman** para probar endpoints en local antes de integrarlos en el frontend.
- **Swagger UI** del propio backend (`/api/swagger-ui.html`) para revisar la superficie pública de la API de forma autoexplorable.

El resultado de esta forma de trabajar es un proyecto cuyo histórico es legible, cuyo estado en `main` es siempre desplegable y cuyo desarrollo ha avanzado en incrementos pequeños y verificables, en lugar de en grandes saltos que habrían sido difíciles de validar.

1.6 Estudio económico y presupuestario

El desarrollo de FCTNow se ha realizado exclusivamente con herramientas y servicios de código abierto o con planes gratuitos suficientes para un piloto académico. Esta decisión ha permitido minimizar los costes directos sin renunciar a un *stack* tecnológico profesional ni a una experiencia real de despliegue en producción.

Costes directos (bajos o nulos)

- **Mano de obra.** El recurso de mayor coste real ha sido el tiempo dedicado por el equipo al análisis, diseño, desarrollo, pruebas y documentación del proyecto. Considerando una jornada estimada de unas 600-700 horas repartidas entre los dos autores a lo largo del curso, el coste de mercado equivalente para perfiles júnior de desarrollo *full-stack* se situaría entre 12.000 € y 18.000 €, dependiendo de tarifas. Al tratarse de un proyecto académico, este coste no se ha facturado.
- **Infraestructura de desarrollo.**
  - IntelliJ IDEA Ultimate (plan educativo gratuito mientras se mantiene la matrícula) y Visual Studio Code.
  - GitHub Free para repositorios públicos y privados.
  - Docker Desktop para levantar PostgreSQL en local.
  - Maven y npm para gestionar dependencias.
- **Infraestructura de producción.**
  - **Render** (plan gratuito) para el backend Spring Boot empaquetado en contenedor Docker.
  - **Vercel** (plan Hobby) para el frontend Angular estático.
  - **Supabase** (plan Free) para PostgreSQL gestionado con copias de seguridad automáticas.
  - **Adzuna API** con plan gratuito limitado a *requests* por mes y por aplicación, suficiente para un piloto.
  - **Gmail SMTP** para el correo transaccional, con cuenta dedicada de proyecto.

Costes a medio y largo plazo

- **Mantenimiento evolutivo.** Las dependencias usadas (Spring Boot, Angular, PostgreSQL) son productos con ciclos LTS bien definidos. El coste real de mantenimiento se reduce a actualizaciones periódicas, sin licencias propietarias asociadas.
- **Escalado de los planes gratuitos.** En un eventual uso real, el plan gratuito de Render fuerza a un *cold start* tras un periodo de inactividad y limita la potencia del contenedor. Pasar a un plan de pago en Render (~7 €/mes) elimina ese inconveniente. Vercel mantiene precios bajos para tráfico moderado, y Supabase escala a planes de pago a partir de los 25 €/mes si el volumen de base de datos supera el plan gratuito.
- **Implantación en un centro real.** El coste relevante en una hipotética implantación recaería en el tiempo de adaptación a la operativa concreta del centro (familias profesionales, ciclos, calendarios), la formación del profesorado y la posible contratación de un plan de pago para garantizar disponibilidad sin *cold starts*.

Como referencia, una implantación piloto en un solo ciclo de DAW podría hacerse cómodamente con menos de 50 €/mes de infraestructura.

Coste-beneficio

Frente a una solución comercial de gestión académica de varios miles de euros anuales, FCTNow ofrece una respuesta específica al problema de la FCT con un coste de explotación marginal y, sobre todo, con código fuente disponible para adaptarse al centro. El retorno principal no es económico directo, sino una reducción significativa del tiempo administrativo dedicado a la FCT por parte de coordinación y tutoría, una mejora medible de la transparencia para el alumnado y la posibilidad de tomar decisiones informadas curso tras curso con datos estructurados.

2. Descripción del proyecto

Este apartado profundiza en cómo se materializa FCTNow desde el punto de vista técnico: los requisitos identificados, la comparativa de tecnologías valoradas frente a las finalmente escogidas, la arquitectura general del sistema, los componentes que la conforman y las funcionalidades clave expuestas a los usuarios. El objetivo es ofrecer una visión coherente de la solución que permita comprender no solo qué hace la aplicación, sino también por qué se ha construido así.

2.1 Análisis de requisitos

2.1.1 Requisitos funcionales

A continuación se enumeran los requisitos funcionales principales agrupados por área. Cada uno está realmente implementado en el código entregado, no son requisitos teóricos.

Autenticación y gestión de usuarios

- **RF1.** Iniciar sesión mediante correo electrónico y contraseña, recibiendo un token JWT firmado con HS256 y vida configurable.
- **RF2.** Recuperar el perfil del usuario autenticado a partir del token (`GET /api/auth/me`), exponiendo identidad, roles y `centro_email` cuando aplica.
- **RF3.** Asignar a cada usuario uno o varios roles entre `ALUMNO`, `EMPRESA`, `TUTOR_CENTRO`, `COORDINADOR` y `ADMIN`, con permisos diferenciados en cada endpoint.
- **RF4.** Almacenar las contraseñas hasheadas con BCrypt; nunca se persisten en claro.
- **RF5.** Generar un usuario tutor inicial al desplegar el backend en producción a partir de variables de entorno, mediante una migración *repeatable* de Flyway, sin hardcodear credenciales en el repositorio.

Catálogo de empresas y ofertas internas

- **RF6.** Alta, edición y baja lógica de empresas colaboradoras, con NIF/CIF, sector, descripción, dirección, localidad, provincia, código postal y contactos.
- **RF7.** Alta, edición y baja lógica de ofertas internas con título, descripción, familia profesional, ciclo formativo objetivo, localidad, provincia, modalidad (presencial, híbrida, remota), fechas, número de plazas, requisitos y tareas.
- **RF8.** Listado público filtrable de ofertas internas por familia profesional, localidad, modalidad y palabra clave.
- **RF9.** Listado privado para una empresa de sus propias ofertas y de las solicitudes recibidas.

Alumnado y preferencias

- **RF10.** Mantenimiento del perfil del alumno con datos básicos y preferencias de FCT (familia, ciclo, localidad preferente, modalidad preferente, fecha de disponibilidad, observaciones).
- **RF11.** Subida de un CV en formato PDF (o equivalente), almacenado como binario en la propia base de datos.
- **RF12.** Subida de foto de perfil para alumno y para cualquier otro rol, almacenada también en la base de datos.

Candidaturas y asignaciones

- **RF13.** El alumnado puede postularse a una oferta interna del catálogo. La unicidad por par (alumno, oferta) está garantizada en la base de datos.
- **RF14.** Las empresas pueden aceptar o rechazar las solicitudes recibidas.
- **RF15.** El tutor y el coordinador pueden registrar manualmente solicitudes externas que el alumno ha gestionado fuera de FCTNow (típicamente desde una oferta de Adzuna).
- **RF16.** El coordinador puede crear una **asignación oficial** a partir de una candidatura aceptada, indicando horas totales (entre 40 y 1000), horas diarias estimadas (entre 1 y 12), fecha de inicio, condiciones de retribución (remunerada / no remunerada, importe mensual) y observaciones.
- **RF17.** Una candidatura aceptada solo puede dar lugar a una asignación; el sistema impide duplicar asignaciones por la misma solicitud.

Ofertas externas (Adzuna)

- **RF18.** Consultar la API pública de Adzuna desde el backend, con palabras clave por defecto pensadas para FCT (“prácticas”, “becario”, “internship”, “pasantía”).
- **RF19.** Filtrar las ofertas externas por palabra clave, localidad, categoría y paginación, manteniendo siempre la atribución a Adzuna.
- **RF20.** Si Adzuna no responde o no hay credenciales configuradas, devolver un `503 Service Unavailable` controlado para que el frontend siga ofreciendo el catálogo interno sin error fatal.

Mensajería y notificaciones

- **RF21.** Crear conversaciones uno a uno entre dos usuarios y enviar mensajes con contenido textual de hasta 2000 caracteres.
- **RF22.** Mantener el histórico cronológico de cada conversación, ordenado por fecha y por identificador.
- **RF23.** Emitir notificaciones in-app al destinatario correspondiente cuando ocurren eventos relevantes del flujo (candidatura aceptada o rechazada, asignación creada, recomendación de oferta por parte del tutor, etc.), con título, mensaje, URL de acción y etiqueta.
- **RF24.** Marcar notificaciones como leídas y borrar las que ya no aplican (por ejemplo, las alertas de asignación pendiente una vez creada la asignación).

Onboarding del alumnado por el tutor

- **RF25.** El tutor puede dar de alta alumnos individualmente con nombre, apellido y correo de centro generado automáticamente, o bien importar un fichero con varios alumnos a la vez.
- **RF26.** Al crear un alumno se genera una contraseña temporal y se envía un correo de bienvenida con instrucciones para acceder a la plataforma.
- **RF27.** El tutor puede consultar y mantener actualizada la información de sus alumnos.

Documentación, listados y trazabilidad

- **RF28.** El backend expone su superficie de API en OpenAPI/Swagger UI para que el resto del equipo y otros desarrolladores puedan explorarla.
- **RF29.** Cada asignación queda enlazada con la candidatura, el alumno, la oferta y la empresa, garantizando la trazabilidad del proceso de extremo a extremo.

2.1.2 Requisitos no funcionales

Seguridad

- **RNF1.** Autenticación basada en tokens JWT firmados con HS256, con expiración configurable mediante `FCTNOW_JWT_EXPIRATION` (por defecto, 1 hora).
- **RNF2.** Autorización por roles aplicada tanto en la capa HTTP (`SecurityFilterChain`) como en la capa de servicio mediante comprobaciones explícitas, para evitar acciones cruzadas (por ejemplo, que un alumno modifique ofertas de otra empresa).
- **RNF3.** Contraseñas hasheadas con BCrypt; las cabeceras `Authorization: Bearer ...` se exigen en todos los endpoints bajo `/api/**` excepto los explícitamente abiertos.
- **RNF4.** Credenciales sensibles (JWT secret, credenciales SMTP, claves de Adzuna, contraseña BCrypt del tutor inicial) gestionadas como variables de entorno y nunca persistidas en el repositorio.
- **RNF5.** Comunicación HTTPS extremo a extremo en producción: Vercel ofrece HTTPS para el frontend, Render lo ofrece para el backend y el cliente Adzuna se llama también por HTTPS.

Rendimiento

- **RNF6.** El catálogo de ofertas internas y las pantallas más frecuentes responden por debajo de 1 segundo en condiciones normales, gracias a índices específicos (`idx_ofertas_fct_estado`, `idx_ofertas_fct_familia`, `idx_ofertas_fct_localidad`, `idx_ofertas_fct_modalidad`) y a la caché *stale-while-revalidate* en frontend.
- **RNF7.** La consulta a Adzuna tiene un timeout configurable de 8 segundos por defecto para evitar bloqueos prolongados ante caídas o lentitud del proveedor externo.
- **RNF8.** Las consultas críticas (listado de mensajes por conversación, listado de notificaciones por destinatario, asignaciones por estado) están respaldadas por índices compuestos pensados para el patrón de acceso real.

Usabilidad y accesibilidad

- **RNF9.** Frontend responsive en escritorio y móvil, con CSS modular por componente.
- **RNF10.** Paneles diferenciados por rol que evitan exponer al usuario opciones que no le corresponden.
- **RNF11.** Validación de formularios en cliente con mensajes claros y validación equivalente en servidor con Bean Validation (`jakarta.validation`).

Mantenibilidad

- **RNF12.** Arquitectura en capas en el backend: controladores REST → servicios → repositorios JPA → entidades.
- **RNF13.** Organización modular en frontend por dominio (`auth`, `alumnos`, `empresas`, `practicas`, `asignaciones`, `mensajes`, `notificaciones`, `fct`, `admin`, `core`, `layout`).
- **RNF14.** Esquema versionado con Flyway y migraciones pequeñas y enfocadas (`V1` a `V15`), aplicadas automáticamente al arranque del backend.
- **RNF15.** Convenciones de nombrado consistentes (`*Controller`, `*Service`, `*Repository`, `*Response`, `*Request`, `*Page`, `*Service` en Angular).

Fiabilidad

- **RNF16.** Gestión consistente de errores HTTP con `ResponseStatusException` y un *exception handler* dedicado para autenticación (`AuthExceptionHandler`).
- **RNF17.** Uso de transacciones (`@Transactional`) en operaciones sensibles (creación de asignación, login, importación masiva de alumnos).
- **RNF18.** La integración con Adzuna degrada con elegancia: si falta configuración o el proveedor cae, el catálogo interno sigue funcionando.

Observabilidad

- **RNF19.** El backend expone un endpoint `/api/health` público para que la plataforma de despliegue (Render) pueda monitorizarlo.
- **RNF20.** Las llamadas críticas a Adzuna registran *warnings* con SLF4J cuando el proveedor falla, suficientes para diagnóstico inicial sin invadir el log con ruido innecesario.

2.2 Tecnologías

La elección de cada componente tecnológico ha sido clave para garantizar la seguridad, la mantenibilidad y la eficiencia del sistema. La selección se ha basado en criterios de madurez, alineación con el temario del CFGS de DAW, compatibilidad con Java 21 y Spring Boot 3.5, y soporte de comunidad consolidado.

2.2.1 Comparativa de las tecnologías valoradas

Backend: Spring Boot (Java) vs Node.js/Express vs Laravel (PHP)

- **Spring Boot 3.5 (Java 21).** Ecosistema maduro, fuerte integración con seguridad (Spring Security, OAuth2 Resource Server, JWT), JPA, validación, correo, migraciones (Flyway) y documentación (springdoc-openapi). Curva de aprendizaje algo más elevada en la parte de seguridad, pero un soporte de comunidad excelente y una superficie de funcionalidades muy alineada con un sistema empresarial.
- **Node.js + Express.** Permite prototipado rápido y compartir lenguaje con el frontend, pero exige seleccionar manualmente librerías de seguridad, ORM, validación y migraciones, sumando muchas decisiones para un equipo de tamaño reducido.
- **Laravel (PHP).** Framework muy completo con autenticación y ORM integrados, pero el equipo había trabajado previamente con Java y Spring, por lo que adoptar PHP habría supuesto descartar buena parte de la experiencia previa.

Frontend: Angular vs React vs Vue

- **Angular 20.** Framework completo y opinionado, con arquitectura clara basada en componentes y servicios, inyección de dependencias, RxJS, Angular Router, TypeScript estricto y soporte oficial de SSR. Curva de aprendizaje mayor que otras opciones, compensada por la coherencia interna.
- **React.** Librería flexible muy adoptada en la industria, pero requiere ensamblar manualmente router, gestor de estado, manejo de formularios y SSR (Next.js u otro framework superior), añadiendo decisiones de diseño que para un proyecto académico de este tamaño no aportaban valor.
- **Vue.** Opción con curva de aprendizaje suave, pero menos presente en el temario del ciclo y en los entornos profesionales locales que Angular o React.

Base de datos: PostgreSQL vs MySQL/MariaDB

- **PostgreSQL.** Muy robusta, soporte avanzado de SQL (CTEs, *upserts*, expresiones regulares en `UPDATE`), tipos como `BYTEA` para almacenar archivos y `TIMESTAMPTZ` para fechas con zona horaria, gran comunidad y excelente integración con Spring Data JPA. Es la opción soportada por Supabase, lo que simplifica el despliegue gestionado.
- **MySQL/MariaDB.** Muy extendido y familiar, pero las funciones de manipulación de texto y algunos detalles del estándar SQL son menos cómodos que en PostgreSQL. La migración `V13`, por ejemplo, depende de `regexp_replace`, `translate` y `substring(... FROM '...')`, mucho más naturales en PostgreSQL.

Comunicación con APIs externas: cliente sincrónico vs reactivo

- **`RestClient` (Spring 6, sincrónico).** Sencillo de configurar, suficiente para una llamada por petición HTTP entrante y compatible con un *timeout* configurable. Es la opción elegida.
- **`WebClient` (reactivo).** Aporta mejor *throughput* en escenarios de muchas llamadas concurrentes, pero exige adoptar un modelo reactivo en toda la cadena para aprovecharlo, lo que para un endpoint puntual era una sobreingeniería innecesaria.

Despliegue: Render + Vercel + Supabase vs todo en uno

- **Heterogéneo (Render + Vercel + Supabase).** Cada plataforma se especializa en una pieza: contenedores con *health check* en Render para el backend, edge global y *rewrites* en Vercel para el frontend, base de datos gestionada con copias automáticas en Supabase. Los planes gratuitos cubren un piloto.
- **Todo en una sola plataforma (por ejemplo, Render para frontend y backend).** Más sencillo de operar, pero los planes gratuitos suelen ser más restrictivos en uno u otro aspecto y los tiempos de despliegue del frontend son peores que en Vercel.

2.2.2 Tecnologías escogidas

Backend

- Java 21 (LTS).
- Spring Boot 3.5.7 con los *starters* Web, Data JPA, Security, Validation, OAuth2 Resource Server y Mail.
- Spring Security + Nimbus JWT (HS256) para autenticación y autorización stateless.
- Spring Data JPA + Hibernate como capa de acceso a datos.
- PostgreSQL como base de datos (15+).
- Flyway (`flyway-core` + `flyway-database-postgresql`) para migraciones.
- springdoc-openapi 2.8 para la documentación OpenAPI / Swagger UI.
- Apache POI 5.4 para procesar ficheros Excel en la importación de alumnos.
- BCrypt (`spring-security-crypto`) para el hashing de contraseñas.
- H2 (alcance `runtime`) como base de datos en memoria para los tests.

Frontend

- Angular 20.3 con SSR (Angular Universal) servido sobre Express 5.
- TypeScript 5.9.
- RxJS 7.8 y Angular Router.
- Estilos CSS por componente sin frameworks externos pesados (decisión consciente para mantener el bundle pequeño y la responsabilidad sobre el diseño).
- Jasmine + Karma para las pruebas unitarias de servicios y componentes.

Desarrollo, pruebas y despliegue

- Git + GitHub para control de versiones y revisión.

- Maven para el backend y npm para el frontend.
- Docker (y Docker Compose) para el PostgreSQL local.
- IntelliJ IDEA y Visual Studio Code como IDEs.
- Postman para validar endpoints en aislamiento.
- Render para el backend en producción, Vercel para el frontend, Supabase para PostgreSQL gestionado.
- Gmail SMTP para el correo transaccional.
- Adzuna API para las ofertas externas.

En conjunto, este *stack* proporciona un equilibrio óptimo entre solidez, mantenibilidad y curva de aprendizaje razonable, alineado con los entornos profesionales actuales y con las competencias del ciclo.

2.3 Estructura del proyecto

FCTNow se organiza como un **monorepo** con dos subproyectos independientes pero coordinados, más el cableado de despliegue y documentación.

```
FCTNow/├──
 backend/                          # Spring Boot 3.5 + PostgreSQL│
   ├── Dockerfile                    # Build multi-stage Maven → JRE 21│
   ├── pom.xml│
   └── src/main/java/com/fctnow/backend/│
       ├── api/                      # HealthController│
       ├── auth/                     # Login, JWT, exception handler│
       ├── config/                   # SecurityConfig, JwtProperties, OpenApi│
       ├── user/                     # UserAccount, roles, UserDetailsService│
       ├── empresas/                 # Empresas colaboradoras│
       ├── ofertas/                  # Ofertas internas│
       │   └── externas/             # Cliente Adzuna│
       ├── alumnos/                  # Preferencias y CV│
       ├── solicitudes/              # Candidaturas internas y externas│
       │   └── externas/│
       ├── asignaciones/             # Asignación FCT formal│
       │   └── externas/│
       ├── tutor/                    # Onboarding del alumnado por tutor│
       ├── mensajes/                 # Mensajería interna│
       ├── notificaciones/           # Notificaciones in-app│
       └── home/                     # Resumen para la pantalla inicial del alumno│
   └── src/main/resources/│
       ├── application.yml│
       ├── application-local.yml│
       ├── application-prod.yml│
       └── db/migration/…V1__init V15__add_asignacion_seguimiento.sql│
   └── src/test/java/com/fctnow/backend/   # 15 clases de test│├──

 frontend/                         # Angular 20 con SSR│
   ├── package.json│
   ├── vercel.json                   # Output dir + rewrites a Render│
   ├── proxy.conf.json               # Proxy /api → localhost:8080 en dev│
   └── src/app/│
       ├── core/                     # api.config, auth services, role guard│
       ├── layout/                   # Navegación principal│
       ├── auth/                     # Login y storage de sesión│
       ├── home/                     # Feed del alumno│
       ├── practicas/                # Catálogo público + Adzuna│
       ├── alumnos/                  # Panel alumno, mis solicitudes│
       ├── empresas/                 # Panel empresa, ofertas, solicitudes│
       ├── asignaciones/             # Panel coordinador + cache + progreso│
       ├── fct/                      # Panel tutor y coordinador│
       ├── mensajes/                 # Chat│
       ├── notificaciones/           # Bandeja de notificaciones│
       ├── perfil/                   # Perfil de cualquier rol│
       ├── admin/                    # Panel administrativo│
       └── not-found/                # 404│├──

 docker-compose.yml                # PostgreSQL local en puerto 15432├──
 README.md                         # Documentación raíz del monorepo├──
 MemoriaFCTConnect.pdf             # Anteproyecto inicial└──
 MEMORIA.md                        # Esta memoria
```

A nivel lógico, el sistema responde a una **arquitectura en capas** clásica del lado del servidor combinada con una **SPA modular por dominio** del lado del cliente:

Capa de presentación (Angular SPA + SSR)

- Componentes y plantillas, organizados por dominio de negocio.
- Servicios Angular que envuelven las llamadas HTTP a la API.
- *Guards* de ruta basados en rol que protegen el panel del coordinador, el panel del tutor y la administración.
- Caché *stale-while-revalidate* en `practicas-cache.service.ts`, `asignaciones-cache.service.ts` y `tutor-cache.service.ts` para reducir latencia percibida.

Capa de aplicación (servicios Spring)

- Lógica de negocio que orquesta autenticación, autorización fina por rol, integridad de candidaturas y asignaciones, envío de correos transaccionales y consumo de la API externa.
- Validaciones con `jakarta.validation` y transformación entre DTO y entidad.

Capa de acceso a datos (Spring Data JPA)

- Entidades JPA mapeadas a tablas PostgreSQL.
- Repositorios `Spring Data` con métodos derivados de su nombre (`findByEmailIgnoreCase`, `findByDestinatarioIdOrderByCreatedAtDesc`, ...).
- Esquema versionado con Flyway y aplicado al arranque.

Capa de integración

- Cliente HTTP sincrónico (`RestClient`) para Adzuna.
- `JavaMailSender` (Spring Mail) para el correo transaccional vía SMTP de Gmail.

Capa de despliegue

- Backend empaquetado como JAR ejecutable dentro de un contenedor Docker multi-stage y publicado en Render.
- Frontend compilado a estáticos y publicado en Vercel con *rewrite* hacia `https://fctnow-backend.onrender.com/api/...`.
- Base de datos en Supabase con conexión SSL.

2.4 Descripción de los componentes

A continuación se detallan los módulos principales del sistema y sus responsabilidades.

Módulo de autenticación y seguridad

- **`SecurityConfig`** declara la cadena de filtros: deshabilita CSRF (la API es stateless), establece `SessionCreationPolicy.STATELESS`, abre los endpoints estrictamente públicos (`POST /api/auth/login`, listado público de ofertas, `health`, OpenAPI y Swagger UI) y exige autenticación JWT para el resto de `/api/**`.
- **`JwtTokenService`** emite tokens HS256 con *claims* `sub` (email), `user_id`, `display_name`, `roles` y `centro_email` cuando aplica. El secreto y la expiración se leen de `JwtProperties`.
- **`AuthService`** orquesta el login: delega en el `AuthenticationManager` para validar usuario y contraseña con BCrypt y, si la autenticación tiene éxito, recupera el `UserAccount` y emite el token.
- **`UserAccountDetailsService`** convierte un `UserAccount` en el `UserDetails` que necesita Spring Security, aplicando el prefijo `ROLE_` a cada rol declarado en la base de datos.
- **`AuthExceptionHandler`** y **`RestAuthenticationEntryPoint`** uniformizan las respuestas de error de autenticación (`401` y `403`) con un cuerpo JSON consistente.

Módulo de empresas y ofertas internas

- **`Empresa`** y **`EmpresaService`** gestionan la entidad colaboradora con CIF/NIF, sector, dirección, contactos y estado.
- **`OfertaFct`** representa una oferta de prácticas con familia profesional, ciclo, localidad, modalidad, fechas, plazas, requisitos, tareas y estado.
- **`OfertaFctController`** expone el catálogo público (`GET /api/ofertas`, `GET /api/ofertas/{id}`) sin necesidad de autenticación, con filtros por familia profesional, localidad, modalidad y palabra clave.
- **`EmpresaOfertaController`** expone la gestión privada de las ofertas para el rol `EMPRESA` (alta, edición, cambio de estado).

Módulo de alumnado y preferencias

- **`AlumnoPreferencias`** almacena las preferencias del alumno (familia, ciclo, localidad, modalidad, fecha de disponibilidad, observaciones) más los binarios de CV y foto.
- **`AlumnoPreferenciasService`** mantiene el perfil, sube/descarga el CV (PDF) y la foto (JPEG/PNG) con sus metadatos.
- **`AlumnoCvResource`** expone la descarga del CV bajo demanda.

Módulo de candidaturas y asignaciones

- **`SolicitudFct`** representa una candidatura del alumno a una oferta interna, con estados `PENDIENTE`, `ACEPTADA` y `RECHAZADA`.
- **`SolicitudFctController`** y **`EmpresaSolicitudController`** dan al alumno la vista “mis solicitudes” y a la empresa la bandeja de “solicitudes recibidas”.

- **`SolicitudExterna`** y **`SolicitudExternaService`** registran el ciclo de vida de las candidaturas externas (típicamente derivadas de una oferta Adzuna).
- **`AsignacionFctController`** y `AsignacionFctService` permiten al coordinador formalizar una asignación a partir de una candidatura aceptada, con horas totales, horas diarias, fecha de inicio y datos de retribución, garantizando unicidad por candidatura.
- **`AsignacionFctExternaController`** hace lo equivalente para asignaciones derivadas de candidaturas externas.

Módulo de ofertas externas (Adzuna)

- **`AdzunaProperties`** centraliza la configuración (base URL, país, *app id*, *app key*, palabras clave por defecto y *timeout*).
- **`AdzunaConfig`** construye un `RestClient` con `SimpleClientHttpRequestFactory` y los *timeouts* configurados.
- **`AdzunaClient`** ejecuta la petición, captura y traduce los errores en `AdzunaUnavailableException`.
- **`AdzunaService`** valida parámetros, aplica límites (resultados por página entre 1 y 50, página mínima 1), llama al cliente y mapea cada `AdzunaJob` a `OfertaExternaResponse`, incluyendo la atribución obligatoria a Adzuna.
- **`OfertaExternaController`** expone `GET /api/ofertas/externas` con `q`, `where`, `category`, `page` y `resultsPerPage`, devolviendo un `503 Service Unavailable` controlado si Adzuna falla.

Módulo de mensajería

- **`Conversacion`** y **`Mensaje`** modelan conversaciones uno a uno entre dos `app_users` distintos, con índices por participante y por orden cronológico.
- **`MensajeController`** y `ConversacionRepository` ofrecen los endpoints para listar conversaciones del usuario actual, crear una nueva, listar mensajes de una conversación y enviar mensajes nuevos.

Módulo de notificaciones

- **`Notificacion`** soporta notificaciones in-app dirigidas a cualquier `app_user` (no solo alumnado, gracias a la generalización introducida en `V12`), con título, mensaje, URL de acción, etiqueta y enlaces opcionales a una oferta interna o a los datos snapshot de una oferta externa.
- **`NotificacionService`** se invoca desde el resto de módulos cuando ocurre un evento relevante (candidatura aceptada o rechazada, asignación creada, alumno creado por el tutor, recomendación de oferta).
- **`NotificacionController`** expone el listado paginado del usuario actual, el marcado como leída y el borrado de notificaciones obsoletas.

Módulo de onboarding por tutor

- **`TutorAlumnoController`** ofrece tres operaciones principales: crear un alumno individual, importar un fichero Excel/CSV con varios alumnos a la vez y listar los alumnos del tutor.
- **`AlumnoWelcomeMailer`** envía un correo de bienvenida con la contraseña temporal y el enlace de login (`FCTNOW_MAIL_LOGIN_URL`).
- El correo de centro se genera automáticamente a partir del nombre y apellido del alumno siguiendo la convención `nombreApellido@elpuig.xeill.net`.

Módulo Home del alumno

- **`StudentHomeController`** y **`StudentHomeService`** componen un *feed* personalizado para el alumno con ofertas recomendadas, candidaturas activas, próximas asignaciones y notificaciones recientes, evitando que el frontend tenga que orquestar varias llamadas independientes.

Capa de cliente Angular

- **`core`** contiene la configuración de la API base (`api.config.ts`), los servicios de autenticación (`auth.service.ts`, almacenamiento del JWT en `sessionStorage`) y los *guards* (`role.guard.ts`).
- **`layout`** define la navegación principal y el chrome común (cabecera, *footer*, accesos rápidos).
- Cada módulo de dominio (`practicas`, `alumnos`, `empresas`, `asignaciones`, `fct`, `mensajes`, `notificaciones`, `perfil`, `admin`) sigue la misma estructura: un componente página, los componentes auxiliares, los servicios HTTP, los modelos y, cuando aplica, un servicio de caché.

2.5 Definición de las funcionalidades

A continuación se describen las funcionalidades clave vistas desde el flujo de uso real de la aplicación.

1. Inicio de sesión y acceso según rol El usuario accede a `/login`, introduce su correo y contraseña, el backend valida las credenciales con BCrypt y devuelve un JWT con sus *claims*. El frontend almacena el token en `sessionStorage` y lo añade como `Authorization: Bearer ...` a cada petición a `/api/**`. Las rutas protegidas por `roleGuard(['TUTOR_CENTRO', 'COORDINADOR'])` o `roleGuard(['COORDINADOR'])` redirigen al login si el token no contiene el rol adecuado.

2. Alta de empresa y creación de ofertas Un usuario con rol `EMPRESA` accede a su panel (`/empresa`), gestiona sus ofertas (`/empresa/ofertas`) y crea una nueva mediante un formulario validado (`/empresa/ofertas/nueva`). La oferta incluye familia profesional, ciclo objetivo, localidad, modalidad, fechas, plazas, requisitos y tareas. Tras guardarse, la oferta aparece inmediatamente en el catálogo público filtrable.

3. Búsqueda de prácticas Cualquier visitante puede consultar el catálogo `/practicas` con filtros por familia profesional, localidad, modalidad y palabra clave. Las ofertas internas se complementan con ofertas externas obtenidas en tiempo real de Adzuna, claramente etiquetadas con la atribución correspondiente. Si Adzuna no está disponible, el frontend muestra solo el catálogo interno sin que la pantalla rompa.

4. Gestión del perfil del alumno y preferencias Tras iniciar sesión, el alumno completa su perfil en `/perfil`: preferencias (familia, ciclo, localidad, modalidad, fecha de disponibilidad), observaciones, CV en PDF y foto. Estos datos alimentan posteriores recomendaciones y el panel del tutor.

5. Candidaturas internas Desde el detalle de una oferta interna (`/practicas/:id`) el alumno se postula con un solo clic. La candidatura aparece en `/alumno/solicitudes` y, simultáneamente, en la bandeja de solicitudes de la empresa. La empresa acepta o rechaza desde `/empresa/solicitudes`, generando una notificación in-app para el alumno.

6. Candidaturas externas (Adzuna) Cuando un alumno aplica a una oferta externa de Adzuna fuera de FCTNow, puede registrar manualmente esa candidatura desde la pantalla de detalle. El registro queda almacenado con su `id_externo`, título, empresa, localidad, URL de aplicación y categoría, evitando perder la trazabilidad por el simple hecho de que la oferta no sea interna.

7. Onboarding del alumnado por el tutor El tutor accede a `/tutor`, crea alumnos uno a uno o importa un fichero. El sistema genera contraseña temporal, envía un correo de bienvenida con las instrucciones de acceso y deja al alumno listo para usar FCTNow desde su primer login. El tutor también puede actualizar datos o reenviar el correo si fuera necesario.

8. Asignación final por el coordinador El coordinador accede a `/asignaciones`, ve todas las candidaturas aceptadas pendientes de asignar (internas y externas) y crea la asignación oficial introduciendo horas totales, horas diarias estimadas, fecha de

inicio y condiciones de retribución. El sistema valida que las horas estén entre 40 y 1000, que las diarias estén entre 1 y 12 y que el importe mensual solo exista si la asignación está marcada como remunerada. Una vez creada, la asignación aparece en el panel del alumno con un componente de progreso de horas (`fct-progress.ts`).

9. Mensajería interna Cualquier usuario autenticado accede a `/mensajes`, inicia o continúa conversaciones uno a uno con otro participante y consulta el histórico ordenado cronológicamente. La pantalla es responsive y se integra con las notificaciones para resaltar conversaciones con mensajes no leídos.

10. Notificaciones in-app Las acciones relevantes del flujo (aceptación de candidatura, creación de asignación, recomendación de oferta por parte del tutor) generan notificaciones que aparecen en `/notificaciones` con título, mensaje, URL de acción y etiqueta. El usuario las marca como leídas o las descarta. Cuando una notificación deja de ser relevante (por ejemplo, una asignación pendiente que ya se ha creado), el sistema la limpia automáticamente.

11. Pantalla inicial personalizada para el alumno La home del alumno (`/`) compone en una única vista las ofertas recomendadas (internas y externas), las candidaturas activas, la próxima asignación y las notificaciones recientes, ahorrando al alumno tener que abrir cada sección por separado.

12. Admin El panel `/admin`, accesible para usuarios con rol `ADMIN`, ofrece una vista agregada del estado del sistema (cuentas, empresas, ofertas activas, asignaciones en curso) pensada para coordinación de centro.

3. Pruebas y validación

Para asegurar que FCTNow cumple sus requisitos funcionales y no funcionales, se ha definido una estrategia de verificación basada en pruebas unitarias y de controlador automatizadas en backend, pruebas de servicio y componente en frontend, y pruebas funcionales manuales que han recorrido el flujo de extremo a extremo en el entorno de producción real.

3.1 Estrategia de pruebas

El alcance de las pruebas ha abarcado los módulos críticos del sistema:

- **Autenticación.** Emisión y verificación de JWT, control de acceso por rol, manejo de errores `401` y `403`.
- **Catálogo interno.** Filtrado de ofertas, alta y edición desde el rol empresa, validación de campos obligatorios.
- **Candidaturas.** Unicidad por par (alumno, oferta), aceptación, rechazo y emisión de notificaciones.
- **Asignaciones.** Creación a partir de candidatura aceptada, restricciones `CHECK` de horas y retribución, bloqueo de duplicados.
- **Solicitudes externas.** Registro manual de candidaturas externas, transición de estados.
- **Integración Adzuna.** Manejo del *timeout*, mapeo de respuestas, degradación cuando faltan credenciales o el proveedor falla.
- **Mensajería.** Creación de conversaciones, envío de mensajes, listado por participante.
- **Onboarding por tutor.** Alta individual, importación masiva, generación de correo de centro y envío del *welcome mail*.

La estrategia se ha apoyado en tres niveles complementarios:

1. **Pruebas unitarias y de controlador automatizadas** en backend (JUnit 5 + `spring-boot-starter-test` + Mockito + `MockMvc`), apoyadas en H2 como base en memoria para los tests que requieren persistencia.
2. **Pruebas de servicio y de componente** en frontend (Jasmine + Karma) que verifican los servicios HTTP y el comportamiento de los componentes Angular más relevantes.
3. **Pruebas funcionales manuales** en local y en producción, simulando los escenarios reales: alta de empresa, publicación de oferta, candidatura, aceptación, asignación, mensajería, recepción de correo de bienvenida.

Los criterios de validación han sido los siguientes:

- Comprobación visual y funcional de todos los flujos críticos desde el navegador, con varias cuentas distintas para cubrir cada rol.

- Verificación de que la API devuelve los códigos de estado HTTP correctos (`200`, `201`, `204`, `400`, `401`, `403`, `404`, `503`).
- Comprobación de que las restricciones de la base de datos (`UNIQUE`, `CHECK`, `FOREIGN KEY`) hacen su trabajo cuando se intenta romper la consistencia desde el cliente o desde Postman.
- Pruebas iterativas durante el desarrollo, con regresión automática mediante `mvn test` y `npm test` en cada *pull request*.

3.2 Pruebas funcionales manuales

Se han ejecutado los siguientes escenarios de validación manual:

Autenticación y autorización

- Login con credenciales correctas → se recibe JWT y se redirige al panel correspondiente al rol.
- Login con credenciales incorrectas → respuesta `401` y mensaje claro en la interfaz.
- Acceso a `/asignaciones` con rol distinto de `COORDINADOR` → bloqueado por el guard.
- Acceso a `/empresa/...` sin sesión → redirección a `/login`.
- Token expirado → la siguiente petición devuelve `401`, el frontend limpia la sesión y reenvía a `/login`.

Empresas y ofertas

- Alta de empresa con NIF duplicado → bloqueada por la restricción `UNIQUE` (`empresas_identificador_fiscal_key`), con mensaje en la interfaz.
- Creación de oferta con campos obligatorios vacíos → mensajes de error de Bean Validation devueltos por el backend y mostrados por el formulario.
- Listado público filtrable → los filtros se traducen en *query params* y los resultados son consistentes con los esperados por el SQL.
- Cambio de estado de oferta de `ACTIVA` a `CERRADA` → la oferta deja de aparecer en el catálogo público.

Candidaturas

- Postulación a la misma oferta dos veces por el mismo alumno → bloqueada por `uk_solicitudes_fct_alumno_oferta` y reportada en la interfaz.
- Aceptación por la empresa → el alumno recibe notificación in-app y un correo opcional.
- Rechazo por la empresa → el alumno recibe notificación in-app, la candidatura aparece como rechazada en su bandeja.

Asignaciones

- Creación de asignación con `horas_totales` fuera de `[40, 1000]` → bloqueada por `ck_asignaciones_fct_horas_totales`.
- Creación con `remunerada = false` y `importe_mensual = 600` → bloqueada por `ck_asignaciones_fct_importe_remunerada`.

- Doble asignación para la misma candidatura → bloqueada por `uk_asignaciones_fct_solicitud`.
- Vista del alumno tras la asignación → se muestra la barra de progreso de horas y los datos económicos.

Ofertas externas y Adzuna

- Búsqueda en `/practicas` con Adzuna disponible → el catálogo combina ofertas internas y externas con atribución visible.
- Búsqueda con Adzuna caído (simulado quitando temporalmente las credenciales en producción) → `503` controlado y el catálogo interno sigue funcionando.
- Registro manual de candidatura externa → aparece en `/alumno/solicitudes` con la fuente etiquetada como Adzuna.

Mensajería

- Inicio de conversación entre alumno y empresa → ambos ven la conversación en su bandeja, el orden cronológico es correcto.
- Envío de mensaje de más de 2000 caracteres → bloqueado por la validación.
- Envío de mensaje a uno mismo → bloqueado por `chk_conversaciones_participantes_distintos`.

Onboarding del alumnado

- Alta individual con nombre "David Barbosa" → se genera `centro_email = davidbarbosa@elpuig.xeill.net`, contraseña temporal, correo de bienvenida.
- Importación de fichero Excel con 30 alumnos → todos los alumnos se crean correctamente, se devuelve un resumen con los éxitos y los errores fila a fila.

3.3 Consideraciones sobre integración y automatización

El backend dispone de **15 clases de test automatizadas** que cubren los controladores principales (auth, health, empresas, ofertas internas, ofertas externas Adzuna, solicitudes internas y externas, asignaciones internas y externas, mensajes, tutor, preferencias). Estas pruebas se ejecutan con `mvn test` y se apoyan en `MockMvc` para verificar respuestas HTTP, Mockito para aislar la capa de servicio y H2 (cuando aplica) para validar consultas reales contra una base de datos compatible.

El frontend cuenta con **más de 30 archivos** `*.spec.ts` que prueban tanto los servicios HTTP como los componentes página (login, panel alumno, panel empresa, asignaciones, mensajes, perfil, prácticas, etc.). Se ejecutan con `npm test` y se han mantenido verdes a lo largo del desarrollo, sirviendo como red de seguridad frente a regresiones.

De cara a una continuación natural del proyecto, las siguientes mejoras se han considerado pero no implementado por estar fuera del alcance del TFG:

- **Pipeline de CI/CD con GitHub Actions** que ejecute automáticamente `mvn test` y `npm test` en cada PR y bloquee el *merge* si fallan, eliminando la dependencia de la disciplina manual.

- **Pruebas de extremo a extremo** con Cypress o Playwright que reproduzcan los escenarios de "alta de empresa → publicación de oferta → candidatura → asignación" sin intervención humana.
- **Pruebas de contrato (Pact o similar)** entre frontend y backend que garanticen que ningún cambio en los DTOs rompa el cliente sin previo aviso.

3.4 Pruebas de rendimiento

No se han ejecutado pruebas de carga formales (JMeter o Gatling), por estar fuera del alcance del proyecto, pero se han realizado **comprobaciones de tiempos de respuesta en producción** con cargas realistas para un centro de FP:

- Listado público de ofertas internas con 50-100 registros: tiempo de respuesta inferior a 200 ms una vez el backend está caliente, gracias a los índices por estado, familia profesional, localidad y modalidad.
- Búsqueda Adzuna con 20 resultados por página: dominada por la latencia de la API externa (entre 500 ms y 2 s habitualmente).
- Listado de notificaciones por destinatario: por debajo de 50 ms gracias al índice compuesto `idx_notificaciones_destinatario_created`.
- Listado de mensajes por conversación con varias decenas de mensajes: por debajo de 100 ms gracias a `idx_mensajes_conversacion_created_at`.
- Login: dominado por el cálculo de BCrypt (~100 ms), aceptable para una operación de baja frecuencia.

El principal cuello de botella detectado en el entorno de producción no es la aplicación sino el ***cold start* del plan gratuito de Render**: tras un periodo de inactividad, la primera petición tarda decenas de segundos en responder mientras arranca la JVM. Una vez caliente, los tiempos son los esperables para un Spring Boot empaquetado como JAR. Este coste no se elimina con cambios en el código, sino pasando a un plan de pago de Render o configurando un ping periódico al `/api/health` para mantener la instancia activa.

4. Conclusiones

4.1 Conclusiones generales del proyecto

FCTNow se entrega como una plataforma web *full-stack* completa, desplegada y validada de extremo a extremo, que cubre los flujos clave de la gestión de FCT de un centro de Formación Profesional. El proyecto ha conseguido aglutinar en una sola aplicación los procesos que tradicionalmente se reparten entre hojas de cálculo, correos electrónicos, formularios en papel y la memoria informal del profesorado, ofreciendo a cada uno de los cinco roles del dominio (`ALUMNO`, `EMPRESA`, `TUTOR_CENTRO`, `COORDINADOR`, `ADMIN`) una superficie funcional pensada para su trabajo concreto.

El sistema resultante no es una maqueta ni un prototipo: el backend está desplegado en Render con su base de datos PostgreSQL en Supabase, el frontend en Vercel y el dominio público es accesible a través de un navegador. La integración con la API de Adzuna ofrece ofertas reales del mercado, el correo transaccional vía Gmail SMTP entrega bienvenidas a los alumnos creados por el tutor y las notificaciones in-app responden a eventos reales del flujo. El histórico de Git (más de 90 *commits* y 80 *pull requests* fusionados) refleja un desarrollo iterativo y revisado, no un sprint final apresurado.

Desde el punto de vista formativo, el proyecto ha permitido al equipo aplicar prácticamente todas las áreas troncales del CFGS de DAW: modelado de bases de datos, desarrollo de servicios REST con un framework empresarial, seguridad con JWT, desarrollo de SPAs con un framework moderno, pruebas automatizadas, integración con APIs externas, despliegue en la nube y operación real en un entorno con usuarios potenciales. La curva de aprendizaje ha sido sostenida pero gestionable, y la decisión de avanzar en fases verticales pequeñas ha sido determinante para evitar bloqueos prolongados.

4.2 Consecución de los objetivos

Repasando los objetivos específicos planteados en el apartado 1.3.2:

- **Modelado completo del dominio FCT.** Conseguido. El esquema cuenta con 15 migraciones Flyway que cubren autenticación, empresas, ofertas internas, candidaturas, asignaciones, ofertas externas, mensajería, notificaciones, perfil con foto y seguimiento de horas y retribución.
- **Autenticación y autorización robustas.** Conseguido. JWT HS256 firmado y verificado por Spring Security en modo *resource server*, BCrypt para las contraseñas, cinco roles funcionales con control en endpoint y servicio.
- **Catálogo de empresas y ofertas internas.** Conseguido. Catálogo público filtrable, panel privado de empresa con CRUD completo y baja lógica de ofertas mediante estado.

- **Integración con fuentes externas de ofertas.** Conseguido. Cliente Adzuna completo, fallback `503` controlado, atribución obligatoria, configuración por variables de entorno.
- **Flujo completo de candidaturas y asignación.** Conseguido. Candidaturas internas y externas, aceptación o rechazo por la empresa, asignación oficial por el coordinador con horas, fecha de inicio y retribución.
- **Gestión del alumnado por parte del tutor.** Conseguido. Alta individual, importación masiva por fichero, contraseña temporal y correo de bienvenida, generación automática del correo de centro.
- **Comunicación interna entre actores.** Conseguido. Mensajería uno a uno con histórico cronológico, notificaciones in-app para los eventos clave, generalización del destinatario a cualquier rol.
- **Frontend moderno, responsive y consciente del rol.** Conseguido. SPA Angular 20 con SSR, *guards* por rol, caché *stale-while-revalidate*, paneles diferenciados, pulido visual de cards y estados vacíos.
- **Despliegue en producción.** Conseguido. Backend en Render como contenedor Docker, frontend en Vercel con *rewrite* a la API, PostgreSQL gestionado en Supabase, correo transaccional vía Gmail SMTP, *bootstrap* del tutor inicial por variables de entorno.
- **Calidad del software.** Conseguido en su tramo central. 15 clases de test en backend con `MockMvc` y Mockito; más de 30 archivos `*.spec.ts` en frontend con Jasmine y Karma; OpenAPI/Swagger UI disponible. La automatización por CI queda como una mejora prevista.

En conjunto, **los objetivos planteados al inicio se han cumplido en su totalidad**, con algunos casos (mensajería, importación masiva de alumnos, integración Adzuna) que han ido más allá del ámbito mínimo previsto en el anteproyecto.

4.3 Valoración de la metodología y planificación

4.3.1 Aspectos positivos

- **El enfoque “vertical y por *issue*”** ha funcionado muy bien. Tener una *issue* por cada bloque funcional, una rama corta por *issue* y un *pull request* con criterios de aceptación claros ha mantenido la trazabilidad del proyecto sin ahogarlo en burocracia.
- **`main` siempre desplegable.** El despliegue automático en Render y Vercel ha forzado a tratar `main` como rama de producción real. Cualquier regresión llegaba en minutos al entorno público, lo que ha mantenido el listón de calidad alto sin necesidad de procesos pesados.
- **Conventional Commits.** A los pocos días de adoptarlos, el historial pasó a ser legible de un vistazo, sin necesidad de abrir cada diff. Cuando hubo que diagnosticar regresiones o documentar cambios para esta memoria, esa decisión ha ahorrado horas de trabajo.
- **Versionado del esquema con Flyway desde el primer día.** Empezar la primera migración con una tabla técnica de bootstrap sonó excesivo en su momento, pero ha pagado con creces: nunca se ha perdido tiempo intentando reconciliar

manualmente esquemas entre entornos, y el seed del tutor en producción se ha podido entregar como otra migración más sin hacks fuera de banda.
- **Decisión consciente sobre el alcance.** Renunciar explícitamente a algunas funcionalidades vistas en el anteproyecto (generación automática de convenios en PDF, integración con un SGA real, módulo completo de evaluación) ha permitido completar de forma sólida lo que sí estaba en el alcance, en lugar de entregar muchas cosas a medias.

4.3.2 Áreas de mejora

- **Falta de CI/CD automático.** Aunque los tests se ejecutaban antes de cada PR, no había un *pipeline* en GitHub Actions que lo hiciera obligatorio. Una primera mejora natural sería añadir un *workflow* que corra `mvn test` y `npm test` y bloquee los *merges* si algo falla.
- **Cobertura de tests no medida.** Las pruebas existen y son numerosas, pero no se han recogido métricas de cobertura ni se ha fijado un umbral mínimo. JaCoCo en backend e Istanbul en frontend cubrirían esta carencia.
- **Falta de pruebas E2E.** Los flujos críticos (alta de empresa → oferta → candidatura → asignación) se validan manualmente. Sustituir esa validación por pruebas Cypress o Playwright daría confianza adicional con cada cambio.
- **Generación de documentación oficial.** El alcance inicial preveía la generación o adjunto de convenios y anexos en PDF; finalmente se ha priorizado el flujo digital y la documentación oficial queda fuera. Es una pieza identificada para una segunda iteración.
- **Internacionalización.** Toda la interfaz está en castellano, sin soporte para catalán o inglés. Para un centro de Cataluña como el IES Puig Castellar, soportar al menos catalán sería natural y enriquecería la solución.
- ***Cold start* del plan gratuito de Render.** Aceptable para un piloto académico, no para una implantación real. La solución pasa por un plan de pago o por un *ping* periódico al `/api/health`.

4.4 Visión de futuro

FCTNow está pensado para evolucionar más allá de la entrega académica. A continuación se proponen líneas de continuación ordenadas por valor potencial.

4.4.1 Generación automática de documentación oficial

Generar automáticamente el convenio de colaboración, los anexos y los partes de horas en PDF a partir de los datos ya estructurados en la base de datos. Esto cerraría el círculo del flujo administrativo: hoy FCTNow registra la asignación con todos sus detalles, pero la documentación oficial sigue rellenándose fuera. La integración con plantillas (por ejemplo, con OpenPDF o iText) y la firma electrónica (eIDAS o un proveedor sencillo) son la siguiente frontera.

4.4.2 Integración con sistemas académicos del centro

Conectar FCTNow con el Sistema de Gestión Académica del centro (SGA) para evitar duplicar el alta de alumnos y para empujar el resultado final del módulo de FCT directamente a la gestión académica oficial. La integración puede plantearse en dos niveles: importación bajo demanda mediante CSV/Excel (ya parcialmente soportada por el módulo de tutor) o integración por API si el SGA expone una.

4.4.3 Evaluación estructurada y rúbricas de empresa

Ampliar el modelo de seguimiento con rúbricas estructuradas que permitan a la empresa y al tutor valorar al alumno sobre criterios concretos del ciclo (competencias técnicas, autonomía, trabajo en equipo, comunicación). El modelo actual ya registra la asignación; faltaría una capa de evaluación con campos numéricos y comentarios libres por criterio, junto con su exportación a informes.

4.4.4 Mejora del *matching* alumno-oferta

Hoy el alumno consulta el catálogo y se postula manualmente, y el coordinador filtra las candidaturas por intuición y conocimiento del perfil. Un futuro motor de recomendación podría sugerir, para cada alumno, las ofertas más adecuadas según familia profesional, ciclo, localidad preferente y modalidad. Una primera versión puede implementarse con reglas simples; una segunda, con un modelo más fino.

4.4.5 Aplicación móvil para tutores y alumnos

La SPA Angular es responsive, pero un cliente móvil nativo (o una PWA bien instrumentada) ofrecería notificaciones push reales, acceso offline limitado y mejor integración con el calendario y el correo del dispositivo. Para tutores que visitan empresas, una app móvil con registro de visita simplificaría mucho el día a día.

4.4.6 Análisis y métricas agregadas para coordinación

Aprovechar que toda la información está ya estructurada para ofrecer al coordinador un panel con métricas agregadas: número de ofertas por ciclo, tasa de aceptación por empresa, distribución de modalidad, tiempo medio entre candidatura y asignación, incidencias por curso. Este panel convierte FCTNow en una herramienta no solo operativa sino también de decisión a medio plazo.

4.5 Conclusión final

FCTNow se entrega como un sistema completo, real, desplegado en producción y validado de extremo a extremo, que demuestra cómo un proyecto académico bien acotado puede ofrecer una respuesta seria a un problema concreto de un centro educativo. La combinación de un backend Spring Boot 3 con un frontend Angular 20, una base de datos PostgreSQL versionada con Flyway, una integración con la API pública de Adzuna y un despliegue en Render, Vercel y Supabase ha permitido

cubrir todas las etapas del flujo de FCT —empresas, ofertas, candidaturas, asignaciones, seguimiento, mensajería y notificaciones— manteniendo al mismo tiempo una calidad técnica defendible y un coste de explotación prácticamente nulo.

Más allá del cumplimiento estricto de los objetivos del TFG, el proyecto deja un mensaje sostenido en el tiempo: que las soluciones software pueden mejorar de manera tangible la vida de los centros de Formación Profesional, reduciendo carga administrativa, ganando trazabilidad y devolviendo al alumnado información clara sobre un proceso que les afecta directamente. FCTNow no pretende ser un producto cerrado, sino una base sólida sobre la que crecer: tanto técnicamente, hacia las líneas de futuro descritas, como organizativamente, hacia una eventual implantación piloto en el propio centro.

5. Glosario

API REST Interfaz de programación basada en el estilo arquitectónico REST que expone recursos a través de HTTP (GET, POST, PUT, PATCH, DELETE).

Adzuna Proveedor internacional de ofertas de empleo agregadas con una API pública. FCTNow lo consume para enriquecer el catálogo con ofertas reales del mercado, manteniendo la atribución obligatoria.

Asignación FCT Registro formal en FCTNow que oficializa la colocación de un alumno en una empresa concreta para realizar la FCT, con horas totales, fecha de inicio, condiciones de retribución y observaciones.

BCrypt Algoritmo de *hashing* de contraseñas adaptable a la potencia de cálculo disponible, considerado robusto para proteger credenciales en reposo. Es el algoritmo que FCTNow usa para almacenar las contraseñas de los usuarios.

Candidatura (solicitud) Acto por el cual un alumno se postula a una oferta interna del catálogo. Tiene estados `PENDIENTE`, `ACEPTADA` y `RECHAZADA`.

Candidatura externa Equivalente a la candidatura interna pero para ofertas que no están en el catálogo de FCTNow (normalmente venidas de Adzuna). El alumno la registra manualmente para mantener la trazabilidad.

CFGS Ciclo Formativo de Grado Superior. Nivel formativo postobligatorio dentro de la Formación Profesional española.

Cold start Demora añadida en la respuesta de un servicio cuando la plataforma de hosting lo arranca desde frío tras un periodo de inactividad. Es característica de los planes gratuitos de muchas plataformas, incluido Render.

CRUD Acrónimo de *Create, Read, Update, Delete*: las cuatro operaciones básicas que se aplican a un recurso persistente.

DAW Desarrollo de Aplicaciones Web. Ciclo formativo del que es proyecto integrador FCTNow.

Docker Plataforma de contenedores que permite empaquetar una aplicación junto con su entorno de ejecución para reproducirla en cualquier máquina. Se usa para PostgreSQL en local y para empaquetar el backend en producción.

FCT Formación en Centros de Trabajo. Módulo obligatorio de la Formación Profesional española en el que el alumnado realiza prácticas profesionales en una empresa colaboradora.

Flyway Herramienta de versionado y aplicación automática de migraciones SQL para bases de datos relacionales. FCTNow la usa para evolucionar el esquema de PostgreSQL de forma reproducible.

JPA / Hibernate *Java Persistence API*: estándar Java para mapear objetos a tablas relacionales. Hibernate es su implementación de referencia. Spring Data JPA es la capa que añade Spring por encima para reducir el código boilerplate.

JWT (JSON Web Token) Formato compacto de token utilizado para autenticación y autorización, que se transporta en la cabecera `Authorization: Bearer ...`. FCTNow firma sus JWT con HS256.

Maven Herramienta de construcción y gestión de dependencias para proyectos Java. Define el ciclo de vida estándar (`compile`, `test`, `package`, `install`, `deploy`).

OpenAPI / Swagger UI Estándar para describir APIs REST y herramienta web (Swagger UI) que muestra esa descripción de forma navegable. FCTNow las expone en `/api/openapi` y `/api/swagger-ui.html`.

PostgreSQL Sistema gestor de bases de datos relacional de código abierto, robusto y orientado a transacciones. Es el SGBD elegido para FCTNow.

Render Plataforma cloud que aloja servicios web (entre otros) y soporta despliegue desde un repositorio Git o desde imágenes Docker. FCTNow la usa para el backend Spring Boot.

Resource Server (OAuth2) Patrón de Spring Security en el que la aplicación valida tokens JWT emitidos previamente por un *issuer* conocido, sin mantener estado de sesión. FCTNow opera como Resource Server consumiendo sus propios JWT.

Rol Etiqueta funcional asociada a un usuario que determina qué operaciones puede realizar. En FCTNow los roles son `ALUMNO`, `EMPRESA`, `TUTOR_CENTRO`, `COORDINADOR` y `ADMIN`.

Spring Boot Framework Java que simplifica la creación de aplicaciones empresariales sobre el ecosistema Spring, con configuración por convención y autoconfiguración.

Spring Data JPA Módulo de Spring que reduce el boilerplate de acceso a datos sobre JPA, generando implementaciones de repositorio a partir de interfaces.

Spring Security Módulo de Spring que aporta autenticación y autorización a la aplicación, con soporte para múltiples mecanismos (formulario, HTTP Basic, OAuth2, JWT…).

SSR (Server-Side Rendering) Técnica de renderizado de aplicaciones SPA en el servidor antes de enviarlas al navegador, mejorando el tiempo a primer pintado y el SEO. Angular Universal es la implementación oficial en Angular.

Stale-While-Revalidate Estrategia de caché en la que se sirve inmediatamente el dato cacheado (aunque esté potencialmente obsoleto) y, en paralelo, se solicita la versión fresca al servidor. FCTNow la aplica en frontend para reducir la latencia percibida.

Supabase Plataforma que ofrece PostgreSQL gestionado (entre otros servicios). FCTNow la usa exclusivamente como base de datos en producción.

Vercel Plataforma cloud especializada en aplicaciones frontend con despliegue continuo desde Git y *edge functions*. FCTNow la usa para servir el frontend Angular.

6. Bibliografía

[1] Spring. *Spring Boot Reference Documentation* (v3.5.x). https://docs.spring.io/spring-boot/docs/current/reference/html/ (Consulta: 2025).

[2] Spring. *Spring Security Reference*. https://docs.spring.io/spring-security/reference/ (Consulta: 2025).

[3] Spring. *Spring Data JPA — Reference Documentation*. https://docs.spring.io/spring-data/jpa/reference/ (Consulta: 2025).

[4] Angular Team. *Angular Documentation* (v20). https://angular.dev/ (Consulta: 2025).

[5] Angular Team. *Angular SSR Guide*. https://angular.dev/guide/ssr (Consulta: 2025).

[6] PostgreSQL Global Development Group. *PostgreSQL Documentation*. https://www.postgresql.o (Consulta: 2025).

[7] Flyway. *Flyway Documentation*. https://documentation.red-gate.com/fd (Consulta: 2025).

[8] RFC 7519. *JSON Web Token (JWT)*. IETF, 2015. https://datatracker.ietf.org/doc/html/rfc7519 (Consulta: 2025).

[9] RFC 6749. *The OAuth 2.0 Authorization Framework*. IETF, 2012. https://datatracker.ietf.org/doc (Consulta: 2025).

[10] OWASP Foundation. *OWASP Top Ten Web Application Security Risks*. https://owasp.org/www-project-top-ten/ (Consulta: 2025).

[11] OWASP Foundation. *Password Storage Cheat Sheet*. https://cheatsheetseries.owasp.org/chea (Consulta: 2025).

[12] Adzuna. *Adzuna Developer Documentation*. https://developer.adzuna.com/ (Consulta: 2025).

[13] Render. *Web Services Documentation*. https://render.com/docs/web-services (Consulta: 2025).

[14] Vercel. *Vercel Documentation*. https://vercel.com/docs (Consulta: 2025).

[15] Supabase. *Supabase Documentation — PostgreSQL*. https://supabase.com/docs/guides/datal (Consulta: 2025).

[16] Walls, C. *Spring in Action* (6.ª edición). Manning Publications, 2022.

[17] Bauer, C., King, G., Gregory, G. *Java Persistence with Hibernate* (2.ª edición). Manning Publications, 2015.

[18] Baeldung. *Spring Security Tutorials*. https://www.baeldung.com/security-spring (Consulta: 2025).

[19] Baeldung. *Spring Data JPA Tutorials*. https://www.baeldung.com/spring-data-jpa (Consulta: 2025).

[20] Conventional Commits. *Conventional Commits 1.0.0 Specification*. https://www.conventionalc (Consulta: 2025).

[21] Provos, N., Mazières, D. *A Future-Adaptable Password Scheme*. USENIX Annual Technical Conference, 1999. (Base teórica de BCrypt.)

[22] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures* (Tesis doctoral). University of California, Irvine, 2000. (Origen de REST.)

7. Anexos

7.1 Manual de instalación local

Requisitos previos:

- Java 21.
- Maven 3.6+.
- Node.js 20+ y npm.
- Docker con `docker-compose` clásico o el plugin `docker compose`.

Pasos:

1. Clonar el repositorio.

```
git clone https://github.com/DavidBarbosaOlayo/FCTNow.git
cd FCTNow
```

2. Arrancar PostgreSQL local.

```
docker compose up -d postgres
```

3. Arrancar el backend.

```
cd backend
mvn spring-boot:run
```

4. En otra terminal, arrancar el frontend.

```
cd frontend
npm install
npm start
```

5. Abrir el navegador en `http://localhost:4200`.

Variables de entorno relevantes para el backend en local:

```
FCTNOW_DB_URL=jdbc:postgresql://localhost:15432/fctnow
FCTNOW_DB_USERNAME=fctnow
FCTNOW_DB_PASSWORD=fctnow
FCTNOW_JWT_SECRET=change-this-local-secret-with-at-least-32-bytes
ADZUNA_APP_ID=...
ADZUNA_APP_KEY=...
FCTNOW_MAIL_USERNAME=...
FCTNOW_MAIL_PASSWORD=...
```

7.2 Manual de usuario resumido

1. **Alumno.** Tras recibir el correo de bienvenida del tutor, inicia sesión en `/login`. Completa su perfil en `/perfil` (preferencias y CV). Consulta el catálogo en `/practicas`, se postula a las ofertas internas que le interesen y registra manualmente las externas que haya tramitado fuera de FCTNow. Sigue el estado de sus candidaturas en `/alumno/solicitudes` y, una vez asignado, ve el progreso de horas y los datos económicos en la home.
2. **Empresa.** Inicia sesión y accede a `/empresa`. Gestiona sus ofertas en `/empresa/ofertas`. Recibe candidaturas en `/empresa/solicitudes` y las acepta o rechaza desde allí.
3. **Tutor.** Inicia sesión y accede a `/tutor`. Da de alta a sus alumnos individualmente o por importación. Hace seguimiento del estado de cada uno.
4. **Coordinador.** Accede a `/asignaciones`, revisa las candidaturas aceptadas pendientes de asignación (internas y externas) y crea la asignación oficial con horas, fechas y datos de retribución.
5. **Administrador.** Accede a `/admin` para una visión agregada del sistema.

7.3 Diagrama del modelo de datos (resumen textual)

- `app_users` (id, email, password_hash, display_name, enabled, centro_email, empresa_id?, foto_*)
- `user_roles` (user_id, role)
- `empresas` (id, nombre, tipo_identificador_fiscal, identificador_fiscal, sector, …)
- `ofertas_fct` (id, empresa_id → empresas, titulo, familia_profesional, modalidad, fechas, plazas, estado, …)
- `solicitudes_fct` (id, alumno_id → app_users, oferta_id → ofertas_fct, estado, UK(alumno, oferta))
- `asignaciones_fct` (id, solicitud_id UK → solicitudes_fct, alumno_id, oferta_id, empresa_id, estado, horas_totales, fecha_inicio, horas_diarias_estimadas, remunerada, importe_mensual, observaciones_retribucion, observaciones, fecha_asignacion)
- `solicitudes_externas` (id, alumno_id, fuente, id_externo, titulo, empresa_nombre, url_aplicacion, estado, …, UK(alumno, fuente, id_externo))
- `asignaciones_fct_externas` (id, solicitud_externa_id UK, alumno_id, estado, horas_totales, fecha_inicio, …)
- `alumno_preferencias` (id, alumno_id UK → app_users, familia, ciclo, localidad, modalidad, fecha_disponibilidad, observaciones, cv_, *foto*_)
- `conversaciones` (id, titulo?, participante_a_id, participante_b_id, CHECK distintos)
- `mensajes` (id, conversacion_id → conversaciones, remitente_id → app_users, contenido, created_at)
- `notificaciones` (id, destinatario_id → app_users, alumno_id?, recomendada_por?, tipo, titulo, mensaje, action_url, action_label, oferta_id?, oferta_externa_*, leida, created_at)
- `backend_metadata` (tabla técnica de bootstrap)

7.4 Enlaces del proyecto

- Repositorio público: `https://github.com/DavidBarbosaOlayo/FCTNow`
- Frontend en producción (Vercel): URL del proyecto desplegado.
- Backend en producción (Render): `https://fctnow-backend.onrender.com`.
- Documentación OpenAPI: `https://fctnow-backend.onrender.com/api/openapi`.
- Swagger UI: `https://fctnow-backend.onrender.com/api/swagger-ui.html`.

7.5 Pruebas automatizadas: inventario

Backend (15 clases de test, ejecutables con `mvn test`)

- `HealthControllerTest`
- `AuthControllerTest`
- `EmpresaControllerTest`
- `EmpresaOfertaControllerTest`
- `OfertaFctControllerTest`
- `EmpresaSolicitudControllerTest`
- `SolicitudFctControllerTest`
- `AsignacionFctControllerTest`
- `AsignacionFctExternaControllerTest`
- `SolicitudExternaControllerTest`
- `AdzunaServiceTest`
- `OfertaExternaControllerTest`
- `MensajeControllerTest`
- `AlumnoPreferenciasControllerTest`
- `TutorAlumnoControllerTest`

Frontend (más de 30 `.spec.ts`, ejecutables con `npm test -- --watch=false --browsers=ChromeHeadless`)

- Tests de servicios HTTP: `auth.service.spec`, `ofertas.service.spec`, `ofertas-externas.service.spec`, `empresa-ofertas.service.spec`, `empresa-solicitudes.service.spec`, `empresa-perfil.service.spec`, `asignaciones.service.spec`, `mensajes.service.spec`, `home.service.spec`, `solicitudes.service.spec`, `preferencias.service.spec`.
- Tests de componentes página: `login.spec`, `home.spec`, `practicas.spec`, `oferta-detail.spec`, `mis-solicitudes.spec`, `mis-ofertas-empresa.spec`, `mis-solicitudes-empresa.spec`, `oferta-empresa-form.spec`, `empresa-perfil.spec`, `mensajes.spec`, `asignaciones.spec`, `tutor.spec`, `perfil.spec`, `preferencias.spec`.
- Tests transversales: `app.routes.spec`, `app.spec`, `route-pages.spec`, `auth-session-storage.spec`, `role.guard.spec`, `app-navigation.spec`.