Alex García Jardí

SISTEMA DE GESTIÓN Y SEGUIMIENTO DE PRÁCTICAS CLÍNICAS EN ENFERMERÍA

TRABAJO DE FIN DE GRADO

Dirigido por Pere Millán Marco

Grado en Ingeniería Informática

UNIVERSITAT ROVIRA I VIRGILI

Tarragona
2025

Resum.

En l'àmbit de les pràctiques clíniques en infermeria, tant els tutors com els coordinadors sovint s'enfronten a la manca d'eines per accedir i gestionar la documentació, les activitats i la informació necessàries per fer un bon seguiment dels estudiants d'infermeria. En aquest projecte es proposa una solució mitjançant dues aplicacions web interrelacionades que resolen aquestes mancances de manera eficient i segura.

La primera aplicació és dirigida als tutors clínics i organitza el contingut en cinc grans blocs:

- Documentació institucional, com normatives i protocols.
- Activitats que l'estudiant ha de completar durant les pràctiques.
- Documentació de suport, com el calendari acadèmic i les competències a assolir.
- Avaluació de l'estudiant.
- Registre de les hores realitzades per l'estudiant.

A més, l'aplicació mostra altres dades rellevants, com les persones que coordinen les assignatures, entre altres.

La segona aplicació està dissenyada per als coordinadors de pràctiques. Compta amb una interfície pensada per facilitar la gestió de tots els recursos accessibles des de la plataforma del tutor. Entre les seves funcionalitats s'inclouen:

- Pujar i organitzar documents.
- Gestionar els campus.
- Gestionar les assignatures.
- Generar i administrar codis de registre (explicats més endavant).
- Consultar el llistat de tutors registrats.
- Visualitzar els registres d'hores enviats pels tutors i exportar-los en format Excel.
- Gestionar tant els coordinadors com els administradors del sistema.

Ambdues aplicacions han estat desenvolupades amb Vue.js, integren serveis de Firebase (Firestore, Authentication i Storage) i es troben actualment en producció.

Resumen.

En el ámbito de las prácticas clínicas en enfermería, tanto tutores como coordinadores suelen enfrentarse a la falta de herramientas para acceder y gestionar la documentación, actividades e información necesarias para hacer un buen seguimiento de los estudiantes de enfermería. En este proyecto se propone una solución mediante dos aplicaciones web interrelacionadas que resuelven estas carencias de forma eficiente y segura.

La primera aplicación está dirigida a los tutores clínicos y organiza el contenido en cinco grandes bloques:

- Documentación institucional, como normativas y protocolos.
- Actividades que el estudiante debe completar durante sus prácticas.
- Documentación de apoyo, como el calendario académico y las competencias a alcanzar.
- Evaluación del estudiante.
- Registro de las horas realizadas por el estudiante.

Además, la aplicación muestra otros datos relevantes, como las personas que coordinan las asignaturas, entre otros.

La segunda aplicación está diseñada para los coordinadores de prácticas. Cuenta con una interfaz pensada para facilitar la gestión de todos los recursos accesibles desde la plataforma del tutor. Entre sus funcionalidades se incluyen:

- Subir y organizar documentos.
- Gestionar los campus.
- Gestionar las asignaturas.
- Generar y administrar códigos de registro (explicados más adelante).
- Consultar el listado de tutores registrados.
- Visualizar los registros de horas enviados por los tutores y exportarlos en formato Excel.
- Gestionar tanto los coordinadores como los administradores del sistema.

Ambas aplicaciones han sido desarrolladas con Vue.js, integran servicios de Firebase (Firestore, Authentication y Storage) y se encuentran actualmente en producción.

Abstract.

In the field of clinical nursing practice, both tutors and coordinators often face a lack of tools to access and manage the documentation, activities, and information needed to effectively monitor nursing students. This project proposes a solution through two interconnected web applications that address these gaps efficiently and securely.

The first application is aimed at clinical tutors and organizes its content into five main sections:

- Institutional documentation, such as regulations and protocols.
- Activities that the student must complete during their nursing practice.
- Supporting documentation, such as the academic calendar and the competencies to be archived.
- Student evaluation.
- Recording of the hours completed by the student.

In addition, the application displays other relevant information, such as the course coordinators, among others.

The second application is designed for nursing practice coordinators. It features an interface aimed at facilitating the management of all resources accessible from the tutor's platform. Its functionalities include:

- Uploading and organizing documents.
- Managing campuses.
- Managing subjects.
- Generating and administering registration codes (explained later).
- Viewing the list of registered tutors.
- Viewing and exporting timesheet records submitted by tutors in Excel format.
- Managing both coordinators and system administrators.

Both applications have been developed using Vue.js, integrate Firebase services (Firestore, Authentication, and Storage), and are currently in production.

Índice

1 INTRODUCCIÓN 4

2 PALABRAS CLAVE DEL PROYECTO 6

3 OBJETIVOS DEL TFG 7

4 PLANIFICACIÓN 8
4.1 DIAGRAMA DE GANTT 8
4.2 FASES DEL PROYECTO 9

5 REQUISITOS DEL PROYECTO 10
5.1 REQUISITOS DE LA APLICACIÓN DE GESTIÓN 10
5.1.1 Requisitos funcionales 10
5.1.2 Requisitos no funcionales 31
5.2 REQUISITOS DE LA APLICACIÓN DE TUTORES 33
5.2.1 Requisitos funcionales 33
5.2.2 Requisitos no funcionales 48

6 DISEÑO 50
6.1 ARQUITECTURA DE LAS APLICACIONES 50
6.2 DISEÑO DE LAS INTERFACES 50
6.2.1 Aplicación de gestión 51
6.2.2 Aplicación de tutores 57
6.3 DISEÑO DE LA ARQUITECTURA DE FIREBASE 63
6.3.1 Diseño de la base de datos en Firestore 63
6.3.2 Diseño de Firebase Storage 69
6.3.3 Firebase Authentication 71

7 IMPLEMENTACIÓN 72
7.1 REUTILIZACIÓN DE COMPONENTES 72
7.2 CALIDAD DEL CÓDIGO Y CONTROL DE VERSIONES 72
7.3 ORGANIZACIÓN DEL CÓDIGO 73
7.4 CONEXIÓN DE VUE CON FIREBASE 74
7.5 OPERACIONES CON FIRESTORE 74
7.6 OPERACIONES CON FIREBASE STORAGE 76
7.7 OPERACIONES CON FIREBASE AUTHENTICATION 77
7.7.1 Gestión de las sesiones 77
7.7.2 Protección de las rutas 78
7.8 INCORPORACIÓN DE IONIC EN VUE 78

8 EVALUACIÓN 79
8.1 JUEGOS DE PRUEBA EN LA APLICACIÓN DE GESTIÓN 79
8.2 JUEGOS DE PRUEBA EN LA APLICACIÓN DE TUTORES CLÍNICOS 83

9 EVALUACIÓN DE COSTES 86
9.1 DETALLE DE RECURSOS DEL PROYECTO 86
9.2 COSTE DE SOFTWARE / HARDWARE 86
9.3 PROVISIONES 87
9.4 RESUMEN DE COSTES 87

10 LEGISLACIÓN Y PROTECCIÓN DE DATOS 88

11 IMPLICACIONES ÉTICAS, DE IGUALDAD Y MEDIOAMBIENTALES 89

12 VALORACIÓN 90

13 BIBLIOGRAFÍA 91

14 ANEXOS 92

Índice de tablas

TABLA 1. APLICACIÓN DE GESTIÓN - JUEGOS DE PRUEBA 82
TABLA 2. APLICACIÓN DE TUTORES - JUEGOS DE PRUEBA 85
TABLA 3. RECURSOS DEL PROYECTO 86
TABLA 4. COSTE DE SOFTWARE / HARDWARE 86
TABLA 5. PROVISIONES 87
TABLA 6. RESUMEN DE COSTES 87

Índice de figuras

Figura 1. Diagrama de Gantt (primera parte) 8
Figura 2. Diagrama de Gantt (segunda parte) 8
Figura 3. Aplicación de gestión - Diagrama de casos de uso 12
Figura 4. Aplicación de Tutores - Diagrama de casos de uso 34
Figura 5. Aplicación gestión – Inicio de sesión 51
Figura 6. Aplicación gestión – Subida de documentos 51
Figura 7. Aplicación gestión – Listado de documentos 52
Figura 8. Aplicación gestión – Gestión de asignaturas 52
Figura 9. Aplicación gestión – Gestionar una asignatura 53
Figura 10. Aplicación gestión – Gestionar campus 53
Figura 11. Aplicación gestión – Gestionar coordinadores 54
Figura 12. Aplicación gestión – Gestionar administradores 54
Figura 13. Aplicación gestión – Gestionar tutores 55
Figura 14. Aplicación gestión – Consultar registro de horas 55
Figura 15. Aplicación gestión – Gestionar códigos de registro 56
Figura 16. Aplicación Tutores – Inicio de sesión 57
Figura 17. Aplicación Tutores – Registro 57
Figura 18. Aplicación Tutores – Listado de campus 58
Figura 19. Aplicación Tutores – Listado de asignaturas 58
Figura 20. Aplicación Tutores – Contenido de una asignatura 59
Figura 21. Aplicación Tutores – FAQ 59
Figura 22. Aplicación Tutores – Avisos importantes 60
Figura 23. Aplicación Tutores – Documentos de soporte 60
Figura 24. Aplicación Tutores – Actividades del estudiante 61
Figura 25. Aplicación Tutores – Registro de horas 61
Figura 26. Aplicación Tutores – Visualización de los Protocolos / Normativas 62
Figura 27. Aplicación Tutores – Perfil del usuario 62

1 Introducción

En el ámbito de las Ciencias de la Salud, especialmente en el Grado en Enfermería, las prácticas clínicas son una parte muy importante en el desarrollo profesional de los estudiantes. Gracias a estas experiencias, los estudiantes pueden poner en práctica lo que han aprendido en clase y adquirir las habilidades necesarias para convertirse en profesionales de Enfermería.

Sin embargo, este proceso de aprendizaje implica no solo a los estudiantes, sino también a tutores clínicos y coordinadores académicos, quienes deben realizar un seguimiento personalizado del progreso de cada alumno. Para ello, es necesaria una gestión continua de documentos oficiales, cronogramas, actividades, evaluaciones, control de horarios y, sobre todo, una comunicación fluida entre todos los implicados.

Cada año, alrededor de 1.000 estudiantes del Grado de Enfermería de la URV se desplazan a distintas instituciones sanitarias para realizar sus prácticas. Estos alumnos están supervisados por los tutores clínicos. No obstante, estos tutores carecen de acceso a la información alojada en el Moodle de la Universidad Rovira i Virgili, por lo que desconocen las tareas y prácticas que deben supervisar en el centro de trabajo.

Es por este motivo que, hasta hace poco, gran parte de esta gestión se llevaba a cabo con recursos improvisados: correos electrónicos, carpetas compartidas, formularios en papel e incluso a través de indicaciones verbales. Aunque esto es funcional a corto plazo, este método dispersa la información, dificulta el seguimiento individual y aumenta el riesgo de errores con consecuencias importantes.

Este Trabajo de Fin de Grado nace precisamente como respuesta a este problema. La oportunidad me vino a través de una oferta publicada en un canal oficial de la universidad, en la que se buscaba a un estudiante del Grado en Ingeniería Informática para llevar a cabo el desarrollo Frontend de una plataforma orientada a la gestión de prácticas clínicas. Una vez superado el proceso de selección, conseguí una beca de desarrollo vinculada al proyecto.

Desde el momento en que empecé, tuve muy claro que era una propuesta alineada con mis competencias y motivaciones, pues me ofrecía la posibilidad de aplicar mis conocimientos en diseño de interfaces, trabajar con tecnologías modernas de Frontend y participar en el desarrollo de una herramienta real que sería utilizada por personal docente y sanitario.

Cuando empecé, el proyecto contaba con una base inicial desarrollada por otro estudiante, pero el código proporcionado requería una reestructuración completa. En realidad, más allá del código, la experiencia de usuario no era la adecuada, y el diseño carecía de claridad y coherencia. Por eso, asumí el reto de repensarlo todo, desde la arquitectura interna hasta la organización de los menús, el flujo de navegación y la integración con la base de datos.

Durante el proceso de desarrollo, estuve haciendo reuniones periódicas con una de las coordinadoras académicas, Rosa Raventós Torner, la cual ha sido la principal promotora en el desarrollo de este proyecto. Estas sesiones semanales se hacían con el fin de garantizar que lo que se estaba construyendo respondía a las necesidades reales del problema planteado.

El sistema que se ha implementado está compuesto por dos aplicaciones web. La primera está destinada a los tutores clínicos, esta permite acceder a toda la documentación necesaria, registrar las horas de prácticas, consultar las actividades del estudiante y acceder a formularios de evaluación.

En esta primera aplicación, debido a que en el entorno clínico un estudiante puede tener múltiples tutores supervisándolo, y cada tutor puede estar a cargo del seguimiento de varios estudiantes de distintas asignaturas, se dispone de cuentas de usuario personalizadas para los tutores y un mecanismo de protección adicional mediante contraseñas por asignatura, las cuales tienen el fin de garantizar un acceso controlado y restringido.

La segunda aplicación está diseñada para coordinadores académicos, permite gestionar todo el contenido de la primera web desde una interfaz sencilla y estructurada. Algunas de las cosas que permite hacer son las siguientes: gestionar documentos, campus, asignaturas, contraseñas, generar códigos de registro[1], consultar registros de horas, gestionar coordinadores, administradores[2], etc.

Estas dos aplicaciones han sido desarrolladas con el Framework Frontend Vue.js y se ha utilizado Firebase como Backend.

Además, para el diseño de las interfaces se utilizó Figma[3]. Esto me sirvió para crear pequeños conceptos de las páginas web de forma que pudiera diseñar las interfaces de una forma rápida, con el objetivo de asegurarme que el diseño cumplía las expectativas antes de empezar a desarrollarlo.

El sistema ya está desplegado en producción y se encuentra funcionando en un entorno real, de forma que los usuarios han podido validar su estabilidad y confirmar su utilidad antes de su uso real.

---

1 Los códigos de registro sirven para dar una capa adicional de seguridad. Debido a la gran cantidad de tutores clínicos a gestionar, se ha creado un mecanismo para que los tutores clínicos puedan registrarse ellos mismos rellenando sus datos. No obstante, para evitar registros de usuarios no autorizados, se crearon los códigos de registro para que solo aquel individuo que durante su registro proporcione un código válido, pueda registrar una cuenta en el sistema.

2 De acuerdo con las políticas de la aplicación, no todos los coordinadores deben tener acceso al programa de gestión. Por lo tanto, para restringir más el acceso, solo aquellos que figuren como administradores podrán gestionarla.

3 Figma es una aplicación ampliamente utilizada por diseñadores gráficos para realizar el diseño de las interfaces y ofrecer una buena experiencia para el usuario (UX).

2 Palabras clave del proyecto

- Informática aplicada (77011001)
- Bases de datos (77011003)
- Seguridad Informática (77011005)
- Ingeniería del software (77011007)
- Programa (77011009)
- Lenguajes de programación (77011012)
- Programación (77011014)
- Sistemas de información (77011018)
- Programación web (99999907)
- Seguridad y privacidad (99999912)
- Cloud computing (99999925)
- Redes (99999937)

3 Objetivos del TFG

Como ya se mencionó, el tema de este Trabajo de Fin de Grado no fue una propuesta personal. Surgió a partir de una oferta externa publicada por la universidad, a la que decidí presentarme porque me pareció una buena oportunidad para aplicar los conocimientos adquiridos durante el grado y para aprender tecnologías nuevas que no habíamos trabajado en el grado.

Mi principal objetivo en este proyecto fue aprender Vue.js. Esto es porque, aunque he trabajado con tecnologías Frontend parecidas como React, trabajar con más Frameworks de Frontend mejora mi visión como futuro Ingeniero a la hora de escoger entre las diferentes tecnologías.

Además, como ya se dijo, se utilizó Firebase como Backend. Con esta plataforma ya estaba familiarizado, pues ya había realizado algún proyecto en los que utilizaba varios de sus servicios.

No obstante, sí que hubo algo en lo que profundicé y aprendí mucho en el entorno de Firebase: la gestión de las reglas. Esta parte era muy importante porque como tal no hay un Backend (del estilo REST API, por ejemplo) donde poder gestionar toda la seguridad, por lo que la seguridad de los datos dependía directamente de una correcta definición de estas reglas.

Otro reto era desplegar las dos aplicaciones en producción. La aplicación web para tutores clínicos se configuró con un dominio de la URV fácil de recordar, mientras que la web de gestión se configuró con un dominio que proporciona Firebase Hosting de forma gratuita.

A nivel personal y formativo, ha sido una experiencia que me ha hecho conseguir el objetivo de crecer como desarrollador, porque me ha ayudado a adaptarme a tecnologías nuevas y entender cómo se trabaja cuando hay usuarios reales esperando que lo que haces funcione de verdad.

4 Planificación

Para organizar adecuadamente el proyecto, se ha definido una planificación que abarca desde el análisis inicial del proyecto hasta su entrega final. Este proyecto empezó el 1 de noviembre del 2024 (inicio de la beca).

Como el desarrollo del proyecto se ha realizado con una dedicación parcial de 10 horas semanales, **normalmente** repartidas entre sábado y domingo, la planificación se ha adaptado a este ritmo de trabajo. Cada sesión de trabajo se ha aprovechado para avanzar en las distintas fases del proyecto, desde el análisis y diseño inicial, pasando por el desarrollo de las dos aplicaciones Frontend, las pruebas funcionales y la redacción de la presente memoria.

Esta planificación también refleja las reuniones realizadas durante el desarrollo del proyecto, las cuales figuran como validaciones y pruebas.

A continuación, se representará gráficamente esta dedicación con el Diagrama de Gantt seguido de las diferentes fases que se realizaron.

4.1 Diagrama de Gantt

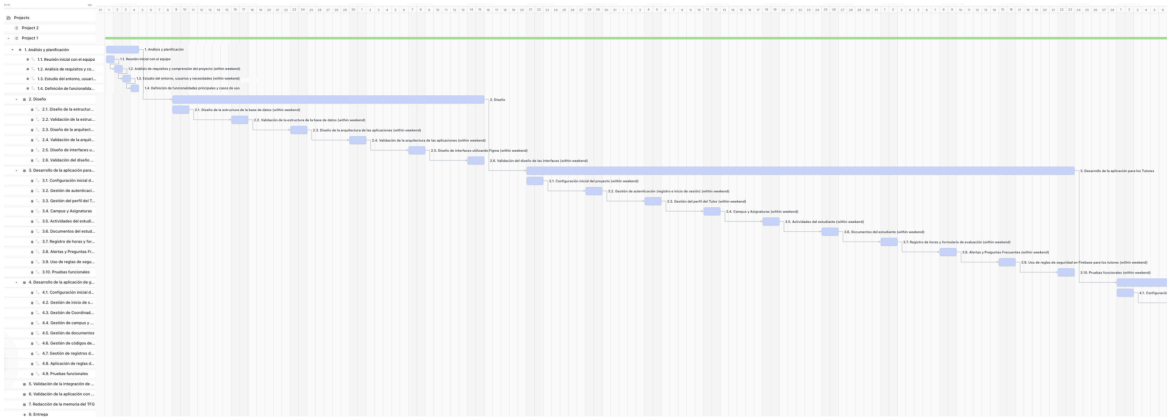


Figura 1. Diagrama de Gantt (primera parte)

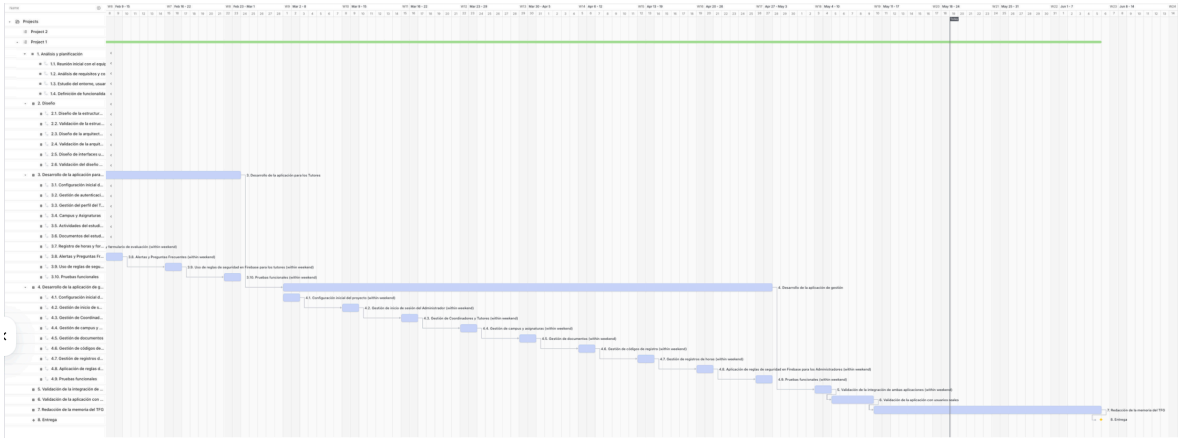


Figura 2. Diagrama de Gantt (segunda parte)

4.2 Fases del proyecto

1. Análisis y planificación
   - Reunión inicial con el equipo académico
   - Análisis de requisitos y comprensión del proyecto
   - Estudio del entorno, usuarios y necesidades
   - Definición de funcionalidades principales y casos de uso
2. Diseño
   - Diseño de la estructura de la base de datos
   - Validación de la estructura de la base de datos
   - Diseño de la arquitectura de las aplicaciones
   - Validación de la arquitectura de las aplicaciones
   - Diseño de las interfaces con Figma
   - Validación del diseño de las interfaces
3. Desarrollo de la aplicación de tutores
   - Configuración inicial del proyecto
   - Gestión de autenticación (registro e inicio de sesión)
   - Gestión del perfil del tutor
   - Campus y asignaturas
   - Actividades del estudiante
   - Documentos del estudiante
   - Registro de horas y formulario de evaluación
   - Alertas y preguntas frecuentes
   - Uso de reglas de seguridad en Firebase para los tutores
   - Pruebas funcionales
4. Desarrollo de la aplicación de gestión
   - Configuración inicial del proyecto
   - Gestión de inicio de sesión del administrador
   - Gestión de coordinadores y tutores
   - Gestión de campus y asignaturas
   - Gestión de documentos
   - Gestión de códigos de registro
   - Gestión de registros de horas
   - Aplicación de reglas de seguridad en Firebase para los administradores
   - Pruebas funcionales
5. Validación de la integración de ambas aplicaciones
6. Validación de la aplicación con usuarios reales
7. Redacción de la memoria del TFG
8. Entrega (hito)

5 Requisitos del proyecto

Antes de desarrollar las aplicaciones, fue necesario definir con claridad qué funcionalidades debían ofrecer y qué características debían cumplir. A continuación, se detallan los requisitos del sistema, tanto funcionales como no funcionales, que han servido como base para orientar el desarrollo del proyecto. Como se han implementado dos proyectos Frontend, este apartado se dividirá en dos bloques.

5.1 Requisitos de la aplicación de gestión

A continuación, se procederá a indicar los requisitos, tanto funcionales como no funcionales, de la aplicación que utilizarán los coordinadores académicos para gestionar la aplicación de los tutores académicos.

5.1.1 Requisitos funcionales

En el siguiente apartado se incluye el glosario de términos, las reglas de negocio, el diagrama de casos de uso y la especificación textual detallada de cada uno de ellos.

5.1.1.1 Glosario de términos

- **Tutor:** Profesional de la salud, externo a la URV, que hace el seguimiento del estudiante durante las prácticas.
- **Coordinador:** Persona de la URV responsable de la gestión académica de las prácticas.
- **Administrador:** Coordinador/a con permisos para gestionar el sistema.
- **Código de registro:** Clave única que permite a los tutores registrarse de forma segura.
- **Asignatura:** Módulo académico de las prácticas.
- **Campus:** Centro o sede donde se imparten las asignaturas de prácticas.
- **Documento:** Archivo oficial, como normativa, protocolo, etc.
- **Registro de horas:** Registro que refleja las horas que el estudiante ha realizado en el centro de trabajo (los introduce el tutor).

5.1.1.2 Reglas del negocio

1. El acceso a la aplicación de gestión está restringido a usuarios que son **Administradores**.
2. No puede haber dos **Administradores** con el mismo correo electrónico.
3. No se puede subir un documento sin especificar su categoría, campus y asignatura al que pertenece.
4. No se puede subir un documento de 0 bytes.
5. Los documentos se ordenan alfabéticamente.
6. No se puede crear/modificar una asignatura sin especificar su nombre.
7. No se puede crear/modificar una asignatura sin especificar el curso al que pertenece.
8. No se puede crear una asignatura sin especificar el campus al que pertenece.
9. Los nombres de las asignaturas deben ser únicos dentro de un mismo campus.
10. Al crear una contraseña para una asignatura, la fecha de expiración de esta debe estar en el futuro.
11. Las contraseñas de las asignaturas se deben confirmar para poder ser creadas.
12. Al crear una contraseña nueva para una asignatura, si ya existía una previamente, la contraseña previa quedará revocada y se creará la nueva. Los usuarios que iniciaron sesión con la contraseña previa (cuando era válida) seguirán teniendo acceso a la asignatura hasta su fecha de expiración. En cambio, los usuarios que no tengan acceso a la asignatura necesitarán usar la contraseña nueva.
13. La contraseña nueva no puede haberse usado en el pasado en la misma asignatura.
14. No se puede crear un campus sin especificar su nombre.
15. Los nombres de los campus deben ser únicos.
16. No se puede borrar un campus que contiene asignaturas.
17. No se puede crear un coordinador sin especificar su nombre completo.
18. No se puede crear un coordinador sin especificar su correo electrónico.
19. No se pueden crear **Administradores** nuevos, la gestión la debe hacer un súper administrador del sistema.
20. No se puede eliminar la cuenta de un **Administrador** si esa cuenta es la del usuario actual.
21. No se puede desactivar una cuenta de un **Administrador** si esa cuenta es la del usuario actual.
22. No se puede desactivar una cuenta de un **Administrador** si solo hay una única cuenta activa.
23. No se puede añadir un curso nuevo si los años del curso no tienen un formato válido (YYYY-YYYY).
24. No se puede exportar el registro de horas si el curso seleccionado no tiene datos.
25. No se puede crear un nuevo código de registro sin especificar el código de registro.
26. El código de registro debe confirmarse para poder crearlo.
27. No se puede crear un nuevo código de registro donde su fecha de expiración no sea en el futuro.
28. No se puede almacenar el código de registro sin aplicar masking[4].
29. Un correo electrónico debe tener siempre un formato válido.

4 Los códigos de registro son altamente confidenciales. Por lo tanto, se acordó que los Administradores deberán anotarlos y el sistema simplemente mostrará las dos primeras y las dos últimas letras del código, mostrando asteriscos en el medio.

5.1.1.3 Diagrama de casos de uso

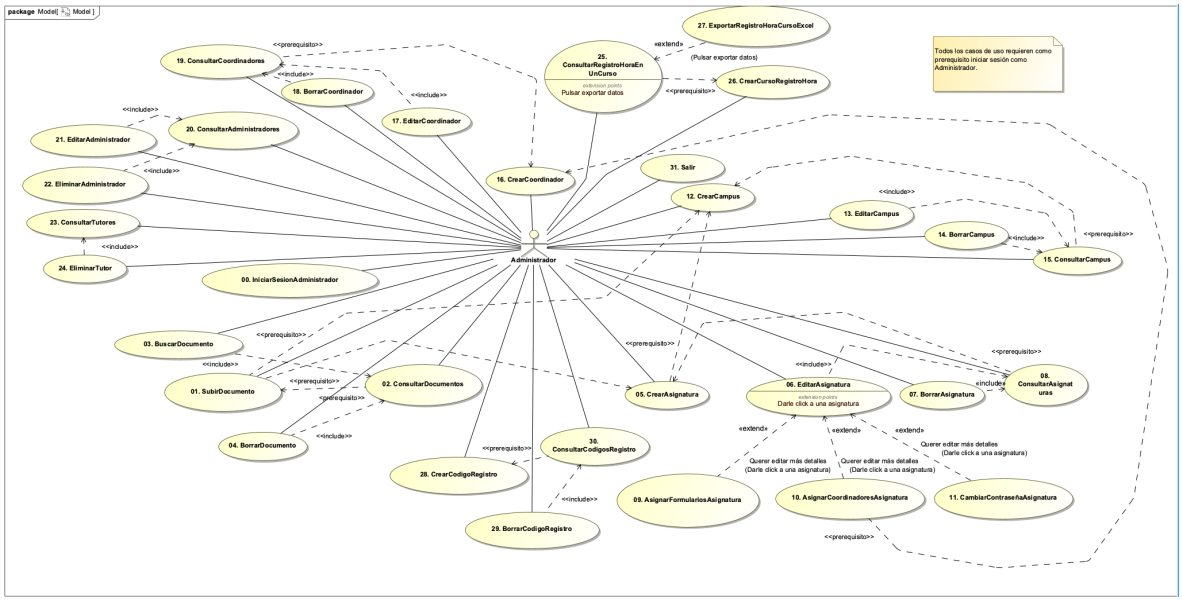


Figura 3. Aplicación de gestión - Diagrama de casos de uso

5.1.1.4 Especificación textual de los casos de uso

En el siguiente apartado, se procederá a especificar textualmente los casos de uso referentes a la Figura 3 del apartado anterior.

Caso de uso 00. IniciarSesionAdministrador

Resumen de la funcionalidad: Autenticar al usuario como **Administrador** para permitir el acceso a la aplicación de gestión.

- Parámetros de entrada: Correo electrónico, contraseña.
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: Estar registrado en el sistema (solo un súper administrador puede registrar a los administradores, por lo que no hay un caso de uso para registrarlos).
- Postcondición: El **Administrador** ha iniciado sesión correctamente.

Proceso normal principal:

1. El sistema solicita el correo electrónico y la contraseña.
2. El **Administrador** introduce las credenciales.
3. El sistema verifica que el usuario es **Administrador** y que la cuenta está activa.
4. El sistema permite el acceso a la aplicación de gestión.

Alternativas de proceso y excepciones:

3a. Las credenciales no son válidas:
3a1. El sistema muestra un mensaje de error y no permite el acceso.

Caso de uso 01. SubirDocumento

Resumen de la funcionalidad: Subir documentos relacionados con las asignaturas o normativas.

- Parámetros de entrada: Archivo, categoría, campus, asignatura.
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe existir al menos un campus y una asignatura.
- Postcondición: Se ha subido el documento en Firebase Storage.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Subir documentos".
2. El **Administrador** selecciona un archivo y especifica: categoría, campus y asignatura.
3. El sistema muestra un diálogo de confirmación.
4. El **Administrador** confirma la acción.
5. El sistema valida que el documento no esté vacío y que no falten campos.
6. Se guarda el documento en Firebase Storage.

Alternativas de proceso y excepciones:

4a. El **Administrador** no confirma la acción:
    4a1. El sistema no realiza ningún cambio.
5a. El documento está vacío o faltan campos:
    5a1. El sistema notifica que faltan campos.

Caso de uso 02. ConsultarDocumentos

Resumen de la funcionalidad: Consultar todos los documentos almacenados.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Listado de documentos disponibles.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse subido al menos un documento.
- Postcondición: Se ha podido consultar uno o varios documentos disponibles.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Documentos".
2. El sistema recupera y muestra los documentos disponibles.
3. El **Administrador** selecciona un documento.
4. El sistema permite la visualización o descarga del archivo.

Alternativas de proceso y excepciones:

2a. No hay documentos disponibles:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 03. BuscarDocumento

Resumen de la funcionalidad: Buscar un documento específico introduciendo su nombre (o parte de él) en un campo de búsqueda.

- Parámetros de entrada: Texto introducido en el campo de búsqueda.
- Parámetros de salida: Listado de documentos disponibles que coinciden con el nombre.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un documento.
- Postcondición: Se ha podido consultar uno o varios documentos disponibles en base al filtro.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Documentos".
2. El sistema ejecuta el caso de uso 02. ConsultarDocumentos.
3. El **Administrador** introduce un texto en el campo de búsqueda.
4. El sistema filtra los documentos según el texto introducido.
5. El sistema muestra el listado actualizado con los documentos coincidentes.

Alternativas de proceso y excepciones:

4a. No existen documentos que coincidan con el filtro:
    4a1. El sistema muestra un mensaje informativo.
4b. El **Administrador** borra el texto del campo de búsqueda:
    4b1. El sistema restablece la lista completa de documentos.

Caso de uso 04. BorrarDocumento

Resumen de la funcionalidad: Borrar un documento concreto.

- Parámetros de entrada: Identificador del documento.
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse consultado al menos un documento.
- Postcondición: Se ha podido eliminar el documento.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Documentos".
2. El sistema ejecuta el caso de uso 02. ConsultarDocumentos.
3. El **Administrador** selecciona el documento que desea eliminar.
4. El sistema muestra una confirmación de borrado.
5. El **Administrador** confirma la acción.
6. El sistema elimina el documento de Firebase Storage.

Alternativas de proceso y excepciones:

5b. El **Administrador** cancela la operación:
    5b1. El sistema no realiza ninguna acción.

Caso de uso 05. CrearAsignatura

Resumen de la funcionalidad: Crear una asignatura asociada a un campus.

- Parámetros de entrada: Nombre de la asignatura, curso académico, identificador del campus.
- Parámetros de salida: Lista actualizada con la nueva asignatura.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe existir al menos un campus.
- Postcondición: Se ha añadido la asignatura en la base de datos.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Asignaturas".
2. El sistema ejecuta el caso de uso 08. ConsultarAsignaturas.
3. El **Administrador** selecciona "Añadir asignatura".
4. El sistema muestra un formulario para introducir los campos necesarios.
5. El **Administrador** introduce el nombre, curso y campus de la asignatura.
6. El sistema valida los campos introducidos.
7. El sistema guarda la asignatura en la base de datos.

Alternativas de proceso y excepciones:

5b. No se han introducido todos los campos:
    5b1. El sistema notifica que faltan campos.

Caso de uso 06. EditarAsignatura

Resumen de la funcionalidad: Modificar los datos básicos de una asignatura.

- Parámetros de entrada: Identificador de la asignatura seleccionada, nombre de la asignatura y curso académico.
- Parámetros de salida: Lista con la asignatura actualizada.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y se debe haber podido consultar al menos una asignatura.
- Postcondición: Se ha editado la asignatura seleccionada en la base de datos.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Asignaturas".
2. El sistema ejecuta el caso de uso 08. ConsultarAsignaturas.
3. El **Administrador** selecciona la asignatura que quiere modificar.
4. El sistema carga los datos y muestra un modal para actualizar los datos necesarios.
5. El **Administrador** introduce el nombre y curso actualizado.
6. El sistema valida los campos introducidos.
7. El sistema actualiza la asignatura en la base de datos.

Alternativas de proceso y excepciones:

6b. No se han introducido todos los campos:
    6b1. El sistema notifica que faltan campos.

Caso de uso 07. BorrarAsignatura

Resumen de la funcionalidad: Borrar una asignatura específica.

- Parámetros de entrada: Identificador de la asignatura.
- Parámetros de salida: Lista actualizada sin la asignatura que se ha eliminado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y se debe haber podido consultar al menos una asignatura.
- Postcondición: Se ha borrado la asignatura seleccionada de la base de datos.

Proceso normal principal:

1. **Administrador** accede a la sección de "Asignaturas".
2. El sistema ejecuta el caso de uso 08. ConsultarAsignaturas.
3. El **Administrador** selecciona la asignatura que quiere eliminar.
4. El sistema muestra un diálogo para confirmar la eliminación.
5. El **Administrador** confirma la eliminación.
6. El sistema elimina la asignatura de la base de datos.

Alternativas de proceso y excepciones:

5b. El **Administrador** cancela la acción:
    5b1. El sistema no realiza ningún cambio.

Caso de uso 08. ConsultarAsignaturas

Resumen de la funcionalidad: Consultar todas las asignaturas disponibles.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Listado de asignaturas disponibles.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse creado como mínimo una asignatura.
- Postcondición: Se ha podido consultar una o varias asignaturas disponibles.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Asignaturas".
2. El sistema recupera y muestra las asignaturas disponibles.

Alternativas de proceso y excepciones:

2a. No hay asignaturas disponibles:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 09. AsignarFormulariosAsignatura

Resumen de la funcionalidad: Asignar enlaces de Microsoft Forms a una asignatura.

- Parámetros de entrada: Identificador de la asignatura, enlaces (evaluación inicial, final, autoevaluación).
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe existir la asignatura.
- Postcondición: Los enlaces quedan asociados a la asignatura.

Proceso normal principal:

1. El **Administrador** accede a la asignatura deseada.
2. El **Administrador** introduce los enlaces de los formularios.
3. El sistema almacena los enlaces en la base de datos asociados a la asignatura.

Alternativas de proceso y excepciones: Ninguna.

Caso de uso 10. AsignarCoordinadoresAsignatura

Resumen de la funcionalidad: Asociar uno o más coordinadores a una asignatura.

- Parámetros de entrada: Identificador de la asignatura, lista de coordinadores.
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**, debe existir la asignatura y al menos un coordinador.
- Postcondición: Los coordinadores han sido asignados a la asignatura en la base de datos.

Proceso normal principal:

1. El **Administrador** selecciona una asignatura.
2. El sistema muestra la lista de coordinadores disponibles.
3. El **Administrador** selecciona los coordinadores.
4. El sistema guarda configuración en la base de datos.

Alternativas de proceso y excepciones: Ninguna.

Caso de uso 11. CambiarContraseñaAsignatura

Resumen de la funcionalidad: Crear o sustituir contraseña para que los tutores puedan acceder a una asignatura.

- Parámetros de entrada: Contraseña, confirmación, fecha de caducidad.
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe existir la asignatura.
- Postcondición: La asignatura tiene una contraseña activa.

Proceso normal principal:

1. El **Administrador** selecciona una asignatura.
2. El **Administrador** introduce la nueva contraseña, la confirma y establece una fecha de expiración.
3. El sistema valida que la fecha de expiración es futura y las contraseñas (confirmación) coinciden.
4. El sistema guarda la nueva contraseña y se inactiva la anterior (si existe).

Alternativas de proceso y excepciones:

3a. Las contraseñas no coinciden o la fecha de expiración no es futura:
    3a1. El sistema no realiza ninguna acción.

Caso de uso 12. CrearCampus

Resumen de la funcionalidad: Crear un nuevo campus dentro del sistema.

- Parámetros de entrada: Nombre del campus.
- Parámetros de salida: Lista actualizada con el nuevo campus.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha creado un nuevo campus registrado en el sistema.

Proceso normal principal:

1. El **Administrador** accede a la sección de “Campus”.
2. El sistema ejecuta el caso de uso 15. ConsultarCampus.
3. El **Administrador** selecciona “Añadir campus”.
4. El sistema muestra un formulario para introducir el nombre del campus.
5. El **Administrador** introduce el nombre.
6. El sistema valida el campo.
7. El sistema guarda el nuevo campus en la base de datos.

Alternativas de proceso y excepciones:

6a. El nombre del campus está vacío o ya existe:
    6a1. El sistema muestra un mensaje de error y no permite la creación.

Caso de uso 13. EditarCampus

Resumen de la funcionalidad: Modificar el nombre de un campus existente.

- Parámetros de entrada: Identificador del campus, nuevo nombre.
- Parámetros de salida: Lista con el campus actualizado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un campus.
- Postcondición: Se ha editado el campus seleccionado.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Campus".
2. El sistema ejecuta el caso de uso 15. ConsultarCampus.
3. El **Administrador** selecciona el campus que quiere modificar.
4. El sistema muestra un formulario con el nombre actual.
5. El **Administrador** introduce el nuevo nombre.
6. El sistema valida el campo.
7. El sistema guarda el cambio en la base de datos.

Alternativas de proceso y excepciones:

6a. El nombre está vacío o coincide con otro existente:
    6a1. El sistema muestra un mensaje de error y no actualiza el nombre.

Caso de uso 14. BorrarCampus

Resumen de la funcionalidad: Eliminar un campus específico del sistema.

- Parámetros de entrada: Identificador del campus.
- Parámetros de salida: Lista de campus actualizada sin el campus eliminado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un campus.
- Postcondición: Se ha eliminado el campus seleccionado del sistema.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Campus".
2. El sistema ejecuta el caso de uso 15. ConsultarCampus.
3. El **Administrador** selecciona el campus a eliminar.
4. El sistema muestra un mensaje de confirmación.
5. El **Administrador** confirma la eliminación.
6. El sistema elimina el campus.

Alternativas de proceso y excepciones:

5a. El Administrador cancela la acción:
    5a1. El sistema no realiza ningún cambio.

6a. El campus tiene asignaturas asociadas:
    6a1. El sistema impide el borrado y muestra un mensaje de error.

Caso de uso 15. ConsultarCampus

Resumen de la funcionalidad: Mostrar el listado de campus registrados en el sistema.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Lista de campus registrados.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha mostrado el listado completo de los campus.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Campus".
2. El sistema obtiene los campus registrados.
3. El sistema muestra la lista en la interfaz.

Alternativas de proceso y excepciones:

2a. No hay campus registrados:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 16. CrearCoordinador

Resumen de la funcionalidad: Crear un nuevo coordinador académico.

- Parámetros de entrada: Nombre, apellidos, correo electrónico.
- Parámetros de salida: Lista actualizada con el nuevo coordinador.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: El nuevo coordinador se ha creado en la base de datos.

Proceso normal principal:

1. El **Administrador** accede a la sección "Coordinadores".
2. El sistema ejecuta el caso de uso 19. ConsultarCoordinadores.
3. El **Administrador** selecciona "Añadir coordinador".
4. El sistema muestra un formulario para introducir los datos del nuevo coordinador.
5. El **Administrador** llena los campos.
6. El sistema valida la información introducida.
7. El sistema crea el nuevo coordinador en la base de datos.

Alternativas de proceso y excepciones:

6a. El correo electrónico ya existe, tiene un formato inválido o faltan campos:
    6a1. El sistema notifica el error.

Caso de uso 17. EditarCoordinador

Resumen de la funcionalidad: Modificar los datos de un coordinador existente.

- Parámetros de entrada: Identificador del coordinador, nombre, apellidos y correo electrónico.
- Parámetros de salida: Lista con el coordinador actualizado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un coordinador.
- Postcondición: El coordinador seleccionado ha sido actualizado correctamente.

Proceso normal principal:

1. El **Administrador** accede a la sección "Coordinadores".
2. El sistema ejecuta el caso de uso 19. ConsultarCoordinadores.
3. El **Administrador** selecciona el coordinador que desea modificar.
4. El sistema muestra un formulario con los datos actuales.
5. El **Administrador** edita los campos deseados.
6. El sistema valida la información.
7. El sistema guarda los cambios en la base de datos.

Alternativas de proceso y excepciones:

6a. El correo electrónico ya existe, tiene un formato inválido o faltan campos:
    6a1. El sistema notifica el error.

Caso de uso 18. BorrarCoordinador

Resumen de la funcionalidad: Eliminar un coordinador específico del sistema.

- Parámetros de entrada: Identificador del coordinador.
- Parámetros de salida: Lista de coordinadores actualizada sin el usuario eliminado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un coordinador.
- Postcondición: El coordinador ha sido eliminado del sistema.

Proceso normal principal:

1. El **Administrador** accede a la sección "Coordinadores".
2. El sistema ejecuta el caso de uso 19. ConsultarCoordinadores.
3. El **Administrador** selecciona el coordinador a eliminar.
4. El sistema muestra un mensaje de confirmación.
5. El **Administrador** confirma la acción.
6. El sistema elimina al coordinador.

Alternativas de proceso y excepciones:

5a. El **Administrador** cancela la acción:
    5a1. El sistema no realiza ningún cambio.

Caso de uso 19. ConsultarCoordinadores

Resumen de la funcionalidad: Muestra el listado de coordinadores existentes.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Lista de coordinadores registrados.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha mostrado la lista de coordinadores.

Proceso normal principal:

1. El **Administrador** accede a la sección "Coordinadores".
2. El sistema recupera los coordinadores existentes.
3. El sistema muestra la lista.

Alternativas de proceso y excepciones:

2a. No hay coordinadores en la base de datos:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 20. ConsultarAdministradores

Resumen de la funcionalidad: Muestra el listado de Administradores existentes.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Lista de Administradores.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se obtiene y visualiza la lista de Administradores.

Proceso normal principal:

1. El **Administrador** accede a la sección "Administradores".
2. El sistema recupera los Administradores registrados.
3. El sistema muestra la lista.

Alternativas de proceso y excepciones:

2a. No hay Administradores registrados:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 21. EditarAdministrador

Resumen de la funcionalidad: Modificar los datos de un Administrador existente.

- Parámetros de entrada: Identificador del Administrador, su nuevo nombre, apellidos y correo.
- Parámetros de salida: Lista con el Administrador actualizado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un Administrador.
- Postcondición: El Administrador seleccionado ha sido actualizado correctamente.

Proceso normal principal:

1. El **Administrador** accede a la sección "Administradores".
2. El sistema ejecuta el caso de uso 20. ConsultarAdministradores.
3. El **Administrador** selecciona el Administrador que desea modificar.
4. El sistema muestra un formulario con los datos actuales.
5. El **Administrador** edita los campos deseados.
6. El sistema valida la información.
7. El sistema guarda los cambios en la base de datos.

Alternativas de proceso y excepciones:

6a. Algún campo obligatorio está vacío o el correo no tiene un formato válido:
    6a1. El sistema muestra un mensaje de error y no actualiza los datos.

Caso de uso 22. BorrarAdministrador

Resumen de la funcionalidad: Eliminar un Administrador específico del sistema.

- Parámetros de entrada: Identificador del Administrador.
- Parámetros de salida: Lista de Administradores actualizada sin el eliminado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un Administrador.
- Postcondición: El Administrador ha sido eliminado del sistema.

Proceso normal principal:

1. El **Administrador** accede a la sección "Administradores".
2. El sistema ejecuta el caso de uso 20. ConsultarAdministradores.
3. El **Administrador** selecciona el Administrador a eliminar.
4. El sistema muestra un mensaje de confirmación.
5. El **Administrador** confirma la acción.
6. El sistema elimina al Administrador.

Alternativas de proceso y excepciones:

5a. El **Administrador** cancela la acción:
    5a1. El sistema no realiza ningún cambio.

Caso de uso 23. ConsultarTutores

Resumen de la funcionalidad: Mostrar el listado de tutores registrados en el sistema.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Lista de tutores registrados.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha mostrado el listado completo de tutores.

Proceso normal principal:

1. El **Administrador** accede a la sección "Tutores".
2. El sistema recupera los tutores registrados.
3. El sistema muestra la lista con los Tutores.

Alternativas de proceso y excepciones:

2a. No hay tutores registrados:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 24. EliminarTutor

Resumen de la funcionalidad: Eliminar un tutor específico del sistema.

- Parámetros de entrada: Identificador del tutor.
- Parámetros de salida: Lista de tutores actualizada sin el tutor eliminado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un tutor.
- Postcondición: El tutor ha sido eliminado del sistema.

Proceso normal principal:

1. El **Administrador** accede a la sección "Tutores".
2. El sistema ejecuta el caso de uso 23. ConsultarTutores.
3. El **Administrador** selecciona el tutor que desea eliminar.
4. El sistema muestra un mensaje de confirmación.
5. El **Administrador** confirma la acción.
6. El sistema elimina al tutor.

Alternativas de proceso y excepciones:

5a. El **Administrador** cancela la acción:
    5a1. El sistema no realiza ningún cambio.

Caso de uso 25. ConsultarRegistroHoraEnUnCurso

Resumen de la funcionalidad: Permite al **Administrador** consultar los registros de horas enviados por los tutores en un curso concreto.

- Parámetros de entrada: Identificador del curso.
- Parámetros de salida: Listado de registros de horas asociados al curso.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y se ha creado previamente el curso en el registro de horas.
- Postcondición: Se ha mostrado el listado de registros de horas correspondientes al curso seleccionado.

Proceso normal principal:

1. El **Administrador** accede a la sección "Registros de horas".
2. El sistema muestra un listado de cursos activos.
3. El **Administrador** selecciona un curso.
4. El sistema muestra los registros de horas asociados al curso.

Alternativas de proceso y excepciones:

4a. El curso no tiene registros:
    4a1. El sistema muestra un mensaje informativo.

Caso de uso 26. CrearCursoRegistroHora

Resumen de la funcionalidad: Crear un nuevo curso de prácticas para el registro de horas de los tutores.

- Parámetros de entrada: Nombre del curso con formato YYYY-YYYY.
- Parámetros de salida: Curso de registro creado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha creado un nuevo curso de prácticas.

Proceso normal principal:

1. El **Administrador** accede a la sección "Registros de horas".
2. El sistema muestra un listado de los cursos actuales.
3. El **Administrador** selecciona la opción "Crear nuevo curso".
4. El sistema muestra un formulario con el campo del nombre.
5. El **Administrador** introduce el nombre del curso.
6. El sistema valida el campo.
7. El sistema guarda el curso en la base de datos.

Alternativas de proceso y excepciones:

6a. No se ha introducido el nombre o tiene un formato inválido (debe ser YYYY-YYY):
    6a1. El sistema notifica del error y no realiza cambios.

Caso de uso 27. ExportarRegistroHoraCursoExcel

Resumen de la funcionalidad: Exportar los registros de horas de un curso a un archivo Excel.

- Parámetros de entrada: Identificador del curso.
- Parámetros de salida: Excel con los registros de hora.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y ha consultado previamente un curso.
- Postcondición: Se ha descargado un archivo Excel con los registros de horas del curso.

Proceso normal principal:

1. El **Administrador** accede a la sección de "Registros de horas".
2. El sistema ejecuta el caso de uso 25. ConsultarRegistroHoraEnUnCurso.
3. El **Administrador** pulsa el botón "Exportar datos".
4. El sistema genera el archivo Excel y lo descarga.

Alternativas de proceso y excepciones:

3a. No hay registros que exportar:
    3a1. El botón "Exportar datos" no es visible.

Caso de uso 28. CrearCodigoRegistro

Resumen de la funcionalidad: Generar un nuevo código de registro para permitir el alta de tutores clínicos en el sistema.

- Parámetros de entrada: Código de registro y fecha de expiración.
- Parámetros de salida: Código de registro generado con "masking" aplicado.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha generado un código de registro.

Proceso normal principal:

1. El **Administrador** accede a la sección "Códigos de registro".
2. El sistema ejecuta el caso de uso 30. ConsultarCodigosRegistro.
3. El **Administrador** selecciona la opción "Crear código".
4. El sistema muestra un formulario para introducir el código y la fecha de expiración.
5. El **Administrador** introduce los datos.
6. El sistema valida los campos.
7. El sistema genera el código, le aplica "masking" y lo guarda.

Alternativas de proceso y excepciones:

6a. Faltan campos por rellenar, la fecha no es en el futuro o el código es muy corto:
    6a1. El sistema muestra un mensaje de error y no guarda el código.

Caso de uso 29. BorrarCodigoRegistro

Resumen de la funcionalidad: Eliminar un código de registro específico del sistema.

- Parámetros de entrada: Identificador del código de registro.
- Parámetros de salida: Lista actualizada de códigos de registro.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador** y debe haberse podido consultar al menos un código de registro.
- Postcondición: El código seleccionado ha sido eliminado del sistema.

Proceso normal principal:

1. El **Administrador** accede a la sección "Códigos de registro".
2. El sistema ejecuta el caso de uso 30. ConsultarCodigosRegistro.
3. El **Administrador** selecciona el código que desea eliminar.
4. El sistema muestra una confirmación de eliminación.
5. El **Administrador** confirma la acción.
6. El sistema elimina el código.

Alternativas de proceso y excepciones:

5a. El **Administrador** cancela la acción:
    5a1. El sistema no realiza ningún cambio.

Caso de uso 30. ConsultarCodigosRegistro

Resumen de la funcionalidad: Visualizar el listado de códigos de registro disponibles en el sistema.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Lista de códigos de registro creados.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión como **Administrador**.
- Postcondición: Se ha mostrado el listado completo de códigos de registro.

Proceso normal principal:

1. El **Administrador** accede a la sección "Códigos de registro".
2. El sistema recupera todos los códigos de registro existentes.
3. El sistema muestra la lista.

Alternativas de proceso y excepciones:

2a. No hay códigos de registro generados:
    2a1. El sistema muestra un mensaje informativo.

Caso de uso 31. Salir

Resumen de la funcionalidad: Cerrar la sesión del usuario autenticado y redirigirlo a la pantalla de inicio de sesión.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Ninguno.
- Actores: **Administrador**.
- Precondición: El usuario ha iniciado sesión en la aplicación como **Administrador**.
- Postcondición: La sesión ha sido cerrada y el usuario ha sido redirigido a la pantalla de inicio de sesión.

Proceso normal principal:

1. El **Administrador** pulsa el botón "Cerrar sesión" desde cualquier vista de la aplicación.
2. El sistema cierra la sesión del usuario autenticado.
3. El sistema redirige al **Administrador** a la pantalla de inicio de sesión.

Alternativas de proceso y excepciones: Ninguno.

5.1.2 Requisitos no funcionales

A continuación, se presentan los requisitos no funcionales, es decir, aquellas condiciones y restricciones de calidad que debe cumplir el sistema.

Requisitos de seguridad:

1. El acceso al sistema estará protegido mediante autenticación basada en correo electrónico y contraseña.

2. Solo los usuarios autenticados como **Administradores** podrán acceder a la aplicación de gestión.

3. Los datos estarán almacenados en Firestore con reglas de seguridad que impedirán la visualización de información a los usuarios no autorizados.

4. Los códigos de registro se almacenarán en la base de datos enmascarados para evitar posibles accesos no autorizados.

5. La plataforma debe impedir que los usuarios desactivados accedan al sistema, incluso si poseen credenciales correctas.

Requisitos de diseño y construcción:

6. El lenguaje de programación utilizado es TypeScript, utilizando el Framework Vue.js para el Frontend.

7. La gestión de datos se realiza mediante Firebase (Firestore, Authentication y Storage).

8. La aplicación debe poder ejecutarse correctamente en navegadores modernos sobre sistemas operativos Windows, macOS, Android, iOS o Linux.

9. El sistema debe seguir una estructura de componentes reutilizables, con separación clara entre lógica de presentación, control de acceso y servicios.

10. Las reglas de seguridad y control de acceso deben estar centralizadas en Firebase, facilitando su gestión y modificación sin necesidad de volver a implementar la lógica en el Frontend.

11. El despliegue del sistema se debe realizar mediante Firebase Hosting.

Requisitos de usabilidad:

12. El sistema debe ofrecer una interfaz clara y coherente, pensada para su uso por parte de personal administrativo sin formación técnica.

13. La navegación entre las distintas secciones debe realizarse con un menú lateral, accesible desde cualquier vista de la aplicación.

14. Cada formulario debe contar con validación de campos y mensajes de error específicos en caso de datos incorrectos o incompletos.

15. Las operaciones críticas, como eliminación de datos o desactivación de cuentas, deben requerir una confirmación explícita por parte del usuario mediante un diálogo.

16. Las vistas deben ser adaptables a distintos tamaños de pantalla. Deben garantizar una correcta visualización en equipos de escritorio, tabletas y teléfonos.

17. Deben existir diálogos que informen al usuario sobre el éxito o fallo de las operaciones realizadas.

18. Las acciones deben estar agrupadas de forma lógica, y los formularios deben mostrar solo los campos necesarios según el contexto, para evitar sobrecargar al usuario.

5.2 Requisitos de la aplicación de tutores

A continuación, se procederá a indicar los requisitos, tanto funcionales como no funcionales, de la aplicación que utilizarán los tutores clínicos.

5.2.1 Requisitos funcionales

En el siguiente apartado se incluye el glosario de términos, las reglas de negocio, el diagrama de casos de uso y la especificación textual detallada de cada uno de ellos.

5.2.1.1 Glosario de términos

- **Estudiante**: Estudiante URV de enfermería que realiza las prácticas clínicas.
- **Tutor:** Profesional de la salud, externo a la URV, que hace el seguimiento del estudiante durante las prácticas.
- **Coordinador:** Persona URV responsable de la gestión académica de las prácticas.
- **Administrador:** Coordinador/a con permisos para gestionar el sistema.
- **Código de registro:** Clave que permite a los tutores registrarse. Se utiliza para evitar registros no autorizados.
- **Campus:** Centro donde se imparten las asignaturas.
- **Asignatura:** Módulo académico de prácticas clínicas.
- **Contraseña de asignatura**: Estas contraseñas están asignadas a las asignaturas y son necesarias para poder acceder a ellas.
- **Documento**: Archivo oficial, como normativa, protocolo, etc.
- **Actividad**: Tarea o ejercicio que el estudiante debe completar durante sus prácticas.
- **Registro de horas:** Tiempo anotado por los tutores respecto a las prácticas realizadas por los estudiantes.
- **Alerta**: Notificación para informar al tutor sobre novedades o incidencias.
- **Preguntas frecuentes (FAQ)**: Listado de dudas y respuestas comunes sobre el uso de la aplicación.

5.2.1.2 Reglas del negocio

1. Solo los **Tutores** registrados y activos tienen acceso a la aplicación.
2. No puede existir más de un **Tutor** con el mismo correo electrónico.
3. Para registrarse, el **Tutor** debe proporcionar un código de registro válido y no expirado.
4. Los DNI deben tener un formato válido.
5. Los correos electrónicos deben tener un formato válido.
6. Los campos obligatorios no pueden estar vacíos.
7. Todas las asignaturas están protegidas mediante contraseña.
8. Para acceder a las asignaturas se requiere proporcionar una contraseña válida para esa asignatura.
9. Una vez el **Tutor** ha introducido una contraseña válida de una asignatura, la sesión activa almacena el acceso a la asignatura hasta la fecha de expiración de la contraseña. El sistema no la volverá a pedir hasta que la contraseña expire.
10. Las fechas de registro de horas deben corresponder a periodos de prácticas activos.
11. No se puede usar un código de registro ni una contraseña si su fecha de expiración está en el pasado.
12. Todos los archivos deben estar ordenados alfabéticamente.

5.2.1.3 Diagrama de casos de uso

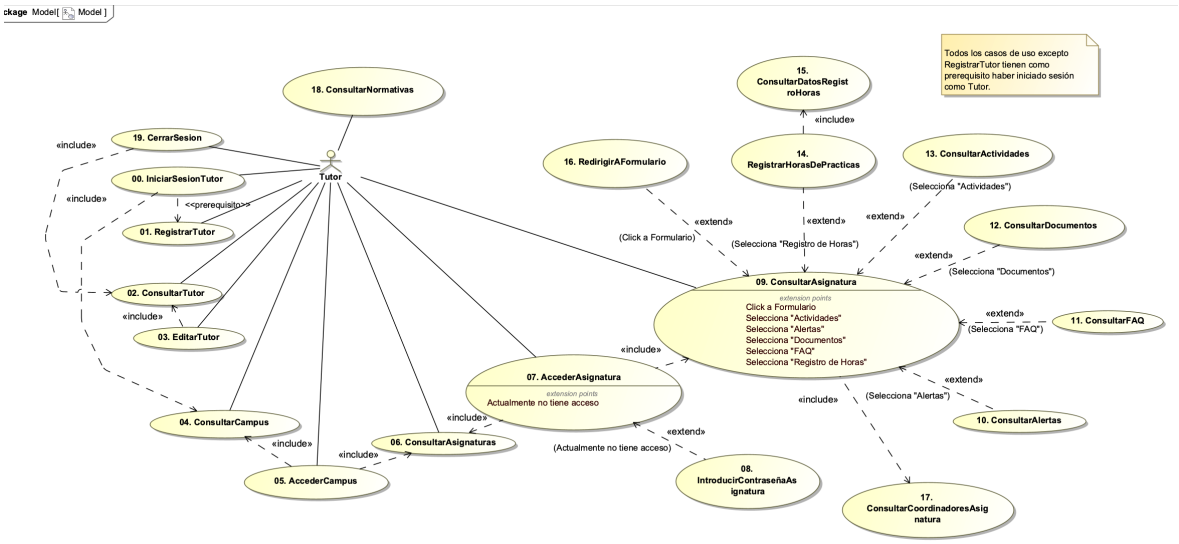


Figura 4. Aplicación de Tutores - Diagrama de casos de uso

5.2.1.4 Especificación textual de los casos de uso

En el siguiente apartado, se procederá a especificar textualmente los casos de uso referentes a la Figura 4 del apartado anterior.

Caso de uso 00. IniciarSesionTutor

Resumen de la funcionalidad: Autenticar al usuario como **Tutor** para permitir el acceso a la aplicación de tutores clínicos.

- Parámetros de entrada: Correo electrónico, contraseña.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: El **Tutor** se registró previamente.
- Postcondición: El **Tutor** ha iniciado sesión correctamente.

Proceso normal principal:

1. El sistema solicita correo electrónico y la contraseña.
2. El **Tutor** introduce sus credenciales.
3. El sistema valida que las credenciales sean válidas.
4. El sistema ejecuta el caso de uso 04. ConsultarCampus.

Alternativas y excepciones:

3a. El correo no existe o la contraseña es incorrecta:
3a1. El sistema muestra mensaje de “Credenciales inválidas” y no permite el acceso.

Caso de uso 01. RegistrarTutor

Resumen de la funcionalidad: Dar de alta un nuevo **Tutor** clínico en el sistema empleando un código de registro.

- Parámetros de entrada: Nombre, apellidos, DNI, correo electrónico, centro de trabajo, área de trabajo, tipo de centro, código de registro, contraseña, confirmación de contraseña.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ninguna.
- Postcondición: Se ha creado la cuenta del **Tutor**.

Proceso normal principal:

1. El **Tutor** accede al formulario de registro.
2. Introduce sus datos personales justo al código de registro, contraseña y confirmación.
3. El sistema valida que el correo y el DNI tengan un formato válido, mira que el código de registro sea válido, que no queden campos obligatorios y que la contraseña esté confirmada (introducida dos veces, deben coincidir).
4. El sistema crea la cuenta.

Alternativas de proceso y excepciones:

3a. Correo electrónico ya registrado para otro Tutor:
    3a1. El sistema muestra “Correo ya en uso” y no crea la cuenta.
3b. Código de registro inválido o expirado:
    3b1. El sistema muestra “Código de registro inválido” y no crea la cuenta.
3c. Contraseña y confirmación no coinciden:
    3c1. El sistema muestra “Contraseña no válida” y no crea la cuenta.
3d. Quedan campos pendientes por completar:
    3d1. El sistema notifica qué campos quedan por completar.

Caso de uso 02. ConsultarTutor

Resumen de la funcionalidad: Mostrar los datos del perfil del **Tutor**.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Datos del **Tutor**.
- Actor: **Tutor**.
- Precondición: El **Tutor** ha iniciado sesión.
- Postcondición: Se muestran los datos actuales del **Tutor**.

Proceso normal principal:

1. El **Tutor** accede a la sección “Mi perfil”.
2. El sistema recupera y muestra los datos del **Tutor**.

Alternativas de proceso y excepciones: Ninguna.

Caso de uso 03. EditarTutor

Resumen de la funcionalidad: Modificar alguno de los datos de perfil del **Tutor**.

- Parámetros de entrada: Área de trabajo, centro de trabajo, tipo de centro.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha consultado los datos del **Tutor**.
- Postcondición: Los cambios quedan guardados en Firestore.

Proceso normal principal:

1. El **Tutor** pulsa “Editar perfil”.
2. El sistema ejecuta el caso de uso 02. ConsultarTutor.
3. El **Tutor** modifica los campos deseados.
4. El sistema valida campos obligatorios y formato.
5. El sistema actualiza los datos en Firestore.

Alternativas y excepciones:

4a. Campos inválidos o vacíos:
    4a1. El sistema notifica los campos con formato inválido o que están vacíos.

Caso de uso 04. ConsultarCampus

Resumen de la funcionalidad: Listar los campus disponibles.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Listado de campus.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión como **Tutor**.
- Postcondición: Se obtiene una lista con los campus.

Proceso normal principal:

1. El sistema recupera la colección de campus de Firestore.
2. El sistema presenta los campus en una lista.

Alternativas de proceso y excepciones:

1a. No hay campus registrados:
    1a1. El sistema notifica que no hay campus disponibles.

Caso de uso 05. AccederCampus

Resumen de la funcionalidad: Seleccionar un campus para proceder con posterioridad a visualizar sus asignaturas.

- Parámetros de entrada: Identificador de campus.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha consultado al menos un campus.
- Postcondición: Navega hasta la ventana de asignaturas de ese campus para consultarlas.

Proceso normal principal:

1. El sistema ejecuta el caso de uso 04. ConsultarCampus.
2. El **Tutor** selecciona un campus de la lista.
3. El sistema ejecuta el caso de uso 06. ConsultarAsignaturas.

Alternativas de proceso y excepciones: Ninguna.

Caso de uso 06. ConsultarAsignaturas

Resumen de la funcionalidad: Ver todas las asignaturas disponibles para el campus específico.

- Parámetros de entrada: Identificador del campus.
- Parámetros de salida: Listado de las asignaturas del campus especificado.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión como **Tutor**.
- Postcondición: Listado de asignaturas del campus especificado.

Proceso normal principal:

1. El sistema recupera las asignaturas de la base de datos que correspondan al identificador especificado.
2. El sistema presenta la lista de las asignaturas del campus.

Alternativa de proceso y excepciones:

2a. No hay asignaturas asignadas:
    2a1. El sistema notifica que no hay asignaturas disponibles.

Caso de uso 07. AccederAsignatura

Resumen de la funcionalidad: Abrir la pestaña de detalle de una asignatura concreta.

- Parámetros de entrada: Identificador del campus e identificador de asignatura.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha consultado al menos una asignatura.
- Postcondición: Se abre la pestaña con los detalles de la asignatura.

Proceso normal principal:

1. El sistema ejecuta el caso de uso 06. ConsultarAsignaturas.
2. El **Tutor** selecciona una asignatura de la lista.
3. El sistema comprueba si ya existe en la sesión actual una entrada válida de contraseña para esa asignatura y que no esté caducada:
    a. Si existe y sigue vigente, va al paso 5.
    b. Si no existe o ha expirado, va al paso 4.
4. El sistema ejecuta el caso de uso 08. IntroducirContraseñaAsignatura.
5. El sistema ejecuta el caso de uso 09. ConsultarAsignatura.

Alternativa de proceso y excepciones: Ninguna.

Caso de uso 08. IntroducirContraseñaAsignatura

Resumen de la funcionalidad: Validar y almacenar en la sesión el acceso a una asignatura tras introducir la contraseña de una asignatura.

- Parámetros de entrada: Contraseña de la asignatura.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha intentado acceder a una asignatura protegida sin tener acceso en la sesión actual.
- Postcondición: Se almacena en la sesión un acceso a la asignatura hasta la fecha de expiración de la contraseña.

Proceso normal principal:

1. El sistema muestra modal solicitando la contraseña de la asignatura.
2. El **Tutor** introduce la contraseña.
3. El sistema verifica la contraseña contra la base de datos y comprueba que la fecha de expiración sea futura.
4. Si la contraseña es correcta y no ha expirado, el sistema almacena el identificador de la asignatura en la sesión del usuario junto a la fecha de expiración de la contraseña (no almacena la contraseña).

Alternativa de proceso y excepciones:

3a. Contraseña incorrecta o expirada:
3a1. El sistema notifica que la contraseña es inválida o está caducada y se cierra el modal.

Caso de uso 09. ConsultarAsignatura

Resumen de la funcionalidad: Muestra los detalles más importantes de la asignatura especificada.

- Parámetros de entrada: Identificador del campus e identificador de la asignatura.
- Parámetros de salida: Datos de la asignatura.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha abierto el panel de asignatura.
- Postcondición: Muestra una pestaña con acceso rápido a todos los recursos y muestra datos relevantes como los coordinadores de la asignatura.

Proceso normal principal:

1. El sistema ejecuta el caso de uso 17. ConsultarCoordinadoresAsignatura.
2. El sistema obtiene de la base de datos los campos básicos de la asignatura.
3. El sistema muestra una pestaña con accesos rápidos a todos los recursos de la asignatura (Actividades, documentos, alertas, FAQ, registro de horas y formularios).

Alternativa de proceso y excepciones: Ninguna.

Caso de uso 10. ConsultarAlertas

Resumen de la funcionalidad: Mostrar una lista con las alertas de la asignatura (si las hubiera).

- Parámetros de entrada: Identificador del campus, identificador de la asignatura.
- Parámetros de salida: Lista de alertas.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha accedido a la sección de “Alertas” desde el panel de la asignatura.
- Postcondición: Se muestran las alertas existentes.

Proceso normal principal:

1. El sistema recupera de la base de datos las alertas asociadas a la asignatura.
2. El sistema muestra una lista con las alertas.

Alternativa de proceso y excepciones:

2a. No hay alertas registradas:
    2a1. El sistema notifica de que no hay alertas.

Caso de uso 11. ConsultarFAQ

Resumen de la funcionalidad: Mostrar las preguntas frecuentes de la plataforma.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Listado de preguntas y respuestas.
- Actor: **Tutor**.
- Precondición: Ha accedido a la sección de FAQ desde el panel de la asignatura.
- Postcondición: Se muestran las FAQs disponibles.

Proceso normal principal:

1. El sistema muestra las preguntas y respuestas en acordeones desplegables.

Alternativa de proceso y excepciones: Ninguna.

Caso de uso 12. ConsultarDocumentos

Resumen de la funcionalidad: Listar los documentos disponibles para la asignatura.

- Parámetros de entrada: Identificador del campus, identificador de la asignatura.
- Parámetros de salida: Listado de documentos.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha accedido a la sección de "Documentos" en el panel de la asignatura.
- Postcondición: Se presentan los documentos disponibles.

Proceso normal principal:

1. El sistema recupera de la base de datos los documentos que serán visibles en la asignatura, este conjunto de documentos se extrae de mezclar diferentes documentos con 4 ámbitos diferentes[5]:
    1. **Globales**: para todos los campus y todas las asignaturas.
    2. **Por campus**: para un campus y todas sus asignaturas.
    3. **Por asignatura**: para una asignatura concreta (en todos los campus).
    4. **Mixto**: para una asignatura concreta en un campus específico.
2. El sistema muestra la lista con los documentos obtenidos.

Alternativa de proceso y excepciones:

2a. No hay documentos disponibles:
    2a1. El sistema notifica que no hay documentos disponibles para la asignatura.

5 Se decidió entre todos dividir los documentos y actividades en 4 grandes bloques con el fin de evitar redundancia, pues hay múltiples documentos que estarán en múltiples lugares a la vez, tanto a nivel de campus como de asignatura.

Caso de uso 13. ConsultarActividades

Resumen de la funcionalidad: Listar las actividades disponibles para la asignatura.

- Parámetros de entrada: Identificador del campus, identificador de la asignatura.
- Parámetros de salida: Listado de actividades.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha accedido a la sección de "Actividades" en el panel de la asignatura.
- Postcondición: Se presentan las actividades disponibles.

Proceso normal principal:

1. El sistema recupera de la base de datos las actividades que serán visibles en la asignatura, este conjunto de actividades se extrae de mezclar diferentes actividades con 4 ámbitos diferentes:
    1. **Globales**: para todos los campus y todas las asignaturas.
    2. **Por campus**: para un campus y todas sus asignaturas.
    3. **Por asignatura**: para una asignatura concreta (en todos los campus).
    4. **Mixto**: para una asignatura concreta en un campus específico.
2. El sistema muestra la lista con las actividades obtenidas.

Alternativa de proceso y excepciones:

2a. No hay actividades disponibles:
    2a1. El sistema notifica que no hay actividades disponibles para la asignatura.

Caso de uso 14. RegistrarHorasDePracticas

Resumen de la funcionalidad: Permitir al **Tutor** introducir el registro de horas prácticas del/la estudiante.

- Parámetros de entrada: Curso actual, nombre y apellidos del estudiante, asignatura que realiza, horas realizadas, comentarios.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y ha accedido a la sección de "Registro de Horas" en el panel de la asignatura.
- Postcondición: Se crea un nuevo registro de horas en la base de datos.

Proceso normal principal:

1. El sistema ejecuta el caso de uso 15. ConsultarDatosRegistroHoras.
2. El sistema muestra un formulario con datos rellenados y con los campos por rellenar por parte del **Tutor** vacíos (los datos del alumno).
3. El **Tutor** llena los campos del alumno.
4. El sistema valida los campos.
5. El sistema guarda el registro en la base de datos.

Alternativa de proceso y excepciones:

4a. Hay campos obligatorios vacíos:
    4a1. El notifica qué campos faltan por completar.

Caso de uso 15. ConsultarDatosRegistroHoras

Resumen de la funcionalidad: Carga los cursos académicos existentes para realizar el formulario y obtiene ciertos datos del perfil del usuario y el campus y asignatura actual para autocompletar el formulario con esos valores con tal de agilizar trabajo.

- Parámetros de entrada: Identificador del campus e identificador de la asignatura.
- Parámetros de salida: Listado de cursos (periodos académicos YYYY–YYYY), datos del perfil del usuario y datos del campus y la asignatura actual.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión.
- Postcondición: Se obtienen datos necesarios para autocompletar campos en el formulario.

Proceso normal principal:

1. El sistema recupera los cursos académicos disponibles.
2. El sistema obtiene los campos del usuario activo.
3. El sistema obtiene el nombre del campus actual.
4. El sistema obtiene el nombre de la asignatura actual.

Alternativa de proceso y excepciones: Ninguna.

Caso de uso 16. RedirigirAFormulario

Resumen de la funcionalidad: Abrir un formulario externo (Microsoft Forms) asociado a la asignatura.

- Parámetros de entrada: Enlace de formulario.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y está en el panel de la asignatura.
- Postcondición: Se abre el formulario en una nueva pestaña del navegador.

Proceso normal principal:

1. El **Tutor** pulsa alguno de los 3 formularios de Microsoft en el panel de la asignatura.
2. El sistema abre el enlace configurado en una nueva pestaña del navegador.

Alternativa de proceso y excepciones:

1a. Enlace no configurado o inválido:
    1a1. El sistema notifica que el formulario no está disponible.

Caso de uso 17. ConsultarCoordinadoresAsignatura

Resumen de la funcionalidad: Mostrar los datos de contacto de los coordinadores académicos de la asignatura (puede haber uno o dos).

- Parámetros de entrada: Identificadores de los coordinadores, identificador del campus, identificador de la asignatura.
- Parámetros de salida: Listado de coordinadores (nombre, correo electrónico).
- Actor: **Tutor**.
- Precondición: Ha iniciado sesión y está en el panel de la asignatura.
- Postcondición: Se muestran los datos de los coordinadores.

Proceso normal principal:

1. El sistema recupera de la base de datos la lista de coordinadores asignados.
2. El sistema muestra los datos de los coordinadores.

Alternativa de proceso y excepciones:

2a. No hay coordinadores asignados:
    2a1. El sistema notifica que no hay coordinador o coordinadores asignados.

Caso de uso 18. ConsultarNormativas

Resumen de la funcionalidad: Mostrar las normativas y protocolos disponibles en toda la aplicación.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Listado de normativas y protocolos.
- Actor: **Tutor**.
- Precondición: Haber iniciado sesión en la aplicación.
- Postcondición: Se presentan las normativas disponibles.

Proceso normal principal:

1. El **Tutor** selecciona la opción “Normativas y protocolos” accesible en toda la aplicación.
2. El sistema recupera de la base de datos todas las normativas y protocolos.
3. El sistema muestra la lista de normativas.

Alternativa de proceso y excepciones: Ninguna.

Caso de uso 19. CerrarSesion

Resumen de la funcionalidad: Finalizar la sesión del **Tutor** y volver a la pantalla de inicio de sesión.

- Parámetros de entrada: Ninguno.
- Parámetros de salida: Ninguno.
- Actor: **Tutor**.
- Precondición: El **Tutor** ha iniciado sesión.
- Postcondición: Sesión cerrada y navega a la pantalla de inicio de sesión.

Proceso normal principal:

1. El **Tutor** navega hacia su Perfil.
2. El sistema ejecuta el caso de uso 02. ConsultarTutor.
3. El **Tutor** selecciona “Cerrar sesión”.
4. El sistema cierra la sesión del usuario.
5. El sistema redirige a la pantalla de inicio de sesión.

Alternativa de proceso y excepciones: Ninguna.

5.2.2 Requisitos no funcionales

A continuación, se presentan los requisitos no funcionales, es decir, aquellas condiciones y restricciones de calidad que debe cumplir el sistema.

Requisitos de seguridad:

1. El acceso al sistema estará protegido mediante autenticación basada en correo electrónico y contraseña.

2. Solo los usuarios identificados como **Tutores** podrán acceder a la aplicación.

3. Los datos estarán almacenados en Firestore, con reglas que impedirán la visualización de información a los usuarios no autorizados.

4. Los códigos de registro se almacenarán enmascarados para evitar accesos no autorizados.

5. La plataforma debe impedir que los usuarios desactivados accedan al sistema, incluso si poseen credenciales correctas.

6. Los datos personales se deben anonimizar utilizando claves criptográficas.

Requisitos de diseño y construcción:

7. El lenguaje de programación utilizado es TypeScript, utilizando el Framework Vue.js para el Frontend.

8. La gestión de datos se realiza mediante Firebase (Firestore, Authentication y Storage).

9. La aplicación debe poder ejecutarse correctamente en navegadores modernos sobre sistemas operativos Windows, macOS, Android, iOS o Linux.

10. El sistema debe seguir una estructura de componentes reutilizables, con separación clara entre lógica de presentación, control de acceso y servicios.

11. Las reglas de seguridad y control de acceso deben estar centralizadas en Firebase, facilitando su gestión y modificación sin necesidad de volver a implementar la lógica en el Frontend.

12. El despliegue del sistema se debe realizar mediante Firebase Hosting.

Requisitos de usabilidad:

13. El sistema debe ofrecer una interfaz clara y coherente, pensada para su uso por parte de los **Tutores** sin formación técnica.

14. Cada formulario debe contar con validación de campos y mensajes de error específicos en caso de datos incorrectos o incompletos.

15. Las vistas deben ser adaptables a distintos tamaños de pantalla, para garantizar una correcta visualización en equipos de escritorio, tabletas y teléfonos.

16. Deben existir diálogos que informen al usuario sobre el éxito o fallo de las operaciones realizadas.

17. Las acciones deben estar agrupadas de forma lógica, y los formularios deben mostrar solo los campos necesarios según el contexto para evitar sobrecargar al usuario.

6 Diseño

En esta sección se presentarán las decisiones y principios de diseño que se han aplicado a las dos aplicaciones. Se comentará desde la arquitectura general y la organización de datos, hasta la presentación visual y la experiencia del usuario.

6.1 Arquitectura de las aplicaciones

La solución se compone de dos aplicaciones independientes, una para tutores clínicos y otra para coordinadores académicos. Ambas han sido desarrolladas con Vue.js y construidas utilizando Vite, se decidió utilizar este Framework debido a que la convocatoria de la beca establecía el uso de esta herramienta.

Cada aplicación Frontend está publicada de manera independiente en Firebase Hosting. Firebase Hosting nos permite desplegar las aplicaciones y configurarles un dominio. Se decidieron desplegar las aplicaciones por separado, por seguridad y para evitar confusión a los usuarios. Ambas aplicaciones soportan tráfico HTTPS/TLS.

En cuanto a la persistencia de datos, se utiliza Cloud Firestore para organizar la información, Firebase Storage para alojar todos los archivos y Firebase Authentication para determinar quién puede leer o escribir en Firestore y Storage según su rol.

Como estos 3 servicios son propios de Firebase y están bien integrados, el contenido y seguridad del sistema es fácil de mantener.

Además, referente a los archivos, estos se dividen en 4 ámbitos diferentes para evitar tener redundancia de datos. Esto está diseñado de esta manera porque un archivo puede pertenecer a un campus y asignatura concretos, a cualquier asignatura de un campus concreto, a una asignatura concreta en todos los campus o en todas las asignaturas de cualquier campus.

6.2 Diseño de las interfaces

En este apartado se muestra, para cada aplicación, diferentes capturas de su interfaz, tanto en versión de escritorio como en versión móvil, acompañadas de una breve explicación de su función. Durante el desarrollo del proyecto, la creación de interfaces adaptativas, que responden al tamaño de pantalla de cada dispositivo, ha sido muy relevante para ofrecer una buena experiencia de usuario. Esto es importante porque los diferentes coordinadores y tutores se habituarán en su mayoría a utilizar las aplicaciones en su versión móvil.

Como ya se dijo, estas interfaces han sido previamente diseñadas en Figma para poder trabajar con el diseño y mostrar una maqueta previa al desarrollo a las entidades coordinadoras de este proyecto.

Se han seguido buenas prácticas de diseño tanto en tipografía como en organización de contenido y en la jerarquización de la información. Dado que la mayoría de los usuarios son mujeres, el estilo general tiende a ser más femenino; sin embargo, se ha buscado un equilibrio cromático y tipográfico que resulte cómodo y atractivo también para los hombres. De este modo, se favorece la adopción y el uso continuado de la plataforma por parte de ambos géneros.

6.2.1 Aplicación de gestión

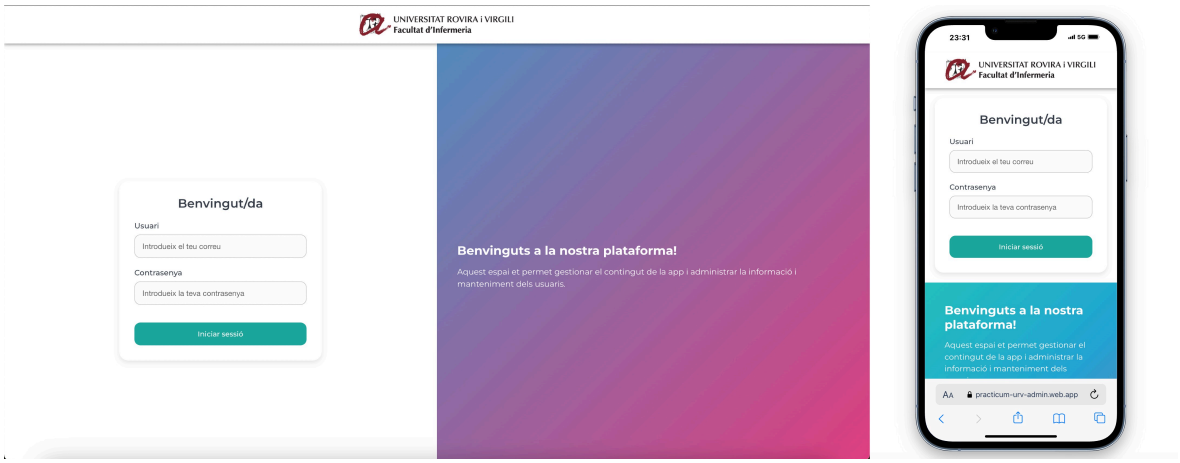


Figura 5. Aplicación gestión – Inicio de sesión

Esta es la pantalla de inicio. Aquí el usuario puede iniciar sesión como **Administrador**.

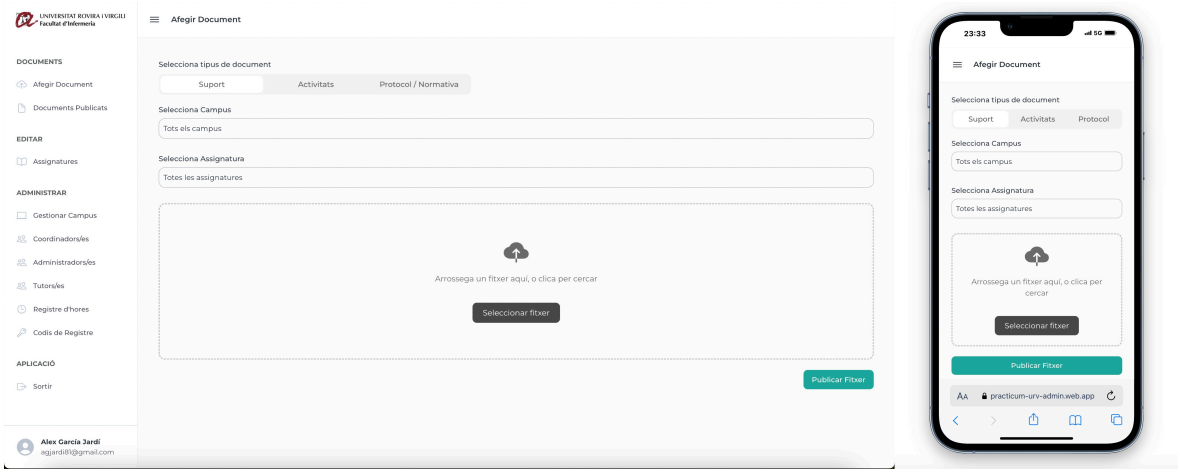


Figura 6. Aplicación gestión – Subida de documentos

Cuando el usuario inicia sesión, se le redirige a esta página. Aquí el usuario podrá subir diferentes tipos de documentos (actividad, soporte, protocolo / normativa) y especificar dónde quiere publicarlo. Se puede acceder a él en el futuro navegando en el menú lateral, en "Subir documento".

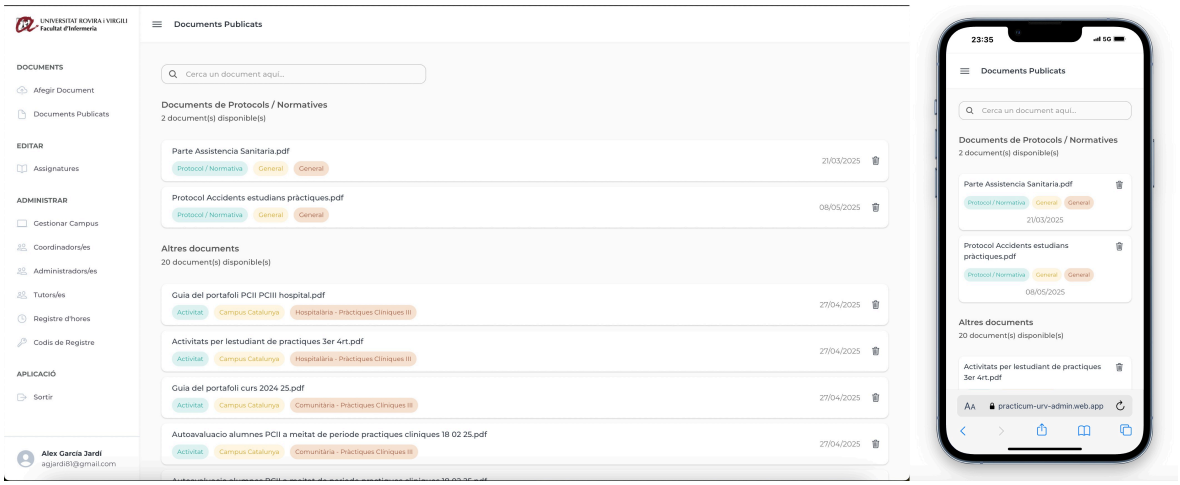


Figura 7. Aplicación gestión – Listado de documentos

Esta pantalla muestra el listado de los documentos publicados. Para llegar a ella, se puede hacer fácilmente mediante el menú lateral en “Documentos Publicados”. Esta vista clasifica los documentos según los que son de protocolo / normativa y los que no lo son. Además, ofrece información sobre el documento, como su tipo, su localización y su fecha de publicación. También permite filtrar documentos por su nombre y, si así lo desea el usuario, los puede eliminar. Al intentar eliminar se muestra un diálogo de confirmación para evitar borrados accidentales. Esta confirmación es requerida en todos los botones de eliminación de toda la aplicación.

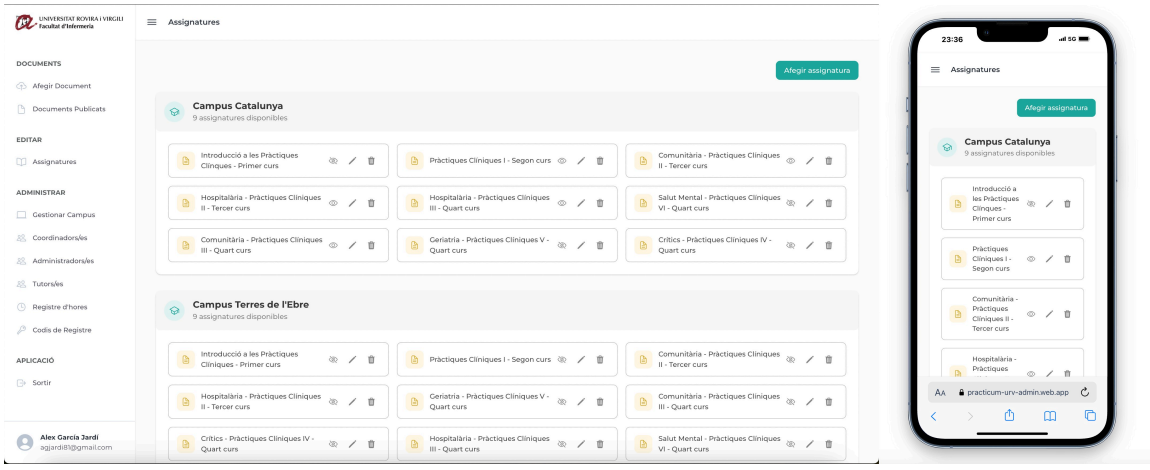


Figura 8. Aplicación gestión – Gestión de asignaturas

Esta vista muestra todas las asignaturas disponibles de todos los campus. Se puede llegar a ella mediante el menú lateral en “Asignaturas”. De hecho, permite visualizar, crear, modificar y borrar las diferentes asignaturas. Aquellos campos como el nombre y el curso de una asignatura pueden modificarse directamente en esta vista dándole al icono del lápiz. Para modificar datos más específicos de la asignatura, se puede seleccionar una de ellas y se abrirá otra ventana para poder modificar dichos detalles.

Cuando el usuario selecciona “Añadir asignatura” se abre un diálogo donde el usuario puede introducir los campos para crear una asignatura.

Además, el usuario también puede cambiar la visibilidad de una asignatura con un solo clic, pulsando el botón representado por un ojo. Esto tendrá efecto en la aplicación de los tutores, pues la asignatura estará bloqueada o disponible en función de la visibilidad que se desee.

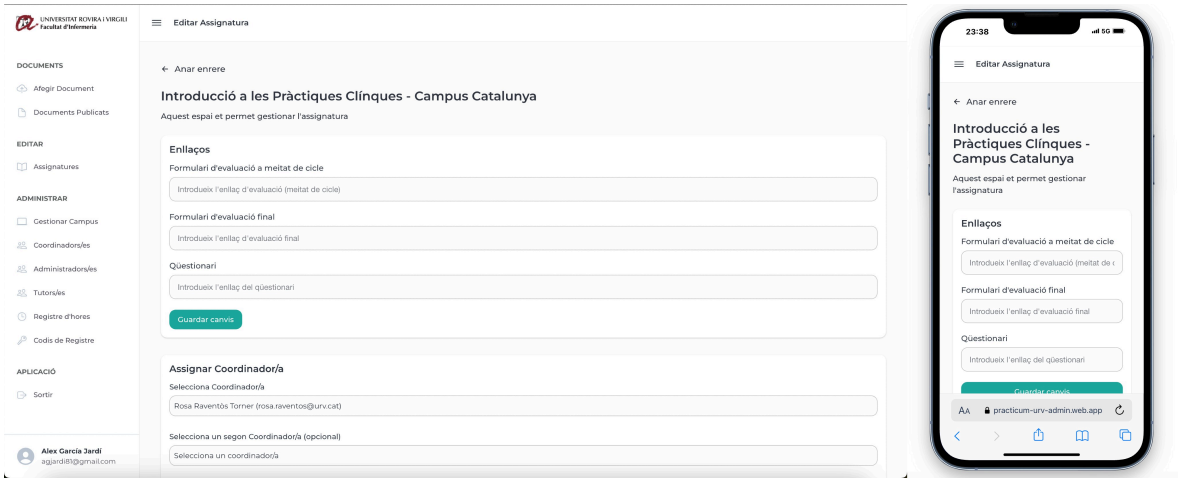


Figura 9. Aplicación gestión – Gestionar una asignatura

Esta ventana es la que se abre cuando se selecciona a una asignatura en la sección de "Asignaturas" del menú lateral. Aquí el usuario puede modificar los enlaces de los formularios de la asignatura, asignar los coordinadores, y a continuación, hay una sección para crear o cambiar la contraseña de la asignatura (a las contraseñas se les debe asignar una fecha de expiración).

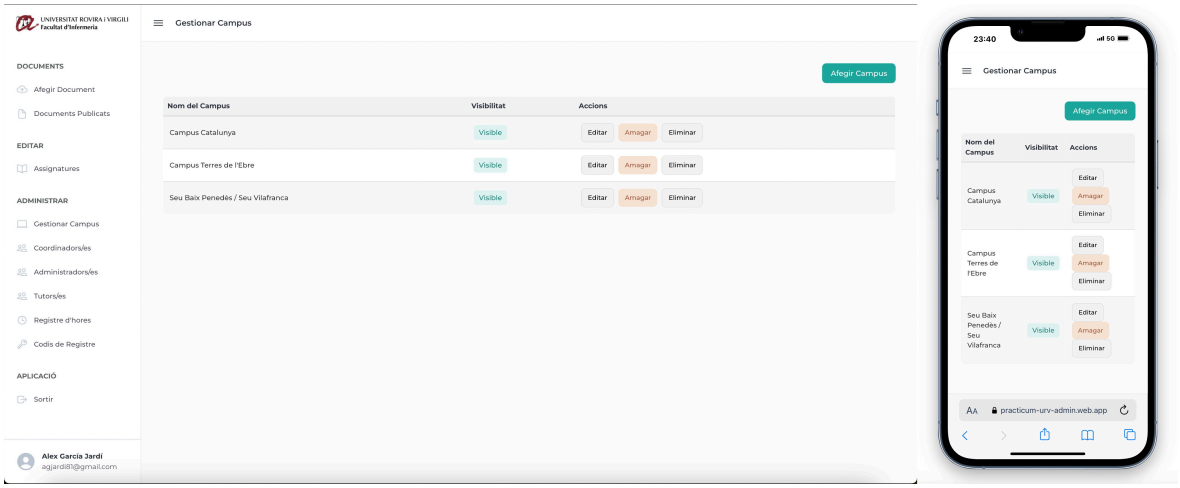


Figura 10. Aplicación gestión – Gestionar campus

Esta vista permite crear, modificar y borrar los campus. Además, permite cambiar su visibilidad. Se puede acceder a ella pulsando en "Gestionar Campus" en el menú lateral.

Cuando se selecciona "Añadir campus" o se procede a modificar alguno, se abre un diálogo para realizar estas acciones con sus correspondientes campos. Estos campos, además, al igual que en toda la aplicación, son validados y el sistema muestra un mensaje cuando hay algún error.

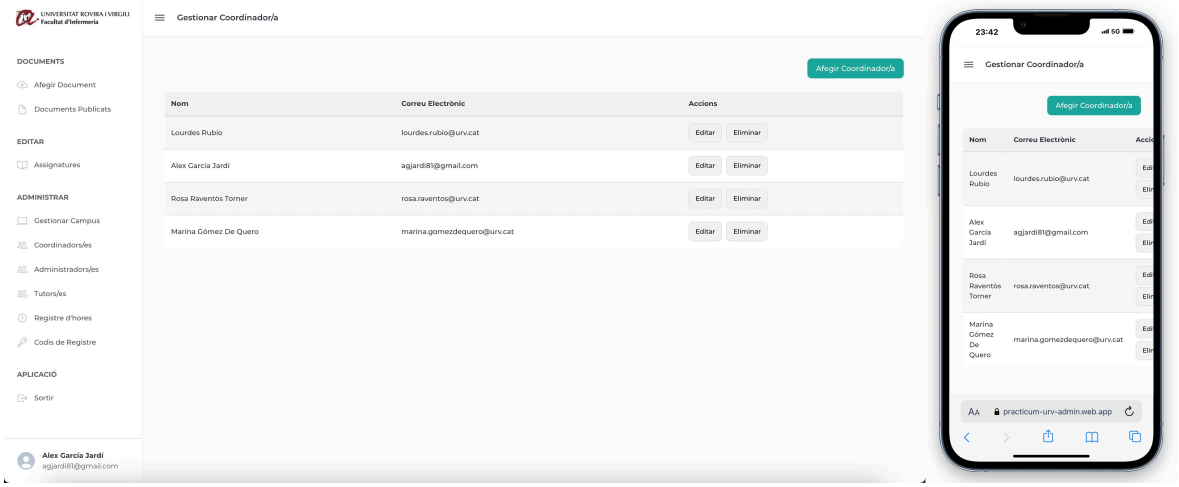


Figura 11. Aplicación gestión – Gestionar coordinadores

En esta vista se puede visualizar, crear, modificar y borrar los diferentes coordinadores. Crear un coordinador es necesario si en la vista de la *Figura 9* se quiere asignar uno de ellos. Para llegar a esta vista, se puede hacer fácilmente mediante el menú lateral en "Coordinadores/as".

Esta vista también abre un diálogo de creación y edición si el usuario quiere realizar alguna de esas dos acciones.

Aunque pareciera que la versión móvil no está adaptada, no es el caso. Para este tipo de tablas con muchas columnas se ha implementado un "scroll" horizontal, de forma que el usuario podrá deslizarse por las diferentes columnas de las diferentes tablas de la aplicación deslizándolas con el dedo.

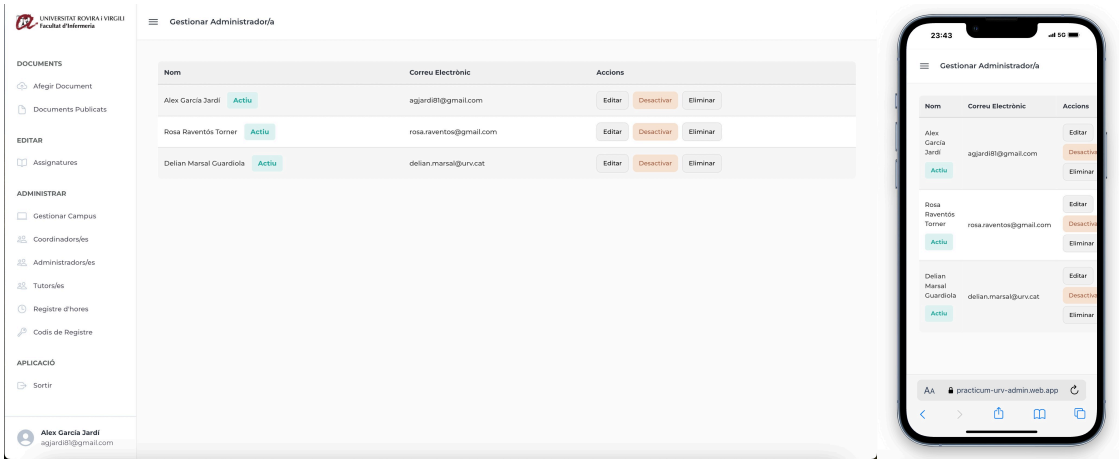


Figura 12. Aplicación gestión – Gestionar administradores

Esta vista permite visualizar, modificar y borrar a los administradores. Para llegar a ella, se puede hacer fácilmente mediante el menú lateral en "Administradores/as". La creación de un **Administrador** lo debe gestionar un súper administrador. Esto es debido a las limitaciones de Firebase, pues como estamos en un entorno gratuito, no se pueden crear usuarios manteniendo la sesión del usuario actual activa. Para ello, se necesitarían custom claims, las cuales son de pago.

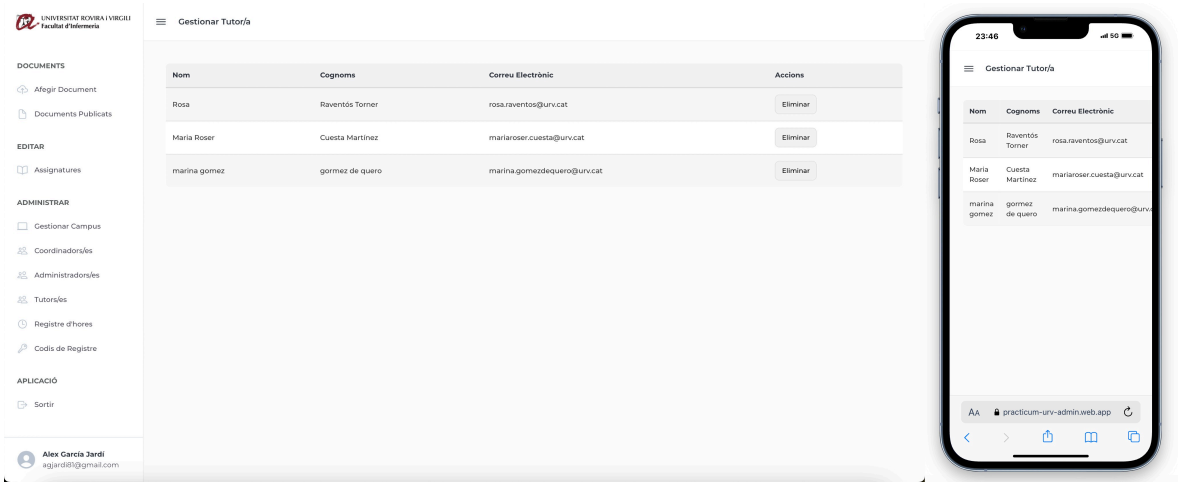


Figura 13. Aplicación gestión – Gestionar tutores

Esta vista permite al **Administrador** supervisar qué Tutores se dieron de alta en el sistema y, en caso de que se detecte un acceso no autorizado, también podrá eliminarlo. Para llegar a esta vista, se puede hacer fácilmente mediante el menú lateral en “Tutores/as”.

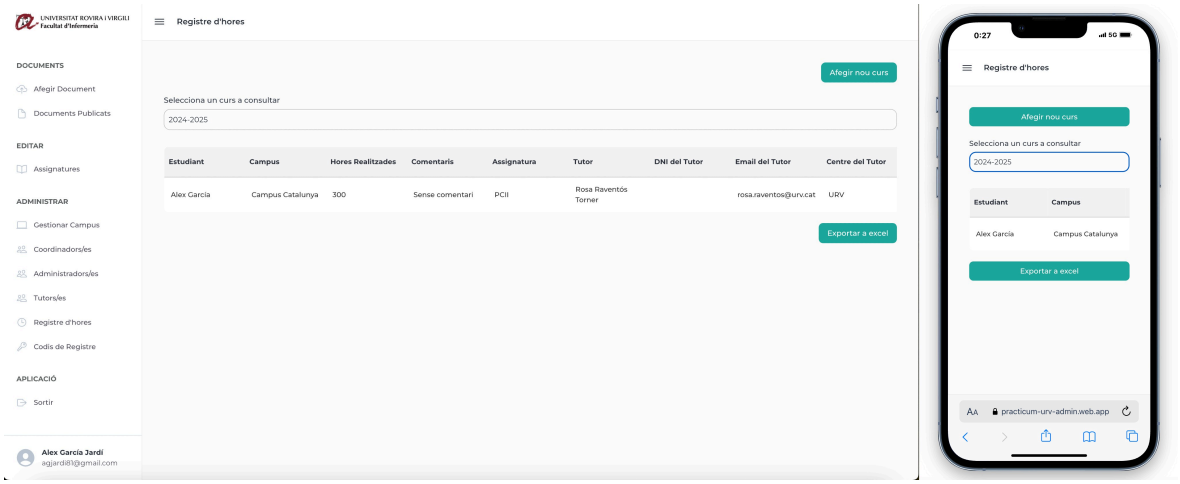


Figura 14. Aplicación gestión – Consultar registro de horas

En esta vista el **Administrador** puede visualizar, por curso lectivo, todos los registros de horas que los Tutores académicos han enviado. Además, para una mejor gestión, se ha habilitado un botón para exportar estos registros en formato Excel. Para llegar a esta vista, se puede hacer fácilmente mediante el menú lateral en “Registro de horas”.

Para añadir un nuevo curso lectivo y así habilitar ese curso también a los Tutores académicos, se puede hacer pulsando el botón “Añadir curso”, el cual abrirá un diálogo de creación. El curso debe tener un formato correcto para ser creado (YYYY-YYYY), en caso contrario se notificará. Por defecto, cuando se abre el diálogo, el sistema automáticamente sugiere el próximo curso lectivo (respecto al último disponible).

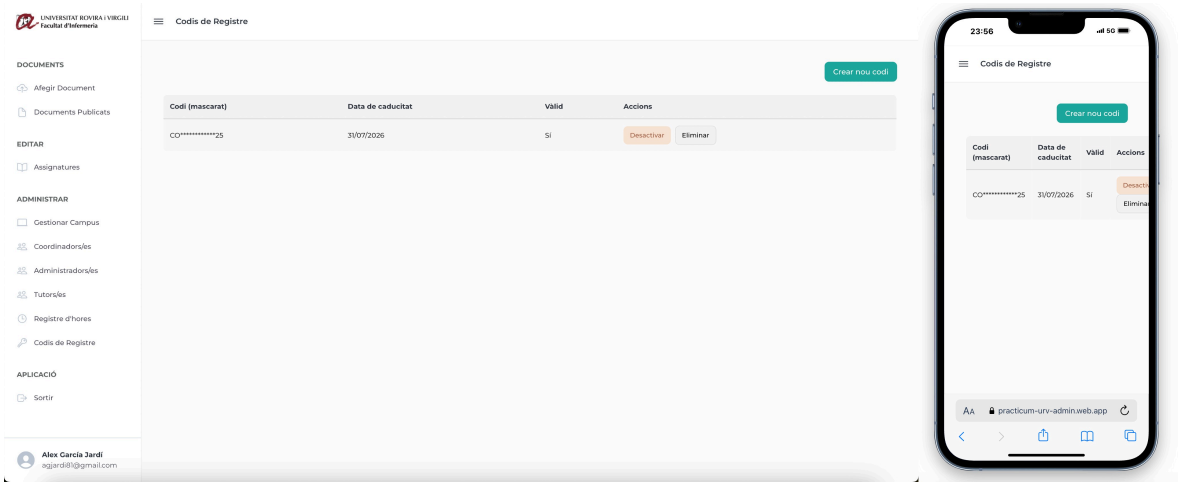


Figura 15. Aplicación gestión – Gestionar códigos de registro

En esta vista el **Administrador** podrá gestionar los diferentes códigos de registro. Para acceder a él se puede hacer desde el menú lateral en “Códigos de registro”.

Permite crear, desactivar, activar y borrar los códigos. Además, como el código de registro es muy sensible en el sistema, este código se guardará y se visualizará enmascarado.

Para crear uno, el **Administrador** puede pulsar el botón “Crear nuevo código” donde se le pedirá el código (debe tener una longitud mínima de 5 caracteres) y la fecha de expiración del código.

Si no hay ningún código de registro activo, ningún Tutor clínico podrá registrarse en el sistema.

6.2.2 Aplicación de tutores

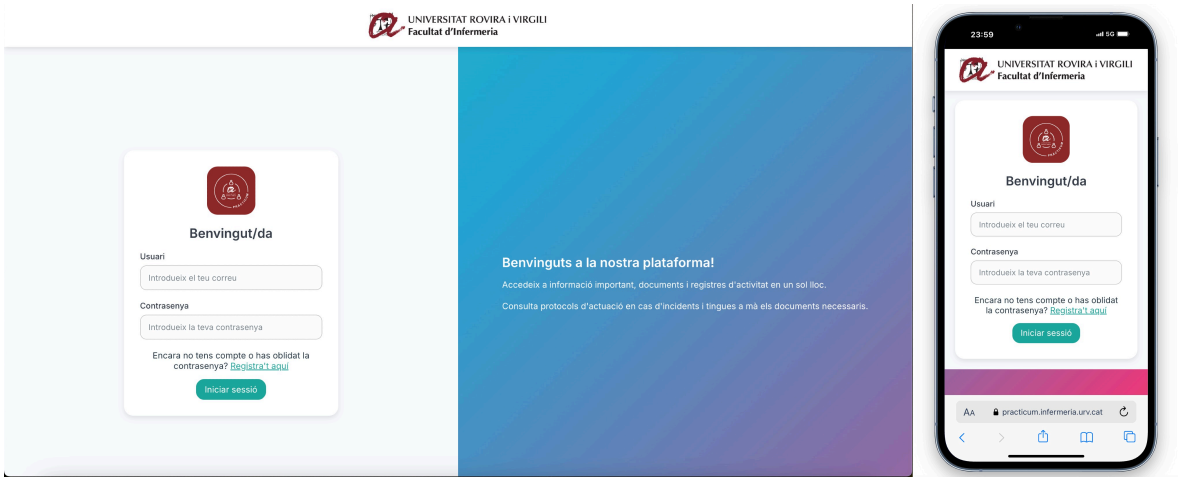


Figura 16. Aplicación Tutores – Inicio de sesión

Esta es la vista principal. Aquí el **Tutor** puede iniciar sesión mediante correo electrónico y contraseña. En el caso de no estar registrado, puede hacerle clic en "Registrarse aquí" para navegar a la vista de registro.

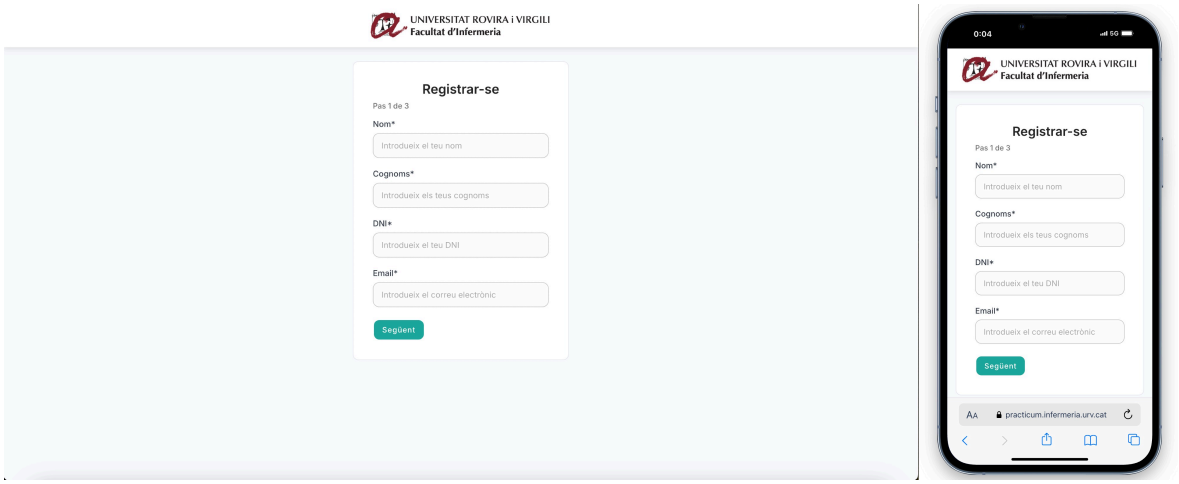


Figura 17. Aplicación Tutores – Registro

Esta es la vista donde el **Tutor** puede registrarse. Esta vista contiene 3 pasos, en cada paso se introducen diferentes campos. En concreto, datos personales, datos de su lugar de trabajo y credenciales (contraseña que utilizará para acceder y el código de registro), respectivamente.

El usuario puede navegar por los diferentes pasos, puede ir tanto hacia atrás como hacia adelante.

Si el código de registro proporcionado es válido, se ha introducido todos los campos y estos contienen un formato válido (por ejemplo, el DNI y el correo electrónico son correctos), entonces el usuario se creará con éxito.

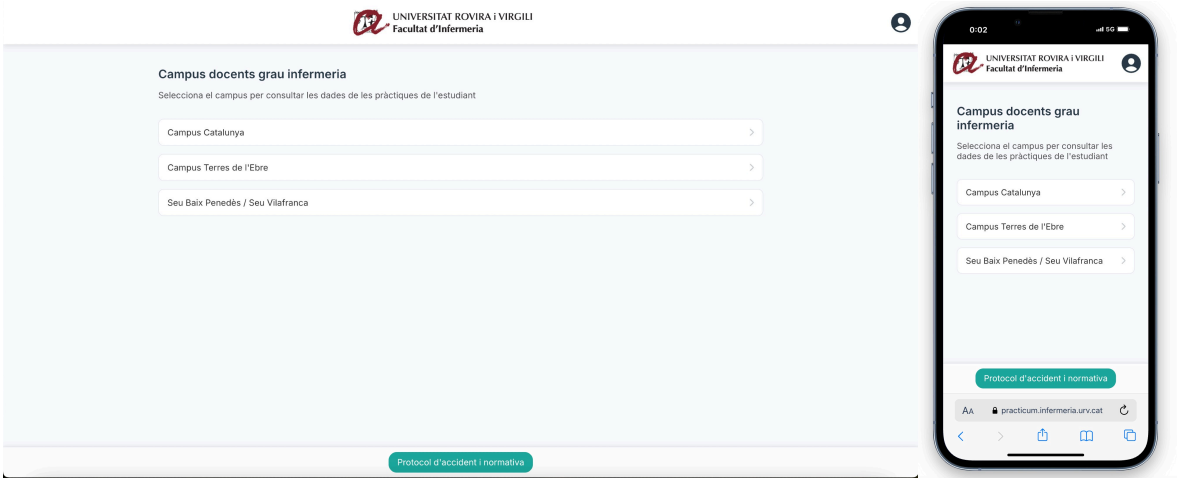


Figura 18. Aplicación Tutores – Listado de campus

Tras iniciar sesión, el usuario es redirigido a una vista que muestra los campus disponibles. En caso de que alguno de ellos esté oculto, quedará bloqueado de tal forma que no se podrá acceder a él. Para acceder a las diferentes asignaturas, hay que seleccionar una opción.

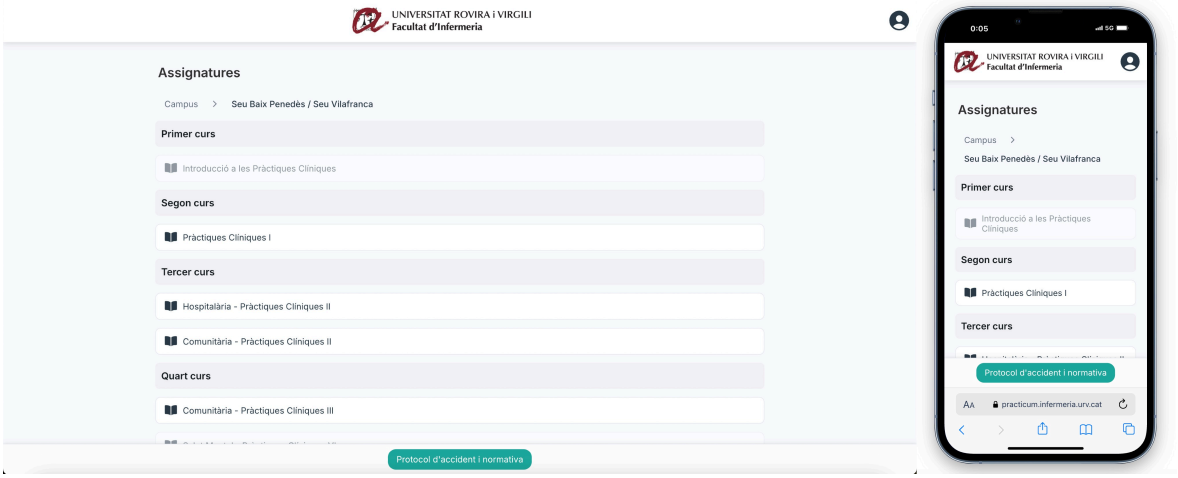


Figura 19. Aplicación Tutores – Listado de asignaturas

Cuando el usuario selecciona un campus, se muestra un listado con las diferentes asignaturas disponibles (tanto ocultas como no ocultas).

Cuando el usuario selecciona una asignatura, si este nunca había entrado a la asignatura, se le pedirá una contraseña. Una vez introducida, si es correcta, el usuario tendrá acceso a la asignatura sin necesidad de especificarla de nuevo hasta la fecha de expiración de la contraseña introducida.

Además, a partir de esta vista se habilita un método alternativo de navegación por la aplicación (ubicado debajo del título).

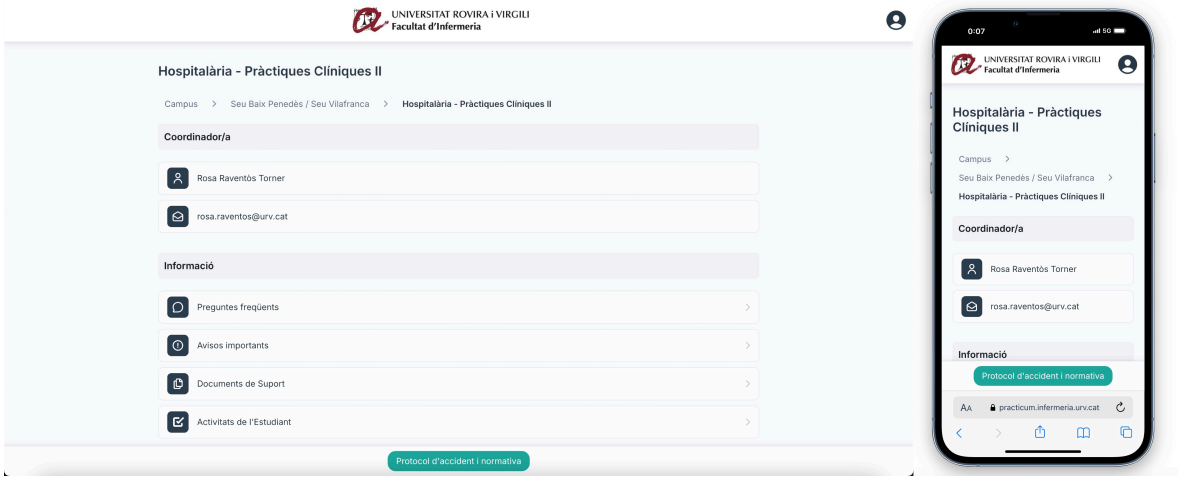


Figura 20. Aplicación Tutores – Contenido de una asignatura

Tras seleccionar una asignatura, se mostrará una vista que mostrará el contenido básico de la asignatura: su coordinador/a o coordinadores (puede haber hasta dos) y diferentes secciones para acceder a las preguntas frecuentes, avisos, documentos, actividades y registro de horas.

Además, contiene 3 elementos los cuales en realidad son enlaces de Microsoft Forms. Cuando el usuario selecciona uno, se le avisa de que se le redirigirá. En caso de que confirme la acción se abrirá una nueva ventana del navegador con el formulario deseado.

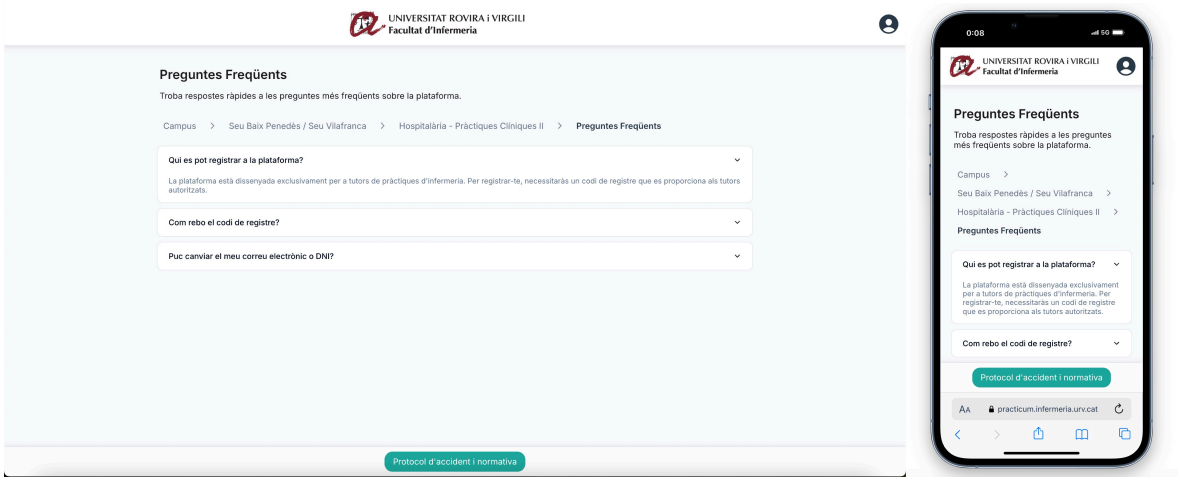


Figura 21. Aplicación Tutores – FAQ

En esta vista el **Tutor** podrá ver las preguntas más frecuentes en caso de que tenga alguna duda sobre el uso de la aplicación.

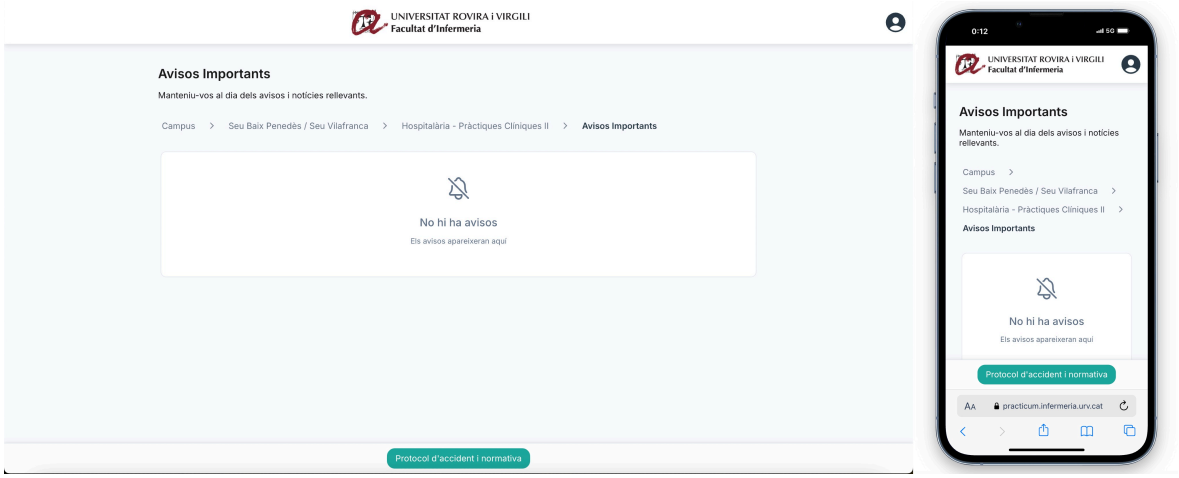


Figura 22. Aplicación Tutores – Avisos importantes

En esta vista el **Tutor** podrá consultar si hay algún aviso importante que deba tener en cuenta; en este caso no hay ningún aviso activo.

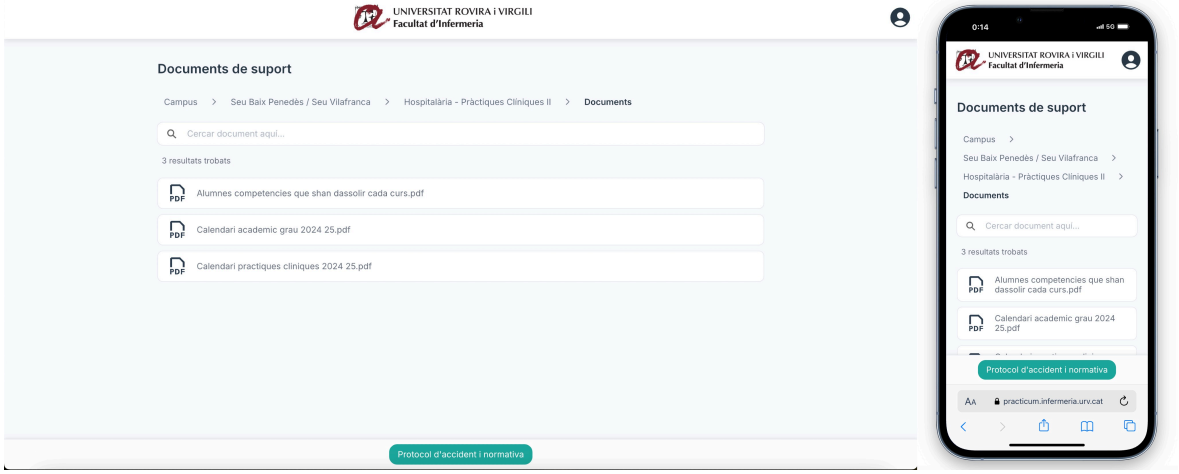


Figura 23. Aplicación Tutores – Documentos de soporte

En esta vista el **Tutor** podrá visualizar los documentos de soporte disponibles para el campus y asignatura actuales. Además, podrá filtrarlos por nombre.

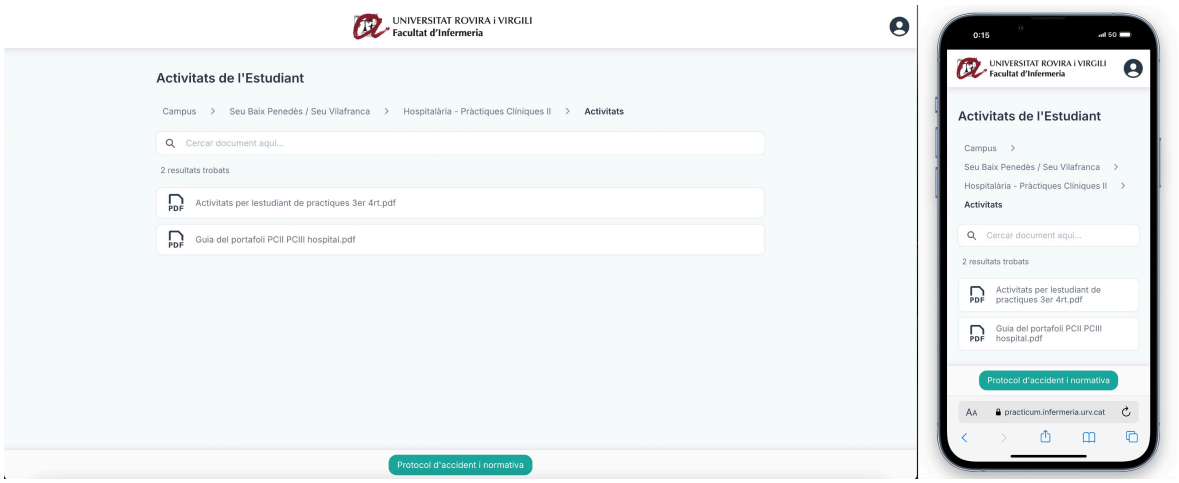


Figura 24. Aplicación Tutores – Actividades del estudiante

En esta vista el **Tutor** podrá visualizar las actividades del estudiante disponibles para el campus y asignatura actuales. También podrá filtrarlos por nombre.

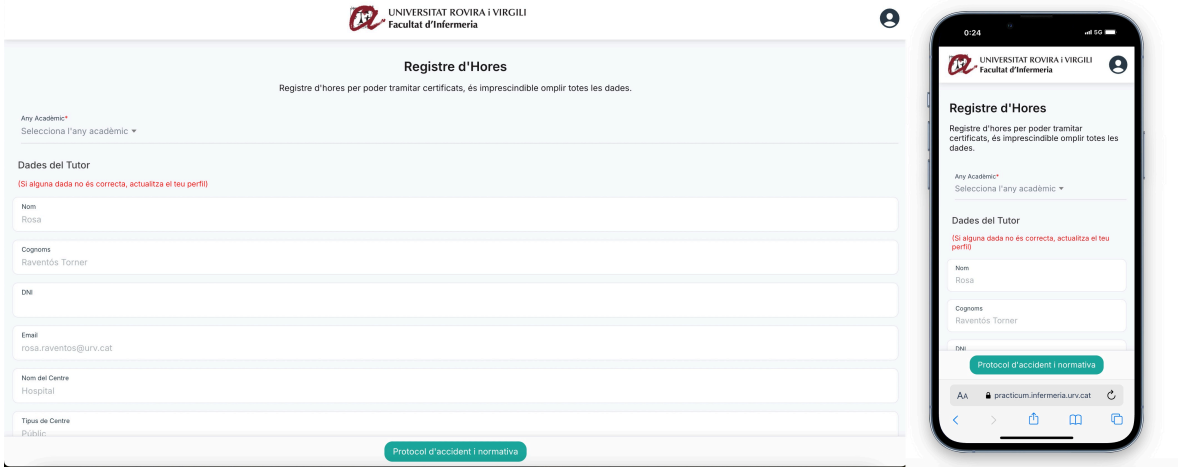


Figura 25. Aplicación Tutores – Registro de horas

Desde la vista de detalles de la asignatura también se puede acceder al registro de horas. Aquí el **Tutor** puede introducir la cantidad de horas realizadas por el alumno en el centro de prácticas, junto a un comentario, si así lo deseara.

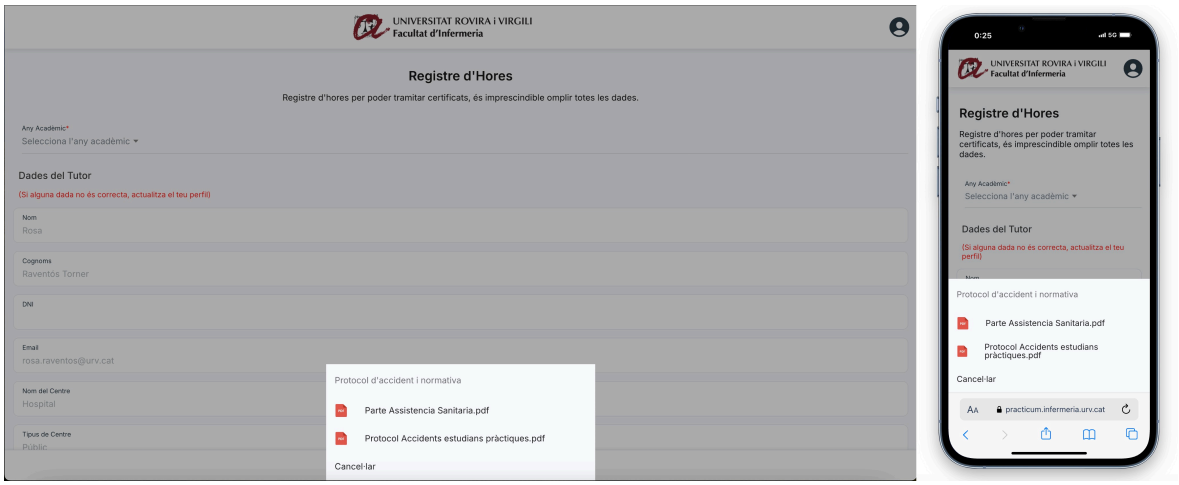


Figura 26. Aplicación Tutores – Visualización de los Protocolos / Normativas

El **Tutor** tendrá siempre accesible los protocolos / normativas en caso de cualquier accidente en la barra inferior de la aplicación.

Al pulsar el botón de "Protocolo / Normativa" se desplegará un cuadro donde podrá navegar rápidamente para consultar estos documentos.

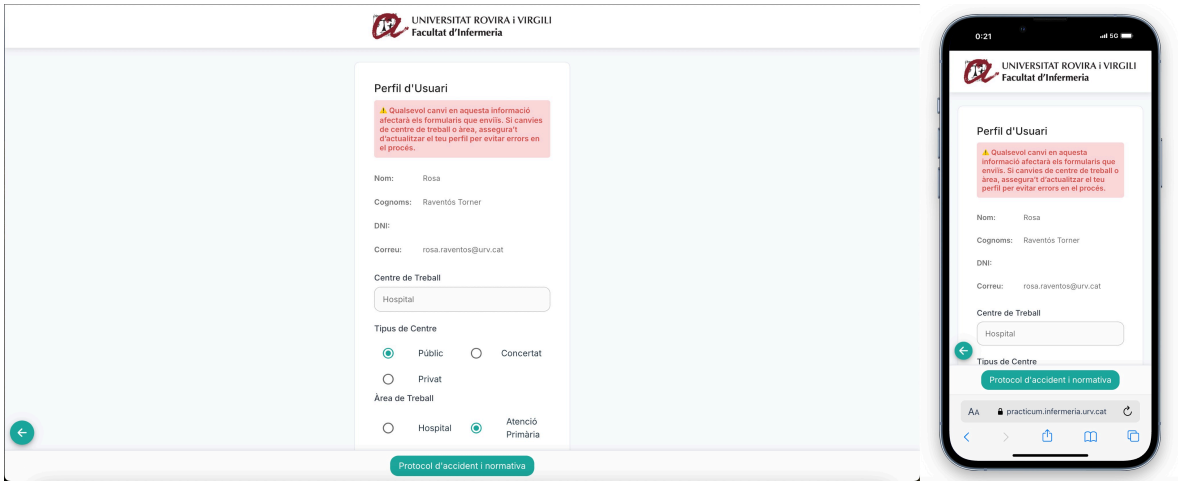


Figura 27. Aplicación Tutores – Perfil del usuario

Si el **Tutor** pulsa el icono de su usuario en la barra superior de la aplicación, se le abrirá una vista como la siguiente. En dicha vista podrá modificar ciertos datos referentes al centro de trabajo del **Tutor** y podrá cerrar su sesión, si así lo deseara.

6.3 Diseño de la arquitectura de Firebase

En esta sección se describe la configuración de Firebase empleada en el proyecto, organizada en tres grandes bloques.

6.3.1 Diseño de la base de datos en Firestore

En esta sección se detalla la arquitectura de Firestore, incluyendo sus colecciones y subcolecciones, la lógica de relaciones que sostiene el modelo de datos y las reglas de seguridad que protegen la información.

6.3.1.1 Estructura de colecciones y campos

Firestore es una base de datos documental (NoSQL). Por lo tanto, se ha decidido mostrar su estructura en forma de árbol:

```
/
│
├─ AccessControl
│   └─ {userId}
│       └─ {campusId}
│           └─ {subjectId}
│               └─ expirationDate      (timestamp)
│
├─ Admins
│   └─ {userId}
│       ├─ active            (boolean)
│       ├─ email             (string)
│       └─ name              (string)
│
├─ Coordinators
│   └─ {coordinatorId}
│       ├─ email             (string)
│       └─ name              (string)
│
├─ Mentors
│   └─ {userId}
│       ├─ centerName        (string)
│       ├─ centerType        (string)
│       ├─ dni               (string)
│       ├─ email             (string)
│       ├─ name              (string)
│       ├─ surnames          (string)
│       └─ workAreas         (string)
│
├─ RegistrationCodes
│   └─ {codeId}
│       ├─ codeHash          (string)
│       ├─ codeMasked        (string)
│       ├─ expiresAt         (timestamp)
```

```
        └─ valid                (boolean)

├─ Campus
   └─ {campusId}
      ├─ locked                 (boolean)
      ├─ name                   (string)
      └─ subjects               (subcolección)
         └─ {subjectId}
            ├─ coordinator_id               (string)
            ├─ course                       (string)
            ├─ locked                       (boolean)
            ├─ name                         (string)
            ├─ url_evaluation_tutor         (string)
            ├─ url_midterm_eval_tutor       (string)
            ├─ url_survey                   (string)
            └─ PasswordHistory              (subcolección)
               └─ {historyId}
                  ├─ createdAt                      (timestamp)
                  ├─ encryptedPassword              (string)
                  ├─ hashedPassword                 (string)
                  ├─ expiryDate                     (timestamp)
                  └─ isRevoked                      (boolean)

└─ TimeRecord
   └─ {academicYear}
      └─ mentors                (subcolección)
         └─ {userId}
            └─ students                 (subcolección)
               └─ {studentId}
                  ├─ campus             (string)
                  ├─ fullName           (string)
                  ├─ hoursCompleted     (number)
                  ├─ idDocument         (string)
                  ├─ notes              (string)
                  └─ subject            (string)
```

La estructura de Firestore se ha pensado para maximizar la eficiencia de las consultas y mantener un modelo fácil de entender.

En este escenario, separar los datos por colecciones raíz según el rol (**Admins**, **Mentors**) simplifica tanto la lógica en el Frontend como las reglas de seguridad.

Los Tutores (**Mentors**) y los Administradores (**Admins**) son realmente los únicos usuarios registrados en el sistema. Eso implica que ellos disponen de su propio identificador generado por Firebase Authentication. Esta división de ambas colecciones es muy útil, pues se pueden definir reglas de Firebase en base a ellos. Si un usuario con cierto identificador tiene una correspondencia en la colección **Admins**, entonces se le considera Administrador; lo mismo sucede con la colección **Mentors**, pero este está limitado a las reglas de los Tutores. Esto implica que aquel usuario considerado como Tutor tendrá un acceso limitado en el sistema.

En cambio, la colección **Coordinators** tiene identificadores aleatorios. Esto es porque en el sistema el único uso que tienen los Coordinadores es informar quién coordina una asignatura. Los únicos que tienen acceso a la aplicación son aquellos Coordinadores que son Administradores.

Para el acceso a las asignaturas se ha definido la colección **AccessControl**, la cual permite comprobar con una sola lectura si un Tutor aún tiene permiso para acceder a las diferentes asignaturas. Esta estructura ayuda a evitar búsquedas globales y reduce el número de lecturas. Esto además es importante porque, especialmente en infraestructuras en la nube, cada operación cuenta sobre la facturación.

La base de datos incluye una colección llamada **TimeRecord** que reúne, en documentos independientes por año académico (por ejemplo "2024-2025"), toda la información de registro de horas. Debajo de cada año aparece la subcolección **mentors**, y dentro de esta, la subcolección **students** con los datos y horas realizadas por cada estudiante. Este diseño se pensó así por cómo era el flujo de trabajo: primero se selecciona el curso, luego el Tutor y finalmente el estudiante concreto (se rellenan sus datos). Además, esta estructura nos permite restringir las escrituras, pues se puede definir una regla en Firebase para que ningún Tutor pueda escribir o sobrescribir en los registros de otro Tutor.

En la colección **Campus** se han definido ciertos campos para controlar la visibilidad del Campus y su nombre. Además, contiene una subcolección **subjects**, donde se almacenan todas las asignaturas; en ella se definen datos como el **identificador** del Coordinador (con el identificador es suficiente, pues se dispone de la colección **Coordinators**). También se definen los formularios, el nombre de las asignaturas, el curso al que pertenecen, entre otros.

Además, en la colección **subjects**, se definió la subcolección **PasswordHistory** para tener un histórico de las contraseñas. Esto se diseñó así para mejorar la trazabilidad. Siempre que se genera o revoca una contraseña, queda un registro cronológico. Además, esto también se utiliza para evitar reutilizar contraseñas (mejora la seguridad ante posibles filtraciones).

También se ha incorporado una capa de seguridad para proteger datos sensibles. Se utilizó Hashing con Salt en las **contraseñas** y **códigos de registro**. Esto es importante porque, de esta manera, se protegen las claves. Se debe tener en cuenta que en este entorno no se dispone de un Backend tradicional (como por ejemplo una REST API). En este escenario, es el Frontend el encargado de hacer las validaciones (comparar que el hash coincide). Por lo tanto, las claves deben estar bien protegidas para evitar accesos no autorizados.

Además, las contraseñas de las asignaturas tienen adicionalmente una versión donde no está hasheada, sino cifrada. Esto es especialmente útil porque hay muchas asignaturas y no se puede pretender que los Administradores se acuerden de todas las contraseñas. El motivo de utilizar hashing y una versión cifrada al mismo tiempo viene porque se dispone de dos aplicaciones: en la aplicación del Tutor no hay que almacenar claves criptográficas (hay que recordar que no se dispone de un Backend tradicional, por lo que se podría interceptar esta clave de cifrado en el navegador) entonces esta aplicación usará el hash para comparar. En cambio, la aplicación de gestión sí conoce esta clave para descifrar el contenido, pero esta aplicación no será de dominio público, por lo que la clave está en un lugar más seguro.

En cambio, el código de registro no se cifró, se decidió acompañar la versión hasheada con una versión enmascarada. Esto se hizo porque hay que recordar que este código es la pieza que realmente nos permite restringir quién se registra en el sistema. Dada la naturaleza del código de registro, la cual es que se usará muy probablemente solo un código para todo un curso lectivo (el mismo para todos), este código es fácil de recordar por los Administradores. El enmascarado tiene como fin mostrar unas pistas de cómo empezaba y cómo terminaba el código por si de casualidad lo olvidaron. En el caso de que lo olviden, podrían simplemente generar un código nuevo y revocar el anterior.

Otras partes de la aplicación donde sí que se utilizó criptografía simétrica (como con las contraseñas de las asignaturas) fue en todos los datos personales; tanto de los Tutores como de los Estudiantes (aquellos campos que introduce el Tutor en el formulario de registro de Horas). Esto se hizo con el fin de anonimizar los datos.

6.3.1.2 Reglas de seguridad en Cloud Firestore

A continuación, se mostrará la configuración realizada en las reglas de Firestore para restringir los accesos no autorizados:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

      // Permitir acceso solo a usuarios autenticados que estén en la
colección Admins
      function isAdmin() {
      return
exists(/databases/$(database)/documents/Admins/$(request.auth.uid))
      &&
get(/databases/$(database)/documents/Admins/$(request.auth.uid)).data.act
ive == true;
    }

    // Mira si el usuario inició sesión
    function isAuthenticated() {
      return request.auth.uid != null;
    }

    // Solo los Admins pueden leer/escribir en cualquier documento
    match /{document=**} {
      allow read, write: if isAdmin();
    }

            match /Admins/{userId} {
      // Permitir lectura solo si el usuario está autenticado y su UID
coincide
      allow read: if request.auth != null && request.auth.uid == userId;
            }

    // Permite a un usuario leer las claves de registro
            match /RegistrationCodes/{document=**} {
                  allow read: if true;
            }

    // Permite a un usuario leer y escribir solo en su propio documento
    match /Mentors/{mentorId} {
```

```
      allow read, write: if isAuthenticated() && request.auth.uid ==
mentorId;
    }

      // Solo permite al usuario escribir en su propio AccessControl
    match /AccessControl/{userId} {
      allow read, write: if isAuthenticated() && request.auth.uid ==
userId;
    }

            // Cualquier persona puede leer los años académicos
            match /TimeRecord/{academicYear} {
                  allow read: if isAuthenticated();

                  // Los mentores solo pueden escribir en su campo
                  match /mentors/{mentorId} {
            allow read, write: if request.auth.uid == mentorId;

        match /students/{studentId} {
                              allow read, write: if request.auth.uid ==
mentorId;
        }
      }
            }

      // Control de acceso a asignaturas
    match /Campus/{campusId} {
      allow read: if isAuthenticated();

      match /subjects/{subjectId} {
            allow read: if isAuthenticated();

        match /PasswordHistory/{passwordDoc=**} {
          allow read: if isAuthenticated();
            }
      }
    }

    match /Coordinators/{allPaths=**} {
      allow read: if isAuthenticated();
    }
  }
}
```

Esta configuración de seguridad de Firestore está pensada para cumplir con el principio de mínimo privilegio, de tal forma que cada usuario solo acceda a aquello que realmente necesita. Para ello, se definieron dos funciones para separar los dos roles: una de ellas comprueba si el usuario es un Administrador activo y la otra verifica simplemente si está autenticado (es decir, es un Tutor).

Por defecto, solo los Administradores deberían poder leer o modificar cualquier documento de la base de datos. Por lo tanto, se definió una regla para que el Administrador pueda realizar cualquier operación en la base de datos.

A continuación de esta regla, se definen diferentes reglas que están diseñadas específicamente para el Tutor. Algunas de ellas son bastante generales, pero muchas otras están pensadas para que un mismo usuario sea el único que pueda modificar ciertos registros.

De esta forma, cada Tutor puede gestionar únicamente su propio perfil y los registros de sus estudiantes, sin ver ni tocar los datos de otros Tutores. De igual modo, los usuarios que ya están autenticados pueden consultar la información de los campus, asignaturas y coordinadores y todos aquellos datos necesarios para tener una buena navegación por la aplicación de Tutores.

Por último, los códigos de registro se mantienen públicos para que cualquier persona pueda consultarlos y darse de alta como Tutores (en el caso de introducir un código de registro correcto).

6.3.2 Diseño de Firebase Storage

Firebase Storage ha sido muy importante en todo el desarrollo de las aplicaciones, de hecho, es la parte más importante.

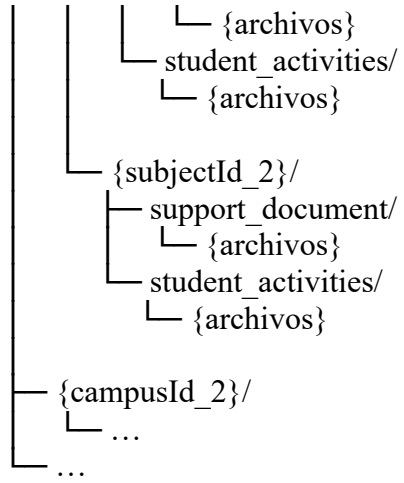
Es el repositorio central donde se alojan todos los documentos, protocolos, imágenes y cualquier otro recurso descargable o compartido entre tutores y coordinadores. Dichos documentos fueron los que realmente impulsaron la creación de este proyecto, pues se necesitaba tener estos documentos disponibles para los Tutores en tiempo real.

A continuación, se expone la organización de Firebase Storage, especificando la estructura de los ficheros y las políticas de acceso aplicadas.

6.3.2.1 Organización de Firebase Storage

Para una mejor visualización de la jerarquía, se ha representado en forma de árbol:

```
/
├── QuickAccess/
│   └── {archivos}
│
├── General/
│   ├── General/
│   │   ├── support_document/
│   │   │   └── {archivos}
│   │   └── student_activities/
│   │       └── {archivos}
│   │
│   ├── {subjectId_1}/
│   │   ├── support_document/
│   │   │   └── {archivos}
│   │   └── student_activities/
│   │       └── {archivos}
│   │
│   ├── {subjectId_2}/
│   │   ├── support_document/
│   │   │   └── {archivos}
│   │   └── student_activities/
│   │       └── {archivos}
│   └── …
│
├── {campusId_1}/
│   ├── General/
│   │   ├── support_document/
│   │   │   └── {archivos}
│   │   └── student_activities/
│   │       └── {archivos}
│   │
│   ├── {subjectId_1}/
│   │   ├── support_document/
```



La estructura se divide básicamente en 4 niveles de alcance, definidos para evitar copias innecesarias y mantener una única fuente de verdad para cada recurso:

- **Global-Global (General/General/)**: Archivos compartidos en cualquier campus y cualquier asignatura.
- **Global-Asignatura (General/{subjectId}/)**: Archivos compartidos por una asignatura concreta, pero en todos los campus.
- **Campus-Global ({campusId}/General/)**: Archivos comunes en todas las asignaturas de un campus concreto.
- **Campus-Asignatura ({campusId}/{subjectId}/)**: Archivos de un campus y asignatura específicos.

Como se puede ver, los 4 niveles van desde un nivel más general hasta el nivel más individual, en el que solo se encontrará ese recurso en una asignatura específica de un campus concreto.

En estos 4 niveles de enlace, se observa que además hay una tercera capa donde se indica el tipo de documento (actividad del alumno y documento de soporte). Entonces, el sistema realmente lo que hace es almacenar los documentos en base al identificador del campus (si no especifica, es el General), su identificador de la asignatura (si no se especifica, es el General) y el tipo de documento.

Además, se puede observar en la jerarquía la presencia de **QuickAccess**: este directorio almacena directamente un conjunto de documentos, los cuales son precisamente los que se muestran en la sección de Protocolos / Normativas.

Todos estos documentos en el sistema están organizados por orden alfabético, por lo que se les suele añadir al nombre un número al inicio para que el Administrador pueda ordenarlos a su gusto.

6.3.2.2 Reglas de seguridad en Firebase Storage

Se han definido las siguientes reglas de seguridad para controlar el acceso a los archivos:

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {

    // Función para saber si el usuario está autenticado
    function isAuthenticated() {
      return request.auth != null;
    }

    // Aplica a todos los ficheros
    match /{allPaths=**} {

      // Permitir leer y escribir a cualquier usuario autenticado
      allow read, write: if isAuthenticated();

    }
  }
}
```

En este escenario, solo los usuarios autorizados podrán leer y modificar archivos.

En un principio se valoró restringir la escritura únicamente a Administradores para evitar que los tutores suban o modifiquen documentos. No obstante, el uso del plan gratuito de Firebase impone limitaciones en las reglas de Storage (no permite reglas tan finas basadas en rutas complejas o custom claims).

Por ello, y conforme al acuerdo de este proyecto, se ha optado por la regla global de autenticación, asumiendo este compromiso hasta contar con un plan que soporte un control de acceso más detallado (ya sea utilizar el plan de pago de Firebase o incluso desarrollar un Backend tradicional).

6.3.3 Firebase Authentication

Firebase Authentication se utiliza como sistema de gestión de usuarios y acceso en ambas aplicaciones. Los Tutores y Administradores se registran e inician sesión con correo y contraseña, y Firebase se encarga de almacenar y renovar de forma segura sus credenciales sin necesidad de un Backend propio.

Gracias a este sistema de autenticación, las reglas de Firestore y Storage pueden basarse en la identidad del usuario para permitir o denegar operaciones. Cada vez que un usuario intenta leer o escribir datos o archivos, su token de Firebase se verifica antes de aplicar las reglas, de modo que solo quienes han iniciado sesión y cumplen con el rol adecuado (Administrador o Tutor) pueden acceder a los recursos correspondientes.

7 Implementación

Durante el desarrollo de ambas aplicaciones, se ha priorizado que el código cumpliera criterios de modularidad, calidad y reutilización. En esta sección se explica cómo se ha organizado el código y cómo funcionan internamente las aplicaciones. Ambos proyectos se han generado de forma independiente con Vue.js y se han compilado con Vite, un Framework moderno que ofrece tiempos de arranque rápidos y optimiza el rendimiento en producción.

7.1 Reutilización de componentes

La reutilización de componentes ha sido una tarea muy relevante en el proyecto. Se han extraído elementos frecuentes, como por ejemplo tablas, tarjetas informativas, diálogos de confirmación, … en componentes genéricos que reciben sus datos y comportamiento mediante parámetros. Gracias a dividir estos componentes, se obtienen módulos con el mismo diseño en diferentes partes de la aplicación, que utilizan diferentes parámetros. De este modo, una misma pieza de UI puede emplearse en distintas vistas sin duplicar código y manteniendo una apariencia uniforme.

Además, antes de empezar a desarrollar se han diseñado prototipos en Figma y se ha validado la experiencia de usuario con las entidades coordinadoras. Se ha definido una guía de estilo (paleta de colores, tipografías y espaciados) para asegurar la coherencia visual de todos los componentes. Cada uno de ellos, construido con estos principios, se adapta a diferentes tamaños de pantalla (responsive) y cumple los criterios básicos de accesibilidad.

Del mismo modo, se han creado servicios en TypeScript que centralizan la lógica de negocio; así, cualquier cambio solo requiere actualizar el servicio correspondiente y se propaga automáticamente a toda la aplicación.

7.2 Calidad del código y control de versiones

Cada aplicación tiene su propio repositorio en GitHub, donde se han ido publicando los diferentes avances de ambas aplicaciones. Antes de hacer alguna publicación del código, se ha utilizado SonarQube para analizar métricas de calidad, vulnerabilidades y cumplir con las mejores prácticas en cuanto a calidad del código.

En ambas aplicaciones, la configuración de entornos se ha gestionado con archivos ".env", de manera que las credenciales de Firebase y otro tipo de credenciales o datos sensibles estén en un entorno protegido. Se ha evitado en todo momento que estas credenciales se suban a los repositorios.

7.3 Organización del código

A continuación, se muestra a modo de guía la jerarquía de archivos compartida por ambos proyectos, destacando los elementos más relevantes:

```
/
├── firebase.json
├── src/
│   ├── shared/
│   │   └── utils/
│   │       └── firebaseConfig.ts
│   ├── components/
│   │   ├── layout/
│   │   └── ui/
│   ├── views/
│   ├── router/
│   │   └── index.ts
│   ├── App.vue
│   └── main.ts
```

El archivo **firebase.json** gestiona el despliegue en Firebase Hosting.

El archivo **shared/utils/firebaseConfig.ts** carga los parámetros de Firebase del archivo de entorno (.env) para inicializar el servicio de Firebase en la aplicación.

El directorio **components** contiene todos aquellos elementos que pueden ser reutilizados en la aplicación. En concreto se ha decidido separarlo en dos subcategorías: **layout** contiene todos aquellos elementos que en general están presentes una vez, como por ejemplo la barra lateral. En cambio, en **ui** están aquellos componentes que se reutilizan más a lo largo de la aplicación, como botones o campos.

En el directorio **views** se encuentran las páginas principales de las aplicaciones, entre ellas podría estar, por ejemplo, la vista para subir un documento.

En **router/index.ts** se controlan las rutas de la aplicación y se especifican restricciones para que no se pueda navegar a ellas sin autorización. En caso de que se intente ir a una vista que no se tiene acceso, el **router** redirige al usuario a una ventana que indica que el usuario no está autorizado para estar en esa página.

Por último, **main.ts** monta Vue con el router y Firebase, y **App.vue** actúa como contenedor raíz de las vistas.

7.4 Conexión de Vue con Firebase

Para habilitar la comunicación con Firebase, se ha creado un bloque de configuración donde se agrupan todas las credenciales obtenidas de variables de entorno. Cuando las aplicaciones arrancan, llaman a la librería oficial de Firebase para inicializar los servicios principales (autenticación, base de datos y almacenamiento). A continuación, se muestra un ejemplo simplificado de cómo se realiza dicha inicialización:

```
import { initializeApp }    from "firebase/app";
import { getAuth }          from "firebase/auth";
import { getFirestore }     from "firebase/firestore";
import { getStorage }       from "firebase/storage";
import firebaseConfig       from "./shared/utils/firebaseConfig";

const firebaseApp     = initializeApp(firebaseConfig);
const auth            = getAuth(firebaseApp);
const db              = getFirestore(firebaseApp);
const firebaseStorage = getStorage(firebaseApp);

export { auth, db, firebaseStorage };
```

Este código se ubica en el punto de entrada de la aplicación (**main.ts**) en ambas aplicaciones, de modo que cualquier componente puede importar directamente **auth**, **db** o **firebaseStorage** sin tener que volver a inicializarlos.

La inicialización de estos servicios se realiza a partir de un objeto de configuración (firebaseConfig) que incluye las credenciales necesarias para establecer la conexión.

7.5 Operaciones con Firestore

En ambas aplicaciones, la interacción con Firestore se realiza a través del objeto **db**, el cual ya está inicializado al arrancar la aplicación. Una vez se dispone de este objeto, se pueden llevar a cabo todas las operaciones sobre la base de datos.

Para leer datos, primero se obtiene una referencia a la colección o documento que se va a consultar. Por ejemplo, **collection(db, "Campus")** devuelve la referencia a la colección "Campus". A partir de esa referencia, se puede llamar a la función **getDocs()**, que devuelve un array de documentos almacenados en esa colección. Cada uno de esos documentos se convierte en un objeto JavaScript para facilitar el acceso a sus campos. Durante el desarrollo, esta lógica se ha encerrado en funciones asíncronas de forma que, al completarse, se asigna el resultado a una variable reactiva en Vue, de modo que la interfaz se actualiza automáticamente con los datos obtenidos.

En lo que respecta a la inserción de nuevos registros (por ejemplo, la creación de una asignatura o el registro de horas), el componente que contiene la lógica de inserción recoge los datos del formulario y los transfiere a una función que llama a **addDoc()**. Es importante destacar que, previamente a la invocación de la función **addDoc()**, se realizan validaciones a nivel de cliente. Estas validaciones garantizan que haya coherencia en la base de datos.

En el caso de querer modificar un documento existente, primero se obtiene una referencia al registro que se desea cambiar. A continuación, se validan en el cliente los datos ingresados (por ejemplo, que el nombre no esté vacío o que el formato sea correcto). Si los

campos son válidos, se invoca la función **updateDoc()**, indicando únicamente los campos a modificar, sin alterar el resto de la información del documento.

El proceso de eliminación es similar. Cuando el usuario pulsa “Eliminar”, se abre primero un diálogo de confirmación para asegurarse de que realmente desea borrar ese registro. Si el usuario confirma la eliminación, se obtiene la referencia de la colección o del documento y, posteriormente, se llama a la función **deleteDoc()** para eliminar el documento de la base de datos.

Como ilustración práctica, se presenta un pseudocódigo simplificado de una función destinada a obtener todos los campus y, a continuación, otra para la inserción de un nuevo campus:

```
async function fetchCampus() {
  const colRef = collection(db, "Campus");
  const snapshot = await getDocs(colRef);
  return snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
}

async function createCampus(nombre, locked) {
  if (!nombre) throw new Error("El nombre no puede estar vacío");
  await addDoc(collection(db, "Campus"), { nombre, locked });
}
```

Lo que se consigue con esta estrategia es encapsular la lógica de Firestore en funciones reutilizables a lo largo de la aplicación. De esta forma, las vistas se restringen a la invocación de estas funciones y a la presentación de los resultados.

7.6 Operaciones con Firebase Storage

En ambas aplicaciones, para la gestión de archivos se utiliza el objeto **firebaseStorage**, el cual ya está inicializado al inicio de la aplicación. Con él se pueden realizar todas las operaciones sobre los archivos.

Para listar archivos, el proceso implica la construcción de una ruta específica que identifique la ubicación de los archivos que se quieren consultar. A modo de ejemplo, para consultar los archivos de soporte de una asignatura en un campus concreto, la ruta adoptaría la siguiente estructura: **"<id_campus>/<id_asignatura>/support_document"**. A continuación, se llama a la función **listAll()** sobre esta referencia. Esta función devuelve un array que contiene todos los archivos presentes en el directorio especificado. Posteriormente, el resultado obtenido se asigna a una variable reactiva en Vue, de modo que los resultados se actualizan automáticamente en la interfaz del usuario.

Cuando un usuario desea descargar uno de los archivos listados, se hace una llamada a la función **getDownloadURL()**. Esta función proporciona un enlace directo al archivo, el cual se abre en una nueva pestaña del navegador, permitiendo al usuario su visualización o descarga. Cabe destacar que no es necesario almacenar el enlace del fichero en la base de datos, pues al consultar los documentos de un directorio Firebase, este nos devuelve como parámetro de los archivos su ubicación.

En caso de querer subir un nuevo archivo, las aplicaciones realizan una validación inicial en el cliente para asegurar que se cumplen los requisitos establecidos, como el tamaño máximo y el formato permitido. Si la validación es exitosa, se construye la ruta de destino del archivo siguiendo la convención previamente definida. Con esta ruta, se obtiene una referencia en Storage y se invoca la función **uploadBytes()**, que es la responsable de subir el documento a Firebase Storage. Esta opción solo se encuentra en la aplicación de gestión, pues es la única con la opción de subir nuevos documentos.

Para eliminar un archivo, se muestra primero un diálogo que pide confirmación (para evitar pulsar “Eliminar” por error). Si el usuario acepta la eliminación del archivo, se obtiene la ruta del archivo seleccionado y se llama a la función **deleteObject()**. Esta función elimina el archivo de Firebase Storage. Después, se elimina el objeto de la lista reactiva en el Frontend para que no aparezca archivo en la interfaz. Al igual que en el caso anterior, esta opción solo se encuentra en la aplicación de gestión.

Toda esta lógica se ha encapsulado en diferentes funciones en ambas aplicaciones. De esa manera, los componentes de Vue solo tienen que invocar estas funciones cuando sea necesario, y el resto del tiempo se centran en mostrar la interfaz.

7.7 Operaciones con Firebase Authentication

Para la gestión del acceso a la aplicación, en ambas aplicaciones Frontend se utiliza Firebase Authentication. Este servicio se inicializa al inicio de cada aplicación en el objeto **auth**. Con este servicio, se implementan las funciones básicas de inicio y cierre de sesión, y se mantiene un control constante sobre el estado de autenticación del usuario.

Se ha dividido la sección en dos apartados: uno dedicado a la gestión de sesión, donde se explica cómo el usuario inicia y cierra sesión y cómo se emplean los tokens de Firebase; y otro centrado en la protección de rutas, que muestra cómo el router impide el acceso a vistas restringidas.

7.7.1 Gestión de las sesiones

En ambas aplicaciones, el usuario primero debe iniciar sesión. Cuando un usuario introduce sus credenciales en el formulario de inicio de sesión, el sistema realiza una validación inicial en el cliente. Esta verificación asegura que los campos requeridos no estén vacíos y que el formato del correo electrónico sea válido. A continuación, se invoca la función **signInWithEmailAndPassword** de Firebase. Esta función verifica si las credenciales son correctas; en caso de serlo, la sesión se establece y el usuario es redirigido a la vista principal de la aplicación. En caso contrario, se presenta un mensaje de error.

Una vez iniciada la sesión, Firebase almacena en el navegador un token de autenticación. De esta manera, cada vez que la aplicación intenta acceder a una ruta protegida o a datos en Firestore o Storage, basta con enviar ese token para que tanto las reglas de seguridad en el Backend como el router de Vue verifiquen su validez de forma totalmente stateless. Esto reduce la superficie de ataque, al no necesitar guardar información sensible en variables reactivas ni en stores locales, ya que toda la lógica de autorización se basa en la validez del token. Además, el propio token lleva incorporada su expiración, de modo que cuando caduque deja de ser válido sin requerir intervención del Frontend. Además, si en cualquier momento se revoca el token en el Backend, el acceso queda bloqueado automáticamente.

Para cerrar la sesión, se invoca **signOut()**. Esta función invalida el token en el cliente y hace que, en el siguiente intento de acceso a rutas o datos protegidos, tanto las reglas del Backend como del router rechacen el acceso al no contar con un token válido.

7.7.2 Protección de las rutas

Se ha añadido una capa de protección adicional a nivel de rutas en ambas aplicaciones, para que no se pueda acceder al contenido si el usuario no tiene una sesión válida activa. La protección de rutas se implementa en el router de Vue. El router verifica, antes de cada intento de navegación, si la ruta requiere autenticación. Si un usuario no autenticado intenta acceder a una vista protegida, el router lo redirige automáticamente a una página de "Permisos insuficientes". Esta medida por si sola evita que la interfaz se muestre a usuarios no autorizados. Si no se dispusiera de esta capa adicional y alguien intentara forzar la URL manualmente, esa persona podría ver la interfaz de alguna de las dos aplicaciones, pero no se cargaría ningún dato ni componente, ya que las reglas de Firebase bloquearían cualquier petición al Backend.

Adicionalmente, específicamente en la aplicación de Tutores Académicos, al intentar acceder a una asignatura el router consulta en Firestore la colección **AccessControl** correspondiente al UID del usuario. Dentro de ese documento busca la entrada que asocia el campus y la asignatura. Si existe esa entrada y su campo **expirationDate** es posterior a la fecha actual, el permiso sigue vigente; de lo contrario, se deniega el acceso. Así, la ruta solo carga la vista cuando el usuario tiene permiso activo para esa asignatura.

7.8 Incorporación de Ionic en Vue

Se ha integrado Ionic en ambas aplicaciones Frontend para proporcionar componentes y comportamientos de aspecto nativo sin tener que escribir gran parte del CSS manualmente. Gracias a esta integración, las vistas utilizan contenedores prediseñados que gestionan automáticamente detalles como el scroll, el espaciado y la adaptabilidad a distintos tamaños de pantalla, para tener una experiencia fluida en dispositivos móviles.

Para la iconografía se ha aprovechado Ionicons. La gran ventaja que tiene es que no ha sido necesario descargar iconos e incorporarlos en el proyecto, sino que ha bastado con escoger un icono de preferencia en Ionicons e importarlo. Además, al ser todos los iconos de la misma procedencia y estilo, se mantiene una apariencia coherente en las aplicaciones.

Además, para personalizar mucho más el estilo de las aplicaciones y hacerlas a medida, cada componente importa su propio archivo ".module.css". Estos archivos aseguran que los estilos queden encapsulados y no se propaguen a otros componentes. Con esto, es posible modificar o refactorizar estilos concretos sin miedo a generar efectos secundarios en otras partes de la aplicación, ya que los nombres de clase se transforman en identificadores únicos durante la compilación.

8 Evaluación

En esta sección se describe el proceso de validación funcional de las aplicaciones, mediante un conjunto de juegos de prueba que cubren los casos de uso más relevantes. El objetivo es comprobar que cada módulo responde correctamente, tanto en su interfaz de usuario como a nivel de persistencia en Firebase (Firestore y Storage).

Todas las pruebas indicadas a continuación se han verificado con Google Chrome sobre macOS. En todos los juegos de prueba se han verificado los casos en que los faltan campos, hay campos con formato inválido o las credenciales son inválidas.

8.1 Juegos de prueba en la aplicación de gestión

Caso	Descripción	Esperado	OK
Login válido	El usuario introduce credenciales correctas en la página de inicio de sesión.	Redirigido a la pantalla de subida de documentos.	Sí
Login inválido	El usuario introduce contraseña o email incorrectos.	Mensaje “Credenciales inválidas”.	Sí
Acceso sin autenticar	Intentar acceder a una ruta protegida sin una sesión activa.	Redirigido a vista indicando “No tienes acceso a este recurso”.	Sí
Subir documento	Seleccionar tipo de documento, ubicación, seleccionar un fichero y subirlo. Si no se adjunta, no se subirá ningún recurso.	El archivo se ha subido correctamente. Si no se adjunta nada, el sistema lo notifica.	Sí
Listar documentos	Acceder a “Documentos publicados”.	Se listan todos los documentos subidos indicando el tipo de documento y localización.	Sí
Buscar documentos	Se introduce un nombre de un documento.	El filtrado de los documentos coincide con el nombre introducido. Si no hay	Sí

		coincidencias, se notifica.	
Descargar documento	Se hace clic encima de un documento.	Se descarga el documento.	Sí
Eliminar documento	Se selecciona un documento y se pulsa el botón de eliminar.	El sistema muestra un diálogo de confirmación. Solo si se confirma, se elimina el documento.	Sí
Listar campus	Acceder a “Campus”	Se muestra una tabla con todos los campus y sus atributos.	Sí
Modificar visibilidad Campus	Se pulsa el icono de un ojo en uno de los campus.	La visibilidad del campus cambia.	Sí
Añadir campus	Completar y enviar formulario de nuevo campus.	El nuevo campus aparece en la tabla.	Sí
Editar campus	Modificar el nombre de un campus y guardar.	El nombre aparece actualizado en la tabla.	Sí
Eliminar campus	Seleccionar un campus y darle al botón de eliminar.	Se muestra un diálogo de confirmación; solo en el caso de confirmarlo se borra el campus.	Sí
Listar asignaturas	Acceder a “Asignaturas”.	Lista con todas las asignaturas.	Sí
Añadir asignatura	Enviar formulario de nueva asignatura.	La nueva asignatura aparece en la lista.	Sí
Modificar visibilidad asignatura	Se pulsa el icono de un ojo en una de las asignaturas.	La visibilidad de la asignatura cambia.	Sí

Editar asignatura	Modificar datos de una asignatura y guardar.	Los cambios quedan reflejados en la lista.	Sí
Eliminar asignatura	Seleccionar una asignatura y darle al botón de eliminar.	Asignatura eliminada de la lista.	Sí
Listar Coordinadores	Acceder a “Coordinadores”.	Lista con todos los coordinadores.	Sí
Añadir Coordinador	Enviar formulario de nuevo coordinador.	El nuevo coordinador aparece en la tabla.	Sí
Editar Coordinador	Modificar datos de un coordinador y guardar.	Los cambios quedan reflejados en la lista.	Sí
Eliminar Coordinador	Seleccionar coordinador y darle al botón de eliminar.	Coordinador eliminado de la lista.	Sí
Listar Administradores	Acceder a “Administradores”.	Se muestra la tabla con todos los Administradores.	Sí
Desactivar / Activar Administrador	Se pulsa Activar / Desactivar en un administrador.	Solo si el Administrador está activo, entonces puede iniciar sesión.	Sí
Editar Administrador	Modificar datos del Administrador y guardar (nombre y correo electrónico).	Los cambios quedan reflejados en la tabla.	Sí
Eliminar Administrador	Seleccionar un administrador y darle al botón de eliminar.	Administrador eliminado de la tabla.	Sí
Listar Tutores	Acceder a “Tutores”.	Se muestra la tabla con todos los Tutores.	Sí

Eliminar Tutor	Seleccionar un tutor y darle al botón de eliminar.	Tutor eliminado de la lista.	Sí
Seleccionar curso académico en Registro de Horas	Acceder a “Registro de horas” y seleccionar un curso a consultar.	Se muestran los registros de horas del curso académico seleccionado.	Sí
Añadir nuevo curso	Enviar formulario con el nuevo curso académico.	Si el formato del curso es correcto y se envía, solo en ese caso se crea el nuevo curso.	Sí
Exportar a formato Excel	Seleccionar curso en registro de horas y seleccionar “Exportar a Excel”.	Se descarga un documento de Excel con todos los campos correctos.	Sí
Listar Códigos de Registro	Acceder a “Códigos de Registro”.	Se muestra tabla con todos los códigos de registro.	Sí
Añadir Código de Registro	Enviar formulario de nuevo código de registro.	Si el código de registro es suficientemente largo y la fecha de expiración es futura, se crea el código; en caso contrario no se añade.	Sí
Desactivar / Activar Código de Registro	Se pulsa activar / desactivar en un código de registro.	Si el código está inactivo, no se podrá usar. Si está activo, se podrá usar.	Sí
Eliminar Código de Registro	Seleccionar código de registro y darle al botón de eliminar.	Código de registro eliminado de la lista.	Sí

Tabla 1. Aplicación de Gestión - Juegos de prueba

8.2 Juegos de prueba en la aplicación de Tutores Clínicos

Caso	Descripción	Esperado	OK
Login válido	El usuario introduce unas credenciales correctas en la página de inicio de sesión.	Redirigido a la pantalla de subida de documentos.	Sí
Login inválido	El usuario introduce una contraseña o un email incorrectos.	Mensaje “Credenciales inválidas”.	Sí
Registrar usuario	El usuario introduce todos sus campos y un código de registro válido.	El sistema registra al usuario.	Sí
Registro inválido	El usuario introduce campos con mal formato, incompletos o el código de registro no es válido.	El sistema avisa que hay campos inválidos.	Sí
Acceso sin autenticar	Intentar entrar a una ruta protegida sin una sesión activa.	Redirigido a una vista indicando “No tienes acceso a este recurso”.	Sí
Listar campus	Se ha iniciado sesión correctamente.	Se muestra una lista con todos los campus.	Sí
Listar asignaturas	Se ha seleccionado un campus.	Lista con todas las asignaturas del campus.	Sí
Introducir contraseña correcta asignatura	Se selecciona una asignatura a la que no se tenía acceso y se introduce una contraseña válida.	El sistema deja acceder a la asignatura.	Sí
Introducir contraseña incorrecta asignatura	Se selecciona una asignatura a la que no se tenía acceso y se introduce una contraseña inválida.	El sistema no deja acceder a la asignatura.	Sí
Acceder sin contraseña a una asignatura	Se selecciona una asignatura a la cual se tenía acceso y la fecha actual es inferior a la	El sistema deja acceder a la asignatura sin pedir contraseña.	Sí

	fecha de caducidad de la contraseña.		
Consultar Normativa / Protocolo	Se selecciona “Protocolo de accidentes y normativa” en la aplicación.	Se listan los protocolos y las normativas.	Sí
Listar Coordinadores Asignatura	Se accede a una asignatura.	Se muestran los coordinadores de la asignatura.	Sí
Listar preguntas frecuentes	Se accede a “Preguntas Frecuentes” en una asignatura.	Se listan las preguntas frecuentes.	Sí
Listar avisos importantes	Se accede a “Avisos importantes” en una asignatura.	Se listan los avisos importantes.	Sí
Listar documentos de soporte	Se accede a “Documentos de Soporte” en una asignatura.	Se listan los documentos de soporte.	Sí
Filtrar documentos de soporte	Se introduce un nombre en un campo para filtrar por nombre.	Se muestran los documentos que coinciden con el nombre.	Sí
Listar actividades del estudiante	Se accede a “Actividades del Estudiante” en una asignatura.	Se listan las actividades del estudiante.	Sí
Filtrar actividades del estudiante	Se introduce un nombre en un campo para filtrar por nombre.	Se muestran las actividades que coinciden con el nombre.	Sí
Navegar a formulario Microsoft Forms	Se selecciona “Evaluación de Mitad de Ciclo”, “Evaluación Final” o “Encuesta”.	El sistema muestra un diálogo de confirmación y solo si se acepta, se redirige al usuario al formulario de interés.	Sí
Consultar Registro de Horas	Se accede a “Registro de Horas” en una asignatura.	El sistema carga todos los datos del usuario	Sí

		importantes junto al campus y la asignatura de interés.	
Enviar Formulario Registro de Horas	El usuario rellena el formulario de registro de horas y lo envía.	Si los campos obligatorios se han introducido, el formulario se enviará con éxito.	Sí
Consultar Perfil	Se accede a “Perfil”.	Se muestran todos los datos del usuario.	Sí
Editar Perfil	Modificar datos del Perfil y guardarlos.	Los cambios quedan reflejados en el perfil.	Sí

Tabla 2. Aplicación de Tutores - Juegos de prueba

9 Evaluación de costes

En esta sección se presenta un análisis detallado de los recursos económicos invertidos en el desarrollo de las dos aplicaciones Frontend, desglosando los costes asociados tanto al personal implicado como a los elementos materiales y servicios necesarios.

9.1 Detalle de recursos del proyecto

A continuación, se detallan los perfiles profesionales implicados, las horas dedicadas a cada función y el importe total. La tarifa horaria asignada a cada recurso se corresponde con la establecida en la beca de desarrollo; por este motivo, se ha aplicado la misma tarifa en los diferentes perfiles profesionales.

Recurso	Rol	Horas	€ / h	Importe Total
Alex García	Analista de Requisitos	35	9,5 €	332,5 €
Alex García	Arquitecto de software	50	9,5 €	475 €
Alex García	Diseñador UI/UX	20	9,5 €	190 €
Alex García	Desarrollador Frontend	140	9,5 €	1.330 €
Alex García	Tester	15	9,5 €	142,5 €
Alex García	Documentador técnico	20	9,5 €	190 €
Total				**2.660 €**

Tabla 3. Recursos del proyecto

9.2 Coste de software / hardware

En este apartado se detallan los costes asociados al software / hardware utilizado.

Dado que se han empleado exclusivamente herramientas y servicios en sus planes gratuitos, el importe económico es nulo. No obstante, la siguiente tabla refleja los activos tecnológicos empleados durante el desarrollo.

Concepto	Descripción	Coste
Firebase (Authentication, Firestore, Storage)	Plan gratuito	0 €
Herramienta de desarrollo	VS Code	0 €
Herramienta de diseño	Figma Free	0 €
Control de versiones	Github	0 €
Firebase Hosting	Plan gratuito	0 €
Total		**0 €**

Tabla 4. Coste de software / hardware

9.3 Provisiones

Referente a las provisiones, se prevén las siguientes para casos imprevistos. En concreto, se han añadido provisiones en caso de que el número de horas de los diferentes trabajadores acabe siendo superior al esperado, debido a posibles complicaciones en el desarrollo del proyecto.

Concepto	€ / u	Uds.	Coste
Margen de aumento del puesto de arquitecto de Software	9,5 €	20	190 €
Margen de aumento del puesto de diseñador	9,5 €	10	95 €
Margen de aumento del puesto de desarrollador	9,5 €	30	285 €
Total			**570 €**

Tabla 5. Provisiones

9.4 Resumen de costes

Para finalizar, se indican los costes de todo el proyecto, incluyendo el subtotal y el precio final tras aplicar el IVA.

Concepto	Coste
Coste recursos humanos	2.660 €
Coste software / hardware	0 €
Provisiones	570 €
Subtotal	**3.230 €**
IVA (21%)	678,3 €
Total	**3.908,3 €**

Tabla 6. Resumen de costes

10 Legislación y protección de datos

El proyecto se desarrolla en el ámbito de un derecho fundamental reconocido por la Unión Europea: la protección de los datos personales. Este principio se materializa a nivel supranacional en el Reglamento (UE) 2016/679 (RGPD), cuyo objetivo es armonizar las normas de tratamiento de datos en todos los Estados miembros, garantizando libertades y derechos fundamentales de las personas físicas en relación con la recopilación y uso de sus datos personales. En el ordenamiento español, esta regulación se adapta y desarrolla mediante la Ley Orgánica 3/2018 (BOE, s.f.), de 5 de diciembre, de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPD-GDD), que incorpora obligaciones adicionales y refuerza los derechos de los ciudadanos en el entorno digital.

El RGPD y la LOPD-GDD establecen varios principios que guían cualquier tratamiento de datos: la licitud, la lealtad y la transparencia (la necesidad de basar el tratamiento en una base jurídica válida e informar al interesado de forma clara); la minimización de datos (recabar solo lo estrictamente necesario para el propósito buscado); la limitación de la finalidad (no usar los datos para fines incompatibles con los declarados); la integridad y la confidencialidad (asegurar la protección contra accesos no autorizados, pérdidas o destrucción); y la responsabilidad proactiva (demostrar ante posibles inspecciones el cumplimiento de estos principios). (LOPD, s.f.)

En el diseño e implementación de ambas aplicaciones Frontend se ha revisado el cumplimiento de estos principios. Por un lado, el uso de Firebase Authentication permite identificar de forma segura a los Tutores y Administradores antes de acceder a cualquier recurso. Gracias a este sistema, cada intento de lectura o escritura en la base de datos o en Storage se verifica frente al token de Firebase, de forma que solo usuarios autenticados, y con el rol adecuado, obtienen acceso. Este control de acceso se basa en un modelo de mínimo privilegio, donde los Administradores tienen permisos globales y los Tutores solo pueden gestionar su perfil, completar formularios y consultar la información general de campus y asignaturas.

Para proteger la confidencialidad de los datos sensibles (como la información de tutores y registros de horas de estudiantes), se ha aplicado criptografía simétrica y hashing con “salt” en las contraseñas de las asignaturas, códigos de registro y datos personales, evitando almacenar texto plano y anonimizando los datos lo máximo posible. En el caso de los códigos de registro, además se utiliza además enmascarado para ofrecer pistas al Administrador sin revelar la clave completa, para evitar comprometer la seguridad. Además, todo el tráfico de ambas aplicaciones navega encriptado mediante HTTPS/TLS.

En la pantalla de registro, se explica claramente al usuario cómo se van a usar sus datos y se le facilita nuestra política de privacidad, buscando siempre la claridad y la honestidad. Además, en la sección "Mi perfil", cada usuario puede acceder cuando quiera para ver y cambiar su información personal, facilitando así que ejerzan sus derechos a saber qué datos tenemos y a corregirlos, todo ello de forma directa.

Si un usuario deseara que eliminásemos sus datos (el llamado "derecho al olvido") o se opone a que los usemos, los Administradores tienen herramientas dentro del programa para desactivar o borrar cuentas de usuarios, asegurando que, si lo piden, se pueda borrar sus datos.

11 Implicaciones éticas, de igualdad y medioambientales

Desde la perspectiva de igualdad de género, el diseño de ambas aplicaciones se centra en ofrecer una experiencia intuitiva y cómoda para el público mayoritariamente femenino. Además, a nivel de interfaz se han utilizado títulos que incluyen tanto al género femenino como el masculino.

Cabe señalar que, aunque en este documento se ha empleado el masculino genérico (por ejemplo, “el usuario” o “el tutor”) en algunas ocasiones, se ha redactado así únicamente por convención lingüística. Estos términos han de entenderse en sentido inclusivo, referidos a personas de cualquier género.

A nivel medioambiental, al ofrecer a los tutores clínicos acceso directo en la web a todos los recursos necesarios para supervisar a los estudiantes, se elimina la impresión de documentos y se reduce de forma significativa el consumo de papel.

12 Valoración

Cerrar este capítulo de mi proyecto es mucho más que terminar un trabajo. Ha sido un camino constante de aprender y organizarme. Ver cómo una simple idea se convertía en dos aplicaciones me enseñó a medir mis tiempos y a documentar cada detalle. Cada error, cada ajuste, era una oportunidad para aprender.

Me planteé unos objetivos al principio y siento que los he cumplido con éxito. La aplicación para Tutores facilita su día a día, les da acceso rápido a lo que necesitan. Y el gestor para los Administradores centraliza toda la gestión de la aplicación de Tutores.

Pero lo que realmente me llevo de todo esto es la mirada al usuario. Entender de verdad qué necesitan los tutores y los coordinadores, y transformar eso en algo sencillo y práctico. Comprobé que el verdadero valor de un desarrollador está en escuchar, probar y ajustar, hasta que lo que creas realmente marca una diferencia en la vida de alguien.

En el fondo, este proyecto me permitió dar un salto grande de la teoría a la práctica. Aprendí a encontrar el equilibrio entre la usabilidad, el rendimiento y la seguridad, siempre pensando en quienes usarán las aplicaciones. No solo me llevo un producto que funciona, sino una confianza renovada en mi capacidad para afrontar proyectos más complejos y una comprensión más profunda de todo el proceso de desarrollo.

13 Bibliografía

BOE. (s.f.). *BOE*. Obtenido de Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673

LOPD, P. D. (s.f.). *Protección Datos LOPD*. Obtenido de Principios de protección de datos (LOPD y RGPD): https://protecciondatos-lopd.com/empresas/principios-generales/

14 Anexos

En este anexo se informa que la totalidad del código fuente está alojada en dos repositorios privados de GitHub:

- https://github.com/pmillan/TIC_PC_URV_APP
- https://github.com/pmillan/TIC_PC_URV_WEB

Ambos repositorios se encuentran en modo privado, pero están a disposición de quien lo solicite.