



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INGENIERÍA INFORMÁTICA

Assistance Management

Sistema de Control de Asistencia Automático

Autor

Martín José García Muñoz

Director

Juan Manuel Fernández Luna



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, Septiembre de 2022



Assistance
Management

Sistema de Control de Asistencia Automático.

Autor

Martín José García Muñoz

Directores

Juan Manuel Fernández Luna

Assistance Management: Sistema de Asistencia Automático

Martín José García Muñoz

Palabras clave: control de asistencia, gestión de asignaturas, gestión de horarios, gestión alumnos, automático

Resumen

A lo largo de la historia, los profesores han controlado las asistencias de los alumnos a sus clases de forma manual, con lo que conlleva una pérdida de tiempo extra a la hora de impartir una asignatura.

El objetivo de este proyecto es diseñar y desarrollar un sistema de asistencia automático, capaz de controlar la asistencia de los alumnos gracias a un sistema en el que intervienen los dispositivos móviles de los alumnos para registrar su asistencia en una asignatura determinada y el dispositivo móvil del profesor que actúa como lector para recibir los datos de los alumnos y registrar su asistencia, con el fin de ahorrar tiempo y evitar confusiones entre alumnos y profesores. Por ello se realizará un estudio de las tecnologías con las que se implementará el sistema y a continuación, se diseñará e implementará.

Por otro lado, con este proyecto se busca incentivar el uso del sistema por parte de las instituciones educacionales, dando una introducción hacia el uso de las tecnologías en la educación y avanzando en la evolución de las mismas.

Assistance Management: Automatic Assistance System

Martín José García Muñoz

Keywords: attendance control, subject management, schedule management, students management, automatic

Abstract

Throughout history, teachers have controlled the attendance of students in their classes manually, which leads to a loss of extra time when teaching a subject.

The objective of this project is to design and develop an automatic attendance system, capable of controlling student attendance thanks to the implemented system and the students' mobile devices, in order to save time and avoid confusion between students and teachers.

The objective of this project is to design and develop an automatic attendance system, capable of controlling student attendance thanks to a system in which the students' mobile devices intervene to record their attendance in a given subject and the teacher's mobile device who acts as a reader to receive student data and record their attendance, in order to save time and avoid confusion between students and teachers. For this reason, a study of the technologies with which the system will be implemented will be carried out and then it will be designed and implemented.

On the other hand, this project seeks to encourage the use of the system by educational institutions, giving an introduction to the use of technologies in education and advancing in their evolution.

Yo, **Martín José García Muñoz**, alumno del grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77554737W, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Martín José García Muñoz

Granada a 7 de Septiembre de 2022 .

D. **Juan Manuel Fernández Luna**, Profesor del Área Ciencias de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *Assistance Management, Sistema de Control de Asistencia Automático*, ha sido realizado bajo su supervisión por **Martín José García Muñoz**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 7 de Septiembre de 2022.

El director:

Juan Manuel Fernández Luna

Agradecimientos

A mi padre, mi madre y mi hermana, que me han apoyado durante todo el proceso de aprendizaje en esta etapa de mi vida, sin su apoyo, no podría haber llegado al punto en el que me encuentro actualmente, finalizando mi carrera.

A mis amigos, por todos los motivos que me han dado para continuar y no dejar la carrera, pese algunas situaciones que han sido malas.

A mi tutor, que me ha apoyado durante todo el proyecto y me ha ayudado en todo lo posible para sacar este proyecto adelante, más que un profesor, es un amigo y compañero que me llevó de esta carrera y con el que he podido aprender muchísimo.

A mis familiares que hoy en día no pueden estar junto a mí, pero siempre han confiado en mi futuro y me decían que llegaría lejos durante mi camino en la vida.

Índice general

1. Introducción	23
1.1. Motivación	23
1.2. Descripción del problema y solución propuesta.	24
1.3. Objetivos	24
1.4. Estructura del documento	26
2. Estado del Arte	29
2.1. Aplicaciones ya existentes	29
2.2. Tecnologías actuales	31
2.2.1. IrDA	32
2.2.2. Bluetooth	33
2.2.3. Biometría	35
2.2.4. GPS	37
2.2.5. QR	38
2.2.6. RFID	40
2.2.7. NFC	41
2.3. Decisión final sobre la tecnología a usar en nuestro proyecto .	43
2.3.1. Solución Propuesta	44
3. Planificación y Presupuesto	45
3.1. Tipo de desarrollo	45
3.1.1. Fase inicial	46
3.1.2. Fase de Diseño	46
3.1.3. Fase de Análisis	47
3.1.4. Fase de Implementación o Desarrollo	47
3.1.5. Fase de pruebas	47
3.2. Diagrama de Gantt	48
3.3. Presupuesto	48
3.3.1. Gastos del personal	48
3.3.2. Gastos de recursos informáticos y energía	49
3.3.3. Resumen del presupuesto	50

4. Análisis	51
4.1. Descripción de los implicados	51
4.2. Especificación de requisitos	54
4.2.1. Requisitos Funcionales	54
4.2.2. Requisitos No Funcionales	55
4.2.3. Requisitos de Información	56
4.3. Casos de Uso	56
4.3.1. Sistema de gestión de usuarios	56
4.3.2. Sistema aplicación móvil (Profesor)	58
4.3.3. Sistema aplicación móvil (Alumno)	58
4.3.4. Sistema aplicación Web	59
4.4. Diagramas de Casos de Uso	62
4.4.1. Sistema Gestión de Usuarios	62
4.4.2. Sistema Aplicación Web	63
4.4.3. Sistema Aplicación Móvil Profesor	64
4.4.4. Sistema Aplicación Móvil Alumno	64
4.5. Diagrama de Secuencia	65
4.5.1. Inicio/cierre de sesión	65
4.5.2. Crear/eliminar profesor	66
4.5.3. Crear/eliminar asignatura	67
4.5.4. Editar asignatura	68
4.5.5. Consultar asistencias	69
4.5.6. Lista de alumnos	69
4.5.7. Alta estudiantes/ baja estudiante	70
4.5.8. Asignar horario	71
4.5.9. Editar horario	71
4.5.10. Visualizar asignaturas (App móvil alumno)	72
4.5.11. Enviar datos a través de NFC	72
4.5.12. Activar lector de NFC	73
4.6. Diagramas Entidad-Relación	74
5. Diseño	77
5.1. Arquitectura del software	77
5.1.1. Diagrama de Base de datos	79
5.2. Diseño de las interfaces de usuario	80
5.2.1. Interfaz App web	80
5.2.2. Interfaz App Móvil Alumnos	82
5.2.3. Interfaz App Móvil Profesor	83
6. Implementación	85
6.1. Herramientas y software utilizado	85
6.2. Implementación de la base de datos	89
6.2.1. Authentication	89
6.2.2. Cloud Firestore	90

6.2.3. Ejemplos del código	92
6.3. Diagrama de Clases	95
6.4. Detalles de implementación	96
6.4.1. Estudio de las herramientas a utilizar	96
6.4.2. Despliegue App Móvil Alumnos	96
6.4.3. Despliegue App Móvil Profesor	101
6.4.4. Despliegue App Web	104
6.5. Repositorio GitHub	109
7. Problemas Encontrados	111
7.1. Idea Principal del Proyecto	111
7.2. Despliegue de la App Móvil Alumno	112
8. Pruebas	113
8.1. Pruebas de Caja Negra	113
8.1.1. Pruebas	114
8.2. Pruebas de Caja Blanca	117
8.2.1. Pruebas	118
9. Conclusiones y Trabajos Futuros	121
9.1. Conclusiones sobre el proyecto	121
9.2. Objetivos Alcanzados	122
9.3. Futuras Mejores	124
10. Anexo I. Manual de Usuario	129
10.1. Aplicación Web	129

Índice de figuras

2.1. Logo de la aplicación Alexia.	30
2.2. Vista en la aplicación móvil.	31
2.3. Puerto infrarrojo IrDA de un teléfono móvil.	32
2.4. Sistema de dispositivos conectado por Bluetooth.	33
2.5. Sistema donde se conectan los beacons con un dispositivo móvil con bluetooth.	34
2.6. Dispositivo de reconocimiento facial.	35
2.7. Dispositivo de reconocimiento dactilar.	36
2.8. Sistema GPS.	38
2.9. Teléfono Móvil escaneando código QR.	39
2.10. Sistema RFID básico.	40
2.11. Tecnologías que usan NFC.	42
3.1. Modelo en Cascada.	46
3.2. Diagrama de Gantt (Meses).	48
3.3. Diagrama de Gantt (Semanas).	48
3.4. Diagrama de Gantt (Semanas).	48
3.5. Presupuesto de ejecución del trabajo	50
4.1. Sistema Gestión de Usuarios.	62
4.2. Sistema aplicación web.	63
4.3. Aplicación móvil profesor.	64
4.4. Aplicación móvil alumno.	64
4.5. Iniciar Sesión.	65
4.6. Cierre Sesión.	66
4.7. Crear Profesor.	66
4.8. Eliminar Profesor.	67
4.9. Crear Asignatura.	67
4.10. Eliminar Asignatura.	68
4.11. Editar Asignatura.	68
4.12. Consultar asistencias.	69
4.13. Lista de Alumnos.	69
4.14. Alta alumnos.	70

4.15. Baja alumno.	70
4.16. Asignar Horario.	71
4.17. Editar Horario.	71
4.18. Visualizar Asignaturas.	72
4.19. Enviar datos NFC.	72
4.20. Lector NFC.	73
4.21. Diagrama E/R.	74
4.22. Paso a tablas.	75
4.23. Fusión de tablas.	76
 5.1. Diagrama de paquetes.	78
5.2. Diagrama Base de Datos.	79
5.3. App Web. Inicio de Sesión.	80
5.4. App Web. Página principal Administrador.	81
5.5. App Web. Página principal Profesor.	81
5.6. App Web. Página principal Profesor.	82
5.7. App Web. Página principal Profesor.	82
5.8. App Web. Página principal Profesor.	83
 6.1. Logos de las herramientas utilizadas en las aplicaciones.	85
6.2. Interfaz Firebase.	86
6.3. Firebase Authentication.	87
6.4. Firebase Cloud Firestore.	87
6.5. Logos de las herramientas utilizadas para el desarrollo del documento y del proyecto.	88
6.6. Conexión Firebase.	89
6.7. Funciones Inicio/Cierre Sesión.	90
6.8. Documento.	91
6.9. Colección.	91
6.10. Estructura almacenamiento de los datos.	92
6.11. Lectura de datos Cloud Firebase.	93
6.12. Actualización de datos Cloud Firebase.	93
6.13. Crear nuevos registros con Cloud Firebase	94
6.14. Eliminar datos Cloud Firebase	94
6.15. Diagrama de clases.	95
6.16. Conexión Authentication.	96
6.17. Conexión Firebase.	97
6.18. Activar NFC message.	98
6.19. Arrancado aplicación móvil alumno.	100
6.20. Vistas de la aplicación móvil alumno.	100
6.21. Dependencias, directorios y ficheros.	101
6.22. Registrar asistencia. App Móvil Profesor.	102
6.23. Home Screen App Profesor.	103
6.24. Dependencias, directorios y ficheros.	103

6.25. Index.js	104
6.26. App.js	105
6.27. Renderizado del componente	106
6.28. Componente HomeAdmin y funciones	106
6.29. useState()	107
6.30. useEffect()	107
6.31. NavBar	108
6.32. Dependencias, directorios y ficheros.	109
8.1. Prueba login administrador.	114
8.2. Lista de asignaturas sin scroll.	115
8.3. Alta de varias asignaturas.	116
8.4. Alta de varias asignaturas.	116
8.5. Calendario lleno.	117
8.6. App móvil Profesor.	118
8.7. App móvil Alumno. Trazas	119
8.8. App móvil Profesor. Trazas	119
8.9. Output app móvil alumno.	120
8.10. Output app móvil profesor.	120
8.11. Registro asistencia de alumno.	120
10.1. Login App Web.	129
10.2. Página Principal App Web Administrador.	130
10.3. Alta profesor App Web.	130
10.4. Página Principal App Web Profesor.	131
10.5. Asignar Horario App Web.	131
10.6. Página principal de una asignatura.	132
10.7. Sistema para cargar fichero excel.	132
10.8. Lista de alumnos de una Asignatura.	133

Índice de cuadros

4.1. Descripción de los Implicados	52
4.2. Usuario de la aplicación web	52
4.3. Usuario de la aplicación móvil	53
4.4. Administrador	53
4.5. Iniciar Sesión	56
4.6. Cerrar Sesión	57
4.7. Alta profesor	57
4.8. Baja profesor	57
4.9. Activar lector NFC	58
4.10. Visualizar asignaturas	58
4.11. Seleccionar asignatura y activar NFC	58
4.12. Crear Asignatura	59
4.13. Editar Asignatura	59
4.14. Eliminar Asignatura	59
4.15. Alta estudiantes en asignatura	60
4.16. Eliminar estudiante de asignatura	60
4.17. Consultar lista de alumnos	60
4.18. Consultar asistencias	61
4.19. Asignar horario a asignatura	61
4.20. Editar Horario Asignatura	61

Capítulo 1

Introducción

1.1. Motivación

El sistema de asistencia automática desarrollado en este proyecto va dirigido a todas las instituciones educacionales con el fin de poder realizar un seguimiento de los alumnos que acuden a clase (dependiendo de cada asignatura y profesor) de la manera más automática posible, solo con la interacción de los dispositivos móviles del alumno y profesor registrando así las asistencias de manera automática y sin la intervención del profesor.

En la actualidad, es importante tener un seguimiento del alumnado para así poder calificarlo correctamente y para ello, la asistencia a las clases es un hecho importante para que los alumnos se motiven y puedan sacar su máxima nota.

En las instituciones, cada profesor tiene una forma de pasar asistencia y esto conlleva a una serie de problemas tanto para el alumnado como para el profesor:

- Pérdida de tiempo lectivo, es decir, el tiempo que transcurre desde que se pasa lista hasta que comienza la clase puede llegar a ser relativamente grande.
- Muchas veces, el profesor debe encargarse de reconocer físicamente al alumno para poder apuntarlo y esto puede llevar a confusiones.
- El profesor tiene que pasar las hojas de asistencia a alguna base de datos u hoja Excel para calcular notas en base a las mismas y esto conlleva pérdida de tiempo.
- Si el proceso de asistencia es oral y el profesor no conoce al alumno, cualquier persona puede hacerse pasar por él.

- Si el proceso es mediante firma en una hoja, cualquier alumno podría falsificar la firma de cualquier otro.

Todos estos casos dan lugar a que no tenemos un sistema universal para pasar asistencia y puede ocasionar confusión y descontrol entre los alumnos y los profesores.

Por todas estas razones, he decidido desarrollar un *sistema de asistencia automático* capaz de ahorrar tiempo y evitar confusiones, con el que estaremos dando una introducción hacia el uso de las tecnologías en la educación y avanzando en la evolución de las mismas.

1.2. Descripción del problema y solución propuesta.

Como se ha hablado en el punto anterior, lo que se pretende con este proyecto es agilizar el método de pasar asistencia en las clases y hacerlo automático de manera que el profesor no tenga que intervenir.

Para ello, se va a utilizar un sistema de comunicación entre dos dispositivos móviles mediante la tecnología NFC con el objetivo de intercambiar información entre ambos móviles y registrar la asistencia de un alumno en una asignatura determinada.

También se creará una aplicación web donde el profesor podrá gestionar las asistencias y asignaturas correspondientes.

1.3. Objetivos

El objetivo principal del proyecto es el desarrollo de un sistema de asistencia automático capaz de controlar la asistencia de los alumnos e informarla a los profesores. El sistema utilizará una de las tecnologías actuales, *NFC*, para poder realizar la comunicación entre el alumno y el sistema.

Los objetivos del **TFG** son:

- **OBJ 1:** Desarrollar un sistema de comunicación con la tecnología NFC.
- **OBJ 2:** Aprender a usar Firebase Cloud Firestore.

- **OBJ 3:** Aprender a usar el framework de React y todo lo relacionado con su programación.
- **OBJ 4:** Aprender a desarrollar aplicaciones web y móviles mediante React y React-native.

Entre los objetivos específicos de la **aplicación móvil (Alumnos)** a desarrollar cabe destacar:

- **OBJ 1:** Mostrarle al usuario una interfaz de inicio de sesión.
- **OBJ 2:** Mostrar una lista de las asignaturas disponibles temporalmente.
- **OBJ 3:** Mostrar los datos de la asignatura donde se registrará la asistencia.
- **OBJ 4:** Registrar asistencia mediante el sistema NFC.

Los objetivos de la **aplicación móvil (profesor)** se listan de la siguiente forma:

- **OBJ 1:** Recibir los datos enviados por el dispositivo de cada alumno.
- **OBJ 2:** Registrar la asistencia correctamente en el sistema.

Los objetivos de la **aplicación web** se listan de la siguiente forma:

- **OBJ 1:** Interactuar con la aplicación de manera simple y sencilla por parte del profesor.
- **OBJ 2:** Gestionar una asignatura.
- **OBJ 3:** Gestionar profesores.
- **OBJ 4:** Gestionar asistencias.
- **OBJ 5:** Gestionar horarios.

Gracias a este proyecto, la carga y responsabilidad del profesor se verá reducida y tendrá fácil acceso al sistema pudiendo consultar cualquier día las asistencias de cualquier asignatura sin miedo a perder dicha información.

1.4. Estructura del documento

Este documento sigue la siguiente estructura de capítulos con sus respectivas secciones:

1. **Introducción:** Sección que incluye la motivación o justificación que lleva a la elección del tema para la elaboración del proyecto y los objetivos que se pretenden alcanzar con el mismo.
2. **Estado del Arte:** Este apartado incluye la explicación de todas las tecnologías con las que se podría implementar el sistema y la decisión final que se tomará para implementar este proyecto.
3. **Planificación:** Esta sección incluye cada una de las fases de la planificación temporal del proyecto, qué se ha hecho en cada fase, durante cuanto tiempo, etc. Esta planificación viene formalizada por medio de un diagrama de *Gantt*.
4. **Análisis:** Este capítulo habla de la fase de análisis del proyecto. En esta fase se describen los implicados, los requerimientos del software a implementar y diagramas de casos de uso y comportamiento del producto.
5. **Diseño:** Este apartado incluye los principales diagramas *UML* que conforman el diseño del software del proyecto. Concretamente, encontramos un diagrama de paquetes, el diseño de clases y diagramas de secuencia. Se describe también un primer diseño de la interfaz.
6. **Implementación:** En este capítulo se exponen las herramientas y software utilizado en el desarrollo del proyecto. Se habla de la metodología del trabajo, que en este caso es la metodología ágil *scrum*; y del uso de issues y pull request de *GitHub* y *git flow*. Se describen también las fases del desarrollo y el diseño final de la interfaz.
7. **Problemas Encontrados:** En este capítulo se exponen los problemas que han surgido durante el desarrollo del proyecto. Se podrán observar como han surgido dichos problemas y las soluciones a los mismos.
8. **Pruebas:** Esta sección contiene la batería de pruebas realizada al producto software para comprobar su correcto funcionamiento, así como rendimiento y eficacia.
9. **Conclusiones y Trabajos Futuros:** En este capítulo, se describe un resumen final de lo que se ha conseguido en la realización del proyecto. En este apartado se habla de los resultados obtenidos y se expresan posibles mejoras o avances futuros.

10. **Bibliografía:** En este apartado se incluyen las referencias bibliográficas usadas a lo largo del proyecto.
11. **Anexo I:** Incluye el manual de usuario de la aplicación web.

Capítulo 2

Estado del Arte

En este capítulo vamos a hablar de todas aquellas tecnologías actuales que podrían usarse para implementar un sistema de control de asistencia automático.

En la actualidad, existen diversas tecnologías con las que se podrían implementar nuestro sistema de asistencia automático, incluso existen algunos proyectos que las utilizan.

Estas tecnologías tienen sus pros y sus contras y por ello, realizaremos una comparación entre las mismas y nuestra elección final, demostrando por qué nuestra tecnología va a mejorar algunas funcionalidades que éstas no tienen.

2.1. Aplicaciones ya existentes

Hoy en día existen infinidad de aplicaciones, ya sean móviles o de escritorio con las que los profesores pueden registrar la asistencia de los alumnos. Con esta manera, evitan el uso de papel y por lo tanto, se centran en estructurar la información de cada alumno, almacenarla en hojas de Excel, base de datos... y administrarla.

Ejemplos de algunas aplicaciones como [1]:

- **Alexia Aula.** Aplicación con la que los profesores pueden controlar la asistencia desde sus dispositivos móviles e incluso compartir la misma con las familias de los estudiantes. Es una herramienta intuitiva que permite acceder a toda la información de las clases y alumnos, así como realizar tareas propias del aula desde el móvil. Su pantalla principal es una agenda que muestra por defecto todas

las sesiones planificadas para ese día, destacando aquella que esté más próxima a la hora de conexión, y con un acceso directo para pasar lista.

Podrás moverte entre la programación de los diferentes días y semanas, gestionar las tareas, actividades y anotaciones relativas a las sesiones y evaluar los controles.[2]



Figura 2.1: Logo de la aplicación Alexia.

Fuente: <https://eduemocion.com/iii-edicion/logo-alexia-suite-educativa-2>

- **Sistema de asistencia de prado(UGR).** Mediante Prado, los profesores tienen la posibilidad de registrar la asistencia de los alumnos mediante un código que tienen que generar y que los alumnos introducen en un apartado de la asignatura correspondiente dentro de la plataforma.
De esta manera, los profesores pueden ver la hora a la que han introducido el código pero además, pueden establecer un tiempo en el que los alumnos tienen para introducir dicho código. Una vez pasado el tiempo, no podrán registrar su asistencia.
- **Dinantia.** Dinantia es una plataforma de comunicación web y móvil dirigida a colegios, profesores, padres y alumnos. La plataforma ofrece enviar mensajes en tiempo real, sin necesidad de compartir ningún dato personal. Ahorra papel y tiempo en tareas administrativas con las diferentes funcionalidades de Dinantia, que incluyen, entre otras, el envío de mensajes con archivos adjuntos, preguntas y autorizaciones para salidas, control de asistencia, etc.

Además, brinda acceso a estadísticas con los porcentajes de asistencia de cada alumno o grupo y su organización en torno a períodos concretos.[3]

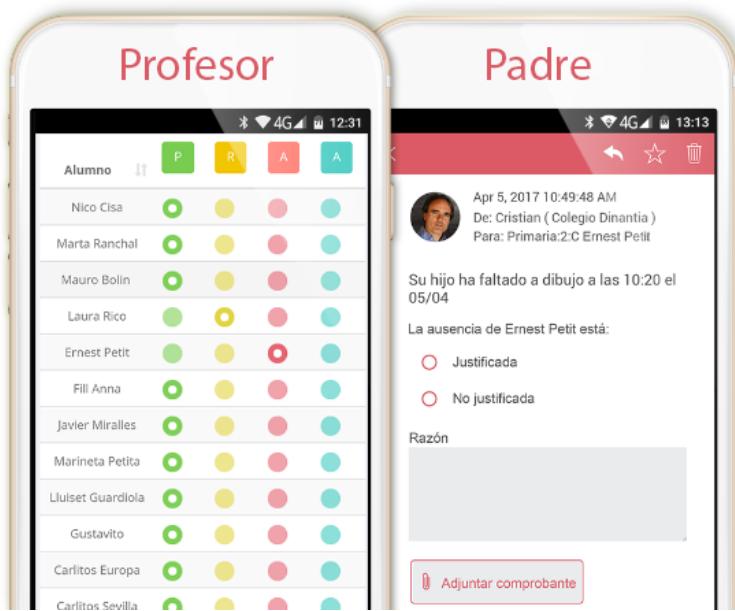


Figura 2.2: Vista en la aplicación móvil.

Fuente: <https://dinantia.com/es/funcionalidades/centro>

Estas aplicaciones y muchas más, suelen utilizarse en muchas instituciones para que los profesores tengan un control sobre los alumnos pero no me centraría en una aplicación como estas ya que supone pérdida de tiempo para el profesor al tener que realizar los registro manualmente.

Para ello, nos vamos a centrar en las siguientes tecnologías con las que se podría implementar nuestro sistema.

2.2. Tecnologías actuales

En esta sección se va a realizar un análisis de todas las tecnologías actuales con las que se podría implementar el sistema, justificando su posible implementación y el por qué no se va a utilizar en este proyecto para realizar una buena comparación entre las mismas.

2.2.1. IrDA

La tecnología *Infrared Data Association* está basada en rayos luminosos que se mueven en el espectro infrarrojo. Los estándares IrDA soportan una amplia gama de dispositivos eléctricos, informáticos y de comunicaciones que permiten la comunicación bidireccional entre dos extremos a velocidades que oscilan entre los 9600 bit/s y los 4 Mbit/s. [4]

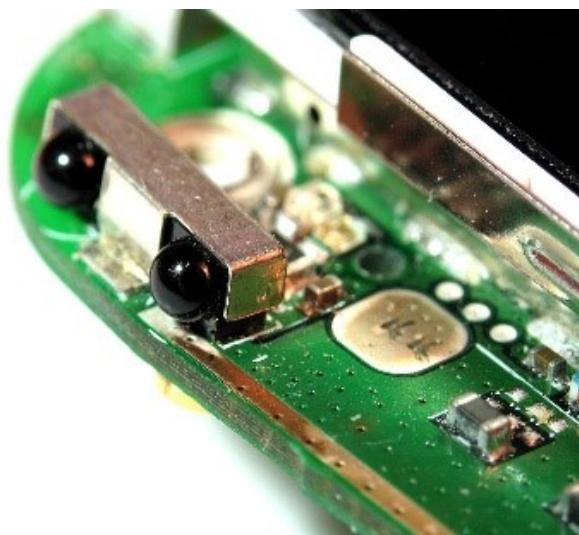


Figura 2.3: Puerto infrarrojo IrDA de un teléfono móvil.

Fuente: https://es.wikipedia.org/wiki/Infrared_Data_Association

El infrarrojo es un método de comunicación de corto alcance, y requiere que coloques los puertos de infrarrojo entre móviles frente a frente a un alcance muy corto. Una vez que se establece la comunicación entre los dos dispositivos, no debes mover los puertos de infrarrojos fuera de la línea de visión mútua o la comunicación puede interrumpirse.

Una manera de crear un sistema de asistencia utilizando esta tecnología sería utilizando una aplicación que conectara con un receptor IrDA mediante infrarrojos y así cada alumno podría grabar su asistencia conectando su dispositivo móvil a dicho receptor.

Podría ser una manera interesante pero por desgracia puede ser interrumpida fácilmente además de que necesita que el receptor permanezca a la espera, forzando al sensor receptor a consumir energía para detectar cuándo llega la luz.

También es una tecnología que está en desuso, cada vez tiene menos soporte y utilización ya que esta ha sido gradualmente desplazada por tecnologías como Wi-Fi y Bluetooth.

2.2.2. Bluetooth

Bluetooth es una tecnología de red desarrollada por el grupo de trabajo IEEE 802.15.1 del Institute of Electrical and Electronics Engineers estadounidense como estándar industrial para conexiones inalámbricas. La tecnología Bluetooth sirve para la transferencia de voz y datos punto a punto sin conexión u orientada a la conexión entre dos dispositivos digitales diferentes.

El objetivo principal de esta tecnología es reemplazar las conexiones por cable, es decir, dejarlas obsoletas, lo cual supone una ventaja, sobre todo, para dispositivos móviles como smartphones o tabletas. En comparación con otras tecnologías de transferencia de datos como USB, LAN o Wi-Fi, Bluetooth está especializada en la transferencia de datos en distancias cortas, así como en el establecimiento de conexiones sencillas y de bajo consumo. Si el objetivo es enviar archivos individuales o servicios y aplicaciones menos complejos, Bluetooth representa, sin lugar a dudas, la solución ideal.[5]

Esta tecnología podría utilizarse en un sistema de asistencia al igual que se implementa en un proyecto existente, el cuál consiste en la comunicación bidireccional mediante Bluetooth entre los dispositivos móviles Android del profesor y sus alumnos, registrando la asistencia de éstos de manera automática, y almacenándola en la base de datos interna del dispositivo. [6]

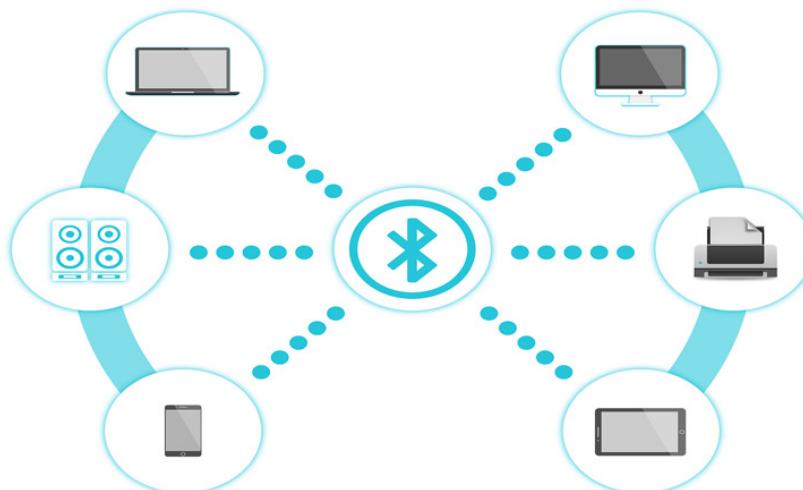


Figura 2.4: Sistema de dispositivos conectado por Bluetooth.

Fuente: <https://blog.orange.es/red/que-es-bluetooth/>

El inconveniente al utilizar esta tecnología podría ser el uso malintencionado del alumnado, pudiendo registrar su asistencia desde el exterior de la clase, aunque como he comentado anteriormente, se usa para la emisión de datos en emisiones cortas.

También podría desarrollarse un proyecto con la tecnología de los beacons y BLE (*Bluetooth Low Energy*). Los beacons son pequeños dispositivos basados en la tecnología BLE que emiten una señal que identifica de forma única a cada dispositivo. Esta señal puede ser recibida e interpretada por otros dispositivos y conocer la distancia a la que se encuentran de esta. [7] La diferencia entre Bluetooth y BLE es que, esta última está diseñada para proporcionar bajo consumo de energía manteniendo el mismo rango de alcance.

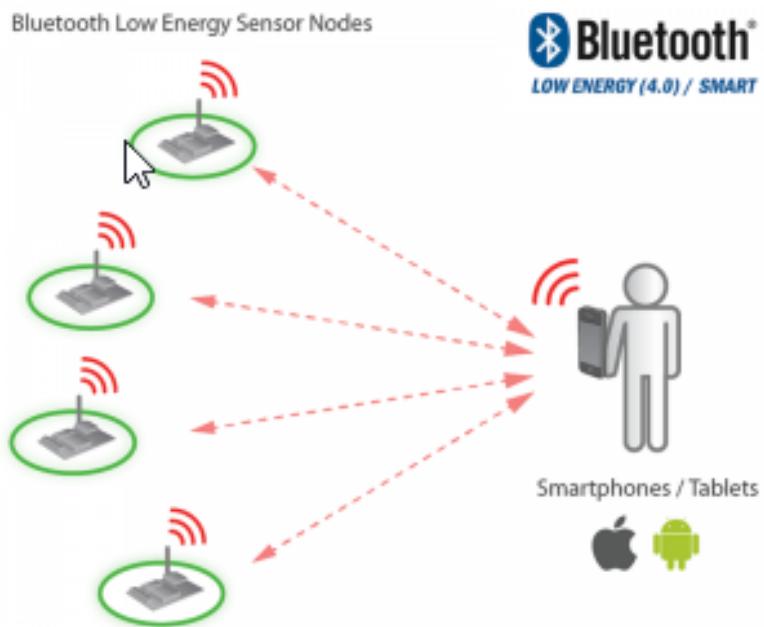


Figura 2.5: Sistema donde se conectan los beacons con un dispositivo móvil con bluetooth.

Fuente: <https://www.mokoblue.com/es/why-bluetooth-iot/>

Una propuesta de proyecto sería que la tecnología de beacons envía señales continuamente y cuando un estudiante ingrese en el aula, se comunicará con él a través del bluetooth de su dispositivo móvil.[8]

Esta tecnología es barata y fácil de construir pero por el contrario, no

utilizaría esta tecnología ya que es muy actual y la mayoría de las personas la desconocen y sobre todo porque la emisión de información diferente de todos los estudiantes podría dar lugar a generar información incorrecta.

2.2.3. Biometría

La *biometría* son las medidas biológicas o características físicas que se pueden utilizar para identificar a las personas. Las tecnologías biométricas más conocidas son el reconocimiento facial y las huellas dactilares.[9]

Reconocimiento facial

El *reconocimiento facial* es una manera de identificar la identidad de una persona mediante su rostro. Estos sistemas también pueden ser utilizados para identificar a las personas mediante fotos o videos. El objetivo de este sistema es, dada una imagen ya sea captada al momento o almacenada, encontrar una imagen de la misma cara en un conjunto de imágenes.

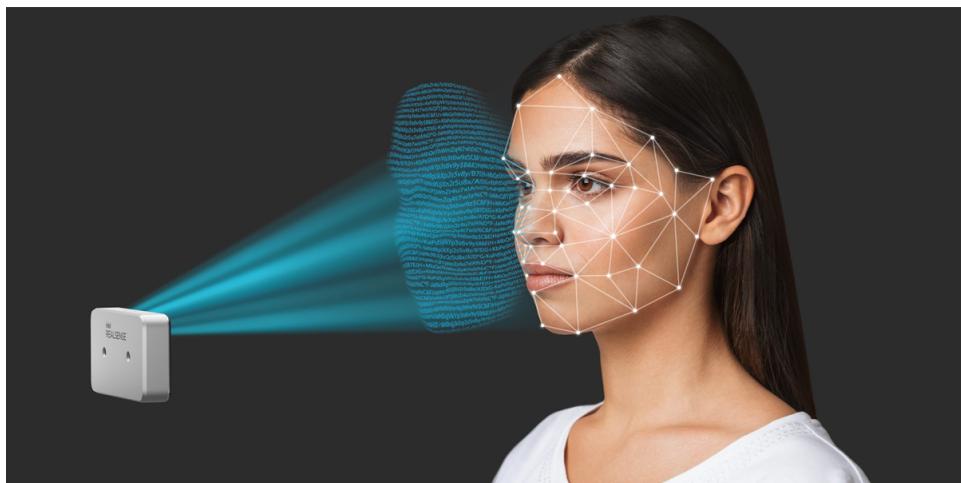


Figura 2.6: Dispositivo de reconocimiento facial.

Fuente: <https://www.profesionalreview.com/2021/01/07/realsense-id-reconocimiento-facial-intel>

Puede operar de dos modos:

- **Verificación o autentificación de caras:** compara una imagen de la cara con otra imagen con la cara de la que queremos saber la identidad. El sistema confirmará o rechazará la identidad de la cara.

- **Identificación o reconocimiento de caras:** compara la imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en la base de datos para determinar su identidad. [10]

Un sistema de asistencia mediante reconocimiento fácil podría ser aquel en el que utilicemos un dispositivo que analice la cara de cada alumno y al encontrarla en su base de datos, registrar su asistencia correspondiente. Sin embargo, este sistema es muy costoso y en a veces, el alumno no cuenta con un feedback necesario para asegurarse de que su asistencia a sido registrada correctamente.

Reconocimiento dactilar

El *reconocimiento de huella dactilar* se hace a través un sensor que digitaliza el dedo del usuario para que el algoritmo extraiga puntos de la imagen de la huella y convierta la información en un único modelo. Este modelo único es como un password que se encripta y se archiva en la base de datos representando al usuario. [11]



Figura 2.7: Dispositivo de reconocimiento dactilar.

Fuente: https://kimaldi.com/blog/biometria/que_aplicaciones_pueden_tener_los_sistemas_de_reconocimiento_biometrico

Para llevarlo a nuestro caso, haría falta un lector de huella dactilar que compruebe un dedo de cada alumno, compruebe que existe en su base de datos y registrar su asistencia. Al igual que en el caso anterior, este es un

método muy costoso debido al precio de los dispositivos con el que se implementaría.

Por lo tanto, no veo adecuado el uso de la biometría sobre todo por el tema económico que supone.

2.2.4. GPS

Los dispositivos GPS (*Global Position System* o Sistema de Posicionamiento Global), son dispositivos de rastreo a nivel mundial que te indican dónde te encuentras gracias a la información enviada por satélite.

Es un sistema que nos permite saber la localización de cualquier persona, vehículo o cosa, la velocidad a la que se mueve y otros datos como su altura en cualquier momento y punto del globo terrestre. El GPS funciona mediante una red de mínimo 24 satélites que se encuentran en órbita sobre nuestro planeta, aproximadamente a unos 20.000 km de altura, con órbitas distribuidas para que en todo momento haya al menos cuatro satélites visibles en cualquier punto de la Tierra.

Cuando queremos determinar la posición exacta de alguien o algo, el receptor que utilizamos debe localizar como mínimo cuatro de estos satélites de la red, de los cuales recibirá unas señales indicando la identificación y hora del reloj de cada uno de ellos y la información sobre la constelación. En base a estas señales, el receptor sincroniza su propio reloj con el tiempo del sistema GPS y calcula el tiempo que tardan en llegar las señales al equipo para calcular la distancia con el satélite. A continuación, teniendo en cuenta la velocidad de la señal y mediante el método de trilateración inversa, calcula su propia posición. Podríamos decir que el posicionamiento satelital funciona básicamente de esta manera, pero lo cierto es que se usan algoritmos muy complejos en dichos cálculos.

[12]

Comentar un caso actual en China donde se controlan a los alumnos mediante la tecnología GPS incorporada en sus uniformes. Los uniformes inteligentes fueron desarrollados por la firma Ghizou Guanyu Technology y cuenta con una serie de sensores y GPS instalados en la tela de las prendas en la zona de los hombros. El sistema funciona con una serie de cámaras que registran los movimientos y detecta que los alumnos utilicen la prenda que les corresponde, luego de cotejar el rostro mediante un sistema de reconocimiento facial.

El uniforme inteligente permite notificar a los padres si sus hijos se ausentan de la escuela y emite una alarma de forma automática si se encuentra fuera del área delimitada por el colegio y los padres.[13]

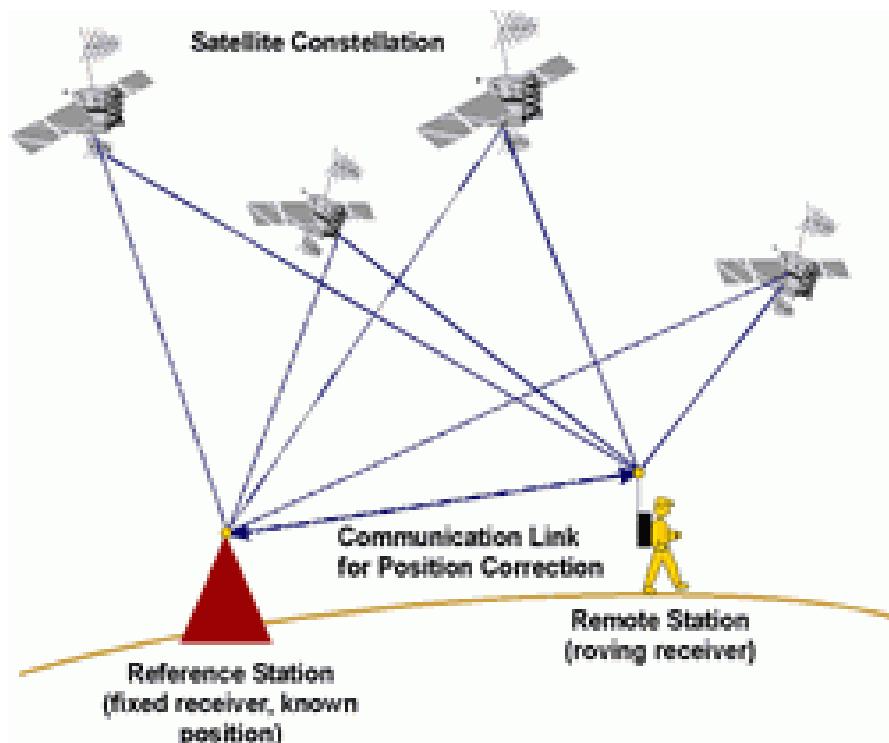


Figura 2.8: Sistema GPS.

Fuente: [https://www.biolaster.com/rendimiento-deportivo/
control-entrenamiento/gps-deporte/tecnologia-gps-sports](https://www.biolaster.com/rendimiento-deportivo/control-entrenamiento/gps-deporte/tecnologia-gps-sports)

Básicamente, implementaría un sistema donde los dispositivos de los alumnos, al entrar a un aula determinada, registrarán su asistencia gracias a la tecnología GPS. Con este sistema podemos asegurarnos la presencia de cada alumno en la clase y evitar así la posibilidad de falsificación, pero descarté esta opción por la gestión de una conexión GPS y la imposibilidad de realizar una conexión bidireccional entre el profesor y el alumno.

2.2.5. QR

Los códigos *QR* son la evolución digital del código de barras. Su estructura está compuesta por una matriz bidimensional de módulos de dos colores contrastados, generalmente blancos y negros. Éstos presentan tres cuadrados en las esquinas que permiten detectar la posición del código al lector. Dichos códigos almacenan información (como sitios web o aplicaciones) a

la que se tiene acceso al escanearlos mediante una cámara con un software específico.

Existen dos tipos de códigos QR en la actualidad:

- **Códigos QR estáticos**, que contienen siempre la misma información una vez creados.
- **Códigos QR dinámicos**, que se crean para aquellos casos en los que la información incluida debe ser actualizada periódicamente sin necesidad de cambiar de código.

La estructura que conforma un código QR está diseñada para mantener la información legible para un escáner, aunque esté oscuro o dañado.[14]



Figura 2.9: Teléfono Móvil escaneando código QR.

Fuente: https://www.segurilatam.com/actualidad/codigos-qr-que-son-riesgos-asociados-y-consejos-de-seguridad_20211017.html

Una forma para pasar asistencia utilizando código QR sería que el profesor genere un código QR para cada alumno y cuando quiera pasar lista,

simplemente tendrá que escanear mediante un escáner o su propio móvil, el código QR de cada alumno y así registrar su asistencia. Este tipo de sistema implicaría que el profesor tuviera que generar los códigos QR para todos los alumnos, lo cual supone una carga de tiempo elevada. A parte, en mi universidad tuve a un profesor que utilizaba este método.

Él nos asignó a todos los alumnos un código QR que teníamos almacenado en nuestros móviles y a principio de clase, escaneaba con su móvil cada uno de los códigos de los alumnos para anotar la asistencia. Como es un sistema que existe y conozco su funcionamiento, he decidido descartar esta opción.

2.2.6. RFID

La tecnología *RFID* es una forma de comunicación inalámbrica entre un lector y un emisor. Se puede comparar con un código de barras, aunque en lugar de marcas de tinta se utilizan ondas de radio. De hecho, las etiquetas con esta tecnología son muy utilizadas en la industria, tanto para localizar objetos como para asegurarse de que estos no se sacan de un establecimiento sin los permisos pertinentes.

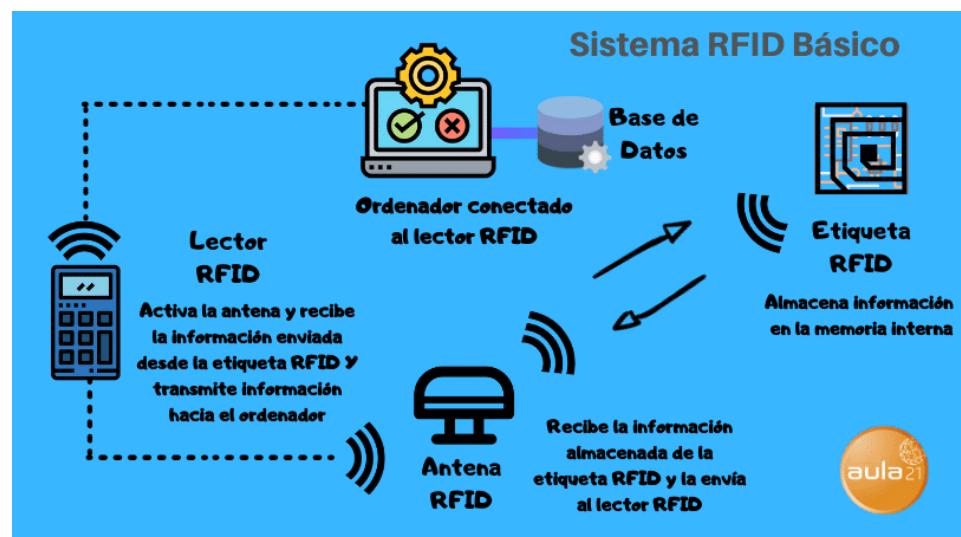


Figura 2.10: Sistema RFID básico.

Fuente: <https://www.cursosaula21.com/que-es-el-rfid>

RFID utiliza las ondas de radio para comunicarse con un microchip con una gran capacidad de almacenamiento de datos, por lo que permite guardar mucha más información que las etiquetas de código de barras tradicional. Su tecnología hace que sean muy difíciles de duplicar lo que aumenta su seguridad y además, permiten realizar la lectura de forma prácticamente

instantánea, a distancia y sin necesidad de línea de visión.[15]

Los sistemas RFID constan de tres componentes: una etiqueta RFID o etiqueta inteligente, un lector RFID y una antena RFID. Las etiquetas RFID contienen un circuito integrado y una antena, que se utilizan para transmitir datos al lector RFID. El lector entonces convierte las ondas de radio en una forma de datos más utilizable. La información recogida de las etiquetas se transfiere a través de una interfaz de comunicaciones a un sistema informático principal, donde la información puede ser almacenada en una base de datos y analizada posteriormente.[16]

En nuestro caso, implementaríamos un sistema similar entregando a cada alumno una tarjeta RFID y que estando a cierta distancia de un lector RFID conectado al pc del profesor, se recogen sus datos y se registra la asistencia en una base de datos. Este sistema es sencillo y fácil de mantener, aunque deberíamos explicar antes a los alumnos la manera de usarlo al igual que en los anteriores casos además de, tener que entregar a cada alumno un identificador RFID, lo cuál supone un coste extra para la institución.

2.2.7. NFC

Near-field communication (NFC) o comunicación de campo cercano es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos. Los estándares de NFC cubren protocolos de comunicación y formatos de intercambio de datos, y están basados en RFID.

NFC se comunica mediante inducción en un campo magnético, en donde dos antenas de espiral son colocadas dentro de sus respectivos campos cercanos. Trabaja en la banda de los 13,56 MHz, esto hace que no se aplique ninguna restricción y no requiera ninguna licencia para su uso.[17]

Soporta dos modos de funcionamiento:

- **Activo:** ambos dispositivos generan su propio campo electromagnético, que utilizarán para transmitir sus datos.
- **Pasivo:** solo un dispositivo genera el campo electromagnético y el otro se aprovecha de la modulación de la carga para poder transferir los datos. El iniciador de la comunicación es el encargado de generar el campo electromagnético.

El uso de la tecnología NFC es muy similar al de la conexión bluetooth sólo que ahorra en tiempo y esfuerzo.

Esta tecnología destaca por su alta velocidad de transmisión de datos entre teléfonos sin necesidad de emparejamiento previo, lo que la distingue de otras tecnologías como infrarrojos o bluetooth. Los equipos con tecnología NFC son capaces de enviar y de recibir información simultáneamente y pueden alcanzar más de 420 kbit/segundo de velocidad. Además, este sistema consume muy poca batería cuando está en uso.

El alcance de la tecnología es de 20 cm, por lo que si queremos pasar datos de un teléfono a otro los dos dispositivos tendrán que estar muy cerca. Esta característica es una ventaja en términos de seguridad, pues podrás ver con tus propios ojos cómo la información que envías llega al destinatario adecuado a diferencia que con la tecnología RFID.[18]



Figura 2.11: Tecnologías que usan NFC.

Fuente: https://www.elespanol.com/elandroidelibre/tutoriales/trucos/20150118/puedes-hacer-nfc-smartphone/4249912_0.html

Por lo tanto, en nuestro sistema de asistencia que vamos a implementar, la distancia máxima permitida entre dispositivos para realizar la conexión

debe ser inferior a 10.16 cm (aunque la distancia máxima efectiva suele ser de aproximadamente 5 cm), de forma que se asegura la asistencia del alumno a la clase a la hora de realizar el intercambio de datos.

2.3. Decisión final sobre la tecnología a usar en nuestro proyecto

El sistema de asistencia automático lo voy a desarrollar con la tecnología NFC, registrando la asistencia de cada alumno una vez que cada uno de ellos pase su móvil junto al lector NFC (dispositivo móvil del profesor) y enviando los datos a una base de datos.

He llegado a esta decisión por las siguientes ventajas que plantea este sistema respecto los sistemas mencionados anteriormente:

- Sistema que destaca por la alta velocidad y rapidez de transmisión de la información entre dispositivos.
- No necesita un previo emparejamiento con otros dispositivos para realizar la comunicación.
- Alcance mínimo para realizar la transmisión respecto a las demás tecnologías, por lo tanto, podemos asegurarnos en todo momento que el alumno ha entrado a la clase.
- Ahorro de tiempo ya que lo único que tienen que hacer los alumnos es logearse en la aplicación móvil y pasar su dispositivo por el lector NFC para registrar la asistencia.
- No necesita que el profesor esté pasando lista de manera manual con cualquier aplicación o con una hoja.
- La implementación del sistema no es muy costosa a diferencia de la tecnología de biometría (reconocimiento dactilar o facial)
- El sistema proporcionará al alumno la confirmación de su asistencia.

Como inconvenientes tiene:

- Se requiere de conexión a internet para poder subir los datos a un servidor de base de datos al que se accederá desde la aplicación web.
- La velocidad es más lenta que la del Bluetooth, ya que es de 424kbits/s

4.2.3. Decisión final sobre la tecnología a usar en nuestro proyecto

2.3.1. Solución Propuesta

Como he explicado en el punto anterior, mi proyecto va a consistir en un sistema de control de asistencia automático usando la tecnología NFC.

El sistema va a estar compuesto por 3 elementos esenciales:

- Una aplicación móvil sencilla que se descargarán los alumnos para iniciar sesión con su correo de universidad y activar el NFC en su dispositivo móvil.
- Otra aplicación móvil para el profesor que hará de lector NFC de la información enviada por el dispositivo del alumno.
- Una aplicación web para el profesor donde podrá gestionar las clases, horarios, asistencias ...

Capítulo 3

Planificación y Presupuesto

En este apartado se realiza una planificación de las tareas necesarias para llevar a cabo este TFG.

3.1. Tipo de desarrollo

El desarrollo del proyecto se llevará a cabo utilizando una metodología en cascada (Waterfall). El desarrollo en cascada es un procedimiento lineal que se caracteriza por dividir los procesos de desarrollo en sucesivas fases de proyecto. Al contrario que en los modelos iterativos, cada una de estas fases se ejecuta tan solo una vez. Los resultados de cada una de las fases sirven como hipótesis de partida para la siguiente. El waterfall model se utiliza, especialmente, en el desarrollo de software.[19]

Las fases de este modelo son:

- **Análisis:** planificación, análisis y especificación de los requisitos.
- **Diseño:** diseño y especificación del sistema.
- **Implementación:** programación y pruebas unitarias.
- **Verificación:** integración de sistemas, pruebas de sistema y de integración.
- **Mantenimiento:** entrega, mantenimiento y mejora.

Con la siguiente imagen se puede ver como el procedimiento lineal se denomina metodología en cascada.



Figura 3.1: Modelo en Cascada.

Fuente: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada>

Para la realización de este proyecto, se va a hacer referencia a las tres primeras fases, incluyendo antes de la fase de análisis, una fase inicial y al final, una fase de pruebas donde se probarán todos los objetivos desarrollados.

3.1.1. Fase inicial

En cuanto a la primera fase del desarrollo del proyecto, se comenzará con una introducción donde se explicará la motivación y objetivos del proyecto.

En esta fase también se realizará una investigación sobre las tecnologías actuales con las que se podría implementar el sistema que utilizaremos en nuestro proyecto, se explicará la tecnología escogida y se definirán las herramientas software que se utilizarán durante el proyecto.

3.1.2. Fase de Diseño

En esta segunda fase, se realizará la planificación del proyecto y la especificación de requisitos funcionales, no funcionales y de información, así

como la descripción de los implicados que intervienen en el proyecto.

3.1.3. Fase de Análisis

En esta tercera fase del desarrollo, se propondrá el diseño a alto nivel de la aplicación. Se diseñará la arquitectura del software a desarrollar con un diagrama de paquetes, se crearán los diagramas de secuencia y el diagrama de clases.

3.1.4. Fase de Implementación o Desarrollo

En esta cuarta fase del desarrollo, se comenzará con la implementación del proyecto software, que se traduce al correspondiente lenguaje de programación.

3.1.5. Fase de pruebas

En esta fase se realizarán pruebas y se incorporarán algunos cambios y mejoras a las aplicaciones, tanto móviles como web.

3.2. Diagrama de Gantt

A continuación se muestra el diagrama de Gantt que hace referencia a las fases del proyecto anteriormente mencionadas.

Diagrama de Gantt por Meses:



Figura 3.2: Diagrama de Gantt (Meses).

Diagrama de Gantt por Semanas:

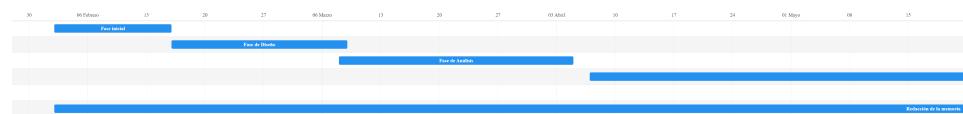


Figura 3.3: Diagrama de Gantt (Semanas).



Figura 3.4: Diagrama de Gantt (Semanas).

3.3. Presupuesto

En este apartado se va a realizar una aproximación de los posibles costes del desarrollo del proyecto. Se detallan aquí los gastos por personal, equipos o recursos informáticos, licencias de softwares, etc.

3.3.1. Gastos del personal

En esta sección se calcula el gasto total referente a pago de personal que ha estado trabajando en el desarrollo del proyecto.

El trabajo ha sido realizado por un grupo de una sola persona o desarrollador con la ayuda de un profesor mediante tutorías.

Se estima que el desarrollador va a recibir un salario base promedio de un programador Junior en Granada a jornada completa de 18000€/año, lo que sería igual a 1500 €/mes.

Observando el diagrama de Gantt realizado en el punto anterior, se puede observar que el desarrollo del proyecto tiene una estimación de 8 meses. Suponiendo que el desarrollador trabajará a media jornada (4 horas diarias), el pago total sería de $1.500 * 8 / 2 = 6000 \text{ €}$.

También se tienen que sumar las horas invertidas por el tutor del proyecto en las tutorías. Estimando un número de 6 tutorías a lo largo del proyecto y que el coste por tutoría es de 20 €, se tiene una inversión de 120 €.

3.3.2. Gastos de recursos informáticos y energía

En este apartado se va a realizar una aproximación de los costes referentes a los recursos informáticos necesarios para el desarrollo del proyecto.

- **Equipo usado para el desarrollo del proyecto:** Torre (Intel 5 11th, 16GB RAM, 512GB SSD, 2 TB HDD, Nvidia RTX2060-6GB, Windows 11).
Coste: 1100 €
- **Periférico:** 2 Monitores, ratón y teclado.
Coste: 300 €
- **Software:** Las aplicaciones y licencias que se han utilizado son de uso gratuito.

Se estima también el coste del gasto energético, que se supone que el gasto medio actual en España es de 50 €/mes en electricidad. Por lo tanto, tenemos un gasto adicional de $50 * 8 = 400\text{€}$.

3.3.3. Resumen del presupuesto

En la figura 3.5 se resumen todos los gastos vistos en las secciones anteriores.

Presupuesto	
Sueldo desarrollador	6000 €
Horas de tutorías	120 €
Hardware	1400 €
Electricidad	400 €
Total gastos previstos	7.920 €

Figura 3.5: Presupuesto de ejecución del trabajo

Capítulo 4

Análisis

En este capítulo se describe la fase de análisis del problema a tratar. En concreto desarrollo la descripción de los implicados en el proceso y la especificación de requisitos funcionales, no funcionales y de información que debe cubrir la solución al problema. Como refuerzo para los requisitos del proyecto, se incluyen diagramas de casos de uso en los que interviene cada uno de los implicados.

4.1. Descripción de los implicados

En nuestro sistema podemos destacar tres principales implicados: el administrador del sistema y de las aplicaciones, tanto móviles como web, usuario de la aplicación web (profesor, usuario de la aplicación móvil para alumnos y usuario de la aplicación móvil para el profesor.

- **Administrador del sistema:** Este implicado se encargará de realizar las actividades relacionadas con el desarrollo del sistema y de las aplicaciones: corregir errores, añadir o actualizar mejoras en la aplicación web, mantenimiento del sistema, de las aplicaciones y de la base de datos.
- **Usuario de la aplicación web y de la aplicación móvil del profesor:** Este implicado se corresponde con el profesor, el cual será el usuario final de la aplicación web. Se encargará de administrar las asignaturas, alumnos y asistencias.
- **Usuario de la aplicación móvil del alumno:** Este implicado se corresponde con el alumno, el usuario final de la aplicación móvil. Se encargará de hacer uso de la aplicación móvil y de registrar sus datos correctamente para poder registrar su asistencia.

En la tabla de a continuación se describen cada uno de los implicados del sistema:

Nombre	Descripción	Tipo	Responsabilidad
Profesores	Profesor de una asignatura determinada	Usuario Sistema	Organizar clases, horarios, asistencias, activar lector NFC ...
Alumnos	Alumnos de una asignatura determinada	Usuario Sistema	Dar los datos de identificación al sistema
Administrador del sistema	Se encarga del soporte y desarrollo del sistema	Usuario Producto	Gestionar el sistema y mantener la aplicación web sin errores, añadiendo cosas que el profesor pueda necesitar

Cuadro 4.1: Descripción de los Implicados

En esta tabla se describen cada una de las implicaciones en el sistema del usuario de la aplicación web, en este caso el profesor:

Profesor	
Descripción	Profesor de una asignatura
Tipo	Utiliza el sistema para organizar sus clases
Responsabilidades	Organizar clases, horarios, asistencias de los alumnos...
Criterios de éxito	Debe intentar realizar una gestión adecuada de la clase
Implicación	Utiliza el sistema para controlar y organizar una asignatura determinada
Comentarios/Cuestiones	Estar familiarizado con el sistema informático

Cuadro 4.2: Usuario de la aplicación web

En esta tabla se describen cada una de las implicaciones en el sistema del usuario de la aplicación móvil del alumno:

Alumno	
Descripción	Alumno de una asignatura
Tipo	Utiliza el sistema para registrar su asistencia en el sistema
Responsabilidades	Dar los datos de identificación al sistema
Criterios de éxito	Que la información que utilizar para registrarse en la aplicación sea correcta
Implicación	
Comentarios/Cuestiones	

Cuadro 4.3: Usuario de la aplicación móvil

En esta tabla se describen cada una de las implicaciones en el sistema del administrador:

Administrador del Sistema	
Descripción	Administrador general del sistema
Tipo	Tiene la responsabilidad total sobre la coherencia y gestión del sistema
Responsabilidades	Agregar y eliminar información del sistema. Mantener coherencia en la base de datos.
Criterios de éxito	Mantener el sistema a prueba de errores
Implicación	Todos los niveles
Comentarios/Cuestiones	Estar familiarizado con el sistema informático

Cuadro 4.4: Administrador

4.2. Especificación de requisitos

4.2.1. Requisitos Funcionales

Descripción de los requisitos más importantes a nivel de funciones que debe incluir el sistema, realizando una clasificación en categorías, a cada uno de los requisitos se le ha asignado un código y un nombre, con el fin de identificarlos fácilmente a lo largo de todo el proyecto.

▪ **RF-1. Gestión de la aplicación móvil del alumno**

- **RF-1.1. Iniciar sesión en la aplicación móvil.** El usuario de la aplicación deberá iniciar sesión en la aplicación para hacer uso de la misma.
- **RF-1.2. Listar asignaturas disponibles.** El usuario podrá ver la lista de asignaturas disponibles en un día y hora concretos.
- **RF-1.3. Activar sistema NFC.** Una vez seleccionada una asignatura, el usuario podrá registrar la asistencia activando la tecnología NFC de su dispositivo móvil para transmitir sus datos al lector NFC.

▪ **RF-2. Gestión de la aplicación móvil del profesor**

- **RF-2.1. Activar lector NFC.** El profesor activará la tecnología NFC de su dispositivo móvil para recibir los datos enviados por los alumnos.

▪ **RF-3. Gestión de la aplicación Web**

- **RF-3.1. Dar de alta una asignatura.** El usuario podrá dar de alta una clase y gestionarla.
- **RF-3.2. Dar de baja una asignatura.** El usuario podrá dar de baja una asignatura.
- **RF-3.3. Dar de alta una lista de alumnos en una asignatura.** El usuario podrá dar de alta a una lista de alumnos en una asignatura del sistema.
- **RF-3.4. Dar de baja a un alumno en una asignatura.** El usuario podrá dar de baja a un alumno en el sistema.
- **RF-3.5. Crear horario.** El usuario podrá crear el horario correspondiente a una asignatura determinada.
- **RF-3.6. Gestionar las asistencias.** El usuario podrá consultar las asistencias registradas correctamente en un día específico.

- **RF-3.7. Consultar lista de alumnos en una asignatura.** El usuario podrá consultar la lista de alumnos pertenecientes a una asignatura.
- **RF-3.8. Dar de alta a un profesor.** El administrador podrá dar de alta a un profesor
- **RF-3.9. Dar de baja a un profesor.** El administrador podrá dar de baja a un profesor

4.2.2. Requisitos No Funcionales

- **RNF-1. Usabilidad**

- **RNF-1.1.** Se proporcionará un teléfono de contacto para cualquier duda del usuario.
- **RNF-1.2.** Se informará de cualquier cambio o modificación en el sistema mediante un correo.

- **RNF-2. Fiabilidad**

- **RNF-2.1.** Para prever caídas del sistema, se harán copias de seguridad.
- **RNF-2.2.** Se garantizará la fiabilidad del sistema, que todo funcione correctamente.

- **RNF-3. Rendimiento**

- **RNF-3.1.** El sistema debe estar preparado para almacenar la información de los alumnos.
- **RNF-3.2.** Se usará Firebase RealTime Database como base de datos noSQL.
- **RNF-3.3.** Se usará React como framework (React app web y React-native apps móviles) y javascript como lenguaje de programación.

- **RNF-4. Interfaz**

- **RNF-4.1.** La interfaz de las aplicaciones móviles como de la aplicación web serán sencillas y de fácil uso para todo tipo de usuarios del sistema.

- **RNF-5. Soporte**

- **RNF-5.1.** La responsabilidad del correcto funcionamiento y mantenimiento del sistema corre de parte del administrador del sistema.

4.2.3. Requisitos de Información

- **RI-1.** El sistema gestor de base de datos utilizado será Firebase Cloud Firestore.
- **RI-2.** Se almacena en el sistema información de los alumnos y del profesor, DNI, correo electrónico, teléfono ...

4.3. Casos de Uso

4.3.1. Sistema de gestión de usuarios

Estos casos de uso son los mismos tanto para el sistema de la aplicación móvil como el sistema de la aplicación web, esto quiere decir, que ambos sistemas tienen un usuario implicado (alumno o profesor) y el administrador del sistema.

Iniciar Sesión		CU-01
Actores:	Usuario	
Descripción:	Acceso al sistema mediante credenciales de usuario.	
Precondición:	Es necesario ser usuario del sistema.	
Postcondición:	N/A	
Flujo de eventos:	Actividades del actor	Respuesta del sistema
	1. Insertar datos en el formulario. 3. Enviar datos.	2. Validar datos del formulario. 4. Procesar información. 5. Mostrar perfil al usuario.
Excepciones:	Si los datos del formulario no son válidos, no se continúa con el flujo. Si el usuario no está registrado, no podrá acceder al sistema y se mostrará un mensaje de información.	

Cuadro 4.5: Iniciar Sesión

Cerrar Sesión			CU-02
Actores:	Usuario		
Descripción:	El usuario cierra la sesión en el sistema. Para acceder de nuevo tiene que iniciar sesión.		
Precondición:	El usuario debe haber iniciado sesión anteriormente.		
Postcondición:	N/A		
	Actividades del actor	Respuesta del sistema	
Flujo de eventos:	1. Cerrar sesión.	2. Mostrar formulario de acceso.	
Excepciones:	N/A		

Cuadro 4.6: Cerrar Sesión

Alta profesor			CU-03
Actores:	Administrador		
Descripción:	El administrador crea un nuevo profesor.		
Precondición:	N/A		
Postcondición:	N/A		
	Actividades del actor	Respuesta del sistema	
Flujo de eventos:	1. . Rellenar formulario para crear un nuevo profesor. 3. . Dar de alta profesor.	2. Validar datos del formulario. 4. Procesar información.	
Excepciones:	Si los datos del formulario no son válidos, no se continúa con el flujo. Si los datos del profesor ya existen, mostrar un mensaje de información.		

Cuadro 4.7: Alta profesor

Baja profesor			CU-04
Actores:	Administrador		
Descripción:	El profesor a eliminar debe existir en el sistema.		
Precondición:	El administrador debe haber iniciado sesión anteriormente.		
Postcondición:	N/A		
	Actividades del actor	Respuesta del sistema	
Flujo de eventos:	1. Eliminar profesor.	2. Actualizar lista de profesores.	
Excepciones:	N/A		

Cuadro 4.8: Baja profesor

4.3.2. Sistema aplicación móvil (Profesor)

Activar lector NFC			CU-05
Actores:	Profesor		
Descripción:	El profesor podrá activar el lector NFC para registrar asistencias.		
Precondición:	N/A		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: activar lector.	Respuesta del sistema 2. Activar sistema de lectura para procesar datos NFC.	
Excepciones:	Si ocurre algún error al activar el sistema del lector, se muestra un mensaje al estudiante.		

Cuadro 4.9: Activar lector NFC

4.3.3. Sistema aplicación móvil (Alumno)

Visualizar lista de asignaturas disponibles			CU-06
Actores:	Estudiante		
Descripción:	El estudiante podrá ver las asignaturas disponibles.		
Precondición:	El estudiante ha accedido al sistema.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. Logearse.	Respuesta del sistema 2. Ir a la siguiente pantalla y mostrar lista de asignaturas.	
Excepciones:	Si ocurre algún error a la hora de mostrar los datos, se muestra un mensaje al estudiante.		

Cuadro 4.10: Visualizar asignaturas

Seleccionar asignatura y activar NFC			CU-07
Actores:	Estudiante		
Descripción:	El estudiante seleccionara asignatura y podrá activar el NFC para registrar su asistencia.		
Precondición:	El estudiante ha accedido al sistema.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. Seleccionar asignatura.	Respuesta del sistema 2. Ir a la pantalla de la asignatura 3. Registrar asistencia	
Excepciones:	Si ocurre algún error al activar el sistema, se muestra un mensaje al estudiante.		

Cuadro 4.11: Seleccionar asignatura y activar NFC

4.3.4. Sistema aplicación Web

Crear Asignatura		CU-08
Actores:	Profesor	
Descripción:	El profesor crea una nueva asignatura.	
Precondición:	El profesor ha iniciado sesión en el sistema.	
Postcondición:	N/A	
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: Nueva Asignatura. 3. Insertar datos en el formulario. 5. Seleccionar opción: Crear. 7. Navegación página principal	Respuesta del sistema 2. Mostrar formulario de nueva asignatura. 4. Validar datos del formulario. 6. Procesar datos de nueva asignatura.
Excepciones:	Si ocurre algún error con los datos del formulario, se muestra un mensaje al profesor.	

Cuadro 4.12: Crear Asignatura

Editar Asignatura		CU-09
Actores:	Profesor	
Descripción:	El profesor edita una asignatura.	
Precondición:	La asignatura ha sido previamente creada.	
Postcondición:	N/A	
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: Editar Asignatura. 3. Insertar datos en el formulario. 5. Seleccionar opción: Editar. 7. Navegación página principal	Respuesta del sistema 2. Mostrar formulario de edición de asignatura. 4. Validar datos del formulario. 6. Procesar datos de edición asignatura.
Excepciones:	Si ocurre algún error con los datos del formulario, se muestra un mensaje al profesor.	

Cuadro 4.13: Editar Asignatura

Eliminar Asignatura		CU-10
Actores:	Profesor	
Descripción:	El profesor elimina una asignatura.	
Precondición:	El profesor ha iniciado sesión en el sistema. El profesor ha creado una asignatura.	
Postcondición:	N/A	
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: Eliminar Asignatura. 3. Navegación página principal	Respuesta del sistema 2. Procesar borrado de asignatura.
Excepciones:	Si ocurre algún error en el proceso de borrado, se muestra un mensaje al profesor.	

Cuadro 4.14: Eliminar Asignatura

Alta estudiantes en asignatura			CU-11
Actores:	Profesor		
Descripción:	El profesor da de alta una lista de alumnos en una asignatura		
Precondición:	El profesor ha iniciado sesión en el sistema. El profesor ha creado una asignatura.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: Matricular alumnos. 3. Navegación página actual (asignatura)	Respuesta del sistema 2. Procesar matriculación y asignación de alumnos en asignatura.	
Excepciones:	Si ocurre algún error en el proceso de matriculación, se muestra un mensaje al profesor.		

Cuadro 4.15: Alta estudiantes en asignatura

Eliminar estudiante de una asignatura			CU-12
Actores:	Profesores		
Descripción:	El profesor elimina a un alumno de una asignatura.		
Precondición:	El profesor ha creado una asignatura y ha dado de alta al alumno en esa asignatura.		
Postcondición:	Se eliminarán todos los registros de asistencia en dicha asignatura.		
Flujo de eventos:	Actividades del actor 1. . Seleccionar opción: Eliminar estudiante. 3. Navegación página actual (lista de alumnos)	Respuesta del sistema 2. Procesar la eliminación del alumno en dicha asignatura.	
Excepciones:	Si ocurre algún error al eliminar al alumno de la asignatura, se muestra un mensaje al usuario.		

Cuadro 4.16: Eliminar estudiante de asignatura

Consultar lista de alumnos			CU-13
Actores:	Profesor		
Descripción:	El profesor consulta la lista de alumnos en una asignatura.		
Precondición:	Se han registrado alumnos en esa asignatura.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. . Seleccionar opción: Lista de alumnos.	Respuesta del sistema 2. Muestra la lista de alumnos en una nueva página.	
Excepciones:	Si ocurre algún error al mostrar la lista de alumnos, se muestra un mensaje al usuario.		

Cuadro 4.17: Consultar lista de alumnos

Consultar asistencias			CU-14
Actores:	Profesor		
Descripción:	El profesor consulta las asistencias de un dia específico.		
Precondición:	Se han registrado asistencias en esa asignatura.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. . Seleccionar día en el calendario.	Respuesta del sistema 2. Mostrar lista de asistencias para ese día.	
Excepciones:	Si ocurre algún error al mostrar las asistencias, se muestra un mensaje al usuario.		

Cuadro 4.18: Consultar asistencias

Consultar asistencias			CU-15
Actores:	Profesor		
Descripción:	El profesor consulta las asistencias de un dia específico.		
Precondición:	Se han registrado asistencias en esa asignatura.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: asignar horario. 3. Introducir datos de horario. 4. Seleccionar opción: asignar. 6. Navegación vista principal.	Respuesta del sistema 2. Mostrar formulario con horas disponibles. 5. Procesar datos del horario.	
Excepciones:	Si ocurre algún error al crear el horario, se muestra un mensaje al usuario.		

Cuadro 4.19: Asignar horario a asignatura

Editar Horario de Asignatura			CU-09
Actores:	Profesor		
Descripción:	El profesor edita el horario de una asignatura.		
Precondición:	El horario ha sido previamente creado.		
Postcondición:	N/A		
Flujo de eventos:	Actividades del actor 1. Seleccionar opción: Editar Horario. 3. Insertar datos en el formulario. 5. Seleccionar opción: Editar. 7. Navegación página principal	Respuesta del sistema 2. Mostrar formulario de edición de horario. 4. Validar datos del formulario. 6. Procesar datos de edición horario.	
Excepciones:	Si ocurre algún error con los datos del formulario, se muestra un mensaje al profesor.		

Cuadro 4.20: Editar Horario Asignatura

4.4. Diagramas de Casos de Uso

4.4.1. Sistema Gestión de Usuarios

Este diagrama engloba dos casos en nuestro proyecto:

- **Administrador y profesor:** El administrador del sistema se encarga de crear los perfiles de los profesores y de su gestión. El profesor es el implicado que se encarga de iniciar/cerrar sesión en la aplicación web.
- **Profesor y alumno:** El profesor se encarga de crear los perfiles de los alumnos al añadirlos a una asignatura. El alumno es el implicado que inicia/cierra sesión en la aplicación móvil del alumno para registrar su asistencia.

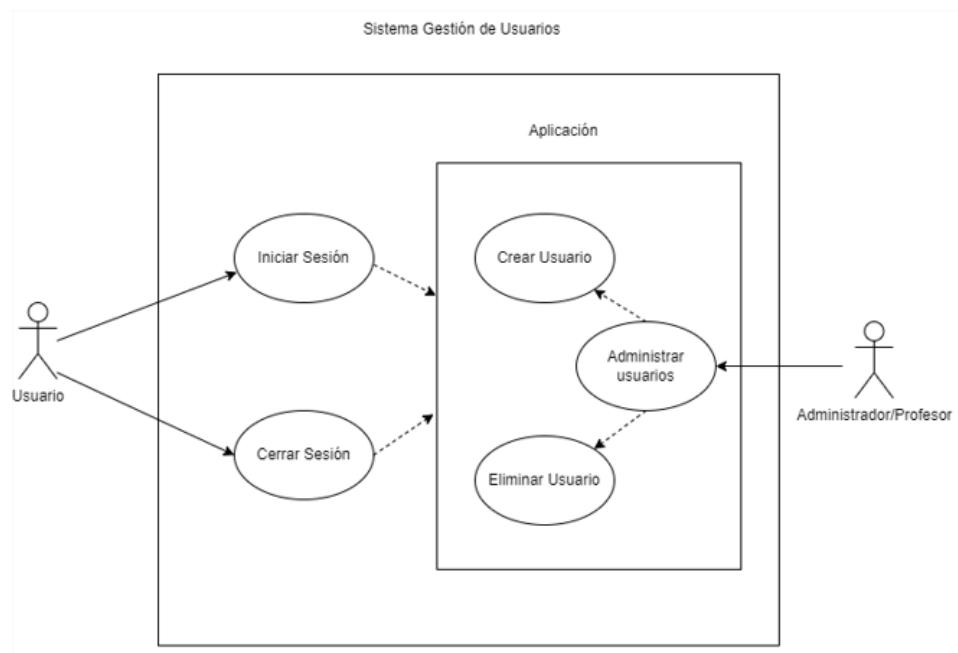


Figura 4.1: Sistema Gestión de Usuarios.

4.4.2. Sistema Aplicación Web

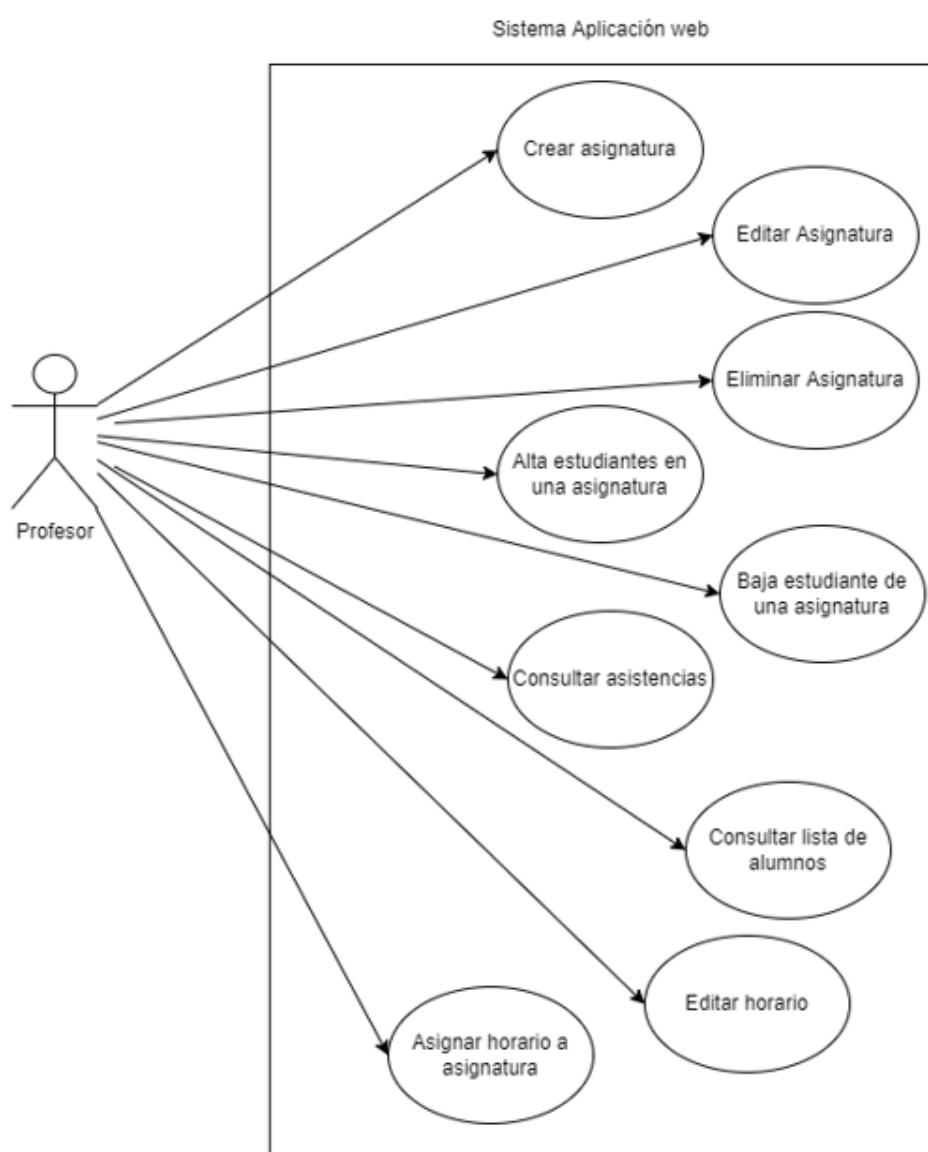


Figura 4.2: Sistema aplicación web.

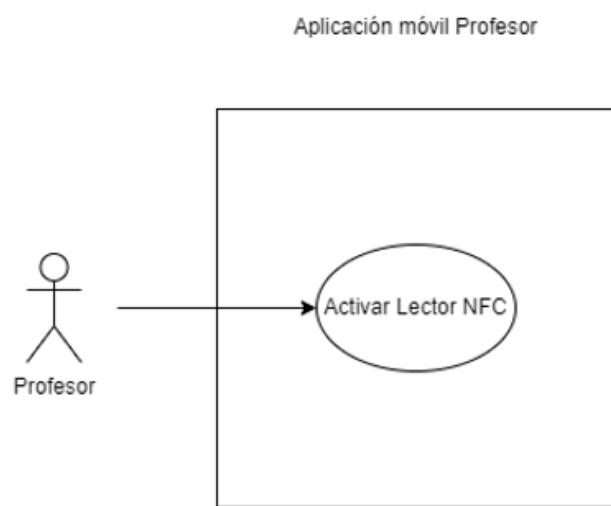
4.4.3. Sistema Aplicación Móvil Profesor

Figura 4.3: Aplicación móvil profesor.

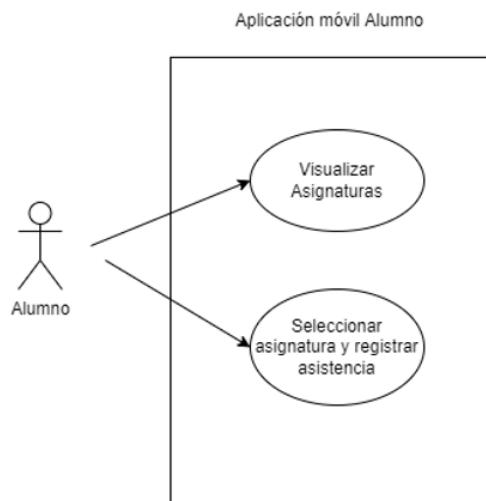
4.4.4. Sistema Aplicación Móvil Alumno

Figura 4.4: Aplicación móvil alumno.

4.5. Diagrama de Secuencia

En este apartado incluyo los diagramas de secuencia referentes a las distintas funcionalidades/operaciones de la aplicación. El objetivo de esta diferenciación es dejar claro qué cosa representa cada una de las partes del patrón de arquitectura de software Modelo-Vista-Controlador.

4.5.1. Inicio/cierre de sesión

Estos diagramas de secuencia reflejan los procesos de inicio y cierre de sesión de los usuarios.

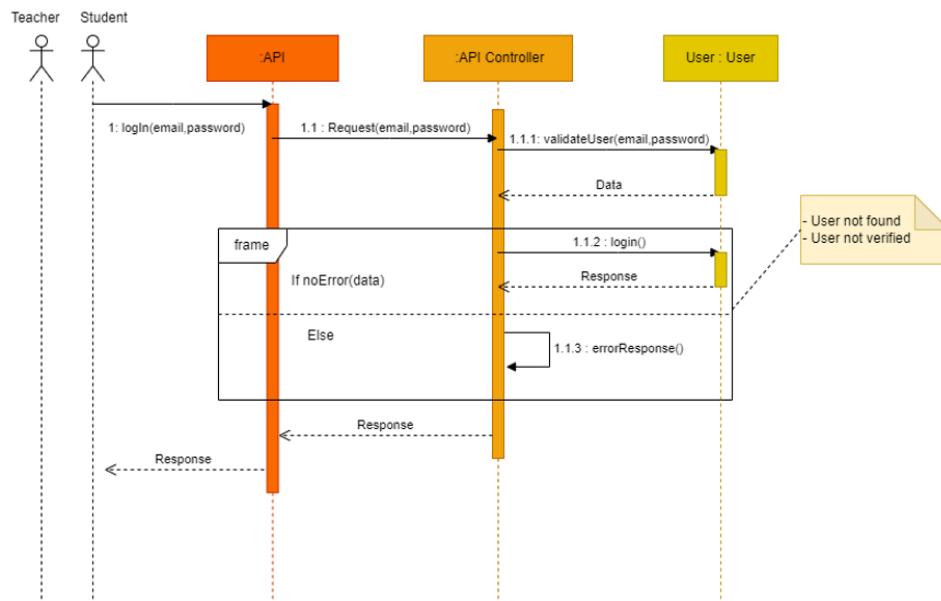


Figura 4.5: Iniciar Sesión.

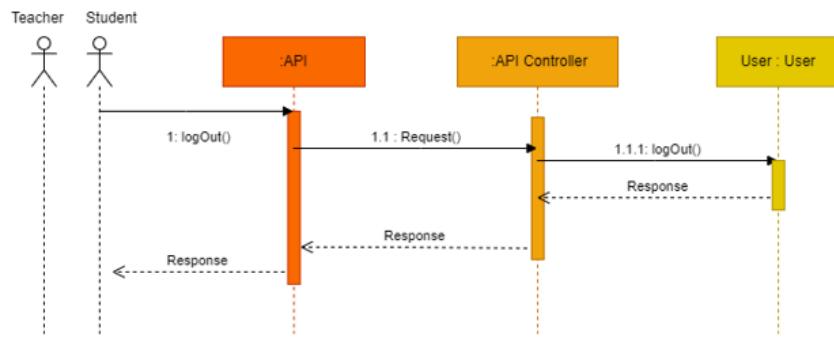


Figura 4.6: Cierre Sesión.

4.5.2. Crear/eliminar profesor

Estos diagramas de secuencia hacen referencia al proceso de crear o eliminar un profesor por parte del administrador del sistema.

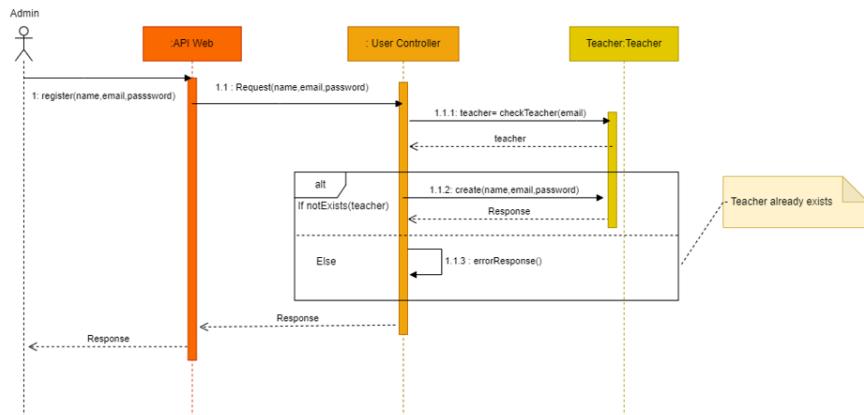


Figura 4.7: Crear Profesor.

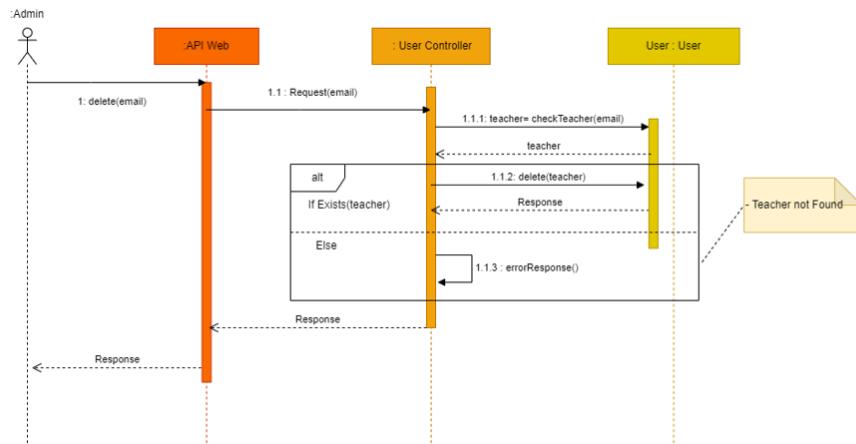


Figura 4.8: Eliminar Profesor.

4.5.3. Crear/eliminar asignatura

El diagrama de secuencia de eliminar una asignatura sirve para el proceso que invoca el profesor para eliminarla de su perfil.

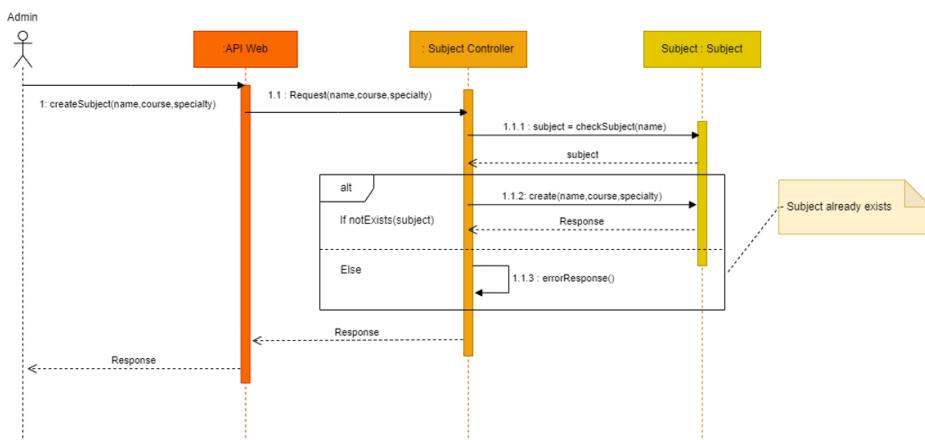


Figura 4.9: Crear Asignatura.

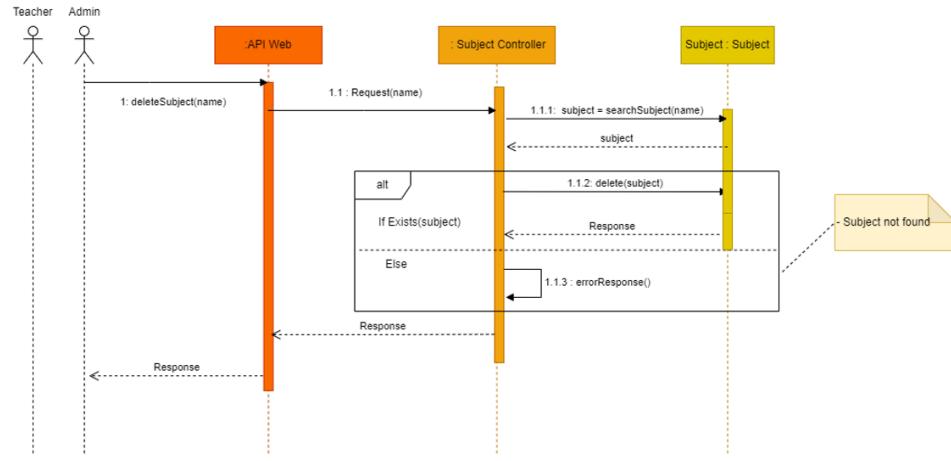


Figura 4.10: Eliminar Asignatura.

4.5.4. Editar asignatura

Este diagrama de secuencia hace referencia al proceso de editar una asignatura.

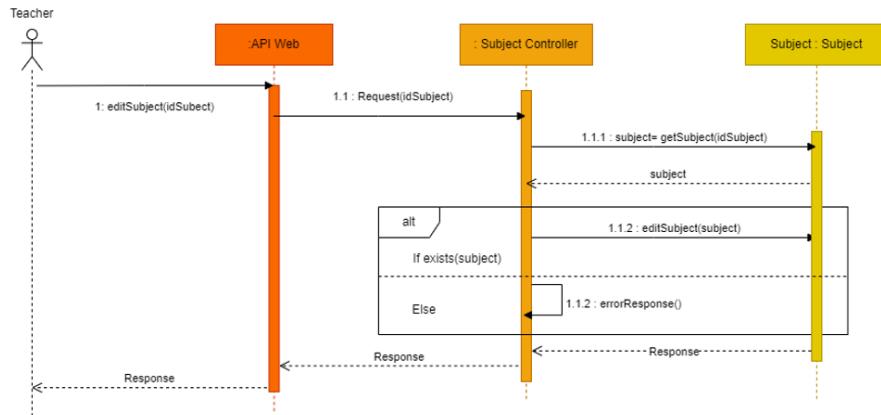


Figura 4.11: Editar Asignatura.

4.5.5. Consultar asistencias

Este diagrama de secuencia se hace referencia al proceso de consultar las asistencias de todos los alumnos de una determinada asignatura en un día determinado.

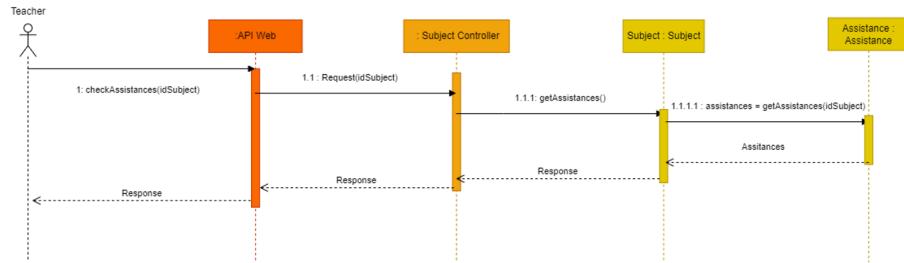


Figura 4.12: Consultar asistencias.

4.5.6. Lista de alumnos

En este diagrama de secuencia se hace referencia al proceso de lista los alumnos de una determinada asignatura.

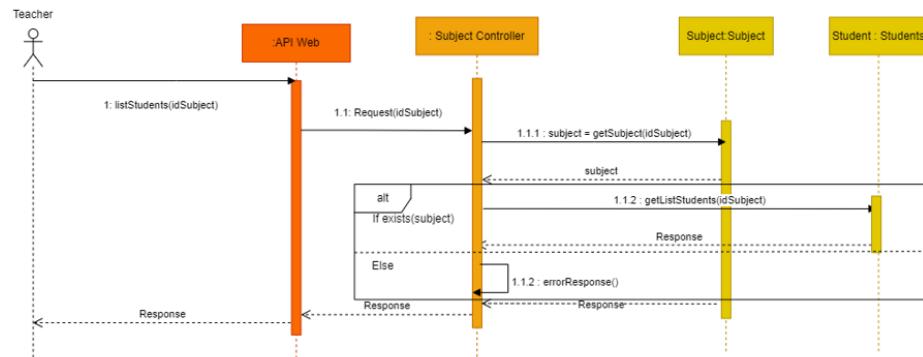


Figura 4.13: Lista de Alumnos.

4.5.7. Alta estudiantes/ baja estudiante

Estos diagramas de secuencia hacen referencia al proceso de dar de alta una lista de alumnos en una asignatura o eliminar a un alumno de una determinada asignatura.

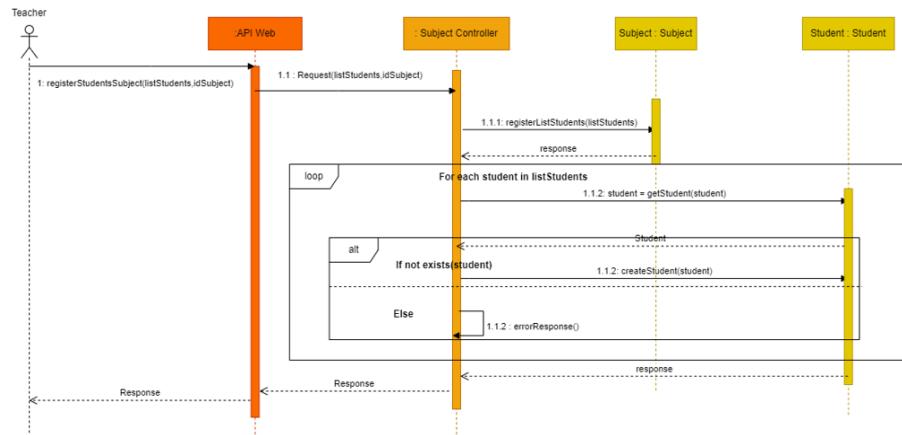


Figura 4.14: Alta alumnos.

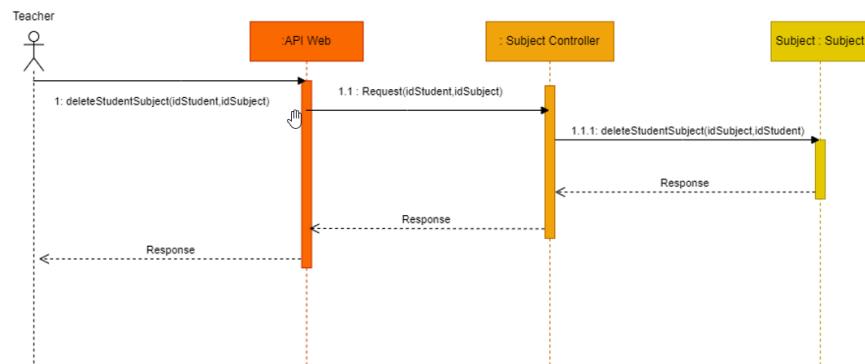


Figura 4.15: Baja alumno.

4.5.8. Asignar horario

Este diagrama de secuencia refleja el proceso de agregar un horario a una asignatura determinada.

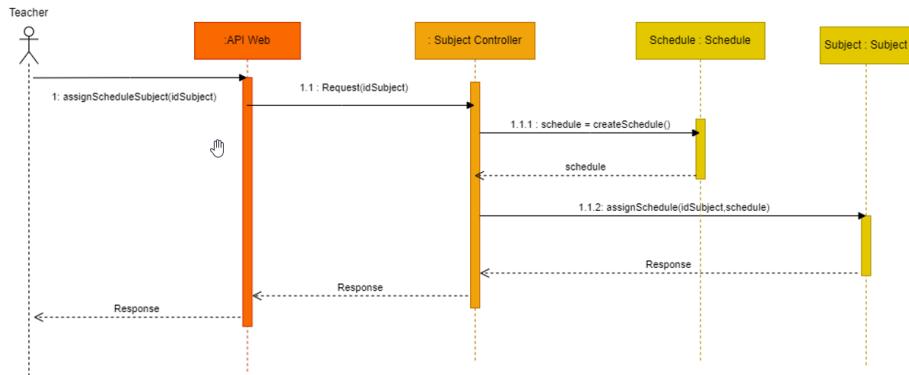


Figura 4.16: Asignar Horario.

4.5.9. Editar horario

Este diagrama de secuencia refleja el proceso de editar un horario de una asignatura determinada.

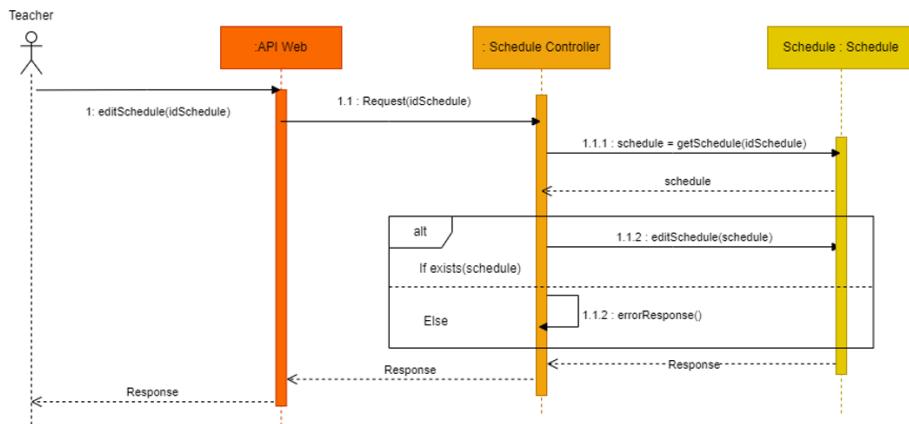


Figura 4.17: Editar Horario.

4.5.10. Visualizar asignaturas (App móvil alumno)

Este diagrama de secuencia hace referencia al proceso de visualizar la lista de asignaturas en las que está un alumno registrado.

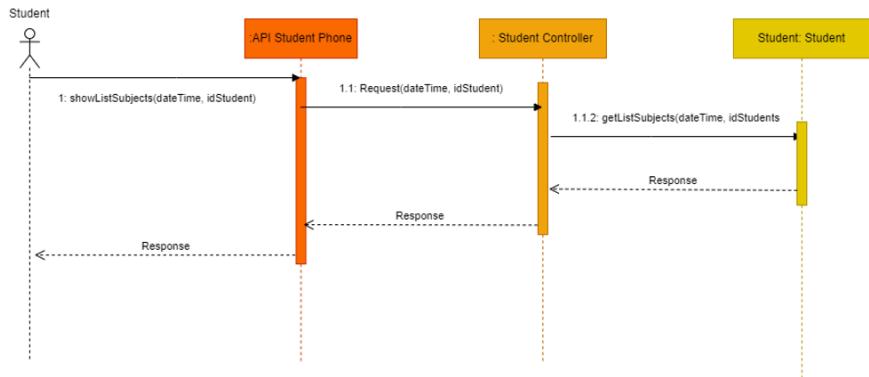


Figura 4.18: Visualizar Asignaturas.

4.5.11. Enviar datos a través de NFC

Este diagrama de secuencia refleja el proceso de enviar la información de una asignatura mediante la tecnología NFC.

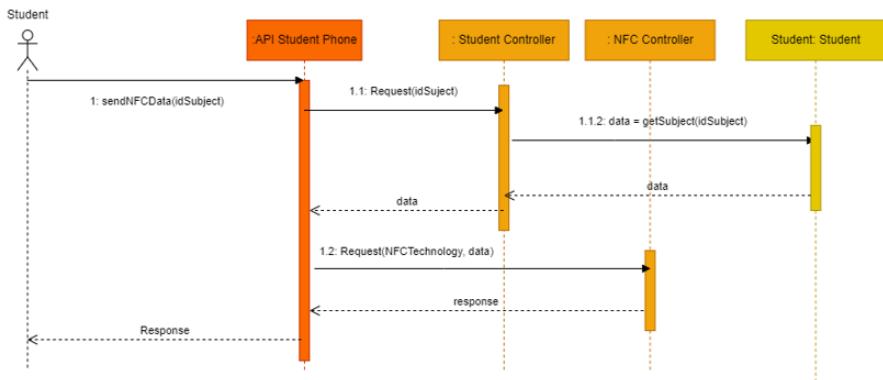


Figura 4.19: Enviar datos NFC.

4.5.12. Activar lector de NFC

Este diagrama de secuencia refleja el proceso de activar la tecnología NFC para recibir los datos enviados en el caso anterior.

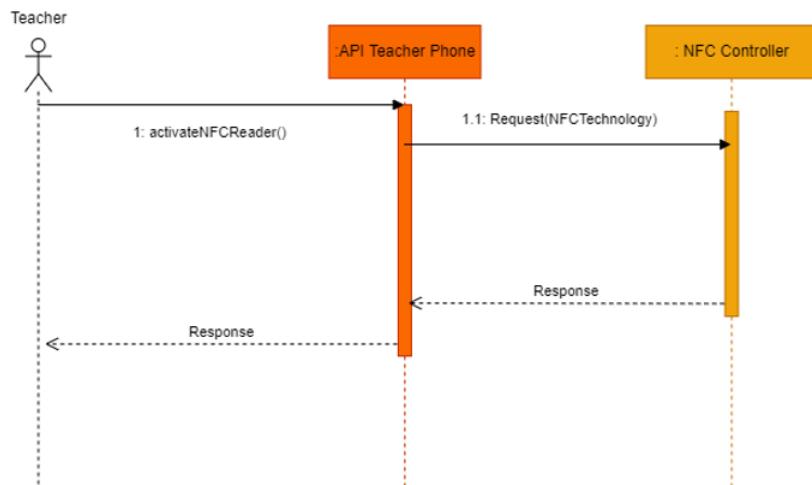


Figura 4.20: Lector NFC.

4.6. Diagramas Entidad-Relación

En este diagrama podemos ver la relaciones entre las distintas entidades que componen el sistema.

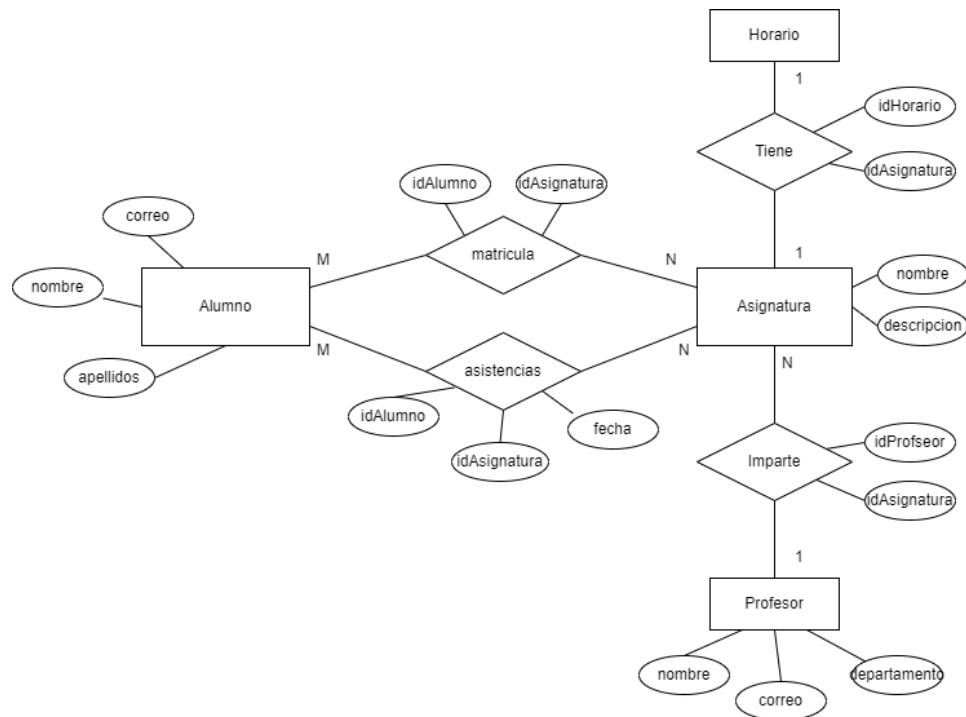


Figura 4.21: Diagrama E/R.

El paso a tablas se puede observar a continuación:

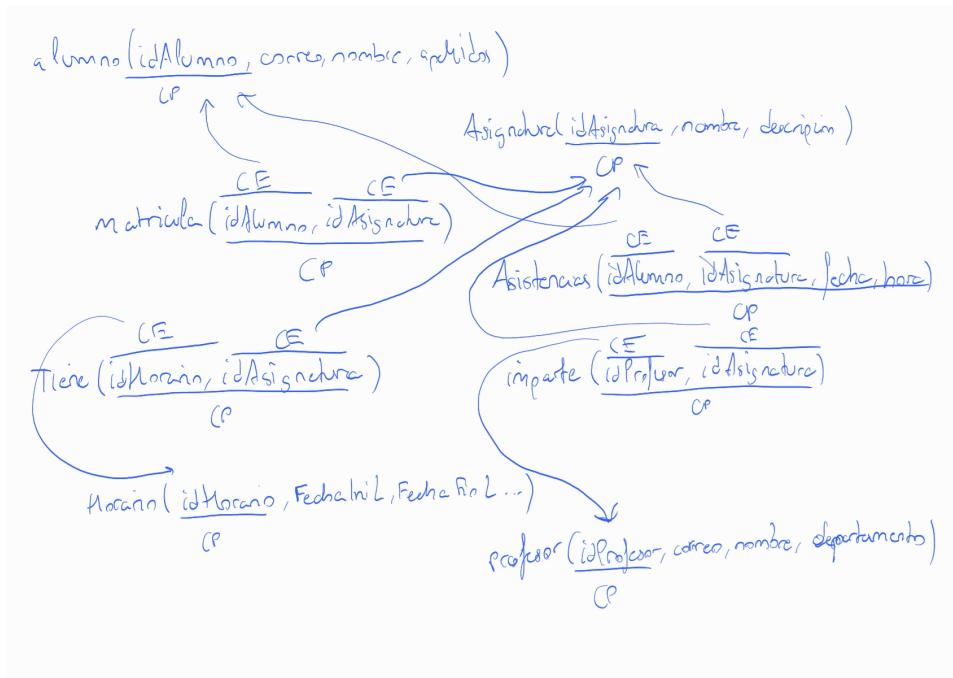


Figura 4.22: Paso a tablas.

En el paso a tablas se van a realizar las siguientes fusiones:

- Se va a fusionar la tabla alumnos con matrícula y así tener por cada alumno, una lista de asignaturas.
- Se fusiona la tabla horario con tiene para tener por cada horario el id de su asignatura correspondiente.
- Se fusiona la tabla asignatura con imparte, para que cada asignatura tenga el id del profesor correspondiente.

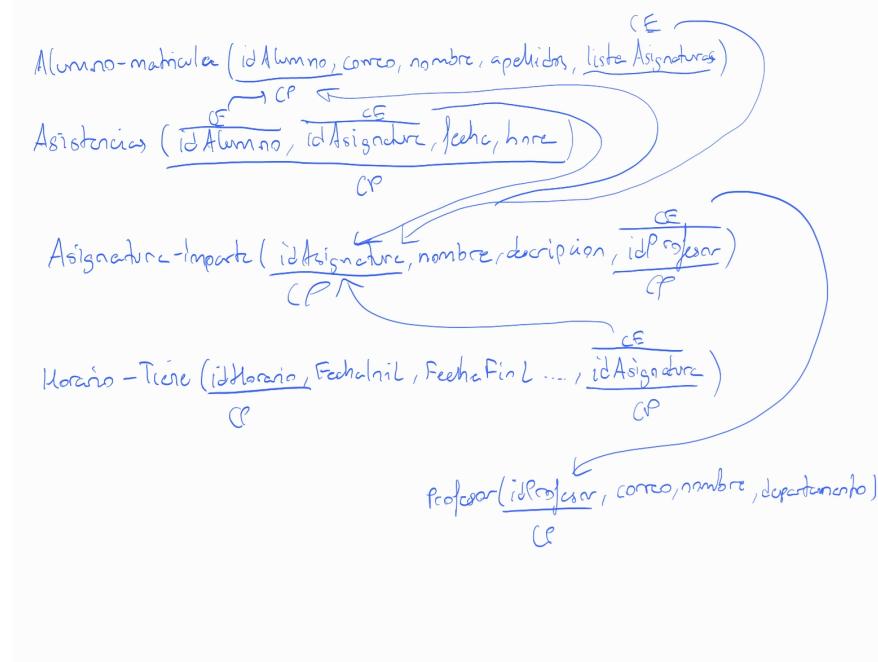


Figura 4.23: Fusión de tablas.

Capítulo 5

Diseño

5.1. Arquitectura del software

En esta sección muestro un esquema que representa la arquitectura general del software mediante el diagrama de paquetes. Como resumen, en el esquema podemos ver cuál es la estructura principal del proyecto.

El estudiante será el que haga uso directo de la interfaz de la aplicación móvil del alumno y el profesor será el que haga un uso directo de la interfaz de la aplicación web y de la aplicación móvil del profesor. El administrador solo interviene en la gestión de los profesores.

A continuación, presento cada uno de los módulos con sus funciones y demás utilidades:

- **Aplicación móvil alumno:** es un módulo básico ya que solo tiene el objetivo de comunicar la información del alumno que haya iniciado sesión, al módulo de la aplicación móvil del profesor.
- **Aplicación móvil profesor:** es un módulo que se encargará de enviar los datos para registrar la asistencia a la base de datos.
- **Aplicación web:** es un módulo complejo, carga con la mayoría de las funcionalidades del profesor como gestionar asignaturas, gestionar los usuarios que la componen, horarios...
- **Firebase Database:** este módulo se corresponderá con la base de datos y las operaciones que hagamos con dichos datos (insertado de datos, consultas ...) Será usado por todas las aplicaciones, la aplicación móvil del alumno con el objetivo de comprobar si existe un usuario y si se ha registrado la asistencia, la aplicación del profesor

para trasmirle los datos del alumno y la aplicación web para gestionar todos los datos del proyecto.

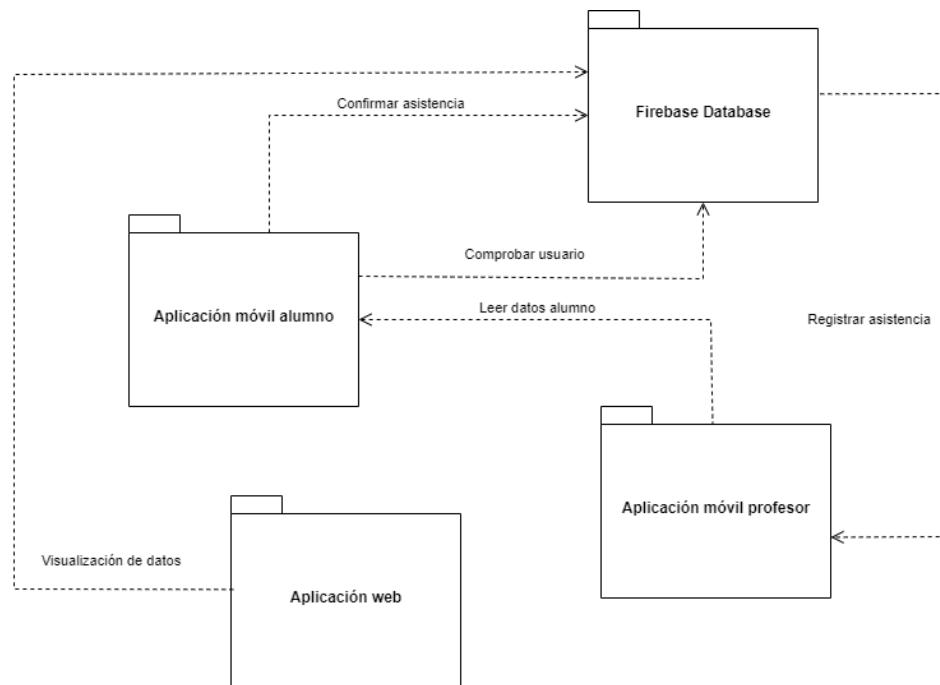


Figura 5.1: Diagrama de paquetes.

5.1.1. Diagrama de Base de datos

A continuación se muestra el Diseño del diagrama de la Base de Datos que se va a implementar, en este caso con Cloud Firebase, se explica con más detalle en el capítulo de la implementación.

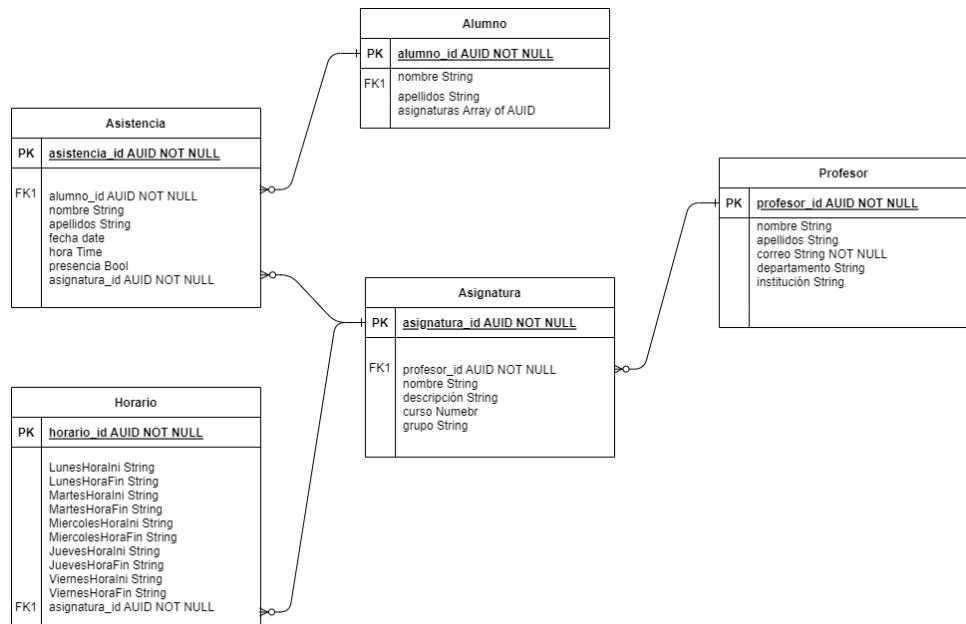


Figura 5.2: Diagrama Base de Datos.

5.2. Diseño de las interfaces de usuario

En esta sección se van a enseñar los bocetos de las interfaces que implementaremos en nuestro sistema, tanto de las apps móviles como de la app web.

5.2.1. Interfaz App web

En la siguiente imagen se puede observar como va a ser la página de inicio de sesión de la aplicación, dónde se le pedirá al usuario los datos para poder acceder al sistema, ya sea el administrador o el profesor.

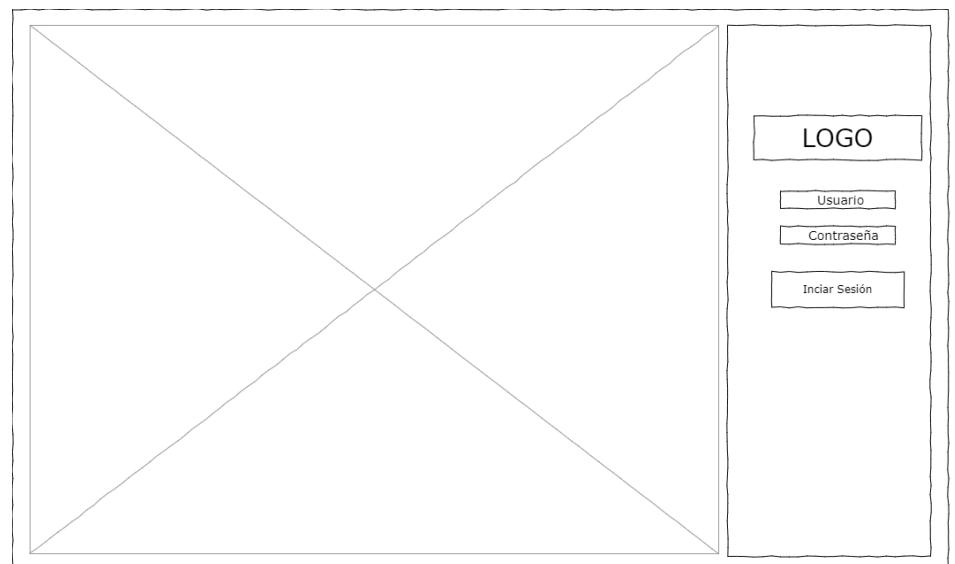


Figura 5.3: App Web. Inicio de Sesión.

En las siguientes imágenes tenemos la página principal de la aplicación, tanto la del administrador como la del profesor. La diferencia entre ambas es la opción de la barra de navegación y los elementos que se listan. En el caso del administrador tenemos la opción de dar de alta profesores y la correspondiente lista y en el caso del profesor, tenemos la opción de dar de alta asignaturas y listar las asignaturas.

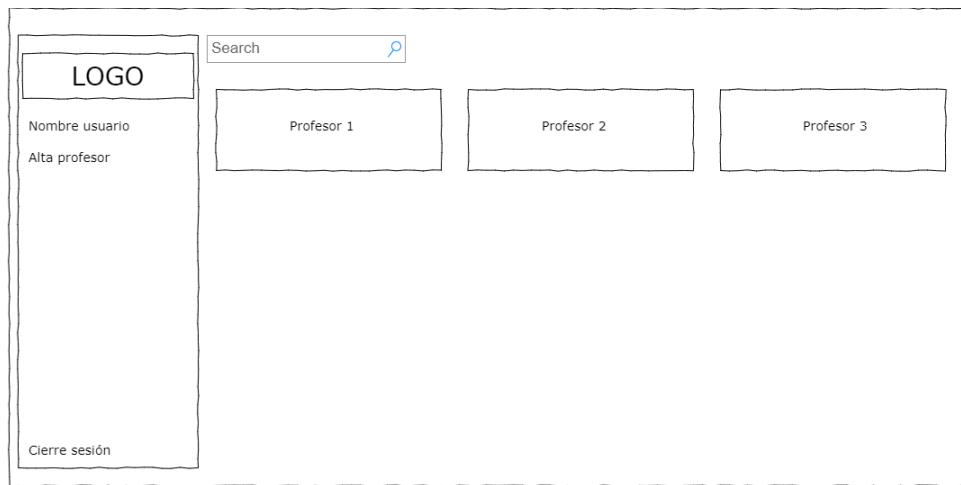


Figura 5.4: App Web. Página principal Administrador.

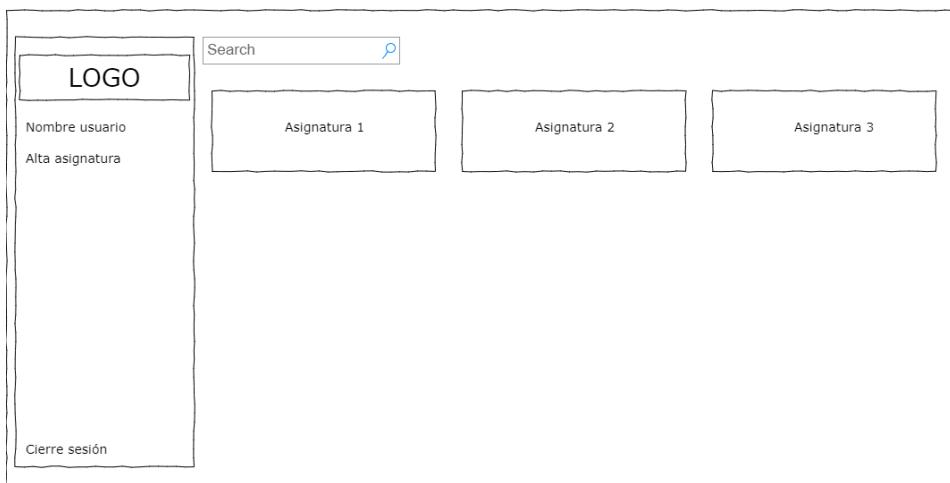


Figura 5.5: App Web. Página principal Profesor.

Finalmente, tenemos un boceto de lo que sería la página de cada asignatura, dónde nos encontraremos con el horario que tiene asignado, un calendario para consultar las asistencias en un día específico y una serie de opciones, editar el horario, consultar la lista de alumnos de la asignatura y cargar alumnos en dicha asignatura.

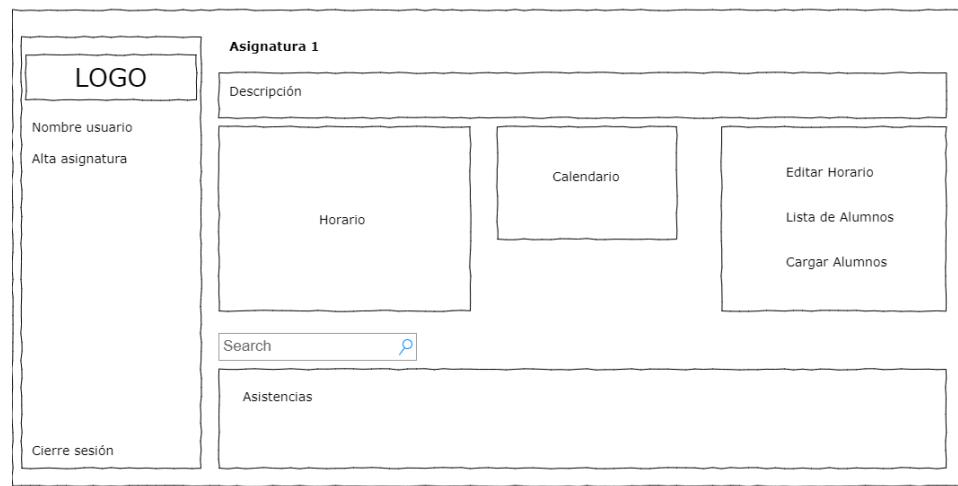


Figura 5.6: App Web. Página principal Profesor.

5.2.2. Interfaz App Móvil Alumnos

En la siguiente imagen se pueden observar las 3 vistas que tendremos en la aplicación móvil del alumno. Se tiene la vista de inicio de sesión (izquierda), página principal donde aparecerán las asignaturas disponibles en el momento para registrar asistencia (centro) y el detalle de cada asignatura donde se puede registrar la asistencia activando el sistema NFC (derecha).

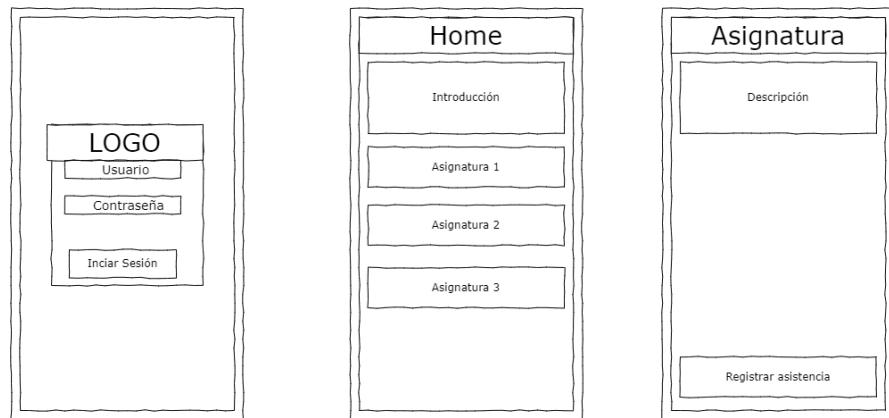


Figura 5.7: App Web. Página principal Profesor.

5.2.3. Interfaz App Móvil Profesor

La siguiente imagen contiene un diseño sencillo de la aplicación móvil del profesor, donde solo tendrá una opción para activar el lector NFC para recibir los datos de los alumnos y registrar la asistencia.

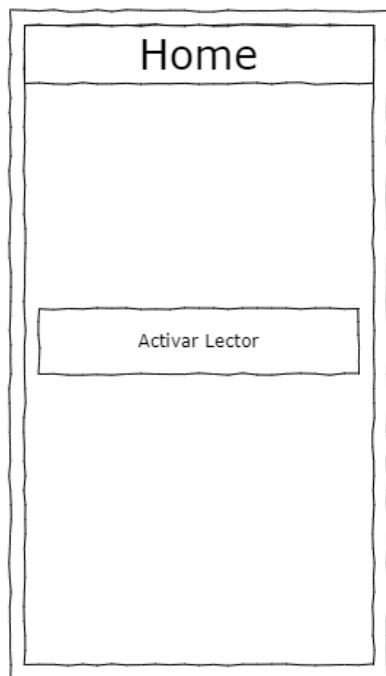


Figura 5.8: App Web. Página principal Profesor.

Capítulo 6

Implementación

6.1. Herramientas y software utilizado

En esta sección se va a hablar de las herramientas y el software utilizado en el proyecto.

El desarrollo de la aplicación web se ha realizado con el framework de React, una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Como lenguaje de programación, javascript utilizando las diferentes librerías y estilos que nos aportan.

Las aplicaciones móviles se han desarrollado con el framework de React-Native, un framework de código abierto creado por Meta Platforms, Inc. Se utiliza para desarrollar aplicaciones para Android, Android TV, iOS, macOS, tvOS, Web, Windows y UWP al permitir que los desarrolladores usen React con las características nativas de estas plataformas.

Como lenguaje de programación javascript al igual que en la aplicación web. [20]

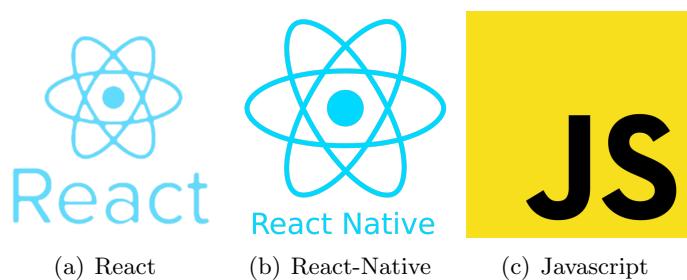


Figura 6.1: Logos de las herramientas utilizadas en las aplicaciones.

Se han escogido los frameworks React y React-native porque son herramientas actuales y que cada vez se utilizan más debido a su comodidad y facilidad para crear entornos e interfaces para las aplicaciones web y móviles. También se quería aprender a utilizar para en un futuro poder trabajar usándolas. El lenguaje de programación que utilizan es Javascript y por eso es el que se ha utilizado, también es sencillo y fácil de utilizar, de ahí su elección.[21]

Como base de datos se ha utilizado Firebase Cloud Firestore, una base de datos noSql en la nube.

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

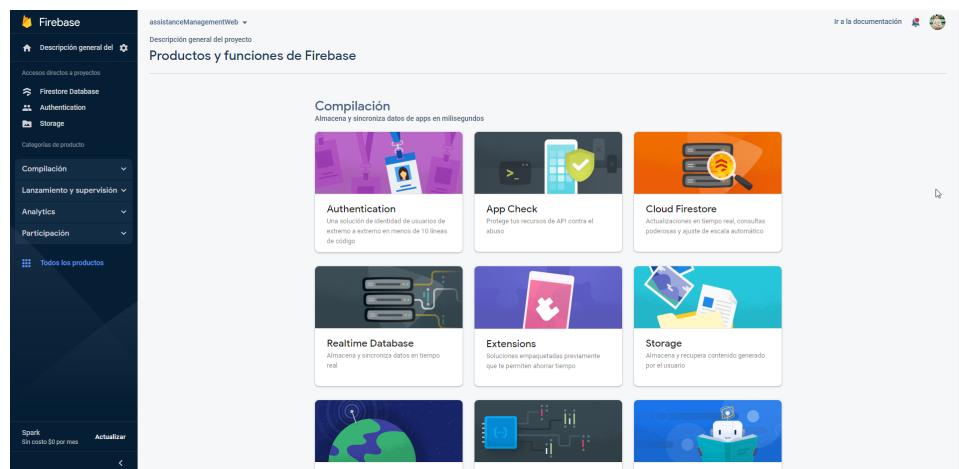


Figura 6.2: Interfaz Firebase.

Las herramientas utilizadas de Firebase en este proyecto son:[22]

- **Authentication:** una solución de identidad de usuarios utilizada para los inicio de sesiones (LogIn) de las aplicaciones.

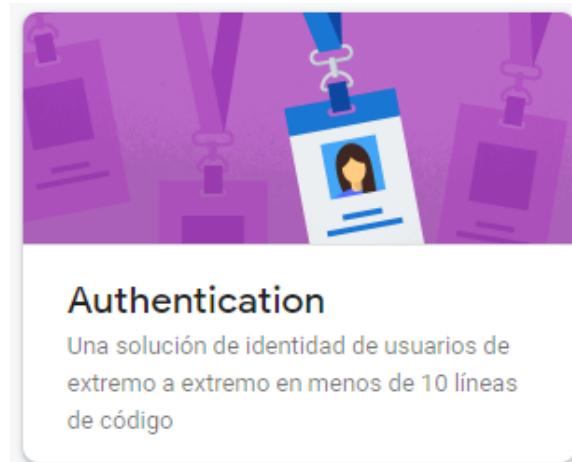


Figura 6.3: Firebase Authentication.

- **Cloud Firestore:** una solución que garantiza actualizaciones en tiempo real y consultas realmente poderosas, utilizada para todo el almacenamiento de datos de las aplicaciones.



Figura 6.4: Firebase Cloud Firestore.

Se ha elegido esta plataforma sobre todo para aprender sobre sus herramientas y por las siguientes ventajas que ofrecen a la hora de desarrollar aplicaciones:

- Sincronizar fácilmente los datos de sus proyectos sin tener que administrar conexiones o escribir lógica de sincronización compleja.
- Usa un conjunto de herramientas multiplataforma: se integra fácilmente para plataformas web como en aplicaciones móviles. Es compatible con grandes plataformas, como IOS, Android, aplicaciones web, Unity y C++.
- Crear proyectos sin necesidad de un servidor: Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.

Para el desarrollo del código se ha utilizado Visual Studio Code en un sistema operativo Windows 11 y para el desarrollo de este documento, se ha utilizado LaTex con el IDE online Overleaf.

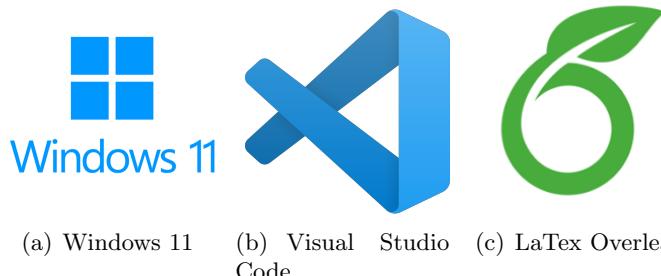


Figura 6.5: Logos de las herramientas utilizadas para el desarrollo del documento y del proyecto.

6.2. Implementación de la base de datos

Para realizar la conexión con Firebase se ha utilizado el siguiente código:

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore, connectFirestoreEmulator } from 'firebase/firestore';
import { initializeFirestore } from 'firebase/firestore';

const firebaseConfig = {
  apiKey: "AIzaSyBflzJExVG3I6Vq2i3JpMGGx_53iFTdylM",
  authDomain: "assistancemanagementweb.firebaseio.com",
  projectId: "assistancemanagementweb",
  storageBucket: "assistancemanagementweb.appspot.com",
  messagingSenderId: "680753831655",
  appId: "1:680753831655:web:fe48623ccf0425ab404c42",
  measurementId: "G-VZRKGQJWPR"
};

const app = initializeApp(firebaseConfig)
const auth = getAuth()
//const db = getFirestore()
//connectFirestoreEmulator(app, 'localhost', 8081);
const db = initializeFirestore(app, {
  experimentalForceLongPolling: true,
});

export [auth, db]
```

Figura 6.6: Conexión Firebase.

6.2.1. Authentication

Para la parte de Authentication se ha creado la variable *auth* que se utilizará para los inicios de sesión y para la conexión a la base de datos de Cloud Firestore, se utiliza la función, que nos da la biblioteca de Firebase, '*initializeFirestore*'.

Las funciones para realizar el inicio y cierre de sesión y que nos proporciona firebase son las siguientes. Se utilizan en la vista del login de la aplicación web y de la aplicación móvil de los alumnos para poder comprobar los datos introducidos de los usuarios.

```

import { createContext, useContext, useEffect, useState } from 'react';
import { createUserWithEmailAndPassword, signInWithEmailAndPassword, onAuthStateChanged, signOut } from 'firebase/auth'
import { auth } from '../firebase-config'

export const authContext = createContext()

export const useAuth = () => {
  const context = useContext(authContext)
  if (!context) {
    throw new Error('There is no auth provider')
  }
  return context
}

export function AuthProvider({children}){
  const [user, setUser] = useState(null)
  const [loading, setLoading] = useState(true)

  const signup = (email, password) => createUserWithEmailAndPassword(auth, email, password)
  const login = async (email, password) => signInWithEmailAndPassword(auth, email, password)

  const logout = () => {
    signOut(auth)
    setLoading(true)
  }

}

```

Figura 6.7: Funciones Inicio/Cierre Sesión.

6.2.2. Cloud Firestore

Cloud Firestore es una base de datos NoSQL orientada a los documentos. A diferencia de una base de datos SQL, no hay tablas ni filas; En su lugar, almacena los datos en documentos, que se organizan en colecciones.

Cada documento contiene un conjunto de pares clave-valor. Cloud Firestore está optimizado para almacenar grandes colecciones de documentos pequeños. Todos los documentos se deben almacenar en colecciones, y pueden contener subcolecciones y objetos anidados. Además, ambos pueden incluir campos primitivos, como strings, o tipos de objetos complejos, como listas.

Las colecciones y los documentos se crean de manera implícita en Cloud Firestore; solo debes asignar datos a un documento dentro de una colección. Si la colección o el documento no existen, Cloud Firestore los crea.

Documentos

En Cloud Firestore, la unidad de almacenamiento es el documento. Un documento es un registro liviano que contiene campos con valores asignados. Cada documento se identifica con un nombre.

Un documento que representa a un usuario alovelace puede tener el siguiente aspecto:

```
alovelace
first : "Ada"
last : "Lovelace"
born : 1815
```

Figura 6.8: Documento.

Tal vez te parezca que los documentos son muy similares a JSON; de hecho, básicamente son JSON. Existen algunas diferencias (por ejemplo, los documentos admiten tipos de datos adicionales y su tamaño se limita a 1 MB), pero en general, puedes tratar los documentos como registros JSON livianos.

Colecciones

Los documentos viven en colecciones, que simplemente son contenedores de documentos. Por ejemplo, podrías tener una colección llamada users con los distintos usuarios de tu app, en la que haya un documento que represente a cada uno:

```
users
alovelace
first : "Ada"
last : "Lovelace"
born : 1815
aturing
first : "Alan"
last : "Turing"
born : 1912
```

Figura 6.9: Coleccion.

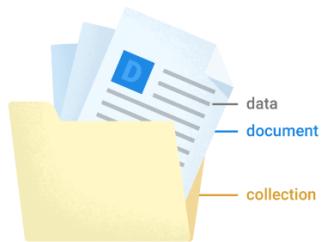


Figura 6.10: Estructura almacenamiento de los datos.

Cloud Firestore no usa esquemas, por lo que tienes libertad total sobre los campos que pones en cada documento y los tipos de datos que almacenas en esos campos. Los documentos dentro de una misma colección pueden contener campos diferentes o almacenar distintos tipos de datos en esos campos. Sin embargo, se recomienda usar los mismos campos y tipos de datos en varios documentos, de manera que puedas consultarlos con mayor facilidad.

Colecciones del sistema

A continuación se puede ver la interfaz de Cloud Firestore donde se encuentran las colecciones que se han creado para este proyecto. Como se observa en la imagen, tenemos

6.2.3. Ejemplos del código

En esta parte se podrán entender con ejemplos utilizados en el código sobre CRUD, las funciones de insertar, buscar, actualizar y borrar.

En la siguiente imagen se tiene el acceso a los datos de la colección de asistencias (`assistanceCollectionRef`) filtrando por la fecha actual y el id de la asignatura. Con la función `getDocs` de Cloud Firebase, se obtienen los datos de la consulta. En este caso se están inicializando los datos en una variable con `setAssistances`, donde se introducen cada una de las filas de la consulta y su id.

```

const onChange = async (date) => {
  const q = query(assistanceCollectionRef, where("fecha", "==", date.toDateString()), where("asignatura", "==", id))

  const data = await getDocs(q)

  setAssistances(data.docs.map((doc) => ({...doc.data(), id: doc.id})))
}

 setShow(true)
 setDate(date)
}

```

Figura 6.11: Lectura de datos Cloud Firebase.

En la siguiente imagen se la función de actualizar datos. Primeramente, se obtiene la colección de un único estudiante con su idStudent y se almacena en studentRef. A continuación se obtiene esta colección con la función getDoc y las asignaturas de la misma. Lo que se pretende en esta función es actualizar la lista de asignaturas a la que pertenece un alumno, eliminando la asignatura de la lista y después actualizándola.

```

const studentsCollectionRef = collection(db, "students")

const [listStudents, setListStudents] = useState([]);
const navigate = useNavigate()

const{id} = useParams()

const deleteStudentFromSubject = async (idStudent) => {
  const studentRef = doc(db, "students", idStudent)
  const student = await getDoc(studentRef)

  var asignaturas = student.data().asignaturas
  var newAsignaturas = []
  newAsignaturas = asignaturas.filter((item) => item !== id)

  await updateDoc(studentRef, {
    asignaturas: newAsignaturas
  })
  window.location.reload()
}

```

Figura 6.12: Actualización de datos Cloud Firebase.

En esta imagen se va a crear un registro perteneciente a un profesor gracias a la función addDoc de Cloud Firebase que permite añadir a una colección (en este caso teacherCollectionRef) un nuevo registro. También se puede observar otra función muy útil 'sendPasswordResetEmail', que básicamente envía un email al correo que se ha registrado para que pueda establecer una contraseña que se desee.

Finalmente, se observan unas comprobaciones validando si el email se en-

cuentra en uso.

```
const createTeacher = async () => {
  const q = query(teacherCollectionRef, where("correo", "==", email))
  const querySnapshot = await getDocs(q)
  var contraseña = await generateP()

  if(querySnapshot.empty) {
    try{
      const res = await signup(email, contraseña)
      const user = res.user
      await addDoc(teacherCollectionRef,{uid:user.uid,nombre:nombre,apellidos:apellidos,departamento:departamento})
      //await sendPasswordResetEmail(auth,email)
      //await sendEmailVerification()
      navigate('/HomeAdmin')
    }catch(error){
      console.log(error)
      if(error.code === "auth/invalid-email"){
        setError("Correo invalido")
      }else if (error.code === "auth/email-already-in-use"){
        setError("Correo registrado")
      }
    }
  }else
    setError("correo en uso")
}
```

Figura 6.13: Crear nuevos registros con Cloud Firebase

En la siguiente imagen se puede observar la función que nos ofrece Cloud Firebase para eliminar un documento de una colección, en este caso una asignatura de la colección de objetos 'subjects'.

```
const deleteSubject= async(id) => {
  const subjectDoc = doc(db,"subjects",id)
  await deleteDoc(subjectDoc)
  window.location.reload()
}
```

Figura 6.14: Eliminar datos Cloud Firebase

6.3. Diagrama de Clases

En el Diagrama de Clases del Diseño se representan todas las clases necesarias en el proyecto, sus propiedades y tipo, y las relaciones entre ellas.

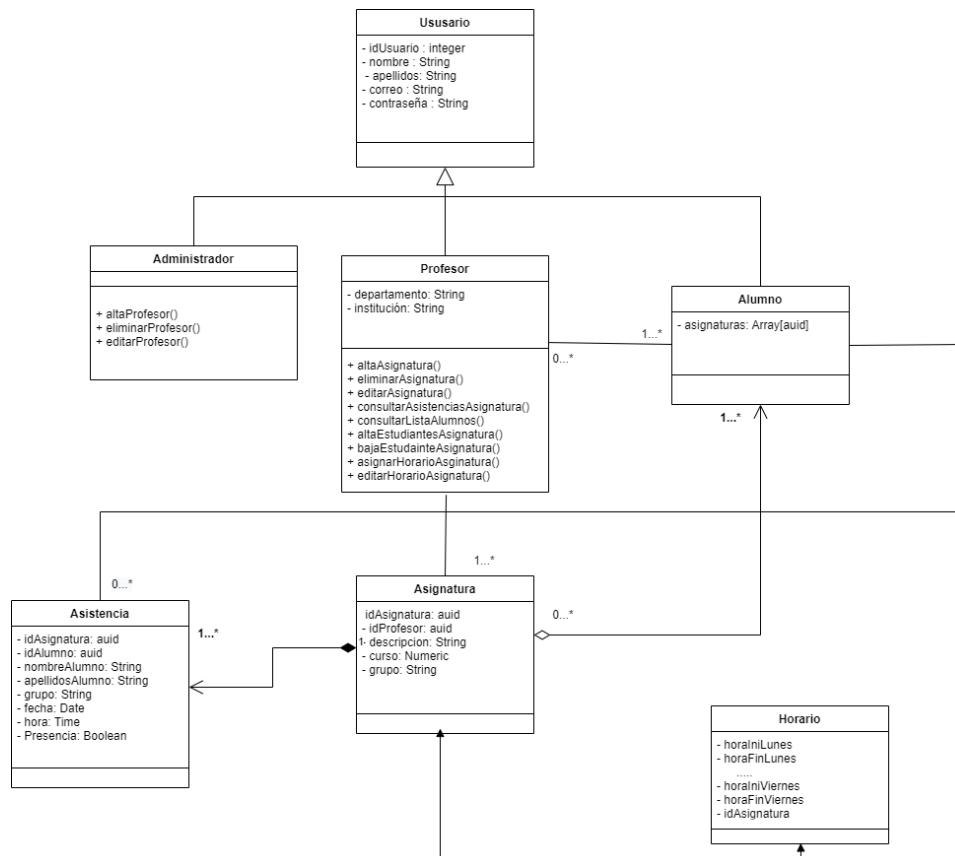


Figura 6.15: Diagrama de clases.

6.4. Detalles de implementación

En este apartado se detallan todos los objetivos que se han ido desarrollando a lo largo del desarrollo de la implementación del proyecto.

6.4.1. Estudio de las herramientas a utilizar

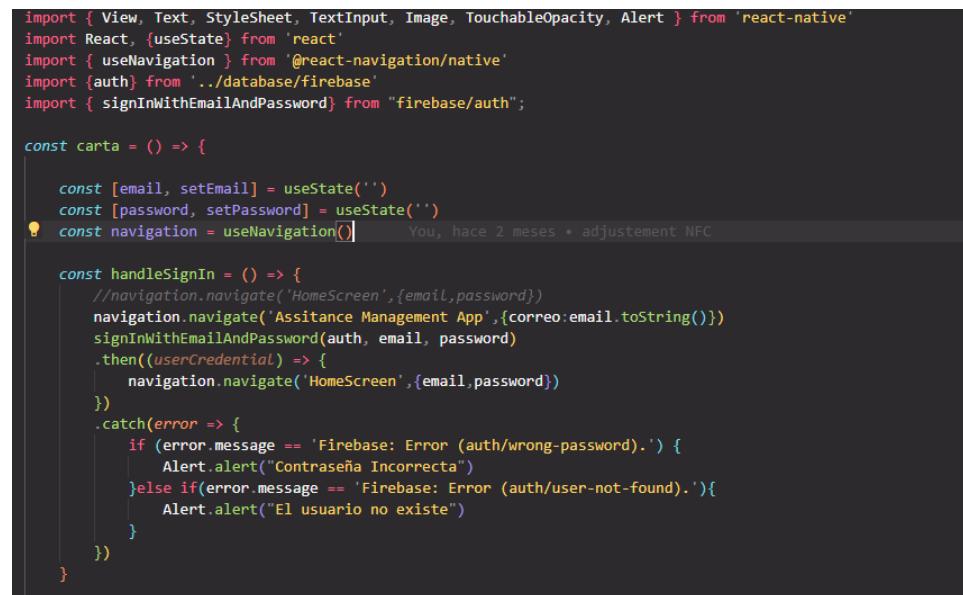
El objetivo de esta sección fue elegir las herramientas que se van a utilizar para el desarrollo del proyecto, intentando buscar aquellas con las que el desarrollo fuese mucho más eficiente y simple y con las que aprender para utilizarlas en un futuro. Las herramientas utilizadas se mencionan en el apartado anterior.

6.4.2. Despliegue App Móvil Alumnos

Primeramente, se realizó un previo estudio al desarrollo de la app para aprender a utilizar la tecnología con la que se iba a desarrollar y realizar algunas pruebas para entenderla.

Se comenzó creando las vistas de la aplicación:

- **Login**, es la vista donde el usuario introducirá sus credenciales para poder acceder al sistema. Para ello, se han tenido que importar la biblioteca para realizar la conexión con Firebase Authentication.



```
import { View, Text, StyleSheet, TextInput, Image, TouchableOpacity, Alert } from 'react-native'
import React, {useState} from 'react'
import { useNavigation } from '@react-navigation/native'
import {auth} from '../database.firebaseio'
import {signInWithEmailAndPassword} from "firebase/auth";

const carta = () => {

  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const navigation = useNavigation()

  const handleSignIn = () => {
    //navigation.navigate('HomeScreen',{email,password})
    navigation.navigate('Assitance Management App',{correo:email.toString()})
    signInWithEmailAndPassword(auth, email, password)
      .then((userCredential) => {
        navigation.navigate('HomeScreen',{email,password})
      })
      .catch(error => {
        if (error.message == 'Firebase: Error (auth/wrong-password).') {
          Alert.alert("Contraseña Incorrecta")
        }else if(error.message == 'Firebase: Error (auth/user-not-found).'){
          Alert.alert("El usuario no existe")
        }
      })
  }
}

export default carta
```

Figura 6.16: Conexión Authentication.

Esta conexión es muy simple ya que el propio firebase te da los datos para realizar la conexión con la aplicación creada en dicha plataforma.

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore, connectFirestoreEmulator } from 'firebase/firestore';
import { initializeFirestore } from 'firebase/firestore';

const firebaseConfig = {
  apiKey: "AIzaSyBflzJExVG3I6Vq2i3JpMGGx_53iFTdy1M",
  authDomain: "assistancemanagementweb.firebaseio.com",
  projectId: "assistancemanagementweb",
  storageBucket: "assistancemanagementweb.appspot.com",
  messagingSenderId: "680753831655",
  appId: "1:680753831655:web:fe40623ccf0425ab404c42",
  measurementId: "G-VZRKGQJWPR"
};

const app = initializeApp(firebaseConfig)
const auth = getAuth()
//const db = getFirestore()
//connectFirestoreEmulator(app, 'localhost', 8081);
const db = initializeFirestore(app, {
  experimentalForceLongPolling: true,
});

export [auth, db]
```

Figura 6.17: Conexión Firebase.

El resto de elementos de visualización, formulario... se pueden observar en el código de la aplicación.

- **Home Screen** es la vista donde van a aparecer la lista de asignaturas en las que el alumno esté registrado y estén disponibles en ese momento, es decir, corresponde con las horas en las que el horario de la asignatura está definido.
- **Subject Screen** es la vista donde podemos ver la información de la asignatura y un botón para registrar la asistencia. Con este botón lo que estamos haciendo es activar la tecnología NFC importada a través de la librería react-native-nfc-manager para generar un mensaje Ndef capaz de ser leído por un lector NFC.
El mensaje Ndef va a contener una cadena string con el id de la asignatura y el correo del alumno, para que en la app del lector se pueda

registrar la asistencia de dicho alumno en dicha asignatura.

```
import { Card, ListItem, Button, Icon } from 'react-native-elements'
import { Text, View, FlatList, TouchableOpacity, StyleSheet, Image, ScrollView, Alert } from 'react-native'
import React, { useState } from 'react'
import { useRoute } from '@react-navigation/native'
import NfcManager, { Ndef, NfcEvents } from 'react-native-nfc-manager'

const Subject = () => {
  const route = useRoute()

  let simulation
  const [content, setContent] = useState({
    correo: route.params.subject.profesor,
    idAsignatura: route.params.subject.id
  })

  const registrarAsistencia = () => {
    Alert.alert(
      '',
      'Acerca el dispositivo al lector NFC',
      [
        {text: 'OK', onPress: () => console.log('OK Pressed')},
      ]
    );
    NfcManager.start()
    let content = route.params.subject.id + route.params.correoAlumno
    const bytes = Ndef.encodeMessage([ Ndef.textRecord('"' + content) ])
    // Stores message to Senders phone. Now other android with enabled NFC can receive it
    NfcManager.setNdefPushMessage(bytes)
  }
}
```

Figura 6.18: Activar NFC message.

Básicamente, la aplicación móvil del alumno está formada por esas 3 vistas, que realizan la consulta de datos a Firebase Authentication y Firebase Realtime Database.

En alguno tutoriales sobre react-native y javascript, se observó que había dos maneras de desplegar la aplicación:

- Mediante **Expo**, un ecosistema de herramientas que facilitan el uso de React Native y apoyan el desarrollo, build e incluso publicación a las stores.
- Mediante **React-Native Client**, el cliente de react native.

Al ver la facilidad con la que se podía trabajar con expo, se decidió optar por esa alternativa. Para desplegar la app y generar el apk para ejecutarlo en el dispositivo móvil, se tenía que realizar una compilación de todos los ficheros que la hacía automáticamente expo con EAS (Expo Application Services), el comando eas build y eas submit.

Aquí se encuentra el primer error del proyecto y es que saltaba el error de que no podía encontrar ciertas librerías al hacer el build. Tras varias semanas de investigación, se observa que expo no es capaz de reconocer determinadas librerías, en nuestro caso, la del sistema NFC react-native-nfc-reader. Por ello, se tuvo que cambiar de opción y utilizar react-native client.

Una vez se conocía la forma correcta para generar el apk, bastaba con seguir lo siguientes comandos para generararlo:

- **react-native bundle** –platform android –dev false –entry-file index.js –bundle-output android/app/src/main/assets/index.android.bundle –assets-dest android/app/src/main/res/ El bundle es un formato de publicación que incluye todos los recursos y el código compilado de tu app, pero delega la generación del APK y la firma a Google Play. Este comando es necesario para poder construir la app antes de generar el apk.
- **./gradlew assembleRelease** Para generar el apk mediante gradle. Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Con lo cual, estamos compilando el código de manera rápida y eficiente. Además es el sistema de compilación oficial para Android y cuenta con soporte para diversas tecnologías y lenguajes.

Para no tener que generar constantemente el apk para poder probar la aplicación en un dispositivo android real, se utilizaba un emulador de android. Dicho emulador de android se arranca con el siguiente comando:

npx react-native run-android

Gracias a este comando, Metro, un empaquetador que realiza la compilación de muchos archivos javascript en un solo archivo, se compila la aplicación y se ejecuta directamente en el emulador dando así una interfaz de la aplicación con la que poder realizar múltiples pruebas.

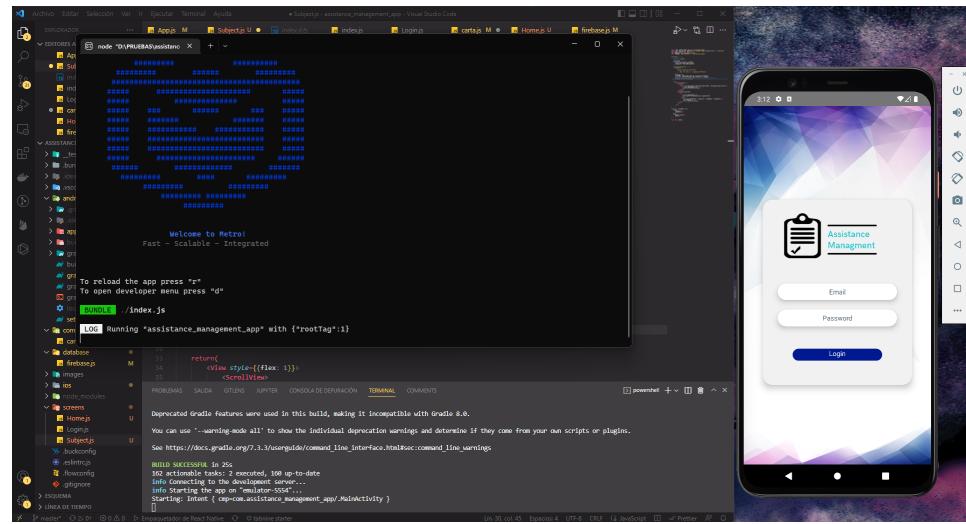


Figura 6.19: Arrancado aplicación móvil alumno.

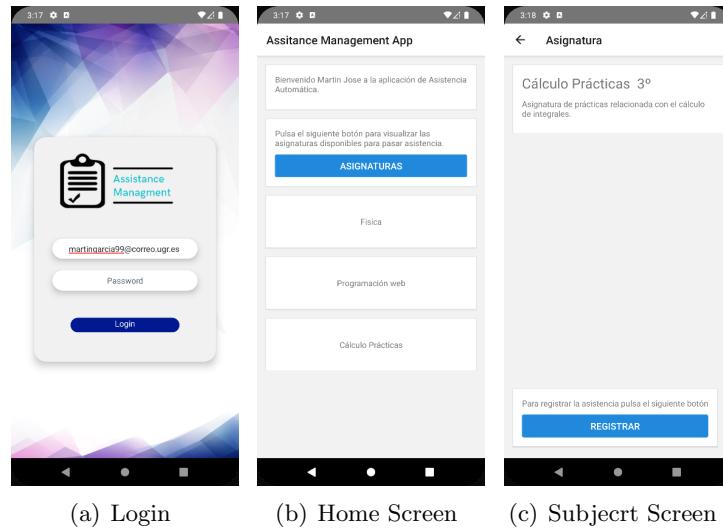
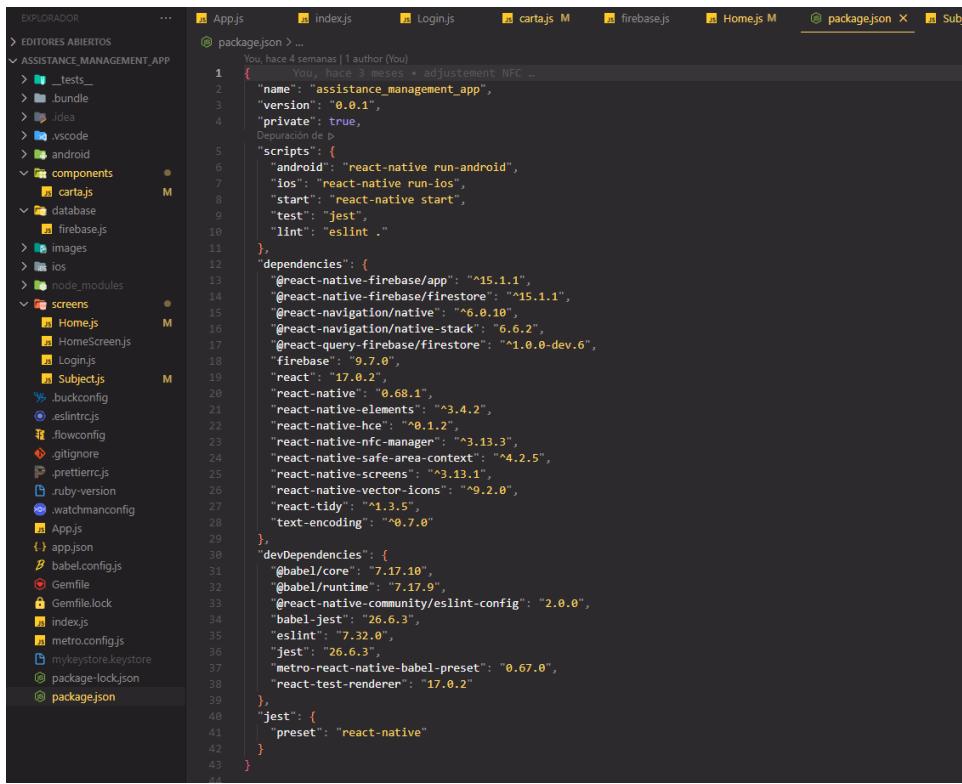


Figura 6.20: Vistas de la aplicación móvil alumno.

Dependencias instaladas (package.json) y directorios y ficheros



The screenshot shows a code editor interface with several tabs at the top: App.js, index.js, Login.js, cartajs M, firebase.js, Home.js M, package.json X, and Sub. The package.json tab is active, displaying the following JSON code:

```
You, hace 4 semanas | 1 author (You)
1 {
2   "name": "assistance_management_app",
3   "version": "0.0.1",
4   "private": true,
5   "scripts": {
6     "android": "react-native run-android",
7     "ios": "react-native run-ios",
8     "start": "react-native start",
9     "test": "jest",
10    "lint": "eslint ."
11  },
12  "dependencies": {
13    "@react-native-firebase/app": "^15.1.1",
14    "@react-native-firebase/firestore": "^15.1.1",
15    "@react-navigation/native": "^6.0.10",
16    "@react-navigation/native-stack": "6.6.2",
17    "@react-query-firebase/firestore": "1.0.0-dev.6",
18    "firebase": "9.7.0",
19    "react": "17.0.2",
20    "react-native": "0.68.1",
21    "react-native-elements": "^3.4.2",
22    "react-native-hce": "0.1.2",
23    "react-native-nfc-manager": "3.13.3",
24    "react-native-safe-area-context": "4.2.5",
25    "react-native-screens": "3.13.1",
26    "react-native-vector-icons": "9.2.0",
27    "react-tidy": "1.3.5",
28    "text-encoding": "0.7.0"
29  },
30  "devDependencies": {
31    "@babel/core": "7.17.10",
32    "@babel/runtime": "7.17.9",
33    "@react-native-community/eslint-config": "2.0.0",
34    "babel-jest": "26.6.3",
35    "eslint": "7.32.0",
36    "jest": "26.6.3",
37    "metro-react-native-babel-preset": "0.67.0",
38    "react-test-renderer": "17.0.2"
39  },
40  "jest": {
41    "preset": "react-native"
42  }
43 }
```

Figura 6.21: Dependencias, directorios y ficheros.

6.4.3. Despliegue App Móvil Profesor

Al principio se tuvo que investigar cómo realizar la conexión entre dos dispositivos móviles mediante NFC.

Esto llevó a realizar determinadas pruebas con distintas bibliotecas que manejan la tecnología NFC, entre ellas:

- **react-native-nfc-manager.**
- **react-native-nfc.**
- **react-native-nfc-hce.**
- **react-native-nfc-card-reader.**

Primeramente, se intentó la siguiente lógica. Un dispositivo móvil usaría la librería **react-native-nfc-hce** para emular un tag NFC, donde se escibe el mensaje que leerá el receptor. El otro dispositivo móvil se encargará de reconocer dicho dispositivo y leer el mensaje. Se tuvieron problemas con esta forma, ya que no se podía leer el mensaje del emisor con el lector, ya sea por desconocimiento del uso de la librería o que estaba sin mantener...

Finalmente, se probó con la librería **react-native-nfc-manager**, que se utiliza en ambas aplicaciones. En la del alumno para generar el mensaje NDEF y enviarlo y la del profesor para leer correctamente dicho mensaje.

Para probar la aplicación se sigue el mismo método usado en el punto anterior.

A continuación se muestra un pedazo de código donde se encuentra la función para registrar las asistencias una vez el alumno pasa su dispositivo por el lector NFC del profesor.

Se pueden observar las funciones para obtener la hora Actual (getHours(), getMinutes(), getSeconds()) y la fecha (date = new Date()). Se comprueba si existe ya alguna asistencia en esa asignatura para ese alumno en esa fecha y hora. Si no existe, se crea un nuevo registro en la colección de asistencias (assistanceCollectionRef).

```
const registerAssistance = async (idAlumno, idAsignatura) => {
    console.log("alumno", idAlumno)
    console.log("asignatura", idAsignatura)
    const studentRef = doc(db, 'students', idAlumno)
    const student = await getDoc(studentRef)
    let nombre = student.data().nombre
    let apellidos = student.data().apellidos

    let date = new Date()
    let hours = date.getHours();
    let minutes = date.getMinutes();
    let seconds = date.getSeconds();
    if (hours < 10) hours = "0" + hours;
    if (minutes < 10) minutes = "0" + minutes;
    if (seconds < 10) seconds = "0" + seconds;
    let horaActual = hours + ":" + minutes + ":" + seconds
    console.log(horaActual)
    console.log(date.toDateString())
    console.log(date)

    //comprobar para ese estudiante, esa asignatura y esa fecha si hay algun registro
    const q = query(assistanceCollectionRef, where("asignatura", "==", idAsignatura), where("alumno", "==", idAlumno), where("fecha", "==", date.toDateString()))
    const querySnapshot = await getDocs(q)

    if(querySnapshot.empty) {
        await addDoc(assistanceCollectionRef, {alumno: idAlumno, nombre: nombre, apellidos: apellidos, hora: horaActual, fecha: date.toDateString(), asignatura:idAsignatura})
    } else {
        setError("Ya se ha registrado la asistencia para el alumno " + nombre + " " + apellidos + " el dia " + date.toDateString())
    }
}
```

Figura 6.22: Registrar asistencia. App Móvil Profesor.

La interfaz de esta aplicación es muy sencilla, ya que el único objetivo de la misma es el de leer el mensaje NFC del emisor y registrar la asistencia. Por lo tanto, solo se cuenta con una única vista a la que el profesor accede al abrir la app.

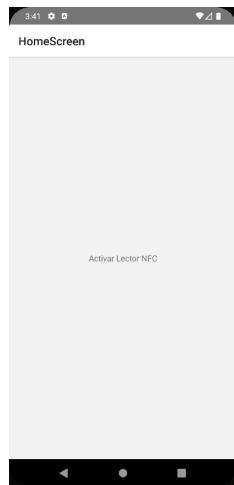


Figura 6.23: Home Screen App Profesor.

Dependencias instaladas (package.json) y directorios y ficheros

```
④ DOUTOREASAPP ⑤ package.json ⑥ dependencies ⑦ react
  > _tests_
  > .bundle
  > .vscode
  > android
  > components
    > NDefMessages
  > ios
  > node_modules
  > screens
    > HomeScreen.js
    > buckconfig
    > eslintrc.js
    > flowconfig
    > .gitignore
    > prettier.js
    > ruby-version
    > watchmanconfig
  > App.js
  > app.js
  > babel.config.js
  > Gemfile
  > Gemfile.lock
  > index.js
  > metro.config.js
  > package-lock.json
  > package.json 1
  1   {
  2     "name": "teacherEasyApp",
  3     "version": "0.0.1",
  4     "private": true,
  5     "scripts": {
  6       "android": "react-native run-android",
  7       "ios": "react-native run-ios",
  8       "start": "react-native start",
  9       "test": "jest",
 10       "lint": "eslint --ext .js --ext .jsx"
 11     },
 12     "dependencies": {
 13       "@great-navigation/native": "^6.0.11",
 14       "@great-navigation/native-stack": "^6.7.0",
 15       "react": "17.0.2",
 16       "react-native": "0.68.2",
 17       "react-native-fnc": "0.3.0",
 18       "react-native-fmc-manager": "3.13.5",
 19       "react-native-safe-area-context": "4.3.1",
 20       "react-native-screens": "3.14.1"
 21     },
 22     "devDependencies": {
 23       "@babel/core": "7.18.6",
 24       "@babel/runtime": "7.18.6",
 25       "@react-native-community/eslint-config": "2.0.0",
 26       "babel-jest": "26.6.3",
 27       "eslint": "7.38.0",
 28       "jest": "26.6.3",
 29       "metro-react-native-babel-preset": "0.67.0",
 30       "react-test-renderer": "17.0.2"
 31     }
 32     "jest": {
 33       "preset": "react-native"
 34     }
 35   }
 36 }
```

Figura 6.24: Dependencias, directorios y ficheros.

6.4.4. Despliegue App Web

En esta sección se va a explicar parte de la aplicación web, dando a conocer la estructura de la aplicación internamente gracias al framework de React y al lenguaje de programación javascript.

En la siguiente imagen se observa el archivo index.js que se encarga de renderizar toda nuestra aplicación 'App', es decir, actualizar los elementos de nuestra aplicación.

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import App from './App';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap-icons/font/bootstrap-icons.css';

const root = createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    |   |   <App/>      You, hace 4 meses • first commit ...
    |   |   </React.StrictMode>
);

```

Figura 6.25: Index.js

A continuación se encuentra toda la estructura de enrutamiento de la aplicación. El enrutamiento es la manera de desplazarse entre las vistas de la aplicación y como indicar a nuestro proyecto, cuál es el componente al que se va a desplazar y mostrar. Esto se consigue gracias a la biblioteca 'Route' que contiene un elemento principal Route que a su vez está formado por las distintas rutas 'routes'.

```

import { Routes, Route, BrowserRouter as Router } from 'react-router-dom';
import { AuthProvider } from './context/authContext';
import Login from './components/Login';
import HomeAdmin from './components/Admin/HomeAdmin';
import CreateTeacher from './components/Admin/CreateTeacher';
import EditTeacher from './components/Admin/EditTeacher';
import HomeTeacher from './components/Teacher/HomeTeacher';
import CreateSubject from './components/Teacher/CreateSubject';
import ShowSubject from './components/Teacher>ShowSubject';
import ShowStudentsSubject from './components/Teacher>ShowStudentsSubject';
import EditSubject from './components/Teacher>EditSubject';
import Editschedule from './components/Teacher>EditSchedule';
import CreateSchedule from './components/Teacher>CreateSchedule';
import SidebarLayout from './components/sidebarLayout';
import SidebarLayoutTeacher from './components/sidebarLayoutTeacher';
import { ProtectedRoute } from './components/ProtectedRoute';
import { ProtectedRouteTeacher } from './components/ProtectedRouteTeacher';

function App() {
  return (
    <Router>
      <div className="flex">
        <AuthProvider>
          <Routes>
            <Route path="/" index exact={true} element={<Login />}/>
            <Route element={<ProtectedRoute><SidebarLayout/></ProtectedRoute>}>
              <Route path="/HomeAdmin" element={<ProtectedRoute><HomeAdmin /></ProtectedRoute>}>
                <Route path="/createTeacher" element={<ProtectedRoute><CreateTeacher /></ProtectedRoute>}>
                  <Route path="/editTeacher/:id" element={<ProtectedRoute><EditTeacher /></ProtectedRoute>}>
                    </Route>
                  <Route element={<ProtectedRouteTeacher><SidebarLayoutTeacher/></ProtectedRouteTeacher>}>
                    <Route path="/HomeTeacher" element={<ProtectedRouteTeacher><HomeTeacher /></ProtectedRouteTeacher>}>
                      <Route path="/createSubject" element={<ProtectedRouteTeacher><CreateSubject /></ProtectedRouteTeacher>}>
                        <Route path="/showSubject/:id" element={<ProtectedRouteTeacher><ShowSubject /></ProtectedRouteTeacher>}>
                          <Route path="/editSubject/:id" element={<ProtectedRouteTeacher><EditSubject /></ProtectedRouteTeacher>}>
                            <Route path="/editSchedule/:id" element={<ProtectedRouteTeacher><EditSchedule /></ProtectedRouteTeacher>}>
                              <Route path="/createSchedule/:id" element={<ProtectedRouteTeacher><CreateSchedule /></ProtectedRouteTeacher>}>
                                <Route path="/showStudentsSubject/:id" element={<ProtectedRouteTeacher><ShowStudentsSubject /></ProtectedRouteTeacher>}>
                                  </Route>
                                </Route>
                              </Route>
                            </Route>
                          </Route>
                        </Route>
                      </Route>
                    </Route>
                  </Route>
                </Route>
              </Route>
            </Route>
          </Routes>
        </AuthProvider>
      </div>
    </Router>
  );
}

```

Figura 6.26: App.js

Ahora se va a mostrar cuál es la estructura de un componente (fichero) de React con la página principal de la aplicación. El componente 'HomeAdmin' esta compuesto por una parte donde se renderiza toda la vista y su código, y otra parte que contiene las distintas funciones para acceder a los datos que se van a mostrar.

En la parte del renderizado 'render()' tenemos el código HTML junto con algunos componentes importados de Bootstrap como los botones 'Button'. Para darle forma a estos elementos hemos utilizado Tailwind css, una biblioteca que contiene los estilos para que se pueda pintar cada componente al gusto del usuario.

```

return (
  <div className="p-7 text-2xl font-semibold flex-1 overflow-y-hidden h-screen">
    <h1>PÁGINA PRINCIPAL</h1>
    <input
      className="border-none hover:border-none p-2 flex-start"
      placeholder="Búsqueda de Profesores"
      name="term"
      onChange={e => setTerm(e.target.value)}
    />
    <div className="grid grid-cols-3 mt-6 overflow-y-scroll h-5/6">
      {teachers.filter(searchingTerm).map((teacher) => {
        return (
          <div className="card flex flex-row h-fit shadow-xl ml-2 mt-2 items-center">
            <BlUser size="2rem" className="ml-4"/>
            <div className="card-body">
              <p className="card-text">Nombre: {teacher.nombre}</p>
              <p className="card-text">Correo: {teacher.correo}</p>

              <Button className="text-center items-center justify-center" onClick={() => {editTeacher(teacher.id)}}>
                Editar
              </Button>
              <Button className="text-center items-center justify-center ml-2" onClick={() => {deleteTeacher(teacher.id, teacher.correo)}}>
                Eliminar
              </Button>
            </div>
          </div>
        );
      })
    </div>
  </div>
)

```

Figura 6.27: Renderizado del componente

En la parte de las funciones del componente, se puede observar la función 'deleteTeacher', donde se realizan varias consultas a Cloud Firebase para ir eliminando de manera anidada un profesor y todo lo que conlleva, es decir, asignaturas y alumnos de las mismas.

```

import React from 'react'
import {useState, useEffect} from 'react'
import {db} from '../../firebase-config'
import {collection, getDocs, deleteDoc, updateDoc, query, where, doc} from 'firebase/firestore'
import Button from 'react-bootstrap/Button'
import {useNavigate} from 'react-router-dom'
import {BlUser} from 'react-icons/bi'
import {auth} from '../../firebase-config'
import {useAuth} from '../../../../../context/authContext'

const HomeAdmin = () => {
  const [teachers, setTeachers] = useState([])
  const [term, setTerm] = useState('')
  const teacherCollectionRef = collection(db, "teachers")
  const subjectsCollectionRef = collection(db, "subjects")
  const navigate = useNavigate()

  const deleteTeacher = async(id, correo) => {
    //eliminar asignaturas creadas por ese profesor
    const q2 = query(subjectsCollectionRef, where("profesor", "==", correo))

    const querySnapshot = await getDocs(q2)

    querySnapshot.forEach(async(docu) => {

      const querySnapshot2 = await getDocs(collection(db, "students"));

      querySnapshot2.forEach(async(doc2) => {

        if(doc2.data().asignaturas.includes(docu.id)){
          const newAsignaturas = doc2.data().asignaturas.filter((item) => item !== docu.id)
          const ref = doc(db, "students", doc2.id)
          await updateDoc(ref, {
            asignaturas: newAsignaturas
          })
        }
      });
    });

    const q3 = query(collection(db, "schedule"), where("idAsignatura", "==", docu.id))
    const querySnapshot3 = await getDocs(q3)
  }
}

```

Figura 6.28: Componente HomeAdmin y funciones

Otros detalles importantes que nos ofrece React son las fucniones, useState y useEffect. useState() se utiliza para crear elementos sin tipo, es decir, sin la necesidad de especificar si es un número, string, array... Está compuesto por dos elementos, la variable que se llama y la función para setear (establecer) el valor de la variable.

```
const [teachers, setTeachers] = useState([])
const [term, setTerm] = useState([])
```

Figura 6.29: useState()

Por otro lado, tenemos useEffect() que le indica que tiene que hacer el componente después del renderizado de los elementos, en otras palabras, que ocurrirá cuando se actualice la página.

```
useEffect(() => {
  const getTeachers = async () => {
    const data = await getDocs(teacherCollectionRef)
    setTeachers(data.docs.map(doc => ({...doc.data(), id: doc.id})))
  }
  getTeachers()
}, [])
```

Figura 6.30: useEffect()

Finalmente, otra parte importante de la aplicación es la barra de navegación, donde se pueden encontrar el logo de la aplicación, algunos accesos directos a componentes y el cierre de sesión. Para la implementación de la misma en el renderiazado, se han utilizado los enlaces 'Link', un componente de React que nos ofrece poder desplazarnos también entre vistas.

```

const Navbar = () => {
  const {logout} = useAuth()
  const handleLogout = async () => {
    await logout()
  }

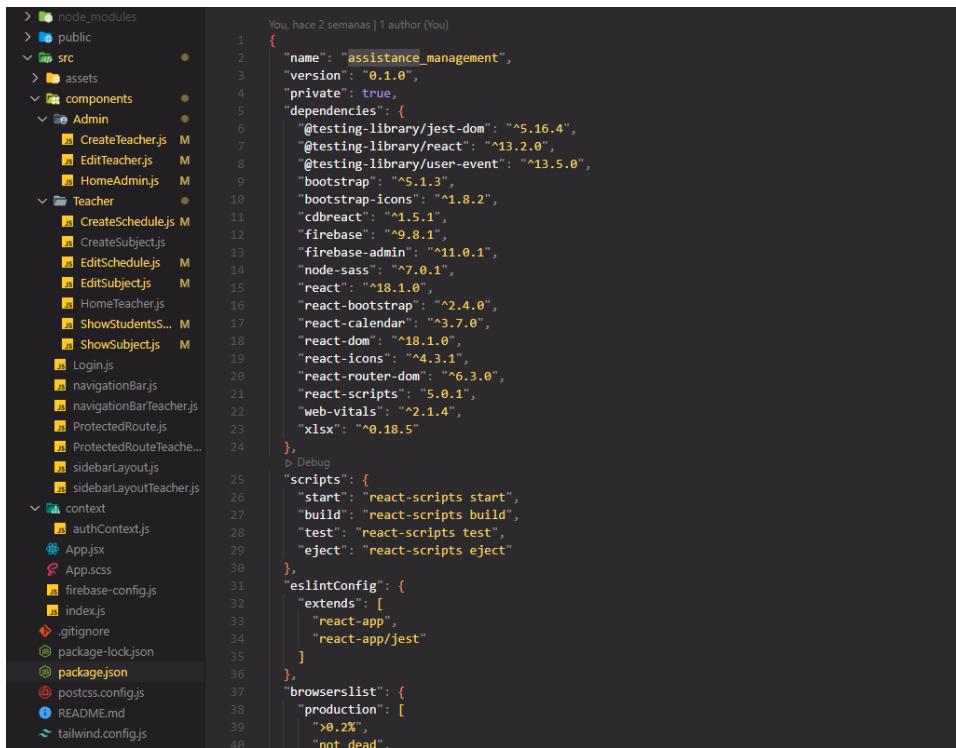
  return (
    <div className='flex'>
      <div className='w-72 bg-blue-700 relative duration-300 drop-shadow-lg h-screen'>
        <div className='flex gap-x-4 items-center mt-0'>
          <img
            src={logo}
            className='w-fit h-18'
          />
        </div>
        <p className='flex mt-3 h-3 p-8 text-gray-50 text-sm items-center gap-x-4 no-underline'>
          Bienvenido Administrador
        </p>
        <Link to="/HomeAdmin" className='flex mt-3 h-3 p-8 cursor-pointer hover:bg-slate-200 text-gray-50 text-sm items-center gap-x-4 no-underline'>
          <Image size='2rem' />
          <span className='origin-left'>
            Inicio
          </span>
        </Link>
        <Link to="/createTeacher" className='flex mt-3 h-3 p-8 cursor-pointer hover:bg-slate-200 text-gray-50 text-sm items-center gap-x-4 no-underline'>
          <Image size='2rem' />
          <span className='origin-left'>
            Alta profesor
          </span>
        </Link>
        <button onClick={handleLogout} className='flex w-full mt-3 h-3 bottom-0 absolute p-8 cursor-pointer hover:bg-slate-200 text-gray-50 text-sm hover:text-blue-700 item'>
          <Image size='2rem' />
          Cerrar Sesión
        </button>
      </div>
    </div>
  )
}

export default Navbar

```

Figura 6.31: NavBar

Dependencias instaladas (package.json) y directorios y ficheros



```
> node_modules
> public
└ src
  └ assets
  └ components
    └ Admin
      └ CreateTeacher.js M
      └ EditTeacher.js M
      └ HomeAdmin.js M
    └ Teacher
      └ CreateSchedule.js M
      └ CreateSubject.js
      └ EditSchedule.js M
      └ EditSubject.js M
      └ HomeTeacher.js
      └ ShowStudents... M
      └ ShowSubject.js M
      └ Login.js
      └ navigationBar.js
      └ navigationBarTeacher.js
      └ ProtectedRoute.js
      └ ProtectedRouteTeache...
      └ sidebarLayout.js
      └ sidebarLayoutTeacher.js
  └ context
    └ authContext.js
  └ App.jsx
  └ App.scss
  └ firebase-config.js
  └ index.js
  └ .gitignore
  └ package-lock.json
  └ package.json
  └ postcss.config.js
  └ README.md
  └ tailwind.config.js

You, hace 2 semanas | 1 author (You)
1  {
2    "name": "assistance_management",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.16.4",
7      "@testing-library/react": "^13.2.0",
8      "@testing-library/user-event": "^13.5.0",
9      "bootstrap": "^5.1.3",
10     "bootstrap-icons": "^1.8.2",
11     "cdbreact": "^1.5.4",
12     "firebase": "^9.8.1",
13     "firebase-admin": "11.0.1",
14     "node-sass": "^7.0.1",
15     "react": "^18.1.0",
16     "react-bootstrap": "^2.4.0",
17     "react-calendar": "^3.7.0",
18     "react-dom": "^18.1.0",
19     "react-icons": "^4.3.1",
20     "react-router-dom": "6.3.0",
21     "react-scripts": "5.0.1",
22     "web-vitals": "2.1.4",
23     "xlsx": "^0.18.5"
24   },
25   "scripts": {
26     "start": "react-scripts start",
27     "build": "react-scripts build",
28     "test": "react-scripts test",
29     "eject": "react-scripts eject"
30   },
31   "eslintConfig": {
32     "extends": [
33       "react-app",
34       "react-app/jest"
35     ]
36   },
37   "browserslist": {
38     "production": [
39       ">0.2%",
40       "not dead"
41     ]
42   }
43 }
```

Figura 6.32: Dependencias, directorios y ficheros.

6.5. Repositorio GitHub

En el siguiente enlace se encuentra el repositorio de GitHub donde se han ido subiendo las versiones de las aplicaciones desarrolladas en este proyecto:

<https://github.com/martingarcia99/TFG>

Capítulo 7

Problemas Encontrados

En este capítulo se hablará de todos aquellos problemas que se han tenido durante el desarrollo del proyecto y las soluciones que se la han dado.

7.1. Idea Principal del Proyecto

La idea principal de este proyecto era el desarrollo de un sistema de asistencia automática utilizando un sistema diferente. En este sistema, el objetivo era usar un dispositivo móvil (del alumno) que emulara un tag NFC con un mensaje y un lector NFC que leyera dicho mensaje. En este caso, el lector NFC era un dispositivo arduino capaz de leer tarjetas NFC mediante su hardware.

En concreto, se iba a utilizar el chip PN532, un chip NFC que podemos conectar a un procesador como Arduino para leer y escribir tarjetas NFC, comunicarse con móviles, o incluso actuar como tag NFC.

La comunicación es muy sencilla, ya que podemos comunicarnos a través de SPI, I2C o HSU (High Speed UART). El PN532 opera 3.3V, pero dispone de conversión de nivel por lo que es posible conectarlo con un procesador de 5V.[23]

Este chip dispone de 6 modos de operación:

- **ISO/IEC 14443A/MIFARE Lector/Grabador.**
- **ISO/IEC 14443A/MIFARE Card MIFARE Classic 1K y MIFARE Classic 4K Card.**
- **ISO/IEC 14443B Lector/Grabador.**
- **FeliCa Lector/Grabador.**

- FeliCa Card emulación.
- ISO/IEC 18092, ECMA 340 Peer-to-Peer

Por lo tanto, se pretendía emular un tag en el dispositivo móvil del alumno usando la librería de react-native-hce, mencionada en el Punto de Implementación, y que este chip, conectado a una placa de arduino, trasmitiera la información a la base de datos.

Al tener todo el sistema ya montado, se empezaron a realizar varias pruebas intentando que el lector NFC (chip PN532) recogiera y mostrara el mensaje del emisor, pero por desgracia no lo recibía y tras varias semanas de investigación se encontró el problema. El caso era que el chip PN532 no era capaz de reconocer el tipo de tag que emulaba el dispositivo android, ya que era de tipo ISO/DEP 7816 y como se puede observar en la lista de los modos de operación anterior, no es compatible.

Por ello, se estuvo planeando realizar un cambio en el sistema eficiente y rápido debido a las semanas que se había perdido en la implementación y pruebas del sistema anterior. Se llegó a la conclusión de que se va a utilizar otro dispositivo móvil capaz de leer mensajes enviados a través del NFC por el dispositivo móvil emisor.

De esta forma, el dispositivo lector de NFC utilizará la misma librería que utiliza el emisor (react-native-tag-manager) para recibir los mensajes enviados y poder registrar la asistencia correctamente.

7.2. Despliegue de la App Móvil Alumno

Este problema se ha comentado en el punto de la Implementación y resumiendo, la causa del error era la manera con la que se iba a desplegar la aplicación, utilizando Expo. Dicha herramienta no es capaz de reconocer algunas librerías de react y en este caso, la librería para usar la tecnología NFC. Por tanto, se tuvo que cambiar la manera de desplegar la app y se utilizó directamente el cliente de react-native (react-native CLI).

Capítulo 8

Pruebas

En este capítulo, se van a realizar una serie de pruebas para comprobar la funcionalidad del proyecto, para eliminar los errores del lenguaje de codificación y mejorar la calidad del producto final. Para ello se van a realizar dos tipos de pruebas, caja blanca y caja negra.

8.1. Pruebas de Caja Negra

Las pruebas de caja negra son un proceso de desarrollo de software que examina si un programa funciona y cumple su objetivo. Las personas que realizan este tipo de pruebas suelen conocer las especificaciones del programa y no su lenguaje de codificación. También es una forma de prueba de alto nivel, lo que significa que implica un análisis más amplio de las características de un programa.[24]

He aquí algunos tipos de procesos de pruebas de caja negra:

- **Pruebas funcionales.** Este proceso examina si un programa implementa las características correctamente. Por ejemplo, un probador puede revisar si se puede iniciar sesión en una red utilizando la identificación correcta.
- **Pruebas de seguridad.** Esta evaluación observa si un programa puede proteger la información confidencial de los hackers internos o externos. A menudo señala las áreas de un programa que pueden necesitar alguna fortificación.
- **Pruebas de regresión.** Después de otros procedimientos que dieron lugar a cambios en la codificación, un probador puede volver a evaluar un programa para ver si la nueva versión funciona. También

pueden identificar nuevas áreas a tratar o comprobar su velocidad de procesamiento.

8.1.1. Pruebas

Primeramente, se van a realizar pruebas para comprobar que se inicia sesión con los datos proporcionados. En la aplicación web, con los datos, correo: admin@gmail.com y contraseña: 123456 se consigue el acceso para el administrador.

Con los datos correo: martinillora@gmail.com y contraseña: 12345678 se consigue acceder al portal del profesor y en la aplicación del alumno, se prueba con el correo: martingarcia99@correo.ugr.es y contraseña: martin1 y se consigue acceder correactamente.

También se ha probado a cerrar sesión en la aplicación web y funciona, por lo tanto las pruebas de inicio y cierre de sesión se realizan satisfactoriamente.

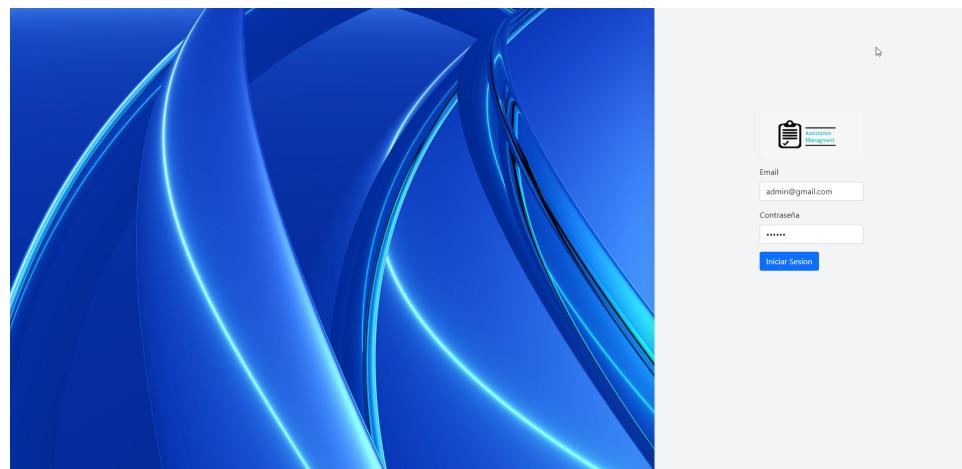


Figura 8.1: Prueba login administrador.

La siguiente prueba es crear asignaturas, asignarles horarios con la fecha y hora actual y asignar un alumno para que le salgan en el dispositivo móvil. A la hora de comprobar la lista de asignatura, se comprueba que falta un scroll vertical para poder navegar entre las distintas asignaturas desde el móvil, por lo que se procede a añadirlo y corregirlo.

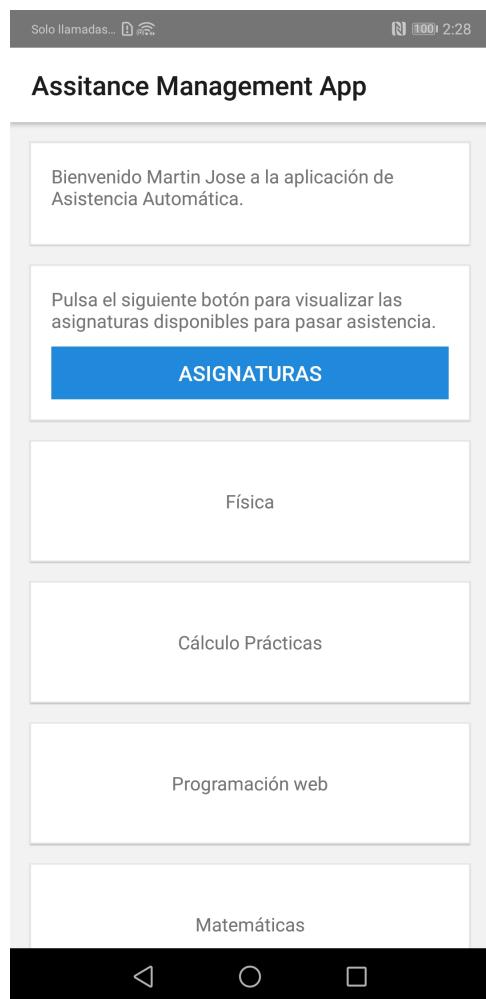


Figura 8.2: Lista de asignaturas sin scroll.

Se realizan pruebas similares en las páginas principales del administrador y el profesor, creando asignatura y profesores hasta que supere el número disponible para toda la pantalla. Se comprueba que el scroll vertical funciona correctamente y que no hay ningun tipo de problema con tener tantos registros.

PÁGINA PRINCIPAL		
Prueba 3 Curso: 2º [checkbox] [Edit] [Eliminar] [Asignar Horario]	Matemáticas Curso: 2º [checkbox] [Edit] [Eliminar] [Asignar Horario]	prueba 10 Curso: 0º [checkbox] [Edit] [Eliminar] [Asignar Horario]
Cálculo Prácticas Curso: 3º B [checkbox] [Edit] [Eliminar] [Asignar Horario]	prueba 4 Curso: 4º [checkbox] [Edit] [Eliminar] [Asignar Horario]	Física Curso: 4º [checkbox] [Edit] [Eliminar] [Asignar Horario]
prueba 5 Curso: 2º [checkbox] [Edit] [Eliminar] [Asignar Horario]	prueba 7 Curso: 0º [checkbox] [Edit] [Eliminar] [Asignar Horario]	prueba 8 Curso: 0º [checkbox] [Edit] [Eliminar] [Asignar Horario]
Prueba 2 Curso: 1º [checkbox] [Edit] [Eliminar] [Asignar Horario]		

Figura 8.3: Alta de varias asignaturas.

Otra prueba ha sido realizar la carga de un documento excel con varios alumnos, y se comprueba que se han cargado todos correctamente en esa asignatura.

Correo	Nombre	Apellidos	
martin3@gmail.com	martins	perez	<button>Eliminar</button>
martinw@gmail.com	martins	perez	<button>Eliminar</button>
martin5@gmail.com	martins	perez	<button>Eliminar</button>
arcadio99@correo.ugr.es	Arcadio	Herrera Abril	<button>Eliminar</button>
martin1@gmail.com	martins	martins	<button>Eliminar</button>
martin9@gmail.com	martins	perez	<button>Eliminar</button>
martingarcia99@correo.ugr.es	Martin Jose	Garcia Muñoz	<button>Eliminar</button>

Figura 8.4: Alta de varias asignaturas.

Se prueba a llenar varias horas todos los días suponiendo que se asisten 4h al día en los horarios, no hay ningún tipo de problema.

	Lunes	Martes	Miercoles	Jueves	Viernes
9:30 - 10:30					
10:30 - 11:30					
11:30 - 12:30					
12:30 - 13:30					
15:30 - 16:30					
16:30 - 17:30					
17:30 - 18:30					
18:30 - 19:30					

Figura 8.5: Calendario lleno.

8.2. Pruebas de Caja Blanca

Las pruebas de caja blanca son un procedimiento de desarrollo de software para verificar el lenguaje de codificación de un programa, es decir, las palabras y los números introducidos que permiten que un programa funcione. Las personas que realizan este tipo de pruebas suelen tener conocimiento del código para poder examinar su estructura interior, su diseño y sus especificaciones técnicas. También es una forma de prueba de bajo nivel, lo que significa que implica la evaluación de las características individuales de un programa para asegurarse de que se ejecutan con éxito.

Estos son algunos tipos de procesos de examen de caja blanca:

- **Pruebas unitarias.** Este proceso evalúa si determinadas líneas de código son funcionales. Las personas suelen realizar pruebas unitarias mientras escriben un lenguaje de codificación para comprobar si hay errores. También pueden realizarlas entre otras pruebas para comprobar si surgen nuevos errores.
- **Análisis estático.** Las personas realizan esta prueba para confirmar si el código fuente de un lenguaje de codificación' es decir, si otras líneas de código pueden utilizar su información para realizar diferentes funciones. Por ejemplo, si un código fuente contiene el tamaño y tipo de letra correctos, puede aparecer en el texto de un programa.

- **Cobertura de la delcaración.** Este proceso implica el seguimiento de las veces que un individuo ejecuta con éxito el código fuente de un programa. Es un valor de fórmula que permite a un desarrollador diseñar los parámetros para otros casos de prueba de caja blanca.

8.2.1. Pruebas

Una vez iniciada sesión en las aplicaciones tanto web como móvil, se van a comprobar el proceso de registro de asistencia. Nos vamos al dispositivo móvil del profesor, lanzamos la aplicación y le damos a activar NFC para leer los datos.

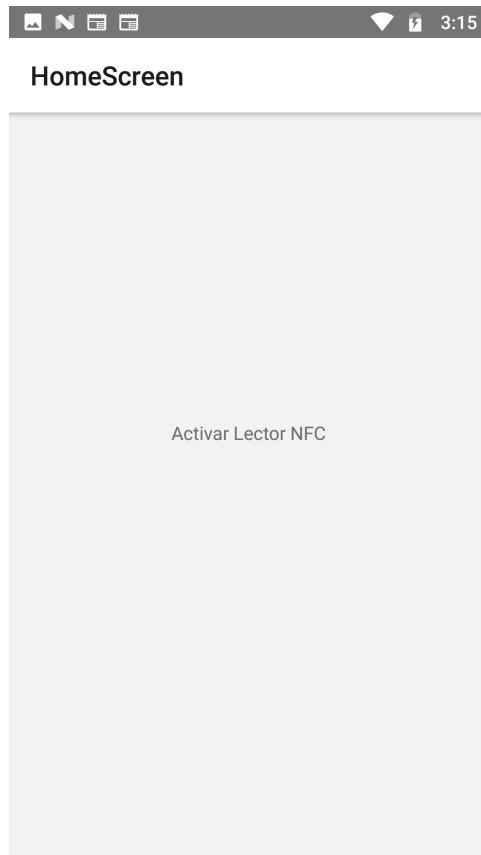


Figura 8.6: App móvil Profesor.

A continuación, escribimos trazas en el código tanto de la app móvil del alumno como en la del profesor. En la app del alumno, antes de enviar el mensaje se comprueba si contiene el id del alumno y el id de la asignatura.

```

const Subject = () => {
  const route = useRoute()

  const registrarAsistencia = () => {
    Alert.alert(
      '',
      'Acerca el dispositivo al lector NFC',
      [
        {text: 'OK', onPress: () => console.log('OK Pressed')},
      ]
    );
    NfcManager.start()
    let content = route.params.subject.id + route.params.idAlumno
    const bytes = Ndef.encodeMessage([ Ndef.textRecord('==' + content) ])
    console.log(content)
    // Stores message to Senders phone. Now other android with enabled NFC can receive it
    NfcManager.setNdefPushMessage(bytes)
  }
}

```

Figura 8.7: App móvil Alumno. Trazas

En la app del profesor, se puede observar la función readNdef() que se encarga de leer el código NFC que le hayan pasado. En este caso, se muestra en la consola el mensaje que recibe para ver si los datos están llenos o falta alguno.

```

//comprobar para ese estudiante, esa asignatura y esa fecha si hay algun registro

const q = query(assistanceCollectionRef, where("asignatura", "==", idAsignatura),where("alumno", "==", idAlumno),where("fecha", "==", date))

const querySnapshot = await getDocs(q)

if(querySnapshot.empty) {
  await addDoc(assistanceCollectionRef,{alumno: idAlumno, nombre: nombre, apellidos: apellidos, hora: horaActual, fecha: fecha})
} else {
  setError("Ya se ha registrado la asistencia para el alumno " + nombre + " " + apellidos + " el dia " + date.toDateString())
}

async function readNdef() {
  You, hace 4 semanas * last changes
  NfcManager.start()
  NfcManager.registerTagEvent()
  // Listens for NFC when u put phones together and tap the screen
  NfcManager.setEventListener(NfcEvents.DiscoverTag, ({ ndefMessage: [{ payload }] }) => {
    const [{ type }] = Ndef.decodeMessage(payload)
    console.log(type) //Message u should receive from sender
    registerAssistance(type.substring(20,41),type.substring(0,20))
  })
}

```

Figura 8.8: App móvil Profesor. Trazas

Como se puede observar, en el log de la consola nos aparecen bien los datos del id del alumno y del id de la asignatura, así que se van a enviar correctamente al móvil del profesor, ya solo quedaría comprobar que llegan correctamente.

```

LOG m6wvWlhCjmgbjdjoIThK
LOG {"curso": "3", "descripcion": "Asignatura en la que podrás aprender a programar tu propia página web con los conceptos básicos del lenguaje HTML y CSS", "id": "05iqqqIBxWALwKW66b8Bn", "nombre": "Programación web", "profesor": "martinil.lora@gmail.com"}
LOG 05iqqqIBxWALwKW66b8Bnm6wvWlhCjmgbjdjoIThK
LOG OK Pressed
LOG 05iqqqIBxWALwKW66b8Bnm6wvWlhCjmgbjdjoIThK
LOG OK Pressed
LOG 05iqqqIBxWALwKW66b8Bnm6wvWlhCjmgbjdjoIThK
LOG OK Pressed

```

Figura 8.9: Output app móvil alumno.

Efectivamente, llegan los datos correctamente a la app del profesor y pasan a ser procesados y listos para registrar la asistencia del alumno.

```

LOG 05iqqqIBxWALwKW66b8Bnm6wvWlhCjmgbjdjoIThK
LOG alumno m6wvWlhCjmgbjdjoIThK
LOG asignatura 05iqqqIBxWALwKW66b8Bn
LOG 04:04:14
LOG Tue Sep 06 2022
LOG 2022-09-06T02:04:14.480Z

```

Figura 8.10: Output app móvil profesor.

Figura 8.11: Registro asistencia de alumno.

Capítulo 9

Conclusiones y Trabajos Futuros

En este último capítulo, destaco las conclusiones que podemos sacar de este trabajo de fin de grado, así como las posibles mejoras futuras.

9.1. Conclusiones sobre el proyecto

Considero que la temática de este proyecto ha sido acertada y como se explicaba en el capítulo Introducción junto con sus objetivos, va a ser una gran ventaja para los profesores a la hora de pasar asistencia en las instituciones y para la introducción de las nuevas tecnologías en el ámbito educacional.

En este proyecto se buscaba conseguir un sistema capaz de ahorrar tiempo al profesor a la hora de pasar lista en las clases, por ello se ha implementado un sistema automático de control de asistencia utilizando la tecnología NFC para realizar un intercambio de datos entre los dispositivos móviles de los alumnos y del profesor y así poder registrar las asistencias de los alumnos con tan solo acercar el móvil. Para la gestión de las asistencias y del sistema, se ha creado una aplicación web en la que el profesor puede consultarlas y gestionarlas.

En este proyecto se han utilizado gran cantidad de tecnologías actuales dando experiencia al estudiante en su utilización en el ámbito laboral y ofreciendo conocimientos de un desarrollador Full-Stack. El framework como React y las herramientas de Firebase han sido clave para este proyecto.

Se ha conseguido lograr cada uno de los objetivos nombrados en la introducción utilizando tecnologías actuales e innovadoras, como el framework de React-Native para la implementación de las aplicaciones móviles, como

el de React, para la implementación de la aplicación web.

La utilización de la tecnología NFC ofrecerá una mayor comodidad tanto al profesor como a los alumnos, ya que con el simple hecho de pasar el dispositivo móvil del alumno por el dispositivo móvil del profesor ya se estaría registrando la asistencia.

Por último, se van a mencionar aquellas asignaturas estudiadas durante la carrera que han servido como base y experiencia para la realización de este proyecto.

- **Fundamentos de Ingeniería del software y Diseño y desarrollo de Sistemas de información** para el desarrollo de los puntos de Análisis y Diseño.
- **Fundamentos de la programación, Metodología de la Programación, Estructuras de datos y Programación y diseño orientado a objetos** para el desarrollo de todo el código.
- **Programación Web y Advanced Web Technologies (ERASMUS)** para el diseño y desarrollo de las aplicaciones.
- **Fundamentos de Bases de Datos y Administración de Bases de Datos** para la programación y conocimiento de SQL.

9.2. Objetivos Alcanzados

En esta sección se va a hablar sobre los objetivos que se han cumplido respecto a los descritos en la introducción.

Respecto a los objetivos del TFG:

- Se ha desarrollado un sistema de comunicación con la tecnología NFC.
- Se ha aprendido a utilizar la herramienta **Firebase Realtime Database** y **Firebase Authentication**.
- Se ha aprendido a usar el framework de **React** y todo lo relacionado con su programación.

- Se ha aprendido a desarrollar aplicaciones web y móviles mediante **React** y **react-native**.

Respecto a los objetivos específicos de la **aplicación móvil (Alumnos)**:

- Se muestra una **interfaz de login** al usuario para poder iniciarse en la aplicación.
- Se comprueban los **datos introducidos** por los usuarios (alumnos).
- Se muestra una **lista de asignaturas** disponibles temporalmente.
- Se muestran los **datos de la asignatura** donde se registrará la asistencia.
- Se permite registrar una asistencia **activando el sistema NFC** para transmitir los datos al dispositivo móvil receptor.

Respecto a los objetivos de la **aplicación móvil (profesor)**:

- Se puede **leer correctamente** los datos recibidos por el dispositivo móvil del alumno con la **tecnología NFC**.
- Permite **registrar la asistencia** correctamente en el sistema.

Finalmente, respecto a los objetivos de la **aplicación web**:

- Se puede interactuar con la aplicación de manera **simple y sencilla** por parte del profesor.
- Se permite dar de **alta una clase / asignatura**.
- Se permite dar de **baja una clase / asignatura**.
- Se permite dar de **alta y baja a profesores**. (Rol administrador).
- Se pueden **registrar alumnos** en una asignatura concreta.
- Se permite **asignar un horario** a una asignatura.
- Se puede **gestionar y visualizar** las asistencias en una fecha concreta.

9.3. Futuras Mejores

En esta sección se proponen las posibles mejoras de las aplicaciones utilizadas en este sistema.

Se añadirán más funcionalidades como exportar las asistencias de los alumnos a un documento excel para que los profesores puedan trabajar mejor con los datos ofrecidos por la aplicación. También se intentará mejorar el detalle de los datos, por ejemplo, ver las asistencias de un alumno determinado con más precisión y mostrando una gráfica sobre su progreso.

Se reforzará la seguridad de la aplicación y del acceso a los datos y por la parte visual, se realizará un desarrollo para poder adaptar la aplicación web a distintos dispositivos con diferentes tamaños.

Bibliografía

- [1] Algunos programas para pasar asistencia automática. Retrieved 7 September 2022, from <https://www.universia.net/es/actualidad/orientacion-academica/3-programas-pasar-lista-manera-automatica-1162404.html>.
- [2] Aplicación Alexia. Retrieved 7 September 2022, from https://play.google.com/store/apps/details?id=alexia_teachers.educaria.es.alexiaprofesores&hl=es&gl=US.
- [3] Aplicación Dinantia. Retrieved 7 September 2022, from <https://play.google.com/store/apps/details?id=com.dinantia.dinantia&hl=es>.
- [4] Información sobre la tecnología infrarojos. Retrieved 7 September 2022, from https://es.wikipedia.org/wiki/Infrared_Data_Association.
- [5] Información sobre la tecnología bluetooth. Retrieved 7 September 2022, from <https://www.ionos.es/digitalguide/servidores/know-how/que-es-bluetooth/>.
- [6] Más información sobre la tecnología bluetooth. Retrieved 7 September 2022, from https://repositorio.uam.es/bitstream/handle/10486/662279/grange_garcia_cristian_eric_tfg.pdf?sequence=1&isAllowed=y.
- [7] Información sobre la tecnología Beacons (bluetooth low energy). Retrieved 7 September 2022, from <https://thevalley.es/blog/que-son-los-beacons-y-cual-es-su-potencial/>.

- [8] Más información sobre los Beacons. Retrieved 7 September 2022, from <https://www.mokoblu.com/es/how-beacon-technology-can-be-applied-in-the-education-industry/>.
- [9] Información sobre la tecnología biometría. Retrieved 7 September 2022, from <https://latam.kaspersky.com/resource-center/definitions/biometrics>.
- [10] Información sobre el Reconocimiento Facial. Retrieved 7 September 2022, from https://es.wikipedia.org/wiki/Sistema_de_reconocimiento_facial.
- [11] Información sobre biometría y el reconocimiento dactilar. Retrieved 7 September 2022, from <https://www.kimaldi.com/biometria/>.
- [12] Información sobre el GPS. Retrieved 7 September 2022, from <https://www.adslzone.net/reportajes/tecnologia/gps-que-es-redes/>.
- [13] Uso real del GPS en alumnos de una institución China. Retrieved 7 September 2022, from <https://www.lanacion.com.ar/tecnologia/en-china-utilizan-uniformes-sensores-gps-controlar-nid2206403/>.
- [14] Información sobre la tecnología de los códigos QR. Retrieved 7 September 2022, from <https://es.godaddy.com/blog/que-es-un-codigo-qr-y-como-funciona/>.
- [15] Información sobre la tecnología RFID. Retrieved 7 September 2022, from <https://www.tecnipesa.com/blog/69-tecnologia-rfid-que-ventajas-tiene>.
- [16] Más información sobre la tecnología RFID. Retrieved 7 September 2022, from <https://www.cursosaula21.com/que-es-el-rfid/>.
- [17] Información sobre la tecnología NFC. Retrieved 7 September 2022, from <https://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve>

- [18] Más información sobre la tecnología NFC. Retrieved 7 September 2022, from <https://blog.masmovil.es/que-es-tecnologia-nfc-movil/>.
- [19] Información sobre el modelo en Cascada. Retrieved 7 September 2022, from <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/>.
- [20] Información sobre React. Retrieved 7 September 2022, from <https://es.wikipedia.org/wiki/React>.
- [21] Información sobre Firebase. Retrieved 7 September 2022, from <https://es.wikipedia.org/wiki/Firebase>.
- [22] Información sobre los productos de Firebase. Retrieved 7 September 2022, from <https://firebase.google.com/products-build>.
- [23] Información sobre el chip pn5322 de arduino. Retrieved 7 September 2022, from <https://www.luisllamas.es/arduino-nfc-pn532/>.
- [24] Información sobre la caja negra y la caja blanca a la hora de realizar pruebas sobre un software. Retrieved 7 September 2022, from <https://historiadelaempresa.com/pruebas-de-caja-negra-y-caja-blanca>.

Capítulo 10

Anexo I. Manual de Usuario

En este capítulo extra de la memoria añado un manual de uso de la aplicación web, para poder usarla desde cero.

10.1. Aplicación Web

Para el uso de la aplicación web tanto para el profesor como para el administrador, primeramente tienen que arrancarlo en local con el comando:

```
npm start
```

donde se le lleva a localhost:?, siendo ? el puerto que tengan configurado por defecto. Dependiendo de los datos que se rellenen en el formulario, se accederá a la parte del administrador o a la del profesor, por ello, se ha configurado un perfil único para el administrador.

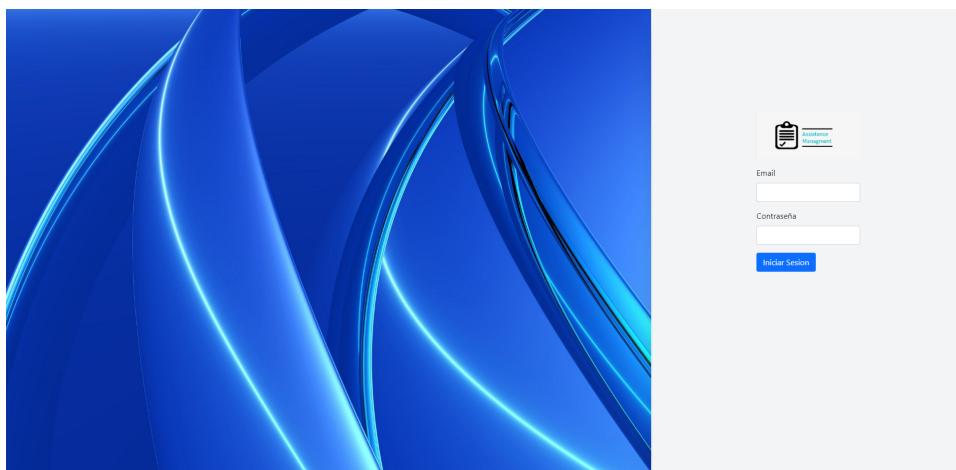


Figura 10.1: Login App Web.

App Web Administrador

En la página principal del administrador se listarán todos los profesores creados por él mismo. Para su creación, se encuentra un botón “Alta profesor” que nos redirige a una vista con un formulario para llenar los datos del profesor.

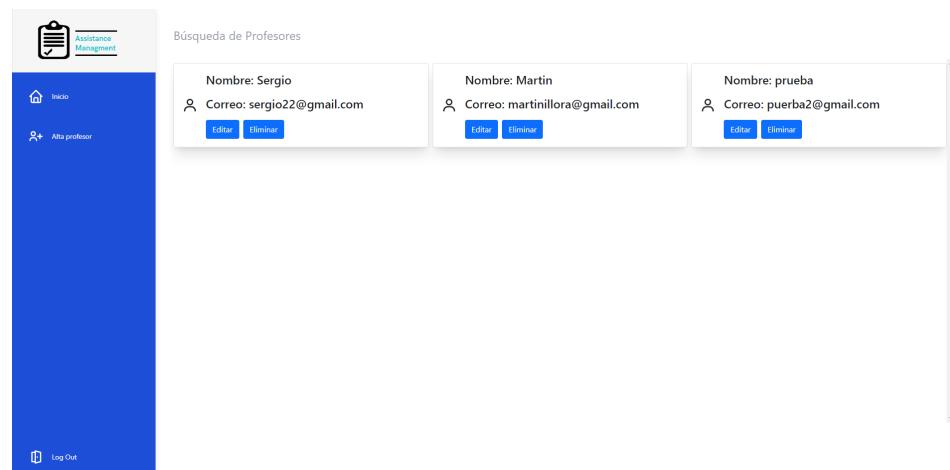


Figura 10.2: Página Principal App Web Administrador.

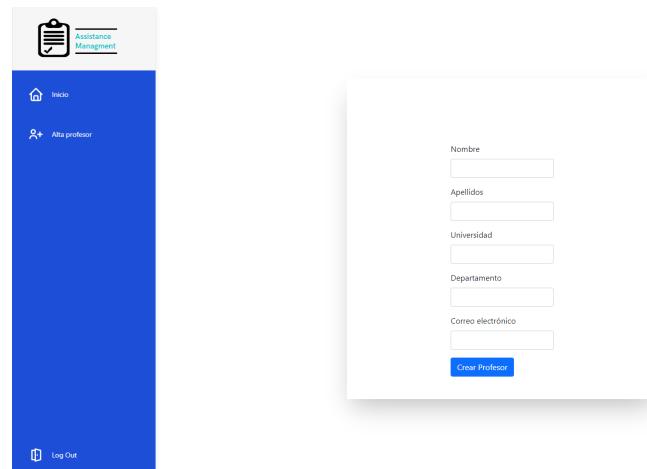


Figura 10.3: Alta profesor App Web.

Como se puede observar en la figura 10.2 se puede ver junto a la carta de cada profesor las opciones para editarla y eliminarla.

App Web Profesor

En esta parte se utiliza la misma barra de navegación implementada en la parte anterior pero con distintas funcionalidades. En este caso vamos a tener la opción .“alta asignatura” que llevará a una nueva vista donde llenar un formulario para crear la asignatura.

Las asignaturas se van a listar en la página principal y cada una de ellas tienen las opciones de editar, eliminar y asignar horario. Las funciones editar y eliminar son similares a las implementadas en la parte anterior. La nueva opción .“asignar horario” lleva a una nueva vista donde se llenará un formulario con las horas y días en las que se imparte la asignatura y a continuación, se asignará dicho horario a la asignatura seleccionada.

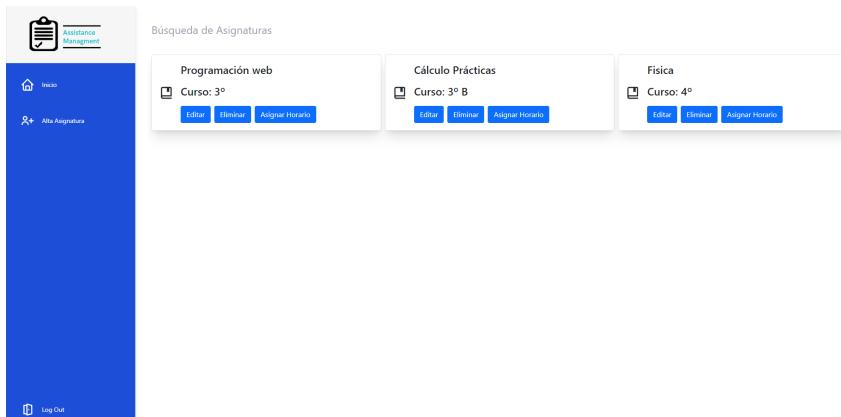


Figura 10.4: Página Principal App Web Profesor.

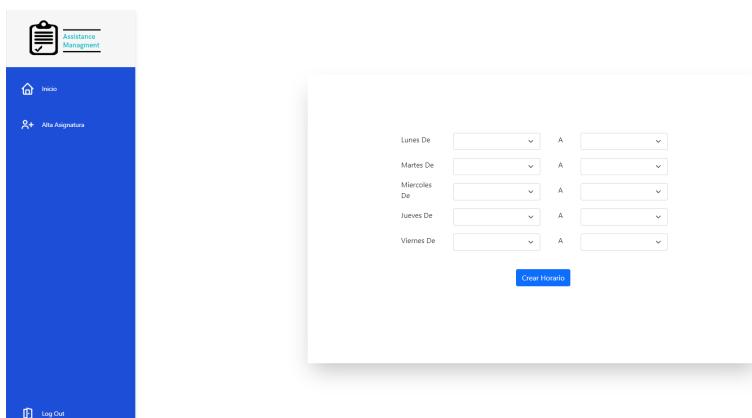


Figura 10.5: Asignar Horario App Web.

Al pinchar sobre el título de cada asignatura, se redirige a una nueva vista donde se encuentra toda la información de la asignatura. Se encuentra el título de la asignatura junto con la descripción y a continuación, el horario, un calendario para seleccionar un día concreto y que muestre la lista de asistencias en ese día y un panel de configuración para editar el horario, mostrar la lista de alumnos y realizar una subida de la lista de alumnos.

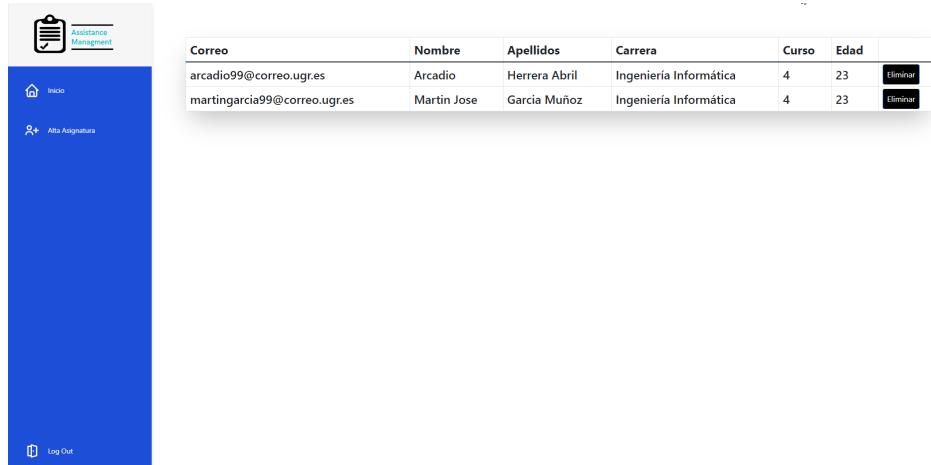
The screenshot shows the main interface for managing assignments. On the left, there's a sidebar with icons for Home, Add Assignment, and Log Out. The main area has a title 'Assistance Management' with a clipboard icon. Below it is a weekly schedule grid from Monday to Friday, 9:30 to 19:30. A specific slot on Wednesday from 12:30 to 13:30 is highlighted in green. To the right of the schedule is a calendar for June 2022, showing dates from 30 to 3. Below the calendar is a table for a student named Arcadio Herrera Abril, listing Name, Surname, Time (01:22:22), Date (Tue Jun 28 2022), and Presence (checkbox). On the far right, there's a sidebar with buttons for Edit Horario, Lista de alumnos, Cargar Alumnos, and Registrar Asistencia.

Figura 10.6: Página principal de una asignatura.

Esta última opción consiste en que el profesor subirá un fichero excel con la lista de alumnos que va a dar de alta en el sistema si no existen y dar de alta en la propia asignatura.

This screenshot shows the 'Carga de Alumnos' (Import Students) dialog box. It contains a file input field labeled 'Seleccionar archivo' (Select file) with the placeholder 'Ninguno archivo selec.' (No file selected). At the bottom right of the dialog is a 'Close' button. The background shows the same assignment management interface as Figure 10.6, with the weekly schedule, calendar, and sidebar visible.

Figura 10.7: Sistema para cargar fichero excel.



The screenshot shows a web-based application titled "Asignatura Management". The left sidebar is blue and contains icons for "Inicio" (Home), "Alta Asignatura" (Add Subject), and "Log Out". The main content area has a white header with the title "Asignatura Management". Below the header is a table with the following data:

Correo	Nombre	Apellidos	Carrera	Curso	Edad	
arcadio99@correo.ugr.es	Arcadio	Herrera Abril	Ingeniería Informática	4	23	<button>Eliminar</button>
martingarcia99@correo.ugr.es	Martin Jose	Garcia Muñoz	Ingeniería Informática	4	23	<button>Eliminar</button>

Figura 10.8: Lista de alumnos de una Asignatura.

