UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Digitalització de la gestió del servei de menjador escolar

Ivan Sierra Moreno

Director: Santi Seguí Mesquida
Realitzat a: Departament de
Matemàtiques i Informàtica

Barcelona, 20 de juliol de 2024

Índice

Resum

En aquest projecte, aprofitant el coneixement adquirit en el grau d'Enginyeria Informàtica, he volgut resoldre un problema trobat al meu lloc de treball com a coordinador de menjador escolar, aplicant tot aquest saber per a desenvolupar una aplicació de programari que ajudés a gestionar el servei i tota la informació que l'envolta.

L'aplicació es desenvoluparà seguint la metodologia Waterfall, complint amb els requisits plantejats. S'utilitzarà el llenguatge Kotlin i l'entorn de desenvolupament Android Studio.

L'objectiu principal del projecte és centralitzar tota la informació i gestió del menjador en una única aplicació mòbil, agilitzant els processos i millorant el control i la seguretat del servei. En un futur, espero poder afegir noves funcionalitats que centralitzin la gestió d'altres àrees del col·legi, optimitzant els processos i beneficiant tant a treballadors com a usuaris del centre.

Resumen

En este proyecto, aprovechando el conocimiento adquirido en el grado de Ingeniería Informática, he querido resolver un problema encontrado en mi lugar de trabajo como coordinador de comedor escolar, aplicando todo este saber para desarrollar una aplicación de software que ayudara a gestionar el servicio y toda la información que le rodea.

La aplicación se desarrollará siguiendo la metodología Waterfall, cumpliendo con los requisitos planteados. Se utilizará el lenguaje Kotlin y el entorno de desarrollo Android Studio.

El objetivo principal del proyecto es centralizar toda la información y gestión del comedor en una única aplicación móvil, agilizando los procesos y mejorando el control y la seguridad del servicio. En un futuro, espero poder añadir nuevas funcionalidades que centralicen la gestión de otras áreas del colegio, optimizando los procesos y beneficiando tanto a trabajadores como a usuarios del centro.

Summary

In this project, taking advantage of the knowledge acquired in the Computer Engineering degree, I wanted to solve a problem found in my workplace as a school cafeteria coordinator, applying all this knowledge to develop a software application that would help manage the service and all the information that surrounds it.

The application will be developed following the Waterfall methodology, complying with the requirements raised. The Kotlin language and the Android Studio development environment will be used.

The main objective of the project is to centralize all the information and management of the cafeteria in a single mobile application, streamlining the processes and improving the control and security of the service. In the future, I hope to be able to add new functionalities that centralize the management of other areas of the school, optimizing the processes and benefiting both workers and users of the center.

1. Introducción y antecedentes

Este proyecto surge a partir de un problema que encontré en mi trabajo como coordinador de comedor escolar, el cual requería de una necesaria optimización del proceso de gestión de este y de control de la información implicada.

He sido testigo de cómo la gestión de los comedores escolares se realiza, en gran medida, de forma manual, utilizando métodos tradicionales como papel y bolígrafo. Esto implica:

- **Control de asistencia y recuento de alumnos manual**: Se anota la asistencia de los alumnos en papel y luego se transcribe al ordenador, teniendo que contar los alumnos a mano de uno a uno listado a listado para saber el número de asistencia diaria, entre otras cosas.
- **Gestión de menús y alérgenos**: Los menús no detallan alérgenos, obligando a los trabajadores a consultar con cocina constantemente. Esto genera ineficiencia y riesgos de errores a la hora de servir la comida a los niños. La información sobre las alergias de los alumnos no está centralizada ni es accesible, especialmente para nuevos empleados. Esto dificulta la atención adecuada a sus necesidades y aumenta el riesgo de errores involuntarios.
- **Comunicación con otros miembros de la comunidad educativa**: La comunicación con el profesorado es verbal, lo que genera en muchas ocasiones que no llegue toda la información al responsable del alumno en caso de incidencias si no se ha coincidido presencialmente con este en el momento de finalizar el servicio. Toda la comunicación con familiares ha de pasar por el profesorado, cosa que genera en ocasiones que no llegue la información al destinatario (En caso de solicitar dieta para el alumno, por ejemplo, y no llegue a cocina).
- **Control de pagos**: En caso de contratar el servicio de comedor puntualmente, esto se hace manualmente, lo que puede conllevar errores en el cobro o impagos.

La gestión manual de los comedores escolares presenta diversos inconvenientes:

- **Falta de eficiencia**: Se requiere una gran cantidad de tiempo y esfuerzo para realizar las tareas administrativas.
- **Errores humanos**: La transcripción manual de datos puede generar errores.
- **Falta de información en tiempo real**: No se dispone de información actualizada sobre la asistencia, alergias e intolerancias, dietas, pagos, etc.
- **Dificultad en la comunicación**: La comunicación entre implicados de la comunidad educativa puede ser lenta y poco efectiva.

Las limitaciones de la gestión manual hacen evidente la necesidad de una solución tecnológica que permita optimizar los procesos del comedor escolar.

La aplicación móvil que propongo en este proyecto tiene como objetivo mejorar la gestión de los comedores escolares mediante la automatización de tareas, la centralización de la información y la mejora de la comunicación entre los miembros de la comunidad educativa.

Los beneficios de la aplicación móvil incluyen:

- **Eficiencia**: Se reduce el tiempo y el esfuerzo necesario para la gestión del comedor.
- **Precisión**: Se eliminan los errores humanos asociados a la transcripción manual de datos.
- **Información en tiempo real**: Se dispone de información actualizada sobre la asistencia, alergias e intolerancias, dietas, pagos, etc.
- **Comunicación efectiva**: Se facilita la comunicación entre implicados de la comunidad educativa de forma rápida y directa.

La digitalización de los procesos en los centros educativos puede ser algo complicada en un principio, sobre todo en centros concertados y privados, ya que implica un cambio en la organización y en la forma en la que se hacen las cosas desde hace mucho tiempo. El proceso de adaptación y aprendizaje de nuevos métodos de funcionamiento en los que la tecnología está implicada suele ser complejo y tedioso en el sector educativo, sobre todo en aquellos centros en los que los miembros de la comunidad educativa abarcan edades muy diversas. Aquellos que llevan muchos años aplicando un sistema concreto o que están más cerca de jubilarse pueden ser muy reticentes en el momento de digitalizar cualquier sistema. No obstante, hay que evidenciar las ventajas que ello implica. En mi experiencia personal, en el centro donde estudiaba, no hace mucho se ha empezado a realizar la gestión y administración de las aulas de forma digital, agilizando mucho el proceso de control de asistencia y comunicación con las familias, por ejemplo. Esto ha mejorado el control de las ausencias no justificadas, que al tratarse de menores es algo muy importante sobre lo que ha de tener control un centro escolar.

1.1. Puntos clave

Problema Identificado

La gestión de los comedores escolares todavía depende mucho de procesos manuales como el papel y bolígrafo para tomar asistencia, gestionar menús y manejar pagos. Esto consume mucho tiempo y es propenso a errores.

Solución Propuesta

Desarrollar una aplicación móvil que simplifique estas tareas, automatizando la gestión y centralizando la información en una única plataforma. Esto incluye mejorar la manera en que se manejan los datos de los estudiantes, como sus alergias, y optimizar la comunicación entre el personal.

Beneficios Esperados

- **Mayor eficiencia**: Reducir el tiempo necesario para la gestión diaria del comedor.
- **Reducción de errores**: Limitar los errores de transcripción y otros errores humanos comunes en los sistemas manuales.
- **Comunicación más efectiva**: Facilitar un canal directo de comunicación entre el personal del comedor, los profesores y las familias.

Aunque la transición a sistemas digitales puede ser un reto, especialmente en centros con personal acostumbrado a métodos tradicionales, las ventajas a largo plazo en eficiencia y manejo de información justifican el esfuerzo de adaptación. La experiencia en otros ámbitos educativos muestra que la digitalización puede significar un gran paso adelante en la administración más efectiva y segura.

2. Objetivos y Motivación

Como ya mencioné en la introducción, este TFG surge como idea a partir de una problemática con la que me encontré en uno de los trabajos que tuve mientras cursaba el grado de Ingeniería Informática. Como Coordinador de comedor escolar, toda la gestión del mismo, la asistencia de los niños, alergias, dietas, incidentes, extraescolares de mediodía, etc. tenía que hacerla a mano, con papel y bolígrafo. ¿2022 y no se utilizaba ningún sistema informático para hacer esta tarea? Investigué un poco y encontré escuelas que usan alguna aplicación para gestionar el día a día del centro, pero para el comedor se sigue haciendo uso del sistema tradicional. También vale la pena decir que estas aplicaciones que utilizan para evaluar al alumnado o comunicarse con las familias, por ejemplo, tienen un margen de mejora considerable. Sobre todo, teniendo en cuenta que hay suficientes profesores que carecen de la habilidad para usarlas de forma fluida y sin necesidad de pedir ayuda a algún compañero, por lo que deberían simplificarse y hacerse más accesibles. Poder dar solución a todo esto aplicando todo lo aprendido durante el grado ha sido mi motivación principal para llevar a cabo este proyecto.

Teniendo en cuenta lo mencionado anteriormente, surge un objetivo principal ya mencionado:

- Centralizar toda la gestión e información del servicio de comedor escolar en una aplicación.

A partir de este objetivo principal, podemos derivar otros secundarios, pero igualmente importantes y que tendré que tener presentes en el desarrollo de la aplicación:

- Digitalizar el proceso de control de asistencia y alergias de los alumnos de los centros escolares.
- Mejorar la seguridad en la gestión de información de los usuarios, especialmente los alumnos, asegurando la privacidad y cumplimiento de normativas de protección de datos, y atender adecuadamente situaciones particulares como alergias o intolerancias.
- Mejorar la comunicación, implementando un sistema de comunicación directo y eficiente entre familias, personal del comedor y profesorado.
- Crear una base de datos única con información actualizada sobre alumnos, menús, alergias, pagos, etc.
- Hacer accesible la aplicación a todos los trabajadores del centro, independientemente de su edad y habilidad con el uso del móvil.

3. Metodología y Desarrollo del proyecto

Una vez definidos los objetivos, empezaré con el proceso de desarrollo de la aplicación, que es en lo que se basa el objetivo principal. Para ello, después de reflexionar sobre qué metodología sería mejor aplicar entre *Agile* y *Waterfall* me decidí por esta última, debido a sus características que se adaptan mejor a proyectos con requisitos bien definidos y estables, como es el caso del desarrollo de esta aplicación. A continuación, podemos ver una tabla comparativa de ambas metodologías para ayudar a entender mejor en qué consisten cada una y sobre las ventajas de estas, para explicar después el porqué de mi elección con más detalle:

Características	Waterfall	Agile
Enfoque	Secuencial y lineal.	Iterativo e incremental.
Flexibilidad	Menos flexible, cambios difíciles de manejar.	Altamente adaptable.
Entregables	Producto completo al final.	Incrementos funcionales.
Planificación	Completa al inicio.	Adaptativa a lo largo del proyecto.
Feedback	Limitado hasta pruebas finales.	Feedback continuo a lo largo del proceso.
Riesgos	Cambios tardíos pueden ser costosos.	Reducidos por iteraciones cortas.
Clientes	Interacción limitada.	Colaboración continua.
Adaptabilidad	Baja.	Alta.
Control de Calidad	Al final de ciclo.	Continuo

Como podemos ver en la tabla, parece que la metodología *Agile* tiene una serie de ventajas muy interesantes en comparación a la *Waterfall*, ya que permite una mayor flexibilidad y capacidad de adaptación a cambios y condiciones del proyecto, y una implicación mayor del cliente. No obstante, este proyecto surge de una idea propia, a partir de mi experiencia laboral en el sector, y tengo una idea bastante clara de cómo ha de ser la aplicación y el camino que he de seguir. Aunque consultaré ideas y pediré opinión al que sería mi posible cliente, teniéndolo en cuenta en cierto sentido en el desarrollo del proyecto, no es un encargo, sino una idea a la que he de darle valor y, una vez hecha la aplicación, ofrecer el servicio a escuelas que valoren la utilidad de este. Además, al realizar todo el proyecto yo solo, encuentro que las características de la metodología *Waterfall* se adecuan más a mi situación actual, y me facilitará en mayor medida poder desarrollarlo con éxito. En definitiva, debido a la estabilidad de los requisitos y la estructura del proyecto (bien definidos desde un

inicio y sin proyección a aceptar modificaciones), la flexibilidad que nos da la metodología *Agile* es menos necesaria.

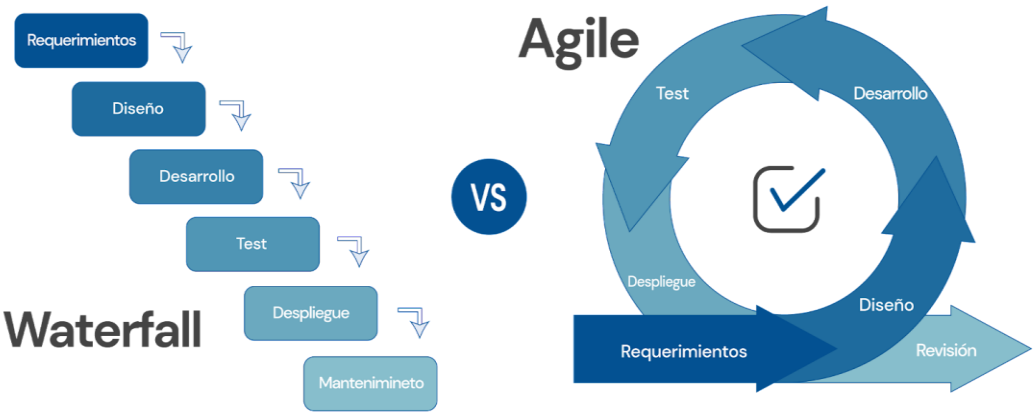


Figura 1. Concepto secuencial vs iterativo

La metodología *Waterfall* es una metodología secuencial y lineal, en la que se sigue una secuencia fija de fases, las cuales deben completarse para poder pasar a la siguiente. Los requisitos del proyecto se establecen al inicio de este, y se espera que sean consistentes y no se desvíen drásticamente mientras se desarrolla. Todo esto hace que haya poco margen para modificaciones, siendo una metodología inflexible a cambios. Por ello, seguir cada etapa paso a paso es crucial para poder desarrollar el proyecto con éxito. Las fases a seguir son las siguientes:

- **Requisitos**: Esta es la fase inicial donde se recopilan y documentan detalladamente los requisitos del proyecto. Se identifican las necesidades y expectativas de los interesados y se establecen los objetivos del proyecto.
- **Diseño**: En esta etapa se desarrollan los diseños técnicos y funcionales adecuados para guiar la correcta implementación.
- **Desarrollo**: En esta fase se codifica y se desarrolla el producto siguiendo las especificaciones establecidas en la etapa de diseño.
- **Pruebas**: Una vez completada la implementación, se lleva a cabo un proceso de pruebas exhaustivo para garantizar que el funcionamiento sea correcto según lo diseñado y cumpla con los criterios predefinidos. Además, se abordan y corrigen todas las anomalías identificadas durante las pruebas.
- **Despliegue**: En esta fase, el producto se prepara y se pone en funcionamiento en el entorno de producción o en manos de los usuarios finales.
- **Mantenimiento**: Una vez finalizado y publicado el producto, se brinda un soporte continuo y se aplican las actualizaciones/correcciones relevantes para garantizar que

el sistema o producto funcionará de manera óptima durante su vida útil (DoneTonic, 2022).

Por tiempo y logística, para este TFG completaré las fases de requisitos, diseño y desarrollo.

Antes de comenzar con la fase de requisitos, realizaré un análisis de mercado para asegurarme de que el producto final sea competitivo y cumpla con las necesidades actuales y futuras de los usuarios potenciales.

3.1. Análisis del mercado

Aunque los orígenes de la alimentación en el entorno de la educación escolar se remontan tiempo atrás, los comedores escolares, tal y como los conocemos ahora, nacen en los años 50 con la ley de Educación Primaria y el Servicio Escolar de Alimentación y Nutrición. Es fácil de entender que, desde entonces hasta la actualidad, la mayor parte del tiempo se haya estado gestionando este servicio de forma manual. Principalmente porque el desarrollo tecnológico necesario para poder gestionarlo de forma digital como para optimizar el proceso no se ha dado hasta, podríamos decir, 10/15 años atrás. Todo este tiempo de gestión manuscrita de los comedores escolares, hacen que el proceso a papel y boli se haya perfeccionado para minimizar tiempo y costes todo lo posible, pero la digitalización de este proceso puede optimizarlo aún más. El hecho de que, en mi entorno más cercano, por lo que he podido investigar, se siga utilizando el método tradicional para esta gestión me hace pensar cuál podría ser el motivo de que no se haya optado por una alternativa digital para ello.

Investigando un poco, he podido encontrar alguna aplicación que realiza estas funciones de gestión y que podrían ser utilizadas por los colegios:

- **Iara Comedores (LOAD Servicios Informáticos, 2024)**: es una plataforma dedicada a la gestión de comedores escolares, elaborada por la empresa LOAD, con sede en Málaga. Esta plataforma se estructura en torno a tres áreas principales: Gestión Comercial, Plataforma de Comunicación y Gestión Online. Funcionalidades clave:
    - Gestión de Alumnos y Datos Personales
    - Facturación y Control de Pagos
    - Control de Mensajería e Incidencias
    - Gestión Dietética y Nutricional
    - Plataforma de Comunicación Web

- Estadísticas y Auditoria

Estas son solo algunas de las funciones que ofrece la plataforma, por lo que en sí es una opción muy completa que da solución a la digitalización del proceso de gestión de los comedores escolares.

- **APP comedores escolares (Roca González)**: es una aplicación desarrollada por la empresa Roca González que se centra en facilitar la interacción entre las familias y los centros educativos respecto a la gestión del comedor escolar. Funcionalidades clave:
  - Menús mensuales con información nutricional y recomendaciones de cenas.
  - Comunicación de la asistencia y faltas diarias.
  - Gestión de los recibos y posibilidad de pago virtual.
  - Comunicación directa con la coordinación del comedor.
  - Estadísticas e informes sobre el uso del comedor.

  Aunque está diseñada principalmente para las familias, esta aplicación también puede ser útil para los centros educativos, aunque presenta margen de mejora en cuanto a la integración y funcionalidad total.

- **Aplicación comedores escolares (Casa Intur, 2021)**: es una aplicación de gestión de comedores escolares ofrecida por Intur Restauración Colectiva, accesible tanto para las familias como para los centros educativos. Funcionalidades clave:
  - Notificaciones personalizadas para padres.
  - Comunicación bidireccional familia-centro.
  - Consultas de menús con fotos.
  - Gestión eficiente de menús con control de asistencia de los alumnos.
  - Generación de informes en Excel.

  Así pues, es una aplicación que se parece por características más a la ofrecida por la empresa Roca González, aunque ofrece alguna funcionalidad distinta y la presentación de su página web hace que parezca algo más elaborada.

- **Colechef (ColeChef, 2024)**: es una plataforma especializada en la gestión de comedores escolares que integra funcionalidades tanto para centros educativos como para las familias. Su enfoque está en mejorar la experiencia de alimentación escolar a través de tecnología y servicios de soporte. Funcionalidades clave:

- Gestión de menús con información nutricional y adaptaciones a necesidades especiales.
- Comunicación directa y eficiente entre la escuela y las familias.
- Control de accesos y asistencia para una mejor seguridad y organización.
- Reservas y cancelaciones de menús en tiempo real.

Incluye un componente de nutrición diseñado por expertos, lo que puede ser un diferencial importante para promover hábitos saludables en el entorno escolar.

- **Easilys f&b de Mapal OS (MapalOS, s.f.)**: es una solución de gestión que se centra en la eficiencia operativa de los servicios de alimentación, incluidos los comedores escolares. Ofrece un conjunto integrado de herramientas para simplificar la gestión diaria y reducir costes operativos. Funcionalidades clave:
  - Gestión de compras y proveedores para optimizar costes y garantizar la calidad de los alimentos.
  - Control de inventarios y trazabilidad de alimentos para cumplir con las regulaciones de seguridad alimentaria.
  - Análisis de consumo y costes para mejorar la toma de decisiones y la sostenibilidad.

  Su enfoque en la gestión de costes y eficiencia puede ofrecer a las escuelas una manera de mejorar sus servicios de comedor mientras se controlan los gastos.

- **No Problem Cooking (NoProblemCooking, s.f.)**: es una solución integral diseñada para optimizar la gestión culinaria en comedores escolares, mejorando la interacción entre el personal del comedor y las familias. Funcionalidades clave:
  - Planificación y Diseño de Menús: Permite a los comedores escolares planificar menús basados en criterios nutricionales y preferencias alimentarias, garantizando una dieta equilibrada y adecuada para todos los estudiantes.
  - Gestión de Alergias e Intolerancias: Proporciona herramientas robustas para registrar y gestionar información sobre alergias e intolerancias alimentarias, asegurando que los menús se adapten a las necesidades de seguridad alimentaria de cada alumno.
  - Comunicaciones y Notificaciones Automáticas a Padres: Facilita la comunicación entre el comedor y las familias, enviando actualizaciones automáticas sobre los menús, cambios y eventos especiales relacionados con el comedor.

Su enfoque en la nutrición y seguridad alimentaria la hace una opción interesante para escuelas que priorizan la salud y bienestar de sus estudiantes.

Estas son algunas de las aplicaciones más completas y con una idea muy similar a la que tengo en este proyecto que dan solución a la gestión de comedores escolares. No obstante, no hay una gran variedad de opciones y, sin poder juzgar la calidad de estas porque se ofrecen con un servicio de pago mensual y no las he podido probar, creo que la falta de visibilidad y de conocimiento hace que muchas escuelas no lo utilicen todavía. Es por esto que queda mucho mercado por cubrir a pesar de la competencia, ya solo teniendo en cuenta la cantidad de colegios que hay en Barcelona[1].

Las aplicaciones existentes muestran que hay una demanda clara para la digitalización de la gestión de comedores escolares, pero también indican que muchas soluciones no están completamente adaptadas a las necesidades de todos los usuarios potenciales, especialmente en lo que respecta a la flexibilidad y facilidad de uso. Mi propuesta busca llenar este vacío, ofreciendo una solución más integrada y fácil de usar, especialmente diseñada para adaptarse tanto a los trabajadores del comedor como a las necesidades administrativas de los centros educativos.

Quiero remarcar el potencial de una aplicación como esta, con muchas funcionalidades como las que ofrece ya la competencia y otras que pueden ser diferenciales con esta. Aunque me gustaría poder desarrollar todas las funcionalidades que creo que suman valor a la aplicación, por una cuestión de logística y tiempo me centraré en aquellas que considero más importantes: poder llevar un registro de asistencia diaria de los alumnos del centro escolar y tener un control exhaustivo de las alergias e intolerancias de estos.

No obstante, mi idea es continuar desarrollándola en un futuro, de manera que aborde no solo las funcionalidades básicas ya presentes en el mercado, sino que también introduzca mejoras significativas con otras funcionalidades como:

- **Gestión de pagos a través de los perfiles de Alumnos**: Habilitar pagos de cuotas y servicios esporádicos a través de la aplicación.
- **Menús Personalizados**: Adaptación de la oferta de alimentos según necesidades dietéticas específicas.
- **Mejora de la Comunicación**: Integración de un sistema de comunicación efectivo que conecte comedor, tutores y familias.
- **Actividades del Comedor**: Presentación de actividades relacionadas con el espacio del comedor escolar, enriqueciendo la experiencia educativa.

[1] En Barcelona hay, según el Consorcio de Educación de Barcelona en el curso 2021-2022, aproximadamente 461 colegios de primaria y secundaria, de los cuales unos 220 son concertados y privados, por lo que no sería necesario entrar en el sector público directamente (Consorci d'Educació de Barcelona, 2022).

Una vez analizado el mercado, para identificar áreas clave para el desarrollo e implementación del proyecto, se hace necesario un análisis DAFO. Este análisis me ayuda a entender claramente no solo las fortalezas en las que puedo apoyarme y las debilidades que debo mejorar, sino también a identificar las oportunidades que debo aprovechar y las amenazas que necesito considerar. Así puedo planear mejor cómo llevar adelante la aplicación y asegurarme de que tenga un buen recibimiento en las escuelas.

Figura 2. Tabla del análisis DAFO

3.2. Requisitos

Como ya he mencionado antes, en esta fase se identifican las necesidades y expectativas de los interesados para poder definir los requisitos del proyecto. Para ello haré un formulario de Google, el cual pasaré a los empleados del centro donde he ejercido de monitor y coordinador de comedor para que lo completen, formulando una serie de preguntas que me ayudarán a determinar los requisitos para la aplicación. Este cuestionario lo responden tanto monitores de comedor, coordinadores, empleados de cocina y personal administrador del centro. Podréis encontrar más detalles del cuestionario en el apartado de anexos. A partir de

la información extraída y de mi experiencia en el sector, haré uso de las historias de usuario para reflejar, desde la perspectiva del usuario final, las funcionalidades o acciones que se espera que tenga la aplicación.

Una de las preguntas planteadas en el formulario consistía en seleccionar funcionalidades que se consideren importantes para la aplicación. En la imagen 2 podéis ver la gráfica que muestra las selecciones que han hecho los usuarios.

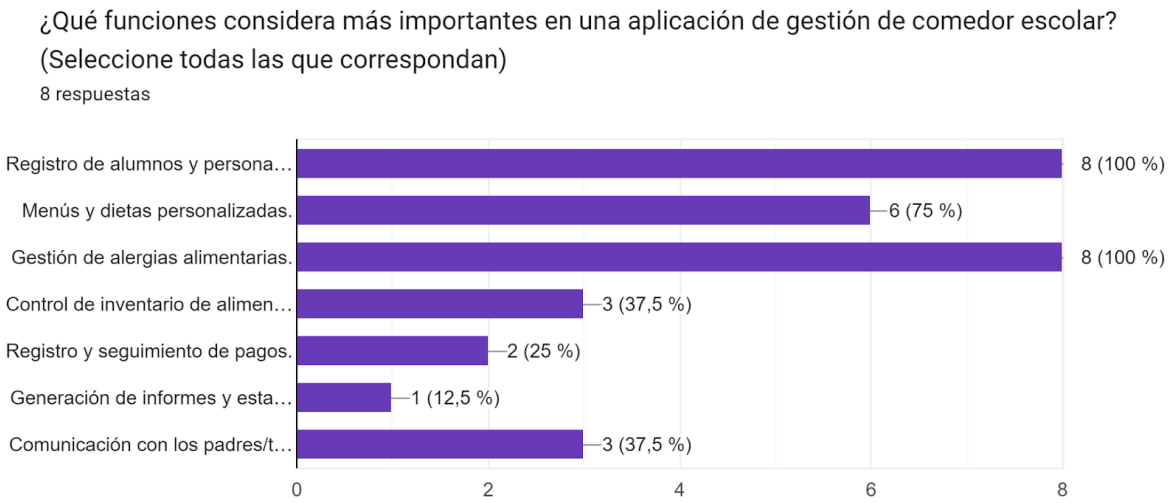


Figura 3. Gráfica que muestra las funcionalidades seleccionadas por los trabajadores como las más relevantes.

Como podéis comprobar en la gráfica, las funcionalidades a las cuales más importancia han dado son las siguientes:

- **Registro de alumnos y personal de comedor**: fundamental para llevar un control adecuado y eficiente de los asistentes y empleados.
- **Gestión de alergias alimentarias**: vital para garantizar la seguridad y adecuación nutricional de los menús ofrecidos.
- **Menús y dietas personalizadas**: esencial para adaptar la oferta alimentaria a las necesidades específicas de cada alumno.

Coincido en que estas serían las funcionalidades más importantes por delante de las otras. Desarrollarlas, por otra parte, me ayudarán a cumplir uno de los objetivos propuestos, digitalizar el proceso de control de asistencia y alergias de los alumnos en los centros escolares. Además de estas funcionalidades, uno de los encuestados sugiere poder gestionar las bajas de los trabajadores a través de la aplicación, de manera que facilite la organización de la substitución por parte de la empresa, la cual encuentro una funcionalidad muy interesante que puede añadir valor al producto. Tanto esta funcionalidad, como el poder gestionar los pagos a través de la aplicación y llevar el registro de asistencia de los empleados, entre otras, son funcionalidades que me gustaría desarrollar en actualizaciones

futuras. Pero tanto por su complejidad como por la importancia que puedan tener dentro de la aplicación, no las desarrollaré por el momento.

Teniendo en cuenta mi experiencia como coordinador de comedor escolar y las respuestas de algunos empleados, para poder capturar los requisitos desde la perspectiva del usuario final, crearé historias de usuario detalladas que describan las acciones que un usuario desea realizar en la aplicación y los resultados esperados. Las historias de usuario tienen como objetivo comprender las necesidades del usuario, proporcionando una visión clara de que se espera del sistema.

3.2.1. Historias de usuario

Para desarrollar las historias de usuario hay que plantear cuáles van a ser los agentes implicados, aquellos que harán uso de la aplicación, y que acciones necesitarán poder realizar con esta:

- Administrador: Para poder gestionar correctamente el acceso y la información, habrá un único usuario administrador, el cual ha de poder.
    - Poder iniciar sesión en la aplicación[2].
    - Crear usuarios para los trabajadores de comedor (monitores).
    - Modificar la información relacionada con cada usuario.
    - Eliminar usuarios.
    - Gestionar el acceso a la información por parte de los trabajadores. Es decir, habilitar o deshabilitar el acceso al registro de niños según los cursos de los que sean responsables.
    - Poder consultar y generar informes de asistencia de los alumnos.
    - Poder registrar alumnos con toda la información sobre estos (clases, alergias o intolerancias, etc.
    - Modificar la información de los alumnos.
    - Eliminar alumnos
    - Poder realizar las mismas acciones que el resto de usuarios (trabajadores).

[2] El registro del usuario administrador se hará de forma manual una vez contratado el servicio, por lo que el usuario administrador no ha de poder registrar-se por sí solo.

- Usuario común (trabajador): Como indicamos en el apartado anterior, los administradores crearan usuarios para los trabajadores para que puedan realizar las gestiones necesarias que les correspondan.
    - Poder iniciar sesión en la aplicación.
    - Realizar el registro de asistencia de alumnos.
    - Consultar alergias e intolerancias de los alumnos.
    - Poder consultar el menú escolar.

Teniendo en cuenta que la gestión principal de un comedor escolar es el registro de alumnos y controlar las alergias e intolerancias de estos, los principales agentes implicados son, entonces, el administrador del centro y sus trabajadores. Conforme la aplicación crezca y abarque muchas otras funcionalidades, se tendrán en cuenta otros agentes implicados como pueden ser el personal de cocina o los alumnos en sí (familias).

Una vez planteados los agentes implicados y las acciones que deberán poder realizar, formalizamos las historias de usuario que reflejaran los requisitos funcionales que la aplicación ha de cumplir:

Épicas

- **E1: Registro de Alumnos.**
  Como usuario, quiero poder registrar la asistencia de los alumnos del comedor en el sistema para mantener un registro completo y organizado de los comensales diarios.

- **E2: Consulta de Menús y Dietas Personalizadas.**
  Como usuario del sistema, quiero poder consultar los menús y dietas personalizadas disponibles para los alumnos, permitiéndome acceder fácilmente a la información nutricional y adaptada a las necesidades específicas de cada uno.

- **E3: Gestión de Alergias Alimentarias.**
  Como administrador, necesito poder gestionar las alergias alimentarias de los alumnos, permitiendo registrarlas, mantenerlas actualizadas y asegurarme de que se tengan en cuenta al planificar los menús y las dietas personalizadas.

Usuario Administrador

- **USA-1: Iniciar sesión en la aplicación.**
  Como administrador, quiero poder iniciar sesión en la aplicación con mis credenciales para acceder a las funcionalidades de la aplicación.

- **E1/USA-2: Crear usuarios.**
  Como administrador, quiero poder crear usuarios para los trabajadores del comedor (monitores) para que puedan acceder a la aplicación.

- **E1/USA-3: Modificar información de los usuarios.**
  Como administrador, quiero poder modificar la información de los trabajadores del comedor (monitores), para adaptarla a nuevas circunstancias como cambios de grupos en los que son responsables, o modificación de la contraseña, por ejemplo.

- **E1/USA-4: Eliminar usuarios.**
  Como administrador, quiero poder eliminar usuarios de la aplicación cuando ya no sean necesarios.

- **E1/USA-5: Gestionar acceso a la información.**
  Como administrador, quiero poder habilitar o deshabilitar el acceso al registro de niños para los trabajadores según los cursos de los que sean responsables.

- **E1/USA-6: Consultar y generar informes de la asistencia de los alumnos.**
  Como administrador, quiero poder consultar y generar informes de asistencia de los alumnos para mantener un registro preciso de su asistencia al comedor.

- **E1/USA-7: Crear clase.**
  Como administrador, quiero poder crear nuevas clases dentro del sistema para organizar a los alumnos según su nivel educativo.

- **E1/USA-8: Registrar alumnos.**
  Como administrador, quiero poder registrar alumnos con toda su información relevante (clases, alergias, intolerancias, etc.) para mantener un registro completo y actualizado de los estudiantes.

- **E1/USA-9: Modificar información de los alumnos.**
  Como administrador, quiero poder modificar la información de los alumnos registrados para mantenerla actualizada y precisa**.**

- **E1/USA-10: Eliminar alumno.**
  Como administrador, quiero poder eliminar alumnos de la aplicación cuando ya no pertenezcan al centro escolar.

Usuario (Monitor)

- **US-1: Iniciar sesión en la aplicación.**
  Como monitor, quiero poder iniciar sesión en la aplicación con mis credenciales para acceder a las funcionalidades necesarias para mi trabajo en el comedor.

- **E1/US-2: Registrar asistencia de alumnos.**
  Como monitor, quiero poder registrar la asistencia de los alumnos en la aplicación, para poder llevar un control preciso de los alumnos que hay ese día y de los que soy responsable.

- **E3/US-3: Consultar alergias e intolerancias de alumnos.**
  Como trabajador, quiero poder consultar las alergias e intolerancias de los alumnos registrados para garantizar que se les sirvan alimentos seguros y adecuados.

- **E2/US-4: Consultar menú escolar.**
  Como trabajador, quiero poder consultar el menú escolar en la aplicación para preparar adecuadamente las comidas y responder a las preguntas de los alumnos sobre los alimentos servidos.

Estas historias de usuario reflejan los requisitos funcionales que la aplicación debe cumplir para satisfacer las necesidades del administrador y los trabajadores del comedor escolar. Una vez concretados los requisitos del proyecto, podemos pasar a la fase de Diseño.

3.3. Diseño

En esta fase voy a desarrollar el diseño del sistema para guiar la correcta implementación de este. Para ello, haré uso de diferentes diagramas y mockups para facilitar el desarrollo de la aplicación, que iniciaré en la siguiente fase:

- **Diagramas de Casos de Uso:** Los diagramas de casos de uso describen las interacciones entre los usuarios y el sistema, identificando los diferentes escenarios de uso y las acciones que pueden realizar los usuarios.

- **Diagramas de Clases:** Los diagramas de clases sirven para modelar la estructura y las relaciones entre los diferentes objetos en el sistema. Con ellos, puedo visualizar mejor cómo se conecta todo dentro del sistema.

- **Mockups:** Diseñar mockups detallados permite visualizar la interfaz de usuario y la disposición de los elementos en pantalla. Esto me permitirá obtener retroalimentación temprana del cliente y realizar ajustes antes de avanzar en el desarrollo.

Todo esto me permitirá optimizar la usabilidad de la aplicación desde un inicio, ya que para ello es crucial tener claro las interacciones necesarias que habrá entre usuario y sistema, así como tener un diseño preestablecido el cual poder validar con el usuario y poder basarme en él en el momento de implementar la parte visual de la aplicación.

3.3.1. Diagramas de Casos de Uso

A partir de las historias de usuario que he descrito en el apartado de requisitos, elaboraré los casos de uso que reflejarán las acciones específicas que el usuario realiza para lograr un objetivo particular dentro del sistema, junto con las respuestas del sistema a esas acciones, para identificar los diferentes escenarios de interacción entre el usuario y el sistema, cosa que facilitará la organización y planificación en la fase de Desarrollo.

Caso de uso	**Iniciar Sesión en la Aplicación**
ID	USA-1
Actores	Administrador
Descripción	Permite al administrador iniciar sesión en la aplicación utilizando sus credenciales.
Precondiciones	-
Flujo principal	1. El administrador ingresa su nombre de usuario y contraseña. 2. El sistema verifica las credenciales del administrador. 3. El sistema autentica al administrador y le da acceso a las funcionalidades de la aplicación.
Flujo alternativo	-

Caso de uso	**Crear Usuarios**
ID	USA-2
Actores	Administrador
Descripción	Permite al administrador crear usuarios para los trabajadores del comedor.
Precondiciones	El administrador ha iniciado sesión en la aplicación.
Flujo principal	1. El administrador accede al panel de administración de usuarios. 2. El administrador selecciona la opción para crear un nuevo usuario. 3. El administrador ingresa los datos del nuevo usuario (nombre, contraseña, rol, etc.). 4. El sistema valida la información ingresada y crea el nuevo usuario.
Flujo alternativo	-

Caso de uso	**Modificar Información de los Usuarios**
ID	USA-3

Actores	Administrador
Descripción	Permite al administrador modificar la información de los trabajadores del comedor (monitores).
Precondiciones	El administrador ha iniciado sesión en la aplicación. Hay usuarios creados.
Flujo principal	1. El administrador accede al panel de administración de usuarios. 2. El administrador selecciona el monitor cuya información desea modificar. 3. El administrador edita la información necesaria (por ejemplo, cambios de grupos, modificación de la contraseña). 4. El administrador guarda los cambios. 5. El sistema muestra un mensaje de confirmación indicando que los cambios se han guardado exitosamente. 6. El administrador confirma la modificación. 7. El sistema actualiza la información del monitor en la aplicación.
Flujo alternativo	-

Caso de uso	**Eliminar Usuarios**
ID	USA-4
Actores	Administrador
Descripción	Permite al administrador eliminar usuarios de la aplicación.
Precondiciones	El administrador ha iniciado sesión en la aplicación. Hay usuarios creados.
Flujo principal	1. El administrador accede al panel de administración de usuarios. 2. El administrador selecciona el usuario que desea eliminar. 3. El administrador confirma la eliminación del usuario. 4. El sistema muestra un mensaje de confirmación solicitando al administrador que confirme la eliminación. 5. El administrador confirma la eliminación. 6. El sistema elimina al usuario seleccionado de la aplicación.
Flujo alternativo	-

Caso de uso	**Gestionar Acceso a la Información**
ID	USA-5
Actores	Administrador
Descripción	Permite al administrador habilitar o deshabilitar el acceso al registro de niños para los trabajadores según los cursos de los que sean responsables.
Precondiciones	El administrador ha iniciado sesión en la aplicación. Hay usuarios creados y se ha seleccionado un usuario.

Flujo principal	1. El administrador accede al panel de configuración de acceso. 2. El administrador selecciona los cursos para los que desea habilitar o deshabilitar el acceso. 3. El administrador guarda los cambios realizados.
Flujo alternativo	-

Caso de uso	**Consultar y Generar Informes de Asistencia de Alumnos**
ID	USA-6
Actores	Administrador
Descripción	Permite al administrador consultar y generar informes de asistencia de los alumnos.
Precondiciones	El administrador ha iniciado sesión en la aplicación.
Flujo principal	1. El administrador accede al panel de informes de asistencia. 2. El administrador selecciona el período de tiempo para el que desea generar el informe. 3. El sistema genera el informe de asistencia y lo muestra al administrador.
Flujo alternativo	-

Caso de uso	**Crear Clase**
ID	USA-7
Actores	Administrador
Descripción	Permite al administrador crear una nueva clase dentro del sistema.
Precondiciones	El administrador ha iniciado sesión en la aplicación.
Flujo principal	1. El administrador accede al panel de administración de clases. 2. El administrador selecciona la opción para crear una nueva clase. 3. El administrador ingresa los datos de la nueva clase (nombre). 4. El sistema valida la información ingresada y crea la nueva clase.
Flujo alternativo	-

Caso de uso	**Registrar Alumnos**
ID	USA-8
Actores	Administrador
Descripción	Permite al administrador registrar alumnos con toda su información relevante.

Precondiciones	El administrador ha iniciado sesión en la aplicación.
Flujo principal	1. El administrador accede al panel de registro de alumnos. 2. El administrador ingresa los datos del nuevo alumno (nombre, clase, alergias, intolerancias, asistencia, etc.). 3. El sistema valida la información ingresada y registra al nuevo alumno.
Flujo alternativo	-

Caso de uso	**Modificar Información de los Alumnos**
ID	USA-9
Actores	Administrador
Descripción	Permite al administrador modificar la información de los alumnos registrados.
Precondiciones	El administrador ha iniciado sesión en la aplicación.
Flujo principal	1. El administrador accede al panel de gestión de alumnos. 2. El administrador selecciona el alumno cuya información desea modificar. 3. El administrador realiza los cambios necesarios en la información del alumno. 4. El sistema valida y guarda los cambios realizados.
Flujo alternativo	-

Caso de uso	**Eliminar Alumno**
ID	USA-10
Actores	Administrador
Descripción	Permite al administrador eliminar alumnos de la aplicación cuando ya no pertenezcan al centro escolar.
Precondiciones	El administrador ha iniciado sesión en la aplicación. Hay alumnos creados..
Flujo principal	1. El administrador accede al panel de administración de usuarios. 2. El administrador selecciona el alumno que desea eliminar. 3. El administrador confirma la eliminación del alumno. 4. El sistema muestra un mensaje de confirmación solicitando al administrador que confirme la eliminación. 5. El administrador confirma la eliminación. 6. El sistema elimina al alumno seleccionado de la aplicación.
Flujo alternativo	-

Caso de uso	**Iniciar Sesión en la Aplicación**
ID	US-1
Actores	Monitor
Descripción	Permite al monitor iniciar sesión en la aplicación utilizando sus credenciales.
Precondiciones	Ha sido registrado por el administrador.
Flujo principal	1. El monitor ingresa su nombre de usuario y contraseña. 2. El sistema verifica las credenciales del monitor. 3. El sistema autentica al monitor y le da acceso a las funcionalidades de la aplicación.
Flujo alternativo	-

Caso de uso	**Registrar Asistencia de Alumnos**
ID	US-2
Actores	Monitor
Descripción	Permite al monitor registrar la asistencia de los alumnos en la aplicación.
Precondiciones	El monitor ha iniciado sesión en la aplicación.
Flujo principal	1. El monitor accede al panel de registro de asistencia. 2. El monitor selecciona los alumnos presentes y los registra como asistentes. 3. El sistema guarda la información de asistencia registrada por el monitor.
Flujo alternativo	-

Caso de uso	**Consultar Alergias e Intolerancias de Alumnos**
ID	US-3
Actores	Monitor
Descripción	Permite al monitor consultar las alergias e intolerancias de los alumnos registrados.
Precondiciones	El monitor ha iniciado sesión en la aplicación.
Flujo principal	1. El monitor accede al panel de consulta de alergias e intolerancias. 2. El monitor selecciona un alumno y consulta su información de alergias e intolerancias. 3. El sistema muestra la información de alergias e intolerancias del alumno seleccionado al monitor.
Flujo alternativo	-

Caso de uso	**Consultar Menú Escolar**
ID	US-4
Actores	Monitor
Descripción	Permite al monitor consultar el menú escolar en la aplicación.
Precondiciones	El monitor ha iniciado sesión en la aplicación.
Flujo principal	1. El monitor accede al panel de consulta de menú escolar. 2. El monitor visualiza el menú escolar para el día o la semana actual. 3. El sistema muestra el menú escolar actual al monitor.
Flujo alternativo	-

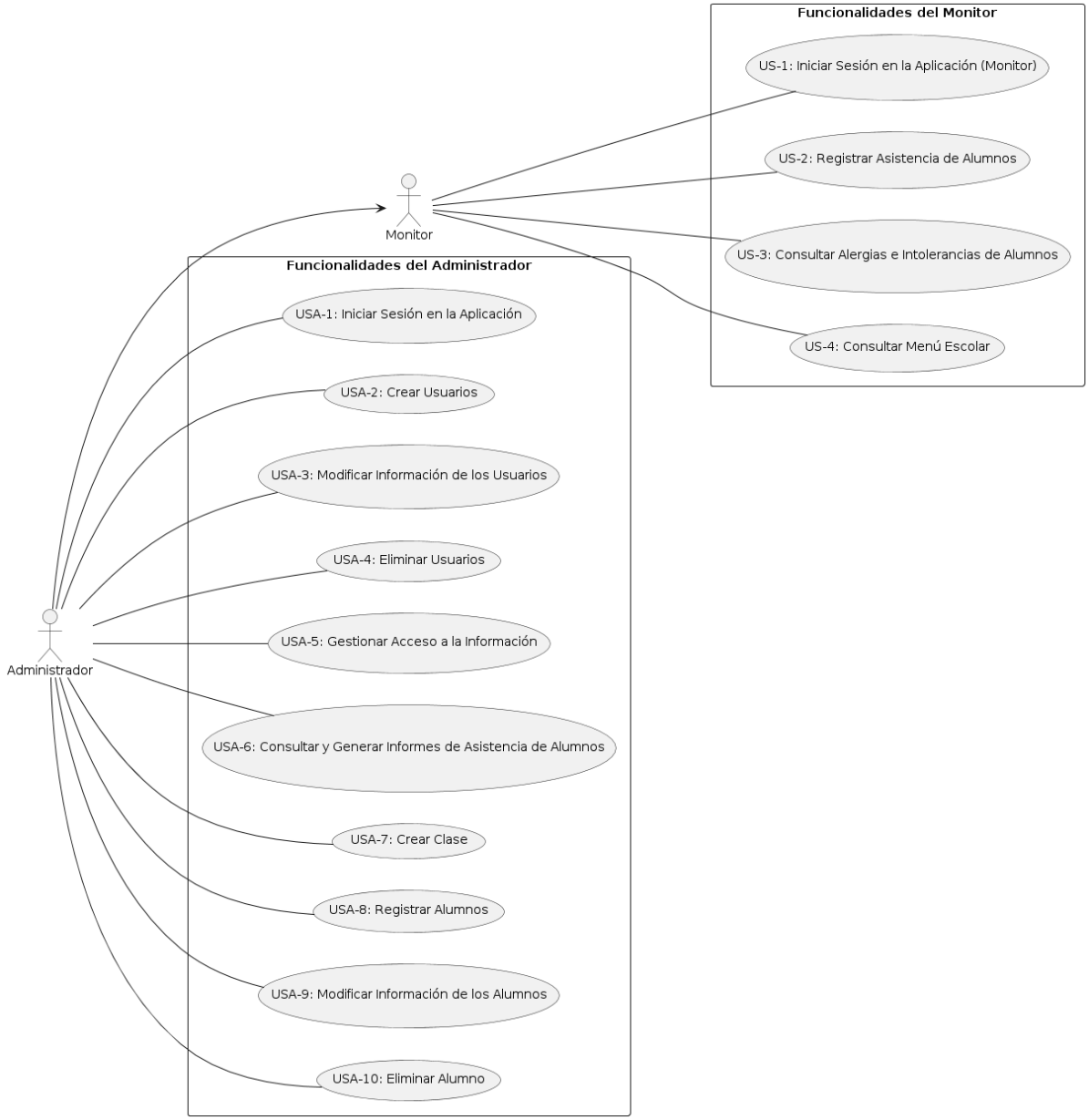


Figura 4. Diagrama de Casos de Uso que refleja las interacciones entre los usuarios y el sistema

3.3.2. Diagrama de clases

Una vez especificadas las funcionalidades y las acciones que han de poder realizar los agentes implicados, el siguiente paso es plantear las entidades que serán necesarias desarrollar para gestionar eficientemente el sistema. Estas entidades representan los elementos fundamentales del sistema y proporcionan una estructura sobre la cual se construirá la lógica y la funcionalidad de la aplicación. Estas son:

- Centro Escolar: Representa la entidad principal del sistema, que estará organizado en cursos y será administrado por un usuario administrado. Incluye atributos como nombre, dirección, teléfono, correo electrónico, y una referencia al administrador del centro. Sobre esta entidad se estructura toda la información del colegio.

- Usuario: Representa a los usuarios del sistema, como administradores, monitores u otros roles. Tiene atributos como nombre, contraseña, acceso y rol.

- Alumno: Representa a los alumnos del comedor. Tiene atributos como nombre, alergias, intolerancias, días habituales y lista de asistencia.

- Alergia: Representa las alergias e intolerancias que pueden tener los alumnos. Tiene atributos como nombre, tipo y severidad.

- Curso: Representa los cursos educativos en los que están organizados los alumnos. Tiene atributos como nombre, cursoId, ciclo y lista de clases.

- Clase: Representa las clases a las que pertenecen los alumnos. Tiene atributos como nombre, claseId y lista de alumnos.

- Asistencia: Representa la asistencia de los alumnos a las clases. Tiene atributos como fecha, estado de presencia y si es habitual.

A continuación, en la siguiente página, podemos ver el diagrama de clases que representa las relaciones que tienen entre ellas:

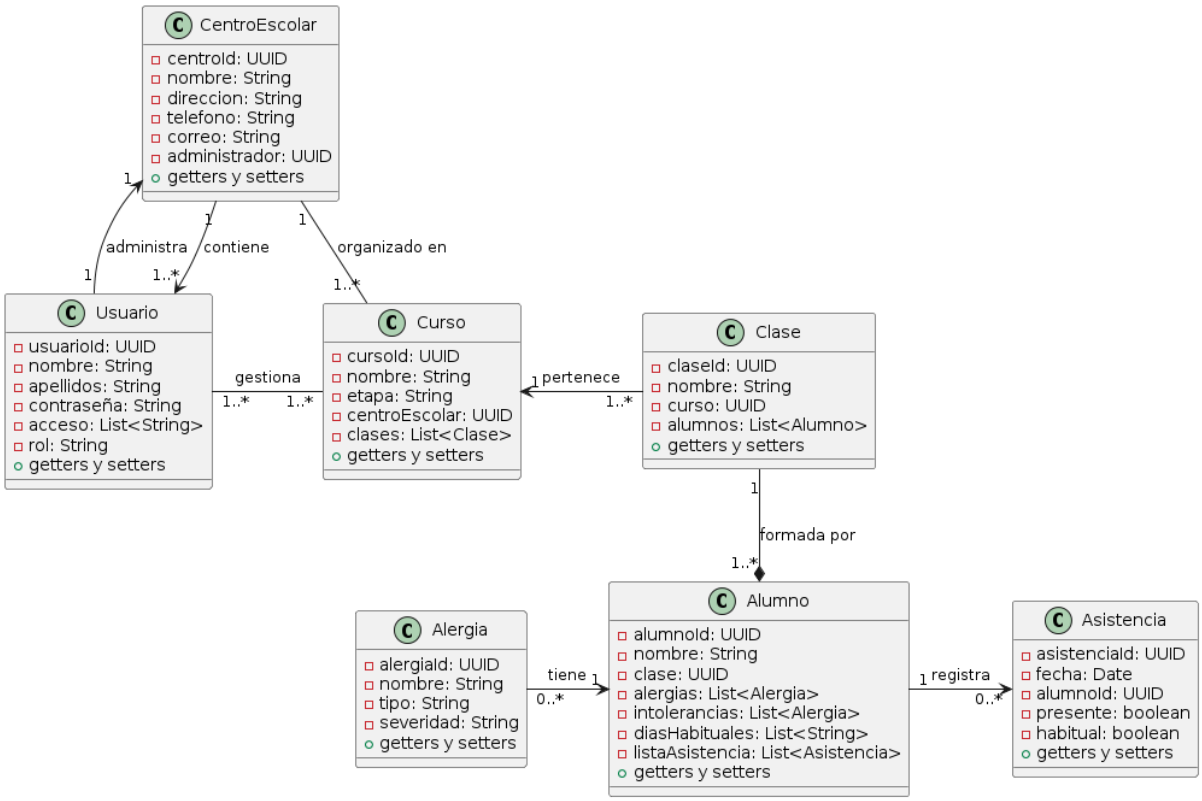


Figura 5. Diagrama de clases donde podemos observar la relación entre entidades

- **Centro Escolar - Curso**: "1" -- "1..*": Cada Centro Escolar está organizado en varios cursos. Esta relación implica que un centro escolar puede tener uno o varios cursos relacionados (por lógica más de uno), pero cada curso está vinculado a un único centro escolar.

- **Usuario - Centro Escolar**: "1" -- "1": Un usuario, específicamente aquel con rol administrador (vinculado a ese centro), gestiona un centro escolar. Esta es una relación directa donde el administrador tiene control completo sobre el centro escolar.

- **Centro Escolar - Usuario**: "1" -- "1..*": Un centro escolar contiene varios usuarios. Esto incluye a administradores, monitores y posiblemente otro personal, todos operando dentro del mismo centro escolar.

- **Usuario - Curso**: "1.." -- "1..": Los usuarios pueden gestionar uno o más cursos, y cada curso puede ser gestionado por uno o más usuarios. Esto facilita la flexibilidad en la administración y la posibilidad de colaboración entre varios usuarios.

- **Alumno - Alergia**: "1" -- "0..*": Un alumno puede tener cero o más alergias asociadas. Esta relación es crucial para el manejo de salud dentro de la escuela, permitiendo un registro detallado de cualquier condición que podría afectar la participación del alumno en actividades escolares, especialmente las relacionadas con la comida. Se incluye como alergia también las intolerancias, cosa que explicaré más adelante.

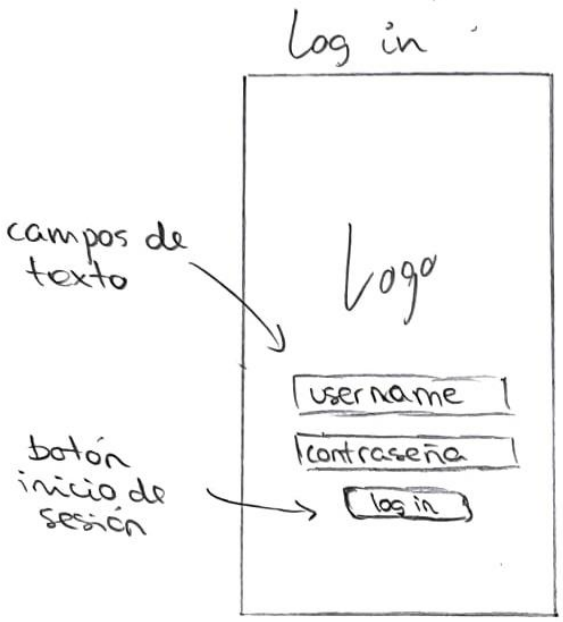
- **Alumno - Asistencia**: "1" -- "0..*": Cada alumno tiene registros de asistencia asociados, los cuales pueden ser muchos o ninguno si el alumno nunca ha usado el servicio de comedor. Esta relación es esencial para llevar un seguimiento de la asistencia diaria de los alumnos.

- **Curso - Clase**: "1" -- "1..*": Cada curso incluye una o más clases. Esta estructura ayuda a organizar los alumnos por nivel educativo y facilita la administración académica dentro del curso.

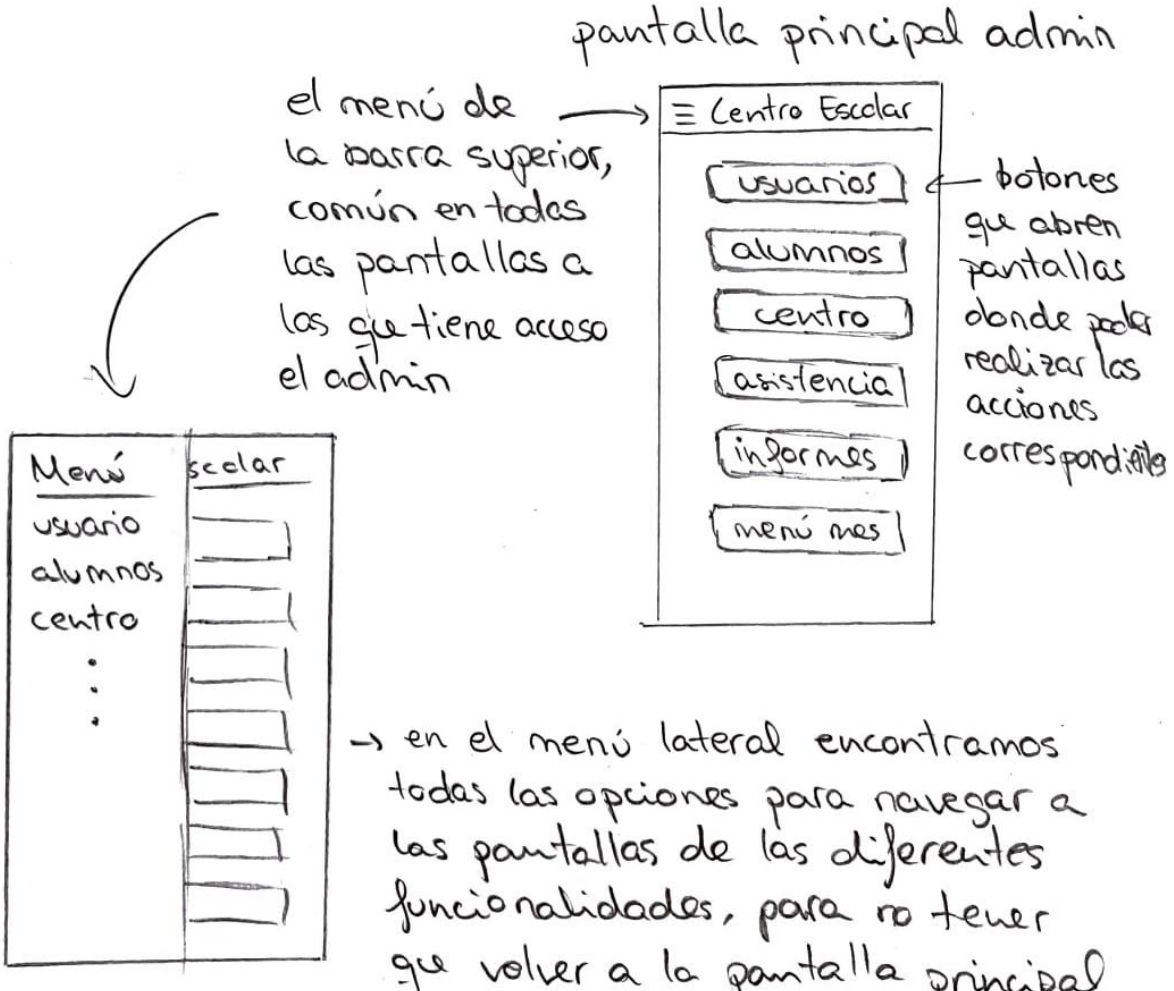
- **Clase - Alumno**: "1" -- "1..*": Cada clase está formada por uno o más alumnos. Esta relación subraya que una clase debe tener al menos un alumno y cada alumno pertenece a una única clase, asegurando la organización adecuada de los estudiantes (Se entiende que las clases tendrán más de un alumno).

Estas relaciones ayudan a establecer un sistema organizado dentro de la aplicación, facilitando la gestión educativa y administrativa de cada centro. Están diseñadas para reflejar la estructura organizativa de un centro escolar y asegurar la gestión adecuada de la información relacionada con alumnos, personal, etc.

3.3.3. Mockups

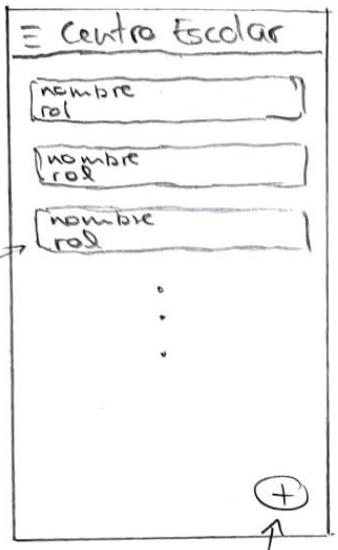


→ pantalla principal. Se abre al iniciar la app. Contiene el logo, dos campos de texto para introducir los datos y el botón correspondiente para iniciar sesión al clicarlo



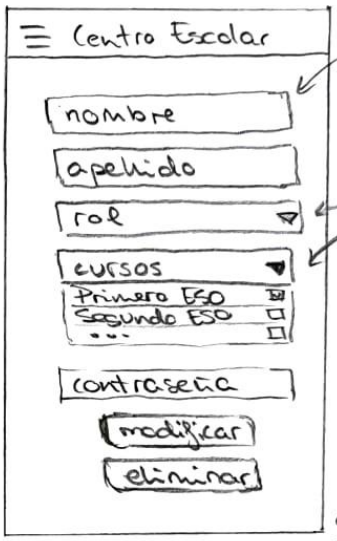
→ en el menú lateral encontramos todas las opciones para navegar a las pantallas de las diferentes funcionalidades, para no tener que volver a la pantalla principal.

gestión de usuarios



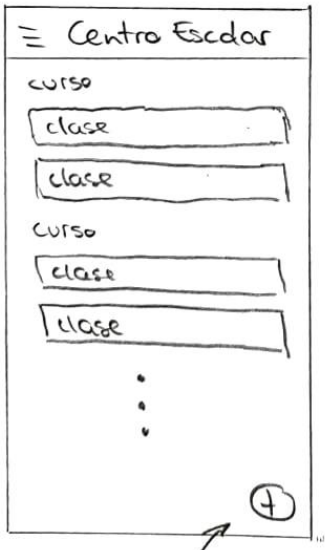
→ en esta pantalla saldrán listados los usuarios. Al seleccionar uno, nos llevará a la página de edición donde poder modificar los datos o eliminar al usuario

pantalla de edición / creación de usuarios



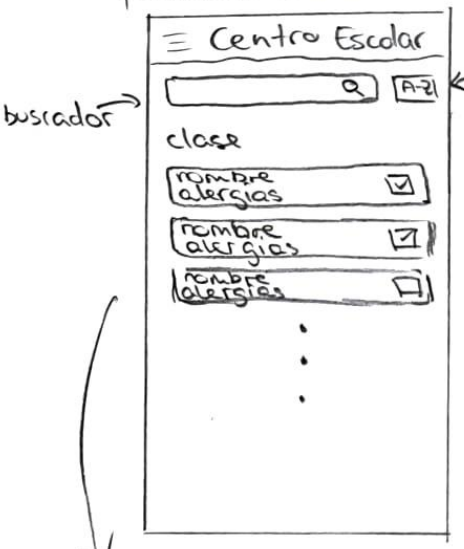
La pantalla de edición será muy similar a la de creación, quitando alguna pequeña modificación de los campos.

gestión de centro



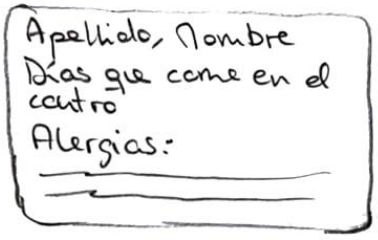
→ seguirá una distribución muy similar a la de usuario. Al clicar el botón de añadir curso ⊕, se abre una pantalla muy similar a la de edición con los campos correspondientes, en función de la selección escogida.

pantalla asistencia ← (pantalla principal monitor y coordinador)

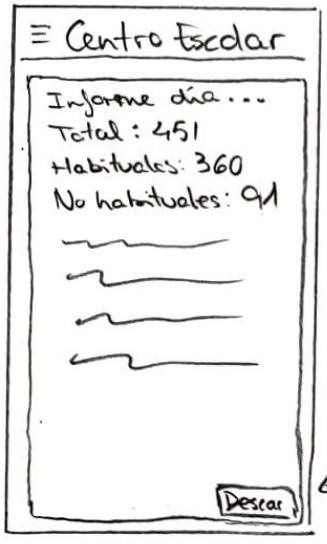


→ esta pantalla será muy similar a la de gestión de alumnos. En esta, tendremos un botón ⊕ para añadir alumnos y no se verá la checkbox en cada uno de estos. La diferencia también estará al clicar en un alumno, que abrirá la pantalla de edición.

en esta, al clicar un alumno se abrirá una tarjeta con toda la información de estos. →



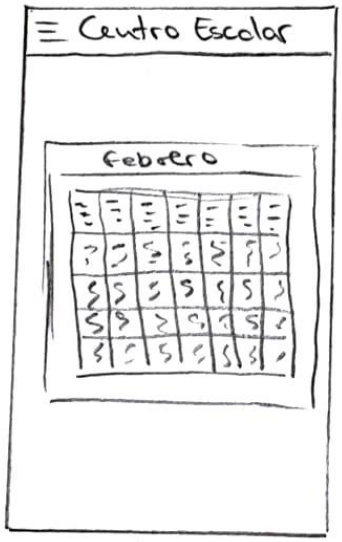
pantalla Informes



→ en esta pantalla encontraremos toda la información de interés de un día es concreto

botón para descargar el informe en pdf

pantalla menú mensual



en esta pantalla podremos ver el menú del mes actual con toda la información correspondiente.

3.3.4. Arquitectura del sistema

Llegados a este punto, ya podemos plantear la arquitectura del sistema. Como mencioné previamente, el objetivo es desarrollar una aplicación nativa para Android que sea compatible con la mayoría de los dispositivos, ya que se utilizará principalmente durante el servicio de comedor y por los trabajadores (monitores) que conforman la plantilla. Dado que la aplicación ofrecerá un servicio que puede ser adoptado por numerosos centros educativos, es probable que algunos prefieran utilizar un dispositivo común para realizar todas las gestiones durante el servicio. Esto puede deberse a que el encargado de controlar la asistencia sea un único empleado (por ejemplo, el coordinador), o por razones de seguridad y protección de datos, prefieran utilizar un dispositivo compartido que permanezca dentro del centro escolar. Por lo tanto, se planea adaptar la aplicación para su uso en tabletas en el futuro.

En cuanto a la protección de datos, como se detallará más adelante, se abordará antes del despliegue de la aplicación (aunque no forma parte de la entrega de este proyecto), ya que es un aspecto crucial que debe considerarse en todo momento y debe implementarse de manera adecuada en la gestión de la información. Dado que la mayoría de los usuarios serán menores de edad y los datos serán relevantes, es aún más esencial aplicar protocolos adecuados para cumplir con todas las normativas de protección de datos y seguridad en la gestión de la información.

He decidido desarrollar la aplicación para dispositivos móviles, ya que considero que es la forma más práctica de usarla inicialmente, basándome en mi experiencia personal y en la forma en que suele desarrollarse el servicio. He elegido el sistema operativo Android, ya que es el más utilizado por los españoles, según un estudio realizado por la CNMC (Comisión Nacional de los Mercados y la Competencia, 2023), lo que facilitará su utilización en los dispositivos de los usuarios.

Con el fin de facilitar la gestión de la información por parte del personal administrativo del centro, tengo como objetivo desarrollar en el futuro una versión más completa de la aplicación para ordenadores. Será mucho más fácil gestionar la información de los alumnos y otros datos necesarios para futuras funcionalidades a través de un ordenador, dispositivo que utilizan principalmente para administrar el centro y donde se encuentran gran parte de los datos necesarios. Por ejemplo, será posible importar la información de los alumnos desde un archivo Excel a la base de datos o exportar informes y datos de asistencia de los alumnos con el mismo formato.

Para el almacenamiento de los datos, haré uso de una base de datos local usando las funcionalidades de Django. Este proporciona una solución completa para la creación y gestión de una base de datos local, así como para la implementación de una API RESTful para interactuar con la base de datos desde una aplicación de Android. Su potente ORM (Object-Relational Mapping) y sus características integradas hacen que sea una opción ideal para el desarrollo de aplicaciones móviles robustas y escalables.

3.3.5. Diseño de la base de datos

Como ya he mencionado, para el almacenamiento de datos crearé una base de datos local utilizando las funcionalidades proporcionadas por Django (Django Software Foundation, 2024). Django permite interactuar con la base de datos utilizando objetos Python en lugar de consultas SQL directas, facilitando enormemente la gestión de la base de datos y habiéndolo utilizado durante la carrera me ha parecido una buena opción.

Para crear la base de datos local con Django, primero definiré los modelos de datos que representan las entidades de la aplicación. Por ejemplo, tendré un modelo para representar a los alumnos, usuarios (monitores y administrador), etc. Estos modelos se definen como clases en Python, utilizando los campos proporcionados por Django para definir la estructura de la base de datos.

Una vez que haya definido los modelos, ejecutaré migraciones en Django para crear las tablas correspondientes en la base de datos. Las migraciones son scripts de Python que describen los cambios en la estructura de la base de datos y se pueden aplicar de manera incremental para mantener la consistencia entre el modelo de datos y la base de datos real.

Para interactuar con estos modelos mediante una API web, utilizaré el Django REST Framework, que facilita la creación de API RESTful. Los serializadores de Django me permitirán transformar los objetos del modelo en formatos JSON para su fácil consumo por la aplicación cliente, y viceversa. Las vistas en Django, especialmente las vistas basadas en clases que ofrece Django REST Framework, me ayudarán a definir la lógica para manejar las diferentes solicitudes de la API, como obtener, añadir, modificar y eliminar registros.

Aunque no desarrollaré aspectos avanzados de seguridad en este proyecto, Django ofrece varias opciones robustas para gestionar la autenticación y la seguridad. Por defecto, Django maneja la autenticación de usuario y la sesión, y permite configurar permisos y grupos de usuarios para controlar el acceso a las funcionalidades de la aplicación de manera segura. Estas herramientas serán suficientes para asegurar que sólo los usuarios autorizados puedan acceder a la API y realizar operaciones en ella.

Una vez que la base de datos esté configurada, mediante el ORM de Django realizaré consultas y manipularé los datos en la base de datos. Por ejemplo, para recuperar todos los alumnos de una clase de la base de datos utilizando una consulta sencilla en Python. Además de gestionar la base de datos local, usaré la API RESTful creada para que la aplicación de Android pueda interactuar con la base de datos de forma remota.

3.3.6. Diseño de software

En el desarrollo de mi aplicación, he decidido aplicar la arquitectura Clean Architecture (Robert C. Martin, 2012), principalmente por recomendación de un compañero que trabaja

en el sector del desarrollo de software, y por los beneficios que yo mismo he podido observar al investigar sobre ella.

Esta arquitectura tiene como objetivo crear un software modular, flexible y mantenible a largo plazo, lo cual es fundamental para el éxito de cualquier proyecto. Los principios fundamentales de Clean Architecture que guiarán el diseño de mi aplicación son:

- **Independencia de dominios:** La lógica de negocio debe estar separada de las interfaces de usuario, bases de datos y frameworks externos. Esto significa que las entidades y casos de uso del dominio no deben depender de tecnologías específicas, lo que les permite ser reutilizados en diferentes contextos.

- **Entidades de dominio:** Las entidades del dominio deben representar conceptos del mundo real y no estar ligadas a tecnologías específicas. Por ejemplo, una entidad Alumno debe representar las características de un alumno real, como su nombre, apellidos, curso, alergias, etc., sin depender de cómo se almacenará o se presentará en la interfaz de usuario.

- **Casos de uso:** Los casos de uso definen el comportamiento del sistema y encapsulan la lógica de negocio. Cada caso de uso se encarga de una tarea específica, como registrar una asistencia, consultar un menú o gestionar una alergia. Los casos de uso interactúan con las entidades de dominio para realizar su función.

- **Adaptadores:** Los adaptadores traducen las entidades de dominio y los casos de uso a formatos compatibles con las interfaces de usuario, bases de datos y frameworks externos. Por ejemplo, un adaptador puede convertir una entidad Alumno en un objeto JSON para ser enviado a la interfaz web, o puede convertir un caso de uso de registro de asistencia en una consulta SQL para ser ejecutada en la base de datos.

- **Estructura en capas:** El código se organiza en capas concéntricas, con las capas internas independientes de las capas externas. Esto significa que las capas internas no deben conocer las tecnologías específicas utilizadas en las capas externas, lo que aumenta la modularidad y la flexibilidad del sistema.

Para implementar Clean Architecture en mi aplicación, seguiré la siguiente estructura de capas:

- **Capa de Entidades:** En esta capa se define las entidades de dominio que representan los conceptos fundamentales del sistema, como Alumno, Menú, Alergia, Asistencia, etc. Estas entidades no dependerán de ninguna tecnología específica y se enfocarán en representar el modelo de negocio de la aplicación.

- **Capa de Casos de Uso:** En esta capa se define los casos de uso que definen el comportamiento del sistema y encapsulan la lógica de negocio. Cada caso de uso se encargará de una tarea específica, como registrar una asistencia, consultar un menú

o gestionar una alergia. Los casos de uso interactuarán con las entidades de dominio para realizar su función.

- **Capa de Adaptadores:** En esta capa se define los adaptadores que traducen las entidades de dominio y los casos de uso a formatos compatibles con las interfaces de usuario, bases de datos y frameworks externos. Por ejemplo, un adaptador puede convertir una entidad Alumno en un objeto JSON para ser enviado a la interfaz web, o puede convertir un caso de uso de registro de asistencia en una consulta SQL para ser ejecutada en la base de datos.

- **Capa de Infraestructura:** En esta capa se implementa las interfaces de usuario, bases de datos y frameworks externos utilizados por la aplicación. Esta capa no dependerá de las otras capas y se enfocará en proporcionar los mecanismos necesarios para que el sistema funcione.

La implementación de Clean Architecture en mi aplicación me permitirá obtener varios beneficios:

- **Modularidad:** El código se organizará en módulos independientes, lo que facilitará su desarrollo, mantenimiento y pruebas.

- **Flexibilidad:** La arquitectura será flexible y se podrá adaptar a cambios en los requisitos del sistema sin afectar el núcleo del mismo.

- **Mantenibilidad:** El código será más fácil de entender y modificar, lo que reducirá los costos de mantenimiento a largo plazo.

- **Reusabilidad:** Los componentes de la arquitectura podrán ser reutilizados en otros proyectos.

- **Testabilidad:** La arquitectura facilitará la realización de pruebas unitarias y de integración, lo que garantizará la calidad del código.

Además de los beneficios mencionados anteriormente, Clean Architecture también me permitirá:

- **Reducir el riesgo de errores:** La modularidad y la independencia de las capas ayudarán a prevenir la propagación de errores y facilitarán su detección y corrección.

- **Facilitar la incorporación de nuevas tecnologías:** La arquitectura flexible de Clean Architecture me permitirá incorporar nuevas tecnologías en el futuro sin tener que realizar cambios drásticos en el código existente.

En definitiva, Clean Architecture me permitirá crear una aplicación sólida, robusta y escalable, que cumpla con las necesidades de los usuarios y que me permita alcanzar los objetivos de mi proyecto.

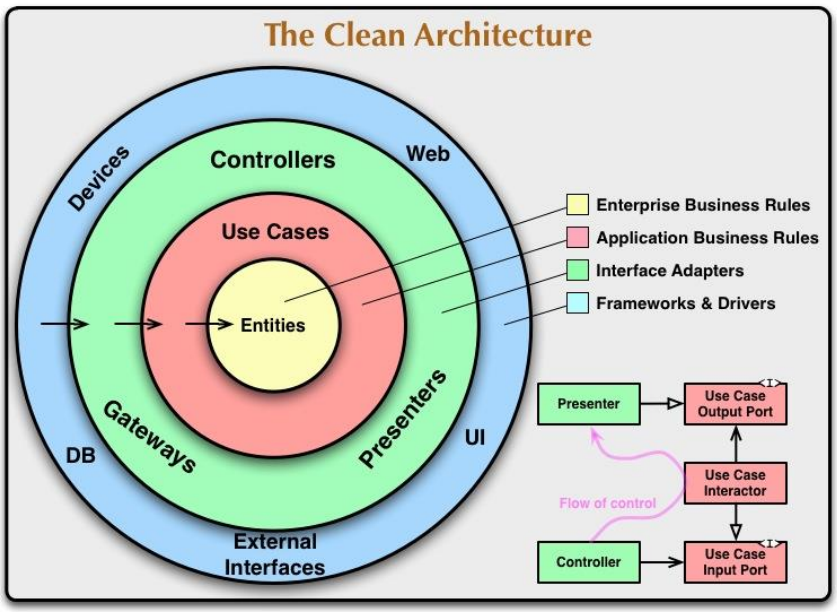


Figura 6. Ilustración de Clean Architecture

3.4. Desarrollo

La fase de desarrollo ha sido a la que más horas he tenido que dedicar. Como más adelante explicaré también en la planificación temporal, la he organizado alrededor de los Use Case. Aunque en algún momento me he encontrado desarrollando más de una al mismo tiempo, por lo general he ido implementándolas de una a una, cosa que me ha facilitado mucho la organización y que todo fuese encajando a medida que iba avanzando.

Antes de profundizar más en detalle, aunque es algo que me he encontrado a medida que iba desarrollando, quiero enfatizar la importancia de las entidades en el desarrollo. Al aplicar la arquitectura Clean, estas representan el modelo de negocio de la aplicación, y definirlas correctamente (o plantearlas, mejor dicho) influirá en la lógica y en el funcionamiento óptimo de la aplicación. A medida que he ido avanzando en el desarrollo, me he dado cuenta de que algunas de las entidades no las había planteado bien o aplicando alguna modificación facilitaba y mejoraba el funcionamiento de la aplicación, ya sea influenciando en la manera en que se obtiene y se almacena la información como en la lógica necesaria para implementar alguna funcionalidad.

Las principales modificaciones que he hecho y los motivos son las siguientes:

- **Alumnos pasa a tener una única lista de alergias**: Las intolerancias se tendrán en cuenta como una alergia más y pasa a ser una categoría de severidad.

- **Se elimina el atributo ‘tipo’ en Alergia**: Aunque lo ideal sería plantear la gestión de las alergias con un dietista, cosa que haré antes de empezar a ofrecer el servicio, he

decidido clasificar las alergias por su severidad directamente para simplificar la gestión de estas.

- **La gestión de Asistencia se modifica de la siguiente manera**: Asistencia deja de tener los atributos alumnoId, presente o habitual. Esto es porque ya no van a depender de los alumnos en sí, y a generarse una por alumno y día, sino que cada centro generará un objeto de asistencia por día, y esta almacenará los alumnos que han asistido en dos listas, en función de si aquel día se deberían de quedar o son esporádicos. Esto hace que tanto las llamadas a la base de datos se minimicen, como la lógica para recopilar información después para los informes sea más sencilla, optimizando los procesos.

La relación entre las entidades es crucial para realizar consultas eficientes y para mantener la integridad y coherencia de los datos. Es por esto que he decidido realizar estas modificaciones. Aunque en el caso de Asistencia implicara tener que rehacer código que ya funcionaba, considero que estos cambios han contribuido en ese sentido y era necesario invertir tiempo en hacer estas correcciones.

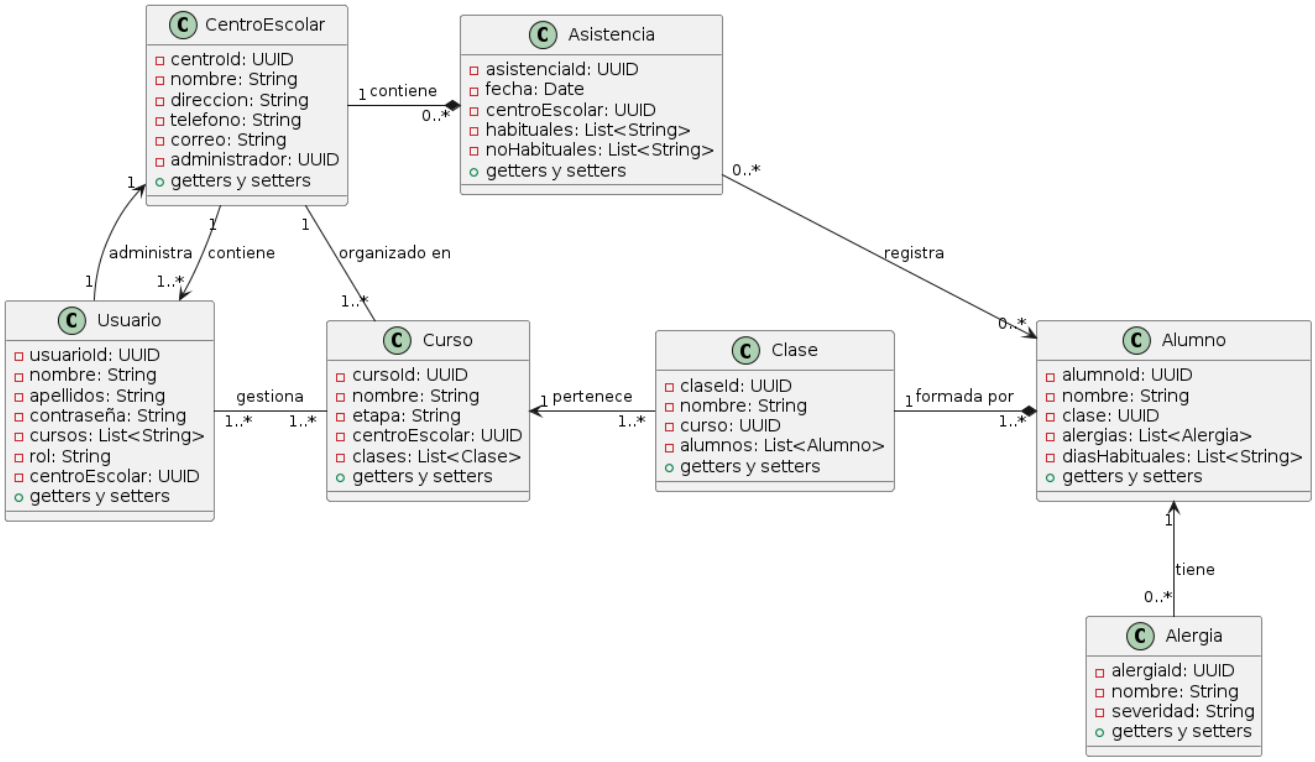


Figura 7. Diagrama de clases actualizado

3.4.1. Documentación de la API y Arquitectura

Backend

Como ya he mencionado anteriormente, decidí implementar una base local en Django porque por tiempo y recursos era lo más práctico a la vez que funcional y robusto de desarrollar para el proyecto, por todas las herramientas que facilita. He usado Django Rest Framework para construir la API, ya que facilita la creación de interfaces RESTful con características como serialización, autenticación, y manejo de permisos, ayudando a desarrollar APIs de manera rápida y con un código mantenible.

Por un lado, encontramos los modelos, que definen la estructura de la base de datos y contienen toda la información que se necesita almacenar ('models.py'). El modelo Usuario, por ejemplo, extiende la clase 'AbstractUser' de Django, facilitando la integración con el sistema de autenticación de Django y añadiendo campos personalizados como el rol dentro del sistema (Administrador, Coordinador, Monitor). O el modelo Alumno, que Contiene datos de los alumnos, incluyendo relaciones a las clases a las que asisten y sus posibles alergias, representadas mediante una relación 'ManyToMany' con el modelo.

Por otro lado, he implementado los serializadores para transformar conjuntos de datos complejos en JSON y viceversa. Esto es esencial para la comunicación entre el backend y el frontend. Por ejemplo, el serializador de alumno (AlumnoSerializer) maneja no solo la serialización de datos básicos del alumno, sino también la integración de datos relacionales como las alergias, asegurando que toda la información relevante sea accesible de manera eficiente y segura.

También he desarrollado las vistas (ViewSets) necesarias, y utilizado el enrutamiento automático en Django REST Framework para organizar las peticiones a la API de manera eficiente. Cada operación de datos, desde crear hasta eliminar registros (GET, POST, PUT, DELETE), es manejada consistentemente a través de métodos HTTP. He tenido que personalizar algunos de los métodos para asegurarme que gestiona bien los modelos más complejos en los que encontramos relaciones 'ManyToMany', por ejemplo. Las rutas están definidas en 'urls.py', asociando URLS con vistas específicas, lo que facilita la expansión o modificación del API.

Para añadir una capa de seguridad, he usado la autenticación basada en tokens que ofrece Django de la siguiente forma:

- Al autenticar a un usuario, si las credenciales son válidas, se genera un token asociado al usuario (Token.objects.get_or_create(user=user)). Este token se devuelve en la respuesta y debe ser utilizado por el cliente en las siguientes peticiones para acceder a las vistas protegidas de la API.

- Para las vistas que requieren autenticación, como CentroEscolarViewSet o UsuarioViewSet, entre otros, se configura las clases de autenticación (authentication_classes = [TokenAuthentication]). Esto significa que cada solicitud a estos endpoints debe incluir un token válido en su encabezado HTTP para ser

autorizada. Si el token es válido, la solicitud continua de forma correcta; si no, se devuelve un error de autenticación.

- Junto con la autenticación por token, además, se utiliza controles de permisos (permission_classes = [IsAuthenticated]). Esto garantiza que solo los usuarios autenticados, es decir, aquellos que han proporcionado un token válido, puedan realizar operaciones como leer, crear, actualizar o eliminar recursos.

Cuando un usuario intenta acceder a los datos de 'Alumno', por ejemplo, debe proporcionar un token que se verifica con los tokens almacenados en la base de datos. Si el token es correcto, se le permite ver o modificar los datos según los permisos que tenga configurados. Esto garantiza que todas las interacciones con el backend sean seguras. Además de esto, las contraseñas se almacenan de manera segura utilizando hash.

Para la gestión de dependencias y configuraciones, se utiliza el archivo 'settings.py', donde se define la configuración de la base de datos, aplicaciones instaladas, middleware, y opciones de autenticación y seguridad.

En el backend también se gestiona la creación de objetos de 'Asistencia', de manera que al iniciar sesión un usuario del centro, se comprueba si es la primera vez que se inicia sesión en ese centro y en caso afirmativo se crea un objeto para cada día de la semana.

```python
def inicializar_asistencia_semanal(user):
    hoy = timezone.now().date()
    inicio_semana = hoy - timedelta(days=hoy.weekday())  # Lunes de la semana actual
    centro = user.centro_escolar  # Asume que cada usuario tiene un centro asociado

    if centro and (centro.ultima_inicializacion is None or centro.ultima_inicializacion < inicio_semana):
        for i in range(7):
            fecha = inicio_semana + timedelta(days=i)
            Asistencia.objects.get_or_create(fecha=fecha, centro_escolar=centro)
        centro.ultima_inicializacion = inicio_semana
        centro.save()
```

Figura 8. Método con el que inicializo las asistencias cada semana por centro

```python
class CustomAuthToken(APIView):
    permission_classes = []

    def post(self, request, *args, **kwargs):
        logger.debug("Datos recibidos para autenticación: %s", request.data)
        username = request.data.get('username')
        password = request.data.get('password')
        user = authenticate(username=username, pass
        if user:
            token, _ = Token.objects.get_or_create(user=user)
            usuario_data = UsuarioSerializer(user).data
            usuario_data['token'] = token.key  # Añadir el token a los datos del usuario
            inicializar_asistencia_semanal(user)  # Inicializar la asistencia semanal
```

(function) user: Any

Figura 9. Llamada al método dentro al autenticar el usuario.

Django ofrece un panel de administración integrado muy útil, que se genera automáticamente y permite manejar las entidades de la base de datos de forma visual. Lo mejor es que no requiere código adicional para configuraciones básicas, aunque se puede

personalizar ampliamente. Es ideal para realizar tareas administrativas como añadir, editar o eliminar registros. Esta herramienta me ha sido de gran ayuda para supervisar rápidamente los datos y gestionar la aplicación sin necesidad de escribir consultas complejas o interfaces de usuario desde cero. Lo he usado principalmente para crear centros escolares y añadirles un usuario para que lo administre, ya que no es una funcionalidad que quiero que se realice desde la app.

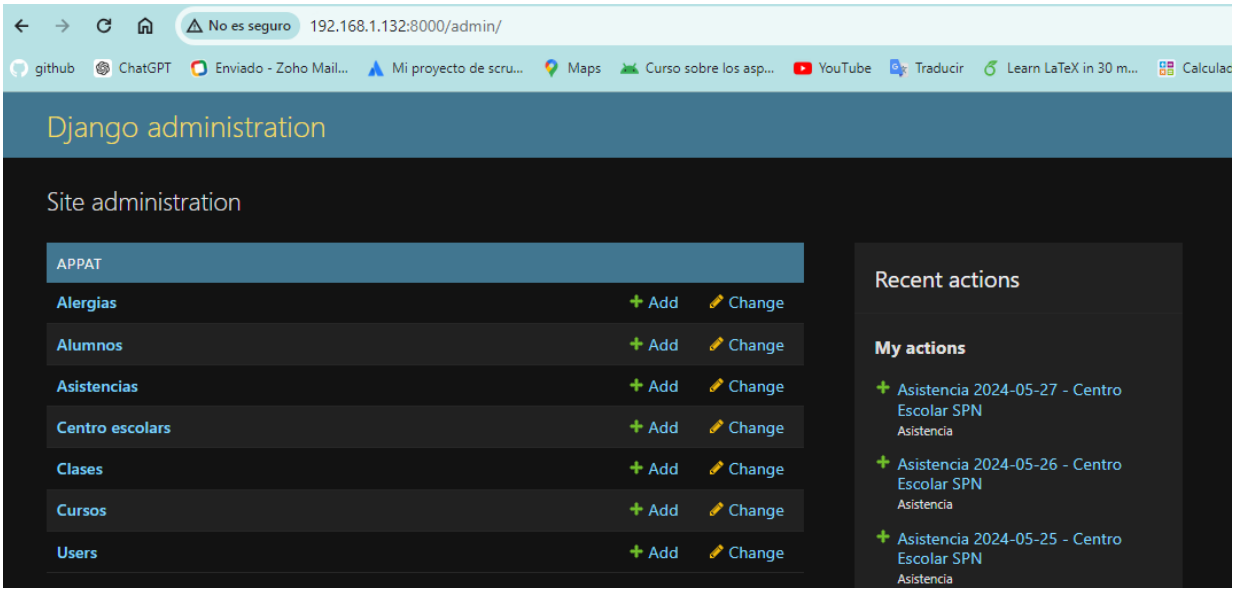


Figura 10. Panel de administrador que ofrece Django.

Frontend

La aplicación la he desarrollado en Android Studio, usando el lenguaje de programación kotlin, siguiendo la arquitectura Clean y el patrón Model-View-ViewModel (MVVM). Esto me ha permitido separar la lógica de la interfaz, lo que facilita la mantenibilidad y escalabilidad de la app.

Aunque durante la carrera ya use el entorno de Android Studio para desarrollar una aplicación, esta la hice con mis compañeros en java, por lo que me enfrentaba a un nuevo reto al implementar la aplicación en kotlin, del cual había oído hablar, pero nunca lo había usado. Gente de mi entorno que se dedica al mundo del desarrollo de software hacía tiempo que me hablaban de él y me recomendaban que aprendiera a usarlo, por lo que aproveché la oportunidad de hacerlo para este proyecto. Entre las ventajas de kotlin encontramos que usa una sintaxis más concisa, se reduce el código repetitivo y un manejo de concurrencia más simple con coroutines. Además, es totalmente compatible con Java y tiene el respaldo oficial de Google para Android.

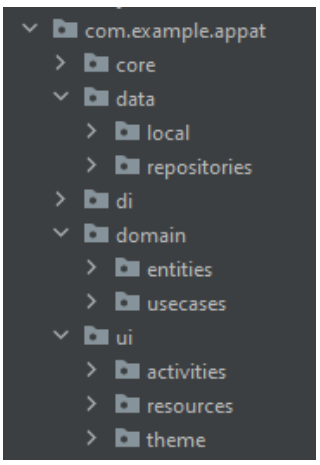


Figura 11. Directorios de la app

Para el desarrollo de la aplicación, he aplicado la Arquitectura Clean. Como he explicado en el apartado de diseño de software, es una metodología de diseño de software que enfatiza la separación de responsabilidades en diferentes capas. Aunque en la introducción de este apartado he

mencionado que he distribuido el desarrollo de la aplicación en torno a los Use Case, voy a explicar el código según como está distribuido en las capas en las que se basa Clean Architecture. En la figura 11, podemos ver la distribución de los archivos, los que se encuentran en su mayoría distribuidos en tres directorios principales, que son *domain*, *ui* y *data*:

- **Domain**: Contiene las entidades de dominio y los casos de uso, representando la lógica de negocio y las reglas del sistema.
    - **Entities (capa de Entidades)**: En este están las clases que representan los conceptos fundamentales del sistema, como *Alumno.kt*, *Alergia.kt*, *CentroEscolar.kt*, etc. Se corresponde a la capa
    - **UseCases (capa de casos de Uso)**: En este se agrupan las clases que definen los casos de uso, que encapsulan la lógica de negocio y el comportamiento del sistema, como *IniciarSesionUseCase.kt* o *ModificarAlumnoUseCase.kt*.

- **Data (capa de Adaptadores)**: Contiene los adaptadores que traducen las entidades de dominio y los casos de uso a formatos compatibles con las interfaces de usuario, bases de datos y frameworks externos. En este encontramos dos directorios importantes:
    - **Local**: Por una parte, contiene los Data Transfer Objects (DTOs), que son los objetos que representan los datos que se transfieren entre capas, como *AlumnoDTO.kt* o *AlergiaDTO.kt*. Por otro lado, encontramos el archivo que maneja la comunicación con el backend, *ApiService.kt* (capa de Infraestructura). Esto lo hace usando Ktor, y en él se configuran las solicitudes HTTP para interactuar con los endpoints del servidor, permitiendo operaciones como el registro de usuarios, la consulta de asistencia y la gestión de datos de alumnos. Este servicio facilita el envío y recepción de datos en formato JSON, asegurando una conexión eficiente y segura. Ktor (JetBrains, s.f.) es un framework de Kotlin para construir aplicaciones web y servicios backend de forma rápida y eficiente. Destaca por su modularidad, soporte para coroutines y compatibilidad multiplataforma. Ideal para APIs RESTful, microservicios y aplicaciones web, facilita la configuración flexible y el manejo de solicitudes HTTP asíncronas.
    - **Repositories**: En este están las Interfaces que definen los métodos para interactuar con las fuentes de datos y las clases que los implementan, como *AlumnoRepository.kt*.

- **UI (capa de Infraestructura)**: Maneja la interacción con el usuario y refleja los cambios en la UI.

- **Activities:** contiene las activities, que son los componentes que interactúan directamente con el usuario, como *AsistenciaManagementActivity.kt*.

- **ViewModels**: aquí están los ViewModels, que gestionan la lógica de presentación y mantienen el estado de la UI, utilizando LiveData y StateFlow (librerías de kotlin que aplican el patrón observer, donde observan los datos de la data base) para la reactividad, como *AsistenciaManagementViewModel.kt*.

Para entender mejor cómo interactúan estas capas, voy a explicar el flujo usando el ejemplo del caso de uso registrar alumno:

1. **Capa de Infraestructura (UI + MVVM)**: El usuario introduce los datos del alumno en *RegistrarAlumnoActivity.kt* para que *RegistrarAlumnoViewModel.kt* reciba estos datos y llame al caso de uso correspondiente.

2. **Capa de Casos de Uso (Use Cases)**: *RegistrarAlumnoUseCase.kt* maneja la lógica para registrar un alumno. Valida los datos y realiza cualquier procesamiento necesario antes de interactuar con los repositorios.

3. **Capa de Entidades (Entities)**: Las entidades como *Alumno.kt*, *Alergia.kt* y *Curso.kt* representan los datos y la lógica de negocio central que son manipulados durante el proceso de registro.

4. **Capa de Adaptadores (Repositories + DTOs)**: *AlumnoRepository.kt* maneja la solicitud para registrar el alumno, haciendo uso de *APIService.kt* para enviar los datos al backend a través de una llamada HTTP utilizando Ktor. Convierte las entidades de dominio a DTOs para la transferencia de datos, y viceversa.

5. **Capa de Infraestructura (Servicios de Red)**: *APIService.kt* realiza la solicitud HTTP para registrar el alumno en el backend y devuelve la respuesta.

Este enfoque facilita la prueba, el mantenimiento y la escalabilidad de la aplicación, asegurando que cada componente esté aislado y pueda ser desarrollado y testeado independientemente.

Además de los tres directorios mencionados, en la figura 11 podemos observar un directorio llamado ‘di’. En él encontraremos el archivo *KoinModule.kt* donde implemento el patrón de inyección de dependencias, el cual es fundamental en el desarrollo de software que facilita la creación y gestión de las dependencias entre los distintos componentes de la aplicación. En mi proyecto, he utilizado Koin (Kotzilla, 2024) para implementar este patrón. La inyección de dependencias me permite desacoplar las clases, mejorando la mantenibilidad y la escalabilidad del código. Con Koin, puedo definir módulos donde se especifican cómo se crean y proporcionan las instancias de las clases que mi aplicación necesita. Esto simplifica la gestión de dependencias, ya que puedo cambiar fácilmente las implementaciones sin modificar el código que las utiliza.

En resumen, para desarrollar la aplicación en kotlin he aplicado la arquitectura Clean y el patrón MVVM. Al separar las responsabilidades en capas bien definidas, he logrado una estructura más organizada, facilitando el desarrollo y el mantenimiento. Complementariamente, la implementación de la inyección de dependencias con Koin ha mejorado la flexibilidad y la capacidad de prueba del sistema.

El resultado de la implementación de la aplicación lo podéis encontrar en el apartado 4, donde detallo, además, la implementación de la Use Case US-3 y USA-6.

3.5. Planificación Temporal

4 de enero de 2024

El plan temporal que me propongo abarcara desde enero hasta mayo de 2024, repartido en cuatro etapas. La idea es empezar el año adentrándome en el aprendizaje de Kotlin y familiarizándome con Android Studio, seguido de un mes de febrero enfocado en sentar las bases del proyecto a través de las etapas de requisitos y diseño. Los meses de marzo y abril los destinaré a la implementación del software, culminando con la finalización de la redacción de la memoria en mayo.

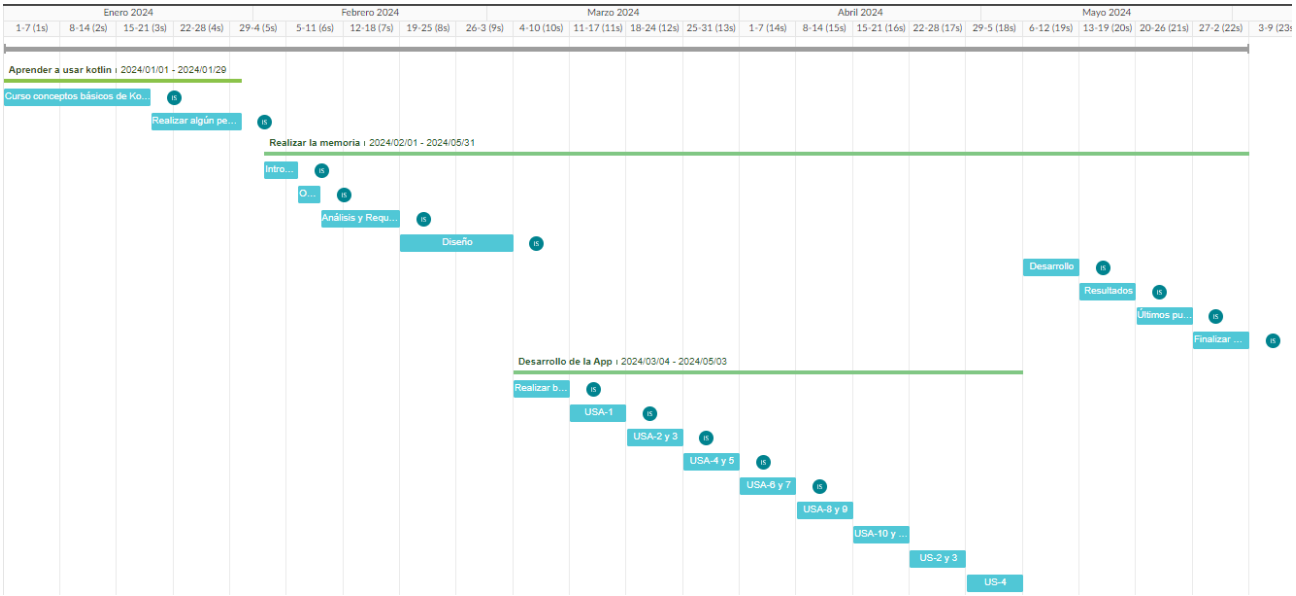


Figura 12. Diagrama de Gantt con la planificación inicial.

26 de mayo de 2024

Aunque he logrado seguir en gran parte la planificación inicial, por diferentes motivos ha habido etapas que he tenido que reajustar en el tiempo de dedicación. A continuación, explico con detalle cada una de ellas.

El mes de enero lo dediqué a familiarizarme con el entorno de programación de Android Studio, concretamente con el lenguaje kotlin, del cual había oído hablar, pero nunca lo había usado. Durante este mes, dediqué el tiempo que tenía en desarrollar algún pequeño proyecto para empezar a comprender conceptos básicos y tener una base sobre la que empezar a desarrollar mi proyecto más adelante (Google for Developres, s.f.).

El mes de febrero me centré en estructurar el proyecto de forma detallada. Durante este mes empecé a elaborar la memoria y a trabajar en la fase de requisitos y de diseño de la aplicación. Aunque desde un inicio tenía bastante claro cómo enfocar el proyecto, fui reformulando conceptos y planteamientos a lo largo de las semanas y, sumado a motivos personales que me quitaron más tiempo del esperado durante los tres primeros meses del año, tardé más de lo esperado en dar por completada estas fases y en empezar con el desarrollo.

A finales de marzo empecé con el desarrollo del proyecto. Las primeras semanas fueron las más complicadas, ya que, a pesar de la preparación previa al inicio, al ponerme con la implementación de la aplicación tardé un poco en coger ritmo y, sumado a algún que otro error que me mantuvo centrado en la resolución de este, no fue hasta ya entrado abril que empecé a avanzar a un buen ritmo. Esto hizo que acabara la fase de desarrollo a mediados de mayo.

Para organizarme en la elaboración de la aplicación, utilicé la herramienta Kanban para visibilizar el trabajo que tenía que realizar, concretamente enfocado en el desarrollo de cada Use Case. Distribuidos en tres columnas: Todo, InProgress y Done. Esta herramienta es comúnmente utilizada en equipos de trabajo de desarrollo de software que están formados por más de un miembro, pero a mí me ha ayudado a organizarme sobre todo en el momento de clarificar que Use Case ya tenía completamente desarrolladas.

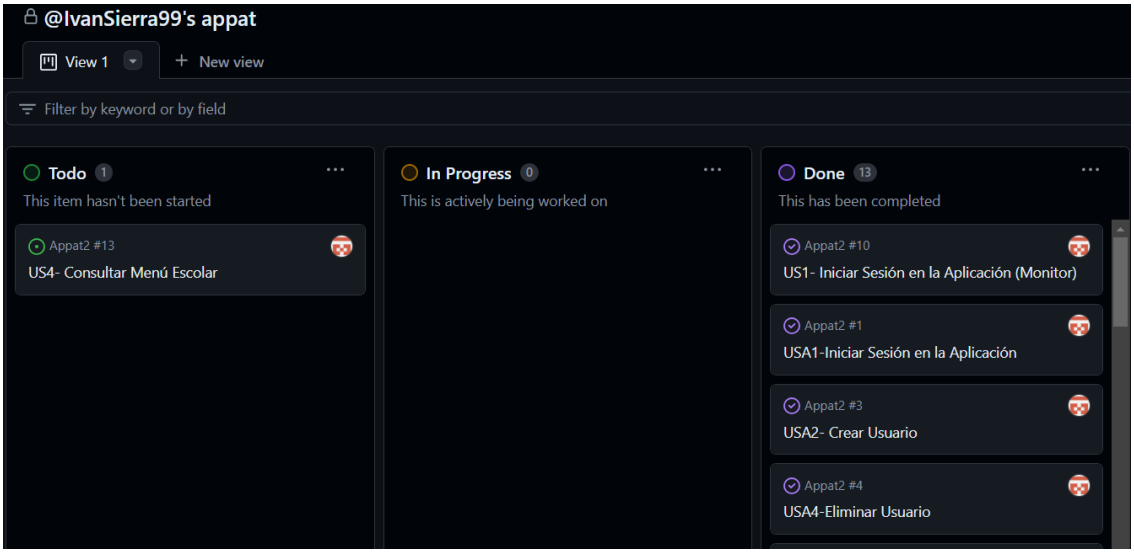


Figura 13. Tabla Kanban que facilita GitHub.

La última etapa, en la que he estado trabajando durante el mes de mayo y en la que me encuentro en el momento de redactar esta sección, es la de redacción de toda la parte de la

memoria que prosigue a las fases de requisitos y diseño. Esta etapa finalizará con la conclusión de este trabajo y con la entrega de este.

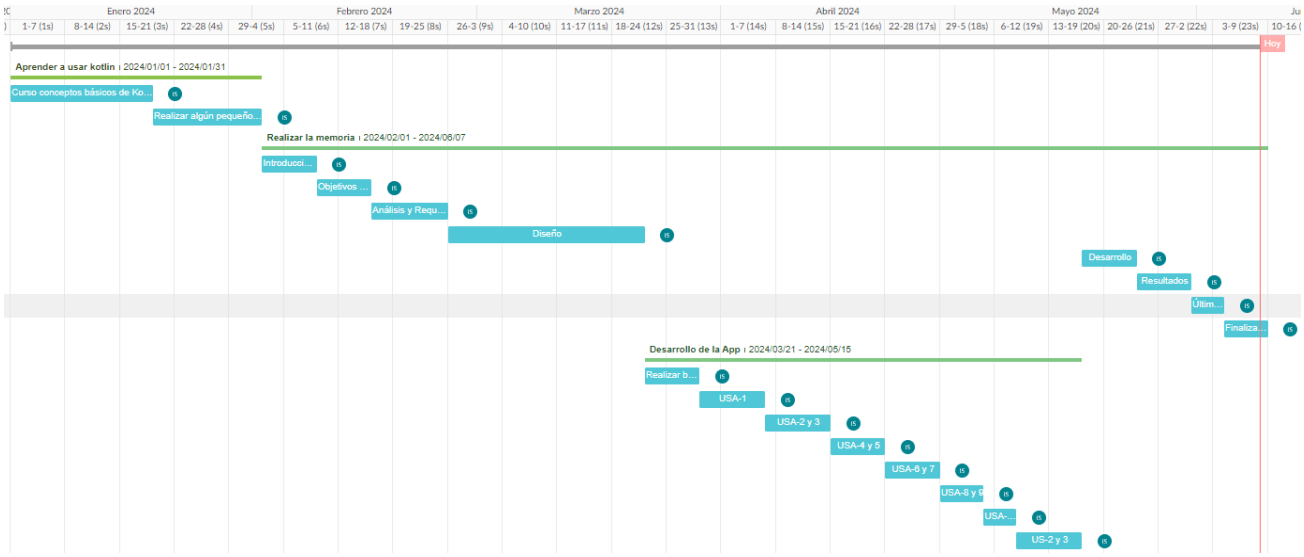


Figura 14. Diagrama de Gantt con la temporización final.

3.6. Aspectos Legales y Éticos

Uno de los aspectos más importantes a tener en cuenta al desarrollar una aplicación que gestiona comedores escolares, es decir, que gestiona datos personales de menores de edad, es la legislación y la ética que hay detrás. Estos requisitos son cruciales no solo para el cumplimiento normativo, sino también para asegurar la confianza y la seguridad entre los usuarios, que en este caso incluyen menores de edad, educadores y personal administrativo.

En la elaboración de mi aplicación, es fundamental tener en cuenta las directrices y pautas esenciales establecidas tanto por el Reglamento General de Protección de Datos (RGPD) como por la normativa española, en particular la Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD).

Podemos encontrar toda la información al respecto en la web de la Generalitat (Autoritat Catalana de Protección de Datos, 2018). A continuación, expongo algunos puntos clave y consideraciones que tendré que tener en cuenta:

- **Consentimiento y Protección de Datos de Menores**: Dado que la aplicación gestionará datos sensibles de menores, como alergias e intolerancias alimentarias, es necesario obtener el consentimiento explícito y verificable de los padres o tutores legales antes de cualquier procesamiento de datos personales.

- **Categorías Especiales de Datos**: Los datos sobre salud, como las alergias o intolerancias alimentarias, están clasificados como categorías especiales de datos

bajo el RGPD. Esto significa que su tratamiento requiere precauciones y medidas de seguridad adicionales, así como una justificación legal clara para su procesamiento.

- **Medidas de Seguridad y Gestión de Datos**: Implementar controles de acceso robustos, cifrado de datos y otras medidas de seguridad técnicas son esenciales para proteger los datos contra el acceso no autorizado o la pérdida de datos. Además, las Evaluaciones de Impacto sobre la Protección de Datos (EIPD) son cruciales cuando se manejan datos sensibles, ayudando a identificar y mitigar riesgos potenciales.

- **Transparencia y Derechos de Acceso**: Es importante asegurar que los usuarios de la aplicación (incluidos los padres y tutores) puedan acceder fácilmente a la información sobre cómo se manejan sus datos personales, cómo pueden ejercer sus derechos de acceso, rectificación y borrado de datos, y cómo pueden retirar su consentimiento.

Al requerir, pues, la manipulación de datos sensibles, seguir de cerca estas pautas no solo asegura el cumplimiento con las leyes de protección de datos, sino que también fortalece la integridad y la confianza en mi aplicación.

Para garantizar el cumplimiento normativo y la protección ética de los datos sensibles en mi aplicación, además de implementar las medidas mencionadas, es fundamental realizar auditorías regulares de seguridad para identificar y corregir vulnerabilidades. Debería establecer protocolos claros para el manejo de incidentes de seguridad, asegurando una respuesta rápida y eficaz en caso de violaciones de datos. Además, en caso de establecer una empresa y contratar personal, deberé asegurarme de proporcionar formación continua a este sobre las mejores prácticas en protección de datos y privacidad. Por último, tendré que asegurarme de que la aplicación cumpla con las actualizaciones legales y tecnológicas vigentes en todo momento.

Por lo que a ética se refiere, es esencial considerar el potencial estrés o estigma que podría surgir para los alumnos cuyas alergias e intolerancias son gestionadas a través de la aplicación. Asegurarse de que la aplicación maneje esta información de manera discreta y respetuosa, fomentando un ambiente inclusivo y seguro para todos los estudiantes, es tan importante como cumplir con los aspectos legales.

4. Resultados

A continuación, muestro las pantallas de las principales funcionalidades implementadas en la aplicación. Estas se muestran en orden, según la navegación de la aplicación, empezando por la pantalla de inicio de sesión (figura 15). La siguiente pantalla es la principal para los usuarios administradores, desde la que tienen acceso a todas las funcionalidades (figura 16) y el menú lateral de la barra superior, accesible desde todas las pantallas con la misma finalidad (figura 17). Las pantallas que vemos después de estas se corresponden a cada una de las funcionalidades disponibles, empezando por la gestión de usuarios (figuras 18, 19 y 20), y siguiendo por la gestión de las clases (figura 21), la de alumnos (figuras 22, 23 y 24), registro de asistencia (figuras 25 y 26) y la de consulta y generación de informes (figura 27 y 28). Estas pantallas siguen en gran medida el diseño planteado en los mockups, a excepción de algún pequeño cambio como en el caso de la pantalla de gestión de alumnos y registro de asistencia que implementan la Use Case US-3 (Consultar Alergias e Intolerancias de Alumnos) mediante una barra con cuatro botones para filtrar por la severidad de las alergias. De esta forma, se puede consultar los alumnos que tienen determinadas alergias y de los cuales hay que tener un control especial en las comidas de forma rápida, intuitiva y sencilla. La pantalla de consulta y generación de informes (USA-6) también tiene una pequeña modificación, ya que se ha añadido una barra con tres botones los cuales generan informes por día, pudiendo seleccionarlo mediante el calendario, mes y te dan la opción de descargar en PDF o Excel.

Figura 15. Inicio Sesión.

Figura 16. Pantalla principal admin.

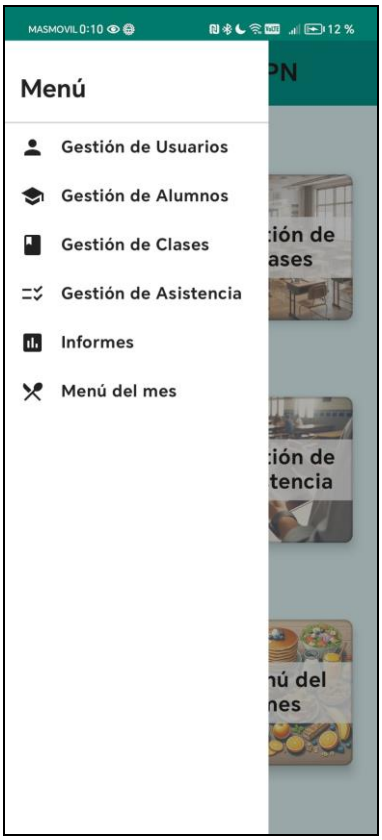


Figura 17. Menú lateral.

Figura 18. Gestión usuarios.

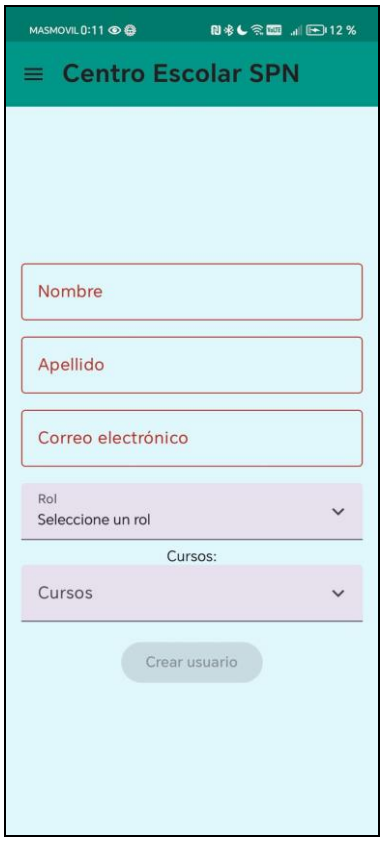


Figura 19. Pantalla creación usuarios.

Figura 20. Pantalla modificar/ eliminar usuario.

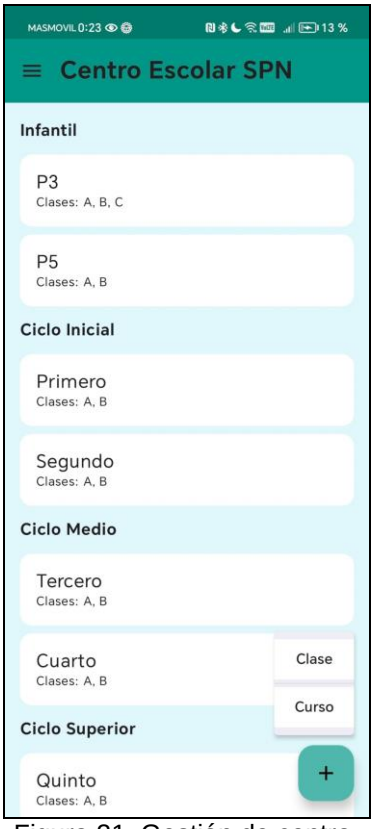


Figura 21. Gestión de centro.

Figura 22. Gestión alumnos.

Figura 23. Pantalla registrar alumno.

Figura 24. Pantalla modificar/ Eliminar alumno.

Figura 25. Pantalla asistencia.

Figura 26. Tarjeta información alumno.

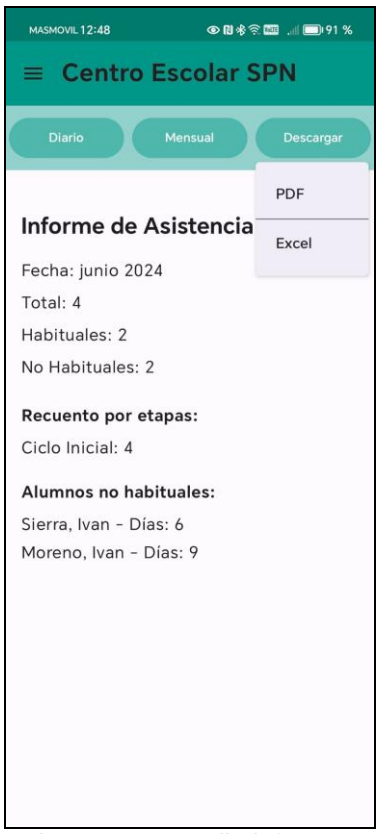


Figura 27. Pantalla informes.

Figura 28. Selección de día para el informe.

5. Impacto Medible

Para evaluar el impacto de la aplicación desarrollada, he definido una serie de métricas que me permitirán medir su efectividad y aceptación entre los usuarios. Algunas de las métricas que se utilizarán para medir el impacto una vez que la aplicación esté en funcionamiento son:

- Reducción del Tiempo de Gestión: Medir el tiempo promedio requerido para registrar la asistencia de los alumnos, gestionar menús y reportar incidencias antes y después de la implementación de la aplicación. El objetivo es reducir este tiempo al menos un 50%.

- Satisfacción del Usuario: Realizar encuestas de satisfacción a los diferentes usuarios de la aplicación (administradores, monitores, padres) para evaluar su experiencia con la aplicación. La meta es alcanzar una puntuación de satisfacción del 85% o superior.

- Número de Incidencias Reportadas: Monitorear el número de incidencias relacionadas con la gestión del comedor escolar reportadas antes y después de la implementación de la aplicación. Se espera una reducción notable en la cantidad de incidencias reportadas.

- Adopción y Uso de la Aplicación: Medir el porcentaje de usuarios que utilizan activamente la aplicación diariamente. El objetivo es lograr que al menos el 90% del personal relevante y padres utilicen la aplicación regularmente.

Estas métricas me proporcionarán una visión clara del impacto de la aplicación y ayudarán a identificar áreas de mejora continua, asegurando que la aplicación no solo cumple con sus objetivos iniciales, sino que también evoluciona para satisfacer mejor las necesidades de sus usuarios.

6. Planificación de costes

Cuando llevas a cabo un proyecto el cual quieres comercializar y del cual sacar un beneficio económico, como es el caso del desarrollo de un servicio ofrecido a través de una aplicación de software, es importante llevar a cabo una planificación de los costes que esto supone. A continuación, voy a desglosar la estimación de los costes implicados no solo en el desarrollo de la aplicación, si no también en la comercialización de esta:

- **Costes de desarrollo y mantenimiento**:

Concepto	Tiempo Estimado (Horas)	Coste por Hora (€)	Total (€)	Mensual (€)
Desarrolladores[3]	200	14,62	2924	243,67
Diseñadores	75	10,63	797,25	66,44
Servidores	-	-	600	50
Mantenimiento	200	14,62	2924	243,67
Coste total	-	-	7245,25	603,77

Para el coste de los desarrolladores estoy teniendo en cuenta la integración de las nuevas funcionalidades. El tiempo invertido en la aplicación, tal y como está ahora, estimo ha sido de alrededor de las 100 h, sumadas unas 50 h de formación. No obstante, un desarrollador experimentado habría invertido no más de 50 h, y teniendo en cuenta que es una inversión por mi parte y que forma parte del mi trabajo de fin de grado no lo tendré en cuenta.

Para el servidor, al no requerir mucha capacidad de almacenamiento por el tipo de datos que se gestionan, empezaría con un plan básico, aunque quizás, en caso de superar las expectativas de éxito iniciales, tenga que contratar un plan superior al cabo de unos meses.

Por lo que se refiere a licencias de software, por ahora no preveo que tenga que adquirir ninguna, por lo que no lo tendré en cuenta en el gasto.

[3] Tanto el salario por hora del desarrollador como el del diseñador, es un valor promedio de lo que cobran en España (Talent, 2024).

- **Costes de formación y soporte al cliente**:

Concepto	Tiempo Estimado (Horas)	Coste por Hora (€)	Total (€)	Mensual (€)
Formación al cliente (base)	5	10	50	4,17
Soporte Técnico (base)	20	14,62	292,4	23,37
Coste total	-	-	342,4	28,54

- **Costes legales y de seguridad**:

Concepto	Tiempo Estimado (Horas)	Coste por Hora (€)	Total (€)	Mensual (€)
Auditorías de Seguridad[4]	-	-	1500	125
Coste total	-	-	1500	125

Esto hace un coste total base de **757,31 €** mensuales, lo que al año suma un importe total de **9.087,72 €**. Teniendo en cuenta que el salario mensual de un monitor escolar es de unos 400€, y que mi aplicación, además de reducir horas de trabajo administrativo y minimizar errores en los cobros, también reducirá la carga laboral de los trabajadores de comedor, pudiendo llegar a prescindir de un trabajador y, por tanto, suponer un ahorro para la empresa de alrededor de los 400 € mensuales, creo que el precio por el servicio que debo cobrar por centro educativo ha de rondar entre los 100 € y 200 €.

El objetivo es arrancar con, como mínimo, 10 centros educativos que hayan contratado el servicio, y creo que es razonable pensar que podemos ir incorporando 3 centros por trimestre. Por centro educativo, tendremos que sumar 5 h de formación al cliente y 20 h de soporte técnico al año. Esto supone que, en el primer trimestre, tendremos unos gastos de 1042,64 € y en caso de cobrar 100 € mensuales, tendremos unas pérdidas de unos 42 € mensuales (128 € en total), y a partir del segundo trimestre empezaremos a obtener beneficios. En caso de cobrar 150 € o 200 € obtendremos beneficios desde el inicio.

[4] El gasto de la auditoria es una estimación, sacada de la web Profesional Hosting (ProfesionalHosting, 2024)

7. Trabajo futuro

Como ya he ido mencionando a lo largo de la memoria, la gestión de los centros escolares y del servicio de comedor, tiene un margen de mejora considerable, sobre todo en cuanto a digitalización se refiere. Aunque me hubiese gustado poder haber ido más allá en el desarrollo para este proyecto, la complejidad y la gran cantidad de funcionalidades que podrían añadir para dar mucho más valor a mi aplicación, ha hecho que tenga que decantarme por las que he considerado principales para poder tener una aplicación básica con la que poder empezar a buscar clientes. No obstante, soy consciente de que con esto no basta y antes de sacar la aplicación al mercado tendré que desarrollar algunas funcionalidades más que pueden ser esenciales para el éxito de la aplicación, así como ir añadiendo otras en actualizaciones futuras una vez ya haya empezado a comercializar el servicio. Así pues, soy consciente de que todavía queda mucho trabajo por hacer y de que esto ha sido solo el principio de lo que puede ser un gran producto que revolucione la gestión de los centros escolares. Algunas de las funcionalidades y tareas que quiero implementar y realizar son:

- **Mejorar el diseño y la navegación de la aplicación**: aunque estoy bastante satisfecho con el resultado final, para mejorar y aumentar las probabilidades de éxito del producto creo que será necesario replantear algunos aspectos del diseño para que el aspecto de la aplicación sea mucho más atractivo e intuitivo para el usuario. Soy consciente de que mis capacidades en ese sentido son limitadas y para ello tendré que contratar a un profesional para que se encargue de esto.

- **Sistema de pago integrado**: añadir un sistema de pago pensado tanto para mensualidades como para días esporádicos. Para ello será necesario crear perfiles de usuario para los alumnos.

- **Perfiles de usuario para alumnos (familias)**: crear un perfil dedicado a los alumnos a través del cual las familias puedan realizar diversas funciones, entre ellas modificar la información del alumno, realizar pagos, consultar información relevante sobre el comedor escolar, etc.

- **Perfiles de usuario para los trabajadores de cocina**: para mejorar la comunicación y la gestión de la información para los miembros de cocina (escuelas que no trabajen con catering), tanto disponer de forma rápida de la información sobre alergias de los alumnos, poder recibir notificaciones de especificaciones puntuales de las comidas de los alumnos (dietas, por ejemplo) o incluso poder realizar la gestión del producto alimentario a través de la app.

Además de las funcionalidades explicadas referentes al comedor escolar, me gustaría poder integrar otras relacionadas con la gestión del centro, como el control de asistencia de los alumnos en las aulas, cosa que facilitaría el conocimiento temprano de los asistentes al comedor y mejoraría el cálculo de las cantidades de comida a elaborar, por ejemplo.

8. Conclusiones

Este proyecto me ha supuesto una oportunidad de aplicar los conocimientos adquiridos durante la carrera para dar solución a un problema encontrado en mi trabajo, cosa que fue uno de los motivos por los que me decidí a estudiar ingeniería informática: poder aplicar el conocimiento adquirido en mi día a día y llegar a montar mi propio negocio. Es por esto que no encuentro mejor forma de poner punto y final a los cuatro años de carrera.

Como ya he ido mencionando a lo largo del trabajo, la aplicación tiene mucho que ofrecer y muchas funcionalidades que pueden darle valor y hacerla una opción competitiva en el mercado. Llegados a este punto, he conseguido desarrollar una aplicación básica pero funcional, cumpliendo con la mayoría de los objetivos planteados al inicio del proyecto y a partir de la cual podré seguir desarrollando nuevas funcionalidades. Además, con ella puedo a empezar a buscar clientes de cara al próximo curso académico, que era uno de mis objetivos personales al proponerme este reto como trabajo de fin de grado.

Aunque algunos de los objetivos, como el de mejorar la seguridad en la gestión de información de los usuarios y la comunicación, no se hayan conseguido como cabría esperar, puedo concluir que la realización de este trabajo ha sido una experiencia satisfactoria.

Esto solo ha sido el inicio de lo que espero sea un proyecto que siga evolucionando y en el que seguir trabajando hasta conseguir comercializar la aplicación. Confío en que todo el esfuerzo realizado y por realizar, valdrá la pena y poder ofrecer un servicio de calidad a los futuros usuarios de la aplicación.

Bibliografía

Autoritat Catalana de Protección de Datos. (2018). *Pautas de protección de datos para los centros educativos*. Obtenido de https://apdcat.gencat.cat/web/.content/04-actualitat/menors-i-joves/documents/GUIA-PAUTAS-DE-PROTECCION-DE-DATOS-PARA-CENTROS-EDUCATIVOS.pdf

Casa Intur. (2021). *Casa Intur*. Obtenido de https://casaintur.com/la-aplicacion-mas-completa-para-comedores-escolares/

ColeChef. (2024). *ColeChef*. Obtenido de https://colechef.com/

Comisión Nacional de los Mercados y la Competencia. (3 de Noviembre de 2023). *CNMC*. Obtenido de https://www.cnmc.es/prensa/panel-hogares-usos-internet-20231103#:~:text=Un%2078%2C8%20%25%20de%20los,los%20servicios%20OTT%20en%20Espa%C3%B1a.

Consorci d'Educació de Barcelona. (2022). *Memoria_consorci_2021_2022_web.* Obtenido de https://www.edubcn.cat/rcs_gene/extra/01_documents_de_referencia/Memoria_consorci_2021_2022_web.pdf

Django Software Foundation. (2024). *Django*. Obtenido de https://docs.djangoproject.com/es/5.0/

DoneTonic. (2022). Obtenido de https://donetonic.com/es/metodologia-waterfall-vs-metodologia-agile/

Google for Developres. (s.f.). *Developers*. Obtenido de https://developer.android.com/courses?hl=es-419

JetBrains. (s.f.). *Ktor*. Obtenido de https://ktor.io/

Kotzilla. (2024). *Koin*. Obtenido de https://insert-koin.io/

LOAD Servicios Informaticos. (2024). *Iara Catering*. Obtenido de https://iaracatering.es/comedores/

MapalOS. (s.f.). *MapalOS*. Obtenido de https://mapal-os.com/es/soluciones/easilys-collective

NoProblemCooking. (s.f.). *NoProblemCooking*. Obtenido de https://noproblem.es/

PlantText . (s.f.). Obtenido de https://www.planttext.com/

ProfesionalHosting. (2024). *ProfesionalHosting*. Obtenido de https://www.profesionalhosting.com/auditoria-de-seguridad/

Robert C. Martin. (13 de agosto de 2012). *The Clean Code Blog*. Obtenido de https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Roca Gonzalez. (s.f.). *Roca Gonzalez*. Obtenido de https://www.rocagonzalez.com/es/app-comedores-escolares/

Talent. (2024). *Talent*. Obtenido de https://es.talent.com/salary?job=programador#:~:text=%C2%BFCu%C3%A1nto%20gana%20un%20Programador%20en%20Espa%C3%B1a%3F&text=El%20salario%20programador%20promedio%20en,hasta%20%E2%82%AC%2039.934%20al%20a%C3%B1o.

Anexos

Cuestionario de Google

https://docs.google.com/forms/d/1vC2OvJki2w7TZ1tlPhM57SEq6KTcodF2uacYqoZUqfk/edit

Nombre y Logo

Al elegir el nombre para mi aplicación, decidí combinar elementos que reflejaran tanto su función tecnológica como su finalidad. El resultado fue "Appat", un nombre que integra "App", refiriéndose claramente a su naturaleza de aplicación, y "Apat", que en catalán significa comida. Esta combinación no solo captura perfectamente la esencia de la aplicación como una herramienta de gestión de comedores escolares, sino que también ofrece ventajas adicionales:

- **Posicionamiento en búsquedas**: Iniciar con 'A' coloca a "Appat" en una posición ventajosa en directorios y búsquedas, donde los nombres al principio de la lista suelen recibir más atención.
- **Relevancia y fácil de recordar**: El nombre es corto, fácil de recordar y describe de manera efectiva su función, lo que facilita que se quede en la mente de los usuarios potenciales.
- **Conexión cultural**: El uso del término "Apat", refuerza la conexión con los usuarios de regiones de habla catalana, región donde nace la aplicación y donde tengo intención de empezar a ofrecer el servicio, añadiendo un plus de familiaridad y relevancia cultural.

Considero que este nombre no solo contribuye a darle una identidad fuerte a la aplicación, sino que también me será de ayuda en su marketing y aceptación por parte del público objetivo, por lo que creo que es una muy buena elección.

Para el logo, he optado por un diseño que incorpora elementos de la cocina y la tecnología, reflejando el propósito de la aplicación. En la parte central encontramos un plato con un diseño que simula el circuito de un microchip (un toque tecnológico) con un documento en el centro, simbolizando en conjunto la gestión digital de datos. Está rodeado por un tenedor y un cuchillo, remarcando su vinculación con el momento de la comida. En la parte central inferior destaca el nombre de la aplicación.

Figura 29. Logotipo de la aplicación.