UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Optimización de la Higiene y Productividad en el Sector de la Restauración: Implementación de la Aplicación CleanScan

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Sáez Rodríguez, Miguel Ángel

Tutor/a: Escobar Román, Santiago

CURSO ACADÉMICO: 2023/2024

Resumen

Este trabajo de final de grado tiene como objetivo el diseño, la implementación y el despliegue de una aplicación destinada a la organización de los turnos de limpieza en el sector de la restauración. Para lograr un desarrollo satisfactorio de la aplicación, se ha abordado minuciosamente cada fase del proceso de creación de software, utilizando tecnologías modernas, como Flutter o Figma, para desarrollar tanto el diseño como la implementación de la aplicación.

Las conclusiones resaltan las fases del proyecto que han sido completadas con éxito y aquellas que podrían mejorarse en futuros desarrollos de aplicaciones similares. Este trabajo también contribuye de manera significativa a algunos de los Objetivos de Desarrollo Sostenible, ya que CleanScan pone de manifiesto la importancia de optimizar los recursos en el sector de la limpieza.

Palabras clave: Implementación, Flutter, Figma, Android, Firebase.

Abstract

This final degree project aims to design, implement, and deploy an application for organizing cleaning shifts in the restaurant industry. To achieve a successful application development, every phase of the software creation process has been meticulously addressed, using modern technologies such as Flutter and Figma to develop both the design and implementation of the application.

The conclusions highlight the phases of the project that have been successfully completed and those that could be improved in future developments of similar applications. This work also significantly contributes to some of the Sustainable Development Goals, as CleanScan emphasizes the importance of optimizing resources in the cleaning sector.

Keywords : Implementation, Flutter, Figma, Android, Firebase.

Resum

Aquest treball de final de grau té com a objectiu el disseny, la implementació i el desplegament d'una aplicació destinada a l'organització dels torns de neteja en el sector de la restauració. Per aconseguir un desenvolupament satisfactori de l'aplicació, s'ha abordat minuciosament cada fase del procés de creació de programari, utilitzant tecnologies modernes per desenvolupar tant el disseny com la implementació de l'aplicació.

Les conclusions ressalten les fases del projecte que han estat completades amb èxit i aquelles que podrien millorar-se en futurs desenvolupaments d'aplicacions similars. Aquest treball també contribueix de manera significativa a alguns dels Objectius de Desenvolupament Sostenible, ja que CleanScan posa de manifest la importància d'optimitzar els recursos en el sector de la neteja.

Palabres clau: Implementació, Flutter, Figma, Android, Firebase.

Índice de contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

1.1 Introducción

Mantener un entorno higiénico y seguro es fundamental en el sector de la restauración, pero también es una tarea compleja y exigente. La falta de una limpieza adecuada puede costar mucho más que la reputación del negocio. Según el Instituto de Investigación de la Industria de Restaurantes, el 78% de los clientes considera la higiene y la limpieza como uno de los factores más importantes al elegir un restaurante.

Actualmente, es común encontrar hojas pegadas en las puertas o paredes de los aseos para gestionar y comprobar los turnos de limpieza. Sin embargo, estos métodos son propensos a fallos y muchas veces no solucionan los desperfectos en materiales o utensilios debido a una mala gestión o comunicación entre el empleado y el empleador. Estos problemas a menudo resultan de una ineficaz gestión y comunicación.

Aunque es un problema fácil de solucionar, actualmente no existen herramientas que simplifiquen y modernicen estos procesos. Aquí es donde surge la idea de CleanScan, una herramienta creada con el propósito de registrar, controlar y gestionar todas las tareas de limpieza de manera oportuna y digital.

CleanScan consta de dos componentes principales: un panel de control y una aplicación disponible para iOS y Android. El panel, dirigido al administrador del local, permite gestionar los turnos de limpieza, los espacios a limpiar, las limpiezas realizadas y los empleados. La aplicación, diseñada tanto para administradores como para empleados, facilita el registro de limpiezas de forma visual y escrita, además de permitir la gestión de turnos. En este TFG se desarrollará únicamente la aplicación para dispositivos móviles.

En los siguientes apartados se describirán las etapas del proceso de creación del software, así como la arquitectura del proyecto y las tecnologías utilizadas. La primera solución, el panel, está desarrollada con el framework Angular, que utiliza el lenguaje TypeScript, un superconjunto de JavaScript. La segunda solución, la aplicación, está

desarrollada con el SDK de Flutter, que emplea el lenguaje multiplataforma Dart. Ambas soluciones se comunican directamente con la base de datos ubicada en Firebase, una plataforma que permite alojar datos con un enfoque no relacional, además de facilitar la monetización y el análisis de la aplicación.

1.2 Motivación

Lamentablemente, es muy común ir a restaurantes, bares, cines y otros locales de ocio y encontrar un estado insalubre en sus servicios. Esta situación, aunque riesgosa para la salud, está en algunos casos incluso normalizada. A medida que la salud se vuelve una prioridad creciente para la sociedad, la presencia de virus y bacterias como el E.coli y el norovirus en estos entornos se convierte en una preocupación significativa. Estos patógenos pueden infectar a los usuarios de estos servicios, pero su probabilidad de propagación puede reducirse considerablemente con una buena higiene personal y un riguroso régimen de higienización por parte del establecimiento.

Por ello, es fundamental contar con una herramienta que permita a los propietarios y gerentes de establecimientos controlar de forma eficiente la higienización de sus locales desde su lugar de trabajo. CleanScan surge como una solución innovadora, permitiendo a los administradores gestionar y supervisar las tareas de limpieza de manera efectiva, asegurando un entorno seguro y limpio para los clientes. Esta gestión no solo mejora la salud pública, sino que también incrementa la satisfacción y confianza de los usuarios, quienes pueden disfrutar de las ventajas de una higiene adecuada y sus comodidad en los lugares que frecuentan.

1.3 Objetivos

Este Trabajo de Fin de Grado se centra en dos objetivos principales. Primero, consiste en desarrollar una herramienta que facilite la gestión eficiente de la higienización y limpieza en el sector de la restauración. Segundo, busca detallar y explicar los procesos involucrados en el desarrollo de aplicaciones, así como las funcionalidades implementadas. Los subobjetivos específicos son los siguientes:

- Conceptualización de la herramienta: Definir claramente el propósito, funcionalidades principales y beneficios de la herramienta de gestión de higienización.

- Planificación y definición de requisitos: Establecer una planificación detallada que incluya la identificación de requisitos funcionales y no funcionales necesarios para la implementación efectiva de la herramienta.
- Creación de diseños: Desarrollar interfaces de usuario y diseños visuales que sean intuitivos y eficientes para los usuarios finales.
- Desarrollo de la aplicación: Implementar las funcionalidades definidas en la fase de diseño utilizando las tecnologías adecuadas y siguiendo las mejores prácticas de desarrollo de software.
- Pruebas (Testing): Realizar pruebas exhaustivas para verificar el funcionamiento correcto de la aplicación, asegurando la calidad y fiabilidad del producto final.

Estos subobjetivos se combinan para proporcionar una comprensión completa del ciclo de vida del desarrollo de software, desde la concepción de la idea hasta la entrega de una solución funcional y efectiva para mejorar los estándares de limpieza en establecimientos de restauración.

2. Estado del arte

Actualmente, no existe una herramienta específica para gestionar los servicios de limpieza en el contexto de la restauración. La aplicación CleanScan está compuesta por una serie de funcionalidades que, en términos generales, se comparará con la posible competencia:

- Sistema de creación de turnos de limpieza
- Sistema de edición de los turnos de limpieza
- Sistema de creación de limpiezas
- Sistema de edición de las limpiezas
- Sistema de códigos QR para fichar y registrar la limpieza de un espacio
- Sistema de creación de espacios de limpieza
- Sistema de edición de los espacios de limpieza
- Panel para la administración de todo el ecosistema
- Aplicación móvil para el registro de información

Herramientas	ventajas	desventajas
Sesame	Potente con una gran funcionalidad para la gestión de turnos	No está enfocada en la restauración y no permite llevar un registro de las tareas disponibles y sus resultados. Es una herramienta exclusiva para PC.
Plaky	Muy versátil para gestionar tareas	Es más compleja de lo necesario, no permite adjuntar imágenes de los resultados y no está enfocada en los requisitos propios de la restauración. Es una herramienta exclusiva para PC.

connecteam	Muy adaptada y preparada para los servicios de limpieza, posee una app móvil	Tiene una funcionalidad muy alta, lo que hace que su curva de aprendizaje y uso sea compleja.

Tabla 1. Comparación de herramientas con funcionalidades similares a CleanScan

El análisis de la competencia revela que no existe una plataforma que reúna todas las características de CleanScan de manera sencilla e intuitiva. Existen aplicaciones para gestionar la limpieza, pero no poseen la sencillez que necesita un usuario común. Por otra parte, hay aplicaciones para gestionar turnos, pero no poseen una infraestructura para registrar las tareas. Finalmente, también encontramos aplicaciones para gestionar tareas, pero sin la capacidad de gestionar turnos. Por estas razones, CleanScan se convierte en la mejor solución actual para gestionar la limpieza en el sector de la restauración.

3. Tecnologías utilizadas

3. Motivación de las tecnologías utilizadas

Para la implementación de la aplicación CleanScan, se han seleccionado diversas tecnologías según ciertos criterios. Estos criterios incluyen que sean económicas, tengan una curva de aprendizaje rápida, posean la potencia necesaria para el proyecto y sean compatibles con múltiples plataformas, minimizando las desventajas en comparación con el desarrollo en lenguajes nativos.

3.1 Visual Studio Code

Visual Studio Code es un potente editor de texto gratuito desarrollado por Microsoft. Permite el desarrollo de software en una gran variedad de lenguajes de programación, como Python, Java, C++, JavaScript, entre muchos otros. Además, facilita en gran medida el desarrollo en equipo, ya que su interfaz permite ver quién escribió un trozo de código y cuándo se realizó el commit. Sin embargo, ninguna de estas funcionalidades es el motivo por el que se ha utilizado este editor de texto en el desarrollo de la aplicación. La función de Visual Studio Code en el proyecto ha sido la de gestionar las versiones con Git, ya que permite, mediante una interfaz gráfica, seleccionar los archivos individualmente para realizar los commits, lo que facilita en gran medida la organización de los mismos.

Figura 1. Logo de visual studio

3.2 Git

Git es un sistema de control de versiones que permite administrar los diferentes estados de un proyecto, facilitando la posibilidad de volver a un estado anterior o combinar distintos estados. Además, al utilizar un host combinado con Git, como GitHub o Bitbucket, es posible gestionar las diferentes versiones de manera remota. En este proyecto, se ha utilizado Bitbucket para alojar el repositorio de Git en la nube.

Figura 2. Logo de git

Figura 3. Logo de bitbucket

3.3 Android Studio

Android Studio es un entorno de desarrollo integrado (IDE) creado por Google para el desarrollo de aplicaciones Android, aunque también permite desarrollar aplicaciones para iOS, web y Windows. Este ha sido el IDE seleccionado para el desarrollo de CleanScan por varios motivos, siendo el principal su alta compatibilidad con Flutter. La combinación de Android Studio y Flutter permite compilar y ejecutar la aplicación tanto para iOS como para Android a través de la interfaz del IDE. Además, facilita la utilización de varios emuladores de Android, la gestión de paquetes descargados y el uso de recursos de la aplicación.

Figura 4. Logo de android studio

3.4 Dart

Dart es un lenguaje de programación multiplataforma creado por Google que permite, mediante un único lenguaje, desarrollar aplicaciones en lenguaje nativo para dispositivos móviles, web y de escritorio. Dart está optimizado para su utilización en el lado del cliente, es decir, para ser ejecutado en el dispositivo del usuario. Desde el punto de vista del desarrollador, es un lenguaje muy cómodo ya que permite la recarga de estado en menos de un segundo, reflejando así los cambios de inmediato sin tener que volver a compilar. Debido a estas características, además de ser gratuito y tener una comunidad muy grande, ha sido elegido como el lenguaje de programación para el proyecto.

Figura 5. Logo de dart

3.5 Flutter

Flutter es un framework de código abierto desarrollado por Google, al igual que Dart. Este framework facilita el desarrollo del front-end y permite, mediante la utilización de un tipo de componente llamado Widget, crear interfaces de usuario de manera fácil, eficiente y sencilla. Debido a su gran comunidad, existe una amplia cantidad de librerías creadas por otros usuarios, disponibles para descargar e implementar de manera sencilla. Al usar Dart como lenguaje, se pueden aprovechar sus grandes ventajas. Por todas estas facilidades y características previamente mencionadas, se ha utilizado este framework en el proyecto.

Figura 6. Logo de flutter

3.6 Firebase

Firebase es la plataforma de desarrollo de aplicaciones móviles de Google. Ofrece una amplia gama de funcionalidades, permitiendo, entre otras cosas, actuar como base de datos, almacenamiento de archivos y servir como backend con la capacidad de escalar según la demanda, proporcionando así una alta elasticidad. Utilizar Firebase en un proyecto permite reducir los costos en la codificación del backend, permitiendo destinar más recursos al desarrollo del front-end. Por todas estas ventajas, se ha decidido utilizar Firebase en el proyecto.

Figura 7. Logo de firebase

4. Análisis

El análisis del problema es uno de los puntos clave en el desarrollo de una aplicación. Una vez está planteado el problema, se hace un análisis, descomponiendo en pequeñas partes el proyecto, para tener una visión más clara y sencilla en el proceso de implementación.

4.1 Requisitos del sistema

4.1.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que prestará el sistema, en la forma en que reacciona a determinados insumos. Cuando hablamos de las entradas, no necesariamente hablamos sólo de las entradas de los usuarios. Pueden ser interacciones con otros sistemas, respuestas automáticas, procesos predefinidos [1].
Los requisitos funcionales de CleanScan son:

- **Requisitos de usuario(UR)**
  - **UR-R01:** Permitir a los usuarios iniciar sesión con un correo electrónico y una contraseña.
  - **UR-R02:** Recordar al usuario mediante una casilla de verificación.
  - **UR-R03:** Permitir la recuperación de contraseña mediante el correo electrónico.
  - **UR-R04:** Permitir a los usuarios cerrar sesión.
  - **UR-R05:** Permitir a los usuarios eliminar su cuenta.
  - **UR-R06:** Visualizar las limpiezas ejecutadas en un intervalo de tiempo especificado.
  - **UR-R07:** Escanear un código QR para crear una limpieza.
  - **UR-R08:** Visualizar los turnos asignados a un espacio.
  - **UR-R09:** Agregar turnos a un espacio.
  - **UR-R010 :** Editar las limpiezas.
  - **UR-R11:** Editar un turno.
- **Requisitos de Limpiezas(CR)**
  - **CR-R01:** Mostrar limpiezas en un intervalo de tiempo especificado.
  - **CR-R02:** Permitir la edición de limpiezas.

- **CR-R03:** Permitir la creación de una limpieza a partir de un escáner de código QR.
- **CR-R04:** Ejecutar una limpieza con fecha, hora, nombre, fotos y comentarios.
- **CR-R05:** Permitir la cancelación de una limpieza que está siendo creada.
- **CR-R06:** Permitir la eliminación de una limpieza.
- **CR-R07:** Permitir la visualización de imágenes de la limpieza desde la pantalla de visualización de limpiezas.
- **CR-R08:** Permitir la edición de una limpieza.

- **Requisitos de Turnos(SR)**
  - **SR-R01:** Mostrar los turnos asignados a los espacios.
  - **SR-R02:** Crear turnos con una fecha, un espacio y un horario específico.
  - **SR-R03:** Permitir la cancelación de la creación de un turno de limpieza.
  - **SR-R04:** Permitir la edición de un turno de limpieza.
  - **SR-R05:** Permitir la eliminación de un turno de limpieza.
  - **SR-R06:** Permitir la visualización de un turno de limpieza.

4.1.2 Requisitos no funcionales

Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad, usabilidad, escalabilidad, mantenibilidad, compatibilidad y fiabilidad. En palabras más sencillas, no hablan de “lo que” hace el sistema, sino de “cómo” lo hace. Alternativamente, definen restricciones del sistema tales como la capacidad de los dispositivos de entrada/salida y la representación de los datos utilizados en la interfaz del sistema[2].
Los requisitos no funcionales de CleanScan son:

- **Rendimiento**
  - RNF-R01: Los tiempos de carga de las pantallas deben ser inferiores a 2 segundos, exceptuando los procesos de autenticación y carga inicial de la aplicación, los cuales deben ser inferiores a 5 segundos.
- **Seguridad**
  - RNF-R02: Las contraseñas deben mostrarse cifradas desde el primer momento y almacenarse cifradas en la base de datos.

    - RNF-R03: La comunicación con el backend debe realizarse mediante el protocolo HTTPS para garantizar la seguridad de los datos en tránsito.
- **Usabilidad**
    - RNF-R04: La aplicación debe ser intuitiva, con textos claros y uniformes, y sin botones difíciles de alcanzar para asegurar una buena experiencia de usuario.
- **Escalabilidad**
    - RNF-R05: El backend debe ser capaz de escalar horizontalmente en función de la demanda de usuarios para mantener el rendimiento óptimo.
- **Mantenibilidad**
    - RNF-R06: El código debe ser modular para facilitar la reutilización de componentes y permitir modificaciones de manera sencilla y eficiente.
- **Compatibilidad**
    - RNF-R07: La aplicación debe ser compatible con versiones de Android superiores a 11.0 Red Velvet Cake y con versiones de iOS superiores a 14.0.
- **Disponibilidad**
    - RNF-R08: La aplicación debe tener una disponibilidad del 99.99%, siempre que esté actualizada a la última versión y tenga acceso a una conexión de internet estable.

4.1 Casos de uso

En este apartado se va a abordar los diferentes casos de uso de la aplicación. Un caso de uso representa la lista de tareas que los actores pueden realizar y está directamente relacionado con los requisitos del proceso de negocio. Los casos de uso son un reconocimiento de los requisitos que debe cumplir el proyecto[3]. En CleanScan se han encontrado los siguientes casos de uso:

CU1-Inicio de sesión

Referencia	1
Nombre	Inicio de sesión
Descripción	El usuario abre la aplicación y accede a la pantalla principal. Introduce el correo electrónico asociado a su cuenta y su contraseña. Una vez validados los datos de forma satisfactoria accede a la pantalla de limpiezas, en caso contrario, se muestra un mensaje de credenciales erróneas.
Actor	Usuario sin sesión activa
Relaciones	
Precondición	El usuario no debe tener recordar inicio de sesión activo

Tabla 2. Caso de uso de inicio de sesión

CU2- Recordar inicio de sesión

Referencia	2
Nombre	Recordar inicio de sesión
Descripción	El usuario abre la aplicación y accede a la pantalla principal. Introduce el correo electrónico asociado a su cuenta y su contraseña, seguidamente, marca la casilla “Recuérdame”. Una vez validados los datos de forma satisfactoria accede a la pantalla de limpiezas y el sistema almacena en memoria sus credenciales, en caso contrario, se muestra un mensaje de credenciales erróneas.
Actor	Usuario sin sesión activa
Relaciones	Extiende CU1-inicio de sesión
Precondición	

Tabla 3. Caso de uso de recordar inicio de sesión

CU3- Inicio de sesión automático

Referencia	3
Nombre	Inicio de sesión automático
Descripción	El usuario abre la aplicación y accede a la pantalla de limpiezas automáticamente, sin pasar previamente, por la pantalla de inicio de sesión.
Actor	Usuario sin sesión activa
Relaciones	Incluye CU1-inicio de sesión
Precondición	

Tabla 4. Caso de uso de inicio de sesión automático

CU4- Recuperar Contraseña

Referencia	4
Nombre	Recuperar contraseña
Descripción	El usuario accede a esta pantalla desde el texto “¿Has olvidado la contraseña?” de la pantalla de inicio de sesión. Una vez en la pantalla de recuperar la contraseña, introduce su correo electrónico y pulsa “Enviar”. El usuario recibe un correo electrónico con los pasos que debe seguir para recuperar su contraseña.
Actor	Usuario sin sesión activa
Relaciones	
Precondición	

Tabla 5. Caso de uso de Recuperar contraseña

CU5- Visualización de limpiezas

Referencia	5
Nombre	Visualización de limpiezas
Descripción	El usuario accede a la pantalla de limpiezas desde la pantalla de inicio de sesión o desde el menú desplegable lateral. Visualiza por defecto las limpiezas ejecutadas durante los últimos 7 días.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 6. Caso de uso de visualización de limpiezas

CU6-Visualización de limpiezas últimos siete días

Referencia	6
Nombre	Visualización de limpiezas últimos siete días
Descripción	El usuario accede a la pantalla de limpiezas desde la pantalla de inicio de sesión o desde el menú desplegable lateral. Selecciona desde el menú desplegable la opción “Últimos siete días” y visualiza las limpiezas ejecutadas durante los últimos siete dias.
Actor	Usuario con sesión activa
Relaciones	Generaliza a CU3-Visualización de limpiezas
Precondición	

Tabla 7. Caso de uso de visualización de limpiezas últimos siete días

CU7-Visualización de limpiezas último mes

Referencia	7
Nombre	Visualización de limpiezas último mes
Descripción	El usuario accede a la pantalla de limpiezas desde la pantalla de inicio de sesión o desde el menú desplegable lateral. Selecciona desde el menú desplegable la opción “Último mes” y visualiza las limpiezas ejecutadas durante el último mes.
Actor	Usuario con sesión activa
Relaciones	Generaliza a CU3-Visualización de limpiezas por defecto
Precondición	

Tabla 8. Caso de uso de visualización de limpiezas último mes

CU8-Visualización de limpiezas personalizada

Referencia	8
Nombre	Visualización de limpiezas personalizada
Descripción	El usuario accede a la pantalla de limpiezas desde la pantalla de inicio de sesión o desde el menú desplegable lateral. Selecciona desde el menú desplegable la opción “Personalizado”, seguidamente, se abre un selector de fecha. El usuario selecciona un periodo de tiempo y posteriormente visualiza las limpiezas ejecutadas durante el periodo de tiempo seleccionado.
Actor	Usuario con sesión activa
Relaciones	Generaliza a CU3- Visualización de limpiezas por defecto
Precondición	

Tabla 9. Caso de uso de visualización de limpiezas personalizada

CU9- Escanear código qr

Referencia	9
Nombre	Escanear código qr
Descripción	El usuario desde la pantalla de turnos o limpiezas selecciona el icono de qr ubicado en la parte superior derecha de la pantalla y accede a la pantalla de escanear código qr. Mediante la cámara escanea un código qr asociado a un espacio.
Actor	Usuario con sesión activa
Relaciones	
Precondición	El código qr escaneado debe estar asociado a un espacio

Tabla 10. Caso de uso de escanear código qr

CU10- Crear nueva limpieza

Referencia	10
Nombre	Crear nueva limpieza
Descripción	El usuario después de escanear un código qr asociado a un espacio, accede a la pantalla de nueva limpieza. Introduce la fecha y hora en la que se hizo la limpieza y su nombre. Posteriormente, pulsa el icono ubicado en la parte superior derecha de la pantalla y ejecuta la limpieza.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 11. Caso de uso de crear nueva limpieza

CU11- Crear nueva limpieza con fotos

Referencia	11
Nombre	Crear nueva limpieza con fotos
Descripción	El usuario después de escanear un código qr asociado a un espacio, accede a la pantalla de nueva limpieza. Introduce la fecha y hora en la que se hizo la limpieza y su nombre. Pulsa en el botón “hacer fotos”, a continuación, se abre la cámara y realiza algunas fotos del espacio. Posteriormente, pulsa el icono ubicado en la parte superior derecha de la pantalla y ejecuta la limpieza.
Actor	Usuario con sesión activa
Relaciones	Extiende a CU10-Crear nueva limpieza
Precondición	

Tabla 12. Caso de uso de crear nueva limpieza con fotos

CU12- Crear nueva limpieza con comentarios

Referencia	12
Nombre	Crear nueva limpieza con comentarios
Descripción	El usuario después de escanear un código qr asociado a un espacio, accede a la pantalla de nueva limpieza. Introduce la fecha y hora en la que se hizo la limpieza y su nombre. Finalmente, introduce algunos comentarios acerca de la limpieza y pulsa el icono ubicado en la parte superior derecha de la pantalla para ejecutar la limpieza.
Actor	Usuario con sesión activa
Relaciones	Extiende a CU10-Crear nueva limpieza
Precondición	El usuario debe introducir de forma obligatoria los campos fecha, hora y nombre para poder ejecutar la limpieza.

Tabla 13. Caso de uso de crear nueva limpieza con comentarios

CU13- Eliminar una limpieza

Referencia	13
Nombre	Eliminar una limpieza
Descripción	En la pantalla de limpiezas, el usuario pulsa sobre el icono de papelera de la limpieza que desea eliminar. A continuación, se abre una ventana modal para confirmar la acción.
Actor	Usuario con sesión activa
Relaciones	
Precondición	Debe haber alguna limpieza ejecutada en el periodo de tiempo seleccionado.

Tabla 14. Caso de uso eliminar una limpieza

CU14- Editar una limpieza

Referencia	14
Nombre	Editar una limpieza
Descripción	En la pantalla de limpiezas, el usuario pulsa sobre una limpieza. Seguidamente se abre la pantalla de editar limpieza. El usuario modifica los campos necesarios y guarda la limpieza.
Actor	Usuario con sesión activa
Relaciones	
Precondición	Debe haber alguna limpieza ejecutada en el periodo de tiempo seleccionado.

Tabla 15. Caso de uso de editar una limpieza

CU15- Visualizar imágenes de una limpieza

Referencia	15
Nombre	Visualizar imágenes de una limpieza
Descripción	En la pantalla de limpiezas, el usuario pulsa sobre el icono de galería de una limpieza. Acto seguido, se abre una pantalla modal y el usuario mediante gestos de deslizar visualiza todas las imágenes.
Actor	Usuario con sesión activa
Relaciones	
Precondición	Debe haber alguna limpieza ejecutada en el periodo de tiempo seleccionado. La limpieza debe tener imágenes asociadas.

Tabla 16. Caso de uso de visualizar imágenes de una limpieza

CU16- Visualización de turnos

Referencia	16
Nombre	Visualización de turnos
Descripción	El usuario accede a la pantalla de turnos desde el menú desplegable lateral pulsando “Turnos”. Seguidamente se abre la pantalla de turnos y visualiza los turnos de limpieza.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 17. Caso de uso de visualización de turnos

CU17- Agregar un turno de limpieza

Referencia	17
Nombre	Agregar un turno de limpieza
Descripción	El usuario desde la pantalla de turnos, pulsa el botón “Agregar turno”. Acto seguido, se abre la pantalla de Nuevos turnos. El usuario selecciona un espacio, después el sistema carga el horario del espacio seleccionado. El usuario selecciona una fecha o un periodo de tiempo que se asociará a ese turno e introduce los nombres en cada campo asociado a una hora. Finalmente, pulsa sobre el botón ubicado en la parte superior derecha de la pantalla y guarda el turno.
Actor	Usuario con sesión activa
Relaciones	
Precondición	No debe existir un turno asociado a la fecha seleccionada.

Tabla 18. Caso de uso de agregar un turno de limpieza

CU18- Visualizar un turno de limpieza

Referencia	18
Nombre	Visualizar un turno de limpieza
Descripción	En la pantalla de turnos, el usuario pulsa sobre un turno. A continuación, se abre la pantalla de ver turno y visualiza: el espacio, fecha y horario asociada al turno seleccionado.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 19. Caso de uso de visualizar un turno de limpieza

CU19- Editar turno de limpieza

Referencia	15
Nombre	Editar turno de limpieza
Descripción	Desde la pantalla de turnos, el usuario pulsa sobre el icono del lápiz para editar el turno deseado. Seguidamente, el usuario modifica algunos nombres y guarda el turno.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 20. Caso de uso de editar un turno de limpieza

CU20- Eliminar turno de limpieza

Referencia	20
Nombre	Eliminar turno de limpieza
Descripción	Desde la pantalla de turnos, el usuario pulsa sobre el icono de la papelera del turno que desea eliminar. Acto seguido, se abre una ventana modal para confirmar la acción.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 21. Caso de uso de visualizar un turno de limpieza

CU21- Cerrar Sesión

Referencia	21
Nombre	Cerrar sesión
Descripción	Desde el menú desplegable lateral, el usuario pulsa sobre el botón “Cerrar sesión”. A continuación, el sistema cierra la sesión del usuario, en caso de tener el inicio de sesión automático activo, el sistema elimina la información de inicio de sesión y navega a la pantalla de iniciar sesión.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 22. Caso de uso de cerrar Sesión

CU22- Eliminar cuenta

Referencia	22
Nombre	Eliminar cuenta
Descripción	Desde el menú desplegable lateral, el usuario pulsa sobre el botón “Eliminar cuenta”. A continuación, el sistema cierra la sesión del usuario y elimina su cuenta. Navega a la pantalla de iniciar sesión.
Actor	Usuario con sesión activa
Relaciones	
Precondición	

Tabla 23. Caso de uso de eliminar cuenta

En estos casos de uso se definen dos actores, el usuario sin sesión activa, que representa todas las acciones que un usuario puede realizar previa a iniciar sesión. El diagrama de dicho actor es el siguiente:

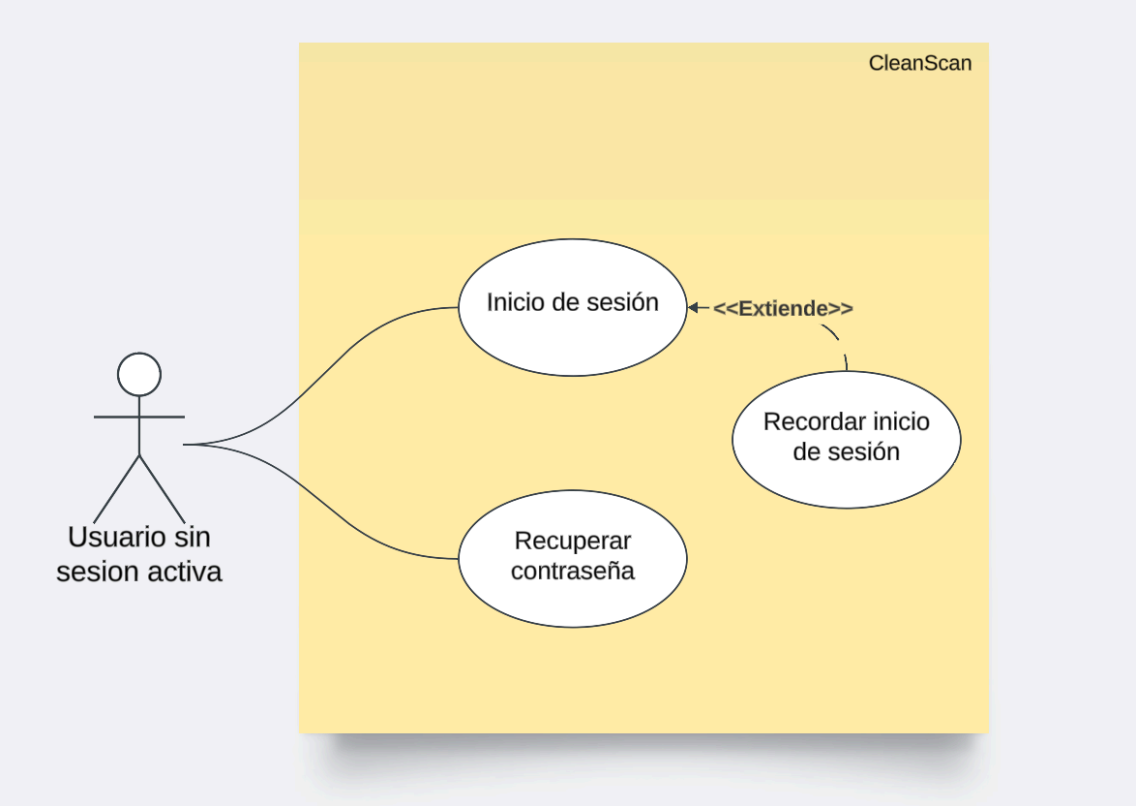


Figura 8. Diagrama de casos de uso de un usuario sin una sesión activa

Finalmente, el segundo actor es el usuario con sesión activa que representa todas las acciones que el usuario puede realizar una vez ha iniciado sesión de forma satisfactoria. El diagrama que representa a este actor se describe a continuación:

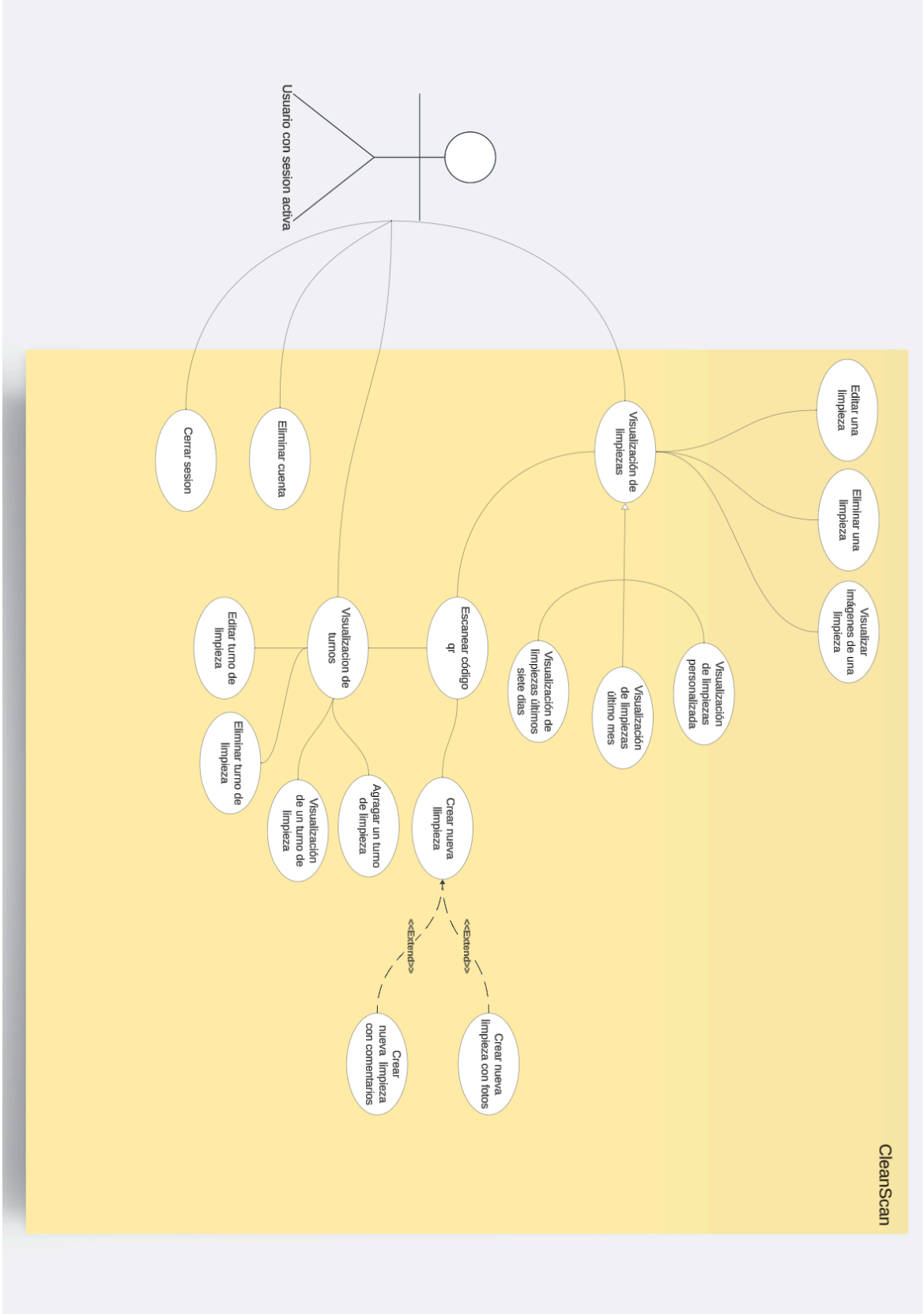


Figura 9. Diagrama de casos de uso de un usuario con una sesión activa

5. Diseño

Para el diseño de la lógica, es fundamental conocer el problema en profundidad, de manera que se pueda construir un modelo detallado enfocado en la solución. En este apartado, se diseñará un diagrama de clases UML (Lenguaje Unificado de Modelado), que representará las distintas entidades y sus relaciones. Gracias a este diseño, será posible implementar las entidades de manera más consistente y eficiente, optimizando las clases, los atributos y las relaciones.

5.1 Diseño de la lógica

Para el diseño de la lógica, es de vital importancia conocer el problema en profundidad, para así poder construir un modelo detallado enfocado en la solución. En este apartado, se diseñará un diagrama de clases UML (Lenguaje Unificado de Modelado), con las distintas entidades y sus relaciones. Gracias a este diseño, se podrá implementar de una manera más consistente y eficiente las entidades, optimizando las clases, los atributos y las relaciones.

En esta aplicación se han encontrado las siguientes entidades:

- **User**: La entidad User hace referencia al usuario. Está compuesta por los siguientes atributos:
    - **‘id’**: Clave primaria de la entidad User, de tipo String.
    - **‘corpId’**: Clave ajena de la entidad Corporation, de tipo String.
    - **‘email’**: De tipo String, hace referencia al correo electrónico del usuario.
    - **‘enabled’**:De tipo booleano, indica si un usuario tiene la cuenta activa.
    - **‘name’**: De tipo String, hace referencia al nombre del usuario.
    - **‘rol’**: De tipo int, hace referencia al rol del usuario dentro del sistema.

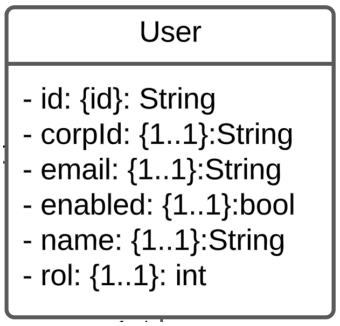


Figura 10. Definición de la entidad User en el diagrama UML

- **Cleaning**: La entidad Cleaning hace referencia a las limpiezas ejecutadas por un usuario. Está compuesta por los siguientes atributos:
  - **'Id'**: Clave primaria de la entidad Cleaning, de tipo String.
  - **'spaceId'**: Clave ajena de la entidad Space, de tipo String.
  - **'userId'**: Clave ajena de la entidad User, de tipo String.
  - **'comments'**: De tipo String, hace referencia a los comentarios que introduce el usuario al ejecutar una limpieza.
  - **'date'**: De tipo DateTime, hace referencia al día y la hora en que el usuario ejecuta la limpieza.
  - **'photos'**: De tipo lista de String, hace referencia a las fotos realizadas por el usuario cuando ejecuta la limpieza.

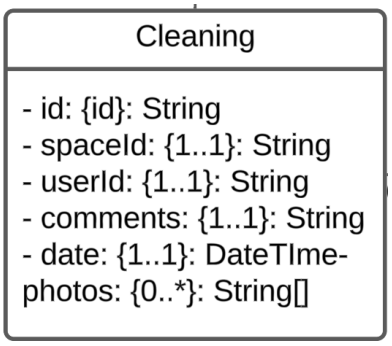


Figura 11. Definición de la entidad Cleaning en el diagrama UML

- **Corporation**: La entidad Corporation hace referencia a las diferentes empresas que están en el sistema. Está compuesta por los siguientes atributos:
  - **'id'**: Clave primaria de la entidad Corporation, de tipo String.
  - **'logo'**: De tipo String, hace referencia a la URL del logo de la empresa.
  - **'name'**: De tipo String, hace referencia al nombre de la empresa.

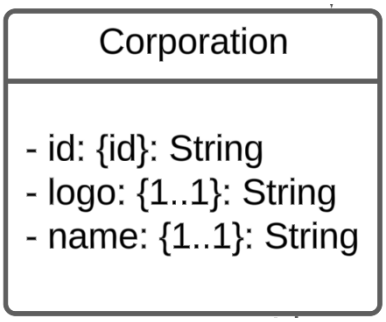


Figura 12. Definición de la entidad Corporation en el diagrama UML

- **Space**: La entidad Space hace referencia a los espacios que forman una empresa. Cada espacio tiene un horario fijo de limpieza (schedule). Está compuesta por los siguientes atributos:
  - **'id'**: Clave primaria de la entidad Space, de tipo String.
  - **'corpId'**: Clave ajena de la entidad Corporation, de tipo String.
  - **'name'**: De tipo String, hace referencia al nombre del espacio.
  - **'schedule'**: De tipo lista de String, hace referencia a las horas en las que se debe ejecutar una limpieza en ese espacio.

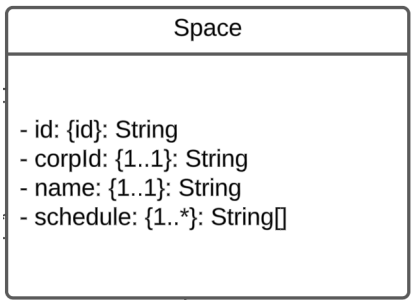


Figura 13. Definición de la entidad Space en el diagrama UML

- **Schedule**: La entidad Schedule hace referencia al conjunto de horarios que existen en un espacio. Está compuesta por los siguientes atributos:
  - **'id'**: Clave primaria de la entidad Schedule, de tipo String.
  - **'spaceid'**: Clave ajena de la entidad Space, de tipo String.
  - **'date'**: De tipo DateTime, hace referencia al día en que se cumple dicho horario de limpieza.

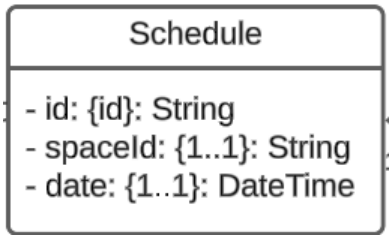


Figura 14. Definición de la entidad Schedule en el diagrama UML

- **Shift**: La entidad Shift hace referencia a los turnos de limpieza que existen en un horario. Está compuesta por los siguientes atributos:
  - **'id'**: Clave primaria de la entidad Shift, de tipo String.
  - **'scheduleId'**: Clave ajena de la entidad Schedule, de tipo String.
  - **'userId'**: Clave ajena de la entidad User, de tipo String.
  - **'hour'**: De tipo String, hace referencia a la hora en que se cumple dicho turno.

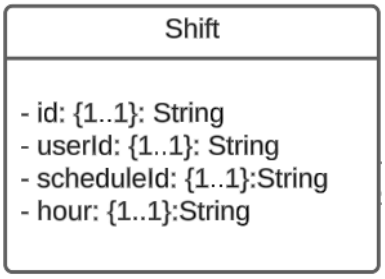


Figura 15. Definición de la entidad Shift en el diagrama UML

A continuación, se muestra el UML completo con las relaciones entre las entidades y sus multiplicidades:

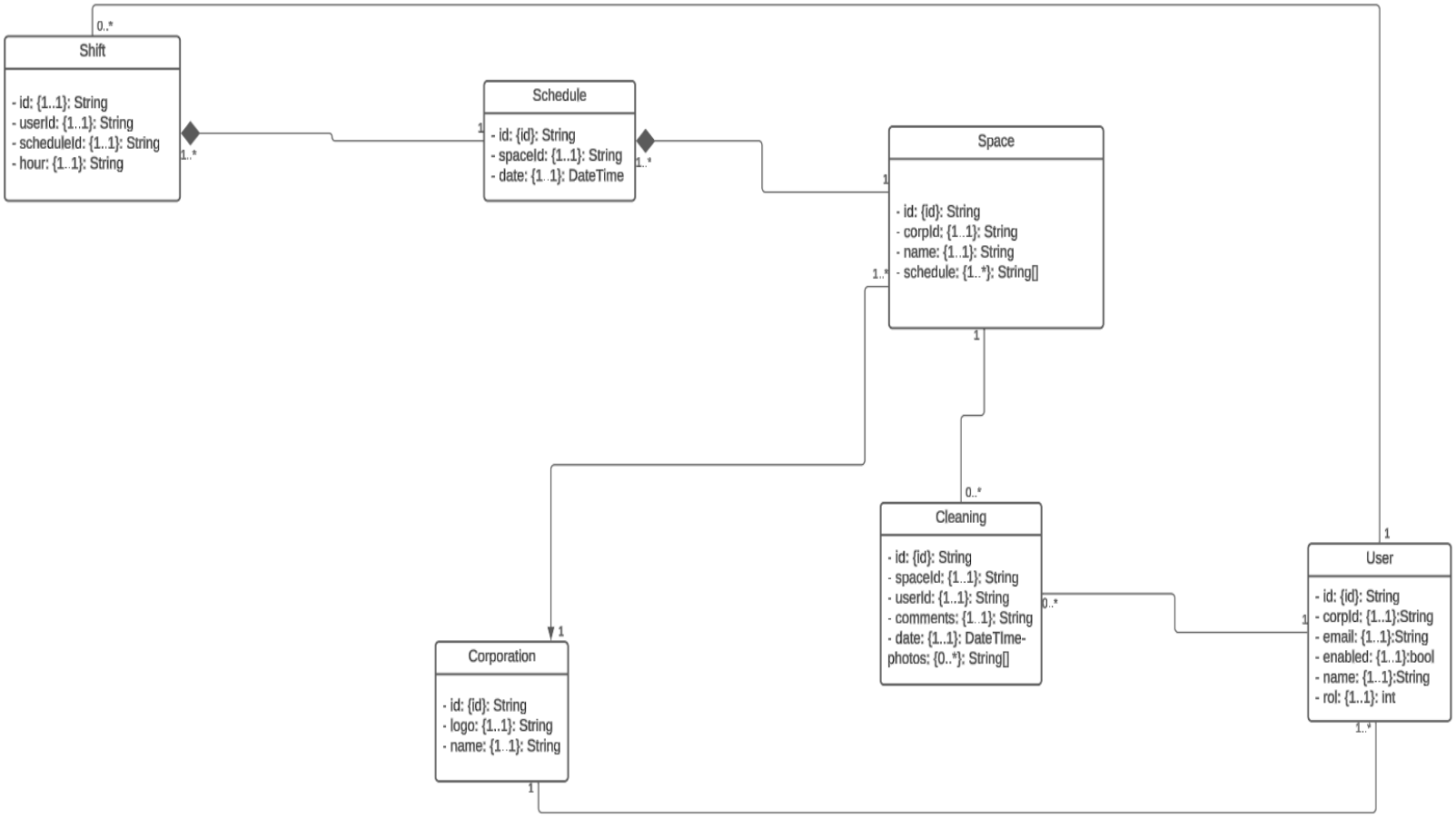

Shift
- id: {1..1}: String
- userId: {1..1}: String
- scheduleId: {1..1}: String
- hour: {1..1}: String
Schedule
- id: {id}: String
- spaceId: {1..1}: String
- date: {1..1}: DateTime
Space
- id: {id}: String
- corpId: {1..1}: String
- name: {1..1}: String
- schedule: {1..*}: String[]
Cleaning
- id: {id}: String
- spaceId: {1..1}: String
- userId: {1..1}: String
- comments: {1..1}: String
- date: {1..1}: DateTIme-
photos: {0..*}: String[]
User
- id: {id}: String
- corpId: {1..1}:String
- email: {1..1}:String
- enabled: {1..1}:bool
- name: {1..1}:String
- rol: {1..1}: int
Corporation
- id: {id}: String
- logo: {1..1}: String
- name: {1..1}: String
0..*
1..*
1
1..*
1
1..*
1
0..*
1
0..*
1
1..*
1

Figura 16. UML de CleanScan

5.2 Diseño de la persistencia

Para la persistencia de los datos se ha seleccionado la tecnología ofrecida por Google Firebase. Esta emplea un modelo no relacional que organiza la información mediante colecciones y documentos. Cada colección representa una entidad y cada objeto de una entidad se representa como un documento dentro de la colección.

Entre la gran variedad de funcionalidades que ofrece FireBase, en este proyecto se van a utilizar las siguientes:

- Authentication
- FireStore Database
- Storage

5.2.1 *Authentication*

Firebase Authentication se utiliza para autenticar a los usuarios mediante sus direcciones de correo electrónico y contraseñas. El SDK de Firebase Authentication proporciona métodos para crear y gestionar usuarios que utilizan estas credenciales para acceder. Además, Firebase Authentication maneja el envío de correos electrónicos para restablecer contraseñas y permite iniciar sesión a través de plataformas como Google y Apple ID. No obstante, en este trabajo, solo se utiliza la funcionalidad para iniciar sesión con correo electrónico y contraseña [4].

Cuando Firebase Authentication persiste un nuevo usuario, almacena su correo electrónico, el proveedor (en caso de haber utilizado una plataforma para registrarse), la fecha en que el usuario se registró y su identificador de usuario (UID). Siguiendo el Artículo 32 del GDPR, que garantiza el tratamiento adecuado de los datos para asegurar la privacidad de los usuarios, las contraseñas se almacenan de manera cifrada y no están visibles [4].

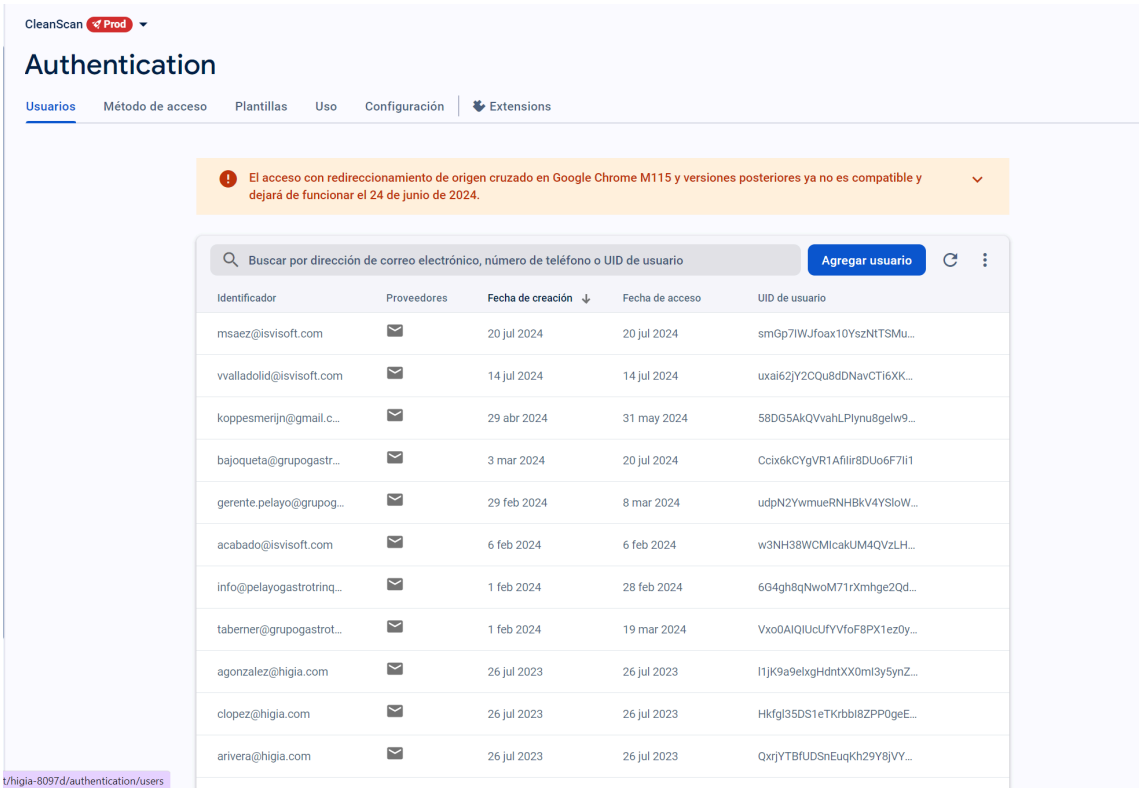


Figura 17. Imagen con información de la autentificación obtenida del panel Authenticaction

En la siguiente imagen se puede observar el panel de Authentication, que permite ver de manera muy intuitiva, sencilla y ordenada los distintos usuarios con su respectiva información.

5.2.2 *FireStore Database*

Firestore Database se utiliza para persistir la información en forma de documentos. En una base de datos no relacional no existen tablas ni columnas, por lo que no es necesario el diseño de tablas con sus atributos (columnas) para crear la base de datos. Sin embargo, para mantener la consistencia de los documentos que se generan en la aplicación, se ha decidido detallar los atributos que contienen los documentos según la colección a la que pertenecen.

Dado que las colecciones corresponden a las mismas entidades detalladas en el apartado 5.1, en este apartado se enumeran junto a sus atributos:

- **User**: id (String), corpId (String), email (String), enabled (boolean), name (String), rol (String).
- **Cleaning**: id (String), spaceId (String), userId (String), comments (String), date (DateTime), photos (String[]).

- **Corporation**: id (String), logo (String), name (String).
- **Space**: id (String), corpId (String), name (String), schedule (String).
- **Schedule**: id (String), spaceId (String), date (DateTime).
- **Shift**: id (String), scheduleId (String), userId (String), hour (String).

En la siguiente imagen se puede observar la estructura en la que Firebase organiza los documentos dentro de las colecciones. A la izquierda se encuentran las colecciones, que pueden asemejarse a las carpetas comúnmente conocidas. En el centro de la imagen se encuentran los documentos asociados a una colección, en este caso, la colección "Cleanings". Debido a la gran cantidad de documentos, Firebase ofrece la posibilidad de buscar y ordenar en función de sus atributos. Finalmente, a la derecha se encuentra la información del documento seleccionado, junto con sus atributos.

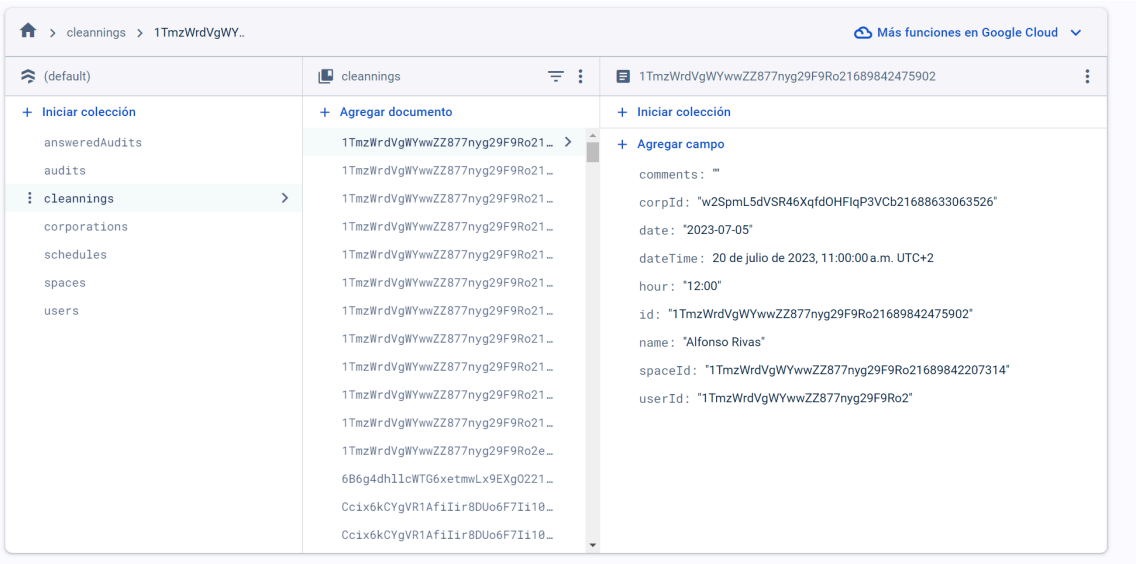


Figura 18. Imagen que muestra una limpieza almacenada en la base de datos

5.2.3 *Storage*

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable, construido para el escalamiento de Google. Los SDK de Firebase para Cloud Storage aportan la seguridad de Google a las operaciones de carga y descarga de archivos en tus aplicaciones Firebase, sin importar la calidad de la red [5].

En este proyecto, Cloud Storage ha sido utilizado para almacenar las fotos asociadas a las limpiezas y a los logos de las empresas. Cuando un usuario guarda una foto,

Firebase genera un enlace que posteriormente se asocia a una limpieza, siguiendo el mismo procedimiento para el almacenamiento de los logos.

En las siguientes imágenes, se puede observar la estructura de Cloud Storage. La organización se basa en un sistema de carpetas, donde cada carpeta contiene las imágenes correspondientes a su categoría específica.

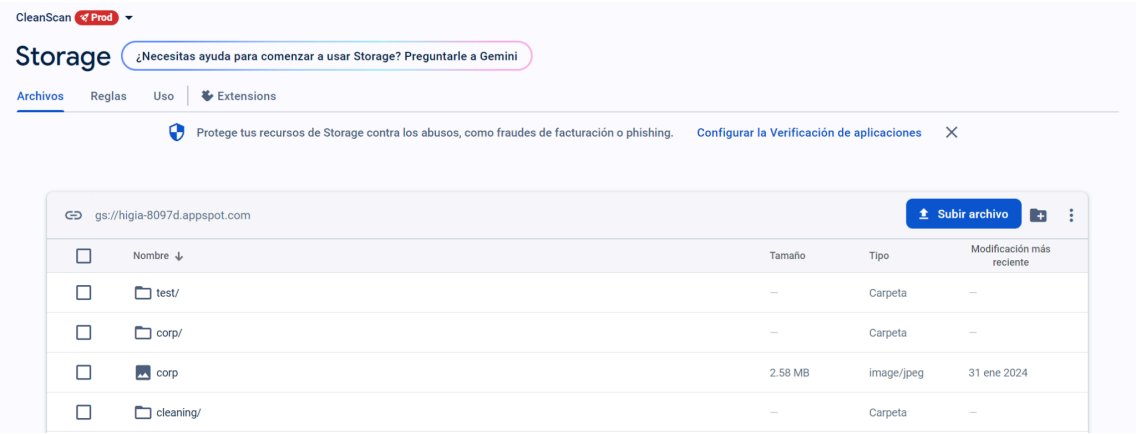


Figura 19. Imagen de la estructura de directorios del panel Storage

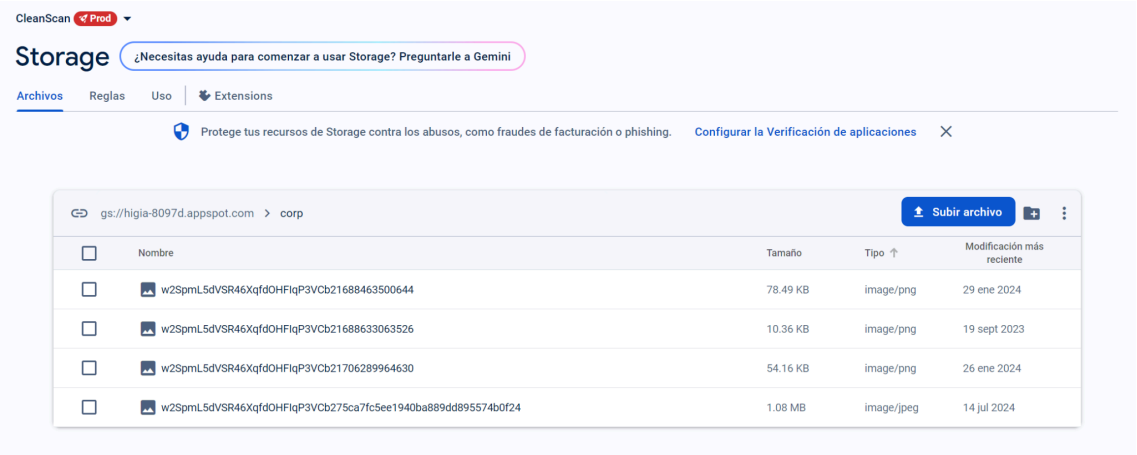


Figura 20. Imagen que muestra diversos ficheros pertenecientes a imágenes de limpiezas

5.3 Diseño de la arquitectura

La arquitectura de software se refiere a la estructura y diseño de un sistema de software. Es una representación de alto nivel que define cómo los componentes del software interactúan entre sí, cómo se organizan y cómo cumplen con los requisitos funcionales y no funcionales del sistema.

La arquitectura de software proporciona una visión global del sistema, lo que permite a los desarrolladores y arquitectos comprender su estructura y tomar decisiones informadas durante el proceso de desarrollo.

Una buena arquitectura de software es fundamental para el éxito de un proyecto, ya que afecta a la calidad, el rendimiento y la escalabilidad del software. Además, facilita la colaboración entre los miembros del equipo de desarrollo y brinda una visión clara de cómo se estructura el sistema, lo que ayuda a minimizar problemas y errores a medida que avanza el desarrollo del software [6].

En este proyecto se ha decidido implementar una arquitectura en tres capas: la capa de presentación, la capa de lógica de negocio y la capa de persistencia. El objetivo de esta arquitectura es aprovechar las ventajas de su modularidad. Gracias a la separación de los componentes, el equipo de trabajo puede desarollar de manera independiente y simultánea las distintas capas. Además, esto ofrece grandes beneficios durante el proceso de desarrollo, ya que se puede apuntar a diferentes backends en el entorno de pruebas y, en el momento del despliegue, apuntar a las versiones de producción de los entornos.

No obstante, la única desventaja de este diseño es la dependencia de Internet de la aplicación. Sin embargo, esta desventaja tiene una relevancia nula debido a la alta disponibilidad que ofrecen estos servicios en la actualidad.
En los siguientes subepígrafe se explican las distintas capas de la arquitectura de la aplicación.

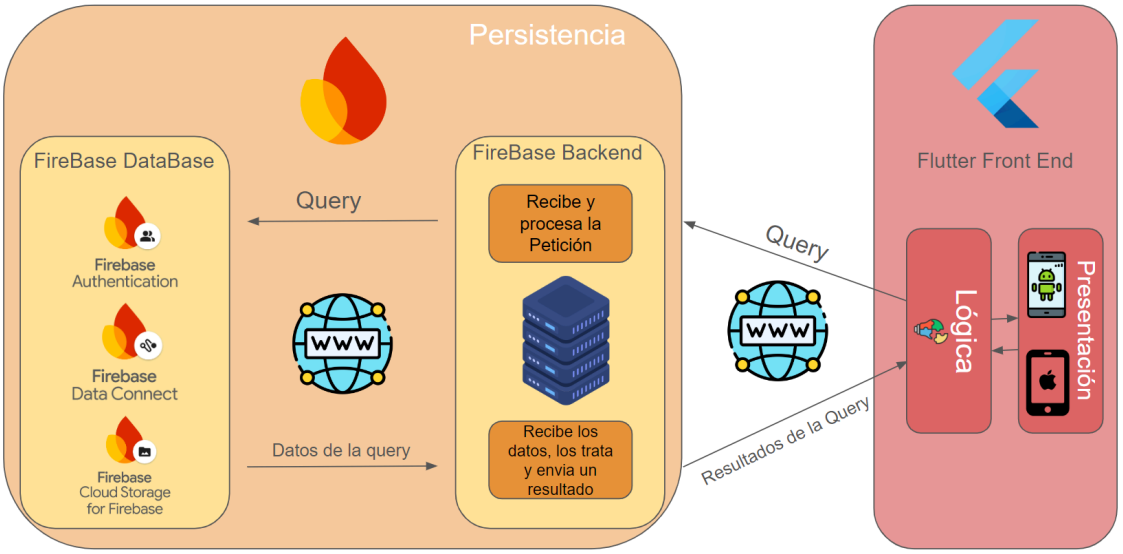


Figura 21. Esquema de la arquitectura simplificada

5.3.1 ***Capa de presentación***

La capa de presentación de un software, comúnmente llamada interfaz de usuario, representa la parte visual de la aplicación. Esta capa únicamente se encarga de mostrar información y capturar las acciones que el usuario ejecuta. Nunca tomará decisiones ni hará operaciones con la información capturada. Esta capa se comunica exclusivamente con la capa lógica.

En CleanScan, la capa de presentación se ejecuta en el propio dispositivo que ejecuta la aplicación y se comunica de manera local con la capa lógica. A continuación se muestra la estructura de archivos de la capa y un ejemplo de la inyeccion de dependencias con la capa lógica:

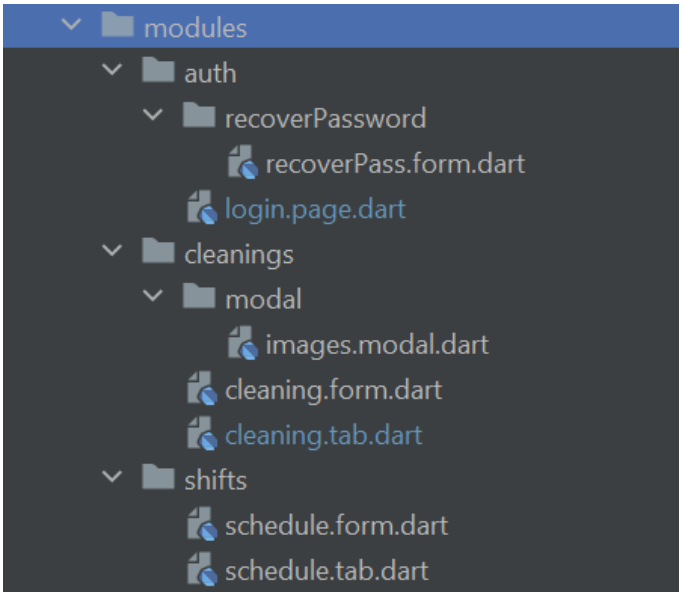


Figura 22. Imagen que muestra la estructura de la carpeta modules

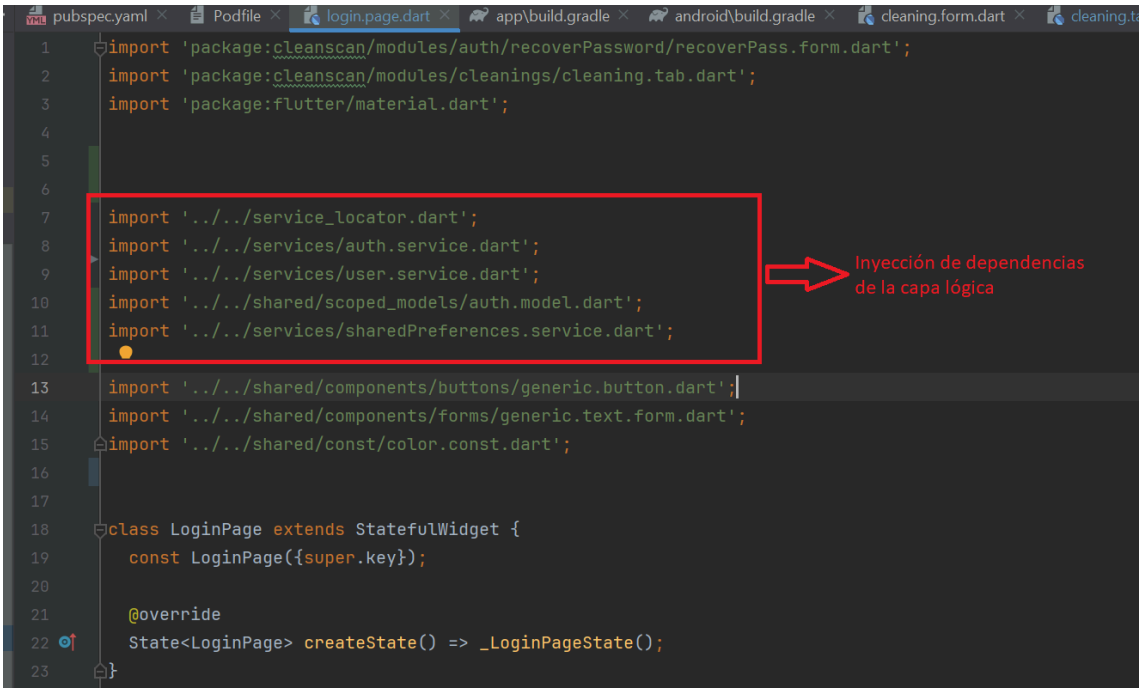


Figura 23. Ejemplo de inyección de dependencias de la capa lógica

5.3.2 *Capa de lógica de negocios*

La capa de lógica de negocio se encarga de la comunicación entre la capa de persistencia y la capa de presentación. Esta capa maneja dos flujos principales: el flujo de datos hacia la capa de presentación, que prepara la información para su visualización, y el flujo de datos hacia la capa de persistencia, que se ocupa de almacenar la información. En esta capa, se definen los modelos de las entidades,

incluyendo sus atributos y métodos necesarios para transformar la información bruta recibida en datos listos para ser mostrados al usuario o almacenados en la base de datos.

En CleanScan se han definido los siguientes modelos:

- Cleaning: es el modelo encargado de definir las limpiezas y tratar su información.
- Corporation: es el modelo encargado de definir una corporación.
- Schedule: es el modelo encargado de definir un horario.
- Space: es el modelo encargado de definir un espacio.
- CleanScanUser: es el modelo encargado de definir un usuario.
- AuthModel: es el modelo encargado de gestionar la autentificación y almacenar la información de esta.

```dart
class CleanScanUser {
  String corpId = "";
  String id = "";
  String email = "";
  String name = "";
  dynamic rol = "";

  CleanScanUser();

  CleanScanUser.fromData({
    required this.corpId,
    required this.id,
    required this.email,
    required this.name,
    required this.rol,
  });

  toMap() {
    return {
      "corpId": corpId,
      "id": id,
      "email": email,
      "name": name,
      "rol": rol,
    };
  }

  static fromMap(Map<String, dynamic> map) {
    return CleanScanUser.fromData(
      corpId: map['corpId'],
      id: map['id'],
      email: map['email'] ?? "",
      name: map['name'] ?? "",
      rol: map['rol'] ?? "",
    );
  }
```

Figura 24. Ejemplo del modelo de la entidad User

5.3.3 *Capa de persistencia*

Una capa de persistencia encapsula el comportamiento necesario para mantener los objetos. O sea: leer, escribir y borrar objetos en el almacenamiento persistente (base de datos). La persistencia de la información es la parte más crítica en una aplicación de software [7].

Para la tercera capa, la capa de persistencia, como se ha mencionado en el apartado “5.2 Diseño de la persistencia”, se ha utilizado Firebase. Para comunicar la capa de lógica de negocio con la capa de persistencia, se han utilizado las siguientes librerías:

- **Firebase Core**: Permite conectar todas las aplicaciones de Firebase.
- **Firebase Auth**: Permite crear una instancia de FirebaseAuth para la conexión con Authentication.
- **Firebase Storage**: Permite crear una instancia de FirebaseStorage para la conexión con Cloud Storage.

Finalmente, para especificar los detalles de la conexión, se añade el archivo *GoogleService-info.plist* a la raíz del proyecto de iOS y el archivo *google-services.json* al nivel de la aplicación en la raíz del proyecto de Android.

google-services.json


```
{
  "project_info": {
    "project_number": "408855687591",
    "project_id": "higia-8097d",
    "storage_bucket": "higia-8097d.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:408855687591:android:4954c29c6a20b98be40682",
        "android_client_info": {
          "package_name": "com.isvisoft.cleanscan.cleanscan_app"
        }
      },
      "oauth_client": [
        {
          "client_id": "408855687591-92fuatq2umctrvaagcsc31tu4ljo61d9.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyBHMXMQ-bDO5pDjdCdHYN346pMKd-s48qg"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": [
            {
              "client_id": "408855687591-92fuatq2umctrvaagcsc31tu4ljo61d9.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        }
      }
    }
  ],
  "configuration_version": "1"
}
```


Figura 25. Contenido del fichero google-services.json

GoogleService-Info.plist

*.plist files are supported by Rider

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>API_KEY</key>
    <string>AIzaSyA3oZMb2_dtLWsSIOn4jHJAYL279md2_F0</string>
    <key>GCM_SENDER_ID</key>
    <string>408855687591</string>
    <key>PLIST_VERSION</key>
    <string>1</string>
    <key>BUNDLE_ID</key>
    <string>com.isvisoft.cleanscan</string>
    <key>PROJECT_ID</key>
    <string>higia-8097d</string>
    <key>STORAGE_BUCKET</key>
    <string>higia-8097d.appspot.com</string>
    <key>IS_ADS_ENABLED</key>
    <false></false>
    <key>IS_ANALYTICS_ENABLED</key>
    <false></false>
    <key>IS_APPINVITE_ENABLED</key>
    <true></true>
    <key>IS_GCM_ENABLED</key>
    <true></true>
    <key>IS_SIGNIN_ENABLED</key>
    <true></true>
    <key>GOOGLE_APP_ID</key>
    <string>1:408855687591:ios:57a15c747c83f72ce40682</string>
</dict>
</plist>
```

Figura 26. Coontenido del fichero GoolgeService-info.plit

5.3.3 ***Capa de presentación***

Para realizar un diseño preliminar antes del proceso de implementación, se ha utilizado una herramienta gratuita llamada Figma. Esta web app permite realizar un prototipado de manera sencilla, arrastrando imágenes y contenedores al mock-up [8].

Es importante destacar que un prototipo es una representación o simulación cuyo objetivo es verificar el diseño, comprobar si cumple con todas las características específicas que el usuario final necesita o demanda, y confirmar su funcionalidad. Por lo tanto, es posible que en la versión final del producto existan ciertas diferencias con respecto al prototipo inicial.

Durante el diseño del mock-up, se han tenido en cuenta aspectos de usabilidad, accesibilidad y consistencia en el diseño para mejorar la experiencia del usuario. Se ha prestado especial atención en mantener una coherencia en la línea de colores y formas en toda la aplicación, se han evitado botones de difícil acceso y se han considerado todos los casos de uso para mostrar la información necesaria en caso de error.

A continuación, se explicará un flujo de ejemplo para navegar por todas las pantallas de la aplicación.

En la primera pantalla, se presenta la interfaz de inicio de sesión. El usuario introduce sus credenciales, con la opción de recordar su usuario para mantener la sesión activa. Si las credenciales introducidas son incorrectas, el sistema mostrará un mensaje de error en la interfaz. En caso de que las credenciales sean correctas, el sistema redirige al usuario a la siguiente pantalla.

Además, si el usuario no recuerda sus credenciales, tiene la opción de hacer clic en el texto "¿Has olvidado la contraseña?" para acceder al formulario de recuperación de contraseña.

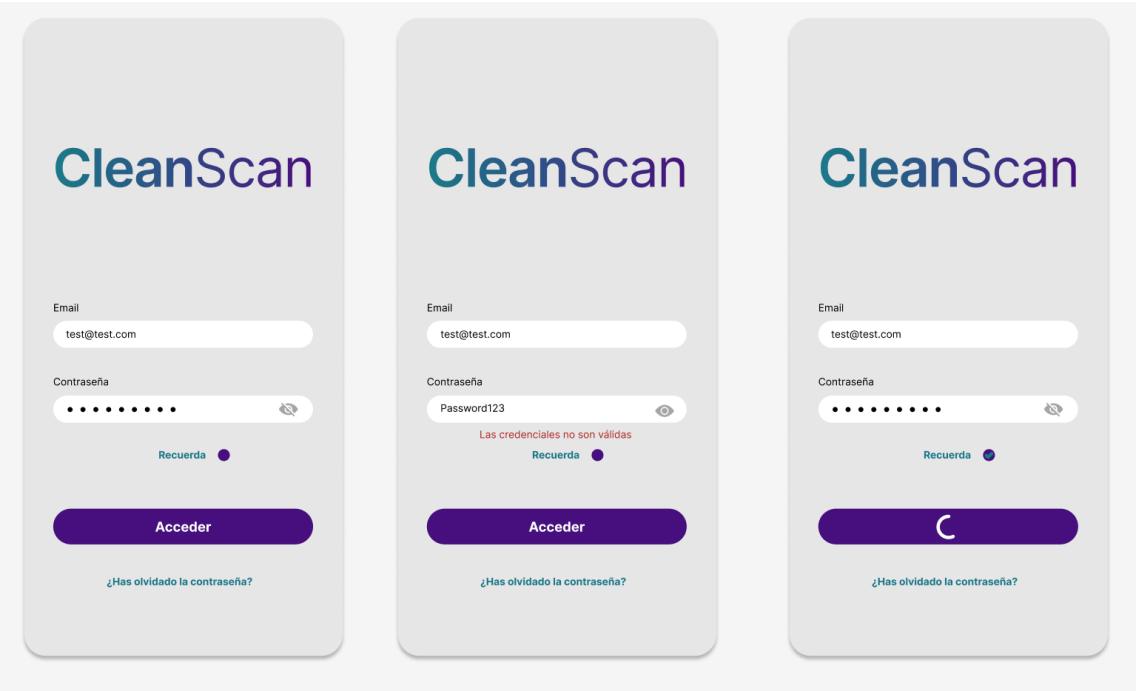


Figura 27. Mock-up de la pantalla de inicio de sesión

En la segunda pantalla, el usuario tiene la opción de recuperar su contraseña introduciendo el correo electrónico asociado a su cuenta. Una vez que el usuario pulsa el botón de enviar, se muestra una ventana modal que confirma que el correo ha sido procesado y enviado satisfactoriamente.

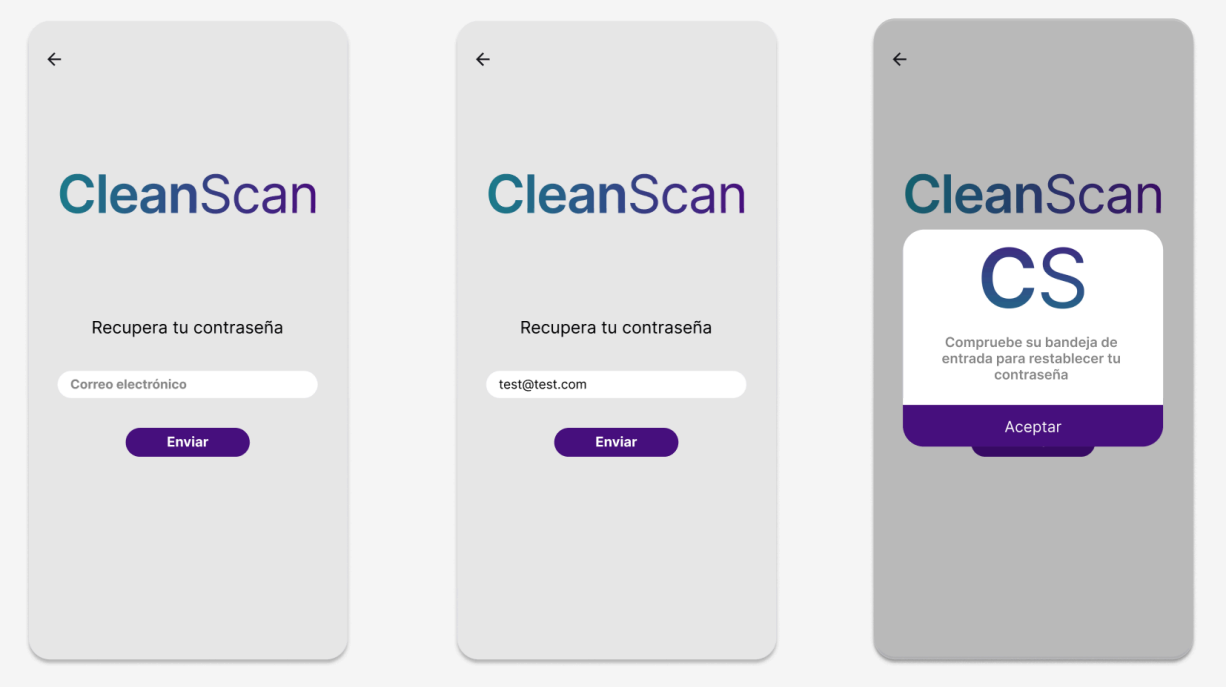


Figura 28. Mock-up de la pantalla de Recuperación de contraseña

Una vez que el usuario ha iniciado sesión, se navega a la pantalla de limpiezas. Por defecto, se selecciona la opción para visualizar las limpiezas ejecutadas en los últimos siete días. Si no existe ninguna limpieza en este periodo de tiempo, se muestra un mensaje de aviso. El usuario tiene dos opciones con periodos de tiempo predefinidos y una tercera opción para seleccionar un período personalizado. En caso de seleccionar esta última opción, se despliega una ventana modal con un calendario.

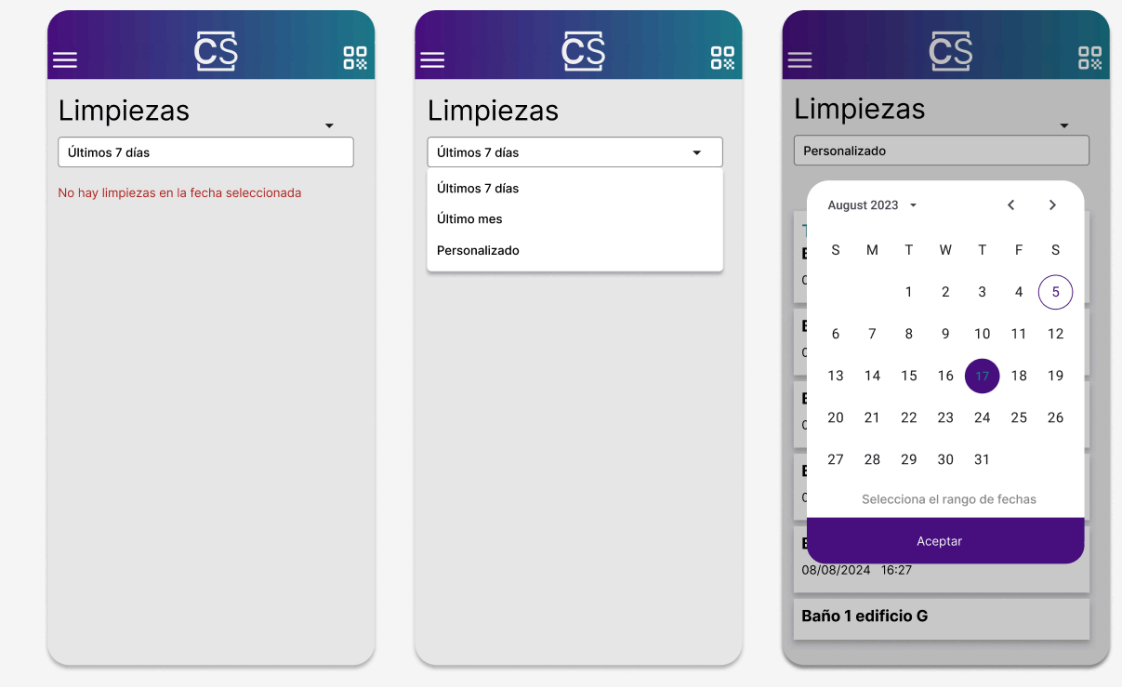


Figura 29. Mock-up de la pantalla de Limpiezas sin limpiezas y del selector de fecha

Una vez seleccionada una franja de tiempo con limpiezas ejecutadas, se muestra una lista de tarjetas que incluyen la siguiente información: nombre del espacio, día y hora en que se realizó la limpieza, y un ícono de imagen si hay fotos disponibles. Si el usuario pulsa sobre el ícono de imagen, se despliega una ventana modal con un menú deslizante que muestra todas las imágenes disponibles junto con el momento en que fueron tomadas.

Al pulsar sobre una tarjeta, se puede visualizar la información de la limpieza de manera más detallada. Además, en la parte superior aparece una barra de aplicación (app bar) con el ícono de la aplicación y dos íconos adicionales: uno que despliega un menú lateral y otro con un ícono QR que navega a la pantalla del escáner QR.

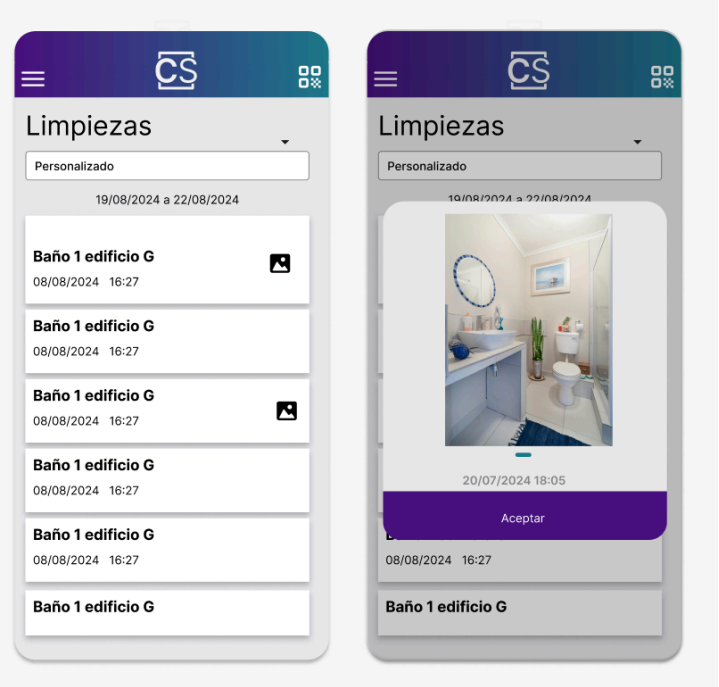


Figura 30. Mock-up de la pantalla de limpiezas con limpiezas y del visor de imagenes

En la pantalla del escáner QR, se muestra una interfaz sencilla donde se activa la cámara para escanear el código QR. Además, se presentan tres botones: uno para cambiar de cámara, otro para activar el flash, y un tercero para cancelar la acción.

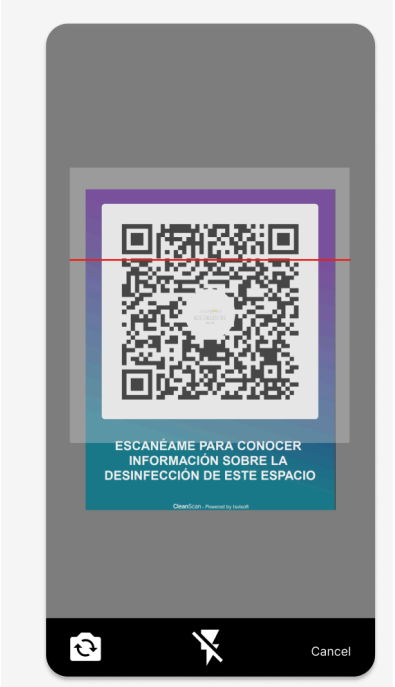


Figura 31. Mock-up del escáner qr

Si el usuario navega desde la pantalla de escaneo de QR, se muestra la pantalla para crear y ejecutar una nueva limpieza. Después de introducir los campos requeridos, el usuario puede pulsar el ícono de “check” para ejecutar la limpieza. Además, tiene la opción de tomar fotos pulsando el botón “Hacer fotos”.

Si el usuario navega desde la pantalla de limpiezas, se mostrará en pantalla la información de la limpieza seleccionada; sin embargo, no podrá modificarla.

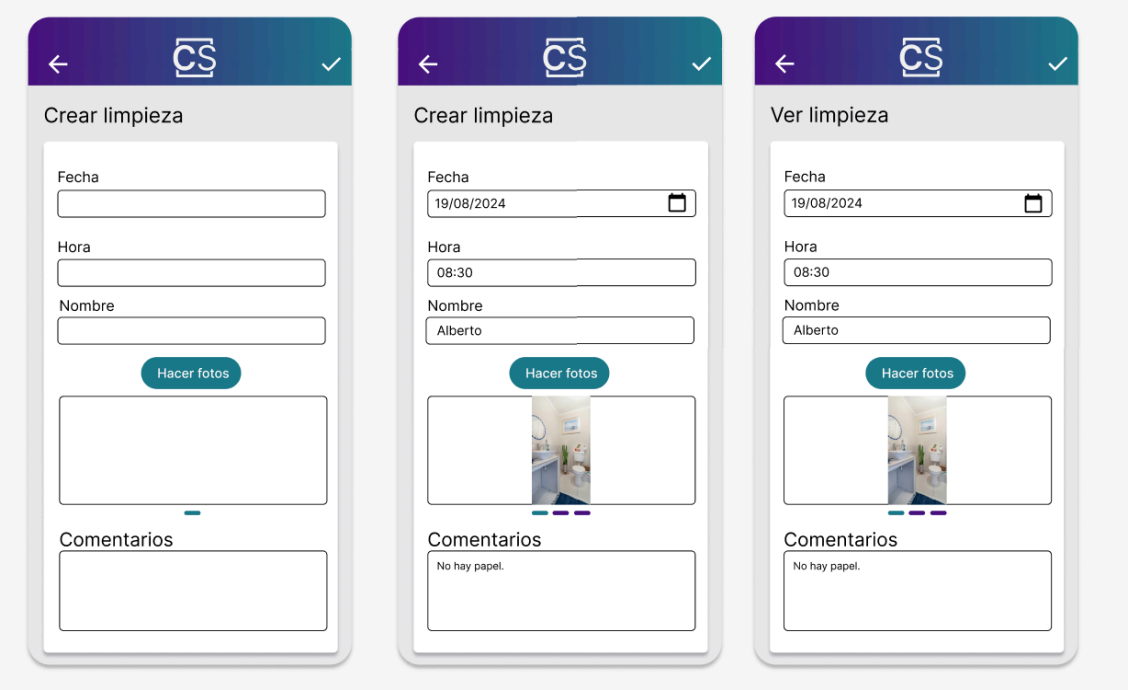


Figura 32. Mock-up de la pantalla de creación y edición de limpiezas

Desde la pantalla de limpiezas o turnos, al pulsar el ícono situado en la parte izquierda de la AppBar, se despliega el siguiente menú lateral. En este menú se muestra el logotipo y nombre de la aplicación, el avatar del usuario, su correo electrónico y el nombre de la compañía. Además, se presentan cuatro botones: “Limpiezas”, que navega a la pantalla de limpiezas; “Turnos”, que lleva a la pantalla de turnos; “Cerrar sesión”, que cierra la sesión del usuario y lo redirige a la pantalla de inicio de sesión; y “Eliminar cuenta”, que borra la cuenta del usuario y también lo redirige al inicio de sesión.

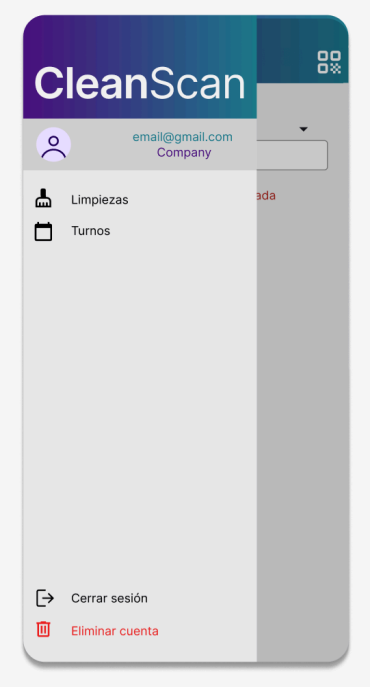


Figura 33. Mock-up del menú de navegación lateral

La pantalla de turnos muestra un botón para agregar nuevos turnos y una lista de tarjetas con la siguiente información: el nombre del espacio y la fecha en que se debe cumplir dicho turno. Al pulsar sobre una tarjeta, se navega a la pantalla de ver turno, donde se pueden visualizar los detalles del turno seleccionado.

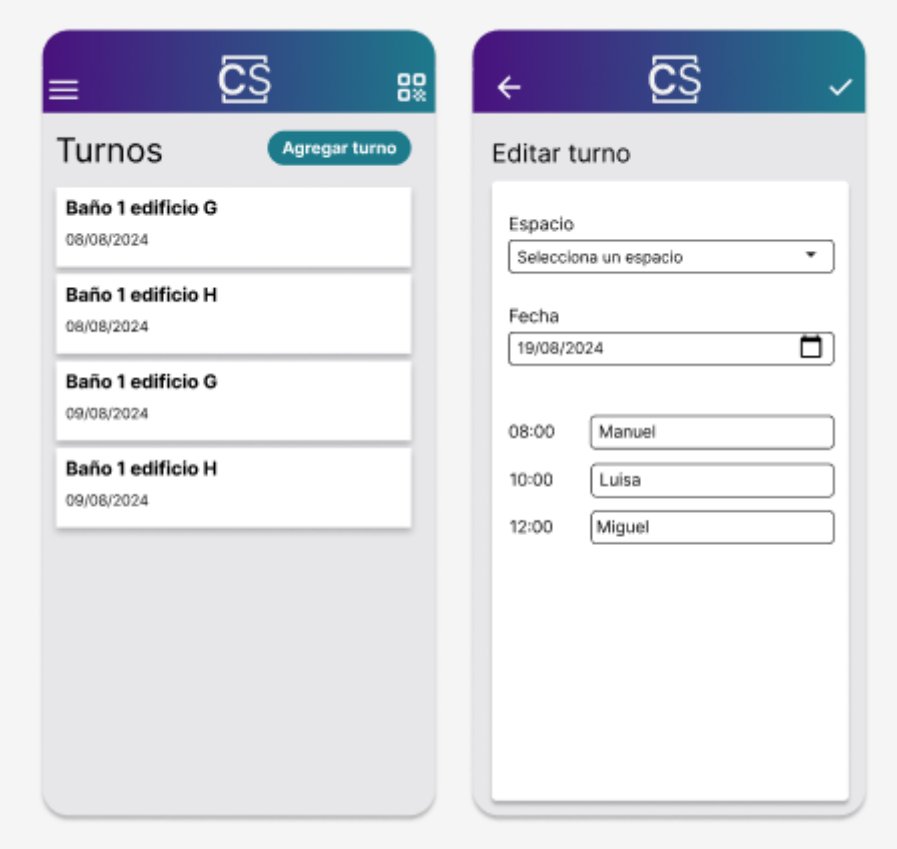


Figura 34. Mock-up de la pantalla de visualización y edición de turnos.

En la pantalla de crear limpieza, se presenta un formulario para crear un nuevo turno de limpieza. El usuario debe seleccionar un espacio, tras lo cual se carga el horario correspondiente a dicho espacio. A continuación, el usuario introduce los nombres de los empleados responsables de limpiar ese espacio. También se debe seleccionar la fecha o el rango de fechas en las que se debe cumplir el horario. Si ya existe un turno programado para ese espacio en la misma fecha, el sistema muestra un mensaje de advertencia. Una vez que todos los campos han sido completados, el usuario puede pulsar el ícono de “check” para confirmar la creación del turno.

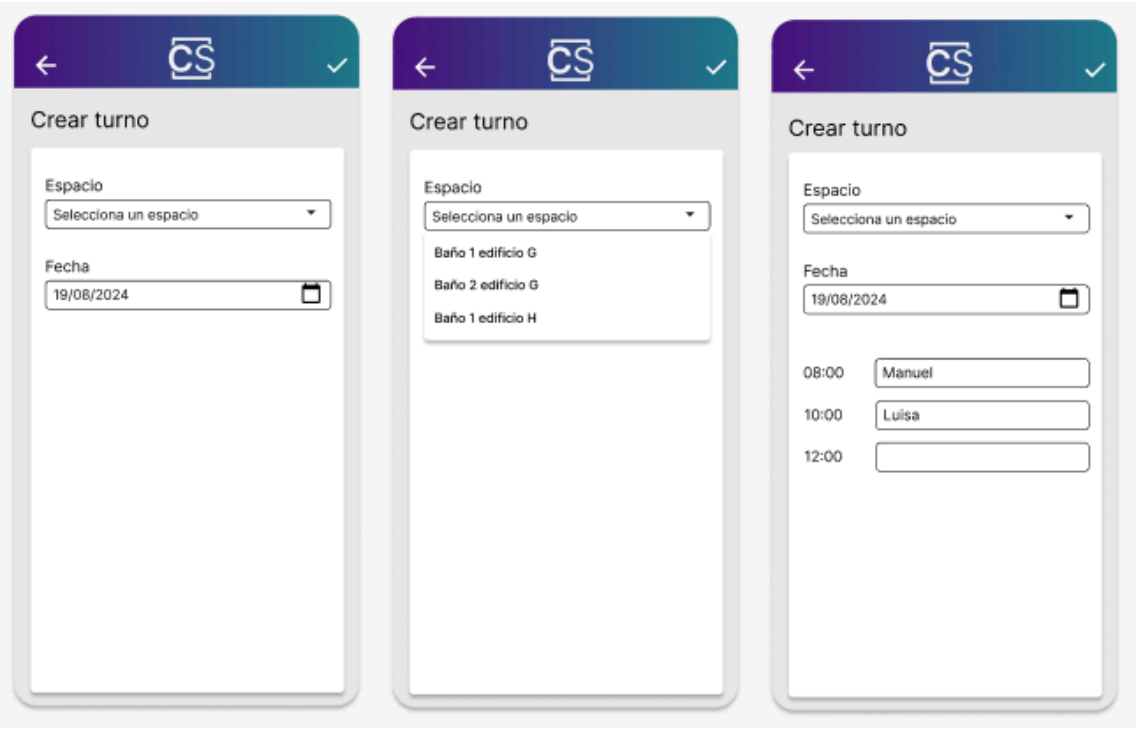


Figura 35. Mock-up de el flujo de creación de turno

6. Implementación

En este apartado se describe cómo se lleva a cabo la implementación de una aplicación en Flutter, utilizando como ejemplo práctico la aplicación CleanScan. Debido a la gran cantidad de clases involucradas, se emplea la clase Cleaning como ejemplo para explicar la implementación en cada paso. Los aspectos del proceso de implementación que se cubrirán en las siguientes secciones son: la preparación del entorno de desarrollo, la creación de un proyecto Flutter y de una estructura básica, la conexión con Firebase, la implementación del servicio, la implementación de los modelos, la implementación de la lógica de negocio, la implementación de la interfaz de usuario y el uso de librerías de terceros.

6.1 Preparación del entorno de desarrollo

Como se ha descrito en el apartado tres, el entorno de desarrollo utilizado es Android Studio para implementar la aplicación mediante el SDK de Flutter. Antes de preparar el entorno, es necesario verificar que el dispositivo de desarrollo donde se instalarán las herramientas tenga las siguientes características, o superiores: un procesador con cuatro núcleos de procesamiento, ocho gigabytes de memoria RAM, un monitor de resolución 1366 x 768 y cuatro gigabytes libres en el disco de almacenamiento. En los siguientes subepígrafes se detallarán los pasos a seguir para la preparación del entorno.

Primero, es necesario instalar el SDK de Flutter, que se puede descargar preferiblemente desde la página oficial[1]. Una vez descargado e instalado el SDK, al ejecutar el comando flutter doctor en el terminal, se instalará automáticamente la última versión del canal estable de Flutter junto con su respectiva versión de Dart.

Después, es necesario instalar la última versión de Android Studio, descargándola desde la fuente oficial[2]. Una vez instalado, se debe descargar durante el proceso de instalación la última versión del SDK de Android. En caso de no disponer de un dispositivo físico para pruebas, se debe configurar un emulador con la última versión de Android.

[1] Enlace a la pagina oficial de Flutter: "https://docs.flutter.dev/get-started/install ".
[2] Enlace a la pagina oficial de Android Studio: "https://docs.flutter.dev/get-started/install ".

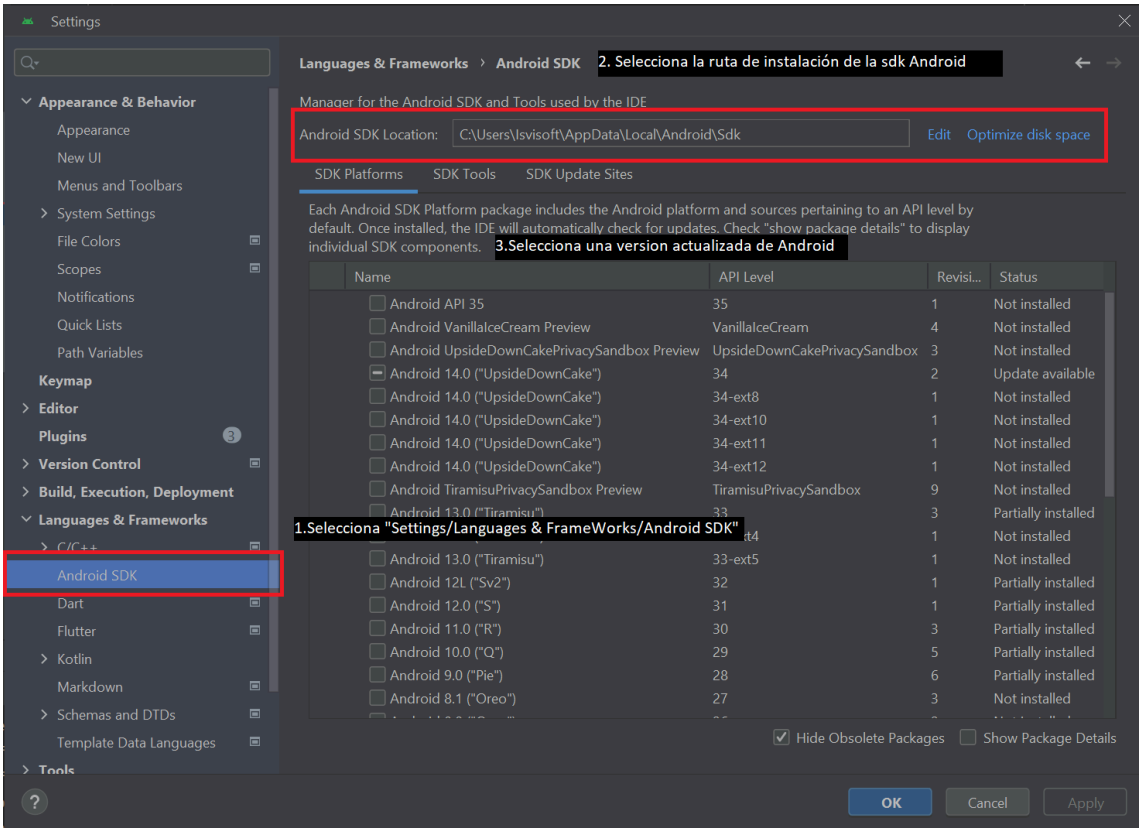


Figura 36. Pantalla de ajustes de Android Studio donde se selecciona el SDK de Android

6.2 Creación de un proyecto Flutter y de su estructura básica

Una vez que hayas iniciado Android Studio, es necesario crear un nuevo proyecto Flutter. Para ello, utiliza el generador de proyectos de Flutter que te permitirá configurar los detalles esenciales del proyecto. Entre los pasos a seguir se incluyen la especificación del nombre del proyecto, el nombre de la organización y la ruta del proyecto en el sistema de archivos.

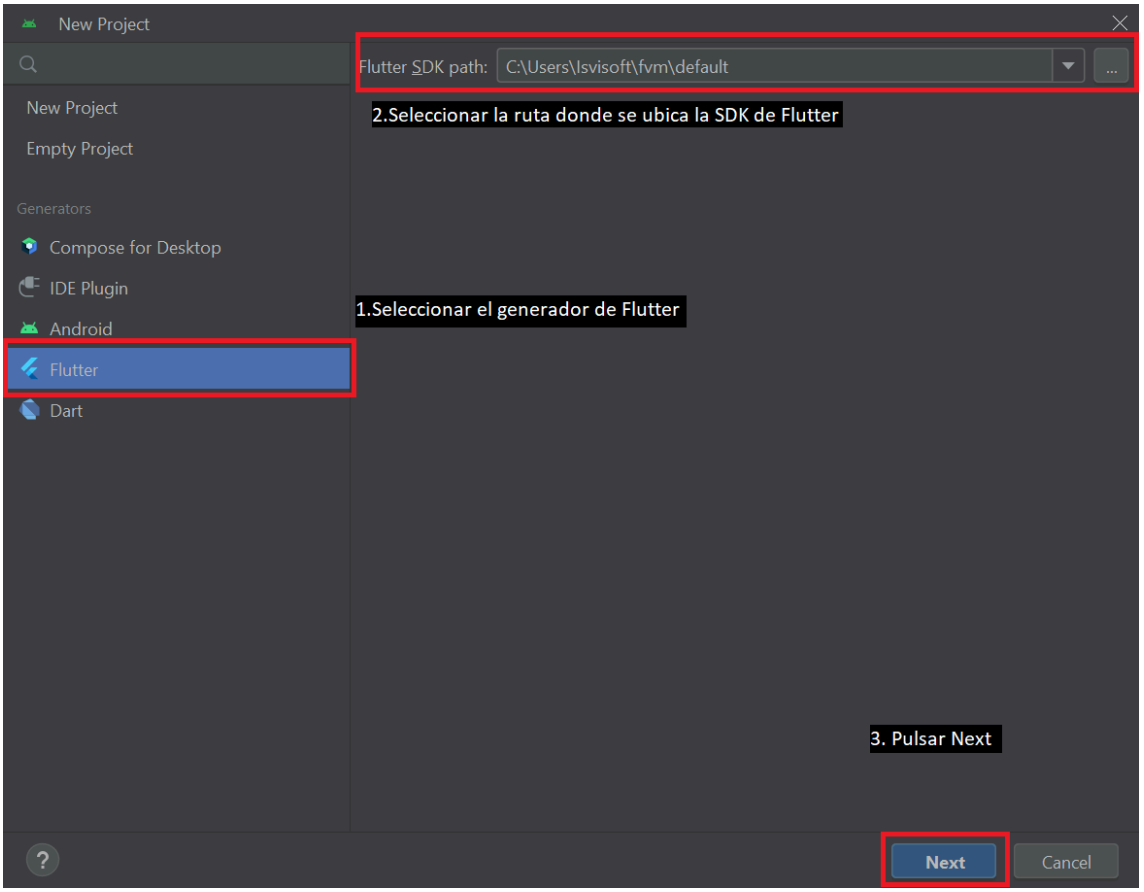


Figura 37. Pantalla de ajustes de Android Studio para crear un nuevo proyecto Flutter

En el caso de CleanScan, se ha elegido como nombre del proyecto "higia-app", y el nombre de la organización es "com.isvisoft".

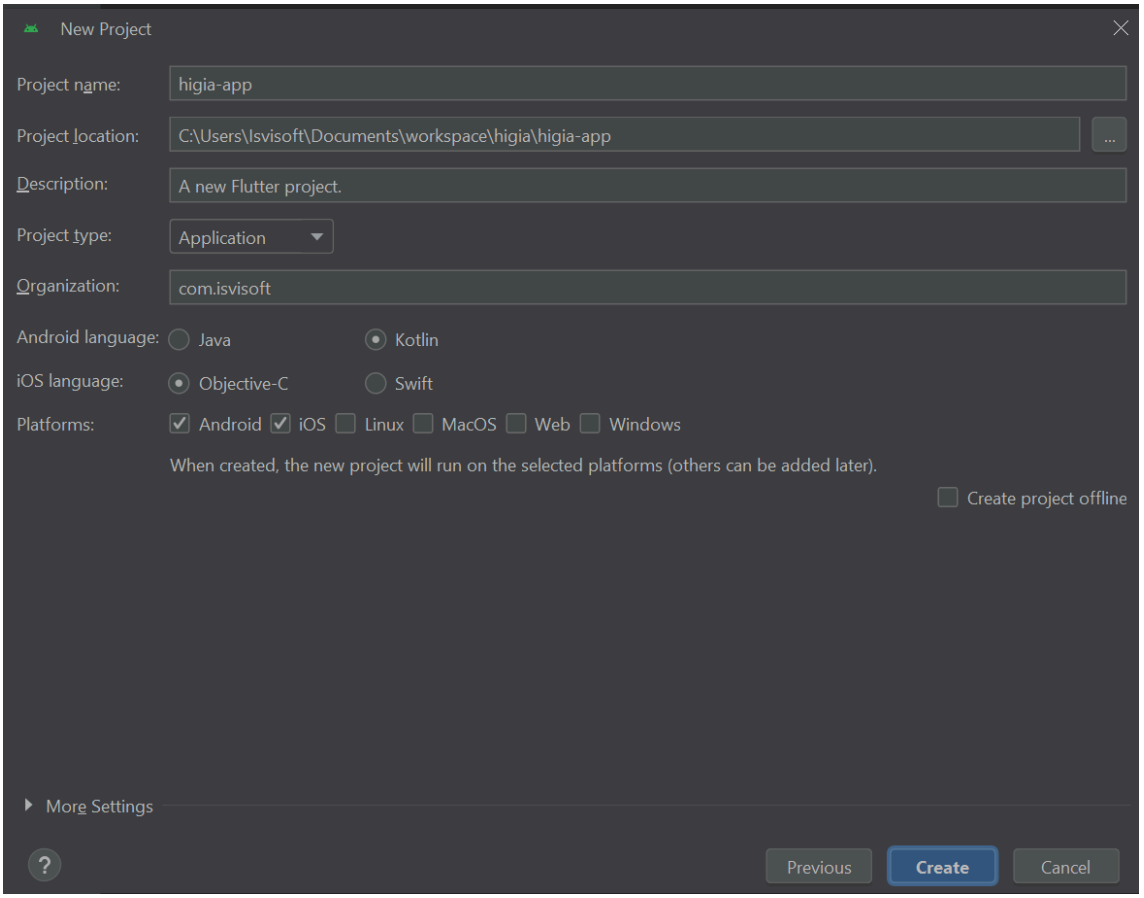


Figura 38. Pantalla de ajustes de Android Studio donde se introducen los parámetros de nuevo proyecto

De igual manera, también es posible crear un proyecto Flutter mediante el comando **`flutter create --org com.<ORGANIZATION> --project-name <NAME> <DIRECTORY>`**, donde **<ORGANIZATION>** es el nombre de la organización, **<NAME>** es el nombre de la aplicación y **<DIRECTORY>** es el nombre del directorio en el que se almacenará el proyecto.

Tras crear el proyecto, se genera automáticamente una estructura básica, la cual puede ser organizada de manera más eficiente añadiendo las siguientes carpetas en el directorio lib:

- models: Esta carpeta contendrá las definiciones de los modelos de datos.
- modules: En este directorio se desarrollará la interfaz de usuario.
- services: Esta carpeta estará destinada a la implementación de servicios.
- shared: Aquí se almacenarán constantes y componentes que se utilizan en distintas partes del proyecto.
- utils: Esta carpeta albergará diversas utilidades y funciones auxiliares.

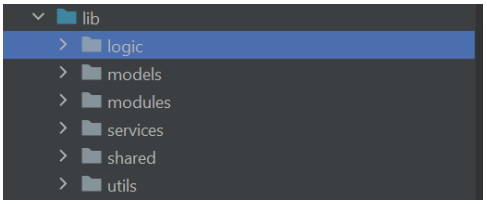


Figura 39. Estructura de los directorios del proyecto

Además, es necesario definir Locator, una clase que nos permitirá implementar el patrón de diseño Singleton, asegurando una única instancia de un objeto. Este patrón de diseño es muy útil en el proyecto para crear una única instancia de los servicios. Para lograr esto, primero se debe instalar el paquete correspondiente ejecutando el comando flutter pub add locator en la terminal. Luego, se debe crear un archivo denominado service_locator.dart e implementar el código mostrado en la figura 40.

Una vez definida la clase, cada vez que se desee crear un Singleton de una clase, se debe añadir la siguiente línea de código, sustituyendo <CLASE> por la clase deseada, en el método setupLocator():

- **locator.registerLazySingleton<<CLASE>>(() => <CLASE<());**

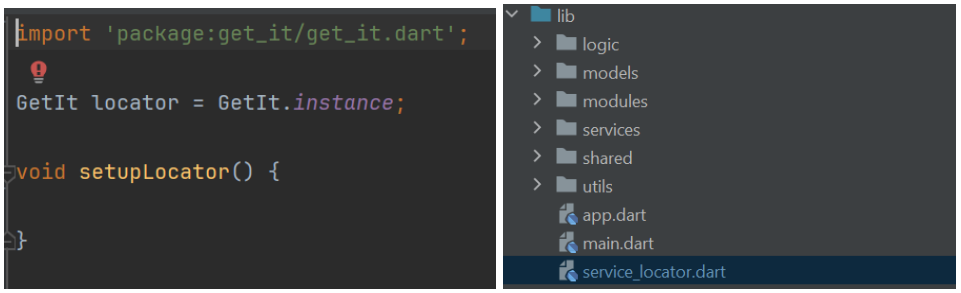


Figura 40. Código base de la librería locator y ubicación del archivo en el proyecto

Finalmente, por defecto, Flutter crea la clase MyApp, HomePage y el método main() dentro del archivo main.dart. Con el objetivo de mantener un código más limpio y modular, se realiza una refactorización extrayendo la clase MyApp al archivo app.dart (como se muestra en la figura 41) y trasladando la clase HomePage a un módulo dentro del directorio modules.

```dart
import 'package:cleanscan/modules/auth/login.page.dart';
import 'package:cleanscan/modules/cleanings/cleaning.tab.dart';
import 'package:cleanscan/service_locator.dart';
import 'package:cleanscan/shared/scoped_models/auth.model.dart';
import 'package:flutter/material.dart';
import 'package:scoped_model/scoped_model.dart';

import 'shared/const/color.const.dart';

class MyApp extends StatelessWidget {
  MyApp({super.key});
  final AuthModel userModel = locator<AuthModel>();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        title: 'CleanScan',
        theme: ThemeData(
            colorScheme: ColorScheme.fromSwatch().copyWith(
          primary: purpleCleanScan,
          secondary: secondaryColor,
        )),  // ThemeData
        debugShowCheckedModeBanner: false,
        home: ScopedModel<AuthModel>(
            model: userModel,
            child: ScopedModelDescendant<AuthModel>(
                builder: (modelContext, child, model) => userModel.isAuth ? const CleaningsForm() : const LoginPage())  // ScopedModelDescendant
    ));  // ScopedModel, MaterialApp
  }

  Future<bool> isAuth() async{
    return userModel.isAuth;
  }
}
```

Figura 41. Ejemplo de MyApp en CleanScan

6.3 Conexión con firebase

Para realizar la conexión a Firebase, primero es necesario agregar la aplicación, tanto para Android como para iOS, en la configuración del proyecto de Firebase.

Para Android, debes registrar la aplicación con el nombre que elegiste previamente al crear el proyecto Flutter, siguiendo el formato: com.<ORGANIZATION>.<NAME>. Una vez introducido, se generará un archivo llamado google-services.json, que debe ser copiado a la raíz del proyecto Android, a nivel de la carpeta app.

Figura 42. Registro de la aplicación en FIrebase

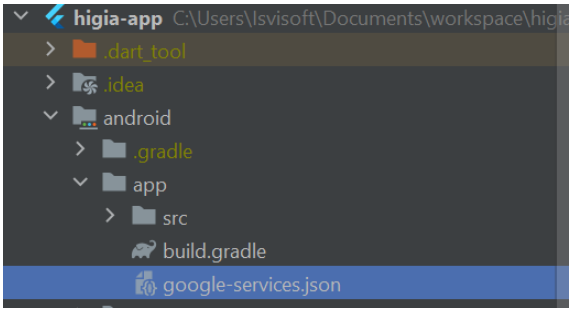


Figura 43. Ubicación del fichero google-services.json en el proyecto

Para iOS, se siguen los mismos pasos, pero cuando se genere el archivo GoogleService-Info.plist, este debe añadirse a la raíz del proyecto en Xcode.

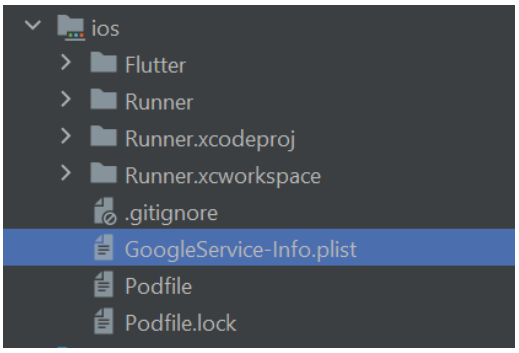


Figura 44. Ubicación del fichero GoogleService-Info.plist en el proyecto

Una vez que se han agregado los archivos correspondientes (google-services.json para Android y GoogleService-Info.plist para iOS) al proyecto, es necesario proceder con la integración del SDK de Firebase en el entorno de desarrollo. Para llevar a cabo esta integración, se deben seguir los pasos detallados a continuación:

- Agregar la dependencia a nivel de proyecto en Android: se debe abrir el archivo build.gradle correspondiente al nivel de proyecto (es decir, el archivo situado en la raíz del proyecto de Android) y agregar la siguiente dependencia en la sección dependencies: **classpath 'com.google.gms:google-services:4.3.25'**.

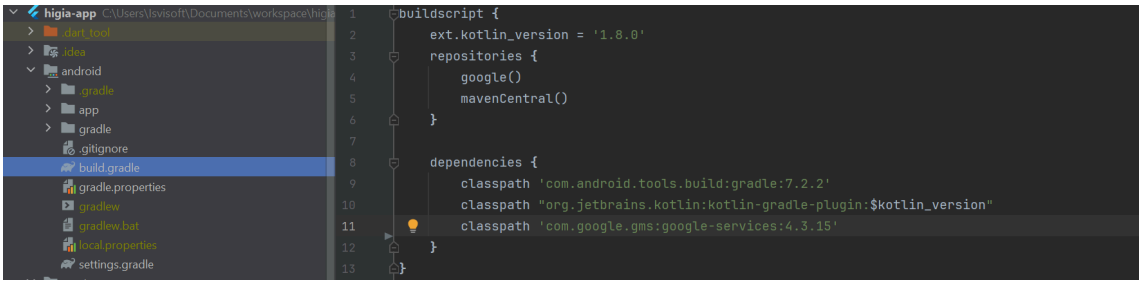


Figura 45. Ejemplo de como se debe añadir la conexión de Firebase al proyecto de android

- Agregar a nivel de app del proyectos android los complementos de google-services y las dependencias de las sdk de firebase que se vayan a utilizar, en el caso de cleanscan, no es necesario agregar ningún extra.

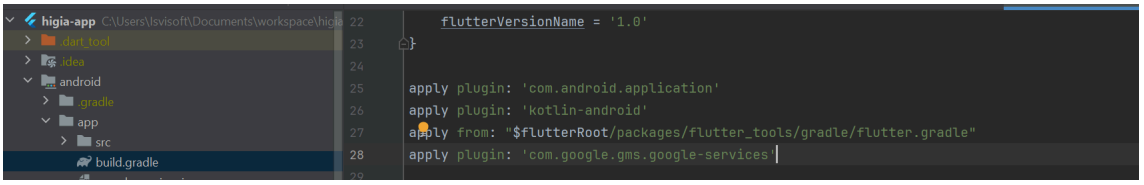


Figura 46. Ejemplo de como se debe añadir la conexión de Firebase a nivel de app

6.4 Creación de los modelos de las entidades

Cuando se recibe un objeto de la base de datos, este se presenta en formato JSON. Para transformar dicho objeto en una entidad dentro de la aplicación, es necesario definir un modelo que represente dicha entidad de manera estructurada. Este modelo debe incluir todos los atributos que caracterizan a la entidad, así como un constructor y los métodos necesarios para convertir un Map<String, dynamic> en una instancia de la entidad y viceversa.

Como ejemplo, se ha utilizado el modelo de la entidad Cleaning. Se han declarado los atributos de la entidad e inicializado con valores vacíos o nulos. Los atributos son: comments, date, hour, id, name, spaceId, userID, dateTime y photos. El constructor se ha definido vacío, y se han implementado los métodos fromData, fromMap y toMap.

El método fromData crea una instancia del modelo a partir de los datos proporcionados como parámetros, con la siguiente estructura: Cleaning.fromData(

```
  comments: "comentario",
  date: DateTime(),
  hour: "13:04",
  id: "9fes99k3902mdk",
  name: "Antonio",
  spaceId: "9fes99k3902faww",
  userId: "9fes99k3902mdkowakwdkow",
  dateTime: DateTime(),
  photos: []
);
```

El método toMap convierte una instancia del modelo en un mapa con la estructura Map<String, dynamic>. Finalmente, el método fromMap transforma un Map<String, dynamic> en una instancia del modelo.

```
class Cleaning {
```

```
String comments = "";
String date = "";
String hour = "";
String id = "";
String name = "";
String spaceId = "";
String userId = "";
dynamic dateTime;
List<dynamic> photos = [];

Cleaning();

Cleaning.fromData({
  required this.comments,
  required this.date,
  required this.hour,
  required this.id,
  required this.name,
  required this.spaceId,
  required this.userId,
  required this.dateTime,
  required this.photos,
});

toMap() {
  return {
    "comments": comments,
    "date": date,
    "hour": hour,
    "id": id,
    "name": name,
    "spaceId": spaceId,
    "userId": userId,
    "dateTime": dateTime,
    "photos": photos,
  };
}
```

```
 static fromMap(Map<String, dynamic> map) {
   return Cleaning.fromData(
     comments: map['comments'] ?? "",
     date: map['date'].toString(),
     hour: map['hour'] ?? "",
     id: map['id'] ?? "",
     name: map['name'] ?? "",
     spaceId: map['spaceId'] ,
     userId: map['userId'] ?? "",
     dateTime: map['dateTime'] ?? "",
     photos: map['photos'] ?? [],
   );
 }

}
```

Figura 47. Ejemplo de Modelo de la entidad Cleaning

6.5 Creación de los servicios

Para realizar las peticiones a Firebase, es necesario implementar una capa de servicios donde se lleven a cabo dichas operaciones. Para ello, se crea una clase base en la que se definen las acciones principales, como guardar un objeto, obtener un objeto por ID, recibir todos los objetos, crear un ID para un objeto y eliminar por ID. Además, en esta clase se instancia FirebaseFirestore para permitir la conexión con la base de datos, y FirebaseAuth para gestionar la instancia del usuario actual.

En la clase base, también se define el atributo collection, que es especificado por la clase que extiende esta base, asignando el nombre de su colección. Asimismo, Firebase se inicializa en el constructor de la clase.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:uuid/uuid.dart';

class BaseService {
 final FirebaseAuth auth = FirebaseAuth.instance;
```

```
final db = FirebaseFirestore.instance;
String collection = "";

BaseService(this.collection) {
  Firebase.initializeApp();
}

getById(String id) async {
  var res = await db.collection(collection).doc(id).get();
  return res.data();
}

save(doc) async{
  createId(doc);
  return await db
      .collection(collection)
      .doc(doc.id)
      .set(doc.toMap());
}


getAll() async {
  var res = await db.collection(collection).get();
  return res.docs;
}

createId(doc) {
  if (doc.id == null || doc.id == '') {
              doc.id   =   auth.currentUser!.uid   +   const
Uuid().v4().replaceAll('-', '');
  }
}

deleteById(id) async {
  try {
                                var      res      =      await
db.collection(collection).doc(id).delete();
```

```
      return res;
    } catch (_) {}
  }


}
```

Figura 48. Ejemplo del servicio BaseService

Por otra parte, los servicios restantes extienden la clase BaseService para heredar todos los métodos básicos necesarios para interactuar con la base de datos. En el ejemplo mostrado en la figura 49, CleaningService extiende BaseService y define el atributo collection con el nombre de su colección: "Cleaning". Además, CleaningService añade métodos específicos para realizar las peticiones requeridas, como recibir las limpiezas por periodos de tiempo.

```
import 'dart:io';
import 'package:cleanscan/models/cleaning.model.dart';
import 'package:cleanscan/models/space.model.dart';
import 'package:cleanscan/services/base.service.dart';
import 'package:cleanscan/services/spaces.service.dart';
import 'package:cleanscan/services/storage.service.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:http/http.dart' as http;
import 'package:path/path.dart' as path;
import 'package:path_provider/path_provider.dart';

import '../shared/const/dates.const.dart';
import '../utils/date.utils.dart';

class CleaningService extends BaseService {
 CleaningService() : super("cleannings");

getAllBetweenDates(String iniDate, String endDate, String
corpId) async {
   QuerySnapshot cleanings = await db
       .collection(collection)
       .orderBy("date")
       .where("date", isGreaterThanOrEqualTo: iniDate)
       .where("date", isLessThanOrEqualTo: endDate)
```

```
        .where("corpId", isEqualTo: corpId)
        .get();

          final    allData    =    cleanings.docs.map((doc)    =>
doc.data()).toList();
   return allData;
 }

  Future  getCleanings(String  iniDate,  String  endDate,  String
corpId) async {
       final  cleanings  =  await  getAllBetweenDates(iniDate,
endDate, corpId);
   List<Cleaning> cleaningList = [];
                                cleanings.map((doc)          =>
cleaningList.add(Cleaning.fromMap(doc))).toList();
   return cleaningList;
 }

Future  getCleaningsByPeriod(String  period,  String  corpId)
async {
   DateTime currentDate = DateTime.now();
         DateTime   lastWeekDate   =   DateTime.now().add(const
Duration(days: -7));
   int currentYear = currentDate.year;
   int currentMonth = currentDate.month;

                              String          today          =
DateUtil.formatDateToStringEnglish(currentDate);
                            String         lastWeekDay         =
DateUtil.formatDateToStringEnglish(lastWeekDate);
   if (period == lastWeek) {
     return getCleanings(lastWeekDay, today, corpId);
   } else if (period == lastMonth) {
     String iniDate = "";
     String endDate = "";
     if (currentMonth < 10) {
       iniDate = "$currentYear-0$currentMonth-01";
```

```
      endDate = "$currentYear-0$currentMonth-31";
    } else {
      iniDate = "$currentYear-$currentMonth-01";
      endDate = "$currentYear-$currentMonth-31";
    }
    return getCleanings(iniDate, endDate, corpId);
  }
}

}
```

Figura 49. Ejemplo del servicio CleaningService

6.6 Creación de la presentación

Toda la interfaz de una aplicación Flutter se define mediante widgets, que pueden ser de dos tipos: Stateful o Stateless. Estos widgets pueden contener otros widgets como Container, Text, ListView, entre otros. La característica principal de un StatefulWidget es que posee un estado, lo que permite, mediante el método setState, actualizar el estado y, en consecuencia, refrescar la interfaz con los valores actualizados. Esto resulta muy útil en pantallas que reciben información de una base de datos, ya que, al recibir nuevos datos, la pantalla puede actualizarse automáticamente con estos valores. Por el contrario, un StatelessWidget no tiene estado, es decir, no puede ser refrescado por sí mismo. Es óptimo para pantallas que no requieren actualización continua.

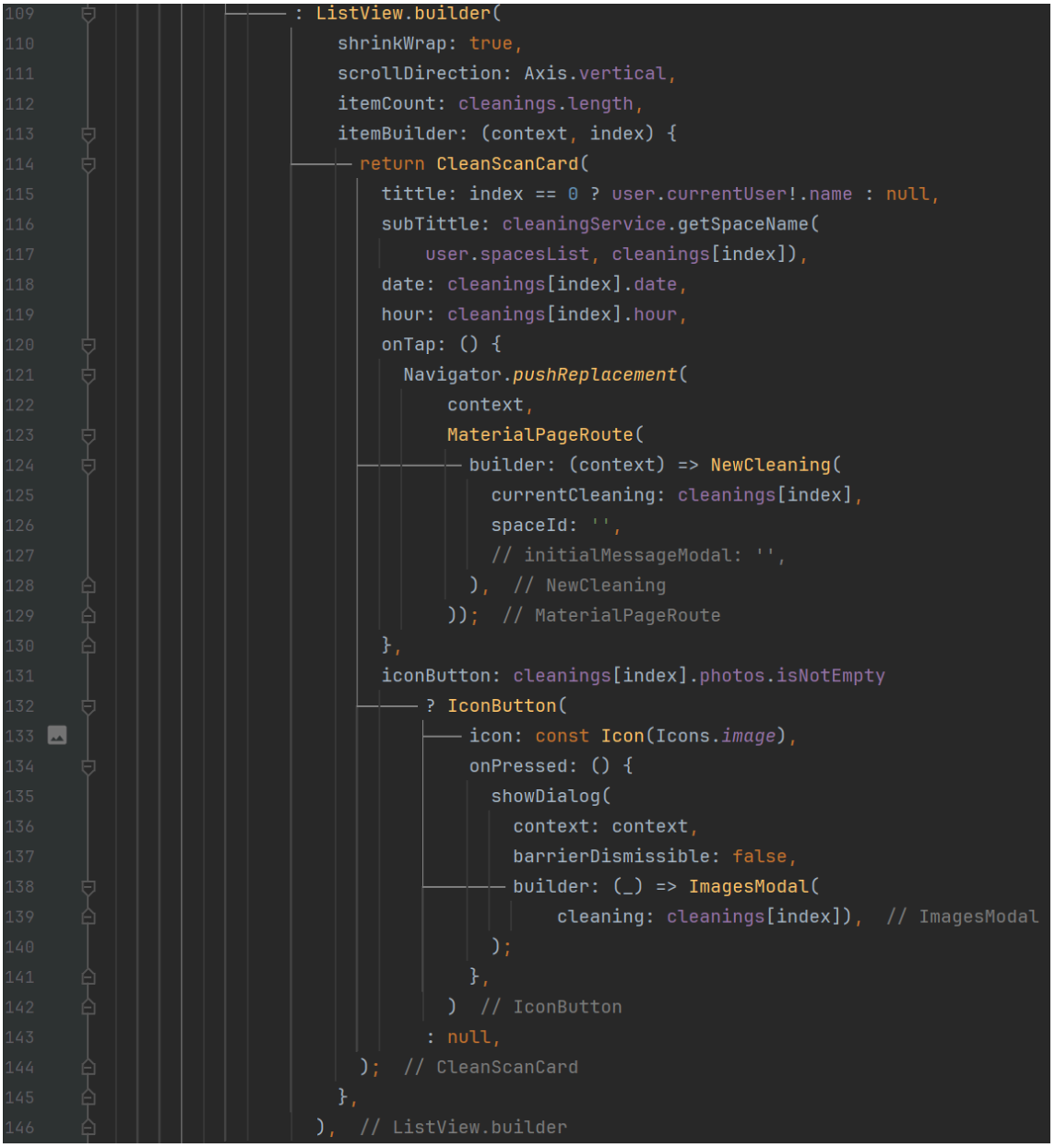
En la clase MyApp, se define un widget sin estado (StatelessWidget) donde se configuran las características principales de toda la aplicación, como la fuente predeterminada, los colores primarios y secundarios, y el idioma. Además, se selecciona qué pantalla se va a cargar según si el usuario ha optado por mantener la sesión activa o no.

```dart
class MyApp extends StatelessWidget {
  MyApp({super.key});
  final AuthModel userModel = locator<AuthModel>();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        title: 'CleanScan',
        theme: ThemeData(
            colorScheme: ColorScheme.fromSwatch().copyWith(
          primary: purpleCleanScan,
          secondary: secondaryColor,
        )),  // ThemeData
        debugShowCheckedModeBanner: false,
        home: ScopedModel<AuthModel>(
            model: userModel,
            child: ScopedModelDescendant<AuthModel>(
                builder: (modelContext, child, model) => userModel.isAuth ? const CleaningsForm() : const LoginPage())  // ScopedModelDescendant
    ));  // ScopedModel, MaterialApp
  }

  Future<bool> isAuth() async{
    return userModel.isAuth;
  }
}
```

Figura 50. Ejemplo de MyApp: scopped model permite refrescar al hijo cuando el modelo ejecuta el metodo notify()

Una vez que se ha lanzado MyApp y se crea la instancia de la primera pantalla de la aplicación, es posible navegar entre pantallas utilizando la clase Navigator. Esta clase incluye métodos como push, que permite cambiar de ruta conservando la pila de rutas, y pushReplacement, que permite cambiar de ruta eliminando la pila de rutas anteriores. El método push es particularmente útil cuando se desea conservar la pila de rutas, ya que permite, mediante el método pop, volver a la pantalla anterior.

Finalmente, es posible crear y personalizar todo tipo de widgets en Flutter. Uno de los más utilizados es Container, que permite crear contenedores con un tamaño específico o que ocupen todo el ancho o alto de la pantalla. Además, se pueden aplicar estilos a un contenedor utilizando DecoratedBox, que permite añadir un color de fondo, bordes, colores de borde, imágenes de fondo, degradados, entre otros. Del mismo modo, es posible agregar un widget hijo dentro del contenedor para cargar otros elementos en su interior.



```
: ListView.builder(
    shrinkWrap: true,
    scrollDirection: Axis.vertical,
    itemCount: cleanings.length,
    itemBuilder: (context, index) {
      return CleanScanCard(
        tittle: index == 0 ? user.currentUser!.name : null,
        subTittle: cleaningService.getSpaceName(
            user.spacesList, cleanings[index]),
        date: cleanings[index].date,
        hour: cleanings[index].hour,
        onTap: () {
          Navigator.pushReplacement(
              context,
              MaterialPageRoute(
                builder: (context) => NewCleaning(
                  currentCleaning: cleanings[index],
                  spaceId: '',
                  // initialMessageModal: '',
                ),  // NewCleaning
              ));  // MaterialPageRoute
        },
        iconButton: cleanings[index].photos.isNotEmpty
            ? IconButton(
                icon: const Icon(Icons.image),
                onPressed: () {
                  showDialog(
                    context: context,
                    barrierDismissible: false,
                    builder: (_) => ImagesModal(
                        cleaning: cleanings[index]),  // ImagesModal
                  );
                },
              )  // IconButton
            : null,
      );  // CleanScanCard
    },
  ),  // ListView.builder
```

Figura 51. Ejemplo de Listview en la pantalla de limpiezas

6.7 instalación de paquetes de terceros

Una de las grandes virtudes de Flutter es la amplia comunidad y el sólido soporte que tiene detrás. Gracias a ello, es posible encontrar paquetes útiles en la página oficial https://pub.dev/, lo que permite reducir significativamente el tiempo de desarrollo.

Estos paquetes se deben definir en el archivo pubspec.yml, ubicado en la raíz del proyecto. Algunos de los paquetes utilizados en CleanScan son:

- **calendar_Date_picker2**: Es una biblioteca que permite crear calendarios para seleccionar una fecha. Es altamente personalizable, lo que permite cambiar los colores de todas sus funcionalidades y elegir una fecha o un rango de fechas.
- **dropDownButton2**: Este paquete ofrece un selector fácil de implementar y personalizable.
- **carouselSlider2**: Permite crear un widget que contiene una o varias imágenes, y navegar entre ellas deslizando a los lados. Se ha utilizado para visualizar las imágenes de las limpiezas.
- **shared_preferences**: Es una biblioteca que permite guardar información en el almacenamiento del dispositivo, siendo muy útil para implementar mecanismos de sesión permanente.
- **qr_flutter**: Permite escanear todo tipo de códigos QR y leer la información contenida en ellos.

```yaml
dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.2
  get_it: ^7.6.0
  flutter_platform_widgets: ^6.0.2
  flutter_native_splash: ^2.3.2
  flutter_launcher_icons: ^0.13.1
  permission_handler: ^10.4.3
  scoped_model: ^2.0.0
  app_tracking_transparency: ^2.0.4
  firebase_core: ^2.8.0
  firebase_auth: ^4.19.0
  cloud_firestore: ^4.16.0
  package_info_plus: ^4.0.2
  uuid: ^3.0.7
  calendar_date_picker2: ^0.5.3

  firebase_storage: ^11.7.0
  flutter_native_image: ^0.0.6+1
  path_provider: ^2.1.0
  universal_html: ^2.2.3
  multiple_image_camera: ^0.0.2
  image_picker: ^1.0.7
  intl: ^0.19.0
  dropdown_button2: ^2.3.9
  qr_flutter: ^4.0.0
  flutter_barcode_scanner: ^2.0.0
  carousel_slider: ^4.2.1
  shared_preferences: ^2.2.2
  carousel_indicator: ^1.0.6
  cached_network_image: ^3.3.1
```

Figura 52. Dependencias del fichero pubspec.yaml de CleanScan

7. Pruebas

En un proyecto, una de las fases más importantes es la de pruebas, ya que, en caso de haber errores graves, se podría entregar un producto final que no cumpla con los estándares de calidad. Existen diferentes tipos de pruebas: las pruebas automatizadas, que se implementan en código, y las pruebas manuales, que se realizan comprobando que los resultados coinciden con lo esperado.

En CleanScan, se ha optado por realizar las pruebas de manera manual. Esta elección permite no solo verificar que el comportamiento de la aplicación es el esperado, sino también asegurarse de que la interfaz visual cumple con los requisitos establecidos.

En primer lugar, se realiza una prueba para verificar el funcionamiento de la funcionalidad de recuperación de contraseña. En esta prueba, se introduce un correo electrónico válido y se envía la solicitud de recuperación. Tras enviar la solicitud, el usuario debería recibir un enlace en la bandeja de entrada de su correo electrónico, el cual le permitirá seguir los pasos necesarios para restablecer su contraseña.

Figura 53. Captura que valida la primera prueba

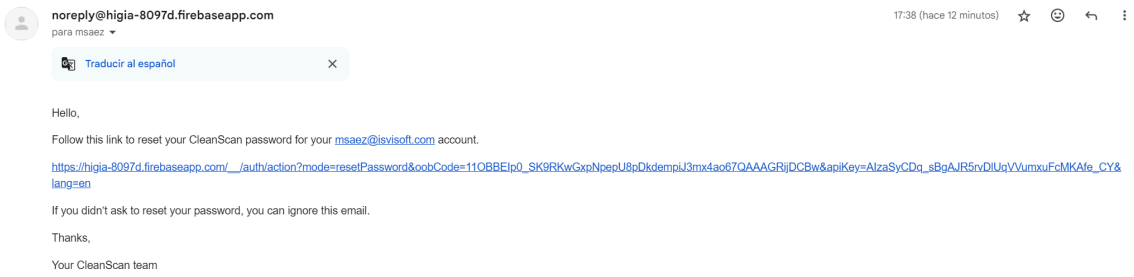


Figura 54. Captura que valida la primera prueba

Como se puede observar el requisito queda satisfecho.

La segunda prueba se centra en verificar el correcto funcionamiento del inicio de sesión. Para ello, se rellenan los campos con las credenciales del usuario, se activa la opción de "mantener sesión activa" y se procede a iniciar sesión.

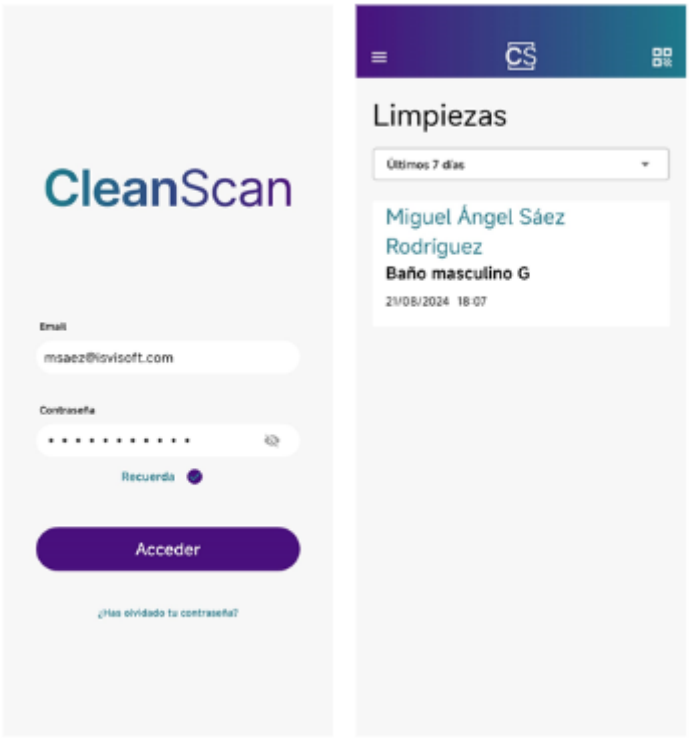


Figura 55. Captura que valida la segunda prueba

Una vez que la sesión se inicia satisfactoriamente, se muestra una lista de limpiezas ejecutadas en los últimos siete días. Posteriormente, se cierra y vuelve a abrir la aplicación para comprobar que la opción de "mantener sesión activa" funciona correctamente. Tras realizar estas pruebas, se verifican los requisitos UR-R01 y UR-R02.

La tercera prueba se enfoca en escanear un código QR, ejecutar una limpieza en un día específico y luego buscar dicha limpieza utilizando la opción de selector personalizado en la pantalla de limpiezas.

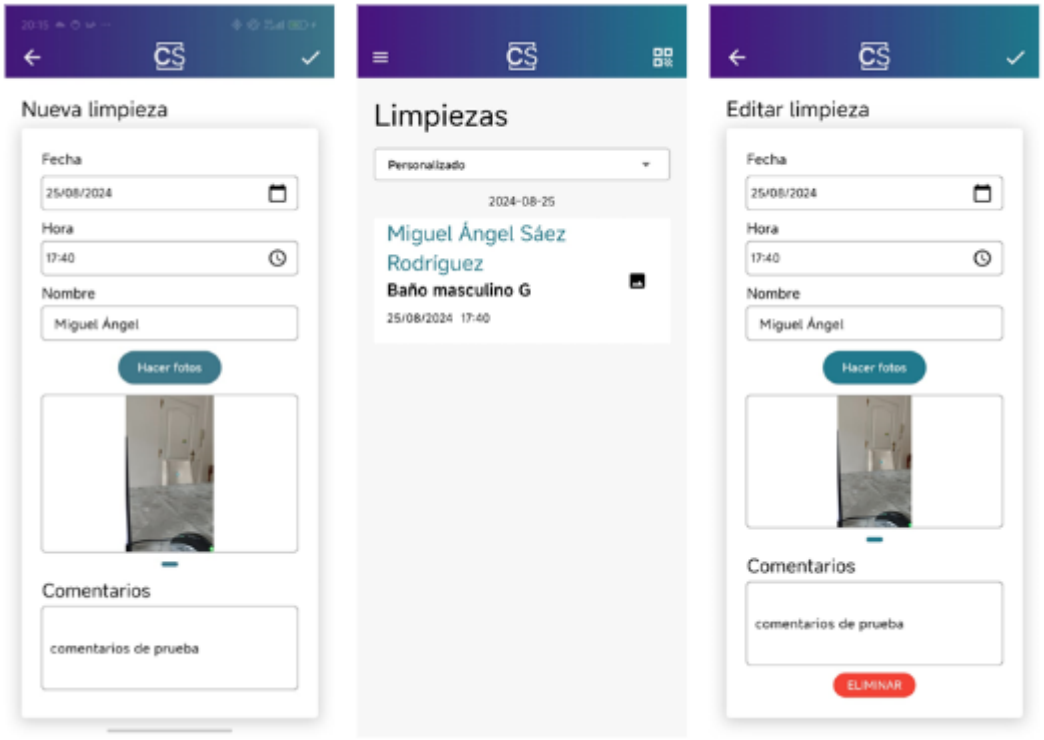


Figura 56. Captura que valida la tercera prueba

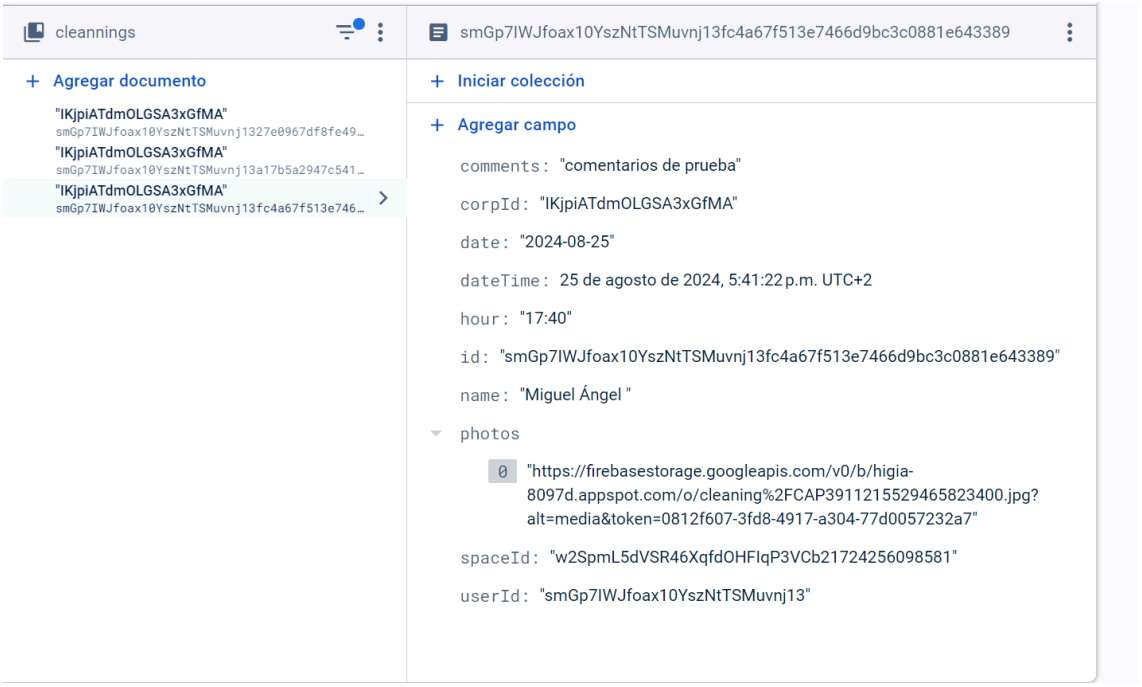


Figura 57. Captura que valida la tercera prueba

Después de realizar estas pruebas, se puede confirmar el correcto funcionamiento de los requisitos: UR-R01, UR-R02, UR-R03, CR-R01, CR-R02, CR-R03, CR-R04, CR-R05, CR-R06, CR-R07 y CR-R08.

La cuarta prueba consiste en desplegar el menú lateral, navegar a la pantalla de turnos y visualizar los turnos existentes.

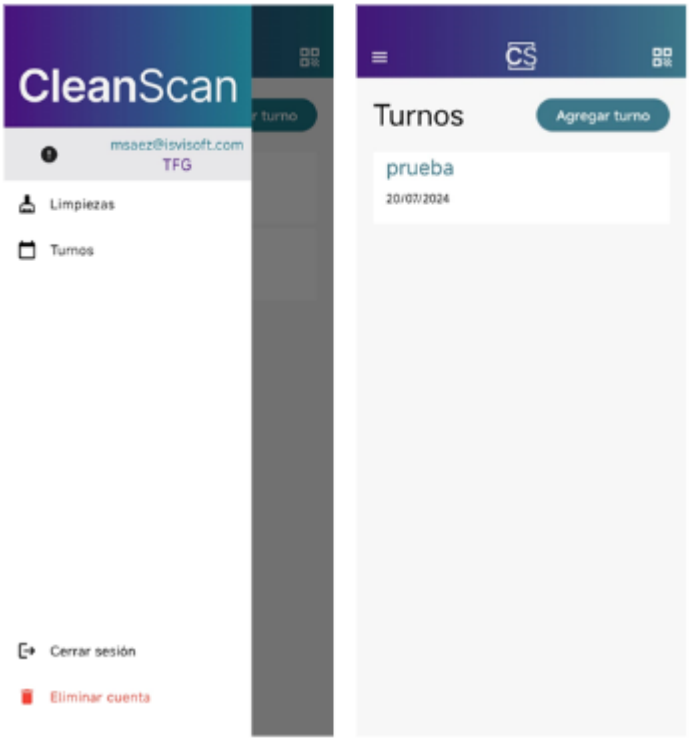


Figura 58. Captura que valida la cuarta prueba

Después de realizar esta prueba, se confirma el correcto funcionamiento del requisito SR-R01.

La quinta prueba consiste en agregar un turno, verificar que se ha agregado correctamente y proceder a editarlo.

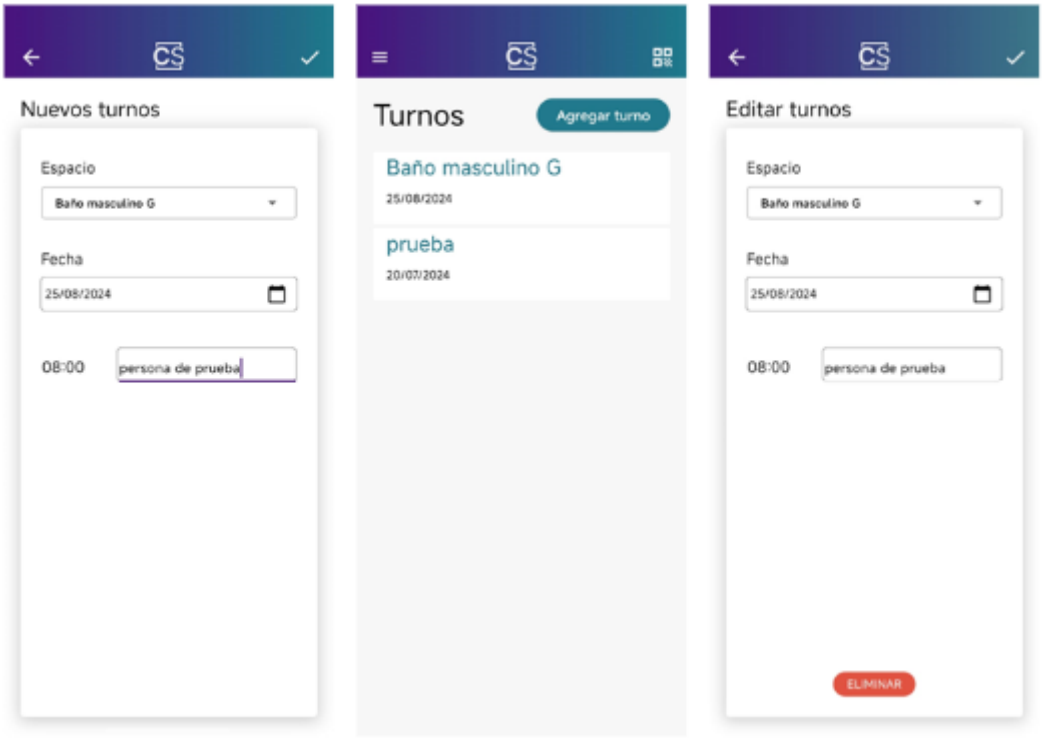


Figura 59. Captura que valida la quinta prueba

Tras completar la prueba, se confirman los siguientes requisitos: UR-R08, UR-R09, UR-R11, SR-R02, SR-R03, SR-R04, SR-R05 y SR-R06.

Finalmente, la última prueba consiste en cerrar sesión y eliminar la cuenta. Una vez completada la prueba, se verifican los requisitos: UR-R04 y UR-R05.

8. Despliegue

En el proceso de desarrollo de CleanScan, se propone realizar el despliegue de la aplicación tanto en la tienda de Android como en la de iOS. Aunque existen otras tiendas, como las de Samsung o Huawei, se ha decidido realizar el despliegue en Google Play y App Store debido a la alta aceptación y alcance que tienen estas plataformas.

Dado el costo asociado al lanzamiento de una aplicación, el proceso de despliegue en Android se aborda de manera informativa en este trabajo. Por otro lado, el despliegue de la aplicación en iOS se propone como trabajo futuro, debido a la falta de disponibilidad de un dispositivo Mac en el momento del despliegue.

Como primer paso, es necesario registrarse en la consola de Google Play[3] (Google Play Console). Se debe realizar un pago único de 25 dólares para acceder, lo cual otorga acceso vitalicio a la consola.

Una vez completados todos los datos de registro, es el momento de crear la aplicación. En el panel de control, se debe introducir el nombre de la aplicación, el idioma predeterminado, seleccionar si la aplicación será un juego o una app, y especificar si será gratuita o de pago. En el caso de CleanScan, será una aplicación gratuita. Además, es necesario aceptar las declaraciones requeridas.

[3] Enlace a la página oficial de Google Play Console: “ https://play.google.com/intl/es/console/about/ ".

Figura 60. Imagen que muestra un ejemplo de como seria el apartado de creación de una aplicación en Google Play Console

A continuación, se solicitará una gran cantidad de información para completar la ficha principal de Play Store. En esta sección, se deben introducir detalles como el nombre de la aplicación, una breve descripción, el ícono de la app, una portada, y las capturas de pantalla que se mostrarán en la tienda.

Llegado este punto, es el momento de subir la aplicación. Para ello, es necesario que las versiones de la app estén firmadas, lo cual implica generar un archivo Bundle/APK firmado.

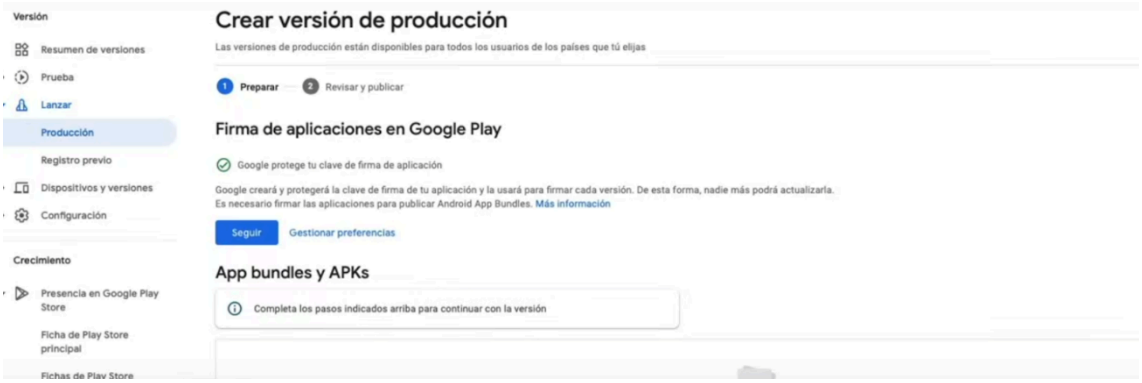


Figura 61. Panel de creación dé versión de la aplicación en Google Play Console

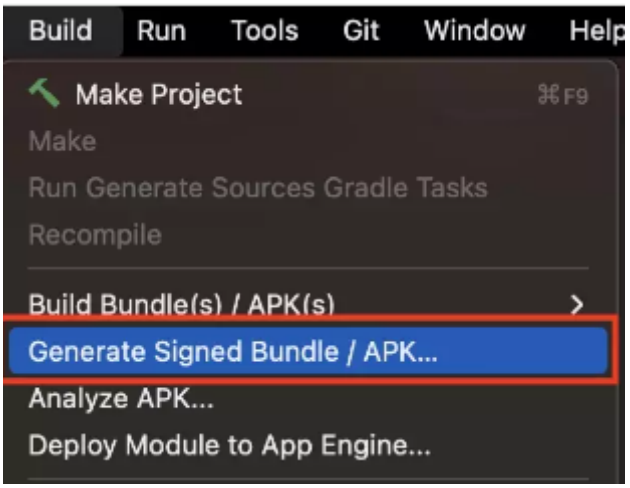


Figura 62. Menú que muestra cómo crear un Bundle/APK firmado

Finalmente, se debe arrastrar la aplicación al contenedor correspondiente y pulsar "Revisar versión". En este último paso, se configurarán aspectos finales de la aplicación, como definir el público al que va dirigida, la información de contacto, los países y regiones en los que estará disponible, entre otros. Una vez configurado todo, se puede lanzar la aplicación a producción.

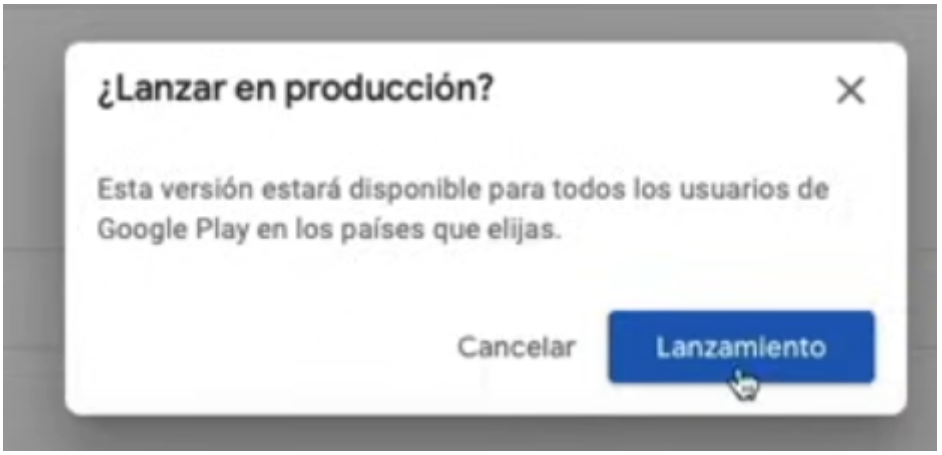


Figura 63. Paso final para lanzar la aplicación en la Play Store

9. Trabajo a futuro

Como trabajo a futuro, se proponen tres objetivos: lanzar la aplicación en la tienda de iOS, realizar una campaña de marketing para captar clientes y añadir nuevas funcionalidades.

Para el despliegue en iOS, es necesario adquirir un Mac de una de las últimas generaciones, debido a la alta capacidad de cómputo que requiere compilar y ejecutar una aplicación. Se estima un presupuesto de 2000 euros para la compra del equipo y así poder alcanzar este objetivo.

Por otra parte, dado que CleanScan es una aplicación poco conocida y con escasa confianza en el sector, es esencial realizar una campaña de marketing para captar nuevos clientes, preferiblemente en la zona local (Valencia), con el fin de extenderse a nivel comunitario y, finalmente, encontrar un nicho a nivel nacional. El presupuesto para esta campaña es desconocido, ya que se debe llevar a cabo una exhaustiva investigación entre las diferentes empresas de marketing para decidir la mejor opción.

Finalmente, como último objetivo, aunque con menor prioridad, es crucial mantener a los clientes obtenidos ofreciendo nuevas funcionalidades, lo que dará una sensación de mejora continua y permitirá mantener y mejorar las funcionalidades ya ofrecidas.

10. Conclusiones

Tras meses de planificación y desarrollo del proyecto, se han alcanzado varios objetivos de manera satisfactoria; sin embargo, otros, aunque logrados, no han sido completados de la mejor manera posible.

La conceptualización de la herramienta ha sido un éxito. Se definieron claramente las funcionalidades principales y sus beneficios, lo cual fue clave para mantener una dirección clara en la toma de decisiones durante todo el proceso.

La planificación y definición de requisitos, aunque consistentes, podrían haber tenido una base más sólida en cuanto a la definición de metas temporales. Al tratarse de un proyecto desarrollado por una única persona, no se consideró necesario invertir tiempo en la organización y planificación de tareas. No obstante, para mantener una referencia temporal clara, habría sido muy beneficioso establecer tareas con fechas límite. Por otro lado, la definición de requisitos fue un éxito, ya que se establecieron de manera clara y precisa, lo que facilitó el proceso de diseño e implementación de las funcionalidades.

La creación de los diseños fue una decisión muy acertada. Utilizar Figma para realizar los mock-ups permitió crear una versión preliminar muy similar a la esperada y a la finalmente alcanzada en el proceso de implementación. Además, la rápida curva de aprendizaje de esta herramienta permitió ahorrar tiempo valioso, que se destinó al desarrollo de la aplicación.

El desarrollo de la aplicación ha sido, sin duda, la fase más desafiante del proyecto. Se tuvo que enfrentar a un lenguaje completamente nuevo, como Dart, y a tecnologías desconocidas como Firebase, Git, Android Studio y Flutter. Sin embargo, gracias a los conceptos sobre programación adquiridos durante el grado, la curva de aprendizaje fue más rápida de lo esperado. No obstante, en las primeras fases de la implementación, se cometieron algunas malas prácticas de programación debido al desconocimiento de ciertas tecnologías.

Las pruebas de la aplicación fueron la fase más afectada por la falta de tiempo. Debido a los inconvenientes mencionados anteriormente en la fase de implementación, las

pruebas se limitaron a comprobar manualmente que los resultados eran los esperados y que cumplían con todos los requisitos definidos.

Finalmente, se explicó el proceso de despliegue en Android, el cual no se pudo realizar por motivos económicos, y se detalló un manual a seguir en el futuro para asegurar el correcto funcionamiento de la aplicación en el mercado.

Bibliografía

[1] Requeridos Blog, “Requerimientos Funcionales y No Funcionales, ejemplos y tips”. https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a Cita consultada el 27 de julio de 2024.

[2] IBM, “Definición de casos de uso”. https://www.ibm.com/docs/es/product-master/12.0.0?topic=processes-defining-use-cases Cita consultada el 21 de julio de 2024.

[3] Vollmer N, “Artículo 32 UE RGDP “Seguridad del tratamiento"”. https://www.privacy-regulation.eu/es/32.htm Cita consultada el 28 de julio de 2024.

[4] Google developers ,“Cloud Storage para Firebase”. https://firebase.google.com/docs/storage?hl=es-419 Cita consultada el 28 de julio de 2024.

[5] Arizbé Ken, “Arquitectura de software: ¿Qué es y qué tipos hay?”. https://www.gluo.mx/blog/arquitectura-de-software-que-es-y-que-tipos-hay Cita consultada el 2 de Agosto de 2024

[6] Junta de Andalucía, “Funcionalidades de la capa de persistencia”. https://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/13#

[7] Universidad Europea, “¿Qué es un prototipo y para qué sirve”. https://creativecampus.universidadeuropea.com/blog/que-es-prototipo/ Cita consultada el 29 de Agosto de 2024

ANEXO 1

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**			x	
ODS 2. **Hambre cero.**			x	
ODS 3. **Salud y bienestar.**	x			
ODS 4. **Educación de calidad.**			x	
ODS 5. **Igualdad de género.**		x		
ODS 6. **Agua limpia y saneamiento.**		x		
ODS 7. **Energía asequible y no contaminante.**			x	
ODS 8. **Trabajo decente y crecimiento económico.**	x			
ODS 9. **Industria, innovación e infraestructuras.**			x	
ODS 10. **Reducción de las desigualdades**.		x		
ODS 11. **Ciudades y comunidades sostenibles**.	x			
ODS 12. **Producción y consumo responsables.**			x	
ODS 13. **Acción por el clima.**	x			
ODS 14. **Vida submarina.**			x	
ODS 15. **Vida de ecosistemas terrestres.**			x	
ODS 16. **Paz, justicia e instituciones sólidas.**			x	
ODS 17. **Alianzas para lograr objetivos.**			x	

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

CleanScan guarda una estrecha relación con algunos Objetivos de Desarrollo Sostenible (ODS), ya que, en su esencia, es una aplicación diseñada para optimizar procesos, los cuales, al estar relacionados con la higiene, suelen utilizar productos contaminantes. Gracias a CleanScan, se puede reducir el número de limpiezas necesarias en un local, además de obtener estadísticas de género en la plantilla. A continuación, se detalla la relación de CleanScan con los ODS más relevantes:

- **ODS 3, Salud y bienestar:** Propone garantizar una vida sana y promover el bienestar para personas de todas las edades. La aplicación desarrollada contribuye a mantener una higiene beneficiosa para la salud, lo que a su vez mejora el bienestar general.

- **ODS 8, Trabajo decente y crecimiento económico:** Uno de los objetivos de este ODS es lograr un crecimiento económico sostenible. Gracias a la optimización de los procesos de limpieza, es posible lograr la máxima limpieza de los locales utilizando la mínima cantidad de recursos.

- **ODS 11, Ciudades y comunidades sostenibles:** Este ODS promueve que las ciudades y comunidades sean inclusivas y sostenibles. Como se ha mencionado, CleanScan permite reducir los recursos utilizados en el sector de la limpieza, ayudando a construir comunidades más sostenibles.

- **ODS 13, Acción por el clima:** Gracias a la optimización de los procesos de limpieza, es posible reducir el uso de productos contaminantes, contribuyendo a mantener la higiene de los locales con un menor impacto ambiental.

En conclusión, CleanScan tiene una estrecha relación con los objetivos que buscan reducir la contaminación y hacer del mundo un lugar más habitable.