Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE VALLADOLID

Grado en Ingeniería Informática

Mención de Ingeniería de Software

Desarrollo de aplicación multiplataforma para la prevención de riesgos laborales con Xamarin

---

Alumno: David González Cosgaya

Tutor: Blas Torregrosa García

Dedicado a
mi familia y a mis amigos por el apoyo y afecto que siempre me han brindado

Agradecimientos

A mi familia por haberme apoyado siempre y haber cuidado de mí.

A mis amigos que siempre estan ahí cuando les necesito y que me apoyan en todo lo que haga.

A los compañeros que he conocido a lo largo de la carrera y con los que he podido trabajar para enriquecer mis conocimientos.

A mi tutor Blas por darme la oportunidad de realizar este trabajo.

Summary

The main purpose of this project it´s to create a cross-platform application that allows the user to produce job inspections in a simple and faster way than the original methods. This is an important goal, because to create an inspection is necessary to gather to much information, and because of that the inspector must save that amount of information with the objective of create it later. In order to simplify that task the creation of an application that allows the user to save that information in an easiest and comfortable way may to the users to do their work more productively and quickly than in the other ways.

The idea of creating a cross-platform app comes from that now a days everyone has an smart-phone. So it´s really common to use apps to perform everyday tasks, and so it is to improve production on job. Because of that creating an application that works on the main mobile devices could help people to do their tasks the best way possible.

The app will allow to the users to create different inspections as well as to get the inspections that were created previously. It would allow them too to create templates that could be used to create new inspections. The information of each user will be stored in his personal account so in that way only they could access to it.

To achieve that goal in this project is going to be used the framework Xamarin Forms, owned by Microsoft. The main language in which it works is **C#**. This is an object-oriented language created on the year 2000 by Microsoft. And for persistent storage is going to be used the online database Firebase from Google.

Resumen

El objetivo principal de este proyecto es crear una aplicación multiplataforma que permita al usuario realizar inspecciones de trabajo de una manera más sencilla y rápida que con los métodos tradicionales. Este es un objetivo importante, porque para crear una inspección es necesario reunir mucha información, y por esto el inspector debe guardar dicha información con el objetivo de crear la inspección posteriormente. Para simplificar esta tarea, la creación de una aplicación que permita al usuario guardar esa información de una manera mas fácil y cómoda puede permitir a los usuarios realizar su trabajo de una forma más productiva y rápida que de las otras formas.

La idea de crear una aplicación multiplataforma viene de que hoy en día todo el mundo dispone de un teléfono inteligente. Por lo que es muy común utilizar aplicaciones para realizar tareas cotidianas, así como para mejorar la productividad en el trabajo. Por esto, crear una aplicación que funcione en los principales dispositivos móviles puede ayudar a la gente a realizar sus tareas de la mejor manera posible.

La aplicación permitirá a los usuarios crear diferentes inspecciones así como obtener las inspecciones que fueron creadas previamente. Les permitirá también crear plantillas que podrían usarse para crear nuevas inspecciones. La información de cada usuario será almacenada en su cuenta personal de forma que solo ellos puedan acceder a la misma.

Para lograr este objetivo en este proyecto se va a utilizar el framework Xamarin Forms, propiedad de Microsoft. El lenguaje principal en el cual funciona es **C#**. Este es un lenguaje orientado a objetos creado en el año 2000 por Microsoft. Y para el almacenamiento persistente se va a utilizar la base de datos online de Google Firebase.

Índice general

Índice de figuras

Índice de cuadros

Parte I

Memoria del Proyecto

Capítulo 1

Descripción del proyecto

1.1. Introducción

Todos los puestos de trabajo tienen asociados una serie de riesgos, ya sea bien un trabajo de oficina o en una obra podemos encontrar situaciones o características que pueda perjudicar a los trabajadores en dicho entorno. Estas son de diversa índole, como pueden ser accidente físicos como lesiones o heridas, o bien algunos no superficiales como podría ser el estrés. Esto implica que en todos los puestos de trabajo se deba de llevar a cabo la prevención de riesgos con el fin de garantizar unas condiciones laborales mínimas para todos los trabajadores.

La principal herramienta que se emplea para esta tarea son las inspecciones, las cúales se basan en un análisis mediante la observación directa de los entornos de trabajo para determinar los posibles riesgos o peligros de los mismos [5]. Estos análisis se realizan en base a una serie de parámetros estipulados.

Por esto este proyecto se podría integrar en cualquier empresa u organización, debido a lo enunciado previamente todas deben de gestionar la prevención de riesgos y mediante el uso de la aplicación desarrollada esta tarea se podría ver simplificada.

1.2. Objetivos del trabajo

El principal objetivo de este trabajo se basa en el desarrollo de una aplicación multi-plataforma que permita la realización rápida y sencilla de inspecciones laborales.

Por tanto los objetivos que debe cumplir esta aplicación es permitir la realización de dichas inspecciones en base a los distintos riesgos laborales y parámetros que se deben de tener en cuenta para las mismas, permitiendoles de la misma manera poder consultar facilmente aquellas inspecciones que se han realizado previamente. En este aspecto encontramos una lista muy extensa lo que dificulta la tarea de realizar dichas inspecciones sin tener sistemas especializados en ello.

Otro objetivo que se debería de alcanzar sería la posibilidad de incluir desde la misma aplicación nuevas plantillas en base a las cuales poder realizar nuevas inspecciones.

Y los principales objetivos desde un punto de vista personal se basarían en realizar un proceso de desarrollo real siguiendo todas las fases que este implica así como la posibilidad de en dicho proceso aprender un lenguaje nuevo como es en este caso **C#** y un framework nuevo como **.NET**.

1.3. Entorno de aplicación

Este trabajo no es si no la continuación de un TFM que tenía como objetivo el mismo pero en este caso mediante la utilización de Flutter para el desarrollo del proyecto.

Con respecto al ámbito o sector al que se destina esta aplicación podemos encontrar que a día de hoy existen aplicaciones cuya función se basa en evaluar las condiciones o medios de trabajo de distintos puestos y sectores pero con el defecto de estar enfocadas unica y exclusivamente a un pequeño sector dentro del ámbito de la prevención. Esto supone un problema debido a 4 principales causas:

1. El hecho de que las aplicaciones esten tan enfocadas a un determinado ámbito obligaría a los inspectores a tener varias de ellas para poder realizar su trabajo existiendo la posibilidad de que no tengan todas la misma fiabilidad o no cumplan los mismos criterios de calidad.

2. Otro problema que esto supondría sería el hecho de que para algunos trabajos o puestos no exista la aplicación específica lo que supondría una perdida de eficiencia obligando al inspector en cuestión a cambiar su mecánica de trabajo para un único proceso.

3. Al tener que emplear distintas aplicaciones para una misma inspección la información relacionada con esta estaría dividida entre las distintas aplicaciones que se empleasen, lo que dificultaría el acceder a dicha información en caso de necesidad.

4. La prevención no es algo estático en el tiempo, con el surgimiento de nuevos trabajos, tareas, herramientas... surgen nuevos riesgos que deben ser tratados. Lo que refuerza la necesidad de una aplicación que permita un cierto grado de actualización con respecto al ámbito del servicio que se suministra.

Esto refuerza lo enunciado en el punto anterior de la necesidad de una aplicación que reuna todo el conocimiento existente con respecto al ámbito de la prevención con el objetivo de facilitar y agilizar el trabajo de los inspectores mas aún tratandose de un ámbito como este pues es necesario en todos los trabajos de una forma u de otra.

Capítulo 2

Metodología

En este capítulo se va a profundizar tanto en la metodología que se va a seguir para el desarrollo del trabajo así como las principales herramientas que se van a emplear para el desarrollo de la misma.

2.1. Proceso de desarrollo

En el desarrollo del proyecto se va a seguir una metodología iterativa, siguiendo un esquema base en el desarrollo de software, incluyendo este las fases de análisis, diseño e implementación. Esto no implica que en caso necesario se pueda volver a una fase previa para modificar, añadir o eliminar si fuera necesario algunos requisitos de los planteados en inicio. Una vez transcurrido el desarrollo se realizarán test con el fin de encontrar bugs y en caso de encontrarlos poder subsanarlos.

En la Figura 2.1 se puede observar un esquema que muestra el desarrollo base del proyecto.

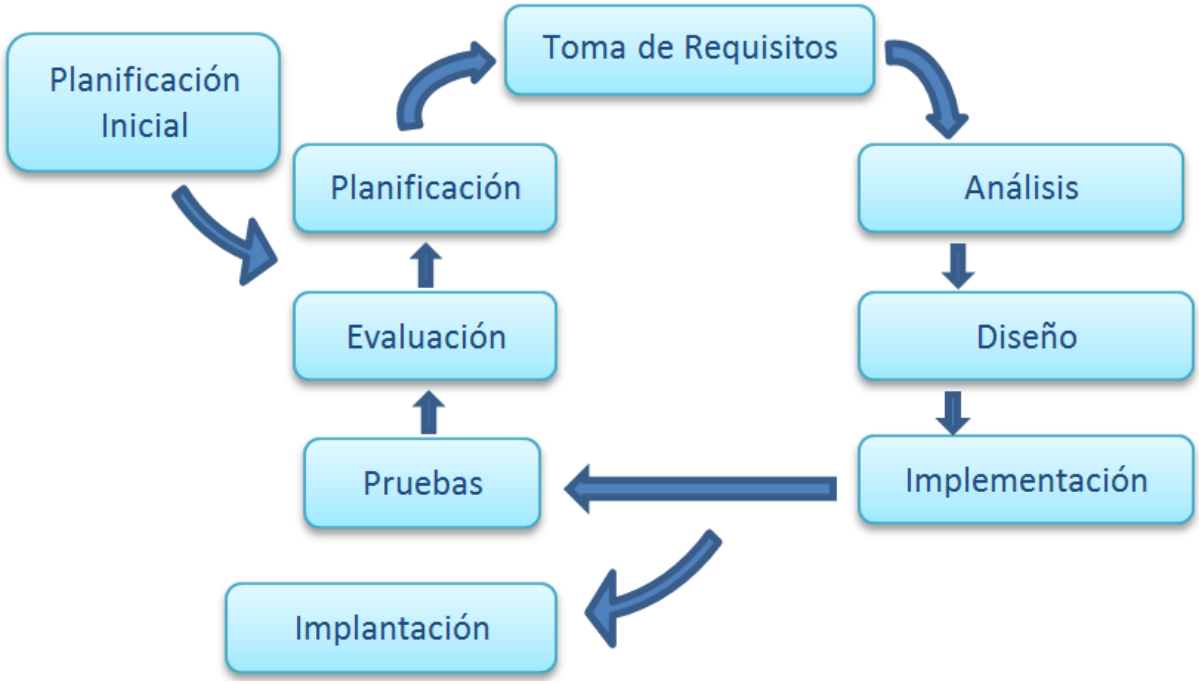

Figura 2.1: Modelo de desarrollo iterativo

En el desarrollo de este proyecto se va a emplear el paradigma de orientación a objetos, la razón principal para esto es que el lenguaje que se va a emplear **C#** es un lenguaje basado en dicho paradigma.

La metodología que se va a emplear es **SCRUM**, debido principalmente a las ventajas que aportan las metodologías ágiles permitiendo realizar cambios sobre lo ya realizado manteniendo la documentación necesaria para estos proyectos. Con respecto a los roles existentes en dicha metodología yo hare de SCRUM Master, PO y equipo de desarrollo planificando los distintos Sprints en función del propio alcance que vea adecuado.

2.2. Herramientas utilizadas

Respecto a las herramientas empleadas para el desarrollo del proyecto se indicarán a continuación los principales mótivos por los cuáles han sido seleccionadas cada una de ellas.

Como se indicó previamente el lenguaje de programación elegido para el desarrollo de este proyecto será **C#**, a parte de dicho lenguaje el IDE que se emplea también permite el desarrollo en **F#**. Se ha optado por la primera opción debido principalmente al paradigma en el que se fundamenta, la orientación a objetos resulta mas adecuada para un desarrollo multiplataforma como el que se plantea en este proyecto, otro factor que

favorece la elección de dicho lenguaje en contra parte con el segundo es su gran similitud a Java, con el cuál se trabaja constantemente a lo largo del grado lo que asienta unas bases que pueden producir unos mejores resultados finales con respecto al software desarrollado.

Con respecto a lo dicho en el parrafo anterior el IDE seleccionado para el desarrollo de esta práctica es Visual Studio Code, y se ha seleccionado dicho entorno debido principalmente a que el framework que se va a emplear para el desarrollo, Xamarin Forms, es propiedad de Microsoft por lo que emplear su principal IDE facilita mucho la tarea y permite disponer de las versiones más recientes. Pese a que el desarrollo se realizará en Visual Studio Code, se dispone tambien de Android Studio y XCode principalmente para poder disponer de los correspondientes emuladores de Android y iOS con el fin de poder probar la aplicación correctamente durante el desarrollo.

Respecto al gestor de base de datos que se va a emplear Firebase, debido principalmente a la naturaleza multiplataforma de la aplicación. Al tratarse de una aplicación que se va a poder emplear en dispositivos móviles que el almacenamiento se gestione en un servidor en la nube permite reducir notablemente el espacio de las aplicaciones que por su propia naturaleza lo beneficia. La principal dificultad que presenta esta elección a pesar de ser la más adecuada y lógica es el hecho de que se trate de una base de datos no relacional en la cuál no dispongo de tanta experiencia.

Con respecto a las herramientas que se van a emplear para el desarrollo de la documentación del proyecto cabe destacar Overleaf como editor de LaTex para el desarrollo de la memoria, principalmente por los buenos resultados y la versatilidad que aporta pese a que algunas veces pueda resultar complicado ordenar los elementos del documento como se desee. Mientras que para el desarrollo de los diagramas tanto de análisis como de diseño se empleara la herramienta Astah Proffesional, gracias a la licencia de la que dispongo por la universidad y la experiencia previa en el uso de dicha aplicación para la construcción de distintos tipos de diagramas.

Por último con respecto al sistema operativo con el que se va a trabajar se ha elegido MacOs Big Sur debido a que se trata del sistema que empleo con naturalidad y al que más acostumbrado estoy, además del óptimo rendimiento en el trabajo que permite.

2.3. Arquitectura

Con respecto a la arquitectura de la aplicación, la gran ventaja que aporta Xamarin forms con respecto a otros frameworks es el hecho de que permite reuitilizar entre un 75-85 % del código entre las distintas plataformas en las que se desarrolle. Principalmente encontraríamos 2 secciones diferenciadas en dichas arquitectura: la primera compartida se referiría a la lógica de negocio, los modelos de datos y las interfaces de usuario de las aplicaciones, y por otra parte tendríamos las partes correspondientes al implementación

caracteristica de cada sistema con respecto a los intercambios con BBDD, que como se explicón anteriormente será Firebase. Cada proyecto referente a los distintos sistemas operativos implementará una interfaz de acceso a datos común que se encuentra en el código compartido.

Lo explicado en el párrafo anterior se puede observar gráficamente en la Figura 2.2. Asi podemos ver como la aplicación quedaría devidida en 3 capas principalmente, la última referente a la base de datos se ha separa puesto que se alojará en un servidor remoto.

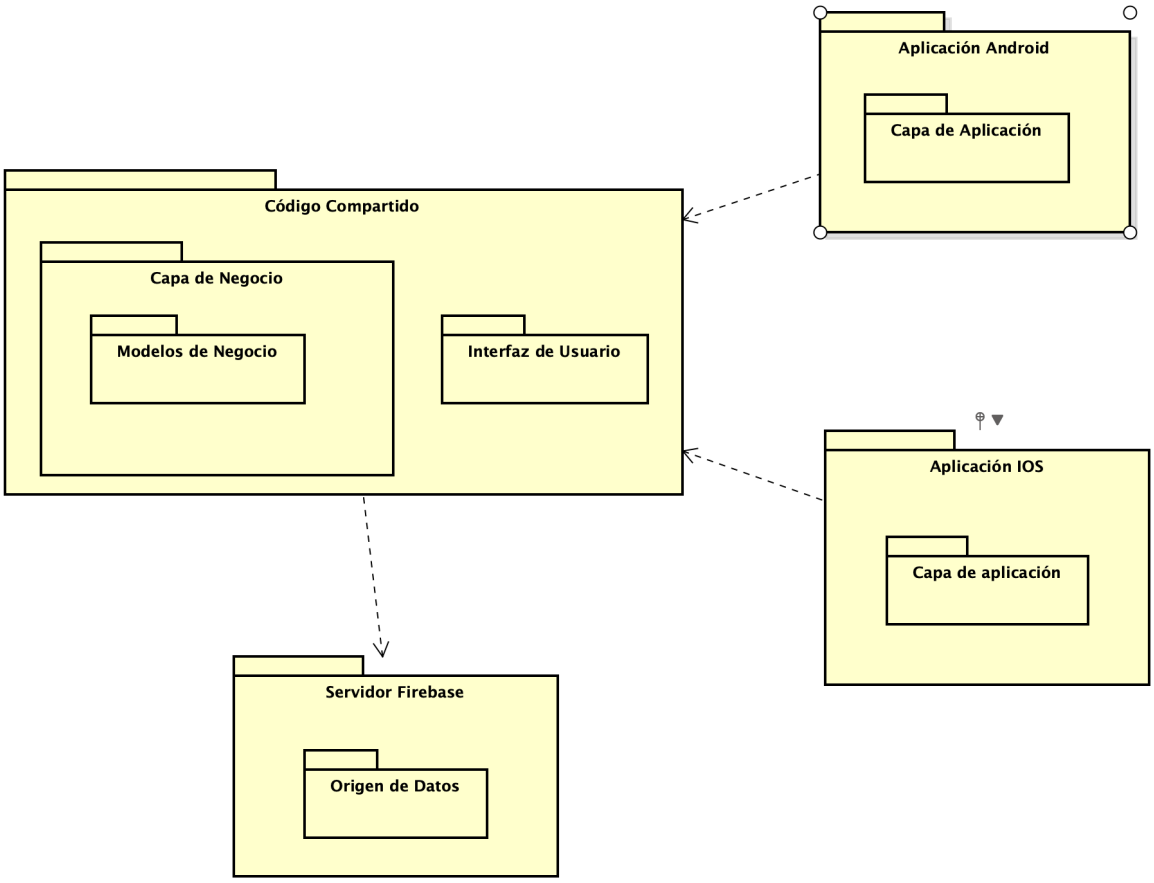

Figura 2.2: Arquitectura base de la aplicación

Capítulo 3

Planificación

Como se indicó en el capítulo anterior, la metodología que se va a emplear en el desarrollo del proyecto será **SCRUM**. Para la organización de trabajo no se han podido emplear tecnologías especificas debido a que me era complicado cuadrar la planificación con las prácticas en empresa.

En este capítulo se dará primero una visión básica de la metodología y se mostrara un calendario de Sprints del mismo.

3.1. SCRUM

SCRUM es lo que se denomina una medodología ágil, una metodología ágil es un conjunto de técnicas de la gestión de proyectos orientadas a generar entregables en un menor periodo de tiempo, sin perder los estándares de calidad en el proceso. Permitiendo de esta misma forma un feedback más constante con el cliente, facilitando así los cambios que se puedan realizar en el producto final con respecto a la idea inicial sin que supongan estos unos costes desorbitados. [12]

SCRUM es un tipo de metodología ágil, muy orientada al trabajo colaborativo y a la autogestión de los equipos incluidos en el desarrollo del mismo. Pese a tratarse de una metodología ágil SCRUM hace uso constante de reuiniones para ver los progresos del proyecto (conocidas como *Daily SCRUM*), que permiten ver en muy poco tiempo, durando la mayoría entre 10-15 minutos, los avances y dificultades que se están produciendo. Estas reuniones generan pequeños informes que condensan la información que se ha obtenido durante las mismas.

En un desarrollo SCRUM existen 3 artefactos principales que son los siguientes:

1. **Backlog del Producto**: que se corresponde con el conjunto de tareas que el cliente desea realizar sobre el proyecto, este artefacto es gestionado por el Product Owner.

2. **Backlog del Sprint**: que se corresponde con el conjunto de tareas que se realizarán en el actual Sprint y del cuál se desea obtener producto.

3. **Incremento**: que se corresponde con el producto final utilizable del sprint en cuestión.

Los artefactos mencionados previamente son importantes en el desarrollo de los Sprints, los cuáles son periodos de tiempo fijos en los que los diferentes equipos deben tratar de completar una cantidad de trabajo previamente acordada. [10]

3.2. Planificación del proceso de desarrollo

La planificación seguida durante el desarrollo del TFG se ha basado en los principios de la Ingeniría de Software, produciendose de esta misma forma las distintas fases que componen los mismos.

Los Sprints que se han realizado se han centrado principalmente en dichas fases, de manera que estos han sido de una duración variable pero en general extensa. Es común que en esta metodología los sprints sean más constantes en duración y no muy extensos pero se ha realizado así para poder avanzar las fases del proceso de una manera mas constante.

Sprint	Fecha de inicio	Fecha de Fin
Sprint 1	01/07/2021	16/07/2021
Sprint 2	17/07/2021	02/08/2021
Sprint 3	03/08/2021	07/08/2021
Sprint 4	08/08/2021	15/08/2021
Sprint 5	16/08/2021	24/08/2021
Sprint 6	24/08/2021	09/08/2021
Sprint 7	10/09/2021	14/09/2021
Sprint 8	15/09/2021	16/09/2021

Como se puede ver en la tabla el primer Sprint se correspondería con el inicio del proyecto y la fase de análisis del mismo. El segundo se centraría en la fase de diseño de este junto con algunas correciones del análisis. La tercera fase se correspondería con el inicio del desarrollo, en concreto con la construcción básica de la aplicación así como la construcción del modelo en el código, preparando también aqui el repositorio remoto en Github. El cuarto se centro en la construcción del login de la aplicación y la conexión inicial con Firebase, que sobre todo en los inicios del desarrollo fue problemática. La quinta fase se centro en el desarrollo de los primeros intercambios de datos con el servicios Cloud Firestore, esta fase es muy extense pues se encontraron numerosos problemas para conseguir realizar las operaciones CRUD con la base de datos. El sexto se corresponde con el desarrollo completo de la aplicación, una vez se conocía como realizar las operaciones mencionadas anteriormente el proceso de desarrollo de la aplicación se vio acelerado al

no surgir nuevos problemas con respecto a esto. El séptimo se corresponde con la realización de cambios en el front-end de la aplicación, así como los test de integración de la misma. Y por último el noveno se corresponde con la adecuación final de la memoria en base a los cambios realizados durante el desarrollo, así como la redacción de la fase de implementación.

Capítulo 4

Conclusiones

En líneas generales se han alcanzado los requisitos establecidos para el proyecto, incluyendo las funciones que podían llegar a resultar más complejas. Si bien es cierto el front-end de la aplicación es el aspecto que más descontento me genera, es posible que con más tiempo y un mayor conocimiento del entorno y del lenguajes podría haber resultado más atractiva.

Como posibles funcionalidades extras que se me ocurren estaría la posibilidad de crear plantillas personales, de forma que unicamente el usuario pueda emplearlas. Otra funcionalidad que sería interesante incluir en un futuro sería el hecho de poder realizar no solo fotografías sino también videos de las inspecciones para tener estos como evidencia. Por último sería interasante incluir una funcionalidad que permita recoger también como evidencias las declaraciones de los involucrados en las inspecciones, tanto accidentados como testigos.

En líneas generales considero que proyecto me ha mostrado en mayor profundidad y de una forma directa como sería todo el proceso completo de desarrollo del software. Las prácticas realizadas en las distintas asignaturas son instructivas pero se centran mucho en una parte del proceso, dejando el resto de partes más de lado. Además el proyecto me ha permitido aprender **C#** así como aprender mucho sobre Firebase y Xamarin.

Parte II

Documentación técnica

Capítulo 5

Análisis

5.1. Requisitos

En esta sección se especificarán los principales requisitos de la aplicación de todos los tipos. Al tratarse como se explicó previamente de una aplicación multiplataforma la aplicación deberá de funcionar indistintamente en los dos sistemas móviles principales hoy en día en el mercado como son Android y iOS. A continuación procedo a redactar una lista con los distintos tipos de requisitos de la aplicación.

Dada la principal funcionalidad de la aplicación los requisitos principales de la misma se centran en la posibilidad y la facilidad para realizar las distintas inspecciones de trabajo que este desee. Así como poder acceder a las mismas y modificarlas si lo encuentra conveniente.

5.1.1. Requisitos Funcionales

Nº	Requisito
RF01	El sistema permitirá al usuario iniciar sesión en la aplicación
RF02	El sistema permitirá al usuario crear una cuenta de usuario
RF03	El sistema permitirá al usuario generar una nueva inspección de trabajo
RF04	El sistema permitirá al usuario generar nuevos cuestionarios de cara a ampliar los que se pueden realizar en las inspecciones
RF05	El sistema permitirá al usuario revisar las inspecciones previamente realizadas desde la misma aplicación
RF06	El sistema permitirá al usuario modificar los valores de las revisiones realizadas previamente
RF07	El sistema permitirá al usuario modificar los datos y la información de su perfil si asi lo desea
RF08	El sistema permitirá al usuario descargar en formato PDF las revisiones que este haya realizado desde la aplicación
RF9	El sistema pemitirá al usuario eliminar una inspección si así lo desea
RF10	El sistema permitirá al usuario cerrar sesión en la aplicación

Cuadro 5.1: Requisitos funcionales de la aplicación

5.1.2. Requisitos no Funcionales

Nº	Requisito
RNF01	La aplicación será intuitiva y fácil de manejar sin necesidad de conocimientos previos
RNF02	La aplicación deberá funcionar indistintamente en los principales sistemas operativos móviles, es decir Android y iOS
RNF03	La aplicación hara uso de Firebase para el almacenamiento persistente

Cuadro 5.2: Requisitos no funcionales de la aplicación

5.2. Casos de uso

A continuación especificare el diagrama de casos de uso del sistema, asi como realizaré una explicación en detalle de los distintos casos de uso que lo componen.

5.2.1. Diagrama de casos de uso del sistema

Como podemos observar en el diagrama de casos de uso del sistema (Figura 5.1) los casos de uso se dividen principalmente en 2 secciones, los primeros que corresponden con la sesión del usuario y el resto con la creación y manipulación de las distintas inspecciones que se realizan en la aplicación.

Como también se puede observar solo se cuenta con un tipo de actor que es el Inspector, en este caso la aplicación tiene un fin muy concreto por lo tanto es lógico que su funcionalidad vaya enfocada unicamente hacia un rol de usuario.

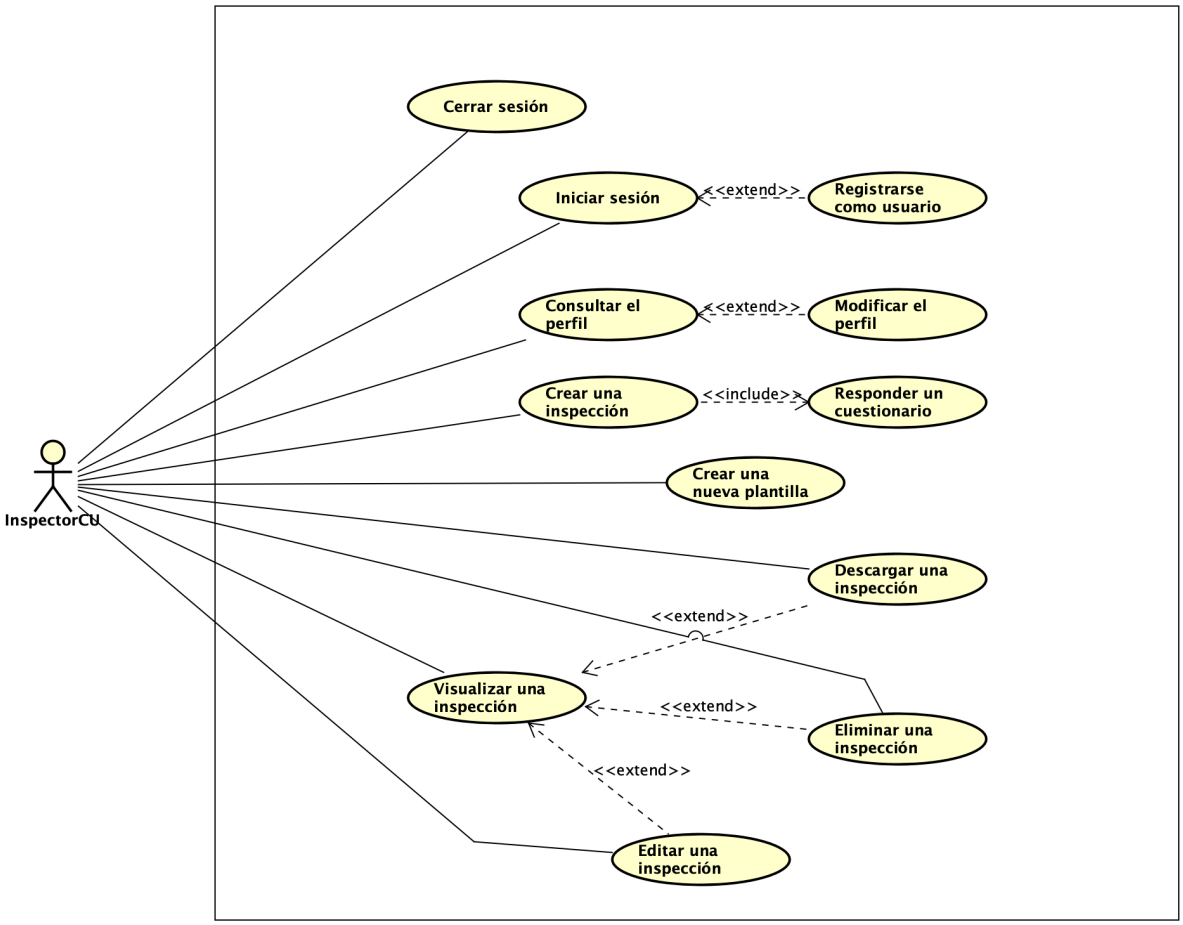

Figura 5.1: Diagrama de casos de uso del sistema

5.2.2. Especificación de casos de uso

En esta sección se van a redactar y explicar en mayor profundidad los casos de uso que componen el sistema.

Nombre e ID del CU	CU-01. Iniciar sesión
Actor	Inspector
Descripción	El usuario emplea las credenciales de su cuenta para acceder a la aplicación.
Precondiciones	PRE-1. El usuario no está identificado en el sistema.
Postcondiciones	POST-1. El usuario accede al sistema.
Flujo normal	FN1 En la pantalla de login el actor introduce su dirección de correo electrónico y su contraseña. FN2 El sistema verifica con la información de la base de datos si las credenciales introducidas son correctas. FN3 Si son correctas el sistema devuelve la información de su cuenta al usuario y le permite acceder a la aplicación.
Flujo alternativo 1	FA3 Si los datos introducidos no son correctos el sistema informará al usuario por pantalla del error con un mensaje y le volverá a solicitar los datos.
Flujo alternativo 2	FA1 Si el usuario no esta registrado en la aplicación deberá de registrarse primero para poder acceder, en este caso el caso de uso queda sin efecto
Excepciones	E1 El usuario ha dejado alguno de los campos requeridos sin rellenar.
Prioridad	Alta

Cuadro 5.3: CU-01. Iniciar sesión

Nombre e ID del CU	CU-02. Registrarse como usuario
Actor	Inspector
Descripción	El usuario rellena una serie de formularios con sus datos para obtener una cuenta y poder acceder a la aplicación.
Precondiciones	PRE-1. El usuario no está registrado en el sistema.
Postcondiciones	POST-1. El usuario pasa a estar registrado en el sistema.

Flujo normal	FN1 En la pantalla de login el actor pulsa sobre el boton Registrarse como usuario. FN2 El sistema muestra al usuario un formulario con una serie de campos que se corresponden a los datos del usuario, todos ellos obligratorios, tales como: nombre, apellidos, DNI, fecha de nacimiento, dirección de correo electrónico y contraseña. FN3 El actor rellena dichos campos con la información correspondiente y presiona el boton Registrarse. FN4 El sistema valida la información introducida por el actor y registra la información en base de datos, tras esto redirecciona al actor a la pantalla de login.
Flujo alternativo 1	FA5 El sistema comprueba que existen campos que no han sido rellenados e informa al usuario del error mediante un mensaje por pantalla.
Flujo alternativo 2	FA5 La contraseña introducida ya esta en uso, por lo tanto el sistema informa al usuario para que introduzca una nueva.
Excepciones	E1 El usuario ha dejado alguno de los campos requeridos sin rellenar.
Prioridad	Alta

Cuadro 5.4: CU-02. Registrarse como usuario

Nombre e ID del CU	CU-03. Cerrar sesión
Actor	Inspector
Descripción	El usuario cierra la sesión en la App.
Precondiciones	PRE-1. El usuario tiene una sesión iniciada en la App.
Postcondiciones	POST-1. El usuario cierra su sesión correctamente.
Flujo normal	FN1 El usuario accede a la sección Mi Perfil de la aplicación y pulsa el boton Cerrar sesión. FN2 El sistema cierra la sesión del usuario y le redirige a la pantalla de login.
Prioridad	Alta

Cuadro 5.5: CU-03. Cerrar sesión

Nombre e ID del CU	CU-04. Consultar el perfil
Actor	Inspector
Descripción	El usuario puede comprobar sus datos desde el perfil de la aplicación.
Precondiciones	PRE-1. El usuario disponde cuenta en la aplicación y tiene su sesión iniciada.
Postcondiciones	POST-1. El usuario puede consultar los datos que corresponden con su perfil.
Flujo normal	FN1 El usuario accede a la sección Mi Perfil en la aplicación. FN2 El sistema recupera la información de usuario y la muestra en un formulario no editable.
Flujo alternativo 1	FA2 El usuario desea modificar su información personal, para ello pulsa sobre el boton Editar perfil y la ejecución continua en el CU-06.
Prioridad	Alta

Cuadro 5.6: CU-04. Consultar el perfil

Nombre e ID del CU	CU-05. Modificar el perfil
Actor	Inspector
Descripción	El usuario modifica la información relativa a su cuenta porque desea cambiar o actualizar algún dato.
Precondiciones	PRE-1. El usuario disponde cuenta en la aplicación y tiene su sesión iniciada, además se encuentra en la pantalla de Mi Perfil.
Postcondiciones	POST-1. El usuario modifica la información referente a su perfil.

Flujo normal	FN1 El usuario que se encuentra en la sección Mi Perfil de la aplicación pulsa sobre el botón Editar Perfil. FN2 El sistema abre un formulario editable con la información actual del usuario. FN3 El usuario modifica los datos correspondientes y pulsa en el botón Guardar. FN4 El sistema verifica y almacena los nuevos datos en la BBDD, y redirecciona al usuario de nuevo a la pantalla de Mi Perfil.
Flujo alternativo 1	FA4 El usuario deja alguno de los campos sin rellenar y pulsa Guardar, en ese caso el sistema informa al usuario del error y no le permite actualizar el perfil.
Excepciones	E1 El actor deja alguno de los campos sin rellenar.
Prioridad	Media

Cuadro 5.7: CU-05. Modificar el perfil

Nombre e ID del CU	CU-06. Crear una inspección
Actor	Inspector
Descripción	El usuario genera una nueva inspección de trabajo que quedará registrada en la aplicación.
Precondiciones	PRE-1. El usuario dispone de cuenta en la aplicación y tiene su sesión iniciada.
Postcondiciones	POST-1. Se genera una nueva inspección de trabajo y se almacena en la cuenta del usuario.

Flujo normal	FN1 El usuario que se encuentra en la pantalla principal pulsa sobre el botón Nueva inspección. FN2 El sistema muestra al usuario un formulario con los datos básicos para crear una inspección. FN3 El usuario rellena los campos que aparecen en dicho formulario y pulsa continuar. FN4 El sistema valida la información introducida y si es correcta muestra un formulario con un listado de las plantillas sobre las que realizar la inspección. FN5 El usuario pulsa sobre una de dichas plantillas. FN6 El sistema le muestra al usuario una nueva pantalla con un formulario para rellenar los bloques correspondientes a dicha plantilla, el caso de uso continua en CU-07. FN7 Cuando el usuario haya terminado de rellenar la nueva inspección pulsa sobre el botón Guardar inspección. FN8 El sistema almacena la nueva inspección en BBDD y devuelve al usuario a la pantalla de inicio.
Excepciones	E1 El actor pulsa continuar sin haber rellenado la información básica de la plantilla. E2 El actor pulsa Guardar inspección sin haber rellenado ninguna de las plantillas previamente.
Prioridad	Alta

Cuadro 5.8: CU-06. Crear una inspección

Nombre e ID del CU	CU-07. Rellenar una plantilla
Actor	Inspector
Descripción	El usuario contesta a las respuestas de los bloques que desee para una determinada plantilla
Precondiciones	PRE-1. El usuario había seleccionado una plantilla de la lista mostrada por la pantalla de "Nueva inspección".
Postcondiciones	POST-1. Se almacenan las respuestas de dicha plantilla y se guardan los datos en BBDD.

Flujo normal	FN1 El usuario selecciona uno de los bloques de la plantilla. FN2 El sistema le muestra al usuario las preguntas de dicho bloque en un nuevo formulario. FN3 El usuario responde a las preguntas mediante las respuestas predefinidas que se le muestran por pantalla, ocasionalmente puede incluir fotos como pruebas. Una vez contestadas las preguntas pulsa el botón Guardar. FN4 El sistema almacena la información del bloque correspondiente y devuelve al usuario a la pantalla con el listado de bloques de la plantilla.
Flujo alternativo 1	FA5 Tras haber respondido a las preguntas y estar de nuevo en la lista de bloques el usuario decide rellenar un bloque más de la inspección, el caso de uso se repite desde el primer paso hasta el último.
Prioridad	Alta

Cuadro 5.9: CU-07. Rellenar una plantilla

Nombre e ID del CU	CU-08. Crear una nueva plantilla
Actor	Inspector
Descripción	El usuario genera una nueva plantilla con sus correspondientes bloques para poder emplearla en futuras inspecciones.
Precondiciones	PRE-1. El usuario disponde cuenta y ha iniciado sesión en la aplicación.
Postcondiciones	POST-1. La nueva plantilla queda registrada en la BBDD de la aplicación y se puede emplear para generar nuevas inspecciones.

Flujo normal	FN1 El usuario pulsa el botón Crear nueva plantilla. FN2 El sistema le muestra un formulario con los campos que componen la información básica de la plantilla. FN3 El usuario rellena dichos campos y pulsa Siguiente. FN4 El sistema le muestra al usuario una nueva pantalla que le permite ir agregando bloques a las plantillas con sus correspondientes preguntas. FN5 El usuario agrega bloques a la plantilla, cuando finaliza pulsa el botón Guardar. FN6 El sistema guarda la información de la nueva plantilla en BBDD y devuelve al usuario a la pantalla de inicio.
Flujo alternativo 1	FA5 El usuario pulsa guardar sin haber creado ningún bloque para dicha plantilla, el sistema informa al usuario con un mensaje de error por pantalla.
Prioridad	Alta

Cuadro 5.10: CU-08. Crear una nueva plantilla

Nombre e ID del CU	CU-9. Visualizar una inspección realizada
Actor	Inspector
Descripción	El usuario accede a una de las inspecciones que realizó previamente para comprobar los resultado de esta.
Precondiciones	PRE-1. El usuario tiene una sesión iniciada en la aplicación y ha realizado alguna inspección previa.
Postcondiciones	POST-1. El usuario visualiza los datos de la inspección que realizó.
Flujo normal	FN1 El usuario que se encuentra en la pantalla de inicio va a la sección de Mis Inspecciones. FN2 El sistema muestra la pantalla Mis Inspecciones con un listado de las inspecciones el usuario. FN3 El usuario selecciona una inspección. FN4 El sistema muestra un formulatio no editable con la información de la inspección elegida.
Flujo alternativo 1	FA3 El usuario no dispone de ninguna inspección previa.

Prioridad	Alta

Cuadro 5.11: CU-9. Visualizar una inspección realizada

Nombre e ID del CU	CU-10. Eliminar una inspección
Actor	Inspector
Descripción	El usuario selecciona una de las inspecciones que tiene guardadas y la elimina.
Precondiciones	PRE-1. El usuario tiene una sesión iniciada en la aplicación y está visualizando una inspección que realizó previamente.
Postcondiciones	POST-1. La inspección es eliminada de la BBDD.
Flujo normal	FN1 El usuario pulsa sobre el botón Eliminar inspección en la pantalla en la que se visualizan los datos de la misma. FN2 El sistema advierte al usuario de dicha acción mediante una ventana emergente con un mensaje y 2 opciones, eliminar y cancelar. FN3 El usuario pulsa eliminar. FN4 El sistema elimina dicha inspección de la base de datos y todos los riesgos relacionados con la misma, tras esto devuelve al usuario a la pantalla de inicio.
Flujo alternativo 1	FA3 El usuario pulsa sobre el botón cancelar, el sistema le devuelve a la pantalla anterior y el caso de uso queda sin efecto.
Prioridad	Alta

Cuadro 5.12: CU-10. Eliminar una inspección

Nombre e ID del CU	CU-11. Editar una inspección
Actor	Inspector
Descripción	El usuario selecciona una de las inspecciones que tiene guardadas y tras seleccionar una opción se abre un formulario que le permite editar los datos de los distintos riesgos que la componen.
Precondiciones	PRE-1. El usuario tiene una sesión iniciada en la aplicación y está visualizando una inspección que realizó previamente.
Postcondiciones	POST-1. La inspección es editada y sus correspondientes riesgos también, guardandose dicha información en BBDD.

Flujo normal	FN1 El usuario pulsa sobre alguno de los riesgos que componen la inspección FN2 El sistema carga los datos de dicho riesgo en un formulario no editable. FN3 El usuario pulsa sobre el botón Editar riesgo. FN4 El sistema muestra la misma pantalla siendo en este caso editable. FN5 El usuario edita los valores que desee de dicho formulario y pulsa sobre el botón Guardar. FN6 El sistema muestrá al usuario una pantalla de confirmación con 2 opciones, guardar y cancelar. FN7 El usuario selecciona guardar. FN8 El sistema almacena la información actualizada de dicho riesgo y muestra al usuario la pantalla con el formulario no editable de dicho riesgo y sus datos ya actualizados.
Flujo alternativo 1	FA7 El usuario pulsa sobre el botón cancelar, el sistema le devuelve a la pantalla anterior y el caso de uso queda sin efecto. FA8 El usuario selecciona un nuevo riesgo de la lista y el caso de uso continua en el paso 2.
Prioridad	Media

Cuadro 5.13: CU-11. Editar una inspección

Nombre e ID del CU	CU-12. Descargar una inspección
Actor	Inspector
Descripción	El usuario selecciona una de las inspecciones que tiene guardadas y tras pulsar el botón Descargar se genera un fichero en formato PDF que se descarga a su dispositivo.
Precondiciones	PRE-1. El usuario tiene una sesión iniciada en la aplicación y está visualizando una inspección que realizó previamente.
Postcondiciones	POST-1. Los datos de la inspección se pasan a formato PDF y se genera un archivo de dicho tipo que se descarga al dispositivo.

Flujo normal	FN1 Cuando el usuario esta en la pantalla de visualización de la inspección pulsa sobre el botón Descargar. FN2 El sistema guarda la información de dicha inspección en un archivo PDF y dicho archivo se descarga al dispositivo del usuario.
Prioridad	Media

Cuadro 5.14: CU-12. Descargar una inspección

5.3. Modelo de dominio

A continuación se analizará al modelo de dominio desarrollado para la aplicación, este se puede observar en la Figura 5.2.

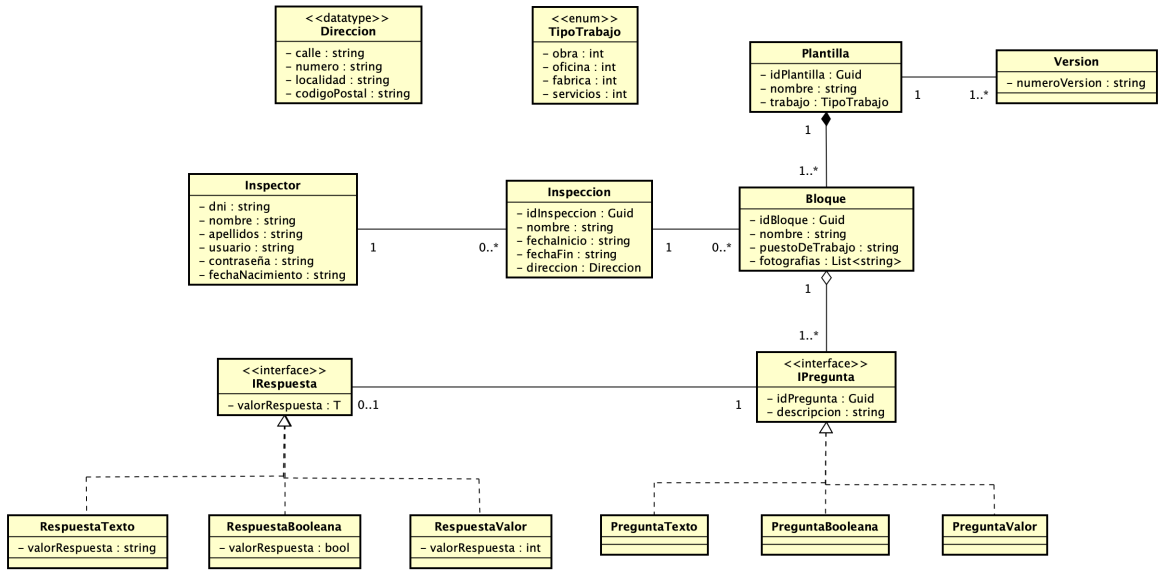

Figura 5.2: Diagrama dominio del sistema

En el modelo de domino encontramos que la aplicación estaría formada por 6 clases, 3 enumeraciones y un datatype. Las primeras se correspondería con los inspectores, es decir los usuarios de la aplicación de los cuáles se dispondría de una serie de datos básicos obtenidos durante el resgistro en la misma. Estos tendrían una serie de inspecciones las cuales dispondrán de un identificador único empleando la clase GUID de **C#**, esta misma clase se empleara en el resto con el mismo fin, para asegurar que los identificadores de los distintos objetos sean siempre únicos.

Las inspecciones a su vez dispondrán de una seria de fechas tanto de inicio como de fin y de una direccion compuesta por: calle, número, código postal y localidad. Además de un nombre para identificar de forma mas sencilla a la misma.

Las inspecciones estarán formadas por un conjunto de bloques los cuales a su vez forman en conjunto una plantilla de inspección. La plantilla representa un esquema de cuestionario para las inspecciones, se le ha asigna un id único, un nombre para distinguirla con facilidad y se clasifica en función del tipo de puesto de trabajo que cubre. Asi mismo a las plantillas habrá asociadas una serie de versiones de la misma, se ha empleado una clase para representar dicha característica por la posibilidad de conservar las versiones previas.

Las plantillas se forman por bloques que a su vez componen una inspección, estos bloques representas secciones de una plantilla específicas como podría ser la correcta iluminación del espacio de trabajo. El bloque esta compuesto por un nombre y un id único, y dentro del bloque disponemos de las preguntas que cubren dicha sección. Estas preguntas se forman a partir de un id, una descripción y el tipo de pregunta al que corresponde, con esto se determina si son preguntas de respuesta afirmativa o negativa, cuantitativa, etc...

Por último tenemos las respuestas, una pregunta puede tener como mucho una respuesta en el ámbito de la inspección. Las respuestas se componen un valor que dependiendo del tipo de pregunta a la que corresponda será de un tipo u otro.

5.4. Atributos de calidad

La aplicación deberá cumplir una serie de estándares mínimos en cuanto a calidad, de los que cabría destacar los siguientes:

Rendimiento: con respecto a los tiempos de espera se desea obtener un tiempo máximo de espera de 5 segundos, ya que según un estudio realizado por Google en el año 2018, este es el tiempo máximos que esta dispuesto a esperar un usuario sin que llegue a suponerle excesivo [2]. Con respecto a la memoria se decidió emplear **Firebase** como BBDD para evitar almacenar una gran cantidad de datos en el dispositivo lo que ayuda a que el consumo de memoria de la aplicación sea mucho menor y su vez esta sea más eficiente, sin embargo esto también supone que la aplicación requiere de conexión a Internet para funcionar pues todos los aspectos de la misma incluido el login requieren de conexión.

Seguridad: el acceso a la aplicación y a los datos de la misma solo se podrá realizar a través del login en esta, para ello se ha decidido emplear **Firebase Authentication**. Este sistema permite acceso a mecanismos de autenticación seguros que son facilmente compatibles con los distintos lenguajes hoy en día y que cada vez se emplean más en el desarrollo de aplicaciones que emplean BBDD Firebase. [3]

Robustez: para asegurar esta cualidad se realizaran diversos test que prueben la funcionalidad de aplicación previniendo de esta forma bugs o errores así como la posibilidad de que la misma se quede colgada durante su funcionamiento básico.

Capítulo 6

Diseño

En este capítulo detallaremos el diseño de la aplicación. Lo primero es destacar que al emplear **C#** nos encontramos con un lenguaje construido en base al paradigma de la orientación a objetos, y por tanto el desarrollo de esta aplicación se basa en dicho paradigma.

La Orientación a Objetos es un paradigma que modela en entidades los elementos del dominio que se desea representar. Estas entidades se construyen en clases que se componen de una serie de atributos y operaciones que cubren sus funciones y cualidades básicas, y en tiempo de ejecución se crean instancias de dichas clases denominadas objetos que tienen identidad propia y estados.

En la orientación a objetos existen otras cualidades así como la herencia que permite relacionar 2 clases distintas en base aspectos comúnes que poseen, de esta forma se emplea una menor cantidad de código para representarlas porque la clase hija de la relación hereda dichas cualidades comúnes de la otra. [9]

6.1. Diseño de datos

En concreto dado que se va a emplear Firebase como base de datos, el diseño se trata de una base de datos no relacional. Las bases de datos no relacionales funcionan a partir de colecciones que almacenan documentos, estos documentos representan la información y se podrían considerar como objetos del dominio, es decir en este caso existiría una colección inspectores que almacenaría la información de los distintos usuarios de la base de datos.

Los documentos de las colecciones pueden contener a su vez subcolecciones, las cuales permiten guardar numerosos documentos dentro de otro. En este caso dentro de cada inspector existirá una colección de inspecciones que el haya realizado. De esta forma se puede acceder rápidamente a las inspecciones que un determinado usuario sin la necesidad de emplear claves foráneas como en el caso de una base de datos relacional.

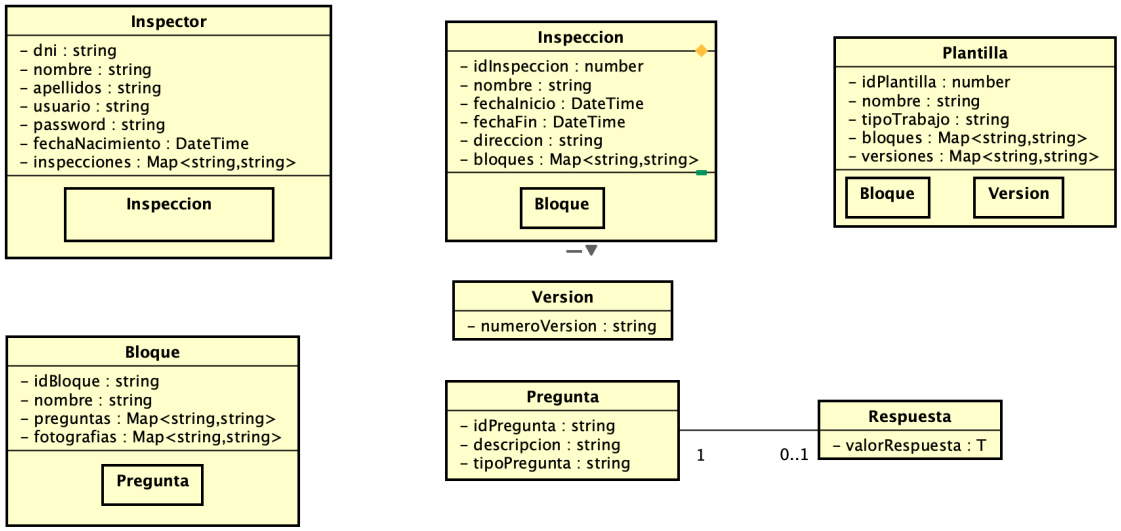

Figura 6.1: Modelo de datos del sistema

En la Figura 6.1 podemos observar el modelo de datos que empleará en el desarrollo de la aplicación. Dado que la aplicación empleada para generar los diagramas no posee este tipo de diagrama en concreto al ser para una base de datos no relacional la estructura se ha recreado a partir de la base de un modelo de dominio.

Como se puede ver el inspector tendrá una serie de datos personales además de tener una colección de inspecciones, las cuáles se han representado aqui mediante el uso de una lista. Este esquema mantiene los atributos mostrados en el modelo de domino de la Figura 5.2, y cada clase cuenta con la colección en forma de lista que le correspondería, como ya se ha enunciado el inspector posee sus inspecciones, cada una de estas inspecciones tendrá una lista de bloques que la conforman y los bloques contendrán una seria de preguntas las cuales pueden o no tener una respuesta.

La colección Plantilla hace referencia a las plantillas que se podrían seleccionar para las inspecciones. En su caso no es necesario guardar su información en la inspección, simplemente se accedería a ellas para poder obtener los distintos bloques y por consecuente sus preguntas. Como se explicó en el modelo de dominio dentro de las plantillas existe una subcolección de versiones que dicha plantilla ha tenido.

6.2. Diagrama de clases

En esta sección se detallara el diagrama de clases desde el punto de vista del diseño de la aplicación, en la Figura 6.2 se puede observar dicho diagrama.

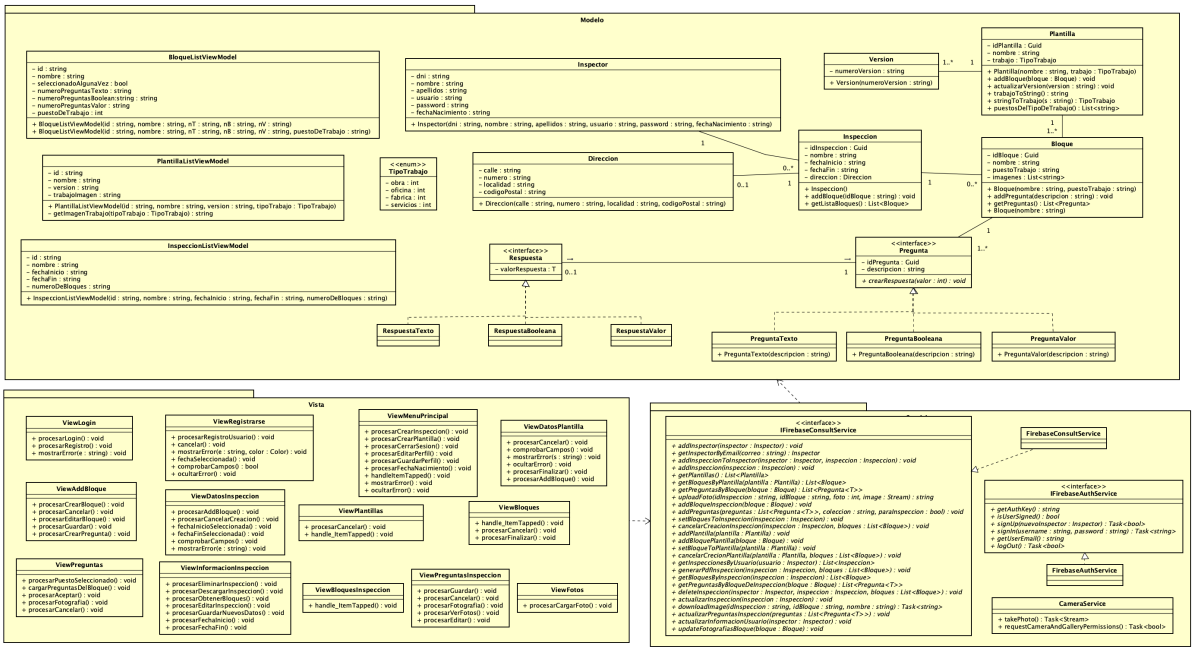

Figura 6.2: Modelo de dominio del sistema

Como se muestra en la figura se puede observar un modelo similar al del capítulo de Análisis, pero en este caso no se indican agregaciones o composiciones y a su vez si se muestran los métodos de las clases que forman el sistema. En este diagrama se pueden observar en detalle los parámetros y sus tipos, asi como los valores que dichas funciones retornarán. En cuanto a las clases y su multiplicidad encontramos las mismas que en el modelo de dominio de análisis.

Como se explicó en el punto anterior al tratarse de una base de datos no relacional se operará con colecciones sin embargo cada clase va a contar un identificador único para poder distinguir los distintos datos en función de ello. En el caso del inspector se trata del DNI que permite identificar a una persona, y para el resto de elementos se empleará la clase Guid de **C**# para generar un valor único.

La plantilla será la única clase independiente al resto del sistema pués sus datos se emplearán para poder construir las inspecciones pero no serán alterados ni asociados por ningún usuario. Se debe concretar la posibilidad de actualizar dichas plantillas con el objetivo de generar una nueva versión, para ello tenemos la clase versión que consta de un atributo que corresponde con el número de la versión y un constructor para crearlas.

Para identificar la versión actual la clase inspección dispondra de 2 atributos, uno que se corresponde con la versión en la que se encuntra y una lista de sus distintas versiones. Sin embargo el resto de las clases contendrán su correspondiente información, así como en la base de datos cada colección contendra sus correspondientes subcolecciones. Esto compondría el modelo de la aplicación, siendo los 2 paquetes restantes los servicios y las vistas que se emplearán en el sistema.

Con respecto a los servicios, la interfaz **IFirebaseAuthService** es la encargada de controlar el login y el registro de los usuarios en la aplicación. Sus diferentes métodos gestionan todos los aspectos relacionados con dichas tareas, lo que permite separar las clases del modelo de dichos aspectos. Por otra parte la interfaz **IFirebaseConsultService** será la encargada de gestionar los intercambios de datos con el servicio Cloud Firestore de Firebase, esta tendrá una implementación en cada sistema operativo atendiendo a las particularidades de los mismos.

Por último dentro de este paquete encontramos la clase **CameraService**, la cuál es la encargada de operar con la cámara del dispositivo, en este caso a diferencia de las 2 anteriores se emplea una única clase para los 2 sistemas pues comparten características y por tanto no es necesario construir una interfaz para este servicio.

Finalmente el paquete Vista contiene las distintas interfaces que compodrán la aplicación y que serán las encargadas de gestionar los intercambios de información con el usuario.

6.3. Diagramas de secuencia

6.3.1. Secuencia Login

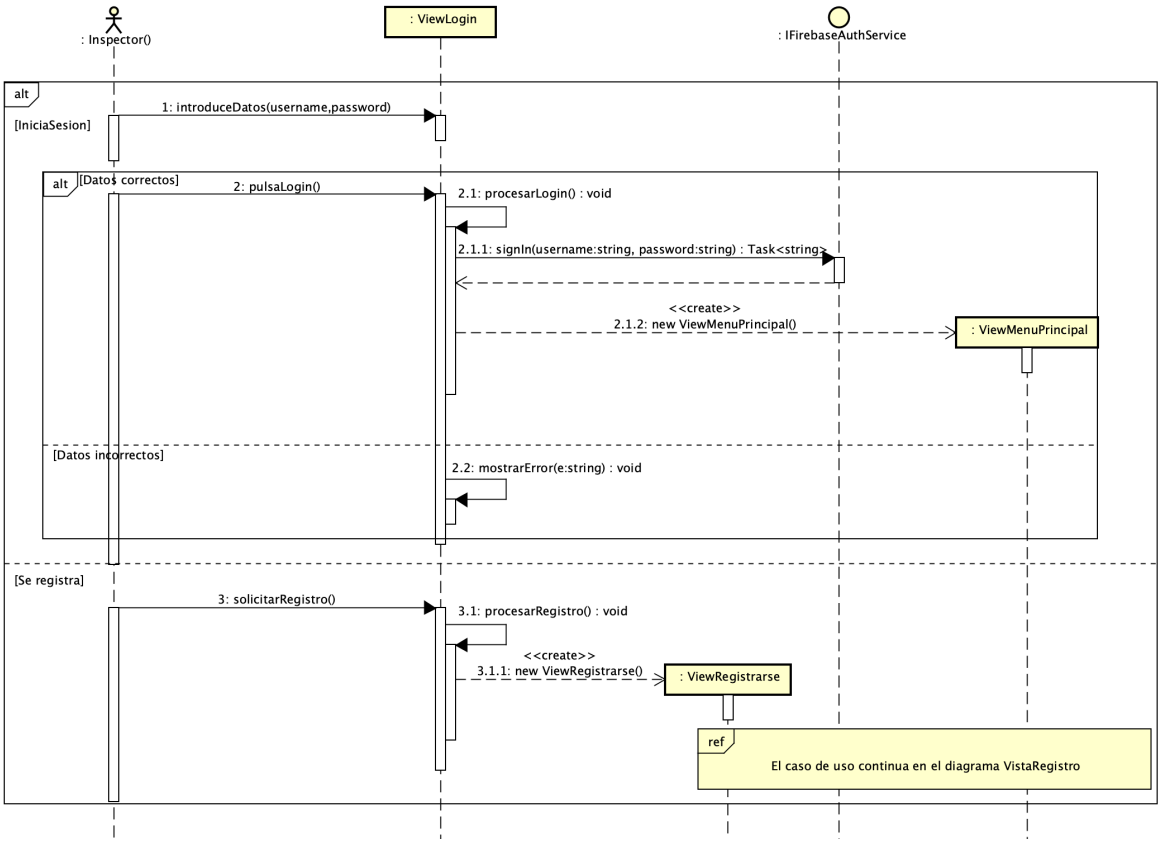

Figura 6.3: Diagrama de secuencia del caso de uso de Iniciar Sesión

6.3.2. Secuencia Registro

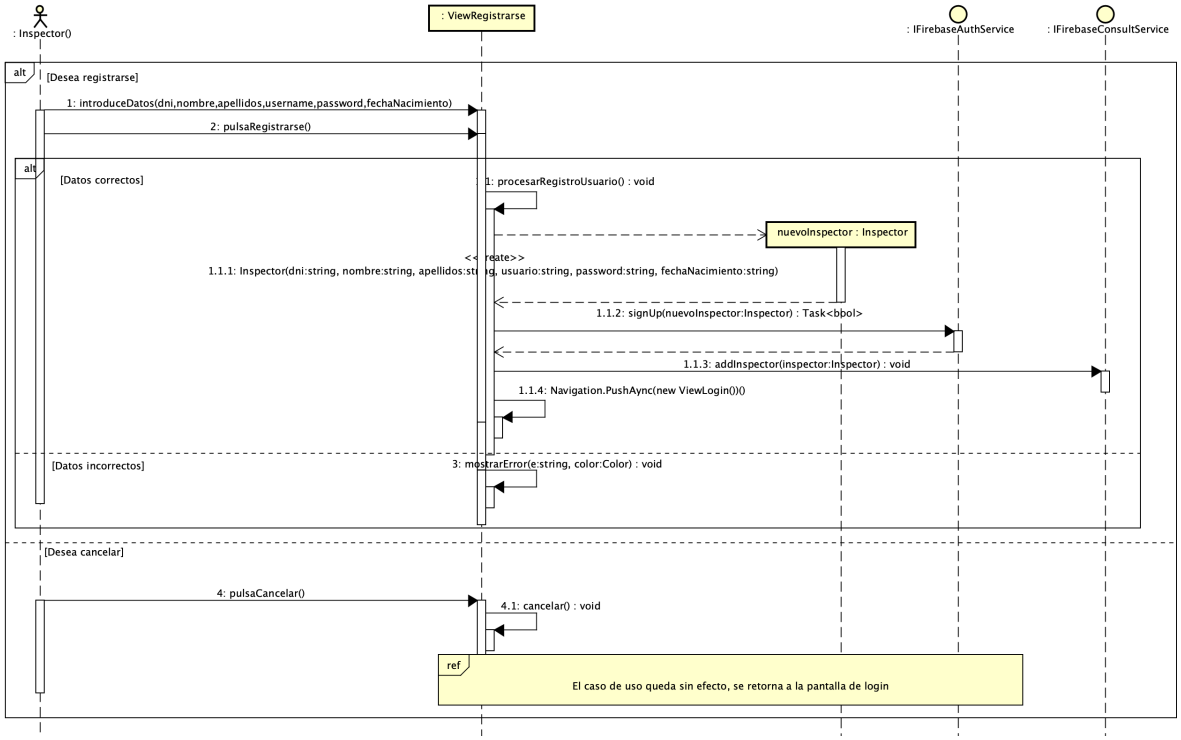

Figura 6.4: Diagrama de secuencia del caso de uso Registrarse como usuario

6.3.3. Secuencia Cerrar sesión

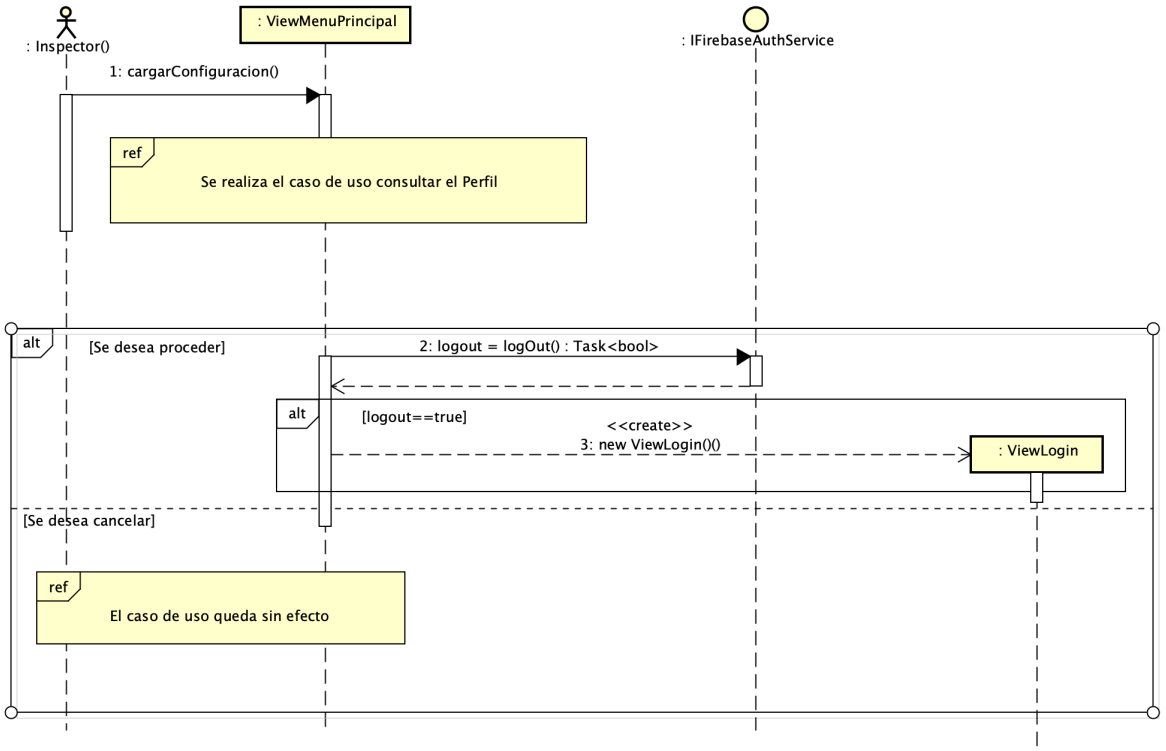

Figura 6.5: Diagrama de secuencia del caso de uso Cerrar sesión

6.3.4. Secuencia Consultar el Perfil

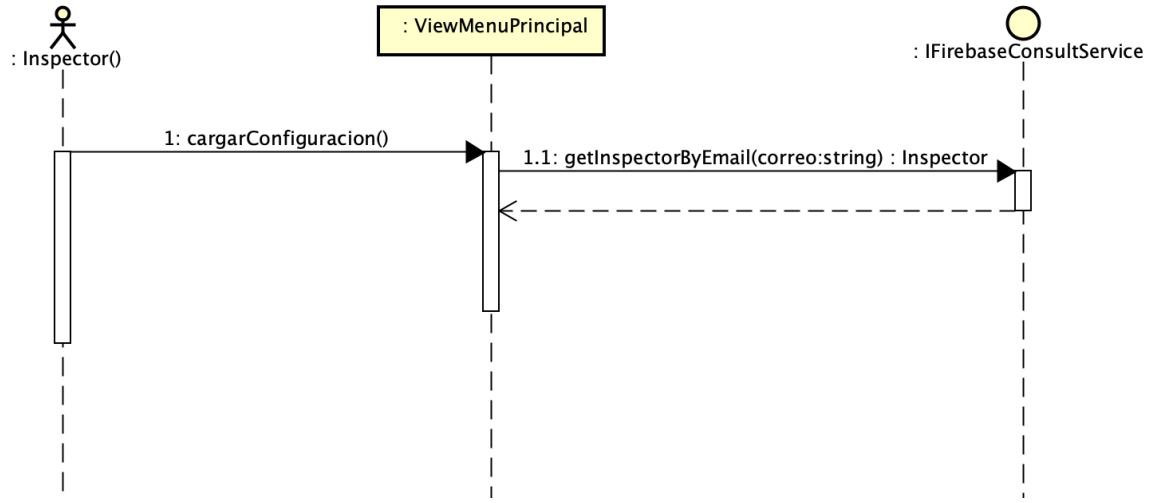

Figura 6.6: Diagrama de secuencia del caso de uso Consultar el Perfil

6.3.5. Secuencia Editar el Perfil de Usuario

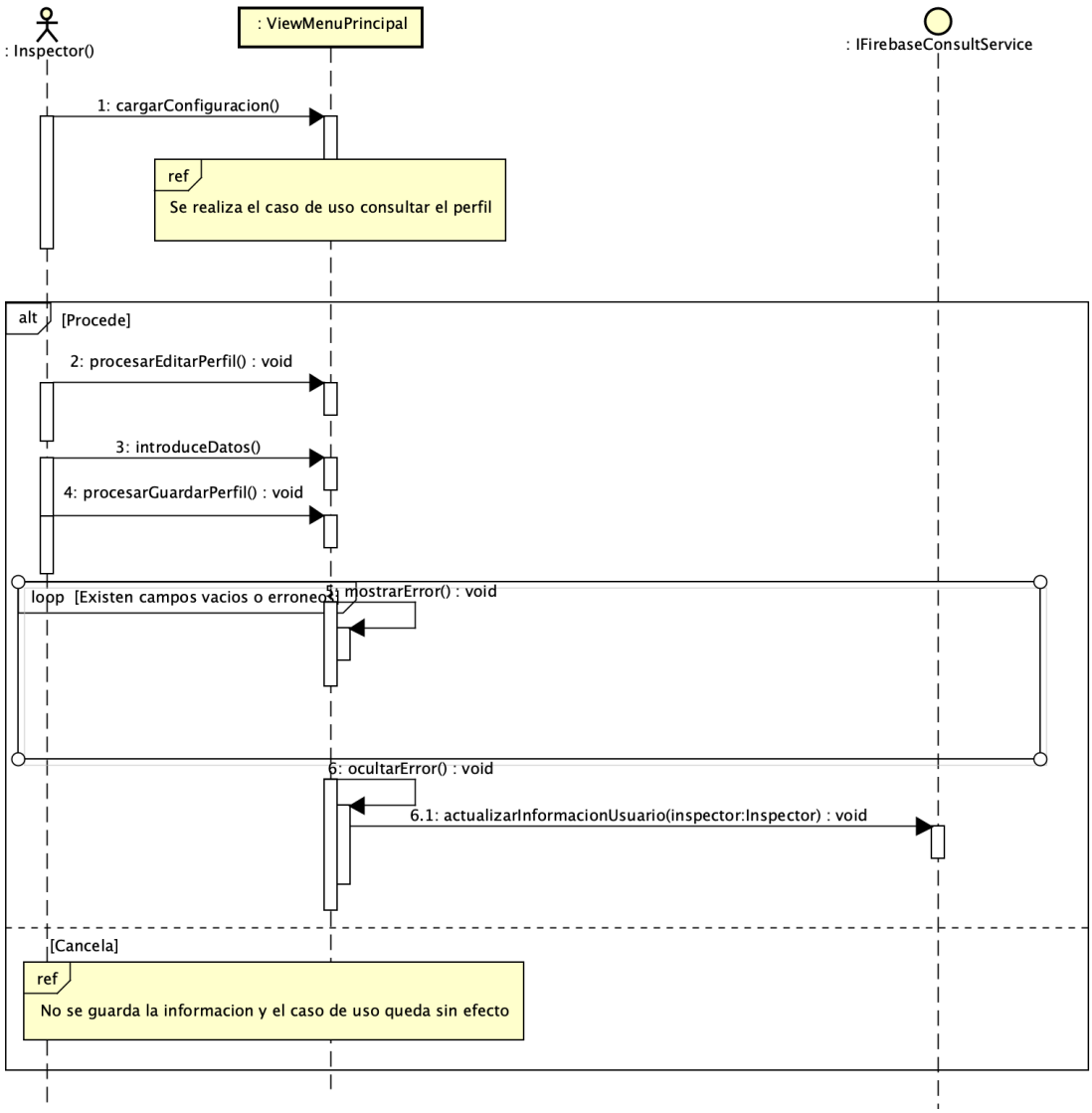

Figura 6.7: Diagrama de secuencia del caso de uso Modificar el Perfil

6.3.6. Secuencia Crear una Nueva Plantilla

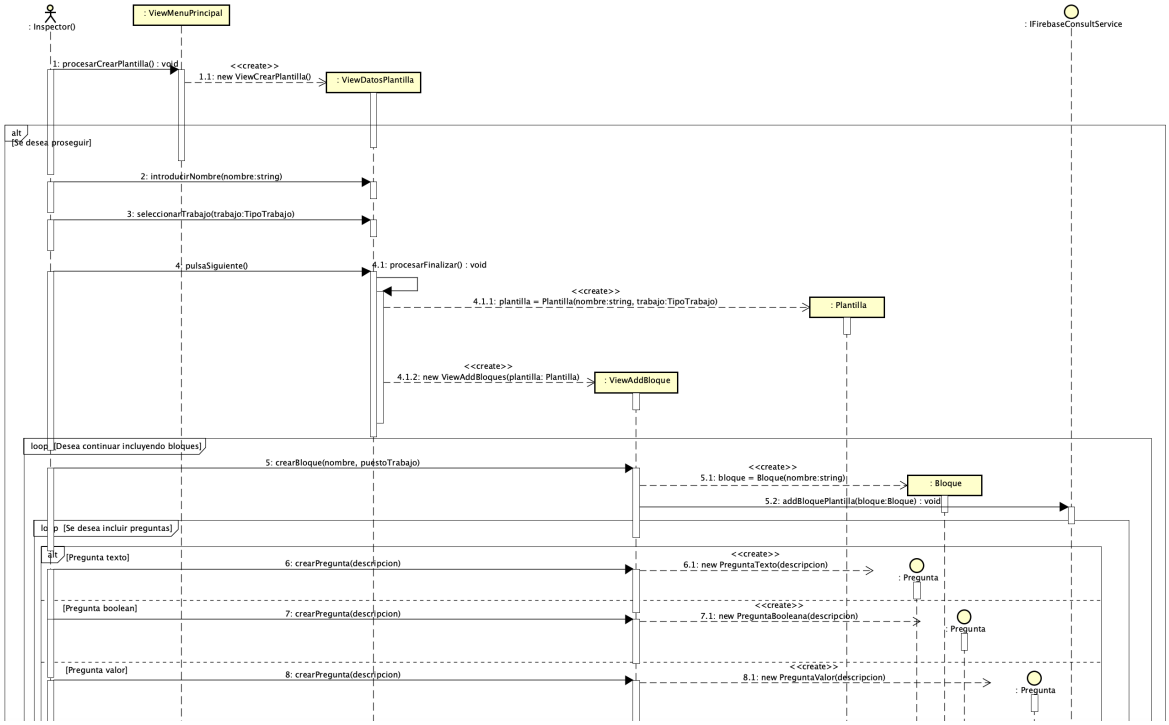

Figura 6.8: Diagrama de secuencia del caso de uso Crear una nueva Plantilla, primera mitad

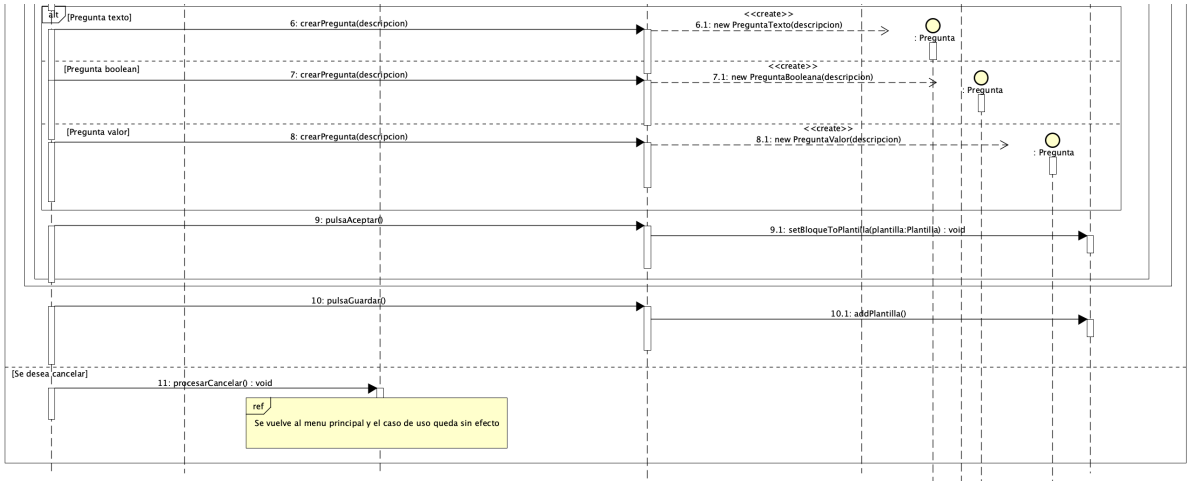

Figura 6.9: Diagrama de secuencia del caso de uso Crear una nueva Plantilla, segunda mitad

6.3.7. Secuencia Crear una Inspección

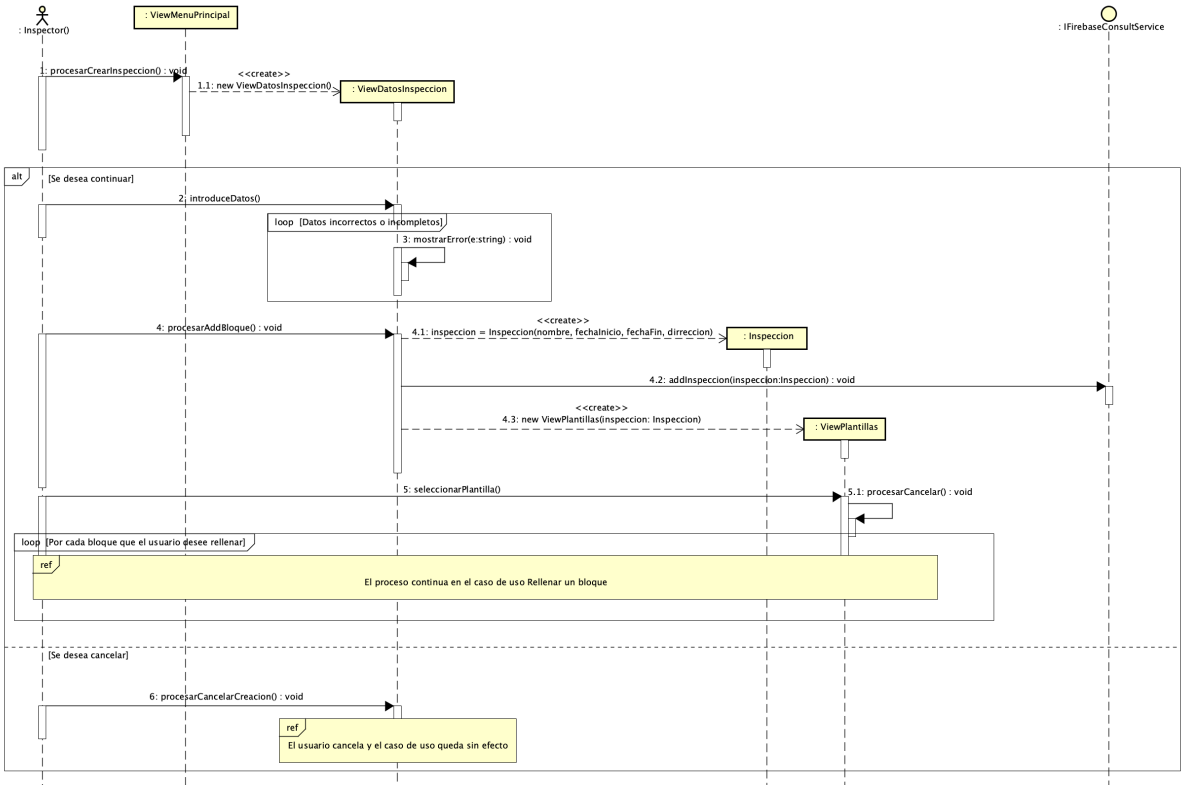

Figura 6.10: Diagrama de secuencia del caso de uso Crear una Inspección

6.3.8. Secuencia Rellenar bloques de una Inspección

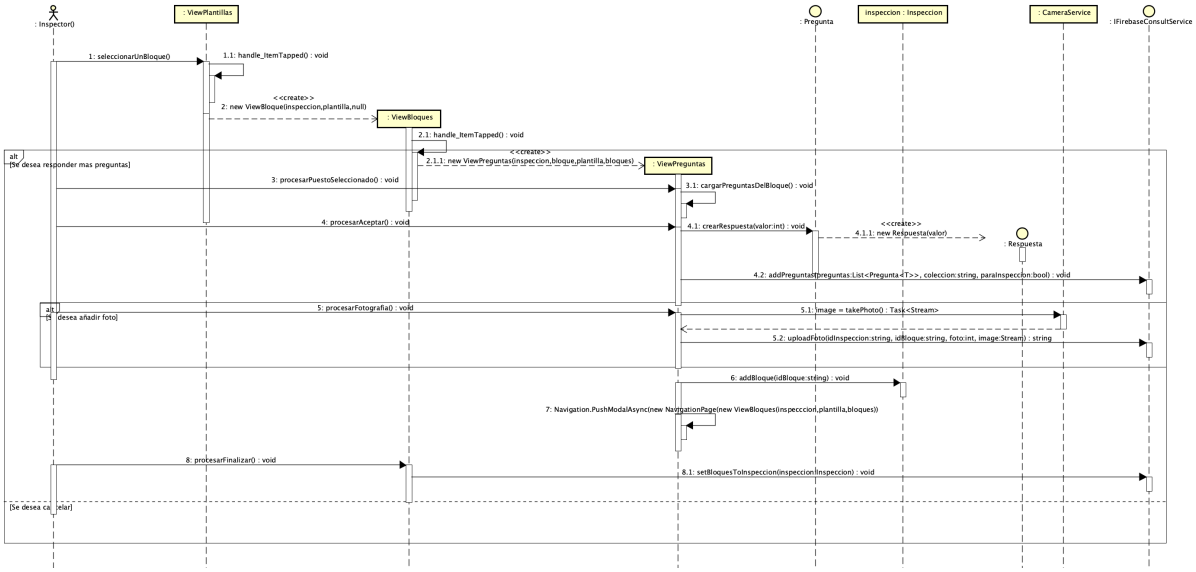

Figura 6.11: Diagrama de secuencia del caso de uso Responder un cuestionario

6.3.9. Secuencia Visualizar una Inspección

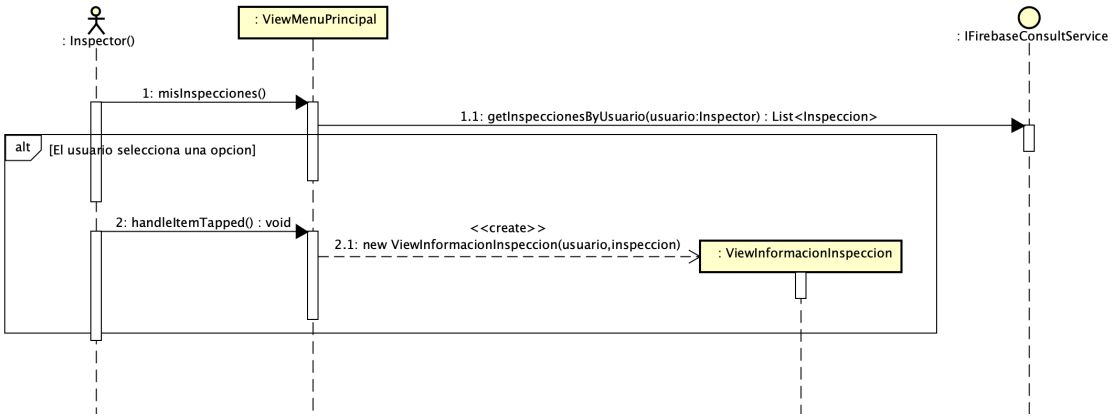

Figura 6.12: Diagrama de secuencia del caso de uso Visualizar una Inspección

6.3.10. Secuencia Editar una Inspección

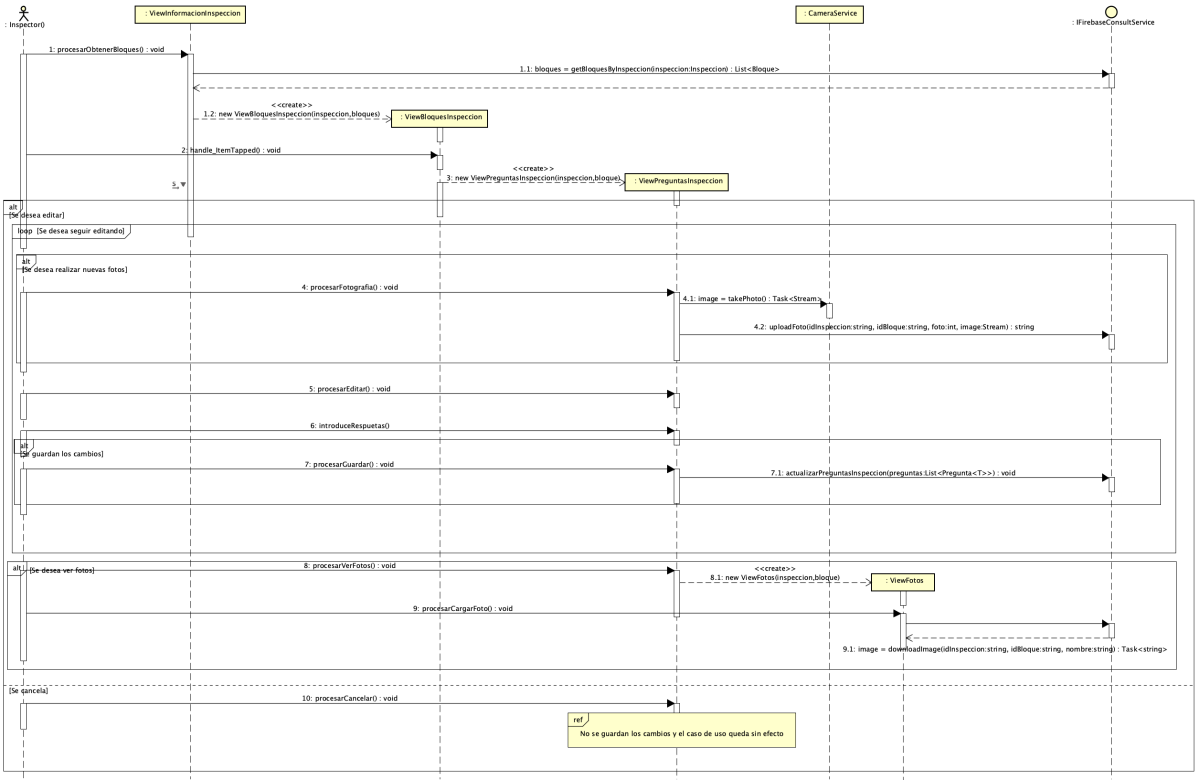

Figura 6.13: Diagrama de secuencia del caso de uso Editar una Inspección

6.3.11. Secuencia Descargar una Inspección

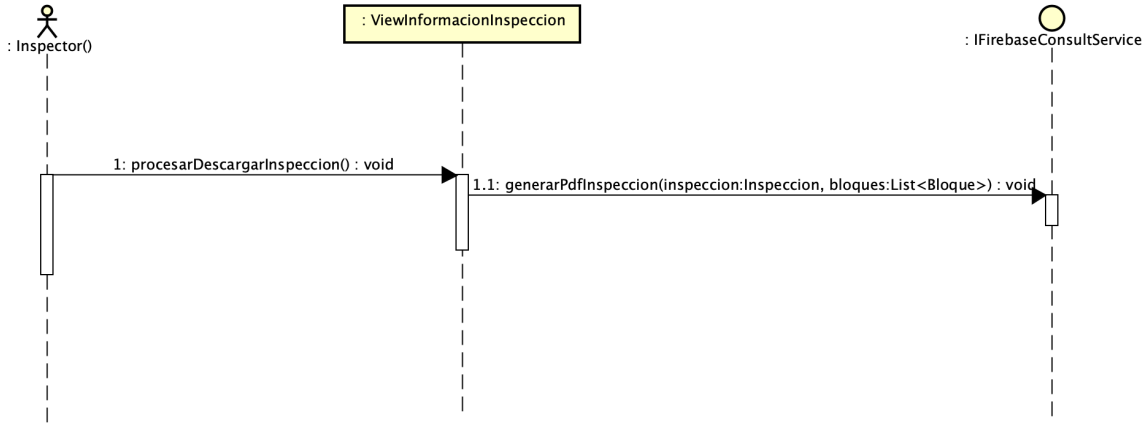

Figura 6.14: Diagrama de secuencia del caso de uso Descargar una Inspección

6.3.12. Secuencia Eliminar una Inspección

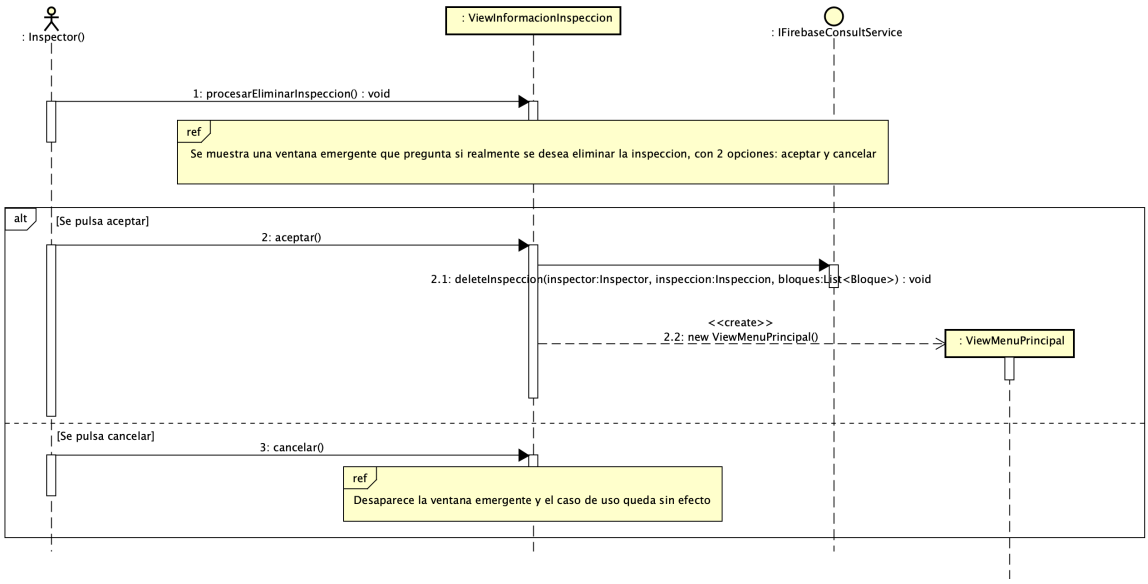

Figura 6.15: Diagrama de secuencia del caso de uso Eliminar una Inspección

Capítulo 7

Patrones de Diseño

Respecto a los patrones de diseño empleados en el desarrollo de la aplicación encontramos los siguientes.

7.1. Patrón Singleton

El patrón Singleton permite restringir la creación de objetos de una determinada clase en una única instancia de la misma. De esta forma todos aquellos que empleen dicha clase trabajaran sobre la misma instancia permitiendo un mayor controlo sobre el flujo de información de la misma.

En concreto se ha empleado dicho patrón en la clase DatabaseConecction. La función de dicha clase en la aplicación es la de obtener la conexión con la base de datos de la aplicación alojada en Firebase, en concreto en el servicio Cloud Firestore.

La conexión con la base de datos se realiza al lanzar la aplicación y de esta forma el servicio FirebaseConsultService solo requiere de obtener la instancia de DatabaseConecction para trabajar con normalidad sobre la base de datos. [11]

7.2. Patrón Experto

El patrón Experto determina que la clase encargada de implementar un constructor y los distintos métodos que la componen es aquella que tiene toda la información sobre dicho objeto.

Las clases que componen la aplicación son cada una de ellas las encargadas de implementar los distintos constructores y métodos que emplearán sus instancias. [4]

7.3. Patrón Creador

El patrón Creador determina que clase es la adecuada para crear cada objeto. En concreto suele ser aquella que posea la información necesaria para crearlo o requiera de un objeto de dicha clase en su composición.

En concreto en la aplicación la mayoría de las clases componen sus propios objetos exceptuando la clase Rspuesta, ya que el constructor de la Respuesta es invocado desde el método Responder de la Pregunta correspondiente, siendo el valor que se suministra distinto dependiendo del tipo de Pregunta del que se trate. [4]

7.4. Patrón MVVM

El patrón MVVM es comunmente empleado en el desarrollo de aplicaciones multiplataforma. Esta compuesto por 3 elementos principales: Modelo, Vista y Modelo-Vista.

El Modelo contiene toda la información referente al modelo y la lógica de la aplicación. Formado este por las distintas clases que componen el dominio de la misma.

La Vista se corresponde con la interfaz de usuario, esta es ajena al modelo y se encarga de mostrar la información al usuario y de la interacción directa con el mismo.

El Modelo-Vista tiene la función de comunicar y conectar el modelo con la vista, esta compuesto por la lógica de la interfaz y los comandos que en esta se pueden realizar.

Es común cuando se emplea este patrón usar los Binding que relacionan directamente valores mostrados en la interfaz con elmentos de Modelo-Vista. Estos en concreto se emplean en la aplicación en las clases que muestran listas, como la lista de plantillas ya que al generar la interfaz se desconoce la información que es necesario mostrar. Primero se obtiene la información de las distintas plantillas en el Modelo-Vista y luego los elementos de la interfaz emplean el Binding para determinar la información que se mostrará en cada caso[6, 8, 1].

Capítulo 8

Pruebas

Las pruebas desarrolladas durante la fase final del desarrollo han sido pruebas de integración, aquellas que permiten probar que los distintos elementos que componen el sistema funcionan correctamente de manera conjunta, empleando para ello UITest de Xamarin. [7]

Se han desarrollado test por cada uno de los casos de uso que componen la aplicación probando en estos la distinta funcionalidad que los compone así como la validación de los datos introducidos durante el proceso de los mismos. Lamentablemente no se ha podido probar de esta forma el funcionamiento de la cámara en las distintas partes de la aplicación, debido a que actualmente no existe forma de controlar la cámara del dispositivo con dicho Framework.

Para realizar los test con este Framework es necesario que exista un archivo **.apk** de la aplicación, ya que requiere emplear este para poder realizar los test sobre el mismo. Para obtenerle es necesario seguir los siguientes pasos:

1. Seleccionar la aplicación InspectionManager.Android del proyecto.
2. Seleccionar la opción Compilar y dentro de esta Archivo para publicar.
3. Se abrirá una nueva pestaña en el editor con el archivo que se va a generar, indicando su nombre, fecha de creación y el número de versión.
4. Seleccionar el archivo deseado y pulsar el la opción Firmar y distribuir.
5. En la nueva ventana que se abrirá se debe seleccionar la opción Ad Hoc.
6. En la siguiente pestaña se selecciona la firma ya existente y se pulsa en Siguiente.
7. Por último se pulsa en Publicar y se introduce la clave de la firma, la cuál es **H23jkZ2P9t**, seleccionando tras esto el directorio en el que se almacenará el archivo **.apk**.

Una vez realizado todo el proceso se debe acceder a la clase AppInitializer de InspectionManagerTest y determinar la ruta donde se almacena el apk. Hecho todo esto bastara con encender el emulador que se este empleando y ejecutar la aplicación de test.

El proceso seguido en el desarrollo de las pruebas es el siguiente. Primero se crea un nuevo usuario validando previamente los campos que componen el resgistro para finalmente registrar un usuario válido. Una vez se ha registrado dicho usuario se inicia sesión en la aplicación con el usuario recien creado y se verifica que no tiene inspecciones y sus datos son los que se han introducido previamente.

El tercer paso es crear una plantilla. Una vez más se validan los campos que componen la información propia de la plantilla así como los que componen los distintos bloques de la misma. Si en algún momento se pulsa en la opción cancelar el proceso se detiene, se elimina la información y se vuelve a la pantalla de inicio como se muestra en el test. De la plantilla se crearán 2 bloques y antes de guardar el segundo se cambiara su nombre para probar que se puede editar dicha información durante el proceso.

En el cuarto paso se probará la edición de los datos de usuario, existen datos que no pueden cambiarse como el DNI del usuario, ya que es el valor que identifica a los usuarios en base de datos. Así como su email y contraseña pues son los elementos empleados por Firebase Auth para registrar a los usuarios.

El quinto paso es el más complejo y extenso. Se trata del proceso de crear una inspección, en este caso también se validarán los campos de la inspección y se mostrara como en este caso el numero de la dirección de la inspección puede estar vacío ya que existen direcciones que no constan de número. En el proceso se empleará la plantilla que se ha creado en el paso anterior, rellenando su primer bloque 2 veces para distintos puestos de trabajo y el segundo una única vez. Y en cada uno de esos bloques se dejará sin responder alguna pregunta verificando asi que no es necesario responder todas las preguntas que componen el bloque. Una vez se ha terminado el proceso se guarda la inspección y se verifica que aparece en la lista de inspecciones del usuario.

En el sexto paso se edita la información básica de la inspección y se válida una vez más los campos de la misma.

En el séptimo paso se accede a la inspección y se pulsa sobre la opción Descargar, de esta forma aparece un aviso en pantalla indicando que se ha generado un PDF con la información de la inspección. Al igual que ocurre con la cámara no es posible manipular elementos ajenos a la aplicación con este Framework, por lo que no se puede acceder en la prueba al archivo generado, sin embargo al finalizar la prueba se puede acceder a los archivos del dispositivo y en la carpeta InspectionManager se habrá generado un archivo PDF con el nomnbre de la inspección en cuestión.

En el octavo paso se válida y verifica la posibilidad de editar las respuestas de una aplicación realizada, al igual que se puede comprobar como la inspección esta formada por el primer bloque de la plantilla que se ha rellenado 2 veces para distintos puestos de trabajo y que el segundo bloque solo se ha rellenado una vez.

En el noveno paso se elimina la inspección que se ha creado en el quinto paso y se verifica como esta desaparece de la lista de inspecciones del usuario

Por último en el décimo paso se comprueba que el usuario puede cerrar sesión correctamente.

El proceso se ha desarrollado siguiendo un guión establecido para probar toda la funcionalidad de la aplicación, de manera que las distintas pruebas guardan dependencia con las anteriores. Sin embargo los casos de prueba se han construido de forma separada por lo que para probar solo uno en concreto bastaría con que aquellos elementos de los que requiere existan y se podría ejecutar unicamente este comentando la línea que hay encima del resto de tests en la que pone la palabra **Test** entre corchetes.

Capítulo 9

Implementación

En este capítulo se detallará el proceso seguido durante la implementación de la aplicación, se mostrará la estructura básica de la misma y se indicarán los cambios o modificaciones realizados respecto a la idea original.

9.1. Proceso

Lo primero que se realizó fue un repositorio en github para emplear git como herramienta de control de versiones del proyecto. Esto además de ser una medida de seguridad extra frente a posibles perdidas de información ha permitido probar distintas soluciones simultáneamente con respecto a los problemas que iban surgiendo en el desarrollo del proyecto.

Una vez vinculado el repositorio local al remoto, se comenzó con el desarrollo de la aplicación y se comenzó construyendo el modelo de la misma. En inicio se realizó el modelo conforme a lo mostrado en la fase de análisis existiendo cambios actuales que se realizaron durante el proceso y que se explicarán posteriormente.

Tras haber construido el modelo inicial se comenzó a desarrollar la integración con Firebase, en concreto en inicio con el servicio Firebase Auth que como se indicó previamente sería el empleado para controlar tanto el inicio como el cierre de la sesión en la app, así como el registro de nuevos usuarios en la misma. Para ello se creo una interfaz en el proyecto InspectionManager denominada **IFirebaseAuthService**, compuesto por todos los métodos que se emplearía para dicho servicio.

En cada implementación, tanto la de Android como la de iOS, se construyo la misma carpeta servicios y las correspondientes clases que implementases esta interfaz. En un inicio se propuso la posibilidad de iniciar sesión con la cuenta de Google pero es algo que se desestimó durante el proceso, empleando definitivamente el inicio de sesión mediante e-mail y contraseña.

Tras haber implementado dichas interfaces en cada sistema se probarón de forma que la implementación en Android funcionaba correctamente, pero en iOS no respondía como se esperaba. Tras buscar información y soluciones que no llegaron a funcionar se decidió dejar de lado la implementación en iOS debido a que desde la versión 9.0 del correspondiente sistema operativo los emuladores empleados en Xamarin no responden correctamente con los servicios de Firebase debido a un bug que existe en el IDE. Sin embargo como ambas implementaciones comparten modelo, lógica y vistas el proceso que se ha realizado para Android se podría emplear en la versión de iOS.

Tras conseguir que el login y el registro funcionasen correctamente se procedió a construir la interfaz que trabajaría sobre el servicio de almacenamiento de Firebase Cloud Firestore. Fue para este servicio para el que se desarrollo la clase DatabaseConecction empleando el patrón Singleton, ya que la instancia que se crea de la misma al lanzar la aplicación crea la conexión con la base de datos que se emplea para obtener y almacenar documentos en Cloud Firestore.

En inicio resultó complicado trabajar con la base de datos de Firestore, sobre todo ya que tanto para recibir como enviar documentos esta trabaja con objetos Java, lo que hacia necesario parsear la información para dicho lenguaje antes de emplearla. Fue en este punto donde se decidió no emplear subcolecciones como era la idea inicial, ya que al tener que manipular de esta forma los datos habría sido muy complicado construir la información del mismo modo que obtenerla. En su lugar se optó por emplear mapas dentro de los documentos que almacenasen los ids correspondientes, por ejemplo un usuario almacenará en sus inspecciones el id de cada una de ellas con el cuál se podrán obtener de Firebase cuando sea necesario.

Para mantener un modelo más común al que se iba a emplear en la base de datos se decidio alterar las clases del dominio de forma que estas guardasen al igual que Firestore solo los id de los objetos que contienen, a excepción de la clase pregunta que sigue manteniendo como parte de sus atributos la respuesta, puesto que esta última no disponde una colección en Firestore sino que se genera como un mapa de su correspondiente pregunta.

Tras haber adecuado el modelo y realizar correctamente tanto las insercciones de datos como el poder traerlos de la base de datos se procedió a construir el resto de la aplicación mediante los casos de uso que se indicarón en la fase de análisis. De esta forma se comenzó por el caso de uso Crear una nueva Inspección, cabe destacar que la vista correspondiente al menú principal se construyo empleando una *TabbedPage*. Esta es una vista que esta compuesta por varios hijos los cuales son pantallas independientes que permiten una navegación rápida y fluida entre todas ellas, de esta forma se unificó en una sola pantalla las vistas correspondientes a la pantalla principal, la de configuración o perfil y la correspondiente a la lista de inspecciones del usuario.

Durante el desarrollo del caso de uso de Crear una nueva Inspección, se realizarón

ciertas modificaciones referentes a la base de datos. En concreto como se debía permitir al usuario rellenar el mismo bloque de una plantilla múltiples veces para distintos puestos de trabajo, de forma que se crearón nuevas colecciones en firebase en referencia a los elementos que componen una Inspección, de esta forma esta colecciones se pueden identificar debido a que terminan con la palabra Inspeccion. Aquellas colecciones para las que se crearon variaciones son Bloques y Preguntas de los distintos tipos, sus diferencias con respecto a las colecciones normales e basan en el caso de Bloques en que en la variante de inspección el id del documento es una combinación de su id con el puesto para el que se ha rellenado así como que guarda un puesto de trabajo y una lista de fotografias que se han realizado en dicho bloque. Y en el caso de las preguntas en las referentes a una inspección encontramos su correspondiente respuesta en función del tipo que esta sea. Como se acaba de mencionar se decidio guardar las fotografías en el bloque en lugar de la respuesta puesto que tiene mas sentido que las fotografías se referencien en función del bloque para el que se tomarón.

Para el almacenamiento de las fotografías se empleó un nuevo servicio de Firebase, en este caso Firebase Storage. El cuál permite la creación de distintas carpetas en las que se pueden almacenar archivos, en concreto cuando se realizan fotografías en una inspección y más concretamente en un bloque se crea una estrutura de carpetas en dicho servicio que primero crea una carpeta por la inspección y dentro de esta tantas como bloques en los que se guarden fotografías, las cuales se almacenarán en la carpeta de su correspondiente bloque.

En un inicio la información se iba a conjuntar al final para realizar la insercción en base de datos pero debido a múltiples comprobaciones que se deben realizar durante la creación de la inspección la información se va almacenando progresivamente en las correspondientes colecciones de forma que al final todo queda correctamente relacionado, y en el caso de que se cancele el proceso se elimina la información existente en base de datos hasta el momento.

En el caso de uso de Crear una nueva Plantilla, el proceso seguido es más similar a la idea original pero al igual que en el caso anterior la información se va almacenando en base de datos de forma progresiva y en el caso de cancelar esta se elimina.

Para poder ver una inspección ya creada como se enunció antes se sustituyo el buscador por una lista de inspecciones que se encuentra en la pantalla principal dentro de la sección Mis inspecciones. Desde dicha lista se puede acceder a la información de una inspección asi como editar sus datos básicos, descargar dicha inspección en formato PDF, eliminarla o ver sus bloques y las respuesta de cada uno, las cuales también se pueden editar si así se desea.

Para construir los PDF de las inspecciones se empleo la biblioteca iTextsharp, y estos se construyeron mostrando en inicio la información básica de la inspección. Tras esto se muestra por cada bloque las preguntas con sus correspondientes respuestas y por último

las fotografias que estos tienen.

En lo referente a las preguntas de los bloques, desde la lista estas se pueden editar de forma libre asi como en cada bloque podemos acceder a una pantalla desde la cuál descargar y visualizar las imágenes que tomamos al realizarlo. Igualmente se pueden incluir nuevas imágenes si asi se desea.

Por último se implementó la posibilidad de editar la información del usuario si así se desea.

9.2. Estructura de la aplicación

En esta sección se detallará la estrutura básica de la misma, indicando donde se guardan las distintas clases y archivos que componen la aplicación, así como las partes individuales de cada sistema operativo y el proyecto de testing.

9.2.1. Visión general de la solución

Como se puede observa en la Figura 9.1, un proyecto Xamarin o Solución esta compuesto de 3 proyectos básicos, siendo el cuarto que aparece en la imagen el proyecto de Test. El proyecto InspectionManager se corresponde con aquel que tendrá la lógica común de la aplicación.

En la carpeta Modelo se almacena el modelo de la aplicación completo, ya que este será el proyecto que hará uso de dichas clases. En Servicios se encuentran tanto la interfaz **IFirebaseAuth** como **IFirebaseConsultService**, además de incluir la clase CameraService, la cuál se encargará de emplear la biblioteca *Xam.Plugin.Media*, puesto que en este caso la implementación para ambos sistemas operativos es la misma, simplemente con iniciar en ellos en plugin podrán emplear sus servicios.

Por último la carpeta Vistas contiene tanto los archivos **.xaml** que se corresponden a las interfaces como una clase que hace la función de Modelo-Vista o Controlador para cada una de ellas, dentro de dichas clases se hace uso de la inyección de Dependencias para obtener la implementación de los servicios necesaria en función del sistema operativo que se ejecute.

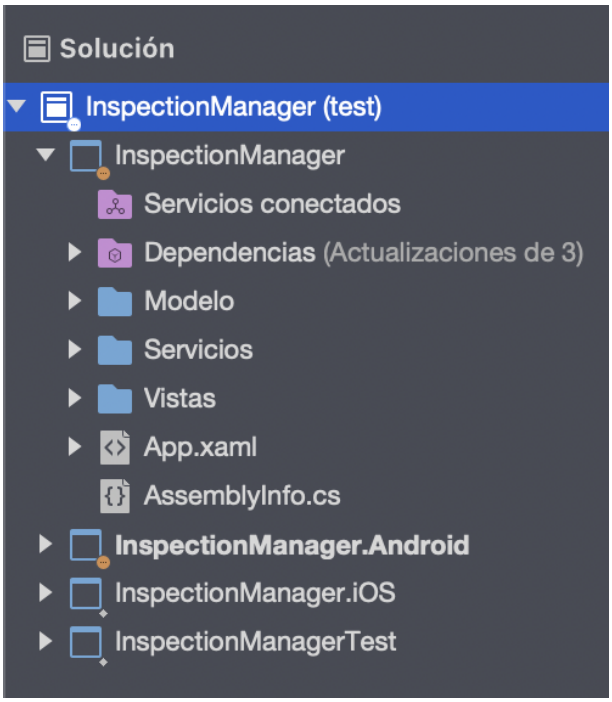

Figura 9.1: Estructura básica del proyecto Xamarin

9.2.2. Proyecto Android

En la Figura 9.2 podemos observar la estructura del proyecto Android de la solución. La clase MainActivity es la inicia la aplicación y donde se obtiene la conexión con Firebase que será empleada posteriormente. El archivo google-service.json contiene la configuración del proyecto Firebase con el que se conecta la aplicación.

En la carpeta Properties se encuentra el archivo AndroidManifest el cuál contiene la información del proyecto así como la versión y los permisos que tiene la aplicación. Y en la carpeta Resources encontramos elementos que se emplearán en el código de la aplicación, dentro de esta esta la carpeta drawable que contiene todas las imágenes que se emplean posteriormente en la aplicación.

Por último la carpeta Servicios contiene las implementaciones de las 2 interfaces de Firebase, así como la clase DatabaseConecction.

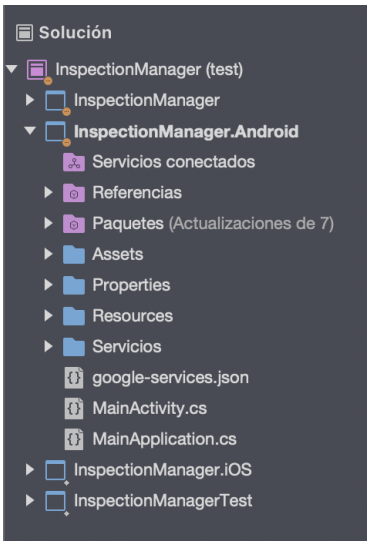

Figura 9.2: Estructura básica del proyecto Android

9.2.3. Proyecto iOS

En la Figura 9.3 se puede observar la estructura que compone el proyecto iOS de la solución.

En este caso los recursos que empleará la aplicación como por ejemplo imágenes se guardan en el paquete *Assets.xcassets*, introduciendo en este nuevos campos a los que se asocia la imagen.

La clase que lanza la aplicación es en este caso AppDelegate y es donde se realiza la configuración con Firebase, el archivo GoogleService-Info.plist es el equivalente al visto en el último proyecto y que servía como archivo de configuración de Firebase. Y por último el archivo Info.plist es el que guarda y determina la configuración básica de la aplicación.

En la carpeta Servicios se encontrarían las correspondientes implementaciones de las interfaces de Firebase, pero debido al problema ya mencionado previamente esto no se ha podido probar ni terminar de desarrollar.

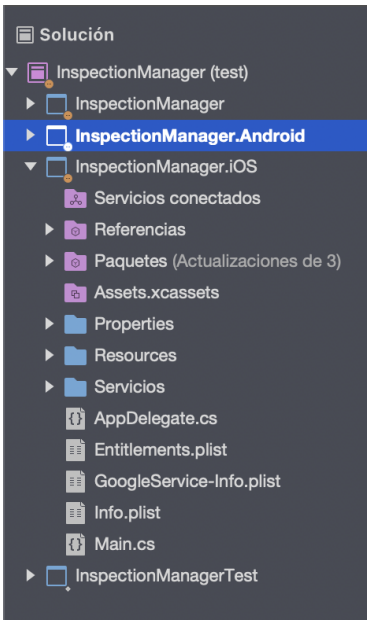

Figura 9.3: Estructura básica del proyecto iOS

9.2.4. Proyecto de Test

Por último en la Figura 9.4 encontramos la estructura del proyecto de Test de la aplicación.

Este consta principalmente de 2 clases, la primera AppInitializer es la encargada de desplegar la aplicación del sistema operativo correspondiente empleando el archivo de aplicación para ello. Mientras que la clase Tests contiene las distintas pruebas que se ejecutarán.

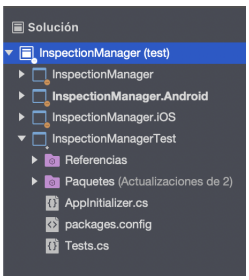

Figura 9.4: Estructura básica del proyecto de Test

9.3. Cambios y modificaciones

En este apartado se detallarán los cambios que se han realizado con respecto a la idea original del proyecto. Muchos han sido ya comentados en la parte de Proceso pero aqui se indicará con mas detalle por que se llevaron a cabo.

En primer lugar los cambios en el modelo que se han enunciado previamente. La razón principal de estos cambios se debe a la forma de intercambiar información con Firebase, en concreto es necesario parsear toda la información que se pose para que al enviarse se envie como objetos Java, lo que hace más complicado tener clases que esten compuestas por objetos de otras distintas. Hacerlo de esta forma supondría tener que parsear mucha información constantemente lo que se podría traducir en una mayor facilidad para cometer errores en el proceso. Además haciendolo de esta forma existe una relación más lógica con la información que se almacena en la base de datos. También como se indicó antes las interfaces Pregunta y Respuesta ya que cuando la información de la pregunta se almacena en la base de datos, se considerá la respuesta como un campo más de esta.

El segundo cambio principal en lo referente a las vistas se ha producido pues durante el desarrollo la idea inicial de vistas que se había planteado no siempre encajaba tan bien como se esperaba en un inicio. Por lo cuál estas se fueron adaptando conforme la aplicación se iba desarrollando. Un buen ejemplo de esto es el hecho de haber convertido las vistas correspondientes al menú principal, el perfil del usuario y la lista de sus inspecciones en una sola vista. De esta forma las secciones de la aplicación a las que con mayor frecuencia se puede acceder son facilmente accesibles unas a otras, mejorando la experiencia del usuario.

Por último se cambió también a causa del parseo los objetos de tipo DateTime de las clases. Debido a que ocurrían errores ocasionales al traer la información de Firebase, por lo tanto la información obtenida de los campos de fecha (DatePickers), se parsea a string empleando el formato dd/MM/yyyy.

Parte III

Manuales de la Aplicación

Capítulo 10

Manual de Instalación

Al tratarse de una aplicación móvil la instalación es un proceso muy sencillo. Basta con disponer de un dispositivo Android con la versión del software 9.0 o superior y ejecutar el archivo **.apk** en el dispositivo que se desee.

Al emplear Firebase para el almacenamiento persistente no es necesario configurar nada en referencia a la base de datos, pero debido a esto la aplicación requiere de conexión a Internet para poder funcionar.

Capítulo 11

Manual de Usuario

En este capítulo se detallará el funcionamiento de la aplicación y la administración de la base de datos. La primera parte se basará en una serie de instrucciones acompañadas de fotografías para visualizar el normal funcionamiento de la app. Y la segunda parte se centrará en detallar como se almacena la información en la base de datos así como su gestión en caso de necesidad.

11.1. Manual de Usuario

Al iniciar la aplicación por primera vez esta mostrará una pantalla de login como la que se puede ver en la Figura 11.1. Desde esta ventana se podrá acceder a la aplicación introduciendo un usuario y una contraseña que hayan sido registrados previamente en la misma. En el caso de no disponer de una cuenta de usuario basta con pulsar en el boton **Registrarse** para acceder a una pantalla que nos permite crear una nueva cuenta.

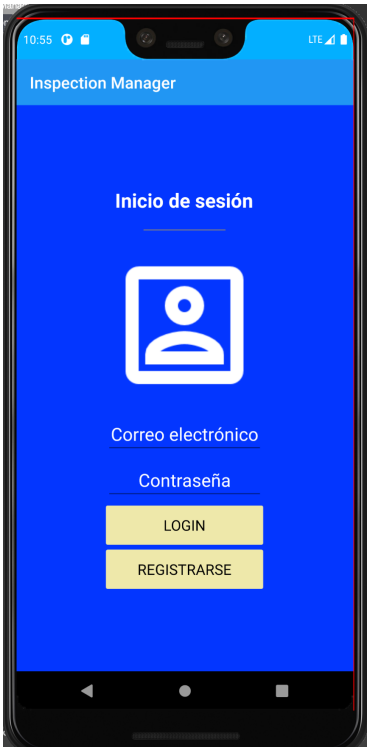

Figura 11.1: Captura de pantalla de la vista de Login

En la pantalla de resgitro que se puede ver en la Figura 11.2, encontramos una serie de campos que se corresponden con la información básica de todo usuario. Todos y cada uno de ellos deben ser cumplimentados para poder crear correctamente un usuario. En concreto, la fecha de nacimiento introducida debe indicar que el usuario es mayor de edad, en caso contrario la aplicación no permitirá registrar a dicho usuario. Si el registro es correcto la aplicación redirecciona al usuario de nuevo a la pantalla de login.

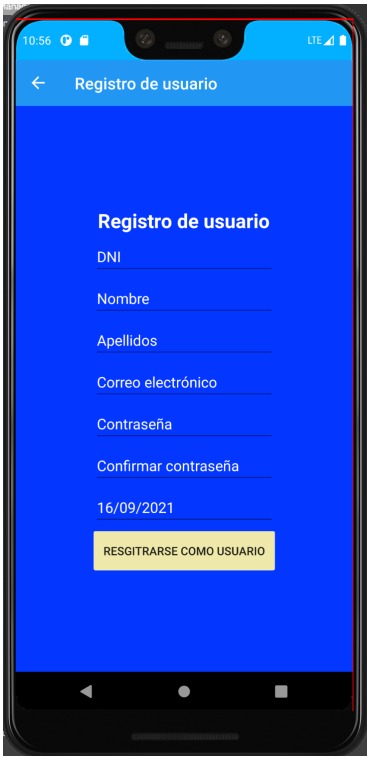

Figura 11.2: Captura de pantalla de la vista de Resgistro de usuario

Una vez se inicia sesión en la aplicación, se muestra la pantalla que corresponde al menú principal de la misma, el cuál se puede observar en la Figura 11.3. Esta muestra 2 opciones que son **Crear una nueva inspección** y **Crear nueva plantilla**, estas opciones enviarían al usuario a sus correspondientes vistas pero antes de continuar cabe destacar que como se enunció previamente el menú principal esta compuesto por 3 vistas independientes las cuales se pueden ver en la barra de navegación inferior de la pantalla. De esta forma el usuario puede acceder a dichas vistas o bien pulsando los correspondientes iconos de dicha barra o desplazando la pantalla lateralmente.

Figura 11.3: Captura de pantalla de la vista Menú Principal, pantalla de Creación de Inspecciones

La opción **Mi Perfil** muestra la información del usuario tal y como se puede ver en la Figura 11.4, desde esta misma pantalla se pueden tanto editar los datos del usuario como cerrar sesión si así lo desea. Los datos que se pueden modificar son el nombre, los apellidos y la fecha de nacimiento, verificandose igualmente que la nueva fecha indique que el usuario es mayor de 18 años. El resto de datos no son modificables debido a que tienen una función importante en la administración de la base de datos, esta función es permitir al usuario iniciar sesión empleando el servicio Firebase Auth y guardar la información del mismo en Cloud Firestore empleando su Dni como clave.

Figura 11.4: Captura de pantalla de la vista Menú Principal, pantalla de Mi Perfil

Por último la pantalla de **Mis Inspecciones** muestra una lista con las inspecciones que tiene el usuario registradas en ese momento como se puede ver en la Figura 11.5. El usuario puede pulsar en cualquiera de ellas para poder ver su información más en detalle así como editar la misma, eliminarla o descargar dicha inspección en formato PDF.

Figura 11.5: Captura de pantalla de la vista Menú Principal, pantalla de Mis Inspecciones

La siguiente pantalla que se puede ver en la Figura 11.6 se corresponde con la información básica de una plantilla que se va a crear. En esta será necesario introducir el nombre que recibirá dicha plantilla así como el sector o tipo de trabajo para el que se esta construyendo. Actualmente en la aplicación existen los tipos que se pueden ver en el enum TipoTrabajo del modelo de dominio, pero esto podría ser ampliable añadiendo nuevos valores asi como sus correspondientes iconos. Una vez se han introducido los datos básicos se pulsa en la opción **Nuevo Bloque** para crear un bloque para dicha plantilla, cabe destacar que no se permitirá guardar la plantilla hasta que por lo menos este compuesta de un bloque, pues no sería lógico almacenar una plantilla vacía.

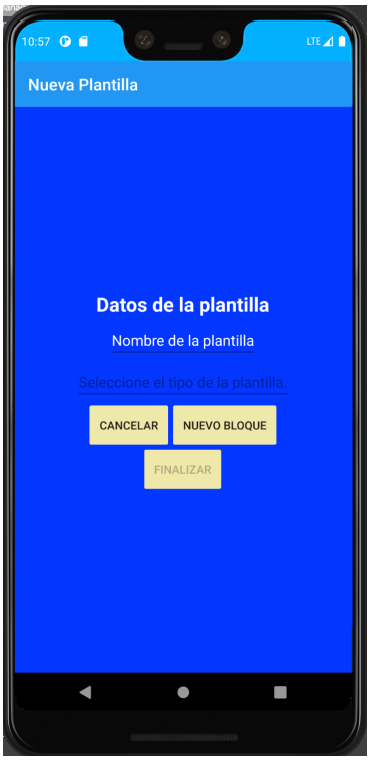

Figura 11.6: Captura de pantalla de la vista de Nueva Plantilla

Al pulsar en **Nuevo Bloque** se muestra la pantalla que se puede ver en la Figura 11.7. Aquí se rellenará la información correspondiente al bloque, el cuál a la hora de crearlo consta de un nombre y un conjunto de preguntas asociadas. Una vez se introduzca el nombre del bloque y se cree se permitirá al usuario crear tantas preguntas de los distintos tipos como este desee. Una vez el usuario termine pulsará en guardar y la información se registrará en la base de datos, redirigiendo al usuario a la pantalla anterior por si desea añadir más bloques a la plantilla o guardarla debido a que ya disponde de al menos un bloque. Si en algún punto el usuario pulsa en la opción cancelar se eliminará la información guardada hasta ese punto y el caso de uso quedará sin efecto.

Figura 11.7: Captura de pantalla de la vista de Nuevo Bloque

Si el usuario decide crear una nueva inspección se le mostrará la pantalla que se puede ver en la Figura 11.8. En esta pantalla el usuario introducirá los datos básicos de la inspección que va a construir. El campo de fecha de inicio fija como fecha mínima para iniciar una inspección el día en curso y como fecha mínima para terminarla el día posterior. En este caso todos los campos vuelven a ser obligatorios de cumplimentar a excepción del número, pues existen direcciones que no constan de número, en caso de dejar este campo vacío el sistema lo registrará como *Sin número* en la base de datos. Una vez esta cumplimentada toda la información el usuario pulsará en el botón **Seleccionar Plantilla**, desde el cuál se mostrará una lista con las plantillas disponibles para seleccionar.

Figura 11.8: Captura de pantalla de la vista de Nueva Inspección

La lista de plantillas se puede ver en la Figura 11.9. En esta se muestrán las plantillas disponibles para emplear, con información básica de cada una de ellas así como un icono que se obtiene en función del campo *TipoTrabajo* de la plantilla. Pulsando sobre una de ellas se muestra la lista de bloques que la conforman.

Figura 11.9: Captura de pantalla de la vista de Lista de Plantillas

En la Figura 11.10 se puede observar la lista de bloques referentes a una determinada plantilla. Estos contienen una serie de preguntas que se indican en su información básica, las cuales se pueden responder pulsando sobre el bloque que se desee rellenar. Un bloque puede ser rellenado varias veces en una misma inspección pero cada una de esas veces se rellenará para un puesto de trabajo distinto, este se seleccionará al pulsar sobre él. Si se pulsa el botón **Cancelar** se eliminará la información guardada y el caso de uso queda sin efecto, y solo se podrá pulsar **Finalizar** cuando al menos se haya rellenado un bloque para la inspección.

Figura 11.10: Captura de pantalla de la vista de Lista de Bloques

En la Figura 11.11 se encuentra la pantalla de Preguntas del bloque, en concreto lo primero que se mostrará será un selector para elegir el puesto de trabajo para el cuál se va a rellenar este bloque.

Figura 11.11: Captura de pantalla de la vista de Preguntas, selector de puesto de trabajo

Una vez se seleccione el puesto y este sea válido la vista se actualizará mostrando las preguntas que componen el bloque así como una serie de campos para responderlas, tal y como se puede ver en la Figura 11.12. Si las preguntas de texto no se responden se guardará una cadena vacía al no haber sido rellenada, las preguntas de verdadero un falso constan de un check según el cual si este está marcado o no se considera la respuesta como verdadera o falsa, y por último las preguntas de respuesta númerica guardan o bien el valor o bien 0 si esta se ha dejado en blanco. Si no se desea rellenar dicho bloque se podrá volver a la pantalla anterior pulsando atrás, **Cancelar** si se desea cancelar el proceso, el icono de la foto para guardar una fotografía como evidencia o bien en **Aceptar** para guardar dicho bloque en la inspección.

Figura 11.12: Captura de pantalla de la vista de Preguntas, preguntas del bloque

Al acceder a una inspección desde el listado de inspecciones del usuario se muestra una pantalla como la que se puede ver en la Figura 11.13. En esta se muestra la información básica de la inspección, la cuál se puede editar si se pulsa en dicho botón, guardando posteriormente dichos cambios. Desde esta misma pantalla es posible tambien descargar la inspección en formato PDF, guardandose dicho archivo en una carpeta dentro del dispositivo llamada InspectionManager, y también se puede eliminar la inspección pulsando sobre el botón correspondiente. Si se elimina la inspección se redirige al usuario a la pantalla principal de la aplicación. La última opción de esta pantalla, **Ver Bloques** permite al usuario ver y editar la información de los bloques de la inspección.

Figura 11.13: Captura de pantalla de la vista de Información de la Inspección

La Figura 11.14 muestra la pantalla de selección de bloques de la inspección, esta es en esencia muy similar a la ya vista cuando se rellena una inspección pero en esta los bloques muestran también el puesto para el que se cumplimentaron. Pulsando en ellos se accede a las preguntas que se repondieron en dicho bloque.

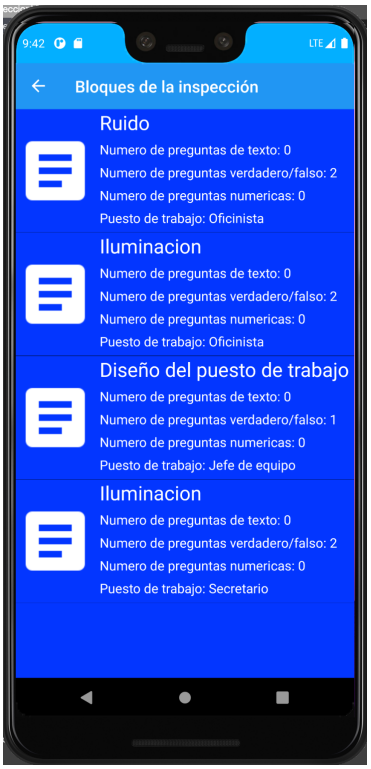

Figura 11.14: Captura de pantalla de la vista de Bloques de la inspección

Al pulsar sobre alguno de los bloques se muestra una pantalla con las distintas preguntas y respuestas del bloque, como se puede ver en la Figura 11.15. Al igual que el caso anterior, esta pantalla es similar a la vista a la hora de crear una inspección, permitiendo en este caso editar las respuesta si se selecciona la opción y añadir nuevas fotos al bloque. Si se pulsa el botón **Ver Fotos** se accede a una nueva pantalla que permitirá al usuario ver las distintas fotografías de la inspección.

Figura 11.15: Captura de pantalla de la vista de Preguntas de la inspección

Por último la Figura 11.16 muestra la pantalla de visualización de fotos de una inspección. Bastará con elegir una foto del selector que se muestra en pantalla y pulsar en el botón **Descargar** para que la imagen se muestre en la pantalla. En la Figura 11.17 podemos observar el selector de fotos con las que están disponibles para el bloque y en la Figura 11.18 la imagen que ha sido descargada desde Firebase Storage.

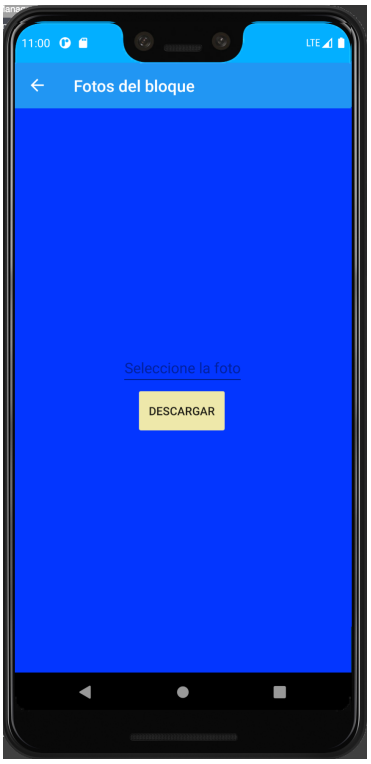

Figura 11.16: Captura de pantalla de la vista del visor de fotografías

Figura 11.17: Captura de pantalla de la vista del visor de fotografías, con el selector de evidencias

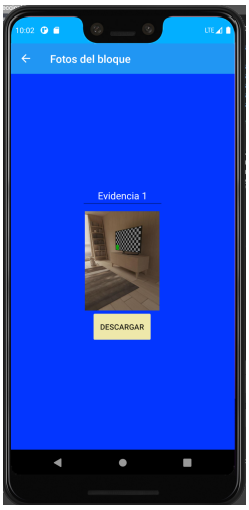

Figura 11.18: Captura de pantalla de la vista del visor de fotografías, con la imagen descargada

11.2. Manual de Administración

En esta sección se detallará la administración referente a la base de datos de la aplicación alojada en Firebase.

La cuenta empleada para alojar la aplicación se llama: **davidtfg2021@inspectionmanager.com**, esta cuenta ha sido creada explicitamente para el desarrollo de la app, por lo que no contiene ningún tipo de información de carácter sensible. La contraseña de dicha cuenta es **H23jkZ2P9t**. La cuál es la misma que se emplea a la hora de firmar la aplicación para poder generar el apk de la misma.

Una vez se accede a la consola de Firebase, se debe iniciar sesión con la cuenta previamente mencionada y acceder a la consola y desde ahí al proyecto Inspection Manager. En este punto se puede acceder libremente a los distintos servicios que se emplean de la base de datos y manipular su información si fuera necesario, como se comentó previamente estos servicios son **Firebase Auth**, **Cloud Firestore** y **Firebase Storage**.

Bibliografía

[1] *Aplicando el patrón de diseño MVVM*. Leomaris Reyes. 2018. URL: https://medium.com/@reyes.leomaris/aplicando-el-patrÃşn-de-diseÃńo-mvvm-d4156e51bbe5.

[2] *Find out how you stack up to new industry benchmarks for mobile page speed*. Google. 2018. URL: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/.

[3] *Firebase Authentication*. Google. 2021. URL: https://firebase.google.com/products/auth?gclid=Cj0KCQjwub-HBhCyARIsAPctr7zpPa2KkkbjVuZDu8_cVbsPUVUrF1sFnMysR1aaZ8Q26uZv-FiwaAugWEALw_wcB&gclsrc=aw.ds.

[4] *GRASP*. Wikipedia. 2020. URL: https://es.wikipedia.org/wiki/GRASP.

[5] *INSPECCIONES DE SEGURIDAD LABORAL*. Asociación Española para la Calidad. 2019. URL: https://www.aec.es/web/guest/centro-conocimiento/inspecciones-de-seguridad-laboral.

[6] *Introducción a MVVM con Xamarin Forms*. Xamarin Latino. 2017. URL: https://xamarinlatino.com/introducciÃşn-a-mvvm-con-xamarin-forms-daabfc36c0c0.

[7] *Introducción a UITest y Xamarin.Forms*. Microsoft. 2021. URL: https://docs.microsoft.com/es-es/appcenter/test-cloud/frameworks/uitest/xamarin-forms?tabs=macos.

[8] *Modelo–vista–modelo de vista*. Wikipedia. 2021. URL: https://es.wikipedia.org/wiki/ModeloâĂŞvistaâĂŞmodelo_de_vista.

[9] *Programación orientada a objetos*. Wikipedia. 2021. URL: https://es.wikipedia.org/wiki/ProgramaciÃşn_orientada_a_objetos.

[10] *Scrum*. Atlassian. 2021. URL: https://www.atlassian.com/es/agile/scrum.

[11] *Singleton*. Wikipedia. 2020. URL: https://es.wikipedia.org/wiki/Singleton.

[12] *¿Qué es la metodología ágil? ¿Para qué sirve?* zendesk. 2021. URL: https://www.zendesk.com.mx/blog/metodologia-agil-que-es/.