Universidad de Valladolid

Escuela Técnica Superior de Ingenieros de Telecomunicación

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías Específicas de Telecomunicación

Mención en Telemática

PrepColon: propuesta de una aplicación móvil multiplataforma para la preparación de colonoscopias

Autor

David Sánchez Olmedo

Tutores

Dr. Guillermo Vega Gorgojo
Dr. Juan Ignacio Asensio Pérez

Valladolid, 9 de julio de 2025

Título:	***PrepColon: propuesta de una aplicación móvil multiplataforma para la preparación de colonoscopias***
Autor:	***David Sánchez Olmedo***
Tutores:	***Dr. Guillermo Vega Gorgojo*** ***Dr. Juan Ignacio Asensio Pérez***
Departamento:	***Teoría de la Señal y Comunicaciones e Ingeniería Telemática***

Tribunal

Presidente:	***Dr. Juan Pablo Casaseca de la Higuera***
Vocal:	***Dr. Rodrigo de Luis García***
Secretario:	***Dr. Miguel Luis Bote Lorenzo***

Fecha:	***17 de julio de 2025***

Resumen

El cáncer colorrectal (CCR) es, de entre todas las variantes de cáncer, la tercera con mayor incidencia y la segunda con mayor mortalidad en España. Es por esto que, desde 2014, se ha realizado en España un programa de cribado que tiene como objetivo la detección precoz de CCR para aumentar la probabilidad de curación del paciente. Este cribado se realiza tomando una muestra de heces a pacientes de entre 50 y 69 años para buscar restos de sangre invisible a simple vista y, en caso de encontrarlos, se cita al paciente para realizarse una colonoscopia. No obstante, el paciente debe acudir a la colonoscopia habiendo realizado una correcta preparación, o de lo contrario una limpieza inadecuada del colon podría dificultar la identificación de lesiones y provocar falsos negativos en la detección de CCR.

Ante esta problemática, y dado el hecho de que actualmente los dispositivos móviles forman parte del día a día en nuestra sociedad, miembros del Servicio del Aparato Digestivo de los centros Hospital Universiario Río Hortega (Valladolid), Hospital Virgen de la Concha (Zamora) y Hospital de Medina del Campo (Medina del Campo) decidieron realizar un estudio clínico con el que determinar si el uso de una aplicación móvil que sirviese de asistente durante la preparación de las colonoscopias podía ayudar a incrementar el número de pacientes que acudían a sus citas con una limpieza adecuada. En este Trabajo de Fin de Grado se propone PrepColon, una aplicación móvil multiplataforma desarrollada con el *framework* Flutter para el apoyo a los usuarios durante el proceso de preparación para sus colonoscopias. Para ello, los usuarios deben completar un formulario en el que se solicitan la fecha de la cita y datos médicos con los que la aplicación podrá determinar, en función de aspectos como el índice de masa corporal del usuario o los medicamentos que toma habitualmente, y siguiendo unas directrices proporcionadas por el equipo médico encargado del estudio clínico, qué pasos deberá realizar y cuándo para conseguir una correcta limpieza de colon. Además, para que el equipo médico pueda analizar cómo ha hecho su preparación el paciente, PrepColon recoge las interacciones y eventos que ocurren entre la aplicación y el usuario para enviarlos a un servidor propio que, a través de su API, forma un correo electrónico con los datos recopilados y lo envía al equipo médico.

PrepColon está siendo utilizada actualmente por pacientes reales que, voluntariamente, han decidido participar en el estudio clínico que busca determinar si el uso de aplicaciones móviles para el apoyo a la preparación de colonoscopias son útiles para mejorar la detección temprana de lesiones producidas por el CCR. La aplicación está disponible públicamente en Google Play para dispositivos Android, y a través de la aplicación *TestFlight* para dispositivos iOS, para que los pacientes que participen en el estudio puedan descargarla en sus móviles de manera rápida y sencilla.

Palabras Clave

Colonoscopia, cáncer colorrectal, aplicación distribuida, aplicación móvil, aplicación multiplataforma, desarrollo software, desarrollo multiplataforma, Flutter, Android, iOS

Abstract

Colorectal cancer (CRC) has, among all cancer variants, the third highest incidence and the second highest mortality rate in Spain. For this reason, since 2014, Spain has implemented a screening program for the early detection of CRC to increase the chances of recovery. This screening is done by taking a stool sample from patients between 50 and 69 years of age to look for traces of blood invisible to the naked eye. If blood is found, the patient is scheduled for a colonoscopy. However, the patient must get properly prepared for the colonoscopy, otherwise, an inadequate colon cleansing could hinder the identification of wounds and lead to false negatives in the CRC detection.

Faced with this problem, and given the fact thar mobile devices are currently part of everyday life in our society, a group of members of the Digestive System Service at Río Hortega University Hospital (Valladolid), Virgen de la Concha Hospital (Zamora) and Medina del Campo's Hospital (Medina del Campo) decided to conduct a clinical study to determine whether the use of a mobile application that served as an assistant during colonoscopy preparation could help increasing the number of patients who attend their appointments with and adequate cleansing. This Final Degree Project proposes PrepColon, a cross-platform mobile application developed with the the framework Flutter to support users during their colonoscopy preparation process. For this reason, users must fill out a form requesting the date of their appointment and other medical data. This information, ranging from factors such as the user's body mass index to their regular medication usage, along with guidelines provided by the medical team in charge of the clinical study, allows the application to determine which steps should be taken to achieve a succesful colon cleansing. Furthermore, PrepColon collects the interactions and events that occur between the application and the user and sends the information to its own server which, through its API, generates an email with the collected data and sends it to the medical team so they can analize the patient's preparation.

PrepColon is currently being used by real patients who have decided to participate voluntarily in the clinical study that aims to determine if the usage of mobile applications for helping users in their colonoscopy preparation is useful in improving the early detection of wounds produced by CRC. This application is avaiable publicly on Google Play for Android devices, and through the *TestFlight* app for iOS devices, so patients who participate in the study can easily download it on their devices.

Keywords

Colonoscopy, colorectal cancer, distributed application, mobile application, cross-platform application, software development, cross-platform development, Flutter, Android, iOS

Agradecimientos

A Guillermo, Miguel y Asensio, por acompañarme y guiarme durante el desarrollo de este TFG. A Henar y Pilar, por su dedicación y compromiso con el proyecto. A Pablo, por compartir sus conocimientos conmigo. A Sofía, por permanecer a mi lado incondicionalmente. A mi familia y amigos, por el apoyo incondicional que me brindan diariamente. A los que aceptaron participar como *testers*, sin ellos no hubiera sido posible publicar la aplicación. Y a ti, que estás leyendo esto, por darle sentido a estas páginas.

A todos vosotros, gracias.

Índice general

1. Introducción **1**
- 1.1. Objetivos . . . 3
- 1.2. Metodología . . . 4
- 1.3. Estructura del documento . . . 5

2. Estado del arte **7**
- 2.1. Introducción . . . 7
- 2.2. Estudios clínicos . . . 8
- 2.3. Aplicaciones de apoyo a la preparación de colonoscopias . . . 9
- 2.4. Discusión . . . 11
- 2.5. Conclusiones . . . 13

3. Análisis **15**
- 3.1. Introducción . . . 15
- 3.2. Ejemplo de uso de la aplicación . . . 16
- 3.3. Requisitos funcionales . . . 17
  - 3.3.1. Historias de usuario . . . 17
  - 3.3.2. Casos de uso . . . 18
- 3.4. Requisitos no funcionales . . . 24
- 3.5. Conclusiones . . . 25

4. Diseño **26**
- 4.1. Introducción . . . 26
- 4.2. Arquitectura del sistema . . . 27
- 4.3. Aplicación móvil . . . 28
  - 4.3.1. Cliente . . . 28
  - 4.3.2. Interfaz de Usuario . . . 35
- 4.4. Servidor . . . 43
  - 4.4.1. API RESTful . . . 43
  - 4.4.2. Esquema de recursos . . . 44
- 4.5. Conclusiones . . . 46

5. Implementación **47**
- 5.1. Introducción . . . 47
- 5.2. Aplicación móvil . . . 47
  - 5.2.1. Tecnologías . . . 48
  - 5.2.2. Estructura de directorios . . . 51
  - 5.2.3. Clases controladoras . . . 52
  - 5.2.4. Implementación del patrón *State* . . . 53
  - 5.2.5. Configuración de la aplicación . . . 56
  - 5.2.6. Validación del código de paciente . . . 57

5.2.7. Envío de datos . . . 60
5.2.8. Internacionalización . . . 65
5.3. Servidor . . . 66
5.3.1. Estructura de directorios . . . 67
5.3.2. Validador e interfaz de usuario . . . 67
5.3.3. Implementación de operaciones . . . 68
5.4. Conclusiones . . . 74

6. Evaluación **76**
6.1. Introducción . . . 76
6.2. Casos de prueba . . . 77
6.2.1. Pruebas realizadas por distintos usuarios . . . 77
6.2.2. Pruebas modelando distintos tipos de pacientes . . . 78
6.3. Estudio clínico . . . 86
6.3.1. Resultados de colonoscopias . . . 86
6.3.2. Escala de usabilidad . . . 87
6.4. Conclusiones . . . 91

7. Conclusiones **92**
7.1. Conclusiones del proyecto . . . 92
7.2. Ideas de mejora . . . 93
7.3. Uso posterior al estudio clínico . . . 94

Referencias **100**

A. Información de referencia **101**
A.1. Laxantes . . . 102
A.2. Dieta . . . 103
A.3. Estado de las deposiciones . . . 103
A.4. Recomendaciones para el paciente . . . 104

B. Aplicación inicial **105**
B.1. Funcionamiento de la aplicación . . . 105
B.1.1. Pantalla principal . . . 105
B.1.2. Menú lateral . . . 106
B.1.3. Aspectos . . . 107
B.2. Código . . . 107
B.2.1. Directorio `/lib/l10n` . . . 107
B.2.2. Directorio `/lib/widgets` . . . 108
B.2.3. Directorio `/lib/tools` . . . 108
B.2.4. Directorio `/lib/services` . . . 108
B.2.5. Directorio `/lib/models` . . . 109
B.2.6. Directorio `/lib/screens` . . . 110
B.2.7. Directorio `/lib/states` . . . 110

C. Guía de colores **113**

D. Logo de la aplicación **114**

Índice de figuras

1.1. Incidencia y mortalidad del cáncer en Europa, 2022 [GCO]. 1
1.2. Incidencia y mortalidad de los diferentes tipos de cáncer en España, 2022[GCO]. 2
1.3. Esquema de los ciclos de desarrollo de PrepColon. 6

2.1. Iconos de (a) «My Colonoscopy», (b) «NurScope» y (c) «SB Colonoscopy Prep». 9
2.2. Capturas de pantalla de la aplicación «My Colonoscopy». (a) Formulario, (b) pantalla de recordatorios y (c) pantalla de información. 10
2.3. Guía de uso de la aplicación «NurScope» 11
2.4. Capturas de pantalla de la aplicación «SB Colonoscopy Prep». (a) Instrucciones para la toma del laxante, (b) información sobre la cita y (c) preparación interactiva. 12

3.1. Imágenes representativas de los posibles aspectos de las heces. 17

4.1. Esquema de la arquitectura propuesta para la aplicación PrepColon. 27
4.2. Diagrama de estados relacionados con el proceso de preparación. Para cualquiera de los estados desde `A.III` hasta `A.VI`, si la aplicación detecta que la preparación no es adecuada, se pasará al estado `A.VII` y luego al estado `VIII`. 31
4.3. Diagrama de estados relacionados con la dieta del usuario. 31
4.4. Diseño de los cuatro tipos de tarjeta. 37
4.5. Pantalla principal de PrepColon para los casos: (a) sin datos guardados, (b) antes de empezar a tomar el laxante y (c) tras detectar una mala preparación. 38
4.6. Pantalla «Mi cita» de la versión final de PrepColon para los casos: (a) sin datos guardados, (b) antes de empezar a tomar el laxante y (c) tras detectar una mala preparación. 39
4.7. Pantallas (a) «Mis datos» y (b) «Ajustes» 40
4.8. Pantalla «Ayuda» (a) antes y (b) después de comenzar la preparación. 41
4.9. Capturas de pantalla del formulario de la aplicación. 42

5.1. Cuota de mercado mundial de los sistemas operativos para dispositivos móviles de 2010 a 2024[Sta24]. 49
5.2. Árbol de directorios de la aplicación PrepColon. 51
5.3. Diagrama de clases de las clases controladoras de la aplicación PrepColon. 53
5.4. Ejemplo de uso de la aplicación PrepColon (a) antes y (b) después de introducir un código de paciente incorrecto. 59
5.5. Árbol de directorios del servidor PrepColon. 67
5.6. Ejemplo de la IU de Swagger para la operación `OP00`. 68
5.7. Información proporcionada por la IU de Swagger sobre la petición y la respuesta para la operación `OP00`. 70
5.8. Formulario de contacto disponible en la página web asociada a la aplicación PrepColon [Prep]. 74

6.1. Pregunta del formulario de hora de la cita en los casos (a) que se introduzca una hora fuera del horario de citas y (b) que se introduzca una hora válida. 84
6.2. Análisis del tipo de distribución de las muestras disponibles [Nav10]: (a) gráfica cuantil-cuantil y (b) gráfica de caja o *boxplot*. 89
6.3. Representación gráfica del análisis estadístico de los resultados del SUS de la aplicación PrepColon: superposición de valores estadísticos. 90

7.1. Gráficas estadísticas de la aplicación PrepColon entre marzo y julio de 2025, proporcionadas por Google: (a) descargas, (b) usuarios activos diarios y (c) usuarios totales. Google no proporciona estadísticas para el número de usuarios activos diarios anteriores al 22 de abril de 2025. 95

B.3. Imágenes representativas de los posibles aspectos de las heces. 107
B.1. Capturas de pantalla de la aplicación. (a) Pantalla principal sin datos guardados. (b) Pantalla principal tras completar el formulario. (c) Resumen con las respuestas del formulario. (d) Menú lateral desplegable. 111
B.2. Capturas de pantalla del formulario de la aplicación. 112

D.1. Icono de PrepColon. 114
D.2. Logo de PrepColon. 114

Índice de tablas

2.1. Comparativa de diferentes estudios clínicos sobre el uso de una aplicación para el apoyo a la preparación de colonoscopias. 8
2.2. Comparativa entre diferentes aplicaciones de apoyo a la preparación de colonoscopias. 14

3.1. Historias de usuario para la aplicación PrepColon. 18
3.2. Requisitos no funcionales de la aplicación PrepColon. 24

4.1. Catálogo de patrones de diseño. [Gam+95] 29
4.2. Estados de la aplicación PrepColon relacionados con el proceso de preparación de colonoscopias. 30
4.3. Estados de la aplicación PrepColon relacionados con las dietas del usuario. . . . 30
4.4. Frecuencia de las notificaciones en función de las horas restantes para completar la tarea asociada. 32
4.5. Listado de notificaciones enviadas por la aplicación PrepColon. Las notificaciones `NTF01` a `NTF05` sólo se enviarán si, según la información del formulario, el paciente requiere de una preparación intensiva. Las notificaciones `NTF06` a `NTF08` sólo se enviarán si, según la información del formulario, el paciente está tomando los medicamentos indicados. 33
4.6. Fechas de inicio y fin en las que se enviarán las notificaciones de la aplicación PrepColon. Las fechas de las tomas de los laxantes—tanto el principal como el adicional—dependerán de si la cita tiene lugar por la mañana o por la tarde. En caso de ser por la mañana, las tomas deberán realizarse antes. 34
4.7. Descripción de las diferentes tarjetas generadas por la aplicación. 36
4.8. Operaciones disponibles en el servidor de la aplicación PrepColon. 44
4.9. Descripción de las operaciones disponibles en el servidor de la aplicación PrepColon. 44

5.1. *Frameworks* multiplataforma para desarrollo móvil más populares según el porcentaje de desarrolladores que los usan a nivel mundial, de 2019 a 2023[Jet23]. . 51
5.2. Descripción del contenido de los directorios de la aplicación PrepColon. 52
5.3. Descripción de las clases contenidas en el directorio `/lib/menus` relacionadas con los estados de la aplicación PrepColon. La clase `NoPreparation()` puede recibir como parámetro un valor verdadero si se ha completado el formulario, en cuyo caso se mostrará el tiempo restante hasta que el usuario deba comenzar la toma de alguno de los laxantes; un valor falso si ya ha completado las tomas de los laxantes, o ningún valor si no ha completado el formulario, en cuyo caso se indicará que no hay ningún laxante registrado. 56

6.1. Número de pacientes y de resultados disponibles en los datos proporcionados por el equipo médico en mayo de 2025. 86

6.2. Resultados de las colonoscopias a los que ha podido acceder el autor de este documento, proporcionados por el equipo médico: puntuación individual de los segmentos del colon, puntuación de la Escala Boston y número de pólipos encontrados. 87
6.3. Formulario utilizado por el equipo médico para calcular el SUS de la aplicación PrepColon. 88
6.4. Resultados del SUS a los que ha tenido acceso el autor de este documento, proporcionados por el equipo médico. 88
6.5. Análisis estadístico de los resultados del SUS. 90
6.6. Rango de calificativos que se pueden asignar a un sistema en función del valor medio del SUS obtenido [BKa08; BKM09]. 90

7.1. Estadísticas de la aplicación PrepColon entre marzo y julio de 2025, proporcionadas por Google. 93

B.1. Medicamentos a tener en cuenta durante la preparación para la colonoscopia. . . 106
B.2. Estados y tipos de las tarjetas de avisos de [Fer23a]. 109
B.3. Estados de la aplicación y la clase asociada. 110

Índice de códigos

5.1. Fragmento de código del fichero `main.dart` en el que se registran instancias de las clases controladoras como «providers». 52
5.2. Fragmento de código del fichero `menu_state.dart` con la implementación del método `checkStatus()`. 53
5.3. Fragmento de código del fichero `menu_state.dart` con la implementación de los métodos `getMenuItems()`, `getDietItems()` y `getProductItems()`. 54
5.4. Fragmento de código del fichero de configuración `appconfig.json`. 56
5.5. Fragmento de código del fichero `validation_screen.dart` que muestra la implementación del validador del campo de texto de introducción de código de paciente. 58
5.6. Fragmento de código del fichero `validation_screen.dart` que muestra la implementación del botón «Aceptar». Al pulsar este botón se comprueba la validez del código introducido y, de ser válido, se pasa a la siguiente pantalla. 59
5.7. Fragmento de código del fichero `network_manager.dart` que muestra los métodos `checkServerConnection()`, para comprobar la conexión con el servidor, y `userValidation()`, para hacer la solicitud al servidor que compruebe la validez del código del usuario. 60
5.8. Fragmento de código del fichero `network_manager.dart` que muestra la implementación del método que envía los datos de la aplicación al servidor para su reenvío al equipo médico. 61
5.9. Fragmento de código del fichero `file_manager.dart` en el que se muestra la implementación de los métodos `convertJsonToCsv()` y `convertJsonListToCsv()`. Estos métodos también «traducen» valores para facilitar su entendimiento por parte de personas poco familiarizadas con la programación de forma que `null` se convierte en «nada», `true` en «sí», y `false` en «no». La clase `FileManager` es una clase auxiliar diseñada para facilitar el manejo de ficheros por parte de la aplicación. 62
5.10. Fragmento de código del fichero `main.dart` en el que se inicializa la clase `Workmanager` para el funcionamiento en segundo plano. Dado que el uso de esta clase está pensado para enviar los datos después de la colonoscopia, tratará de borrar los datos almacenados una vez se complete el envío mediante el método `wipeData() de la clase FileManager`. 64
5.11. Fragmento de código que permite registrar la tarea de envío de datos en segundo plano. 64
5.12. Fragmento de código del fichero `main.dart`. Para la aplicación PrepColon, actualmente sólo los idiomas español e inglés están soportados. 65
5.13. Ejemplo de uso de la librería `flutter_localizations`. Si el dispositivo del usuario está configurado en inglés, en la aplicación se mostrará el texto «Hello World!». En cambio, si el idioma es español, se verá el texto «Hola Mundo!». 66
5.14. Fragmento del fichero `app_es.arb`. 66
5.15. Definición de la operación `OP00` para la IU de Swagger. 69

5.16. Definición del esquema `ServerResponse` que especifica el formato de las respuestas del servidor. 69
5.17. Fragmento de código del fichero `app.js` en el que se inicializan las rutas a los recursos del servidor. 70
5.18. Expresión regular para verificar el formato de los códigos de paciente. 71
5.19. Implementación de la función `validateUser()` para la validación de los códigos de paciente. 71
5.20. Fragmento de código de la función `sendUserData()` en el que se genera y se envía el correo electrónico con los datos de usuario. 72

"No importa cuántas veces empieces de nuevo, cada viaje es único."

— *Outer Wilds*

Capítulo 1

Introducción

El cáncer es, actualmente, la principal causa de muerte por causas naturales a nivel mundial en todos los rangos de edad; además de una de las enfermedades con mayor índice de mortalidad que se conoce. Tan sólo en 2020 se registraron aproximadamente 10 millones de fallecimientos provocados por algún tipo de cáncer [WHO]. En el año 2022 se diagnosticaron cerca de 20 millones de nuevos casos y se produjeron, de nuevo, casi 10 millones de muertes relacionadas con esta enfermedad [GCO].

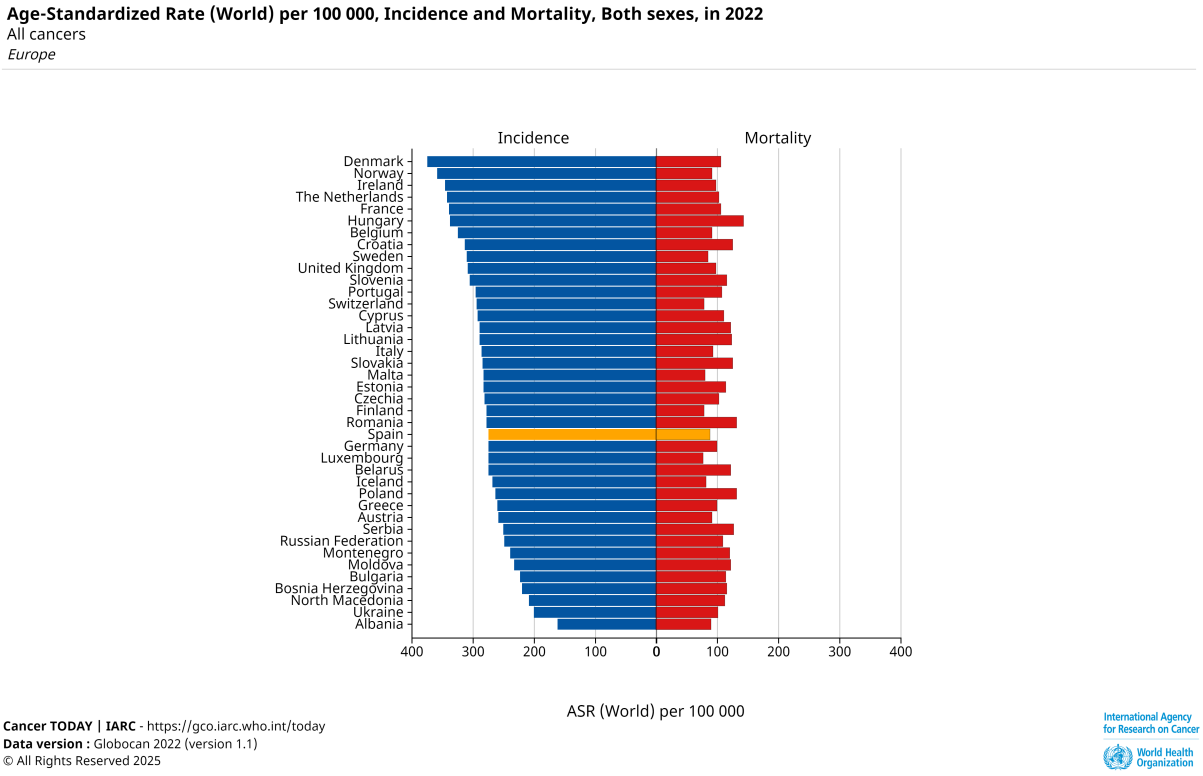


Fig. 1.1: *Incidencia y mortalidad del cáncer en Europa, 2022 [GCO].*

Una de las principales características de estos tumores malignos es su capacidad para desarrollarse en prácticamente cualquier órgano del cuerpo humano, lo que da lugar a múltiples variantes con diferentes síntomas y gravedades. Si nos fijamos en la Figura 1.2, uno de los tipos de cáncer con mayor incidencia y con mayor mortalidad es el cáncer colorrectal (CCR). Esta

enfermedad afecta a más de 30 de cada 100.000 españoles, siendo el tercer tipo de cáncer más incidente y el segundo más mortal a nivel nacional. No obstante, al igual que ocurre con los demás tipos de cáncer, la probabilidad de curar a un paciente diagnosticado con CCR puede llegar a ser muy elevada si se detecta y se comienza a tratar en las primeras fases de su tratamiento [WHO]. Es por esto que, en 2014, se creó el Programa de Cribado de Cáncer Colorrectal [San25] con el objetivo de ofrecer pruebas que permitan esta detección precoz, que puede resultar determinante en la evolución de un paciente que haya desarrollado cáncer, a toda la población española. Este programa, incluido en la cartera común de servicios del Sistema Nacional de Salud según [Est14], especifica que el cribado de cáncer colorrectal debe realizarse cada 2 años en personas que pertenezcan al rango de edad de 50 a 69 años, mediante una prueba de sangre oculta en heces. Si se detecta sangre en la muestra de heces proporcionada por el paciente, el siguiente paso sería la realización de una colonoscopia para determinar si el origen del sangrado pudiera estar relacionado con el CCR.

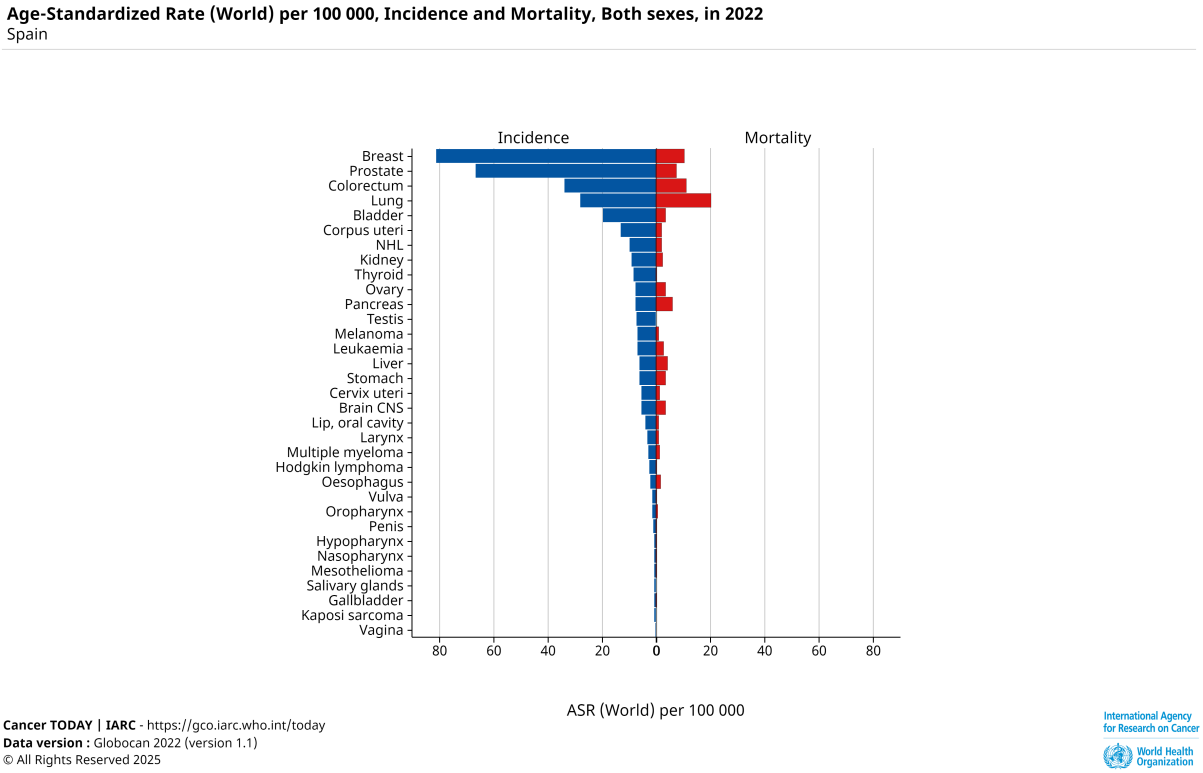


Fig. 1.2: *Incidencia y mortalidad de los diferentes tipos de cáncer en España, 2022[GCO].*

La problemática aparece cuando, en el momento de realizarse la colonoscopia, el paciente no ha seguido correctamente las indicaciones para lograr una correcta limpieza del colon. Esto dificulta enormemente la detección de lesiones, por lo que se vuelve imposible determinar con certeza la causa de la sangre observada en las heces. De acuerdo con [Pan22], en esta situación se recomienda repetir la colonoscopia en el intervalo de un año. Según un estudio realizado de forma conjunta por diferentes hospitales del país, cerca del 60 % de los pacientes que tuvieron que repetir la prueba lograron una limpieza óptima en su segunda colonoscopia. Además, más de la mitad de las lesiones detectadas se observaron en colonoscopias repetidas [Pan22]. Según otro estudio realizado en Estados Unidos, se observa también cómo más del 20 % de los pacientes que acuden a una colonoscopia llevan una mala preparación y, de ellos, el 18 % admitió haber incumplido las indicaciones que recibieron de su médico respecto a cómo prepararse para la prueba [Nes+01].

Son estos datos los que preocupan a los profesionales sanitarios, quienes sienten que los métodos tradicionales de dar las indicaciones a los pacientes—esto es, de forma hablada y mediante infografías físicas—no son suficientes, y detectan la necesidad de nuevas formas de informar a la población. Para ello, hay quienes han decidido aprovechar el hecho de que, actualmente, una gran cantidad de personas poseen un dispositivo móvil que llevan consigo la mayor parte del tiempo para complementar las indicaciones orales y escritas con aplicaciones móviles. Algunas de estas aplicaciones han sido puestas a prueba en diferentes estudios, demostrando que el uso de *software* permite a los pacientes realizar una mejor preparación de cara a sus colonoscopias [Wal+21; Zhu+23; WSD17; Zan+21]. Sin embargo, todas estas aplicaciones presentan alguna de las limitaciones que se listan a continuación:

- No están disponibles públicamente.
- No son multiplataforma.
- No son accesibles fuera de un área geográfica concreta.
- Tienen un alcance lingüístico limitado.
- Muestran indicaciones generalizadas, sin adaptarse a las necesidades individuales del usuario.
- Requieren que el usuario establezca las fechas de los recordatorios.

La motivación de este Trabajo de Fin de Grado (TFG) deriva de un proyecto de investigación planteado por miembros del Servicio de Aparato Digestivo de los centros Hospital Universitario Río Hortega (Valladolid), Hospital Virgen de la Concha (Zamora) y Hospital de Medina del Campo (Medina del Campo), liderado por el hospital vallisoletano. Dichas personas han detectado los puntos débiles anteriormente mencionados en aplicaciones ya existentes y han determinado la necesidad de una aplicación móvil que supere esas restricciones. Una vez dicha aplicación esté completada, se probará su efectividad en el estudio clínico «PrepColon APP», enmarcado dentro del proyecto de investigación «Validación e implementación de una aplicación móvil para la limpieza colónica personalizada y su repercusión en la calidad de la colonoscopia de cribado de cáncer colo-rectal» liderado por el Hospital Universitario Río Hortega y aprobado por el Comité de Ética de la Investigación con Medicamentos (CEIm) de las Áreas de Salud de Valladolid. Para ello, además, se tomará como punto de partida el proyecto descrito en [Fer23a]. Aunque en el Anexo B se realiza un análisis más detallado de esta primera aplicación, el hecho de que estuviera incompleta y que faltasen varias funcionalidades clave, además de la inexistencia de un servidor que recogiese los datos necesarios para llevar a cabo el estudio anteriormente mencionado, hizo necesario desarrollar una aplicación más sólida y estable. En el presente TFG se propone un sistema que permita realizar el estudio clínico, reutilizando el código de la versión inicial cuando sea posible.

1.1. Objetivos

El principal objetivo de este TFG es desarrollar una aplicación móvil multiplataforma (PrepColon) que cumpla con una serie de requisitos definidos por un equipo médico profesional formado por miembros del Servicio de Aparato Digestivo de tres hospitales castellanoleoneses—Hospital Universitario Río Hortega (Valladolid), Hospital Virgen de la Concha (Zamora) y

Hospital de Medina del Campo (Medina del Campo)—, los cuales están detallados en el Anexo A. Esta aplicación deberá servir como una suerte de asistente virtual que ayude a mejorar la limpieza colorrectal de los pacientes que deban someterse a colonoscopias, así como para reducir el número de personas que no asisten, de manera injustificada, a las mismas. Para ello el programa deberá proporcionar al usuario una serie de indicaciones y recordatorios, similares a los que le daría un profesional sanitario, que le permitan acudir a su cita médica con una preparación adecuada que permita identificar más fácilmente posibles indicios de CCR.

Teniendo en cuenta que el rango de edad de los potenciales usuarios está entre 50 y 69 años, la aplicación deberá, además, mostrar una interfaz gráfica intuitiva, sencilla de utilizar, y desde la cual el usuario pueda acceder sin problemas a toda la información relevante que pueda necesitar. Además, el programa no se limitará a mostrar información generalizada, sino que recopilará datos, tanto introducidos por el usuario como derivados de la propia interacción del usuario con la aplicación, para poder determinar las necesidades específicas de cada persona y, de esta forma, proporcionar las indicaciones que mejor se adapten a la situación personal de cada usuario.

Además, para realizar el estudio clínico que determine la efectividad de PrepColon, será imprescindible implementar un servidor que recoja los datos—totalmente anonimizados e identificados mediante un código de paciente aleatorio que se proporcionará a un grupo controlado de usuarios que acepten participar voluntariamente en el estudio—, y los envíe al equipo médico para su posterior análisis.

1.2. Metodología

El proyecto PrepColon se ha llevado a cabo siguiendo una metodología ágil. Esta metodología se basa en [Bec+01], donde se recogen una serie de principios basados en cuatro axiomas principales[SS05]:

- Los individuos y las interacciones son más importantes que los procesos y las herramientas.
- Tener un producto funcional es prioritario frente a la documentación excesiva—por lo que resulta imprescindible tener un código limpio, claro y autodescriptivo.
- La clave es la involucración del cliente en el proceso de desarrollo, son necesarias la colaboración y la comunicación continuas con él.
- El producto debe ser flexible y poder adaptarse fácilmente a cambios que surjan durante el desarrollo.

Con este enfoque se definen tres roles principales[SS05]:

- El **cliente** es quien establece los requisitos del producto y ofrece realimentación sobre las diferentes versiones del producto para identificar posibles errores antes de tener una versión final del mismo.
- Los **desarrolladores** son los encargados de interactuar con el cliente para extraer los requisitos funcionales y no funcionales a partir del producto que quiere el cliente. Además,

recogen la realimentación del propio cliente para lograr un producto que se adecúe lo mejor posible a sus requerimientos.

- Los **directores** garantizan que la comunicación entre desarrolladores y clientes sea efectiva y eficaz.

En el desarrollo de este TFG, se ha seguido una metodología ágil en la que el autor del presente documento ha tomado el rol de desarrollador, los tutores del mismo han tomado el rol de directores, y el equipo médico con el que se ha colaborado ha tenido el rol de cliente.

Una técnica ampliamente utilizada en la metodología ágil consiste en programar ciclos de desarrollo periódicos [SS05] con el objetivo de obtener realimentación más o menos frecuente por parte del cliente. En este sentido, los ciclos han consistido en reuniones cada dos o tres semanas en las que se mostraba una nueva versión de la aplicación y se decidían los objetivos más prioritarios a cumplir en la siguiente versión. Asimismo, entre dos ciclos adyacentes, el cliente probaba la última versión disponible para, en la siguiente reunión, aportar realimentación sobre dicha versión. De esta manera, no solo se ha simplificado el proceso de desarrollo de la aplicación, al irse añadiendo funcionalidades paulatinamente; sino que también se han podido subsanar errores antes de llegar a la versión final. Concretamente, para este proyecto, se han realizado 24 ciclos que se pueden distribuir en 4 fases diferenciadas. En la Figura 1.3 se presenta un gráfico en el que puede verse la duración aproximada de cada una de estas fases.

- **Fase I:** En esta primera fase el foco estaba puesto en el análisis. Se estudió la aplicación de [Fer23a] para ver qué funcionalidades faltaban por implementarse, y cuáles se podían reutilizar en PrepColon; y se definieron los principales requisitos que debía cumplir la aplicación.
- **Fase II:** En la segunda fase se implementaron las funcionalidades necesarias de la aplicación, así como el servidor y la página web. Además, en esta fase se realizó un rediseño de la interfaz de usuario, ya que hasta este momento se había mantenido la misma que en la versión inicial.
- **Fase III:** Finalmente, en esta última fase se centró el esfuerzo en poner a prueba la aplicación para detectar errores o puntos de mejora. Para ello, se obtuvo una licencia de desarrollador tanto de Google como de Apple para poder distribuir más fácilmente la aplicación entre los *testers*.
- **Fase IV:** Tras las pruebas realizadas sobre las diferentes versiones de la aplicación, una vez se consiguió una versión estable, comenzó el estudio clínico para el que PrepColon fue diseñado. En esta fase, pacientes reales comenzaron a utilizar la aplicación, de manera voluntaria, para ayudar al equipo médico a determinar si el uso de una aplicación como esta puede mejorar los resultados de las colonoscopias.

1.3. Estructura del documento

La estructura de este documento pretende reflejar las diferentes fases del desarrollo del proyecto PrepColon, de manera que le sea más sencillo al lector entender todo el proceso que dió como resultado dicha aplicación. Es por esto que, en el **Capítulo 2 – Estado del arte** se

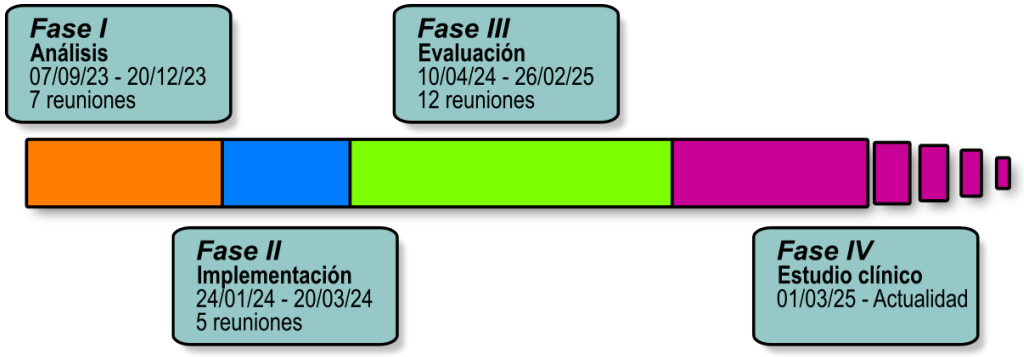


Fig. 1.3: *Esquema de los ciclos de desarrollo de PrepColon.*

comienza analizando qué aplicaciones de propósito similar existen, con el objetivo de tomarlas como fuentes de ideas para el diseño de PrepColon. También en este capítulo se hace un breve estudio de la aplicación creada en [Fer23a], pues será el punto de partida para este proyecto.

En el **Capítulo 3 – Análisis** se definen los requisitos que debe cumplir el sistema, tanto la parte cliente, es decir, la aplicación móvil, como la parte servidora; para lograr los objetivos propuestos. En este capítulo también se incluirán los casos de uso que detallarán el comportamiento esperado de la aplicación y un ejemplo que ilustre cómo la utilizaría un posible usuario real.

Una vez definidos los requisitos, en el **Capítulo 4 – Diseño** se explica, por un lado, cómo evolucionó la interfaz de usuario desde unas primeras versiones hasta la versión definitiva, y por otro lado, qué arquitectura se definió para la lógica del cliente y para el servidor.

Tras esto, en el **Capítulo 5 – Implementación** , se describe, a raíz de los requisitos y de la arquitectura propuesta, cómo se ha desarrollado el código que permite el correcto funcionamiento de todos los elementos que componen el sistema. También en este capítulo se estudiarán las diferentes opciones relativas a las tecnologías que pueden ser utilizadas para desarrollar la aplicación.

Finalmente, en el **Capítulo 6 – Evaluación** se muestran los resultados tanto de las pruebas realizadas con la aplicación como del análisis realizado por el equipo médico, y en el **Capítulo 7 – Conclusiones** se hace una reflexión sobre la eficacia de la aplicación PrepColon en vista de los resultados obtenidos.

Adicionalmente, se incluye documentación complementaria distribuida en varios anexos: el **Anexo A – Información de referencia** describre la información proporcionada por el equipo médico con el que se colaboró para la realización de este proyecto, y a partir de la cual se definieron los requisitos que debían cumplirse; en **Anexo B – Aplicación inicial** se analiza el estado de la aplicación de [Fer23a], que derivó en la actual PrepColon; en el **Anexo C – Guía de colores** se listan los colores a los que se hace referencia en este documento para que el lector del mismo pueda identificarlos más fácilmente; y, por último, en el **Anexo D – Logo de la aplicación** se muestra el icono y el logotipo de la aplicación, creados en colaboración con una empresa externa.

Capítulo 2

Estado del arte

Resulta clave conocer qué aplicaciones existen en el mercado que sirvan de apoyo a las personas que deban realizarse colonoscopias. De esta forma, se pueden analizar los puntos fuertes y las debilidades de estos programas para determinar qué características puede aprovechar PrepColon y establecer los aspectos que la diferenciarán de otras aplicaciones similares. En este capítulo se hará un breve análisis de algunas de estas aplicaciones, así como de la versión inicial en la que se basa PrepColon. Tras este análisis, se hará una reflexión sobre los motivos por los que estas aplicaciones no cumplen con los requisitos propuestos por el equipo médico.

2.1. Introducción

Los profesionales sanitarios especializados en el sistema digestivo detectan que las indicaciones orales y escritas que se han venido dando tradicionalmente a los pacientes que han de someterse a colonoscopias resultan insuficientes, en vista de la cantidad de pacientes que, o bien no se presentan a sus citas médicas sin previo aviso, o bien acuden con una limpieza inadecuada que impide detectar correctamente posibles pólipos en el colon. Ante esta problemática hay quienes han pensado que, en una sociedad donde predominan los dispositivos móviles, podría ser buena idea complementar las indicaciones orales y escritas tradicionales con información transmitida a través de *software* instalado en el *smartphone* del paciente [Wal+21; WSD17; Zhu+23; Zan+21].

Por esta razón, en los últimos años han surgido diversas aplicaciones móviles cuyo objetivo es ayudar a lo largo del proceso de preparación para que acudan a sus colonoscopias con una limpieza más adecuada. Algunas de estas aplicaciones han sido evaluadas en estudios que han puesto a prueba su eficacia, demostrando ser útiles en su propósito de mejorar los resultados de las colonoscopias. En estos estudios se midieron los resultados siguiendo la Escala de Preparación Intestinal Boston (BBPS, *Boston Bowel Preparation Scale*) [Lai+09], en la que se asigna una puntuación entre 0 y 3 a cada segmento del colon—colon derecho, colon transversal y colon izquierdo— en función del nivel de limpieza que presente. La puntuación BBPS será la suma de todas ellas, dando como resultado un valor entre 0 y 9. Se considera que la limpieza es adecuada si el paciente obtiene una puntuación BBPS total de 6 o más puntos, y además una puntuación

individual en cada segmento del colon de al menos 2 puntos.

2.2. Estudios clínicos

Haciendo una búsqueda rápida de los términos «*preparation colonoscopy app*» en *PubMed* [NLM], una base de datos de libre acceso que contiene artículos de investigación científica relacionados con el campo de la medicina, se pueden encontrar entradas que muestran resultados de estudios similares al que el equipo médico de los hospitales Río Hortega, Virgen de la Concha y Medina del Campo pretenden realizar a través de la aplicación PrepColon.

Por poner algunos ejemplos, en [Zan+21] se habla de un estudio realizado en Países Bajos en el que participaron 173 pacientes, de los cuales la mitad utilizó una aplicación móvil y la otra mitad no. En el grupo «app» se obtuvo una puntuación BBPS media de 8.3, con un 99 % de pacientes que acudieron a su cita con una limpieza adecuada; frente a la puntuación media de 7.9 en el grupo que no usó la aplicación, donde el 95 % de los pacientes presentaron una limpieza adecuada.

En [Zhu+23] se utilizó una aplicación que incorporaba un sistema de inteligencia artificial predictiva que evaluaba imágenes que los pacientes debían tomar de sus defecaciones y así generar instrucciones en tiempo real que ayudasen a mejorar la preparación para la colonoscopia. De las 500 personas que participaron voluntariamente en el estudio, el grupo que utilizó la aplicación obtuvo una puntuación media de 6.74, presentando casi el 90 % de los pacientes una limpieza adecuada, de los cuales cerca del 30 % mostraron una limpieza excelente[1]. En el grupo de control se obtuvo una puntuación media de 5.97, y un 65 % de pacientes con limpieza adecuada, de los cuales el 20 % acudieron a su cita con una preparación excelente.

En otro estudio similar realizado en Alemania [Wal+21; WSD17], los pacientes que utilizaron la aplicación lograron una puntuación media de 7.6 frente a los 6.7 puntos obtenidos en el grupo de control. En el grupo «app» hubo un 92 % de limpiezas adecuadas y un 60 % de limpiezas excelentes, frente a las 83 % adecuadas y 34 % excelentes del grupo de control.

	Grupo de control	**Grupo de la app**
Estudio	**Número de pacientes**	**Número de pacientes**
[Wal+21]	243	246
[Zan+21]	86	87
[Zhu+23]	247	253

	Grupo de control		**Grupo de la app**	
Estudio	**BBPS medio**	**Preparaciones adecuadas(%)**	**BBPS medio**	**Preparaciones adecuadas(%)**
[Wal+21]	6.70	83.10	7.60	92.30
[Zan+21]	7.90	95.00	8.30	99.00
[Zhu+23]	5.97	65.59	6.74	88.54

Tabla 2.1: *Comparativa de diferentes estudios clínicos sobre el uso de una aplicación para el apoyo a la preparación de colonoscopias.*

[1] Puntuación BBPS total mayor o igual que 8

En la Tabla 2.1 puede verse claramente que el uso de una aplicación móvil puede mejorar la preparación de los pacientes para sus colonoscopias. En la Sección 2.3 se presentarán algunas aplicaciones diseñadas para ese propósito. En la Sección 2.4 se discutirá la razón por la que estas aplicaciones no cumplen con los requisitos propuestos para la aplicación PrepColon por el equipo médico con el que se colaboró para su desarrollo.

2.3. Aplicaciones de apoyo a la preparación de colonoscopias

Haciendo una búsqueda por las tiendas de aplicaciones se puede encontrar rápidamente software relacionado con las colonoscopias. Sin embargo, es necesario filtrar los resultados de la búsqueda para descartar las aplicaciones meramente informativas, cuyo público objetivo está formado principalmente por profesionales sanitarios, de las aplicaciones enfocadas específicamente a asistir a los usuarios durante su preparación. Tras este filtrado las opciones disponibles en España se reducen mucho.

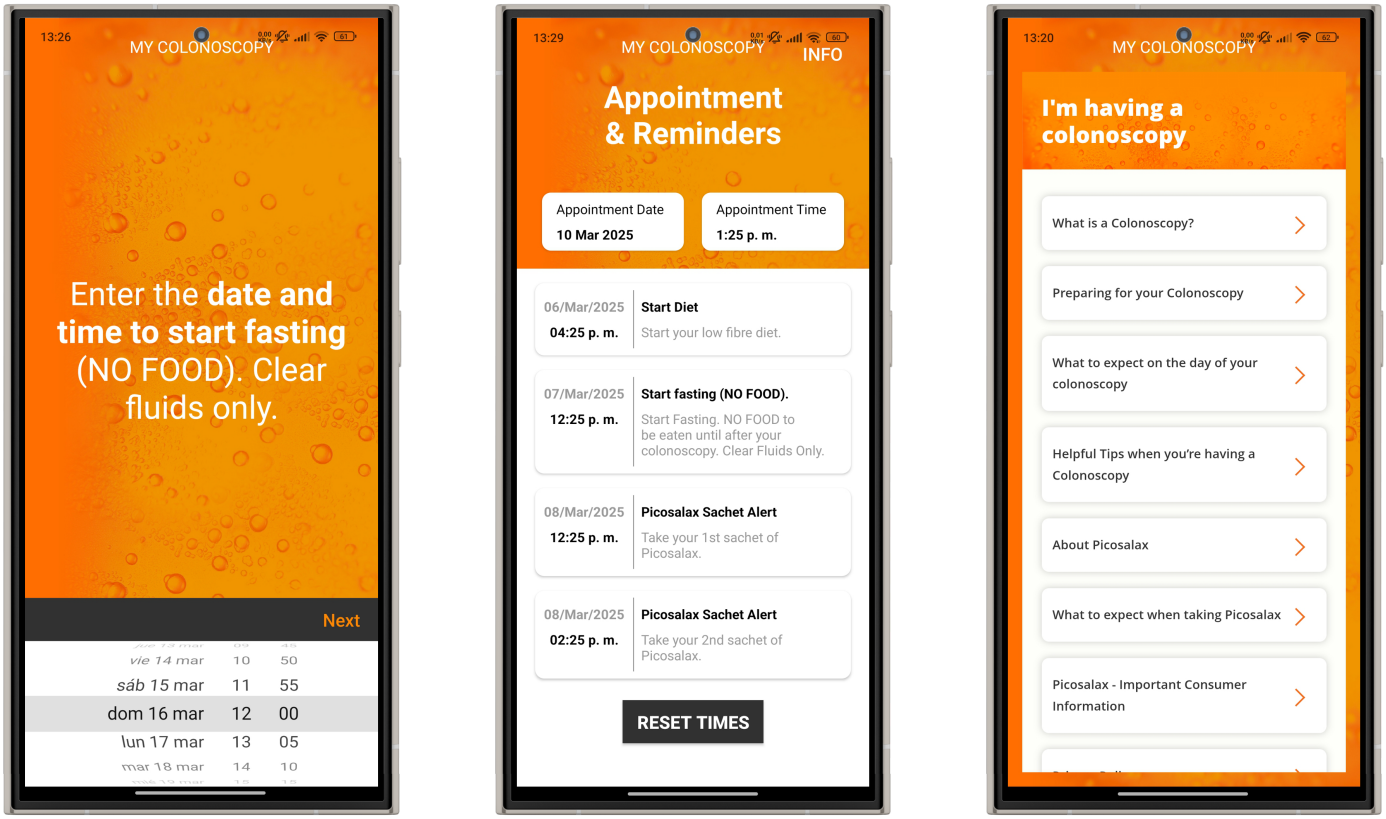
(a) **(b)** **(c)**

Fig. 2.1: *Iconos de (a) «My Colonoscopy», (b) «NurScope» y (c) «SB Colonoscopy Prep».*

En la tienda de Google se puede encontrar la aplicación «My Colonoscopy». Este programa muestra, inicialmente, mensajes que describen brevemente su propósito y funcionalidad. Posteriormente, el usuario pasa a una segunda pantalla en la que un mensaje hace hincapié en la importancia de tomar líquidos claros durante la preparación, y desde la cual se puede o bien comenzar la preparación o bien acceder información relevante sobre las colonoscopias y sobre el laxante a utilizar. Una vez el usuario decide comenzar la preparación, la aplicación solicita la fecha de la cita, la fecha en la que debe comenzar cada una de las dietas y las fechas en las que debe tomar el laxante. Tras esto, el usuario debe confirmar que los datos introducidos son correctos antes de que la aplicación pase a una última pantalla en la que muestra las tareas a realizar y sus correspondientes fechas. Una vez la preparación ha comenzado, la aplicación envía periódicamente recordatorios en forma de notificaciones.

Otra aplicación es «Upadr», para la cual hace falta iniciar sesión con un correo electrónico. Esta aplicación solicita al usuario indicar el procedimiento al que se va a someter, así como la fecha y la hora en la que se realizará. Una vez comienza la preparación, se ofrece al usuario información detallada sobre las diferentes fases del proceso de preparación y se le notifica con recordatorios personalizados de las tareas que debe completar.

«NurScope» es una aplicación japonesa en la que el usuario debe tomar imágenes al váter tras defecar durante su preparación para la colonoscopia. El programa evaluará y puntuará el estado de las heces, de manera que el propio usuario pueda saber si está realizando correctamente la limpieza colorrectal.



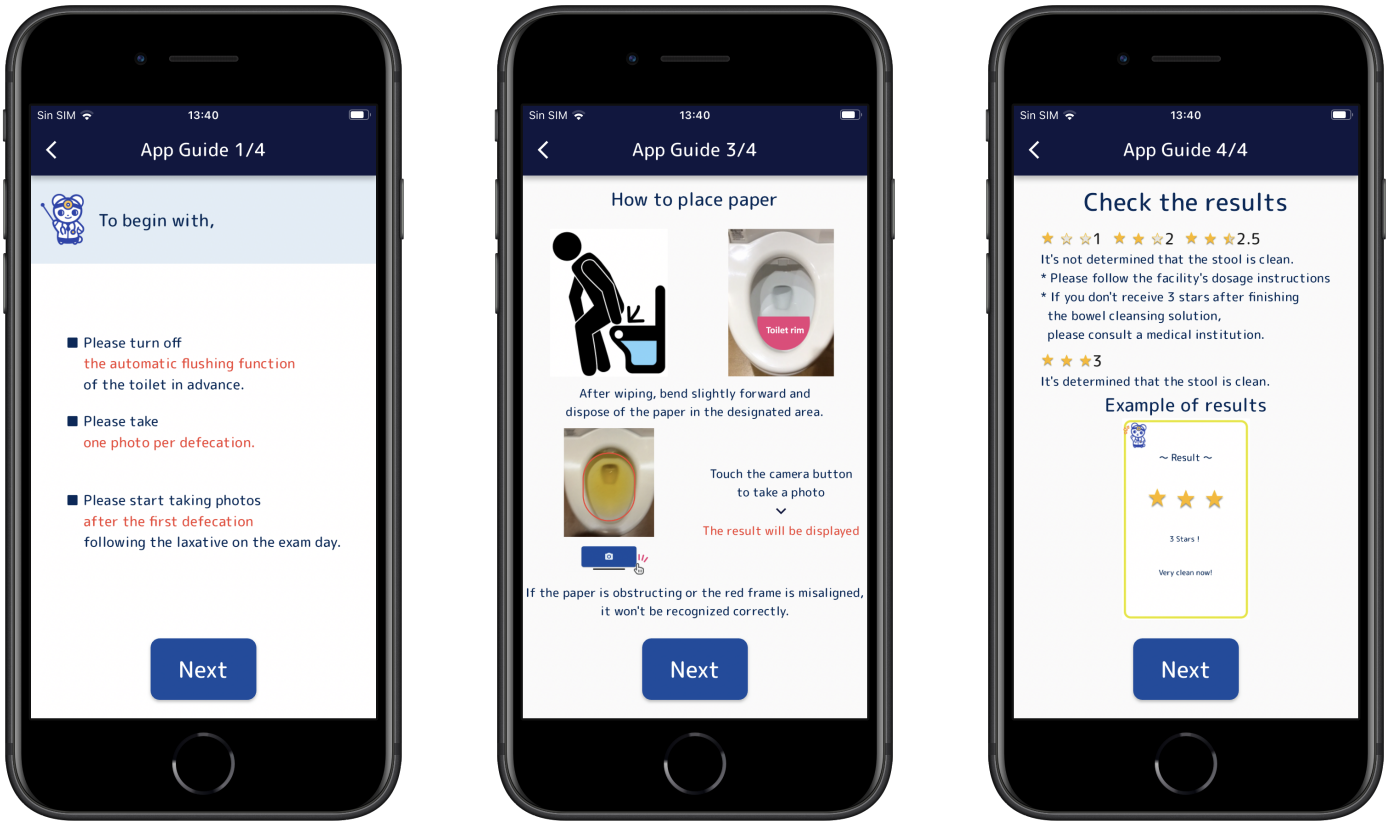
(a) **(b)** **(c)**

Fig. 2.2: *Capturas de pantalla de la aplicación «My Colonoscopy». (a) Formulario, (b) pantalla de recordatorios y (c) pantalla de información.*

En iOS se puede encontrar, aparte de las anteriores, alguna alternativa más. Una primera aplicación a mencionar es «Colonoscopy Prep Reminder», que permite una comunicación directa entre el médico y el paciente. El paciente deberá identificarse con su número de teléfono para acceder a información sobre colonoscopias, información sobre su cita, y a recordatorios personalizados generados por el Departamento de Sistema Digestivo de su hospital.

Otra aplicación prometedora es «SB Colonoscopy Prep», que destaca sobre las demás gracias a su preparación interactiva, una funcionalidad que permite guiar al usuario más fácilmente durante las tomas del laxante. Para ello, muestra por pantalla las instrucciones que deben seguirse en cada paso—por ejemplo, disolver el producto en una determinada cantidad de agua—, y será el propio usuario quien indique que ha completado ese paso pulsando un botón. Además, mediante un temporizador, el usuario puede ver el tiempo restante hasta que comience la siguiente fase de la preparación.

Para finalizar la sección cabe mencionar la versión inicial de PrepColon, descrita en [Fer23a]. Dicha aplicación prometía cumplir con los requisitos propuestos, derivados de la información que se puede leer en el Anexo A. La aplicación recogía información médica del usuario que pudiera ser relevante para el proceso de limpieza colorrectal, así como la fecha y la hora de la cita médica, y generaba avisos personalizados en función de las necesidades del propio usuario. Además, permitía llevar un registro del estado de las deposiciones de forma que se pudiera determinar si la limpieza estaba siendo eficaz o si era necesario reprogramar la colonoscopia. No obstante,



(a) **(b)** **(c)**

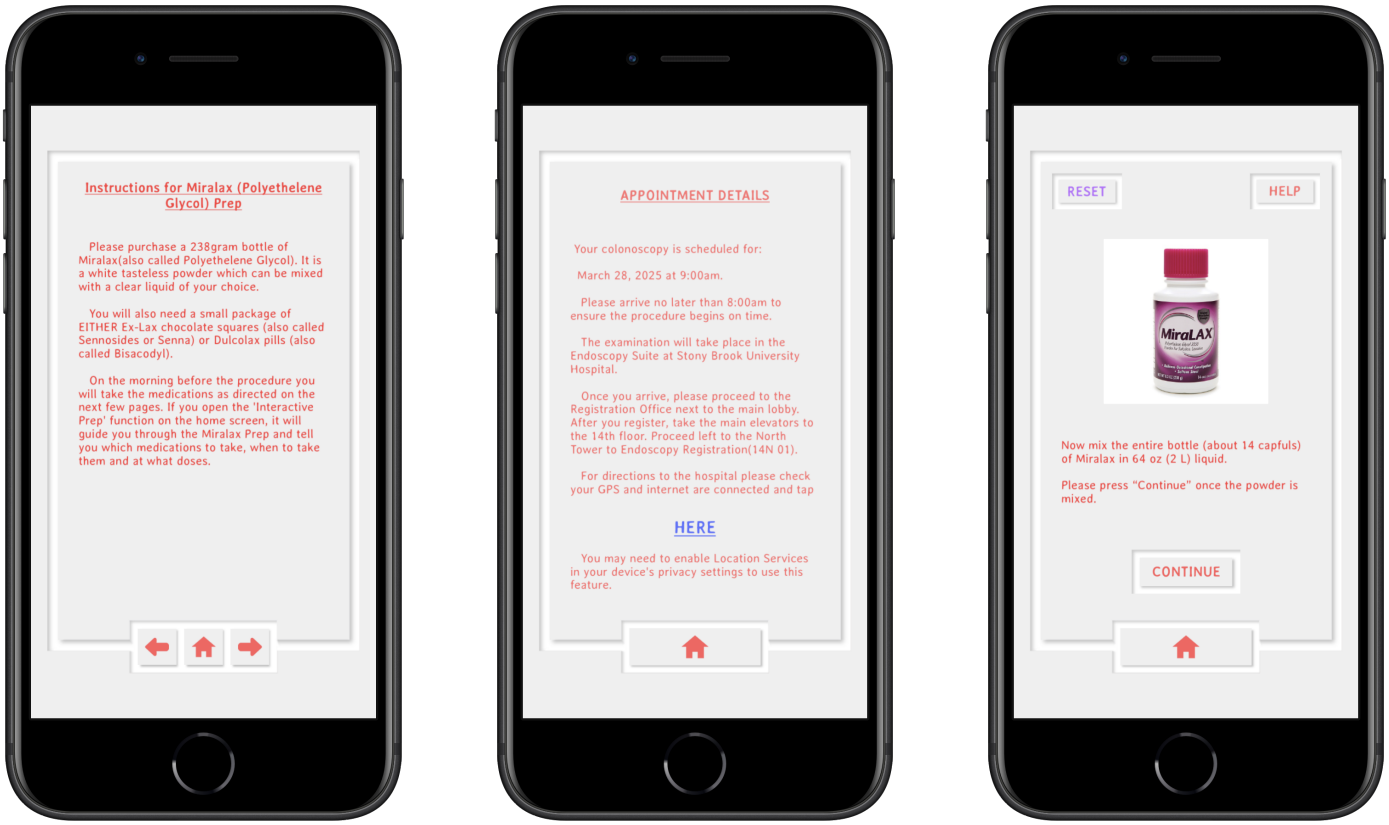
Fig. 2.3: *Guía de uso de la aplicación «NurScope»*

a pesar de que esta aplicación tenía potencial para destacar sobre otras aplicaciones similares, debido precisamente a su capacidad para adaptarse al usuario, no llegó a terminarse. La falta de implementación de varios elementos de la interfaz de usuario dejaban la aplicación, más pronto que tarde, inutilizable, y tampoco había un servidor que recogiese los datos necesarios para realizar el estudio clínico que pruebe la efectividad de esta aplicación. En el Anexo B se hace un análisis más detallado de esta versión.

2.4. Discusión

El equipo médico con el que se ha colaborado para la realización de este proyecto buscaba una aplicación disponible tanto para Android como para iOS, en idioma español, y que fuese accesible desde España. Esta aplicación, además, debía poder permitir al usuario seleccionar diferentes tipos de soluciones laxantes—aquellos utilizados en los hospitales con los que se colaboró para el desarrollo de este proyecto—, y ser capaz de adaptarse a sus necesidades para mostrar unas indicaciones u otras en función de variables como, por ejemplo, el índice de masa corporal (IBM) del paciente. Esta aplicación debía poder enviar recordatorios al usuario para que éste no olvidase las tareas a realizar en cada fase de la preparación, así como mostrar información relevante sobre el proceso.

Las aplicaciones utilizadas en [Zhu+23], [Wal+21] y [Zan+21] parecen estar disponibles úni-



(a) **(b)** **(c)**

Fig. 2.4: *Capturas de pantalla de la aplicación «SB Colonoscopy Prep». (a) Instrucciones para la toma del laxante, (b) información sobre la cita y (c) preparación interactiva.*

camente en sus países de origen—estos son, respectivamente, China, Alemania y Países Bajos—, por lo que quedan descartadas de la búsqueda. También hay que descartar las aplicaciones «Upadr» y «Colonoscopy Prep Reminder», ya que para usarlas parece ser necesario que el usuario esté registrado en alguna base de datos, probablemente del sistema de salud de Estados Unidos. «NurScope», por su parte, únicamente evalúa el estado de las deposiciones del usuario, por lo que más allá de hacerle saber si la limpieza está siendo exitosa o no, no aporta ninguna otra funcionalidad que ayude al proceso de preparación para la colonoscopia.

«My Colonoscopy» parecía prometedora, pero presenta varios inconvenientes que la sacan de la lista de posibles candidatos. Para empezar, la información que muestra es información genérica, no está enfocada al perfil que pueda tener el usuario. Además, parece diseñada exclusivamente para pacientes que deban tomar Picosalax®, un laxante que no está contemplado en la lista de laxantes que puede verse en el Anexo A. Esta aplicación solicita las fechas y las horas en las que deben generarse los avisos, datos que el usuario pordría olvidar o no recordar correctamente. La barrera del idioma también es un punto en contra importante, pues únicamente está disponible en inglés.

«SB Colonoscopy Prep» es, quizás, la más completa de todas. Genera avisos personalizados, proporciona mucha información y su funcionalidad de preparación interactiva facilita enormemente su uso. Sin embargo, tampoco se libra de los aspectos negativos. Y es que esta aplicación

está únicamente disponible en inglés, y también está diseñada para un uso concreto, pues los usuarios que la utilicen deben adquirir el laxante MiraLAX® y realizarse la colonoscopia en el Hospital Universitario Stony Brook (Nueva York). Además, está pensada para ayudar en el proceso de preparación de colonoscopias infantiles, algo que no es compatible con el intervalo de edad de los potenciales usuarios de PrepColon.

Es por estas razones por las que se hace necesaria una propuesta como PrepColon, que responde a las carencias detectadas en otras aplicaciones ya existentes con el objetivo de ayudar a mejorar los resultados de las colonoscopias en el ámbito nacional.

2.5. Conclusiones

En la Tabla 2.2 se puede apreciar una comparativa[2] entre las diferentes aplicaciones de apoyo a la preparación de colonoscopias, cuyas características se han discutido en la sección anterior. Gracias a esta tabla, el lector podrá hacerse una primera idea sobre los requisitos que se pretenden cumplir con la aplicación PrepColon y que se definirán más detalladamente en el próximo capítulo de este TFG.

[2]Para las aplicaciones no disponibles en España, las características indicadas en la tabla se han extraído del artículo sobre el estudio que usó cada una.

	[Wal+21]	[Zan+21]	[Zhu+23]	«My Colonoscopy»	«NurScope»	«SB Colonoscopy Prep»	[Fer23a]
Funcional[3]	✓	✓	✓	✓	✓	✓	✗
Apoyo durante todo el proceso[4]	—	✓	✓	✓	✗	✓	✓
Se adapta al usuario[5]	—	—	—	✗	✗	✓	✓
Disponible en España	✗	✗	✗	✓	✓	✓	✓
Disponible en español	—	—	—	✗	✗	✗	✓
Admite distintos laxantes	✗	✗	✗	✗	✗	✗	✓
Proporciona información sobre colonoscopias	✓	✓	✓	✓	✗	✓	✗

Tabla 2.2: *Comparativa entre diferentes aplicaciones de apoyo a la preparación de colonoscopias.*

[3]El usuario puede acceder a todas sus funcionalidades.
[4]La aplicación ayuda al usuario durante todo el proceso de preparación, incluyendo dietas y tomas de laxantes.
[5]La aplicación tiene en cuenta las necesidades individuales del usuario.

Capítulo 3

Análisis

Antes de comenzar a diseñar una aplicación como PrepColon, es necesario saber qué funcionalidades debe ofrecer para cumplir con su cometido, así como pensar en posibles usos de la misma para poder prever el comportamiento de un posible usuario y anticiparse a los fallos que pudieran surgir durante su ejecución. Para ello, en este capítulo se combinarán historias de usuario con los contenidos del Anexo A para determinar tanto los casos de uso de la aplicación como los requisitos que debe cumplir.

3.1. Introducción

Una vez definidos en el Capítulo 1 tanto la problemática que motiva este TFG, como los objetivos a cumplir; y tras explicar, en el Capítulo 2, qué alternativas existen y qué limitaciones tienen, el siguiente paso en el desarrollo de este proyecto consiste en realizar la fase de análisis. En esta fase se aprovecharon las reuniones celebradas con el cliente para extraer los requisitos solicitados para la aplicación PrepColon. Dado que el cliente es un grupo formado por profesionales con una amplia experiencia en el ámbito de las colonoscopias, dichos requisitos derivan de aspectos en los que, en su día a día, detectan carencias importantes que acaban resultando en preparaciones deficientes por parte de los pacientes. Para ello se sintetizaron los puntos más importantes extraídos de las primeras reuniones con el cliente en un documento que se resume en el Anexo A, que se complementaron con nuevos requisitos que fueron añadiéndose a medida que el proyecto avanzaba y se detectaban aspectos que inicialmente se habían pasado por alto.

Para definir los requisitos que deben cumplirse, en este capítulo se expondrán ejemplos de uso de la aplicación e historias de usuario [Koe16], a través de los cuales se tratará de identificar las características más relevantes que debería tener la aplicación. A estos elementos se sumarán algunos casos de uso [Koe16] que especificarán algunas de las interacciones más representativas del usuario con la aplicación. Una vez definidos todos estos elementos, el último paso en esta fase de análisis es formalizar los requisitos funcionales y no funcionales para ayudar a garantizar que la aplicación final es capaz de satisfacer las necesidades del cliente.

3.2. Ejemplo de uso de la aplicación

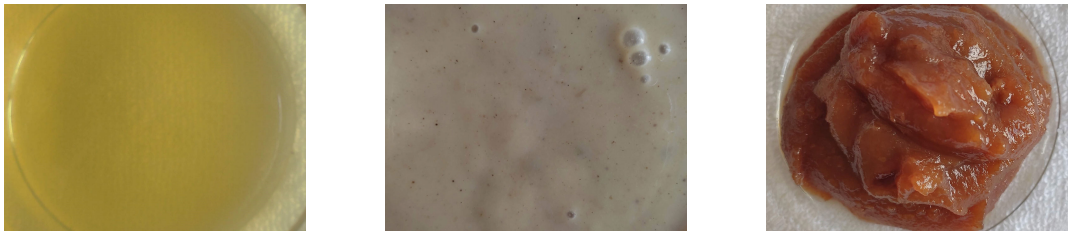
El equipo médico con el que se colaboró para desarrollar este proyecto pretende, a través de la aplicación PrepColon, aumentar el número de pacientes que acuden a las colonoscopias con una preparación adecuada. De esta forma, mejora la tasa de detección precoz de lesiones que puedan estar relacionadas con el CCR. De acuerdo con el propio equipo médico, el intervalo de tiempo más común desde el momento en el que se concerta la cita hasta el día de la misma suele ser de una o dos semanas.

Un paciente que ha participado en el programa de cribado recibe una mala noticia: se ha detectado sangre oculta en la muestra de heces que entregó en su Centro de Salud. Esta persona acude a la consulta y le dan cita para realizarse una colonoscopia en 10 días, por la mañana. El médico le da las indicaciones necesarias de forma oral, como es habitual, pero también le recomienda instalarse PrepColon, una aplicación novedosa que promete ayudar durante el proceso de preparación para colonoscopias, para lo cual necesitará un código único que el propio médico le proporcionará. El paciente decide hacer caso a la recomendación, por lo que instala PrepColon en su dispositivo móvil y, tras permitir a la aplicación enviar notificaciones, introducir el código que le han dado en el Centro de Salud, y leer detenidamente toda la información relevante acerca del proceso de preparación y del uso de la aplicación, el paciente comienza la preparación rellenando el formulario inicial sobre la cita y sus características personales. El protagonista de este ejemplo es una persona sana, que no toma ningún medicamento de forma recurrente, sin sobrepeso, con un flujo intestinal adecuado y que no se ha realizado ninguna intervención quirúrgica en la zona abdominal. En definitiva, es alguien que no necesitará llevar a cabo una preparación intensiva. Tras rellenar el formulario, la aplicación le muestra la primera tarea, que consiste en recoger la preparación laxante en su Centro de Salud.

El paciente recoge la preparación laxante, marca la tarea como completada en la aplicación, y continúa haciendo vida normal. El tiempo pasa, y a 5 días de la colonoscopia llega una nueva notificación de la aplicación indicando al usuario que debe comenzar a hacer dieta. A partir de este momento, el paciente empieza a revisar periódicamente el menú de dieta de la aplicación para saber qué alimentos puede comer y cuáles debe evitar, así como para ver ideas de platos que puede preparar que sean aptos para la dieta. Dado que debe mantener la dieta a lo largo del tiempo—durante los cinco días previos a la cita—, esta tarea no podrá marcarla como completada.

Posteriormente, el día antes de la cita por la tarde-noche—según el laxante que le hayan recetado—, recibe una nueva notificación de la aplicación que le recuerda tomar el laxante. El paciente busca la información sobre cómo tomar el laxante, a la que puede acceder pulsando en el botón «Ayuda» de la tarjeta de tareas correspondiente, o pulsando en «¿Cómo tomo el laxante?» dentro de la pantalla «Ayuda». En cualquier caso, el paciente verá un carrusel en el que podrá deslizar entre las diferentes tarjetas de información para saber cómo preparar la toma del laxante y conocer algunas recomendaciones que puede seguir si el sabor del laxante le resulta demasiado desagradable. Junto con este recordatorio aparece otra tarea pendiente que debe completarse cuando el usuario vaya al baño. El paciente pulsa «Marcar como completado» y se le muestra una pantalla con las tres imágenes de la Figura 3.1 para que éste seleccione el aspecto de su última deposición. En caso de presentar un aspecto inadecuado, que podría significar que el proceso de preparación no está realizándose correctamente o que el laxante no está haciendo efecto debidamente, se mostrará una advertencia indicando al paciente que reprograme la cita para la colonoscopia. Sin embargo, todo está yendo bien y las heces del paciente son líquidas, por lo que no tiene de qué preocuparse. Gracias a las notificaciones enviadas por la aplicación, el

paciente detecta, además, que el menú de dieta ha cambiado, y ahora no puede tomar alimentos sólidos, lácteos de ningún tipo, ni sucedáneos de la leche.



(a) *Líquido* **(b)** *Pastoso* **(c)** *Sólido*

Fig. 3.1: *Imágenes representativas de los posibles aspectos de las heces.*

Cinco horas antes de la cita, el paciente recibe una segunda notificación recordando la toma del laxante, ante lo cual actúa de la misma forma en la que lo hizo para la primera toma. 2 horas antes de la cita, la aplicación le vuelve a pedir indicar el aspecto de sus deposiciones, y vuelve a detectar otro cambio en el menú de dieta. Ahora debe realizar ayuno absoluto hasta que finalice la colonoscopia.

3.3. Requisitos funcionales

3.3.1. Historias de usuario

Una forma de describir las funciones y características que debe cumplir un sistema, o una aplicación en este caso, de manera sencilla para que no sea complicado su comprensión por parte de todos los actores involucrados en su desarrollo es mediante las historias de usuario [Koe16]. Este modo de presentar los requisitos sigue una estructura fija:

Como *[rol]*, quiero *[característica]* para *[beneficio]*

Donde **rol** representa al tipo de usuario, **característica** indica una funcionalidad que dicho usuario necesita utilizar y **beneficio** es el motivo por el que el usuario quiere utilizar esa característica. Para el caso de PrepColon, todas las historias de usuario comenzarían igual, pues el único usuario que utilizaría la aplicación sería un paciente que necesite realizarse una colonoscopia, por lo que podrían empezar de la siguiente manera:

Como *paciente*, quiero...

En la Tabla 3.1 se listan unas posibles historias de usuario para la aplicación PrepColon. En ellas se omite el rol, pues en todas será el mismo, tal y como se ha mencionado anteriormente.

ID	Historia de usuario
US00	Ver información sobre el proceso de preparación de colonoscopias para saber qué voy a tener que hacer.
US01	Registrar la hora y la fecha de la colonoscopia para que la aplicación pueda indicarme qué tengo que hacer en cada momento.
US02	Introducir mis características personales para que las tareas propuestas por la aplicación estén adaptadas a mis necesidades.
US03	Recibir un aviso si estoy realizando la preparación incorrectamente para saber cómo actuar.
US04	Recibir recordatorios en forma de notificaciones para no saltarme ningún paso de la preparación.
US05	Recibir instrucciones sobre cómo tengo que preparar el laxante para tomarlo adecuadamente.
US06	Ver recomendaciones sobre cómo tomar el laxante para evitar mareos, náuseas o vómitos al hacerlo.
US07	Registrar el laxante que me ha sido recetado para recibir indicaciones específicas para ese producto.
US08	Ver qué alimentos puedo o no puedo comer estando en dieta para realizarla correctamente.
US09	Ver ideas de platos que puedo preparar estando en dieta para ayudarme a realizarla mejor.
US10	Recibir instrucciones sobre qué hacer en caso de que algo vaya mal para saber cómo actuar.
US11	Cambiar la fecha y la hora registradas para poder reprogramar la cita si he hecho algo mal durante la preparación.

Tabla 3.1: *Historias de usuario para la aplicación PrepColon.*

3.3.2. Casos de uso

Otra manera de describir las características del sistema, pero con más detalle que las historias de usuario, son los casos de uso (CU) [Koe16]. En ellos no solo se definen las funcionalidades de las que hará uso el usuario final, sino que se detalla todo el proceso que permite acceder a esas características: el actor que inicia el proceso, las interacciones que se dan entre los actores, las condiciones que han tenido que darse para que se pueda iniciar ese proceso, etc. Los casos de uso que se describirán en esta sección se basarán en las historias de usuario expuestas en la sección anterior.

Siguiendo la estructura definida en [Koe16], un CU estará formado por diferentes campos:

- **Título** que identifique el CU.
- Breve **descripción** del CU.
- Los **actores**, cualquier entidad que haga uso del sistema.
- Las **precondiciones** que deberán cumplirse antes de iniciar el CU.
- Las **postcondiciones** que se cumplirán al finalizar el CU.
- El **disparador**, un evento que iniciará el CU.

- El **flujo normal**, la secuencia de eventos e interacciones que formarán el CU si éste transcurre de la manera prevista.
- Un **flujo alternativo** que describe eventos e intereacciones adicionales si no se puede seguir el flujo normal.

A continuación se describen de manera detallada los casos de uso de la aplicación PrepColon.

`CU00` Ver información de colonoscopias

`Descripción`	El usuario visualiza información sobre el proceso de preparación de colonoscopias.
`Actor`	Usuario.
`Precondición`	El sistema no tiene datos guardados.
`Postcondición`	—
`Disparador`	El usuario ha iniciado la aplicación.
`Flujo normal`	1. El sistema solicita permiso para enviar notificaciones. 2. El usuario selecciona «Permitir». 3. El sistema genera un *log*[1] con información sobre el tipo de dispositivo y la versión de la aplicación, y muestra la pantalla inicial, que contiene información sobre el proceso de preparación de colonoscopias.
`Flujo alt.`	2.a. Si el usuario pulsa «No permitir», la aplicación se cierra y el caso de uso finaliza.

`CU01` Inicio de sesión

`Descripción`	El usuario introduce el código único que ha recibido en su Centro de Salud y que confirma que es un participante del estudio clínico.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU00`.
`Postcondición`	El usuario ha sido identificado como un participante válido del estudio clínico.
`Disparador`	El usuario pulsa el botón «¡Comencemos!».
`Flujo normal`	1. El sistema muestra un campo de introducción de texto y solicita el código de paciente. 2. El usuario introduce el código que recibió en su Centro de Salud y pulsa «Aceptar». 3. El sistema genera un *log* con el código de paciente, lo registra y lleva al usuario a la pantalla principal.
`Flujo alt.`	2.a. Si el usuario pulsa «Retroceder», el caso de uso finaliza y se inicia el `CU00`. 3.a. Si el código de paciente no coincide con el formato especificado[2], se mostrará un mensaje indicando que el código introducido no es válido y el CU volverá al paso 1.

[1] Los *logs* tienen el siguiente formato: `[<fecha>]:[<tipo>]:[<identificador>]:<información>`

[2] Dependiendo del hospital en el que el paciente tenga la cita, será `HURH-NNN`, `HMC-NNN` o `HZ-NNN`, siendo `NNN` un número aleatorio.

`CU02` Rellenar formulario

`Descripción`	El usuario completa el formulario de la aplicación con la fecha y hora de la cita y su información médica.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU01`.
`Postcondición`	Los datos de la cita y del usuario se han guardado en el almacenamiento local. El estado interno del sistema ha cambiado. El usuario ha comenzado el proceso de preparación de colonoscopias.
`Disparador`	El usuario ha pulsado el botón «Rellenar formulario» en la pantalla principal[3].
`Flujo normal`	1. El sistema genera un *log* indicando que el usuario va a comenzar el formulario, y muestra las opciones «Rellenar formulario» y «Volver». 2. El usuario selecciona «Rellenar formulario». 3. El sistema muestra una pregunta del cuestionario. 4. El usuario selecciona la respuesta y pulsa el botón «Continuar». 5. El sistema almacena la respuesta del usuario para esa pregunta y genera un *log* con la respuesta introducida. 6. Se repiten los pasos 3 a 5 hasta completar todas las preguntas del formulario. 7. El sistema muestra las respuestas introducidas por el usuario. 8. El usuario comprueba que las respuestas son correctas y selecciona «Terminar». 9. El sistema muestra una advertencia indicando que los datos introducidos no se pueden cambiar. 10. El usuario selecciona «Aceptar» 11. El sistema genera un *log* indicando que el usuario ha terminado el formulario, procesa los datos introducidos, cambia su estado interno, genera los avisos[4] necesarios en función de las respuestas del usuario y lleva al usuario de nuevo a la pantalla principal.
`Flujo alt.`	2.a. Si el usuario pulsa «Volver», éste vuelve a la pantalla principal y el caso de uso finaliza. 3 - 9.a. Si el usuario pulsa «Retroceder» o «Cancelar», éste vuelve a la pantalla previa y el CU seguirá su flujo normal. 11.a. Si, dadas las fechas de la colonoscopia y del momento de la realización del formulario, el usuario debería haber comenzado el proceso de preparación, se iniciará el `CU03`. Si finaliza de manera exitosa, este CU sigue su flujo normal. 11.b. Si, dadas las respuestas, el sistema detecta que el usuario requiere realizar una preparación intensiva, mostrará un mensaje mostrando brevemente las tareas adicionales que deberá realizar.

`CU03` Completar una tarea

[3] «Pantalla principal» hace referencia a la pantalla «Mi cita» en la barra de navegación de la aplicación.

[4] Al generar un nuevo aviso también se programan notificaciones que se lanzarán de manera periódica hasta que el usuario complete la tarea correspondiente.

`Descripción`	El usuario marca una tarea propuesta como completada.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU01`.
`Postcondición`	La tarea cambia su estado de *Pendiente* a *Completada*.
`Disparador`	El usuario ha pulsado el botón «Mis tareas» en la pantalla principal.
`Flujo normal`	1. El sistema genera un *log* indicando que se están consultando las tareas pendientes y muestra tarjetas de aviso, que contienen una descripción de la tarea propuesta, asociadas a tareas pendientes de completarse. 2. El usuario selecciona «Marcar como completado» en una tarea pendiente. 3. El sistema pregunta al usuario si desea continuar. 4. El usuario selecciona «Aceptar». 5. El sistema genera un *log* indicando que se ha completado una tarea y registra la fecha y la hora actual. Si fuese necesario, el sistema también genera nuevos avisos.
`Flujo alt.`	1.a. Si no hay tareas pendientes, el sistema lo indica en un mensaje que se muestra por pantalla. El CU finaliza. 2.a. Si el usuario pulsa «Volver», éste vuelve a la pantalla principal y el caso de uso finaliza. 4.a. Si el usuario pulsa «Cancelar», el caso de uso finaliza.

`CU04` Completar una tarea fuera de tiempo

`Descripción`	El usuario marca una tarea propuesta como completada en un momento temporal posterior a la fecha límite estipulada para dicha tarea.
`Actor`	Usuario, tiempo.
`Precondición`	Se ha completado el `CU01`. No se ha completado el `CU02` para esa tarea.
`Postcondición`	La tarea cambia su estado de *Pendiente* a *Completada*.
`Disparador`	El sistema ha detectado que la fecha actual es posterior a la fecha límite de esa tarea.
`Flujo normal`	1. El sistema muestra una advertencia preguntando si la tarea ha sido completada. 2. El usuario pulsa «Sí». 3. El sistema genera un *log* indicando que se ha completado una tarea registra la fecha y la hora actual. Si fuese necesario, el sistema también genera nuevos avisos.
`Flujo alt.`	2.a. Si el usuario pulsa «No», y la tarea es imprescindible para lograr una preparación adecuada, éste vuelve a la pantalla principal y el caso de uso finaliza. Se indica al usuario que deberá contactar con su Centro de Salud para reprogramar la cita. 2.b. Si el usuario pulsa «No», y la tarea es una recomendación para ayudar a mejorar la preparación, el sistema muestra una advertencia insistiendo en la importancia de realizar todas las tareas propuestas.

`CU05` Ver alimentos permitidos

`Descripción`	El usuario visualiza información sobre los alimentos que puede o no puede tomar.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU01`.
`Postcondición`	—
`Disparador`	El usuario ha pulsado el botón «Mi dieta» en la pantalla principal o en la pantalla «Ayuda».
`Flujo normal`	1. El sistema genera un *log* indicando que se está consultando la información de la dieta y muestra un listado de los alimentos permitidos para el usuario y los que tiene prohibido comer.
`Flujo alt.`	1.a. Si el usuario pulsa «Volver», éste vuelve a la pantalla principal y el caso de uso finaliza. 1.b. Si quedan más de 5 días hasta el momento de la cita, el sistema indica que no existen restricciones en la dieta del usuario. 1.c. Si quedan menos de 2 horas hasta el momento de la cita, el sistema indica que el usuario no puede comer ni beber ningún tipo de alimento.

`CU06` Ver información del laxante

`Descripción`	El usuario visualiza información sobre cómo preparar y tomar el laxante.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU01`.
`Postcondición`	—
`Disparador`	El usuario ha pulsado el botón «¿Cómo tomo el laxante?» en la pantalla «Ayuda».
`Flujo normal`	1. El sistema genera un *log* indicando que se está consultando la información del laxante y muestra información sobre cómo preparar el laxante principal o adicional, junto con recomendaciones sobre cómo tomarlo.
`Flujo alt.`	1.a. Si el usuario pulsa «Volver», éste vuelve a la pantalla principal y el caso de uso finaliza. 1.b. Si el usuario aún no debe tomar el laxante, el sistema indicará la fecha y la hora a la que tiene que hacerlo. 1.c. Si el usuario ha completado las tomas del laxante, el sistema indicará que no necesita tomar más laxantes.

`CU07` Modificar fecha y hora de la cita

`Descripción`	El usuario cambia la fecha y la hora de la colonoscopia que están guardados en el sistema.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU01`.

`Postcondición`	Los datos de la cita se han modificado.
`Disparador`	El usuario ha pulsado el botón «Cambiar fecha» de la pantalla principal.
`Flujo normal`	1. El sistema muestra una advertencia indicando que cambiar la fecha en la aplicación no equivale a cambiar la fecha real de la cita, y que debe contactar con su Centro de Salud para hacerlo. 2. El usuario seleccciona «Cambiar fecha». 3. El sistema pide una confirmación de que el usuario ha contactado previamente con su Centro de Salud. 4. El usuario selecciona «Sí». 5. El sistema genera un *log* indicando que se está modificando la fecha de la cita y muestra la pregunta del formulario relativa a la fecha. 6. El usuario selecciona la nueva fecha y pulsa «Continuar». 7. El sistema muestra la pregunta del formulario relativa a la hora. 8. El usuario selecciona la nueva hora y pulsa «Guardar». 9. El sistema genera un *log* con las nuevas fecha y hora de la cita, las registra y vuelve a generar los avisos.
`Flujo alt.`	2,4,6.a. Si el usuario pulsa «Retroceder» o «No», éste vuelve a la pantalla principal y el CU finaliza. 8.a. Si el usuario pulsa «Retroceder», el CU vuelve al paso 5. 9.a. Si, dadas la nueva fecha de la colonoscopia y del momento actual, el usuario debería haber comenzado el proceso de preparación, se iniciará el `CU03`.

`CU08` Cancelar cita

`Descripción`	El usuario cancela el proceso de preparación. Se borran todos los datos del sistema.
`Actor`	Usuario.
`Precondición`	Se ha completado el `CU01`.
`Postcondición`	El sistema ha vuelto a su estado inicial.
`Disparador`	El usuario ha pulsado el botón «Borrar datos» de la pantalla «Ajustes».
`Flujo normal`	1. El sistema muestra una advertencia indicando que borrar los datos de la aplicación no equivale a cancelar la cita, y que para ello debe contactar con su Centro de Salud. También ofrece la posibilidad de desactivar las notificaciones. 2. El usuario seleccciona «Borrar datos». 3. El sistema muestra una advertencia indicando que los datos no se podrán recuperar una vez se borren. 4. El usuario selecciona «Aceptar». 5. El sistema inicia el `CU08`.
`Flujo alt.`	2,4.a. Si el usuario pulsa «Retroceder» o «Cancelar», éste vuelve a la pantalla «Ajustes» y el CU finaliza.

`CU09` Enviar datos

`Descripción`	Los datos recopilados por la aplicación son enviados al equipo médico para su análisis.
`Actor`	Tiempo.
`Precondición`	Se ha completado el `CU01`.
`Postcondición`	El sistema ha vuelto a su estado inicial.
`Disparador`	La fecha actual es posterior a la fecha registrada por el usuario.
`Flujo normal`	1. El sistema envía los datos recogidos al equipo médico. 2. El sistema borra todos los datos almacenados e inicia el `CU00`.
`Flujo alt.`	1 - 2.a. Si no es posible completar la acción, el caso de uso finalizará. Una vez transcurrido un tiempo determinado, el caso de uso volverá a iniciarse. Si la aplicación está funcionando en primer plano, se mostrará una ventana emergente que indique la causa del problema.

3.4. Requisitos no funcionales

En las secciones anteriores se han presentado las funcionalidades a las que el usuario final debería poder acceder mientras utiliza la aplicación PrepColon. Pero antes de terminar este capítulo es importante definir también otras características del sistema que, aunque pasen más desapercibidas para el usuario final, también resultan clave para el correcto funcionamiento de la aplicación. Estas propiedades, relacionadas con aspectos como la eficiencia, la arquitectura, la tolerancia a fallos, la escalabilidad, la portabilidad, etc.; son lo que se conocen como requisitos no funcionales [Koe16].

En la Tabla 3.2 se pueden observar algunos de los requisitos no funcionales más relevantes, en su mayoría sintetizados a partir de los requerimientos solicitados por los clientes.

ID	Requisito no funcional
`NFR00`	Los usuarios deben poder utilizar la aplicación desde un dispositivo móvil.
`NFR01`	La aplicación debe estar disponible en las principales tiendas de aplicaciones—Play Store y App Store.
`NFR02`	La aplicación debe presentar compatibilidad con el mayor número de dispositivos móviles posible.
`NFR03`	La aplicación debe estar preparada para utilizarse en diferentes idiomas.
`NFR04`	El sistema debe permitir la incorporación de nuevas características sin necesidad de realizar grandes cambios en el código.
`NFR05`	El sistema debe garantizar la privacidad, seguridad e integridad de los datos del usuario.
`NFR06`	La aplicación debe estar disponible para su uso a cualquier hora del día, todos los días del año.
`NFR07`	El sistema debe presentar una interfaz gráfica capaz de adaptarse a diferentes tamaños y resoluciones de pantalla.
`NFR08`	La aplicación debe presentar una interfaz gráfica intuitiva, amigable y sencilla de utilizar.
`NFR09`	El sistema debe enviar los datos recopilados al equipo médico cuando se produzca un evento que interrumpa o finalice el proceso de preparación de colonoscopias.

Tabla 3.2: *Requisitos no funcionales de la aplicación PrepColon.*

3.5. Conclusiones

En este capítulo se han podido identificar las diferentes necesidades que la aplicación PrepColon debe poder cubrir para cumplir con las expectativas de los clientes del proyecto. A modo de resumen, se ha visto que PrepColon debe ser una aplicación multilingüe que funcione en iOS y en Android, que recoja datos de los usuarios para su uso en un estudio clínico, que proporcione al usuario información relevante relativa a la preparación de colonoscopias, y que realice un seguimiento del proceso para poder dar indicaciones personalizadas al usuario. En el siguiente capítulo, se aprovecharán los requisitos funcionales y no funcionales ya descritos para definir el diseño de las partes cliente y servidora que compondrán el sistema.

Capítulo 4

Diseño

En el Capítulo 2 se han presentado las problemáticas de otras aplicaciones de apoyo a la preparación de colonoscopias y en el Capítulo 3 se han descrito las características que debería tener la aplicación para poder cumplir correctamente con su cometido. El siguiente paso en el proceso de desarrollo es diseñar el sistema. Por ello, en este capítulo, se definirá tanto la arquitectura, que caracterizará al sistema en conjunto, como los distintos componentes que la formarán: aplicación móvil—cliente e interfaz de usuario—y servidor de recogida de datos.

4.1. Introducción

Llegados a este punto del desarrollo del proyecto, y gracias a lo expuesto en el capítulo anterior, ya se dispone de una idea bastante clara de qué requisitos debe cumplir la aplicación PrepColon para poder satisfacer las necesidades del cliente y permitir a sus usuarios realizar una correcta preparación para sus colonoscopias. Pero antes de comenzar a programar conviene reflexionar previamente sobre cómo se va a enfocar el diseño para cumplir dichos requisitos.

En el proceso de diseño de este proyecto ha resultado determinante el uso de una metodología ágil en el desarrollo de la aplicación. Gracias a esto, durante los diferentes ciclos del desarrollo, se han ido puliendo los detalles que han permitido conseguir un producto que cumpliese con las expectativas del cliente de la mejor manera posible, y que se descibirán en este capítulo. Primeramente, será necesario discutir qué arquitectura va a tener el sistema al que pertenecerá la aplicación PrepColon. Tras ello, se definirán los elementos de la aplicación que buscarán satisfacer los requisitos propuestos en el capítulo anterior. Estos elementos son la lógica del cliente y la interfaz de usuario, dos aspectos sumamente importantes que determinarán cómo interactuará el usuario con el sistema. Por último, pero no por ello menos importante, se presentará la parte servidora que complementará a la aplicación, y cuya necesidad se discutirá en la Sección 4.2

4.2. Arquitectura del sistema

El equipo médico que toma el rol de cliente en este proyecto busca poner a prueba la efectividad de PrepColon en un estudio clínico, y para ello necesita poder almacenar los datos de los diferentes pacientes en una base de datos alojada en un servidor propio.

A priori la propia aplicación podría ser capaz de enviar directamente los datos a la base de datos. No obstante, y de acuerdo a la información proporcionada por el propio equipo médico, el uso de dicho servidor no está limitado únicamente a este proyecto, por lo que su acceso está restringido a personal sanitario autorizado. Además, ese servidor genera códigos aleatorios y únicos que se asocian a cada paciente que decide participar de manera voluntaria en el estudio, y esa relación entre el código y el paciente solo deberían conocerla los médicos encargados del estudio, de forma que los datos queden totalmente anonimizados. En definitiva, acceder directamente desde la aplicación a la base de datos del estudio podría generar más problemas de los que soluciona, por lo que una mejor solución es que sea el equipo médico quien introduzca los datos manualmente en la base de datos.

Ante esta problemática, el equipo médico solicitó recibir los datos por correo electrónico. Esto puede hacerse accediendo desde la propia aplicación al cliente de correo electrónico que tenga el usuario instalado en su dispositivo móvil, lo cual, con los permisos necesarios, no debería suponer un problema demasiado grande. Pero si se revisan los requisitos no funcionales descritos en la sección 3.4, y en el requisito `NFR05` más concretamente, utilizar el cliente de correo electrónico del usuario no es tan buena idea. Y es que, de hacerse así, los datos se enviarían a través la cuenta de correo electrónico del propio paciente, con lo que se perdería su anonimato.

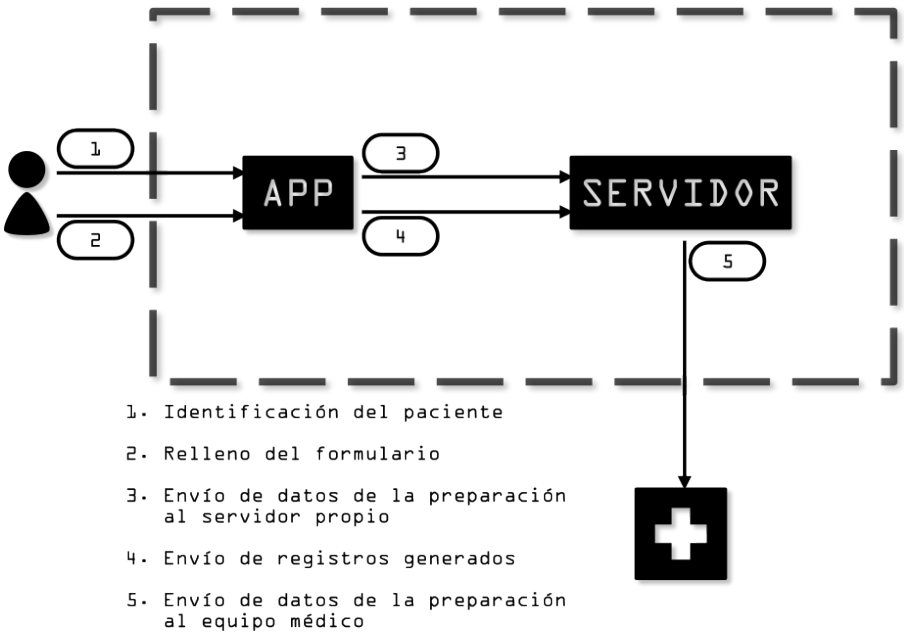


Fig. 4.1: *Esquema de la arquitectura propuesta para la aplicación PrepColon.*

Teniendo en cuenta todo esto, finalmente se decidió que, al contrario de lo que se pensaba en un principio, sí era necesario que la aplicación formase parte de un sistema distribuido, como se muestra en la Figura 4.1. De esta manera, el sistema constará de dos componentes fundamentales, tal y como puede verse en la Figura 4.1. Por un lado, la aplicación móvil que, a través de la aplicación PrepColon, recogerá la información necesaria de los pacientes que la

utilicen. Por otro lado, habrá un componente servidor que se encargue de recibir los datos de la aplicación y, sin almacenarlos, los reenvíe al equipo médico. De esta forma, este componente puede tener un cliente de correo electrónico que envíe todos los datos desde una misma cuenta, independientemente de a qué paciente correspondan dichos datos, para mantenerlos anonimizados. Asimismo, teniendo en mente el `CU01` , se puede delegar la verificación del código del paciente al servidor. Esto permitiría no sólo comprobar que los códigos de paciente son válidos, sino también registrar aquellos que ya se han utilizado para evitar situaciones en las que varios usuarios introduzcan el mismo código de paciente.

4.3. Aplicación móvil

Una vez definida la arquitectura del sistema, el siguiente paso es diseñar la propia aplicación móvil, ya que será la base principal del sistema. Para ello, dicho diseño se dividirá en dos partes. Primeramente se hablará de la lógica del cliente en sí, cómo interactuará con el usuario de la aplicación y qué datos utilizará. Tras ello, se explicará el proceso de rediseño de la interfaz gráfica, desde la versión inicial inspirada en el prototipo—descrito en el Apéndice B—hasta la versión final disponible en las tiendas de aplicaciones.

4.3.1. Cliente

En este documento se define «cliente» como el conjunto de aquellos componentes cuyo cometido se centra en lograr cohesión dentro de la aplicación y permitir un correcto funcionamiento tanto de manera interna como en su interacción con servicios externos—en este caso, el servidor. En esta sección se describirá el diseño propuesto para el cliente PrepColon, buscando satisfacer las necesidades identificadas en el capítulo anterior.

4.3.1.a. Patrón *State*

Para facilitar y agilizar la fase de diseño, es común utilizar patrones de diseño [Gam+95] y, para este proyecto, se hará lo propio. Los patrones de diseño son guías que describen formas de solucionar problemas frecuentes relativos, principalmente, a la estructuración del software de un programa. Estos patrones se clasifican en tres categorías—creacionales, estructurales y de comportamiento—según su objetivo principal, tal y como se puede ver en la Tabla 4.1.

Categoría	Objetivo	Patrones
Creacionales	Conseguir independencias en la creación de elementos para aumentar la reutilización del código.	*Abstract Factory* *Builder* *Factory Method* *Prototype* *Singleton*
Estructurales	Definir cómo formar estructuras más complejas a partir de elementos más sencillos.	*Adapter* *Bridge* *Composite* *Decorator* *Facade* *Flyweight* *Proxy*
De comportamiento	Gestionar la interacción entre diferentes elementos y las responsabilidades de cada uno.	*Chain of Responsibility* *Command* *Interpreter* *Iterator* *Mediator* *Memento* *Observer* *State* *Strategy* *Template Method* *Visitor*

Tabla 4.1: *Catálogo de patrones de diseño. [Gam+95]*

Volviendo a los requisitos definidos en el capítulo anterior, se puede ver que la clave de la aplicación es la interacción entre el usuario y el sistema. Es por esto que los patrones de comportamiento podrían ser de gran ayuda en el desarrollo de PrepColon.

De acuerdo con el análisis realizado en el Capítulo 3, y, junto con la información adicional proporcionada en el Anexo A, el proceso de preparación de colonoscopias puede dividirse en varias fases. Habrá inicialmente una fase previa en la que el paciente tenga la cita ya programada pero aún no haya comenzado la preparación. Después, una segunda etapa en la que deba comenzar la dieta sólida y, si fuese necesario, la toma de un laxante adicional. Tras esto, comenzará la tercera fase, donde el paciente deba iniciar la dieta líquida. De aquí, se pasaría a una cuarta fase en la que el paciente comienza la toma del laxante principal. Finalmente, el proceso termina en una última fase en la que el paciente debe guardar ayuno absoluto.

Como aplicación de apoyo a la preparación de colonoscopias, PrepColon debería reflejar todas estas fases. Esto puede lograrse cambiando la información mostrada y las interacciones que el usuario puede tener con la aplicación en función de la etapa actual de la preparación. De esta manera, en cada fase el usuario encontrará contenidos específicos que le ayuden durante esa fase concreta. Es decir, el comportamiento de la aplicación debe cambiar en cada etapa.

Es por esto que el patrón estado [Gam+95] parece el más adecuado para aplicar en el desarrollo de la aplicación. Cada una de las fases anteriores se corresponderán con un estado

diferente, y la información que reciba el usuario cambiará en función del estado interno de la aplicación, que a su vez dependerá de las interacciones que se hayan producido entre el usuario y la aplicación o de la fecha actual.

La aplicación PrepColon necesitaría, como mínimo, cinco estados—uno por cada fase descrita anteriormente—, pero eso no es suficiente para abarcar todos los requisitos que debe cumplir. Además, hay situaciones en los que dos estados se pueden solapar ya que, por ejemplo, la toma del laxante principal debe realizarse mientras el paciente se encuentra en dieta líquida. Por esta razón se tomó la decisión de separar los estados en dos flujos diferentes e independientes. Por un lado, se encuentran los estados relacionados con la preparación en sí, mostrados en la Tabla 4.2, y, por otro lado, los estados relacionados con la dieta, que se pueden ver en la Tabla 4.3. El flujo de estados en ambos casos se puede observar en las Figuras 4.2 y 4.3.

Estado	Descripción	Requisitos relacionados
A.I	Estado inicial previo, la aplicación no tiene datos guardados.	CU00 US00
A.II	Estado previo al inicio de la preparación, el usuario ha iniciado sesión.	CU01
A.III	Estado inicial de la preparación, el usuario ha rellenado el formulario.	CU02 US01 US02 US04 US07
A.IV	El usuario ha comenzado la toma del laxante adicional.	CU06 CU07 CU08 US04 US05 US06
A.V	El usuario ha comenzado la toma del laxante principal.	CU06 CU07 CU08 US04 US05 US06
A.VI	El usuario ha finalizado la toma del laxante, espera a la colonoscopia.	US04
A.VII	Se ha detectado una mala preparación por parte del usuario.	NFR09 US03 US10 US11
A.VIII	Está pendiente el envío de datos.	CU09 NFR09

Tabla 4.2: *Estados de la aplicación PrepColon relacionados con el proceso de preparación de colonoscopias.*

Estado	Descripción	Requisitos relacionados
B.I	El usuario no tiene restricciones en su dieta.	CU05 US08
B.II	El usuario tiene restricciones en su dieta.	CU05 US08 US09
B.III	El usuario únicamente puede tomar líquidos claros.	CU05 US08
B.IV	El usuario debe hacer ayuno absoluto.	CU05 US08

Tabla 4.3: *Estados de la aplicación PrepColon relacionados con las dietas del usuario.*

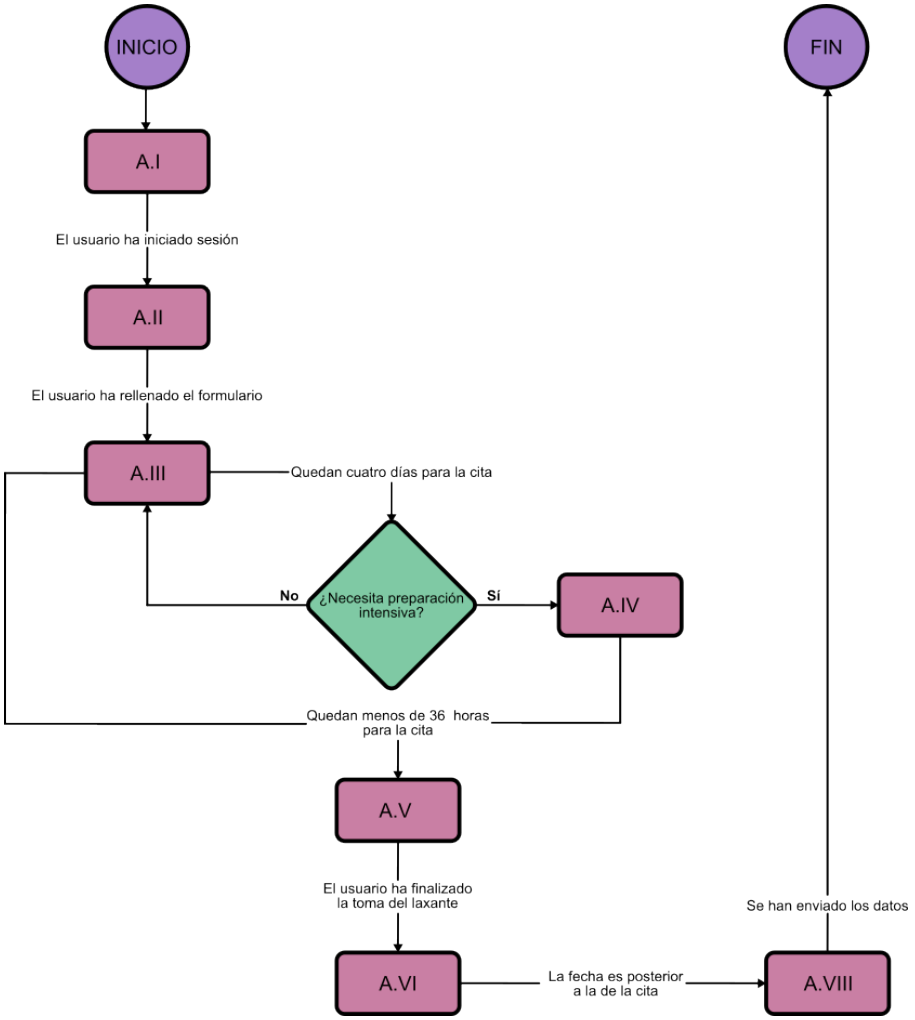


Fig. 4.2: *Diagrama de estados relacionados con el proceso de preparación. Para cualquiera de los estados desde* `A.III` *hasta* `A.VI`*, si la aplicación detecta que la preparación no es adecuada, se pasará al estado* `A.VII` *y luego al estado* `VIII`*.*

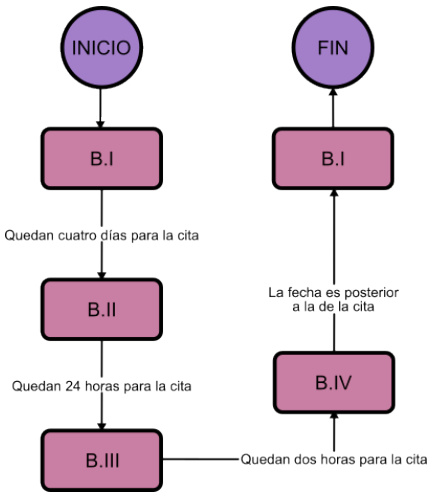


Fig. 4.3: *Diagrama de estados relacionados con la dieta del usuario.*

4.3.1.b. Notificaciones de la aplicación

La clave de la eficacia de la aplicación PrepColon reside en las notificaciones. Gracias a estas alarmas, los usuarios no necesitarán estar todo el tiempo pendientes de la aplicación, sino que podrán hacer vida normal hasta que reciban una alerta indicándoles que deben realizar una tarea relacionada con la preparación para su colonoscopia.

Las notificaciones van asociadas a las tareas que se muestran en la aplicación y, dado que estas tareas deben completarse en un intervalo de tiempo determinado, la frecuencia de las notificaciones aumentará a medida que se aproxime la fecha límite. Así, la aplicación tendrá un comportamiento más «insistente» con aquellos usuarios que ignoren las alertas, para evitar que se salten pasos de la preparación. En la Tabla 4.4 puede verse la frecuencia de las notificaciones en función del número de horas restantes hasta la fecha límite de la tarea.

Diferencia (horas)	**Frecuencia**	**Frecuencia (horas)**
Más de 128	Muy baja	96
Entre 128 y 64	Baja	48
Entre 64 y 32	Media	24
Entre 32 y 16	Alta	12
Entre 16 y 8	Muy alta	6
Menos de 8	Extremadamente alta	2

Tabla 4.4: *Frecuencia de las notificaciones en función de las horas restantes para completar la tarea asociada.*

Al estar asociadas a tareas que debe completar el usuario, las notificaciones deberían mostrar un texto que permita identificar fácilmente la tarea a la que pertenece el recordatorio. En la Tabla 4.5 pueden verse los mensajes que aparecen en las notificaciones enviadas por la aplicación PrepColon—consensuados con el equipo médico—y, en la Tabla 4.6, el intervalo de tiempo que determinará la frecuencia de las mismas.

Tarea	ID	Mensaje
Tarea genérica, por defecto	`NTF00`	Tienes tareas pendientes.
Adquisición de laxante suplementario	`NTF01`	¿Has comprado Evacuol®?
Tomas de laxante suplementario	`NTF02`	¿Has tomado la primera dosis de Evacuol®?
	`NTF03`	¿Has tomado la segunda dosis de Evacuol®?
	`NTF04`	¿Has tomado la tercera dosis de Evacuol®?
	`NTF05`	¿Has tomado la cuarta dosis de Evacuol®?
Suspensión de medicamentos	`NTF06`	¿Has contactado con tu Médico de Atención Primaria para valorar la Suspensión de Antiagregantes?
	`NTF07`	¿Has contactado con tu Médico de Atención Primaria para valorar la Suspensión de Anticoagulantes?
	`NTF08`	¿Has suspendido el tratamiento de hierro?
Adquisición del laxante principal	`NTF09`	¿Has recogido la preparación en tu Centro de Salud?
Tomas del laxante principal	`NTF10`	¿Has tomado la dosis de Bohn®?
	`NTF11`	¿Has tomado la dosis de Pleinvue®?
	`NTF12`	¿Has tomado la dosis de Citrafleet®?
	`NTF13`	¿Has tomado la dosis de Moviprep®?
	`NTF14`	¿Has tomado la dosis de Clensia®?
Recordatorio de las dietas	`NTF15`	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.
	`NTF16`	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.
	`NTF17`	¿Estás siguiendo la dieta? Recuerda que no puedes comer ni beber nada.

Tabla 4.5: *Listado de notificaciones enviadas por la aplicación PrepColon. Las notificaciones `NTF01` a `NTF05` sólo se enviarán si, según la información del formulario, el paciente requiere de una preparación intensiva. Las notificaciones `NTF06` a `NTF08` sólo se enviarán si, según la información del formulario, el paciente está tomando los medicamentos indicados.*

ID	Inicio	Fin
NTF00	—	—
NTF01	Fecha en la que se completó el formulario	4/5 días antes de la cita
NTF02	4/5 días antes de la cita; 19:00h	4/5 días antes de la cita; 23:00h
NTF03	3/4 días antes de la cita; 19:00h	3/4 días antes de la cita; 23:00h
NTF04	2/3 días antes de la cita; 19:00h	2/3 días antes de la cita; 23:00h
NTF05	1/2 días antes de la cita; 19:00h	1/2 días antes de la cita; 23:00h
NTF06	Fecha en la que se completó el formulario	Fecha de la cita
NTF07	Fecha en la que se completó el formulario	Fecha de la cita
NTF08	Fecha en la que se completó el formulario	Fecha de la cita
NTF09	Fecha en la que se completó el formulario	3 días antes de la cita
NTF10	Día antes de la cita; 19:00h/Mismo día de la cita; 07:00h	Día antes de la cita; 21:30h/Mismo día de la cita; 09:30h
NTF11	Día antes de la cita; 21:00h/Mismo día de la cita; 09:00h	Día antes de la cita; 23:30h/Mismo día de la cita; 11:30h
NTF12	Día antes de la cita; 19:00h/Mismo día de la cita; 08:00h	Día antes de la cita; 21:30h/Mismo día de la cita; 10:30h
NTF13	Día antes de la cita; 20:00h/Mismo día de la cita; 08:00h	Día antes de la cita; 22:30h/Mismo día de la cita; 10:30h
NTF14	Día antes de la cita; 20:00h/Mismo día de la cita; 08:h00	Día antes de la cita; 22:30h/Mismo día de la cita; 10:30h
NTF15	4 días antes de la cita	Día antes de la cita
NTF16	Día antes de la cita	Dos horas antes de la cita
NTF17	Dos horas antes de la cita	Fecha de la cita

Tabla 4.6: *Fechas de inicio y fin en las que se enviarán las notificaciones de la aplicación PrepColon. Las fechas de las tomas de los laxantes—tanto el principal como el adicional—dependerán de si la cita tiene lugar por la mañana o por la tarde. En caso de ser por la mañana, las tomas deberán realizarse antes.*

4.3.1.c. Estructura de los datos de configuración

El requisito `NFR04` se propuso con el objetivo de facilitar el mantenimiento de la aplicación y agilizar el desarrollo de la misma. La forma de lograr esto es definir en algún lugar una serie de datos que, al arrancar la aplicación, puedan ser accesibles dinámicamente. De esta manera, si estos valores se utilizan en diferentes sitios dentro del programa, sólo es necesario modificarlos, eliminarlos, o añadirlos en un único sitio y no en todas las partes en las que se utilicen. Existen diferentes maneras de conseguir este propósito. Una primera forma de hacerlo sería mediante un servidor externo. Al arrancar la aplicación, ésta debería conectarse al servidor para solicitar los datos necesarios. Sin embargo, esta solución requiere que la aplicación siempre tenga conexión a Internet para poder comunicarse con el servidor, lo que podría interferir con el requisito `NFR06` .

Otra forma de hacerlo es incluyendo estos datos en algún fichero dentro del código fuente. Esta solución no pone en riesgo el requisito `NFR06` , dado que esta información estaría almacenada localmente, aunque puede dificultar el cumplimiento del requisito `NFR04` al ser necesario modificar el código fuente.

Finalmente, otra posible solución es almacenar los datos en un fichero independiente del código localizado en la memoria interna del dispositivo y que, al arrancar la aplicación, se lea el contenido de este fichero. Esta solución es similar a la anterior, pero además permite mantener un código más limpio al separar claramente la lógica del programa de la configuración, logrando así también el requisito `NFR04` . Es por esto que este método fue el que se eligió finalmente para PrepColon. Y más concretamente se utilizó un formato JSON [JSON], más sencillo y fácil de comprender que otras alternativas como XML. En este fichero se definen principalmente los siguientes parámetros:

- La hora que determina si la cita es por la mañana o por la tarde, establecida en las 15:00 horas.
- Las frecuencias con las que se lanzarán las notificaciones, que dependen del tiempo restante hasta la fecha límite de la tarea asociada—cuanto menos tiempo quede, más frecuentes serán las notificaciones.
- La lista de medicamentos que puede seleccionar el usuario en el formulario.
- La lista de laxantes principales que puede seleccionar el usuario en el formulario, junto con la hora a la que debe comenzar a tomarlos en función de si la cita es por la mañana o por la tarde, y si la primera toma debe ser el día anterior o no.
- La ruta de las imágenes que se muestran en la aplicación.
- URIs de acceso al servidor.

4.3.2. Interfaz de Usuario

Como se mencionó en la Sección 2.3, PrepColon tiene su origen en un prototipo no funcional de una aplicación de apoyo a la preparación de colonoscopias, del cual se hace un análisis en el Apéndice B. Es por eso que, en esta sección, el lector podrá apreciar la evolución que sufrió la interfaz de usuario desde la primera versión, descrita con mayor detalle en el Anexo B, hasta la

versión definitiva. De esta forma, a medida que se generaban versiones nuevas de PrepColon, se buscaba adaptar la aplicación mejor al requisito `NFR07` de la Tabla 3.2. Además, los comentarios proporcionados por el equipo médico durante las reuniones periódicas—fruto de la metodología ágil descrita en la Sección 1.2—ayudaron también a conseguir una interfaz de usuario que cumpliese con sus expectativas.

4.3.2.a. Versiones iniciales

En un principio se optó por mantener una interfaz similar a la de [Fer23a], conservando la paleta de colores. En esta primera versión predomina el azul—color `#29B6F6` —, un color asociado a conceptos como la calma, la sabiduría o la confianza [Tav16]. Y para lograr un contraste adecuado, se ha continuado utilizando un color blanco como color secundario. También se mantuvieron los colores rojo— `#F44336` —y verde— `#4CAF50` —para los botones del formulario, de forma similar a como se puede observar en B.2.

Sin embargo, se cambió el diseño de las tarjetas de avisos que se mostraban al usuario para que fuesen más llamativas y consiguiesen captar mejor la atención del paciente. Para ello, se distinguieron los cuatro tipos de tarjetas con distinto nivel de urgencia que se describen en la Tabla 4.7 y que se muestran en la Figura 4.4.

Tipo	**Urgencia**	**Descripción**	**Color**
Información	—	Esta tarjeta muestra información adicional, como la dieta u otras indicaciones.	`#283593`
Avisos	—	Estas tarjetas muestran tareas que debe realizar el usuario	`#388E3C`
	Media		`#FF6F00`
	Alta		`#AD1457`

Tabla 4.7: *Descripción de las diferentes tarjetas generadas por la aplicación.*

Sin embargo, se comprobó que algunas personas que no estuvieran familiarizadas con el uso de dispositivos móviles podrían tener dificultades para utilizar la aplicación debido, principalmente, al desconocimiento sobre cómo utilizar el menú desplegable lateral. Y teniendo en cuenta que los potenciales usuarios de esta aplicación son personas de 50 años o más, la probabilidad de que un usuario de la aplicación pudiese tener problemas a la hora de usarla era considerable. Es por esto que se optó por un rediseño radical de la interfaz, con el objetivo de mostrar en pantalla la mayor cantidad de información posible—toda la que la pequeña pantalla de un dispositivo móvil permite. También se trató de utilizar botones con etiquetas autoexplicativas para facilitar el entendimiento de su propósito.

En esta nueva fase se sustituyó el azul como color principal por el verde, concretamente las tonalidades `#004D40` , `#009688` y `#E0F2F1` . La motivación para este cambio es la relación del color verde con la salud, ya que al asociarse con ideas como la paz, la naturaleza o la armonía no es raro encontrarse este color en hospitales o centros de salud [Tav16]. Ahora, desde la pantalla principal, se podía acceder a toda la información relevante según el estado de la preparación en la que se encontrase, tal y como puede comprobarse en la Figura 4.5—y en la Figura 4.6 para la versión final. A continuación se listan los botones y mensajes que se muestran al usuario según el estado de la preparación.

- Sin datos guardados (ver Figuras 4.5a y 4.6a).

Fig. 4.4: *Diseño de los cuatro tipos de tarjeta.*

- Mensaje que indica que se debe rellenar el formulario, o terminarlo si no se completó.
- Botón que lleva al formulario.

➤ Con datos guardados, antes de empezar a tomar el laxante (ver Figuras 4.5b y 4.6b).

- Botón que lleva a la pantalla de avisos. Este botón parpadea en diferentes colores si el usuario tiene tareas pendientes.
- Botón a la pantalla de dieta.
- Botón al resumen del formulario.
- Botón que lleva al formulario, para modificarlo.
- Botón para borrar los datos.
- Fecha y hora de la cita.
- Mensaje con un recordatorio.

➤ Durante la toma del laxante adicional.

- Se añade a lo anterior un botón que muestra información sobre qué hacer en caso de vómitos.

➤ Durante la toma del laxante principal.

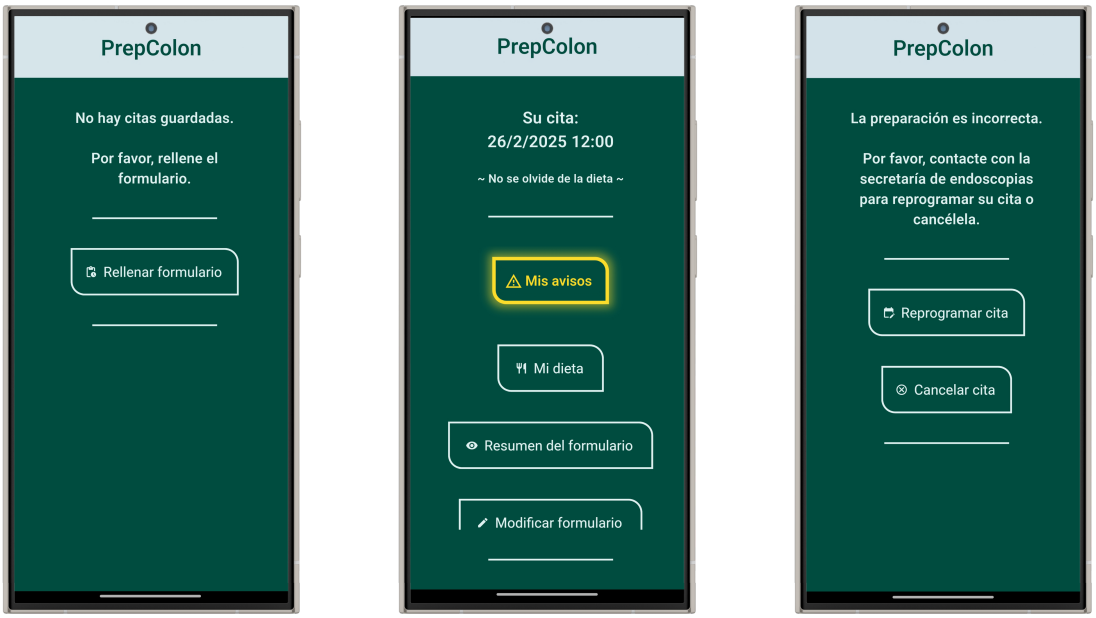
- Se añade a lo anterior un botón que lleva a la pantalla de selección del aspecto de las deposiciones.

➤ Después de la cita.

- Se añade a lo anterior un botón para enviar los datos recopilados por la aplicación, si no se han podido enviar automáticamente.

➤ Si se ha detectado una mala preparación por parte del usuario (ver Figuras 4.5c y 4.6c).

- Mensaje que advierte de la mala preparación y sobre las posibles acciones que puede realizar el usuario.
- Botón para modificar la fecha de la cita.
- Botón para borrar los datos.



(a) **(b)** **(c)**

Fig. 4.5: *Pantalla principal de PrepColon para los casos: (a) sin datos guardados, (b) antes de empezar a tomar el laxante y (c) tras detectar una mala preparación.*

Este prototipo fue probado principalmente por los clientes del proyecto—es decir, el equipo médico que propuso la idea para PrepColon y con el que se colaboró para su desarrollo—y también alguna persona cercana al proyecto. Aunque este nuevo diseño cumplía con su cometido, resultaba poco atractivo para el usuario final, y algunas de las personas que utilizaron esta versión tuvieron pequeñas dificultades a la hora de manejar la aplicación. Y es que, en esta versión, era necesario deslizar la pantalla para acceder a todos los botones disponibles, lo cual era poco intuitivo para algunos usuarios. Para solventar esto, no solo se modificó el diseño de la interfaz de usuario, sino que además se añadió una barra de navegación [Fluh] para distribuir mejor el contenido en diferentes pantallas.

4.3.2.b. Versión final

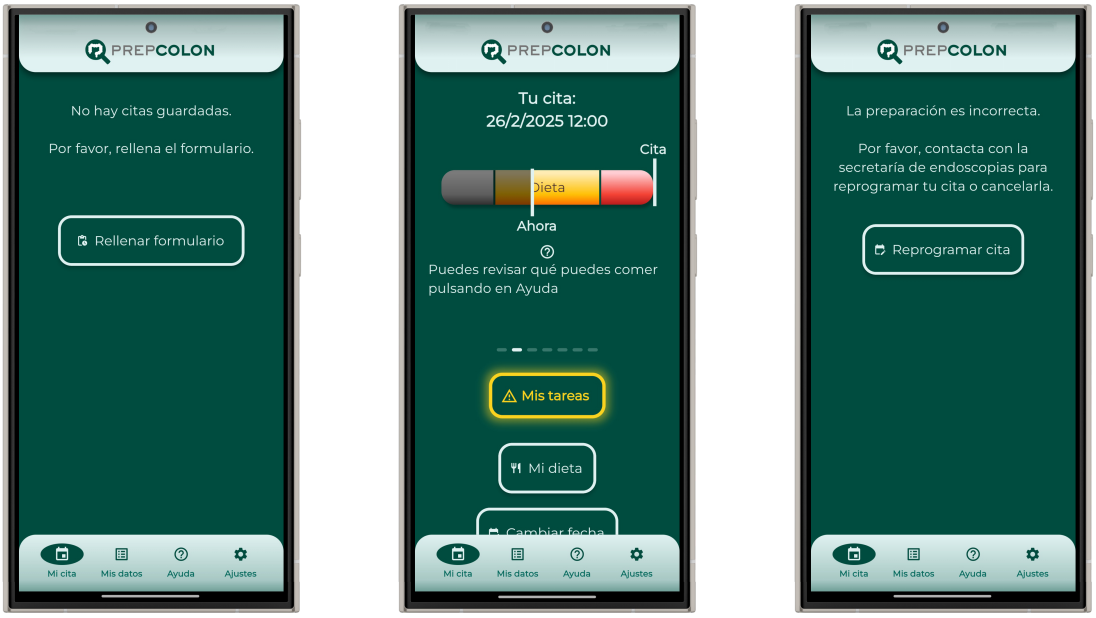
Gracias a la realimentación proporcionada por las diferentes personas que fueron probando la aplicación, y a la visión del equipo médico, finalmente se consiguió una interfaz de usuario lo suficientemente intuitiva como para que usuarios poco familiarizados con el uso de dispositivos

móviles pudieran utilizarla sin problemas. Esto fue posible, principalmente, gracias a la barra de navegación, la cual permitió separar de forma más clara y visual todo el contenido de la aplicación en diferentes pantallas, como se puede apreciar en las Figuras 4.6, 4.7 y 4.8.

Pantalla «Mi cita»

Esta es la primera pantalla que ve el usuario al entrar en la aplicación. En ella se mantuvieron la información sobre la fecha y la hora de la cita y los botones de avisos y de dieta. Se incluyó también un botón para modificar la fecha y la hora de la cita, de manera que no sea necesario repetir todo el formulario si ya se completó. El mensaje de recordatorio se sustituyó por un *carousel* con diferentes mensajes entre los que se incluyen recordatorios, ayuda sobre la aplicación o información adicional que pudiera ser de ayuda para el usuario. También se añadió un `widget` personalizado que muestra una barra de progreso con tres etapas (ver Figura 4.6b):

- Etapa previa al comienzo de la preparación.
- Etapa de dieta, previa al comienzo de la toma del laxante principal.
- Etapa final, durante las últimas horas antes de la cita.



(a) **(b)** **(c)**

Fig. 4.6: *Pantalla «Mi cita» de la versión final de PrepColon para los casos: (a) sin datos guardados, (b) antes de empezar a tomar el laxante y (c) tras detectar una mala preparación.*

Pantalla «Mis datos»

En esta pantalla el usuario puede ver sus respuestas del formulario, así como modificarlo si fuese necesario.

Pantalla «Ayuda»

(a) **(b)**

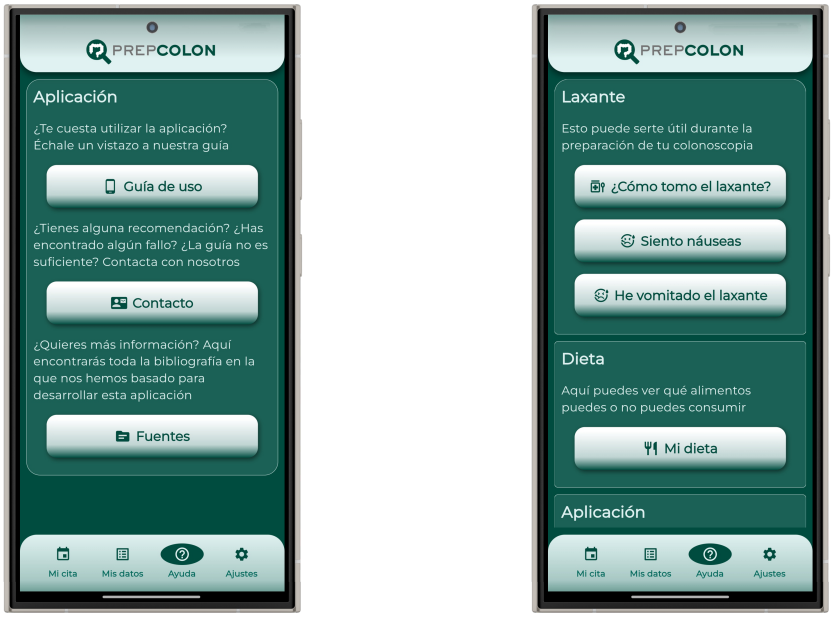
Fig. 4.7: *Pantallas (a) «Mis datos» y (b) «Ajustes»*

Se añadió esta pantalla para ofrecer al usuario información relevante adicional. Si no ha comenzado la dieta o la toma del laxante, sólo se muestran botones de ayuda relacionados con la aplicación en sí, que llevan a la página web de PrepColon, como se muestra en la Figura 4.8. Estos son:

- Guía de uso.
- Contacto.
- Bibliografía[1].

Una vez comenzadas la dieta o la toma de laxante, también aparecen en «Ayuda» botones que ofrecen información sobre el laxante o sobre la dieta.

[1] Al tratarse de una aplicación médica, uno de los requisitos de Apple para publicarla era incluir las fuentes bibliográficas en las que nos habíamos basado.



(a) **(b)**

Fig. 4.8: *Pantalla «Ayuda» (a) antes y (b) después de comenzar la preparación.*

Pantalla «Ajustes»

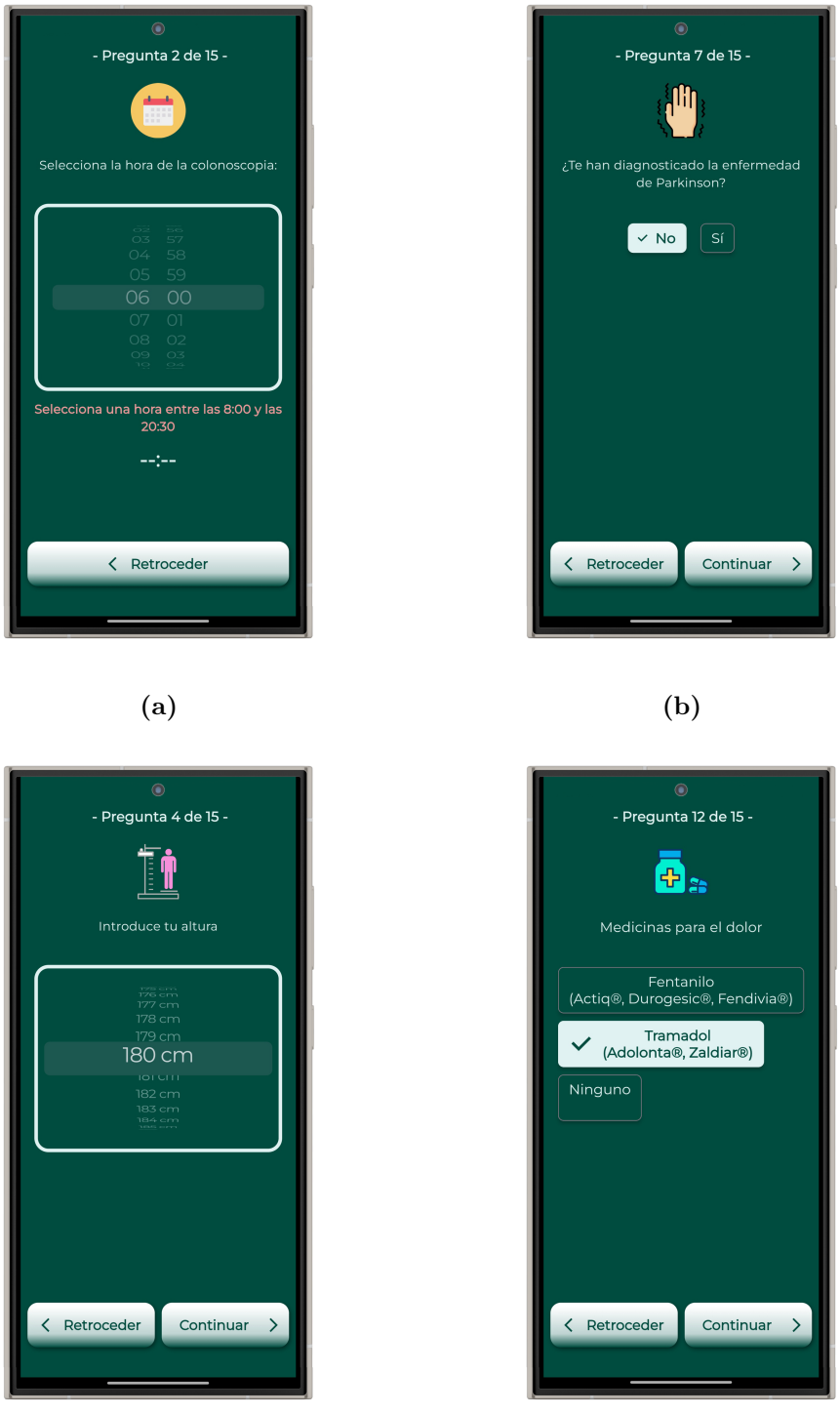
Aunque en un principio no estaba contemplada esta parte de la aplicación, se pensó que podía ser útil para añadir funciones que, de algún modo, no encajaban del todo en las otras pantallas. Dichas funciones se pueden ver en la Figura 4.7b.

- Notificaciones: Aunque no es recomendable desactivarlas, el equipo médico creyó conveniente incluir esta opción por si los usuarios se sentían agobiados por todas las notificaciones que genera la aplicación. Aun así, si se desactivan, se muestra un mensaje que anima a volver a activarlas.
- Registros: La aplicación, al igual que la inmensa mayoría de *software* actual, genera *logs* que registran los eventos que ocurren durante su ejecución. Estos registros totalmente anónimos son muy útiles para detectar errores o para análisis estadístico, por lo que son enviados automáticamente. Sin embargo, el usuario final tiene la opción de desactivar este envío automático si así lo desea.
- Borrar datos: El usuario puede decidir eliminar todos sus datos por diferentes motivos. En las versiones anteriores, la funcionalidad implementada por este botón era la del botón «Cancelar cita». Sin embargo, esta etiqueta se cambió ya que los usuarios debían tener claro que la aplicación no está ligada al hospital en el que se van a realizar la colonoscopia, y por lo tanto borrar los datos de la aplicación no equivale a cancelar su cita.

Formulario

El formulario es una parte clave de la aplicación, y en sus versiones iniciales éste ya estaba correctamente implementado. Sin embargo, en las últimas versiones se decidió modificar

ligeramente su diseño para adaptarlo a la nueva paleta de colores. Se aprovechó este rediseño para incorporar `chips` de *Material Design 3* [MD3] en las respuestas del formulario. Se puede observar en la Figura 4.9 el nuevo aspecto del formulario, en comparación con la Figura B.2.



(a) **(b)**

(c) **(d)**

Fig. 4.9: *Capturas de pantalla del formulario de la aplicación.*

4.4. Servidor

Como ya se discutió anteriormente, la aplicación PrepColon formará parte de un sistema distribuido que permita tanto enviar los datos recogidos por la propia aplicación al equipo médico mediante correo electrónico como verificar los códigos de los pacientes para garantizar que cada paciente tiene un código único y válido. El servidor deberá presentar, por lo tanto, dos funcionalidades principales. Por un lado, recibir los datos de la aplicación, adjuntarlos a un mensaje automático y enviar dicho mensaje a través de una cuenta y un servidor de correo electrónico propios a la dirección determinada por el equipo médico. Por otro lado, almacenar una lista con los posibles códigos de paciente y los códigos que ya están en uso, para poder decidir si el código que introduzca un paciente en su dispositivo móvil es válido o no.

Aunque existen varias formas de diseñar la API que permita crear un servidor que se adecúe a las necesidades del sistema, actualmente una manera ampliamente utilizada de hacerlo es mediante servicios RESTful, basados en los principios REST [Fie00]. Este diseño presenta varias ventajas que permitirán facilitar el diseño del sistema:

- Ausencia de estado: el servidor no almacena información adicional sobre su comunicación con el cliente, por lo que todos los datos necesarios para que una petición se pueda procesar correctamente deben ir en la propia petición. Esto simplifica el diseño del servidor, ya que no necesita almacenar información adicional sobre cada cliente con el que se comunica.
- Arquitectura cliente–servidor: el desarrollo del cliente y el desarrollo del servidor son completamente independientes, separando así la responsabilidad de cada una de las partes. De esta forma, se facilita el diseño de ambas partes.
- Interfaz uniforme: en una arquitectura REST, los recursos se manejan mediante un identificador único, conocido como URI [RR07; BFM05]. Esto permite el intercambio entre cliente y servidor de representaciones de estos recursos a través de las operaciones HTTP [Nie+99]. Gracias a la interfaz uniforme, la información se transfiere de manera estandarizada, lo que facilita el desarrollo del sistema al conseguir que las interacciones entre el cliente y el servidor sean independientes de la implementación concreta que se use en cada uno de ellos.

4.4.1. API RESTful

En las Tablas 4.9 y 4.8 se listan las operaciones que la aplicación PrepColon puede invocar en el servidor. En la Tabla 4.9 se indica el método HTTP [Nie+99] de cada operación, siendo `GET` para aquellas operaciones en las que toda la información necesaria va en la URI de la petición, y `POST` para las operaciones en las que el servidor recibe datos adicionales que no pueden ir directamente en la URI de la petición, bien por su extensión o bien por tratarse de información sensible. También en la misma Tabla se muestra la URI para cada recurso [BFM05], que hace las veces de un identificador que permite al servidor saber qué recurso está solicitando el cliente.

ID	Método HTTP	Ruta
OP00	GET	/api/validation/{userId}
OP01	POST	/api/{userId}/data
OP02	POST	/api/logs
OP03	POST	/web/mail

Tabla 4.8: *Operaciones disponibles en el servidor de la aplicación PrepColon.*

ID	Descripción
OP00	Comprueba si el código `userId` recibido es válido y no está en uso.
OP01	Recibe los datos recopilados por la aplicación asociados al paciente que tenga asignado el código `userId` recibido. Con los datos recibidos, forma un mensaje de correo electrónico y lo envía al equipo médico.
OP02	Recibe los registros generados por la aplicación, forma un mensaje de correo electrónico con ellos y lo envía a la cuenta de correo electrónico de la aplicación PrepColon.
OP03	Genera un mensaje de correo electrónico a partir de los datos introducidos en el formulario de contacto disponible en la página web asociada y lo envía a la cuenta de correo electrónico de la aplicación PrepColon.

Tabla 4.9: *Descripción de las operaciones disponibles en el servidor de la aplicación PrepColon.*

Este diseño de los recursos podría no ser el más acertado. Por ejemplo, la operación `OP00` combina lectura y escritura, ya que el servidor lleva una lista de todos los códigos utilizados y, si recibe una petición con un código que no existe en la lista, la modifica para que dicho código quede registrado. Por ejemplo, en este caso, una solución más alineada con los principios REST sería dividir esa operación en dos, una para leer de la lista de códigos utilizados y otra para actualizarla. En esa misma operación, un identificador más lógico podría ser `/api/{userId}/validation`, de manera que el recurso solicitado, la validación, quede vinculado al `userId`. Estos fallos se deben a decisiones de última hora tomadas por el cliente, ante las cuales se optó por realizar cambios rápidos para volver a poner el foco lo antes posible en la aplicación. Sin embargo, estas soluciones, aunque funcionales, no han resultado ser las más idóneas y sería conveniente modificarlas en el futuro.

Aunque inicialmente no estaba previsto, uno de los requisitos de Google y Apple para publicar la aplicación en sus tiendas era contar con una página web asociada a la aplicación. Aprovechando esto, se decidió incluir en dicha página web un formulario de contacto para que los usuarios pudiesen comunicarse directamente con el desarrollador de PrepColon—autor de este TFG—en caso de encontrar algún fallo en el funcionamiento de la aplicación o en caso de tener dudas sobre cómo utilizarla. Esto es posible gracias a la operación `OP03` , incluida en las tablas anteriores a pesar de no estar directamente relacionada con la comunicación entre la aplicación PrepColon y el servidor.

4.4.2. Esquema de recursos

En la Tabla 4.9 se ha podido observar cómo algunas operaciones del servidor utilizan el método `POST`. Para estas operaciones, el servidor requiere que el cliente envíe una cantidad relativamente grande de datos, por lo que sería contraproducente incluirlos en la URI de la

petición. En las operaciones `OP02` y `OP03`, los datos que se envían son relativamente poco relevantes, por lo que el servidor acepta cualquier cadena de texto incluida en la petición.

Sin embargo, en la operación `OP01` lo que se envían son los datos recopilados por la aplicación que usará el equipo médico para determinar la eficacia de la aplicación y que, precisamente por esto, resultan especialmente importantes. El servidor debe garantizar que los datos enviados por la aplicación son válidos antes de reenviarlos al equipo médico, y por ello se definió un esquema que indica el formato que deben seguir las peticiones recibidas.

```
userCsv:string
warningCsv:string
appData:
    login:string (date-time)
    id:string
    appointment:
        date:string(date-time)
        morning_shift:boolean
    form:
        started:string(date-time)
        finished:string(date-time)
        last_modified:string(date-time)
        weight:integer
        height:integer
        intensify_prep:boolean
        weekly_stools:boolean
        diabetic:boolean
        parkinson:boolean
        bariatric:boolean
        abdomen:boolean
        medicines:[ string ]
        previous_colonoscopy:boolean
    product:
        name:string
        collected:boolean
        collected_date:string (date-time)
        takes:[ string (date-time) ]
    laxative:
        collected:boolean
        collected_date:string (date-time)
        laxative_takes:[ string (date-time) ]
    warnings:[
        {
            code:integer
            message:string
            created:string (date-time)
            deadline:string (date-time)
            completed:boolean
            was_completed_by_card:boolean
            required:boolean
        }
    ]
    other:
        completed_questions:[ boolean ]
        form_finished:boolean
        form_completed:boolean
        bad_preparation:boolean
        bad_prep_timestamp:[ string (date-time) ]
    notifications_enabled:boolean
```

```
        notification_state_list:[ string ]
```

El esquema presenta tres campos principales. El campo `appData` contiene los datos «en crudo» tal y como los generó la aplicación, compuesto por los campos que pueden observarse en el esquema anterior. El campo `warningCsv` contiene la lista de tareas generada por la aplicación—es decir, aquellas que el usuario debe completar para conseguir una preparación adecuada para su colonoscopia—en formato CSV [Sha05]. El campo `userCsv` contiene el resto de datos, como la información médica del usuario, el laxante que seleccionó o la fecha de la colonoscopia, también en formato CSV. Fue el propio equipo médico el que solicitó recibir los datos en un formato compatible con hojas de cálculo, por lo que se decidió utilizar el formato CSV, ya que es un formato abierto de datos tabulares sencillo de usar. Para evitar trabajar con datos de pacientes en el servidor, es la aplicación la que genera el contenido en formato CSV y lo envía al servidor como una cadena de caracteres. Partiendo de este esquema, el servidor envía correos electrónicos al equipo médico adjuntando tres ficheros:

- Un fichero CSV con los datos recopilados por la aplicación, excepto las tareas generadas, para poder trabajar con ellos desde un programa de hoja de cálculo.
- Un fichero CSV con la lista de tareas generadas, para poder trabajar con ello desde un programa de hoja de cálculo.
- Un fichero JSON con todos los datos «en crudo», por si hubiese habido algún error al generar los ficheros CSV.

4.5. Conclusiones

A lo largo de este capítulo se ha mostrado una propuesta del diseño del sistema que conformará este proyecto, dividido en los diferentes elementos que formarán parte del mismo, buscando en todo momento tanto la cohesión entre ellos como la satisfacción de los requisitos indicados en el capítulo anterior. Sin embargo, hasta ahora sólo se ha trabajado sobre el papel, con una idea más o menos abstracta del par cliente–servidor. En el siguiente capítulo se abordará la forma de extraer las ideas y soluciones que han surgido en este documento para materializarlas a través del código y, de esta forma, poder obtener una aplicación real que pueda ejecutarse en diferentes dispositivos móviles.

Capítulo 5

Implementación

Anteriormente se ha analizado y discutido qué elementos son necesarios para construir un sistema que permita ayudar a los pacientes a prepararse adecuadamente para sus colonoscopias. Y, una vez definidos, el siguiente paso es materializarlos. Este capítulo se divide en dos partes. Por un lado, se discute qué tecnologías y lenguajes se van a utilizar para desarrollar la aplicación móvil, y se describe cómo se ha creado el código de la parte cliente del sistema. Por otro lado, se explica cómo se ha implementado el servidor que servirá de puente entre los dispositivos de los usuarios y el equipo médico.

5.1. Introducción

Tras la búsqueda de aplicaciones similares, el análisis de las necesidades del cliente, la identificación de requisitos, la definición de casos de uso y la toma de decisiones arquitectónicas; la siguiente fase del desarrollo de la aplicación, la cual se abordará en este capítulo, es convertir en código todos esos elementos para crear un programa funcional que cumpla con los requisitos propuestos.

En este capítulo se comenzará explicando cómo se ha desarrollado la aplicación móvil, discutiendo primeramente las diferentes opciones existentes en cuanto a tecnologías y lenguajes para elegir la solución que mejor se adapte a las necesidades del proyecto. Una vez hecha la elección de tecnologías, se describirá el desarrollo de la aplicación, buscando ajustarse a las características definidas en el capítulo anterior. Finalmente, se explicará cómo se ha implementado el servidor para lograr un sistema distribuido en el que los datos recopilados por la aplicación móvil lleguen a la bandeja de entrada del correo electrónico del equipo médico.

5.2. Aplicación móvil

De manera similar a como se hizo en el capítulo anterior, primero se describirá la implementación de la aplicación móvil, el cliente del sistema distribuido cuyo diseño quedó definido

en la Sección 4.2. Para desarrollar la aplicación, el primer paso será determinar qué lenguajes o tecnologías utilizar. Tras ello, se definirá la estructura de los directorios y de los ficheros en los que se escribirá el código. Finalmente, se detallará cómo se programará todo para cumplir con los requisitos propuestos y las funcionalidades deseadas.

La aplicación PrepColon se encuentra disponible actualmente de manera pública en Google Play Store[1] y es compatible con dispositivos que implementen la API 21 (Android 5) y versiones posteriores. También cuenta con una versión compatible con dispositivos de Apple con sistema operativo iOS 12 o posterior; sin embargo, debido a las restricciones para publicar en App Store, esta versión de la aplicación solo es accesible a través de la plataforma de pruebas beta «*TestFlight*»[2].

Debido a la extensión que pudieran tener algunos ficheros de código, en este documento se intentarán representar únicamente los fragmentos más significativos que permitan al lector hacerse una idea de cómo se han implementado las características más destacables de la aplicación.

5.2.1. Tecnologías

El objetivo de este proyecto es el desarrollo de un programa que pueda usarse desde cualquier dispositivo móvil, lo cual nos abre varias alternativas:

- Hacer una aplicación web.
- Hacer una aplicación nativa.
- Hacer una aplicación multiplataforma.

La idea de una aplicación web puede resultar atractiva. Para acceder a ella simplemente hace falta un navegador web, y tanto Android como iOS son compatibles con ellos. Además, sólo sería necesario tener un código base, lo que simplifica las tareas de mantenimiento, y los usuarios utilizarían en todo momento la versión más reciente, sin necesidad de preocuparse por actualizaciones. Sin embargo, esta solución presente algún inconveniente que no se puede pasar por alto. Teniendo una aplicación web, alojada en un servidor web, existe el riesgo de que el servidor, por diferentes motivos, deje de estar disponible en un momento dado o que el usuario no tenga conexión a Internet, lo cual incumpliría el requisito `NFR06` . También, al estar la aplicación en un dispositivo diferente al móvil desde el que accede el usuario, se pone en riesgo el requisito `NFR05` ya que habría que almacenar los datos de usuario en el servidor y enviarlos por la red cada vez que éste se conecte a la aplicación, lo que los expone a ser intereceptados por terceros. Otro requisito que podría no funcionar es el requisito `US04` , ya que las notificaciones funcionan de manera muy diferente en distintos navegadores o sistemas operativos [Netb; ES23]. Los usuarios deberían utilizar un navegador que permita enviar notificaciones no sólo mientras está abierto sino también en segundo plano, para lo que no sólo habría que buscar qué navegadores tienen esa característica sino que también habría que obligar a los usuarios a utilizar esos navegadores concretos, lo cual, teniendo en cuenta que los potenciales usuarios tendrán entre 50 y 69 años [Est14], puede resultar complicado si no están familiarizados con el uso de dispositivos móviles.

[1] Disponible en: https://play.google.com/store/apps/details?id=app.prepcolon
[2] Disponible en: https://apps.apple.com/es/app/testflight/id899247664

Optando por la aplicación nativa, sería necesario hacer un programa independiente para cada sistema operativo en el que se quisiera utilizar la aplicación. Observando la Figura 5.1 se puede ver cómo, en los últimos años, prácticamente todo el mercado de dispositivos móviles ha quedado ocupado por Android e iOS, lo cual simplifica las cosas ya que sólo haría falta desarrollar dos códigos base. Sin embargo, esta opción presenta un aspecto importante a tener en cuenta, y es que ambos sistemas operativos ofrecen diferentes lenguajes para programar aplicaciones nativas: Java o Kotlin para Android [Devc; Deva], y Objective-C o Swift para iOS [IBM]. Escogiendo esta solución habría que hacer un análisis de las ventajas e inconvenientes de cada uno de ellos, estudiar sus curvas de aprendizaje—ya que el autor de este documento no está familiarizado con Kotlin, Objective-C ni Swift—y considerar las tendencias actuales para determinar qué lenguajes prefieren tanto Google como Apple que usen sus desarrolladores.

Por otra parte, la principal ventaja del desarrollo multiplataforma es que un único código puede adaptarse de forma rápida y sencilla a diferentes tipos de dispositivos. También existen múltiples bibliotecas que permiten acceder a APIs propias de cada dispositivo, con lo que superamos también el obstáculo de las aplicaciones web en cuanto a los requisitos `USO4` , `NFR05` y `NFR06` . Además, al necesitar únicamente un código base, las tareas de mantenimiento y de corrección de errores se simplifica también enormemente. Quizás la única desventaja frente a desarrollo nativo pueda ser el rendimiento, pero dado que la aplicación no realizará ningún procesamiento excesivamente complejo, este inconveniente no parece relevante.

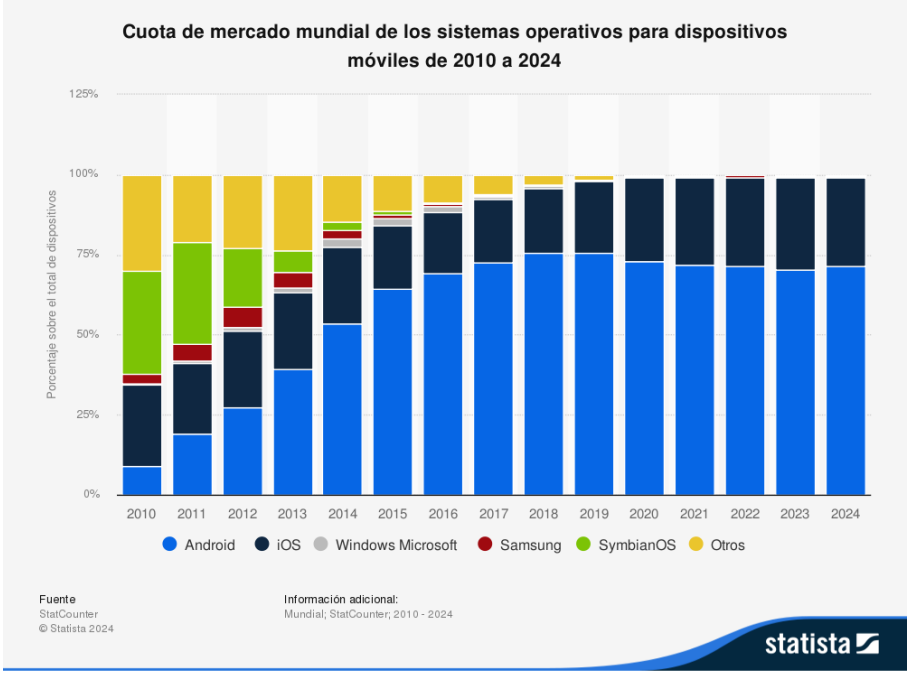


Fig. 5.1: *Cuota de mercado mundial de los sistemas operativos para dispositivos móviles de 2010 a 2024[Sta24].*

El siguiente paso es determinar qué tecnología utilizar. Y es que existe una gran variedad de herramientas y lenguajes que permiten el desarrollo móvil multiplataforma. Analizando la tabla 5.1, vemos que el *framework* más usado en los últimos años es Flutter, que utiliza el lenguaje Dart. Se podrían pensar en varias razones que justifiquen esto [Flutter][Dart]:

- Permite desarrollar aplicaciones y programas para iOS, Android, Linux, Mac, Windows, web y sistemas embebidos en dispositivos inteligentes y vehículos.

- El 90 % del código es común a todas las plataformas, y permite generar automáticamente el 10 % restante, específico de cada dispositivo.
- Dart está optimizado para compilarse directamente sobre los procesadores de los dispositivos, logrando así un rendimiento muy parecido al que se obtendría desarrollando directamente en nativo.
- Por el mismo motivo, Flutter permite crear interfaces de usuario más complejas, lo que mejora la experiencia de usuario.
- Su sintaxis es sencilla y clara, lo que hace que tenga una curva de aprendizaje rápida.
- Está desarrollado por Google, por lo que se puede integrar fácilmente con sus servicios.
- Es utilizado por grandes empresas: LG Electronics, Xiaomi, Universal Studios, Supercell, eBay, BMW, Toyota y muchas más.

No obstante, es importante tener en cuenta las desventajas que presenta este *framework*. El autor de este TFG ha considerado oportuno destacar tres principales debilidades. La primera es el tamaño de las aplicaciones, y es que el uso de bibliotecas que permitan acceder a funcionalidades nativas incrementan considerablemente el tamaño de una aplicación multiplataforma frente al que tendría si fuese desarrollada en nativa [Maj18]. La segunda es el uso de *Widgets*, un arma de doble filo que, por un lado, se convierte en el punto fuerte de Flutter al simplificar la creación de interfaces de usuario; pero por otro lado puede dificultar la legibilidad y la limpieza del código si el diseño es demasiado complejo—podríamos tener una gran cantidad de *Widgets* anidados. Por último, tanto Flutter como Dart forman parte de Google, por lo que existe el riesgo de que, en algún momento, Google decida abandonar alguno de estos proyectos, algo que ya ha ocurrido anteriormente con otros productos como Google Reader o Google+ [Ogd]. En ese caso sería necesario considerar otras alternativas que permitan una migración lo más sencilla posible. No obstante, la popularidad de Flutter—según la Tabla 5.1—y la gran cantidad de empresas que lo usan hace pensar que esta posibilidad es, por el momento, remota.

Dado que el desarrollo multiplataforma permite garantizar los requisitos propuestos en capítulos anteriores sin necesidad de desarrollar varios códigos fuente, y teniendo en cuenta que Flutter tiene una curva de aprendizaje sencilla y posibilita la creación de interfaces más complejas de manera relativamente sencilla, a la vez que cuenta con una amplia comunidad y una enorme cantidad de librerías para diferentes usos, los argumentos a favor de usar este *framework* parecen superar claramente a los argumentos en contra. Es por esto que se decidió desarrollar la aplicación PrepColon con Flutter.

Framework	**2019**	**2020**	**2021**	**2022**	**2023**
Flutter	30 %	39 %	42 %	46 %	46 %
React Native	42 %	42 %	38 %	32 %	35 %
Cordova	29 %	18 %	16 %	10 %	10 %
Unity	12 %	11 %	11 %	12 %	10 %
Ionic	28 %	18 %	16 %	11 %	9 %
Xamarin	26 %	14 %	11 %	12 %	8 %
Kotlin Multiplatforma	–	2 %	2 %	3 %	4 %
NativeScript	11 %	5 %	5 %	3 %	2 %
PhoneGap	11 %	6 %	4 %	2 %	2 %
Apache Flex	5 %	2 %	1 %	1 %	2 %
Kivy	–	1 %	1 %	1 %	1 %

Kendo UI	4 %	1 %	1 %	0 %	1 %
Otros	–	8 %	10 %	8 %	9 %

Tabla 5.1: *Frameworks multiplataforma para desarrollo móvil más populares según el porcentaje de desarrolladores que los usan a nivel mundial, de 2019 a 2023[Jet23].*

5.2.2. Estructura de directorios

La ventaja de Flutter es que permite generar automáticamente el código necesario para cada plataforma en la que se quiera desplegar la aplicación. El programador únicamente debe escribir el código necesario dentro del directorio `/lib` . Adicionalmente, se creará un directorio `/assets` para incluir ficheros adicionales que no formen parte del código en sí, pero que sean necesarios para el correcto funcionamiento de la aplicación PrepColon. En la Figura 5.2 se muestra el árbol de directorios en el que se estructura la aplicación PrepColon, y en la Tabla 5.2 se explican los contenidos de cada uno de ellos.

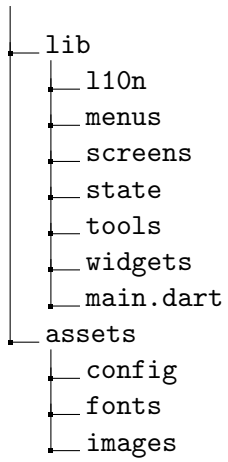


Fig. 5.2: *Árbol de directorios de la aplicación PrepColon.*

Directorio	Descripción
`/lib/l10n`	Ficheros de idiomas. Necesario para el uso de la librería `flutter_localizations` para la internacionalización de la aplicación
`/lib/menus`	Ficheros con los contenidos de las pantallas «Mi cita», «Mi dieta» y «Mi laxante». Estas pantallas cambiarán en función del estado en el que se encuentre la aplicación, por lo que cada fichero corresponde a un estado.
`/lib/screens`	Ficheros con los *Widgets* que representan las diferentes pantallas de la aplicación. Estas pantallas no cambian con el estado de la aplicación.
`/lib/state`	Ficheros con clases controladoras que gestionan el estado de la aplicación, los datos del usuario, las conexiones con el servidor, etc.
`/lib/tools`	Ficheros con clases auxiliares en las que se apoyan las clases del directorio anterior.
`/lib/widgets`	Ficheros con *Widgets* personalizados que ayudan a facilitar la legibilidad del código reduciendo el número de *Widgets* anidados en un mismo sitio.
`/assets/config`	Fichero de configuración.
`/assets/fonts`	Ficheros de fuentes de texto.

`/assets/images`	Imágenes utilizadas en la aplicación.

Tabla 5.2: *Descripción del contenido de los directorios de la aplicación PrepColon.*

5.2.3. Clases controladoras

Como se ha comentado anteriormente, las clases contenidas dentro del directorio `/lib/state` se encargan de gestionar los flujos de estado de la aplicación, los datos del usuario, las conexiones con el servidor, etc. Para asegurar que existe coherencia dentro de estos datos, la solución elegida es utilizar una única instancia de cada una de estas clases durante la ejecución de la aplicación. Así, cuando se necesite invocar algún método o leer algún atributo de ellas, será necesario obtener una referencia a esa instancia para trabajar sobre ella.

Flutter utiliza un estado interno, diferente a los estados descritos en la Sección 4.3.1.a, para permitir aplicaciones reactivas que ofrezcan una experiencia de usuario más fluida [Fluk]. Para gestionar este estado de la aplicación de manera rápida y sencilla, Flutter cuenta con el paquete `provider`, gracias al cual se pueden registrar clases que se encarguen de ello. En el Código 5.1 se muestra cómo se registran estas clases como *providers*[3].

```
1   void main async{
2       . . .
3       final AppState state = AppState();
4       final WarnManager wm = WarnManager();
5       final AppConfig config = AppConfig();
6       final ImageManager img = ImageManager();
7       final MenuState mst = MenuState();
8       final NetworkManager nm = NetworkManager();
9       . . .
10      runApp(
11          MultiProvider(
12              providers:[
13                  Provider(create: (context) => config),
14                  ChangeNotifierProvider(create: (context) => state),
15                  ChangeNotifierProvider(create: (context) => wm),
16                  Provider(create: (context) => img),
17                  Provider(create: (context) => mst),
18                  Provider(create: (context) => nm),
19              ],
20              child: const MyApp()
21          )
22      );
23      . . .
24  }
```

Código 5.1: *Fragmento de código del fichero `main.dart` en el que se registran instancias de las clases controladoras como «providers».*

En la Figura 5.3 se muestra el diagrama de clases que define estas clases controladoras. Todas ellas heredan de la clase `StateInterface` los métodos `initialization()`, que permite inicializar estas clases controladoras, y `cancelAppointment()`, que se llama cuando se produce un evento que interrumpe o finaliza el proceso de preparación de colonoscopias.

[3]La clase `MyApp` es el *Widget* principal de la aplicación PrepColon.

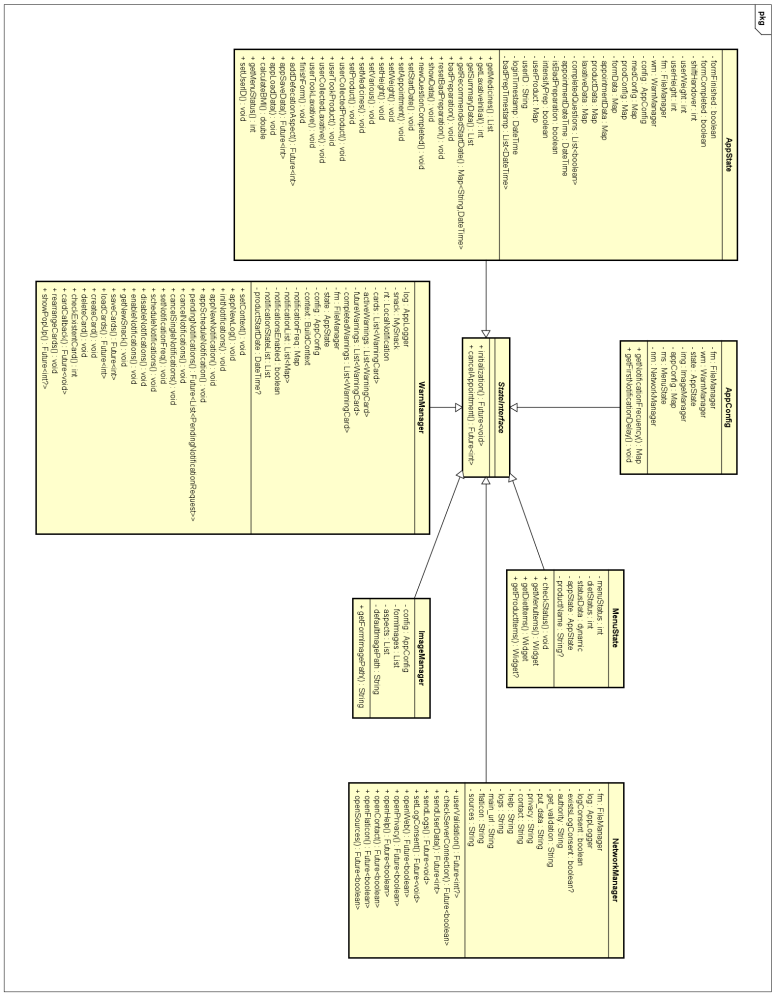


Fig. 5.3: *Diagrama de clases de las clases controladoras de la aplicación Prep-Colon.*

5.2.4. Implementación del patrón *State*

El fichero `/lib/state/menu_state.dart` contiene la clase `MenuState`, que es la encargada de gestionar el estado en el que se encuentra la aplicación. Como se veía en las Figuras 4.2 y 4.3, la aplicación necesita dos flujos de estado diferentes, por lo que esta clase utilizará dos variables diferentes, una para cada flujo. Estas variables son `_menuStatus` y `_dietStatus`, para los estados relacionados con el proceso de preparación y los estados relacionados con la dieta, respectivamente.

Para saber cuándo es la fecha de la colonoscopia y qué acciones ha realizado el usuario, la clase `MenuState` utiliza una instancia de la clase `AppState`, la cual guarda los datos de usuario, a través de la variable `_appState`. En el Código 5.2 se puede ver la implementación del método `checkStatus()`, que permite determinar en qué punto de cada uno de los flujos de estado se encuentra la aplicación.

```
1     checkStatus(){
2         _statusData = _appState.getStatusData ?? {};
3         _productName = _appState.getUserProductId;
4         if(_statusData['empty_data'] || _statusData.isEmpty){
5             /// No hay datos
6             _menuStatus = 0;
7             _dietStatus = 0;
```

```
        }else{
            final now = DateTime.now();
            if(_statusData['appointment_date'].compareTo(DateTime(1))!=0 && now.isAfter(
    _statusData['appointment_date'])){
                /// La fecha actual es [posterior] a la fecha de la cita
                _menuStatus = 7;
                _dietStatus = 0;
            }
            else if(_statusData['bad_preparation']){
                /// Se ha detectado una mala preparación
                _menuStatus = 8;
            }
            else if(!_statusData['form_completed']){
                /// El usuario completó el formulario
                _menuStatus = 1;
            }else if(now.isBefore(_statusData['appointment_date'].subtract(const
    Duration(days: 6)))){
                /// Quedan más de 6 días para la cita
                _menuStatus = 2;
                _dietStatus = 0;
            }else if(now.isBefore(_statusData['appointment_date'].subtract(const
    Duration(days: 1)))){
                /// Queda más de 1 día para la cita
                if(_statusData['intensive']){
                    _menuStatus = 3;
                }else{
                    _menuStatus = 2;
                }
                _dietStatus = 1;
            }else if(now.isBefore(_statusData['appointment_date'].subtract(const
    Duration(hours: 2)))){
                /// Quedan más de 2h para la cita
                _menuStatus = 4;
                _dietStatus = 2;
            }else{
                /// Quedan menos de 2h para la cita
                _menuStatus = 5;
                _dietStatus = 3;
            }
        }
    }
```

Código 5.2: *Fragmento de código del fichero* `menu_state.dart` *con la implementación del método* `checkStatus()`.

Esta clase también implementa métodos que devuelven el *Widget* que se mostrará al usuario en las pantallas «Mi cita», «Mi dieta» y «Mi laxante», respectivamente. La implementación de estos métodos se puede ver el Código 5.3, y en la Tabla 5.3 se describe el estado al que pertenece cada una de las clases devueltas de manera similar a las Tablas 4.2 y 4.3.

```
    Widget getMenuItems(){
        checkStatus();
        switch(_menuStatus){
            case 0:
                return const MenuNoData();
            case 1:
                return const MenuFormNoCompleted();
            case 2:case 3:case 4:case 5:
                return const MainMenu();
            case 6:
                return const MenuBadPreparation();
            case 7:
                return const MenuDataSend();
            default:
                return const MenuNoData();
        }
    }

    Widget getDietItems(){
        checkStatus();
        switch(_dietStatus){
```

```
22              case 0:
23                  return const DietNoRestrictions();
24              case 1:
25                  return const DietRestricted();
26              case 2:
27                  return const DietLiquid();
28              case 3:
29                  return const DietFast();
30              default:
31                  return const DietNoRestrictions();
32          }
33      }
34
35      Widget getProductItems(){
36          checkStatus();
37          if(_menuStatus==3){
38              return const PrepEvacuol();
39          }
40          if(_menuStatus==4){
41              switch (_productName) {
42              case 'bohn':
43                  return const PrepBohn();
44              case 'pleinvue':
45                  return const PrepPleinvue();
46              case 'citrafleet':
47                  return const PrepCitrafleet();
48              case 'moviprep':
49                  return const PrepMoviprep();
50              case 'citraliquid':
51                  return const PrepCitraliquid();
52              case 'clensia':
53                  return const PrepClensia();
54              default:
55                  return const NoPreparation();
56              }
57          }
58          if(_menuStatus==2){
59              return const NoPreparation(code: true,);
60          }
61          if(_menuStatus==5){
62              return const NoPreparation(code: false);
63          }
64          return const NoPreparation();
65      }
```

Código 5.3: *Fragmento de código del fichero* `menu_state.dart` *con la implementación de los métodos* `getMenuItems()`, `getDietItems()` *y* `getProductItems()`.

Clase	Descripción
`DietFast()`	El usuario debe hacer ayuno absoluto.
`DietLiquid()`	El usuario únicamente puede tomar líquidos claros.
`DietNoRestrictions()`	El usuario no tiene restricciones en su dieta.
`DietRestricted()`	El usuario tiene restricciones en su dieta.
`MainMenu()`	El usuario está tomando algún laxante.
`MenuBadPreparation()`	Se ha detectado una mala preparación por parte del usuario.
`MenuDataSend()`	Está pendiente el envío de datos.
`MenuFormNoCompleted()`	Estado intermedio en el que el usuario ha comenzado el formulario pero no lo ha completado.
`MenuNoData()`	Estado previo al inicio de la preparación, el usuario ha iniciado sesión pero no ha rellenado el formulario.
`PrepBohn()`	El usuario ha comenzado la toma de Solución de Bohn® o Casenglicol®.
`PrepCitrafleet()`	El usuario ha comenzado la toma de Citrafleet® (polvos).
`PrepCitraliquid()`	El usuario ha comenzado la toma de Citrafleet® (líquido).

`PrepMoviprep()`	El usuario ha comenzado la toma de Moviprep®.
`PrepPleinvue()`	El usuario ha comenzado la toma de Pleinvue®.
`PrepClensia()`	El usuario ha comenzado la toma de Clensia®.
`PrepEvacuol()`	El usuario ha comenzado la toma de Evacuol®.
`NoPreparation()`	El usuario no tiene que tomar ningún laxante.

Tabla 5.3: *Descripción de las clases contenidas en el directorio `/lib/menus` relacionadas con los estados de la aplicación PrepColon. La clase `NoPreparation()` puede recibir como parámetro un valor verdadero si se ha completado el formulario, en cuyo caso se mostrará el tiempo restante hasta que el usuario deba comenzar la toma de alguno de los laxantes; un valor falso si ya ha completado las tomas de los laxantes, o ningún valor si no ha completado el formulario, en cuyo caso se indicará que no hay ningún laxante registrado.*

5.2.5. Configuración de la aplicación

Como se discutió en el capítulo anterior, para cumplir los requisitos `NFR04` y `NFR06`, la aplicación PrepColon hará uso de un fichero externo de configuración en el que se definen las frecuencias de las notificaciones, la hora límite que marca si la cita es por la mañana o por la tarde, la lista de medicamentos, la lista de laxantes principales, la ruta de las imágenes de la aplicación—almacenadas en el directorio `/assets/images` —y las URIs de acceso al servidor; todo ello en formato JSON [JSON]. Todos estos datos se pueden encontrar en el fichero `appconfig.json`, en el directorio `/assets/config`, cuyo contenido se muestra parcialmente en el Código 5.4.

```
{
    "shift_handover":15,
    "digit_precision":3,
    "notifications":{
        . . .
        "frequency_default":{
            "extra_low":96,
            "low":48,
            "medium":24,
            "high":12,
            "extra_high":6,
            "ultra_high":2
        },
        . . .
    },
    "medicines":{
        "various":[
            {
                "id":"iron",
                "card_code":10,
                "days_before":5
            },
            . . .
        ],
        "nervous":[
            {
                "id":"amytriptiline",
                "card_code":99
            },
            . . .
        ],
        . . .
    },
    "products":{
        . . .
        "options":[
            {
                "id":"bohn",
```

```
                    "card_code":0,
                    "morning_shift":{
                        "start_time":19,
                        "is_day_before":true
                    },
                    "afternoon_shift":{
                        "start_time":7,
                        "is_day_before":false
                    },
                    "second_dose":5,
                    "takes_per_dose":8,
                    "duration":2
                },
                . . .
            ]
        },
        "images":{
            . . .
            "form":[
                {
                    "path":"assets/images/intestine.png",
                    "screen":"initial"
                },
                {
                    "path":"assets/images/calendar.png",
                    "screen":"date"
                },
                . . .
            ],
            . . .
        },
        "server_data":{
            "authority":"prepcolon.gsic.uva.es",
            . . .
            "get_validation":"/api/app/validation/{id}",
            "put_data":"/api/app/{id}/data",
            "privacy":"/#/privacy",
            "contact":"/#/contact",
            "help":"/#/help",
            . . .
        }
        . . .
    }
```

Código 5.4: *Fragmento de código del fichero de configuración `appconfig.json`.*

La clase `AppConfig`—fichero `/lib/state/app_config.dart` —es la encargada de leer este fichero de configuración para almacenar sus valores en variables dentro del código, así como de inicializar las demás clases controladoras. De esta forma se busca que únicamente exista una instancia de cada una de estas clases durante la ejecución de la aplicación, garantizando la integridad y la coherencia de los datos de usuario.

5.2.6. Validación del código de paciente

Como ya se indicó en la Sección 1.1, los usuarios que utilicen la aplicación PrepColon durante la fase del estudio clínico deberán contar con un código único que los identifique para poder anonimizar correctamente los datos generados por la aplicación. Para garantizar que los usuarios de la aplicación PrepColon son participantes del estudio clínico, fue necesario incluir una pantalla de validación en la que deben introducir el código que les fue asignado. Esta pantalla muestra un pequeño campo de texto y, tras introducir el código, se envía una petición al servidor para comprobar si el valor introducido es válido.

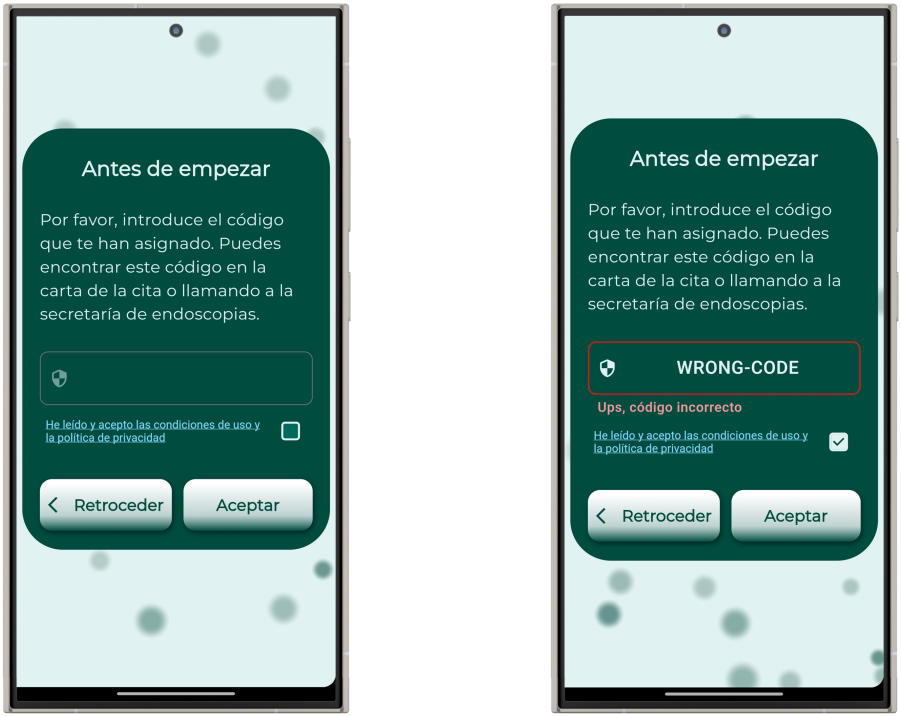
Hay dos clases que permiten esta funcionalidad. Por un lado, está la clase `ValidationScreen`—fichero `/lib/screens/validation_screen.dart` —, que implementa el *Widget* que conforma la interfaz de usuario para esta pantalla. Por otro lado, está la clase `NetworkManager`—fichero `/lib/state/network_manager.dart` —, clase controladora encargada de implementar los métodos que permiten la comunicación entre el cliente—la aplicación—y el servidor.

```
1   Form(
2       . . .
3       validator: (value){
4           if(!_privacyChecked){
5               return AppLocalizations.of(context)?.validation_privacy_wrong;
6           }
7           if(value==null || value.isEmpty){
8               return AppLocalizations.of(context)?.validation_empty;
9           }
10          if(_validCode!=null && !_validCode!){
11              return AppLocalizations.of(context)?.validation_wrong;
12          }
13          return null;
14      }
15      . . .
16  )
```

Código 5.5: *Fragmento de código del fichero `validation_screen.dart` que muestra la implementación del validador del campo de texto de introducción de código de paciente.*

En el Código 5.5 se muestra la función validadora [Flua] del formulario—que consta de un campo de introducción de texto y una casilla. Dentro de ella, se comprueba si la casilla está marcada—atributo *booleano* `_privacyChecked`—, si el formulario se ha rellenado y si el valor introducido es válido—atributo *booleano* `_validCode`. El valor que devuelve es una cadena de texto que se mostrará al usuario. Al tratarse de una serie de sentencias `if`, si no se cumple la condición de ninguna de ellas, se asumirá que el formulario se ha rellenado correctamente—es decir, se ha introducido un código de paciente válido y se ha marcado la casilla de condiciones de uso y política de privacidad—y no se devolverá ningún valor. En la Figura 5.4 se observa cómo reacciona la aplicación ante un código de paciente incorrecto.

En el Código 5.6 se muestra un fragmento del *Widget* `MyGradientButton` dentro del *Widget* principal de la clase `ValidationScreen`. Este elemento pertenece a la lista de *Widgets* personalizados contenidos en el directorio `/lib/widgets` y representa gráficamente algunos de los botones de la aplicación. En el caso del Código 5.6, se observa un fragmento del código que se ejecutará cuando el usuario presione el botón «Aceptar». Se comprueba la validez del código introducido llamando a la función `userValidation()` de la clase `NetworkManager`, que devuelve el código de respuesta enviado por el servidor. Si el valor devuelto es nulo, significa que no se ha podido establecer la conexión con el servidor; si la respuesta es un código 400, el código introducido no es válido, y se informa al usuario de ello a través de un diálogo emergente [Fluc], y si el código es válido—tiene el formato correcto y no está en uso—, se llaman a las funciones `setUserID()` y `appSaveData()` de la clase `AppState` para guardar los datos y se genera un *log* a través de la clase `WarnManager` indicando que el código ha sido validado y registrado.



(a) **(b)**

Fig. 5.4: *Ejemplo de uso de la aplicación PrepColon (a) antes y (b) después de introducir un código de paciente incorrecto.*

```
    Expanded(
        child: MyGradientButton(
            buttonPressed: () async{
                . . .
                String userCode = textController.value.text;
                int? validId = await network.userValidation(userId: userCode);
                . . .
                if(validId==null) {
                    if (context.mounted) {
                        await warning.showPopUp(
                        message:AppLocalizations.of(context)?.network_no_connection ?? '
    lang.network_no_connection',
                        options: 1,
                        ctxt: context,
                        );
                    }
                }else if(validId==400){
                    if(context.mounted){
                        await warning.showPopUp(
                        message: AppLocalizations.of(context)?.network_used_code ?? '
    lang.network_used_code',
                        options: 1,
                        ctxt: context
                        );
                    }
                }else{
                    if (formKey.currentState!.validate()) {
                        try {
                            appState.setUserID(textController.value.text);
                            await appState.appSaveData();
                            await warning.appNewLog(message: "ID registered: $userCode",
     type: LogTypes.info, id: AppLogger.VALIDATION);
                        } catch (e, t) {
                            await warning.appNewLog(message: "[ ERROR ] $e -- [TRACE] $t
```

```
    ", type: LogTypes.error, id: AppLogger.VALIDATION_ERROR);
                        }
                        . . .
                    }
                }
            },
        ),
    ),
```

Código 5.6: *Fragmento de código del fichero* ***validation_screen.dart*** *que muestra la implementación del botón «Aceptar». Al pulsar este botón se comprueba la validez del código introducido y, de ser válido, se pasa a la siguiente pantalla.*

La petición al servidor la hace la clase `NetworkManager` mediante su método `userValidation()`, donde primeramente comprueba que sea posible conectarse al servidor llamando al método `checkServerConnection()`, el cual, a su vez, hace uso de las funcionalidades de la clase `InternetAddress` [Flug]. Una vez comprobada la conectividad con el servidor, forma una URI que servirá para hacer una petición HTTPS [Flud] y esperará a recibir una respuesta, tal y como se puede comprobar en el Código 5.7. Las variables `get_validation` y `authority` corresponden a los campos del mismo nombre del fichero de configuración que se puede ver en el Código 5.4.

```
Future<bool> checkServerConnection() async{
    try{
        final connection = await InternetAddress.lookup(authority);
        if(connection.isEmpty || connection[0].rawAddress.isEmpty){
            return false;
        }
        return true;
    }on SocketException catch(e){
        await _log.e(e,id: AppLogger.SERVER_ERROR);
        return false;
    }
}

Future<int?> userValidation({required String userId}) async{
    if(!await checkServerConnection()) return null;
    String stringUrl = get_validation.replaceAll("{id}",userId);
    final Uri uri = Uri.https(authority,stringUrl);
    var response = await http.get(url);
    return response.statusCode;
}
```

Código 5.7: *Fragmento de código del fichero* ***network_manager.dart*** *que muestra los métodos* ***checkServerConnection()****, para comprobar la conexión con el servidor, y* ***userValidation()****, para hacer la solicitud al servidor que compruebe la validez del código del usuario.*

5.2.7. Envío de datos

El requisito `NFR09` establece que, cuando finaliza o se interrumpe la preparación del usuario, la aplicación debe enviar los datos recopilados al servidor para que éste los reenvíe al equipo médico. Volviendo a las Tablas 4.9 y 4.8, el envío de datos se hace mediante una operación `POST` y, de nuevo, la encargada de esto es la clase `NetworkManager` mediante su método `sendUserData()`, como se muestra en el Código 5.8. En este método, se comprueba primero que la aplicación tiene conexión con el servidor y después carga los datos de usuario almacenados. Tras ello, crea un mapa—formato `‘‘clave’’:valor`— en el que incluye dichos datos.

Una vez se encuentran todos los datos del usuario en una variable de tipo mapa, llama

a las funciones `convertJsonToCsv()` y `convertJsonListToCsv()` de la clase `FileManager`, una de las clases auxiliares disponibles en el directorio `/lib/tools` , que, como su nombre indica, convierten datos en formato JSON o mapa a formato CSV [Sha05], compatible con hojas de cálculo. Se puede ver en el Código 5.9 la implementación de ambos métodos. Finalmente, se genera la petición HTTPS y se envía mediante el método `POST` [Fluj]. Para evitar que la aplicación se bloquee se hace uso de promesas, que en Flutter se pueden implementar mediante la clase `Future` [Flue].

```
Future<int> sendUserData() async{
try{
  if(!await checkServerConnection()){
    return -1;
  }
  final userString = await _fm.loadData(name: FileManager.USER_FILE);
  final warningString = await _fm.loadData(name: FileManager.CARD_FILE);
  if(userString==null || userString.isEmpty){
    throw 'No user data stored';
  }
  final userData  = json.decode(userString);
  final warningData = (warningString!=null) ? json.decode(warningString)['warnings']
 : [];
  final notificationsEnabled = (warningString!=null) ? json.decode(warningString)['
notifications_enabled'] : true;
  final notificationStateList = (warningString!=null) ? json.decode(warningString)['
notification_state_list'] : [];
  final stringUrl = put_data.replaceAll("{id}",userData['user_Id']);
  final Uri url = Uri.https(authority,stringUrl);
  final jsonUserData = {
    . . .
  };
  String csvUserData = FileManager().convertJsonToCsv({
    "version_app":MyApp.APP_VERSION,
    "dispositivo":(Platform.isAndroid)?"Android":(Platform.isIOS)?"iOS":"otro",
    "codigo_usuario":jsonUserData['id'],
    'fecha_registro':jsonUserData['login'],
    'fecha_cita':jsonUserData['appointment']['date'],
    'modificaciones_fecha_cita': jsonUserData['appointment']['modifications'],
    'fecha_inicio_formulario':jsonUserData['form']['started'],
    'fecha_fin_formulario':jsonUserData['form']['finished'],
    'peso': jsonUserData['form']['weight'],
    'altura':jsonUserData['form']['height'],
    'prep_intensiva':jsonUserData['form']['intensify_prep'],
    'defecaciones_por_semana':jsonUserData['form']['weekly_stools'],
    'diabetico':jsonUserData['form']['diabetic'],
    'parkinson':jsonUserData['form']['parkinson'],
    'cirugia_bariatrica':jsonUserData['form']['bariatric'],
    'cirugia_abdominal':jsonUserData['form']['abdomen'],
    'medicamentos':jsonUserData['form']['medicines'],
    'colonoscopia_previa':jsonUserData['form']['previous_colonoscopy'],
    'nombre_preparacion':jsonUserData['product']['name'],
    'preparacion_recogida':jsonUserData['product']['collected'],
    'fecha_recogida_preparacion':jsonUserData['product']['collected_date'],
    'fechas_tomas_preparacion':jsonUserData['product']['takes'],
    'evacuol_recogido':jsonUserData['laxative']['collected'],
    'fecha_recogida_evacuol':jsonUserData['laxative']['collected_date'],
    'fechas_tomas_evacuol':jsonUserData['laxative']['laxative_takes'],
    'preguntas_completadas':jsonUserData['other']['completed_questions'],
    'usuario_vio_todas_preguntas':jsonUserData['other']['form_finished'],
    'usuario_completo_formulario':jsonUserData['other']['form_completed'],
    'mala_preparacion':jsonUserData['other']['bad_preparation'],
    'fechas_de_mala_preparacion':jsonUserData['other']['bad_prep_timestamp'],
    'notificaciones_activadas':jsonUserData['notifications_enabled'],
    'cambios_en_notificaciones_activadas':jsonUserData['notification_state_list'],
  });
  String csvWarningData = FileManager().convertJsonListToCsv(jsonUserData['warnings'
]);
  final jsonBody = {
    'userCsv':csvUserData,
    'warningsCsv':csvWarningData,
    'appData':jsonUserData,
  };
```

```
      var response = await http.post(
          url,
          headers: {
            "accept" : "application/json",
            "Content-Type" : "application/json"
          },
          body: jsonEncode(jsonBody)
      );
      if(response.statusCode==200) {
        return 0;
      } else {
        return -1;
      }
    }catch(e,t){
      await _log.e('[ERROR] $e -- [TRACE] $t',id: AppLogger.SERVER_ERROR);
      return -2;
    }
  }
```

Código 5.8: *Fragmento de código del fichero* ***network_manager.dart*** *que muestra la implementación del método que envía los datos de la aplicación al servidor para su reenvío al equipo médico.*

```
  String convertJsonToCsv(Map<String, dynamic> data) {
      List<String> headers = [];
      List<String> dataList = [];
      String headerString = '', dataString = '';
      if (data.isEmpty) return '';
      Iterable<String> keys = data.keys;
      for (String key in keys) {
          dynamic item = data[key];
          headers.add(key);
          switch (item.runtimeType) {
              case Null:
                  dataList.add("nada");
                  break;
              case String:
                  dataList.add(item.toString());
                  break;
                  case bool:
                  if(item){
                      dataList.add("Si");
                  }else{
                      dataList.add("No");
                  }
                  break;
              case int:
                  dataList.add(item.toString());
                  break;
              case List<bool>:
                  List list = [];
                  if (item.isEmpty) {
                      dataList.add('nada');
                      break;
                  }
                  for(bool value in item){
                      if(value){
                      list.add("Si");
                      }else{
                      list.add("No");
                      }
                  }
                  dataList.add(list.join(","));
                  break;
              case List<String>:
              case List<int>:
                  List list = item;
                  if (item.isEmpty) {
                      dataList.add('nada');
                      break;
                  }
                  dataList.add(list.join(","));
                  break;
```

```
                case List<Object>:
                case List<dynamic>:
                    if(item.isEmpty){
                        dataList.add('nada');
                        break;
                    }
                    List list = [];
                    for(dynamic element in item){
                        list.add(element.toString());
                    }
                    dataList.add(list.join(','));
                    break;
            }
        }
        headerString = headers.join("|");
        dataString = dataList.join("|");
        return "sep=|\n$headerString\n$dataString";
    }

    String convertJsonListToCsv(List<dynamic> data) {
        List<String> headers = [];
        List<String> dataList = [];
        String headerString = '', dataString = '';
        if (data.isEmpty) return '';
        Iterable<String> keys = data[0].keys;
        for (String key in keys) {
            headers.add(key);
        }
        for (int i = 0; i < data.length; i++) {
            dynamic item = data[i];
            for (String key in keys) {
                dataList.add(item[key].toString());
            }
        }

        headerString = headers.join("|");

        for (int j = 0; j < dataList.length; j++) {
            String stringItem = dataList[j].replaceAll("\n", " ");
            dataString += stringItem;
            if (j % headers.length == headers.length - 1) {
                dataString += "\n";
            } else {
                dataString += "|";
            }
        }
        return "sep=|\n$headerString\n$dataString";
    }
```

Código 5.9: *Fragmento de código del fichero* `file_manager.dart` *en el que se muestra la implementación de los métodos* `convertJsonToCsv()` *y* `convertJsonListToCsv()`*. Estos métodos también «traducen» valores para facilitar su entendimiento por parte de personas poco familiarizadas con la programación de forma que* `null` *se convierte en «nada»,* `true` *en «sí», y* `false` *en «no». La clase* `FileManager` *es una clase auxiliar diseñada para facilitar el manejo de ficheros por parte de la aplicación.*

5.2.7.a. Funcionamiento en segundo plano

De acuerdo con el equipo médico, los pacientes que acuden a sus citas para realizarse colonoscopias pueden tardar unas horas en recuperarse por completo de la sedación y de las estrictas dietas. Para evitar que los usuarios se olviden de enviar los datos pulsando el botón correspondiente, se decidió buscar una forma de que la aplicación enviase los datos automáticamente sin necesitar la interacción de los usuarios. La solución se encontró en el paquete `workmanager` [Com].

Su uso es sencillo, y comienza por registrar en el fichero `main.dart` las tareas que se desean ejecutar en segundo plano. En el Código 5.10 se muestran la tarea preparada para enviar los datos al servidor. Si el valor de retorno es `Future.value(true)`, la tarea ha finalizado. Si el valor devuelto es `Future.value(false)`, la tarea ha fallado, pero se intentará realizar de nuevo. Si el valor devuelto es `Future.error()`, la tarea ha fallado y no volverá a intentarse.

```
void main() async{
    . . .
    Workmanager().initialize(callbackDispatcher);
    . . .
}

@pragma('vm:entry-point')
void callbackDispatcher(){
    Workmanager().executeTask((taskName,inputData) async{
        try{
            switch(taskName){
                case MyApp.DATA_TASK:
                    NetworkManager network = NetworkManager();
                    final check = await network.sendUserData();
                    switch(check){
                        case 0:
                            await network.sendLogs();
                            await FileManager().wipeData(fileName: FileManager.
USER_FILE);
                            await FileManager().wipeData(fileName: FileManager.
CARD_FILE);
                            await FileManager().wipeData(fileName: FileManager.
LOG_FILE);
                            return Future.value(true);
                        case -1:
                            return Future.error("Wrong request");
                        case -2:
                            return Future.value(false);
                    }
                    return Future.error("Server not available");
            }
        }catch(e){
            return Future.error(e);
        }
        return Future.value(true);
    });
}
```

Código 5.10: *Fragmento de código del fichero `main.dart` en el que se inicializa la clase `Workmanager` para el funcionamiento en segundo plano. Dado que el uso de esta clase está pensado para enviar los datos después de la colonoscopia, tratará de borrar los datos almacenados una vez se complete el envío mediante el método `wipeData()` de la clase `FileManager`.*

Tras completar el usuario el formulario, y después de generarse todas las tareas del usuario y alertas, se cancelan todas las tareas pendientes de `Workmanager` para el envío de datos,si las hay, para volver a registrarla con los últimos datos proporcionados por el usuario. En el Código 5.11 se muestra cómo se registra esta tarea. La tarea comenzará a ejecutarse en segundo plano 30 minutos después de la fecha de la cita—atributo `initialDelay`—y, cada vez que su ejecución falle, se repetirá cada 30 minutos—atributos `backoffPolicyDelay` y `backoffPolicy`—siempre y cuando el dispositivo tenga conexión a Internet—atributo `constraints`.

```
final Duration delay = appointment.difference(DateTime.now()) + const Duration(
minutes:30);
Workmanager().cancelByUniqueName(MyApp.DATA_TASK);
Workmanager().registerOneOffTask(
    MyApp.DATA_TASK,MyApp.DATA_TASK,
    initialDelay: delay,
    backoffPolicyDelay: const Duration(minutes:30),
    backoffPolicy: BackoffPolicy.linear,
    constraints: Constraints(
```

```
            networkType: NetworkType.connected
        )
    );
```

Código 5.11: *Fragmento de código que permite registrar la tarea de envío de datos en segundo plano.*

5.2.8. Internacionalización

Como se ha comentado anteriormente, Flutter cuenta con una gran variedad de librerías creadas por y para la comunidad. Una de ellas es `flutter_localizations` [Fluf], cuyo propósito es adaptar las aplicaciones automáticamente al idioma del dispositivo en el que se están ejecutando de manera sencilla. Esto permite cumplir con los requisitos `NFR03` y `NFR04` —ya que, para añadir compatibilidad con un nuevo idioma, no será necesario modificar todos los textos de la aplicación.

Para lograr esto, primero será necesario incorporar, en la clase principal de la aplicación—en este caso, dentro de `main.dart`—código que permita detectar el idioma utilizado y las características del mismo—por ejemplo, si se escribe de izquierda a derecha o de derecha a izquierda—, así como indicar con qué idiomas será compatible la aplicación. Esto es posible gracias a los atributos `localizationsDelegates` y `supportedLocales` del *Widget* de `MaterialApp`, tal y como se muestra en el Código 5.12.

```
class MyApp extends StatelessWidget{
    . . .
    @override
    Widget build(BuildContext context){
        . . .
        child: MaterialApp(
            . . .
            localizationsDelegates: const [
                AppLocalizations.delegate,
                GlobalMaterialLocalizations.delegate,
                GlobalWidgetsLocalizations.delegate,
                GlobalCupertinoLocalizations.delegate
            ],
            supportedLocales: const[
                Locale('en'),
                Locale('es')
            ]
        )
        . . .
    }
}
```

Código 5.12: *Fragmento de código del fichero `main.dart`. Para la aplicación PrepColon, actualmente sólo los idiomas español e inglés están soportados.*

De esta manera se evita incluir en el código cadenas de texto con valores constantes. Únicamente será necesario escribir un identificador de la cadena de texto que se quiera mostrar al usuario. Dicho identificador deberá estar presente en un fichero por cada idioma soportado por la aplicación. En el Código 5.13 se muestra un ejemplo de esto. En el Código 5.14 se puede apreciar un fragmento del fichero `app_es.arb`, que contiene los textos en español que se muestran en la aplicación[4].

[4]Existe un fichero `app_en.arb` con cadenas de caracteres en inglés. Sin embargo, dado que los pacientes que usarán la aplicación durante el estudio clínico serán habitantes españoles, la prioridad de crear una versión en inglés era muy baja, por lo que dicho fichero está incompleto.

```dart
// En el fichero .dart
Text(AppLocalizations.of(context).hello)

// En el fichero app_en.arb
"hello":"Hello World!"

// En el fichero app_es.arb
"hello":"Hola Mundo!"
```

Código 5.13: *Ejemplo de uso de la librería `flutter_localizations`. Si el dispositivo del usuario está configurado en inglés, en la aplicación se mostrará el texto «Hello World!». En cambio, si el idioma es español, se verá el texto «Hola Mundo!».*

```arb
 "@@locale":"es",
"entity":"Hospital Universitario Río Hortega",

"@_Main":{},
"main_title":"PrepColon",
"main_appointment":"Tu cita:\n{date} {time}",
"@main_appointment":{
    "placeholders": {
        "date":{
            "type":"DateTime",
            "format":"yMd"
        },
        "time":{
            "type":"DateTime",
            "format": "Hm"
        }
    }
},
"main_appointment_alt":"Tu cita:\n{day,plural,=0{Hoy} other{Mañana}} {time}",
"@main_appointment_alt": {
    "placeholders": {
        "day": {
            "type":"int"
        },
        "time": {
            "type":"DateTime",
            "format":"Hm"
        }
    }
},
"main_no_data":"No hay citas guardadas.\n\nPor favor, rellena el formulario.",
"main_no_completed":"El formulario está incompleto.\n\nPor favor, complétalo.",
"main_bad_preparation":"La preparación es incorrecta.\n\nPor favor, contacta con la secretaría de endoscopias para reprogramar tu cita o cancelarla.",
"main_on_diet":"¡No te saltes la dieta!",
"main_bottom_nav_appointment":"Mi cita",
"main_bottom_nav_data":"Mis datos",
"main_bottom_nav_info":"Ayuda",
"main_bottom_nav_settings":"Ajustes",
"main_tooltip_appointment":"Ver datos de la cita",
"main_tooltip_data":"Ver mis datos",
"main_help":"Tengo problemas\ncon la app",
"main_data_send":"Aún no se han enviado los datos de la anterior cita.\n\nPuedes esperar a que se envíen automáticamente y volver más tarde o pulsar el botón para enviarlos ahora.",
```

Código 5.14: *Fragmento del fichero `app_es.arb`.*

5.3. Servidor

Tal y como se observa en la Figura 4.1, el servidor del sistema distribuido en el que funcionará la aplicación PrepColon es una pieza fundamental de este puzle. Gracias a él, el equipo

desarrollador del proyecto—el autor de este documento—puede acceder a los registros generados por la aplicación para detectar fácilmente posibles errores. Gracias a él, el cliente del proyecto—el equipo médico—puede acceder a los datos de los usuarios para evaluar la eficacia de la aplicación en el apoyo a la preparación de colonoscopias.

De igual manera que en la sección anterior se ha detallado la implementación de las funcionalidades más relevantes de la parte cliente, en esta sección se explicarán las características más significativas del servidor, presentando primero la estructura de directorios del mismo.

5.3.1. Estructura de directorios

Para la implementación del servidor se decidió utilizar Node.js® [Foub], una herramienta versátil, gratuita y de código abierto que cuenta con varias librerías enfocadas al desarrollo *backend*, lo que la convierte en una buena opción para esta tarea. En la Figura 5.5 puede observarse cómo se han estructurado los directorios dentro del servidor. En el fichero `app.js` se encuentra el código que define las rutas a los recursos e inicializa y ejecuta el servidor. En el directorio `/js` se encuentran los ficheros que definen e implementan las principales funcionalidades del servidor—como el código que se ejecuta cuando se reciben las peticiones del cliente o las funciones que permiten el manejo de ficheros. El directorio `/data` contiene un fichero de configuración del servidor y listas de códigos de paciente—por un lado, todos los códigos válidos; por otro lado, los códigos en uso. El directorio `/dist` alberga el código necesario para el funcionamiento de la página web de la aplicación, disponible en este enlace, y el directorio `/logs` almacena los registros que se generan durante la ejecución del servidor.

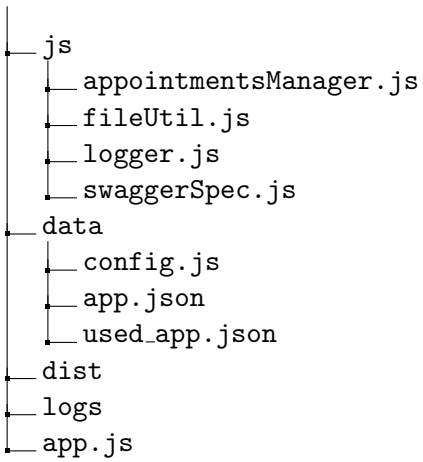


Fig. 5.5: *Árbol de directorios del servidor PrepColon.*

5.3.2. Validador e interfaz de usuario

En el capítulo anterior se propusieron la API RESTful y el esquema de recursos que permitirían al servidor cumplir con las necesidades del proyecto. Para lograr una gestión de los recursos más sencilla y eficiente se utiliza OpenAPI [Fouc], un estándar para definir APIs RESTful, Swagger [Smab], una herramienta de código abierto diseñada para este propósito, y *Express OpenAPI Validator* [M D], una herramienta que valida automáticamente las peticiones y respuestas del servidor en función de las APIs definidas. Una de las ventajas del uso de Swagger es su interfaz

de usuario (IU), una plataforma interactiva desde la que se pueden probar las diferentes operaciones disponibles sobre los recursos. En la Figura 5.6 se muestra, tomando la operación `OPOO` como ejemplo, la funcionalidad ofrecida por esta plataforma y, en la Figura 5.7, se observa la información proporcionada al enviar una petición y recibir la respuesta del servidor.

Para implementar la IU de Swagger, es necesario definir esquemas que especifiquen las peticiones que se pueden realizar sobre los recursos y los formatos de dichas peticiones [Smaa]. Esta característica resulta muy útil a la hora de desarrollar tanto el servidor como el cliente, ya que facilita la implementación del código utilizado para la comunicación entre ambas partes. En el Código 5.15 se muestra cómo se especifica una operación—reutilizando el ejemplo anterior—para poder ejecutarla desde la IU de Swagger y, en el Código 5.16 se define el esquema `ServerResponse` que se utiliza esa operación.

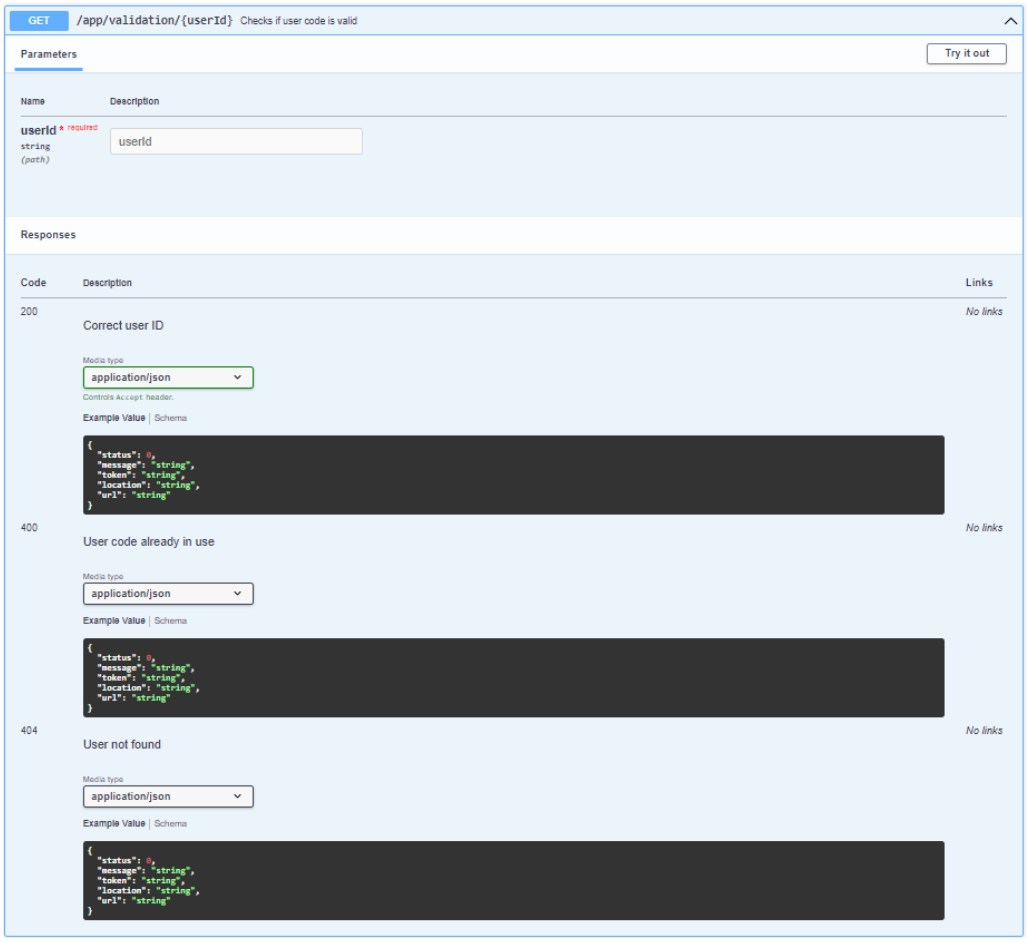


Fig. 5.6: *Ejemplo de la IU de Swagger para la operación `OPOO`.*

5.3.3. Implementación de operaciones

En la Sección 4.4.1 se presentaron las operaciones que formarían parte de la API RESTful que utilizaría la aplicación PrepColon. Cada una de esas operaciones tiene asociada una función que se ejecuta cuando el servidor recibe las peticiones correspondientes y en la que se prepara la respuesta que se enviará al cliente. En esta sección se explicará cómo se han implementado dichas funciones.

```
/**
* @swagger
* /app/validation/{userId}:
*   get:
*     summary: Checks if user code is valid
*     tags:
*       - app
*     operationId: validateUser
*     parameters:
*       - name: userId
*         in: path
*         required: true
*         schema:
*           type: string
*     responses:
*       '200':
*         description: Correct user ID
*         content:
*           application/json:
*             schema:
*               $ref: '#/components/schemas/ServerResponse'
*       '404':
*         description: User not found
*         content:
*           application/json:
*             schema:
*               $ref: '#/components/schemas/ServerResponse'
*       '400':
*         description: User code already in use
*         content:
*           application/json:
*             schema:
*               $ref: '#/components/schemas/ServerResponse'
*/
async function validateUser(req,res,next)
. . .
```

Código 5.15: *Definición de la operación `OPOO` para la IU de Swagger.*

```
/**
* @swagger
* . . .
* components:
*   schemas:
* . . .
*     ServerResponse:
*       type: object
*       additionalProperties: false
*       properties:
*         status:
*           type: integer
*           format: int32
*         message:
*           type: string
*         token:
*           type: string
*         location:
*           type: string
*         url:
*           type: string
*           format: uri
*       required: [status, message]
* . . .
*/
```

Código 5.16: *Definición del esquema `ServerResponse` que especifica el formato de las respuestas del servidor.*

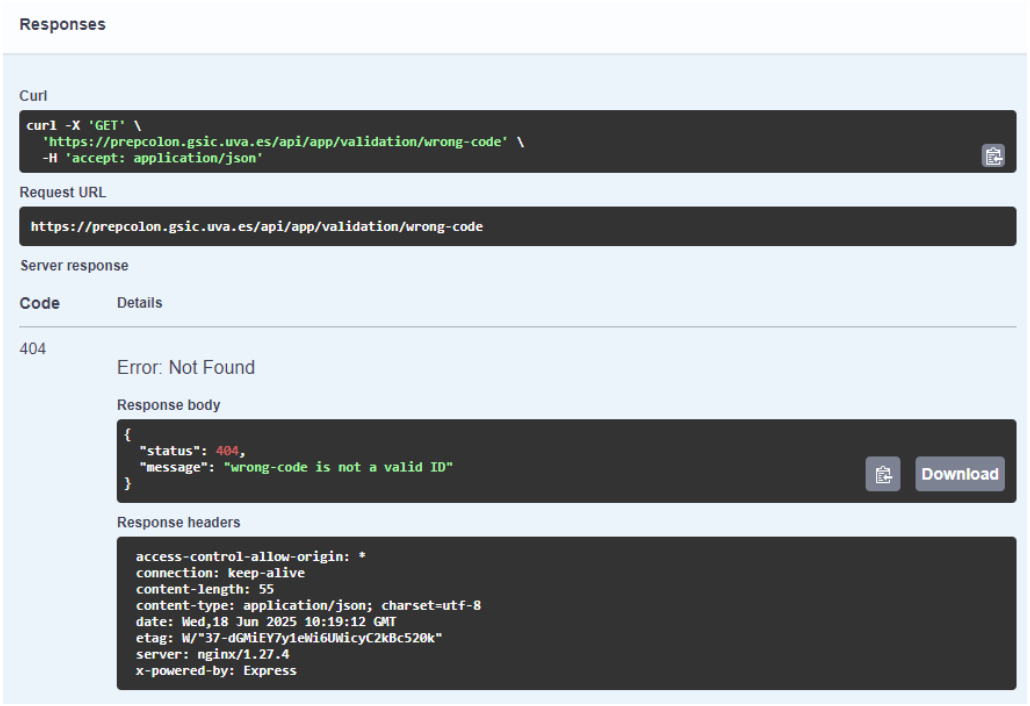


Fig. 5.7: *Información proporcionada por la IU de Swagger sobre la petición y la respuesta para la operación `OPOO`.*

El manejo de las peticiones del cliente se hace mediante las librerías del *framework* Express [Foua]. Esta herramienta permite definir las rutas a los recursos disponibles en el servidor e implementar el código a ejecutar cuando se soliciten esos recursos. En el fichero `app.js` se encuentra la función `initServer()` en el que, entre otras cosas, se definen estas rutas a los recursos. Para ello, primero se guardan en objetos [Neta] las rutas con las funciones que se llamarán al recibir una petición para cada recurso. Después, se utilizan los contenidos de esas variables como parámetros para las funciones `get()` y `post()` de Express [Foua] para definir las operaciones homónimas. En el Código 5.17 se muestra un fragmento de código del fichero `app.js` en el que puede observar cómo se definen las rutas.

```
1   const app = new express();
2   . . .
3   let getRoutes = {};
4   let postRoutes = {};
5   . . .
6   function initServer(){
7       . . .
8     getRoutes['/app/validation/:userId'] = appointmentsManager.validateUser;
9       postRoutes['/app/:userId/data'] = appointmentsManager.sendUserData;
10      postRoutes['/app/logs'] = appointmentsManager.sendLogs;
11      postRoutes['/web/mail'] = appointmentsManager.sendSupportMail;
12      . . .
13    for (let r in getRoutes)
14      app.get(r, getRoutes[r]);
15      for (let r in postRoutes)
16      app.post(r, postRoutes[r]);
17      . . .
18  }
```

Código 5.17: *Fragmento de código del fichero `app.js` en el que se inicializan las rutas a los recursos del servidor.*

La variable `appointmentsManager` incluye los métodos exportados en el fichero `appointmentsManager.js`, los cuales se explicarán a continuación. Estos métodos reciben tres parámetros [Foua]:

- `req` es un objeto que representa la petición enviada por el cliente.
- `res` es un objeto que representa la respuesta que enviará el servidor tras procesar la petición.
- `next` es una variable que permite redirigir las peticiones. Aunque aparece en las funciones, el servidor de la aplicación no la utiliza.

5.3.3.a. Validación de códigos (`OP00`)

Los códigos de paciente están formados por una cadena que identifica al hospital en el que el paciente se va a realizar la colonoscopia—`HURH` para el Hospital Universitario Río Hortega de Valladolid, `HMC` para el Hospital de Medina del Campo y `HZ` para el Hospital Virgen de la Concha de Zamora—, seguida de un número aleatorio de tres dígitos. Para comprobar la validez del código enviado por el cliente, el servidor sigue los siguientes pasos:

1. Se definió un código especial para realizar pruebas en la aplicación. Si el código enviado es este código de pruebas, el procesamiento termina y el servidor envía la respuesta.
2. Comprueba que el código tiene el formato correcto utilizando una expresión regular definida en el fichero de configuración `config.js`, y que puede verse en el Código 5.18.
3. Compara el código con la lista de códigos válidos del fichero `app.json` para comprobar que los tres dígitos sean correctos.
4. Compara el código con la lista de códigos en uso del fichero `used_app.json` para comprobar que ese código no se haya asignado ya a otro paciente.

Si el código es válido, lo almacena en la lista del fichero `used_app.json` para evitar que otro usuario trate de utilizarlo. En el Código 5.19 se muestra la implementación de la función `validateUser()` que lleva a cabo esta tarea.

```
1    /^p?(HURH|HMC|HZ)-\d{3}$/
```

Código 5.18: *Expresión regular para verificar el formato de los códigos de paciente.*

```
1  async function validateUser(req, res, next)
2  {
3    const userId = req.params.userId;
4    let resp = {};
5    let i;
6    let userFound = false;
7    let codeUsed = false;
8    if(userId === config.test){
9      /// Test user code
10     userFound = true;
11     resp.status = 200;
12     resp.message = userId + ' is a valid ID';
13     res.type('json');
14     res.status(resp.status).send(resp);
15     return;
16   }
17   if(config.pattern.test(userId)){
18     /// Is a real code
19     for(i=0; i<appData.length; i++){
20       let id = appData[i];
21       if(id === userId){
```

```
22          userFound = true;
23        }
24      }
25      /// Check if used
26      for(i=0; i<usedCodes.length; i++){
27        let id = usedCodes[i];
28        if(id === userId){
29          codeUsed = true;
30        }
31      }
32    }
33    if(!userFound){
34      resp.status = 404;
35      resp.message = userId + ' is not a valid ID';
36    }else if(codeUsed){
37      resp.status = 400;
38      resp.message = userId + ' is already in use';
39    }else{
40      resp.status = 200;
41      resp.message = userId + ' is a valid ID';
42      usedCodes.push(userId);
43      await fileUtil.saveFile(codeFilePath,usedCodes);
44    }
45    logger.info(resp.message);
46    res.type('json');
47    res.status(resp.status).send(resp);
48    return;
49  }
```

Código 5.19: *Implementación de la función* `validateUser()` *para la validación de los códigos de paciente.*

5.3.3.b. Envío de datos de usuario (`OP01`)

Del manejo de las peticiones de la operación `OP01` se encarga la función `sendUserData()`, que, no sólamente genera un correo electrónico con los datos de usuario para enviarlo al equipo médico, sino que también comprueba que los datos recibidos de la aplicación son correctos. Para ello, primero comprueba que el cuerpo de la petición incluye el código de paciente, y que éste es el mismo que el enviado a través de la URI de la petición. Después, comprueba la validez del código de manera similar a como se hace en el Código 5.19, con la diferencia de que, en este caso, la lista de códigos en uso sí debe contener este código. Una vez determinada la validez del código recibido, se genera y se envía un correo electrónico con tres ficheros adjuntos—los datos «en crudo» en formato JSON [JSON], los datos de usuario en formato CSV [Sha05], y los datos de las tareas propuestas también en formato CSV—a una dirección de correo electrónico proporcionada por el equipo médico. Finalmente, se modifica el fichero `used_app.json` para que el código pueda volver a usarse.

Para enviar correos electrónicos desde el servidor se utiliza la herramienta NodeMailer [Rei] y una cuenta de correo electrónico propia proporcionada por el grupo de investigación GSIC/E-MIC de la Universidad de Valladolid. En el Código 5.20 se muestra la implementación de esta funcionalidad.

```
1  async function sendUserData(req,res,next)
2  {
3    let resp = {};
4    let newData = {};
5    let i,j;
6    let codeToRemoveIndex;
7    let codeUsed = false;
8    const reqData = req.body.appData;
9    const userCsv = req.body.userCsv;
10   const warningsCsv = req.body.warningsCsv;
```

```
  try{
    . . .
      newData = reqData;
      const mailMessage = config.mailMessage + userId;
      var mail = {
        from: config.mailSender,
        to: config.mailReceiver,
        subject: config.mailSubject,
        text: mailMessage,
        html: "<html><p>"+mailMessage.replaceAll('\n','<br>')+"</p></html>",
        attachments: [
          {
            filename: 'paciente'+userId+'.json',
            content: JSON.stringify(newData,null,2),
            contentType: 'application/json',
          },
          {
            filename: 'paciente'+userId+'.csv',
            content: userCsv,
            contentType: 'plain/text',
          },
          {
            filename: 'paciente'+userId+'_avisos.csv',
            content: warningsCsv,
            contentType: 'plain/text',
          },
        ],
      };
      const transporter = nodemailer.createTransport(config.smtpServer);
      transporter.verify(function(error,success){
        if(error!=undefined){
          throw error;
        }
      });
      await transporter.sendMail(mail);
      . . .
    }
        . . .
  }catch(error){
    if(resp.status == undefined){
      resp.status = 500;
    }
    resp.message = "ERROR: " + error;
    logger.error('ERROR FORWARDING USER DATA: '+resp.message);
  }
  res.type('json');
  res.status(resp.status).send(resp);
  return;
}
```

Código 5.20: *Fragmento de código de la función* `sendUserData()` *en el que se genera y se envía el correo electrónico con los datos de usuario.*

5.3.3.c. Envío de registros (OP02) y formulario de contacto (OP03)

El envío de registros se realiza de manera similar al envío de datos de usuario, desde una cuenta propia hasta otra cuenta propia, pero sin comprobar ningún código de paciente. Como se comentó en la Sección 4.4.2, la integridad de los registros generados por la aplicación no es crucial, ya que no se utilizan en el estudio clínico.

Por otra parte, la página web asociada a la aplicación PrepColon—necesaria para cumplir con las condiciones de Google y Apple para poder publicar la aplicación en sus tiendas oficiales—implementa el formulario de contacto que puede verse en la Figura 5.8, pensado para ofrecer soporte a los usuarios de la aplicación. Al pulsar «Enviar», el navegador web genera una petición

al servidor que ejecuta la operación `OP03` y provoca el envío de un correo electrónico con los datos introducidos por el usuario.

El envío de registros se lleva a cabo al ejecutarse la función `sendLogs()`, y el envío del mensaje del formulario de contacto se hace a través de la función `sendSupportMail()`. En ambos casos, se genera un correo electrónico utilizando la herramienta NodeMailer [Rei], de manera similar a como se muestra en el Código 5.20.

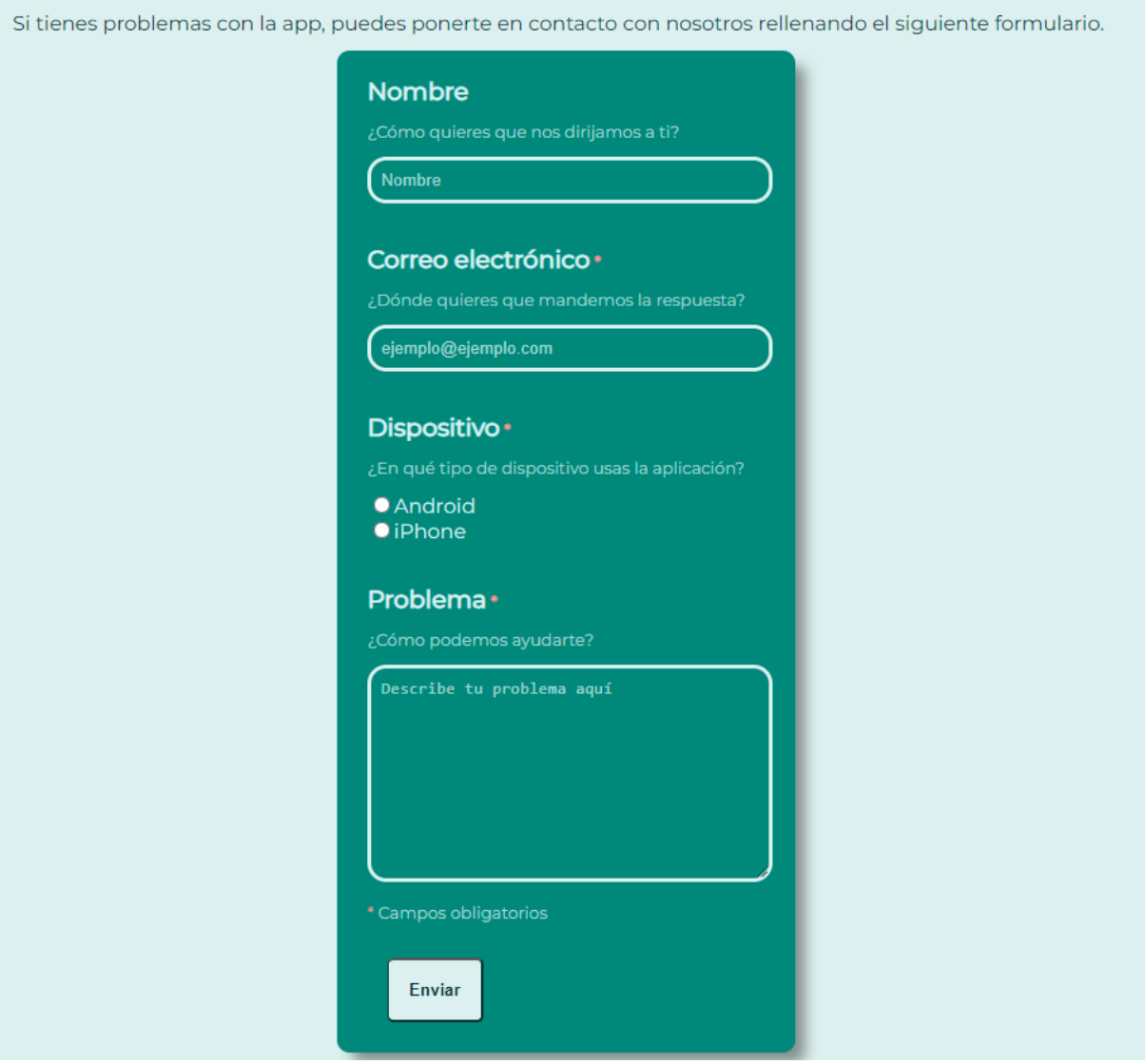


Fig. 5.8: *Formulario de contacto disponible en la página web asociada a la aplicación PrepColon [Prep].*

5.4. Conclusiones

En este capítulo se ha presentado la forma en la que el diseño de la aplicación PrepColon y su servidor, propuesto en el Capítulo 4, se ha implementado para cumplir con las funcionalidades previstas, mostrando los fragmentos de código que pudieran resultar más relevantes.

Respecto a la aplicación móvil, el uso de Flutter [Flutter], un *framework* multiplataforma,

ha facilitado el desarrollo de la aplicación PrepColon gracias a las ventajas que se presentaron en la Sección 5.2.1, aunque el hecho de ser *software* propietario implica que el futuro de la aplicación depende, en cierto modo, de las decisiones que tome Google. En este capítulo se ha mostrado cómo la aplicación valida el código de paciente asignado a los usuarios que decidan participar en el estudio clínico, cómo envía los datos de usuario al servidor para su reenvío al equipo médico y cómo está preparada para usarse en diferentes idiomas; todo ello gesitonado por unas clases controladoras encargadas de manejar los datos y los estados de la aplicación. También se ha presentado la manera en la que se ha implementado el patrón de diseño *State*, cuya importancia se discutió en la Sección 4.3.1.

En cuanto al servidor, este capítulo ha puesto el foco en la API RESTful definida en la Sección 4.4.1. Por un lado, se ha explicado cómo se ha usado el validador de Swagger [Smab], una herramienta muy útil a la hora de desarrollar APIs gracias a us interfaz interactiva que permite probar las operaciones soportadas por el servidor. Por otro lado, se ha mostrado cómo se han implementado las operaciones que conforman la API del servidor de PrepColon, descritas en las Tablas 4.9 y 4.8.

Volviendo a la metodología seguida, descrita en la Sección 1.2, una vez conseguida una aplicación funcional, la siguiente fase es evaluar su funcionamiento para garantizar que los usuarios finales se encuentren una versión estable y sin errores. Después, la aplicación estará lista para su uso, por parte de pacientes reales, en el estudio clínico que buscará determinar la eficacia de una aplicación de apoyo a la preparación de colonoscopias. En el siguiente capítulo se aboradarán ambas cuestiones, presentando las diferentes pruebas realizadas sobre la aplicación PrepColon antes del comienzo del estudio clínico y los resultados actuales del mismo. Cabe destacar que este documento se ha redactado de manera paralela a la realización del ensayo, y la defensa de este TFG se hará meses antes de su finalización, por lo que los datos que se mostrarán en el próximo capítulo no representarán los resultados reales.

Capítulo 6

Evaluación

En el Capítulo 1 se planteó la necesidad de la realización de colonoscopias para la detección precoz de CCR. En el Capítulo 2 se presentaron aplicaciones de apoyo a la preparación de colonoscopias y estudios que mostraban la eficacia de éstas. En el Capítulo 3 se analizaron los requisitos necesarios para una aplicación que pretenda ayudar a los usuarios a lograr mejores resultados en sus colonoscopias. En el Capítulo 4 se propuso un sistema distribuido que cumpliese con esos requisitos y, en el Capítulo 5, se explicó cómo se implementó dicho sistema. En este capítulo se hablará de las pruebas realizadas sobre la aplicación desarrollada y de los resultados obtenidos al ser utilizada por pacientes reales que debían realizarse una colonoscopia.

6.1. Introducción

A lo largo de los capítulos anteriores se ha descrito el proceso de desarrollo de una aplicación destinada al apoyo a la preparación de colonoscopias. El siguiente paso lógico, una vez obtenida una aplicación funcional, es probarla meticulosamente para detectar errores que, por un motivo u otro, pudieron pasarse por alto durante la implementación de la aplicación. Para ello, se han simulado diferentes usuarios con diferentes comportamientos para tratar de abarcar todas las posibles situaciones en las que podría usarse la aplicación.

Tras varios meses probando la aplicación, corrigiendo errores y mejorando aspectos de la misma, a principios del año 2025 el equipo médico decidió que la aplicación se encontraba en una versión lo suficientemente madura como para usarse con pacientes reales. En marzo de ese mismo año comenzó de manera oficial un estudio clínico con el objetivo de determinar la efectividad de la aplicación PrepColon a la hora de lograr que los usuarios acudiesen a sus citas médicas con una preparación adecuada. En la Sección 6.3 se comentarán los resultados obtenidos en el momento de la redacción de este documento. Sin embargo, a julio de 2025, aún quedan cuatro meses hasta la finalización del estudio clínico, por lo que los datos que se presentarán más adelante en este capítulo están incompletos.

6.2. Casos de prueba

Antes de sacar cualquier producto al mercado, conviene someterlo a diferentes pruebas para garantizar que se cumplen los requisitos con una calidad adecuada y que se satisfacen las necesidades de los usuarios. Y la aplicación PrepColon no es diferente en ese aspecto. Tras desarrollar una versión completamente funcional, tal y como pudo verse en la Figura 1.3, se inició una fase de evaluación en la que diferentes personas probaron la aplicación para tratar de encontrar errores que pudieron pasarse por alto en la fase de implementación o aspectos que pudieran pulirse para lograr una mejor experiencia de usuario.

Estas pruebas pueden dividirse en dos grupos. Por un lado, aquellas realizadas por el autor de este documento, en las que se trató de hacer un seguimiento más profundo para detectar más fácilmente las causas de los errores a nivel de código. Por otro lado, las pruebas realizadas por terceras personas, principalmente el cliente del proyecto y otro personal sanitario del Servicio de Aparato Digestivo de los hospitales participantes, más enfocadas a la evaluación de las funcionalidades principales y a la experiencia de usuario.

Estas pruebas pueden dividirse en dos grupos. Por un lado, aquellas realizadas con el objetivo de explotar todas las funcionalidades de la aplicación para hacer un seguimiento más profundo y detectar así más fácilmente las causas de los posibles errores a nivel de código. Por otro lado, pruebas más informales realizadas por distintos usuarios, principalmente personal sanitario del Servicio de Aparato Digestivo de los hospitales participantes, más enfocadas a la evaluación tanto de los aspectos médicos que trata la aplicación como de la experiencia de usuario.

6.2.1. Pruebas realizadas por distintos usuarios

El desarrollador de la aplicación PrepColon no ha sido el único que ha realizado pruebas, sino que ha podido contar con realimentación proporcionada por otras personas que han probado de manera voluntaria la aplicación exhaustivamente, en su mayoría miembros del equipo médico que ha colaborado en este proyecto u otros profesionales de los hospitales participantes, en dispositivos móviles de diferentes marcas y características. Los comentarios facilitados por estas personas estaban mayormente dirigidos al diseño de la interfaz de usuario de la aplicación, cuya evolución—motivada precisamente por estos comentarios—se pudo ver en la Sección 4.3.2. Sin embargo, estas pruebas también sirvieron para identificar elementos de la aplicación que era necesario modificar y que se listan a continuación.

- En la pantalla de selección horaria el usuario podía seleccionar cualquier hora para su cita. Se limitó el rango seleccionable al intervalo entre las 08:00 horas y las 20:30 horas, ya que, según el equipo médico, fuera de ese horario no se dan citas para colonoscopias. En la Figura 6.1 se muestra esta característica de la aplicación.
- El fichero CSV enviado por la aplicación mostraba campos vacíos y datos que no se correspondían con el campo en el que se encontraban. Esto se debía a que la función encargada de convertir los datos de formato JSON a formato CSV estaba mal implementada. En el `switch`, el caso `List<dynamic>` ejecutaba el mismo código que el caso `List<bool>`, por lo que las listas de cadenas de caracteres no se incluían correctamente en el fichero CSV.
- Aunque ya se habló de ello en la Sección 5.2.6, el sistema de validación de código de paciente se incluyó durante la fase de evaluación de la aplicación para garantizar que los

usuarios de la aplicación, al menos hasta que el estudio clínico finalizase, eran únicamente pacientes que participaban en el estudio de manera voluntaria. De lo contrario, existía el riesgo de que el equipo médico recibiese datos de usuarios no participantes en el estudio.

- Inicialmente el intervalo entre dos intentos consecutivos para enviar los datos al servidor era de dos horas. El equipo médico sugirió reducir este intervalo para obtener más rápidamente los datos recopilados por la aplicación, por lo que el intervalo se cambió a 30 minutos, como se muestra en el Código 5.11.

- Se ajustaron los recordatorios de las tomas del laxante adicional, Evacuol®, para que se mostrasen a las horas en las que el paciente debe tomarlo, entre las 19:00 horas y las 23:00 horas.

- Se detectó que, al cambiar la hora y la fecha de la cita, había avisos que no se generaban. Se solucionó incluyendo una llamada a la misma función que generaba todas las tareas al final del formulario, cuando el usuario confirmaba las nuevas fecha y hora.

- Se detectó que las funciones de la clase `DateTime` de Flutter [Flub] para calcular la diferencia en días entre dos fechas no eran exactas. Por ejemplo, comparar las fechas `2025-07-16 20:00:00` y `2025-07-17 08:00:00` daría como resultado que la diferencia en días es cero, porque en horas es menor de 24. Esto daba problemas cuando quedaban pocos días para la cita, ya que la aplicación podía mostrar el texto «Tu cita es hoy» el día antes de la cita, o «Tu cita es mañana» quedando dos días para la cita. Se solucionó implementando una función personalizada que realizase este cálculo.

- Se detectó que, cuando se pulsaba el botón «Borrar datos» para cancelar la preparación, la función mostrada en el Código 5.8 se llamaba dos veces desde diferentes puntos del código, lo que ocasionaba envíos duplicados. Se eliminó la llamada sobrante.

6.2.2. Pruebas modelando distintos tipos de pacientes

Si bien es cierto que las pruebas han estado presentes en todo el proceso de desarrollo de la aplicación PrepColon—cada vez que se implementaba una nueva funcionalidad, ésta era puesta a prueba para comprobar su correcto funcionamiento—, no fue hasta conseguir una versión completa cuando se probaron todas las características disponibles en conjunto. Para ello, se iniciaron varias preparaciones falsas en la aplicación, tratando de simular el comportamiento de pacientes reales. En cada una de estas preparaciones, el «paciente» mostraba una actitud diferente en cuanto a su interacción con la aplicación. A continuación se muestran algunas de estas pruebas, detallando las fechas en las que ocurrían los diferentes eventos y los mensajes generados por la aplicación.

Para el primer caso se simuló un paciente sano, es decir, que no requiriese de una preparación intensiva. Este «paciente» sigue correctamente las indicaciones de la aplicación, completando las tareas propuestas lo antes posible.

CASO I

Descripción	**Fecha de inicio**	**Fecha de la cita**	**Laxante**
Paciente sano que sigue las indicaciones de la app	20/09/2024 – 16:45	27/09/2024 – 16:00	Pleinvue

Evento	**Fecha**	**Mensaje**	**Comentarios**

Tarea pendiente	20/09/2024 – 16:45	¿Has recogido la preparación? Puedes hacerlo en tu Centro de Salud	Completada
Notificación	23/09/2024 – 16:30	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	24/09/2024 – 16:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	25/09/2024 – 04:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	25/09/2024 – 10:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	25/09/2024 – 12:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	25/09/2024 – 16:30	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	26/09/2024 – 16:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	27/09/2024 – 04:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	27/09/2024 – 09:20	¿Has tomado la dosis de Pleinvue?	Completada
Notificación	27/09/2024 – 09:20	Tienes tareas pendientes	—
Tarea pendiente	27/09/2024 – 09:50	Completa esta tarea la próxima vez que vayas a defecar para indicar su aspecto	Completada (aspecto líquido)
Notificación	27/09/2024 – 10:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	27/09/2024 – 11:20	¿Has tomado la dosis de Pleinvue?	Completada

Notificación	27/09/2024 – 12:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	27/09/2024 – 14:30	¿Estás siguiendo la dieta? Recuerda que no puedes comer ni beber nada.	No es una tarea que se pueda completar
Notificación	27/09/2024 – 14:30	Tienes tareas pendientes	—
Tarea pendiente	27/09/2024 – 14:30	Completa esta tarea la próxima vez que vayas a defecar para indicar su aspecto	Completada (aspecto líquido)
Errores detectados	El primer aviso de dieta líquida se muestra con 24 horas de antelación.		

Para el segundo caso el perfil de paciente cambia ligeramente. Este segundo «paciente» también sigue correctamente las indicaciones dadas por la aplicación, pero en este caso ya no se trata de un paciente sano y, por lo tanto, deberá realizar una preparación intensiva.

CASO II

Descripción	**Fecha de inicio**	**Fecha de la cita**	**Laxante**
Paciente con problemas de salud que sigue las indicaciones de la app	29/09/2024 – 12:40	04/10/2024 – 12:00	Bohn

Evento	**Fecha**	**Mensaje**	**Comentarios**
PopUp	29/09/2024 – 12:40	¿Has comprado Evacuol?	Sí
Tarea pendiente	29/09/2024 – 12:40	Recuerda suspender tu tratamiento de hierro.	No es una tarea que se pueda completar
Tarea pendiente	29/09/2024 – 12:40	Recuerda contactar con tu Médico de Atención Primaria para valorar la suspensión o sustitución de Anticoagulantes Orales.	Completada
Tarea pendiente	29/09/2024 – 12:40	¿Has recogido la preparación? Puedes hacerlo en tu Centro de Salud.	Completada
Notificación	29/09/2024 – 13:10	¿Has suspendido el tratamiento de hierro?	No es una tarea que se pueda completar
Tarea pendiente	29/09/2024 – 20:00	Recuerda tomar entre 6 y 8 gotas de Evacuol esta noche.	Completada
Notificación	30/09/2024 – 12:30	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	01/10/2024 – 12:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar

Notificación	01/10/2024 – 12:45	¿Has suspendido el tratamiento del hierro?	No es una tarea que se pueda completar
Notificación	02/10/2024 – 00:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	02/10/2024 – 06:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	02/10/2024 – 10:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	03/10/2024 – 12:30	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	03/10/2024 – 12:45	¿Has suspendido el tratamiento del hierro?	No es una tarea que se pueda completar
Notificación	03/10/2024 – 19:20	¿Has tomado la dosis de Bohn?	Completada
Notificación	03/10/2024 – 19:30	Tienes tareas pendientes	—
Tarea pendiente	03/10/2024 – 19:50	Completa esta tarea la próxima vez que vayas a defecar para indicar su aspecto	Completada (aspecto líquido)
Notificación	04/10/2024 – 00:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	04/10/2024 – 00:45	¿Has suspendido el tratamiento del hierro?	No es una tarea que se pueda completar
Notificación	04/10/2024 – 06:30	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	04/10/2024 – 07:20	¿Has tomado la dosis de Bohn?	Completada
Notificación	04/10/2024 – 07:30	¿Has suspendido el tratamiento del hierro?	No es una tarea que se pueda completar
Notificación	04/10/2024 – 08:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	04/10/2024 – 08:45	¿Has suspendido el tratamiento del hierro?	No es una tarea que se pueda completar

Notificación	04/10/2024 – 09:00	Tienes tareas pendientes	—
Tarea pendiente	04/10/2024 – 09:15	Completa esta tarea la próxima vez que vayas a defecar para indicar su aspecto	Completada (aspecto líquido)
Notificación	04/10/2024 – 10:30	¿Estás siguiendo la dieta? Recuerda que no puedes comer ni beber nada.	No es una tarea que se pueda completar
Notificación	04/10/2024 – 10:40	¿Has suspendido el tratamiento del hierro?	No es una tarea que se pueda completar
Errores detectados		Ninguno.	

El tercer «paciente» de estas pruebas no requiere de una preparación intensiva. Pero al contrario del primero, en este caso se ignorarán los avisos de la aplicación.

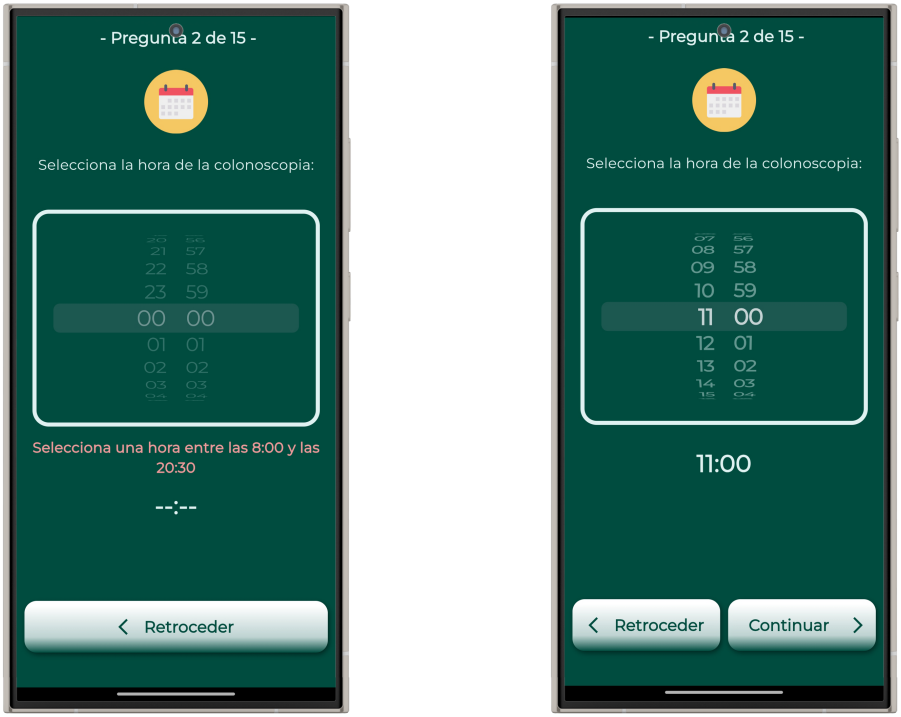
CASO III

Descripción	**Fecha de inicio**	**Fecha de la cita**	**Laxante**
Paciente sano que ignora la app	12/10/2024 – 15:15	18/10/2024 – 18:00	Moviprep

Evento	**Fecha**	**Mensaje**	**Comentarios**
Notificación	12/10/2024 – 15:45	¿Has recogido la preparación en el Centro de Salud?	Ignorada
Notificación	14/10/2024 – 15:15	¿Has recogido la preparación en el Centro de Salud?	Ignorarda
Notificación	14/10/2024 – 18:30	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	15/10/2024 – 03:45	¿Has recogido la preparación en el Centro de Salud?	Ignorada
Notificación	15/10/2024 – 09:15	¿Has recogido la preparación en el Centro de Salud?	Ignorada
Notificación	15/10/2024 – 11:15	¿Has recogido la preparación en el Centro de Salud?	Ignorada
Notificación	15/10/2024 – 13:15	¿Has recogido la preparación en el Centro de Salud?	Ignorada
Notificación	15/10/2024 – 15:15	¿Has recogido la preparación en el Centro de Salud?	Ignorada
Notificación	15/10/2024 – 17:15	¿Has recogido la preparación en el Centro de Salud?	Ignorada

Notificación	15/10/2024 – 18:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	16/10/2024 – 06:30	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	16/10/2024 – 12:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	16/10/2024 – 14:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	16/10/2024 – 16:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	16/10/2024 – 12:00	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	No es una tarea que se pueda completar
Notificación	17/10/2024 – 18:30	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	18/10/2024 – 06:30	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	18/10/2024 – 12:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	18/10/2024 – 14:00	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	18/10/2024 – 16:30	¿Estás siguiendo la dieta? Recuerda que no puedes comer ni beber nada.	No es una tarea que se pueda completar
Errores detectados		Ninguno.	

En la cuarta prueba se simuló un paciente sano, sin requerimientos de preparación intensiva, que fue realizando la preparación por su cuenta siguiendo las indicaciones proporcionadas por su médico a través de un folleto informativo. El día antes de la colonoscopia, recuerda la existencia de la aplicación PrepColon y decide utilizarla.



(a) **(b)**

Fig. 6.1: *Pregunta del formulario de hora de la cita en los casos (a) que se introduzca una hora fuera del horario de citas y (b) que se introduzca una hora válida.*

CASO IV

Descripción	**Fecha de inicio**	**Fecha de la cita**	**Laxante**
Paciente sano que comienza a utilizar la app tarde	20/10/2024 – 19:30	21/10/2024 – 20:00	Moviprep

Evento	**Fecha**	**Mensaje**	**Comentarios**
PopUp	20/10/2024 – 19:30	¿Estás siguiendo la dieta? Recuerda que hay alimentos que no puedes tomar.	Sí
PopUp	20/10/2024 – 19:30	¿Has recogido la preparación en el Centro de Salud?	Sí
Notificación	21/10/2024 – 08:15	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	21/10/2024 – 08:15	¿Has tomado la dosis de Moviprep?	Completada
Notificación	21/10/2024 – 08:25	Tienes tareas pendientes	—

Tarea pendiente	21/10/2024 – 08:50	Completa esta tarea la próxima vez que vayas a defecar para indicar su aspecto.	Completada (aspecto líquido)
Notificación	21/10/2024 – 13:20	¿Estás siguendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	21/10/2024 – 15:20	¿Has tomado la dosis de Moviprep?	Completada
Notificación	21/10/2024 – 15:25	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	21/10/2024 – 17:25	¿Estás siguiendo la dieta? Recuerda que no puedes tomar alimentos sólidos ni lácteos.	No es una tarea que se pueda completar
Notificación	21/10/2024 – 17:30	Tienes tareas pendientes	—
Tarea pendiente	21/10/2024 – 17:35	Completa esta tarea la próxima vez que vayas a defecar para indicar su aspecto.	Completada (aspecto líquido)
Notificación	21/10/2024 – 18:30	¿Estás siguiendo la dieta? Recuerda que no puedes comer ni beber nada.	No es una tarea que se pueda completar
Errores detectados		Ninguno.	

Como puede verse, en estas pruebas no se detectaron apenas errores, a excepción del hecho de que el primer aviso de la dieta líquida se mostró en el primer caso con 24 horas de antelación. Este error estaba relacionado con el problema de la clase `DateTime` de Flutter [Flub] que se comentó en la Sección 6.2.1. También se puede observar en el segundo caso cómo, si las características médicas de un paciente pueden repercutir negativamente en el resultado de la colonoscopia—por ejemplo, sobrepeso, intervenciones quirúrgicas previas en la zona abdominal o uso de fármacos para la presión arterial—, la aplicación propone una serie de pasos adicionales durante la preparación, adaptándose así a las necesidades individuales de cada usuario.

En el tercer caso se aprecia la «insistencia» de la aplicación ante usuarios que ignoren los avisos. Aunque aquí se ha presentado un caso extremo en el que se han ignorado todos los avisos y recordatorios, el uso de frecuencias de notificaciones incrementales puede ser útil para que aquellos usuarios más olvidadizos no se salten pasos durante su preparación. Además, en el cuarto caso se puede ver que la aplicación está preparada para comenzar a guiar al usuario desde cualquier punto de la preparación, incluso si éste olvida utilizarla al comienzo del proceso. En este caso, y siguiendo la información de la Tabla 4.2, se puede observar que la aplicación «salta» del estado `A.II` al estado `A.V`, logrando así una mayor flexibilidad del sistema.

6.3. Estudio clínico

El estudio clínico destinado a comprobar la efectividad de la aplicación PrepColon en su cometido a la hora de apoyar a los pacientes durante la preparación para sus colonoscopias, enmarcado dentro del proyecto de investigación «Validación e implementación de una aplicación móvil para la limpieza colónica personalizada y su repercusión en la calidad de la colonoscopia de cribado de cáncer colo-rectal» y aprobado por el Comité de Ética de la Investigación con Medicamentos (CEIm) de las Áreas de Salud de Valladolid, se realizó de manera paralela a la redacción de este documento. Es por esto que los datos a los que ha tenido acceso el autor del mismo, proporcionados por el equipo médico, son limitados y están, en algunos casos, incompletos. Sin embargo, se intentará en esta sección analizar y comentar estos resultados.

Dichos datos incluyen a 184 pacientes, de los cuales 61 recibieron instrucciones de utilizar la aplicación PrepColon y 59 pertenecen al grupo de control que recibió las indicaciones habituales—explicaciones orales por parte del médico y un folleto informativo con las indicaciones necesarias para realizar la preparación. De los 64 pacientes restantes no se indica a qué grupo pertenecen. De los 120 pacientes que se incluyeron en alguno de los dos grupos—el grupo «app» y el grupo de control—, solo estaban disponibles resultados de colonoscopias de ocho de ellos. Por otra parte, de los 61 pacientes que debían utilizar la aplicación, solo 32 habían rellenado el cuestionario de Escala de Usabilidad de Sistema (SUS) [Bro95; Gri+13] en mayo de 2025—fecha en la que el equipo médico compartió estos datos. Como comentario adicional, se sabe que, dentro del grupo «app», al menos siete pacientes utilizaban un dispositivo Android y al menos dos de ellos utilizaban un dispositivo con sistema operativo iOS.

Número total de pacientes	**Grupo «app»**	**Grupo de control**	**Sin determinar**
184	61	59	64

Número de resultados de colonoscopias	**Números de resultados de SUS**
8	32

Tabla 6.1: *Número de pacientes y de resultados disponibles en los datos proporcionados por el equipo médico en mayo de 2025.*

6.3.1. Resultados de colonoscopias

Como se explicó en la Sección 2.1, una manera común de medir los resultados de las colonoscopias es siguiendo la Escala de Preparación Intestinal Boston [Lai+09], donde se asigna una puntuación entre 0 y 3 a cada segmento del colon—colon derecho, colon transversal y colon izquierdo—en función del nivel de limpieza presente en cada uno de ellos, siendo el resultado final la suma de sus valores. Además, en estudios similares, se definía una preparación como adecuada si se obtenían al menos dos puntos en cada segmento del colon y una suma total de, por lo menos, seis puntos; y una preparación excelente en aquellos casos en los que la puntuación BBPS era de ocho puntos o más [Wal+21; Zan+21; Zhu+23].

En la Tabla 6.2 se muestran los resultados de las colonoscopias proporcionados por el equipo médico: la puntuación individual—entre 0 y 3—de cada uno de los tres segmentos en los que se divide el colon—colon derecho, colon izquierdo y colon transverso—, la puntuación total de la Escala Boston o BBPS—resultado de sumar las tres anteriores—y el número de pólipos encontrados. Aunque no son demasiados valores, se puede ver que en todos ellos la preparación

ha sido adecuada y, exceptuando a uno de los pacientes, también ha sido excelente.

Grupo	Derecho	Transverso	Izquierdo	BBPS	# Pólipos
App	3	3	2	8	1
	2	3	3	8	2
	2	2	2	6	1
	2	3	3	8	1
	3	3	3	9	1
	2	3	3	8	1
No App	3	3	3	9	0
	3	3	3	9	0

Tabla 6.2: *Resultados de las colonoscopias a los que ha podido acceder el autor de este documento, proporcionados por el equipo médico: puntuación individual de los segmentos del colon, puntuación de la Escala Boston y número de pólipos encontrados.*

6.3.2. Escala de usabilidad

En el estudio clínico no solo se miden los resultados de las colonoscopias, sino que también se evalúa la percepción que tenían los pacientes de la aplicación PrepColon. Esto se hace mediante el cálculo de la Escala de Usabilidad de Sistema (SUS)[Bro95; Gri+13], un cuestionario estandarizado de 10 afirmaciones en las que el usuario debe responder con una puntuación entre 1—totalmente en desacuerdo con la afirmación—y 5—totalmente de acuerdo. El resultado se calcula como se muestra en (6.1) y (6.2), donde r_i es la respuesta dada por el usuario a la pregunta i-ésima y v_i es el valor de la respuesta i-ésima que computa para el cálculo del SUS de ese cuestionario[Bro95].

$$v_i = \begin{cases} r_i - 1 & \text{para } i = 1,3,5,7,9 \\ 5 - r_i & \text{para } i = 2,4,6,8,10 \end{cases} \tag{6.1}$$

$$SUS = 2{,}5 * \sum_{i=1}^{10} v_i \tag{6.2}$$

Para este estudio, las preguntas formuladas se pueden ver en la Tabla 6.3, y las respuestas obtenidas a las que ha podido acceder el autor de este TFG se muestran en la Tabla 6.4.

Número	Pregunta
1	Me gustaría usar esta APP con frecuencia
2	Me pareció la APP compleja
3	Pensé que la APP era fácil de usar
4	Necesito el apoyo de un técnico para poder utilizar esta APP
5	Las diversas funciones de esta APP estaban bien integradas
6	Hay demasiada inconsistencia en esta APP
7	La mayoría de las personas aprenderían a usar esta APP muy rápidamente
8	La APP es muy incómoda de usar
9	Me sentí muy seguro/a usando la APP
10	Necesité aprender muchas cosas antes de poder ponerme en marcha con esta APP

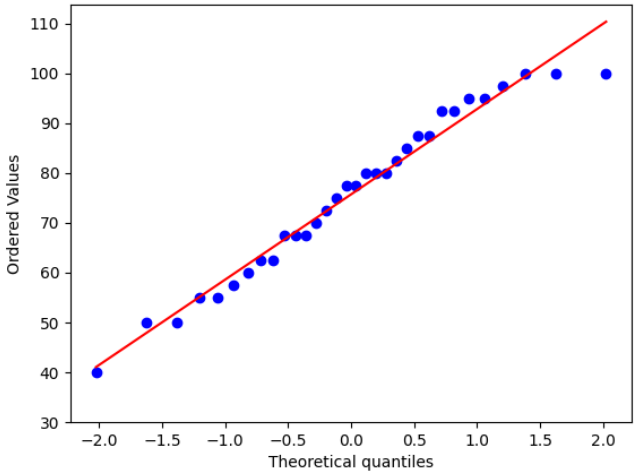
Tabla 6.3: *Formulario utilizado por el equipo médico para calcular el SUS de la aplicación PrepColon.*

# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10	SUS
3	5	5	1	5	1	3	1	2	3	67.5
1	1	3	1	1	1	3	1	1	1	60.0
3	1	3	1	2	2	5	1	3	2	72.5
5	1	5	1	5	1	5	1	5	1	100.0
3	1	2	1	4	3	5	1	5	1	80.0
5	1	5	1	5	1	3	1	5	1	95.0
4	1	5	1	5	4	5	1	3	1	85.0
4	2	5	1	5	4	4	1	3	1	80.0
3	1	3	1	1	1	3	5	1	1	55.0
5	2	5	1	2	3	4	2	3	1	75.0
3	2	3	1	3	4	4	2	2	1	62.5
5	5	3	5	5	1	5	5	5	4	57.5
3	2	4	1	4	1	2	1	5	1	80.0
3	4	5	1	3	3	5	2	2	1	67.5
1	3	5	1	3	3	5	3	2	1	62.5
5	1	5	1	3	2	3	1	5	1	87.5
5	1	5	1	5	1	5	1	5	1	100.0
4	1	5	1	1	1	5	1	3	1	82.5
1	1	5	2	5	1	5	1	5	1	87.5
5	3	4	4	5	4	5	1	5	1	77.5
4	3	3	1	3	3	2	3	2	4	50.0
1	3	3	1	3	3	1	1	1	5	40.0
4	1	5	1	5	1	4	1	5	1	95.0
5	1	1	1	5	4	3	1	5	1	77.5
5	1	4	1	4	2	5	1	5	1	92.5
4	2	3	1	3	3	5	2	3	2	70.0
4	3	2	4	3	5	5	1	3	4	50.0
5	1	5	1	5	1	5	1	5	1	100.0
4	4	3	1	5	3	3	1	3	2	67.5
3	3	3	2	3	3	3	2	3	3	55.0
3	1	4	1	5	1	5	1	5	1	92.5
4	1	5	1	5	1	5	1	5	1	97.5

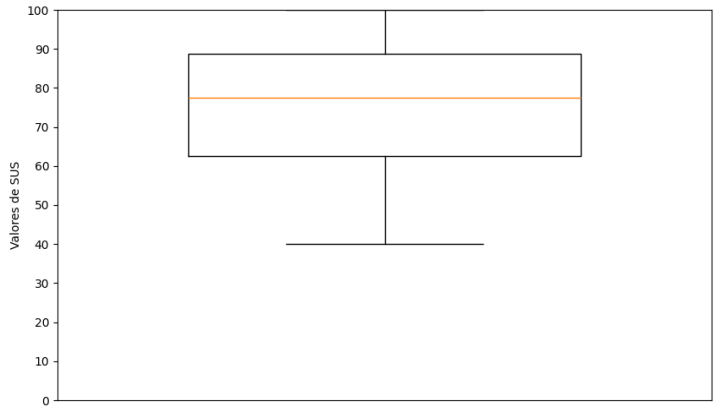
Tabla 6.4: *Resultados del SUS a los que ha tenido acceso el autor de este documento, proporcionados por el equipo médico.*

6.3.2.a. Análisis estadístico

Dado que los resultados de la Tabla 6.4 no conforman el total de los resultados del SUS que habrá al final del estudio clínico, se pueden considerar esos valores como una muestra, un subconjunto de la población total—en este caso la población total serían todos los resultados del SUS obtenidos por todos los pacientes que participen en total en el estudio. Según [Nav10], las muestras de tamaño $n > 30$ se pueden considerar lo suficientemente grandes como para cumplir el Teorema del Límite Central [Nav10] y asumir que siguen una distribución aproximadamente normal. En este caso, la muestra tiene un tamaño $n = 32$, por lo que se puede hacer esa suposición sobre los datos disponibles. Como se puede comprobar en la Figura 6.2a, ya que las muestras se aproximan a la recta—aunque aparecen pequeños escalones que representan valores repetidos—, y en la Figura 6.2b, al no haber una asimetría demasiado marcada, la suposición de que las muestras disponibles siguen una distribución aproximadamente normal o gaussiana no parece alejarse mucho de la realidad.



(a)



(b)

Fig. 6.2: *Análisis del tipo de distribución de las muestras disponibles [Nav10]: (a) gráfica cuantil-cuantil y (b) gráfica de caja o boxplot.*

Puesto que la distribución de la muestra disponible puede aproximarse a una distribución

normal o gaussiana, se puede calcular un intervalo de confianza aproximando la desviación típica total a la desviación típica muestral disponible. El resultado se puede observar en la Tabla 6.5 y, de manera gráfica, en la Figura 6.3. Gracias a esto se puede estimar que la media total estará, con una probabilidad del 95 %, entre 69.92 y 81.49. Según la Tabla 6.6, podría decirse que, actualmente, la aplicación PrepColon tiene una usabilidad buena. Además, puede preverse que la media del SUS al final del estudio se encuentre en el rango marcado por el interalo de confianza, lo que, según la Tabla 6.6, significa una usabilidad aceptable o buena. Aunque son resultados mejorables, cabe recordar que el estudio clínico se está realizando en pacientes de entre 50 y 69 años, un rango de edad en el que es común encontrar personas con habilidades limitadas a la hora de interactuar con la tecnología, por lo que probablemente el resultado del SUS mejorase si el rango de edades fuese más amplio.

Media	Mediana	Moda	Desviación típica muestral	Intervalo de confianza[1]
75.70	77.50	67.50	16.70 59.00 – 92.41	5.78 69.92 – 81.49

Tabla 6.5: *Análisis estadístico de los resultados del SUS.*

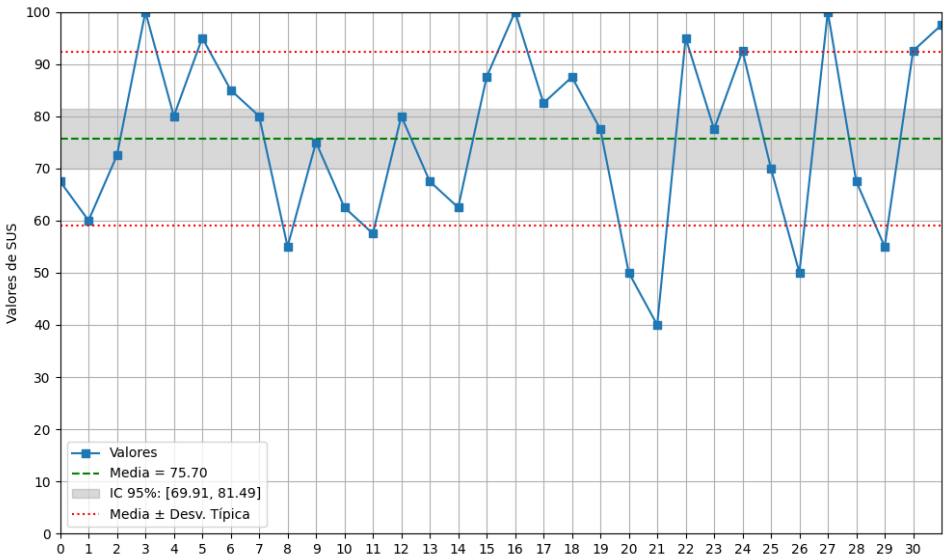


Fig. 6.3: *Representación gráfica del análisis estadístico de los resultados del SUS de la aplicación PrepColon: superposición de valores estadísticos.*

Adjetivo	Rango de valores medios de SUS
Óptimo	100.00
Excelente	Entre 100.00 y 85.58
Bueno	Entre 85.58 y 72.75
Aceptable	Entre 72.75 y 52.01
Pobre	Entre 52.01 y 39.17
Malo	Entre 39.17 y 25.00
Pésimo	Menos de 25.00

Tabla 6.6: *Rango de calificativos que se pueden asignar a un sistema en función del valor medio del SUS obtenido [BKa08; BKM09].*

[1] Intervalo al 95 %

6.4. Conclusiones

Tras desarrollar una aplicación funcional, en este capítulo se ha mostrado cómo se ha probado la aplicación PrepColon para comprobar que su funcionamiento era el correcto y el esperado, gracias no solo a la dedicación del autor de este documento, sino también al esfuerzo y tiempo invertido por personas ajenas al proyecto que han podido aportar otro punto de vista.

También en este capítulo se han analizado los limitados resultados que ha podido proporcionar el equipo médico relativos al estudio clínico en el que se comprueba la efectividad de la aplicación PrepColon en su labor como apoyo para los pacientes que necesiten someterse a colonoscopias. Por un lado, se han comentado los resultados de la Escala de Preparación Intestinal Boston [Lai+09], relativos a las colonoscopias en sí. Sin embargo, estos datos son demasiado pocos como para sacar conclusiones.

Por otro lado, se han podido analizar resultados de la Escala de Usabilidad de Sistemas [Gri+13; Bro95] o SUS, por sus siglas en inglés. En este aspecto se ha podido hacer un estudio más profundo de los datos proporcionados, obteniendo resultados esperanzadores. Se ha tomado el conjunto de los valores disponibles como una muestra del total que se espera conseguir una vez finalizado el estudio clínico y se ha demostrado mediante el uso de gráficas cuantil-cuantil y *boxplot* [Nav10] que su distribución puede aproximarse a una distribución normal o gaussiana. De esta manera, y de acuerdo a [BKa08; BKM09], se prevé que la usabilidad de la aplicación PrepColon acabará siendo aceptable o buena—aunque con los datos actuales la usabilidad puede considerarse buena.

Capítulo 7

Conclusiones

Para finalizar este TFG se hablará, en este último capítulo, de cómo se relacionan los objetivos que se propusieron al principio con la aplicación PrepColon cuyo desarrollo se ha mostrado en este documento. Sin embargo, mirando el proyecto con retrospectiva, surgen algunas ideas que no se pudieron realizar por falta de recursos, tiempo o conocimientos; pero que podrían ser interesantes para llevarse a cabo en un futuro. Estas ideas se comentarán también en este capítulo.

7.1. Conclusiones del proyecto

El cáncer de colon es el tercer tipo de cáncer con mayor incidencia y el segundo con mayor tasa de mortalidad en España. Para reducir su impacto, se realiza un cribado periódico en personas de entre 50 y 69 años en el que se trata de identificar restos de sangre en muestras de heces. Si se encuentran, se cita al paciente para realizarse una colonoscopia con el objetivo de detectar pólipos y pequeñas heridas que puedan haberse causado por el cáncer. Sin embargo, los pacientes que no acuden con una limpieza adecuada o que no se presentan a sus colonoscopias dificultan esta tarea.

Existen estudios extranjeros en los que se probaron aplicaciones que sirviesen de apoyo a los usuarios durante la preparación de colonoscopias, dando resultados positivos al encontrar que aquellos pacientes que utilizaban la aplicación conseguían, de media, mejores resultados en sus colonoscopias. No obstante, las limitaciones en cuanto al uso de dichas aplicaciones en España hizo que un equipo de médicos profesionales de diferentes hospitales de Castilla y León propusieran de forma conjunta un estudio de características similares en el que se probase la efectividad de una aplicación española para ayudar a los pacientes en el proceso de preparación para sus colonoscopias.

En este TFG se ha propuesto PrepColon, una aplicación móvil multiplataforma desarrollada con Flutter diseñada para cumplir una serie de requisitos definidos por el equipo médico. El proyecto, llevado a cabo a lo largo de más de un año en el que se ha seguido una metodología ágil, con reuniones periódicas con el equipo médico y con los tutores de este TFG, buscaba

obtener una aplicación que se adaptase a las necesidades individuales de cada usuario para ofrecerle indicaciones personalizadas que permitiesen lograr una limpieza de colon adecuada. La aplicación implementa el patrón de diseño *State* [Gam+95] para controlar en qué etapa del proceso de preparación para colonoscopias se encuentra en cada momento y generar los avisos adecuados, que se recuerdan al usuario a través de notificaciones programadas cuya frecuencia aumenta con el tiempo para «insistir» al paciente para que complete las tareas propuestas. Además, también ha sido necesario implementar un servidor con una API propia, usando la especificación OpenAPI [Fouc], que permitiese al equipo médico recibir, durante el estudio clínico, datos generados por la aplicación sobre la preparación realizada por el paciente, de forma que pudieran analizar qué pasos han seguido los usuarios durante la preparación.

Este documento se ha redactado de manera paralela a la ejecución del estudio clínico, y su autor ha podido acceder a unos pocos resultados tanto de las colonoscopias realizadas como de los cuestionarios de la Escala de Usabilidad de Sistema que han tenido que rellenar los participantes del estudio que tuvieron que utilizar la aplicación. En el momento de la escritura de este TFG quedan varios meses hasta que finalice el estudio clínico, por lo que los datos proporcionados por el equipo médico son, de momento, demasiado escasos como para sacar conclusiones. No obstante, un pequeño análisis realizado en este documento indica que los resultados son prometedores y se prevé que el estudio clínico concluya favorablemente. Sin embargo, hasta finales de este año 2025 no se podrá determinar si el uso de la aplicación PrepColon supone una ventaja frente a las indicaciones habituales dadas de manera escrita y oral.

Actualmente la aplicación PrepColon se encuentra disponible de manera pública en Google Play Store. Según las métricas ofrecidas por Google, disponibles en la Tabla 7.1 y en la Figura 7.1, durante el estudio clínico—es decir, desde marzo de 2025—, la aplicación se descargó 92 veces y tuvo una media de entre cuatro y cinco usuarios activos diarios—aunque, como se muestra en la Figura 7.1b, Google no proporciona datos de usuarios activos diarios anteriores al 22 de abril de 2025. Además, de media, la aplicación PrepColon ha contado con 49 usuarios en total, siendo 70 el número de usuarios con la aplicación descargada a fecha 4 de julio de 2025.

Descargas totales	**Descargas diarias (media)**	**Usuarios activos diarios (media)**	**Usuarios totales (media)**
92	1.6	4.6	49.0

Tabla 7.1: *Estadísticas de la aplicación PrepColon entre marzo y julio de 2025, proporcionadas por Google.*

7.2. Ideas de mejora

Aunque con este proyecto se ha logrado desarrollar una aplicación completa y funcional, haciendo una reflexión sobre el cómo se hizo lleva a pensar en nuevas propuestas que, aunque en su momento no se llevaron a cabo porque no había tiempo suficiente para ello o porque el desarrollador de la aplicación no tenía los conocimientos necesarios, podrían implementarse en futuras versiones si se decidiese dar continuidad a la aplicación.

La idea quizás más llamativa es la incorporación de inteligencia artificial en la aplicación PrepColon. Ya se observó en la Sección 2.2 que es posible utilizar sistemas de inteligencia artificial predictiva para evaluar si el paciente está llevando una adecuada preparación a través

del aspecto de sus heces. Según [Zhu+23], se tardaron cuatro meses en generar una base de datos completa con imágenes tomadas por pacientes que estuviesen preparándose para una colonoscopia, y fueron necesarias 350 «épocas» de entrenamiento [Ver] hasta conseguir una precisión de más del 95 % en su modelo.

Otra propuesta para una posible futura versión de PrepColon, buscando una aplicación que se adapte aún más al usuario, sería incluir en el formulario preguntas relacionadas con su estilo de vida, como por ejemplo, a qué hora se suele levantar o a qué hora suele comer. De esta forma, los recordatorios y notificaciones mostrados por la aplicación podrían resultar más efectivos si, por ejemplo, salta una notificación recordando seguir la dieta cuando el usuario esté pensando qué cenar.

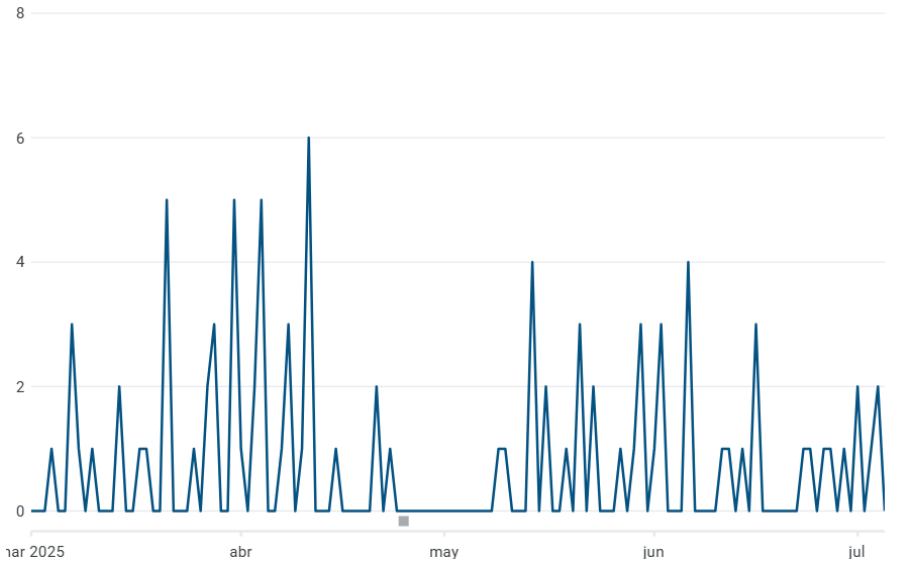
Una tercera idea es modificar la forma en la que PrepColon almacena la información. Actualmente, los datos de usuarios se almacenan en ficheros en la memoria interna del dispositivo. En el momento de su desarrollo, el autor de este documento no estaba familiarizado con el concepto, por eso esta propuesta ni siquiera se llegó a plantear. Sin embargo, dado que los dispositivos Android e iOS pueden almacenar datos persistentemente en bases de datos [Devd; Den], y puesto que Flutter cuenta con un paquete específico para ello [Flui], esta podría ser una alternativa interesante. Utilizar bases de datos no sólo eliminaría la necesidad de clases y métodos para manejar ficheros sino que también facilitaría la exportación de los datos de las tablas de la base de datos a un formato CSV.

7.3. Uso posterior al estudio clínico

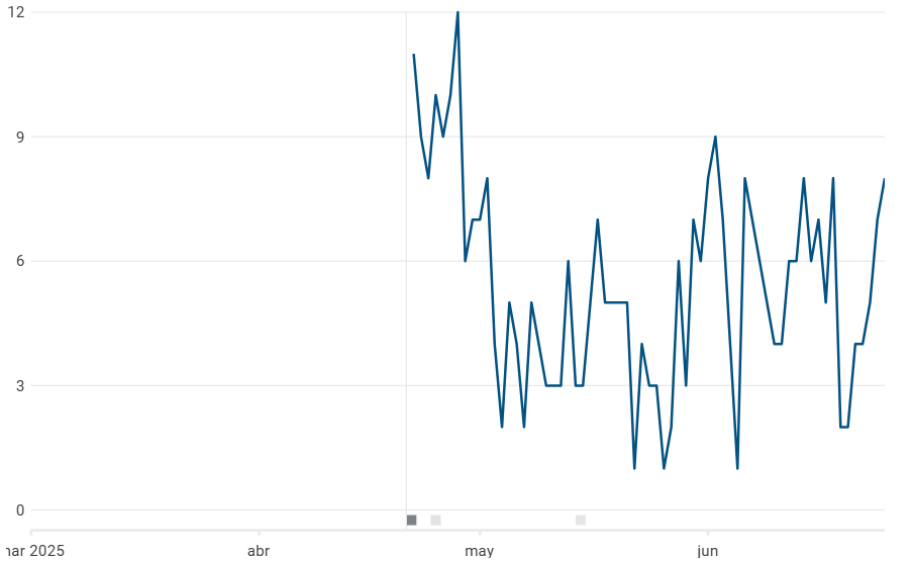
Actualmente el uso de la aplicación PrepColon está restringido únicamente a los participantes del estudio clínico perteneciente al proyecto de investigación que se está llevando a cabo de manera conjunta en el Hospital Universitario Río Hortega, el Hospital Virgen de la Concha y el Hospital de Medina del Campo. Esta decisión se tomó con el fin de evitar mezclar los datos de los pacientes participantes en el estudio con datos de pacientes no participantes, y por ello los usuarios que deseen hacer uso de la aplicación deben iniciar sesión con un código único. No obstante, el futuro de la aplicación depende de las conclusiones finales que saque el equipo médico tras la finalización del estudio.

Si los resultados son favorables y se demuestra que la aplicación PrepColon es útil para ayudar a los pacientes a lograr una limpieza adecuada, sería sencillo modificarla para eliminar la restricción, de forma que cualquier persona que se descargase la aplicación pudiese utilizarla sin necesidad de ningún identificador. Para ello, bastaría con quitar del código el *Widget* correspondiente a la pantalla de solicitud del código, del que se habló en la Sección 5.2.6. De esta forma, PrepColon podría ser utilizada por cualquiera que así lo quisiese, tanto personas individuales que decidan usarla de manera independiente como hospitales o centros de salud que busquen complementar las indicaciones tradicionales que dan a sus pacientes.

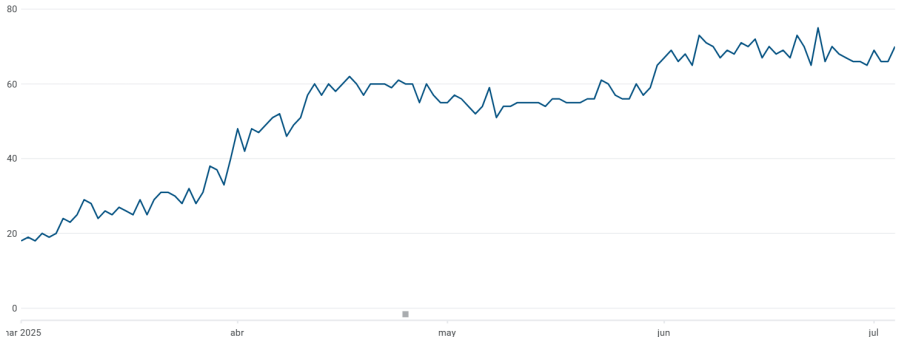
Otra alternativa sería mantener el sistema de validación de códigos y tratar de monetizar la aplicación limitando el acceso a la aplicación a los pacientes de aquellos hospitales o centros de salud que paguen por tener códigos de paciente propios. Sin embargo, hay quienes podrían ver esta opción como poco ética, al buscar aprovecharse de una necesidad médica—como es una colonoscopia—para conseguir una rentabilidad económica. No obstante, esta propuesta se menciona simplemente como una posibilidad hipotética, por lo que la discusión sobre sus implicaciones éticas o sociales no es objeto de este Trabajo de Fin de Grado.



(a)



(b)



(c)

Fig. 7.1: *Gráficas estadísticas de la aplicación PrepColon entre marzo y julio de 2025, proporcionadas por Google: (a) descargas, (b) usuarios activos diarios y (c) usuarios totales. Google no proporciona estadísticas para el número de usuarios activos diarios anteriores al 22 de abril de 2025.*

Referencias

[Bec+01] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Adrew Hunt, Ron Jeffiries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas. *Manifesto for Agile Software Development*. 2001. URL: `http://agilemanifesto.org/iso/en/manifesto.html` (Último acceso, 31 de marzo de 2025).

[BFM05] T. Berners-Lee, R.T. Fielding y L.M. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Enero de 2005. URL: `https://datatracker.ietf.org/doc/html/rfc3986` (Último acceso, 18 de mayo de 2025).

[BKa08] Aaron Bangor, Philip T. Kortum y James T. Miller and. «An Empirical Evaluation of the System Usability Scale». En: *International Journal of Human–Computer Interaction* 24.6 (2008), págs. 574-594. DOI: 10.1080/10447310802205776. eprint: `https://doi.org/10.1080/10447310802205776`. URL: `https://doi.org/10.1080/10447310802205776`.

[BKM09] Aaron Bangor, Phil Kortum y James Miller. «Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale». En: *J. Usability Stud.* 4 (abril de 2009), págs. 114-123.

[Bro95] John Brooke. «SUS: A quick and dirty usability scale». En: *Usability Eval. Ind.* 189 (noviembre de 1995).

[Com] Flutter Community. *Workmanager*. URL: `https://pub.dev/packages/workmanager` (Último acceso, 16 de junio de 2025).

[Dart] Dart. URL: `https://dart.dev` (Último acceso, 13 de marzo de 2025).

[Den] E. Deniz. «iOS Swift: SQLite3 Integration». En: (). URL: `https://medium.com/@emre.deniz/ios-swift-sqlite3-integration-1b3dece47b46` (Último acceso, 28 de junio de 2025).

[Deva] Android Developers. *Develop Android apps with Kotlin*. URL: `https://developer.android.com/kotlin` (Último acceso, 13 de junio de 2025).

[Devb] Android Developers. *Especificaciones de diseño de íconos de Google Play*. URL: `https://developer.android.com/distribute/google-play/resources/icon-design-specifications?hl=es-419` (Último acceso, 23 de febrero de 2025).

[Devc] Android Developers. *Meet Android Studio*. URL: `https://developer.android.com/studio/intro` (Último acceso, 13 de junio de 2025).

[Devd] Android Developers. *Save data using SQLite*. URL: `https://developer.android.com/training/data-storage/sqlite` (Último acceso, 28 de junio de 2025).

[ES23] Brady Eidson y Jen Simmons. «Web Push for Web Apps on iOS and iPadOS». En: (febrero de 2023). URL: https://webkit.org/blog/13878/web-push-for-web-apps-on-ios-and-ipados/ (Último acceso, 15 de junio de 2025).

[Est14] Agencia Estatal Boletín Oficial del Estado. *Orden SSI/2065/2014, de 31 de octubre, por la que se modifican los anexos I, II y III del Real Decreto 1030/2006, de 15 de septiembre, por el que se establece la cartera de servicios comunes del Sistema Nacional de Salud y el procedimiento para su actualización.* Boletín Oficial del Estado, noviembre de 2014. URL: https://www.boe.es/buscar/doc.php?id=BOE-A-2014-11444 (Último acceso, 9 de marzo de 2025).

[Fer23a] Alejandro Fernández Fernández. «Diseño e implementación de una aplicación móvil multiplataforma para la preparación de colonoscopias». Trabajo de Fin de Grado. Universidad de Valladolid, julio de 2023.

[Fer23b] Alejandro Fernández Fernández. «Diseño e implementación de una aplicación móvil multiplataforma para la preparación de colonoscopias». Trabajo de Fin de Grado. Universidad de Valladolid, julio de 2023. Anexo B. Documento de referencia facilitado por el equipo médico.

[Fie00] Roy T Fielding. «Architectural styles and the design of network-based software architectures». Tesis doct. University of California, Irvine, 2000. Cap. 5. Representational State Transfer (REST).

[Flua] Flutter. *Build a form with validation.* URL: https://docs.flutter.dev/cookbook/forms/validation (Último acceso, 16 de junio de 2025).

[Flub] Flutter. *DateTime class.* URL: https://api.flutter.dev/flutter/dart-core/DateTime-class.html (Último acceso, 25 de junio de 2025).

[Fluc] Flutter. *Dialog class.* URL: https://api.flutter.dev/flutter/material/Dialog-class.html (Último acceso, 16 de junio de 2025).

[Flud] Flutter. *Fetch data from the internet.* URL: https://docs.flutter.dev/cookbook/networking/fetch-data (Último acceso, 16 de junio de 2025).

[Flue] Flutter. *Future¡T¿ class.* URL: https://api.flutter.dev/flutter/dart-async/Future-class.html (Último acceso, 3 de julio de 2025).

[Fluf] Flutter. *Internationalizing Flutter apps.* URL: https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization (Último acceso, 14 de junio de 2025).

[Flug] Flutter. *InternetAddress class.* URL: https://api.flutter.dev/flutter/dart-io/InternetAddress-class.html (Último acceso, 16 de junio de 2025).

[Fluh] Flutter. *NavigationBar class.* URL: https://api.flutter.dev/flutter/material/NavigationBar-class.html (Último acceso, 23 de febrero de 2025).

[Flui] Flutter. *Persist data with SQLite.* URL: https://docs.flutter.dev/cookbook/persistence/sqlite (Último acceso, 28 de junio de 2025).

[Fluj] Flutter. *Send data to the internet.* URL: https://docs.flutter.dev/cookbook/networking/send-data (Último acceso, 16 de junio de 2025).

[Fluk] Flutter. *Simple app state management.* URL: https://docs.flutter.dev/data-and-backend/state-mgmt/simple (Último acceso, 16 de junio de 2025).

[Flutter] Flutter. URL: https://flutter.dev (Último acceso, 13 de marzo de 2025).

[Foua] OpenJs Foundation. *Express.* URL: https://expressjs.com/ (Último acceso, 20 de junio de 2025).

[Foub] OpenJs Foundation. *Node.js*. URL: https://nodejs.org/en (Último acceso, 18 de junio de 2025).

[Fouc] The Linux Foundation. *OpenAPI*. URL: https://www.openapis.org/ (Último acceso, 3 de julio de 2025).

[Gam+95] Eric Gamma, Richard Helm, Ralph Johnson y John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GCO] Global Cancer Obsevatory — International Agency for Research on Cancer. URL: https://gco.iarc.who.int/ (Último acceso, 9 de marzo de 2025).

[Gri+13] Rebecca A. Grier, Aaron Bangor, Philip Kortum y S. Camille Peres. «The System Usability Scale: Beyond Standard Usability Testing». En: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 57.1 (2013), págs. 187-191. DOI: 10.1177/1541931213571042. eprint: https://doi.org/10.1177/1541931213571042. URL: https://doi.org/10.1177/1541931213571042.

[IBM] IBM. *What is iOS app development?* URL: https://www.ibm.com/think/topics/ios-app-development (Último acceso, 13 de junio de 2025).

[Inc] Apple Inc. *App icons*. URL: https://developer.apple.com/design/human-interface-guidelines/app-icons (Último acceso, 23 de febrero de 2025).

[Jet23] JetBrains. *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2023*. Junio de 2023. URL: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ (Último acceso, 13 de marzo de 2025).

[JSON] *JavaScript Object Notation*. URL: https://json.org (Último acceso, 10 de mayo de 2025).

[Koe16] George. Koelsch. *Requirements Writing for System Engineering*. 1st ed. 2016. Berkeley, CA: Apress, 2016.

[Lai+09] E.J. Lai, A.H. Calderwood, G Doros, O.K. Fix y B.C. Jacobson. «The Boston Bowel Preparation Scale: A valid and reliable instrument for colonoscopy-oriented research». En: *Gastrointestinal endoscopy* 69 (enero de 2009), págs. 620-625. DOI: https://doi.org/10.1016/j.gie.2008.05.057. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC2763922/ (Último acceso, 22 de marzo de 2025).

[M D] C. M. DiMascio. *express-openapi-validator*. URL: https://www.npmjs.com/package/express-openapi-validator (Último acceso, 3 de julio de 2025).

[Maj18] D. Majmudar. «Comparing APK sizes». En: *AndroidPub* (marzo de 2018). URL: https://medium.com/android-news/comparing-apk-sizes-a0eb37bb36f (Último acceso, 21 de marzo de 2025).

[MD3] Material Design. *Chips*. URL: https://m3.material.io/components/chips/overview (Último acceso, 24 de febrero de 2025).

[Nav10] W. Navidi. *Statistics for Engineers and Scientists*. 3rd ed. 2010. McGraw-Hill, 2010.

[Nes+01] R.M Ness, R. Manam, H. Hoen y N. Chalasani. «Predictors of inadequate bowel preparation for colonoscopy». En: *The American Journal of Gastroenterology* 96.6 (2001), págs. 1797-1802. ISSN: 0002-9270. DOI: https://doi.org/10.1016/S0002-9270(01)02437-6. URL: https://www.sciencedirect.com/science/article/pii/S0002927001024376.

[Neta] Mozilla Developer Network. *JavaScript reference*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (Último acceso, 20 de junio de 2025).

[Netb] Mozilla Developer Network. *Notifications API*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API (Último acceso, 15 de junio de 2025).

[Nie+99] H. Nielsen, J. Mogul, L.M. Masinter, R.T. Fielding, J. Gettys, P.J. Leach y T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Junio de 1999. URL: https://datatracker.ietf.org/doc/html/rfc2616 (Último acceso, 18 de mayo de 2025).

[NLM] National Library of Medicine. *PubMed*. URL: https://pubmed.ncbi.nlm.nih.gov/.

[Ogd] Cody Ogden. *Killed by Google*. URL: https://killedbygoogle.com/ (Último acceso, 2 de julio de 2025).

[Pan22] M. Pantaleón Sánchez et al. «Prevalence of missed lesions in patients with inadequate bowel preparation through a very early repeat colonoscopy». En: *Digestive Endoscopy* 34 (2022). URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9545231/pdf/DEN-34-1176.pdf (Último acceso, 9 de marzo de 2025).

[Prep] D. Sánchez. *PrepColon*. URL: https://prepcolon.gsic.uva.es/ (Último acceso, 21 de junio de 2025).

[Rei] Andris Reinman. *Nodemailer*. URL: https://nodemailer.com/ (Último acceso, 20 de junio de 2025).

[RR07] L. Richardson y S Ruby. *RESTful Web Services*. 1st ed. 2007. O'Reilly y Associates, 2007.

[San25] Ministerio de Sanidad — Gobierno de España. URL: https://www.sanidad.gob.es/areas/promocionPrevencion/cribado/cribadoCancer/cancerColon/home.htm (Último acceso, 9 de marzo de 2025).

[Sha05] Y. Shafranovich. *Common Format and MIME Type for Comma-Separatec Values (CSV) Files*. RFC 4180. Octubre de 2005. URL: https://datatracker.ietf.org/doc/html/rfc4180 (Último acceso, 16 de junio de 2025).

[Smaa] SmartBear. *OpenAPI Specification*. URL: https://swagger.io/specification/v3/ (Último acceso, 18 de junio de 2025).

[Smab] SmartBear. *Swagger*. URL: https://swagger.io/ (Último acceso, 18 de junio de 2025).

[SS05] Alberto Sillitti y Giancarlo Succi. «Requirements Engineering for Agile Methods». En: *Engineering and Managing Software Requirements*. Ed. por Aybüke Aurum y Claes Wohlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, págs. 309-326. DOI: 10.1007/3-540-28244-0_14.

[Sta24] StatCounter. *Cuota de mercado mundial de los sistemas operativos para dispositivos móviles de 2010 a 2024*. Diciembre de 2024. URL: https://es.statista.com/estadisticas/635202/sistemas-operativos-de-telefonos-moviles-cuota-de-mercado-mundial (Último acceso, 13 de marzo de 2025).

[Tav16] M. S. Tavaragi. «Colors and Its Significance». En: *The International Journal of Indian Psychology* 3 (marzo de 2016). URL: https://oaji.net/articles/2016/1170-1457802371.pdf (Último acceso, 21 de febrero de 2025).

[Ver] A. Verma. «Epochs, Batch and Iterations in Deep Learning». En: (). URL: https://medium.com/@akankshaverma136/epochs-batch-and-iterations-in-deep-learning-ed319565e85e (Último acceso, 28 de junio de 2025).

[Wal+21] B Walter, R Frank, L Ludwig, N Dikopulos, M Mayr, B Neu, B Mayer, A Hann, B Meier, K Caca, T Seufferlein y A Meining. «Smartphone Application to Reinforce Education Increases High-Quality Preparation for Colorectal Cancer Screening Colonoscopies in a Randomized Trial». En: *Clinical Gastroenterology and Hepatology* 19 (febrero de 2021), págs. 331-338. ISSN: 1542-3565. DOI: https://doi.org/10.1016/j.cgh.2020.03.051. URL: https://www.cghjournal.org/article/S1542-3565(20)30428-6/fulltext (Último acceso, 21 de marzo de 2025).

[WHO] World Health Organization. «Cancer». En: (febrero de 2025). URL: https://www.who.int/news-room/fact-sheets/detail/cancer (Último acceso, 9 de marzo de 2025).

[WSD17] B. Walter, R. Schmid y S. von Delius. «A Smartphone App for Improvement of Colonoscopy Preparation (ColoprApp): Development and Feasibility Study». En: *JMIR mHealth and uHealth* 5 (septiembre de 2017). DOI: https://doi.org/10.2196/mhealth.7703. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC5628282/ (Último acceso, 21 de marzo de 2025).

[Zan+21] Q.E.W. van der Zander, A. Reumkens, B. van de Valk, B. Winkens, A.A.M. Masclee y R.J.J. de Ridder. «Effects of a Personalized Smartphone App on Bowel Preparation Quality: Randomized Controlled Trial». En: *JMIR mHealth and uHealth* 9 (agosto de 2021). DOI: https://doi.org/10.2196/26703. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8414298/ (Último acceso, 21 de marzo de 2025).

[Zhu+23] Y. Zhu, D. Zhang, H. Wu, P. Fu, L. Feng, K. Zhuang, Z. Geng, K. Li, X. Zhang, B. Zhu, W. Qin, S. Lin, T. Chen, Y. Huang, X. Xu, J. Liu, S. Wang, W. Zhang, Q. Li y P. Zhou. «Improving bowel preparation for colonoscopy with a smartphone applicatino driven by artificial intelligence». En: *NPJ digital medicine* 6 (marzo de 2023), págs. 1-41. DOI: https://doi.org/10.1038/s41746-023-00786-y. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10011797/ (Último acceso, 21 de marzo de 2025).

Anexo A

Información de referencia

Para el desarrollo de esta aplicación se contó con la ayuda de profesionales del ámbito sanitario especializados en el aparato digestivo, que fueron quienes nos ayudaron a decidir cómo debería ser el comportamiento del programa. Concretamente, durante el desarrollo de la aplicación PrepColon nos hemos guiado por un documento similar al que se referencia en [Fer23b], cuyo contenido se especifica en este apéndice.

Este comportamiento vendría determinado por las características individuales del usuario, de forma que las indicaciones dadas por la aplicación pudiesen estar adaptadas a las necesidades del usuario. Para ello, el usuario debe rellenar un formulario inicial en el que no sólo indicará la fecha y la hora de su colonoscopia, sino que además dará información sobre sus características clínicas:

- Si tiene sobrepeso (se pide altura y peso para calcular el IMC)
- Si realiza menos de 3 deposiciones semanales
- Si sufre diabetes
- Si sufre Parkinson
- Si se ha sometido a cirugía bariátrica
- Si ha recibido intervenciones quirúrgicas en la zona abdominal
- Si se ha realizado previamente alguna colonoscopia con malos resultados
- Si toma fármacos relacionados con el sistema nervioso, el control del dolor, la presión arterial
- Si toma fármacos relacionados con la sangre: hierro, anticoagulantes o antiagregantes

Si cumple alguna de las características anteriores, se deberá avisar al usuario de que debe intensificar la preparación de la siguiente forma:

- Deberá aumentar la cantidad de líquidos que toma al día
- Deberá tomar un laxante adicional, Evacuol®, por las noches durante los cuatro días previos al inicio de la toma del laxante principal
- Deberá aumentar el tiempo entre las tomas del laxante principal
- Si toma hierro, deberá suspenderlo 5 días antes de su colonoscopia
- Si toma anticoagulantes o antiagregantes, deberá contactar con su Médico de Atención Primaria, quien le indicará si debe suspenderlos o sustituirlos

En cualquier caso, deberá también indicar el laxante principal que utilizará durante su preparación. El usuario tomará dos dosis, comenzando el mismo día de la cita si ésta es por la tarde, o la tarde-noche anterior si la cita es por la mañana.

A.1. Laxantes

Solución de Bohn® o Casenglicol®

Este laxante se toma en dos dosis formadas por el contenido de 8 sobres del evacuativo diluidos en 2 litros de agua. La dosis deberá tomarse lentamente, aproximadamente 250ml cada 15 minutos.

El paciente debe comenzar a tomar la primera dosis el día anterior a la cita a las 19h si la cita es por la mañana, o el mismo día a las 7h si la cita es por la tarde. La segunda dosis deberá comenzar a tomarse 5h antes de la cita.

Pleinvue®

Este fármaco está compuesto por tres sobres. Para la primera dosis bastará con diluir el contenido de uno de los sobres en 500ml de agua, mientras que para la segunda dosis se deberá diluir el contenido de los dos sobres restantes. Ambas tomas deben beberse poco a poco, aproximadamente 250ml cada 15 minutos, e intercaladas con otros 250ml de algún líquido claro.

La primera dosis se tomará a las 21h del día anterior a la cita si ésta es por la mañana, o a las 9h del mismo día de la cita en caso contrario. En cualquiera de los casos, la segunda toma deberá comenzar 5 horas antes de la cita.

Citrafleet®

Este laxante está comercializado en dos formatos: líquido o polvos. En ambos casos el procedimiento es el mismo: el paciente debe disolver el contenido de un sobre en aproximadamente 250ml de agua y tomarlo lentamente durante 10 ó 15 minutos. Tras esto, se debe esperar 10 minutos y, posteriormente, tomar 1.5 litros de algún líquido claro durante las siguientes 2h.

La primera toma será a las 19h del día anterior a la cita si ésta es por la mañana o a las 7h de la mañana del mismo día de la cita. La segunda dosis debe tomarse 5h antes de la colonoscopia.

Moviprep®

Moviprep® se toma disolviendo el contenido de dos sobres en 1 litro de agua, y bebiendo unos 250ml cada 15 minutos. Una vez terminada la mezcla, la toma debe finalizarse bebiendo al menos medio litro adicional de líquidos claros.

La primera dosis se toma a las 20h o a las 8h, y la segunda, al igual que los anteriores, 5h antes de la cita.

Clensia®

Clensia® es un fármaco más o menos reciente que se incluyó durante el desarrollo de la aplicación. Su preparación es similar a Moviprep®: se disuelve el contenido de cuatro sobres en 1 litro de agua para beberlo en tomas de aproximadamente 250ml cada 15 minutos. Tras finalizar la disolución, el paciente debe tomar otro litro de líquidos claros.

A.2. Dieta

Durante la preparación, el paciente también tiene que cumplir una dieta que irá variando a lo largo del proceso.

- **Dieta sólida:** 4 días antes de la cita, el paciente deberá evitar frutas, verduras y legumbres, productos integrales, carnes grasas y embutidos, frutos secos, pescado azul o lácteos
- **Dieta líquida:** El día anterior a la cita, el usuario no podrá ingerir ningún alimento sólido, restringiendo su dieta únicamente a líquidos claros
- **Ayuno:** Finalmente, dos horas antes de la colonoscopia, el paciente debe ayunar

A.3. Estado de las deposiciones

A lo largo del proceso de preparación, es también importante comprobar el estado de las deposiciones del paciente para corroborar que se está realizando de forma correcta. Lo ideal sería encontrar heces cada vez más líquidas. Si la primera deposición tiene lugar pasadas las 3h desde que el paciente comenzó a tomar el laxante principal, se le recomienda aumentar la cantidad de líquidos en 1.5 litros.

Si a menos de 2h de la cita las heces son más o menos sólidas, entonces la colonoscopia será incorrecta, por lo que el paciente debería notificarlo al servicio de endoscopias para mover la cita.

A.4. Recomendaciones para el paciente

Adicionalmente, si durante la toma de los laxantes el paciente presenta náuseas, se le recomienda beber la solución más despacio y aumentando la separación de 15 minutos a 20 ó 25 minutos, o mezclar el laxante con agua fría o zumo sin pulpa. En cambio, si el paciente vomita gran parte del laxante, se le indica que lo notifique a su hospital para reprogramar la cita y cambiar de laxante.

Anexo B

Aplicación inicial

Este TFG toma como punto de partida el trabajo descrito en [Fer23a], donde se encuentran las bases sobre las que se ha desarrollado PrepColon. En este anexo se hace un análisis de esa primera versión de la aplicación, primero desde un punto de vista más funcional y después con una perspectiva más técnica.

B.1. Funcionamiento de la aplicación

Si bien es cierto que la aplicación de [Fer23a] estaba incompleta, implementaba funcionalidades básicas que permitían determinar las necesidades del paciente según sus características clínicas. Esta primera versión de lo que posteriormente sería PrepColon mostraba al usuario un formulario en el que recogía los datos indicados en el Anexo A para determinar si el paciente necesitaba o no realizar una preparación intensiva, así como almacenar el laxante principal elegido por el usuario y generar algunas tareas que éste debe completar.

Sin embargo, como se menciona anteriormente, la aplicación estaba incompleta. El algoritmo estaba preparado para funcionar como una máquina de estados pero únicamente estaban implementados los dos estados iniciales: aquel en el que el usuario entraba en la aplicación por primera vez—es decir, no había datos guardados—, y el estado al que pasaba la aplicación una vez se completaba el usuario. Para otros estados, la pantalla del dispositivo móvil permanecía en negro debido a la falta de código subyacente que conformase la interfaz de usuario.

B.1.1. Pantalla principal

En esta versión de la aplicación, tal y como se puede apreciar en las Figuras B.1a y B.1b se mostraban diferentes pantallas principales en función del estado en el que se encontrase. Para una primera situación en la que aún no hay datos guardados, el usuario se encontraba con un botón que le llevaba a iniciar un formulario con 15 preguntas diferentes. Una vez terminado el formulario, se mostraba un resumen de las respuestas introducidas para que el usuario pudiera

corroborar que todos los datos eran correctos. Posteriormente, la aplicación volvía a la pantalla principal, que ahora mostraba la fecha de la cita y las tareas que debía completar el usuario durante su preparación para la colonoscopia.

Una vez contestadas estas 9 primeras preguntas, se muestra un aviso al usuario indicando que debe seleccionar todos los medicamentos que tome o, en su defecto, marcar la opción «Ninguno de ellos». Estos medicamentos se especifican en la Tabla B.1. A partir de aquí vemos 4 preguntas de respuesta múltiple con diferentes opciones que incluyen, para facilitar el entendimiento del usuario, ejemplos de versiones comercializadas de algunos de los fármacos. Por último, se pregunta al usuario si se realizó una colonoscopia fallida con anterioridad y se le pide que indique la preparación laxante que le hayan recetado.

Tipo	**Nombre**	**Ejemplo comercial**
Medicamentos relacionados con la sangre	Hierro	—
	Antiagregantes	—
	Anticoagulantes orales	—
Medicamentos relacionados con el sistema nervioso (nervios, ansiedad...)	Amitriptilina	Tryptizol®
	Imipranina	Tofranil®
	Clomipramina	Anafranil®
	Paroxetina	—
	Venlafaxina	—
	Risperidona	Risperdal®
	Clozapina	Leponex® Nemea®
	Olanzapina	—
	Haloperidol	—
	Amisulpiride	—
	Quetiapina	Seroquel® Actiq®
Medicamentos relacionados con el dolor	Fentanilo	Durogesic® Fendivia®
	Tramadol	Adolonta® Zaldiar®
Medicamentos relacionados con la presión arterial	Amlodipino	—
	Diltiazem	—
	Nicardipino	—

Tabla B.1: *Medicamentos a tener en cuenta durante la preparación para la colonoscopia.*

B.1.2. Menú lateral

La aplicación también contaba con un menú lateral, mostrado en la Figura B.1d que contenía las siguientes opciones:

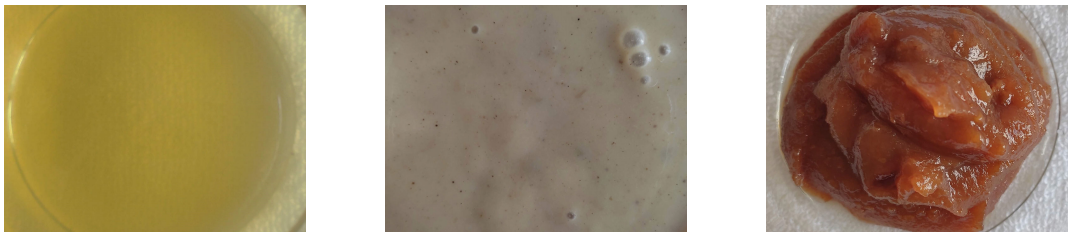
- **Revisar formulario:** Opción para ver el resumen del formulario.
- **Modificar formulario:** Opción para volver a comenzar el formulario, el cual aparece relleno con los datos introducidos la anterior vez que el usuario lo realizó.

- **Historial** (sin implementar)
- **Cancelar preparación:** Opción para borrar todos los datos.
- **Revisar formulario:** Opción para elegir el aspecto de la última deposición del usuario.

B.1.3. Aspectos

Al seleccionar la última opción, «Revisar formulario», del menú lateral, el usuario pasa a una pantalla adicional en la que se le muestran tres imágenes representativas que debe comparar con el aspecto de las heces tras su última deposición. La idea tras esto es hacer un seguimiento de la preparación del paciente para comprobar que los laxantes están teniendo un correcto efecto en el cuerpo del usuario. Los aspectos son:

- **Líquido:** Este es el aspecto ideal que indica una buena preparación.
- **Pastoso:** Aunque no es el aspecto ideal, puede permitir una correcta preparación si el usuario aumenta la cantidad de líquidos que toma.
- **Sólido:** Este aspecto resulta preocupante en estados avanzados de la preparación. El paciente debe aumentar la cantidad de líquidos en su dieta, pero si presenta este aspecto durante las dos horas previas a la cita deberá cancelarla, ya que la preparación habrá resultado fallida.



(a) *Líquido* **(b)** *Pastoso* **(c)** *Sólido*

Fig. B.3: *Imágenes representativas de los posibles aspectos de las heces.*

B.2. Código

El código de [Fer23a] está desarrollado con Dart y Flutter, al igual que la aplicación PrepColon sobre la que trata este trabajo, por lo que el grueso del código se encuentra en el directorio `/lib/d` entro del directorio raíz del proyecto. Aquí encontramos el indispensable fichero `main.dart`, que contiene el código necesario para inicializar la aplicación, junto con otros directorios en los que se reparten el resto de ficheros de la aplicación.

B.2.1. Directorio `/lib/l10n`

Una posible futura mejora de la aplicación es adaptarla a diferentes idiomas. Es por esto que en [Fer23a] se construyeron las bases para permitir esta característica gracias al uso del paquete `gen_l10n` de Flutter y a su clase `AppLocalizations`.

El directorio `/lib/widgets` contiene diferentes ficheros ARB, uno para cada idioma que queramos aplicar en la aplicación - en este caso, inglés (`app_en.arb`) y español (`app_es.arb`). En cada uno de estos ficheros se definen, en formato «`clave:valor`», las cadenas de texto que se mostrarán al usuario dependiendo del idioma que tenga configurado en su dispositivo. La ventaja de este método es la facilidad para la localización, ya que la propia librería elige automáticamente qué idioma utilizar.

En [Fer23a], aunque no se definieron todos los textos que aparecen en la aplicación, sí se realizó la traducción de unos pocos, lo que facilitó enormemente el trabajo posterior para la internacionalización de PrepColon.

B.2.2. Directorio `/lib/widgets`

La principal característica de Flutter es la existencia de los `widgets`, clases que permiten implementar fácilmente elementos de la interfaz gráfica. En el directorio `/lib/widgets` se encuentra definido un `widget` personalizado, `ShowCard`, usado para mostrar las tarjetas con los diferentes avisos para el usuario. Esta clase hereda de `StatefulWidget`, lo que le permite tener asociada una subclase privada `_ShowCardState` cuyas propiedades pueden cambiar durante de la ejecución del código.

Esta clase contiene los atributos `ancho` y `alto`, que permiten definir las dimensiones de la tarjeta, un atributo de tipo `ColonprepInfo`, otro de tipo `Cards` y finalmente un atributo de tipo `Card`. Hablaremos con más detalle de estas clases en el apartado B.2.5.

B.2.3. Directorio `/lib/tools`

Este directorio contiene una clase auxiliar, `Tools`, que implementa métodos para convertir fechas de objetos de tipo `DateTime` a cadenas de texto con diferentes formatos como pueden ser «`DD-MM-YYYY`», «`DD-MM-YYYY hh:mm`» o «`hh:mm`».

B.2.4. Directorio `/lib/services`

Aquí podemos encontrar clases que aportan a la aplicación funcionalidades de más bajo nivel, relacionadas con una capa de control. Este directorio alberga las siguientes clases:

- `AppointmentsServer:` Clase pensada para gestionar las comunicaciones entre la aplicación y un servidor externo. Aunque está implementada, esta funcionalidad no se llegó a utilizar en esta versión.
- `CardsManager:` Clase encargada de la creación, en función de los datos recopilados por la app, de las tarjetas de avisos que se muestran al usuario.
- `LocalFile:` Clase auxiliar para el manejo de los ficheros en los que se almacenan los datos del usuario.
- `LocalNotification:` Clase creada para inicializar, mostrar, programar o cancelar las notificaciones que se muestran al usuario.

- `LocalSharedPreferences:` Clase que se encarga de obtener un objeto de tipo `SharedPreferences` para guardar información adicional en el almacenamiento persistente del dispositivo asociado a la aplicación.
- `NotificationsManager` (sin implementar)

B.2.5. Directorio `/lib/models`

En este directorio se definen diferentes clases diseñadas para serializar y deserializar los datos con los que trabaja la aplicación, de forma que puedan almacenarse en ficheros JSON. Encontramos las siguientes clases encargadas de convertir los datos en formato «`clave:valor`»:

- `Appointment:` Para los datos relacionados con la cita, como la fecha o el turno - mañana o tarde.
- `Card:` Clase que almacena la información relativa a una tarjeta de aviso: la marca de tiempo en la que debe aparecer, el texto a mostrar, su tipo y estado - según se indica en la Tabla B.2 -, y una breve descripción.
- `Cards:` Mientras que `Card` está relacionada con un único aviso, `Cards` guarda una lista con todos los avisos, es decir, todos las instancias de tipo `Card` creadas.
- `Preparation:` Esta clase contiene la información relativa al proceso de preparación para la colonoscopia, como el laxante recetado, la fecha en la que el usuario debe comenzar a tomarlo, o un registro con las fechas en las que el usuario ha tomado el laxante o ha defecado.
- `IntakeReport:` Clase que incluye la fecha de una toma de laxante realizada por el usuario.
- `DefecationReport:` Clase que representa la fecha una deposición realizada por el usaurio.
- `PatientQuestionnaire:` Esta clase maneja los datos recopilados por la aplicación derivados de las respuestas introducidas en el formulario.
- `Log:` Clase que contiene los atributos necesarios para registrar un evento que ocurra durante la ejecución de la aplicación.
- `ColonprepInfo:` Esta clase contiene instancias de las clases anteriores, de forma que a través de ella se tenga acceso a todos los datos generados por la aplicación.

Nombre	**Descripción**
`NoAction`	Estado de la tarjeta con la que el usuario no ha interactuado.
`Correct`	Estado de la tarjeta que ha recibido por parte del usuario una respuesta esperada.
`Incorrect`	Estado de la tarjeta que no ha recibido por parte del usuario una respuesta esperada.
`ToDo`	Tipo de tarjeta que muestra un aviso con importancia alta.
`Completed`	Tipo de tarjeta cuya tarea ha sido completada.

Tabla B.2: *Estados y tipos de las tarjetas de avisos de [Fer23a].*

B.2.6. Directorio `/lib/screens`

En este directorio encontramos las clases que definen los elementos de la interfaz de usuario que representan las distintas pantallas del cuestionario inicial. Todas ellas derivan de la clase `StatefulWidget`, por lo que llevan asociada una subclase privada que representa su estado. De esta manera su diseño puede modificarse mientras se ejecuta la aplicación, permitiéndola reaccionar a los valores seleccionados por el usuario.

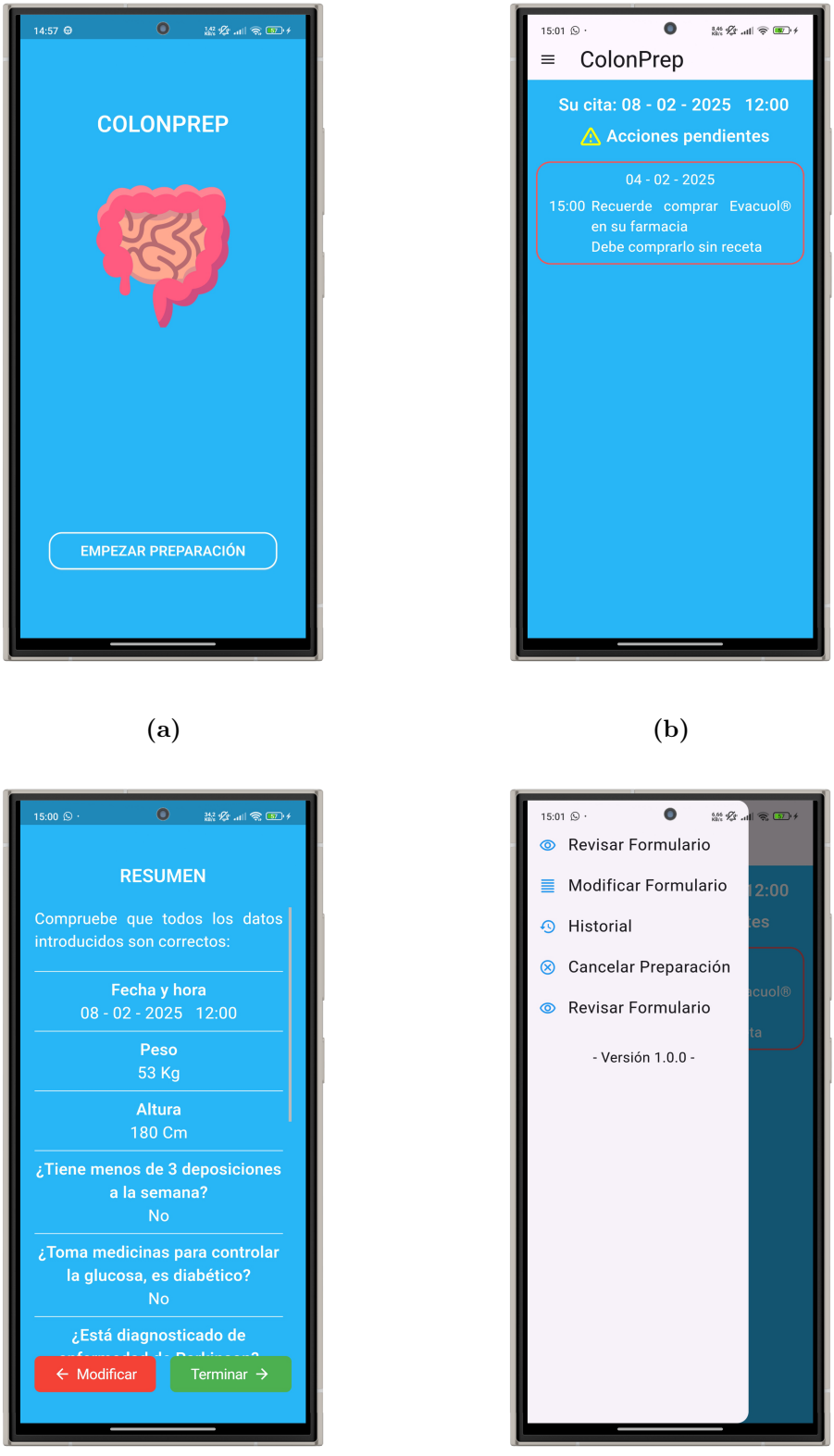
B.2.7. Directorio `/lib/states`

Por último nos movemos al que quizás sea uno de los directorios clave del programa. En él encontramos las clases que definen la interfaz de usuario de la pantalla principal de la aplicación en función del estado de la preparación en la que se encuentre. Este estado se determina mediante una llamada al método `checkState()` de la clase `StateContext`. Dicho método devuelve una clase que implementa la interfaz `State`, y que contiene el código que define qué verá el usuario en la pantalla de su dispositivo. La relación entre el estado y su clase asociada puede verse en la Tabla B.3.

Clase	**Descripción del estado**
`SinCitaState`	Estado inicial en el que no hay datos guardados.
`ConCitaPteDatosState`	El usuario ha comenzado a rellenar el formulario pero no está completo.
`ConCitaListoPrepState`	El usuario ha completado el formulario y puede comenzar la preparación para su colonoscopia.
`EnPreparacionState`	El usuario ha comenzado a tomar el laxante.
`RiesgoMalaPreparacionState`[1]	—
`PreparacionCorrectaState`	El usuario ha completado todas las tareas propuestas por la aplicación.
`MalaPreparacionState`	El usuario no ha confirmado haber tomado la dosis del laxante recetado.

Tabla B.3: *Estados de la aplicación y la clase asociada.*

[1] Esta clase no se utiliza en el código.



(a) **(b)**

(c) **(d)**

Fig. B.1: *Capturas de pantalla de la aplicación. (a) Pantalla principal sin datos guardados. (b) Pantalla principal tras completar el formulario. (c) Resumen con las respuestas del formulario. (d) Menú lateral desplegable.*

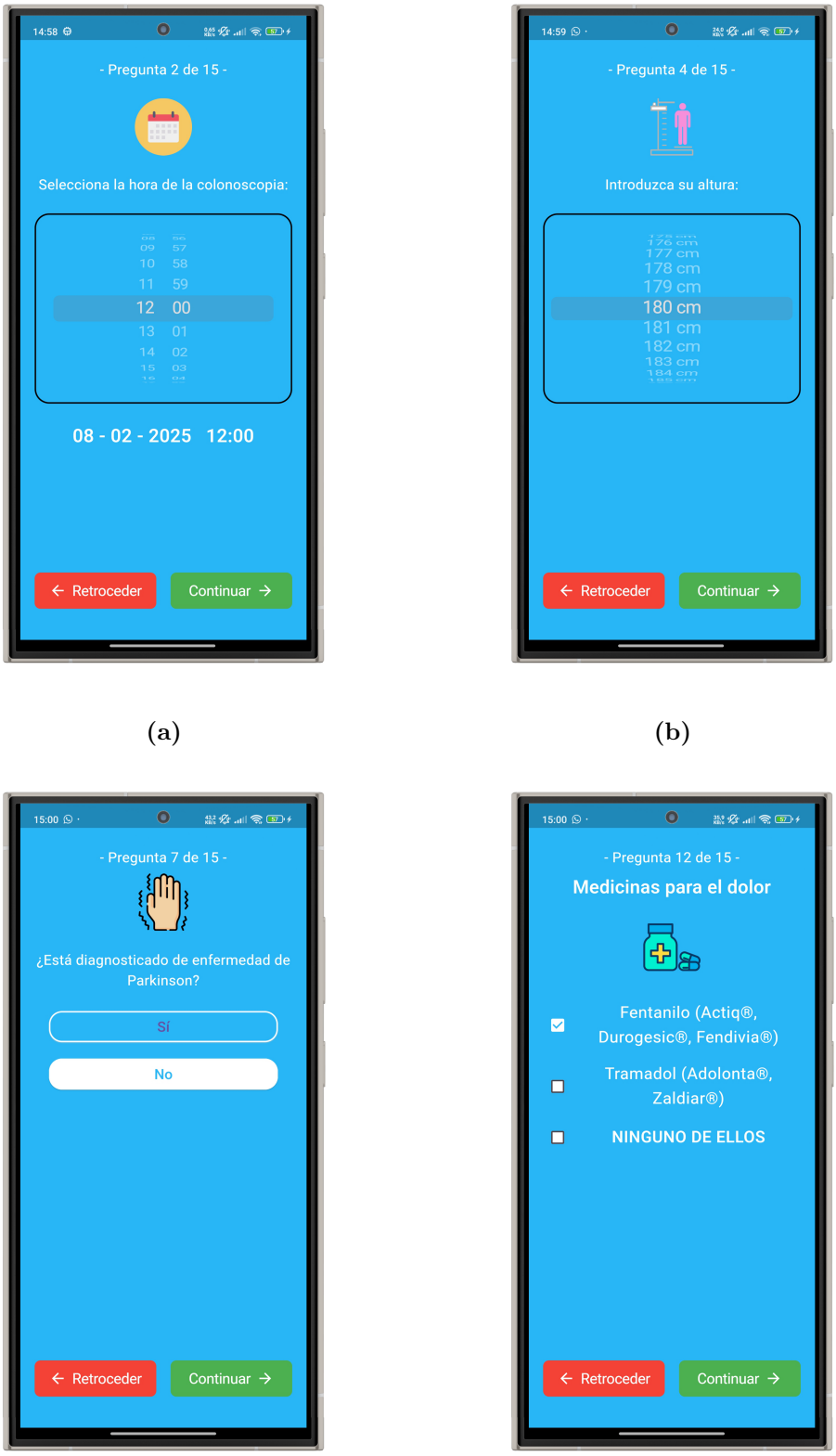


Fig. B.2: *Capturas de pantalla del formulario de la aplicación.*

Anexo C

Guía de colores

A continuación se listan los colores mencionados en este documento.

#29B6F6	#F44336	#4CAF50
#283593	#388E3C	#FF6F00
#AD1457	#004D40	#009688
#E0F2F1		

Anexo D

Logo de la aplicación

Cuando se decidió que el color `#004D40` iba a pasar a ser, de forma definitiva, el color principal de PrepColon, se contactó con una empresa externa para realizar un logotipo para la aplicación utilizando la misma paleta de colores. El resultado sirvió para diseñar un icono que cumpliese los requisitos de las tiendas de Google y Apple[Devb; Inc].

Fig. D.1: *Icono de PrepColon.*

Fig. D.2: *Logo de PrepColon.*