CuidaMed

Memoria Técnica

Ainhoa Lara
Javier Merino
Alex Nagore
Felipe Santos
Manex Tiraplegui
Rubén Zubasti

Descripción

CuidaMed es una aplicación móvil para la gestión y seguimiento de la medicación. Permite registrar los fármacos que el usuario debe tomar, configurar horarios de toma, recibir notificaciones push y llevar un historial de cumplimiento. Además dispone de una herramienta que permite escanear recetas médicas.

Utilidad

- **Evitar olvidos en la toma de medicación:** Notifica a los usuarios para recordarles cuándo y qué medicación tienen que tomar.
- **Facilitar el seguimiento médico y familiar:** Permite a los médicos y familiares realizar un seguimiento sobre las tomas de los medicamentos para
- **Mejorar la adherencia a los tratamientos:** Gracias al sistema de notificaciones y al seguimiento, la adherencia a los tratamientos está asegurada,
- **Dar autonomía a pacientes crónicos y personas mayores:** Los pacientes pueden gestionar las tomas de medicaciones gracias al sistema de notificaciones,
- **Escanear recetas médicas:** Escanea recetas médicas y apunta automáticamente las tomas en el calendario de la aplicación.

Estado del arte

Estudio de mercado / Plan de viabilidad

- Creciente demanda de aplicaciones mHealth.
- Usuarios interesados en control de hábitos y salud digital.
- Segmento creciente: personas mayores + pacientes crónicos.

Aplicaciones / webs de la competencia

- **Medisafe** (Android/iOS).
- **Pill Reminder – Meds Alarm**.
- **MyTherapy**.
- Todas ofrecen recordatorios básicos, pero pocas incluyen personalización, exportación de datos o conexión con cuidadores.

Herramientas, plataformas y lenguajes

- **Frontend móvil:** React Native.

React native es uno de los frameworks más utilizados en el desarrollo móvil. Se puede utilizar para desarrollar tanto en Android como en IOS, característica crucial para este proyecto. Como su propio nombre indica, se basa en la famosa librería "React", por lo que se aprovecha de la mayoría de las ventajas que ofrece.

Hemos elegido React Native, en primer lugar, porque utiliza JavaScript (actualmente el lenguaje más usado en desarrollo web). Es un lenguaje fácil de entender y aprender, con miles de librerías útiles y con compatibilidad prácticamente global. Por otro lado, en vez de crear dos aplicaciones distintas para IOS y Android, React Native nos permite desarrollar para ambas plataformas a la vez, un ahorro del 50% del código.

A la hora de crear código, también nos aporta rapidez. React Native utiliza la función de recarga activa (Hot Reloading), lo que permite al desarrollador implementar cambios en un código y ver el efecto en la aplicación en tiempo real.

También se ha barajado la idea de utilizar Flutter, otro de los frameworks más utilizados, pero la necesidad de aprender un nuevo lenguaje (Dart) y el limitado soporte que tiene en IDEs nos ha hecho elegir React Native.

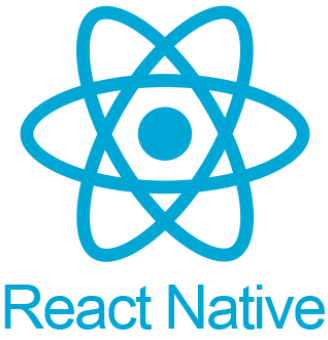


Figura 1.1 Logo de React Native

- **Backend:** Fastify.

  Para el backend, una de nuestras opciones inicialmente, era PHP, ya que tenemos algo de experiencia con él y nos sería bastante más fácil empezar a desarrollarlo por eso mismo. Pero después de barajar varias opciones hemos elegido Fastify. Fastify es un framework de node rápido, moderno y fácil de aprender. A pesar de que no tenemos demasiada experiencia con él, creemos que puede ser interesante aprender a usarlo ya que tanto JavaScript como TypeScript son bastante demandados a día de hoy. Además así usaremos el mismo lenguaje tanto para front como para back.

Figura 1.2 Logo de Fastify

- **Base de datos:** PostgreSQL.

Hemos decidido usar esta base de datos en nuestra aplicación porque es muy versátil. Ofrece las clásicas características de bases de datos relacionales, por lo que podemos usarla como una base MySQL. Sin embargo, también ofrece la opción de guardar los datos de forma no relacional (ficheros JSON, por ejemplo). Por lo tanto, siempre vamos a tener la oportunidad de almacenar datos de otro modo sin necesidad de mudarnos a otra base de datos.

Otras ventajas a destacar pueden ser:

- ➢ Gran escalabilidad
- ➢ Código abierto (instalación gratuita)
- ➢ No se han registrado caídas de la base de datos en sus 20 años de historia
- ➢ Ofrece una herramienta para visualizar nuestra base de datos: **pgAdmin**

Figura 1.3 Logo de PostgreSQL

- **Servidor:** Heroku.

  Para el despliegue seguramente usemos Heroku ya que la hemos usado en alguna ocasión y no nos ha dado demasiados problemas

- **Notificaciones push:** Firebase Cloud Messaging.

Firebase Cloud Messaging es un servicio de mensajería multiplataforma de Google que permite enviar mensajes de forma segura y rápida.

Una de las razones por las que hemos elegido este servicio es que es gratuito, con notificaciones ilimitadas para cualquier usuario. Además, ofrece compatibilidad tanto en IOS como en Android, por lo que se adapta perfectamente a nuestro proyecto.

Otra de las causas es su compatibilidad con la interacción con el usuario. Ofrece la opción de añadir botones al mensaje y, dependiendo de en qué se haga click, se manda un mensaje u otro a la base de datos (característica perfecta para poder indicar si te has tomado la pastilla o no, por ejemplo).

En resumen, es un servicio muy sencillo pero fiable, con la infraestructura de Google, que se adapta perfectamente a nuestros objetivos.

Figura 1.4 Logo de Firebase Cloud Messaging

- **Control de versiones:** GitHub.

  Evidentemente, hemos usado GitHub para gestionar nuestro proyecto. Es la aplicación web de control de versiones más popular y extendida del momento. La hemos elegido por varias razones:

  ➔ Ofrece una gestión de proyectos muy completa y visual, suficiente para la aplicación que queremos hacer ahora.

  ➔ Interfaz clara sobre el código que aporta cada uno

  ➔ Ofrece integración con herramientas de CI/CD (GitHub Actions), por lo que el proceso de despliegue va a estar bien respaldado.

- **Gestión del proyecto:** GitHub Projects.

  GitHub, como hemos mencionado antes, ofrece una sección de proyectos muy completa e intuitiva. Ofrece posibilidades como separar las tareas por categoría, proyectos, versiones... Podemos dividir el trabajo de forma ordenada y limpia.

  Otras alternativas a Github Projects pueden ser Trello, Jira, o GitLab Boards. Todas están bien y completas, pero ya que estamos usando GitHub para el control de versiones, la forma más fácil de unir el desarrollo con la planificación es GitHub Projects.

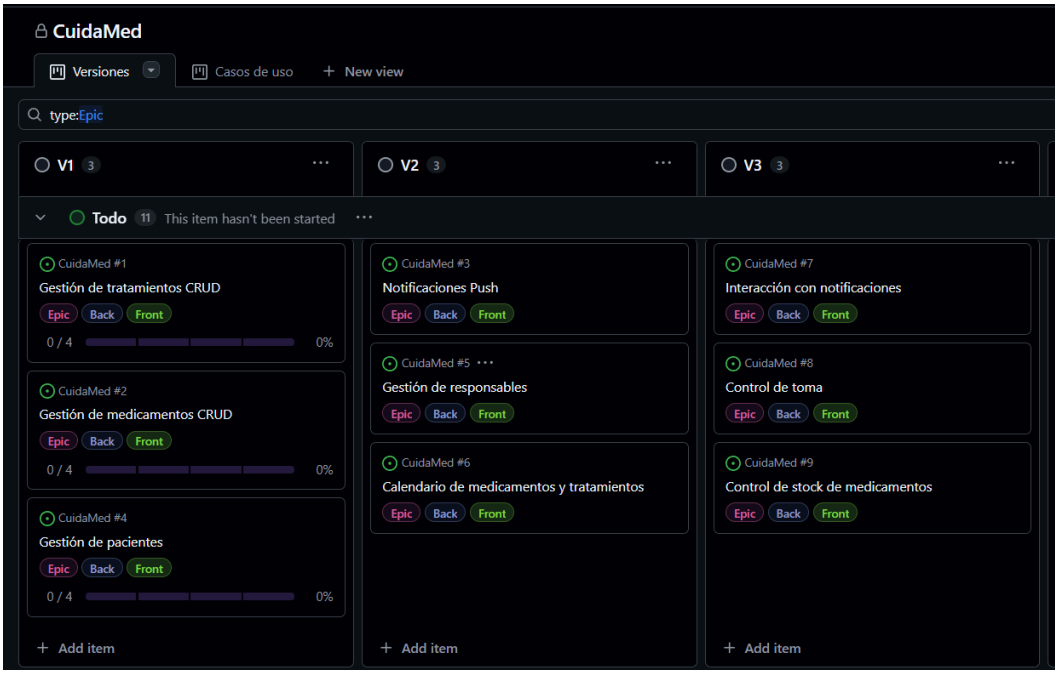


Figura 1.5 Ejemplo de panel de Github Projects

Recursos necesarios

Hardware

- Ordenadores de desarrollo (por miembro de equipo): portátil/PC con CPU quad-core (i5/Ryzen 5 o superior), 16GB RAM, SSD 256+ GB, conexión a Internet estable. Asimismo, macOS, necesario para compilar/testear en iOS con Xcode.
- Dispositivos móviles físicos para pruebas: mínimo 2 Android de distinto perfil (un dispositivo moderno Android 12+ y un dispositivo de gama baja/media, con el objetivo de comprobar compatibilidad y rendimiento); 1 iPhone para pruebas en la versión de iOS.
  Las pruebas en dispositivos reales son imprescindibles para cámara, OCR y notificaciones push.

Software

- Frontend: React Native + Node.js (entorno con npm). Android Studio (SDK + AVD) para Android y Xcode + Cocoa Pods para iOS.
- Backend: Node.js + Fastify (API REST). Herramientas de desarrollo: nodemon (dev), ESLint + Prettier para calidad de código.
- OCR / escaneo de recetas: react-native-vision-camera + ML Kit (Text Recognition) para OCR on-device.
- Testing / QA: Jest + React Native Testing Library.
- IDEs: Visual Studio Code

Limitaciones y restricciones del prototipo

- No incluirá integración con APIs médicas en la primera fase.
- Interfaz simplificada, sin soporte multilenguaje al inicio.
- Implementación básica de las funcionalidades
- Seguridad y privacidad limitadas al cumplimiento básico (no certificado sanitario).

Capacidad de crecimiento

- Integración con APIs médicas para interacciones de fármacos.
- Sincronización con wearables (relojes inteligentes)..
- Algoritmos de recordatorio inteligente (aprendizaje automático).
- Exportación avanzada de datos (para médicos).
- Aplicación web.
- Implementación de traducción.

Público objetivo

- Personas polimedicadas (mayores de 50).
- Pacientes crónicos (diabetes, hipertensión, etc.).
- Cuidadores y familiares de personas dependientes.
- Usuarios jóvenes con tratamientos puntuales (ej. antibióticos).

Objetivos a desarrollar

- **Gestión** y **registro** de medicamentos
- Sistema de **notificaciones push** a la hora de tomar un medicamento
- Sistema de **calendario** para visualizar cuándo y qué medicamento tomar
- **Historial** de adherencia a tratamientos
- Interfaz **accesible** y **clara** para las personas menos familiarizadas con la tecnología
- **Analizar métricas** de uso para futuras mejoras

Funcionalidades del proyecto

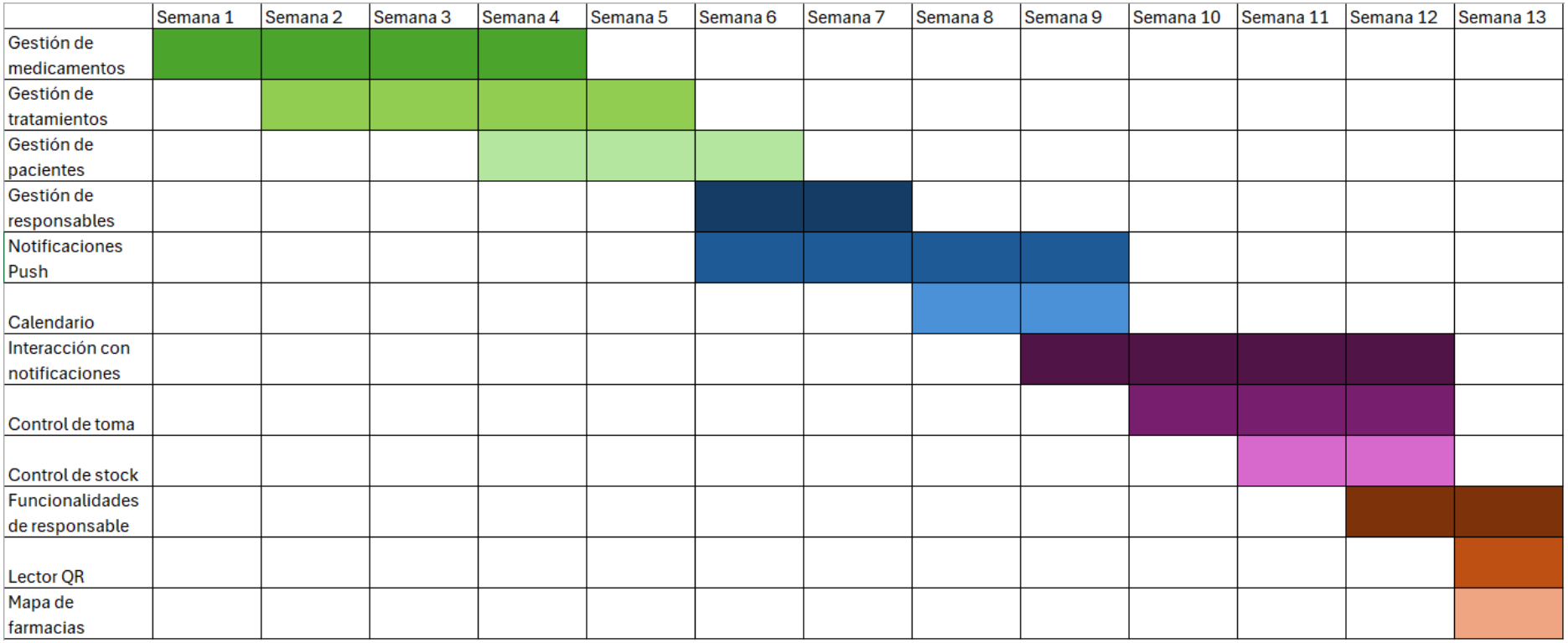
- Registro de medicación (nombre, dosis, horario, recomendaciones).
- Gestión de usuarios: paciente y/o responsable.
- Registro de tratamientos del paciente usuario.
- Notificaciones push en el momento indicado.
- Posponer toma o marcar como realizada.
- Historial de tomas (calendario y porcentajes).
- Control básico de stock de medicación.
- Comunicación de la app con la persona “responsable” de la toma de la medicación.
- Registro / recordatorio de citas médicas.

Plan de trabajo

WBS (Work Breakdown Structure – esquema de paquetes de trabajo)

1. Gestión del proyecto
    - Planificación, asignación de tareas, control de versiones.
2. Análisis y diseño
    - Definición de requisitos.
    - Diseño de arquitectura.
    - Prototipado de interfaz.
3. Desarrollo backend
    - Base de datos.
    - API REST.
    - Notificaciones push.
4. Desarrollo frontend móvil
    - Interfaz de usuario.
    - Registro y gestión de medicación.
    - Pantallas de notificación y confirmación.
    - Calendario de tomas
    - Mapa de farmacias más cercanas
    - Lector de medicamentos
5. Integración y pruebas
    - Pruebas unitarias.
    - Pruebas de usabilidad.
    - Beta testing.
6. Entrega del prototipo
    - Documentación.
    - Presentación.

Diagrama de Gantt (ejemplo simplificado en semanas)



Resultados esperados

- Prototipo funcional de aplicación móvil..
- Registro fiable de tratamientos y notificaciones.
- Interfaz accesible y validada con usuarios.
- Documentación técnica y memoria del proyecto.
- Bases para escalar a una versión completa con más funcionalidades