BABY NANY

[Subtítulo del documento]

Florida Universitària

GENERALITAT VALENCIANA
Conselleria de Educación, Cultura y Deporte

24 DE FEBRERO DE 2026

REALIZADO POR: ADRIÀ GALÁN BORJA, JAUME RUÍZ MORAGA Y ALBERTO PEÑARUBIA GÓMEZ
2ºDAM

Índice

Resumen del proyecto:

El proyecto es una aplicación para móvil con el objetivo de facilitar y monitorear la crianza de un bebé en distintos aspectos. Siendo estos sus características, tales como peso, altura y edad, sus tomas o sus tiempos de sueño. Todo esto acompañado con feedback visual, posibilidad para incluir varios bebés para un solo usuario y una actualización de datos en tiempo real. El proyecto se compone de 3 partes principales.

1. FrontEnd programado con el FrameWork React Native.
2. Una API en un servidor de AWS programada con SpringBoot
3. Una base de datos MongoDB en un servidor de MongoDB Atlas

Justificación y objetivos:

La mayoría de los trabajos he ideas que podíamos ver de otros equipos eran orientadas hacia aplicaciones mas siguiendo un esquema de red social con ciertas temáticas, por lo que queríamos alejarnos de ese estereotipo he intentar crear una aplicación mas orientada hacia la gestión y representación de datos pero que se pudiese crear en la brecha de tiempo que teníamos disponible para la realización del proyecto. Además, una de las compañeras y en un principio integrante del equipo paso por una etapa de crianza de un niño y hubiese agradecido tener una herramienta como la que se plantea.

Con todo esto en mente, llegamos al consenso de crear una app que ayudase a cualquier persona independientemente de su edad, ya sea; padre o madre, padre o madre joven o abuelos intentado hacerse cargo de sus nietos, con el objetivo de facilitar en la medida de lo posible el complejo proceso que es la crianza de un bebé, obviamente la app solo podría ayudar en aspectos de recolección, cálculo y representación, pero en dicho proceso toda ayuda es bienvenida.

Desarrollo del proyecto:

3.1 Análisis de mercado y posibles modelos de negocio: Desde nuestro entendimiento y acceso si existen ya algunas app que pueden ayudar con la crianza y el monitoreo de la crianza de un bebe pero suelen caer en una de 2 categorías: Cuentan con un proceso muy complejo a la hora de la recolección de datos lo que cansa y agobia al usuario produciendo un abandono en su uso o son herramientas no muy conocidas, o por lo menos ninguno de los miembros del equipo a usado o escuchado de alguna aplicación similar.

Por lo tanto, el margen de mejora es claro crear una aplicación de monitoreo de bebés que no sea un formulario interminable y cuente con herramientas que puedan cortar o automatizar en la medida de lo posible la mayor cantidad de recolección de datos.

3.2 Metodología: Para realizar este proyecto iba a ser necesaria la actualización constante de varios puntos de esta durante la duración del tiempo de desarrollo. Es decir, la app constaría de una versión inicial muy básica y se iría incrementando y añadiendo modificaciones una vez realizada la base de la aplicación. Es por esto

que la metodología utilizada necesitaba ser lo suficientemente ágil y flexible como para adaptarse a los diferentes cambios y problemas que pudieran surgir durante el desarrollo de cualquiera de las fases del proyecto. Por estas razones es por las que el equipo decidió la utilización de la metodología "Scrum". Es metodología consiste en la segmentación del trabajo a realizar en distintos "sprints" o fases. Además, dicha metodología requiere de ciertos roles para funcionar con fluidez y correctamente, dichos roles y las personas que los ocuparon se desarrollaran a continuación:

3.2.1 "Product owner": Este rol es el encargado de saber que es lo que tiene que llegar a ser la app, es la persona que sabe lo que quiere y va pidiendo implementaciones y modificaciones durante el proceso de desarrollo. Dentro del equipo de desarrollo el encargado de llevar a cabo dicho rol a sido: Adrià Galán Borja.

3.2.2 "Scrum master": Este rol es el encargado de, en base a los requerimientos del "Product owner" dirigir al equipo en la creación y abordaje de las distintas tareas a realizar. Dentro del equipo de desarrollo, debido al limitado número de integrantes del equipo, no ha existido un solo "Scrum master", dicho título ha variado dependiendo de la fase, momento o estado en el que se encontraba el proyecto en ese momento, por lo que lamentablemente no podemos asignar a nadie con ese rol.

3.2.3 Equipo de desarrollo: Este rol es el encargado de, dirigidos por el "Scrum Master" realizar las tareas asignadas. Son lo que se podría considerar como "la mano de obra" y el motor que mueve todo el proyecto hacia delante. Como ya hemos cubierto en el rol anterior, el número de integrantes del grupo es limitado por lo que todos los integrantes del grupo; Adrià Galán Borja, Jaume Ruíz Moraga y Alberto Peñarrubia Gómez, han ocupado el rol anterior.

3.3 Descripción de los componentes de la aplicación: Como ya esta comentado anteriormente la aplicación esta construida sobre 3 bases importantes, el FrontEnd, el BackEnd y la base de datos. Antes de detallar estos aspectos es importante mencionar ciertos documentos ajenos a estos que ayudaron a la creación del proyecto: Los mockups de las diferentes pantallas del proyecto y un documento en el que se detallaron los principales EndPoints con los que contaría la app en un primer momento. Los mockups se realizaron a papel y se hicieron para todas las pantallas fueran o no del proyecto mínimo viable, por esta razón algunos mockups no fueron utilizados o fueron rediseñados para adaptarse al ritmo del proyecto. Por otro lado, el documento de los EndPoints si estaba únicamente diseñado con los necesarios para la funcionalidad mínima en un primer momento.

3.3.1 FrontEnd ➔ El FrontEnd ha sido diseñado usando React Native y varias librerías externas pertenecientes a este FrameWork. Esta plantedo con una estructura de directorios segmentada en las siguientes carpetas: screens para almacenar todas las pantallas que se mostraran en la app, assets para guardar archivos como imágenes, audios y otro tipo de archivos que no contengan código, services contiene todas las funciones que realizan llamadas o acciones fuera de la app, components donde se almacenan todos los componentes que usan las pantallas para mostrar información, context donde se encuentran todas las variables de uso global de la app y utils carpeta en la que se

almacenan métodos de uso común en toda la app y un archivo destinado a la realización de pruebas de código además de un archivo raíz App.js

3.3.2 BackEnd ➔El BackEnd consta de una API diseñada usando SpringBoot para la realización de funciones y Maven como gestor de dependencias. Este componente está estructurado en “repositoris” que son clases diseñadas para sincronizarse con colecciones de la base de datos, además de poder diseñar funciones que facilitan la realización de operaciones en esta. No todas las clases están pensadas para ser traducidas a “repositoris” algunas simplemente son clases auxiliares que habilitan la realización de algunos métodos. La pieza principal de BackEnd es el controlador, este se encarga de utilizar las clases y “repositoris” mencionados para gestionar todas las peticiones que realiza la API.

3.3.3 Base de datos: Base de datos MongoDB alojada en MongoDB atlas, interacciona con la API para proporcionar la información necesaria al “ForntEnd” a través de esta. Contiene las siguientes colecciones: baby, tokens,intakeRecotd, medicalRecord, email, user, sleepRecoder, featuresRecord.

3.4 Problemas y soluciones:

1. BackEnd ➔ Algunos de los problemas más notorios que han sucedido en este campo del proyecto son:
    a. Git: Debido a nuestra poca experiencia en el uso de Git como servicio de control de versión a sido algo incomodo manejar el trabajo continuo de 2 o más personas durante el diseño del backend, sobre todo a la hora de realizar modificaciones en un mismo archivo.
    b. Compatibilidad IntelliJ y Eclipse: Otro problema que ha surgido es que los miembros del equipo de desarrollo utilizaban frameworks distintos por lo que trabajar en conjunto y sobre todo al momento de realizar push y pull con git a sido un dolor de cabeza. Este problema se soluciono usando el git ignore para no editar algunos archivos que usaban los IDs para poder funcionar, aunque aun así de vez en cuando continuaba dando problemas.
    c. Eliminación de la API: Uno de los miembros del equipo de desarrollo tubo problemas con datos corruptos en su sistema local y de manera no intencional borro el progreso que se tenia hasta ese momento en el backend. El problema fue solucionado y se volvió a sincronizar el repositorio corrupto con el repositorio remoto por lo que no hubo ningún problema, no obstante, provoco que se consumiese una inversión de tiempo innecesaria.
2. FrontEnd ➔Algunos de los problemas más notorios que han sucedido en este campo del proyecto son:
    a. Sincronización: Al momento de sincronizar el BackEnd con el FrontEnd ubo mucho problema de sincronización debido a nuestra falta de conocimiento en el sector. Destacando no saber hacer mas que gets y no saber como construir el header y el body. La solución fue dedicar cierto tiempo al aprendizaje del funcionamiento del resto de tipos de peticiones necesarias en el proyecto, put, post y delete, así como al funcionamiento correcto del header y el body para las peticiones.

b. Sensibilidad con el Git: De manera similar a el backend en la parte del frontend el Git muchas veces se confundía, generando conflictos inexistentes y algo absurdos, así como no copiando o copiando de manera parcial información de algunos push. La solución de este problema fue aprender como funciona el sistema de detección y señalización de errores de merge en el Git y saber gestionarlo
c. Node-modules y packaje.json y packaje-lock.json: Estos son directorios y fixeros encargados de la gestión he instalación de dependencias del proyecto. Estos han sido los causantes de muchos de los problemas de backend debido a que git añade y quita o superpone dependencias a gusto y a diferencia de problemas anteriores no es material que pueda colocarse en git ignore porque eso implicaría que en cada pull se tendrían que editar las dependencias añadiendo o eliminando líneas. Además de que muchas librerías están capadas dependiendo de si el proyecto usa Expo o no. Este último problema si tenía fácil solución, fijarse bien antes de instalar alguna dependencia, no obstante, no sabíamos que había distinción entre react sin expo y react con expo lo que causo algunos problemas al principio del proyecto.
d. Index.js: Similar al problema que hubo en el frontEnd, en el intento de merge de la rama main con la rama api se eliminaron muchos de los archivos imprescindibles para el funcionamiento del proyecto. La solución fue similar al front, buscar un comit en el que estuviese todo y abortar el merge pero fue otro consumo de tiempo considerable.

3. Problemas generales: Problemas que no están relacionados con back ni front pero posiblemente han supuesto el mayor obstáculo para el proyecto.
    a. Tiempo: Este es posiblemente el más ovio, pero no por ello no podemos comentarlo. Todos los integrantes del equipo estamos de acuerdo en que no hay suficiente tiempo como para realizar una app en condiciones. Han quedado muchas funcionalidades por implementar, test por realizar y funcionalidades por depurar. Entendemos que el tiempo que hay es el mismo para todos los módulos, pero, sinceramente, creemos que crear una app no consume el mismo tiempo que lo que podrían ser otros proyectos de varios módulos. La solución a este problema no la tenemos del todo clara puesto que, si bien es cierto que se podría haber dedicado más tiempo propio o fuera de horario lectivo o empezar antes el proyecto, antes del horario fijado, también pensamos que el proyecto no debería estar pensado para tener que consumir horas no lectivas.
    b. Organización: Para todos los miembros del equipo este era el primer proyecto que se realizaba de una envergadura o tamaño consistente y por ende muchas decisiones de planificación que se tomaron al principio del proyecto, tales como estructuración de la base de datos, organización de pantallas y gestión de las peticiones al api, no fueron las más optimas y terminaron siendo algo contraproducentes para el desarrollo del proyecto. El equipo fue capaz de solucionar o mejorar algunas de estas soluciones antes de que llegar a implementarse o durante su implementación, pero también hubo algunas que no se

pudieron solucionar y se tuvieron que cargar hasta el final del proyecto.

c. Florida Expo: Entendemos que florida realice una feria de proyectos para promocionar a sus alumnos y a si misma, incluso no nos molesta la decisión de hacer obligatoria la entrada a algunos módulos, lo que si vemos innecesario es todo el trabajo de "marketing" que lleva detrás la feria, trabajos como el panfleto, el poster, las charlas o los videos promocionales, no el requerido para el modulo de Inglés Profesional sino uno específicamente requerido para la feria. En el quipo estamos muy ilusionados de participar en la feria y exponer el proyecto, pero al fin y al cabo somos programadores y tener que realizar todo este trabajo aparte de haber sido algo que se ha sentido bastante ajeno al módulo nos a acortado bastantes horas de trabajo en el proyecto en sí. Además, creemos que mucha de la información de la expo si podría haber facilitado antes, como los posters o el video que a pesar de no contar con nada físico del proyecto si se podían haber realizado con antelación.

3.5 Resultados obtenidos

A pesar de todos los problemas que se han comentado el equipo esta más que contento y satisfecho con el trabajo realizado y el producto final. Hubo momentos durante el desarrollo que, por como avanzaba el proyecto, pensamos que no daría tiempo a realizar muchas de las funcionalidades que pensábamos necesarias, pero finalmente y gracias a la gran inversión de tiempo propio el producto final si es una versión bastante cercana a lo que se tenia pensado en un primer momento. Además, más allá del proyecto "físico" o tangible, cada integrante del equipo a aprendido lecciones importantes no solo a nivel de programación o aptitudes relacionadas con el campo sino también actitudes relacionadas con la gestión de tiempo o el trabajo en equipo.

Conclusiones:

Para mayor facilidad separaremos las conclusiones obtenidas en 2, las obtenidas a nivel técnico del propio proyecto y las obtenidas a nivel personal o no tan relacionadas con el módulo

Ha nivel técnico se han aprendido a usar ambos framework de manera fluida, tanto React Native como SpringBoot, además de la base de datos de MongoDB. A todo esto, se le pude sumar la experiencia obtenida del trabajo con API y el diseño de interfaces o la gestión o distribución de datos, además de algo de experiencia en diseñar un plan de proyecto y como afrontar proyectos de tamaño más consistente.

Entre los aprendizajes que van más allá del apartado técnico del módulo podemos encontrar el trabajo en conjunto, un mejor control del tiempo y planificación de uso de este sobre todo para identificar que aspectos son clave en un proceso y tiene prioridad o cuales no. También podemos destacar capacidades de liderazgo y asunción de riesgos.

Trabajo a futuro:

Queda mucho por hacer si dirigimos la vista hacia el futuro, pues como ya se ha comentado en el apartado de problemas hay muchas funcionalidades que se tenían pensadas para realizar y no se han podido llevar a cabo.

A continuación, detallaremos con que cambios serian y que cambios supondrían en la estructura del proyecto:

- Implementación de la interfaz de alimentación: Esta seria una de las más secundarias, pero básicamente se trata de una pantalla en la que se mostrarían distintos dipos de recetas para bebés además de que cualquier usuario podría añadir una receta propia a esta lista, lo que implicaría una tabla compartida en la base de datos para todos los usuarios, una cualidad que actualmente no existe en la base de datos. Cada receta tendría una foto, ingredientes, descripción o procedimiento, titulo y nombre del usuario que la ha subido. La implementación de esta funcionalidad implicaría la creación de una nueva colección en la base de datos, así como su respectiva clase y "repositori" en la API junto con su pantalla y componentes en el frontend

- Registro de alimentación: Relacionado con la funcionalidad anterior este nuevo registro se encargaría de guardar junto a la fecha y la cantidad el alimento del bebé. Obviamente para la realización de esta funcionalidad es necesario la anterior por lo que las 2 deberían de realizarse en sincronía. De la misma forma que ya existen registros para sueño, tomas y crecimiento añadir uno más para alimentación. Su implementación seria sencilla y simplemente seria necesaria una nueva colección en la base de datos y añadir un atributo en el objeto bebé que guarde los diferentes registros.

- Mejorar el registro medico: Actualmente el registro médico simplemente almacena aspectos como medicamentos, duración de tratamiento y fechas. El objetivo sería que esta funcionalidad fuese mucho más reactiva, es decir que tomas note de el tratamiento actual en el que se encuentra el bebé, que tratamientos ya a pasado o si este es alérgico a algún medicamento. Es funcionalidad requeriría modificaciones tanto en el atributo medicalRecord como en su gestión en el frontend.

- Mejorar la calendarización: Uno de los aspectos más rigurosos para la crianza de un bebé es tomar nota o recordar todos los eventos o citas medicas obligatorias de las que requiere un bebé y crear un objeto de fechas obligatorias para que los usuarios no tengan que estar recordándolas o, aquellas que sean obligatorias, introducirlas manualmente. Esta funcionalidad seria puramente de frontend puesto que de la misma forma que se ha definido una línea de bebe promedio en el apartado de gráficos también sería necesaria la implementación de una funcionalidad similar en el calendario de evento

- Depurar las gráficas de crecimiento: Actualmente la funcionalidad de las gráficas es funcional porque si dibuja un gráfico dependiendo de los registros de crecimiento del bebé. El problema reside en que no es consistente y no muestra los datos con exactitud. Seguramente el problema resida más en la forma en la que se crean nuevos registros de crecimiento, siendo necesaria seguramente la implementación de una pantalla a parte para esta funcionalidad, así como también investigar de buena forma la librería utilizada para añadir dicha funcionalidad. Esta funcionalidad se implemento la última y fue con el objetivo de sentar la una base para futuro por eso es probable que sea necesario depurarla más que otras funcionalidades.

- Pantalla de configuración de usuario: Actualmente no hay muchas acciones ni de edición ni de personalización del perfil de usuario por lo que seria necesario añadir más, destacando una foto de perfil, posibilidad a cambiar de nombre de usuario, cambiar la contraseña y dar opción a eliminar el usuario. Además de depurar más en general la interfaz de datos del usuario.
- Depuración general de código: Muchas de las funciones y componentes que utiliza la app pueden estar mucho mejor optimizados, así como todo lo relativo a obtención de datos de la API.

- Códigos de error: Con el objetivo de hacer la aplicación mucho más óptima para usuarios no muy acostumbrados a las nuevas tecnologías seria necesario revisar los códigos de error que devuelve la API en la parte de frontend y realizar correcciones o avisos correspondientes en las interfaces

- Feedback de acciones: Si bien es cierto que algunas funcionalidades aportan una respuesta para que el usuario sepa que la app esta realizando acciones hay muchas otras que no y usando algunas librerías se podrían implementar respuestas a muchas otras acciones, acción que no parece hacer mucho por aportan una mejor experiencia de usuario.

- Canciones favoritas: En un principio se tenia pensado que cada usuario pudiese guardar una serie de canciones para poder reproducir. Al final se tuvo que cortar y cambiar por una sola canción predeterminada, no obstante añadir esta funcionalidad no solo permitiría dar una mayor profundidad a la aplicación en el sentido de la personalización sino también añadiría contenido a la pantalla de configuración de usuario. Para esto simplemente seria necesario crear un atributo en el usuario para guardar una colección de canciones favoritas y o bien crear una nueva colección en la base de datos donde almacenar cierta cantidad de imágenes y dejar al usuario elegir alguna o bien dar la opción para poder subir una canción a la app desde el dispositivo local.

- Identificación de síntomas visuales: Actualmente la única función que usa la cámara dentro de la aplicación es en la selección de la foto del bebé, no obstante, se tenía pensado que se pudiese sacar una foto de un síntoma visual

ya bien fuese algo como una mancha en la piel o un eccema y mandar dicha imagen a una API de IA para que generase una respuesta y la mandase al usuario. Se llegó incluso a buscar algunos sitios donde poder realizar la llamada, pero finalmente se termino por descartar la funcionalidad debido a la complejidad de la implementación y por la necesidad de priorizar otras funcionalidades.

- Multiusuario: También se tenia pensado hacer que, como es lógico, cada bebé tuviese más de un usuario asignado y ambos pudiesen interactuar con el bebé. Se barajaron algunas ideas para realizar esto, pero lo más probable es que un usuario “invitase” a otro a poder editar o gestionar el bebé vía la app. No obstante, debido a la complejidad y el desconocimiento de como llevar a cabo tal funcionalidad se decidió descartar del proyecto final.

- Exportación de datos a informe pediátrico: Para terminar con su funcionalidad de asistente de recogida de datos se tenia pensado implementar una funcionalidad que transformara todos los datos de la aplicación en un informe legible para un pediatra o de valor médico, pero se quedó fuera del proyecto final por falta de tiempo. Seguramente haría falta alguna biblioteca externa pero no tenemos del todo claro si sería una función de frontend o backend