UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

BreathBank

Presentado por Eduardo García de Leániz Juarros
en Universidad de Burgos — 3 de julio de 2025
Tutora: Ana Serrano Mamolar

Resumen

En el contexto actual, caracterizado por un ritmo de vida acelerado y una creciente dependencia de la tecnología, se observa un incremento notable en los niveles de estrés, ansiedad y otros trastornos vinculados al bienestar físico y mental. Esta situación, potenciada por la constante demanda de productividad y la hiperconectividad, plantea nuevos desafíos en cuanto al cuidado de la salud individual.

Para dar solución a este problema, este proyecto se centra en la promoción de estrategias accesibles y eficaces para contrarrestar los efectos negativos de este estilo de vida. En particular, se propone el desarrollo de *BreathBank*, una aplicación móvil orientada a facilitar la práctica diaria de la respiración consciente. Esta técnica, respaldada por diversas investigaciones científicas, ha demostrado ser eficaz en la mejora de funciones fisiológicas clave, así como en la regulación emocional y la reducción del estrés.

De esta manera, el proyecto no solo busca ofrecer una herramienta tecnológica útil y de fácil acceso, sino también evidenciar cómo la tecnología, empleada de manera adecuada, puede convertirse en un recurso positivo para mejorar la calidad de vida en un entorno cada vez más demandante.

Descriptores

Respiración consciente, aplicación, *full-stack*, *Flutter*, *Firebase*, bienestar, *mindfulness*, hiperconectividad.

Abstract

In today's context, marked by a fast-paced lifestyle and an increasing reliance on technology, there has been a notable rise in stress, anxiety, and other conditions related to physical and mental well-being. This situation, driven by constant demands for productivity and hyperconnectivity, presents new challenges for individual health care.

To address this issue, this project focuses on promoting accessible and effective strategies to mitigate the negative effects of this lifestyle. Specifically, it proposes the development of *BreathBank*, a mobile application designed to facilitate the daily practice of conscious breathing. This technique, supported by various scientific studies, has proven effective in improving key physiological functions, as well as in regulating emotions and reducing stress.

In this way, the project aims not only to provide a practical and user-friendly technological tool, but also to demonstrate how technology, when used appropriately, can become a positive resource for enhancing quality of life in an increasingly demanding environment.

Keywords

Conscious breathing, application, full-stack, *Flutter*, *Firebase*, well-being, mindfulness, hyperconnectivity.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

La velocidad y el ritmo de vida que lleva la sociedad hoy en día es incomparable al que podían llevar hace unas décadas. Este aumento exponencial en las responsabilidades y en la inmediatez ha sido impulsado principalmente por el desarrollo de nuevas tecnologías, que no solo permiten esta interconexión global constante sino que muchas veces la exigen para no quedarse atrás. Esto causa que cada vez más personas sufran de estrés, ansiedad y otras consecuencias asociadas.

Es evidente que es muy difícil reducir las obligaciones o el tiempo de conexión a ciertos medios digitales, tanto en el tiempo de trabajo como en el de ocio. Pero lo que sí se puede hacer es transformar el uso de estos medios en beneficio de nuestra salud física y mental. Este proyecto pretende no solo proporcionar una herramienta práctica para cumplir con este objetivo, sino también demostrar el impacto positivo que la tecnología puede tener en el cuidado de la salud personal.

Desde hace algunos años, los profesionales del área de la salud son conocedores de los beneficios de la práctica de la respiración consciente [27]. El ideario de este proyecto es uno de ellos que además detectó la necesidad de una herramienta que acercara y facilitara esta práctica a sus pacientes. Entre algunas de esas ventajas se encuentran:

- Mejora de la capacidad pulmonar y respiratoria.
- Disminución del estrés y de la ansiedad.
- Reducción de la irascibilidad y mejora del descanso nocturno.
- Mejora en las funciones cardíaca, circulatoria y digestiva.

- Aumento de la claridad mental y de la capacidad de concentración.

BreathBank es una aplicación para dispositivos móviles que pretende ser un instrumento al alcance de la mano, para que los usuarios, de una forma personalizada, puedan desconectar y relajarse por unos instantes de este mundo tan frenético que nos rodea. Ser conscientes de nuestra respiración y ejercitarla diariamente aporta no solo beneficios a nivel físico, como aumentar la capacidad pulmonar, estabilizar el Sistema Nervioso Central o fortalecer el Sistema Inmunitario, sino también a nivel psicológico, como alcanzar una mayor capacidad de concentración, reducir la irascibilidad y aumentar la energía y la vitalidad.

1.1. Estructura de la memoria

La documentación de este proyecto se divide en dos archivos relacionados: memoria y anexos. La memoria se compone de las siguientes secciones:

- **Introducción:** Pequeña explicación del tema del proyecto, presentación de la herramienta desarrollada y estructura de la documentación y materiales adjuntos.
- **Objetivos del proyecto:** Conjunto de propósitos a satisfacer tras la realización del proyecto.
- **Conceptos teóricos:** Exposición de nociones y términos necesarios para comprender el tema del proyecto y el funcionamiento de la herramienta desarrollada.
- **Técnicas y herramientas:** Medios técnicos y recursos empleados durante el proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** Descripción de las decisiones importantes, dificultades encontradas, opciones consideradas y soluciones adoptadas durante las distintas fases de desarrollo.
- **Trabajos relacionados:** Propuestas, artículos o trabajos relacionados con el tema y que han servido de inspiración para la realización de este proyecto.
- **Conclusiones y líneas de trabajo futuras:** Áreas de mejora y posibles desarrollos a futuro de la herramienta, y reflexiones finales sobre el proyecto realizado.

1.2. Estructura de los anexos

Los anexos recogen la información más específica del proyecto, dividiéndola en los siguientes apartados:

- **Plan de Proyecto:** Cronograma del proyecto, organización temporal de las tareas realizadas y estudios de viabilidad económica y legal.
- **Requisitos:** Listado de requisitos a cubrir y exposición de casos de uso.
- **Diseño:** Especificación del modelo de datos y arquitectura utilizados, y sus procedimientos asociados.
- **Manual de Usuario:** Guía con la información necesaria para poder utilizar la aplicación, sin entrar en detalles de la implementación.
- **Manual de Programador:** Guía que recoge todos los aspectos importantes sobre el código, implementaciones, estructura y todos los pasos necesarios para replicar la herramienta y poder continuar con su desarrollo.

1.3. Materiales adjuntos

- Repositorio que contiene todo el desarrollo del proyecto, la documentación y el *apk* de instalación. (Repositorio BreathBank)
- Enlace a una *demo online* de tres minutos para probar *BreathBank* sin ningún tipo de descarga. (Demo BreathBank)

2. Objetivos del proyecto

2.1. Objetivos generales

- Desarrollar una aplicación móvil híbrida conectada a un *backend* para ayudar a los usuarios a adquirir el hábito de realizar respiraciones conscientes.

- Ofrecer al usuario un seguimiento personalizado y ejercicios para mejorar sus habilidades respiratorias.

- Almacenar los datos de los usuarios de manera organizada y responsable, con el objetivo de representar su progreso y estadísticas de uso.

- Proporcionar al usuario una herramienta sencilla que no requiera un gran conocimiento técnico para su uso.

2.2. Objetivos técnicos

- Llevar a cabo el desarrollo de una aplicación *full-stack*, teniendo en cuenta todos sus componentes: *backend*, *frontend*, infraestructura y lógica de negocio.

- Implementar la arquitectura *MVC (Modelo-Vista-Controlador)* en el desarrollo de la aplicación, con el fin de estructurar el código de forma modular, facilitar el mantenimiento y separar claramente la lógica de negocio de la interfaz de usuario.

- Desarrollar una interfaz intuitiva y cómoda para que el usuario se centre únicamente en los contenidos y no en el funcionamiento de la aplicación.
- Integrar un sistema de *backend* en la aplicación, a través de *Firebase* (Sección 4.6), para gestionar el sistema de autenticación de usuarios y almacenar datos relativos a los usuarios y su interacción con la app de forma segura.
- Utilizar *Flutter* (Sección 4.5) como *framework* de desarrollo para integrar en una misma base de código todas las herramientas necesarias para construir la aplicación, optimizando tiempos y recursos de desarrollo.
- Integrar la herramienta *SonarQube* (Sección 4.7) en el proceso de desarrollo para analizar automática y objetivamente la calidad del código fuente, identificando errores, vulnerabilidades de seguridad o malas prácticas.
- Aplicar la metodología ágil *Scrum* (Sección 4.1) durante el desarrollo del proyecto para su organización.
- Utilizar *GitHub* (Sección 4.2) como plataforma de control de versiones y colaboración, junto con *GitHub Projects* (Sección 4.3) para organizar y dar seguimiento al desarrollo del proyecto mediante tableros, facilitando una gestión ágil y ordenada de tareas.

2.3. Objetivos personales

- Fomentar mi crecimiento personal al trabajar en un proyecto que promueva el bienestar de las personas, desarrollando herramientas que ayuden a mejorar su salud mental y física.
- Desarrollar mi capacidad para estructurar y organizar proyectos de *software*, mejorando la modularidad y mantenimiento del código.
- Mejorar mi capacidad para resolver problemas complejos durante el proceso de desarrollo de *software*, enfrentando desafíos técnicos y encontrando soluciones eficaces.
- Adquirir experiencia práctica en la gestión de la privacidad y seguridad de los datos de los usuarios, especialmente en el contexto de aplicaciones que requieren el almacenamiento de información sensible.

- Aumentar mi comprensión de la importancia de la usabilidad y la experiencia del usuario en el diseño de aplicaciones móviles, garantizando que la interfaz sea intuitiva y fácil de usar.
- Poner en práctica los conocimientos adquiridos durante la carrera y las prácticas en empresa.

3. Conceptos teóricos

Para una mejor comprensión del tema y del contexto donde se desarrolla este proyecto, se han recogido varios conceptos, con enfoques teóricos y técnicos, junto con sus explicaciones.

3.1. Conceptos relacionados con el ámbito de la aplicación

Respiración consciente

Es aquella en la que nuestra mente y nuestros pensamientos están enfocados en la propia respiración. La intención puede ir desde únicamente ser consciente de nuestra respiración, a modificarla también conscientemente, aumentando o disminuyendo la intensidad y la duración de la inspiración y de la espiración, o la duración de las pausas entre ellas. De aquí en adelante, todas las respiraciones que se comentan son de este tipo.

Inversión

En el contexto de la aplicación, se denomina ***inversión*** al ejercicio o prueba que realiza el usuario de la aplicación con el objetivo de mejorar sus capacidades pulmonares y de obtener los beneficios de la respiración consciente. El término hace referencia a la inversión de un bien, como el tiempo, para recoger en un futuro resultados favorables en el ámbito personal.

Inversor

Recibe el nombre de ***inversor*** cada usuario de la aplicación *BreathBank* que destina su tiempo a la práctica de respiraciones conscientes con el objetivo de obtener beneficios en su bienestar físico y mental. La práctica de respiraciones que se incluye en esta aplicación ha sido respaldada por estudios científicos.

Niveles de inversor

Dentro de la aplicación se definen diferentes niveles de inversor. Cada nivel viene determinado por la capacidad del inversor de respirar de forma más lenta, buscando aumentar el tiempo que puede inspirar y espirar de forma continuada sin que aparezca tensión en el cuerpo ni en la mente. La pertenencia de cada usuario a un ***nivel de inversor*** u otro, dependerá de su rendimiento en las pruebas de evaluación. Estos niveles están diseñados para adaptarse progresivamente a las capacidades respiratorias de cada inversor, que irá subiendo de nivel según la experiencia que vaya adquiriendo y a la mejora en su práctica diaria. Pertenecer a un nivel u otro condiciona la duración de las respiraciones durante las inversiones.

Saldo

Cada inversor contará con un ***saldo*** acumulativo donde se irán sumando el número de respiraciones que vaya realizando en cada inversión. Este saldo permitirá realizar evaluaciones para subir de nivel. El número de respiraciones del usuario se restablecerá después de cada evaluación.

Listón de la inversión

A la hora de realizar una inversión y dependiendo del nivel de inversor del usuario, éste podrá establecer una mayor o menor dificultad al ejercicio seleccionando un ***listón de inversión*** más alto o más bajo. Este listón influye en la duración de las respiraciones durante la inversión. Cada nivel permite establecer unos listones diferentes, y estos listones están asociados a respiraciones más largas aumentando la dificultad.

Tipo de inversiones

Se ofrecen dos formas de realizar inversiones dentro de la aplicación. En ambos casos, la duración de las respiraciones vendrá marcada por el nivel de inversor y por el listón de la inversión elegido.

- ***Manual:*** en la ***inversión manual*** el usuario realizará las respiraciones a su ritmo, sin estar pendiente del dispositivo y al final introducirá las respiraciones realizadas. Para facilitar este ejercicio, durante la realización del mismo, el usuario podrá pulsar en la pantalla cada vez que cambie de inspiración a exhalación. De esta forma la aplicación contabilizará las respiraciones que ha realizado y lo mostrará automáticamente al finalizar el tiempo.

- ***Guiada:*** en la ***inversión guiada*** el usuario tendrá que seguir el ritmo marcado de respiraciones. La aplicación indicará el progreso y si el usuario debe inspirar o exhalar en cada momento. Además incluye una ayuda sonora que indica los cambios de una fase de la respiración a la otra para facilitar el ejercicio.

Evaluación de condiciones

Para determinar el nivel inicial del inversor y seguir evaluando a medida que practique con la aplicación, el usuario deberá realizar una evaluación de sus condiciones o habilidades respiratorias. Esta ***evaluación de condiciones*** se compone de tres pruebas. Los resultados obtenidos por el usuario en dichas pruebas determinarán el nuevo nivel adquirido. La realización de estas evaluaciones estará sujeta a la siguiente condición: solo podrán llevarse a cabo cuando el usuario haya alcanzado el saldo establecido por la aplicación. Tras cada evaluación, el saldo se restablecerá a cero.

Término BreathBank	**Analogía**
Inversión	Ejercicio
Inversor	Usuario
Nivel de Inversor	Nivel de usuario
Listón de Inversión	Dificultad del ejercicio
Saldo	Progreso actual
Evaluación	Examen

Tabla 3.1: Analogía entre conceptos de la lógica de negocio y la aplicación

3.2. Conceptos sobre el *software*

Aplicación Híbrida

Una ***aplicación híbrida*** es un tipo de *software* móvil que combina elementos de aplicaciones nativas y aplicaciones web. Se desarrolla utilizando tecnologías web estándar pero se empaqueta dentro de un contenedor nativo que le permite acceder a funcionalidades del dispositivo. Esto permite que la aplicación se ejecute en múltiples plataformas como *Android* e *iOS* sin necesidad de escribir el código desde cero para cada una.

Widget

Un ***widget*** es un componente visual e interactivo que forma parte de la interfaz de usuario de una aplicación. En el contexto del desarrollo de *software*, especialmente en *frameworks* como *Flutter*, los *widgets* son los bloques fundamentales a partir de los cuales se construyen las interfaces. Cada elemento visible en pantalla, como botones, textos, imágenes o contenedores, es un *widget*. Los *widgets* permiten reutilizar código y componer interfaces de forma modular, facilitando el diseño y la personalización de la experiencia del usuario.

Emulador

Un ***emulador*** es un *software* que permite simular el entorno de otro sistema operativo o dispositivo. En el desarrollo de aplicaciones móviles, los emuladores son utilizados para ejecutar y probar aplicaciones sin necesidad de un dispositivo físico, imitando las características de un teléfono o *tablet*. Esto facilita las pruebas en diferentes dispositivos y versiones del sistema operativo.

Frontend / Interfaz

El ***frontend*** es el medio o mecanismo a través del cual los usuarios o sistemas interactúan con un *software* o *hardware*. A través de ella se produce el intercambio de información entre la persona que la utiliza y el programa que esté utilizando.

Backend

El ***backend*** es la parte de la aplicación que gestiona el procesamiento de datos y la lógica de negocio. Aunque no es visible para el usuario, es fundamental para que la aplicación funcione correctamente. El *backend* se encarga de tareas como el manejo de bases de datos, la autenticación de usuarios y la integración con otros servicios.

Persistencia

La ***persistencia*** en el desarrollo de aplicaciones móviles se refiere a la capacidad de guardar datos de manera permanente o a largo plazo, de modo que sobrevivan a la ejecución y parada de la aplicación. Esto es crucial para aplicaciones que requieren almacenamiento de información, como configuraciones de usuario, historial o datos entre sesiones. Se puede lograr mediante bases de datos locales o servicios de almacenamiento en la nube.

Base de datos no relacional

Una ***base de datos no relacional*** *(NoSQL)* es un tipo de base de datos que no sigue el modelo de tablas y relaciones de las bases de datos tradicionales. Son útiles cuando se manejan grandes volúmenes de datos no estructurados. Este tipo de bases de datos permiten almacenar datos en formatos más flexibles como documentos, clave-valor o grafos, lo cual es común en aplicaciones móviles que manejan datos dinámicos.

Colecciones

En el contexto de bases de datos, una ***colección*** es una estructura que agrupa un conjunto de documentos o registros. En bases de datos *NoSQL* las colecciones son similares a las tablas en bases de datos relacionales, pero no requieren un esquema fijo. Se utilizan para almacenar y organizar datos de forma flexible, lo que se ajusta muy bien para aplicaciones móviles que necesitan manejar datos dinámicos y cambiantes.

Concurrencia

La ***concurrencia*** en este contexto hace referencia a la capacidad de la aplicación de ejecutar múltiples tareas de forma aparentemente simultánea. No tienen por qué ejecutarse exactamente en el mismo instante, pero sí

solaparse en su ejecución. Este concepto es relevante, por ejemplo, cuando el usuario interactúa con la interfaz mientras el sistema valida sus credenciales, o cuando se están guardando datos en la base de datos mientras se permite la navegación entre pantallas.

Autenticación de usuarios

La ***autenticación de usuarios*** es el proceso que garantiza que solo los usuarios autorizados puedan acceder a ciertos recursos dentro de la aplicación. Esto se logra normalmente mediante un sistema de credenciales, como un nombre de usuario y una contraseña, aunque también se pueden utilizar métodos más avanzados, como la autenticación multifactor o biométrica.

Programación reactiva

La ***programación reactiva*** es un enfoque que se centra en gestionar flujos de datos y eventos de manera eficiente. En aplicaciones móviles, la *programación reactiva* permite que la interfaz de usuario reaccione automáticamente a los cambios en los datos, como la actualización de información en tiempo real. Patrones de diseño como el Estado o el Observador, son de mucha utilidad a la hora de implementar estas características.

API

Una ***API*** (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite que diferentes programas se comuniquen entre sí. Las *APIs* son fundamentales para integrar servicios externos, como sistemas de autenticación, sistemas gestores de bases de datos, etc. Permiten que la aplicación se conecte a otras plataformas de manera eficiente y controlada.

Debugging

El ***debugging*** es el proceso de identificar y corregir errores en el código de una aplicación. Normalmente, los editores de código (como *Visual Studio Code* [33]) siempre incorporan esta opción para facilitar el desarrollo del *software*. Estas herramientas de depuración sirven para revisar el flujo de la aplicación, inspeccionar variables y ver cómo se comporta el código durante la ejecución.

Hot Reload

El ***Hot Reload*** es una característica que permite ver los cambios del código de la aplicación de manera instantánea en un *emulador*, sin necesidad de reiniciar la aplicación. Esto mejora la productividad de los desarrolladores al reducir el tiempo de espera entre modificaciones en el código y su visualización en el dispositivo. En plataformas como *Flutter* (4.5), el *Hot Reload* facilita la iteración rápida en el proceso de desarrollo de aplicaciones móviles.

Dependencias

Se define ***dependencia*** a cualquier biblioteca, módulo o servicio externo necesario para que funcione correctamente una aplicación o para añadir funcionalidades específicas. Para incluir estos elementos, suele ser necesario modificar la configuración de arranque de la aplicación y descargar ciertos archivos en los que estén contenidas las herramientas que se quieren importar.

Test

Un ***test*** es una prueba encargada de verificar el correcto funcionamiento de una parte específica del *software* en función del comportamiento esperado. En el *framework* que se ha utilizado en este proyecto, *Flutter*, se pueden crear tres tipos distintos de test [21] en función de la naturaleza de lo que queramos testear.

Test unitarios

Los test unitarios se utilizan para verificar que funciones, métodos o clases individuales se comporten de manera correcta y esperada de forma aislada. Su objetivo principal es validar la lógica interna de componentes específicos sin involucrar otras partes de la aplicación. Este tipo de pruebas se realizan cuando se comienzan a crear nuevas clases, detectando errores tempranos de manera más sencilla

Test de *widgets*

Los test de *widgets* están diseñados para comprobar que los componentes de la interfaz gráfica funcionen correctamente cuando se integran con el *framework* de *Flutter*. Estas pruebas permiten verificar tanto la estructura como el comportamiento visual de los *widgets*, evaluando aspectos como la disposición de elementos, el estado interno de los mismos y las interacciones

del usuario. Este tipo de test son más difíciles de probar y suelen presentar más dificultades cuando se aplican a interfaces ya finalizadas.

Test de integración

Los test de integración se enfocan en evaluar el comportamiento completo de la aplicación o de flujos funcionales específicos, simulando la experiencia real del usuario.

Mock

Un *mock* es un objeto simulado que imita el comportamiento de dependencias externas (como una base de datos, una *API* u otro tipo de servicio), de forma que puedes probar una parte específica de tu código sin necesidad de ejecutar toda la aplicación ni depender de servicios reales. En este caso, para testear gran parte de la aplicación, era necesario simular los servicios de *Firebase* mediante objetos *mock* para así evitar inicializar ese servicio en cada test.

4. Técnicas y herramientas

4.1. Metodologías

Scrum

Scrum [4] es una metodología de trabajo ágil utilizada, principalmente, para el desarrollo de *software*, aunque también se aplica en otros contextos. Su propósito es mejorar la colaboración, la flexibilidad y la entrega continua de valor en equipos de trabajo. En el caso de este proyecto, la metodología *Scrum* ha sido adaptada al contexto y características del mismo.

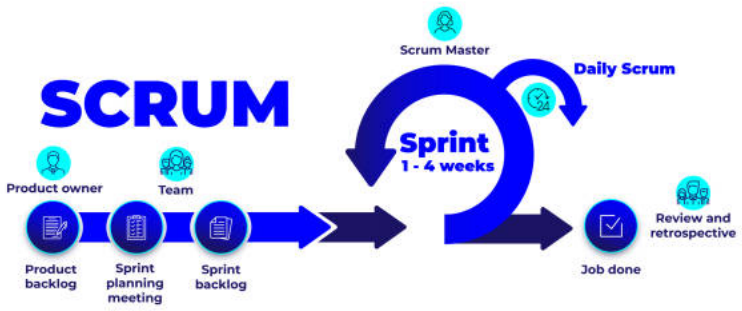

Figura 4.1: Metodología *Scrum* (Imagen extraída de: *istockphoto*)

Roles

Para llevar a cabo esta metodología, se identifican distintos roles con sus tareas asociadas:

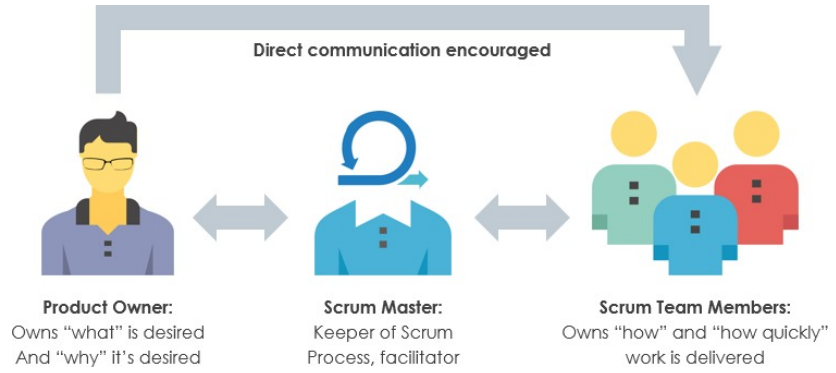

Figura 4.2: Roles *Scrum* (Imagen extraída de: *visual-paradigm*)

- **Product Owner:** Es el encargado de representar al cliente y asegurar que se cumplen sus intereses. En este proyecto ese papel ha sido tomado por el ideador de *BreathBank* junto con la tutora de este TFG, los cuales han velado porque las necesidades que querían cubrir con esta aplicación de verdad se cumplieran.
- **Scrum Master:** Es la persona que dirige y coordina el desarrollo del proyecto. En este caso esa figura la toma totalmente la tutora de este TFG.
- **Equipo Scrum o equipo de trabajo:** Formado por el conjunto de desarrolladores, ingenieros, técnicos, etc. que llevan a cabo el producto final. En el proyecto de *BreathBank* no ha sido un equipo, sino que todo el trabajo de desarrollo, pruebas, implementación, servicios, etc. recayó sobre el alumno.

Elementos *Scrum*

Son componentes fundamentales para la aplicación de este marco de trabajo. Se consideran artefactos a los objetos que se generan y utilizan como medios para abordar y cubrir los desafíos y requisitos del proyecto.

Figura 4.3: Elementos *Scrum* (Imagen extraída de: *blog.buhoos*)

- ***Product Backlog:*** Lista de trabajo que recoge todas las tareas a finalizar para obtener el producto final y finalizar el proyecto. Estas tareas deben estar supervisadas por el *Product Owner*, pero en este proyecto fueron añadidas en su gran mayoría al principio por el alumno, aunque otras ciertas tareas tuvieron que ir añadiéndose según avanzaba el proyecto para cubrir imprevistos y errores que iban surgiendo.

- ***Sprint Backlog:*** Subconjunto de tareas del *Product Backlog* elegidas para completar durante el *sprint* actual. Este *Backlog* deberá definirse teniendo en cuenta la cantidad de trabajo que se puede asumir en función del tiempo y de los recursos de los que se dispone. En este proyecto concreto, el *Sprint Backlog* se consensuaba entre la tutora y el alumno en cada reunión de *sprint*.

- ***Incremento:*** Resultado del producto al finalizar cada *sprint*. En el entorno profesional se requiere que este resultado sea funcional para enseñar los avances al cliente. En este proyecto, no siempre terminaba una versión funcional de la aplicación después de cada *sprint*, puesto que había momentos más enfocados al desarrollo *software* y sin embargo otros más enfocados a la documentación o a la investigación, aunque se procuraba equilibrar ambas antes de cada reunión.

Beneficios de usar *Scrum* en este proyecto de desarrollo

La adaptación de la metodología *Scrum* al contexto específico de este proyecto ha supuesto una serie de beneficios significativos que han contribuido positivamente al desarrollo y a la organización del trabajo:

- **Mejor organización y planificación del trabajo:** El uso del *Product Backlog* y del *Sprint Backlog* permitió estructurar y priorizar las

tareas de forma clara, ayudando a mantener el foco en lo más relevante en cada momento según pasaban las semanas.

- **Seguimiento continuo del progreso:** Gracias a los *sprints* quincenales y a las reuniones de planificación y revisión, se pudo hacer un seguimiento constante de los avances y ajustar la dirección del proyecto según las necesidades e inconvenientes que iban surgiendo.
- **Mayor capacidad de adaptación:** Al no tratarse de un entorno con requisitos totalmente cerrados desde el inicio, el enfoque ágil permitió incorporar nuevas tareas y resolver imprevistos de forma flexible, sin que esto supusiera una disrupción grave en el desarrollo.
- **Refuerzo de la autoevaluación y la autogestión:** La práctica diaria de revisar el estado del *Sprint Backlog* permitió al alumno mejorar su capacidad de autogestión, planificar sus jornadas de trabajo con mayor eficiencia y tomar decisiones informadas en función del estado del proyecto.
- **Equilibrio entre desarrollo técnico y documentación:** La estructura cíclica de *Scrum* facilitó encontrar un balance adecuado entre las distintas fases del proyecto (desarrollo, investigación, documentación), asegurando que ninguna de ellas quedara desatendida, y fuesen avanzando todas de manera equilibrada.

Elemento	**Descripción**
Product Owner	Ideador del proyecto + Tutora
Scrum Master	Tutora del TFG
Equipo de desarrollo	Alumno (desarrollador único)
Duración del Sprint	14 días
Sprint Meeting	Reunión conjunta (30' revisión + 30' planificación)
Daily Scrum	Revisión individual diaria (10' - 15')
Product Backlog	Ajustado durante el desarrollo
Sprint Backlog	Consensuado en cada Sprint con la tutora
Incremento	Depende del enfoque del Sprint (desarrollo o documentación)

Tabla 4.1: Resumen de la aplicación de *Scrum* en el proyecto

4.2. *Hosting* del repositorio – *GitHub*

El repositorio del proyecto se ha alojado en la plataforma *GitHub* [12], una de las herramientas más utilizadas a nivel mundial para el control de versiones y colaboración en proyectos de *software*. Esta plataforma es sencilla, fácil de utilizar y muy integrable con distintos editores de código, como *Visual Studio Code*, para según vas desarrollando poder almacenar los cambios y tenerlo todo siempre actualizado. Solo con vincularlo con su herramienta *Github Desktop*, se pueden visualizar los cambios de todos los archivos y confirmarlos o revertirlos con apenas dos clicks.

Gracias a su interfaz intuitiva y funcionalidades avanzadas, *GitHub* ha permitido mantener el proyecto bien estructurado, facilitando la gestión del código fuente, la documentación y otros archivos asociados.

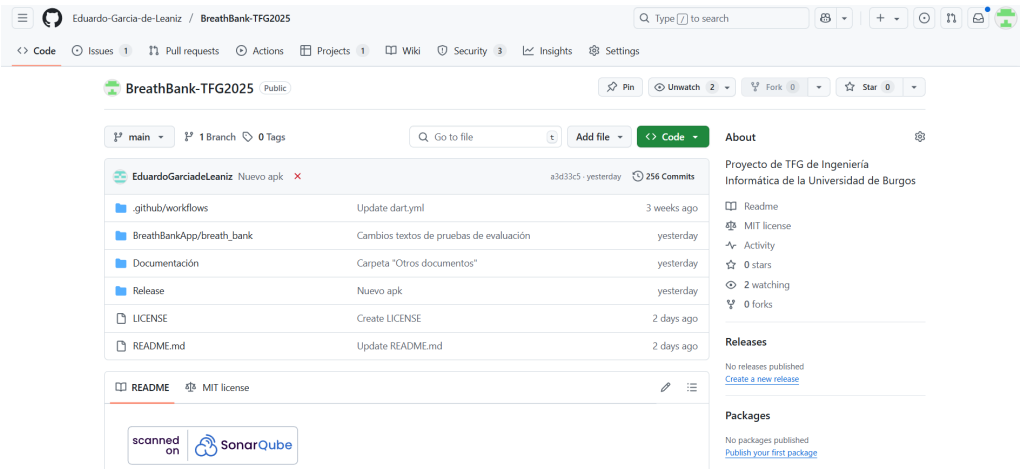

Figura 4.4: Repositorio de GitHub

4.3. Gestión del proyecto – *GitHub Projects*

Para la planificación y seguimiento de las tareas del proyecto, se ha utilizado la funcionalidad de *GitHub Projects*, una herramienta integrada en GitHub que permite crear tableros *Kanban* personalizados. Ha sido realmente útil para llevar a cabo la metodología *Scrum* (Sección 4.1), clasificar las tareas según su estado (por hacer, en progreso, finalizadas) y vincular directamente las tareas con *issues* y *milestones*, para una mejor organización de las mismas.

El uso de esta herramienta ha contribuido a mantener una visión clara del progreso general del proyecto, facilitando tanto la planificación de los *sprints* como el seguimiento de tareas pendientes o en curso. Además, su integración directa con el repositorio ha simplificado la trazabilidad de las decisiones técnicas y los avances conseguidos.

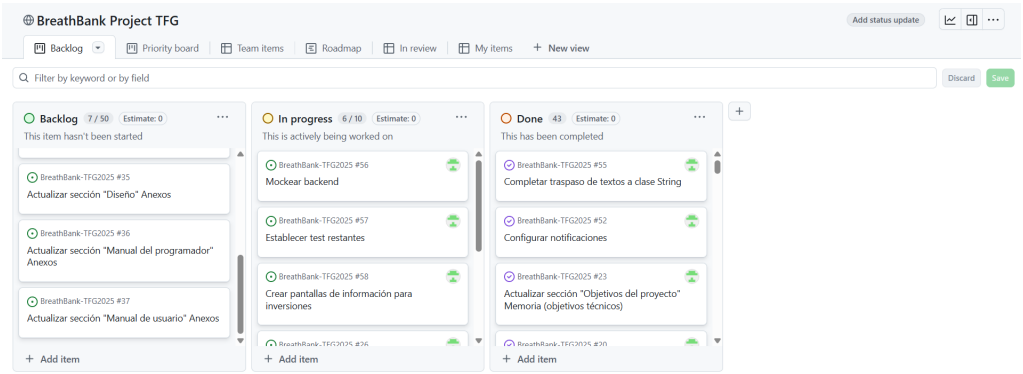

Figura 4.5: GitHub Projects

4.4. Comunicación – *Outlook* y *Microsoft Teams*

La comunicación, como en cualquier proyecto, siempre es una parte fundamental. En este caso los canales utilizados han sido dos. Las dudas, respuestas y avisos entre alumno y tutora se realizaban a través de *Outlook*, mientras que las reuniones telemáticas se organizaban a través de *Microsoft Teams*. Posteriormente, las reuniones pasaron a ser presenciales reduciendo el uso de *Teams* a partir de la mitad del desarrollo del proyecto.

4.5. Entorno de Desarrollo Integrado (IDE), *framework* y lenguaje de desarrollo

Visual Studio Code y extensiones

La elección del editor de código fuente ha cambiado a lo largo del desarrollo de este proyecto. Primero se comenzó utilizando *Android Studio* ya que se pensó que quizá estaría un poco más integrado con el trabajo con aplicaciones móviles. Sin embargo, el no tener experiencia utilizándolo y el

bajo rendimiento que daba en el equipo con el emulador y a la hora de hacer *debug*, llevó a cambiar de opinión y sustituirlo por ***Visual Studio Code***.

Este editor ofrecía mejor rendimiento, además de que ya tenía experiencia trabajando con él en proyectos anteriores. Además, para trabajar con *Flutter* solo eran necesarias instalar tres extensiones: ***Flutter, Dart y Awesome Flutter Snippets***. Otras partes de la configuración, como es la instalación de *Flutter*, el sistema de *backend*, y la integración del propio *backend* en el *framework* de desarrollo sí que fueron bastante más complejas.

Flutter

En el proceso de desarrollo de *BreathBank*, se evaluaron diversas opciones de *frameworks* que permitieran construir la aplicación para múltiples plataformas de manera eficiente. Sin embargo, se decidió optar por *Flutter* [16] por una serie de características y ventajas que se comentan a continuación.

- **Desarrollo multiplataforma con un solo código base:** Aunque *BreathBank* ha sido desarrollada solo para *Android*, *Flutter* ofrece la posibilidad de desarrollar para múltiples plataformas al mismo tiempo, reduciendo significativamente el tiempo de desarrollo.

  Flutter permite escribir el código una vez y ejecutarlo en ambas plataformas sin comprometer la funcionalidad o la experiencia del usuario, lo que es una ventaja clara frente a soluciones que requieren mantener bases de código separadas para cada plataforma.

- **Alto rendimiento en desarrollo y compilación:** A diferencia de otros *frameworks* que dependen de un puente o capas intermedias para comunicar el código con los sistemas operativos nativos, *Flutter* compila directamente a código nativo. Esto significa que las aplicaciones desarrolladas con *Flutter* tienen un rendimiento muy cercano al de aplicaciones nativas. Además, funciona de manera bastante rápida a la hora de mostrar los cambios en el dispositivo o en el emulador que se esté probando, aunque esto también se abordará en el apartado de *Hot Reload*.

- **Control sobre la UI y widgets personalizados:** *Flutter* se basa en un sistema de *widgets* para la creación de interfaces de usuario. Cada componente de la interfaz, desde botones hasta animaciones, es un *widget*, lo que ofrece un control totalmente personalizado sobre el diseño y el comportamiento de la aplicación.

- **Hot Reload: Desarrollo Ágil y Eficiente:** Una de las características más valoradas por los desarrolladores de *Flutter* es el *Hot Reload* (Sección 3.2). Esta funcionalidad acelera enormemente el proceso de desarrollo y permite realizar pruebas e iteraciones rápidas sobre el diseño y la funcionalidad.
- **Fácil integración con sistemas de *backend*:** *Flutter* destaca también por su fácil integración con distintas plataformas que ofrecen servicios de *backend*. Esto permite que con una rápida configuración, la aplicación pueda estar conectada a bases de datos, sistemas de autenticación, servidores de notificaciones...

Dart

Dart [14] es un lenguaje de programación optimizado para aplicaciones móviles, web y de escritorio. Fue desarrollado por *Google* con el objetivo de proporcionar una plataforma eficiente para la creación de aplicaciones modernas. *Dart* es conocido por ser el lenguaje que utiliza el *framework Flutter* para el desarrollo de aplicaciones.

- **Sintaxis moderna y reconocible:** *Dart* tiene una sintaxis que se asemeja a otros lenguajes como *JavaScript, C++ o Java*, lo que facilita su aprendizaje para programadores con experiencia en estos lenguajes.
- **Orientado a objetos:** *Dart* es un lenguaje orientado a objetos, lo que significa que utiliza clases y objetos para organizar el código y la lógica de las aplicaciones.
- **Asincronía y Concurrencia:** *Dart* proporciona soporte para programación asincrónica mediante *async y await*, facilitando la escritura de código no bloqueante y optimizando el rendimiento en tareas concurrentes (Sección 3.2). Esto es muy útil a la hora de implementar sistemas de autenticación o en operaciones de lectura o escritura en base de datos.
- **Tipado estático:** *Dart* permite un tipado estático opcional, lo que ayuda a detectar errores en tiempo de compilación y mejora la mantenibilidad del código.

Overleaf

Respecto a la redacción de la memoria y sus anexos, *Overleaf* [26] resultó ser una herramienta muy cómoda. Hay múltiples editores de archivos de

LaTex, ya que es un lenguaje cada vez más utilizado en documentos oficiales gracias a su control preciso de formato, modularidad, gestión automática de referencias y bibliografía, etc.

Algunos de los motivos por los que me incliné por *Overleaf* son: acceso desde cualquier lugar y sin necesidad de instalación, colaboración en tiempo real y compilación automática mientras editas.

Aunque el del repositorio se llevó a través de *Github* (Sección 4.2), el control de cambios y versiones de la memoria y demás documentación se llevó a través de *Overleaf* por comodidad y rapidez en las correcciones.

4.6. Servicios de *backend*

Uno de los pilares fundamentales dentro del desarrollo de aplicaciones es **la gestión y el almacenamiento de los datos** que vaya a utilizar la misma. Por ello, es importante estudiar las opciones que hay e identificar cual se adapta mejor a cada, en función de sus ventajas e inconvenientes.

Comparación entre distintos SGBD

En el mundo del desarrollo de aplicaciones, elegir el sistema de gestión de bases de datos (SGDB) adecuado es crucial para garantizar un rendimiento óptimo y una buena experiencia de usuario. A día de hoy, existen múltiples opciones a valorar en función de su arquitectura, escalabilidad y otras características que se han tenido en cuenta para elegir el gestor adecuado.

Las **bases de datos relacionales** ofrecen muchas ventajas, tienen una estructura muy definida, permiten realizar consultas más complejas de manera eficiente y ofrecen una mayor integridad y consistencia de los datos. Sin embargo, estas son características que en el contexto de este proyecto no se iban a aprovechar al máximo. Por ello, ciertos SGDB como *MySQL*, *PostgreSQL*, *Oracle* y *SQLite* fueron descartados como opciones para crear la base de datos de *BreathBank*.

Por otro lado, las **bases de datos no relacionales** ofrecían algunas ventajas que sí podían ser de mayor provecho para el contexto de desarrollo de una aplicación para móviles: mayor flexibilidad en el modelo de datos y facilidad para manejarlo, mayor capacidad de adaptación a cambios en los datos, mejor manejo de operaciones de lectura y escritura en tiempo real, mayor tolerancia a fallos y disponibilidad, etc. Pero sobre todo, la mayor ventaja que ofrecían en este caso, era una mayor facilidad a la hora

de integrarlas con la aplicación y de conectarlas con las herramientas de desarrollo que estaba utilizando.

En concreto ***Firebase*** (Sección 4.6) es una plataforma que ofrecía todas las herramientas para cubrir las necesidades de *backend* que requería la aplicación: autenticación de usuarios, almacenamiento de datos, comunicación entre *frontend* y *backend*, notificaciones, etc. y además está muy integrada con *Flutter*, el entorno de desarrollo que se ha decidido utilizar.

Característica	**BBDD Relacionales**	**BBDD No Relacionales**
Escalabilidad	↓	↑
Consistencia de datos (ACID)	↑	↓
Flexibilidad de esquema	↓	↑
Rendimiento en consultas simples	↓	↑
Consultas complejas	↑	↓
Soporte para datos jerárquicos	↓	↑
Transacciones complejas	↑	↓
Rendimiento en grandes volúmenes de datos	↓	↑
Manejo de relaciones complejas	↑	↓
Fácil de usar para desarrolladores	↑	↑
Manejo de datos en tiempo real	↓	↑
Costo de infraestructura	↓	↑

Tabla 4.2: BBDD Relacionales VS BBDD No Relacionales

Firebase

Firebase [15] es una plataforma de desarrollo de aplicaciones móviles y web ofrecida por *Google*, que proporciona una variedad de servicios *backend*

para ayudar a los desarrolladores a construir, gestionar y escalar aplicaciones de manera eficiente.

Servicios de *Firebase* utilizados en el proyecto

- ***Firestore DB* y *Realtime DB*:** Bases de datos *NoSQL* escalables que permiten almacenar y sincronizar datos en tiempo real entre todos los usuarios de la aplicación.

- **Autenticación:** Servicio sencillo para gestionar el registro, inicio de sesión y autenticación de usuarios mediante múltiples proveedores como correo electrónico, *Google*, *Facebook*, etc.

- ***Cloud Messaging:*** Aunque no se ha utilizado completamente en la aplicación, este servicio sí que está conectado con la misma para futuras versiones de *BreathBank*. Este sistema de notificaciones permite enviar distintos mensajes al usuario, programarlos y desde la aplicación gestionar respuestas a los mismos.

Firestore DB vs Realtime DB

Tanto *Realtime Database* como *Firestore Database* son **bases de datos *NoSQL* en tiempo real** proporcionadas por *Firebase*, lo que significa que ambas permiten una sincronización instantánea de datos entre el cliente y el servidor. Ambas están optimizadas para aplicaciones móviles y web, con una infraestructura gestionada que facilita la **escalabilidad**. En cuanto a su estructura, ambas ofrecen almacenamiento flexible de datos (*JSON* en *Realtime Database* y documentos en *Firestore Database*), y permiten la **actualización en tiempo real de los datos**, lo que es fundamental para aplicaciones de este tipo. Además, las dos bases de datos están integradas dentro del ecosistema de *Firebase*, lo que facilita su utilización con otros servicios de la plataforma, como la autenticación y las notificaciones.

Las **principales diferencias** entre ambas radican en varias áreas clave. *Realtime Database* organiza la información en forma de un **árbol *JSON***, mientras que *Firestore Database* emplea una estructura de datos basada en **colecciones y documentos**, lo que proporciona una organización más flexible y escalable. Respecto a las consultas, *Firestore Database* permite realizar **búsquedas complejas y optimizadas a través de índices**, lo que permite mayor eficiencia, mientras que *Realtime Database* dispone de un sistema más básico y limitado para este tipo de operaciones. En términos

de **actualización en tiempo real**, aunque ambos servicios soportan la sincronización instantánea de datos, *Firestore Database* maneja de manera más eficiente conexiones en segundo plano y operaciones con grandes volúmenes de datos, aunque esto no es de gran importancia en este caso, ya que la aplicación no va a llegar a un volumen de datos tan grande como para que esto sea un factor relevante. Por último, *Firestore Database* ofrece un control de acceso más detallado, permitiendo **reglas de seguridad específicas** para cada documento y colección, mientras que *Realtime Database* trabaja con **reglas de acceso más generales**.

Teniendo en cuenta todo esto, he considerado que *Firestore Database* era más adecuado para este tipo de aplicación debido a la flexibilidad que me ofrecía en el modelo de datos, la fácil organización de los mismos, y la mayor capacidad de realizar consultas y devolver datos de manera más rápida.

Aspecto	**Realtime DB**	**Firestore DB**
Arquitectura	Único árbol JSON	Colecciones y documentos
Consultas	Simples, sin índices complejos	Con índices automáticos
Escalabilidad	Menor	Mejor rendimiento a gran escala
Sinc. en Tiempo Real	Menos eficiente	Más eficiente
Latencia	Mayor (con vol. grandes)	Menor
Seguridad	Reglas globales	Reglas por colección
Rendimiento general	Medio	Alto

Tabla 4.3: Firestore DB VS Realtime DB

Ventajas de usar *Firebase*

- Servicios de autenticación, base de datos y notificaciones integrados en una única plataforma.
- Servicio de autenticación ya implementado listo para integrarse directamente en la aplicación.

- Bases de datos flexibles y en tiempo real.
- Fácil integración con *Flutter* (Sección 4.5).
- Permite trabajar con múltiples plataformas de desarrollo.
- Escalabilidad automática.

4.7. Analizador de calidad de código - *SonarQube*

SonarQube [30] es una plataforma gratuita creada para la inspección y análisis de código fuente. Esta herramienta es utilizada por los desarrolladores para identificar y corregir problemas en su código, como errores, vulnerabilidades de seguridad, malas prácticas de programación, código duplicado y problemas de mantenibilidad. Es capaz de analizar código en diferentes lenguajes, incluso con *Dart*, que es un lenguaje poco habitual, funciona bastante bien.

Dentro de los múltiples aspectos que se analizan se encuentran:

- **Errores:** detección de fallos potenciales en tiempo de ejecución, como llamadas a métodos sobre objetos nulos, condiciones siempre verdaderas o falsas, o recursos no cerrados correctamente. Este tipo de errores suelen ser detectados por el propio editor de código, pero *SonarCloud* también los revisa
- **Vulnerabilidades y seguridad:** identificación de posibles fallos de seguridad como inyecciones, uso inseguro de *APIs*, exposición de datos sensibles, o flujos de datos que pueden comprometer la aplicación.
- **Mantenibilidad:** evalúa la facilidad con la que el código puede ser modificado, mejorado o corregido. Las sugerencias de mejora que se dan en este apartado son muy útiles para conseguir que la aplicación siga algunos principios *SOLID* como el de única responsabilidad o el de abierto/cerrado.
- **Cobertura:** integra los resultados de todos las pruebas implementadas (Sección 3.2) para calcular el porcentaje de código cubierto por tests.
- **Duplicidad:** identifica bloques de código repetidos que pueden ser refactorizados. El código duplicado incrementa el esfuerzo de mantenimiento y puede dar lugar a inconsistencias si no se actualiza de forma uniforme.

5. Aspectos relevantes del desarrollo del proyecto

Como todo proyecto, este también ha estado marcado por las decisiones que se han ido tomando y el contexto en el que ha tenido lugar. Todos esos aspectos se recogen en este apartado.

5.1. Origen y contexto del proyecto

La asignación final de la propuesta se produce, tras un par de entrevistas, el día 24 de febrero de 2025. Teniendo en cuenta esta fecha y la de entrega del *TFG*, el proyecto debería completarse en su totalidad en apenas tres meses y medio. Para ello, se estableció un ritmo de trabajo bastante elevado, ya que cubrir la creación de una aplicación completa desde cero en tan poco tiempo, y compaginarlo con una jornada de prácticas de siete horas diarias, no fue tarea fácil. Aunque el compromiso fue total y la velocidad de avance bastante buena, se decidió posponer la entrega hasta el 8 de julio de 2025, con el objetivo de presentar un trabajo más completo y de mayor calidad.

Volviendo al inicio de *BreathBank*, desde el primer momento se supo que iba a haber diferencias con el desarrollo de otros tipos de *TFG*. En este caso, la idea o el tema que se quería abordar con la aplicación ya venía determinada por una persona externa. Debido a esto, las primeras horas se dedicaron a comprender en profundidad toda la información que se quería plasmar en la herramienta. Este contexto ya definido marcó en gran medida el desarrollo del proyecto, ya que la creación de todos los elementos de la aplicación tenía que ajustarse a él. Se intentó desde el principio ser lo más fieles a lo propuesto por el ideador de *BreathBank*, porque establecía un

marco teórico y médico que impactaría en gran medida en la utilización de la aplicación. En concreto, la naturaleza y los procedimientos que se detallan tanto en las evaluaciones como en las inversiones (Sección 3.1), están avalados por profesionales médicos para conseguir los objetivos propuestos (Sección).

5.2. Desarrollo general del proyecto

Sobre el desarrollo general del proyecto, y como se comenta en el apartado (Sección 4.1), se planteó un avance más o menos constante y continuo durante todos los meses. Se consiguió mantener este objetivo gracias a tener en consideración todo tipo de tareas (investigación, aprendizaje de lenguaje y *framework* de programación, configuración del proyecto...) y no solo aquellas relacionadas con la creación de código, permitiendo asignar tiempos de trabajo adaptados a la carga de cada área en específico. También, cada dos semanas se examinaban los avances del *sprint* lo que permitía ajustar regularmente el trabajo restante al tiempo disponible.

Otro aspecto relevante fue la configuración e integración de todas las herramientas que componen *BreathBank*. La primera decisión que se tomó fue cambiar de editor de código. Se comenzó con *Android Studio* ya que suele ser el editor más utilizado para el desarrollo de aplicaciones móviles. Sin embargo, a los pocos días, se sustituyó por *Visual Studio Code* (Sección 4.5) debido, sobre todo, a problemas de rendimiento con el emulador y a la poca comodidad para conectar con el repositorio de *GitHub* (Sección 4.2). La instalación del *framework* de desarrollo *Flutter* (Sección 4.5) fue bastante sencilla. Desde la documentación oficial de *Flutter* se pueden seguir los pasos fácilmente y encontrar las extensiones de *Visual Studio Code* que son necesarias tener instaladas. Para el *backend*, la instalación y configuración fue bastante más compleja y se dilató más en el tiempo debido a su dificultad. La documentación oficial de *Firebase* es tan completa como extensa, solo la lectura de la misma conlleva unas cuantas horas. Resulta bastante confuso que ofrezca la posibilidad de hacer la instalación de muchas maneras distintas, sin dejar claro unos pasos a seguir. Por ello, lo más eficiente es guiarse por tutoriales de instalación (como este tutorial de instalación de *Firebase*). Hay que tener en cuenta que no todos los equipos parten desde la misma configuración, por lo que los pasos a seguir pueden diferir. Respecto al uso de esta plataforma, me pareció interesante este artículo [22] que me ayudó a comprender un poco mejor el funcionamiento de la misma.

Respecto a la arquitectura general del código de la aplicación, desde el comienzo se concretó que el modelo MVC (*Model-View-Controller*) era el más adecuado organizativa y funcionalmente. Sin embargo, el inicio del desarrollo fue un poco anárquico, creando pantallas y funcionalidad sin una estructura clara. Más adelante, cuando el contenido de las pantallas estuvo más detallado, se procedió a refactorizar el código ya creado dividiéndolo en modelos, controladores y vistas. Posteriormente, se fueron añadiendo el resto de pantallas y su funcionalidad siguiendo este esquema, lo que facilitó obtener un código más estructurado, con mayor modularidad, más homogéneo y mucho más sencillo de extender y mantener.

5.3. Detalles del desarrollo del *frontend*

El primer paso que se realizó para comenzar con el desarrollo de la interfaz, fue el de reorganizar los objetivos que se querían conseguir y cómo distribuirlos en las diferentes pantallas. Al comienzo del proyecto, se realizó una estructura de pantallas muy general pero que sirviera de base para comenzar a construir. Para ello, la herramienta *Pencil* [11] fue muy útil y fue la que se utilizó en este caso. Cabe destacar, que como era obvio, esa primera estructura de pantallas difiere en gran medida de la organización actual ya que durante el proceso de desarrollo van surgiendo nuevas necesidades. Para recabar más información sobre buenas prácticas de creación de interfaces, me resultó interesante este artículo [32].

Figura 5.1: Parte de la idea inicial sobre la estructura de pantallas

Posteriormente, los objetivos principales que se querían conseguir con la interfaz son:

- Conseguir una interfaz sencilla, consistente, poco recargada e intuitiva.
- Que el usuario no necesitase apenas conocimientos para usar la interfaz, y que no supusiera un impedimento para usuarios sin experiencia a la hora de beneficiarse de la aplicación.
- Y lo más importante, el objetivo de *BreathBank* es practicar la respiración consciente y ejercicios de relajación. Para ello, la interfaz debe

indicar pasos claros pero sin requerir de una atención máxima por parte del usuario para manejarla.

Para el primer objetivo, fueron imprescindibles crear ciertos *widgets* más o menos generales, que fuesen reutilizables en diferentes pantallas de la aplicación y ayudasen a dar consistencia al utilizar siempre los mismos, pero que a la vez debían permitir cambios de tamaños, posicionamiento o comportamiento para adaptarse a la lógica de cada pantalla.

Para el segundo objetivo, se intentó seguir una estructura de aplicación que resultase familiar, con elementos reconocibles (como *login*, registro de usuario, *logout*, etc). También, se han incluido explicaciones para ciertos términos que pudieran resultar un poco más complejos (en el apartado 6.2 se deja como propuesta la idea de añadir tutoriales en la propia aplicación que ayudarían en este aspecto también).

Para el tercer objetivo, se ha invertido mucho tiempo en pensar como guiar correctamente al usuario en los ejercicios sin ser excesivamente intervencionista y permitiendo que se consigan los objetivos propuestos con estos mismos. Se llegó a la conclusión de que añadir estos aspectos ayudarían a la comodidad del usuario:

- Cuenta atrás antes de comenzar cada ejercicio o prueba, para que el usuario pueda estar preparado.
- Botones de control del tiempo de la prueba con la posibilidad de parar o comenzarla de nuevo en cualquier momento.
- Visualización del tiempo transcurrido o restante y progreso de la prueba.
- Tanto en las inversiones manuales como en una de las pruebas de evaluación, se permite al usuario tocar la pantalla cada vez que termine de inspirar o expirar para llevar automáticamente un recuento de las respiraciones del usuario, evitando que tenga que estar pendiente de la prueba y no consiguiendo los efectos de relajación esperados.

5.4. Detalles del desarrollo del *backend*

La decisión de desarrollar un *backend* viene determinada por tres motivos principales:

- Implementar un servicios de autenticación de usuarios, para un comportamiento más personalizado de la aplicación.
- Mostrar datos e información sobre el progreso del usuario.
- Al ser una aplicación centrada en conseguir beneficios para la salud de las personas, los datos guardados podrían utilizarse con fines médicos en un futuro.

Una de las ventajas de usar *Firebase* es que ofrece métodos que recogen la funcionalidad de autenticación y guardado de datos. Dichos métodos deben recogerse y adaptarse para cada aplicación en concreto. En *BreathBank*, hay varias clases de servicio (*Authentication Service y Database Service* en concreto) que utilizan estos métodos para crear su propio sistema de autenticación y guardado de datos. El agrupar esta funcionalidad en clases de servicio permite modificar este comportamiento sin afectar para nada al resto de la aplicación. De hecho, se podría sustituir *Firebase* por cualquier otra plataforma de *backend* y la aplicación seguiría funcionando sin problemas.

El guardado de datos, como se comenta previamente, podría convertir *BreathBank* en una herramienta de apoyo para profesionales médicos. Esta información de usuarios con ciertas patologías podría ser de gran ayuda para sus especialistas. Pacientes de las áreas de salud mental, rehabilitación respiratoria y pulmonar, incluso personas con dolores crónicos podrían utilizar la aplicación para mejorar sus patologías. A su vez, los médicos correspondientes podrían analizar los datos proporcionados para ofrecer soluciones más contrastadas y eficaces.

Por otro lado, el sistema de *backend* utilizado ofrece un servicio de mensajería y notificaciones al usuario. Dicho servicio no terminó de implementarse porque se salía un poco del ámbito de la aplicación, pero podría llegar a ser útil a la hora de permitir al usuario programar ciertos entrenamientos, recibir recordatorios regulares para no perder progreso en sus habilidades, etc. Las posibles utilidades de las notificaciones se detalla más en este apartado (Sección 6.2).

5.5. Despliegue en *Appetize*

Aunque la aplicación está desarrollada para móviles *Android*, se meditó alguna opción que no conllevase todo el proceso de descarga e instalación. Además, de esta manera, los usuarios podrían probar *BreathBank* de manera

breve antes de decidir si deseaban instalarse la aplicación completamente en su dispositivo.

Appetize [2] es una plataforma que permite ejecutar aplicaciones móviles nativas directamente desde la web, sin necesidad de dispositivos físicos ni instalación local. Esta herramienta es muy cómoda para probar versiones de aplicaciones pero tiene una limitación bastante importante. En este caso, la versión gratuita de *Appetize* solo permite probar la aplicación durante tres minutos. Al cabo de ese tiempo, la aplicación se cierra y debes volver a iniciarla desde la pantalla principal.

Se ha decidido hacer este pequeño despliegue ya que, aunque la limitación del tiempo es bastante importante, permite visualizar y utilizar ciertas funcionalidades de la aplicación solo haciendo *click* en un enlace (Enlace a BreathBank en Appetize).

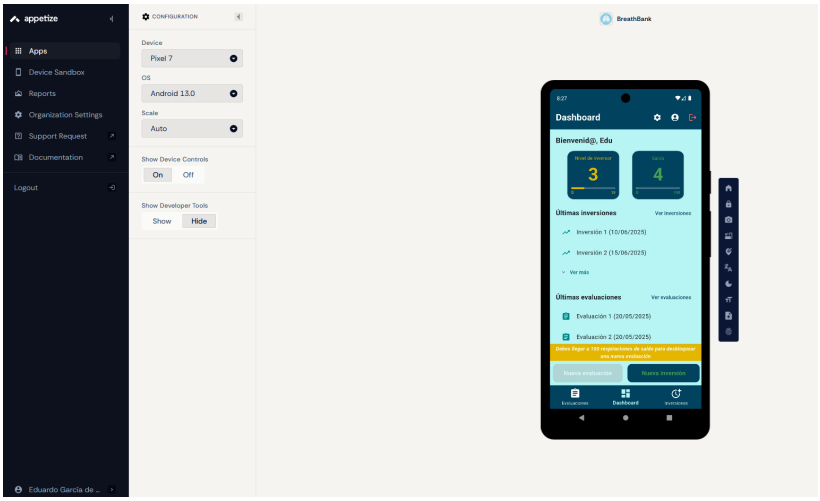

Figura 5.2: *BreathBank* en *Appetize*

5.6. Analizador de calidad de código

Casi desde el comienzo del proyecto, se decidió utilizar *SonarQube* (Sección 4.7) como analizador de calidad de código. Su uso fue bastante beneficioso a la hora de generar un código más mantenible, con menos complejidad y con la mayor cantidad de funcionalidad cubierta por los test. Se pueden ver los resultados de calidad del código realizado proporcionados por *SonarQube* en la imagen 5.3

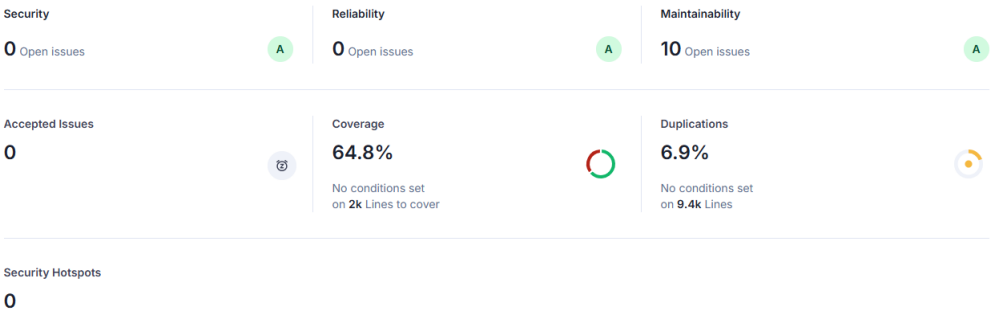

Figura 5.3: Resultados de calidad de código

Como se puede apreciar en la imagen, las métricas de calidad arrojan muy buenos resultados:

- En términos de seguridad no se ha detectado ninguna posible vulnerabilidad o fallo. Calificación de A
- Respecto a la mantenibilidad, se ha obtenido la calificación máxima A, reportando solo 10 *issues* relacionados con alta complejidad en algunas pantallas.
- En cuanto a fiabilidad tampoco se ha detectado ningún problema o fallo obteniendo una calificación de A.
- El 6,9 % de código duplicado obtenido hace destacar una gran reutilización del código. Apenas 600 líneas se han reportado como posibles duplicadas, de las más de 8500 líneas examinadas.
- Por último, el porcentaje de funcionalidad cubierta por tests o pruebas asciende hasta el 65 %, un gran dato por los motivos que se comentan en el siguiente apartado.

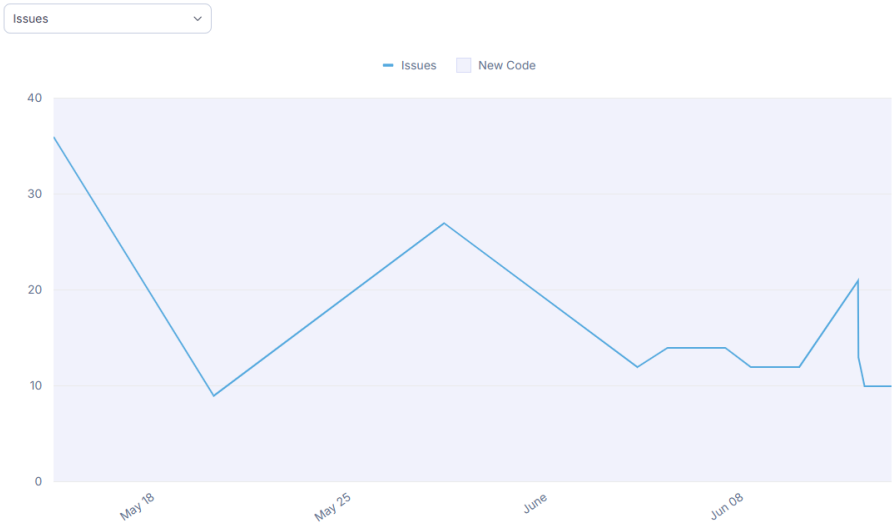

Figura 5.4: Variación de nº de *issues* (último mes)

Figura 5.5: Variación de porcentaje de código duplicado respecto al total de código (último mes)

Figura 5.6: Variación de porcentaje de código cubierto por test respecto al total de código (último mes)

5.7. *Testing*

La parte de *testing*, como en cualquier proyecto de *software*, es muy importante para asegurar un correcto y esperado funcionamiento de la herramienta que se está desarrollando.

En el apartado (3.2) se pueden ver los tipos de test que hay en *Flutter* por lo que no se repetirá aquí. Otro de los beneficios de usar un analizador de calidad de código, es que también ofrece un porcentaje de funcionalidad cubierta por estas pruebas. Para calcular este porcentaje, *SonarQube* detecta todos los ficheros con extensión *.dart* y cada método de estos ficheros contará como código a cubrir por los test. Esto ofrece unos resultados muy realistas de como de cubierta está la funcionalidad de la aplicación, pero requiere de muchísimas pruebas, a veces repetitivas, para conseguir un alto porcentaje.

Aparte de lo anterior, otro inconveniente que presentaba testear esta aplicación era la gran conectividad con servicios externos que presentan muchas clases. A la hora de programar pruebas para estas clases saltaban múltiples errores ya que se necesitaba de una conexión con *Firebase* para comprobar que efectivamente funcionaban. Esta dificultad se superó del siguiente modo:

- Para ciertas funcionalidades como registrar usuarios o guardar una nueva evaluación o inversión, el código de la aplicación utiliza el sistema de *backend*. Pero para comprobar que ese proceso se produce de manera satisfactoria no es necesario registrar exactamente ese usuario o guardar exactamente esa evaluación, sino que se puede simular o *mockear* para ser más precisos. En este caso, se utilizó *Mockito* [10] para generar esos *mocks* que permiten simular el comportamiento ciertos elementos. Verdaderamente se pueden *mockear* todas las clases que sean necesarias, pero en algunos casos presentan ciertas dificultades y no terminan de funcionar correctamente.

- Por ello, la decisión tomada fue la de generar *mocks* únicamente para las clases de servicio (autenticación y persistencia en base de datos). Todo el resto de clases se modificaron para que pudieran funcionar tanto con la instancia real como con la instancia *mockeada*. Esto ofrece muchísimas ventajas ya que hace una separación muy clara del entorno donde se está ejecutando el código, diferenciando los entornos de ejecución de los entornos de pruebas.

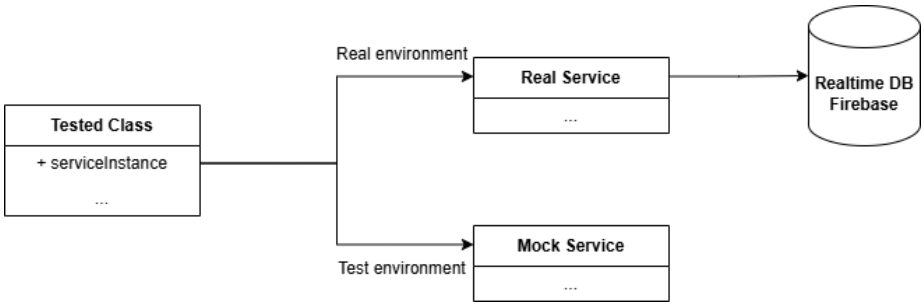

Figura 5.7: Diagrama de test con *mocks*

De esta manera, y teniendo en cuenta la gran cantidad de código que *SonarQube* determina como necesaria de ser cubierta y que parte de los métodos utilizados para la autenticación y el guardado de datos ya vienen dados (por lo que no se hicieron pruebas para ellos), se crearon 194 test con un porcentaje de código cubierto del 65 % (Sección 5.3).

6. Trabajos relacionados

6.1. Fundamentos teóricos

La respiración consciente ha sido ampliamente investigada desde diversas disciplinas, incluyendo la psicología [34], la medicina [18], la neurociencia [24] y otras especialidades. A continuación, se presentan y comentan brevemente varios documentos que fundamentan el enfoque y la utilidad de esta práctica, sirviendo como base teórica para el desarrollo de esta aplicación.

Libros relevantes

En primer lugar, *Jon Kabat-Zinn* en *Full Catastrophe Living* [20] explica los detalles del programa *MBSR* (Reducción del Estrés Basado en Mindfulness), donde la respiración consciente ocupa un lugar central como técnica de atención y regulación emocional. Este programa surgió para ayudar a pacientes con patologías médicas que no respondían a tratamientos convencionales. Es un buen ejemplo que certifica que la respiración consciente puede funcionar muy bien como alternativa a lo tradicional.

The Miracle of Mindfulness de *Thich Nhat Hanh* [25] introduce la respiración consciente desde una perspectiva más espiritual y filosófica. En él se mezclan enseñanzas, anécdotas personales, y ejercicios prácticos sobre cómo introducir el *mindfulness* en distintos momentos del día. El autor enseña cómo observar la respiración permite desarrollar una presencia plena en cada momento de la vida cotidiana. Este enfoque destaca la simplicidad y profundidad de la práctica, mostrando que puede ser accesible para todo tipo de usuarios.

Altered Traits de *Daniel Goleman y Richard Davidson* [13] trata los efectos sostenidos de la meditación, incluyendo la respiración consciente, desde un punto de vista más analítico. En este libro se distinguen dos caminos de la meditación. El primero implica una disciplina intensiva y busca una transformación total del ser. Sin embargo, el segundo es más accesible y menos intenso, pero igualmente beneficioso para muchas personas. Basándose en décadas de investigación y decenas de estudios científicos, los autores demuestran que la práctica regular produce cambios duraderos en la estructura y el funcionamiento cerebral. Entre los principales beneficios documentados se puede encontrar reducción de estrés, mejora de la atención, mayor control emocional y el tratamiento de trastornos mentales.

Artículos científicos

En el ámbito empírico, *Arch y Craske (2006)* [3] evaluaron los efectos de una inducción breve (15 minutos) basada en la respiración consciente, su impacto en la regulación emocional y la disposición de las personas a enfrentarse a estímulos negativos. Mediante un experimento controlado, observaron que los participantes que practicaron previamente la respiración consciente presentaban una menor reactividad emocional ante estímulos negativos, en comparación con otros grupos. Los ejercicios guiados pueden tener un alto impacto en nuestra respuesta ante todo tipo de situaciones.

Khoury et al. (2013) [23] llevan a cabo un análisis exhaustivo sobre terapias basadas en *mindfulness* recogidas en más de 200 estudios, concluyendo que estas son altamente efectivas para reducir síntomas de ansiedad, depresión y estrés. Aunque su eficacia es similar a la de las terapias cognitivas-conductuales (TCC) o tratamientos farmacológicos, los programas que utilizan la respiración consciente y el *mindfulness* son más sencillas de llevar a cabo, tienen menos efectos secundarios y pueden aplicarse a grupos de personas más grandes.

Siguiendo una línea similar al artículo anterior, en *Mindfulness meditation improves cognition: Evidence of brief mental training* [35] se presenta un estudio en el que se analizan los efectos de un breve entrenamiento en *mindfulness*, apenas 4 sesiones de 20 minutos al día. Los resultados muestran mejoras en el estado del ánimo, en la capacidad de mantener la atención, en la memoria de trabajo y en la función ejecutiva. Esta evidencia sugiere que, aún con pequeñas sesiones de trabajo (como el contexto de BreathBank), los beneficios de la respiración consciente están al alcance de todos.

Por su parte, *Schoenberg y David (2014)* [29] reflexionan sobre el uso del *biofeedback* en el tratamiento de trastornos psiquiátricos, destacando entre sus estrategias más eficaces la respiración controlada. El *biofeedback* es una técnica terapéutica que permite a una persona aprender a controlar funciones corporales que normalmente son automáticas. Aunque centrado en entornos clínicos, el estudio sugiere que el control consciente de la respiración tiene efectos reparadores significativos en ciertos pacientes con enfermedades mentales.

Estudios científicos

En los últimos años, la respiración consciente ha despertado un interés creciente en el ámbito de la investigación científica, especialmente por su impacto en la salud mental y el bienestar emocional. A continuación, se presentan estudios recientes que avalan su eficacia desde distintas perspectivas fisiológicas y clínicas.

Un análisis realizado por *Banushi et al.* [7] contrastó la eficacia de las técnicas de respiración (*breathwork*) como tratamiento para personas adultas que padecen de ansiedad, ya que las técnicas convencionales no suelen abordar directamente este aspecto. Tras la revisión de más de 1000 artículos, se incluyeron 16 estudios que mostraron que diversas técnicas de respiración mejoran significativamente los síntomas de ansiedad.

En otro estudio, *Balban et al.* evaluaron cómo prácticas estructuradas de respiración durante solo cinco minutos al día podían mejorar el estado de ánimo y reducir la activación fisiológica asociada al estrés [6]. Se contrastaron tres tipos de técnicas de respiración: respiraciones cíclicas alternando inhalaciones y exhalaciones prolongadas, respiraciones con inhalaciones, apneas, y exhalaciones con la misma duración en cada respiración, y respiraciones con inhalaciones y exhalaciones muy cortas. Los resultados mostraron que la primera técnica mejoraba significativamente el estado de ánimo y redujo la frecuencia cardíaca de quienes la practicaban.

Por otra parte, en *Effects of cons- cious connected breathing on cortical brain activity, mood and state of consciousness in healthy adults* [5] se investigó cómo la respiración consciente guiada influye en la actividad cortical, descubriendo que regula patrones cerebrales asociados a la ansiedad, lo que refuerza su valor como técnica de autorregulación emocional.

Finalmente, Ruiz-Fernández et al. [28] comentan cómo una intervención breve de respiración consciente en pacientes con cáncer, produce mejoras significativas en el estrés percibido y los niveles de *mindfulness*. Obviamente

no se puede considerar la respiración consciente como tratamiento contra patologías oncológicas, pero sí como paliativo y herramienta para mejorar los niveles de estrés y ansiedad en los pacientes que las sufren.

En conjunto, estas fuentes bibliográficas ofrecen un respaldo teórico y empírico sólido al diseño de la aplicación planteada. La integración de respiración consciente como eje principal no solo se justifica por su simplicidad y accesibilidad, sino también por sus efectos positivos comprobados a nivel psicológico, fisiológico y cognitivo.

6.2. *Apps*

Como la herramienta que se iba a desarrollar era una aplicación móvil, se consultaron diferentes *apps* relacionadas como inspiración y estímulo creativo:

- **Calm** [9] es una aplicación para dormir y meditar muy popular entre los usuarios. Incluye meditaciones guiadas de distintas temáticas, historias para dormir y música relajante. Está disponible para múltiples plataformas como *Android*, *iOS* o web.
- **Headspace** [17] también tiene soporte multiplataforma, centrado en cursos de meditación aunque también incluye historias para ayudar a conciliar el sueño. Parte de la aplicación requiere una suscripción de pago.
- **Breathwrk** [8] ofrece múltiples ejercicios de respiración con una interfaz muy elaborada y visual. Además presenta al usuario los resultados de estos ejercicios con gráficas de progreso y logros. También ofrece al usuario crear o suscribirse a programas personalizados. Está disponible en múltiples plataformas.
- **RespiRelax+** [31] es una aplicación gratuita para móviles que ofrece la posibilidad de utilizarse *offline*. Respecto al contenido de la misma, ha desarrollado una interfaz intuitiva e inmersiva logrando muy buenos resultados. Ofrece programas de respiración según la experiencia del usuario, aunque también permite personalizarlos en su duración y efectos visuales y sonoros.
- **Insight Timer** [19] no está enfocada en ejercicios propios de respiración, sino que es más como una plataforma con múltiples contenidos sobre meditación. Música, historias para dormir, incluso eventos en

vivo sobre referentes del *mindfulness* ayudarán al usuario a obtener los beneficios de la relajación.

- **Prana Breath** [1] es la aplicación más similar a lo que se ha intentado desarrollar en *BreathBank*. Se centra principalmente en ejercicios de respiración guiados con una interfaz muy interesante. Estos ejercicios son muy personalizables por parte del usuario, incluso en modo dinámico, lo que permite variar la estructura del ejercicio una vez ya iniciado. También presenta el progreso con múltiples estadísticas, y ofrece la posibilidad de crear alarmas y recordatorios para una experiencia más adaptada.

Aplicación	Interfaz	Personalización	Visualización de progreso	Enfoque científico
Calm	Moderna e intuitiva	Media	Baja	Basado en *mindfulness*
Headspace	Moderna y amigable	Baja	Media	Validada por estudios científicos
Breathwrk	Muy visual e interactiva	Alta	Alta (gráficas, logros)	Respaldada por técnicas de respiración con base científica
RespiRelax+	Intuitiva y simple	Media-Alta	Baja	Sin validación científica directa
Insight Timer	Variada, orientada a contenido	Baja	Baja	Ofrece contenido verificado
Prana Breath	Funcional y detallada	Muy alta	Alta (estadísticas avanzadas)	Sin respaldo clínico formal
BreathBank	Intuitiva y amigable	Media (ejercicios adaptados)	Alta (historial y estadísticas)	Origen y técnicas con base científica

Tabla 6.1: Comparativa de aplicaciones

7. Conclusiones y Líneas de trabajo futuras

Conclusiones

Tras la finalización de este proyecto y después de comparar el resultado final con los objetivos propuestos inicialmente (Sección 2.1), se puede afirmar que:

- Se ha conseguido desarrollar una aplicación con la funcionalidad propuesta al comienzo del proyecto y dando cobertura a todas las necesidades que presentó el ideador de *BreathBank*.
- Se han utilizado todas las herramientas recogidas en el apartado 4.1, de manera que cada una de ellas ha aportado un valor significativo al resultado final de la aplicación.
- La metodología *Scrum* (Sección 4.1) ha ayudado a conseguir entregar el resultado final dentro de plazo. Es cierto que el volumen de trabajo se podría haber distribuido mejor, ya que los primeros *sprints* contaban con muchas menos tareas asignadas que los últimos. Esto provocó, en muchos momentos, en sobrecargas de trabajo para no tener que reducir la funcionalidad implementada y poder seguir entregando a tiempo. En conclusión, este proyecto debería haberse comenzado antes para contar con un plazo más ajustado a los requerimientos del mismo.
- El desconocimiento de muchas de las herramientas utilizadas en el proyecto y la inexperiencia previa en este tipo de trabajos (de tan largo plazo y realizando una aplicación completa desde cero), se ha tenido

que cubrir con un gran ejercicio de investigación y autoaprendizaje para conseguir los objetivos propuestos.

- Se han mejorado múltiples habilidades individuales, tanto a nivel técnico, para superar las dificultades que se presentaban durante el desarrollo, como a nivel personal a la hora de la toma de decisiones.

Líneas de trabajo futuras

Pese a que *BreathBank* se trató de hacer lo más completa posible, el tiempo de desarrollo fue muy limitado. Durante este tiempo, surgieron ideas muy interesantes que se plantearon implementar pero por diferentes motivos no se llevaron a cabo. Algunas de ellas se recogen a continuación:

- **Añadir más métodos de autenticación:** actualmente *BreathBank* dispone de un sistema de acceso y registro de cuentas mediante correo electrónico y su correspondiente contraseña. A día de hoy, muchas aplicaciones cuentan con autenticaciones vinculadas con *Google*, *Facebook*, *Apple* o incluso mediante autenticación biométrica. Implementar estos métodos adicionales agilizaría el proceso de registro a la aplicación y evitaría al usuario tener que memorizar más contraseñas.

- **Crear tutoriales de iniciación:** a pesar de que la aplicación cuenta con numerosas explicaciones sobre las dinámicas de realización de evaluaciones e inversiones, sería un buen punto añadir un tutorial que salte cuando se detecte un nuevo usuario. En él se podría enseñar más en profundidad todos los apartados de la aplicación y dejar los conceptos más claros antes de que el usuario comience a utilizarla. Se podrían incluir textos, imágenes e incluso breves vídeos de demostración.

- **Configurar notificaciones de usuario, alarmas y recordatorios:** aprovechando que se utiliza *Firebase* como plataforma de desarrollo para el *backend*, *Firebase Cloud Messaging* permite emitir notificaciones a los usuarios de la aplicación. Se podrían configurar ciertos eventos que lancen dichas notificaciones, como alcanzar un cierto nivel de inversor, llegar a un número de inversiones o que pase cierto tiempo sin que el usuario abra la aplicación. Por otro lado, hay otras plataformas que podrían ayudar a integrar un sistema de alarmas o recordatorios, que faciliten al usuario la tarea de programar sus entrenamientos y adaptarlos a su tiempo libre.

- **Incluir nuevos idiomas:** actualmente *BreathBank* solo está disponible en castellano. Una posible línea de trabajo futura sería agregar nuevos idiomas. Este aspecto se pensó durante el desarrollo, por lo que todos los textos de la aplicación están en una clase concreta, facilitando la traducción de los mismos y que el sistema pueda cargar la información en tiempo de ejecución en base al idioma seleccionado.

- **Incluir diferentes interfaces para modo claro/oscuro:** esta futura línea de trabajo va un poco en la línea de la anterior. Tampoco aporta nada nuevo funcionalmente, pero sería una manera de hacer más atractiva la aplicación. En este caso la interfaz de *BreathBank* no varía en función del modo del sistema. Pero como esta posibilidad también se barajó durante el desarrollo, se decidió organizar todos los colores, estilos y aspectos visuales en una clase aparte, de manera que su modificación o sustitución fuera más simple y rápida.

- **Añadir contenidos multimedia:** como aparece en algunas de las aplicaciones del apartado (6.2), se podría complementar la práctica de respiraciones conscientes con otro tipo de contenidos: música, historias, *podcasts* de relajación, etc.

- **Desarrollar nuevos tipos de ejercicios y pruebas de evaluación de condiciones del usuario:** los ejercicios que conforman tanto las inversiones como las pruebas de evaluación, venían dados dentro del marco de la aplicación. Una posible innovación dentro de la *app* podría ser incluir otro tipo de ejercicios, o incluso que las pruebas de evaluación variasen en función del nivel al que el usuario quiere subir.

- **Vincular dispositivos de medición de respiraciones:** esta sin duda es la propuesta más ambiciosa y a la vez la más compleja. A día de hoy todavía no existe un dispositivo capaz de registrar de manera automática nuestras respiraciones, aunque, dado el ritmo en el que está avanzando la tecnología, es probable que pronto sea posible. La aplicación se tendría que rediseñar prácticamente entera, pero permitiría aislar totalmente la atención del usuario del dispositivo, logrando resultados mucho mejores y consiguiendo una experiencia de usuario aún más satisfactoria.

Bibliografía

[1] Abdula. Prana breath: Aplicación sobre respiración consciente. https://play.google.com/store/apps/details?id=com.abdula.pranabreath&hl=es_419, 2025.

[2] Appetize Technologies Inc. Appetize.io. https://appetize.io/, 2025.

[3] Joanna J. Arch and Michelle G. Craske. Mechanisms of mindfulness: Emotion regulation following a focused breathing induction. *Behaviour Research and Therapy*, 44(12):1849–1858, 2006.

[4] Asana. Información metodología scrum. https://asana.com/es/resources/what-is-scrum, 2024.

[5] Camile Bahi, Mona Irrmischer, Katrien Franken, George Fejer, Anna Schlenker, Jan Berend Deijen, and Hessel Engelbregt. Effects of conscious connected breathing on cortical brain activity, mood and state of consciousness in healthy adults. *Current Psychology*, 43(12):10578–10589, 2024.

[6] Maxwell Y. Balban et al. Brief structured breathing practices enhance mood and reduce physiological arousal. *Cell Reports Medicine*, 4(1):100930, 2023.

[7] Blerida Banushi, Madeline Brendle, Anya Ragnhildstveit, Tara Murphy, Claire Moore, Johannes Egberts, and Reid Robison. Breathwork interventions for adults with clinically diagnosed anxiety disorders: A scoping review. *Brain Sciences*, 13(2):20542, 2023.

[8] Breathwrk Inc. Breathwrk: Aplicación sobre ejercicios de respiración. https://play.google.com/store/apps/details?id=com.breathwrk.android, 2025.

[9] Calm. Calm: Aplicación sobre meditación y sueño. https://play.google.com/store/apps/details?id=com.calm.android&hl=es_419, 2025.

[10] dart.dev. Información sobre mockito. https://pub.dev/packages/mockito, 2025.

[11] Evolus. Pencil project. https://pencil.evolus.vn/, 2025.

[12] GitHub. Página inicial de github. https://github.com/, 2025.

[13] Daniel Goleman and Richard J Davidson. *Altered Traits: Science Reveals How Meditation Changes Your Mind, Brain, and Body*. Avery, 2017.

[14] Google. Página inicial de dart. https://dart.dev/, 2025.

[15] Google. Página inicial de firebase. https://firebase.google.com/?hl=es-419, 2025.

[16] Google. Página inicial de flutter. https://flutter.dev/, 2025.

[17] Headspace Inc. Headspace: Aplicación sobre mindfulness. https://play.google.com/store/apps/details?id=com.getsomeheadspace.android&hl=es_419, 2025.

[18] Gay Hendricks. *Conscious breathing: Breathwork for health, stress release, and personal mastery*. Bantam, 2010.

[19] Insight Network Inc. Insight timer: Aplicación sobre meditación. https://play.google.com/store/apps/details?id=com.spotlightsix.zentimerlite2&hl=es, 2025.

[20] Jon Kabat-Zinn. *Full Catastrophe Living: Using the Wisdom of Your Body and Mind to Face Stress, Pain, and Illness*. Delta, 1990.

[21] KeepCoding. Testing en flutter. https://keepcoding.io/blog/testing-en-flutter/, 2025.

[22] Chunnu Khawas and Pritam Shah. Application of firebase in android app development-a study. *International Journal of Computer Applications*, 179(46):49–53, 2018.

[23] Bassam Khoury, Tania Lecomte, Guillaume Fortin, Marjolaine Masse, Phillip Therien, Vanessa Bouchard, Marie-Andrée Chapleau, Karine Paquin, and Stefan G. Hofmann. Mindfulness-based therapy: A comprehensive meta-analysis. *Clinical Psychology Review*, 33(6):763–771, 2013. Artículo sobre los beneficios a nivel psicológico de terapias basadas en el mindfulness.

[24] Antoine Lutz, John D Dunne, and Richard J Davidson. Meditation and the neuroscience of consciousness: An introduction. *The Cambridge handbook of consciousness*, 19, 2006.

[25] Thich Nhat Hanh. *The Miracle of Mindfulness*. Beacon Press, 1975.

[26] Overleaf. Página inicial de overleaf. https://es.overleaf.com/, 2025.

[27] David Robson. Why slowing your breathing helps you relax. *BBC Worklife*, March 2020.

[28] María Dolores Ruiz-Fernández et al. Mindful breathing: Effects of a five-minute practice on perceived stress and mindfulness in patients with cancer. *Clinical Journal of Oncology Nursing*, 25(2):203–210, 2021.

[29] Paul LA Schoenberg and Anthony S David. Biofeedback for psychiatric disorders: A systematic review. *Applied Psychophysiology and Biofeedback*, 39(2):109–135, 2014. Artículo sobre los beneficios de terapias basadas en el biofeedback.

[30] SonarSource. Página inicial de sonarcloud. https://www.sonarsource.com/sem/products/sonarqube/, 2025.

[31] Thermes d'Allevard. Respirelax+: Aplicación sobre estrés y respiraciones. https://play.google.com/store/apps/details?id=com.thermesallevard.respi_relax&hl=es, 2025.

[32] Sergio Vergara. Buenas prácticas en el diseño de ux/ui móvil. https://www.itdo.com/blog/buenas-practicas-en-el-diseno-de-ux-ui-movil/, 2024.

[33] Visual Studio Code. Debugging in visual studio code. https://code.visualstudio.com/docs/debugtest/debugging, n.d.

[34] Andrea Zaccaro, Andrea Piarulli, Marco Laurino, Erika Garbella, Danilo Menicucci, Bruno Neri, and Angelo Gemignani. How breath-control can change your life: a systematic review on psycho-physiological correlates of slow breathing. *Frontiers in human neuroscience*, 12:409421, 2018.

[35] Fadel Zeidan, Susan K. Johnson, Bruce J. Diamond, Zhanna David, and Paula Goolkasian. Mindfulness meditation improves cognition: Evidence of brief mental training. *Consciousness and Cognition*, 19(2):597–605, 2010. Artículo sobre los beneficios del mindfulness en terapias cortas.