ÁFRICA Mª
BERMÚDEZ MEJÍAS

LOTUS

DESARROLLO DE APLICACIONES
MULTIPLATAFORMA
FRANCISCO RODRÍGUEZ MARÍN

Índice

CAPÍTULO 1: INTRODUCCIÓN

1.1-. INTRODUCCIÓN DEL PROYECTO

En un mundo donde la sociedad sigue viendo la menstruación como tabú y el bienestar de las mujeres y su salud sigue quedando en segundo plano, surge la necesidad de desarrollar herramientas prácticas y accesibles para que las mujeres puedan llevar un registro de su ciclo menstrual y comprender mejor los cambios que experimentan en sus cuerpos. En respuesta, se propone el desarrollo de una aplicación menstrual cual principal objetivo es proporcionar a las mujeres una herramienta sencilla para poder monitorear su ciclo menstrual.

1.2-. PROPÓSITO

El propósito de Lotus es que las mujeres puedan tener un seguimiento sencillo del ciclo menstrual, además de que puedan registrar los diversos síntomas que hay a lo largo de cada ciclo y se proporcionará recursos e información sobre el bienestar, salud femenina y la menstruación.

1.3-. OBJETIVOS DEL PROYECTO

- Investigación sobre las necesidades de las usuarias que usen la aplicación.
- Desarrollo de una aplicación que realizará la tarea de calendario menstrual que integrará información sobre el ciclo menstrual y salud femenina entre otros temas.
- Diseñar una interfaz gráfica sencilla e intuitiva para que las usuarias puedan acceder a las diferentes funciones sin dificultad.
- Elaboración de un manual de instalación y uso de la aplicación menstrual.
- Se redactará el proceso del proyecto junto con la decisiones tomadas, desafíos enfrentados y resultados obtenidos.

1.4-. COSTE DEL PROYECTO

En este apartado se detallarán y estimarán los costes del software y/o hardware requerido para el desarrollo del proyecto.

1.4.1-. COSTES DE DESARROLLO

Los costes de desarrollo de la aplicación se dividen en **costes de recursos informáticos** y **costes de personal**.

- **Hardware**: Los costes de hardware son un total de 0€ ya que no se usa ningún componente físico para el desarrollo de esta aplicación.
- **Software**: Para poder desarrollar esta aplicación se necesitan los siguientes componentes de software:
    a. **Android Studio**: Es el entorno de desarrollo (IDE) usado para poder desarrollar la aplicación, este entorno es gratuito por lo que no tendría ningún coste.
    b. **Emulación móvil**: Para poder hacer pruebas y ver nuestra aplicación en dispositivos móviles se ha usado el emulador de Android Studio que ya tiene integrado, por lo que no tendría ningún coste.
    c. **Firebase**: Las bases de datos (Realtime Database) y servicios de autenticación (Authentication) son gratuitos con ciertas restricciones. Si Lotus creciese notablemente nos veríamos obligados a pagar por mejoras en los servicios necesarios, por lo que no tendría ningún coste.
    d. **Clip Studio Paint PRO Versión 2.0**: Para el diseño e ilustración de los iconos e imágenes usados en algunas pantallas de Lotus, se ha obtenido un programa con las características necesarias para su desarrollo. El coste es de un pago único, aunque hay diferentes métodos de compra, y su precio es de 42€.
- **Personal**: El coste es 0 ya que un grupo de personas se ofrecieron a probar la aplicación gratuitamente y para el desarrollo de la misma no se ha incurrido en costos externos para contratar desarrolladores o adquirir servicios de desarrollo.

1.4.2-. COSTES DE IMPLANTACIÓN

Los costes para poner en funcionamiento la aplicación consta de 0 euros ya que disponemos con los recursos existentes de software por lo cual no requiere inversión adicional. En cuanto a licencias, se está usando un recurso que aplica la licencia *Apache License, Version 2.0* la cual es una licencia gratuita y se aplicará a nuestra aplicación.

CAPÍTULO 2: ANÁLISIS

2.1-. INTRODUCCIÓN

A lo largo de este capítulo, se detallará el desarrollo del proyecto en concordancia con los criterios previamente establecidos.

2.2-. ANÁLISIS DE REQUISITOS

Una vez se ha definido claramente el propósito último del proyecto, el siguiente paso implica la especificación y análisis detallado de los requisitos. En los siguientes apartados se identifican y documentan los requerimientos específicos del proyecto.

2.2.1-. REQUERIMIENTOS FUNCIONALES.

Se especificarán los servicios que presta Lotus, así como interacciones con los usuarios y otros sistemas.

- Calendario para un seguimiento sencillo del ciclo menstrual. Este calendario incluye y muestra días resaltados para el seguimiento de la menstruación e iconos para la ovulación como iconos para el seguimiento de los síntomas. En el se mostrarán los 6 próximos ciclos.
- Detalle del día elegido en el calendario que muestra los síntomas elegidos por el usuario.
- Una pantalla para añadir o eliminar síntomas.
- Apartado de información sobre el ciclo menstrual y salud femenina entre otros tópicos.
- Registro e Inicio de sesión mediante correo electrónico para usuarios como para administradores.
- Ajustes que incluye lo siguiente:
    - Cerrar sesión.
    - Cambiar datos de tu cuenta: Nombre, nombre de usuario y contraseña.
    - Enviar sugerencias y errores al equipo de Lotus.
    - Recordatorio para Anticonceptivos.
    - Cambiar datos del ciclo, como su duración y del periodo.
- Añadir, modificar y borrar artículos a los a administradores de la aplicación.

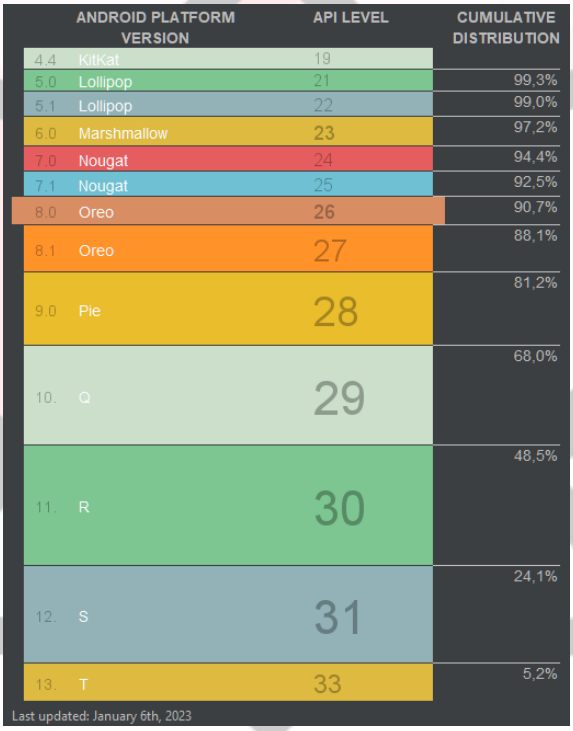
2.2.2-. REQUERIMIENTOS NO FUNCIONALES.

Se especificarán las restricciones y requisitos impuestos al sistema.

- **Seguridad**: Lotus no almacena contraseñas tanto en *Firebase*, como en el código o en el dispositivo móvil. *Authentication* brinda un token único cifrado en la memoria del sistema

para mantener la conexión del usuario en la aplicación hasta que el mismo cierre sesión. Además, los datos del usuario o interacciones del usuario con la base de datos u otros componentes de *Firebase*, nunca se hacen con una conexión directa, por lo que hay una *API* que actúa como intermediario entre *Firebase* y la aplicación, brindando así una mayor seguridad.

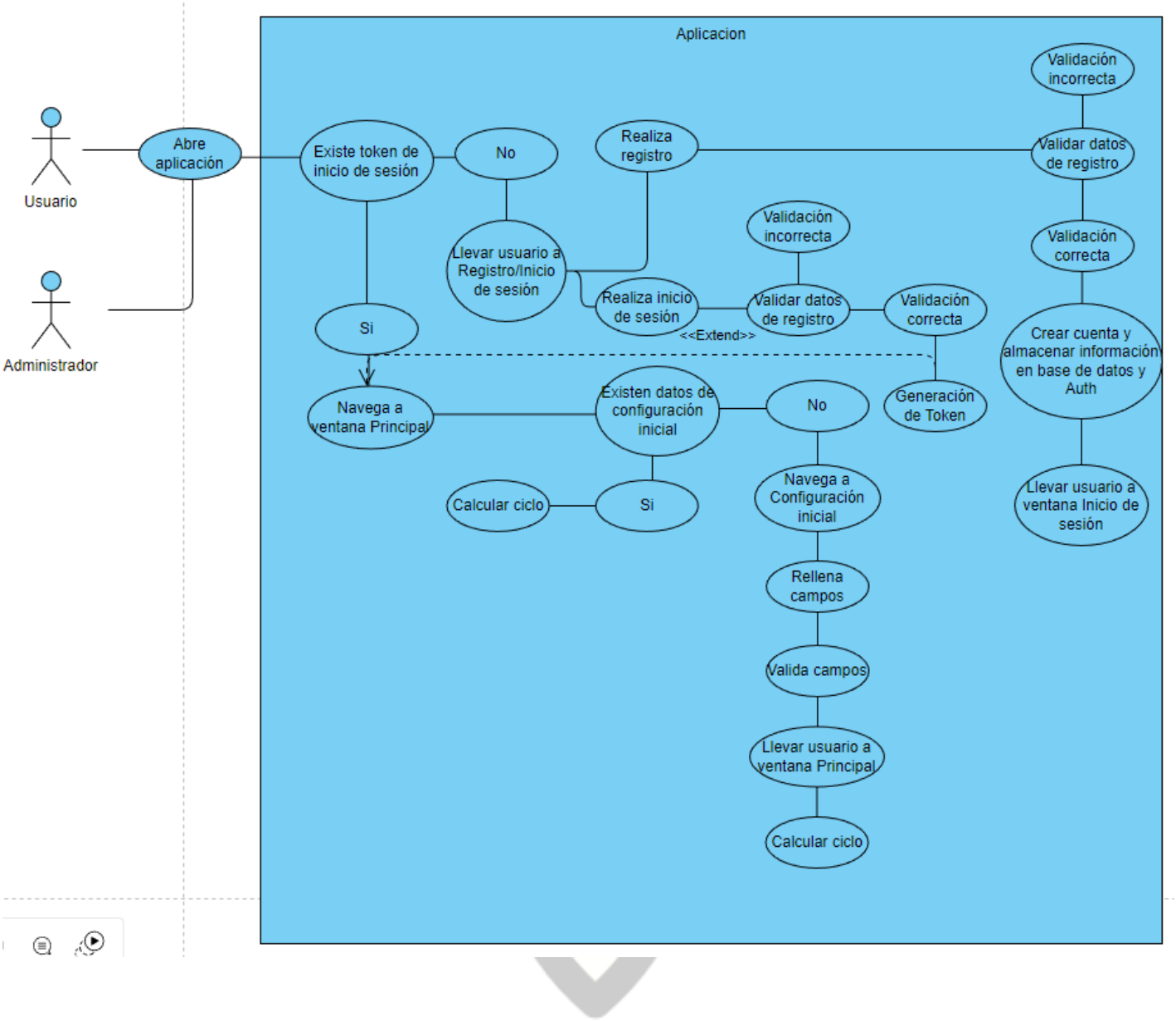
- **Rendimiento**: Proporciona tiempos de respuesta rápidos al cargar los datos de los usuarios desde la base de datos de *Firebase* y maneja de forma eficiente la carga de usuarios y cuenta con una buena escalabilidad para el futuro de la aplicación.
- **Disponibilidad**: La base de datos siempre estará disponible ya que al usar *Realtime Database* (base de datos de Firebase) es un almacenamiento basado en la nube y siempre cuenta con servidores de ayuda por si el servidor en el que se encuentran nuestros datos cae.
- **Compatabilididad**: Lotus funcionará en todos los dispositivos móvilos Andorid 8.0 (Oreo) en adelante, por lo que brinda una compatibilidad en el **90’7%** de los dispositivos.



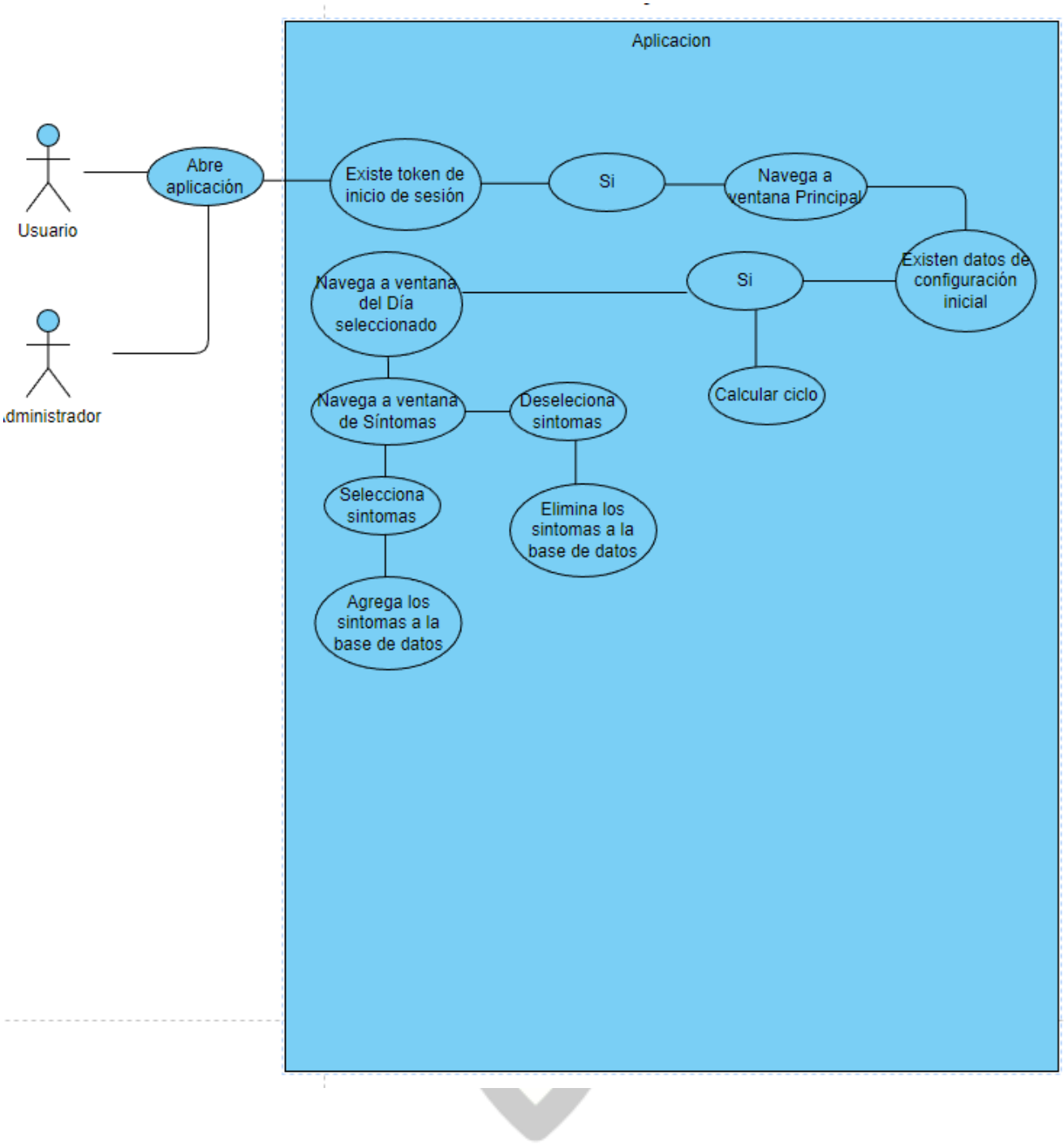
2.2.3-. CASOS DE USO

A continuación, se describen los casos de uso de los usuarios y administradores de la aplicación.

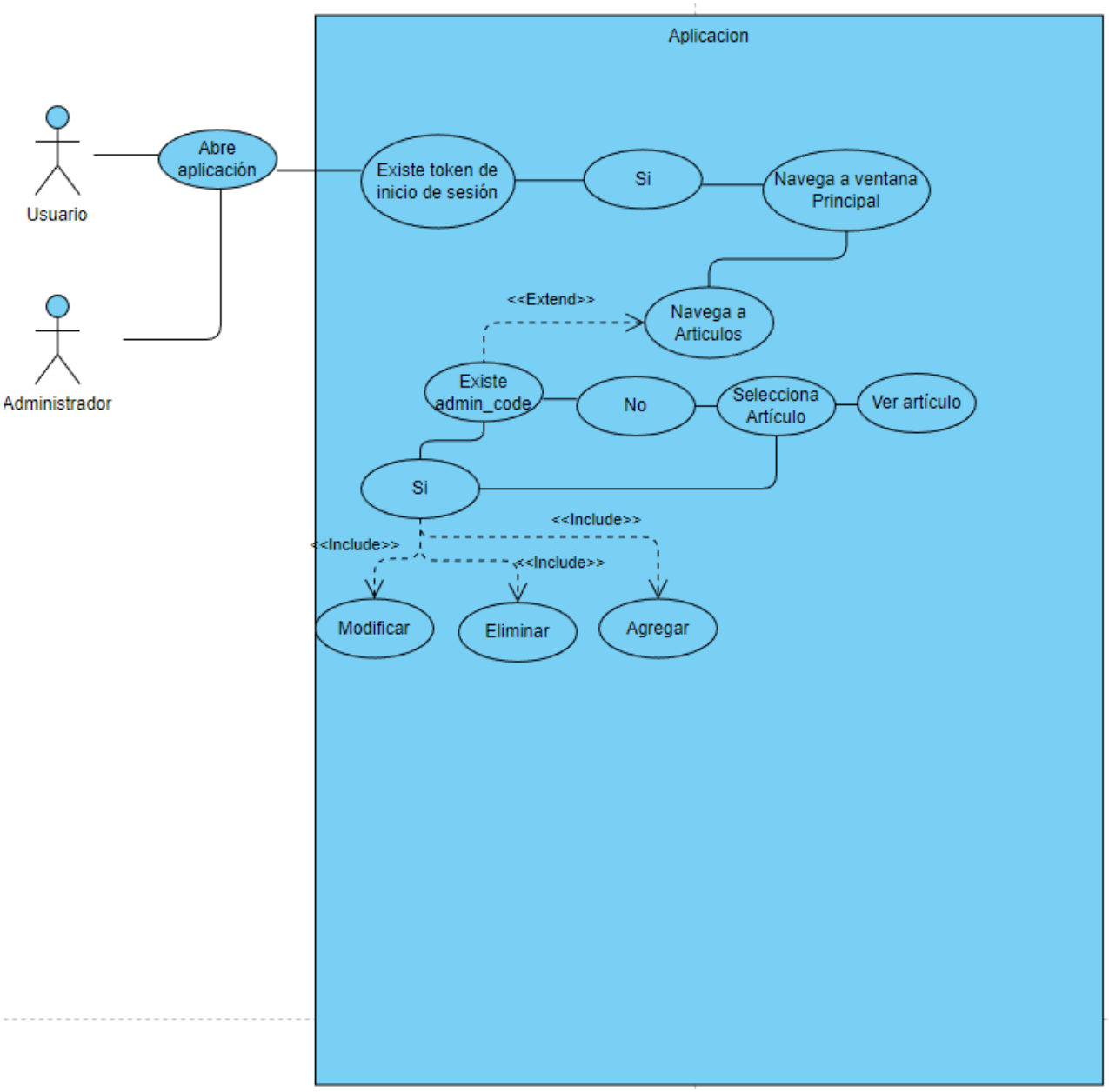
1. Caso de Registro, Inicio de sesión y Configuración inicial.



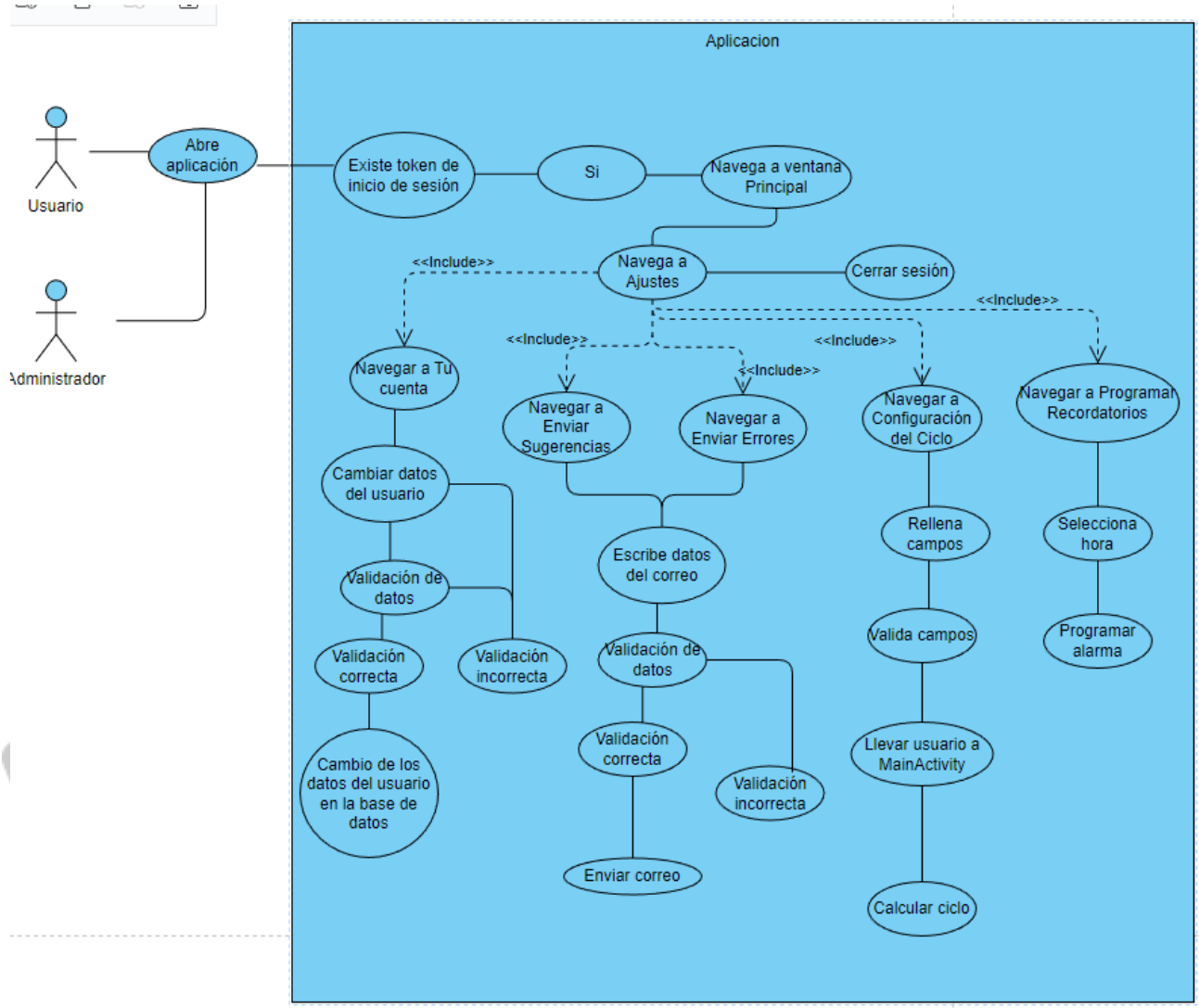
2. Caso de Añadir y Eliminar síntomas



3. Caso de Ver, Añadir, Modificar y Eliminar artículos



4. Caso de Ajustes



2.2.4-. REQUERIMIENTOS HARDWARE

Como se ha dictado en el 1.4.1 Costes de desarrollo, no se ha necesitado ningún hardware para la realización del proyecto ya que contamos con solo herramientas de software para dicho desarrollo.

2.2.5-. REQUERIMIENTOS SOFTWARE

Para el desarrollo de la aplicación se ha necesitado un entorno de desarrollo *Andorid* que nos brindase una emulación móvil y que también soportase Java como lenguaje de programación, por lo que se ha elegido Android Studio porque cuenta con todas esas características necesarias para el desarrollo de la aplicación. Este además nos brinda ayudas para poder conectar *Firebase* con nuestra aplicación de forma sencilla.

También se ha instalado *Clip Studio Paint PRO* para poder realizar y personalizar los iconos de la aplicación.

2.2.6-. OTROS REQUISITOS DE INTERÉS

Para poder usar *Firebase* en nuestra aplicación, hemos tenido que usar una cuenta de correo para poder crear un proyecto nuevo dentro de *Firebase*. Para el desarrollo de la aplicación se han usado *Authentication* que valida las credenciales y cuentas de usuarios de la aplicación y *Realtime Database* que guarda y gestiona los datos de la base de datos en formato JSON.

CAPÍTULO 3: DISEÑO

3.1-. INTRODUCCIÓN

En este tercer capítulo, a partir de los requisitos mencionados anteriormente, se realizará el diseño del desarrollo a implementar.

3.2-. DISEÑO DE BASE DE DATOS

En el diseño de base de datos, se definen las estructuras y relaciones que permitirán almacenar y gestionar los datos de manera eficiente. Se divide en tres etapas: diseño conceptual, diseño lógico y diseño físico.

Hay que tener en cuenta que nos basamos en una Base de datos NoSQL con un formato JSON, por lo que puede variar dependiendo del proyecto y los requisitos específicos.

3.2.1-. DISEÑO CONCEPTUAL

En este diseño se identificarán y definirán las entidades principales de nuestra base de datos, así como las relaciones.

Entidades:

- User (Usuario).
- Codes (Códigos).
- Artículos.
- Categorías.

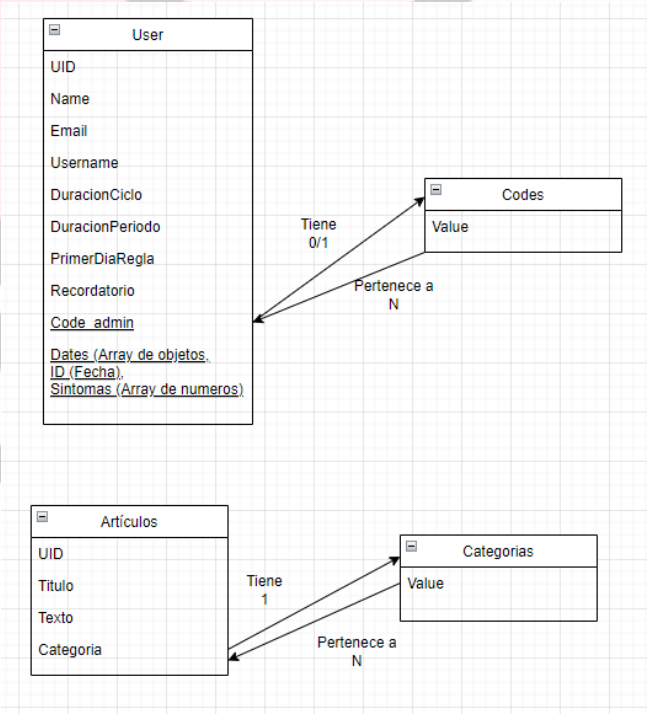
Relaciones:

- Un usuario puede tener o no un código.
- Un código puede pertenecer a muchos usuarios.
- Un artículo puede tener una categoría.
- Una categoría puede pertenecer a muchos artículos.

Atributos:

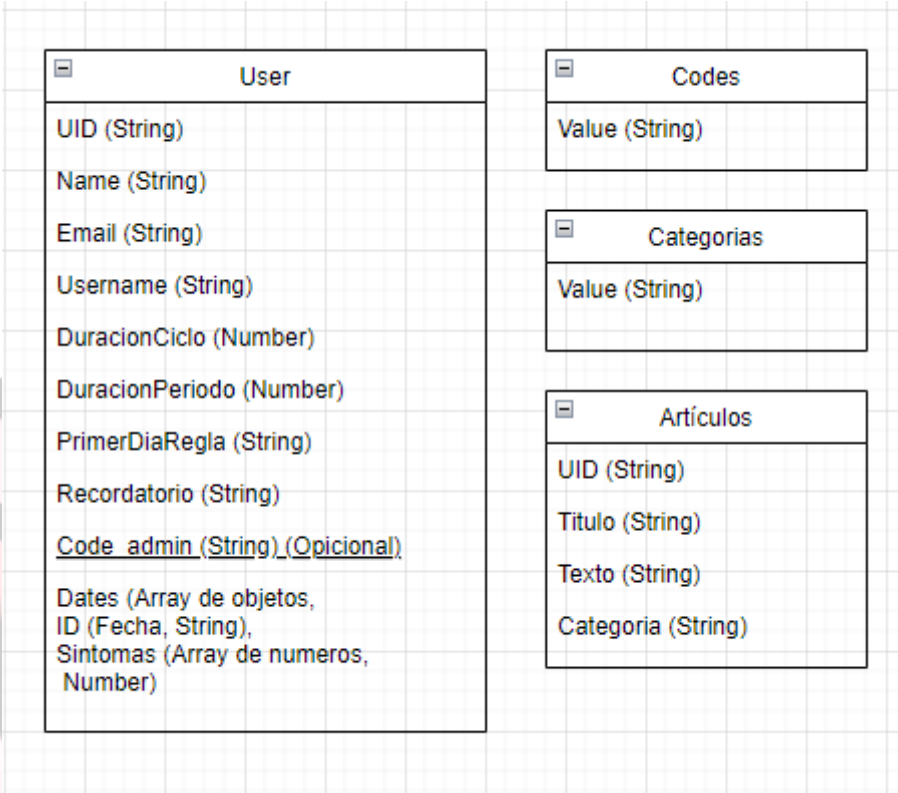
- User:
    - UID (User ID)
    - Name (Nombre)
    - Email (Correo electrónico)
    - Username (Nombre de usuario)
    - DuracionCiclo (Duración del ciclo)

  - DuraciónPeriodo (Duración del periodo)
  - PrimerDiaRegla (Primer día de la regla)
  - Recodatorio (Hora del recordatorio)
  - Code_admin (Codigo de administrador)
  - Dates (Fechas, Array de objetos)
    - ID (Fecha como ID)
    - Síntomas (Array de números)
- Codes:
  - Value (Valor del código)
- Artículos:
  - UID (Article ID)
  - Title (Título)
  - Text (Texto)
  - Categoría
- Categorías:
  - Value (Nombre de la categoría)



3.2.2-. DISEÑO LÓGICO

En el diseño lógico se describen las propiedades a ser utilizadas y sus tipos. Se definirá también un esquema de la base de datos y se establecerán obligaciones, como por ejemplo si un atributo es opcional.



A continuación, se enseñará un texto en formato JSON de como se estructura la base de datos.

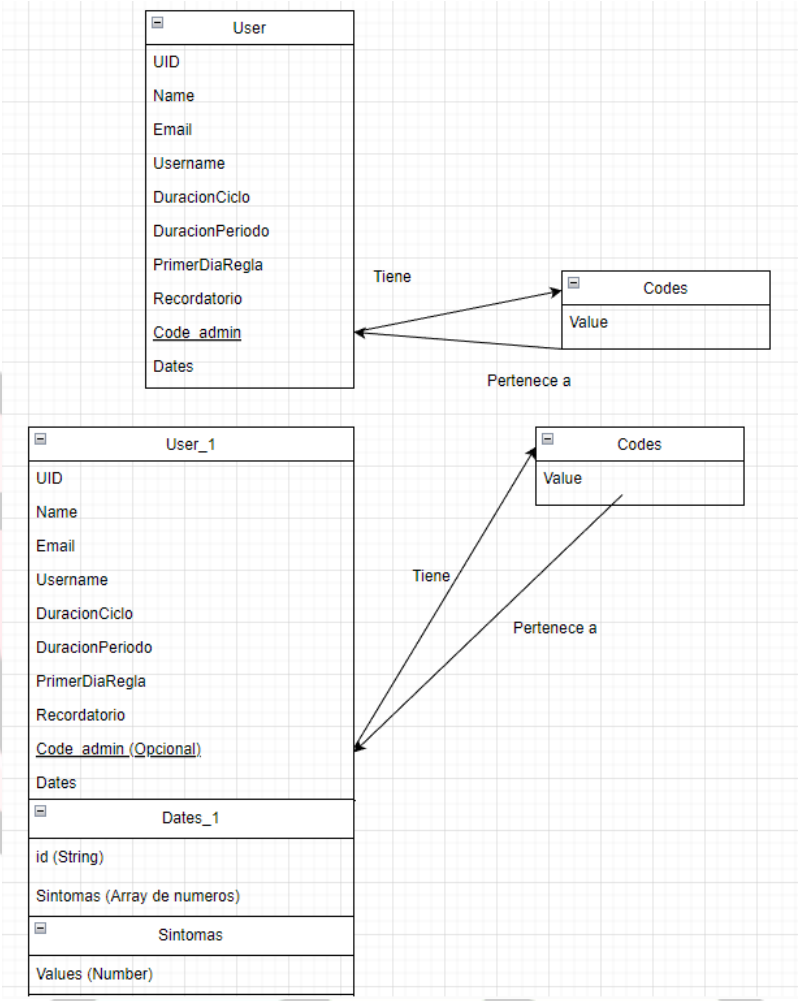
```
{
  "articulos": {
    "UID": {
      "UID": "String",
      "categoria": "String",
      "texto": "String",
      "titulo": "String"
    }
  },
  "categorias": {
    "id1": "String"
  },
  "codes": {
    "Value": "String"
  },
```

```
"users": {
    "uid": {
      "id": {
        "id": "String",
        "sintomas": {
          "-NWfiLbawk3lqmiGfbMt": "String"
        }
      },
      "admin_code": "String, Opcional",
      "duracionCiclo": Number,
      "duracionPeriodo": Number,
      "email": "String",
      "name": "String",
      "primerDiaRegla": "String",
      "uid": "String",
      "username": "String"
    }
  }
}
```

3.2.3-. DISEÑO FÍSICO

En una base de datos NoSQL JSON, los datos se pueden almacenar en documentos JSON, donde cada documento representa un registro o entidad.

En la siguiente imagen se ve un ejemplo de diseño físico de un usuario y códigos.

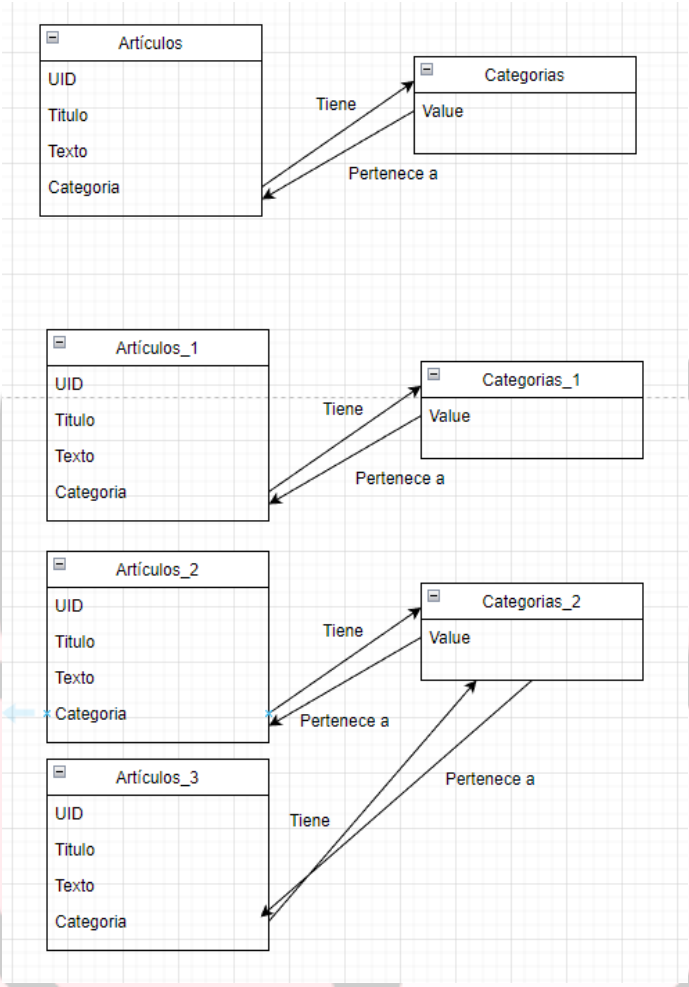


A continuación, se detalla como seria en formato JSON de Users y Codes:


```
"users": {
    "4DjB6tpyB2ae7m0CnyCYLZ1vP8P2": {
      "18-5-2023": {
        "id": "18-5-2023",
        "sintomas": {
          "-NVkEP_YBITyVXjiy4sP": "1"
        }
      },
      "19-5-2023": {
        "id": "19-5-2023"
      },
      "admin_code": "as8923h",
      "duracionCiclo": 28,
      "duracionPeriodo": 5,
      "email": "gmail1@gmail.com",
      "name": "Admin",
      "primerDiaRegla": "16-5-2023",
      "recordatorio": "14:21",
      "uid": "4DjB6tpyB2ae7m0CnyCYLZ1vP8P2",
      "username": "Admin2"
    }


"codes": {
    "FEs5YRE&wezQ": "FEs5YRE&wezQ",
    "as8923h": "as8923h"
  }
```

En la siguiente imagen se verá el diseño físico de artículos y categorías.



Y ahora se verá como seria en formato JSON de Artículos y Categorías:

```
"articulos": {
    "10273d39-59ff-4adf-bf4c-5d94ba3c5f57": {
      "UID": "10273d39-59ff-4adf-bf4c-5d94ba3c5f57",
      "categoria": "Salud femenina",
      "texto": "Owo2",
      "titulo": "Owo"
    }
}

"categorias": {
    "id1": "Ciclo y menstruacion",
    "id2": "Salud femenina",
    "id3": "Sexual",
    "id4": "Medicina"
  }
```

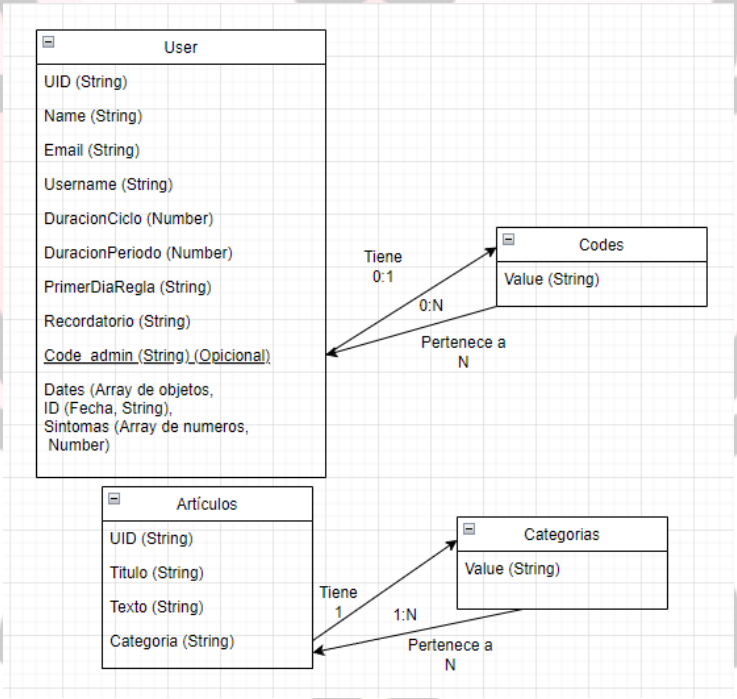
3.3-. DIAGRAMA DE CLASES

En la siguiente imagen se representará el diagrama de clases, que es una representación de las clases de una base de datos o sistema, junto con sus atributos, métodos y las relaciones entre ellas.

En este diagrama de clases, se representan las clases principales del sistema: User, Codes, Artículos y Categorías. Cada clase tiene sus atributos representados y sus tipos.

Las relaciones entre las clases se muestran con flechas. Por ejemplo, hay una relación entre User y Code, indicando que un usuario puede tener o no un código, y un código puede pertenecer a muchos usuarios.

También se muestra una relación entre Article y Categoria, indicando que un artículo puede tener una categoría, y una categoría puede pertenecer a muchos artículos.



3.4-. DISEÑO DE LA INTERFAZ

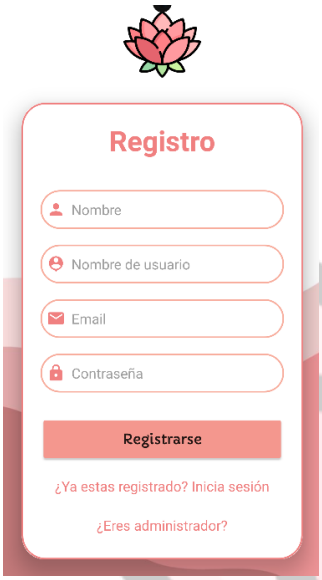
El diseño de la interfaz es un componente crucial en el desarrollo de cualquier proyecto. En este apartado, se presenta y describe el diseño de la interfaz del proyecto.

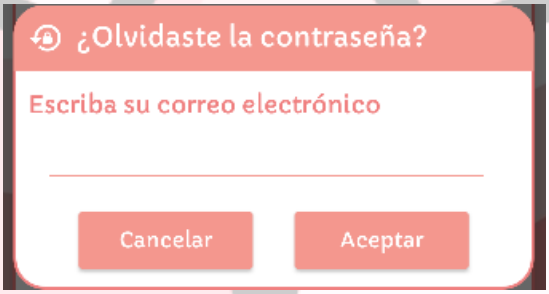
1. **Objetivos del diseño de la interfaz**: Primero, se establecen los objetivos con los cuales de diseñará la interfaz.
    a. **Usabilidad**: Nuestro objetivo principal es crear una interfaz intuitiva y fácil de usar hasta para los usuarios menos experimentados con los dispositivos móviles, por lo que se busca que los usuarios puedan aprender rápidamente como interactuar con la aplicación.
    b. **Eficiencia**: El diseño de la interfaz busca permitir a los usuarios realizar las tareas deseadas en el menor tiempo posible, como por ejemplo minimizando los pasos para realizar una acción.
    c. **Accesibilidad**: Lotus cuenta con un diseño accesible para usuarios con diferentes capacidades, como por ejemplo el tamaño de la tipografía, colores y navegación clara.
    d. **Estética**: La interfaz de Lotus debe ser atractiva visualmente para los usuarios para que puedan disfrutar al máximo de la experiencia. Esta estética se basa en una combinación de colores adecuada, elementos visuales y tipografías, entre otros.
    e. **Retroalimentación**: Lotus proporciona una retroalimentación de lo que está sucediendo en la aplicación todo el tiempo, para que así el usuario sepa lo que está ocurriendo.
    f. **Adaptabilidad**: El diseño de la interfaz está programado para adaptarse a casi cualquier dispositivo móvil de hoy en día.
2. **Requisitos**: A continuación, se detallan los requisitos que se han tenido en cuanta para el diseño de la interfaz.
    a. **Navegación intuitiva y sencilla**: La interfaz cuenta con una navegación clara y sencilla para el usuario, utilizando iconos y textos descriptivos. Además, proporciona un flujo de navegación lógico para que los usuarios puedan navegar sin ningún problema por la aplicación.
    b. **Diseño adaptivo**: La interfaz está diseñada para que pueda adaptarse a diferentes tamaños de pantalla.

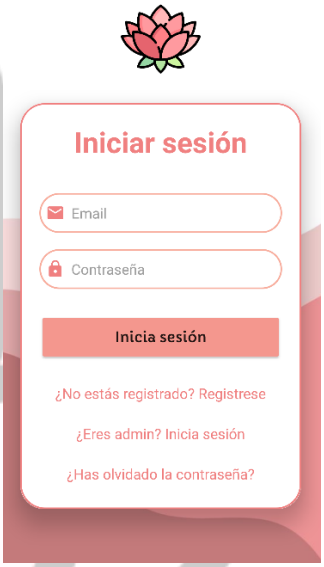
c. **Retroalimentación visual clara**: La interfaz proporciona en todo momento un *feedback* sobre las diferentes acciones que realiza el usuario, por ejemplo: agregar o eliminar síntomas a un día.

3. **Metodología de diseño**: Este apartado consiste en la metodología seguida para diseñar la interfaz.
    a. **Investigación**: Primero se realizó un estudio en el campo de salud femenina y menstruación para comprender las necesidades de los usuarios y se identificaron distintos casos de uso.
    b. **Definición de objetivos y requisitos del usuario**: Se establecieron los objetivos que los usuarios querían lograr, por ejemplo: Añadir síntomas al calendario, y se identificaron los requisitos funcionales y no funcionales para que cumplir esos objetivos.
    c. **Diseño temprano**: Una vez aclarado los apartados anteriores de puso en marcha un prototipo sencillo con diferentes diseños.
    d. **Diseño**: Se desarrolló un prototipo mas complejo con Just In Mind. Este prototipo se le presentó a varios usuarios para recopilar sus comentarios y base a sus comentarios se adaptó el diseño.

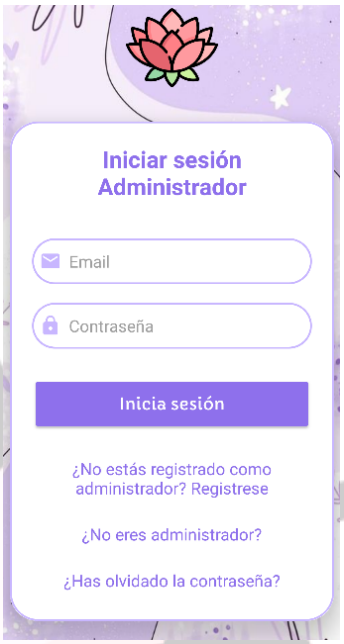
4. **Diseño visual**: En este apartado se mostrará el diseño visual de la aplicación.
   a. **Registro e inicio de sesión**: Si el usuario aun no tiene cuenta, las ventanas de registro e inicio de sesión será lo primero que vea dicho usuario. En esta parte hay 4 ventanas, 2 de usuario y 2 de administrador.
   b. **Registro e inicio de sesión de usuario**: En estas ventanas el usuario podrá registrarse si aun no tiene cuenta o iniciar sesión si ya al tiene. En ambas ventanas el usuario deberá rellenar los campos correspondientes. Si el usuario no completa algún campo requerido o ingresa una contraseña o correo electrónico incorrectos, se mostrará un mensaje de error en el campo correspondiente. Además, se ha implementado una opción para que el usuario recupere su contraseña en caso de olvido. Para ello, deberá proporcionar el correo electrónico asociado a su cuenta y recibirá un correo de recuperación de contraseña.



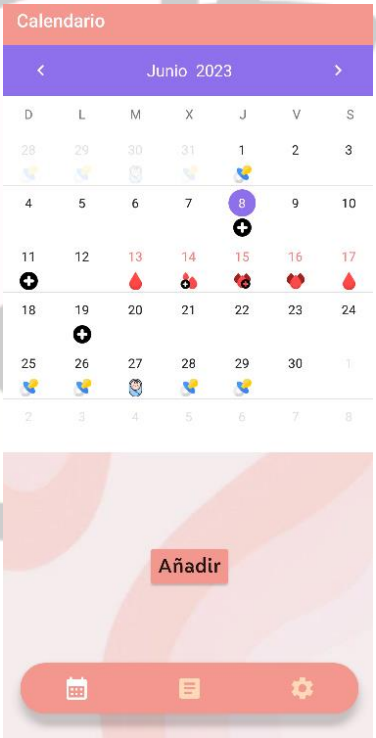




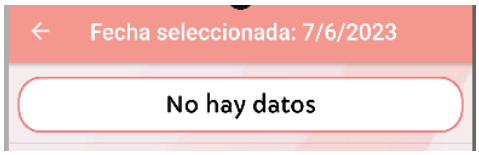
i. **Registro e inicio de sesión de administrador**: Estas ventanas con prácticamente iguales que las del usuario, la única diferencia es que el administrador debe introducir un código.



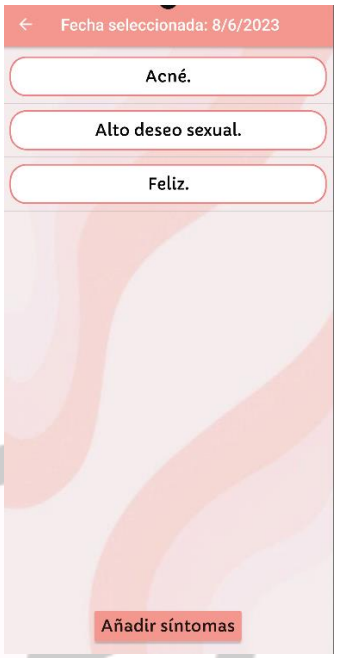
ii. **Calendario**: Esta es la ventana principal de Lotus, la cual muestra un calendario con los días de menstruación y ovulación calculados si ha realizado la configuración inicial. También se mostrarán iconos dependiendo de si el usuario a añadido o no síntomas. Si a realizado la configuración inicial, saldrá en pantalla un botón que lo llevará a una vista detallada del día marcado en el calendario. Abajo sale un navegador para que el usuario pueda moverse por diferentes ventanas.



c. **Detalle día**: En esta ventana se mostrarán los síntomas elegidos por el usuario. En caso de que el día no contenga síntomas se mostraría lo siguiente:



Arriba de la pantalla se incluye una flecha para que el usuario pueda navegar a la pantalla anterior y un titulo con la fecha seleccionada. También se incluye un botón para que el usuario pueda añadir síntomas a ese día en concreto.



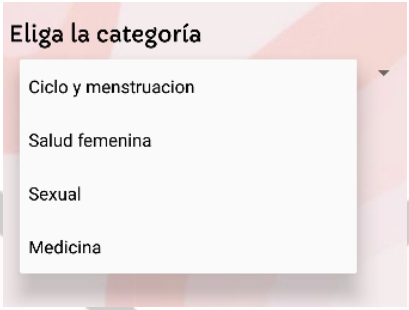
d. **Síntomas**: La ventana de síntomas incluye una serie de botones con una imagen de apoyo para que el usuario seleccione los síntomas que se ajusten a su día. Además, si el usuario esta teniendo su primer día de periodo, puede marcarlo.

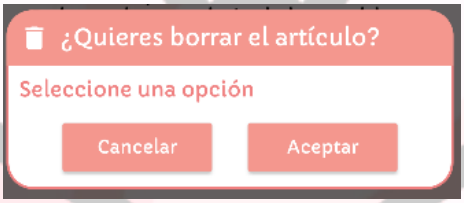
Esta ventana también incluye un botón para ir atrás.

e. **Artículos**: Esta ventana muestra los artículos disponibles para que los usuarios lo lean. En caso de que el usuario administrador, le saldría un botón flotante para añadir un artículo.

f. **Detalle del artículo**: Si el usuario presiona uno de los artículos, lo lleva a una ventana que tiene el contenido de dicho artículo. Si el usuario fuese administrador, se mostrarían dos botones flotantes para eliminar o editar el artículo .

g. **Añadir, editar y eliminar artículo**: Consta de dos actividades prácticamente iguales, sin embargo, la pantalla de editar artículo contiene los datos del artículo y la pantalla de crear está totalmente vacía, solo incluye unos campos para que sean rellenados, además de una lista de las categorías disponibles.
En cuanto a eliminar el artículo, cuando ese botón es presionado salta un diálogo donde el administrador debe confirmar si desea eliminar o no el artículo.

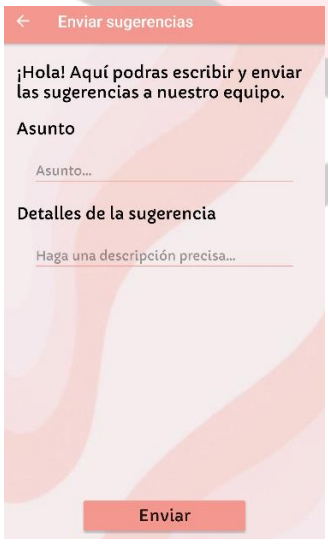


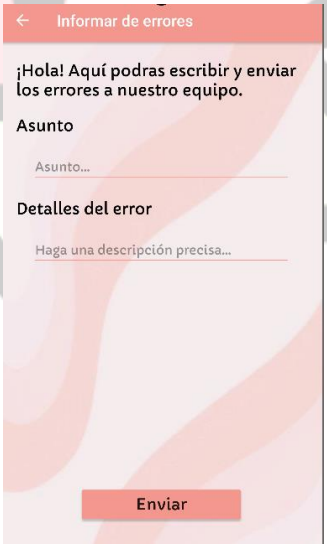


h. **Ajustes**: En esta ventana se muestra información básica del usuario, con un menú de opciones y un botón para cerrar sesión.

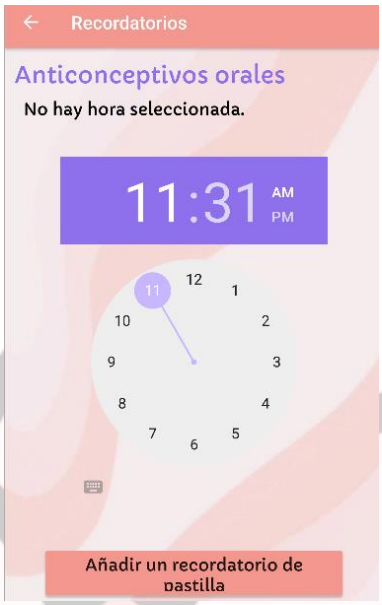
i. Tu cuenta: Esta ventana muestra el nombre de usuario, el nombre y email del usuario, también tiene la funcion de cambiar el nombre de usuario, el nombre real y cambiar la contraseña.

j. **Enviar sugerencias y errores**: Estas ventanas cumplen la función de enviar al equipo de Lotus un correo electrónico sobre alguna sugerencia o error que tenga el usuario.



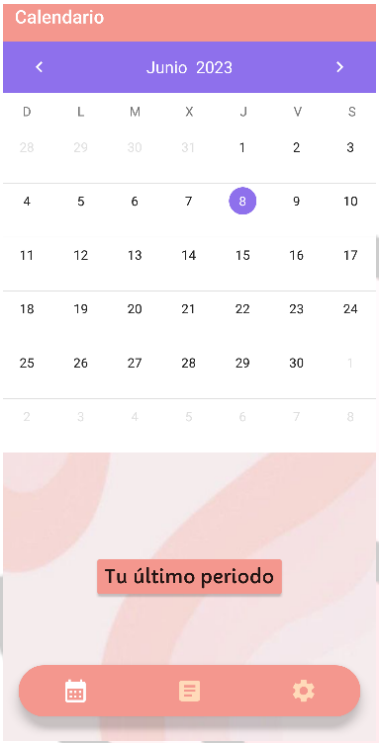


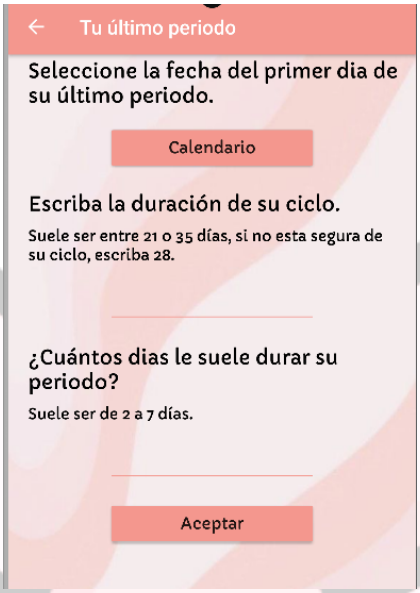
k. **Recordatorios**: Contiene la función de programar un recordatorio que avise el usuario todos los días a la hora seleccionada.

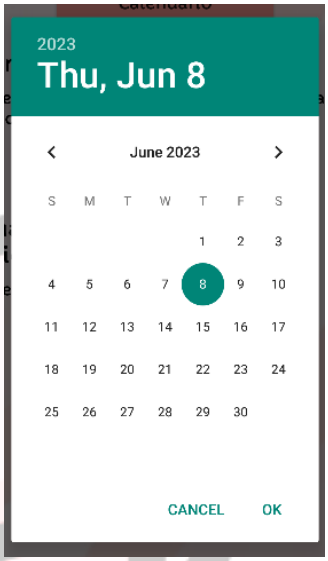


l. **Ciclo**: Esta ventana permite al usuario cambiar la duración de su ciclo y periodo.

m. **Tu último periodo**: Esta es la ventana de configuración inicial del usuario. Esta ventana se la encuentra el usuario si no hay ha hecho dicha configuración y accederá a ella a través de un botón que se encuentra en la pantalla principal. En dicha pantalla de configuración el usuario deberá escoger la fecha de su último periodo y la duración de su ciclo y periodo.







CAPÍTULO 4: IMPLEMENTACIÓN DEL PROYECTO

4.1-. INTRODUCCIÓN

En este capítulo, se presentará una visión general de la implementación del proyecto de la aplicación de calendario menstrual. El objetivo principal de esta implementación fue desarrollar una herramienta útil y sencilla de usar para que las mujeres pudiesen un seguimiento de su ciclo menstrual y brindarles información sobre su salud y bienestar.

4.2-. LENGUAJES DE PROGRAMACIÓN

A continuación, se mencionarán los lenguajes de programación con los cuales se ha trabajado para el desarrollo del proyecto.

- **Java**: Es un lenguaje de programación ampliamente utilizado y reconocido en la industria del desarrollo de software.
  - **Multiplataforma**: Es un lenguaje que se puede ejecutar en diferentes plataformas gracias a JVM (Java Virtual Machine).
  - **Amplia comunidad y soporte**: Java tiene una gran comunidad de desarrolladores, por lo que hay mucha documentación en Internet, así como cursos, por lo que hace que sea muy usado. Además, cuenta con una amplia variedad de bibliotecas y frameworks que facilitan el desarrollo de las aplicaciones.
  - **Seguridad**: Java proporciona un entorno seguro donde ejecutar el código y proporciona mecanismos para gestionar los permisos y proteger la aplicación de vulnerabilidades.
  - **Orientado a objetos**: Gracias a esto se facilita el diseño y la estructuración de aplicaciones.
- **XML**:
  - **Diseño visual**: Facilita la creación de diseños para diferentes tamaños de pantalla y orientaciones.
  - **Declarativo**: XML utiliza etiquetas y atributos para describir la estructura y las propiedades de los elementos de la interfaz de usuario. Al ser un lenguaje declarativo, XML facilita la comprensión y la lectura del código.
  - **Separación del diseño y lógica**: Al usar XML, se logra una separación entre el diseño de la interfaz y la programación de la aplicación.

4.3-. HERRAMIENTAS DE DESARROLLO

En este apartado, se mencionarán las herramientas de desarrollo usadas para desarrollar la aplicación.

- **Android Studio**: Es el IDE (Entorno de desarrollo integrado) oficial de Android. Este proporciona muchas herramientas que facilitan la creación de aplicaciones. Además de ser un editor de código, tiene un depurador, emulador de dispositivos móviles y permite crear elementos gráficos a partir de XML.
- **Firebase**: Es una plataforma de desarrollo móvil y web proporcionada por Google, la cual ofrece una variedad de servicios y herramientas, por ejemplo: base de datos en tiempo real (*Realtime Database o Firestore Database*), autenticación de usuarios (*Authentication*), almacenamiento en la nube (*Cloud Storage*), notificaciones *push* (*Messaging*), etc… Se utilizó esta herramienta ya que esta basada en la nube y no dependemos de un dispositivo físico para almacenar información, además de que ofrece muchas características y es segura.
- **Gradle**: Es un sistema, utilizado en Android Studio para compilar y administrar las dependencias del proyecto. Además, se puede gestionar las bibliotecas y recursos externos utilizados en la aplicación. *Gradle* también genera *APKs* y ayuda automatizar las tareas de compilación.
- *Material Calendar-View*: Es una biblioteca desarrollada por Applandeo que ofrece un calendario personalizado fácil de usar.

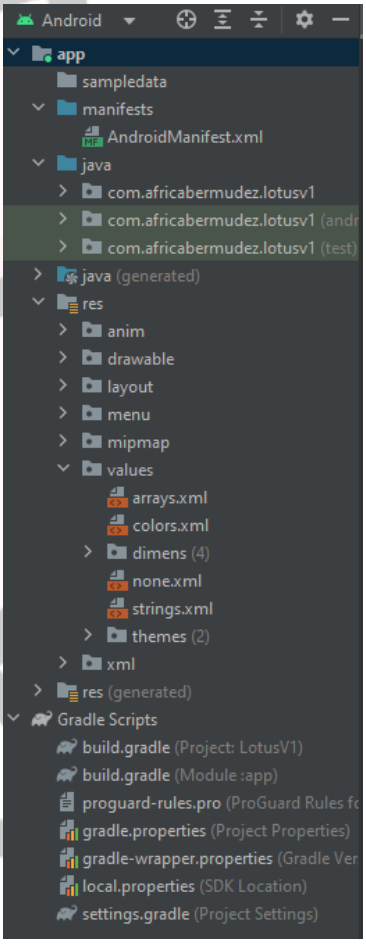
4.4-. CODIFICACIÓN

En este apartado, se detallará el proceso de codificación del proyecto, en la que se describirá la estructura del proyecto, las clases y componentes utilizados, etc. Además, se proporcionarán ejemplos de código relevante y capturas de pantalla para una mejor comprensión.

4.4.1-. ESTRUCTURA DEL PROYECTO

A continuación, se proporcionará una descripción de la estructura del proyecto. De forma predeterminada, Android Studio nos muestra los archivos del proyecto en una vista Android, esta vista no es la que esta reflejada en el disco, pero nos basaremos en ella.

1. **Directorio "app"**:
    a. **"manifest"**: Contiene el archivo manifest.xml, en él se declaran las actividades, permisos y configuraciones de la aplicación.
    b. **"java"**: Contiene los archivos del código fuente de Java o *Kotlin*, separados por paquetes. También incluye el código de prueba de *Junit*.
    c. **"res"**: Contiene todos los recursos sin código de la aplicación, como diseños *XML*, *drawables*, *layouts*, etc.
2. **Directorio "Gradle Scripts"**: Contiene los archivos de configuración de *Gradle*, utilizados para compilar y construir la aplicación.



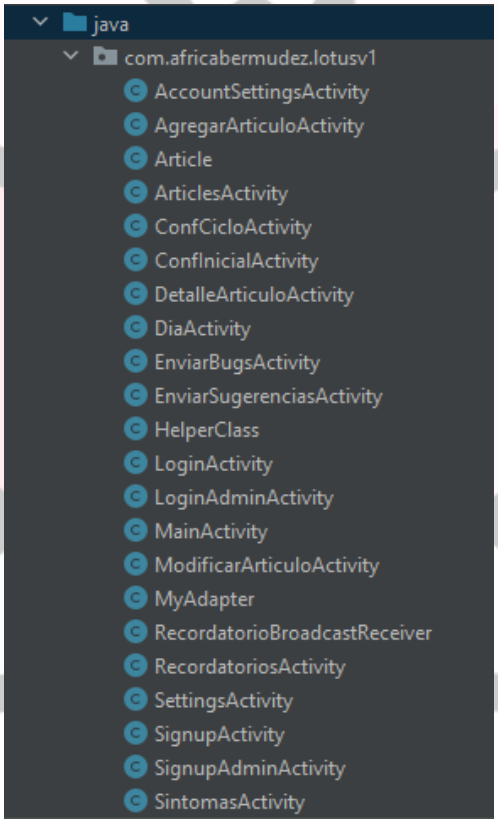
4.4.2-. CLASES

En este apartado, se comentarán las clases que se han utilizado en el proyecto.

1. **Registro e Inicio de sesión**: Estas clases se encargan de manejar el proceso de nuevos registros de usuario y administradores, como el inicio de sesión de usuarios/administradores ya existentes. Estas clases contienen métodos de validación de datos, almacenamiento de datos del usuario en la base de datos, recuperación de contraseñas y autorizaciones, entre otros.
2. **Artículo y HelperClass**: Ambas clases representan objetos utilizados en la aplicación. Estas clases tienen como finalidad, ser usadas para crear objetos de las mismas para una programación y manejo mas sencillo del proyecto. Ambas contienen los atributos, contenedores y *getters/setters* necesarios para su correcto funcionamiento.
3. **MainActvity**: Esta clase es la principal, la cual contiene un calendario que muestra al usuario iconos para un fácil seguimiento de su ciclo, además de un navegador para moverse por la aplicación.
4. **DiaActivity y SintomasActivity**: Estas clases son las encargadas de enseñar en detalle los síntomas añadidos por el usuario (DiaActivity) y de brindar diferentes opciones de síntomas para que el usuario elija los que necesite (SintomasActivity). Además, SintomasActivity, contiene métodos que inserta o elimina dentro de la rama del usuario en la base de datos, lo síntomas elegidos.
5. **ArticlesActivity, AgregarArticuloActivity, DetalleActiculoActivity y ModificarArticuloActivity**: Estas clases se encargan de mostrarte los artículos disponibles (ArticlesActivity), agregar un artículo a la base de datos (AgregarArticuloActivity), ver el contenido de un artículo previamente seleccionado (DetalleActiculoActivity) y modificar un artículo (ModificarArticuloActivity).
6. **ConfInicialActivity y ConfCicloActivity**: Ambas clases se encargan de pedir al usuario unos datos para que la aplicación calcule los ciclos e inserte información sobre ellos en la base de datos.
7. **EnviarSugerenciasActivity, EnviarBugsActivity**: Estas clases contienen unos campos que el usuario deberá rellenar para enviar un mail al equipo de Lotus.
8. **SettingsActivity**: Esta clase enseña información básica del usuario y brinda un menú de opciones en las que el usuario puede elegir y navegar por ellas.

9. **AccountSettingsActivity**: En esta clase el usuario puede ver sus datos y además cambiar algunos de ellos, como su nombre de usuario o cambiar la contraseña. Esta clase incluye verificaciones para que no haya datos repetidos en la base de datos.
10. **RecordatoriosActivity y RecordatorioBroadcastReceiver**: Estas clases se encargan de recoger una hora que el usuario elegir y establecer una alarma diaria a dicha hora. Esto sirve por si el usuario tiene que tomarse una medicación todos los días.
11. **MyAdapter**: Esta clase extiende la clase *ArrayAdapter*, que es una clase proporcionada por Android para adaptar y mostrar una lista de elementos en una vista. En este caso, la clase se ha parametrizado con el tipo "*Article*", lo que indica que se utilizará para adaptar una lista de objetos de tipo "*Article*".

A continuación, se muestra una imagen de las clases descritas anteriormente en la estructura del proyecto.



4.4.3-. FUNCIONALIDADES CLAVE

En este apartado, se mostrará las funcionalidades principales de la aplicación. Las cuales se explicarán brevemente.

1. **Obtener una instancia y referencia de Realtime Database**: Como ejemplo, vamos a obtener una referencia del usuario en la base de datos que usa la aplicación. Para ello se está obteniendo una instancia de la clase *FirebaseDatabase*, que es la clase principal de *Firebase* para interactuar con la base de datos en tiempo real. Con la llamada al método "*getInstance()"* devuelve una instancia de *FirebaseDatabase* que se utiliza para realizar operaciones de lectura y escritura en la base de datos. Además. se está obtiene otra instancia de la clase *FirebaseAuth*, que es la clase principal de *Firebase* para la autenticación de usuarios. Al llamar al método *"getInstance()"* devuelve una instancia única de *FirebaseAuth* que se utiliza para gestionar la autenticación de los usuarios de la aplicación. Con "*userId*", se está obteniendo el *UID* del usuario que esta usando la aplicación en ese momento. Por último, se obtiene una referencia a una ubicación específica, en este caso es donde se encuentran los usuarios. Con "*getReference("users")"* obtiene una referencia a la ubicación *"users"* en la base de datos, y *"child(userId)"* añade un nodo adicional con el UID del usuario actual.

```
FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
String userId = firebaseAuth.getCurrentUser().getUid();
DatabaseReference usuarioRef =
firebaseDatabase.getReference("users").child(userId);
```

Con esto ya podemos añadir o eliminar hijos o datos dentro de ese usuario.

2. **Agregar datos a la base de datos**: A continuación, se explicará como se añaden datos a la base de datos, para ello utilizaremos como referencia lo obtenido en el apartado anterior con el método *"insertarDatosEnFirebase*". Este método recibe un id con la cual trabajaremos mas adelante. Se crea una variable fecha con un formato *"day-month-year"*. Esta variable se utiliza posteriormente para crear una referencia al nodo correspondiente en la base de datos. Se crea una referencia "*fechaRef"* que señala a un nodo en la base de datos con la referencia del usuario obtenida en el apartado anterior. Con el método *"addListenerForSingleValueEvent"* de *"fechaRef*" se escucha un evento de lectura único en ese nodo, es decir, que se realiza una sola vez. Esto verifica si ya existe un nodo con la misma fecha en la base de datos. En *"onDataChange*", se verifica si el nodo con la fecha ya existe. Si el nodo existe, se obtiene una referencia al nodo "síntomas" dentro del nodo existente y se usa *"push().setValue(id)"* para agregar un nuevo valor id sin borrar los valores existentes en el nodo *"síntomas"*. En cambio, si el nodo de la fecha no existe, se crea un nuevo nodo con la fecha y se inserta el valor id utilizando *"fechaRef.child("id").setValue(fecha)"*. Luego se crea una nueva referencia *"sintomasRef"* dentro de *"fechaRef"* y se utiliza *"sintomasRef.push().setValue(id)"* para agregar el valor id al nodo *"sintomas"*. Cuando se hace algún cambio en la base de datos, se muestra un *"Toast"* para indicar que todo a salido bien. En caso de que se produzca algún error, se muestra un *"Toast"* con el error.

```
private void insertarDatosEnFirebase(final String id) {
    final String fecha = day + "-" + month + "-" + year;

    // Crea una referencia al nodo de fecha seleccionada
    DatabaseReference fechaRef = usuarioRef.child(fecha);

    // Verifica si ya existe un nodo con la misma fecha
    fechaRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) {
                // Si el nodo ya existe, se obtiene la referencia a "sintomas" dentro del nodo existente
                DatabaseReference sintomasRef = dataSnapshot.child("sintomas").getRef();
                // Agrega el nuevo valor sin borrar lo que ya existe
                sintomasRef.push().setValue(id);
                Toast.makeText(SintomasActivity.this, "Datos guardados correctamente", Toast.LENGTH_SHORT).show();
            } else {
                // El nodo no existe, se crea un nuevo nodo con la fecha y se inserta el valor "id"
                fechaRef.child("id").setValue(fecha);
                DatabaseReference sintomasRef = fechaRef.child("sintomas");
                sintomasRef.push().setValue(id);
                Toast.makeText(SintomasActivity.this, "Datos guardados correctamente", Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText(SintomasActivity.this, "Error al verificar los datos", Toast.LENGTH_SHORT).show();
        }
    });
}
```

3. **Eliminar datos de la base de datos**: Ahora se enseñará como eliminar un nodo. Como ejemplo usaremos las fechas que se crean en el apartado anterior. Con la referencia de la fecha se apunta al nodo de síntomas dentro de dicha referencia. Se realiza una consulta en *"sintomasRef"* para buscar valores que sean iguales al *"id"* pasado por parámetro. Se utiliza *"addListenerForSingleValueEvent"* para escuchar un evento de lectura de la consulta realizada. Con *"onDataChange"* se recorren los nodos dentro de "*sintomasRef*" y se eliminan usando *"snapshot.getRef().removeValue()"*. Cuando hayan sido eliminados los datos, se enseña al usuario un *"Toast"* para que sepa que los datos fueron borrados correctamente. En caso de que haya algún error, se muestra un *"Toast*" mostrando el error.

```
private void eliminarDatosDeFirebase(final String id) {
    String fecha = day + "-" + month + "-" + year;

    // Elimina el nodo correspondiente a la fecha seleccionada
    final DatabaseReference fechaRef = usuarioRef.child(fecha);
    final DatabaseReference sintomasRef = fechaRef.child("sintomas");
    Query query = sintomasRef.orderByValue().equalTo(id);
    query.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                snapshot.getRef().removeValue();
            }

            // Verifica si no quedan sintomas en el nodo de sintomas
            sintomasRef.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    if (!dataSnapshot.exists()) {
                        // No quedan sintomas, elimina el nodo fechaRef
                        fechaRef.removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task) {
                                if (task.isSuccessful()) {
                                    Toast.makeText(SintomasActivity.this, "Datos eliminados correctamente", Toast.LENGTH_SHORT).show();
                                } else {
                                    Toast.makeText(SintomasActivity.this, "Error al eliminar datos", Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    } else {
                        Toast.makeText(SintomasActivity.this, "Datos eliminados correctamente", Toast.LENGTH_SHORT).show();
                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {
                    Toast.makeText(SintomasActivity.this, "Error al eliminar datos", Toast.LENGTH_SHORT).show();
                }
            });
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText(SintomasActivity.this, "Error al eliminar datos", Toast.LENGTH_SHORT).show();
        }
    });
}
```

4. **Modificar datos**: Para este apartado, vamos a modificar un artículo de la base de datos. Contamos con que el usuario ha cambiado los datos del artículo, que son Título, Texto y Categoría y ha pulsado el botón de aceptar para que dicho artículo se modifique. Obtiene la referencia del artículo *"nuevoArticuloRef*" de la base de datos con el id (este id fue obtenido al inicio de la actividad) y se recogen los datos escritos por el usuario. Al artículo *"nuevoArticuloRef*" se le establecen los valores obtenidos a sus hijos correspondientes para que el artículo sea modificado.

```
aceptar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!tituloEdittext.getText().toString().trim().isEmpty() && !textoEdittext.getText().toString().trim().isEmpty()){

            DatabaseReference articulosRef = firebaseDatabase.getReference("articulos");
            DatabaseReference nuevoArticuloRef = articulosRef.child(id);

            final int position = categorias.getSelectedItemPosition();
            categoria = categorias.getItemAtPosition(position).toString();
            titulo = tituloEdittext.getText().toString();
            texto = textoEdittext.getText().toString();


            nuevoArticuloRef.child("categoria").setValue(categoria);
            nuevoArticuloRef.child("texto").setValue(texto);
            nuevoArticuloRef.child("titulo").setValue(titulo);

            Toast.makeText(ModificarArticuloActivity.this, "Artículo modificado.", Toast.LENGTH_LONG).show();

            Intent intent = new Intent(ModificarArticuloActivity.this, ArticlesActivity.class);
            finish();
            startActivity(intent);

        }else{
            Toast.makeText(ModificarArticuloActivity.this, "Hay algún campo vacío", Toast.LENGTH_LONG).show();
        }

    }
```

5. **Leer y obtener de datos**: En este apartado se mostrará como es una lectura de datos. Para ello se leerán y obtendrán todos los datos que hay dentro de *"categorías"* en la base de datos y se insertarán en un *"Spinner"* para que el usuario puede elegir que categoría necesita para su artículo. Primero se obtiene la referencia de *"categorías"* de la base de datos. Con *"ValueEventListener"* se escuchan todos los cambios en los datos y usando *"onDataChange"*, se recorre cada elemento hijo de *"dataSnapshot"*. Por cada elemento, se obtiene el valor de la categoría y se agrega a una lista llamada *"categoriasList"*. Finalmente se configura el *"Spinner"* con los datos de la lista.

```
private void obtenerCategorias(){

    DatabaseReference categoriasRef = firebaseDatabase.getReference("categorias");

    categoriasRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            List<String> categoriasList = new ArrayList<>();
            for (DataSnapshot categoriaSnapshot : dataSnapshot.getChildren()) {
                String categoria = categoriaSnapshot.getValue(String.class);
                categoriasList.add(categoria);
            }

            // Configura el Spinner con las categorías obtenidas
            ArrayAdapter<String> spinnerAdapter = new ArrayAdapter<>(ModificarArticuloActivity.this, android.R.layout.simple_spinner_item, categoriasList);
            spinnerAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            categorias.setAdapter(spinnerAdapter);

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });

}
```

6. **Registro de usuario**: Contando con que el usuario ha introducido los datos necesarios para el registro y que todos cumplen con los requisitos, se va a mostrar como se realiza dicho registro. Se utiliza *"auth.createUserWithEmailAndPassword(email, password)"* para crear un nuevo usuario en *Firebase Authentication* con el correo electrónico y la contraseña obtenidos. Se añade un "*OnComplete"* para escuchar la finalización de la tarea de creación de usuario. Dentro de este *"listener*", se verifica si la tarea se completó correctamente. Si la tarea es exitosa, se obtiene el usuario actualmente autenticado "*("FirebaseUser currentUser = auth.getCurrentUser())"* y se crea un objeto *"HelperClass*" con la información del usuario (UID, nombre, correo electrónico, nombre de usuario) para guardarlo en la base de datos *"Realtime Database"* utilizando *"reference.child(UID).setValue(helperclass)".* A continuación, se muestra un mensaje con *"Toast"* de que el registro fue realizado con éxito y lleva al usuario a la actividad de Inicio de sesión. En caso de que haya ocurrido algún error, se muestra un *"Toast"* con el error.

```
auth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            FirebaseUser currentUser = auth.getCurrentUser();
            String UID = currentUser.getUid();
            HelperClass helperclass = new HelperClass(UID, name, email, username);
            reference.child(UID).setValue(helperclass);
            Toast.makeText(SignupActivity.this, "Registro Exitoso", Toast.LENGTH_LONG).show();
            startActivity(new Intent(SignupActivity.this, LoginActivity.class));
            finish();
        }else{
            Toast.makeText(SignupActivity.this, "Registro Fallido"+task.getException().getMessage(), Toast.LENGTH_LONG).show();
        }
    }
});
```

CAPÍTULO 5: DESPLIEGUE DEL PROYECTO

5.1-. INTRODUCCIÓN

El despliegue del proyecto es el proceso de implementar y poner en funcionamiento la aplicación para que esté disponible y accesible a los usuarios. Esto involucra la configuración y preparación de la infraestructura necesaria, así como la instalación de la aplicación. En este apartado de detallará como ha sido el proceso de despliegue del proyecto.

5.2-. REQUISITOS DEL SISTEMA

- **Sistema operativo**: Android 8.0 o superior.
- **Navegador Web**: Se recomienda utilizar la última versión de Google Chrome, Mozilla Firefox, Microsoft Edge o Safari. El navegador web se necesita para que el usuario pueda descargarse la aplicación.
- **Hardware**: La aplicación es compatible con equipos con al menos 4 GB de RAM.
- **Conexión a internet**: Para que el usuario pueda disfrutar de esta aplicación necesitará una conexión a internet estable.
- **Resolución de pantalla**: Se recomienda una resolución mínima de 1280x800 píxeles para una visualización óptima de la interfaz.

5.3-. DESCARGA E INSTALACIÓN DE LA APLICACIÓN

Para que los usuarios puedan descargarse la aplicación se ha creado una página web. Esta web incluye una descripción de la aplicación, el equipo y manuales para que el usuario pueda descargarse la aplicación, instalarla y ejecutarla sin problema, además de un manual de uso.

Enlace: https://africabermudezmejias.es

5.4-. SOPORTE Y CONTACTO

Los usuarios de Lotus se pueden poner en contacto con el Email proporcionado en la web como enviando una sugerencia desde la aplicación a nuestro equipo.

CAPÍTULO 6: PRUEBAS DE EJECUCIÓN

6.1.-INTRODUCCIÓN

Las pruebas unitarias es una forma efectiva de comprobar el correcto funcionamiento del código de forma aislada para verificar que funcione correctamente. Estas pruebas se centran en probar el comportamiento de la unidad de código de manera independiente, sin depender de otros componentes o módulos del sistema.

En el caso de este proyecto, involucra consultas y operaciones en una base de datos en tiempo real, autenticación y una interfaz visual interactiva. Por lo cual dependemos de componentes externos como *Firebase* o *Material-Calendar-View* y dificultan la simulación o el aislamiento del código para realizar pruebas unitarias. Igualmente, se han controlado todos los errores posibles con excepciones dentro del código.

En cambio, las pruebas en ejecución son pruebas realizadas por usuarios reales o representantes del público objetivo de la aplicación, las cuales se llevan a cabo para evaluar el funcionamiento general de la aplicación y recopilar comentarios y sugerencias de los usuarios.

Este proyecto a contando en todo momento con un equipo de usuarios que han estado probando la aplicación y proporcionando *feedback* sobre el funcionamiento de la misma. Lo cual ha beneficiado al desarrollo de Lotus.

CAPÍTULO 7: CONCLUSIONES.

7.1.- CONCLUSIONES

En este proyecto, se ha desarrollado una aplicación de ciclo menstrual con el objetivo de proporcionar a las personas que menstrúan una herramienta práctica y sencilla para el seguimiento de su ciclo, la gestión de síntomas y el acceso a información relevante. Esta aplicación está desarrollada en Android Studio y programada en Java y ofrece una interfaz intuitiva y funcionalidades específicas diseñadas para mejorar la experiencia del usuario.

Esta aplicación tiene un enfoque escalable, lo cual permitirá una fácil expansión de funcionalidades en futuras versiones.

También se ha tenido mucho en cuenta la interfaz del usuario, haciéndola atractiva e intuitiva para que los usuarios disfruten al máximo de la aplicación.

Sus principales funciones se encuentran en el cálculo y seguimiento del ciclo menstrual, permitiendo a los usuarios registrar su menstruación y los ciclos, así como los síntomas asociados. Además, la aplicación ofrece información sobre salud femenina, sexual, la menstruación y ciclos, entre otros.

Durante el desarrollo, se han enfrentado desafíos como la gestión de la información sensible de los usuarios y la integración de servicios externos como *Firebase*.

En conclusión, la aplicación de ciclo menstrual desarrollada en este proyecto representa un paso significativo hacia la mejora del bienestar y la salud de las personas que menstrúan. Se espera que la aplicación contribuya a una mayor conciencia y comprensión del ciclo menstrual.

7.2.- PROPUESTAS FUTURAS

Como futuras propuestas, se plantea la posibilidad de expandir la aplicación a otras plataformas como IOS, ofrecer a los usuarios una mejor precisión de los cálculos del ciclo, más síntomas para una mayor personalización del ciclo y mas opciones de registro e inicio de sesión.

CAPÍTULO 8: BIBLIOGRAFÍA Y REFERENCIAS

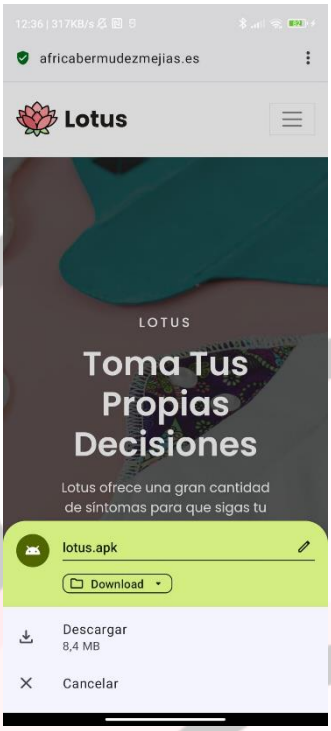
8.1.-REFERENCIAS BIBLIOGRÁFICAS.

En este apartado se citarán las referencias bibliográficas, los artículos y las páginas Web utilizadas en el proyecto.

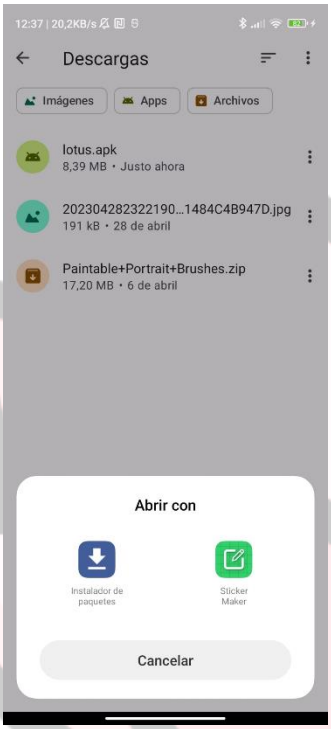
- Material-Calendar-View
- Material-Calendar-View Artículo 1
- Material-Calendar-View Artículo 2
- Coolors
- Como Enviar un Correo en Android (Java)
- Cómo agregar un botón de acción flotante
- Android Custom ListView (Adding Images, sub-title)
- How to create a custom listview in android?
- Custom ListView With CardView Layout In Android Studio Using a Java Programming Language
- Flaticon
- Aprenda Java para Desarrollo en Android: Javadoc Documentación de Código
- TimePicker
- Cómo crear una notificación
- AlarmManager
- Modelos de datos NoSQL
- 5 motivos por los que utilizar Java para desarrollar tus aplicaciones
- Ventajas de desarrollar en Java
- Diseños
- Descripción general de proyectos
- Estructura De Un Proyecto En Android Studio
- Firebase Authentication
- Firebase Realtime Database
- Android Knowledge
- Android Knowledge (YouTube)
- Clue
- Flo

ANEXO 1: MANUAL DE INSTALACIÓN

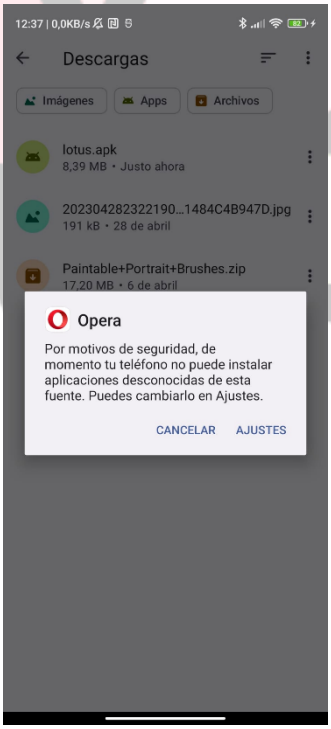
Para poder descargar la aplicación los usuarios deben situarse en la página de Lotus "africabermudezmejias.es" y descargar la aplicación pulsando los botones de descarga.

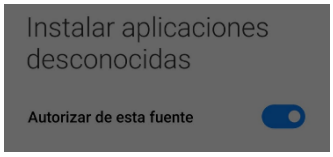


Una vez instalada la abriremos con el "instalador de paquetes".

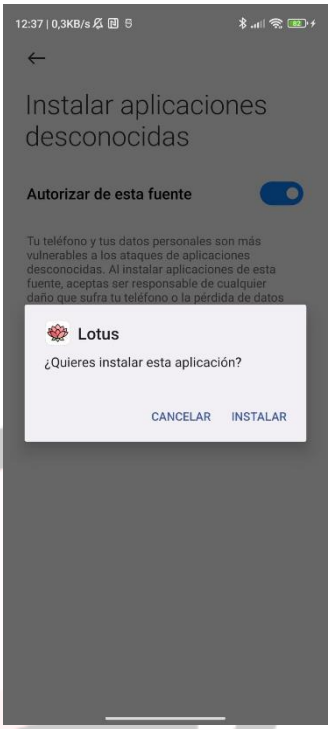


Dependiendo del navegador usado, nos dirá que no se pueden instalar aplicaciones de fuentes desconocidas. Para corregir esto nos iremos a los ajustes, autorizaremos la fuente y tendremos que aceptar que somos conscientes del riesgo.





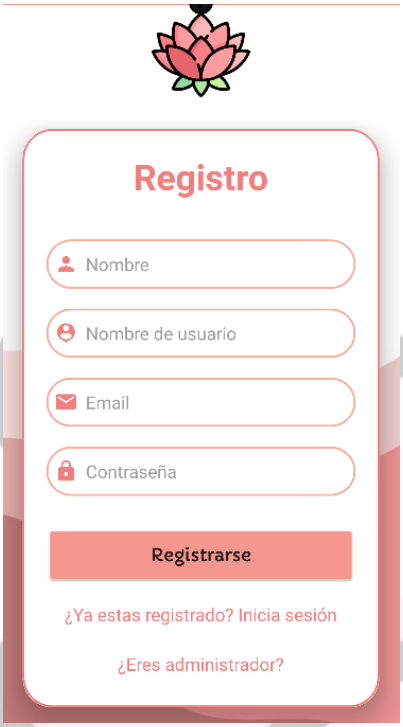
Una vez aceptados los riesgos, nos enseñará un diálogo sobre si queremos instalar la aplicación, le tenemos que dar a “instalar” y con eso ya tendremos Lotus instalado en nuestro dispositivo.



ANEXO 2: MANUAL DE USUARIO

ANEXO 2.1-. REGISTRO

Para crear una cuenta en Lotus, debemos rellenar el formulario de registro que nos aparece.



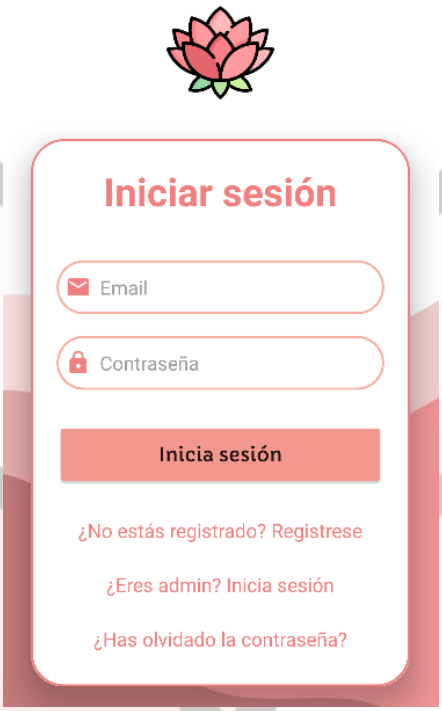
A continuación, se verán los campos del formulario.

Nombre	Nombre real del usuario
Nombre de Usuario	Nick del usuario, no es lo mismo que el nombre real.
Email	Se trata de un correo de email válido.
Contraseña	Contraseña segura que debe incluir: • Letra mayúscula. • Letra minúscula. • Número. • Símbolo. • Mínimo 8 caracteres.

Una vez introducido los datos, para terminar el registro se deberá pulsar el botón de "Registrar".

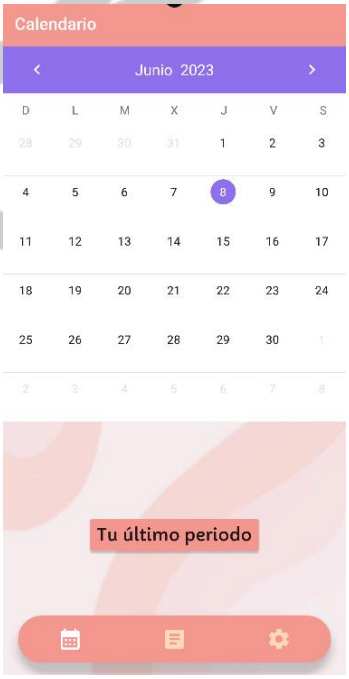
ANEXO 2.2-. INICIO DE SESIÓN

Si el usuario ya está registrado, debe situarse en la pantalla de inicio de sesión e introducir el email y su contraseña y pulsar el botón de "Inicia sesión".



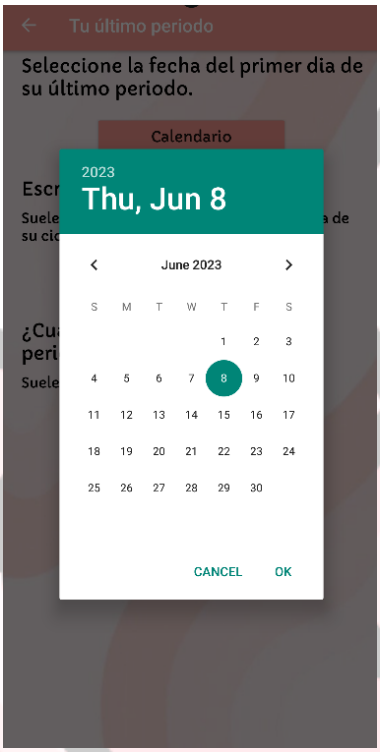
ANEXO 2.3-. CONFIGURACIÓN INICIAL

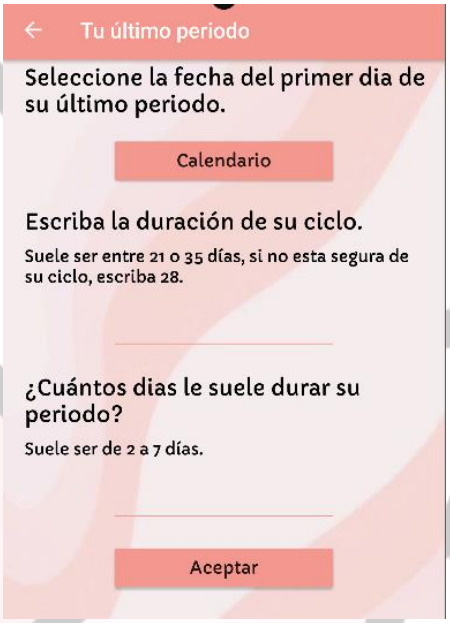
El usuario deberá realizar la configuración inicial antes de añadir cualquier síntoma al calendario. Para ello deberá pulsar el botón de "Tu último periodo".



A continuación, se deberá seleccionar la fecha del primer día del último periodo del usuario, la duración del ciclo y periodo.

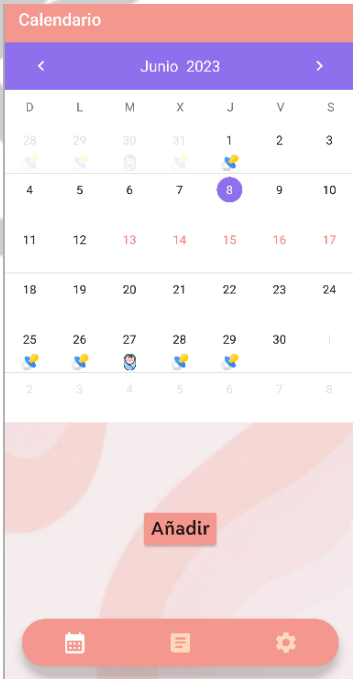
En caso de que el usuario no sepa la duración del ciclo, se recomienda poner 28 días y en cuanto al periodo, se recomienda poner 5 días.





Para confirmar los datos, el usuario deberá pulsar el botón “Aceptar”.

Ahora el usuario verá la estimación de su ciclo.

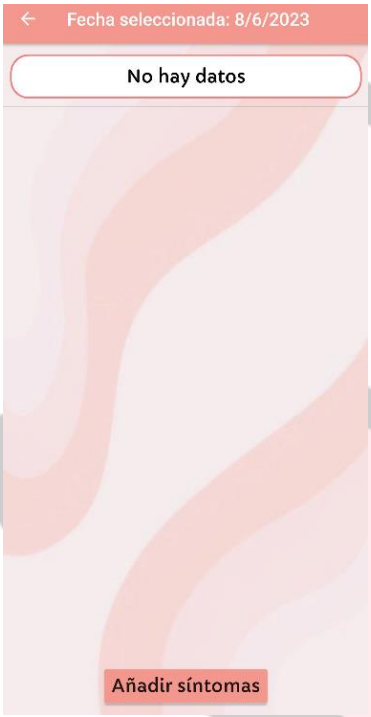


ANEXO 2.4-. AGREGAR Y VER SÍNTOMAS

Para agregar síntomas a un día o ver los que hay agregados, el usuario debe hacer clic en el día deseado del calendario y posteriormente en el botón “Añadir”

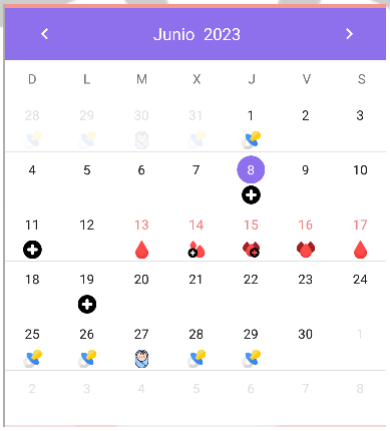
Si no hay ningún síntoma en el día seleccionado se verá en pantalla que no hay datos.

Para añadir síntomas se deberá pulsar en el botón “Añadir síntomas” y el usuario podrá seleccionar los síntomas que se ajusten a él.



ANEXO 2.5-. SÍNTOMAS

A continuación, se verán los iconos disponibles que aparecerán en el calendario y otros elementos.



Elemento	Descripción
Días resaltados en rojo 16 17 18 19 20	Son los días en los que el usuario debería tener el periodo.
	Flujo del periodo ligero.
	Flujo del periodo normal.
	Flujo del periodo abundante.
	Flujo del periodo ligero y síntomas.
	Flujo del periodo normal y síntomas.
	Flujo del periodo abundante y síntomas.
	Síntomas.
	Probabilidad media de ovulación.
	Probabilidad alta de ovulación.

ANEXO 2.5-. ARTÍCULOS

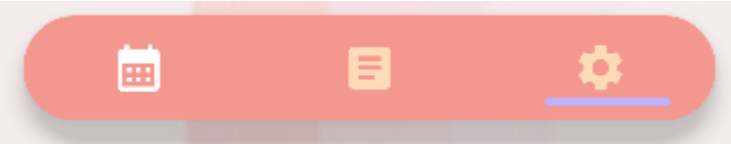
Para ver los artículos, el usuario debe desplazarse a la sección de los artículos usando el panel de opciones inferior.

En esta ventana se verán los artículos disponibles para leer.

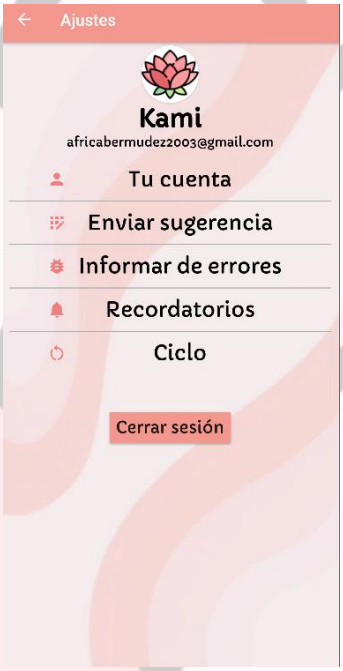
Para acceder a uno solo hay que hacer clic sobre él.

ANEXO 2.6-. AJUSTES

Para acceder a los ajustes, el usuario debe desplazarse usando el panel de opciones inferior.



En esta ventana el usuario podrá hacer cambios en datos de su cuenta, ciclo, recordatorios, enviar mails y cerrar sesión.

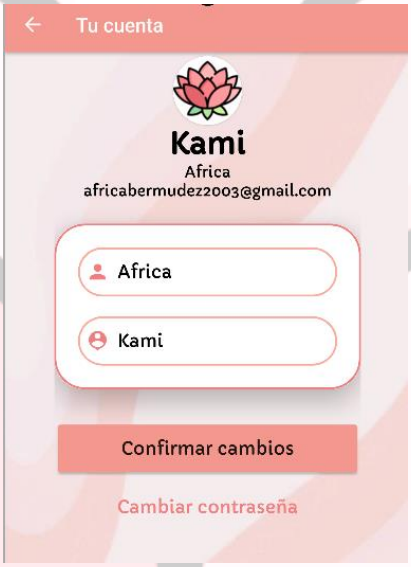


ANEXO 2.6.1-. TU CUENTA

Si el usuario desea cambiar algunos datos sobre su cuenta, debe hacer clic sobre “Tu cuenta”.

Aquí el usuario podrá cambiar los datos deseados sobre su cuenta y para confirmarlos deberá presionar el botón de “Confirmar cambios”.

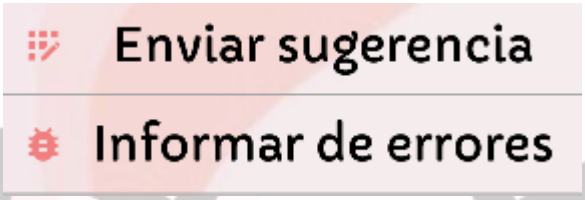
Si el usuario desea cambiar su contraseña, debe presionar sobre “Cambiar contraseña” y se le enviará un correo electrónico a su cuenta de correo.



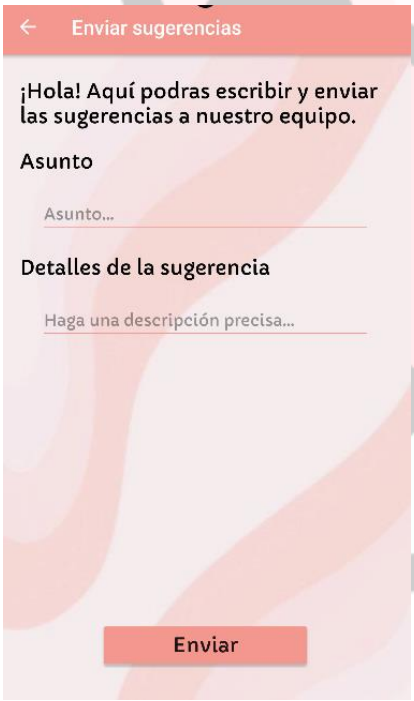
ANEXO 2.6.2-. ENVIAR SUGERENCIAS E INFORMAR DE ERRORES

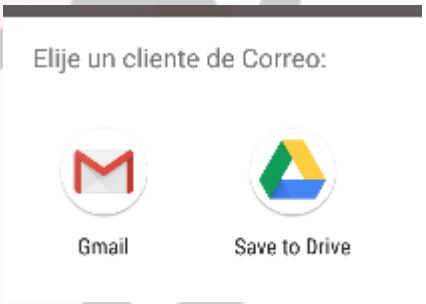
En estas ventanas en usuario puede enviar mails al equipo de Lotus sobre sugerencias o errores.

Para acceder a ellas, deberá presionar sobre “Enviar sugerencias” o “Informar de errores”.



En cada una de ellas, se deberá escribir el asunto y cuerpo del mail. Y cuando esté todo listo, se hara clic en el botón “Enviar”, el cual nos despliega un menú donde deberemos seleccionar por donde queremos enviar el correo.

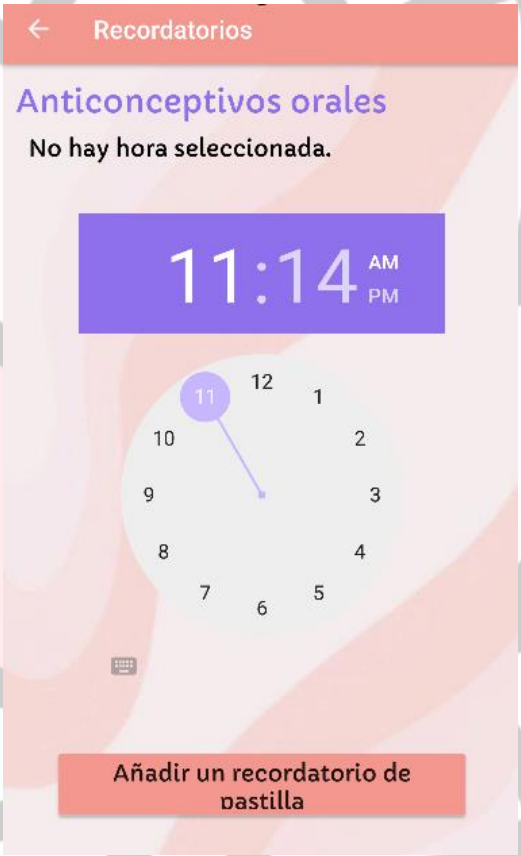




ANEXO 2.6.3-. RECORDATORIOS

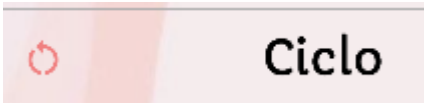
Si el usuario desea establecer recordatorios, deberá presionar sobre “Recordatorios”.

Entonces, deberá seleccionar la hora en la que el usuario quiera que se realice dicho recordatorio en el reloj y para confirmar debe hacer clic en “Añadir un recordatorio de pastilla”.



ANEXO 2.6.4-. CICLO

Si el usuario desea cambiar la duración de su ciclo o periodo, deberá presionar sobre “Ciclo”.



Entonces, el usuario podrá cambiar los datos deseados y para confirmar deberá hacer clic sobre el botón “Aceptar”.

ANEXO 2.6.5-. CERRAR SESIÓN

Si el usuario desea cerrar la sesión de su cuenta, solo debe presionar sobre el botón de “Cerrar sesión”.