

TFG ROUTINES



Alumno: *Gabi Vilaplana Garcia*

Tutor del proyecto: *Iván Martos Gázquez*

Desarrollo de Aplicaciones Multiplataforma

CIPFP BATOI 2024 - 2025

AGRADECIMIENTOS

En primer lugar, me gustaría dar las gracias a todos los profesores que he tenido, por haberme formado e introducido en este sector. Gracias a ellos ahora lo veo todo con más perspectiva y entiendo mejor el funcionamiento y la gestión de los datos.

En segundo lugar, me gustaría agradecer a la empresa en la que he hecho las prácticas, ya que me han hecho adoptar una disciplina constante de organización y optimización del tiempo para poder sacar adelante tanto el proyecto, como las horas solicitadas en la empresa.

También me gustaría agradecer a mi familia el poder permitirse que yo pueda aprovechar esta oportunidad, y que me hayan ayudado en todo lo que hayan podido.

Por último quiero agradecer a mi tutor del proyecto, Ivan por haber estado ahí siempre que he tenido alguna consulta, y haber hecho todo lo que estaba en su mano para ayudarme a solucionar problemas, y a darle un enfoque técnico al proyecto.

ÍNDICE

1. Introducción.....	3
Motivo de la elección del proyecto.....	3
Descripción y objetivos generales.....	3
Posibles aplicaciones prácticas.....	3
2. Fundamentos generales. Teóricos y prácticos.....	4
1º Curso.....	4
2º Curso.....	5
3. Etapas de desarrollo del proyecto.....	6
Primeros pasos.....	6
Crear el proyecto.....	7
Estructura inicial de la aplicación.....	8
Función adicional.....	11
Vista principal de la aplicación.....	12
Funciones clave de la aplicación.....	13
Problemas.....	22
4. Coste económico de implantación y uso.....	22
Alquilar servidor.....	23
Obtener licencias.....	23
Resumen de costes.....	23
5. Comparación a la situación actual y soluciones alternativas.....	24
6. Conclusiones.....	25
Resultados obtenidos.....	25
Puntos pendientes.....	25
Tiempo dedicado / Dificultad / Problemas más importantes.....	25
Valoración personal y auto - evaluación.....	26
Consejos.....	26
7. Webgrafía.....	27

INTRODUCCIÓN

MOTIVO DE LA ELECCIÓN DEL PROYECTO

He escogido hacer este proyecto en Visual Studio porque me pareció muy cómodo cuando lo usamos para trabajar con Unity. Me dió mucha curiosidad y fue una de las cosas que menos pesadas se me hicieron. Por otro lado se usa en Windows, que es el sistema operativo que uso en casa y, por comodidad era la opción más adecuada. Por todo esto decidí darle una oportunidad como esta e intentar aprender algo nuevo por mi cuenta, como es .NET Maui.

DESCRIPCIÓN Y OBJETIVOS GENERALES

Mi proyecto consiste en una aplicación que se encargue de realizar un monitoreo de ciertas actividades previamente disponibles como; permitir hábitos personalizados, definir con qué frecuencia se deberá realizar ese hábito, establecer un seguimiento y progreso del mismo, posibilidad de agruparlos según su categoría... entre muchas otras funciones.

También toco bases de datos sql con SQLite para guardar los perfiles de usuario y cuenta con una función que permite la extracción de un pdf con datos generales sobre el usuario registrado.

POSIBLES APLICACIONES PRÁCTICAS

Para que una persona, en su día a día, pueda llevar un registro detallado de todas las actividades que el decida apuntar en esta aplicación. Con ella el usuario puede tener una autogestión de rutinas saludables o un seguimiento de metas personales.

También le podría servir a un estudiante que quiere planificar rutinas de repaso, asistencia o entrega de tareas.

También puede ser usada por familias, sustituyendo la típica hoja con tareas colgada de la nevera. Pueden compartir esta aplicación para crear rutinas de higiene, estudio o colaboración doméstica.

Estos han sido algunos ejemplos de uso de la aplicación en la vida real de las personas.

FUNDAMENTOS GENERALES. TEÓRICOS Y PRÁCTICOS

1º curso

Programación

Esta asignatura fué la que más me costó entender, tanto que repetí un curso solo con ella. Pese a ello, creo que aproveché la oportunidad para centrar mi atención y poder comprender un poco cuales son las bases de la programación, como funciona y cómo debes interpretar los conocimientos para luego poder llevarlos a cabo.

Aprendí a desenvolverme en el entorno de programación, escribir código coherente, y darle un sentido a toda la parte 'funcional' que tiene mi aplicación.

Entornos de desarrollo

Como he dicho antes, esta asignatura es la que nos enseña a desenvolvernos un poco por el entorno de trabajo que estamos usando para crear y probar código.

Por ejemplo, en mi caso, pese a no haber tocado apenas Visual Studio más que para hacer un par de aplicaciones sencillas para el pc, gracias al haber trabajado y estudiado en el primer año esta asignatura, puedo intuir a primera vista las funciones básicas que te ofrece el entorno y cómo funciona de forma general.

Cada entorno cuenta un poco con las mismas funciones, aunque no siempre se encuentren en el mismo lugar, pero gracias a haberlo visto anteriormente, te puedes hacer una idea de cómo encontrarlas y no pierdes tanto tiempo.

Sistemas informáticos

Recuerdo que aquí dimos algunas cosas básicas como binario, octal, decimal y hexadecimal, pero lo que más me sirvió para el futuro fué usar la consola del sistema y entender que podemos hacer cualquier cosa sin tocar la interfaz gráfica del sistema operativo.

Pese a que las lecciones fueron en linux y yo he desarrollado la aplicación en Windows, he tenido que usar el powershell para unos sistemas de importación/exportación de la base de datos interna y para varios comandos sencillos que suplían ciertos botones que de repente dejaban de funcionar en el entorno de desarrollo. Así que en cierta parte, estoy agradecido de estas lecciones del primer curso por abrirme un poco los ojos en este sistema.

Base de Datos

Antes de empezar el grado había escuchado este conjunto de palabras incontables veces, pero nunca le había dado forma y me parecía un concepto muy abstracto. Gracias a esta asignatura conseguí visualizar varias bases de datos para entender su estructura y composición y pude aprender su funcionamiento y a trabajar con ellas.

En mi proyecto uso una base de datos interna llamada SQLite. En ella guardo un registro de los usuarios, de las tareas, y de las tareas que han sido cumplidas. De esta forma consigo tener una sincronización y persistencia en los datos y que cada usuario tenga los suyos, dependiendo de su actividad.

2º curso

Acceso a datos

Esta asignatura ha sido como la continuación de programación del primer año, o incluso una combinación entre esta y la base de datos. Me ha ayudado sobre todo en la práctica. Haciendo los proyectos he adquirido más soltura, nuevos conocimientos y he asimilado varios conceptos que tenía un poco esparcidos por mi cabeza.

Desarrollo de interfaces

Gracias a esta asignatura conozco Visual Studio. Aprendí a usar el IDE y a desenvolverme bien por él y, aunque para mí no tiene punto de comparación lo que hemos trabajado en clase con este proyecto, el lenguaje de programación sigue siendo C# principalmente y las opciones que te ofrece el entorno son las mismas.

Me ayudó también a la hora de realizar la instalación, conocer los datos en lo que me debía fijar y descargar lo que tocaba.

Programación multimedia y dispositivos móviles

Como mi aplicación está destinada para un dispositivo Android, me ha ayudado mucho la parte en la que en esta asignatura vimos Flutter y dart en Android Studio. Gracias a las prácticas que realizamos en ese entorno, puede extrapolar un poco los conceptos a Visual Studio y trabajar con .NET Maui.

Conseguí conectar mi dispositivo local al entorno para realizar pruebas, entender cómo funcionan las vistas, conocer lo que era SQLite y tener la posibilidad de utilizarlo para evitar tener problemas a largo plazo...

También quiero mencionar lo poco de Unity que dimos. Usamos Visual Studio y C#, aunque no haya podido relacionar tantas cosas como con Android Studio.

En resumen, el proyecto ha sido como una mezcla entre estas últimas dos asignaturas que he comentado, sumando la dificultad de hacerlo en .NET Maui, que era algo completamente desconocido para mí.

Inglés (1º y 2º curso)

Aunque sea algo muy trivial, me veo en la obligación de mencionarlo ya que todo el entorno, código, librerías, documentación, videos informativos, e incluso la aplicación inicialmente la pensé para que estuviese en Inglés, aunque después le añadí la posibilidad de cambiar el idioma.

ETAPAS DE DESARROLLO DEL PROYECTO

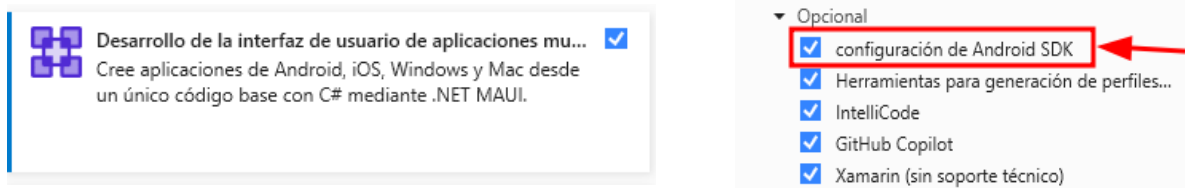
1. PRIMEROS PASOS

En primer lugar, me documenté de lo que era .NET Maui, ya que nunca antes lo había escuchado mencionar. Leí la documentación oficial de Microsoft y miré un curso que se encuentra en youtube en inglés (link en webgrafía). Después de ver y leer varias veces la documentación, comprendí un poco lo que era este framework y cómo debía utilizarse.

Con ayuda del curso y siguiendo un poco sus pasos, instalé todo lo necesario y creé el proyecto.

2. CREAR EL PROYECTO

1. Preparamos un dispositivo Android funcional con su cable correspondiente.
2. Instalamos una versión de Visual Studio superior a la 2022 17.03 (Yo uso la 17.14)
3. En el instalador, seleccionamos la carga de trabajo de la imagen (Importante marcar la opción 'configuración de Android SDK')



Una vez instalado toca sincronizar mi dispositivo.

4. En los ajustes del teléfono revelamos las opciones de desarrollador y activamos la depuración USB.
5. Volviendo al pc, en mi caso tuve que modificar el path, porque estaba situado en una carpeta de un proyecto de Unity que hicimos con Empar.
6. Si todo ha ido bien, introducimos adb devices en la consola y debería detectar el dispositivo con el número de serie y, además, nos debería dejar ejecutar y probar la aplicación en el.

Las primeras semanas las dediqué a todo lo que acabo mencionar ya que, aunque parezca un proceso corto y sencillo, me surgieron incontables errores con los que tuve que lidiar para lograr que todo funcionase. Además que la mayoría de videos útiles que encontré estaban en inglés, cosa que dificulta un poco más el entender los conceptos que se explicaban

3. ESTRUCTURA INICIAL DE LA APLICACIÓN

Empecé desarrollando las dos primeras vistas que ve el usuario al entrar a la app; la página de inicio y el login. (InitPage y LoginRegisterPage)

CONSTRUCCIÓN

InitPage

La vista InitPage actúa como pantalla de bienvenida de la aplicación. Su propósito principal es ofrecer una primera impresión visualmente atractiva al usuario antes de acceder a las funcionalidades principales.

El diseño se estructura utilizando un `AbsoluteLayout`, permitiendo superponer varios elementos:

- Una imagen azul degradada de fondo (blue.png) que ocupa toda la pantalla.

```
<AbsoluteLayout>
  <!-- Imagen de fondo -->
  <Image Source="blue.png"
    Aspect="AspectFill"
    AbsoluteLayout.LayoutBounds="0,0,1,1"
    AbsoluteLayout.LayoutFlags="All" />
</AbsoluteLayout>
```

- Un logo centrado (logo.png) acompañado del título "Welcome to Routines", que refuerza la identidad de la aplicación.

```
<Image Source="logo.png"
  WidthRequest="150"
  HeightRequest="150"
  HorizontalOptions="Center"/>

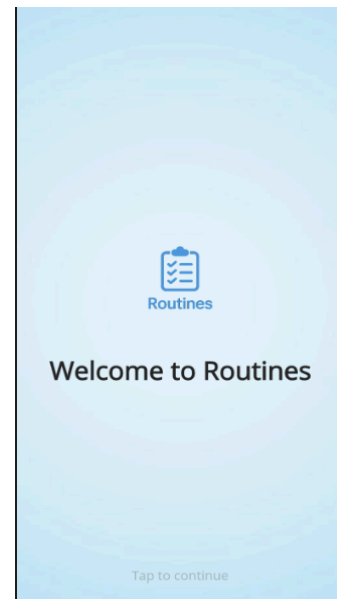
<Label Text="Welcome to Routines"
  FontSize="32"
  FontAttributes="Bold"
  TextColor="#222"
  HorizontalOptions="Center"/>
</VerticalStackLayout>
```

- Un mensaje "Tap to continue" situado en la parte inferior de la pantalla, que invita al usuario a interactuar para continuar.

```
<!-- Tap to continue abajo -->
<Label x:Name="TapToContinueLabel"
      Text="Tap to continue"
      FontSize="16"
      TextColor="Gray"
      AbsoluteLayout.LayoutBounds="0.5,1, -1, -1"
      AbsoluteLayout.LayoutFlags="PositionProportional"
      Margin="0,0,0,30">
  <Label.GestureRecognizers>
    <TapGestureRecognizer Tapped="TapArea_Tapped" />
  </Label.GestureRecognizers>
</Label>
```

Además, se ha implementado una pequeña animación en el texto "Tap to continue", haciendo que este parpadee suavemente mediante cambios de opacidad (FadeTo). Esta animación se ejecuta de forma infinita para captar la atención del usuario de manera sutil.

Cuando el usuario pulsa sobre el mensaje, se navega hacia la página de login y registro (LoginRegisterPage).



LoginRegisterPage

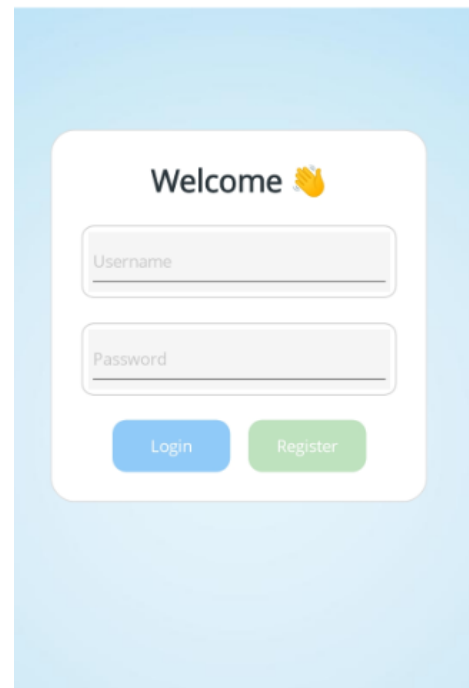
La vista *LoginRegisterPage* gestiona el acceso de los usuarios a la aplicación, permitiendo tanto iniciar sesión como registrarse. Sus funciones clave son:

- **Autenticación:** Los ingresan con su nombre de usuario y contraseña. Una función interna (OnLoginClicked) recorre la tabla de la base de datos y comprueba que el usuario con esa contraseña exista. En caso negativo, salta un mensaje de error.

- **Registro de nuevos usuarios:** Los usuarios pueden crear una nueva cuenta, añadiendo su nombre de usuario y contraseña a la base de datos. Esto ocurre gracias a una función que recoge los datos y los manda a la clase [Database.cs](#).
- **Configuración de idioma:** El idioma de la interfaz se ajusta según la configuración del usuario tras el inicio de sesión. En estas dos primeras vistas es siempre en inglés.

Respecto al diseño:

- Utiliza un **AbsoluteLayout** para colocar la imagen de fondo y un **VerticalStackLayout** para organizar los elementos del formulario de forma centrada.
- El diseño incluye botones con colores suaves y bordes redondeados. Está situada justo donde el teclado no tape nada de información cuando aparezca, optimizando la experiencia visual y la accesibilidad.



[Database.cs](#)

Una vez llegados aquí hace falta registrar los datos de inicio de sesión en algún lado, por tanto cree la clase Database.cs. Esta es la encargada de gestionar la persistencia de datos en la aplicación. Utiliza SQLite como motor de base de datos local y proporciona métodos asíncronos para interactuar con la tabla de usuarios. Antes de poder utilizarla debemos instalar la dependencia: *sqlite-net-pcl*.

Sus principales funciones son:

- **Inicialización:** Establece conexión con la base de datos y crea las tablas de User, Habit y HabitCheck si no existen previamente.

- **Gestión de usuarios:** Permite agregar nuevos usuarios, pasando los datos por varios filtros (*AddUserAsync*), obtener usuarios por nombre y contraseña (*GetUserAsync*), y actualizar la información de los usuarios (*UpdateUserAsync*).
- **Gestión de hábitos:** Permite agregar, actualizar y eliminar hábitos, además de obtener un hábito según el usuario que esté registrado (*GetHabitsByUserAsync*).
- **Validaciones:** Permite añadir validaciones a los hábitos y comprobar si un hábito se ha cumplido hoy (*AddHabitCheckAsync*, *IsHabitDoneTodayAsync*).

Toda la comunicación con SQLite se realiza de forma asíncrona (*async/await*) para no bloquear la interfaz de usuario mientras se ejecutan operaciones de lectura o escritura.

FUNCIÓN ADICIONAL

Cuando me puse a hacer el proyecto, pensé un poco a largo plazo sobre ciertos aspectos de él. Uno de ellos fué la base de datos. Vista mi experiencia anterior presentando proyectos que funcionaban en casa, pero en el centro no iban según lo esperado, decidí asegurarme todo lo posible de que este saliese bien y valoré la posibilidad de cambiar la base de datos externa que había propuesto (XAMPP), por alguna interna.

Como conocimiento previo sobre alguna de ellas tenía SQLite, que usamos en Android Studio y recordé bastante sencilla de implementar y usar, así que decidí darle una oportunidad. Implementar esta base de datos fué relativamente sencillo, lo complicado vino más tarde, cuando quise poder ver y modificar las tablas creadas. No era algo tan sencillo como acceder a un archivo que se encuentra dentro del dispositivo, ya que este era casi inaccesible.

Para ello cree una clase [DebugTools.cs](#), que sería la encargada de realizar la importación y exportación de este archivo oculto, para poder trabajar con él. Fué bastante complicado lograr que el código funcionase, y tuve que añadir varios permisos adicionales a *AndroidManifest.xml* (*WRITE_EXTERNAL_STORAGE*, *READ_EXTERNAL_STORAGE*, *MANAGE_EXTERNAL_STORAGE*), pero finalmente conseguí que funcionase.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" tools:ignore="ScopedStorage" />
```

4. VISTA PRINCIPAL DE LA APLICACIÓN

Cuando logré que todo lo anterior funcionase a la perfección me sentí un poco bloqueado porque no sabía cómo continuar el proyecto. No sabía que forma darle para que se quedara una aplicación bonita y con potencial, así que le consulté a mi tutor para que me orientara un poco. Seguí su consejo de crearme un pequeño esquema de lo que me quedaba por hacer y cómo lo quería reflejar y decidí crear una vista principal a continuación de lo que ya teníamos, en la que pudiéramos acceder a todas las funciones que había propuesto en la solicitud.

Era una forma sencilla y eficaz de seguir adelante con el proyecto. Cree MainMenuPage.

CONSTRUCCIÓN

MainMenuPage

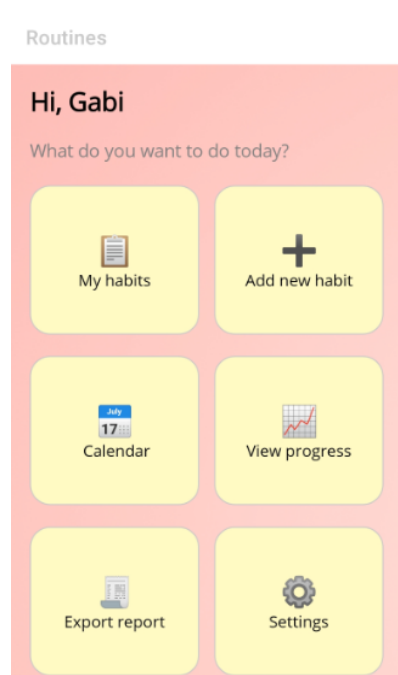
Esta es la vista principal de la aplicación, que se encarga de proporcionar acceso a las funcionalidades clave. Está diseñada para servir como menú principal donde el usuario puede navegar fácilmente entre las distintas opciones de la aplicación, todo con un diseño visual atractivo y fácil de usar. Sus características principales son:

1. Interfaz de usuario:

- Formada por varios botones (representados como Frame con íconos y texto), que permiten al usuario navegar hacia diferentes vistas dentro de la aplicación. (*HabitsListPage*, *AddHabitPage*, *CalendarPage*...)

2. Estructura visual:

- La disposición visual utiliza un Grid para organizar los botones en 3 filas y 2 columnas, logrando una disposición clara y accesible.



- El fondo de la aplicación es configurable, lo que permite personalizar la apariencia según la preferencia del usuario (se aplica mediante `SetUserBackground()`).

3. Acciones de navegación:

- Cada (`On...Clicked`) dirige al usuario a la vista que corresponde a esa función.

También cuenta con los ajustes necesarios para cambiar idioma o fondo en función de lo que haya creado el usuario. Lo veremos más tarde en `SettingsPage`.

5. FUNCIONES CLAVE DE LA APLICACIÓN

Una vez creada la vista principal, tocaba ponerse con las funciones descritas en la propuesta del proyecto. Mi idea fue crear una vista para cada una, aunque luego añadí algunos detalles a muchas de ellas.

CONSTRUCCIÓN

HabitsListPage

Esta vista gestiona la visualización y edición de los hábitos registrados por el usuario. Sus características principales son:

- **Carga de hábitos:** Al cargar la vista, `LoadHabits()` obtiene todos los hábitos asociados al usuario actual desde la base de datos. Luego, se verifica si cada hábito ha sido completado en el día de hoy. El color de fondo de cada registro varía en función si ya ha sido completado o no.
- **Selección de hábito para edición:** Cuando el usuario selecciona un hábito de la lista, la vista navega a la **EditHabitPage**, donde el usuario puede modificar los detalles del hábito seleccionado. (`OnHabitSelected`)





- **Marcar un hábito como completado:** La función OnHabitCheckClicked permite marcar un hábito como completado o eliminarlo, si el hábito ya ha sido marcado. Si no ha sido completado, se agrega una nueva entrada en la base de datos para registrar su cumplimiento. La interfaz se actualiza automáticamente después de estas acciones.

Además, si no hay hábitos, se muestra un mensaje indicando que no hay datos.

Para crear esta vista tuve varios problemas con los que tuve que lidiar creando algunas clases extra.

BoolToIconConverter.cs:

- **Función:** Convierte un valor booleano (true o false) en un icono.
 - True →  (indicando que el hábito ha sido completado).
 - False →  (indicando que el hábito no ha sido completado).

En HabitsListPage se aplica al botón que permite marcar un hábito como completado.

BoolToIconColorConverter.cs:

- **Función:** Convierte un valor booleano (true o false) en un color para el icono.
 - True → Rojo (Colors.Red).
 - False → Verde (Colors.Green).

En HabitsListPage se aplica al icono que nos permite seleccionar una tarea como realizada, o eliminarla.

BoolToBackgroundConverter.cs:

- **Función:** Convierte un valor booleano (true o false) en el color de fondo de un elemento.
 - True → Azul muy suave y claro.
 - False → Transparente.

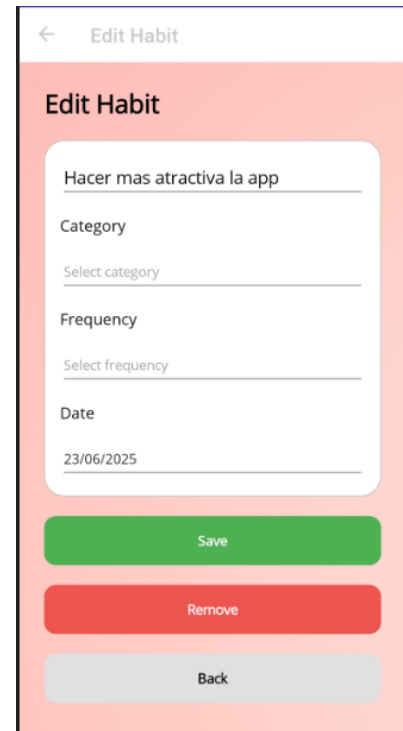
En HabitsListPage se aplica al fondo que contiene cada hábito. Si ha sido completado, se coloreará.

Puede parecer que este conjunto de clases son ciertamente relevantes y que todo se podría solucionar con una, pero lo probé todo para dejarlo lo más comprimido posible y si no era de esta manera, los iconos seleccionables se veían cortados, los verdes a veces no se veían, no se actualizaban según tenían que hacerlo... Así que esta fue la única solución que encontré, hacerlo en clases separadas, guardadas dentro de una carpeta /Utils.

EditHabitPage

Vista que complementa la anterior. Permite a los usuarios modificar los detalles de un hábito existente. Sus características clave incluyen:

1. **Campos de edición:** Permite editar el título, la categoría, la frecuencia y la fecha asignada del hábito.
2. **Acciones:** Tres botones:
 - *Guardar cambios:* Actualiza los detalles del hábito en la base de datos.
 - *Eliminar:* Elimina el hábito tras confirmación.
 - *Cancelar:* Abandona la vista sin guardar cambios.



The screenshot shows a mobile application interface for editing a habit. At the top, there is a back arrow and the title 'Edit Habit'. Below this, the main title 'Edit Habit' is displayed. The form contains the following fields: a title field with the text 'Hacer mas atractiva la app', a category field with a dropdown menu showing 'Select category', a frequency field with a dropdown menu showing 'Select frequency', and a date field with the text '23/06/2025'. At the bottom of the form, there are three buttons: a green 'Save' button, a red 'Remove' button, and a grey 'Back' button.

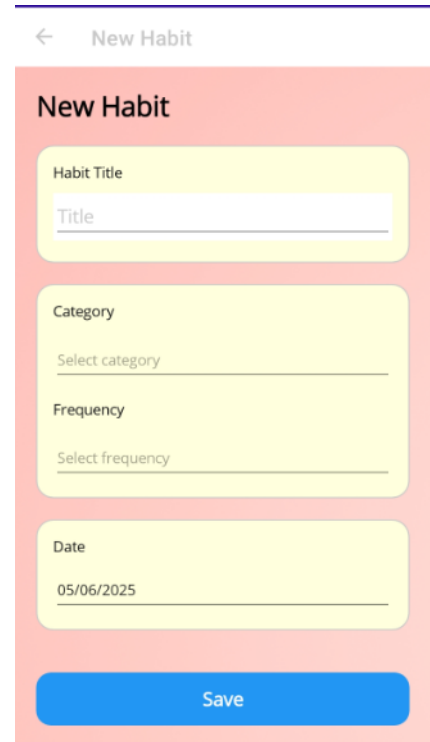
3. **Carga de datos:** Los datos del hábito seleccionado se cargan automáticamente al acceder a la página.
4. **Gestión de base de datos:** Utiliza métodos de la clase Database para actualizar o eliminar hábitos.

Esta página facilita la edición de hábitos con una interfaz clara y fácil de usar, vinculada a la base de datos local para almacenar los cambios.

AddHabitPage

Destinada a permitir a los usuarios crear nuevos hábitos en la aplicación. Sus características clave son:

1. **Campos de entrada:** La vista permite al usuario ingresar los detalles del nuevo hábito (Título, Categoría, Frecuencia y fecha).
2. **Interacción:**
 - El usuario puede guardar el nuevo hábito, lo que lo agrega a la base de datos o ir a la ventana anterior pulsando sobre la flecha que se encuentra arriba a la izquierda.
3. **Datos y funcionalidad:**
 - Utiliza la clase *Database* para agregar el nuevo hábito a la base de datos, lo que permite la persistencia de los datos del usuario.
 - Las categorías y frecuencias están localizadas para adaptarse a diferentes idiomas mediante el uso de un sistema de traducción.



The screenshot shows a mobile application interface for creating a new habit. At the top, there is a white header bar with a back arrow on the left and the text 'New Habit' on the right. Below this is a pink rectangular area containing the form. The form is titled 'New Habit' in bold black text. It consists of four input fields, each with a label and a placeholder text: 'Habit Title' (placeholder: 'Title'), 'Category' (placeholder: 'Select category'), 'Frequency' (placeholder: 'Select frequency'), and 'Date' (placeholder: '05/06/2025'). At the bottom of the pink area is a blue button with the text 'Save' in white.

CalendarPage

Permite a los usuarios visualizar un calendario interactivo donde se muestran las tareas o hábitos asignados para cada día del mes.

Intenté encontrar alguna extensión que me facilitase el trabajo y mostrase un calendario con mejor estructura y forma, pero no hubo forma de hacerlo funcionar. Había muchos conflictos con partes ya hechas del proyecto. Por tanto decidí hacer una vista personalizada que utiliza funcionalidades propias de .NET MAUI. Sus características son:

1. Visualización de los días de la semana:

La vista contiene una fila en la parte superior con los días de la semana (lunes a domingo), los cuales se generan dinámicamente y se localizan según el idioma seleccionado. Esto se logra mediante un Grid que se llena con etiquetas que representan los días.

2. Generación dinámica del calendario:

El calendario se genera de manera dinámica sin utilizar controles predefinidos. Dependiendo del mes y año seleccionados, se calcula la cantidad de días del mes, el primer día de la semana y las celdas correspondientes a cada día.

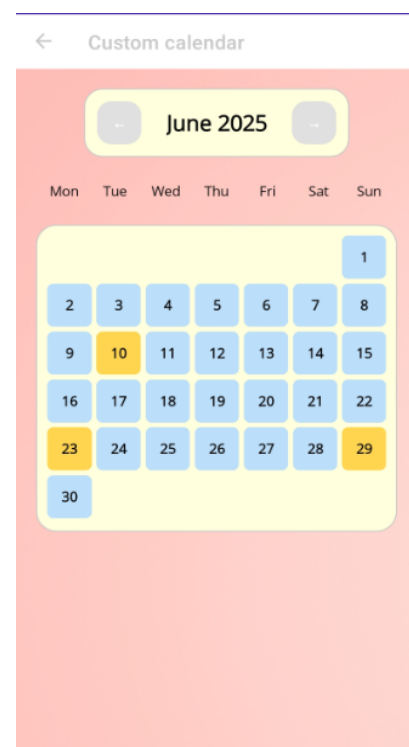
Cada día se representa mediante un Button, cuyo texto es el número del día y cuya apariencia (color de fondo) cambia si hay tareas asignadas para ese día. Las celdas vacías de días no pertenecientes al mes actual también se generan correctamente.

3. Interactividad:

Los botones correspondientes a los días del mes permiten al usuario hacer clic para ver un resumen de las tareas asignadas para esa fecha. Si el día tiene tareas asignadas, se muestra una lista de ellas; si no tiene tareas, se muestra un mensaje indicativo.

4. Navegación entre meses:

Los botones de flecha a la izquierda y a la derecha permiten al usuario navegar entre los meses, actualizando el calendario para mostrar el mes anterior o siguiente.



ProgressPage

Permite al usuario visualizar estadísticas relacionadas con sus hábitos. Muestra información relevante de manera clara y accesible, como la cantidad de hábitos creados, el número de cumplimientos registrados, y los hábitos más realizados. Realiza un recuento de:

1. Hábitos creados y cumplimientos registrados:

Presenta un contador de los hábitos creados (Hábitos que se encuentran en la lista que se muestra en HabitsListPage) y de los cumplimientos registrados. Esto son las tareas que se han llevado a cabo y se han marcado como 'hechas'

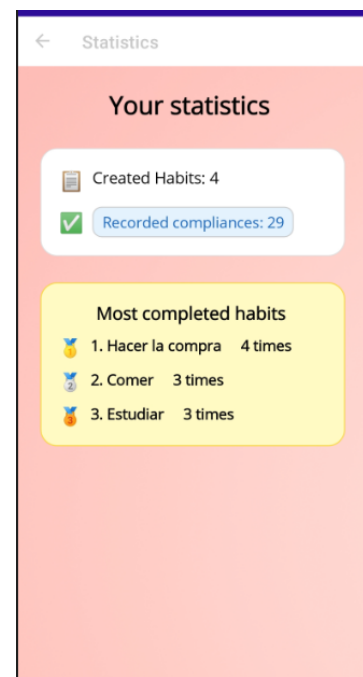
2. Top 3 hábitos más realizados:

La página incluye una sección que destaca los tres hábitos más repetidos por el usuario. Esta sección está organizada en un formato de medallas (🥇, 🥈, 🥉), lo que añade un toque visual y gamificado a la aplicación.

3. Interacción con los cumplimientos:

Vi interesante añadir una vista que muestre todos los hábitos que han sido marcados como hechos. De esta forma el usuario tiene la posibilidad de entrar a hacer el recuento manualmente si lo cree necesario, e incluso puede comprobar si el top 3 está bien calculado.

La mejor forma que se me ocurrió de navegar hacia esta nueva vista es que el texto 'Cumplimientos registrados' sea un botón que te dirija a ella. Lo hice un poco más intuitivo al mejorar la estética de la aplicación.



HabitsChecksPage

Esta vista la consideraría un complemento de la otra, aunque tiene su dificultad. Tuve tantos problemas al intentar leer y mostrar la lista de elementos realizados, que decidí crear una nueva tabla con estos elementos y mostrarla directamente.

```
1 referencia
public Database(string dbPath)
{
    _database = new SQLiteAsyncConnection(dbPath);

    _database.CreateTableAsync<User>().Wait();
    _database.CreateTableAsync<Habit>().Wait();
    _database.CreateTableAsync<HabitCheck>().Wait();
}
```

HabitsCheckPage es una vista que muestra el historial de los hábitos cumplidos por el usuario, permitiendo a este revisarlos todos en caso de que lo crea necesario.

1. Listado de hábitos cumplidos:

La página presenta un ListView donde se visualizan las entradas de hábitos registrados por el usuario. Cada entrada incluye el nombre del hábito y la fecha en que se cumplió.

Están ordenados de forma descendente, de forma que los más recientes aparecen primero.

2. Interacción visual:

El diseño utiliza un ListView con celdas de vista personalizada, mostrando cada cumplimiento con un formato claro y legible. Los hábitos cumplidos están destacados por el texto en negrita, con la fecha de cumplimiento en un tamaño más pequeño y en color gris.



ExportPage

En esta vista me ocurrió lo mismo que con la del calendario. Probé varias extensiones que vi que la gente recomendaba por internet, pero todas daban conflictos en alguna parte del código que ya había construido. Por tanto decidí construir la vista con funcionalidades propias de .NET MAUI. Cree una clase auxiliar llamada AndroidPdfExporter.cs , que se encarga de contener la lógica de creación de la estructura del pdf. Las funciones destacables de ExportPage son:

1. Generación de Informe PDF:

Permite generar un informe que detalle las estadísticas de los hábitos del usuario y las verificaciones realizadas. Es un resumen de la vista de progreso, plasmado en este formato.

2. Interacción con AndroidPdfExporter:

ExportPage depende de la clase AndroidPdfExporter para crear y guardar el PDF. Cuando el usuario hace clic en el botón "Exportar a PDF", el método OnExportClicked se ejecuta.

En este método, se recogen los datos de los hábitos y verificaciones a través de la base de datos local. Posteriormente, se llama a la función CrearInforme de AndroidPdfExporter, pasando la información necesaria (usuario, hábitos y verificaciones) para crear el PDF y guardarlo en la ubicación especificada del dispositivo Android.

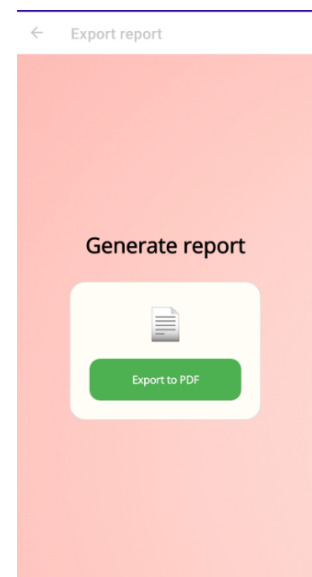
3. Interfaz de Usuario:

La vista de ExportPage está diseñada para ser simple y centrada en el usuario. En ella, el usuario encuentra un título claro y un botón de acción para generar el reporte..

4. Compatibilidad con Android:

Esta funcionalidad está actualmente implementada solo para la plataforma Android, como se indica en la clase AndroidPdfExporter que utiliza APIs específicas de Android para generar y guardar el PDF en el almacenamiento del dispositivo.

En plataformas distintas, se muestra un mensaje de error indicando que la exportación no está disponible. Una posible mejora a futuro puede ser el añadir esta funcionalidad para todo tipo de plataformas.



SettingsPage

SettingsPage es una vista dedicada a permitir que el usuario personalice ciertos aspectos de la aplicación, como el fondo y el idioma. Consta de dos funciones principales:

1. **Cambio de fondo:** El usuario puede seleccionar entre varios colores de fondo (azul, verde, rojo, amarillo, naranja, y morado). Cada vez que se selecciona un color, se actualiza la imagen de fondo de la aplicación, lo que afecta a todas las vistas. Esto se gestiona a través de la clase **BackgroundManager**, que maneja la preferencia del fondo seleccionada y la almacena en las preferencias del dispositivo. Se utiliza un *Button* en la vista para que el usuario elija el color, y se guarda el color seleccionado en la base de datos y en las preferencias locales.

Para evitar la recarga constante de datos, se suscribe a un **MessagingCenter** para que, al cambiar el fondo, todas las vistas se actualicen con la nueva imagen.

2. **Selección de idioma:** A través de un *Picker*, el usuario puede elegir entre Español e Inglés. Al seleccionar un idioma, la aplicación recarga la vista con las traducciones correspondientes. Utiliza la clase **LocalizationResourceManager**, que se encarga de gestionar las traducciones en función del idioma seleccionado. Los cambios de idioma también se actualizan en la base de datos y en las preferencias locales del usuario, lo que permite que la selección del idioma sea persistente en el dispositivo.

Para la correcta actualización del fondo y el idioma en toda la aplicación, se utilizan métodos como *OnBackgroundSelected* y *OnLanguageSelected*, que actualizan tanto la interfaz visual como las preferencias del usuario dinámicamente sin tener que cerrar la aplicación. Estas son dos clases auxiliares que me permiten gestionar:

- **Fondo:** La clase **BackgroundManager** gestiona el fondo global de la aplicación, permitiendo que se actualice dinámicamente en todas las vistas sin necesidad de recargar manualmente cada una.
- **Traducción:** El **LocalizationResourceManager** se encarga de gestionar las traducciones globales. Cualquier cambio en el idioma se refleja automáticamente en toda la aplicación gracias a la notificación del evento de propiedad *OnPropertyChanged*, que se dispara cuando se cambia el idioma. Las vistas utilizan el **TranslateExtension** para aplicar las traducciones a los elementos visuales en función del idioma seleccionado.

Esta fué la vista que más me costó implementar debido a la cantidad de errores que se presentaban. Cuando arreglaba uno, salían otros tres que ya había solucionado previamente, y así le dediqué muchas horas.

6. PROBLEMAS

He tenido muchísimos problemas a lo largo del desarrollo, pero los que más me ha costado resolver han sido lo de la exportación e importación de las tablas a modo de depuración y que el fondo y las traducciones sean persistentes.

Importación y exportación

El tema de la importación y exportación lo arreglé a base de probar a añadir y quitar permisos, buscando tutoriales y preguntado a IAs cómo debía ser la lógica para que me permitiese copiar con éxito el [archivo.db](#) a una carpeta desde la cual pudiera hacer un pull desde el pc. Era un proceso tedioso ya que incluía la consola y estar introduciendo comandos, y el ir reiniciando la app todo el rato.

Fondo e idioma personalizados y dinámicos

Un poco lo mismo ocurría con el fondo y el idioma. Para ver si los cambios surgían efecto debía reiniciar, y cuando no funciona tantas veces empiezas a desesperarte. El problema aquí fué que el fondo y la traducción no cambiaban dinámicamente en la vista MainMenuPage, cuando en el resto de la aplicación sí lo hacía. Para que los cambios surgieran efecto tenía que reiniciar, y ese no es el objetivo de la App. Finalmente con LocalizationResouceManager hallé una forma de que el programa leyese las traducciones y solucionar este problema.

COSTE ECONÓMICO DE IMPLANTACIÓN Y USO

Para lograr implementar la aplicación que hemos desarrollado, debemos tener en cuenta que los costes van a ir dirigidos sobre todo en la **infraestructura de servidores** y en obtener las **licencias** necesarias para su publicación en las plataformas de distribución de aplicaciones móviles.

ALQUILAR SERVIDOR

La aplicación no requiere de un servidor dedicado a su ejecución, ya que la funcionalidad principal se ejecuta en el dispositivo móvil de los usuarios. Sin embargo, se necesita un servidor para manejar aspectos como el almacenamiento de datos de usuario y la base de datos.

Dada la magnitud y el poco impacto que realizaría nuestra aplicación sobre un servidor, es recomendable considerar el alquiler en vez de comprar uno. Comprar es mucho más caro y puede ser un riesgo a largo plazo, en caso de que no obtengamos las ventas suficientes. El precio de un servidor virtual básico es de aproximadamente 5€/mes, que será suficiente para gestionar las solicitudes de datos y el almacenamiento necesario.

OBTENER LICENCIAS

La aplicación será publicada tanto en la App Store de Apple como en la Google Play Store. Ambas plataformas requieren licencias para poder subir aplicaciones.

Licencia para Apple (App Store): La licencia anual para poder subir aplicaciones a la App Store tiene un coste de 99€ al año.

Licencia para Google (Play Store): En Google Play, la licencia es de un pago único de 25€ para crear una cuenta de desarrollador, lo que permite publicar aplicaciones en la tienda.

RESUMEN DE COSTES

COMPONENTE	PRECIO
Alquilar servidor virtual	5€ / mes
Subir a la App Store	99€ / año
Subir a la Play Store	25€ <i>(pago único)</i>
Precio total implantación	129€ <i>(primer mes de servidor incluido)</i>
Precio total anual	164€ <i>(servidor anual + renovación App Store)</i>

El uso de la aplicación es gratuito para los usuarios. Las posibles fuentes de ingresos podrían provenir de futuras implementaciones de publicidad en la app o funciones premium si se considerara una opción de monetización.

El servidor virtual alquilado se podría ubicar en proveedores como ionos.es o AWS.

COMPARACIÓN A LA SITUACIÓN ACTUAL Y SOLUCIONES ALTERNATIVAS

En el ámbito de las aplicaciones de seguimiento de hábitos, existen varias opciones populares que ofrecen funcionalidades similares a las de mi aplicación. He hecho una tabla con algunas de las que he encontrado

Aplicación	Plataforma	Características destacadas
Streaks	iOS	Seguimiento de hasta 12 hábitos, integración con Apple Health, interfaz minimalista.
HabitNow	Android	Plan gratuito con funciones suficientes, seguimiento de hábitos ilimitados en la versión premium.
Way of Life	iOS / Android	Seguimiento de hábitos basado en diarios, flexibilidad en las rachas.
Habitica	iOS / Android	Gamificación del seguimiento de hábitos, recompensas por completar tareas.
Habitify	iOS / Android	Organización de hábitos por áreas de la vida, interfaz atractiva.

Nuestra aplicación se diferencia al ofrecer una integración con la base de datos local, permitiendo un seguimiento detallado de los hábitos y su cumplimiento. Además, implementa características como la personalización del fondo de pantalla y la traducción dinámica, mejorando la accesibilidad y la experiencia del usuario.

En cuanto a soluciones alternativas, existen dispositivos y tecnologías de asistencia que ayudan a las personas con discapacidad visual a interactuar con su entorno. Sin embargo, estas soluciones suelen estar más centradas en la navegación y la identificación de objetos, y no en el seguimiento de hábitos. Por lo tanto, nuestra aplicación ofrece una propuesta única al combinar el seguimiento de hábitos con características de accesibilidad.

CONCLUSIONES

Resultados obtenidos

La verdad es que estoy muy contento con el resultado final de la aplicación y de cómo ha quedado. Me he sorprendido a mi mismo al ser capaz de poder sacar todo adelante cuando el tiempo iba tan ajustado. En un día tenía que hacer las 8 horas de las prácticas, entrenar y dedicar 2 o 3 horas al proyecto. Todo esto habiendo dormido entre 5 y 7 horas.

Por eso estoy bastante orgulloso de la aplicación que se ha quedado. Para haber partido de 0 en la mayoría de los aspectos, tiene casi todas las funciones que propuse y me parece bastante atractiva visualmente.

Puntos pendientes

En principio, en la página de progreso, quería poner algún tipo de tabla o gráfico que me mostrase algún tipo de información relevante para el usuario, pero debido a su complejidad, a la falta de tiempo y a la necesidad de seguir adelantando el proyecto, lo fui aplazando hasta que al final terminé por no implementarlo.

También me ha faltado implementar un sistema de notificaciones. El motivo es que simplemente no me acordé de que lo puse y, por tanto, no lo tuve en cuenta, pero planteándolo ahora se me ocurre donde implementarlo y que función podría cubrir, así que puede ser una buena mejora a futuro. Todo es cuestión de seguir documentándose sobre cómo funciona y implementarlo.

Tiempo dedicado / Dificultad del proyecto / Problemas más importantes

La verdad es que no llevo la cuenta de las horas que le habré dedicado pero, teniendo en cuenta que el proyecto está pensado para hacerse en 40, puedo asegurar que he hecho más del doble o incluso del triple.

He dedicado todos los días al menos un par de horas a trabajar en él y, en este último mes, he pasado los fines de semana encerrado porque había fallos que no sabía solucionar y los tuve que dedicar la mayoría del día a arreglarlos para no quedarme sin tiempo.

La dificultad ha sido bastante elevada porque, aunque sepa moverme por el código y pueda entenderlo si me concentro en leerlo y analizarlo, no sabía absolutamente nada de .NET MAUI ni de cómo se usaba en Visual Studio. Como he dicho antes, partía de absoluto 0.

Ya lo he mencionado y explicado anteriormente, pero los fallos que más me costó solucionar fueron la persistencia en el fondo y en el cambio de idioma, y el importar y exportar la tabla para poder ver y modificar los datos.

Valoración personal y auto - evaluación

Desde mi punto de vista creo que he hecho un buen trabajo, cumpliendo el desarrollo de la aplicación dentro del plazo solicitado y sin parar de auto exigirme más y más para lograr sacarlo adelante.

Quiero mencionar que al igual que hay puntos que no he realizado, como el sistema de notificaciones o añadir algún gráfico informativo, he realizado otras funciones extra que no puse en la propuesta como las dos que más me ha costado realizar.

El sistema de exportación e importación fue un pensamiento que quise llevar a cabo al recordar lo sencillo que fué en Android Studio, y se me complicó muchísimo. Y el cambio de idioma fué una idea que me dió un amigo y me pareció buena idea, y fue otra de las cosas que más problemas me dió y más me costó encontrar información sobre ello.

Me gustaría también mencionar dos posibles implementaciones que se pueden hacer a futuro. Una la he comentado antes, se trata de las notificaciones, y otra es añadir seguridad a la base de datos SQLite. He estado investigando y se suele utilizar la herramienta SQLCipher.

Consejos

Si tuviera que dar algún consejo para alguien que está pasando por esto, le diría que antes de empezar organice bien las ideas en su cabeza y que cuando las esté llevando a cabo, no se frustre si le salen errores desconocidos. Debe ser persistente y, si le dedicas el tiempo suficiente conseguirás solucionarlo y seguir adelante. Parece que no, pero al final se llega donde uno quiere.

WEBGRAFÍA

Curso .NET MAUI:

https://www.youtube.com/watch?v=Hh279ES_FNQ&list=PLdo4fOcmZ0oUBAdL2NwBpDs32zwGqb9DY

Documentación .NET MAUI:

https://learn.microsoft.com/es-es/dotnet/maui/?view=net-maui-9.0&WT.mc_id=dotnet-35129-website

SQLite en .NET

<https://learn.microsoft.com/es-es/dotnet/maui/data-cloud/database-sqlite?view=net-maui-9.0>

Localization (Cambiar idioma):

<https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/localization?view=net-maui-9.0>

<https://www.youtube.com/watch?v=cf4sXULR7os>

Calendario:

<https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/datepicker?view=net-maui-9.0>

<https://www.syncfusion.com/maui-controls/maui-calendar>

Generación de pdf:

<https://help.syncfusion.com/document-processing/pdf/pdf-library/net/create-pdf-file-in-maui>

<https://github.com/jfversluis/MauiGeneratePdfSample>

Algunas de las páginas que he visitado para hacer la tabla comparativa entre las apps que cumplen la misma función que la mía.

<https://time.com/5621109/best-habit-tracking-apps/>

<https://zapier.com/blog/best-habit-tracker-app/>

<https://aswa.es/que-tipo-de-tecnologias-de-asistencia-para-la-comunicacion-pueden-ayudar-a-las-personas-con-discapacidades-visuales-a-acceder-a-la-informacion-escrita/>

https://clockify.me/blog/productivity/best-habit-tracker-apps/?utm_source=chatgpt.com