DAYPILOT

TRABAJO DE CURSO

Mario Pérez Carmona

INDICE

Introducción

DayPilot es un proyecto centrado en el desarrollo personal, utilizado para que poco a poco el usuario pueda desarrollar una rutina la cual utilice día a día, esto se desarrolla con pequeñas metas creada por competencia sana con amigos, la creación y seguimiento de tareas, así como la gestión de hábitos del usuario.

Aunque existen algunas aplicaciones que ya se encargan de estos apartados, el punto que las diferencias con esta es la implementación de todos los apartados en uno, buscando la mayor facilidad a la hora de empezar a desarrollar hábitos sanos.

Este documento hablará de manera sencilla y escalonada todos los puntos a destacar de esta aplicación a un nivel interno, como se desarrolla y él porque del todo.

Objetivos

Si hablamos de los objetivos del proyecto tenemos que dividirlos en objetivos generales con la aplicación a un corto alcance y objetivos específicos unificado a un mayor alcance, así como el alcance del proyecto.

Objetivos generales

Como objetivos generales tendríamos

- La implementación de un sistema de tareas, donde se puedan crear, editar y eliminar (CRUD)
- Un sistema de amigos
- Un contador de pasos del usuario
- Un sistema que permita ayudar al usuario a reducir su consumo de móvil
- Un sistema que motiva al usuario a mantener su rutina

Objetivos específicos

Con los objetivos generales, podemos utilizarlos para distintas cosas, como crear respuestas que respondan a lo que piden, creando nuevas funcionalidades o secciones que las complementen.

- Un calendario que permita traquear y visualizar todas las tareas creadas
- Un sistema de puntos que permita crear una competencia entre usuarios
- Una sección de hábitos que encapsule los pasos, la salud tecnológica y un sistema de recordatorios

Esto permite crear una aplicación que pueda ser de interés a cualquier usuario ya que está orientado a ser una aplicación para todo el mundo

Diseño

DayPilot se ha planteado con el objetivo de que el usuario pueda **entender la aplicación rápidamente** y realizar las acciones principales sin esfuerzo. Para ello, se han seguido tres principios base: **usabilidad**, **simplicidad** y **consistencia**.

Principios de diseño y estructura de pantallas

Usabilidad: La interfaz prioriza que las funciones más importantes estén siempre accesibles y sean fáciles de descubrir. La navegación entre secciones es gobernada por una barra inferior (NavigationBar) que permite cambiar de sección de forma rápida y predecible. Además, se utilizan elementos visuales conocidos como iconos o tarjetas para mayor sencillez de navegación.

Simplicidad: Cada vista intenta centrarse en una tarea concreta: ver información, editar una tarea, revisar perfil, etc, evitando así rellenar la interfaz con información, esto se consigue agrupando el contenido y mostrando solo lo necesario en cada contexto.

Consistencia: Se intenta mantener un estilo coherente en toda la aplicación, mismos patrones de navegación, disposición similar de elementos, y comportamiento uniforme de componentes. Además, el uso de un mismo diseño de interfaz ayuda a que el usuario reconozca rápidamente lo que tiene que hacer en cada sección de la aplicación y se sienta orientado, sin explicaciones o aprendizaje necesario.

Todo esto se consigue separando a el sistema en 3 secciones divididas por lo visual, que separan importancia de sección

Zona principal

La zona principal corresponde a los accesos más destacados de la aplicación, representados mediante los botones de mayor tamaño situados en la parte superior de la pantalla. Esta área agrupa las funcionalidades de uso más frecuente y de consulta rápida, como el calendario y la sección de hábitos, donde se integran elementos como el contador de pasos, la salud tecnológica y los recordatorios.

Zona secundaria

La zona secundaria se sitúa justo debajo de la zona principal y está formada por accesos de menor tamaño. Su objetivo es ofrecer entrada a funciones que requieren interacción más puntual o acciones concretas, como la gestión/creación de tareas y el módulo de rivalidades con otros usuarios.

Zona terciaria

La zona terciaria está representada por la barra de navegación inferior (NavigationBar), que actúa como eje estable de movimiento dentro de la aplicación. Desde esta barra se accede a tres secciones principales:

- Inicio, que devuelve al usuario a la pantalla principal.
- Amigos, donde se pueden buscar usuarios, enviar y gestionar solicitudes de amistad.

- Perfil, que muestra la información del usuario, su lista de amigos, el progreso de estos y un apartado de configuración.

Elementos UI principales

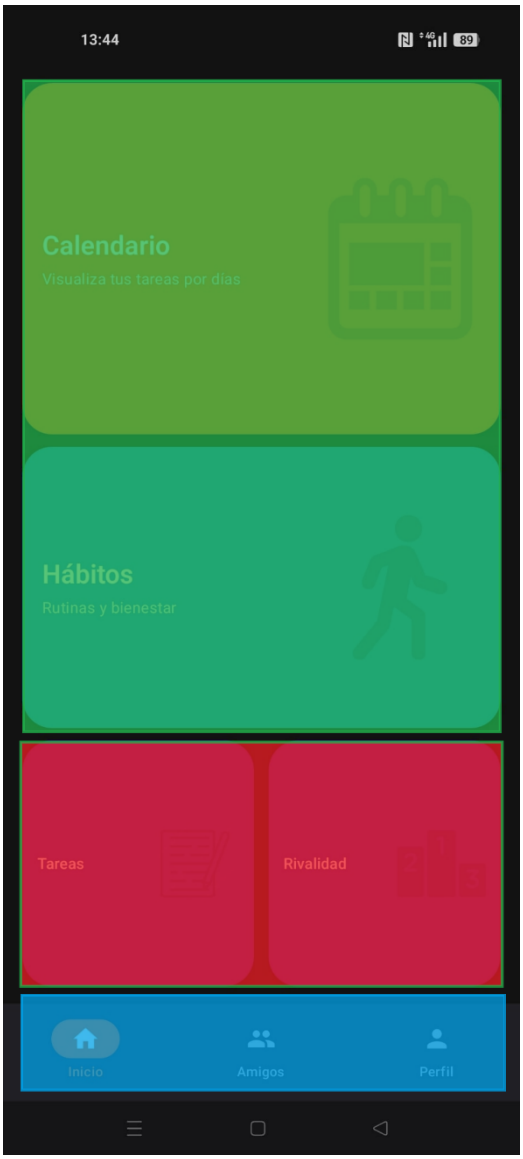
Por último, en el diseño queda explicar la creación formularios, listas y otros apartados los cuales se ha intentado mantener una sincronía con todos ellos para mayor claridad visual. Algunos ejemplos simples serían:

- Formularios: Utilizando **MODALBOTTOMSHEET** de Jetpack Compose para crear un formulario que se sitiera algo más dinámico y visual
- Tarjetas: Utilizado por toda la aplicación usando en cosas como la Salud tecnológica, las tareas, el calendario para mostrar las tareas, en los recordatorios, etc. Para todos estos se utiliza el **CARD** para mostrar cualquier cosa

Verde: Zona Principal

Rojo: Zona Secundaria

Azul: Zona Terciaria



Funcionalidades

Antes de ver la arquitectura es importante entender el conjunto de funcionalidades que ofrece la aplicación actualmente. Muchas de estás ya se han nombrado, pero se explicarán brevemente aquí de nuevo en descripciones breves.

Principales funcionalidades:

Tareas: Se permite al usuario realizar varias actividades con las tareas permitiendo

- **Gestión de tareas:** crear, editar y eliminar tareas con dificultad, duración estimada, categoría y días asignados.
- **Listado visual de tareas:** un listado con tarjetas con estado (pendiente / completada) y que permita realizar las acciones de eliminar o editar.
- **Completar tareas y progreso:** se permite al usuario marcar tareas como completadas y registrar el avance del usuario.
- **Filtros y ordenación:** se permite filtrar por dificultad y luego ordenar por estado o criterios como fecha, nombre, duración o próxima fecha.
- **Calendario:** Un calendario donde se puede ver de manera organizada todas las tareas a lo largo del mes, este permite ver las definiciones de esa tarea, si esta completada o si se quiere añadir una tarea en el momento

Puntos: El usuario podrá ganar puntos a lo largo de la aplicación mediante tareas o pasos

- **Tareas:** Se obtendrán 2 puntos por cada tarea completada, sin importar la dificultad o el tiempo invertido, esto incentiva a la creación de varias tareas lo cual crea un hábito de organización.
- **Pasos:** Se pueden obtener hasta 6 puntos, los cuales se consiguen llegando a las metas de 50%, 75% y 100% de la meta.
- **Mantención de puntos:** Los puntos se mantienen durante 30 días exactos, esto se hace para que alguien que entre después no esté en una desventaja permanente respecto a alguien que empezó después además permite llevar un seguimiento extra en el que el usuario puede ver que consiguió menos puntos este mes respecto al anterior

Perfil de usuario: Pantalla que muestra toda la información del usuario necesaria, como el nombre, correo, etc. Así mismo también tiene acceso a una pestaña que permite ver el progreso en puntos realizado en los últimos 30 días, y de donde viene los puntos. Por último, hay unas secciones extras como ver a tus amigos y su progreso con la misma tabla o la sección de configuración.

Configuración: Sección donde se puede configurar varios apartados del perfil como, si se desea un modo oscuro o no, la activación de notificaciones, el idioma y la región actual

Amistades: El usuario podrá tener amigos dentro de la aplicación, para ello

- **Sección de invitación y aceptación de solicitudes:** Una sección donde se pueden enviar peticiones de amistad a un usuario que luego otro usuario podrá usar la misma sección para aceptarla
- **Rivalidad:** Una de las zonas principales donde se puede ver todos los amigos que tienes y verte en un ranking con ellos para ver quien tiene más puntos en los últimos 30 días
- **Amigos en el perfil:** Aparte de poder cuantos amigos tiene un usuario en el perfil, también se puede ver el progreso de puntos y donde lo ha conseguido

Hábitos: El usuario podrá entrar a la sección de hábitos donde podrá ver bastantes subsecciones tales como

- **Pasos:** La aplicación traqueara los pasos dados por los usuarios permitiendo llegar a una meta, por ejemplo 2000 pasos que es el por defecto al entrar. Al caminar las barreras del 50%, 75% y 100% de la meta se obtendrán puntos, 1 por 50%, 2 por 75% y 3 por el 100%. Además, los pasos traqueados se guardan en la base de datos Firebase solo al llegar a una meta, de mientras se guarda en local. Otro detalle es que la meta es configurable pudiendo poner el valor deseado entre 10 y 1 millón. Por ultimo la meta solo es configurable cuando para el día siguiente, así evitamos trampas fáciles
- **Recordatorios:** Una sección donde se pueden configurar alarmas para ciertos días específicos y a una hora especifica. Esto es configurable para que la alarma salte todos los días o que añada un recordatorio 10 minutos antes.
- **Salud Tecnológica:** Esta última sección se encarga de poder añadir restricciones de uso de aplicaciones sobre la aplicación, cuando dicha cuota se cumple se empieza a mandar notificaciones avisando del límite de cuota cada cierto intervalo. El intervalo y la cuota de tiempo son configurables. Además, se pueden crear grupos que contiene varias aplicaciones y se puede definir una cuota y un intervalo separado para ellas

Persistencia y sincronización: Se utiliza un combinado de Firebase con los datos principales con datos del perfil manteniéndose sobre 30 días, así como los amigos, y el almacenamiento local, utilizado para el guardado de sesión y las preferencias.

Automatización de despliegue: Como ultima característica se añade el uso de pipeline con GitHub Actions para construir APK y publicar releases de la versión final del producto.

Arquitectura

La arquitectura se ha planteado buscando un equilibrio entre mantenibilidad y simplicidad de implementación. El proyecto separa responsabilidades en capas y módulos lógicos, de forma que cada parte tenga un objetivo concreto, dividiéndose en 3 capas

- La UI que se centra en mostrar estado y recoger interacción.
- La lógica que se encarga de transformar y coordinar datos.
- La capa de datos que abstrae el acceso a Firebase y a la persistencia local.

Estructura del proyecto y reparto de responsabilidades

El proyecto se organiza en paquetes

- **Capa de presentación (main/... | login/…)**: contiene las pantallas de la aplicación (UI), divididas por secciones funcionales. Aquí se ubican las Activity y las Screen en Jetpack Compose.
- **Capa de datos remotos (firebaseLogic/...)**: agrupa repositorios orientados a cada dominio, por ejemplo, AuthRepository, TaskRepository, PointsRepository. Su objetivo es **ocultar detalles** de Firebase al resto del proyecto.
- **Persistencia local / utilidades (mainDatabase/...)**: contiene componentes que trabajan con el dispositivo: preferencias, sesión, almacenamiento local de estados, y futuras extensiones como notificaciones del sistema.
- **Modelos compartidos**: estructuras de datos simples, por ejemplo, Task, UserProfile, etc. usadas para transportar información entre capas.

Este reparto permite que el código sea más fácil de extender: por ejemplo, añadir una funcionalidad nueva suele implicar crear o ampliar un repositorio y conectarlo a una screen, sin necesidad de "mezclarlo todo" en la UI.

Presentación: Activities + Screens (UI en Compose)

Se utiliza un patrón sencillo y común en Compose: **una Activity por sección principal** (o flujo), y dentro de ella una **Screen** que contiene la UI real.

Rol de las Activities

Las Activity actúan como "contenedor" y punto de integración con Android:

- **Gestionan el ciclo de vida** (onCreate, navegación hacia atrás, recepción de intents).
- **Obtienen dependencias necesarias** (por ejemplo, SessionManager, repositorios Firebase, etc.).
- **Verifican estado de sesión** (si no hay usuario, se cierra la pantalla o se redirige).
- **Definen el tema** (DayPilotTheme) y **detalles del sistema** (status bar / navigation bar).

- **Pasan parámetros a las Screen** (por ejemplo, uid, openTaskId, callbacks como onBack).

Esto mantiene el código de UI más limpio: la Activity "conecta" con el sistema y la Screen se dedica a dibujar y responder a eventos.

Rol de las Screens (Compose)

Las Screen (por ejemplo, TaskScreen) representan la interfaz y su comportamiento:

- **Mantienen estado con remember y mutableStateOf** (listas de tareas, filtros, loading, errores).
- **Llaman a repositorios mediante corrutinas** (rememberCoroutineScope, LaunchedEffect).
- **Renderizan UI reactiva** (si cambia el estado, la UI se actualiza automáticamente).
- **Contienen componentes reutilizables**: cards, chips, filtros, formularios, etc.

Componentes UI reutilizables

Dentro de cada Screen se utilizan piezas separadas para mantener orden, por ejemplo en task:

- TaskRow para cada tarea en lista.
- TaskFiltersRow para filtros y ordenación.
- TaskFormSheet como formulario de alta/edición.
- Diálogos de confirmación (AlertDialog) para acciones importantes (ej. completar tarea).
- Estructuras de layout (Scaffold, TopAppBar, LazyColumn) para seguir patrones consistentes.

Este enfoque es útil porque Compose facilita dividir pantallas complejas en componentes pequeños, reutilizables y testeables a nivel de UI.

Lógica de aplicación: repositorios y servicios internos

La lógica de la aplicación se apoya en repositorios por dominio, lo que evita que la UI trabaje directamente con Firebase o con detalles de bajo nivel, esto se encapsulan en firebaseLogic

Repositorios como punto central

Los repositorios encapsulan:

- Acceso a datos (lectura/escritura).
- Conversión de datos.
- Manejo de valores por defecto y compatibilidad con campos opcionales.

Esto se ve por ejemplo en:

- **AuthRepository**: registro/login, gestión de perfil, foto, búsqueda de usuarios, sistema de amigos y solicitudes.
- **TaskRepository**: CRUD de tareas, lectura de Firestore, completar tarea.
- **PointsRepository**: sumar puntos según fuente y registrar metadatos.

Esto diseño tiene muchas ventajas tales como que la UI no necesita saber si un dato viene de Firestore, Storage o caché local lo cual aísla partes del código manteniendo el desacoplamiento del código o la posibilidad de cambiar de backend y solo necesitar tocar los repositorios y nada de la UI

Servicios internos y utilidades

Además del repositorio, existen "servicios" internos para lógica puntual, como:

- Formateo de fechas y duraciones.
- Cálculo de fecha más próxima de una tarea
- Mapeo de iconos por categoría.
- Gestión de sesión/preferencias (local).

Datos remotos: Firebase (Auth, Firestore, Storage)

Se utilizan tres piezas principales de Firebase:

Firebase Authentication

- Registro y login con email/contraseña.
- Identificación del usuario actual (currentUser).
- Control de sesión: si no hay usuario logueado se bloquean pantallas protegidas.

Cloud Firestore

Se usa como base de datos NoSQL:

- Colección *users* para perfiles para guardar nombre, email, username, región, etc.
- Subcolección *tasks* para tareas por usuario o *friends* y *friendRequests* para la parte social.

Firestore aporta:

- Estructura flexible.
- Lectura simple y directa.
- Modelo orientado a documentos, que encaja con entidades como Task o UserProfile.

Firebase Storage

Se usa para guardar específicamente las fotos de perfil del usuario.

La lógica sube imágenes (desde Uri o Bitmap), obtiene la URL pública y luego actualiza Firestore con el campo photoUrl.

Persistencia local: sesión, preferencias y caché

Como ya se ha explicado se utiliza la base de datos local para apartados como:

- Guardar preferencias de usuario.
- Mantener estado de sesión o configuración sin depender de red.
- Permitir una UX más rápida y estable.

Conclusiones

Logros y resultados obtenidos

Tras todo este desarrollo se puede sacar que la aplicación resultante, pese a ser sencilla quede con un resultado eficaz y eficiente disponiendo de varios apartados claves para una aplicación como él la gestión de sesión, y amigos, guardados de amigos y todas sus funcionalidades principales. Todo esto deja la aplicación preparada para ser escalable en un futuro.

Funcionalidades no implementadas

Desgraciadamente ha habido algunos apartados de la aplicación que no han podido ser implementadas debido a problemas de tiempo, aquí se muestran de más leve a más importante:

- **Sistema de control de notificaciones y idiomas:** En el apartado de configuraciones pese estar ambas cosas como opciones, no están implementadas.
- **Grafica que muestre el progreso respecto a los 30 días:** Este apartado pretendía ser un apartado extra del menú donde se podía ver los pasos, tareas y puntos conseguidos cada día, la propia base de datos ya está preparada para ello, y el único motivo por el que no esta implementado son problemas con las librerías de las graficas
- **Notificaciones de las tareas:** Pese a tener un apartado de notificaciones las tareas para activarlas y desactivarlas estas no notifican, lo ideal sería es que al inicio de cada día se mandará una notificación explicada las tareas del día, pero como está ahora no se realiza nada

Propuestas de mejora y evolución futura

Aquí hay unas mejoras extra que se podría haber implementado con un poco más de tiempo o para una futura expansión del proyecto

- **Más opciones de gestión** de hábitos como rutinas, manejos de entrenamientos, conexión con un reloj de entrenamiento (Ej: SmartWatch)
- **Mejor gestión de las tareas**, mostrando más tipos de tareas, tareas divididas en secciones para enfatizar en pequeños incrementos, etc.
- **Calendario más expresivo** mostrando rutas recomendadas para solución de tarea, por ejemplo, creando una hoja de ruta sobre las tareas de la próxima semana
- A parte de la ganada de puntos, también se pueda ganar una moneda que permita ganar trajes y mejorar un perfil virtual, fomentado pequeñas recompensas en mejoras de hábitos

Anexo

Aquí se deja el github usado para el desarrollo de este proyecto

https://github.com/MarTonPerCar/DayPilot/tree/master