Grace

Intro

En este breve documento se hablará resumidamente del proceso de implementar la aplicación Grace. Como preámbulo, iniciamos con la idea de llamar a la aplicación "XS" porque es un clon de X, antes conocido como Twitter. Sin embargo, necesitábamos algo que tenga un significado y, en la mitad del proyecto, decidí hacer el cambio a "Grace". La bauticé "Grace" en honor a Grace Hopper, la persona que acuñó el término "Bug". Lo que más hice para desarrollar esta aplicación fue mucho debugging…

Frontend

Grace es un proyecto almacenado en un repositorio de GitHub, que tiene 60+ commits registrados. Del 7 al 15 de noviembre, me encargué de realizar el UI de la app.

Normalmente, al crear aplicaciones en Xcode, tenemos dos opciones: crear la aplicación UIKit, una herramienta desarrollada por Apple para desarrollar interfaces, o la nueva opción SwiftUI, un framework para interfaces desarrollado por Apple en 2019. SwiftUI es el que se utiliza para las aplicaciones modernas hechas para dispositivos de Apple. En SwiftUI, una de las particularidades es que no tenemos que preocuparnos acerca de Auto Layout, el cual es un dolor de cabeza para muchos desarrolladores de iOS ya que hay que adecuar la aplicación a todos los tamaños de pantalla de los modelos de iPhone que existen, sin mencionar para iPad, Watch, etc.

SwiftUI utiliza una sintaxis declarativa, lo cual hace que el desarrollo de interfaces sea un proceso bastante rápido. Además, contiene Live Preview, lo cual permite ver en tiempo real a través del entorno Xcode los cambios en la interfaz, lo cual es una ventaja. Cuando se completó el UI de la aplicación, pude inyectar instantáneamente el código de Google Firebase para que nuestra aplicación tenga la funcionalidad deseada.

Otra ventaja de utilizar SwiftUI es que es cross-platform, lo que significa que Grace no solo funciona para iPhone, sino que también funciona para iPad y MacOS. Siendo una aplicación nativa, tiene una velocidad excelente en comparación con otras herramientas como Flutter o React Native, que se utilizan para desarrollar aplicaciones para Android e iOS simultáneamente.

Backend

En el momento en que supe que el UI para la aplicación estaba hecho, tuve que registrar la aplicación en la consola de Firebase e instalar un archivo que debía ser arrastrado hacia la aplicación. Procedí a hacer la integración de Firebase a través del Swift Package Manager en Xcode y estuve listo para poder investigar cómo ingresar el código que ya está definido en la documentación de Firebase para que se integre a la app.

Para esta aplicación, tuve que habilitar tres servicios de Firebase: Autenticación, Firestore (una base de datos NoSQL de Firebase) y Storage (para guardar las fotos de perfil de los usuarios). En Firestore y en Storage, podemos hacer ciertos cambios como redefinir las reglas de cómo se van a almacenar los datos que vamos a guardar dentro de Firebase.

Para Autenticación, existen diferentes métodos de Sign-In, como con Google, GitHub, Apple, Microsoft, Teléfono, etc. Para Grace, solo decidí utilizar el método con correo.

En Firestore, donde guardamos todos los datos de la aplicación, tuve que declarar seis carpetas o colecciones de datos: "followers" para poder almacenar los seguidores de una cuenta, "following" para poder rastrear las cuentas que el usuario actual está siguiendo, "messages" que se encarga de guardar los mensajes que se envían entre los usuarios, "posts" que se encarga de almacenar el contenido de las publicaciones que realizan los usuarios y, finalmente, "users" que almacena la lista de usuarios que están registrados en la aplicación con todos los datos que posee un usuario al registrarse en Grace.

En Storage, el uso que le dimos es el de almacenar las fotos de perfil de los usuarios. Se puede ver el tamaño que tiene y un URL que, si le damos click, descargamos las fotos en nuestra computadora. Las fotos se ven que están almacenadas en Firebase, pero el único inconveniente que hallé es no poder realizar un fetch adecuado de las fotos. Investigué librerías como KingFisher y URLImage, las cuales integré al proyecto para que podamos mostrarlas dentro de la aplicación, pero no tuve éxito al hacerlo. Así que decidí mostrar una foto de un perfil vacío. Sin más que decir, se pudieron almacenar las fotos de los usuarios en Firebase.

MVVM

Al ser una aplicación que utiliza la arquitectura MVVM, es necesario definir qué va a ir en cada carpeta. En View, estarán los archivos que se utilizaron para definir la interfaz de la aplicación. En Model, se colocaron los datos que necesitábamos para hacer un

fetch de Firebase. Me refiero a que, por ejemplo, para User, necesitamos guardar y sacar los siguientes datos: (id, username, URL de la imagen del usuario, nombre completo, email, si es el usuario actual y las estadísticas). Aquellos son los datos que debemos ingresar y sacar en Firebase. En ViewModel están los archivos que contienen código ya escrito por ingenieros de Google para poder hacer uso del API e inyectar directamente ese código que ya escribieron a nuestra aplicación, funciones como registrar un usuario, cerrar sesión, etc.

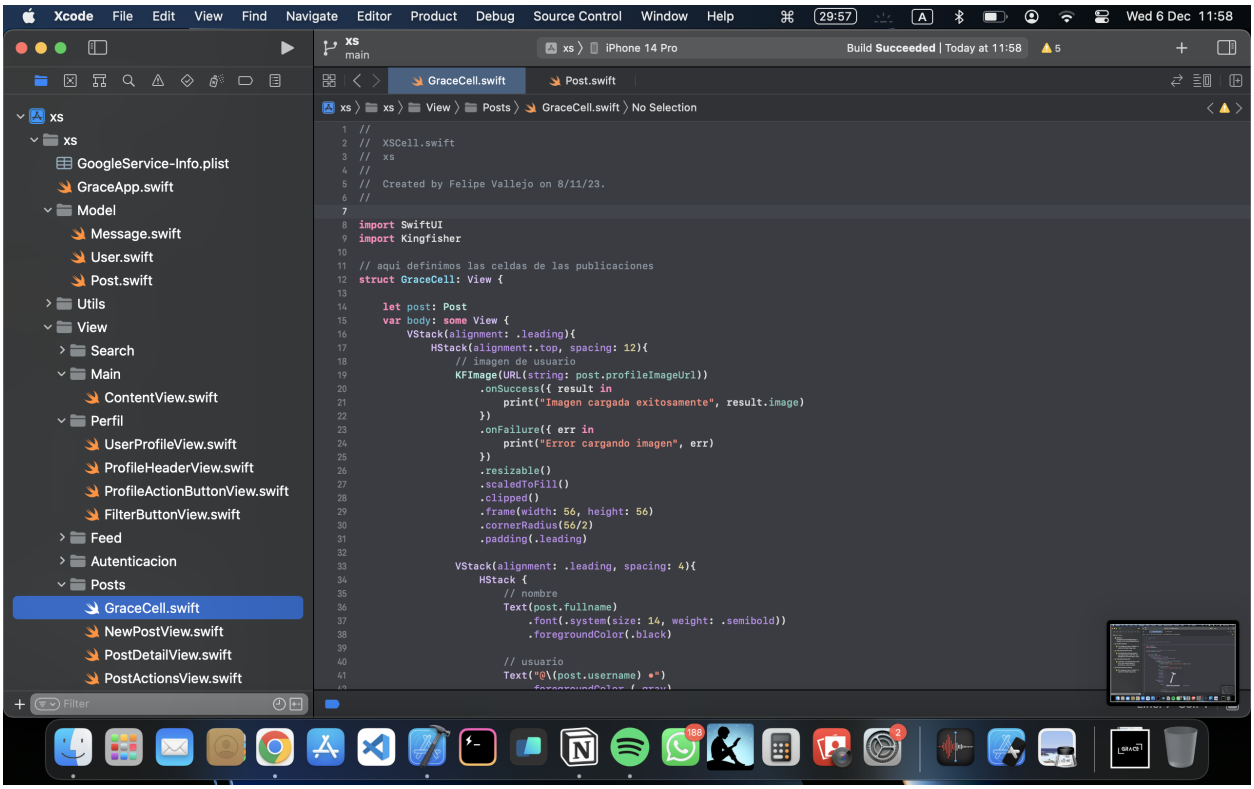
Conclusion

En el ámbito del frontend, la adopción de SwiftUI demostró ser una elección acertada, brindando no solo eficiencia en el desarrollo de interfaces, sino también una versatilidad cross-platform que permite a Grace operar de manera fluida en distintos dispositivos de la familia Apple. La eliminación de preocupaciones relacionadas con Auto Layout, característica de SwiftUI, simplificó significativamente el proceso de diseño y optimizó la velocidad de desarrollo. En el backend, la integración de Firebase se llevó a cabo con éxito, aprovechando servicios clave como Autenticación, Firestore y Storage.

La implementación del patrón de arquitectura MVVM proporcionó una organización clara y modular del código. La asignación de responsabilidades a las carpetas correspondientes (View, Model y ViewModel) facilitó la comprensión y mantenimiento del código, permitiendo una integración fluida con las funciones predefinidas de Firebase.

Aunque la implementación final mostró una foto de perfil predeterminada en lugar de las imágenes reales, este obstáculo no impidió el almacenamiento exitoso de las fotos de los usuarios en Firebase.

En resumen, el desarrollo de la aplicación Grace ha sido un proceso integral que ha abordado tanto los aspectos técnicos como los conceptuales. Desde la elección de tecnologías hasta la gestión eficiente de datos y la implementación de la arquitectura MVVM, cada fase ha contribuido a la creación de una aplicación funcional y significativa. **Este es el MVP para la asignatura.**



Autor

Felipe Vallejo