UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

LUIS ENRIQUE GARCÍA ROSALES

CÓMPUTO MÓVIL

¡OS DEVELOPMENT LAB

GeoGoals

DICIEMBRE DE 2019

Introducción

GeoGoals es una aplicación de exploración y turismo para iOS. Permite que el usuario descubra lugares qué visitar basado en categorías llamadas listas, que agrupan cierto número de sitios, llamados objetivos. Algunas categorías son, por ejemplo, iglesias, parques, rascacielos, etc. Los objetivos de cada lista son del tipo que su nombre indica.

El usuario tiene la posibilidad de completar objetivos haciendo check-in cuando se encuentra en el lugar indicado. Esto marcará el objetivo como completado. Una vez completados todos los objetivos de una lista, la lista aparecerá como completada. De esta manera se puede llevar un registro de los lugares visitados.

En esta primera etapa de desarrollo, la aplicación permite esas funcionalidades, con un número pequeño de objetivos y listas para hacer pruebas. Sin embargo, es fácilmente extensible en sus datos y nuevas características pueden ser agregadas para mejorar la experiencia del usuario.

En este documento se describirá brevemente el proceso de desarrollo de esta primera prueba de la aplicación GeoGoals.

Desarrollo

El entorno de desarrollo fue Xcode, utilizando la opción gráfica para las vistas y la parte de programación en Swift.

Se inició con la creación del esquema de la aplicación, que consiste en cinco vistas (View Controller) conectadas por medio de segues . Algunos segues se hicieron en la interfaz visual, mientras que otros se realizaron programáticamente, dependiendo de las necesidades de la transición de una vista a otra, pues en algunos casos la vista siguiente cambia de acuerdo a la selección que haga el usuario.

Para controlar la navegación entre vistas se utilizó Navigation Controller. De esta manera se puede avanzar y retroceder entre las vistas con facilidad, y no se generan vistas de más en memoria. Cada vista tiene su encabezado con un título y un botón para retroceder a la vista anterior.

La primera vista es la pantalla de inicio de la aplicación. Esta consiste de una UIImageView como fondo, otra UIImageView como logo de la aplicación y un solo UIButton para proceder a las listas.

La segunda vista, ListsViewController, muestra en una UITableView el nombre de las listas disponibles. La UITableViewCell que forma parte de la tabla fue personalizada para que mostrara una imagen UIImageView descriptiva de la lista y texto UILabel en cierto tamaño y forma, así como un espacio para un checkmark que se muestra en caso de que la lista haya sido completada, por lo que se tuvo que hacer una clase ListTableViewCell para determinar el comportamiento de la celda personalizada.

La tercera vista, GoalsViewController, es en diseño idéntica a la vista anterior, pues muestra una tabla con celdas personalizadas, cada una contiene una imagen del objetivo, el nombre y el espacio para el checkmark. Para controlar esta celda se utiliza la misma clase que en la vista anterior.

La cuarta vista, DetailsViewController, muestra en la parte superior la imagen del objetivo y un UIImageView que muestra una equis roja o una marca verde para indicar si se ha cumplido el objetivo o no. También hay un UIButton que lleva a la siguiente vista, mostrando una imagen de un mapa en lugar de texto. En la parte inferior se encuentra un UILabel mostrando el nombre del objetivo, debajo otro UILabel que muestra la descripción y más abajo, en el fondo de la pantalla, hay un UIButton que dice "Estoy aquí" para hacer check-in.

En la quinta y última vista, MapViewController, la pantalla completa muestra un mapa MKMapView y en él un marcador que indica la ubicación del objetivo, junto con su nombre.

La transición desde las vistas con tablas se hace al presionar una fila de la tabla, con lo que se ejecuta un segue y se envían los datos necesarios para que la siguiente vista muestre la información correcta.

Para que todos los objetos que se muestran en pantalla aparecieran en las posiciones correctas, sin ningún traslape entre ellos y en proporciones constantes se utilizaron constraints en todas las vistas.

Los datos que muestra la aplicación se obtienen de dos archivos JSON. El primer archivo, llamado lists.json, contiene el conjunto de listas, que se conforman por los datos de id, name e image. Estos datos se cargan a un array de tipo list, que es una estructura Codable. El segundo archivo, llamado goals.json, contiene todos los objetivos, cada uno con los datos de id, listId, name, image, description, latitude y longitude. Estos se cargan en un array de tipo goal, igualmente una estructura Codable.

Cuando un usuario hace check-in, la aplicación verifica que en realidad el usuario se encuentre en la ubicación del objetivo. Para esto es necesario obtener la ubicación por GPS del usuario. Es por eso que se utiliza MapKit para permitir trabajar con ella. En la clase location se incluyó esta funcionalidad, que se encarga de pedir los permisos al usuario para acceder a su ubicación, y también de devolver la ubicación en el momento del check-in para hacer la comparación con la ubicación proporcionada en el archivo JSON del objetivo.

Una vez que el usuario hace check-in exitoso, el estado de un objetivo cambia de no completado a completado, por lo que la marca verde aparece en la vista de detalle y se desactiva el botón "Estoy aquí" para ese objetivo. La celda en la tabla de objetivos mostrará un checkmark, y si se completaron todos los objetivos de una lista, la celda en la tabla de listas deberá mostrar un checkmark también. Esta información se almacena en un array de estructuras del tipo state, que incluye un booleano que almacena el estado de la lista, y un array de booleanos que almacena el estado de los objetivos para esa lista.

Como debe existir persistencia de datos, es decir, que tras descargar la aplicación de memoria y volverla a cargar, la aplicación debe mostrar los estados correctos, sea no completado o completado, la estructura state es Codable y se guarda en un archivo plist, y los datos se recuperan cuando se carga la primera vista. La primera vista también se encarga de revisar que ya exista un archivo de guardado, si no crea uno nuevo con todos los estados de completado en false.

Conclusión

Este proyecto llevó a la creación de una pequeña app, con funciones básicas pero funcional, que sirvió para mostrar el potencial en el desarrollo para iOS, Swift y en la aplicación misma. Hubo algunos problemas durante la programación que fueron difíciles de solventar, pero se encontró una manera de darle la vuelta a esos problemas. El conocimiento adquirido es invaluable, y sería muy bueno continuar con el desarrollo de GeoGoals para ampliar este conocimiento y lograr una aplicación íntegra con posibilidades de hacerla disponible al público.

Bibliografía

App Development with Swift
Everyone Can Code
Apple Education