Entrega Desarrollo de Aplicaciones iOS

Autor: Jorge Sanzo Hernando

Descripción funcional:

La aplicación desarrollada para esta entrega consiste en la creación de un gestor de recetas, el usuario podrá registrarse en el sistema, iniciar sesión, consultar, eliminar recetas y crear nuevas recetas, también podrá cambiar distintos ajustes de la aplicación.

Primero vemos la pantalla de inicio de sesión. Para registrarse en el sistema hay que rellenar los campos requeridos y presionar el botón de registrarse, si no se produce ningún error la aplicación llevará al usuario a la vista principal y su nombre de usuario y contraseña quedarán guardados en el servidor para poder iniciar sesión con ellos en el futuro.

Una vez registrados podremos iniciar sesión en cualquier otro momento presionando el botón de iniciar sesión tras haber introducido nuestro nombre de usuario y contraseña previamente registrados, si todo ha salido bien el móvil la aplicación llevará al usuario a la vista principal.

En la vista principal el usuario podrá consultar su lista de recetas, al principio aparecerá vacía si el usuario es nuevo, pero si ya hemos creado alguna receta previamente, aparecerá aquí con las opciones de ver más detalles sobre ella o de eliminarla. Para cada receta podemos ver a primera vista su nombre, categoría, duración y dificultad, si accedemos a los detalles se mostrarán también la subcategoría, los ingredientes, la descripción y las calorías

En la pantalla de creación de receta el usuario debe rellenar todos los campos con la información necesaria y tendrá dos botones uno para borrar todos los campos y volver a empezar en caso de que haya puesto alguna información equivocada y otro para crear su nueva receta, una vez creada quedará guardada en el sistema para poder consultarla en el futuro

Por último, tenemos la pantalla de cerrar sesión donde el usuario podrá finalizar su sesión y volver a la pantalla de inicio de sesión de nuevo cuando haya terminado de usar la aplicación.

Descripción técnica:

Se ha utilizado el patrón VIPER (View, Interactor, Presenter, Entity y Router), también se ha utilizado el patrón Repository para implementar el código de acceso a datos externos (pueden ser locales o externos, en este caso son externos del servidor) y Clean Architecture para dividir la aplicación en diferentes carpetas o módulos:

- **Application:** Contiene la configuración general de la aplicación y los assets. En el fichero Application.swift es donde se inicializa toda la arquitectura y se montan las dependencias necesarias de cada clase, también se inicializan y asignan los casos de uso y los ViewControllers de cada vista.

- **Model:** Define todos los modelos de datos que se utilizan, en forma de estructuras de datos.

- **Presentation:** Contiene todas las vistas, Storyboards, ViewControllers y Presenters que permiten realizar al usuario acciones en la aplicación, gestionarlas y ofrecer una respuesta. También se define para cada vista la lista de sus estados posibles en los que se puede encontrar. Este módulo está dividido en dos:
    - **View**: Todas las vistas tienen su ViewController que es el encargado de inicializar y configurar la vista, recoger las acciones del usuario y pasárselas al Presenter
    - **Presenter**: Todas las vistas tienen su Presenter asignado que es el encargado de ejecutar los casos de uso y cambiar el estado de la vista, ya sea para mostrar datos, mostrar un error o realizar la acción que sea necesaria en cada caso.

- **Interactor:** En este módulo es donde se definen e implementan todos los casos de uso de la aplicación, es el intermediario entre los Presenter de las vistas y la capa de datos, en este caso la capa de red. Este módulo también se divide en dos:
    - **Provider**: Define los Protocols que deben implementar las clases que implementen los casos de uso, de esta manera quedan todos bien definidos con sus funciones, parámetros y tipos de datos de retorno.
    - **Implementation**: Implementa los Protocols previamente definidos para cumplir con los requisitos de los casos de uso de la aplicación.

- **Repository:** En este módulo se define e implementa el patrón repository para poder comunicar los casos de uso con la capa de red.

- **Network:** En este módulo se crea e implementa la capa de red de la aplicación. Para cada caso de uso se definen las peticiones a la API necesarias, su tipo (POST, GET o DELETE), sus parámetros incluidos en la petición y las respuestas. En este punto se utiliza la interfaz ImmutableMappable para poder convertir objetos del modelo de datos de la aplicación a formato JSON y viceversa. Para realizar las peticiones a la API se ha utilizado RxAlamofire.

Librerías utilizadas:

Todas las librerías utilizadas han sido incluidas mediante cocoapods, definiendolas en el archivo “Podfile” e instalandolas mediante el comando “pod install”

- **RxSwift y RxSwiftExt:** esta librería nos facilita el uso de la programación reactiva utilizando observadores para detectar cambios de estado de objetos, por ejemplo, en la vista de login el usuario realiza la acción de registrarse, la vista pasa a estado de “Cargando” mientras el Presenter observa el progreso del caso de uso “Registrarse”, una vez que ha terminado ese proceso el Presenter cambiara el estado de la vista de nuevo en función de si el caso de uso fue exitoso o tuvo algún error
- **MaterialComponents/Buttons:** esta librería se ha utilizado para mejorar el aspecto visual de los botones de las vistas de la aplicación.

- **SVProgressHUD:** esta librería se ha utilizado para mostrar una ruleta de cargando cuando la aplicación realiza operaciones que pueden tardar un poco y se necesita avisar al usuario para que no piense que se ha quedado bloqueada.
- **Alamofire y RxAlamofire:** estas librerias nos permiten implementar la capa de red de la aplicación y con el uso de Rx podemos quedarnos observando la petición a la API esperando una respuesta ya sea un código de estado HTTP, un error de aplicación o una respuesta con datos útiles para la aplicación.
- **ObjectMapper**: esta librería nos facilita mucho el proceso de mapear objetos del modelo de datos de la aplicación a formato JSON y viceversa.

Motivación del desarrollo:

La motivación del desarrollo de esta aplicación ha sido la de reutilizar el servidor previamente creado en la asignatura Tecnologías del Lado Servidor: Cloud Computing de este mismo master para darle un posible uso real teniendo una aplicación móvil como cliente, todas las llamadas de red de la aplicación son realizadas a ese servidor que actualmente se encuentra desplegado en Amazon Web Service.

Puntos de mejora:

El siguiente paso en el desarrollo de esta aplicación podría ser el de incluir la opción de editar una receta previamente creada, permitir cambiar los colores de la aplicación o por ejemplo añadir la posibilidad de establecer una hora o fecha en cada receta a modo recordatorio para el usuario, para poder programar cierta receta para determinada fecha y recibir una notificación el día antes, por ejemplo.