Dianelys Saldaña
Interfaz de Usuario en Dispositivos Móviles
Universidad de Alicante

MyNasa

Índice

Sobre MyNasa

MyNasa es una aplicación universal que brinda a los usuarios una experiencia mediante la cual pueden explorar el cosmos, fascinantes vistas del planeta, así como aprender sobre distintos eventos naturales. Esta aplicación aprovecha las funcionalidades disponibles en las APIs de la NASA, ofreciendo una experiencia completa para aquellos apasionados por la astronomía y deseosos de sumergirse en la belleza del espacio mientras se mantienen actualizados sobre eventos significativos.

Diseñada para adaptarse a dispositivos compactos, así como a iPads, la aplicación garantiza una interfaz responsiva y atractiva. Las principales funcionalidades de MyNasa incluyen la visualización del Astronomy Picture of the Day (APOD) junto con información detallada sobre cada imagen, proporcionando a los usuarios una fascinante visión diaria del universo. Además, los usuarios pueden explorar las impresionantes imágenes capturadas por EPIC (Earth Polychromatic Imaging Camera) y acceder a eventos astronómicos actuales mediante la integración de datos de EONET (Earth Observatory Natural Event Tracker).

Controladores

Page View Controller

Para la implementación de las páginas de introducción a la aplicación, se hizo uso de un controlador de vista en Swift que utiliza un UIPageViewController. A continuación, se proporciona una explicación detallada de cada sección del código:

- Declaración de la clase ViewController: Aquí se declara la clase ViewController, que hereda de UIViewController y adopta los protocolos UIPageViewControllerDataSource y UIPageViewControllerDelegate.

- Declaración de la variable skipButton: Esta línea de código declara una variable skipButton que representa un botón en la interfaz de usuario. El botón se conecta a la interfaz mediante una conexión IBOutlet.

- Declaración de las descripciones de las páginas: Aquí se declara una variable pageDescriptions que contiene las descripciones de cada página que se mostrará en el UIPageViewController.

- Declaración de las imágenes de las páginas: Esta línea de código declara una variable pageImages que contiene los nombres de archivo de las imágenes que se mostrarán en cada página del UIPageViewController.

- Declaración de la variable pageViewController: Aquí se declara una variable pageViewController que representa el UIPageViewController utilizado para mostrar las páginas.

- Método viewDidLoad: Este método se llama después de que la vista del controlador se haya cargado en la memoria. Aquí se configura el UIPageViewController y se añade a la vista principal.

- Creación del controlador paginado: En esta sección del código, se crea una instancia del UIPageViewController y se configura como el controlador paginado. Se establece el origen de datos y el delegado del controlador paginado.

- Creación del primer controlador de contenido: Aquí se crea el primer controlador de contenido que se mostrará en el UIPageViewController. Se obtiene el controlador de contenido llamando al método viewControllerAtIndex con un índice de 0.

- Cambio de tamaño del controlador paginado: En esta sección del código, se cambia el tamaño del UIPageViewController para que quepa el botón de abajo. Se establece el marco del controlador paginado utilizando la propiedad frame de la vista.

- Añadir el primer controlador de contenido: Aquí se añade el primer controlador de contenido al UIPageViewController. Se añade como hijo del controlador principal y se agrega su vista a la vista principal.

- Configuración del botón skipButton: En esta sección del código, se configura el botón skipButton para que aparezca en la esquina inferior derecha de la vista. Se establece la propiedad layer.zPosition para asegurarse de que el botón esté por encima de otros elementos de la interfaz de usuario.

- Agregando restricciones al botón skipButton: Aquí se agregan restricciones para mantener el botón skipButton en la esquina inferior derecha de la vista. Se utilizan las propiedades trailingAnchor y bottomAnchor para establecer las restricciones.

- Método viewControllerAtIndex: Este método se utiliza para obtener el controlador de contenido en un índice dado. Se verifica si el índice está dentro del rango válido y se crea un nuevo controlador de contenido con los datos correspondientes.

- Método pageViewController:viewControllerBefore: Este método se llama para obtener el controlador de contenido anterior al controlador actual en el UIPageViewController. Se verifica si el índice es válido y se obtiene el controlador de contenido llamando al método viewControllerAtIndex con el índice anterior.

- Método pageViewController:viewControllerAfter: Este método se llama para obtener el controlador de contenido siguiente al controlador actual en el UIPageViewController. Se verifica si el índice es válido y se obtiene el controlador de contenido llamando al método viewControllerAtIndex con el índice siguiente.

- Método presentationCount: Este método se utiliza para obtener el número total de páginas en el UIPageViewController. Devuelve el número de descripciones de página en la variable pageDescriptions.

- Método presentationIndex: Este método se utiliza para obtener el índice de la página actual en el UIPageViewController. Devuelve siempre 0, ya que el controlador de contenido inicial siempre es el primero.

- Acción del botón "Start": Este método se llama cuando se presiona el botón "Start". Crea una instancia del controlador de pestañas principal y lo presenta en pantalla.

Collection View Controller

Este controlador de vista de colección en Swift se encarga de mostrar una cuadrícula de imágenes utilizando una vista de colección. Proporciona funcionalidades como la selección de una imagen y la transición a una vista de detalle.

- Al inicio se hace la declaración de la clase CollectionViewController, que hereda de UIViewController y adopta el protocolo UICollectionViewDelegate.

- Luego, se definen las variables de instancia epicImages, estimatedWidth y cellMarginSize, que se utilizan para almacenar las imágenes, el ancho estimado de las celdas y el margen entre las celdas, respectivamente.

- En el método viewDidLoad(), se inicializa el array epicImages con los nombres de las imágenes en formato JPEG. Luego, se establece el delegado y el origen de datos de la vista de colección en sí misma (self.collectionView.delegate y self.collectionView.dataSource). A continuación, se registra la celda personalizada "ItemCell" en la colección utilizando el método register(_:forCellWithReuseIdentifier:). Por último, se llama al método setupGridView() para configurar la cuadrícula de la vista de colección.

- El método viewDidLayoutSubviews() se utiliza para llamar al método setupGridView() cada vez que la vista de colección se redimensiona.

- El método setupGridView() configura el espaciado mínimo entre las celdas de la vista de colección utilizando el objeto UICollectionViewFlowLayout.

- El método collectionView(_:didSelectItemAt:) se encarga de manejar el evento de selección de una celda de la vista de colección. En este caso, se obtiene la imagen seleccionada a partir del array epicImages y se realiza una transición a la vista de detalle utilizando el método performSegue(withIdentifier:sender:).

- El método prepare(for:sender:) se utiliza para preparar los datos necesarios para la transición a la vista de detalle. En este caso, se obtiene la imagen seleccionada a partir del array epicImages y se asigna al atributo imageName del controlador de vista de detalle (DetailViewController).

- Luego, se extiende la clase CollectionViewController para implementar los métodos del protocolo UICollectionViewDataSource. Estos métodos se encargan de proporcionar el número de elementos en la sección y la celda correspondiente para cada índice de la vista de colección. En el método collectionView(_:cellForItemAt:), se asigna la imagen correspondiente a la celda utilizando el nombre de la imagen almacenada en el array epicImages.

- Por último, se extiende la clase CollectionViewController para implementar los métodos del protocolo UICollectionViewDelegateFlowLayout. Estos métodos se utilizan para calcular el tamaño de las celdas en función del ancho estimado y el margen entre las celdas. El método calculateWidth() realiza estos cálculos y devuelve el ancho de las celdas.

Search Controller

Este controlador de vista en Swift fue utilizado para implementar una funcionalidad de búsqueda utilizando un UISearchController. A continuación, se explica cada sección del código:

- Declaración de la clase: Se declara la clase EventViewController que hereda de UITableViewController y adopta el protocolo UISearchResultsUpdating.

- Propiedades: Se declaran las propiedades privadas searchController, searchResults y eventsResponse. searchController es una instancia de UISearchController que se utiliza para manejar la funcionalidad de búsqueda. searchResults es un arreglo vacío que se utiliza para almacenar los resultados de búsqueda. eventsResponse es una variable opcional que se utiliza para almacenar la respuesta de eventos obtenida de un archivo JSON.

- Método viewDidLoad(): Este método se llama después de que la vista se haya cargado en la memoria. En este método, se configura la vista y se carga la respuesta de eventos desde un archivo JSON. También se configura el controlador de búsqueda y se agrega la barra de búsqueda a la cabecera de la tabla.

- Métodos de la tabla: Estos métodos son implementaciones de los métodos requeridos por el protocolo UITableViewDataSource y UITableViewDelegate. Se encargan de configurar la tabla y mostrar los resultados de búsqueda en las celdas de la tabla.

- Método didSelectRowAt(): Este método se llama cuando se selecciona una fila en la tabla. En este método, se obtiene el objeto correspondiente a la fila seleccionada y se abre la URL fuente en el navegador web.

- Método updateSearchResults(): Este método se llama cuando se actualizan los resultados de búsqueda. En este método, se filtran los eventos que contienen la cadena de búsqueda y se recargan los datos en la tabla de resultados de búsqueda.

Tab Bar Controllers

A través de todo el proyecto se utilizan dos Tab Bar Controllers, esto con el propósito de manejar múltiples elementos pertenecientes a un mismo grupo. Debajo se mencionan cada uno de estos:

- Tab Bar Controller Principal: Este Tab Bar Controller gestiona las vistas principales de la aplicación y se utiliza para la navegación entre las secciones fundamentales, como APOD (Astronomy Picture of the Day), EPIC Images (Earth Polychromatic Imaging Camera), y EONET Events (Earth Observatory Natural Event Tracker). Proporciona una estructura organizada para acceder a las diversas funcionalidades ofrecidas por la aplicación relacionadas con la exploración espacial, visualización de imágenes y seguimiento de eventos naturales en la Tierra.

- Tab Bar Controller en AstronomyPictureViewController: Se implementa un segundo Tab Bar Controller en la clase AstronomyPictureViewController para gestionar la visualización detallada de la imagen del día y la información asociada a la misma. Este controlador permite la transición entre diferentes vistas relacionadas con la imagen seleccionada, lo que puede incluir detalles adicionales, comentarios u opciones de interacción específicas para la imagen del día.

Navigation Controllers

En la aplicación, se implementaron varios Navigation Controllers para mejorar la navegación entre distintas vistas, proporcionando una estructura jerárquica que facilitara la transición entre ellas. Dos instancias clave donde se utilizaron Navigation Controllers son el CollectionViewController y el SearchController.

- En el CollectionViewController, el Navigation Controller se emplea para permitir una navegación fluida entre las imágenes presentadas en la colección. Cuando el usuario selecciona una imagen específica, el Navigation Controller facilita la transición hacia una vista de detalle, donde la imagen seleccionada se muestra en tamaño completo o con información adicional.

- En el caso del Search Controller, el Navigation Controller contribuye a la gestión de la interfaz de búsqueda. Cuando los usuarios interactúan con el campo de búsqueda, el Navigation Controller facilita la transición hacia una vista de resultados de búsqueda, donde se pueden visualizar y explorar los elementos coincidentes. Además, el Navigation Controller permite una navegación coherente al presentar los resultados de búsqueda y, si es necesario, ofrece la posibilidad de navegar hacia vistas más detalladas o relacionadas con los elementos encontrados.

Modelo de Datos

APOD - Astronomy Picture of the Day

El modelo de datos AstronomyPicture en la aplicación desempeña un papel crucial al estructurar y organizar la información relacionada con las imágenes astronómicas diarias (APOD) proporcionadas por la NASA. Este modelo se utiliza para representar y manejar los datos esenciales asociados con cada imagen APOD, ofreciendo una estructura coherente y accesible para su fácil integración en diferentes componentes de la aplicación. Los siguientes son sus atributos representativos:

- title: el título de la imagen
- date: la fecha en que fue publicada
- url: la URL de la imagen (url)
- explanation: una descripción detallada sobre la imagen

EPIC - Earth Polychromatic Imaging Camera

Las imágenes utilizadas en esta porción de la aplicación fueron obtenidas del programa EPIC de la NASA, el cual posee una cámara diseñada específicamente para capturar imágenes de la Tierra desde una perspectiva orbital única. Esta fue montada en el satélite Deep Space Climate Observatory (DSCOVR), que se encuentra en el punto de Lagrange L1, aproximadamente a 1.5 millones de kilómetros de la Tierra en dirección al Sol, la cámara EPIC ofrece una vista panorámica completa y constante de nuestro planeta.

EONET - Earth Observatory Natural Event Tracker

El modelo de datos EONET en la aplicación MyNASA está representado por una estructura de eventos conformada por las siguientes entidades: Event, Category, Source, y EventsResponse. Esta estructura organiza la información esencial sobre eventos naturales y su contexto.

1. Event (Evento):
   - title: Representa el título del evento.
   - id: Proporciona un identificador único para el evento.
   - categories: Contiene información sobre las categorías asociadas al evento, utilizando la estructura Category.
   - sources: Almacena detalles sobre las fuentes relacionadas con el evento, utilizando la estructura Source.

2. Category (Categoría):

- id: Identificador único de la categoría.
- title: Título descriptivo de la categoría a la que pertenece el evento.

3. Source (Fuente):
   - id: Identificador único de la fuente asociada al evento.
   - url: URL que proporciona más información sobre la fuente.

4. EventsResponse (Respuesta de Eventos):

   - events: Contiene una lista de eventos naturales representados por la estructura Event.