ESCUELA DE SISTEMAS Y TECNOLOGÍAS

Transparencias de **ANALISTA DE SISTEMAS**
Edición 2023 - Materia: **Aplicaciones Móviles con Flutter**

TEMA: Introducción a Flutter

Agenda

- Presentación de Flutter
- Instalación y Configuración
- Estructura de un Proyecto Flutter
- Packages
- Diferencias entre Plataformas

Presentación de Flutter (1)

- **Flutter** es un *framework* de código abierto desarrollado por *Google* para la construcción de aplicaciones multiplataforma que utilizan la misma base de código.
- Suele utilizarse para desarrollar interfaces de usuario para aplicaciones móviles en *Android* e *iOS*, pero también soporta desarrollo *web*, *desktop* (*Windows*, *macOS* y *Linux*) e incluso *embedded devices*.
- Utiliza *Dart* como lenguaje de programación y permite desarrollar aplicaciones compiladas nativamente a diferentes plataformas (*ARM*, *Intel*, *Javascript*), lo que les asegura un gran rendimiento.

Presentación de Flutter (2)

- Cuenta con su propio motor de renderizado gráfico para controlar cada *pixel* en la pantalla y lograr apariencias visuales consistentes y flexibles sin importar la plataforma.
- Su característica llamada **Hot Reload** acelera los tiempos de desarrollo, ya que permite ver casi al instante los cambios aplicados, sin perder el estado de la aplicación.

Instalación y Configuración (1)

- Para instalar y desarrollar aplicaciones *Flutter* en *Windows*, el sistema debe cumplir con los siguientes requerimientos mínimos:
  - *Windows 10* o posterior
  - Espacio en Disco: *1.64 GB* (sin contar el espacio necesario para *IDEs* y otras herramientas)
- Además *Flutter* depende de las siguientes herramientas que deben estar instaladas:
  - *Windows Power Shell 5.0* o posterior (ya viene incluido en *Windows 10*)
  - *Git for Windows 2.x* con la opción de utilizar *Git* desde la línea de comando (*https://git-scm.com/download/win*).

Instalación y Configuración (2)

- Instalar *Flutter* es tan simple como descargar el *SDK* desde su sitio oficial (*https://docs.flutter.dev/get-started/install/windows*) y descomprimirlo en algún directorio que no requiera privilegios elevados, ni contenga caracteres especiales ni espacios en su ruta (por ejemplo: *C:\src\flutter*).
- Luego se debe agregar el directorio *bin* del directorio de instalación de *Flutter* al path.
- En nuestro caso, como editor de código utilizaremos *Visual Studio Code* (*code.visualstudio.com*) con las extensiones de *Dart* y *Flutter*.

Instalación y Configuración (3)

- Además tendremos que instalar y configurar *Android Studio* (*https://developer.android.com/studio*):
  - Instalar el último *Android SDK*, *Android SDK Command-line Tools*, *Android SDK Build-Tools*, *Google USB Driver*, *Intel x86 Emulator Accelerator (HAXM)* o *Android Emulator Hypervisor Driver for AMD Processors*, también los plugins de *Dart* y *Flutter*, y por último configurar los emuladores deseados.
- Una vez realizadas todas las instalaciones anteriores, desde la línea de comando ejecutar el comando *flutter doctor --android-licenses* y aceptar las licencias.

Instalación y Configuración (4)

- Para finalizar, volver a ejecutar el comando *flutter doctor* (sin opciones) para comprobar que se cumplan todos los requerimientos. En caso contrario corregir de acuerdo a las instrucciones que brinda la herramienta y volver a comprobar hasta que no aparezcan advertencias ni errores.

Estructura de un Proyecto Flutter (1)

- En *Visual Studio Code* (habiendo agregado las extensiones de *Dart* y *Flutter*), se puede utilizar el atajo *CTRL* + *SHIFT* + *P* y buscar la opción *Flutter: New Project*.
- Luego se deberá elegir la plantilla *Application*, indicar el directorio donde se guardará el proyecto y por último un nombre (en minúsculas y subguiones).
- En la estructura predeterminada que se crea para el proyecto podemos destacar:
  - **/pubspec.yaml**: Archivo de configuración del proyecto en el que, entre otras cosas, se definen su nombre, descripción, versión, versión del *SDK*, dependencias, *assets* (recursos), fuentes, etc.

Estructura de un Proyecto Flutter (2)

- En la estructura predeterminada que se crea para el proyecto podemos destacar (cont.):
  - **/pubspeck.lock**: Archivo autogenerado que contiene las versiones específicas de los paquetes utilizados en el proyecto, para asegurar que cuando se reconstruya se obtengan siempre las mismas versiones compatibles entre sí.
  - **/analysis_options.yaml**: Archivo que permite configurar el analizador estático de *Dart* para definir los tipos de advertencias, errores y mejoras de código que deben ser informados.

Estructura de un Proyecto Flutter (3)

- En la estructura predeterminada que se crea para el proyecto podemos destacar (cont.):
  - **/android, /ios, /web, /windows, /macos, /linux**: Directorios con los proyectos específicos de cada plataforma (se pueden realizar configuraciones específicas según el tipo de proyecto generado).
  - **/lib**: Directorio donde se guarda el código fuente de la aplicación. Aquí van los archivos *.dart* (se pueden organizar en una estructura de subdirectorios) incluyendo el archivo *main.dart* que contiene el método *main()* (punto de entrada a la aplicación). Desde dicho método se invoca la función *runApp(Widget)* para iniciar la aplicación.

Packages (1)

- *Flutter* proporciona una amplia gama de paquetes (**packages**) que ayudan a los desarrolladores a agregar funcionalidades a sus aplicaciones de manera rápida y fácil.
- Los paquetes son módulos de código reutilizable que pueden ser importados en un proyecto de *Flutter* para agregar funcionalidad.
- Pueden ser encontrados en el repositorio de paquetes de *Dart* (**pub.dev**) o en otros lugares como *Github*.
- Los paquetes pueden ser de dos tipos: paquetes que se mantienen directamente por la comunidad de *Flutter,* o paquetes que son mantenidos por terceros.

Packages (2)

- Ayudan a reducir el tiempo de desarrollo y mejorar la calidad de la aplicación al permitir que los desarrolladores se centren en la funcionalidad principal sin tener que escribir todo desde cero.
- Pueden ser agregados a un proyecto de *Flutter* editando el archivo *pubspec.yaml* y agregando la dependencia del mismo.
- Algunos paquetes populares en *Flutter* son:
  - **http**: para hacer solicitudes *HTTP*.
  - **flutter_bloc**: para implementar el patrón de arquitectura de software *BLoC*.
  - **shared_preferences**: para almacenar datos simples en la aplicación.

Diferencias entre Plataformas (1)

- Como ya se ha mencionado, *Flutter* es un marco de trabajo de desarrollo de aplicaciones móviles multiplataforma que permite a los desarrolladores crear aplicaciones nativas para *iOS* y *Android* desde un único código base.
- Para manejar diferencias entre plataformas, *Flutter* proporciona varias herramientas y enfoques.
- **Widgets específicos de la plataforma**:
  - *Flutter* proporciona una gran cantidad de *widgets* que se adaptan automáticamente al aspecto y la sensación de la plataforma subyacente.

Diferencias entre Plataformas (2)

- **Widgets específicos de la plataforma (cont.)**:
  - Por ejemplo, se puede usar *CupertinoButton* para obtener un botón que se ve y se comporta como un botón de *iOS*, y *MaterialButton* para uno que se ve y se comporta como un botón de *Material Design* en *Android*.
- **Plugins y paquetes específicos de la plataforma**:
  - *Flutter* permite acceder a la funcionalidad específica de la plataforma a través de *plugins* y paquetes.
  - Se pueden integrar fácilmente características como la cámara, el *GPS*, las notificaciones *push*, etc., utilizando paquetes específicos de la plataforma.

Diferencias entre Plataformas (3)

- **Condicionales basados en plataforma**:
  - Se pueden usar condicionales basados en la plataforma para ejecutar código específico de dicha plataforma cuando sea necesario.
  - Por ejemplo, se puede utilizar *Platform.isIOS* y *Platform.isAndroid* para ejecutar diferentes fragmentos de código en función de la plataforma en la que se ejecute la aplicación.
- **Canales de mensajes**:
  - Flutter ofrece un mecanismo llamado "canales de mensajes" (*platform channels*) que permiten comunicar el código *Dart* de la aplicación y el código nativo de la plataforma.

Diferencias entre Plataformas (4)

- **Canales de mensajes (cont.)**:
  - Esto es útil cuando se necesita realizar operaciones específicas de la plataforma que no están disponibles a través de plugins existentes.