Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Curso: Ingeniería de Software 1 (2016701)
Angel Santiago Avenda﷿o Ca﷿on

Flutter

- Flutter puede ser descrito como un SDK o framework, desarrollado por google con fecha de lanzamiento en 2018. Su utilidad principal era la de proporcionar una herramienta fácil de aprender para desarrollar aplicaciones móviles en varias plataformas sin la necesidad de sobre-escribir el código para cada plataforma, de esta manera proporcionando un desarrollo ágil y limpio. En su lanzamiento original, tenía soporte para aplicaciones en plataformas Android y iOS, pero a lo largo de los años se han añadido otras plataformas como Windows, Linux, MacOS, y WEB.

  Esto lo logran a través del concepto del middleware, el desarrollador trabaja sobre el framework, y este traduce al motor gráfico de la plataforma en cuestión. Esto sin embargo, trae consigo problemas notables de rendimiento, en comparación a las aplicaciones elaboradas de manera nativa en su propia plataforma. Un concepto usado por el lenguaje de programación JAVA

  Para el programador, el proceso de desarrollo se simplifica de gran manera, y no necesita conocimientos de la plataforma de manera nativa para poder desarrollar aplicaciones de manera fácil y rápida.

- El framework está desarrollado alrededor del lenguaje de programación Dart. Un lenguaje de código abierto desarrollado por google, lanzado en el año 2011. Lanzado como una alternativa más moderna para programación web, usualmente este lenguaje es referido como un “Lenguaje estructurado pero flexible para la programación web”. Al usar el lenguaje, pueden verse ciertas características que lo asimilan a otros lenguajes, como Java, o C + +, pero ofreciendo una mucha mayor flexibilidad y curva de aprendizaje (Opinión personal).

- Se recomienda el uso de un Entorno de Desarrollo Integrado (IDE) para el desarrollo de aplicaciones usando flutter, si se busca desarrollar para plataformas android, es imperativo la instalación de la herramienta gratuita Android Studio y el JDK
  Para dispositivos propiedad de Apple, se necesita un equipo Apple (O

una máquina virtual con sistema operativo macOS) para desarrollar orientado a estas plataformas.

- Para otras plataformas como WEB, Linux o Windows no es necesario de una plataforma/dispositivo/aplicación adicional para trabajar. Personalmente recomiendo el uso de la suite de Jetbrains, con IntelliJ como la estrella para programar en flutter. Una segunda opción sería Visual Studio Code por su amplia cantidad de plugins y herramientas que ayudan a mantener un código limpio.

- Flutter usa el sistema de control de paquetes por defecto de Dart, conocido como "pub", en la página pub.dev se almacenan todos los paquetes desarrollados por la comunidad, incluyendo grandes empresas que crean API's para uso público, posee su propio sistema de control de versiones, y mantiene un historial inédito de todos los paquetes y sus actualizaciones, esto con el fin de evitar problemas en aplicaciones que dependan de estos paquetes. En los proyectos de flutter, se debe poseer un archivo con el nombre pubspec.yaml, este archivo contiene todas las dependencias del proyecto, incluyendo el número de versión de cada uno, versión de flutter y versión de Dart.

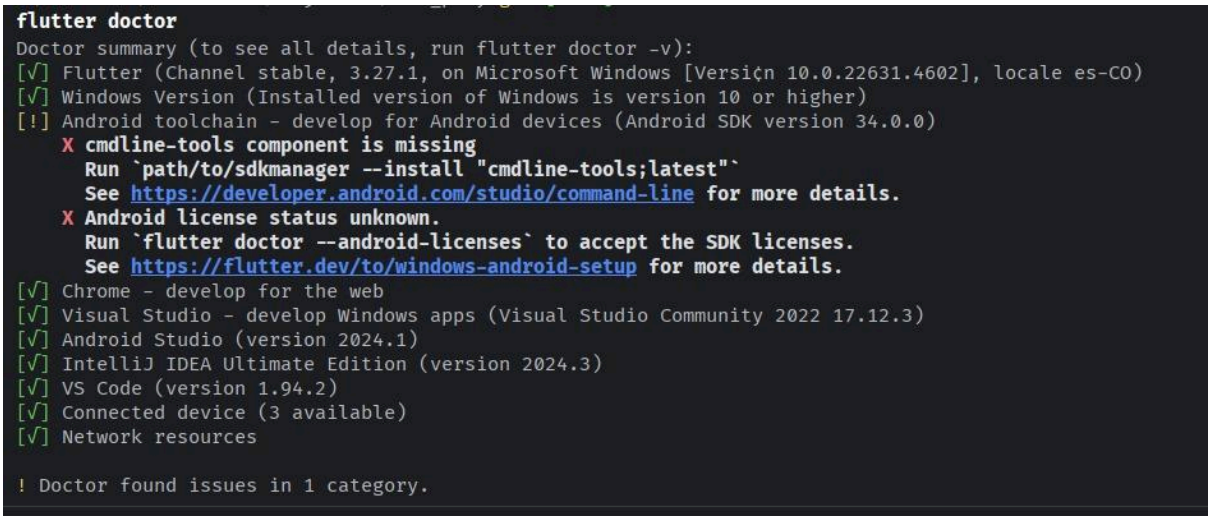
  Añadir, actualizar y eliminar paquetes es un proceso simplificado/unificado para todas las plataformas. Aquí los comandos más esenciales/utilizados en un flujo de desarrollo:

Comando	Función
flutter pub add <package name>	Añade un paquete al archivo de dependencias
flutter pub remove <package name>	Elimina un paquete del archivo de dependencias
flutter clean	Elimina toda la caché de los paquetes instalados en el proyecto
flutter pub get	Descarga e instala todos los paquetes del archivo de dependencias
flutter upgrade	Actualiza todos los paquetes sin dependencias transitivas a la última versión posible (Los conflictos se deben solucionar manualmente)

- Se puede descargar el SDK desde la página oficial de flutter, este traerá incluido Dart dentro de sus archivos:

  https://docs.flutter.dev/get-started/install

- Tras instalar el flutter SDK, se puede validar el estado de la instalación por medio del comando “flutter doctor”, que lanzará un conjunto de estadísticas y recomendaciones sobre el estado de la instalación, incluyendo número de versión, canal (stable o beta) (Hace referencia a la versión de flutter) y otra información relacionada.



Estructura recomendada del proyecto:

- Se recomienda usar los principios de Clean Architecture para flutter, se puede encontrar diversa literatura para el tema. La filosofía se basa en diversos principios, incluyendo Separation of COncerns(SoC), dependency inversion y principio de única responsabilidad.
  En el desarrollo de aplicaciones móviles, es necesario crear estructuras fuertes, fáciles de mantener y escalables. A medida que se añaden más funcionalidades a un proyecto, mantenerlo es más difícil si no se organiza desde un inicio.

  Se usa mucho el modelo “Vista-Modelo-VistaModelo” (MVVM), que es un patrón de diseño usado comúnmente en el desarrollo de interfaces de usuario. También asociado a frameworks que soportan la asociación de información (data binding) cómo React, Vue, etc. Estos son frameworks donde la interfaz se actualiza de manera automática cuando ocurren cambios en la capa de información y viceversa.

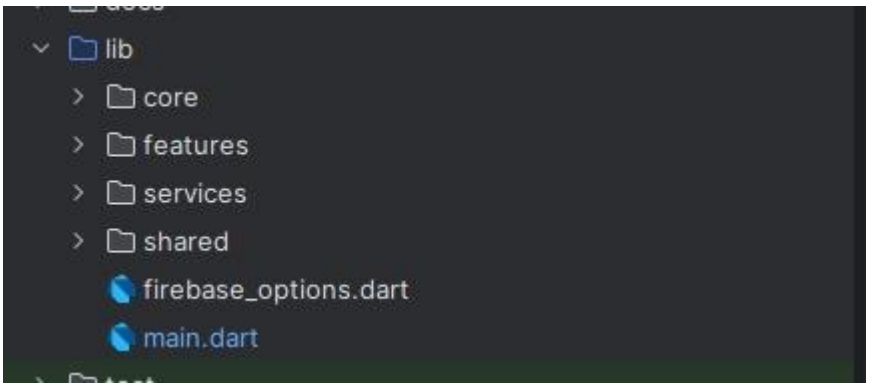
¿Qué es un modelo?: Los modelos hacen referencia a la lógica de negocio de la aplicación, estos son responsable de manejar la información y asegurar su consistencia e integridad. Los modelos son comúnmente independientes de las interfaces de usuario y son reutilizables en cualquier parte de las capas de presentación.

¿Qué es una vista?: Los elementos de vista son los responsables de representar la información de manera visual, lo que verá y con lo que interactúa el usuario final. Se intentan mantener lo más simples a nivel código posible, y su único propósito es el de mostrar información, no maneja lógica de negocio.

¿Qué es una Vista Modelo?: Éste es el intermediario entre la vista y el modelo, contiene la lógica de presentación, exponiendo la información y los comandos/funciones a los que la vista se puede vincular. Es normalmente diseñado para ser testable de manera independiente a la interfaz de usuario, normalmente también maneja las interacciones de usuario.

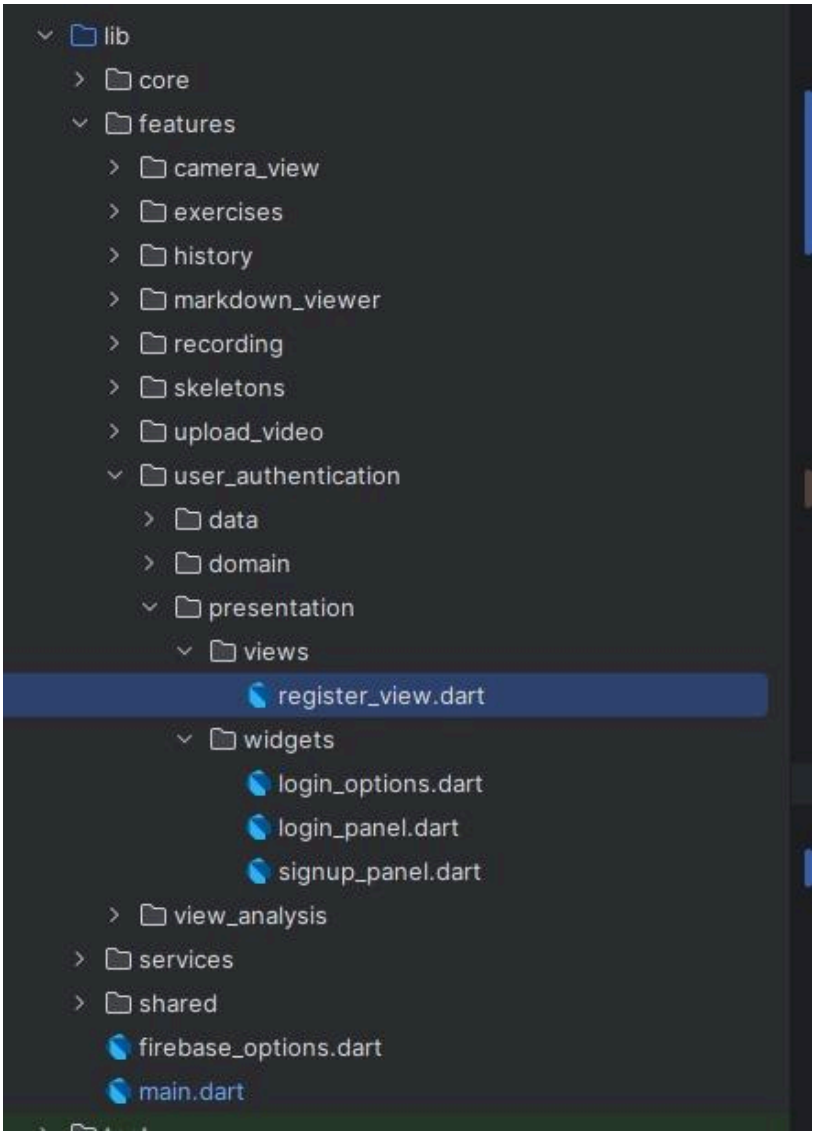
¿Cómo conectar el Vista modelo con la Vista?: Normalmente se usan paquetes de control de estados. Tales como Provider, o Riverpod, Otra forma de manera nativa es usar los Stateful Widget, nativo del framework. Estas herramientas te permiten manejar el estado de un "Widget", y actualizar la interfaz en función de este.

- Cada característica de la aplicación se divide en 3 capas, Capa de presentación, Capa de dominio y capa de información. (PResentation Layer, DOmain Layer, Data Layer). El usuario interactúa únicamente con la capa de presentación.
  La carpeta con todo el código de la aplicación se conoce como la carpeta "lib", esta se subdivide en 4 principales.
  - Core
  - Features
  - Shared



Ejemplo de estructura carpetas principales (Ejemplo propio).

La carpeta features contendrá todas las características principales de la aplicación subdivididas en las 3 capas mencionadas



Ejemplo de estructura features (Ejemplo propio)

La carpeta CORE contiene todos los componentes claves como routers, servicios de red, validadores y estilos de la aplicación

- Para nombres de archivos, clases y carpetas, se usan los principios de clean code.
    - Para nombres de archivos y carpetas se usan sustantivos singulares, separados por _.
    - Para clases y tipos, incluyendo enums y typedefs se usa PascalCase
    - Se sugiere evitar el uso de acrónimos y abreviaciones, es mejor colocar el nombre completo de un componente por largo que sea

- Para más información acerca de los estilos recomendados a usar para el lenguaje de programación dart puede dirigirse al siguiente link: https://dart.dev/effective-dart/style