Repaso Final Del Corte

MAPAS GOOGLE - OSMP

Mapas (Google, OSM)]:

-FRAGMENTOS: en vez de actividades.

-Un pedazo de la pantalla con un ciclo de vida diferente.

-Es como una página web.

-Tiene un papá que es la actividad.

-EL MAPA ES UN FRAGMENTO (Demora en responder y es un pedazo de la pantalla).

-A continuación vemos la implementación y características de algunas APIS de mapas.

-A) GOOGLE:

-Al asistente se le dice new y a “google maps activity”.

-Se necesita una actividad, un layout / Nos genera una llave con la que nos autenticamos en Google.

-Api Key: Nos metemos en la consola de Api de Google, creamos un proyecto nuevo, nos vamos a Apis y Servicios, librerías,

-JAMAS SUBIR ESTE ARCHIVO POR EJEMPLO A GITHUB.

-”MAPS_API_KEY=Aiza….”.

-Se le puede programar el zoom o que el usuario al usar el “setzoomGesture” para acercar la pantalla o alejarla le funcione.

-En el Manifest al crear la actividad con Mapas (New, Activity, Gallery, GoogleMaps) se pone la API KEY de la credencial de Google: “https://console.cloud.google.com/”.

-Se crea un proyecto.

-Se le da a APIS y Servicios.

-Mapas, SDK, ENABLE.

-MARCADOR: Un puntito, se necesita latitud y altitud, usamos “LatLng”.

-Para poder quitar los marcadores se hace teniendo el objeto (medio complicado) o con “mMap.clear()” (la fácil xd).

-Página para configurar el mapa de Google:

-”https://mapstyle.withgoogle.com”.

-Genera un .Json y le damos a “copy Jason”.

-Se lo ponemos dentro en la zona de assets, o raw (new, folder, raw…) → Recordar que no tenga ningún carácter en mayus.

-Para usarlo se pone: “`mMap.setMapStyle(MapStyleOptions.loadRawResourceStyle(this, R.raw.NombreDelJson))`”.

-Sensores:

-Temperatura presión, luminosidad, acelerómetro, giroscopio, proximidad, etc.

-El sensor de de luz (0 - 40.000), acá podemos decir que a cierto rango poner el mapa normal y a otro el mapa #2.

-B) OPENSTREETMAPS:

-Ver diapositivas jijiiji.

-Rutas: Toca importar su librería, añadir en gradle un nuevo repositorio para la dependencia.

FIREBASE:

[Firebase]:

-Multiusuario:

-(Vamos a trabajar con servicios para conectar usuarios.).

-Backend: Se usa uno que ya existe (Google Firebase) o se hace uno manualmente (En ambos casos se necesita una maquina), las empresas cuando son bastantes grandes migran a una solución propia.

-FIREBASE:

-Vamos a ver hoy como en Firebase usar una Base de datos, la autenticcación y el storage.

-AUTENTICACIÓN: (Usamos una API) / tools + firebase / Se necesita una cuenta de Google, en donde tengamos un espacio de capa gratis (firebase.google.com).

-En VS:

(tools + firebase + authentication + using google + connect to firebase + se crea el proyecto (todo en la nube) + Cuando se crea se le da a Connect. y aparece ya en AS en verde eso. /// Esto crea un archivo en el src de “google-services”).

-Ahora se le da en “Add firebase ….” y así sincroniza todo y aparece en verde

-En “https://console.firebase.google.com/”, en Authentication, get started.

-STORAGE:

-Esta es “createUserWithEmailAndPassword” (Ver diapositivas).

-Firebase Databases:

-Realtime (sencillo - Este es el que usaremos /// Un documento).

-Cloud Firestore (más complejo /// Muchos documentos).

-Esta hecho para que se suscriban a un path en particular como por ejemplo “users”(real time).

-Acá entre menos anidamiento mejor, así toque repetir todo con IDs.

-Acá se va a Assitant, se devuelve a firebase y se selecciona realtime database y se sincroniza el gradle “segunda opción”.

-Nos devolvemos a la pagina, a todos los productos, a realtime database, crear base de datos y ya se puede correr la app.

Consulta y Suscripción / Parse Platform / Notificación / Rest:

[Storage (Final de la anterior clase)]:

-Un producto para guardar, fotos, imágenes, videos, audios, nuevamente nos vamos a 2console.firebase.google.com”, le damos en get started al cloud storage.

-Y es lo mismo de nuevo de la anterior.

[Parse]:

-Es gratis toda la infraestructura de Google, como lo mismo que firebase, pero no la infraestructura creo.

-Docker: Fácil de transportar a través de diferentes plataformas, la app se ejecuta de forma aislada y se pueden reutilizar los volúmenes de datos.

-Funciona al definir un archivo al definir la estructura, con esto se puede construir una imagen que representa el contenedor y eso mismo se corre.

-Docker Compose: Se refiera a una herramienta para la ejecución de varios contenedores a partir de un archivo descriptivo (yml).

-Se crea la VM instancia, permitimos trafico http, https.

[Notificación]:

-Ya tu sabes

[Rest]: (Representational state transfer):

-Rest usando Volley, es una librería cpnstruida por Google para consumir servicios Rest, para usarla se añade una dependencia del gradle

---

FLUTTER #1:

[Flutter]:

-Es un framework super reciente multiplataforma, sirve para android y IOs, windows, linux, web.

-Lo hizo Google.

-Basado en Material Design, no depende del navegador.

COMANDO DESDE VISUAL PARA CREAR UNA APP EN FLUTTER:

-"flutter create nombreAPP" (EL NOMBRE SIN ESPACIOS NI CARACTERES RAROS, SI SE PUEDE MAYUS Y MINUS).

-Todo el codigo se usa en "libs", las pruebas en "test", "pubscpec.yaml" es como una mezcla con el manifest.

-Acá hay puros "widget" y todo viene de una "widget raíz" ().

-Con estado: (Puede cambiar durante la ejecución - corren el build cada vez que detectan un cambio en su estado).

-Sin estado: (Lo que pinte al inicio no cambia para nada - corren el build creo que una única vez).

-El mismo codigo que se usa para el controlador es el de la vista.

-Escrito “stl” y con las segunda opción del widget y se le cambia el nombre.

-Al final esta el método build como ya se mencionó

[Dart]:

-Lenguaje de programación diferente a java, kotlin, swift, c++, python, etc.

-Es fuertemente tipado y tiene inferencia de tipos.

-final (una vez le asigno el valor no cambia, pero durante la ejecución se le asigna el valor).

-const (lo mismo de arriba, pero esta es desde antes de arrancar el programa ya asigna el valor).

-Se imprime con print.

-Los mismos constructores de java.

-Existen funciones anónimas.

-A las funciones con corchetes cuadrados significa que se puede llamar a esa función de diversas maneras y no ese fundamental poner todos los datos.

-El “Null safety”

FLUTTER #2:

[Imágenes]:

-Crear en el yaml:

```
# To add assets to your application, add an assets section, like this:
  assets:
     - assets/logi.png
  #   - images/a_dot_ham.jpeg
```

-Y se pone donde se necesite: “child: Image.asset("assets/logi.png")”.

[Provider]:

-agregar dependencia

-Definir el modelo para el estado (se define una clase “GuesState” que extiende de “ChangeNotifier” para qcada uno de los atributos que se identificaron como estado, o sea en message, random, counter y el finish).

-Registrarlo en el contexto (En nuestro caso el registro es en el MaterialApp).

-Operaciones sobre el estado (watch, read, select —→ usarlo en lo que se crea necesario).

IOS-XCODE Intro a Swift:

[IOS-XCODE Intro. a Swift]:

-Es orientado a objetos y funcional

-Una variable puede cambiar su valor a lo largo de la ejecución, una constante no (Mutable vs Inmutable).

-Tiene inferencia típado.

-Tiene interpolación con “\(manzanas)” sin las comillas obvio xd.

-El arreglo comienza desde la posición 0.

-condicionales normales, solo que los paréntesis no son obligatorios.

-Aquí no hay “++” o “--”,

-”repeat” es como el while pero con una diferencia que no me acuerdo xd.

-”…” es como “≤” y “..” es como “<”.

-”?” es una variable que puede ser nula en algún momento, o sea ser un “optional”, tiene dos partes, una que “nil” que aquí es como un nulo y la otra es el tipo de dato..

-”if let” es el “si es diferente de nulo haz lo siguiente” / “guard let” es lo contrario, o sea “si es que si es nula entonces haz esto”.

-Acá todo son “Struct” ya que este lenguaje esta enfocado al lado funcional y no tanto la herencia, encapsulación, etc.

-Las “Tuplas” son un grupo de cosas.

-PROGRAMACIÓN FUNCIONAL:

-Las funciones son tipos de datos.

-”(String) → Bool”: Significa que es una función que recibe un String y retorna un Bool.

-Closures: (Es un bloque de código anónimo que se puede invocar. Una función es un tipo de closure).

-Las estructuras son “value types” y las clases ya que escalan son (referenced types).
