

Jefferson Elías Calix

Inteligencia artificial | Hector helliel sabillon

IAssistant18

Documento de soporte

1. Detalles Técnicos

Propósito

Este documento tiene como propósito explicar todos los elementos de programación y herramientas utilizadas en el desarrollo de la aplicación IAssistant18.

Antecedentes

Anteriormente se presentó un documento “Propuesta” en el cual se dieron detalles e hitos que la aplicación debía cumplir, los cuales se cumplieron y superaron, adelante en el informe serán explicados.

¿Qué es IAssistant18?

Es un asistente de voz que en un principio planteaba solo la charla coherente con el usuario, a través de un solo agente (Jeffy), finalmente me he tomado el tiempo de superar esto y de implementarlo con dos agentes (Jeffy e Izzy).

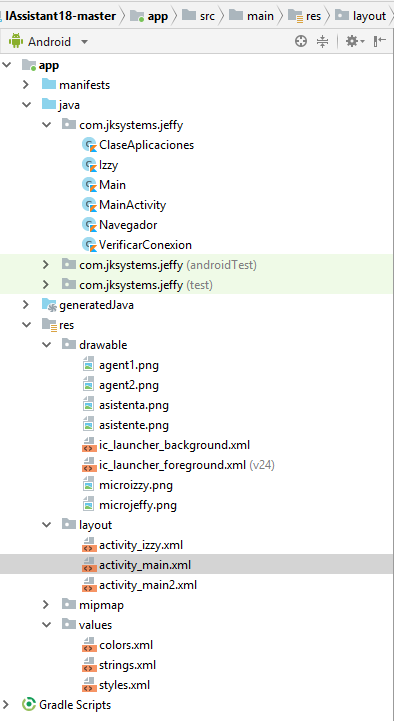
¿Qué acciones cumple IAssistant18?

* Charla convencional con el usuario
* Interfaz para ambos sexos
* Realización de operaciones matemáticas básicas
* Interacción con otras aplicaciones
* Consultas web

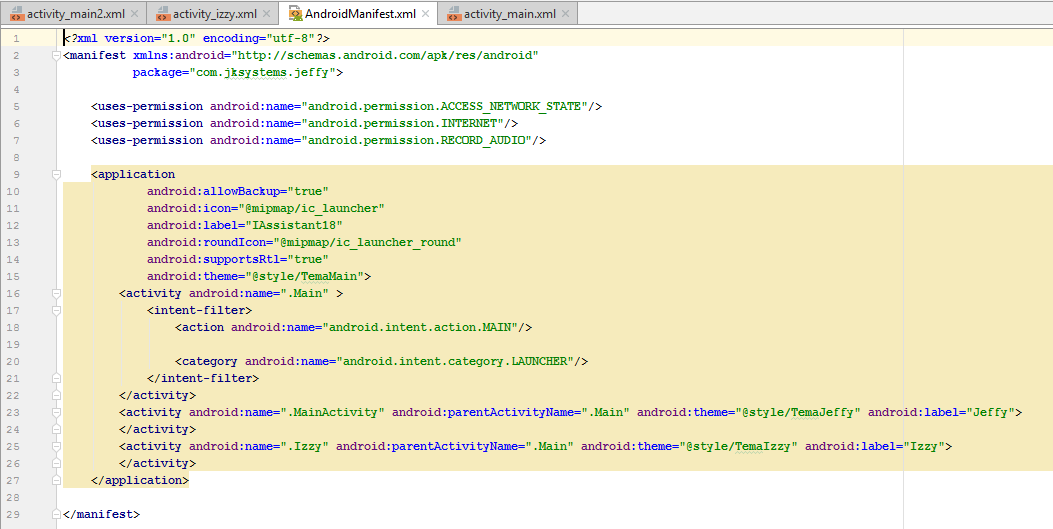
1. Desarrollo de la App

Android Studio

Toda la programación de la app está hecha en la IDE de Android Studio, también su interfaz gráfica, y la lógica de todo su funcionamiento



Manifest



En manifest, podemos destacar los permisos que asignamos

* Android.permission.ACCESS\_NETWORK\_STATE: para permitir a la aplicación saber el estado de conexión del teléfono.
* Android.permission.INTERNET: para permitir a la aplicación que acceda a internet
* Android.permission.RECORD\_AUDIO: permite a la aplicación la grabación de voz

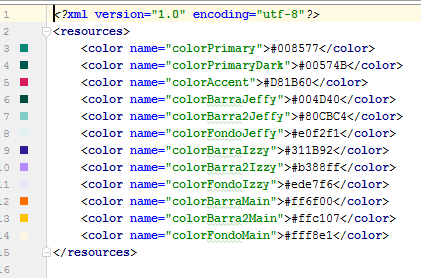
También podemos ver el llamado de los temas que están alojados en “Styles”

* Android\_theme = “styles/TemaJeffy”
* Android\_theme = “styles/TemaIzzy”

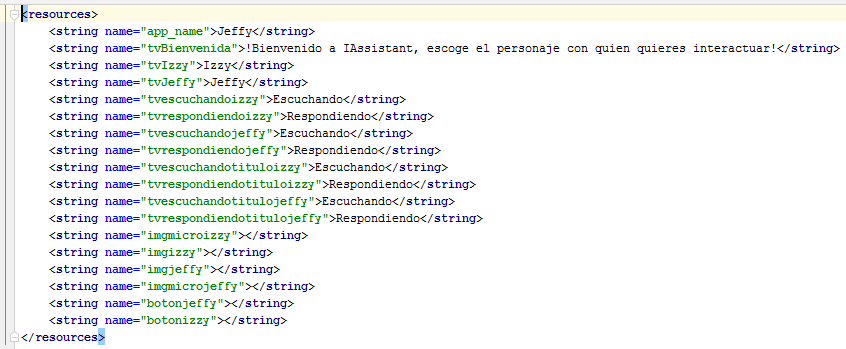
Values

Dentro de values podremos observar los colores utilizados, siguiendo material design, los strings y los estilos de la aplicación.

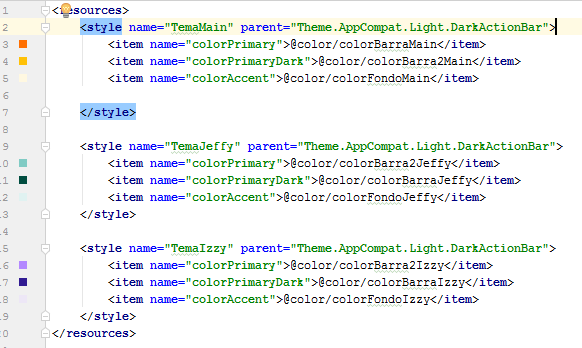
* COLORS



* STRINGS

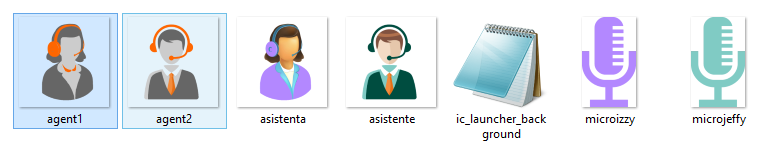


* STYLES



Drawable

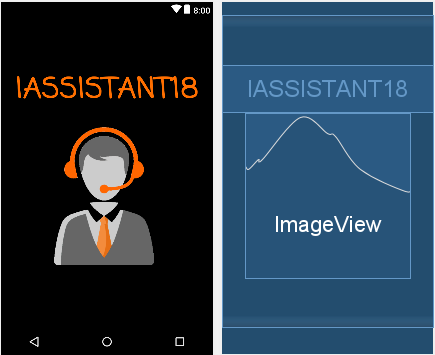
Dentro de este apartado van todas las imágenes que se utilizaron en la app



Layouts

Utilice 3 layouts, uno que representa al main de la aplicación y 2 que representan a los agentes, cada layout ligado a una clase.

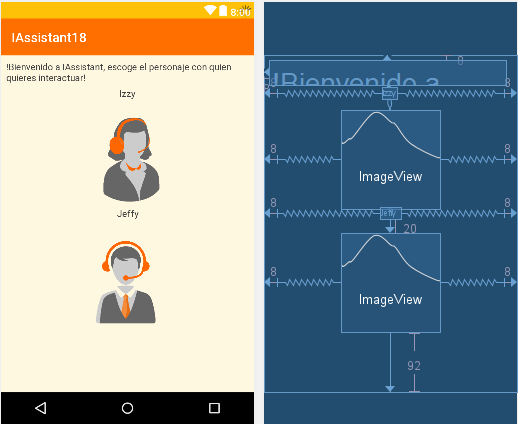
1. Splash



Codigos Externos

* Utilice “HANDLER” y “Delay” para manejar el tiempo que iba a estar activo el splash.

1. Main

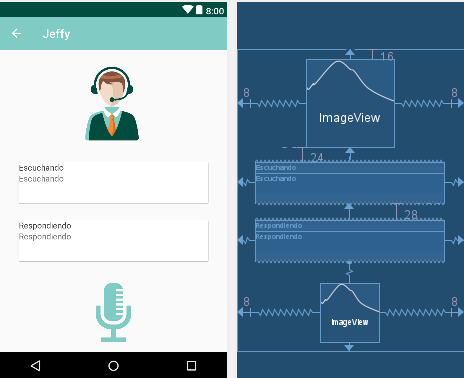


Código del Main



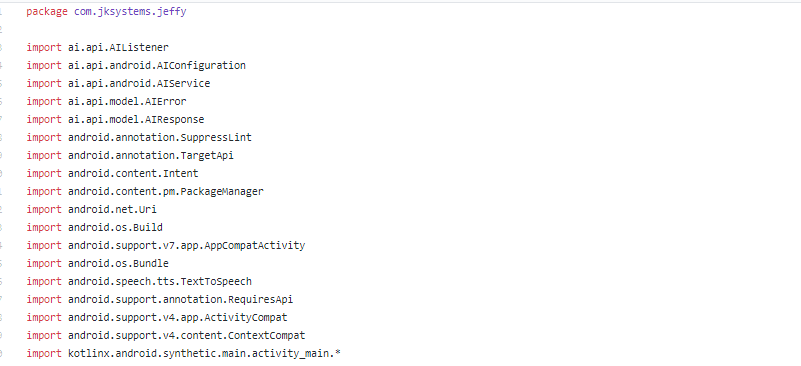
* Arriba importamos todas las clases para que los widgets funcionen
* Convertimos los Imageview en botones (Izzy y Jeffy)
* Colocamos un listener de clics en cada botón para iniciar el intent que abre los agentes, antes de eso, llamamos a la clase “VerificarConexion” para saber el estado de red del teléfono.
* Si VerificarConexion no fuese “true” la aplicación no nos permitirá acceder a los agentes.

1. Jeffy

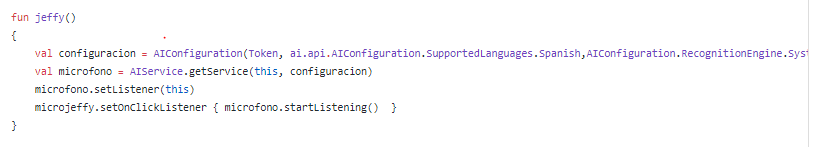


Codigo de Jeffy

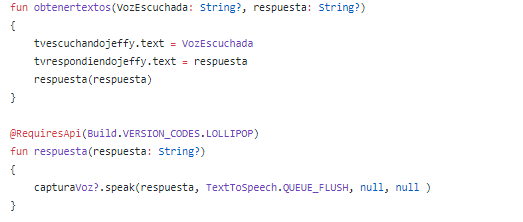
* Librerias implementadas para los widgets de la clase



* Esta función inicializa el servicio de la api de dialogflow(descrito mas adelante), también al botón “micrófono” le colocamos un listener para que comience a escuchar lo que el usuario dice.



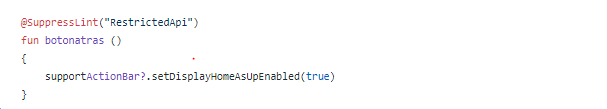
* La función “obtenertextos” recibe 2 parametros, los cuales serán igualados a los 2 textbox que contiene el layout, mientras la función “respuesta” acciona la función “speak” del servicio AI, que interpreta el texto y literalmente lo habla, para que el usuario lo escuche.
* @RequireApi, es un consejo que Android studio nos envía para recordarnos que ciertas funciones requieren de ciertas API para funcionar.



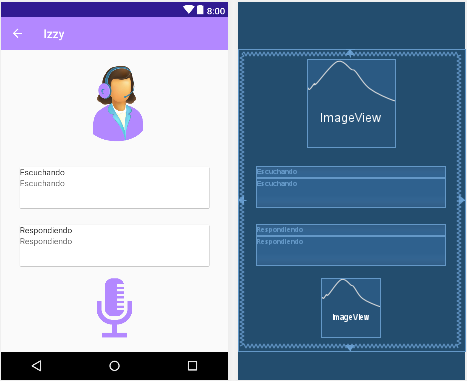
* La función “onListening” trae varias subfunciones, dentro de estas la mas importante es la OnResult, en la cual capturamos en variables, lo que le decimos a la aplicación y lo uqe esta nos contestara, además llamamos a la clase Aplicaciones.



* Esta función añade un botón para retroceder el activity de Jeffy



1. Izzy

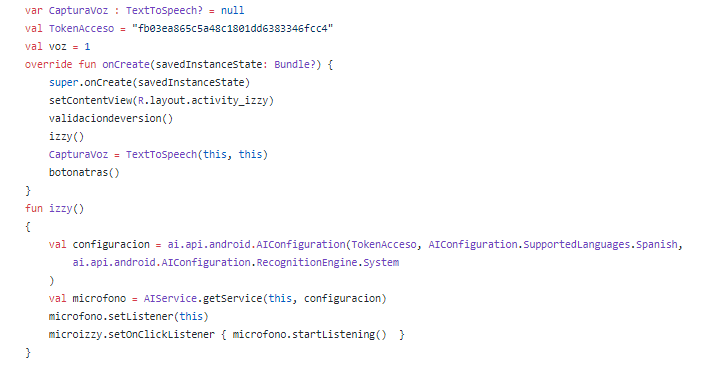


Código de Izzy

• Librerias implementadas para los widgets de la clase

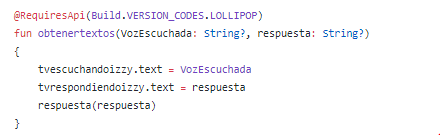


• Esta función inicializa el servicio de la api de dialogflow(descrito mas adelante), también al botón “micrófono” le colocamos un listener para que comience a escuchar lo que el usuario dice, también viene el token de acceso que nos da dialogflow.



• La función “obtenertextos” recibe 2 parametros, los cuales serán igualados a los 2 textbox que contiene el layout, mientras la función “respuesta” acciona la función “speak” del servicio AI, que interpreta el texto y literalmente lo habla, para que el usuario lo escuche.

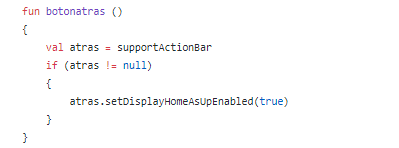
• @RequireApi, es un consejo que Android studio nos envía para recordarnos que ciertas funciones requieren de ciertas API para funcionar.



• La función “onListening” trae varias subfunciones, dentro de estas la mas importante es la OnResult, en la cual capturamos en variables, lo que le decimos a la aplicación y lo uqe esta nos contestara, además llamamos a la clase Aplicaciones.

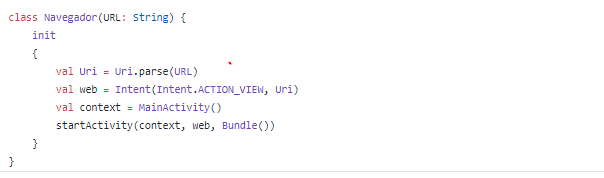


• Esta función añade un botón para retroceder el activity de Jeffy



1. Navegador

* Colocamos “Init” para inicializar todas las variables
* Uri es una variable que detectara la URL que enviemos
* Luego abrimos el intent utilizando ACTION\_VIEW y mandando la URL a la que accederemos



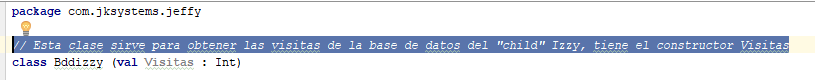
1. VerificarConexion

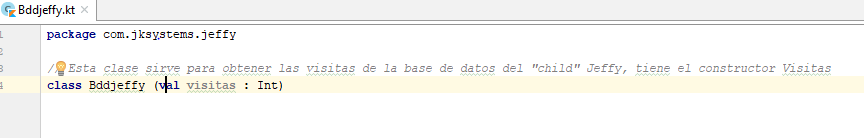
* Obtenemos el servicio de conectividad
* Instanciamos el servicio para obtener el estado de red del teléfono
* Retornamos que el teléfono este conectado.

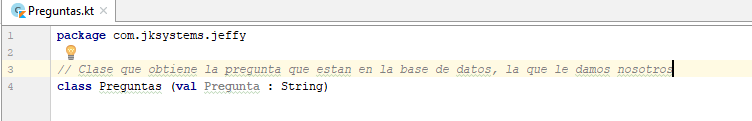


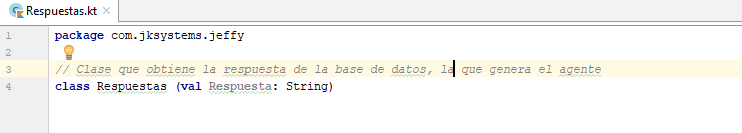
1. Codigos de base de datos

En general:





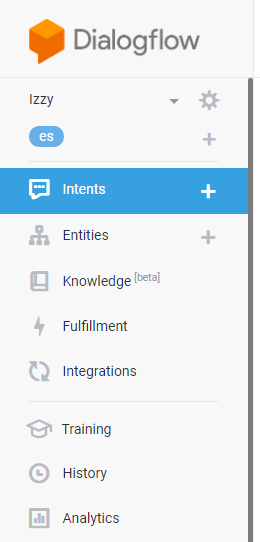
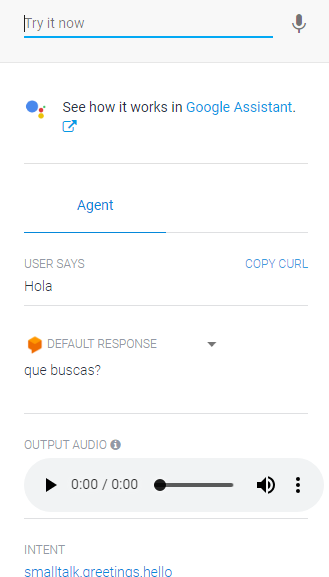




* Cada clase tiene contructores que luego son llamados en las clases utlizadas, tienen implementación Firebase.

1. Software extra utilizado
2. Dialogflow

Es una API de google, la cual nos facilita la creación y entrenamiento de agentes virtuales, dándonos la posibilidad de integrar estos a varias plataformas de desarrollo de software, en mi caso Android Studio, la API nos comparte un token de acceso, y nos da la posibilidad de utilizar otros elementos como los webhooks.

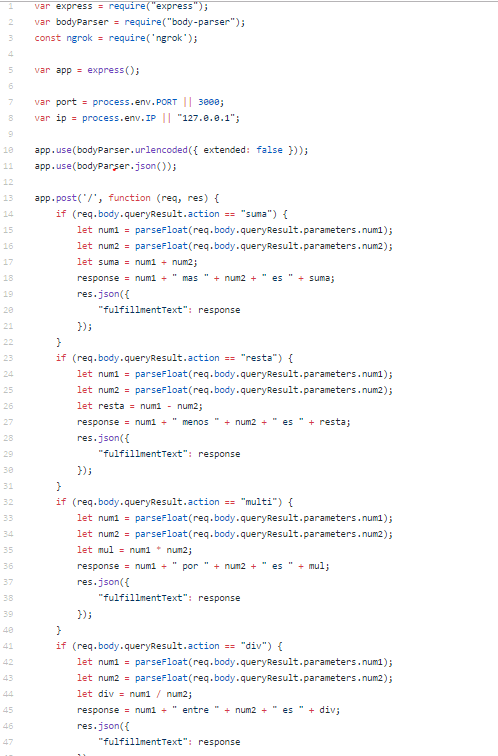
 

1. Webhook

A partir de mi experiencia para mi Webhook es como un servicio interprete entre 2 API, yo lo implemente ayudándome de Node JS, para poder conectar el servicio Webhook de Dialogflow con mi aplicación y que pudiese funcionar de forma plena.

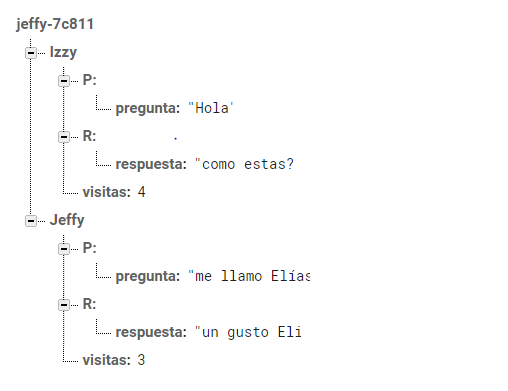
1. Node JS

Con node js a traves de la instalación de paquetes npm pude colgar en la web el servidor, que contiene mi código “js” con el cual la aplicación puede hacer las operaciones matemáticas.



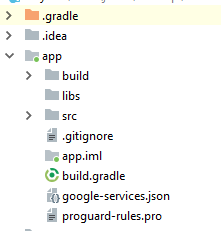
1. Firebase

Firebase es una API de Google que tra una dependencia llamada Database en la cual hay 2 opciones, yo opte por Real Time DataBase, porque ofrece actualizaciones de datos en tiempo real.

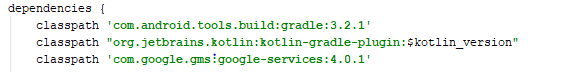


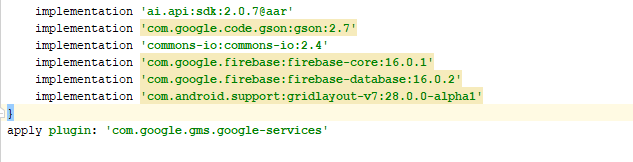
Para implementarlo se debe conectar la API a nuestra aplicación:

* Ponemos el nombre de nuestro paquete
* Descargamos un archivo json



* Procedemos a implementar en el gradle





* Finalmente sincronizamos la aplicación.

Jefferson Elías Calix

Inteligencia artificial | Hector helliel sabillon