



Universidad  
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA  
INFORMÁTICA

GRADO EN INGENIERÍA DE COMPUTADORES

Curso Académico 2019/2020

Trabajo Fin de Grado

AMPLIACIÓN Y MEJORA DE UNA APLICACIÓN PARA EL  
ASISTENTE VIRTUAL ALEXA PARA EL APRENDIZAJE  
DEL ALGORITMO DIJKSTRA

Autor: Alberto Barragán Costo

Director: Maximiliano Paredes Velasco



A mis padres, por cuidarme y enseñarme como lo han hecho. Para quitaros por fin esa espinita y así podáis seguir estando orgullosos de mí. Sin vosotros esto no hubiese sido posible. Gracias. Esto es por y para vosotros.

A mi pareja, por aguantar tanto y durante tanto tiempo que ha llevado la realización de este proyecto. Sin tu apoyo no habría sido posible. Gracias

A mis compañeros de clase, su ayuda desde hace 9 años es la que ha hecho posible que la carrera concluya aquí. Sin vosotros nada habría sido igual. Gracias a todos.

Al tutor, por estar siempre disponible y atento a todas mis necesidades. Por confiar en mi proyecto, apostar por él y ayudarme con todas sus propuestas. Gracias por todo el apoyo.

# Resumen

*Aprendamos Algoritmos* es una aplicación desarrollada para el asistente virtual Alexa capaz de comunicarse con una aplicación Android y cuya finalidad es poder enseñarle al usuario cómo resolver un algoritmo Dijkstra.

El objetivo del presente proyecto es ampliar esta aplicación dotándola de nuevas funcionalidades y cubriendo ciertas carencias que posee para que pueda ser desplegada en la tienda de aplicación y posteriormente utilizada por todos los usuarios que así lo deseen.

Para el desarrollo de este proyecto, se ha realizado un estudio del comportamiento previo de la aplicación, para así ver las carencias que poseía y establecer los objetivos a realizar. La principal carencia que poseía la aplicación era la escalabilidad, por lo que este proyecto va a centrar gran parte de sus esfuerzos en la mejora de este ámbito.

La aplicación previa a la realización del presente proyecto disponía de un grafo a modo de ejemplo para la enseñanza del algoritmo al usuario y un grafo para poner a prueba al mismo. En este proyecto se le va a dotar a la aplicación de distintos niveles de dificultad a la hora de poner a prueba al usuario y por tanto de diferentes problemas a resolver por parte del mismo. La generación del código entendible por el asistente virtual Alexa, se realizará de forma automática para facilitar la escalabilidad a la hora de desarrollador de la aplicación.

Anteriormente, la aplicación para dispositivos móviles utilizada para la comunicación con la aplicación de Alexa, estaba desarrollada en Kotlin y únicamente disponible para el sistema operativo Android. En este proyecto se ha realizado una aplicación para dispositivos móviles en Flutter, estando, por tanto, disponible para dispositivos móviles con sistema operativo Android e iOS.

Además, se le ha dotado a la aplicación de una funcionalidad para que pueda comunicarse únicamente con una aplicación para dispositivos móviles, ya que una de las carencias previas de la aplicación era que la comunicación con el dispositivo móvil no era correcta debido a que el envío de información era hacia todas las aplicaciones abiertas en ese momento y no hacia una única aplicación.

Por último, también se le ha dotado a la aplicación de persistencia de datos entre sesiones debido a que era una de las carencias de la anterior aplicación.



# Índice

1	Introducción	8
1.1	Aprendamos algoritmos	8
1.2	Motivación.	10
1.3	Objetivos	10
2	Fundamentos	12
2.1	Creación automática de código.	12
2.1.1	Node.js	12
2.1.2	Fs-extra	13
2.1.3	WebStorm	13
2.2	Aplicación móvil desarrollada en Flutter	13
2.2.1	Flutter	13
2.2.2	Dart	14
2.2.3	Android Studio	15
2.3	Aplicación Alexa	15
2.3.1	Amazon Alexa	15
2.3.2	Skills	16
2.3.4	Intenciones	17
2.3.5	Alexa-Hosted Skill	17
2.3.6	Amazon S3	18
2.4	Git	18
3	Descripción informática	19
3.1	Metodología de trabajo	19
3.2	Análisis	20
3.2.1	¿Por qué Flutter?	20
3.2.2	¿Por qué una aplicación que cargue las fotos de internet y no las tenga alojadas?	21
3.2.3	Comunicación entre aplicaciones	22
3.2.4	Porque una librería de creación automática de código	23
3.3	Diseño	23
3.3.1	Flujo de ejecución	23
3.3.1.1	Aplicación para dispositivos móviles	23
3.3.1.2	Aplicación Alexa	26
3.4	Implementación	31
3.4.1	Generación código automático	31
3.4.2	Aplicación Flutter	35
3.4.3	Aplicación Alexa	36
3.5	Pruebas	38
3.5.1	Pruebas manuales.	38
3.5.1.1	Pruebas manuales en el generador de código	38

3.5.1.2 Pruebas manuales aplicación Alexa	40
3.5.2 Pruebas con usuarios	42
4 Conclusiones	45
4.1 Conclusiones generales.	45
4.2 Conclusiones personales	47
4.3 Trabajos futuros	48
5 Bibliografía	49

## Tabla de ilustraciones

Ilustración 1 Invocación de una skill	17
Ilustración 2. Fases del desarrollo en cascada	20
Ilustración 3. Flujo de ejecución aplicación móvil	24
Ilustración 4. Pantalla 1 aplicación móvil	24
Ilustración 5. Pantalla 2 aplicación móvil	25
Ilustración 6. Pantalla 3 aplicación móvil	25
Ilustración 7. Pantalla 4 aplicación móvil	26
Ilustración 8. Flujo de ejecución aplicación Alexa	27
Ilustración 9. Pantalla 5 aplicación Alexa	29
Ilustración 10. Pantalla 8 aplicación Alexa	30
Ilustración 11. Pantalla 13 aplicación Alexa	30
Ilustración 12. Pantalla 14 aplicación Alexa	31

# 1 Introducción

En este primer apartado de introducción, se busca que el lector tenga una idea clara del porqué se ha realizado el proyecto descrito en estas páginas y qué se busca con la realización del mismo. La introducción comienza con una explicación del estado actual de la aplicación *Aprendamos Algoritmos*, que será la aplicación de la que parta este proyecto y cuya ampliación será lo expuesto en este documento. A continuación, habrá un apartado donde se describirá cuál ha sido la motivación para la realización del proyecto, así como otro apartado donde se explicará el objetivo que se busca con su realización.

## 1.1 Aprendamos algoritmos

La aplicación llamada *Aprendamos Algoritmos*, es la aplicación sobre la que parte este proyecto. Esta aplicación consiste en un desarrollo realizado para ser ejecutado por el asistente virtual Alexa con capacidad para comunicarse con una aplicación para dispositivos móviles Android y cuyo objetivo es enseñar al usuario a resolver el problema del algoritmo voraz Dijkstra.

Esta aplicación, antes de la realización del proyecto descrito en estas páginas, disponía de:

- Un apartado en el cual, mediante un grafo de ejemplo, se le explica al usuario cómo se resuelve el problema. El usuario únicamente tiene que ir navegando por la explicación diciendo “siguiente” cada vez que desee continuar con dicha explicación.
- Un apartado con un grafo a modo de prueba de comprensión del algoritmo. En este apartado, hay un único grafo y la aplicación va realizando una serie de preguntas a las que el usuario debe ir respondiendo. En el caso de que el usuario responda correctamente, la aplicación le responde con un mensaje de felicitación y le permite continuar en su progreso. En caso de respuesta errónea, la aplicación le responderá con un mensaje de ayuda y volviendo a realizarle la pregunta de nuevo sin permitirle continuar con el progreso hasta que la respuesta sea correcta.
- Un apartado donde se muestra el algoritmo voraz Dijkstra en pseudocódigo para que así el usuario pueda apoyarse en él para la comprensión y resolución del problema propuesto.
- La aplicación muestra en todo momento el estado de resolución del problema mediante una tabla con el estado de los valores aportados por el usuario tanto para el apartado de explicación como para el apartado de resolución del problema.



- La aplicación se apoya en una aplicación para dispositivos móviles con sistema operativo Android para poder mostrar imágenes y de esta forma facilitar la comprensión del problema propuesto por parte del usuario.

La comunicación de la aplicación de Alexa con la aplicación Android, se realiza mediante un webSocket desplegado en Amazon Web Service. La comunicación consiste en:

- La aplicación de Alexa se conecta al webSocket y le envía un mensaje dependiendo de la pantalla en la que se encuentre el usuario.
- La aplicación Android en el momento de su inicio, se conecta al mismo webSocket que la aplicación de Alexa y se mantiene a la espera de recibir un mensaje. Cuando le llega un mensaje, lo compara con una lista de posibles pantallas que espera recibir y muestra las imágenes necesarias en cada momento.
- El webSocket es un API Gateway desplegado en Amazon Web Service. Dicho API dispone de tres rutas:
  - Connect. Será llamada cuando se realice una conexión con el webSocket. Esta ruta activará una función desplegada en AWS Lambda, la cual almacenará en una base de datos, desarrollada en AWS DynamoDB, el identificador del cliente que ha realizado la llamada al API.
  - Disconnect. Será llamada cuando se realice la desconexión con el webSocket. Esta ruta activa una función Lambda la cual elimina de la base de datos el identificador del cliente.
  - SendMessage. Será llamada cuando la skill desee comunicarse con la aplicación y activará una función Lambda capaz de acceder a la base de datos, obtener todos los clientes conectados y enviarle a estos el mensaje recibido.

Todo el código de la aplicación utilizada por el asistente virtual Alexa está desarrollado en NodeJS, así como el código de las funciones Lambda utilizadas para la conexión entre aplicaciones.

La aplicación Android está desarrollada en el lenguaje Kotlin. Según el mensaje recibido por el webSocket muestra una pantalla u otra. Todas las pantallas están definidas y cada una de ellas dispone de una serie de imágenes almacenadas dentro de la aplicación. El usuario no puede de ningún modo interactuar con la aplicación ya que esta únicamente muestra la pantalla indicada en cada momento por la aplicación de Alexa.

## 1.2 Motivación.

La aplicación como estaba hasta el comienzo del presente proyecto no podía ser desplegada en las tiendas de aplicaciones debido a que la comunicación de la aplicación utilizada por Alexa no era capaz de identificar con que aplicación Android debía conectarse, por lo que si hubiese sido desplegada, habría provocado que un único usuario utilizando la aplicación desde Alexa controlase todas las aplicaciones Android abiertas en ese momento.

Además de lo descrito anteriormente, únicamente había un grafo con el que poner a prueba al usuario, por lo que no sería una aplicación con un gran uso ya que una vez resuelta, la aplicación carecería de sentido para el usuario que lo hubiese resuelto. Esta carencia se podría resolver con actualizaciones de código, pero de la forma en la que se encontraba el código resultaba muy complicado la escalabilidad en cuanto a la introducción de nuevos problemas.

Cuando un usuario cerraba la aplicación, esta perdía todo el progreso llevado por el usuario, por lo que su uso podría hacerse tedioso ya que de cara al usuario tener que repetir lo que ya ha resuelto no resulta agradable.

Otro de los motivos por lo que se ha decidido realizar este proyecto es debido a que los usuarios con sistema operativo iOS no disponían de aplicación para sus dispositivos móviles, por lo que el uso de la aplicación estaba destinado a los usuarios con sistema operativo Android, limitando así el público objetivo de la aplicación.

Debido a todas estas carencias surgidas en la aplicación aprendamos algoritmos, se decide continuar con la aplicación resolviendo las carencias encontradas y haciendo más usable, más práctica y poniéndola a disposición de un mayor número de personas.

## 1.3 Objetivos

El objetivo principal de este proyecto es resolver las carencias presentadas por la aplicación *Aprendamos Algoritmos* para que pueda ser desplegada en las tiendas de aplicaciones y sea usada por el mayor número de personas posible. Para ello debe cumplir una serie de requisitos que son:

- La aplicación ha de contener varios problemas a resolver por parte del usuario. Además, estos problemas deben tener diferente nivel de dificultad pudiendo el usuario elegir qué nivel de dificultad desea resolver. Una vez que el usuario elija un nivel, la aplicación ha de responder con un problema elegido aleatoriamente dentro del nivel de dificultad elegido por el usuario.

Con este objetivo se busca que la aplicación ofrezca al usuario una gran variedad de posibilidades para ponerse a prueba.

- El desarrollador de la aplicación debe ser capaz de crear problemas de una manera sencilla permitiendo de este modo la escalabilidad de la aplicación. Mediante este objetivo se busca llegar al punto en el que el desarrollador pueda añadir problemas a la aplicación sin un gran esfuerzo de desarrollo.
- La aplicación de Alexa es capaz de identificar con qué dispositivo móvil tiene que conectarse y de esta manera enviarle la información necesaria únicamente a él. De esta manera conseguimos una usabilidad completa y una comunicación directa uno a uno entre aplicación de Alexa y aplicación móvil.
- Poner a disposición de los usuarios la aplicación para dispositivos móviles con sistema operativo iOS. Una de las carencias de la aplicación está en que únicamente puede ser utilizada por usuarios cuyo dispositivo móvil es Android. Con este objetivo se busca ampliar el rango de usuarios que puedan utilizar la aplicación.
- La aplicación para dispositivos móviles ha de ser capaz de cargar las imágenes mediante urls y mostrarlas para que de esta manera no sea necesario su almacenamiento dentro de la propia aplicación. Con este objetivo se busca reducir el tamaño de la aplicación y facilitar su escalabilidad.
- Mantener persistencia de datos entre sesiones. Con este objetivo se logra que cuando un usuario abandone la aplicación, esta guarda su progreso una actividad posterior en la aplicación por parte del usuario.

## 2 Fundamentos

En este apartado se hablará acerca de las tecnologías, herramientas y lenguajes de programación utilizados para la realización de las distintas partes del proyecto. Se hablará sobre qué son y porqué han sido seleccionadas cada una de ellas.

### 2.1 Creación automática de código.

Como se detallará en el apartado de implementación, para la realización de este proyecto, se ha creado una librería para la generación automática de código. Las tecnologías usadas en la creación de esta librería se exponen a continuación.

#### 2.1.1 Node.js

Javascript es un lenguaje de programación que se ejecuta en el lado del cliente, es decir, es un navegador web quien interpreta el código. Para poder ser ejecutado en el lado del servidor, es necesario un entorno en tiempo de ejecución capaz de interpretar el código y ejecutarlo, es de esta manera como nace Node.js. Node.js es una máquina virtual que utiliza un modelo asíncrono y dirigido por eventos, capaz de transformar el código creado originalmente para ser ejecutado en navegadores, en código capaz de ser ejecutado en un ordenador como si de una aplicación independiente se tratara. En comparación con las técnicas tradicionales de servicio web donde cada conexión (creadora de una solicitud) genera un nuevo subproceso, ocupando la RAM del sistema y regularmente maximizando la cantidad de RAM disponible, Node.js opera en un solo subproceso, utilizando el modelo entrada y salida sin bloqueo de la salida, lo que le permite soportar decenas de miles de conexiones al mismo tiempo mantenidas en el bucle de eventos. Cuando hay una nueva solicitud, se genera un tipo de evento. El servidor empieza a procesarlo y, cuando hay una operación de bloqueo de entrada y salida, no espera hasta que se complete, sino que en su lugar crea una función de devolución de la llamada. El servidor comienza en el acto a procesar otro evento y cuando finaliza la operación de entrada y salida, continúa trabajando en la solicitud ejecutando la devolución de llamada tan pronto como sea capaz. Por tanto, el servidor nunca necesita crear más subprocesos o cambiar entre subprocesos, lo que significa que tiene muy poca sobrecarga. Por tanto, Node es capaz de manejar una gran cantidad de conexiones simultáneas con un alto nivel de rendimiento, lo que equivale a una alta escalabilidad. Gracias al alto nivel de rendimiento y la escalabilidad ofrecida por Node.js será el entorno de ejecución sobre el que se desarrollará la generación de código automático que se expondrá en posteriores apartados.

### 2.1.2 Fs-extra

Node.js pone a disposición por defecto, una serie de librerías con diversa funcionalidad para ayudar a los desarrolladores en la creación de sus scripts. Una de estas librerías es File-System. Esta librería disponibiliza las funciones relacionadas con la entrada/salida de ficheros, por lo que será necesaria en el código desarrollado en este proyecto para la lectura y creación de ficheros y directorios. File-System tiene una serie de carencias como por ejemplo la lectura y escritura de ficheros json, por lo que para la realización de este proyecto se utilizará la librería “fs-extra”, librería que añade una serie de funciones a la librería fs (File-System), puesto que una de estas funciones añadidas es la de creación y lectura de ficheros json.

### 2.1.3 WebStorm

Webstorm es un entorno de desarrollo integrado (IDE) especialmente diseñado para Javascript por la empresa JetBrains. Al ser este su propósito, no se necesitará configurar ningún complemento adicional para poder empezar a utilizarlo para el desarrollo de la skill. Webstorm ofrece la detección de errores mientras se escribe código de programación, permite depurar, lanzar test unitarios con las principales librerías de los mismos, permite una refactorización del código con el mínimo esfuerzo posible, y además, ofrece una integración con git sin necesidad de una configuración adicional, por lo que es el IDE más potente que se puede utilizar para realizar la librería de generación automática de código.

Su licencia tiene un coste de 129€ al año, pero dispone de una licencia gratuita para estudiantes que será la utilizada para la de este proyecto.

## 2.2 Aplicación móvil desarrollada en Flutter

Para la realización del proyecto descrito en estas páginas, se ha realizado una aplicación en Flutter, la cual se detallará en el apartado de implementación. En los siguientes apartados se expondrá las tecnologías y herramientas utilizadas para su desarrollo.

### 2.2.1 Flutter

Flutter es un SDK desarrollado por Google para el desarrollo de aplicaciones móviles nativas Android e iOS, y en fase beta para aplicaciones web. Flutter está basada en una librería con tecnología basada en cross-platform, las cuales al compilar producen apps completamente nativas sin intervención alguna de navegadores web para su ejecución. Flutter es capaz, por tanto, de construir aplicaciones nativas con un alto nivel de rendimiento y fiabilidad para múltiples plataformas a partir de la misma base de código.

La principal idea de Flutter es la utilización de componentes nativos llamados widget, ya que toda la interfaz de usuario se compone gracias a la combinación de diferentes widget, definiendo cada uno de ellos un elemento de la estructura de la aplicación (como puede ser un botón o un menú) un elemento estilístico, (como puede ser la fuente o el color utilizado en un determinado texto) o un aspecto del diseño, (como puede ser el fondo de la pantalla mostrada). Flutter utiliza sus propios widgets, los cuales simulan un aspecto nativo de las aplicaciones Android (Material Design) o iOS (Cupertino).

Ventajas de usar Flutter:

- Creación de un único código para la aplicación de Android e iOS, por lo que se ahorrará gran cantidad de tiempo tanto de desarrollo como de pruebas para la creación de la aplicación para dispositivos móviles
- Es de código abierto y gratuito, por lo se puede acceder a todas las librerías necesarias para la creación de este proyecto, como por ejemplo la librería para la creación del webSocket que servirá como comunicador con la app de Alexa.
- Existe una gran abstracción de compatibilidades con las distintas versiones de los sistemas operativos ya que los widget son disponibilizados y probados por Flutter, por lo que se gana mucho tiempo en pruebas de compatibilidad con los distintos dispositivos móviles.

### 2.2.2 Dart

Dart es el lenguaje de desarrollo utilizado por Flutter. Es un lenguaje de código abierto desarrollado por Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y con análisis estático de tipo. Dart se diseñó con el objetivo de hacer el proceso de desarrollo lo más cómodo y rápido posible para los desarrolladores, por eso, viene con un conjunto bastante extenso de herramientas integrado, como su propio gestor de paquetes, varios compiladores/transpiladores, un analizador y un formateador de código. Una de las principales ventajas de Dart y por la que se diferencia a Flutter es que la compilación de Dart produce instrucciones nativas muy afinadas para los chips arm64 de los dispositivos móviles, teniendo así una mejora en cuanto a rendimiento y velocidad con respecto a la compilación de JavaScript utilizada por la otra librería que también compila aplicaciones nativas como es React-native.

Dart dispone de una curva de aprendizaje muy rápida ya que es un lenguaje, en cuanto a sintaxis, muy parecido a JavaScript pero tipado y con orientación a objetos, por lo que si el desarrollador tiene conocimientos de Java será muy rápido su aprendizaje.

### 2.2.3 Android Studio

Android estudio es un entorno de desarrollo integrado (IDE) diseñado para el desarrollo de apps para Android, basado en los IDEs de JetBrains (como WebStorm, visto en el apartado 2.1.3). Además de las funcionalidades descritas en Webstorm, Android Studio tiene una serie de funcionalidades específicas para el desarrollo de apps como son:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde poder desarrollar para todos los dispositivos Android.
- Capacidad de inserción de cambios en el código sin necesidad de reinicio en la aplicación.
- Integración con GitHub para poder importar código de muestra.
- Variedad de marcos de trabajo y herramientas para probar el código.

Android Studio posee la capacidad de importar complementos de diferentes lenguajes y librerías. Dos de estos complementos, los cuales se usarán para la realización de la aplicación Flutter, son Flutter y Dart. Gracias a estos complementos, desde Android Studio se puede desarrollar una aplicación Flutter utilizando el lenguaje Dart. Como Android Studio está diseñado para la simulación de distintos dispositivos móviles y tiene funcionalidades específicas para el desarrollo de aplicaciones, es el IDE utilizado para el desarrollo de la aplicación Flutter.

## 2.3 Aplicación Alexa

La aplicación para el asistente virtual Alexa es la base del proyecto, es la aplicación sobre la que se basa el proyecto, la encargada de enviar información a la aplicación Flutter y por la que se realiza la librería de generación automática de código. Su código está implementado en NodeJS descrito en el apartado 2.1.1, por lo que en el siguiente apartado veremos qué es Alexa, cómo son sus aplicaciones y de que están formadas las mismas.

### 2.3.1 Amazon Alexa

Alexa es un asistente virtual controlado por voz y lanzado al mercado por Amazon en noviembre de 2014 integrado en su línea de altavoces inteligentes Amazon Echo. Actualmente, Alexa está disponible en español, inglés, francés, alemán, italiano, japonés y portugués.

Alexa, está disponible en una gran cantidad de dispositivos como puede ser desde una nevera hasta un coche. Cualquier compañía puede incorporar Alexa a sus dispositivos gracias a la herramienta Alexa Voice Service. No entraremos en detalle, ya que en el contexto de estas páginas utilizaremos un Amazon Echo con Alexa ya integrado.

Para comenzar a conversar con Alexa, un usuario tiene que decir su nombre en voz alta cerca de un dispositivo donde esté configurada y se activará. A continuación, el usuario debe decir lo que se necesite saber o hacer y Alexa realizará las operaciones necesarias y le contestará.

Alexa trae integradas ciertas funcionalidades desarrolladas por Amazon como son, conocer el estado del tiempo, poner una alarma, hacer llamadas entre dispositivos Amazon Echo, conectarse con otros dispositivos inteligentes de la casa como pueden ser bombillas o cámaras de vigilancia... Además, Alexa permite aplicaciones de terceros, llamadas skills, y aquí es donde entra el desarrollo descrito.

### 2.3.2 Skills

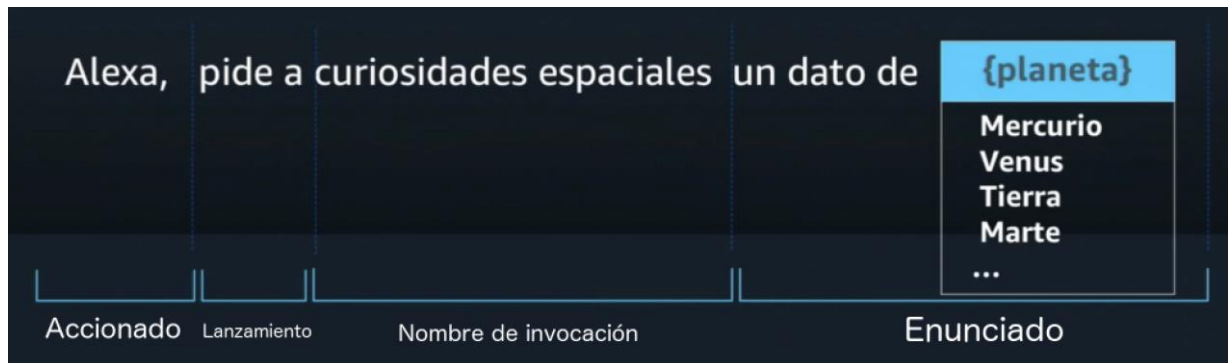
Una skill es una aplicación capaz de añadirle funcionalidad extra a cualquier dispositivo que tenga integrado Alexa. Esta conexión se realiza a través de la herramienta Alexa Skills Kit. Las skills se apoyan en un servicio de voz de Alexa que vive alojado en la nube y desarrollado por Amazon. En dicho servicio se realizan dos etapas de machine learning cuando los usuarios interactúan con una skill. Una primera etapa llamada “Reconocimiento de voz automatizado” donde transforma en texto lo recibido por audio por parte del usuario, y una segunda etapa, llamada “Comprensión del lenguaje natural” donde se le asigna un significado a la frase.

Para invocar una skill:

- Lo primero debe ser la palabra de accionado del dispositivo. Por defecto es Alexa, pero puede ser configurada y tener los nombres de Echo o Amazon. Esta palabra hará que el dispositivo “despierte” y comience a procesar todo lo que el usuario le indica a continuación.
- Seguido, viene la palabra de lanzamiento. Puede tener diferentes valores, como por ejemplo, lanza, abre, inicia, empieza, comienza, pregunta... Esta palabra es necesaria para que el dispositivo utilizado por Alexa conozca que un usuario va a llamar a una skill.
- Después, viene el nombre de invocación. Este será el nombre por el que los usuarios llamarán a una skill determinada y provocará que dicha skill comience su procesamiento.
- De manera adicional, en el momento de invocar una skill, se le puede pasar una frase. Dicha frase se llamará enunciado. Este enunciado será pasado a la skill invocada, que lo procesará y convertirá en una intención. Los enunciados, además pueden llevar variables, que son elementos que pueden tener diferentes valores, pero que la skill tratará y transformará en una



misma intención. Estas variables pueden tener un tipo predefinido, como pueden ser números, países, colores... o puedes crearte tu propio tipo de variable.



*Ilustración 1 Invokación de una skill*

#### 2.3.4 Intenciones

Una intención es una acción que cumple con una petición realizada por un usuario. Cada intención puede ser activada por uno o más enunciados. Las intenciones pueden ser creadas por el desarrollador de la skill, aunque, tiene que haber una serie de interacciones por defecto que tienen que comportarse de la misma manera para todas las skill para que se comporten de la misma manera. Si no cumples con estas intenciones por defecto, no pasarás el proceso de validación de la skill. Estas interacciones por defecto son:

- AMAZON.CancelIntent. Se acciona con los enunciados “cancela”, “olvidalo”... Sirve para parar la acción que se esté realizando dentro de la skill sin cerrarla.
- AMAZON.StopIntent. Se acciona con los enunciados “para”, “apagar”... Sirve para parar la acción que está realizando la skill y salir de ella.
- AMAZON.HelpIntent. Se acciona con los enunciados “ayuda”, “ayúdame”... Sirve para que la skill responda con el mensaje de ayuda que tenga configurado.

#### 2.3.5 Alexa-Hosted Skill

Hasta ahora se ha descrito la interfaz de voz de usuario, que es lo que la skill tiene que comprender cuando un usuario habla con ella. A esta parte de la skill se le llama Front-end. Pero esto no es únicamente lo que forma una skill, sino que tiene que haber un script de programación para que la skill pueda realizar las operaciones que se deseen. A esto se le llama back-end. El back-end es un script de programación que se ejecuta en un servidor y realiza una serie de acciones. A la hora de realizar un back-end para una skill puede alojarse en cualquier servidor que se desee, lo único que se debe tener en cuenta es que el servidor seleccionado permita peticiones HTTPS. Lo más popular a la hora de desarrollar una skill es tener el back-end en una Lambda.

Amazon, además, provee la herramienta Alexa-Hosted skill. Esta herramienta aprovisiona y administra automáticamente el conjunto de servicios de AWS que necesita la skill, como por ejemplo las lambdas necesarias. De esta forma, desde la consola de desarrollador de Alexa se puede construir, editar y publicar una skill sin tener que preocuparse por configurar el back-end. Alexa-Hosted skill será la herramienta que se utilizará para crear la skill, ya que, como se ha hablado anteriormente, abstrae de tener que configurar las lambdas para el back-end de la skill.

### 2.3.6 Amazon S3

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de datos diseñado para facilitar a los desarrolladores recursos de computación escalables, y con una alta disponibilidad seguridad y un alto rendimiento. Amazon S3 tiene una interfaz de servicios web simple que puede utilizar para almacenar y recuperar cualquier cantidad de datos, en cualquier momento, desde cualquier parte de la web. Puede ser utilizado para almacenar sitios web estáticos, realizar copia de seguridad de archivos, alojar imágenes...

Para el almacenamiento de datos en Amazon S3 se utilizan “Buckets”, que es un recurso de almacenamiento en la nube, similar a una carpeta de archivos, en donde se permite el almacenado de datos través de Amazon S3.

Para el presente proyecto se utilizará un Bucket creado y utilizado únicamente por la aplicación de Alexa para el almacenamiento de datos de los usuarios de la misma.

## 2.4 Git

Git es un software de código abierto de control de versiones y pensado para la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Git, provee de muchas herramientas para hacer posible esto. Para este proyecto, se utilizará:

- Creación de ramas por funcionalidad. De esta manera se puede ver de una forma rápida el desarrollo de cada nueva características
- Realizar commits por cada pequeño desarrollo creado para así tener un control y poder volver atrás en caso de error

Git será utilizado para los desarrollos de todas las funcionalidades descritas en este proyecto. Cada proyecto será subido a un repositorio distinto en donde se podrá consultar todo el código desarrollado, así como el histórico de cada uno de los commits.

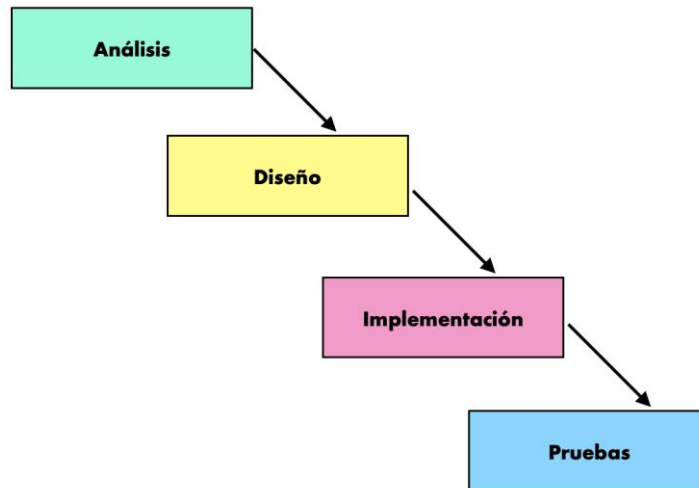
## 3 Descripción informática

### 3.1 Metodología de trabajo

Este proyecto parte de un proyecto anterior gracias al cual se han aprendido ciertos métodos de trabajo y sobre todo se han creado unos objetivos muy concretos para el desarrollo del proyecto descrito en estas páginas. Al no haber tanta incertidumbre, partir de un proyecto ya creado y tener unos objetivos muy claros, la metodología de trabajo elegida para este proyecto es la metodología en cascada, consistente en la realización del desarrollo del proyecto de forma secuencial en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra, es decir, hasta que no se completa una fase de la metodología no puede comenzar la siguiente. Este método requiere un alto detalle de los requisitos debido a que existe una única iteración del camino a recorrer por las fases del desarrollo.

Para este proyecto se ha utilizado la metodología de trabajo en cascada consistente en 4 fases:

- **Análisis.** En esta primera fase se analiza todo lo necesario para la realización del proyecto como puede ser la tecnología utilizada para la creación de la aplicación móvil, cómo va a ser el desarrollo de la aplicación móvil, cómo va a ser el envío de mensajes entre las dos aplicaciones y por qué se decide crear una librería generadora de código. Todas estas decisiones son tomadas para satisfacer los objetivos marcados con el proyecto. Esta fase está detalladamente descrita en el apartado 3.2.
- **Diseño.** En esta segunda fase del proceso de desarrollo se describe cómo se van a establecer la relación entre las distintas pantallas desarrolladas y cómo cada una de ella debe responder a las acciones llevadas a cabo por el usuario. En ella se establece el flujo de ejecución de las dos aplicaciones y ejemplos de su interfaz. Esta fase está detalladamente definida en el apartado 3.3
- **Implementación.** En la fase de implementación se desarrollan los requisitos especificados en la fase anterior siguiendo las conclusiones sacadas en el análisis. Los detalles de la implementación están descritos en el apartado 3.4
- **Pruebas.** En esta fase se verifica que todos los componentes desarrollados en la fase de implementación funcionan correctamente de acuerdo a los requisitos especificados en el diseño. Para esta fase se realizarán pruebas manuales por parte del desarrollador y una serie de pruebas por parte de usuarios externos al proyecto. El detalle de estas pruebas puede verse en el apartado 3.5



*Ilustración 2. Fases del desarrollo en cascada*

## 3.2 Análisis

Para la realización del análisis de los requisitos del proyecto se tomaron en cuenta los objetivos marcados al comienzo del mismo. Con la búsqueda de la resolución de estos objetivos se realizó un análisis descrito en los siguientes apartados

### 3.2.1 ¿Por qué Flutter?

Uno de los objetivos de este proyecto es disponibilizar la aplicación móvil para los sistemas operativos de Android e iOS llegando de esta manera a un mayor número de usuarios ya que el proyecto anterior únicamente disponibilizaba la aplicación para dispositivos móviles con sistema operativo Android. El anterior proyecto tenía una aplicación móvil realizada en el lenguaje de programación Kotlin para Android. Debido a que estaba construida en Kotlin únicamente podía ser desplegada para Android, por lo que había que realizar una nueva aplicación para iOS. En el análisis de esta nueva aplicación se descartó realizar una nueva aplicación exclusivamente para iOS ya que habría que mantener 2 códigos distintos, cada uno para un sistema operativo específico. Además, la aplicación Android, como se verá en posteriores apartados, habrá que rehacer todo su código por lo que habría que realizar dos códigos totalmente distintos en cuanto a su lenguaje de programación se refiere, pero cuyo propósito es el mismo.

Una vez descartada la generación de código específico por sistema operativo, se analizan las herramientas disponibles de generación de código multiplataforma, el cual nos permite con un mismo código poder desplegar en ambos sistemas operativos. Sobre qué librería para el desarrollo de código multiplataforma es mejor, existe un fuerte debate y no hay un consenso único, si no que cada

herramienta es mejor según qué tipo de aplicación se vaya a realizar. Después de realizar un análisis de las distintas herramientas ofrecidas por el mercado se ha decidido utilizar Flutter debido a que:

- Tiene un conjunto muy completo de Widgets para la interfaz de usuario los cuales se compilan directamente a nativo proporcionando así un alto rendimiento en la aplicación. Para la aplicación se utilizarán un componente para la introducción de texto, otro para el mostrado de imágenes, otro para el cuadro de alerta y una librería para la conexión con el webSocket.
- Dispone de una documentación muy completa además de una guía de instalación muy sencilla, por lo que para una aplicación sencilla como será la desarrollada se dispone de toda la documentación necesaria.
- Dispone de un lenguaje novedoso de programación, suponiendo así un nuevo reto para el proyecto dándole de esta manera perspectivas de futuro.
- Cada día la comunidad de Flutter es más grande siendo así una grandísima herramienta de futuro.

### 3.2.2 ¿Por qué una aplicación que cargue las fotos de internet y no las tenga alojadas?

La aplicación Android con la que contaba el proyecto en sus inicios contenía una serie de carencias con respecto al mostrado de información al usuario. Algunas de estas carencias eran:

- Generación de una pantalla por cada paso de cada problema. El desarrollador de la aplicación debía generar por cada paso de cada problema que contuviese la aplicación, una pantalla con un contenido muy similar al resto de pantallas ya que únicamente cambiaba las imágenes a mostrar. Debido a esto se hacía muy tediosa la implementación de nuevos problemas ya que requería desarrollar un mínimo de 12-15 pantallas dependiendo de los nodos que contuviese el problema.
- Gran tamaño de aplicación debido al almacenamiento de todas las imágenes que iban a ser mostradas. Esto hacía que la aplicación fuese adquiriendo cada vez más tamaño a medida que se iban agregando nuevos problemas
- Dificultad en la comunicación con la aplicación de Alexa para la selección de la pantalla a mostrar. Anteriormente, a través del webSocket, la aplicación de Alexa enviaba un código el cual la aplicación recibía y dependiendo del código que le llegase, mostraba una imagen u otra. Esto hacía que el desarrollador de la aplicación móvil tuviese que conocer el código que había decidido el desarrollador de la aplicación de Alexa que correspondía con una pantalla determinada a mostrar.
- Constante actualización por parte del usuario de la aplicación. Debido a que había que realizar cambios cuando se introducía un nuevo problema, una vez estuviese subida la aplicación a la

tienda de aplicaciones de Android, el usuario necesitaría estar actualizando la aplicación con cada nuevo cambio.

Debido a todas estas carencias, en el análisis realizado, se ha decidido crear una aplicación totalmente nueva en Flutter, ya que se ha decidido no alojar las imágenes dentro de la aplicación móvil, si no que será la aplicación de Alexa a través del webSocket quien envíe la información de las imágenes que mostrará la aplicación móvil. Gracias a esto, cuando se implementa un nuevo problema no será necesario el desarrollo de nuevas pantallas, ya que únicamente se implementará una que muestre las imágenes recibidas, ganando así tiempo de desarrollo y mitigando de esta manera la carencia que tenía la aplicación anterior en cuanto a la actualización continua por parte del usuario, ya que, con este nuevo método, no será necesario hacer ningún cambio en la aplicación y por lo tanto no será necesaria ninguna actualización por parte del usuario. Además, como ya no tendrá almacenadas imágenes, la aplicación tendrá siempre el mismo tamaño siendo este muy bajo debido a la baja cantidad de pantallas necesarias.

### 3.2.3 Comunicación entre aplicaciones

Uno de los puntos más importantes a tener en cuenta en el análisis de requisitos es la comunicación que se ha de realizar entre la aplicación para dispositivos móviles y la aplicación de Alexa. Anterior a la realización de este proyecto, esta comunicación se hacía a través de un webSocket. La aplicación de Alexa era la encargada de enviar un mensaje con un código a la aplicación móvil. Para este proyecto se ha analizado esta comunicación y se ha decidido mantenerla cambiando el mensaje enviado.

Uno de los objetivos es poder comunicar correctamente ambas aplicaciones. Para ello, se ha decidido que el usuario introduzca la misma clave en ambas aplicaciones, de esta manera, la aplicación que reciba el mensaje, comprobará la clave recibida y la comparará con la introducida por el usuario en su aplicación para saber de esta manera si es el mensaje que espera recibir o no. Por tanto, debido a esto, se han de crear en ambas aplicaciones la opción de que el usuario pueda introducir una clave. Además, el encargado de enviar el mensaje, la aplicación de Alexa, tiene que enviar en el cuerpo del mensaje la clave introducida por el usuario.

Como se ha visto en el apartado anterior, para este proyecto, la aplicación móvil espera recibir en el mensaje llegado mediante el webSocket qué imágenes necesita mostrar, por lo que ahora en el mensaje enviado desde la aplicación de Alexa tienen que ser enviadas dichas direcciones donde se encuentran almacenadas las imágenes correspondientes a cada pantalla que desee ser mostrada.

Por tanto, el mensaje enviado a través del WebSocket ahora contendrá una clave y urls donde se encuentran las imágenes como número de imágenes que deseen ser mostradas.

### 3.2.4 Porque una librería de creación automática de código

Uno de los principales objetivos de este proyecto es poder dotar a la aplicación de una gran cantidad de problemas para que, de esta forma, el usuario de la aplicación aprenda a resolver el algoritmo correctamente poniéndose a prueba con diferentes problemas siendo estos de diferentes niveles de dificultad. La aplicación, al comienzo de este proyecto, para poder agregar una nueva pantalla era necesario desarrollar un fichero con el código entendible por la aplicación (manejador), un fichero donde se encuentra el texto a expresar por parte de la aplicación Alexa y dos ficheros para el mostrado de imágenes por pantalla, por lo que añadir un nuevo problema consistía en la agregación de unos 60-70 ficheros. Este hecho, hacía a la aplicación muy compleja en cuanto a su escalabilidad. Para ello se ha propuesto la creación de una librería, la cual recibe un fichero Json con una serie de parámetros y genera automáticamente todos los ficheros necesarios para la agregación de un nuevo problema en la aplicación de Alexa.

## 3.3 Diseño

En los siguientes apartados se detallará el recorrido que un usuario puede realizar dentro de la aplicación y una muestra de algunas pantallas tanto para la skill de Alexa como para la aplicación móvil.

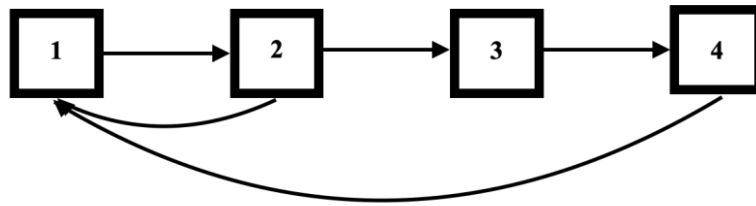
### 3.3.1 Flujo de ejecución

El flujo de ejecución de las aplicaciones va a estar muy marcado por los objetivos a cumplir en el presente proyecto. El anterior proyecto tenía un flujo de ejecución único ya que la aplicación Android se caracterizaba por mostrar imágenes dependiendo de qué orden le llegaba desde la aplicación de Alexa. Para este nuevo proyecto, el usuario si debe interactuar con la aplicación móvil, por lo que al comienzo tiene un flujo de ejecución distinto a la aplicación de Alexa. En las siguientes líneas se describen ambos flujos de ejecución

#### 3.3.1.1 Aplicación para dispositivos móviles

Para la aplicación móvil, el usuario tiene que introducir una contraseña para así poder comunicarse con la aplicación Alexa, por lo que la primera pantalla consistirá en la introducción y confirmación por parte del usuario de la contraseña introducida. Una vez que el usuario ha confirmado la contraseña, se mostrará una pantalla a modo de presentación en la que el usuario ya no podrá realizar más acciones, ya que la aplicación se mantendrá desde ese momento a la espera de recibir mensajes con las imágenes a mostrar. El usuario una vez introducida la contraseña, únicamente puede volver a

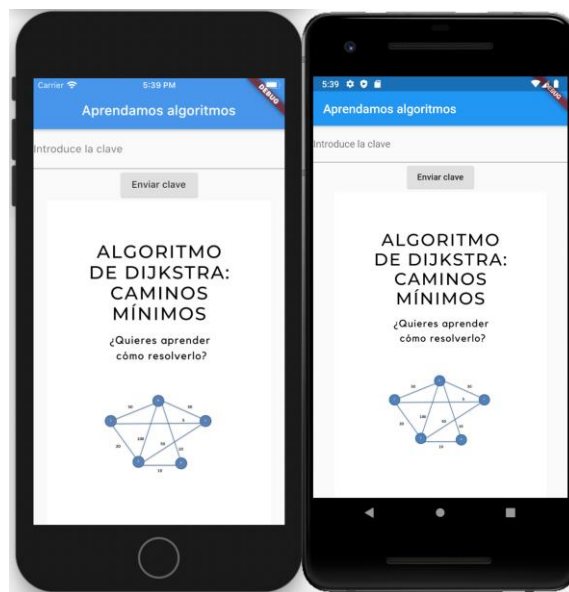
la página de inicio por si desease introducir una nueva contraseña. Por tanto el flujo de ejecución para la aplicación móvil quedaría como en la ilustración número 3.



*Ilustración 3. Flujo de ejecución aplicación móvil*

A continuación, se describen las pantallas que componen el flujo de ejecución de la aplicación móvil junto con su interfaz gráfica.

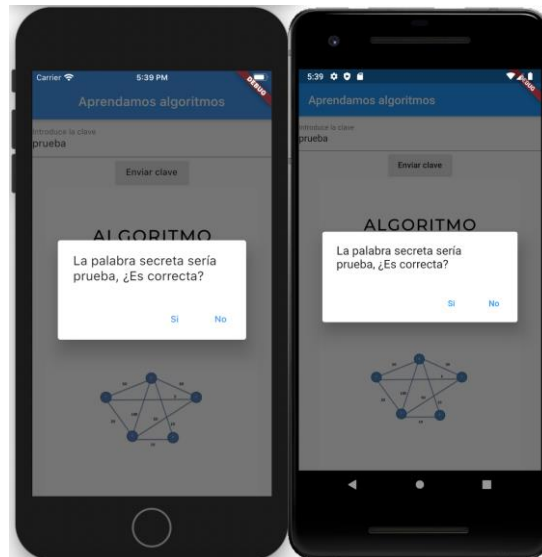
- Pantalla 1. Es la pantalla de inicio de la aplicación, en ella el usuario puede introducir la contraseña para la comunicación con la aplicación de Alexa. Su interfaz queda como puede observarse en la ilustración número 4:



*Ilustración 4. Pantalla 1 aplicación móvil*

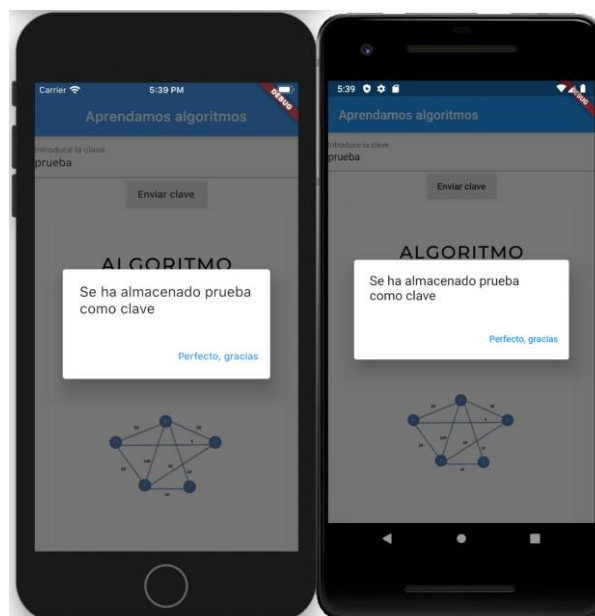


- Pantalla 2. Esta pantalla muestra un cuadro de texto en el que se le indica al usuario qué contraseña ha elegido tendiendo este que elegir si es correcta o no. Si el usuario pulsa sobre la opción “No” volverá a la pantalla 1. Si pulsa sobre la opción “Si” continuará hasta la pantalla 3. Su interfaz puede verse en la ilustración número 5.



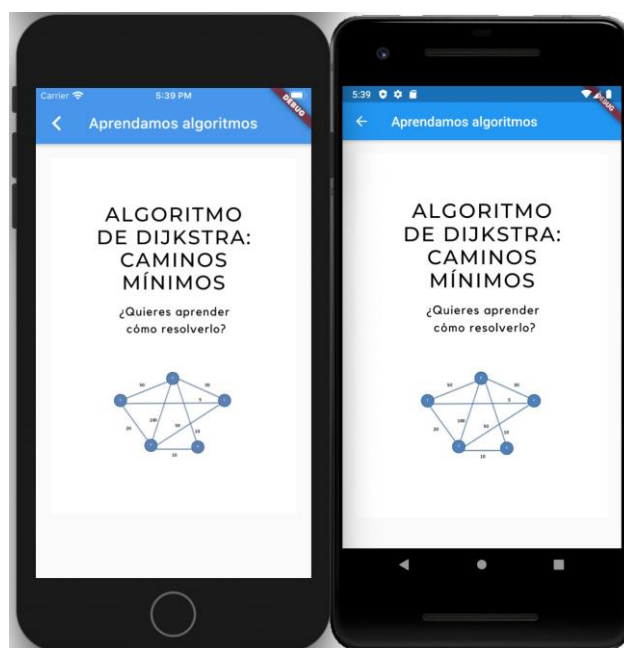
*Ilustración 5. Pantalla 2 aplicación móvil*

- Pantalla 3. A esta pantalla se accede desde la pantalla 2 y pulsando en el botón “Si”. Si ha realizado lo anteriormente expuesto, el usuario se encontrará con un cuadro de diálogo en el que se le informa de la clave introducida y un botón con el mensaje “Perfecto, gracias” que le permitirá continuar hasta la pantalla 4. A continuación, en la ilustración 6 se muestra cómo queda su interfaz.



*Ilustración 6. Pantalla 3 aplicación móvil*

- Pantalla 4. Se accede desde la pantalla 3 pulsando el botón “Perfecto, gracias” disponible en ella. En esta pantalla se le presenta al usuario una imagen a modo de presentación. En esta pantalla el usuario no puede interactuar con la aplicación ya que la aplicación se encuentra a la espera de recibir mensajes con las imágenes que tiene que mostrar. Cuando la aplicación vaya recibiendo mensajes irá mostrando las imágenes correspondientes. Un ejemplo de cómo queda la interfaz es mostrado en la ilustración 7.

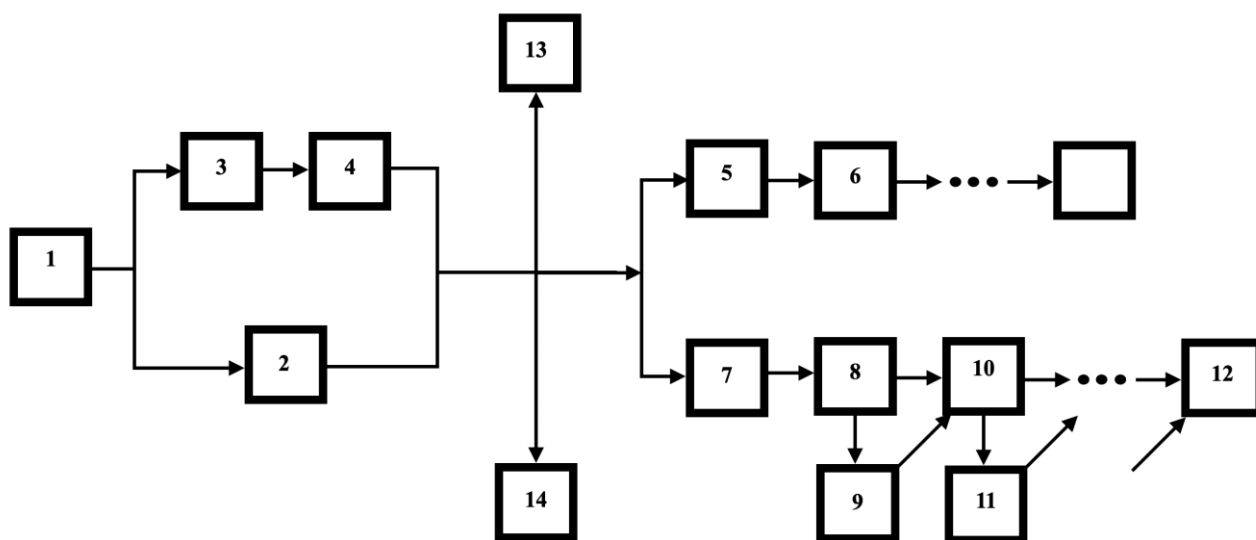


*Ilustración 7. Pantalla 4 aplicación móvil*

### 3.3.1.2 Aplicación Alexa

El flujo de ejecución de la aplicación de Alexa se asemeja al flujo anterior que tenía la aplicación antes de la realización de este proyecto ya que se parte de esa base. A modo de recordatorio, decir que el flujo anterior que contenía la aplicación disponía de una pantalla de bienvenida en la que se exponía la descripción de la aplicación, un camino en el flujo de ejecución en donde se enseñaba al usuario a resolver un problema, un camino en donde se ponía a prueba al usuario, una pantalla en la que se mostraba el algoritmo Dijkstra y una pantalla en la que aparecía el nombre del desarrollador de la aplicación. El actual flujo de ejecución dispone de las mismas pantallas que antes con la salvedad de que el camino en el cual se le pone a prueba al usuario, para este proyecto el problema a solventar se elegirá de manera aleatoria, ya que actualmente la aplicación dispone de más problemas para ser resueltos por parte del usuario. Además de los caminos comentados, la aplicación para este proyecto se le ha añadido una nueva pantalla de inicio en la cual en vez de presentar la aplicación, se le indica al usuario si desea recuperar el progreso llevado en su anterior ejecución o si desea comenzar de nuevo. Según la elección del usuario, se le llevará a una pantalla u otra. Si el usuario desea continuar

con su progreso anterior, la aplicación ya dispondrá de todos los datos guardados anteriormente y por lo tanto el usuario ya podrá continuar con la explicación desde el punto en el que estaba o continuar con la resolución del problema que estuviese resolviendo. Si por el contrario, la elección del usuario es comenzar de nuevo sin importar el progreso anterior, lo primero que se le dirá es que introduzca la clave para comunicarse con la aplicación móvil. Una vez introducida la clave, se le dará el mensaje de bienvenida y estará listo para navegar por la aplicación. Si el usuario desea resolver el problema, como aún la aplicación no tiene ninguna información acerca de que el usuario haya intentado resolver un problema, se le preguntará acerca del nivel de dificultad del problema. Una vez el usuario elija una dificultad, la aplicación elegirá aleatoriamente uno de los problemas que tenga almacenado de esa dificultad y le enunciará el problema al usuario mostrándole por pantalla el grafo del problema a resolver y una tabla del estado de la resolución del problema. En la ilustración 8 se detalla el flujo de ejecución.



*Ilustración 8. Flujo de ejecución aplicación Alexa*

El diagrama de flujo de la ilustración 8 es un gráfico simplificado del flujo de ejecución de la aplicación de Alexa, en las siguientes líneas pasa a describirse con más detalle.

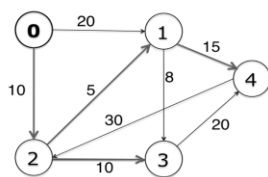
- Pantalla 1. En la pantalla 1 nos encontramos con la página de inicio. En esta pantalla la aplicación le da la bienvenida al usuario y le realiza una pregunta sobre si desea mantener el progreso llevado a cabo en anteriores sesiones o si desea comenzar de nuevo.
- Pantalla 2. Si el usuario se encontraba en la pantalla 1 y responde a la pregunta realizada en la misma diciendo “sí”, la aplicación recuperará el progreso de anteriores sesiones y llevará al usuario a la pantalla 2 expresándole el mensaje de que todo el progreso ha sido recuperado y que puede continuar navegando por la aplicación. Desde la pantalla 2 puede continuar con su progreso

tanto en el grafo de ejemplo diciendo “volver a la explicación”, como en el grafo en el que se pone a prueba diciendo “volver a la resolución”. Además puede acceder a las pantallas en las que puede ver el algoritmo y la pantalla de créditos.

- Pantalla 3. A la pantalla 3 se accede desde la pantalla 1 y respondiendo “no” a la pregunta acerca de si desea continuar con su progreso anterior. En esta pantalla se le comunicará al usuario que debe indicar una contraseña para poder conectarse con la aplicación y la manera que tiene el usuario para responder a la pregunta.
- Pantalla 4. Para acceder a la pantalla 4 es necesario que el usuario se encuentre en la pantalla 3 y le indique la clave a la aplicación para poder comunicarse con el dispositivo móvil. En esta pantalla se le dice al usuario cuál es la contraseña almacenada y se le hace una presentación sobre la aplicación. En dicha presentación se le indica los comandos más comunes para moverse por ella para que de esta forma el usuario sepa cómo utilizar la aplicación. Desde esta pantalla el usuario puede moverse al comienzo del grafo de aprendizaje, puede comenzar un nuevo problema para ponerse a prueba, puede ir a ver el algoritmo de Dijkstra y puede ir a ver los créditos de la aplicación.
- Pantalla 5. Se accede a ella desde cualquier pantalla de la aplicación diciendo “comenzar explicación”. En esta pantalla se le presenta al usuario el grafo que servirá como ejemplo y en el que la aplicación se apoyará para realizar la explicación de cómo resolverlo. Además, la aplicación comenzará a darle los primeros consejos sobre cómo se resuelve el problema planteado. Para ir navegando por la explicación, el usuario tiene que ir diciendo en cada una de las pantallas “siguiente”. Si el abandona la explicación para ir a otra pantalla de la aplicación y desea volver al punto donde dejó la explicación tiene que decir “volver a la explicación” y la aplicación le llevará a la pantalla en la que se encontraba anteriormente, mostrándole esta vez un mensaje distinto ya que la explicación ya fue expuesta la anterior vez que visitó esa misma pantalla. Un ejemplo de cómo se vería una de las pantallas del grafo de explicación tanto en un dispositivo Alexa como en un dispositivo móvil con iOS y con Android puede verse en la ilustración 9.



## APRENDIENDO DIJKSTRA



PASO	NODO SELECCIONADO (N)	CANDIDATOS (C)	DISTANCIAS (D)
Paso inicialización	0	{1, 2, 3, 4}	[20, 10, ∞, ∞]
1	2	{1, 3, 4}	[15, 10, 20, ∞]
2	1	{3, 4}	

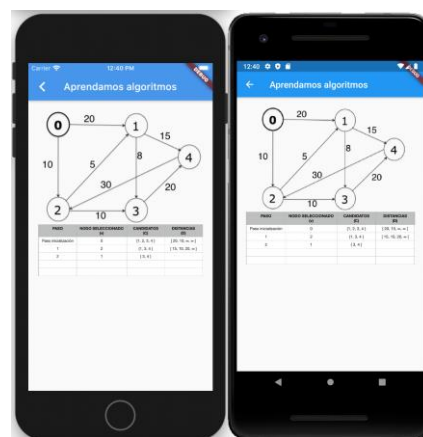


Ilustración 9. Pantalla 5 aplicación Alexa

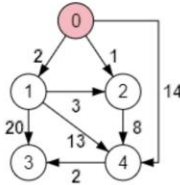
- Pantalla 7. Para acceder a la pantalla 7 el usuario debe decir “resolver problema”. En esta pantalla se le consultará al usuario acerca de qué nivel de dificultad desea que tenga el grafo a resolver.
- Pantalla 8. Para acceder a la pantalla 8 el usuario debe encontrarse en la pantalla 7 e indicar un nivel de dificultad del grafo a resolver a la aplicación. Una vez elegida la dificultad por el usuario, la aplicación seleccionará uno de los problemas que tenga almacenado dentro de ese nivel y mostrará al usuario el enunciado del mismo junto con la primera pregunta que ha de resolver el usuario. Si el usuario responde correctamente a la pregunta, pasa a la siguiente pantalla (representado en el diagrama de flujo como su siguiente pantalla par) en donde se le formular una nueva pregunta correspondiente a esa nueva pantalla hasta llegar a la última pantalla de todas (representada en el diagrama como la pantalla número 12) en donde se le dará la enhorabuena por resolver el grafo. Si el usuario falla cualquiera de las preguntas formuladas, pasa a una nueva pantalla (representada en el diagrama como su siguiente pantalla impar) en donde se le da consejos sobre cómo resolver el problema y se le formula de nuevo la pregunta realizada anteriormente. En cualquier momento el usuario puede volver al punto del grafo donde se encontraba diciendo “volver a la resolución” y la aplicación le devolverá a la pantalla en la que se encontraba resolviendo el grafo. Un ejemplo de cómo se vería esta pantalla tanto en un dispositivo Alexa como en un dispositivo móvil con iOS y con Android sería como el mostrado en la ilustración 10:



## APRENDIENDO DIJKSTRA



### Calcular el camino mínimo de un nodo al resto de nodos



PASO	NODO SELECCIONADO (N)	CANDIDATOS (C)	DISTANCIAS (D)
Paso inicialización	0		
1			
2			
3			



Ilustración 10. Pantalla 8 aplicación Alexa

- Pantalla 13. La pantalla 13 es la correspondiente a la pantalla en donde se le muestra al usuario el algoritmo de Dijkstra en pseudocódigo. Para acceder a ella el usuario debe decir “ver algoritmo”. A continuación, en la ilustración 11 se muestra cómo queda su interfaz en un dispositivo con Alexa integrado, un dispositivo móvil con el sistema operativo iOS y un dispositivo móvil con Android.



## ALGORITMO DE DIJKSTRA

```

función Dijkstra(L[1..n, 1..n]): matriz [2..n]
  matriz D[2..n]
  {Iniciación}
  C ← {2, 3..n}
  para i ← 2 hasta n hacer D[i] ← L[1, i]
  {bucle voraz}
  repetir n - 2 veces
    v ← algún elemento de C que minimiza D[v]
    C ← C \ {v} {e implícitamente S ← S ∪ {v}}
    para cada w ∈ C hacer
      D[w] ← min(D[w], D[v] + L[v, w])
  devolver D
  
```

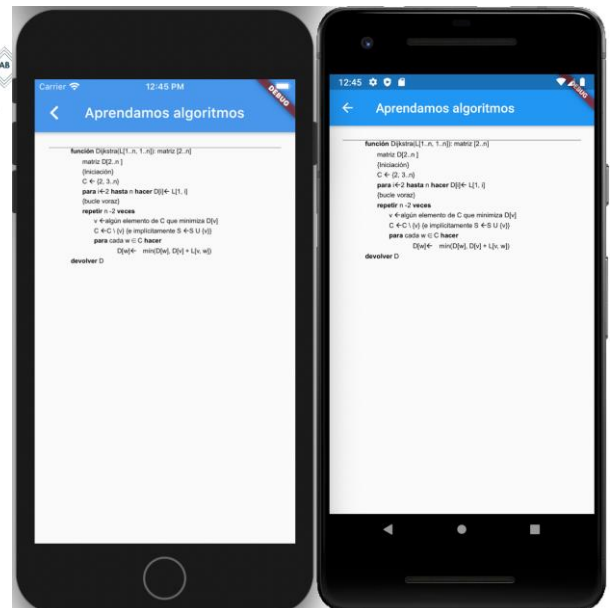


Ilustración 11. Pantalla 13 aplicación Alexa

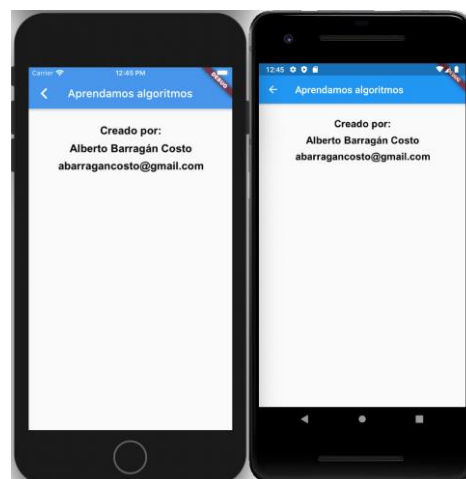
- Pantalla 14. Para acceder a la pantalla 14 el usuario debe decir “ver créditos”. En esta pantalla se le muestra al usuario la persona que ha desarrollado la aplicación junto con su contacto. A continuación, en la ilustración 12, se muestra cómo queda su interfaz en un dispositivo con Alexa integrado, un dispositivo móvil con el sistema operativo iOS y un dispositivo móvil con Android.



CREDITOS



**Creado por:**  
**Alberto Barragán Costo**  
**abarragancosto@gmail.com**



*Ilustración 12. Pantalla 14 aplicación Alexa*

Además de las pantallas mencionadas anteriormente, hay pantallas auxiliares que sirven al usuario como explicación de qué debe decir. Por ejemplo, si al usuario se le indica que debe decir la contraseña y este no responde correctamente a la pregunta, la aplicación le insistirá indicándole cómo debe indicar la contraseña.

Para salir de la aplicación el usuario tiene que decir “salir”. Cuando el usuario cierra la aplicación, esta realiza un almacenado de los datos para posteriores usos por parte del mismo usuario.

### 3.4 Implementación

En los siguientes apartados se describirá la implementación de las dos aplicaciones descritas en este proyecto. Se comenzará detallando cómo se ha desarrollado el código encargado de generar código automáticamente capaz de ser interpretado por el asistente virtual Alexa para la creación de problemas, continuando por la aplicación creada con Flutter y finalizando con una explicación sobre el desarrollo realizado en la aplicación de Alexa.

#### 3.4.1 Generación código automático

Uno de los objetivos principales que tiene el proyecto es la posibilidad de escalar la aplicación añadiéndole problemas que pueda resolver el usuario sin que ello conlleve un gran coste para el desarrollador de la aplicación. Para ello, se ha creado una librería capaz de generar de una manera sencilla todos los ficheros necesarios en la aplicación de Alexa para la generación de problemas.

El intérprete de código de Alexa, para saber qué está diciendo el usuario y de esta manera responderle con la pantalla correspondiente necesita:

- **Intenciones.** Una intención es una acción que cumple con una petición realizada por un usuario. Cada intención puede ser activada por uno o más enunciados. Por ejemplo, cuando el usuario de la aplicación dice “La contraseña es prueba”, Alexa entiende que el usuario ha activado la intención “passwordIntent” ya que es una de las frases que la activan.
- **Manejadores.** Es la parte del código de desarrollo donde se indicada cómo debe comportarse la aplicación para una determinada intención. En los manejadores se ha de indicar el mensaje que ha de recitar Alexa para la pantalla en la que se encuentre el usuario. En esta aplicación, en cada manejador, además del mensaje, se indica qué imágenes se han de mostrar, se envían mensajes a través del webSocket y se almacenan datos para saber en qué pantalla se encuentra el usuario y así poder navegar por la aplicación sin perder el progreso que lleve el usuario.

Primero de todo, nos centraremos en los manejadores y en cómo se han desarrollado y automatizado su creación.

Para facilitar la creación y la mantenibilidad de los manejadores, se han creado una serie de funciones utilizadas por todos ellos. Estas son:

- **AttributeIs.** Recibe el nombre del atributo y el valor que se quiere comprobar. Esta función accederá a los atributos de sesión, obtendrá el atributo recibido y lo compara con el valor esperado.
- **getAttribute.** Esta función recibe el nombre del atributo que se desea obtener, lo obtiene de los atributos de sesión y devuelve su valor.
- **getSpeakOutput.** Recibe la pantalla en la que se encuentra el usuario y obtiene el mensaje que se desea transmitir.
- **intentNameIs.** Recibe el nombre de la intención que se desea comprobar su valor, lo obtiene y lo compara con el valor recibido por parámetro.
- **sendMessageToWebsocket.** Recibe una serie de parámetros con los que compone el mensaje que envía al webSocket.
- **setAttribute.** Recibe el nombre del atributo y el valor que se desea almacenar. Accede a los atributos de sesión y lo almacena.
- **showAPLWithScreen.** Recibe la pantalla que se desea mostrar, comprueba si el dispositivo que utiliza Alexa dispone de pantalla para mostrar imágenes y en caso afirmativo le envía la información al dispositivo.



Gracias a las funciones descritas anteriormente, la generación de los manejadores se reduce a la comprobación sobre qué pantalla se encuentra el usuario y qué ha respondido. Dependiendo de estos parámetros se llama a las funciones auxiliares que almacenarán el cambio de estados de la aplicación, el mensaje a exponer y las imágenes a mostrar. Por tanto, dentro del código de desarrollo de la aplicación, los directorios que encontramos son:

- **Handlers.** En este directorio se encuentra el código de los manejadores. Como se ha comentado anteriormente, en ellos se encuentra el código que hace posible conocer qué intención ha activado el usuario y dependiendo de ella realizar una serie de acciones.
- **Utils.** En este directorio se encuentra el código utilizado por todos los manejadores. Como se ha explicado en el párrafo anterior, estas funciones nos permiten ahorrar gran cantidad de código duplicado dentro de los manejadores y por tanto sencillez, reutilización y mantenibilidad, ya que si en algún momento se desea cambiar, por ejemplo, cómo se envía los mensajes a través del `WebSocket`, únicamente se realizaría en un único lugar y no en todos los manejadores creados.
- **Apl y data.** En estos directorios se encuentra el código que hace posible el mostrado de imágenes en los dispositivos utilizados por el asistente virtual Alexa. En este código se le indican las imágenes a mostrar y cómo estas se comportan dependiendo de las dimensiones del dispositivo o como se distribuyen dentro del marco.
- **Texts.** En este directorio se encuentran los textos que la aplicación expondrá dependiendo de la pantalla que haya seleccionado como la pantalla indicada para la acción realizada por el usuario.

Dentro de todos estos directorios se generan otros directorios, dependiendo del apartado de la aplicación en la que se encuentre la pantalla. Dentro de ese subdirectorio se crean los ficheros necesarios para cada pantalla.

Esta estructura de directorios y funciones comunes, se ha creado para poder automatizar la generación de código y por tanto, la escalabilidad de la aplicación. Para poder hacer esto necesario se ha creado un proyecto independiente llamado “createFiles”.

CreateFiles es una librería con un CLI la cual recibiendo un fichero JSON de entrada, con unas características que se verán más adelante, es capaz de generar todos los directorios y ficheros necesarios para la configuración de problemas en la aplicación de Alexa, es decir, generará todos los ficheros que se encuentran dentro de “handlers”, “apl”, “data” y “texts”.

El CLI de esta librería se compone únicamente de un parámetro de entrada llamado “jsonPath” en donde se le ha de indicar el fichero JSON del que partirá y otro parámetro llamado “outputPath” en donde se le indica cual será el directorio donde se crearán todos los ficheros. Este segundo parámetro

no es obligatorio, ya que si el usuario no le indica ningún directorio utilizará el directorio desde donde se está lanzando.

El fichero JSON de entrada, debe ser un fichero en donde esté toda la información necesaria para la creación de todos los ficheros. En un primer nivel del objeto JSON se ha de indicar el nivel del problema (fácil, medio, difícil). En un segundo nivel, es decir, dentro de la dificultad del problema, se ha de indicar el nombre del problema. Por último dentro del problema tienen que escribirse el nombre de la pantalla y dentro de cada una de ella una serie de parámetros que serán los que la definan. Estos parámetros son:

- `firstTimeText`. Dentro de este parámetro se indicará cual será el texto que la aplicación ha de decir cuando el usuario accede por primera vez a ella habiendo respondido correctamente la pregunta formulada previamente por la aplicación cuya respuesta provoca la llegada a esta pantalla.
- `returnText`. Dentro de este campo se indicará cual es el texto que la aplicación ha de decir cuando el usuario accede a la pantalla indicada diciendo “volver”, es decir, cuando visita la página habiéndola visitado previamente.
- `failText`. Dentro de este campo se ha de escribir el texto que la aplicación dirá cuando un usuario falle la respuesta de la pregunta que la aplicación dirá antes de acceder a esta pantalla.
- `correctAnswer`. Dentro de este parámetro, el desarrollador debe indicar cuál es la respuesta que la aplicación interpretará como correcta para la pregunta previa al acceso de la pantalla a la que se le está dando valor.
- `graphImageSource`. El contenido de este parámetro será una url donde se encuentre el grafo del problema que se desea mostrar en las pantallas de los distintos dispositivos.
- `dataTableImageSource`. Este parámetro contendrá la url donde esté almacenada la imagen de la tabla de datos que se mostrará para la pantalla indicada en los distintos dispositivos.

La librería, con el fichero JSON de entrada cuyo contenido es el descrito anteriormente es capaz de generar todos los ficheros correspondientes a los problemas indicados dentro de él. Para ello, la librería, tiene almacenado el código que se repite dentro de esos ficheros y su funcionamiento se basa en ir recorriendo todos los parámetros del fichero JSON e ir rellenando correctamente esos ficheros, cada uno de ellos con los parámetros necesarios. Además, la librería genera los directorios como la aplicación espera tenerlos almacenados, de esta manera, el desarrollador de la aplicación de Alexa, únicamente tiene que rellenar el JSON, hacer la llamada a la librería, coger todos los ficheros generados y pegarlos dentro de la aplicación. Una vez realizado esto ya tendría todo el código

disponible para que su aplicación tuviese configurado todos los problemas que se deseen introducir haciendo así un código de aplicación totalmente escalable.

Se ha hablado de los manejadores y cómo estos se crean automáticamente, pero la aplicación para Alexa necesita también de intenciones. CreateFiles es capaz, con ese mismo fichero JSON de entrada, de crear todas las intenciones necesarias mediante las respuestas correctas que ha indicado el desarrollador en el fichero. De esta forma, al igual que se ha realizado con los directorios, el desarrollador tiene que copiar estas intenciones y pegarlas dentro de su aplicación para que así Alexa sea capaz de interpretar qué desea decir el usuario.

El funcionamiento interno de la librería consiste en tener unas plantillas que, gracias a las funciones del directorio “utils” son comunes para todos los problemas e ir recorriendo todas las propiedades del objeto json de entrada para ir introduciendo en el sitio correspondiente cada una de ellas. Una vez formado el contenido del fichero que queremos generar, este se crea en el directorio indicado por parámetro.

### 3.4.2 Aplicación Flutter

La aplicación móvil anterior a la realización de este proyecto contenía una serie de carencias analizadas en apartados anteriores como son soporte únicamente para el sistema operativo Android, no existía una comunicación uno a uno con la aplicación de Alexa, almacenamiento de imágenes dentro de la propia aplicación... por lo que para este proyecto se ha decidido realizar la implementación de una aplicación en Flutter desde cero sin tener en cuenta lo desarrollado en el proyecto anterior. Gracias a esta tecnología se consigue, con un mismo código dar soporte al sistema operativo Android e iOS por lo que de esta manera se disponibiliza también la aplicación para el sistema operativo iOS. Para esta nueva aplicación se ha realizado una serie de implementaciones que cubren las carencias de la anterior aplicación las cuales serán descritas a continuación:

- **Conexión con aplicación de Alexa.** Para corregir la carencia anterior de la comunicación con la aplicación de Alexa, para esta nueva aplicación se ha desarrollado un campo de introducción de texto en la pantalla de inicio de la aplicación. Mediante este campo, el usuario puede introducir la clave compartida con la aplicación de Alexa. Una vez el usuario introduce la clave en este campo, la aplicación la almacena y comprobará con ella los mensajes que le lleguen por parte del websocket para así mostrar las imágenes o no. El usuario, dentro de cualquier pantalla, tiene disponible la opción de volver a la pantalla principal para introducir una nueva clave siempre que así lo desee. Cuando el usuario introduce una nueva contraseña, la aplicación levanta una nueva conexión con el websocket con la contraseña actualizada. Se han desarrollado, además, unas

ventanas emergentes para que el usuario pueda verificar que clave ha introducido y en caso de ser correcta poder confirmarla.

- **Mostrado de imágenes.** Anteriormente todas las imágenes se almacenaban dentro de la aplicación. Para esta nueva aplicación se ha desarrollado una pantalla, cuya implementación es un Widget principal con un webSocket, el cual cada vez que recibe un mensaje, lo analiza, y en caso de ser correcto, muestra las imágenes recibidas en él. Las imágenes son recibidas a través de una url que la aplicación se encarga de acceder a dicha dirección y mostrarlas en los contenedores de imágenes correspondientes. De esta manera se ha desarrollado únicamente una pantalla la cual va recargando las imágenes según la aplicación recibe mensajes desde el webSocket, por lo que de cara al usuario no visualiza una transición entre pantallas, si no que únicamente ve unas imágenes desapareciendo y otras apareciendo dentro de una misma pantalla.

### 3.4.3 Aplicación Alexa

Para adaptar la aplicación de Alexa a los nuevos objetivos marcados en este proyecto se han tenido que realizar una serie de cambios significativos en la aplicación como son la persistencia de datos, el envío de clave e imágenes por el webSocket, introducción de niveles de dificultad y aleatoriedad en la elección de problemas dentro de un determinado nivel. Además, con respecto al estado anterior a la realización de este proyecto de la aplicación, se han añadido nuevos problemas para que le usuario puedan ponerse a prueba. A continuación se detallan cada uno de estos desarrollos.

- **Persistencia de datos.** Durante el recorrido del usuario en la aplicación se van almacenando unos datos como pueden ser en qué pantalla de la explicación se encuentra, cuantas respuestas de la resolución lleva acertadas, qué problema ha sido el elegido, qué contraseña introdujo... Estos datos eran únicamente almacenados en la sesión del usuario perdiéndose su progreso cuando el usuario cerraba la aplicación y por tanto cerraba sesión. Para este proyecto, se ha desarrollado que cuando el usuario cierra la aplicación, esas propiedades de sesión pasan a ser almacenadas en un Bucket de S3 cuyo identificador es el mismo que el identificador de la skill, el cual es único. En el Bucket se almacena todos los atributos junto con el usuario de esos atributos de tal manera que cuando un usuario desea recuperar los atributos de la sesión anterior, la aplicación accede al Bucket y recupera dichos atributos pertenecientes al usuario en cuestión. Una vez recuperados los datos del Bucket, la aplicación almacena esos atributos en la sesión para poder utilizarlo durante la sesión. Si el usuario no desea recuperar los datos, la aplicación no inicializará ningún atributo de sesión y los irá creando conforme el usuario vaya accediendo a las distintas pantallas de la aplicación.

- **Envío de clave.** Como se ha explicado en anteriores apartados, para poder realizar una comunicación uno a uno Aplicación de Alexa con aplicación móvil es necesario el envío y comprobación de una clave común. Esta era una de las carencias más importantes que tenía la aplicación, para solventarla se ha desarrollado una pantalla en la cual Alexa le indica al usuario que necesita introducir una clave para poder comunicarse con la aplicación. Una vez que el usuario introduce su clave, la aplicación de Alexa la almacena como un atributo de sesión y la envía en cada mensaje del webSocket. De esta manera, la aplicación móvil tiene acceso a esa clave y en caso de coincidir con su clave muestra las imágenes recibidas por la aplicación de Alexa
- **Envío de imágenes.** Como también se ha expuesto en anteriores apartados, la aplicación Android anteriormente tenía alojadas las imágenes que necesitaba mostrar en cada pantalla por lo que la aplicación de Alexa únicamente le enviaba un código con la pantalla a mostrar. Esto ha cambiado ya que ahora la aplicación para dispositivos móviles no aloja ninguna imagen si no que todo es recibido a través del webSocket. Desde la aplicación de Alexa se ha modificado el mensaje enviado a la aplicación Android añadiendo qué imágenes debe mostrar
- **Selección de niveles de dificultad y aleatoriedad en la elección del problema.** Una de las novedades realizadas en este proyecto es el aumento de problemas a resolver y su división por niveles de dificultad. Para que el usuario pueda seleccionar un nivel de dificultad se ha desarrollado una pantalla en la cual se le pregunta al usuario qué nivel de dificultad desea que contenga el problema a resolver. Una vez seleccionado el nivel de dificultad por parte del usuario, la aplicación recorre todos los problemas que contenga dentro de ese nivel de dificultad y selecciona aleatoriamente uno de ellos. Cuando la aplicación ya tiene con qué problema pondrá a prueba al usuario, almacena su nombre y su nivel de dificultad en un atributo de sesión para así poder enviar las imágenes correspondientes a dicho problema a la aplicación y de esta manera, además, saber cuáles son las preguntas que ha de exponer al usuario y cuáles son las respuestas esperadas para cada una de ellas.
- **Nuevos problemas a resolver.** Además de lo comentado anteriormente, se han añadido 2 nuevos problemas con los cuales el usuario podrá ponerse a prueba y valorar sus conocimientos en el algoritmo Dijkstra.

## 3.5 Pruebas

En este apartado, se describirán las pruebas realizadas para la comprobación del correcto funcionamiento de la aplicación desarrollada. Se dividirá en dos apartados. Por un lado se describirán las pruebas manuales realizadas por parte del desarrollador y por otro lado se describirán las pruebas realizadas por distintos usuarios de la aplicación, así como el cuestionario realizado posteriormente y los resultados obtenidos a través del mismo.

### 3.5.1 Pruebas manuales.

El apartado de pruebas manuales se divide en dos partes, una la correspondiente al generador automático de código y otra a la correspondiente con la aplicación de Alexa. A continuación se detallarán las pruebas realizadas para ambas partes.

#### 3.5.1.1 Pruebas manuales en el generador de código

Para la librería del generador automático de código se han realizado una serie de pruebas consistentes en la introducción de datos mediante diferentes ficheros y una comprobación posterior consistente en la comprobación del código generado. Junto con el código de desarrollo de la librería se encuentran las pruebas realizadas. En todas ellas se realiza la llamada a la función generadora con unos ficheros de entrada y se comprueban todos los ficheros generados con una línea base que se encuentra dentro del directorio de test de la librería. Las pruebas realizadas se describen a continuación en diferentes tablas que contendrán el objetivo buscado con la prueba, cómo se ha realizado la prueba, cual es el resultado esperado y cuál ha sido el resultado de la prueba.

Prueba GC 1

Objetivo	Comprobar la correcta creación de ficheros cuando se introduce un único problema dentro de un nivel
Prueba realizada	En el fichero json de entrada se introduce un problema de nivel difícil
Resultado esperado	La librería genera correctamente todos los ficheros dentro de un directorio llamado con el nombre del problema y que a su vez está dentro un directorio llamado “difícil”
Resultado	OK

### Prueba GC 2

Objetivo	Comprobar la correcta creación de ficheros cuando se introducen varios problemas dentro de un nivel
Prueba realizada	En el fichero json de entrada se introducen dos problemas de nivel medio
Resultado esperado	La librería genera correctamente todos los ficheros dentro de dos directorios llamados cada uno de ellos con el nombre del problema correspondiente y ambos se encuentran dentro de un directorio llamado “medio”
Resultado	OK

### Prueba GC 3

Objetivo	Comprobar la correcta creación de ficheros cuando se introducen varios problemas dentro de varios niveles
Prueba realizada	En el fichero json de entrada se introducen dos problemas de nivel medio y un problema de nivel difícil
Resultado esperado	La librería genera correctamente todos los ficheros dentro de tres directorios llamados cada uno de ellos con el nombre del problema correspondiente y cada uno de ellos se encuentra dentro de un directorio cuyo nombre es el nivel de dificultad al que pertenecen.
Resultado	OK

### Prueba GC 4

Objetivo	Comprobar el correcto funcionamiento de la librería cuando no se introducen algunos datos
Prueba realizada	En el fichero json de entrada se introduce un objeto vacío dentro del nivel difícil y dos problemas vacíos dentro del nivel medio
Resultado esperado	La librería no genera ningún directorio dentro del nivel vacío y dentro del nivel medio genera dos directorios con los nombres de los ficheros pero sin ningún contenido
Resultado	OK

### 3.5.1.2 Pruebas manuales aplicación Alexa

El apartado de pruebas manuales por parte del desarrollador para la aplicación de Alexa se ha realizado en base a los objetivos marcados en el proyecto. A continuación se detallan las pruebas realizadas mediante tablas. En ellas se establece el objetivo que se busca conseguir con la realización de la prueba, desde qué pantalla se parte y a qué pantalla se busca llegar siguiendo el enumerado del flujo de ejecución visto en el apartado 3.3.1 y la frase que expresa el desarrollador para la prueba.

#### Prueba Alexa 1

Objetivo	Comprobar la solicitud de contraseña
Pantalla actual	1
Frase formulada por el usuario	No
Pantalla esperada	3
Resultado	OK

#### Prueba Alexa 2

Objetivo	Comprobar el almacenamiento correcto de la contraseña
Pantalla actual	3
Frase formulada por el usuario	La contraseña es prueba
Pantalla esperada	4
Resultado	OK



### Prueba Alexa 3

Objetivo	Comprobar el almacenado de datos entre sesiones
Pantalla actual	1
Frase formulada por el usuario	Si
Pantalla esperada	2
Resultado	OK

### Prueba Alexa 4

Objetivo	Comprobar el correcto funcionamiento del almacenamiento de la progresión en el apartado de explicación del problema
Pantalla actual	14
Frase formulada por el usuario	Volver a la explicación
Pantalla esperada	La última pantalla visitada dentro del grafo de explicación
Resultado	OK

### Prueba Alexa 5

Objetivo	Comprobar el correcto funcionamiento del almacenamiento de la progresión en el apartado de resolución del problema
Pantalla actual	10
Frase formulada por el usuario	Volver a la resolución
Pantalla esperada	La última pantalla visitada dentro del grafo de resolución
Resultado	OK

### Prueba Alexa 6

Objetivo	Verificar la solicitud por parte de la aplicación de un nivel de dificultad para el nodo de resolución del problema
Pantalla actual	4
Frase formulada por el usuario	Comenzar resolución
Pantalla esperada	7
Resultado	OK

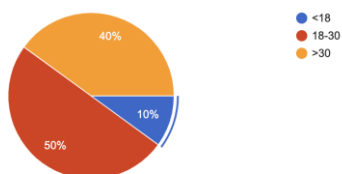
### Prueba Alexa 7

Objetivo	Correcto funcionamiento de selección de dificultad del problema a resolver
Pantalla actual	7
Frase formulada por el usuario	Quiero el nivel medio
Pantalla esperada	8
Resultado	OK

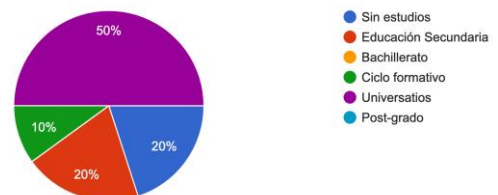
### 3.5.2 Pruebas con usuarios

Para las pruebas con usuarios se ha creado un documento que recoge cómo acceder a la aplicación de Alexa, cómo poder instalarse la aplicación móvil y cómo deben realizarse las pruebas (véase Anexo 2). El documento ha sido enviado a 10 personas de diversa edad y de distintos estudios finalizados. Al final del documento se envía un cuestionario creado expresamente para este proyecto cuyas respuestas se describen a continuación:

¿Qué edad tienes?  
10 respuestas

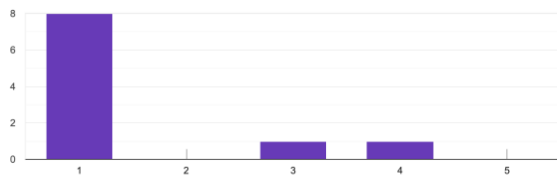


Estudios finalizados  
10 respuestas



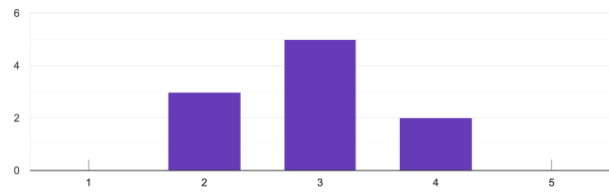
Antes del uso de la aplicación, ¿Cuál considerabas que era el conocimiento que tenías acerca de un algoritmo voraz?

10 respuestas



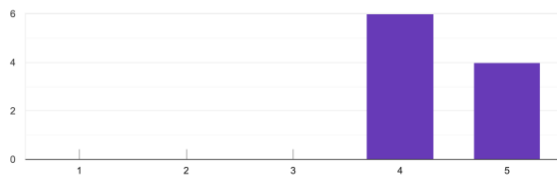
Después del uso de la aplicación ¿Cuál crees que es tu nivel de conocimiento actual?

10 respuestas



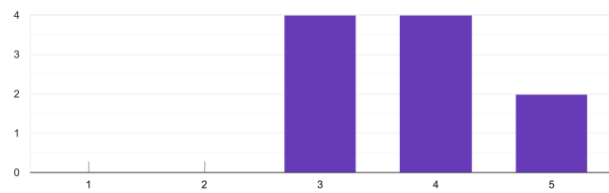
En general, ¿Cuál sería tu valoración sobre la aplicación después de su uso?

10 respuestas



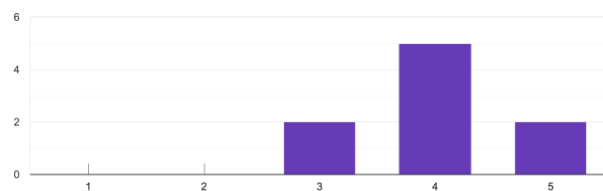
¿Consideras que has aprendido con el uso de la aplicación?

10 respuestas



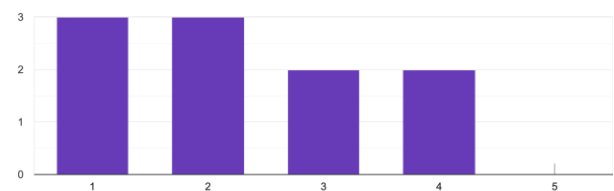
¿Los conceptos están bien explicados?

9 respuestas



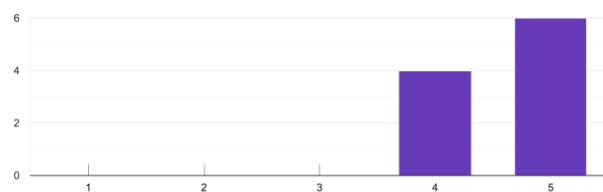
Me he sentido perdido durante el uso de la aplicación

10 respuestas



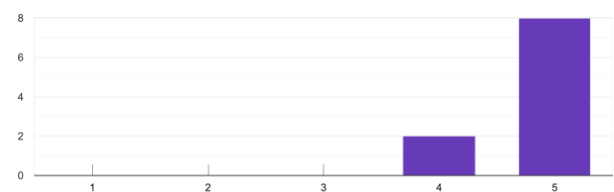
¿Te ha parecido útil y divertida esta manera de resolver un problema?

10 respuestas



Recomendarías la aplicación

10 respuestas



Tras analizar el resultado de las pruebas realizadas a distintos usuarios, se puede llegar a las siguientes conclusiones:

- El 50% de las personas a las que se le ha realizado el estudio han concluido estudios universitarios pero únicamente un 20% del total dice conocer qué es un algoritmo voraz.
- Antes del uso de la aplicación, el 80% de usuarios decía no conocer nada sobre el uso de un algoritmo voraz, siendo del 0% una vez utilizada la aplicación. Además, el 100% de los usuarios ha respondido con 3 o más puntos sobre 5 que si han aprendido algo con el uso de

la aplicación. Estas 2 estadísticas unidas a que todos los usuarios han dado 3 puntos o más a la pregunta acerca de si los conceptos están bien explicados, puede concluir que todos los usuarios, una vez utilizada la aplicación, conocen algo de un algoritmo ya que la explicación de los conceptos es la adecuada.

- La respuesta a la pregunta acerca de si se han sentido perdidos durante el uso de la aplicación es muy diversa por lo que habría que trabajar en definir mejor la explicación de la navegación dentro de la aplicación.
- En general, la aplicación ha parecido útil y recomendable debido a que el 100% de los usuarios ha dado 4 o 5 puntos a las preguntas sobre si les ha parecido entretenida y sobre si recomendaría la aplicación a algún compañero.

Como conclusión a toda la encuesta se podría decir que los usuarios han aprendido, les ha parecido divertida y recomendable pero que hay que trabajar un poco más en la explicación sobre la navegación dentro de la aplicación.

## 4 Conclusiones

En los 2 próximos apartados se expondrán las conclusiones a las que ha llevado la realización de este proyecto. En un primer apartado se expondrán las conclusiones más objetivas en base a los objetivos propuestos al comienzo del proyecto y en el segundo apartado se hablará de una conclusión más subjetiva.

### 4.1 Conclusiones generales.

El objetivo principal de este proyecto es dotar de nuevas características a la aplicación ya existente llamada *Aprendamos Algoritmos* con la finalidad de que pueda ser desplegada en las tiendas de aplicaciones y utilizada por usuarios reales. Para poder lograr este objetivo, se marcaron una serie de objetivos más específicos, los cuales se han realizado satisfactoriamente. En las siguientes líneas se describirá la realización el cómo se ha logrado cada uno de estos objetivos.

- **Facilitar la creación de nuevos problemas.** Para lograr este objetivo se ha desarrollado una librería que genera todos los ficheros necesarios por la aplicación de Alexa. Además, la aplicación móvil se ha implementado de tal manera que la introducción de nuevos problemas no supone cambios en su código
- **Dotar a la aplicación de diferentes niveles con varios problemas.** Para lograr este objetivo se ha utilizado la librería de generación de nuevos problemas. Además, como la aplicación no disponía de niveles y únicamente tenía un problema para poner a prueba al usuario, se ha tenido que rehacer parte de su código para adaptarse a estos nuevos cambios. Se han realizado las pantallas para preguntar al usuario sobre qué nivel desea, se ha desarrollado un código capaz de elegir aleatoriamente entre los problemas que tenga cada nivel y se han creado nuevas variables para que la aplicación conozca cual es el problema a resolver.
- **Comunicación entre aplicación Alexa y aplicación móvil correcta.** Este objetivo se ha logrado añadiendo la posibilidad de añadir tanto en la aplicación de Alexa como en la aplicación para dispositivos móviles una clave que el usuario debe introducir en ambas aplicaciones para que de esta forma la aplicación de Alexa conozca con qué dispositivo móvil tiene que conectarse. Esta clave está almacenada en ambas aplicaciones y viaja a través del WebSocket para la correcta comunicación.
- **Creación de la aplicación móvil para el sistema operativo iOS.** Este objetivo se ha realizado mediante una aplicación desarrollada con Flutter y desplegada tanto en Android como en iOS. Gracias a esto, de cara a futuros desarrollos, con un único código de desarrollo, se pueden realizar cambios en ambas plataformas facilitando así el tiempo de desarrollo en código futuro.

- **Creación de carga de imágenes de manera dinámica en la aplicación móvil.** Para la resolución de este objetivo se ha realizado una pantalla en la aplicación la cual recibe el mensaje del websocket y carga de manera dinámica las imágenes que le llegan mostrándolas al usuario.
- **Persistencia entre sesiones.** Este objetivo se ha logrado almacenando todos los datos de la sesión del usuario en un Bucket de Amazon S3. Una vez que el usuario inicia sesión, si este desea recuperar sus datos, la aplicación accede al Bucket donde están almacenados y los guarda en los atributos de sesión. De esta manera el usuario puede continuar con su progreso.

Si se observa el logro de todos los objetivos propuestos para este proyecto puede concluirse que logra el objetivo principal de estar dotado de las características suficientes como para que sea desplegado en las tiendas de aplicaciones y por tanto probado por un gran número de personas. La aplicación, gracias a estos nuevos desarrollos, puede ser utilizada por cualquier persona, ya que problemas como no tener disponible una comunicación uno a uno con la aplicación móvil, el coste que suponía desarrollar nuevos problemas, no estar disponible para iOS, no tener persistencia entre sesiones... hacían que, pese a ser una aplicación con importantes características, estas no fuesen suficientes como para ser una aplicación final. Ahora, ya si puede considerarse una aplicación lista para ser usada.

El uso de la aplicación por parte de usuarios se irá viendo con el tiempo. Esta aplicación aún falta que sea analizada en su uso en una clase, ahí realmente será donde se vea cuanto puede aportar a estudiantes de algoritmos, pero creo, y esta conclusión es algo más personal, que esta aplicación puede ser el principio de un proyecto más grande. Una aplicación que no solo trate de enseñar un algoritmo Dijkstra, ni un algoritmo cualquiera, sino que trate de enseñar cualquier tipo de problema matemático. La generación automática de problemas, puede dar lugar a llevar a cabo problemas de todo tipo para que cualquier profesor pueda generar su propia aplicación para poner a prueba a sus alumnos. Ahora mismo el generador necesita introducir un fichero json para crear esos problemas, pero se puede trabajar para que sea una interfaz gráfica la que permita el ingreso de datos o incluso que la aplicación sea capaz de resolver ella misma los problemas y generar automáticamente el código, ya que de esta manera se haría más accesible la herramienta a todo tipo de profesores. Cada profesor podría tener sus problemas y mediante un código darles acceso a sus alumnos a un determinado tipo de problemas. Además, con el cambio que se ha realizado en la aplicación móvil, podría mostrar cualquier tipo de problemas sin necesidad de agregar nuevos desarrollos, por lo que para ese caso todo el trabajo ya estaría realizado.

Hay infinidad de caminos que se pueden recorrer después de este proyecto. Lo que sí que se ha avanzado es poder realizar una prueba piloto con alumnos que estén estudiando Dijkstra y poder ver

si les es útil y si realmente les gustaría utilizarlo para el resto de asignaturas. Una vez realizado ese estudio el campo de actuación es muy amplio.

## 4.2 Conclusiones personales

Como conclusión personal me gustaría comentar todo lo que para mí ha supuesto este proyecto. Desde el momento en el que estaba defendiendo el anterior proyecto y un profesor me realizó la pregunta de “¿la aplicación puede escalar fácilmente?” comenzó la idea y el desarrollo de este proyecto. En aquel momento tuve que decirle que no, que el proyecto, tal y como estaba diseñado, era muy difícil añadirle nuevos problemas y por tanto era muy difícil su escalabilidad, pero claro, en ese momento me di cuenta de que para que el proyecto tuviese futuro y de verdad valiese tendría que haber una manera sencilla de generar nuevos problemas. No iba a ser fácil, ya que al final hay que atarse a una determinada arquitectura marcada por Amazon para el desarrollo de sus aplicaciones, pero después de darle muchas y muchas vueltas surgió la idea del generador automático de código, la aplicación con carga de imágenes desde la web y todo el resto de ideas del proyecto. Al final, creo que ha quedado una aplicación fácilmente escalable y sobre todo, puede servir para seguir ampliando por muchos caminos, ya que permite nuevos problemas, realizar estudios de aprendizaje, crear un frontal para que sea aún más fácil crear problemas de Dijkstra, crear otro tipo de problemas...

Si me centrare en qué me ha aportado este proyecto, sin duda ha sido el descubrimiento de Flutter. Creo que tiene un potencial enorme, que es una herramienta que está relativamente empezando, pero que ha venido para reinar las librerías de programación multiplataforma. Se programa de una manera muy visual y se pueden hacer cosas muy potentes con todas las librerías que tiene integradas. Además, ahora mismo hay muy pocos desarrolladores especializados en Flutter y cada vez más empresas están migrando sus aplicaciones a Flutter por lo que creo que puede ser una muy buena oportunidad profesional para la cual, creo que voy a formarme ya que por lo que he visto en este proyecto estoy muy ilusionado.

Otra de las cosas que más me ha gustado de este proyecto es ver como una idea muy vaga en sus comienzos y con muchas dudas sobre qué hacer esté creciendo tanto y tenga tanto potencial de crecimiento. Me gustaría seguir ampliando este proyecto y haciéndolo crecer para ojalá un día varios alumnos de diferentes grados de estudio puedan usar una aplicación cuya idea surgió de la cabeza del tutor y mía. Estaría muy feliz si algún día lograra conseguir eso.

También cabe decir que la creación del proyecto no ha sido fácil para mí. Venía de un larguísimo periodo de tiempo para la realización del anterior y sin descanso, he dedicado todas mis vacaciones

a este proyecto, por lo que aunque no han sido tantos meses como el anterior, sí que llevo mucho tiempo dándole vueltas a este proyecto y a contrarreloj. A partir de ahora seguiré dedicándole tiempo pero desde una mayor calma

### 4.3 Trabajos futuros

Este proyecto ha mejorado muchas de las características de la aplicación de Alexa llamada *Aprendamos Algoritmos* y ha conseguido los objetivos establecidos al comienzo del mismo, pero una vez esté desplegada la aplicación, como se ha venido comentando en apartados anteriores, surgen nuevos objetivos y nuevas vías de desarrollo para enriquecer aún más la aplicación para dotarla de diferentes y diversos tipos de problemas. Algunos de estos objetivos podrían ser:

- **Realizar un estudio de usabilidad.** Con la aplicación ya desplegada, el primer paso podría ser utilizarla con alumnos estudiantes del grado de ingeniería informática cuando estén estudiando el algoritmo Dijkstra y estudiar si les ha sido útil, si han aprendido o si les ha gustado este nuevo método de aprendizaje
- **Generar nuevos problemas de Dijkstra.** Ir dotando a la aplicación de nuevos problemas de cara a que los usuarios no se cansen de utilizar la aplicación, si no que siempre tengan un motivo para seguir utilizándola.
- **Modificar el generador para aceptar otros métodos de entrada.** Actualmente el fichero de entrada para el generador automático de problemas es un fichero json. Se podrían investigar otros métodos de entrada para hacer más accesible la librería.
- **Adaptar la aplicación para nuevos tipos de problemas.** Con este objetivo se conseguiría que profesores de más materias puedan utilizar la aplicación en sus clases y de esta manera acercar la aplicación a más alumnos.
- **Solucionador de problemas.** La aplicación podría tener el script de resolución de problemas, el profesor le pasaría un enunciado y la aplicación podría generar la solución y con ella qué preguntas tiene que transmitir y qué respuestas espera recibir. De esta manera se abriría un campo muy amplio para el uso con profesores de diversas materias.



## 5 Bibliografía

- Definición y uso de Flutter. 2020. <https://flutter.dev/>
- Instalación de Flutter. 2020. <https://flutter.dev/docs/get-started/install>
- Carga de imágenes desde url en Flutter. 2020. <https://flutter.dev/docs/cookbook/images/fading-in-images>
- WebSocket en Flutter. 2020. <https://flutter.dev/docs/cookbook/networking/web-sockets>
- Díálogos en Flutter. 2020. <https://www.youtube.com/watch?v=wfoSGjBOQj8>
- Generación de código automático. 2018. <https://www.freecodecamp.org/news/how-to-create-files-automatically-and-save-time-with-magic-scaffolding-8dcd1b31483/>
- Librería para generar ficheros en node. 2020. [https://nodejs.org/api/fs.html#fs\\_file\\_system](https://nodejs.org/api/fs.html#fs_file_system)
- Persistencia de datos en Alexa. 2019. <https://plataforma.keepcoding.io/courses/572949/lectures/10316803>
- Documentación aplicación *Aprendamos Algoritmos*. 2020. <https://github.com/abarragancosto/aprendamosAlgoritmos2/blob/main/memoriaTFGAlbertoBarraganAprendizajeDijkstra.pdf>
- Código desplegado nueva aplicación. 2020. <https://github.com/abarragancosto/aprendamosAlgoritmos2/tree/main/lambda>
- Generador automático de código. 2020. <https://github.com/abarragancosto/createFiles>

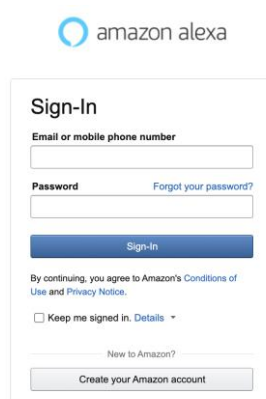
## Anexo 1. Instalación

La aplicación *Aprendamos algoritmos* consta de una aplicación para Alexa la cual se comunica con una aplicación para dispositivos móviles. Actualmente ninguna de las aplicaciones está disponible en su respectiva tienda de aplicaciones, por lo que en esta guía se expondrá cómo poder utilizar las distintas aplicaciones.

### Aplicación de Alexa.

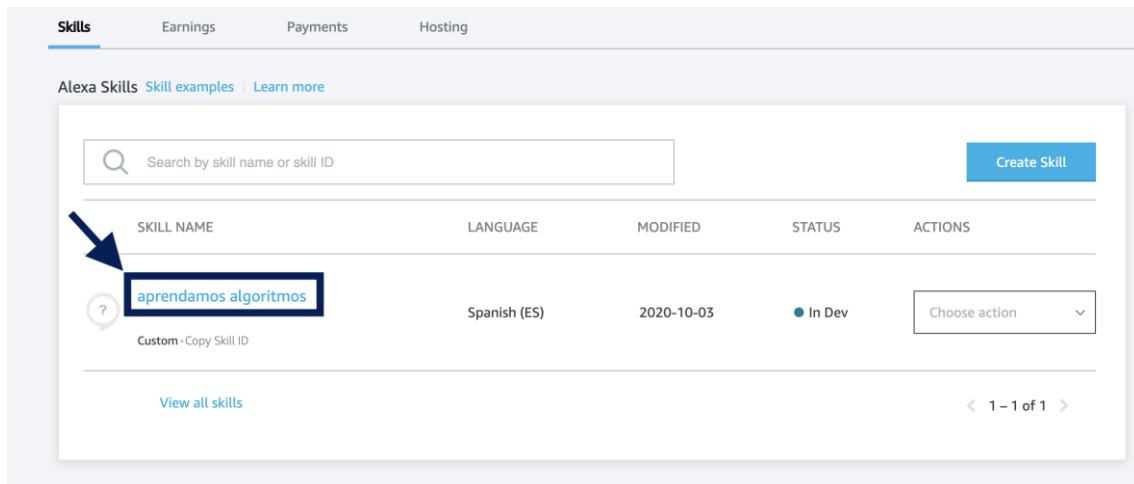
Como se ha comentado en el párrafo anterior, la aplicación de Alexa aún no ha sido desplegada en la tienda de aplicaciones y debido a que Amazon no permite sacar versiones de prueba de una aplicación antes de desplegarla a menos que esta vaya a ser desplegada en países como Estados Unidos, Alemania o Reino Unido, se ha creado una cuenta de desarrollador de aplicaciones de Alexa, donde está la aplicación subida y disponible para ser probada. Para poder acceder a esta cuenta y por tanto, utilizar la aplicación, se deben seguir los siguientes pasos:

1- **Acceder a Alexa Developer Console.** Para ello, se tiene que acceder a un navegador web y colocar en la barra de navegación la url “<https://developer.amazon.com/alexa/console/ask>”. Una vez realizada esta acción debería aparecer una web con un contenido similar al de la siguiente imagen:

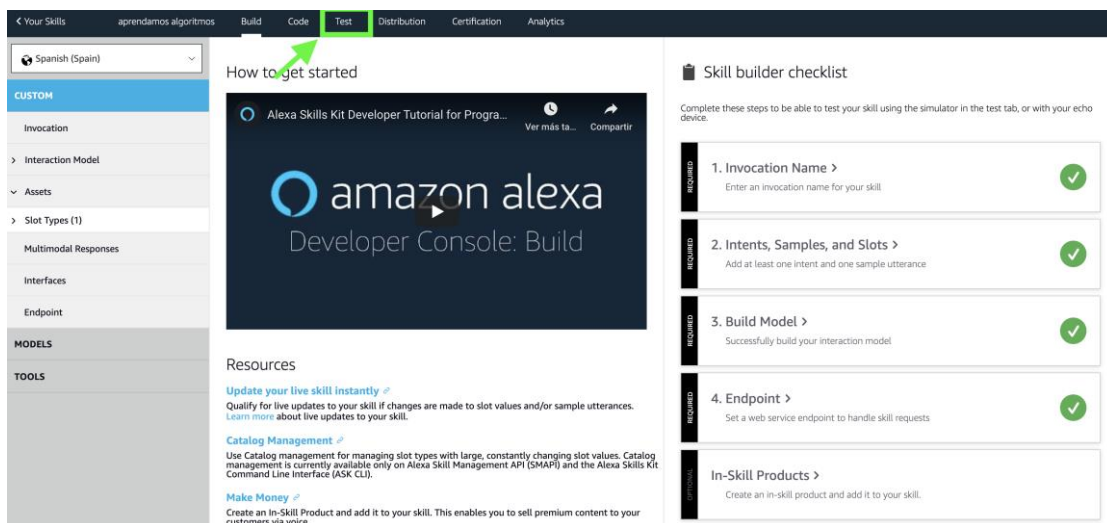


2- **Hacer login en Alexa Developer Console.** Para realizar las pruebas se ha creado un usuario cuyo email es “[aprendamos.algoritmos.tfg@gmail.com](mailto:aprendamos.algoritmos.tfg@gmail.com)” y contraseña es “**aprendamosCuestionario**”. Para realizar el login en la aplicación se debe introducir usuario y contraseña en su lugar correspondiente y hacer click en “Sign-in”.

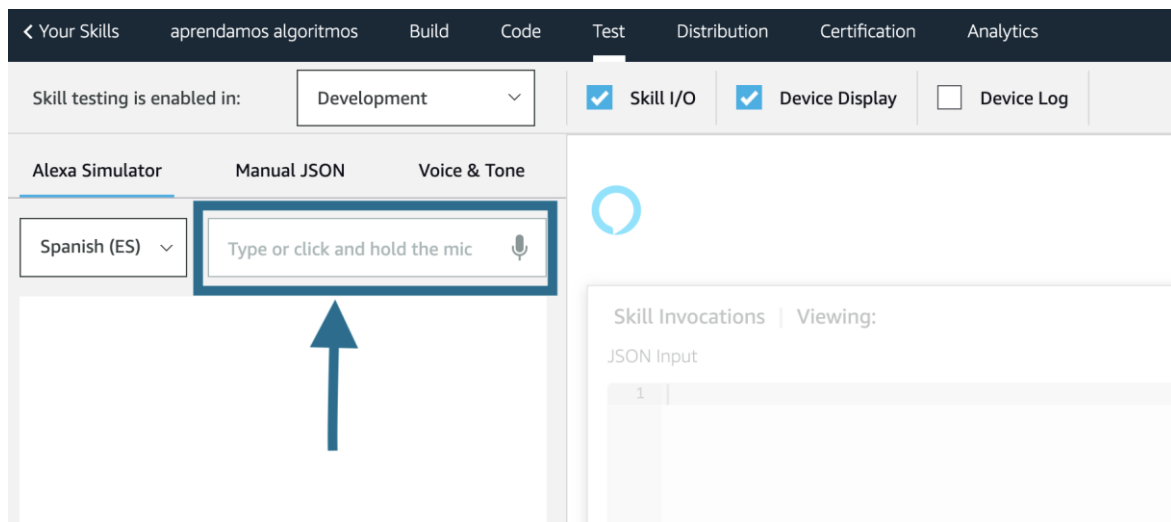
3- **Seleccionar la aplicación “Aprendamos Algoritmos”.** Para ello hay que hacer click en el nombre que se encuentra de color azul como se muestra indicado en la siguiente imagen.



4- **Acceder a la pestaña llamada “Test”**. Una vez dentro de la aplicación, en la parte de arriba a la derecha se puede observar la pestaña “Test”, Hay que hacer click sobre ella como se muestra en la siguiente imagen:



5- Una vez dentro de la pestaña test ya estarás en el apartado donde poder realizar las pruebas. Para la introducción de texto hay habilitado un campo para ello descrito en la siguiente imagen:



Una vez se hayan realizados estos pasos ya puede ser probada la aplicación para Alexa.

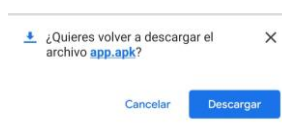
## Aplicación móvil.

La aplicación móvil está desarrollada para dos sistemas operativos como son Android e iOS. Al igual que ocurre con la aplicación de Alexa, al estar en un contexto de pruebas, aún no están subidas las aplicaciones a las tiendas de aplicaciones, por lo que habrá que instalarlas manualmente.

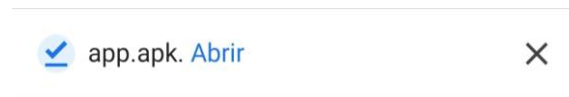
En el caso del sistema operativo iOS la aplicación es necesario firmarla por parte del desarrollador de aplicaciones, por lo que la aplicación para iOS no está disponible para pruebas hasta que no sea subida a la tienda de aplicaciones.

Para instalar la aplicación en el sistema operativo Android hay que realizar los siguientes pasos:

- Acceder a la url donde se encuentra el fichero .apk disponible para su descarga. La url es <https://github.com/abarragancosto/aprendamosAlgoritmos2/raw/main/app.apk> Una vez introducida la url nos saldrá un mensaje como el siguiente:



- Se pulsa sobre el botón “descargar” para descargar la aplicación. Una vez esté completa la descarga nos debería aparecer un mensaje con la palabra “Abrir” como en la siguiente imagen:



- Una vez hayamos pulsado sobre la opción de abrir, nos debería salir una pantalla con la opción de “instalar. Pulsamos sobre dicho botón y debería comenzar a instalar la aplicación



- Una vez la instalación haya sido completada, debería abrirse la aplicación y ya puede comenzar a disfrutar de ella. Una vez la aplicación esté iniciada debería aparecer una pantalla como la siguiente imagen:



## Anexo 2. Tarea de evaluación con usuarios

Este documento sirve como guía para la realización de pruebas por parte de usuarios para la aplicación de Alexa *Aprendamos Algoritmos*. A disposición de los usuarios está el Anexo 1, donde se explica cómo poder acceder e instalar las distintas aplicaciones necesarias para la realización de estas pruebas. Una vez se haya accedido a la aplicación de Alexa e instalado la aplicación Android puede comenzar la realización de las pruebas detalladas que se detallan a continuación.

### Pruebas a realizar:

Una vez estén abiertas las aplicaciones de Alexa y de Android es momento de comenzar con las pruebas. Para ello se pide:

- Introducir una clave en la aplicación para dispositivos móviles, pulsar sobre el botón “Enviar clave” y aceptar.
- En la ventana del navegador donde está iniciado Alexa, introducir en el campo de texto que hay disponible “abre aprendamos algoritmos” para poder iniciar la aplicación.
- Una vez iniciada la aplicación, el usuario debe ir navegando por la aplicación como más lo desee durante un máximo de 30 minutos. Los objetivos a cumplir por el usuario son:
  - Conectar la aplicación de Alexa con la aplicación Android mediante la misma clave.
  - Escuchar la explicación sobre la resolución de un problema Dijkstra.
  - Intentar resolver un problema planteado por la aplicación del nivel que más desee.
  - Ver el algoritmo de Dijkstra.
- Una vez hayan pasado los 30 minutos se debe resolver el cuestionario mostrado a continuación.

### Cuestionario.

Para acceder al cuestionario se debe pulsar en el siguiente enlace que llevará hasta a él. Se debe contestar con total sinceridad.

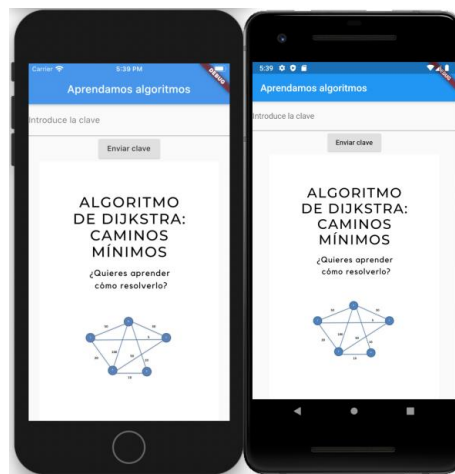
[https://docs.google.com/forms/d/e/1FAIpQLScykIskY\\_dt6orxdTfXEjT8KKwo89Ozv6kQdQObHJ6SWKGDmQ/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScykIskY_dt6orxdTfXEjT8KKwo89Ozv6kQdQObHJ6SWKGDmQ/viewform?usp=sf_link)

## Anexo 3. Guía de uso

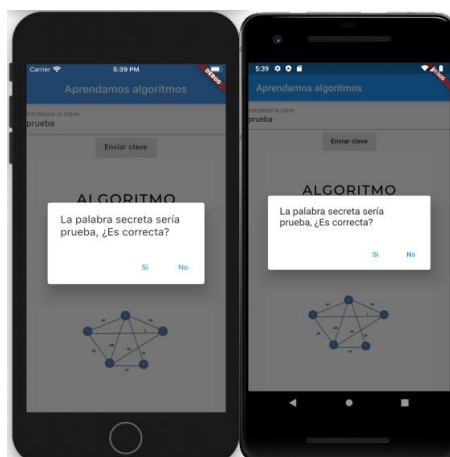
En este anexo se describirá un manual de usuario para la utilización de las aplicaciones desarrolladas para este proyecto.

Para comenzar a utilizar las aplicaciones, primero se debe acceder a ellas. Para el acceso a la aplicación de Alexa e instalación de la aplicación Android, se pone a disposición del usuario el Anexo 1 donde se detalla cómo realizar estas acciones.

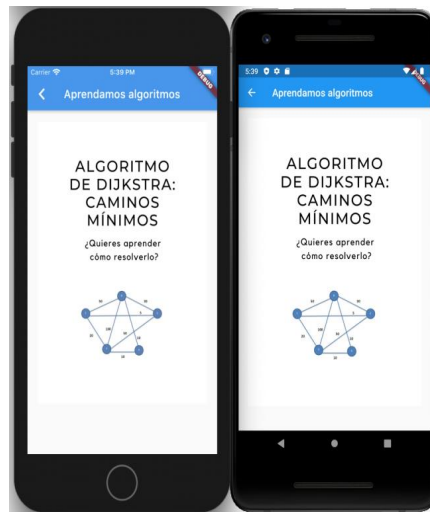
Una vez se tenga instalada la aplicación móvil en el dispositivo deseado, el siguiente paso sería abrirla. Una vez abierta debería aparecer una pantalla como la mostrada a continuación tanto para dispositivos Android como dispositivos con sistema operativo iOS



Una vez abierta la aplicación, el siguiente paso es introducir una clave para permitir la comunicación con la aplicación de Alexa. Para poder realizar esta acción, el usuario debe introducir en el campo habilitado para ello, la palabra que desee emplear como clave. Una vez introducida la clave, el usuario debe pulsar en el botón cuyo texto es “Enviar clave” y aparecerá un mensaje preguntándole si la palabra introducida es la correcta. Este mensaje será como el de la siguiente imagen:



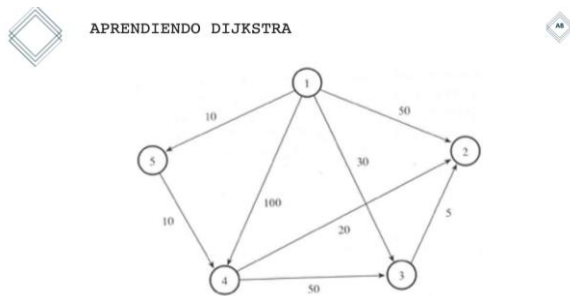
Para confirmar la clave, el usuario debe pulsar sobre la opción “Sí” y para volver a introducir otra clave, el usuario debe pulsar sobre “No”. Si se pulsa sobre la opción “No”, volverá a la pantalla inicial y si pulsa sobre la opción “Sí”, la aplicación enseñará otro mensaje indicándole al usuario cual ha sido la palabra confirmada. Además de esto, mostrará un botón el cual le llevará a una pantalla en la que el usuario ya no tendrá que utilizar más la aplicación debido a que ya se irán mostrando las imágenes automáticamente. Esta pantalla es como la mostrada en la siguiente imagen:



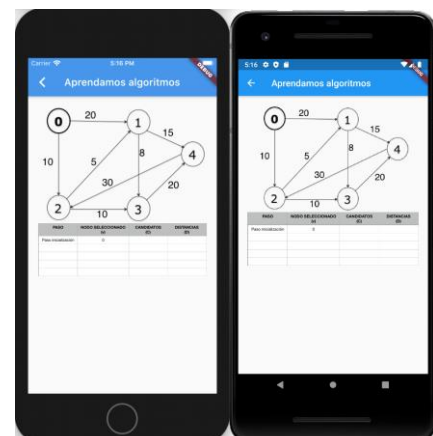
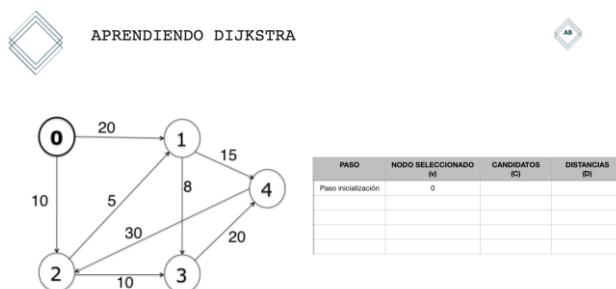
Una vez disponible la aplicación móvil, es momento de utilizar la aplicación de Alexa. Para abrir la aplicación y por tanto comenzar con su utilización, el usuario debe decir “**Alexa, abre aprendamos algoritmos**”. Una vez dicho esto, La aplicación comenzará con el mensaje de “*Hola y bienvenido a aprendamos algoritmos. La primera pregunta que debo hacerte es si deseas continuar con el progreso llevado anteriormente o no. Para contestarme debes decir sí o no.*” Si se desea comenzar con un nuevo progreso la respuesta debe ser “**No**”, a lo que Alexa contestará con “*Necesito que me digas una contraseña para poder comunicarme con la aplicación móvil. Para ello debes decir, la contraseña es, seguido de tu contraseña.*”. En este momento, el usuario debe responder con la contraseña que haya introducido en la aplicación móvil. En el caso de este ejemplo, la contraseña es “prueba” por lo que la respuesta del usuario debe ser “**la contraseña es prueba**”. Ante esta frase, la respuesta de Alexa es “*Se ha almacenado prueba como contraseña, a continuación paso a describir la aplicación en qué consiste. Bienvenido a Aprendamos algoritmos, tu asistente para aprender a resolver un algoritmo Dijkstra. Si aún no sabes en qué consiste y te gustaría aprender a resolverlo prueba a decir. Comenzar explicación. En cambio, si ya eres conocedor de este algoritmo y te gustaría ponerte a prueba e intentar resolver un ejemplo, prueba a decir. Resolver problema. En cualquier momento puedes decir: "muestra el algoritmo" y se presentará el algoritmo en pseudocódigo. Además, puedes decir "ver créditos" y aparecerá el autor de la skill.*” En este



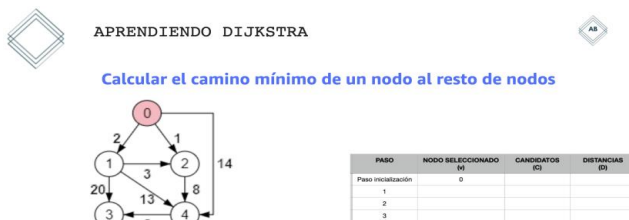
momento, las imágenes mostradas en los distintos dispositivos serían como las mostradas a continuación.



Una vez dentro de la aplicación, si el usuario desea navegar hacia la explicación del problema, debe decir **“comenzar explicación”** a lo que Alexa responderá con *“El problema consiste en calcular el camino mínimo de un nodo al resto de nodos del grafo. En este caso de ejemplo, el nodo del que partimos es el nodo 0, así que debemos calcular las distancias entre el nodo 0 y el resto de nodos. En cada paso debemos ir calculando los caminos mínimos parcialmente. Para ello vamos marcando el conjunto de nodos candidatos, que son aquellos nodos de los que aún no hemos partido, y las distancias desde el nodo 0 al resto de nodos. Si desea volver a escuchar esta explicación diga, repetir, y en caso de querer continuar con la explicación, diga, siguiente.”*. Para ir navegando por las distintas pantallas, el usuario, únicamente deberá decir **“siguiente”** y Alexa le irá dando una explicación específica al paso en el que se encuentre el usuario. Si el usuario se encuentra en otro apartado de la aplicación y desea volver a la pantalla de la explicación en la que se encontraba, debe decir **“volver a la explicación”**. En los diferentes dispositivos, las imágenes mostradas saldrán como las mostradas a continuación.



Si en cualquier momento el usuario desea comenzar a ponerse a prueba y resolver un problema, debe decir **“resolver problema”**. A dicha frase, Alexa contestará diciendo *“Como dispongo de diferentes niveles de problemas, tienes que decir si deseas el nivel medio o el nivel difícil diciendo. Quiero el nivel medio o quiero el nivel difícil”*. El usuario ante esta frase, debe elegir un nivel y decir *“Quiero el nivel medio”*. Alexa a esa frase del usuario contestará con *“Este es el enunciado del problema de nivel medio. ¿Cuál sería el conjunto de candidatos para el paso de inicialización de este nodo? Para dar una respuesta debes decir la solución es, seguida de los nodos de solución”* y el usuario deberá ir respondiendo a las preguntas que le vaya realizando Alexa a lo largo de todo el problema. Para contestar una pregunta, en el caso de ejemplo nos ha salido un nodo de nivel medio cuyo conjunto de candidatos es uno, dos, tres y cuatro, el usuario para responder correctamente debe decir **“la solución es uno dos tres cuatro”** y Alexa cambiará de pantalla, le mostrará un mensaje de enhorabuena y realizará una nueva pregunta. En el caso de que el usuario responda erróneamente, la pantalla no se modificará, Alexa le dará algunos consejos y le realizará de nuevo la pregunta hecha con anterioridad. Si el usuario se encuentra en algún momento en otro apartado de la aplicación y desea volver al progreso que llevaba con anterioridad, debe decir **“volver a la resolución”**. Un ejemplo de cómo se vería la primera pantalla de resolución del problema sería como la siguiente imagen:

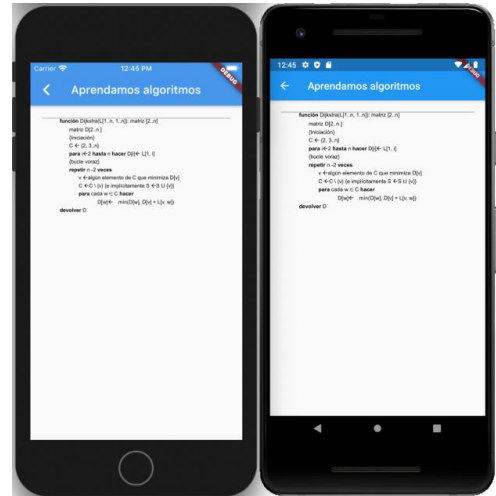


Si el usuario desea ver el algoritmo de Dijkstra en pseudocódigo, debe decir “**ver algoritmo**”, a lo que Alexa responderá diciendo “*En la pantalla se muestra el algoritmo de Dijkstra en pseudocódigo*” y mostrará las siguientes imágenes en los distintos dispositivos:

ALGORITMO DE DIJKSTRA

```

función Dijkstra(L[1..n, 1..n]: matriz [2..n]
    matriz D[2..n]
    {Iniciación}
    C ← {2, 3..n}
    para i ← 2 hasta n hacer D[i] ← L[1, i]
    {bucle voraz}
    repetir n - 2 veces
        v ← algún elemento de C que minimiza D[v]
        C ← C \ {v} (e implícitamente S ← S ∪ {v})
        para cada w ∈ C hacer
            D[w] ← min(D[w], D[v] + L[v, w])
    devolver D
  
```



Por último, si el usuario desea ver quien ha sido el desarrollador de la aplicación y su contacto, debe decir “**ver créditos**”. Alexa, ante esta frase responderá con: “*Esta skill ha sido desarrollada por Alberto Barragán Costo*” y en las distintas pantallas mostrará unas imágenes como las mostradas a continuación:

CREDITOS

Creado por:  
**Alberto Barragán Costo**  
**abarragancosto@gmail.com**

