Trabajo Fin de Grado

Título del trabajo: Desarrollo de experiencias lúdicas destinadas a personas mayores y grupos intergeneracionales para asistentes por voz con salida gráfica

English tittle: Development of recreational experiences for elderly people and intergenerational groups for voice assistants with graphic output

Autor

Alberto Lardiés Getán

Directora

Eva Mónica Cerezo Bagdasari

Titulación del autor

Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura

2024

AGRADECIMIENTOS

A mi directora Eva Cerezo, por su gran labor de ayuda y orientación, y su infinita paciencia, durante todo el proceso de elaboración de este proyecto.

A mis amigos, por haber estado a mi lado, apoyándome, siempre con palabras de ánimo, y ofreciendo su ayuda incansablemente a pesar de mis continuas negativas.

A mi familia por ser siempre optimistas, motivándome a seguir adelante. En especial a mis padres, por estar siempre apoyándome hasta en los momentos más duros del proyecto. Sin ellos no lo habría logrado.

Desarrollo de experiencias lúdicas destinadas a personas mayores y grupos intergeneracionales para asistentes por voz con salida gráfica

RESUMEN

A pesar de que hoy en día existen muchas aplicaciones dedicadas al entretenimiento, existe una brecha generacional que favorece que las personas mayores no suelan ser el foco de atención a la hora de desarrollar estas aplicaciones, alejándolos de los más pequeños, que crecen con este mundo tecnológico alrededor.

El objetivo de este trabajo, aprovechando el auge de los asistentes por voz con salida gráfica, ha sido el desarrollado de una aplicación para personas mayores y grupos intergeneracionales, formados por personas mayores y niños, que les proporcione un medio divertido de interactuar con el asistente por voz y entre ellos. Al tener salida gráfica táctil ofrece también un apoyo visual interactivo que mejora la experiencia de los usuarios, ofreciendo reacciones visuales a sus interacciones con la aplicación.

Después de realizar un análisis previo de aplicaciones similares y de objetivos del trabajo, se desarrolló uno de los juegos que ofrece la aplicación, cuyo diseño proviene de un TFM de Diseño de Producto, con interacción únicamente por voz, que luego se rediseñó e implementó para incluir interacción táctil.

Más adelante se diseñó e implementó un catálogo con tres juegos, uno de ellos el anterior, permitiendo al usuario elegir el modo de interacción con la aplicación entre táctil, por voz o ambos. Además, la aplicación guarda las partidas no terminadas, permitiendo al usuario continuarlas en otro momento.

Adicionalmente, se llevó a cabo una evaluación con usuarios con el objetivo de comprobar el funcionamiento de la aplicación, la comprensión de las directrices dadas por el asistente y posibles aspectos a mejorar para enriquecer la experiencia del usuario.

Development of recreational experiences for elderly people and intergenerational groups for voice assistants with graphic output

ABSTRACT

Although nowadays there are many applications dedicated to entertainment, there is a generational gap that often results in older adults not being the focus when developing these applications, distancing them from younger generations who grow up surrounded by technology.

The objective of this project, taking advantage of the rise of voice assistants with graphical output, was to develop an application for older adults and intergenerational groups, consisting of both seniors and children, that provides a fun way for them to interact with the voice assistant and with each other. With a touch graphical interface, the application also offers interactive visual support that enhances the user experience by providing visual reactions to their interactions with the application.

After conducting a prior analysis of similar applications and the project's objectives, one of the games offered by the application was developed. The design of this game originates from a Master's thesis in Product Design and initially involved only voice interaction, which was later redesigned and implemented to include touch interaction.

Later, a catalog of three games was designed and implemented, one of them being the previously mentioned game, allowing the user to choose their preferred interaction mode with the application (touch, voice, or both). Additionally, the application saves unfinished sessions, allowing the user to continue playing at a later time.

Furthermore, an evaluation with users was conducted to test the application's functionality, the understanding of the assistant's instructions, and to identify potential areas for improvement to enhance the user experience.

Índice

1. Introducción

En este capítulo se habla del contexto del trabajo, los objetivos que se quieren alcanzar y el contenido de este documento.

1.1. Contexto

En la sociedad actual, las tecnologías interactivas basadas en asistentes virtuales como Alexa han ganado popularidad, no solo aplicadas a hogares inteligentes, sino también en el desarrollo de aplicaciones de entretenimiento, educación y bienestar [1]. Sin embargo, a pesar del auge de estas tecnologías, sigue existiendo una importante brecha en el desarrollo de aplicaciones digitales inclusivas y accesibles para las personas mayores, quienes pueden encontrar dificultades para interaccionar con dispositivos complejos debido a posibles limitaciones físicas, cognitivas o tecnológicas[2].

Por otro lado, los juegos son un medio efectivo para mejorar la coordinación, la memoria y las habilidades cognitivas de estas personas [3], creando también una vía para la interacción social, ya sea con otras personas mayores o con niños. El proyecto busca aprovechar la popularidad de dispositivos de interacción con Alexa de Amazon, donde algunos modelos ya tienen pantalla táctil incorporada, para crear una aplicación que permita combinar entretenimiento y funcionalidad, que facilite la interacción intergeneracional.

1.2. Objetivos

El objetivo principal de este trabajo es el desarrollo de una aplicación para dispositivos Alexa con pantalla, dirigida a personas mayores y grupos intergeneracionales formados por mayores y niños, que ofrezca un catálogo de juegos interactivos con instrucciones claras y fáciles de usar. A través de esta aplicación, se busca promover la interacción y la diversión entre los distintos grupos de edad, al mismo tiempo que se pone a prueba la capacidad de los dispositivos Alexa con pantallas táctiles para manejar aplicaciones con una interfaz sencilla y accesible.

2. Análisis previo

Se ha realizado un estudio previo sobre los dispositivos Amazon con Alexa disponibles. En este capítulo también se detalla el público objetivo de la aplicación.

2.1. Dispositivos de Amazon con Alexa

Amazon ofrece una variedad de dispositivos integrados con Alexa y pantalla que proporcionan diversas funcionalidades y capacidades [6]. Aquí se detallan algunos de los dispositivos clave y sus características:

- Echo Pop: es el altavoz inteligente más compacto de Amazon. Ideal para habitaciones pequeñas, puesto que va perdiendo calidad de audio al subir el volumen. El micrófono no es muy bueno y cuando el volumen es alto no oye los comandos de voz.

- Echo Dot: es un dispositivo pequeño, anterior al Echo Pop, pero con mejor calidad de audio. Incluye un botón de acción, que sirve para "despertar" a Alexa si no te oye.

- Echo Show: hay varias generaciones de estos dispositivos, pero su principal cambio con respecto a los anteriores es que llevan una pantalla táctil integrada para mostrar información visual con sus respuestas, reproducir vídeos, hacer videollamadas. La calidad de audio mejora notablemente.

- Echo Spot: es una versión pequeña del Echo Show, con forma redonda y una pantalla circular o semicircular, según la versión. Está pensado principalmente para comprobar la previsión del tiempo, poner alarmas, titulares de noticias.

- Fire HD 8 y 10: estos dispositivos son tablets con Alexa integrada. A diferencia de otras tablets de Amazon, estos modelos tienen un modo "Show" (que viene de los dispositivos Echo Show), lo que les permite actuar como un Echo Show. Una diferencia con los Echo Show es una menor calidad de audio, puesto que su finalidad es distinta.
- Fire TV: estos son dispositivos de streaming para televisores. Permiten también descargar aplicaciones y juegos.

Se descartan el Echo Pop y Echo Dot al no tener pantalla, pues es uno de los requisitos de esta aplicación. El Echo Spot también puesto que la pantalla es muy pequeña para poder incluir toda la información que requiere una aplicación lúdica como la que se plantea.

Los Fire TV tampoco son aptos puesto que los televisores son dispositivos fijos, con los que se interactúa por voz algunas veces pulsando un botón en un mando y no incluyen interacción táctil.

Teniendo en cuenta las posibilidades de cada uno, la aplicación se ha desarrollado para los dispositivos Echo Show y Fire HD. Su característica principal es que tienen pantalla táctil, permitiendo mostrar contenido visual para enriquecer la aplicación, además de incluir la posibilidad de interacción táctil.

2.2. Público objetivo

El público objetivo principal de esta aplicación son las personas mayores. Estas personas pueden hacer uso de la aplicación individualmente o acompañadas, si bien la compañía puede ser de una edad similar o preferiblemente niños.

Concretamente se encuentran los siguientes perfiles:

- Personas mayores de 65 años que no tengan problemas cognitivos y puedan interactuar con el dispositivo, ya sea por voz con una buena vocalización o de manera táctil con los botones.

- Los niños deben ser mayores de 4 años y, al igual que el grupo anterior, han de poder interaccionar vía voz o de forma táctil con el dispositivo. En todo caso jugarán en compañía del adulto mayor.

Son evidentes los beneficios que existen en las relaciones entre los jugadores de este tipo de juegos ya que, aunque puedan no conocerse previamente, su nivel de interacción aumenta, lo que se hace más evidente cuando logran entender bien el juego [16] [17]. Estos juegos son una forma de fomentar la conversación entre personas mayores y niños [18], y de aumentar el entendimiento mutuo más allá del propio juego [19].

2.3. Análisis de aplicaciones similares

En este apartado, en la tabla 1, se detallan las aplicaciones más relevantes para este trabajo encontradas en la tienda de Amazon [7]:

Nombre	**Dispositivo**	**Características**	**Tipo de Juego**	**Soporte táctil**
Akinator	Alexa	El usuario piensa en una persona o personaje y Alexa, a través de preguntas, intenta adivinar de quién se trata.	Preguntas y adivinanza.	No
Reto memoria	Alexa	Es un juego de memoria donde el usuario tiene que repetir la lista de colores que diga Alexa. Cada vez que acierta se añade un color más.	Memoria	No
Simón dice	Alexa	Juego en solitario o por equipos. Si Alexa dice “Simón dice”, hay que realizar la acción que diga. El que lo haga sin oír “Simón dice” queda eliminado.	Acción y concentración	Sí
Desafío diario	Alexa	Alexa te hace una pregunta una vez al día, acumulando dinero ficticio si se responde correctamente.	Preguntas	No
Quién es quién	Alexa	El jugador debe adivinar el personaje		No

Tabla 1. Análisis de aplicaciones existentes.

- Desafío diario: es una aplicación simple. Hace una pregunta por día con tres opciones. Si el jugador acierta o falla, gana o pierde dinero ficticio, respectivamente. La interfaz es solo visual, puesto que no permite seleccionar de manera táctil las opciones mostradas, solo por voz.

- Simón dice: Es el primer juego de Alexa encontrado que tiene incluye interacción táctil, aunque sea solo al elegir la categoría sobre la que Alexa dará las órdenes.

- Quién es quién: El mítico juego en el que lose jugadores (en este caso Alexa y un usuario) ven una serie de rostros de personas y deben hacer preguntas alternadas hasta que uno acierte el personaje del otro. No tiene ninguna opción táctil, ni para descartar personajes ni para elegir el correcto.

- Reto memoria: Un juego de repetir una secuencia de colores. No tiene opción táctil, a pesar de que se podrían mostrar botones de los colores que forman parte de la cadena para que el usuario los pulse en orden.

Aunque hay muchas aplicaciones para Alexa, las comentadas han sido seleccionadas por su similitud en cuanto a objetivos y público con la aplicación a desarrollar en este proyecto. Como se puede observar, son pocas las que cuentan con soporte táctil, y además poco desarrollado.

3. Características del sistema

En este capítulo se presenta la arquitectura del sistema, incluyendo las tecnologías utilizadas y los requisitos de la aplicación.

3.1. Arquitectura del sistema

Cuando se envía un comando de voz a un dispositivo Alexa, se dan una serie de sucesos que terminan con el dispositivo pronunciando un mensaje y, si tiene pantalla, mostrando una salida gráfica. En la figura 1 se muestra un esquema de los elementos que actúan es ese proceso.

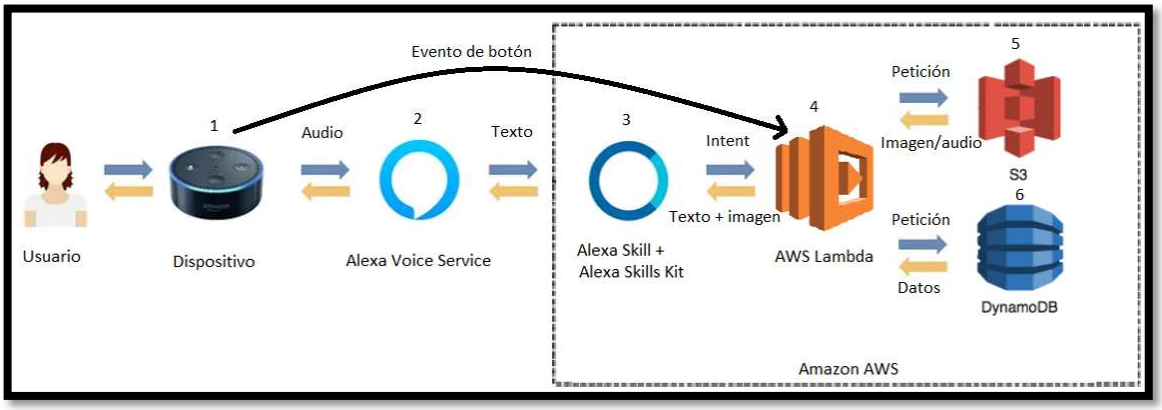


Figura 1. Esquema de interacción de un usuario con Alexa.

En este proceso intervienen los siguientes componentes:

1 Dispositivo Alexa Echo Show/Fire HD: Es el dispositivo físico, que cuenta con micrófonos, altavoces y pantalla táctil, con el que interactúa el usuario y que muestra la respuesta visual. Recibe el comando de voz del usuario y lo envía a AVS para su tratamiento. Si por el contrario el usuario pulsa un botón que genera un evento, se envía dicho evento directamente a Lambda.

2 Alexa Voice Service (AVS) [8]: Es un servicio en la nube que recibe el comando de audio del usuario y lo convierte a texto para enviarlo a la skill.

3 Alexa Skills Kit (ASK) [9]: Kit de desarrollo de Amazon que permite a los usuarios o desarrolladores crear skills para Alexa y definir los intents y slots. Además, se encarga de enviar las solicitudes a Lambda, diciéndole a qué intent corresponde el comando del usuario.

4 AWS Lambda [10] (Backend de la Skill): Plataforma sin servidor donde se gestiona la lógica de la skill. Recibe un intent, o un evento, y ejecuta el handler correspondiente, accediendo si es necesario a servicios de almacenamiento como DynamoDB [11] o S3 [12].

5 Simple Storage Service (S3): se usa para almacenar archivos estáticos (imágenes, audios, etc.).

6 DynamoDB: se usa para almacenar datos estructurados (información del usuario, de la partida, etc.).

7 Alexa Presentation Language (APL) [13]: Es el lenguaje utilizado para diseñar interfaces visuales en dispositivos con pantalla. Permite decidir lo que se mostrará en pantalla (texto, imágenes, gráficos, botones, etc.) como reacción a las acciones del usuario en dispositivos como el Echo Show o Fire HD.

3.2. Tecnologías Utilizadas

Para el desarrollo de esta aplicación se han empleado principalmente tecnologías y herramientas propias de Amazon puesto que ofrece su propia plataforma y recursos de programación para dispositivos Alexa. Las tecnologías principales utilizadas son:

- Alexa-hosted skill [14]. Una skill Alexa-hosted es una aplicación cuyo backend y gestión de recursos está gestionado por Amazon. Concretamente el servicio AWS Lambda permite ejecutar código sin tener que administrar servidores, ejecutando el código en una infraestructura de computación altamente disponible y realiza las tareas de mantenimiento del servidor y el sistema operativo, la escalabilidad, etc. El nivel gratuito de Amazon permite llamadas ilimitadas a Lambda.

  Esto permite construir, programar, desplegar y testear una skill desde la consola de desarrollador de Amazon sin tener que gestionar recursos externos. Proporciona además almacenamiento gratis limitado en Amazon S3 y Amazon DynamoDB.

- Alexa Developer Console [15]: Una plataforma de Amazon que proporciona una infraestructura para la creación y prueba de skills de Alexa, que incluye la configuración de intents, slots, la lógica de la aplicación y las interfaces. Usada también para diseño de interfaces.

- Node.js y JSON: Lenguajes para la programación de la aplicación elegidos por la experiencia previa en su uso para programación. Node.js para la lógica del juego y JSON para las interfaces.

- AWS Lambda: un servicio informático de Amazon Web Services sin servidor que ejecuta el código en respuesta a los eventos y administra los recursos informáticos.

- S3 y DynamoDB para almacenamiento. Se explicará en profundidad en el capítulo 5.1.1.

- Testing con Dispositivos Reales: Para las pruebas durante el proceso de implementación se ha contado con un dispositivo Alexa Echo Show 8, un asistente por voz de Amazon con pantalla incorporada, para asegurar el correcto funcionamiento de la aplicación en el hardware objetivo.

3.3. Requisitos

A partir del estudio realizado previamente sobre aplicaciones similares y sobre los usuarios objetivo, se definen los requisitos que la aplicación ha de cumplir. Dichos requisitos, tanto funcionales como no funcionales, se muestran en la Tabla 2.

Como la aplicación contendrá diversos juegos no será necesario que ºs los juegos hagan uso de todas las funcionalidades, pero se habrá de asegurar que todas ellas se usen en algún juego o etapa de juego.

Requisito	Descripción
RF1	El sistema pide al usuario identificarse al empezar
RF2	El sistema debe permitir al usuario elegir el modelo de interacción: por voz, táctil, o completo (ambos)
RF2.1	El sistema, en el modo táctil, debe permitir al usuario interactuar con la aplicación por medio de botones sin necesidad de hablar en ningún momento
RF2.2	El sistema, en el modo por voz, debe permitir al usuario interactuar con la aplicación por medio de comandos de voz sin necesidad de tocar la pantalla en ningún momento
RF3	El sistema debe permitir al usuario pedir al dispositivo que repita las últimas instrucciones
RF4	El sistema debe soportar tanto el juego individual como grupal
RF5	El sistema debe permitir al usuario volver a alguna fase anterior de un juego
RF6	El sistema debe permitir al usuario en todo momento salir de la partida
RF7	El sistema debe permitir guardar el progreso de una partida de un usuario al salir en mitad de una partida
RF8	El sistema debe ser capaz de guardar estadísticas de un juego
RF9	El sistema debe ser capaz de mostrar las estadísticas del juego durante la partida
RNF1	La aplicación debe funcionar en dispositivos Echo Show (1, 2, 5, 8, …) y tablets Fire de Amazon
RNF2	La aplicación necesita conexión a internet
RNF3	El sistema utiliza los servicios de almacenamiento S3 de Amazon para alojar las imágenes y los archivos de audio que necesita la aplicación
RNF4	El sistema utiliza los servicios de base de datos DynamoDB de Amazon para almacenar los datos de las partidas guardadas

Tabla 2. Tabla de requisitos.

4. Diseño de la aplicación

En este capítulo se va a hablar del diseño de la aplicación y las decisiones más importantes que se tomaron.

La aplicación se plantea como un catálogo de juegos orientados a personas mayores y grupos intergeneracionales para su juego individual o en grupo.

El primero de los juegos es el juego "Encontremos el mapa". El trabajo en ese juego constituyó la primera parte del proyecto y partió de un diseño de juego intergeneracional llevado a cabo en el marco de un Trabajo de Fin de Máster de diseño [5]. A partir del diseño inicial conceptual del juego y de los recursos gráficos creados en dicho TFM se hizo un rediseño del mismo para tener en cuenta las restricciones técnicas impuestas por el dispositivo y para transformarlo en un juego que permitiera no solo la interacción por voz sino también táctil, de cara a aumentar las posibilidades de interacción y la accesibilidad del juego.

Además de ese juego se decidió añadir dos juegos más que permitieran explorar otras formas de juego y funcionalidades no contempladas en el juego de "Encontremos el mapa". Los otros dos juegos son “Encontrar las parejas” y “Adivinar el deporte”, dos juegos muy visuales e intuitivos. Se hablará más en detalle de cada uno en este capítulo.

4.1. Decisiones generales de diseño

Las interfaces serán sencillas, evitando sobrecargar la pantalla de información o estímulos irrelevantes que distraigan al usuario, ya se ha observado que una de las razones por las que las personas mayores no hacen uso de alguna función en Alexa es el desconocimiento de su existencia [4].

Para ello Se quiere hacer que Alexa, al pasar a una nueva pantalla, explique las posibles vías de acción, que serían los comandos posibles y los botones disponibles. Además, se añadirá un botón de ayuda que permita al usuario escuchar de nuevo las instrucciones para el juego, o fase de un juego, donde se encuentre, evitando así que se quede bloqueado. Este botón no tendrá una posición fija porque se considera que reservar un espacio común en todas las interfaces afectaría en gran medida al espacio utilizable por el resto de los elementos visuales de cada juego. Este botón será visible y de un color llamativo para facilitar su localización en las distintas interfaces.

En base al diseño de los distintos juegos, y para mantener una estructura uniforme, el botón que permita salir de un juego o de la aplicación se va a encontrar en la esquina superior izquierda, a la misma altura que el título de la pantalla. Esto se debe a que las dos esquinas superiores son las únicas zonas que están libres en todas las pantallas de los distintos juegos, como se verá a continuación en las imágenes del diseño. Los juegos necesitan bastante espacio para que las imágenes y los textos puedan tener un buen tamaño, dejando poco espacio disponible en pantalla.

4.2. Diseño del catálogo

La primera pantalla de la aplicación pedirá al usuario por voz y en pantalla que le proporcione su nombre. Se incluye el botón de ayuda en caso de que no se entienda la pregunta. Se puede ver esta interfaz en la figura 2.

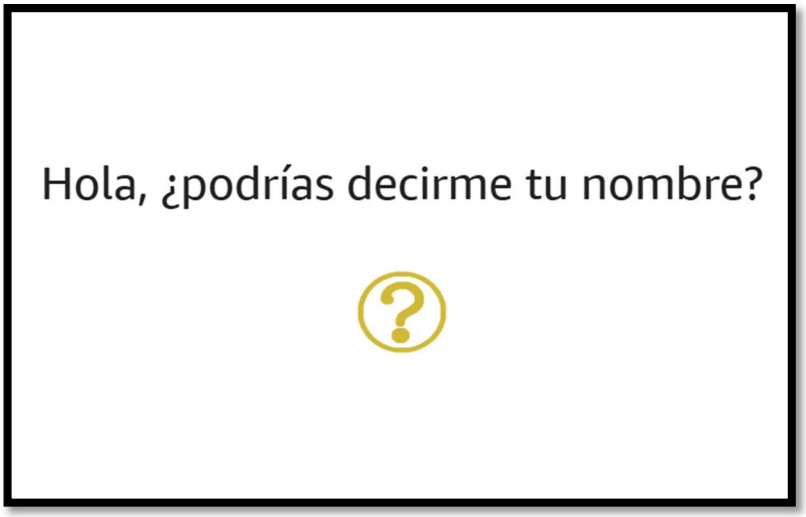


Figura 2. Interfaz de identificación del usuario.

En la siguiente pantalla se preguntará al usuario cómo quiere interactuar con la aplicación. Se ofrecen tres opciones: solo por voz, solo de forma táctil, o ambas. Al igual que en la anterior, el botón de ayuda deberá repetir la pregunta. No se incluye botón de volver porque, en caso de querer cambiar de usuario, se puede salir de la aplicación. En la figura 3 se muestra la interfaz de la elección del modo de interacción.

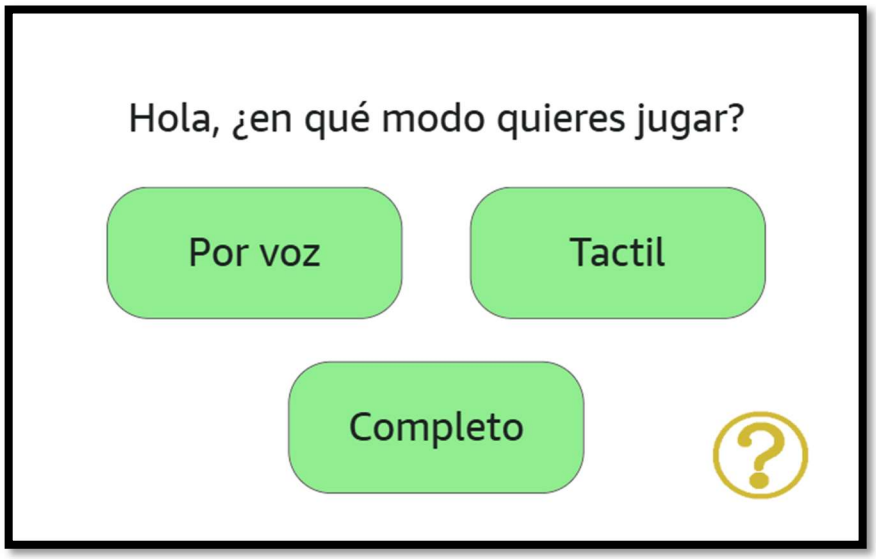


Figura 3. Interfaz de elección del modo de juego.

Una vez elegido el modo de interacción se pasará a la interfaz principal de la aplicación, el catálogo. Como se ve en la figura 4, se han dispuesto tres imágenes, con la intención de que hagan de botones, con una imagen de una partida de cada juego y un título en la parte superior para identificarlos. Se ha aprovechado el espacio en pantalla para permitir que los botones sean grandes y fáciles de pulsar a la hora de elegir el juego.

Figura 4. Interfaz del catálogo.

La última interfaz antes de llegar a los juegos será la que pregunte, en caso de haber una partida empezada, si el usuario quiere continuarla o, por el contrario, empezar una nueva, perdiendo los datos de la partida anterior. En la figura 5 se puede ver que otra vez se simplifica la interfaz para que el mensaje sea directo, continuar o no. Botones grandes y separados para evitar que el usuario pulse el que no quiera.

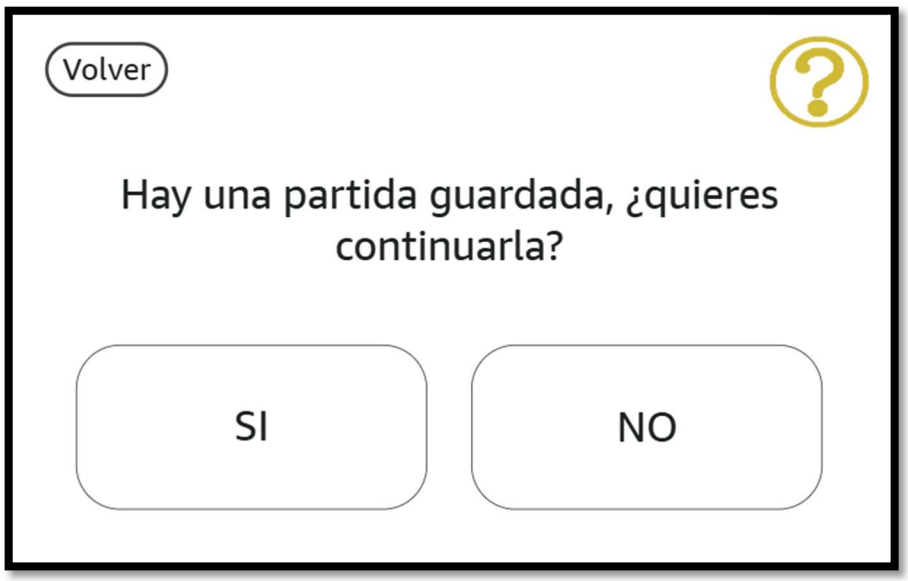


Figura 5. Interfaz de carga de partida guardada.

4.3. Diseño de “Encontremos el mapa”

Partiendo de las interfaces implementadas inicialmente de este juego, correspondientes al diseño de Lucía [5], que se irán viendo a lo largo de este capítulo, se han hecho modificaciones principalmente para añadir interacción táctil, además de ampliar el tamaño de letra de los textos y las imágenes mostrados en pantalla teniendo en cuenta el público al que va dirigido.

Se añade el botón “Continuar” en el lado opuesto al botón de volver mencionado en el capítulo 4.1, en la parte superior derecha, al lado del título, manteniendo así un diseño homogéneo.

Se añade además la posibilidad de cargar una partida anterior de este juego si no se había terminado, dando la opción al usuario de iniciar una partida nueva, borrando los datos de la anterior, o continuarla desde donde se quedase.

La interfaz inicial del juego que se puede ver en la figura 6 estaba pensada para un juego que era en sí una aplicación, y además solo por voz. Por ello solo se han añadido los tres botones mencionados en capítulos anteriores, como se muestra en la figura 7.

Figura 6. Interfaz inicial de bienvenida de “Encontremos el Mapa”.

Figura 7. Interfaz final de bienvenida de “Encontremos el Mapa”.

4.3.1. Encuentra el trozo de mapa

En esta fase del juego se deben identificar mediante descripciones, las imágenes correctas del mapa. Partiendo del diseño inicial (figura 8) y las decisiones generales de diseño, los cambios más importantes son:

- Se ha cambiado el icono de repetir, por el incono de ayuda, que es el que estará en todas las interfaces.
- Se han eliminado los textos con los comandos para elegir cada imagen y pedir que se repita la descripción.
- En caso de jugar en grupos, muestra el nombre del grupo al que le toca.
- Se han numerado en grande las cuatro imágenes para marcar su orden.

Estos cambios se pueden ver en la figura 9.

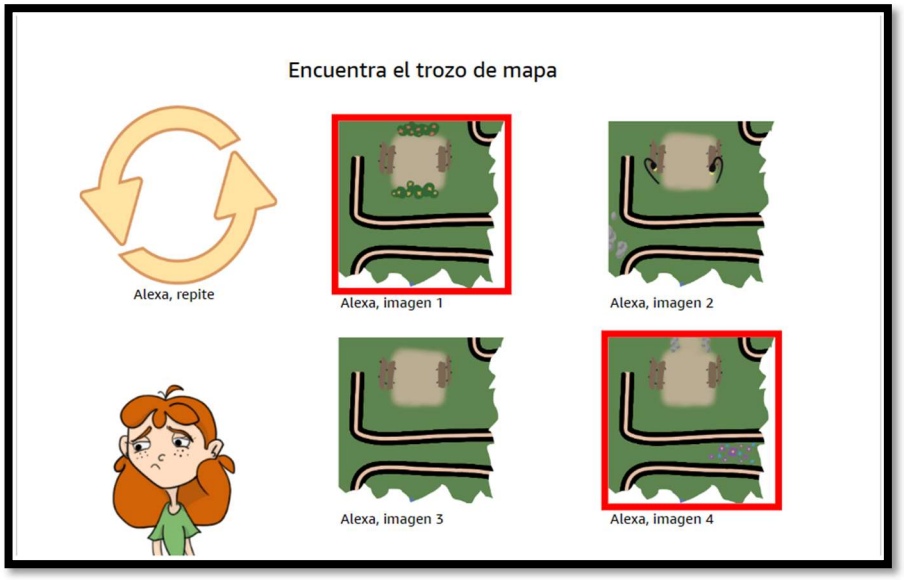


Figura 8. Interfaz inicial de “Encuentra el trozo de mapa”.

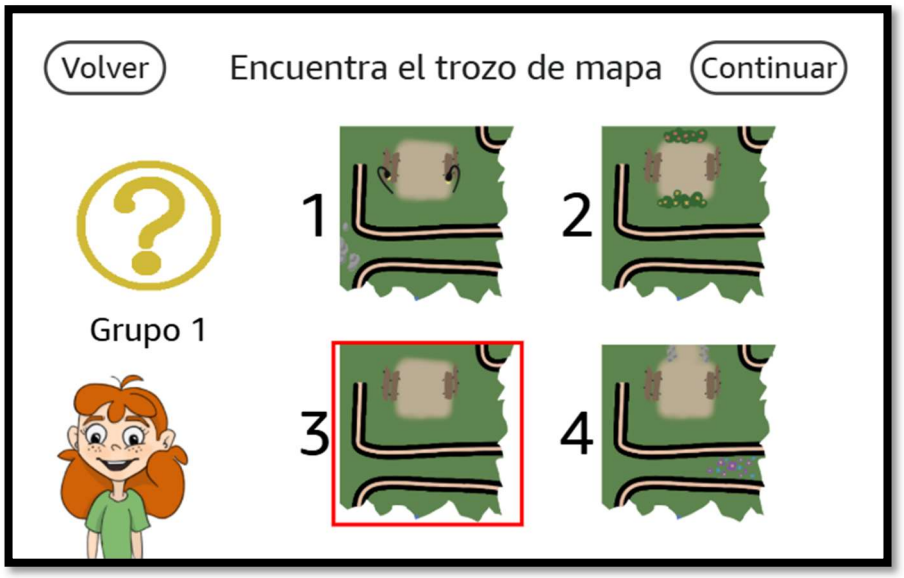


Figura 9. Interfaz final de “Encuentra el trozo de mapa”.

4.3.2. Parque

Esta fase es una introducción al segundo minijuego, donde un niño en el parque tiene los trozos de mapa que nos faltan y el usuario debe ayudarle con sus deberes para que se los devuelva. Al igual que en la anterior, el único cambio entre el diseño inicial y final, mostrados en la figura 10 y la figura 11 respectivamente, es que se han añadido los mismos tres botones.

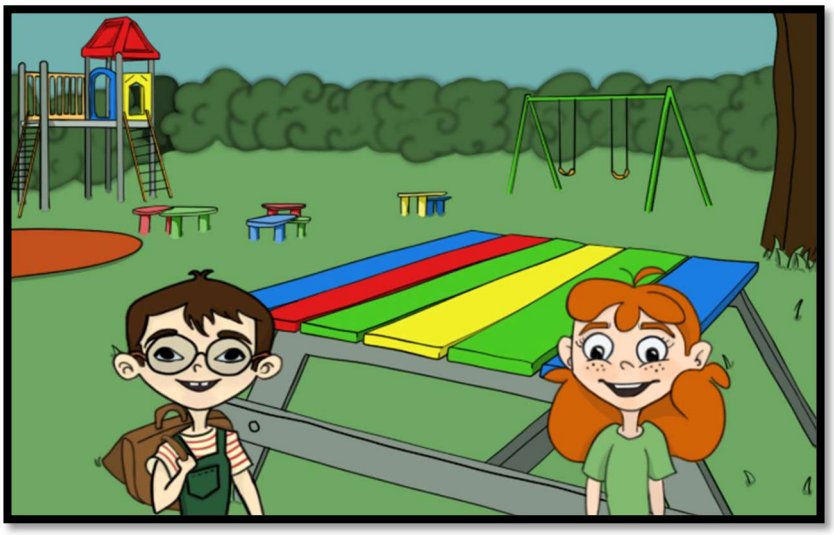


Figura 10. Interfaz inicial del parque.

Figura 11. Interfaz final del parque.

4.3.3. Ayuda al chico con los deberes

Este es el segundo minijuego de “Encontremos el mapa”. Hay que completar cuatro de los cinco refranes que aparecen en la interfaz para que el niño nos devuelva los trozos de mapa.

En este caso, al igual que en el anterior minijuego, se añade el texto con el grupo al que le toca jugar. Además, se incluye el resaltado de las primeras mitades de un refrán, correspondiente al modo táctil, donde se quiere mostrar al usuario qué es lo que ha pulsado para que lo tenga en cuenta. Todos los cambios se pueden ver en la figura 12 y la figura 13.

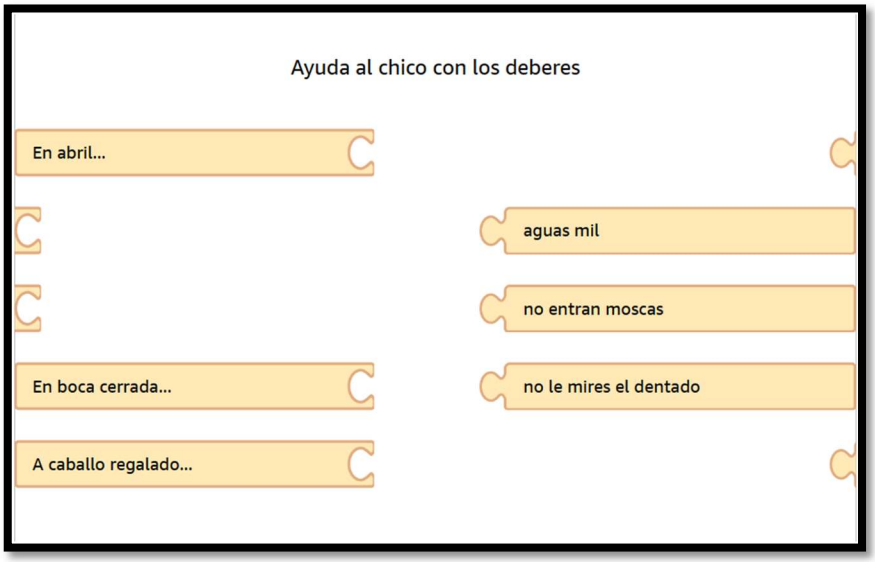


Figura 12. Interfaz final de los refranes.

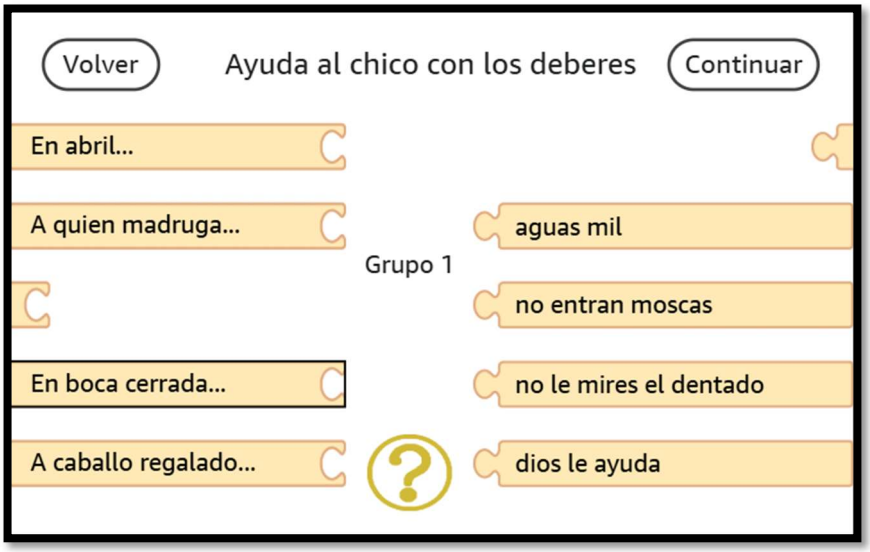


Figura 13. Interfaz final de los refranes.

4.3.4. Montemos el mapa

En este minijuego, para mejorar la interacción con la interfaz, definida inicialmente como se muestra en la figura 14, se han numerado tanto las piezas del mapa como los huecos donde se deben colocar para reconstruir el mapa, como se ve en la figura 15, permitiendo que el usuario pueda asociar las piezas con los huecos mediante comandos por voz. Se pretende que las propias imágenes y los huecos actúen como botones. Además, la pieza seleccionada por el usuario se marcará con un borde negro, permitiéndole saber cual es la pieza que está colocando en cada momento.

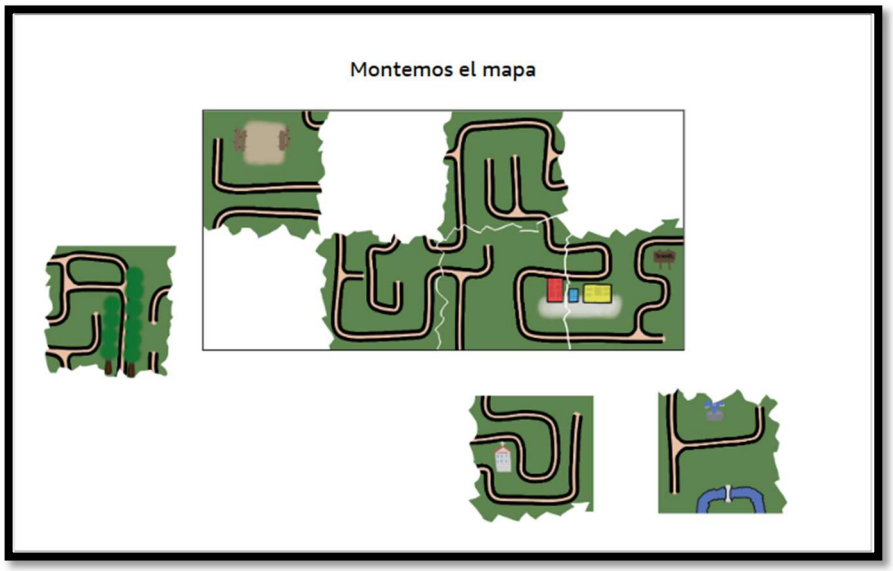


Figura 14. Interfaz final de “Montemos el mapa.

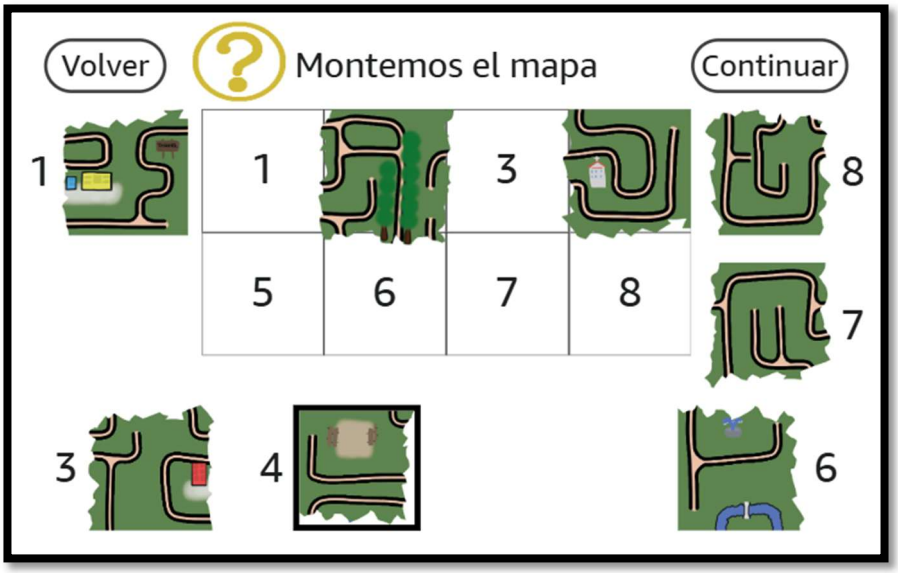


Figura 15. Interfaz final de “Montemos el mapa”.

4.3.5. Hagamos turismo

Este minijuego consta de dos partes para cada grupo. En la primera fase, Alexa preguntará al grupo por un monumento hasta que lo acierten. Aparte de los botones mencionados en todos los capítulos y el tamaño de letra, esta fase no ha sufrido cambios significativos entre su versión inicial, en la figura 16, y la versión final, en la figura 17.

Figura 16. Interfaz inicial de pregunta de “Hagamos turismo”.

Figura 17. Interfaz final de pregunta de “Hagamos turismo”.

En la segunda fase tampoco se observan grandes cambios. Lo único reseñable es que, en el diseño inicial mostrado en la figura 18, estaban escritos los comandos de voz que permiten moverse por el mapa, que han sido eliminados y sustituidos por flechas que harán de botones para dirigir de forma táctil la pieza de cada grupo hasta el monumento que deban visitar. Esto puede verse en la figura 19.

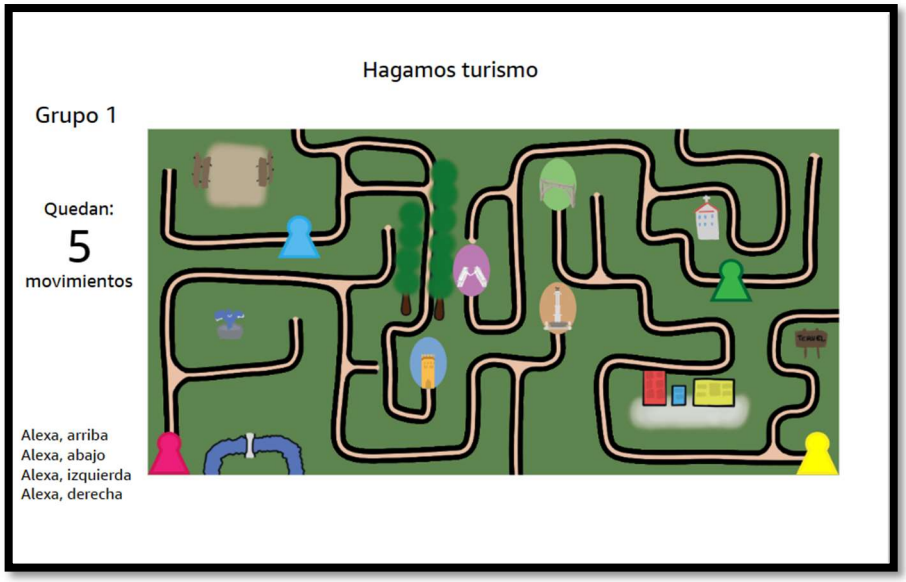


Figura 18. Interfaz inicial del mapa de “Hagamos turismo”.

Figura 19. Interfaz final del mapa de “Hagamos turismo”.

4.4. Encontrar las parejas

Este es el famoso juego de encontrar las parejas. En este caso se hace con tarjetas de colores. Tras elegir el juego “Encontrar las parejas” en la pantalla del catálogo se mostrará una nueva pantalla, que se puede ver en la figura 20, con 3 botones, cada uno con un texto donde estará escrito el nivel al que corresponde. Una vez seleccionado, mostrará la pantalla con el número de tarjetas que corresponda según el nivel.

Lo que se pretende es permitir al usuario elegir el nivel de dificultad, añadiendo más filas o columnas de tarjetas. Tendrá 3 niveles: fácil, medio y difícil. En el nivel fácil (figura 21) habrá tres filas y cuatro columnas, con un total de 12 tarjetas y 6 parejas. El nivel medio (figura 21) constará de 4 filas y cuatro columnas, con 16 tarjetas y 8

parejas. Por último, el nivel difícil (figura 22) estará formado por 4 filas y 5 columnas, con un total de 20 tarjetas y 10 parejas.

Cuando el usuario acierte, se quiere mostrar un texto que informe de ello. También se quiere contar cuantos intentos ha hecho el usuario hasta conseguir emparejar todas las tarjetas, mostrándoselo durante la partida.

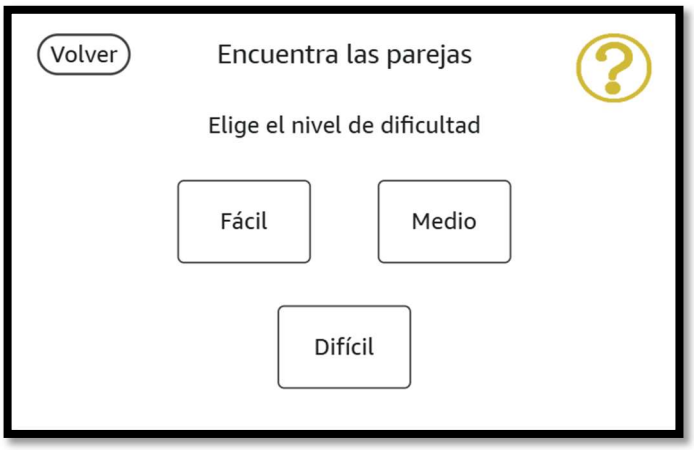


Figura 20. Interfaz de elección de nivel en “Encontrar las parejas”.

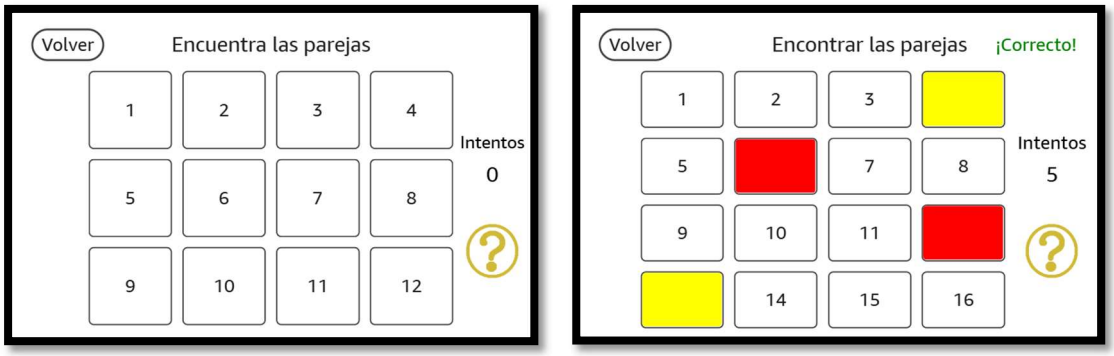


Figura 21. Interfaz de “Encontrar las parejas”, nivel fácil (izquierda) y medio (derecha).

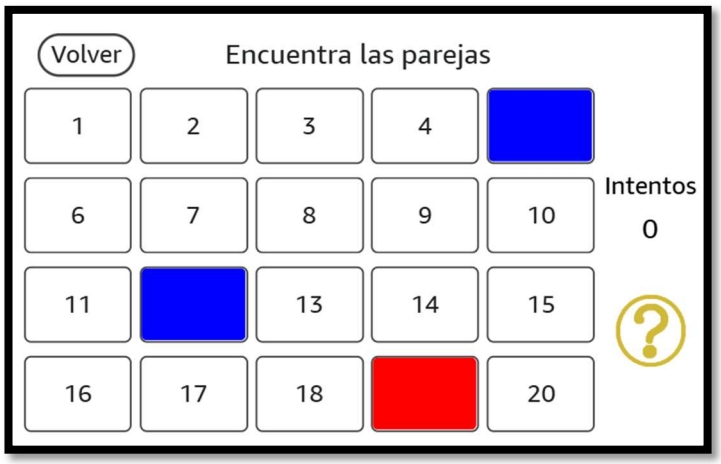


Figura 22. Interfaz de “Encontrar las parejas”, nivel difícil.

4.5. Adivinar el deporte

Este juego consiste en mostrar una imagen relacionada con un deporte (un balón de fútbol, una canasta) y tres opciones, para que el usuario elija la correcta. Se quiere guardar y mostrar el conteo de aciertos seguidos que lleve el usuario y el número máximo de aciertos que halla conseguido en el pasado hasta ese momento.

La imagen, al ser la parte principal de este juego, ocupa gran parte de la pantalla para que el usuario pueda identificar correctamente los elementos de esta. La racha de aciertos y la mejor racha se sitúan a la derecha por ser uno de los únicos espacios libres para añadir información a la pantalla y el diseño de los botones de las opciones es ancho para evitar pulsar dos a la vez. Se puede ver esta interfaz en la figura 23.

Figura 23. Interfaz de “Adivinar el deporte”

5. Implementación

En este capítulo se va a hablar de la implementación de la aplicación diseñada, explicando tanto aspectos generales de la implementación realizada como específicos de cada juego. También se desarrollarán los problemas encontrados y las soluciones llevadas a cabo.

5.1. Aspectos generales

Una aplicación para Alexa consta de varios componentes. El principal es el Modelo de Interacción descrito anteriormente. En la figura 24 se puede ver la definición del intent de *Hagamos turismo* en el modelo, llamado *LabyrinthIntent* por el parecido del mapa con un laberinto. Se puede observar que a dicha petición se le asocian las variables labAnswer y direction. No siempre es necesario decir la frase completa, a veces, si la petición espera variables, con decir un valor para la variable es suficiente, por ejemplo, decir “derecha” en vez de “Ve derecha”, aunque en algunos casos genera conflictos por lo que la aplicación siempre informa de los comandos disponibles en su versión completa.

```
{
    "name": "LabyrinthIntent",
    "slots": [
        {
            "name": "direction",
            "type": "direction"
        },
        {
            "name": "labAnswer",
            "type": "labAnswers"
        }
    ],
    "samples": [
        "{labAnswer}",
        "Ve {direction}"
    ]
},
```

Figura 24. LabyritnhIntent en el Modelo de Interacción.

Una vez que se asocia la frase del usuario con una petición, se pasa a Lambda, que es donde se almacena el código de la aplicación, y se comprueba qué gestor puede tratar esa petición. En la figura 25 se puede ver el gestor simplificado para la petición de elegir el nivel para el juego de encontrar las parejas, que recibe un parámetro level, que recibe como valor de un slot o como argumento del evento generado al pulsar uno de los botones de nivel que ofrece la interfaz, mostrada en la figura 20.

```
const PairsLevelIntentHandler = {
    canHandle(handlerInput) {
        return (Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) === 'PairsLevelIntent')
            || (Alexa.getRequestType(handlerInput.requestEnvelope) === 'Alexa.Presentation.APL.UserEvent'
            && handlerInput.requestEnvelope.request.source.id === 'pairsLevel')
    },
    handle(handlerInput) {
        const sessionAttributes = handlerInput.attributesManager.getSessionAttributes();
        let speakOutput = "";
        // Si el intent anterior es el correcto, se gestiona
        if(sessionAttributes['lastIntent'] === 'ChooseLevel') {
            // Obtener el valor de las variables que haya proporcionado el usuario
            if (Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest') {
                level = level + Alexa.getSlotValue(handlerInput.requestEnvelope, 'level');
            } else {
                level = handlerInput.requestEnvelope.request.arguments[0];
            }
            ...
            // Directiva APL para enviar la interfaz que corresponde
            handlerInput.responseBuilder.addDirective({
                type: 'Alexa.Presentation.APL.RenderDocument',
                version: '1.1',
                document: sessionAttributes["pairsDoc"],
                datasources: {
                    pairsData: {
                        type: 'object',
                        properties: {
                            exitButtonText: handlerInput.t('EXIT_TEXT'),
                            helpIcon: Util.getS3PreSignedUrl("Media/help_icon.png")
                        }
                    }
                }
            });
        }
        // Envío de la respuesta
        return handlerInput.responseBuilder
            .speak(speakOutput + oneMinuteOfSilence)
            .reprompt(speakOutput)
            .getResponse();
    }
};
```

Figura 25. Gestor simplificado de *PairsLevelIntent*.

Como se observa, la función *canHandle* recibe el objeto *handlerInput*, que contiene la petición completa en JSON enviada a la aplicación. Como en este caso, se han implementado los gestores para que comprueben si la petición es de tipo *Intent*, es decir, comando por voz y si es del tipo que gestiona o no. O de tipo evento, que en esta aplicación corresponde con pulsación de un botón, comprobando a continuación el id del botón pulsado. Si la petición cumple las condiciones, será atendida por el gestor. Si no se comprobará el siguiente en el orden en que se hayan dispuesto.

Al terminar de gestionar la petición, si es necesario, se añade una directiva, *addDirective*, a la respuesta a enviar que corresponde con el documento de la interfaz que se precise, pasándole los datos, o *properties*, necesarios para su correcta visualización. Dichos datos pueden ser imágenes, textos (título de interfaz, texto de un botón, …) o valores para algunas propiedades del documento.

En la figura 26 se puede ver el final del código principal de la aplicación, donde se exportan todos los gestores, los interceptores, y un adaptador de persistencia del que se hablará más adelante. La asignación de las peticiones a los gestores se hará según el orden en que estén exportados por lo que, si dos gestores pueden atender la misma petición, se elegirá siempre el que esté primero.

Figura 26. Gestores e interceptores exportados en orden de prioridad.

5.1.1. Almacenamiento

Amazon S3 (Simple Storage Service) y Amazon DynamoDB son dos servicios fundamentales ofrecidos por Amazon Web Services (AWS) de almacenamiento en la nube. Ambos tienen usos muy distintos y, si se usan en conjunto en una Hosted Skill, pueden ser herramientas muy poderosas.

- Amazon S3

Es un servicio de almacenamiento en la nube que permite almacenar y manejar grandes cantidades de datos no estructurados, como videos, imágenes, archivos de texto, etc. Es escalable y fácilmente integrable con Alexa. S3 organiza los datos en contenedores, llamados buckets, donde se pueden almacenar los archivos. A estos archivos se accede a través de URLs o APIs.

En una Hosted Skill, Amazon pone a disposición del programador un bucket gratuito con límite de uso (Figura 27), que para esta aplicación es suficiente. Si se publicara la plicación, habría que controlar que el flujo de datos no sobrepasara esos límites y, en caso de hacerlo, proporcionar recursos AWS propios para soportar la nueva carga de transacciones.

En esta aplicación, su uso será almacenar las imágenes y audios que se emplean en las respuestas a los distintos intents. Las imágenes, que acompañan y enriquecen las respuestas visuales de la skill, y los audios, cuyo objetivo se explica en el capítulo 5.10.

- Amazon DynamoDB

Es un servicio de base de datos NoSQL completamente administrado, diseñado para aplicaciones que requieren baja latencia y escalabilidad, almacenando los datos en tablas. Es una buena opción para gestionar datos estructurados o semiestructurados tales como configuraciones, preferencias y más.

Al igual que Amazon S3, en una Hosted skill, Amazon pone a disposición del programador una tabla DynamoDB gratuita con límite de uso (Figura 27), que para

esta aplicación vuelve a ser suficiente, teniendo que aportar recursos propios si se sobrepasara dicho límite.

Además, Amazon recomienda su uso para gestionar la persistencia de datos en las skills, que es la función que se necesita que haga en esta aplicación. Se pretende usar solo al iniciar y cerrar una sesión, para cargar y guardar los datos persistentes de la skill, respectivamente. Su baja latencia permitirá que no se tarde nada en iniciar la skill.

- Conclusión

Al combinarlos, se pretende dar una solución completa para las necesidades de esta skill, con S3 gestionando los archivos grandes y DynamoDB encargándose de la persistencia de datos.

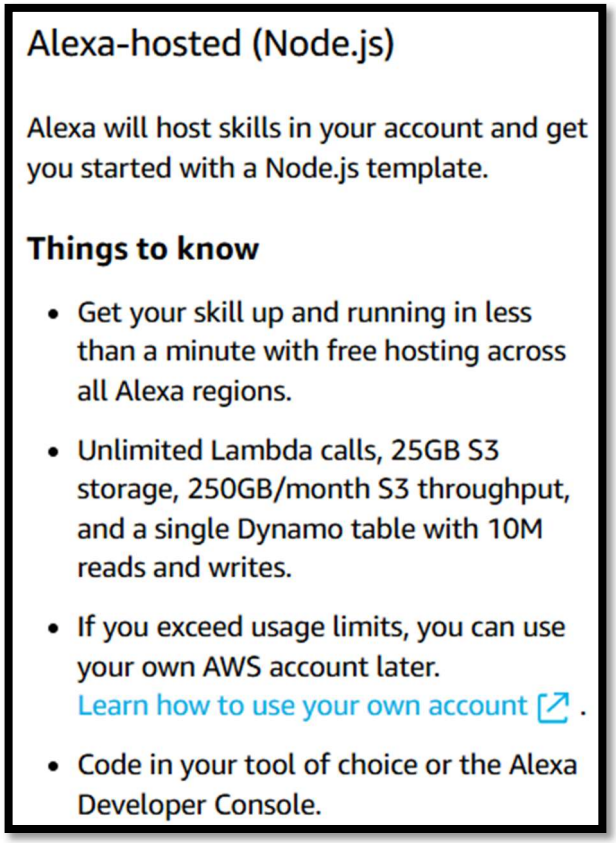


Figura 27. Información al elegir una skill Alexa-hosted.

5.2. Atributos de sesión

Los atributos de sesión son datos de tipo clave valor que se mantienen estables durante la sesión del usuario en la aplicación. A continuación, se explican los más relevantes:

- lastSpeech: Es donde se guardan las instrucciones de Alexa actualizadas para la tarea que el usuario deba realizar en cada momento. Si el usuario pulsa el botón de ayuda o dice alguno de los comandos de voz de ayuda, Alexa transmitirá el mensaje que tenga guardado en ese momento en este atributo.

- lastIntent: Cuando el usuario intenta invocar un intent, se comprueba si está en el estado correcto del juego para evitar que se pase, por ejemplo, de estar buscando parejas a montar el mapa directamente sin pasar por el catálogo, por ejemplo. Esto se hace mediante este atributo, que almacena el nombre del último intent gestionado, que es comprobado al inicio de cada handler. Esto también evita que el usuario reciba mensajes de Alexa erróneos que perturben su experiencia con un juego al recibir mensajes de otro juego distinto, si por ejemplo invoca otro intent sin querer.

- disableButtons: Si el usuario elige el método de interacción solo por voz, los botones se deshabilitan, excepto el de ayuda para dar mayor soporte al usuario. Para eso sirve este atributo, que se pasa como propiedad a la directiva de la interfaz que se quiera mostrar, y se asigna al atributo *disabled* del botón.

- mapTurn: En este atributo se almacena el número del turno en *Encontremos el Mapa* para mantener el orden en sus distintas fases durante toda la partida.

- mode: Es donde se almacena el modo de interacción elegido por el usuario. Tiene tres posibles valores: “TOUCH”, “VOICE”, o “FULL”, que representan la interacción táctil, por voz o completa respectivamente.

5.3. Interceptores

Los interceptores (interceptors) son funciones especiales que se ejecutan antes o después de la ejecución de las acciones de un gestor dependiendo de si son de tipo request o *response*, respectivamente.

En esta aplicación se han utilizado para dos tareas. La primera es la persistencia de datos, donde se intercepta un cierre de sesión y guarda los datos de la partida para que no se pierdan. También se intercepta la petición del nombre del usuario para recuperar sus datos de partidas guardadas una vez está identificado.

También se ha creado un interceptor que asigna a las peticiones una función ‘t’ que permite extraer variables de un archivo, que recibe como parámetro, donde se encuentran todos los diálogos de Alexa para la aplicación, las cadenas de texto a mostrar en las interfaces, y más información que se explica en el capítulo 5.4.

5.3.1. Persistencia de datos

Para la persistencia de los datos de un usuario, que en esta aplicación son los datos de las partidas guardadas, lo primero que se hace es crear en el documento constants.js un array, “PERSISTENT_ATTRIBUTES_NAMES”, al que se han añadido los nombres de los atributos de sesión necesarios para asegurar el guardado y recuperación de los datos de las partidas guardadas. Además, se precisa de un adaptador de persistencia, en este caso el adaptador de DynamoDB, que hace uso de la tabla de tipo [clave, valor] gratuita que ofrece Amazon. Este adaptador se pasa al gestor de atributos.

El interceptor de respuestas comprueba si se va a cerrar la sesión, ya sea por voluntad del usuario o por un fallo de la aplicación o el dispositivo, y guarda los atributos de sesión declarados como persistentes en la base de datos, usando como clave las concatenaciones del nombre del usuario con el nombre de los atributos, permitiendo guardar datos de varios usuarios a la vez. Además, borra los atributos de sesión restantes.

El interceptor de peticiones comprueba en cada petición desde que se inicia la aplicación, de manera asíncrona, si existe el atributo de sesión de nombre del usuario. Cuando lo detecta, le pide a la base de datos, a través del gestor de persistencia, que le entregue todos los datos del usuario que correspondiente, comprobando que el nombre de los atributos incluya el nombre del usuario, que se eliminará del nombre del atributo antes de asignarlo a los atributos de sesión. Esto se hace para no tener que añadir, cada vez que se quiera acceder a un atributo, el nombre del usuario delante.

5.4. Internacionalización

La internacionalización es el proceso de desarrollar aplicaciones de software para que puedan ser adaptadas a otros idiomas y regiones sin cambios en el código. Esto se ha llevado a cabo trasladando todas las cadenas de texto que corresponden a títulos y textos mostrados en las interfaces, todos los diálogos de Alexa y los valores posibles de los slots, para compararlos con los que lleguen en las peticiones, a un documento, en esta aplicación es es-ES.js, correspondiente al idioma español de España. Todos los datos se almacenan en formato [clave: valor], la clave siendo un nombre que identifica la finalidad de su valor.

En el interceptor mencionado en el capítulo 5.3, se asigna a la función 't' como atributo un documento que toma como nombre el idioma en el que está configurado el dispositivo que ejecuta la aplicación, que se obtiene directamente del dispositivo, añadiéndole ".js" al final. Si el idioma del dispositivo es español de España, el documento seleccionado será es-ES.js, que es el disponible en esta aplicación, por lo que la función 't', que recibe como parámetro una cadena de texto, extraerá el valor de ese documento que tenga esa cadena como clave.

Estos valores se leen de ese documento llamando a *handlerInput.t('CLAVE')* siendo CLAVE la cadena de texto que identifica a un valor en el documento.

Esto permite que, si se quiere adaptar la aplicación a otro idioma, por ejemplo, al inglés de Estados Unidos, habría que crear un documento llamado en-US.js donde se copie las claves de todos los valores del documento en español, y se traduzcan sus valores al inglés. Importante mantener el mismo nombre de las claves porque de otro modo no se podrá acceder a los valores correctos.

5.5. Encontremos el mapa (Versión inicial)

En este apartado se va a hablar de la primera implementación del juego Encontremos el Mapa, que solo incluía interacción por voz. Esto implica que los handlers solo comprueban si se ha hecho una petición del intent que gestionan.

En esta fase del desarrollo no se implementó todavía el guardado de partida ni se explicaban los comandos exactos para responder a las pruebas. Tampoco se dicen los comandos exactos que puede decir el usuario para progresar en el juego, sino que es más intuitivo.

El juego inicialmente pregunta al usuario el número de grupos que van a jugar, informando de que las únicas opciones posibles son 1, 2 o 4. Las fases del juego tienen siempre los mismos turnos por lo que, si se elige uno o dos jugadores, se harán más rondas de respuestas hasta completarlas.

5.5.1. Encuentra el trozo de mapa

En esta fase del juego, cuya interfaz se muestra en la figura 8, Alexa lee una descripción de una imagen, y el gestor únicamente Comprueba si el intent invocado es *ImagesIntent*. Si el grupo acierta se pasa al siguiente, si no, se pide al grupo actual que lo siga intentando y se cambia la imagen del avatar de Alexa a su versión triste. Esto se hace cambiando el parámetro de la directiva APL que carga el documento de la interfaz, al que se le pasa la imagen triste de Alexa obtenida de la base de datos S3.

Además, el recuadro rojo alrededor de la respuesta incorrecta es un objeto que siempre está, pero el valor de su color también es un parámetro de la interfaz que, si su valor es cadena vacía, es como si fuera invisible. Al fallar una respuesta, se cambia ese parámetro por la cadena “red” indicando que el cuadro que debe aparecer es rojo.

La interfaz incluía debajo de las imágenes el comando con el que se elegían, por ejemplo, “Alexa, imagen 1”, o “Alexa, repite” que se representa con una imagen que simboliza la repetición de la descripción. Es la única fase del juego que tiene una opción de repetir.

Al identificar las cuatro partes del mapa, Alexa pregunta si el usuario quiere pasar a la siguiente fase. Si dice sí, avanza a la siguiente prueba. Si dice no, Alexa insiste.

5.5.2. Parque

En esta fase solo se detecta si el usuario dice si o no a ayudar al niño. Si dice sí, se avanza a la siguiente fase. Si dice no, Alexa insiste.

5.5.3. Ayuda al chico con los deberes

Este es el juego de los refranes. En esta implementación, los refranes aparecen siempre en las mismas posiciones, facilitando su comprobación, y el usuario debe decir el refrán completo para que lo detecte, no puede decir solo una parte y en el siguiente comando la otra. Se puede observar su interfaz en la figura 12.

Para situar los refranes en la interfaz se emplean los atributos *up, down, left, right,* que simbolizan su separación de los bordes del objeto que los contiene, que en este caso es un contenedor con las mismas dimensiones que el contenedor de la imagen de fondo del refrán. El texto está incluido también en el contenedor de la imagen. Por ello, al querer marcar un refrán como completado en la interfaz se pasa como valor de *left* o *right* un valor ligeramente menor al de la imagen, para que se desplace lateralmente en el contenedor y se vea solo una parte.

Si se acierta, se pasa al siguiente grupo, si no, se le pide al grupo actual que lo vuelva a intentar. Así hasta que se aciertan cuatro refranes, que simbolizan las cuatro partes del mapa que faltan, momento en el que Alexa, al igual que en la fase anterior, pregunta si el usuario quiere avanzar. Cuando responde sí, el juego avanza a la siguiente fase.

5.5.4. Montemos el mapa

En esta fase, cuya interfaz se puede ver en la figura 14, se debe posicionar las piezas en los huecos. En esta primera implementación no se numeraron las imágenes, lo que fue un error pues los usuarios deberían imaginar el orden de las piezas y de los huecos. El comando para probar una pieza en un hueco es, por ejemplo, “la pieza 4 al

hueco 2". La posición de las piezas alrededor del recuadro es fija, en todas las partidas es la misma.

La correspondencia entre imágenes y huecos se guarda en dos variables, una lista de piezas y otra de huecos, representados por números. Los números dichos por el usuario se usan como índices de las dos listas y, si el valor en ambas listas con esos índices es el mismo, la pieza se coloca en el hueco. Esa colocación se logra teniendo las imágenes duplicadas, es decir, inicialmente el mapa está completo, pero tanto las imágenes de alrededor como las del cuadro central reciben un parámetro de opacidad, que inicialmente está a 0 (invisible) para las interiores y a 1 (visible) para las exteriores. Cuando se acierta una posición, se intercambian las opacidades pasadas a las dos versiones de la imagen, haciendo que parezca que ha sido colocada.

El mismo grupo sigue intentándolo hasta que acierta, momento en que se pasa al siguiente. En este caso, hay dos rondas de turnos puesto que hay ocho piezas y cuatro grupos. Al terminar ocurre lo mismo que se ha explicado en las otras fases.

5.5.5. Hagamos turismo

Esta fase fue la más compleja de programar en esta versión. Viendo su interfaz en las figuras 16 y 18 se aprecia la gran cantidad de elementos en pantalla, y estos conllevan muchas variables que controlar y muchos parámetros que pasar a la interfaz.

En la primera parte de este juego, las preguntas sobre los monumentos, el primer parámetro obvio es la opacidad del fondo, que tiene un valor de 0.5 para darle más protagonismo a Alexa y a la pregunta, pero manteniendo la sensación de que tanto pregunta como fondo forman parte de la misma prueba. Tanto las preguntas como el orden de sus posibles respuestas son fijos.

En este caso, después de que Alexa lea la pregunta, si un grupo acierta, no se pasa directamente al siguiente, sino que el mismo grupo pasa a la siguiente fase, cambiando la opacidad de la pregunta a 0 y la del fondo a 1 para realizar sus primeros cinco movimientos por el mapa, apareciendo en éste la imagen del monumento a visitar, que siempre estuvo ahí, pero tenía opacidad 0. Alexa le informará de su objetivo.

Los comandos de los movimientos aparecen en la pantalla, al igual que la información del grupo al que le toca y los movimientos que le quedan por realizar. Los grupos están representados por piezas de colores inicialmente colocadas en las esquinas del mapa. Tanto las posiciones de las piezas como el número de grupo y el numero de movimientos restantes son también parámetros de la interfaz de esta fase del juego. Si un grupo intenta ir en una dirección prohibida, es decir, que no siga un camino del mapa, recibe un mensaje de aviso de Alexa, pero no pierde ese movimiento.

De cinco en cinco movimientos se espera a que cada grupo llegue a su monumento, momento en el que Alexa informa de que ha terminado el juego, preguntando a los grupos si se lo han pasado bien. Si la respuesta es afirmativa, se pasa a una interfaz similar a la inicial del juego, que se puede ver en la figura 6, pero con el texto "¡Hasta pronto!" mientras Alexa da un mensaje sobre lo divertido que ha sido y que espera verlos a todos pronto. Ahí solo se puede decir "Alexa, salir" para cerrar la aplicación.

5.6. Catálogo

Al iniciar la aplicación se pide el nombre del usuario para poder cargar los datos de sus partidas guardadas y guardar datos de partidas nuevas sin terminar.

La implementación del catálogo es la primera que introduce la interacción táctil, permitiendo al usuario elegir uno de los dos modos, o ambos a la vez, después de identificarse, guardando el modo en un atributo de sesión, *mode,* explicado en el capítulo 5.2. Esto influye en las instrucciones que dará. Por ejemplo, en el modo táctil solo informará de qué botones se pueden pulsar en la interfaz que muestre el dispositivo por pantalla, y en el modo por voz solo informará de los comandos de voz que acepta la aplicación en la fase actual. En el modo completo, Alexa informará de todas las opciones, tanto vocales como táctiles.

Los botones se identifican mediante su propiedad id, y al ser pulsados envían argumentos, que es una información adicional que pueden leer los handlers.

Esto se consigue creando tres versiones de los diálogos de Alexa que contengan instrucciones en el documento es-ES.js mencionado en el capítulo 5.4 sobre internacionalización. Por ejemplo, las instrucciones de la elección de un juego del catálogo están guardadas como 'INTRUCCIONES_CATALOGO_VOICE', 'INTRUCCIONES_CATALOGO_TOUCH' e 'INTRUCCIONES_CATALOGO_FULL', cada una con su versión de las instrucciones. Entonces, para acceder a una o a otra, se concatena el valor del atributo de sesión *mode* a la cadena 'INSTRUCCIONES_CATAGOLO_' y se le pasa a la función t, que devolverá el valor de las instrucciones en el modo correcto.

Añadir botones implica que los handlers comprobarán tanto si hay una petición del intent que gestionan como si se ha generado un evento de botón asociado a ese intent, que puede ir acompañado de un argumento.

La implementación de la opción de ayuda hace uso del atributo de sesión *lastSpeech* explicado también en el capítulo 5.2. Cuando se invoca ese intent se pasa a Alexa el valor de ese atributo para que lo lea, ayudando al usuario a seguir jugando.

Se ha implementado también la opción de volver, tanto por voz diciendo "Alexa, volver" como por botón en la interfaz principal del catálogo, permitiendo volver a la pantalla de elección del modo de interacción.

Al elegir uno de los juegos, el handler correspondiente comprueba si el usuario tiene alguna partida guardada de ese juego y Alexa le pregunta si quiere continuarla. Si la tiene, el handler que trata las respuestas de si o no comprueba cual ha sido la respuesta del usuario. Si la continúa, mira la interfaz que le tocaba dependiendo del atributo de sesión *lastIntent* y la muestra, pasándole los valores de los parámetros que tenía guardados y haciendo que Alexa comunique las instrucciones de ese juego, o fase del juego.

5.7. Encontremos el mapa (Versión final)

En este apartado se va a hablar de la implementación final del juego Encontremos el Mapa. En esta fase del desarrollo se implementa el guardado de partida. También se modifican los diálogos de Alexa para ser más concretos y explicar al usuario las acciones que tiene disponibles dependiendo del modo de interacción elegido en cada fase del juego.

Se han añadido botones en todas las interfaces, identificados por un id, para permitir interacción táctil, algunos como nuevos objetos y otros convirtiendo imágenes en elementos táctiles, teniendo que modificar todos los handlers para comprobar si deben gestionar la pulsación de un botón consultando su id.

El botón de continuar añadido en todas las interfaces del juego es la opción táctil para responder afirmativamente a una pregunta de Alexa, que permite avanzar de fase del juego al terminar la anterior.

El botón de volver en las interfaces sirve para volver al catálogo, guardándose la partida automáticamente si no se había acabado.

También se ha añadido un texto a todas las interfaces de las fases del juego informando del número del grupo que tiene el turno, que es un parámetro de texto en cada interfaz, salvo que sea una partida de un solo grupo, en cuyo caso se pasa una cadena vacía y no aparece el texto en pantalla.

5.7.1. Encuentra el trozo de mapa

En esta fase del juego, cuya interfaz se muestra en la figura 9, uno de los mayores cambios de implementación, además del uso de botones ya explicado en el capítulo 5.6, ha sido el añadir un parámetro a la interfaz que, al completar esta fase, ponga opacidad 0 a las cuatro imágenes, al texto del turno, y la numeración de las imágenes. Dando una sensación de juego completado. Además, en esta interfaz, el botón de ayuda sustituye a la imagen de repetir.

Los botones de las imágenes envían como argumento el número de la imagen pulsada por el usuario.

La fase del parque no tiene nada nuevo de implementación aparte de los tres botones de continuar, volver y ayuda, mencionados anteriormente.

5.7.2. Ayuda al chico con los deberes

En esta fase, cuya interfaz se muestra en la figura 13, a nivel de implementación se ha creado una función que escoge aleatoriamente 5 refranes de una lista de 9 para que varíen de una partida a otra, almacenando la posición de ambas partes de cada refrán ya que los botones de cada parte llevarán como argumento su posición.

Además, el modo táctil permite que se seleccione una parte del refrán y luego otra, por lo que e añade un parámetro de color a las primeras partes de los refranes para asignarlo al recuadro que rodea a la opción pulsada, permitiendo al usuario saber si ya ha pulsado la primera parte o no. También permite cambiar de un inicio de refrán a otro directamente sin tener que pulsar un final de refrán para que se deseleccione el pulsado.

5.7.3. Montemos el mapa

A nivel de implementación, en esta fase solo se incluye el mismo detalle que en el capítulo anterior. Al pulsar una pieza de fuera del recuadro aparece un marco rodeándola, pudiendo cambiar la selección entre las demás. Se puede ver en la figura 15.

Los botones envían como argumento su número de pieza, o de hueco, para poder identificarlos y comprobar si pertenecen a la misma pieza.

5.7.4. Hagamos turismo

En esta fase, cuyas interfaces pueden verse en las figuras 17 y 19 los botones de las respuestas a las preguntas de los monumentos envían como argumento la posición de la respuesta pulsada para su comprobación y los botones de las flechas de dirección llevan como argumento la dirección, como texto, a la que corresponden.

También se ha añadido un recuadro rojo que rodea a la pieza del mapa del grupo al que le toca moverse.

El juego se termina igual que en la versión inicial, pudiendo responder con botones también. En este caso, en vez de salir de la aplicación se puede volver, por voz o botón, al catálogo.

5.8. Encontrar las parejas

El juego encontrar las parejas consta de tres niveles del clásico juego “Encontrar las parejas”, con las mismas normas: dar la vuelta las tarjetas de una en una formando parejas.

Cada tarjeta está implementada de la siguiente manera: Un texto independiente con el número de la tarjeta, y en la misma posición un elemento táctil, que contiene un elemento marco, que contiene un elemento imagen. Las imágenes que aparezcan son simplemente cuadrados de colores, que mediante un atributo en el código de la interfaz (scale: “fill”) se expande hasta ocupar el tamaño completo del objeto imagen al que se asocia. Cuando la imagen no existe, no se muestra, por lo que el botón con la imagen se implementa sobre el texto para que, si no hay imagen, se vea el número, y si hay imagen, porque el usuario ha seleccionado la tarjeta, la tarjeta se verá del color correspondiente, tapando el texto que quedaría debajo.

Si se crea una partida nueva, el usuario elige la dificultad, que se guarda en un atributo de sesión para comprobarlo más adelante. Si no, se carga la partida que hubiera guardada con las parejas que hubiera ya descubiertas.

En este juego, a lo que se llama voltear o dar la vuelta es en realidad pasar una imagen con el color de la tarjeta al marco del botón. En este caso, las tarjetas se pueden seleccionar de dos formas: pulsando sobre la tarjeta que se quiera voltear, o diciendo el comando de voz “Alexa, la X”, siendo X un número correspondiente a la tarjeta que se quiere dar la vuelta. Si la pareja es del mismo color, cuando se pulse la siguiente tarjeta, la pareja se quedará volteada, mostrando su color. Si son de distinto color, al pulsar la siguiente, se voltearán las dos anteriores, dejando al descubierto su número, salvo que la pulsada sea una de ellas, en cuyo caso solo se voltearía la otra.

Para los colores de las tarjetas se usa una función, *randomColors*, que a partir del número de parejas n, definido por el nivel de dificultad seleccionado por el usuario, escoge los primeros n colores de un array de colores y los almacena por duplicado en n*2 atributos de sesión de forma aleatoria. Estos atributos tienen de combre “colorX” siendo X el número que los relaciona con su tarjeta, por lo que el valor que recibe “color1” es el color de la tarjeta 1. Estos valores son los que se comprueban al seleccionar dos tarjetas para saber si es una pareja del mismo color o no.

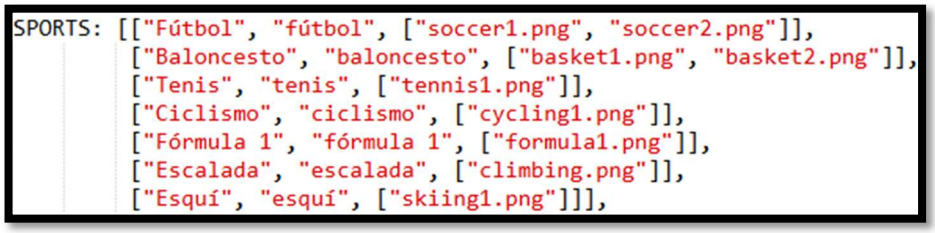
El número de intentos mostrado en pantalla es un atributo de sesión que se pasa como parámetro a la interfaz.

El juego termina cuando el usuario descubre la última pareja, momento en que se le notifica su victoria, pidiéndole que pulse el botón *Volver* o diga “Alexa, volver” para regresar al catálogo. Mientras se espera la acción del usuario, se bloquean todos los botones excepto *Volver* y el botón de ayuda.

5.9. Elegir el deporte

En la figura 28 se observa que existe una matriz que contiene los nombres de los deportes elegidos para este juego y el nombre de las imágenes asociadas a cada uno. Estos nombres corresponden con el nombre con el que han sido guardadas en la base de datos. También se ha definido una función, *randomSports* que toma los valores de esta matriz y crea otra, ordenando aleatoriamente cada imagen, asociada a su deporte, que se ejecuta al iniciar este juego, variando así el orden en que aparecen entre una partida y otra. También asigna dos nombres de deportes más para completar las opciones que se mostrarán.

Cuando el usuario acierta, se incrementa el contador de aciertos, incrementando su atributo de sesión asociado, pero cuando falla, se pone a cero. Si al sumar un acierto supera la mayor racha de aciertos previa, se actualiza ese atributo también, por lo que se muestra actualizado en la pantalla.



```
SPORTS: [["Fútbol", "fútbol", ["soccer1.png", "soccer2.png"]],
         ["Baloncesto", "baloncesto", ["basket1.png", "basket2.png"]],
         ["Tenis", "tenis", ["tennis1.png"]],
         ["Ciclismo", "ciclismo", ["cycling1.png"]],
         ["Fórmula 1", "fórmula 1", ["formula1.png"]],
         ["Escalada", "escalada", ["climbing.png"]],
         ["Esquí", "esquí", ["skiing1.png"]]],
```

Figura 28. Array de deportes con imágenes asociadas.

5.10. Accesibilidad

En todas las pantallas, sin excepciones, todos los elementos tienen sus posiciones y tamaños dependientes del tamaño de la pantalla del dispositivo en el que se esté ejecutando la aplicación. Por ejemplo, la separación de un botón del lateral de la pantalla es, por ejemplo, “3vw”, que significa el tres por ciento de la anchura de la pantalla (viewport width). Por esta razón, ningún elemento se puede superponer a otro al cambiar de dispositivo, permitiendo que se pueda ejecutar en casi todos los dispositivos Alexa con pantalla.

Según el modo de interacción elegido al iniciar la aplicación, las instrucciones de uso que comunica Alexa pueden variar. Si se elige el modo táctil, Alexa solo informará al usuario de los botones que puede pulsar. Además, la aplicación no reaccionará a comandos de voz. Por el contrario, si elige el modo por voz, solo informará de los comandos de voz disponibles y deshabilitará la interacción táctil, ocultando algunos botones para evitar confundir al usuario. Por último, si elige el modo completo, Alexa informará tanto de los comandos de voz como de los botones disponibles. Esto permite acortar los diálogos de Alexa, evitando instrucciones innecesarias.

5.11. Problemas encontrados

- Actualización de la salida gráfica: Una skill de Alexa reacciona a comandos de voz o táctiles, que hacen que el handler correspondiente trate dicha información y actualice el estado del la skill, tras lo que carga el documento que mostrará en pantalla y queda a la espera de una nueva entrada de audio o táctil. Esto hacía que en algunos juegos no se pudiese programar la experiencia visual como se pensó en el diseño. Por ejemplo, en el juego de *unir los refranes*, no se puede poner en verde las partes del refrán si el resultado es correcto y luego eliminarlos de la pantalla pasados unos segundos. En este caso, lo que se acabó haciendo es eliminarlos directamente si el refrán es correcto y no hacer nada si es incorrecto, haciendo que Alexa informe mediante un mensaje de voz si se ha acertado o no.

- Mismo comando de voz para distintos intents: Al ser una aplicación destinada a personas mayores y grupos intergeneracionales, los comandos deben ser lo más sencillos que permita el entorno de programación de Alexa. Esto hace que algunos juegos tengan comandos con los mismos slots, lo que hace que Alexa los confunda y los asigne al primer intent que utilice ese slot. Por ejemplo, en *Encuentra el trozo de mapa*, un comando es “La 1”, y en *Montemos el Mapa* un comando es “La 1 en la 3”. La solución ha sido hacer que las frases de todos los intents con slots numéricos invoquen el mismo intent, y en su handler correspondiente, comprobar el atributo de sesión *lastIntent* para saber a cuál se está queriendo invocar. El resultado es un intent bastante extenso, de 400 líneas de código. Algo similar pasa con el intent de respuestas, que recibe un si o un no, desde varios puntos de la aplicación, como al final de cada fase de *Encontremos el Mapa* o al preguntar si el usuario quiere continuar la partida guardada, por lo que en su handler comprueba en qué estado se encuentra la aplicación y responde tanto por voz como visualmente tras realizar las acciones que corresponda, quedando como el intent más largo, con 700 líneas.

- Alexa da poco tiempo al usuario para responder. Al responder a un comando del usuario, el handler que ha gestionado la última petición crear una respuesta completa, con diálogo, directivas, etc. La respuesta hablada puede ser en forma de texto o de audio, y se pasa como parámetro al método *speak* del constructor de la respuesta.

  En dispositivos con pantalla, la skill esperará 8 segundos con el micrófono abierto a recibir la respuesta del usuario tras terminar el diálogo. Si no recibe nada, apaga el micrófono y espera 30 segundos más. En vez de eso, se puede llamar también al método *reprompt*, que tras los 8 segundos anteriores pronuncia otro diálogo que puede ser un mensaje de ayuda sobre lo que hacer en ese momento, y otra vez 8 segundos con micrófono abierto y 30 con micrófono cerrado. Si no recibe respuesta en esos 30 segundos se cierra la sesión.

  Teniendo en cuenta el público objetivo, dejar el micrófono abierto no se considera una buena elección pues los usuarios podrían ponerse a pensar en voz alta su respuesta o comentarla con otras personas. Por ello ha decidido añadir un audio de 60 segundos en silencio tras el diálogo de Alexa para posponer el encendido del micrófono y dar tiempo a los usuarios a pensar la respuesta sin que Alexa detecte cualquier palabra como respuesta. En consecuencia, para avisar a Alexa de que van a responder, tendrán que decir la palabra “Alexa” primero para que se active el micrófono de nuevo.

También se añaden pausas en medio de los diálogos para que no se haga muy tedioso el escuchar todo seguido a la misma velocidad. El principal ejemplo sería decirle a un usuario que ha elegido correctamente y después ponerse a dar las instrucciones de la siguiente fase del juego. En vez de hacerlo de corrido, se quieren añadir pausas de entre medio segundo y dos segundos para separar la intención de las partes de algunos diálogos, por ejemplo, en *Encontremos el Mapa*, para separar el informar de una acción correcta a un grupo y dar las instrucciones al siguiente.

- En el juego *Encontremos el mapa* se quería que los miembros de los grupos participantes pudieran hacerse una foto con el dispositivo desde la aplicación para guardarla de recuerdo, pero Amazon tiene prohibido el uso de la cámara en aplicaciones o skills de terceros.

6. Evaluación

En este apartado se presentan las pruebas de evaluación con usuarios realizadas al terminar la implementación de la aplicación, detallando la metodología seguida, los resultados y su análisis.

6.1. Metodología

Para evaluar esta aplicación, al no disponer de un grupo de personas, mayores y niños, con el que hacer pruebas de usuario, se ha realizado con un pariente de 78 años, que solo usa el móvil en su día a día para llamar y usar WhatsApp, una prueba de usabilidad con comentarios en voz alta, también llamada *Think-aloud protocol*. Esta prueba consiste en plantear una serie de tareas al usuario y dejar que interaccione con la aplicación a su ritmo, verbalizando sus pensamientos, acciones y decisiones en tiempo real.

Las tareas han sido completar una partida de cada uno de los tres juegos del catálogo.

6.2. Resultados

Durante la prueba, el usuario ha ido haciendo varios comentarios en voz alta sobre lo que piensa de la aplicación y las acciones que iba tomando. Aquí se recogen las más relevantes, citadas textualmente:

1. “Se quita todo el rato el cuadrado y me hace elegirla otra vez”
2. “Está muy callada esta chica(Alexa)”
3. “¿Y aquí quien soy yo? Que me dice que no puedo ir en esa dirección”
4. “Me gusta mucho el interrogante (botón de ayuda), a veces no recuerdo qué tengo que decirle”

6.3. Análisis

Se ha observado que el usuario se maneja bastante bien con la aplicación, consiguiendo completar todos los juegos sin grandes dificultades, aunque durante la prueba hizo algunos comentarios sobre aspectos a mejorar, presentados en el capítulo anterior, que se van a tratar a continuación:

- En el comentario 1, el usuario hace referencia a la fase de montar el mapa, que es tipo puzle. El problema está en que, al seleccionar un hueco del puzle incorrecto para la pieza seleccionada, la pieza se deja de estar seleccionada, perdiendo el cuadro negro que la rodea. Al usuario le gustaría poder probar esa pieza en otro hueco sin tener que volver a elegirla. Se ha decidido cambiar esa parte del código para que siga seleccionada la pieza al fallar.

- En el comentario 2, el usuario se queja de la falta de comentarios por parte de la aplicación, en concreto, al acertar una pareja en el juego de encontrar las parejas, porque solo muestra un mensaje por pantalla. Como respuesta se ha añadido ese mismo mensaje, que dice “¡Correcto!”, a la salida de voz de Alexa cuando se acierta una pareja.

- En el comentario 3, el usuario hace referencia a la fase de turismo del juego *Encontremos el mapa*, donde los usuarios deben visitar monumentos por el mapa, representados por unos iconos de distintos colores. El problema que encuentra el usuario es que no sabe qué icono se mueve en cada turno. Se ha tomado muy en cuenta esta observación, por lo que se ha modificado el código de esa parte y ahora en cada turno se pone un marco rojo alrededor del icono que se va a mover por el mapa.

- En el comentario 4, el usuario informa de su aprecio por el botón de ayuda, pues lo ha usado en varias ocasiones durante su interacción con la aplicación.

7. Gestión del proyecto

Este proyecto se ha desarrollado en diferentes fases:

- Primero se decide hacer un catálogo de juegos, comenzando un análisis previo que incluye el estudio de aplicaciones similares, de los dispositivos Alexa y del público objetivo.

- Después se implementó una versión inicial del juego *Encontremos el mapa* del TFM mencionado anteriormente para el dispositivo Alexa, que tenía interacción únicamente por voz.

- Luego se diseña las interfaces de los juegos, incluyendo en *Encontremos el* mapa lo necesario para soporte táctil, que son los botones.

- A continuación, se llevó a cabo la implementación de la aplicación final, que es la parte más costosa, tomando la mayor parte del tiempo.

- Al mismo tiempo que se realiza la implementación de las distintas partes, se van haciendo pruebas con el dispositivo Echo Show 8 puesto que representa a los dispositivos objetivo de esta aplicación.

- También se va añadiendo información a la memoria conforme se avanza en el proyecto.

- La evaluación final con el usuario se realiza el día 10 de diciembre.

- Al analizar los resultados de la evaluación con el usuario, se realizan algunos cambios en la aplicación.

En la figura 29 se detalla la extensión en el tiempo de este proyecto en un diagrama de Gantt.

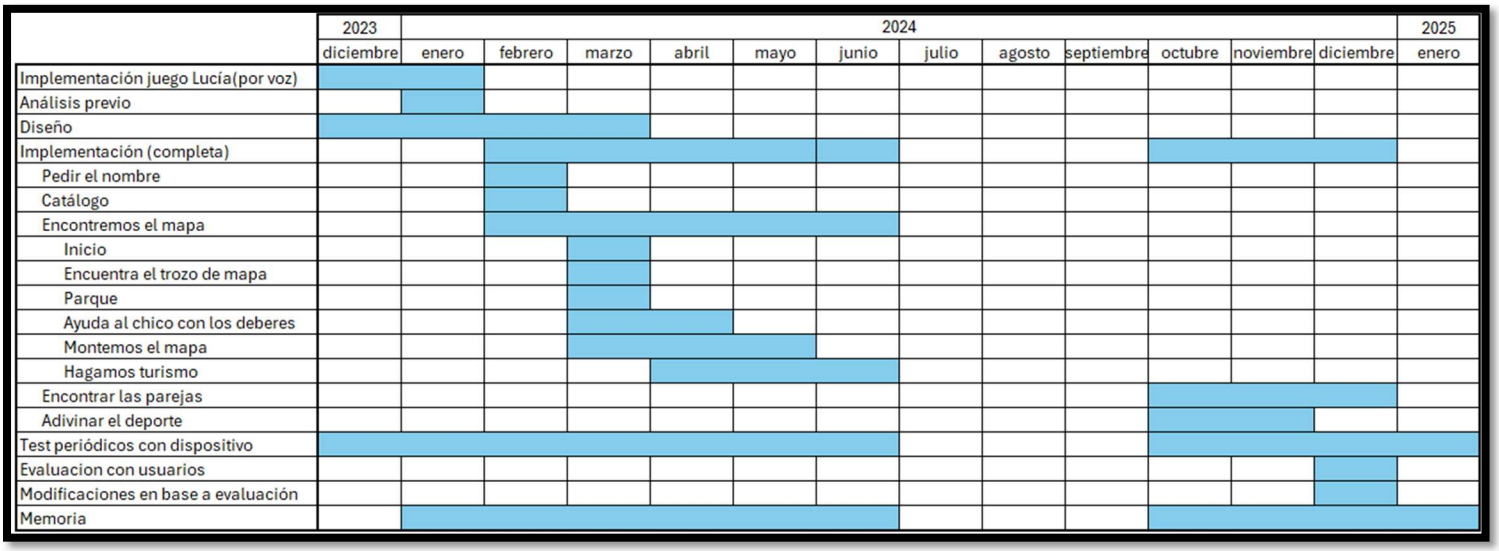


Figura 29. Diagrama de Gantt del desarrollo del proyecto.

Debido a situaciones personales, en julio de 2024 se interrumpió el desarrollo de este proyecto durante 4 meses, retomándolo en octubre de ese mismo año.

El proyecto ha tenido una duración total de unas 450 horas, con la implementación tomando más de la mitad de ese tiempo.

8. Conclusiones

En este último capítulo se explican las conclusiones extraídas de la realización de este proyecto tras su finalización, la valoración personal y las posibles tareas a realizar para continuar con su desarrollo en el futuro.

8.1. Conclusiones

Se han cumplido e implementado los objetivos establecidos en el primer capítulo del proyecto.

- Se han estudiado varias aplicaciones similares a la implementada en este trabajo para poder comprender las necesidades del usuario, observar las ventajas que ofrecen los asistentes por voz y averiguar las aplicaciones ya existentes en el mercado.

- Se han estudiado las posibles tecnologías a utilizar en el desarrollo de la aplicación, eligiendo las más apropiadas para el desarrollador favoreciendo la comodidad a la hora de su implementación.

- En función de los estudios anteriores, se han definido los requisitos de la aplicación y se ha realizado un diseño de acuerdo con esos requisitos, facilitando así el futuro desarrollo de la aplicación.

- Se ha desarrollado una aplicación por voz con apoyo visual y táctil para dispositivos Alexa Echo Show y Alexa Fire HD que permite a los usuarios acceder a un catálogo con tres juegos para jugar solos o acompañados, eligiendo el modo de interacción con la aplicación. Además, guarda el progreso de partidas no terminadas para continuarlas más adelante y guarda y muestra estadísticas del juego.

- Se ha llevado a cabo una evaluación con usuarios con la que se ha verificado el funcionamiento de la aplicación, detectando posibles objetivos de mejora que han sido tratados posteriormente. La aplicación ha demostrado un funcionamiento general satisfactorio para el usuario.

8.2. Valoración personal

Este trabajo me ha servido como puerta al mundo de la programación de aplicaciones para asistentes por voz. Me ha permitido descubrir las limitaciones de los asistentes por voz y, en profundidad, las limitaciones de los dispositivos Alex de Amazon, al igual que las ventajas que suponen las aplicaciones para estos dispositivos frente a las tradicionales para móvil o tablet, donde tú no puedes hablar con el dispositivo.

Pese a la dificultad inicial de manejar las tecnologías y las normas de programación para el desarrollo de estas aplicaciones, he aprendido mucho sobre este tema, quedando con ganas de seguir en un futuro programando alguna otra aplicación para asistentes por voz Alexa, teniendo en cuenta mi sentimiento de satisfacción con el desarrollo de este trabajo.

También me ha ayudado a comprender las limitaciones de las personas mayores y los niños, a la hora de interactuar con nuevas tecnologías, concretamente con asistentes de voz con pantalla táctil, y como solventarlas, algo que he notado sobre todo tras realizar la evaluación con usuarios, permitiéndome valorar la importancia de adaptar el producto al usuario final.

Estoy satisfecho con el proyecto final y espero que esta aplicación pueda ayudar a acercar a estos grupos de edad en una experiencia entretenida y divertida.

8.3. Trabajo futuro

Una vez terminado el proyecto se observan diferentes vías de ampliación y mejora de la aplicación de cara a futuras implementaciones.

La mejora principal debería consistir en llevar a cabo pruebas con un mayor número de usuarios, permitiendo recabar grandes cantidades de información sobre como adaptar mejor la aplicación al usuario final.

También se podrían añadir más juegos a la aplicación para ofrecer al usuario más variedad y diversidad de opciones de entretenimiento, que podrían ser en solitario, en equipo, o en grupos como *Encontremos el Mapa*, para seguir fomentando la interacción entre los usuarios.

Pero sin duda, lo que tendría mayor alcance sería traducir la aplicación a distintos idiomas, internacionalizando sobre todo el juego de *Encontremos el Mapa*, para que los usuarios de distintos países hagan turismo por alguna ciudad propia de su país.

Incluso, al iniciar el juego *Encontremos el Mapa,* se podría dar una serie de ciudades posibles para que el usuario decida cuál visitar. Esto también se podría hacer con el juego *Adivinar el deporte*, eliminando la parte de deporte obligatoria y permitiendo elegir al usuario el tema de las imágenes que debe reconocer.

Bibliografía

[1] Gabriel, Dr.T.Malathi & Kumar, s & Mathew, Binu & Rajkumar, Mallipaamu. (2022). Voice assistance - future of contact. URL: https://www.researchgate.net/publication/385127091_VOICE_ASSISTANCE-FUTURE_OF_CONTACT

[2] Papí-Gálvez, N. y García-Espinosa, S. (1 de agosto 2023). Brecha digital y adultos mayores: desafíos y búsqueda de soluciones. URL: https://brechadigitalgeneracional.com/brecha-digital-y-adultos-mayores-desafios-y-busqueda-de-soluciones/

[3] Yu RWL, Chan AHS. Meta-analysis of the effects of game types and devices on older adults-video game interaction: Implications for video game training on cognition. *Appl Ergon*. 2021;96:103477. URL: https://doi.org/10.1016/j.apergo.2021.103477

[4] Orlofsky, S., & Wozniak, K. (2022). Older adults' experiences using Alexa. *Geriatric nursing (New York, N.Y.)*, *48*, 247–257. URL: https://doi.org/10.1016/j.gerinurse.2022.09.017

[5] García Petrovic, Lucía. (2024). Diseño de experiencias intergeneracionales para asistentes por voz. URL: https://zaguan.unizar.es/record/133538/files/TAZ-TFM-2024-045.pdf

[6] Amazon. Smart devices with Alexa. URL: https://www.amazon.com/smart-home-devices/b?ie=UTF8&node=9818047011

[7] Amazon. Alexa Skills URL: https://www.amazon.es/b?ie=UTF8&node=13944662031

[8] Alexa Voice Service (AVS) | Allion Labs. URL: https://www.allion.com/certification/alexa_voice_service/

[9] Amazon. Alexa Skills Kit Official Site: Build Skills for Voice. URL: https://developer.amazon.com/es-ES/alexa/alexa-skills-kit

[10] Amazon. AWS | Lambda - Gestión de recursos informáticos. URL: https://aws.amazon.com/es/lambda/

[11] Amazon. Características de Amazon DynamoDB – Amazon Web Services. URL: https://aws.amazon.com/es/dynamodb/features/.

[12] Amazon. Características de Amazon s3 – Amazon Web Services. URL: https://aws.amazon.com/es/s3/features/.

[13] Amazon. Add Visuals and Audio to Your Skill | Alexa Skills Kit. URL: https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/add-visuals-and-audio-to-your-skill.html

[14] Amazon. About Alexa-Hosted Skills | Alexa Skills Kit. URL: https://developer.amazon.com/en-US/docs/alexa/hosted-skills/build-a-skill-end-to-end-using-an-alexa-hosted-skill.html

[15] Amazon. About the Alexa Developer Console | Alexa Skills Kit. URL: https://developer.amazon.com/en-US/docs/alexa/devconsole/about-the-developer-console.html

[16] Rice, M., Yau, L., Ong, J., Wan, M., & Ng, J. (2012). Intergenerational gameplay: Evaluating social interaction between younger and older players. In J. A. Konstan (Ed.), CHI '12 extended abstracts on human factors in computing systems (pp. 2333–2338). New York, NY: ACM. URL: https://doi.org/10.1145/2212776.2223798

[17] Chou, W.-H., Li, Y.-C., Chen, Y.-F., Ohsuga, M., & Inoue, T. (2022). Empirical Study of Virtual Reality to Promote Intergenerational Communication: Taiwan Traditional Glove Puppetry as Example. *Sustainability*, *14*(6), 3213. URL: https://doi.org/10.3390/su14063213

[18] Pais, P., Lopes, I., Pereira, F., Vieira. (2022). Asymmetric Roles in Intergenerational Games. Nova School of Science and Technology, Nova University Lisbon. URL: http://hdl.handle.net/10362/138804

[19] Bacca, F., Cerezo, E., Gil, R., Aguelo, A., Blasco, A. C., Coma, T., & Garrido, M. A. (2021). Intergenerational Playful Experiences Based on Digital Games for Interactive Spaces. In Design, Learning, and Innovation: 5th EAI International Conference, DLI 2020, Virtual Event, December 10-11, 2020, Proceedings 5 (pp. 101-119). Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-78448-5_8

Glosario

Intent: Representa una acción que cumple una petición por voz del usuario. Opcionalmente, pueden tener argumentos denominados slots. Los intents se especifican en un documento JSON llamado esquema de interacción. Por ejemplo, un intent que diga la hora se podría llamar mediante el comando “Alexa, ¿qué hora es?”.

Slots: Parámetros que puede tener un intent, que pueden ser palabras, números, frases, etc. Por ejemplo, “Alexa, ¿qué hora es en [ciudad]?”. Pueden ser de varios tipos, algunos predefinidos por Amazon como el tipo número o el tipo fecha, y personalizados, como por ejemplo nombres de verduras.

Utterances: Son las frases asociadas a un intent que el usuario puede decir para invocarlo. Se pueden añadir todas las necesarias. Por ejemplo, “Alexa, ¿Qué hora es?” y “Alexa, dime la hora” podrían pertenecer al mismo intent y, por tanto, devolver la misma información. Si una utterance tiene un slot asociado y el usuario no proporciona un valor válido de ese slot, no se detectará a qué está llamando. Válido: “Alexa, ¿qué hora es en [madrid]?”. No válido: “Alexa, ¿qué hora es en [manzana]?”.

Modelo de Interacción: Es un archivo JSON donde se encuentran todos los intents con sus utterances y slots. Este es el archivo consultado para asociar una utterance con su intent correspondiente y mandárselo a Lambda.

Handler: Es el código responsable de realizar las acciones necesarias de las peticiones entrantes. Tiene dos funciones: canHandle(), que comprueba si puede aceptar la petición, y handle() que, en caso de que canHandle devuelva un valor cierto, realiza todas las acciones del intent que lo invoca y puede devolver una salida de texto, que pronunciará Alexa como respuesta a la petición del usuario, y una interfaz si el dispositivo tiene pantalla. Se suele crear un handler por cada intent de la aplicación.