



Universidad de Granada

Escuela Técnica Superior de Ingenierías Informática y de  
Telecomunicación

Trabajo de Fin de Grado (TFG)

Juego en Alexa para combatir el aislamiento  
de las personas mayores

Número asignado: X

**Autora**

Marina Jun Carranza Sánchez

**Tutora**

Nuria Medina Medina

**Resumen:**

[Aquí iría el resumen del TFG]

**Abstract:**

[Aquí iría el resumen en inglés]

**Palabras clave:** Gamificación, aislamiento, personas mayores, asistentes de voz, asistentes conversacionales, Alexa, skill, envejecimiento activo, juegos serios.

**Key words:** Gamification, isolation, elder people, voice assistants, chatbots, Alexa, skill, active aging, serious games.

# Índice

|  |           |
|--|-----------|
| <b>1. Introducción</b>   | <b>10</b> |
| 1.1. Motivación y contexto . . . . .   | 10        |
| 1.1.1. Impacto social y tecnológico de la pandemia de COVID-19 . . . . .                 | 12        |
| 1.2. Objetivos . . . . .   | 15        |
| <b>2. Estado del arte</b>  | <b>16</b> |
| 2.1. Juegos serios . . . . .   | 16        |
| 2.1.1. Origen de los juegos serios . . . . .   | 16        |
| 2.1.2. Concepto de juego serio . . . . .   | 17        |
| 2.1.3. Juegos serios en la rehabilitación neuropsicológica . . . . .                     | 18        |
| 2.1.4. Juegos tradicionales en grupo y sus ventajas cognitivas . . . . .                 | 19        |
| 2.2. Juegos digitales para adultos mayores . . . . .                                     | 19        |
| 2.2.1. El proyecto WorthPlay . . . . .   | 21        |
| 2.3. Estado del arte de los asistentes virtuales . . . . .                               | 22        |
| 2.3.1. Influencia de la IA y clasificación . . . . .                                     | 22        |
| 2.3.2. Alexa, el asistente conversacional más extendido hoy en día . . . . .             | 24        |
| 2.4. Aplicaciones similares . . . . .  | 25        |
| 2.4.1. CELIA, mucho más que una asistente . . . . .                                      | 25        |
| 2.4.2. Juego de los trayectos orientado a la detección del deterioro cognitivo . . . . . | 25        |
| 2.4.3. Skill de Alexa para la mejora de la inhibición de respuesta . . . . .             | 26        |
| <b>3. Planificación, metodología y presupuesto</b>                                       | <b>28</b> |
| 3.1. Planificación temporal . . . . .  | 28        |
| 3.2. Metodología . . . . .   | 31        |
| 3.3. Presupuesto . . . . .   | 31        |
| 3.3.1. Recursos humanos . . . . .  | 32        |
| 3.3.2. Hardware . . . . .  | 32        |
| 3.3.3. Software . . . . .  | 32        |
| 3.3.4. Resumen de presupuesto . . . . .  | 33        |
| <b>4. Análisis del problema</b>  | <b>34</b> |
| 4.1. Elección de la temática: el juego de la oca . . . . .                               | 34        |
| 4.1.1. Lista de minijuegos dentro de la oca . . . . .                                    | 36        |
| 4.2. Historias de usuario y requisitos . . . . .   | 38        |

|           |  |           |
|-----------|--|-----------|
| 4.2.1.    | Históricas de usuario . . . . .                              | 39        |
| 4.2.2.    | Requisitos funcionales . . . . .                             | 43        |
| 4.2.3.    | Requisitos no funcionales . . . . .                          | 44        |
| 4.3.      | Casos de uso y sus correspondientes diagramas . . . . .      | 46        |
| 4.3.1.    | Casos de uso . . . . .                                       | 46        |
| 4.3.2.    | Matriz de cobertura de requisitos funcionales . . . . .      | 58        |
| 4.3.3.    | Diagramas de secuencia de casos de uso . . . . .             | 59        |
| <b>5.</b> | <b>Diseño</b>  | <b>63</b> |
| 5.1.      | Diseño de la arquitectura . . . . .                          | 63        |
| 5.2.      | GDD: Documento de Diseño del Juego . . . . .                 | 65        |
| 5.3.      | Modelo conceptual del juego de la oca . . . . .              | 68        |
| 5.4.      | Diseño de la interfaz de usuario . . . . .                   | 68        |
| 5.4.1.    | Bocetos y mockups . . . . .                                  | 70        |
| 5.4.2.    | Diagrama de flujo entre pantallas . . . . .                  | 71        |
| 5.4.3.    | Cuestiones de estética, usabilidad y accesibilidad . . . . . | 72        |
| <b>6.</b> | <b>Análisis tecnológico</b>                                  | <b>74</b> |
| 6.1.      | Fundamentos de una skill . . . . .                           | 74        |
| 6.1.1.    | Términos comunes en el desarrollo de skills . . . . .        | 74        |
| 6.1.2.    | Memoria y persistencia de datos . . . . .                    | 74        |
| 6.2.      | Alexa Skills Kit (ASK) . . . . .                             | 74        |
| 6.3.      | AWS Serverless Platform . . . . .                            | 74        |
| 6.4.      | Alexa Presentation Language (APL) . . . . .                  | 75        |
| <b>7.</b> | <b>Desarrollo</b>  | <b>76</b> |
| 7.1.      | Creación de la skill y el modelo de interacción . . . . .    | 76        |
| 7.2.      | Configuración de servicios de AWS . . . . .                  | 80        |
| 7.2.1.    | Identity and Access Management (IAM) . . . . .               | 81        |
| 7.2.2.    | Amazon Simple Storage Service (S3) . . . . .                 | 84        |
| 7.2.3.    | DynamoDB . . . . .   | 85        |
| 7.3.      | Implementación y estructura del código . . . . .             | 87        |
| 7.3.1.    | Lógica del juego de la oca . . . . .                         | 88        |
| 7.3.2.    | Fichero «handlers.js» . . . . .                              | 100       |
| 7.3.3.    | Fichero «index.js» . . . . .                                 | 104       |
| 7.3.4.    | Directorio «apl» . . . . .                                   | 105       |
| 7.3.5.    | Directorio «exports» . . . . .                               | 112       |

|   |            |
|---|------------|
| 7.3.6. Fichero «db.js» . . . . .          | 115        |
| 7.4. Pruebas . . . . .                    | 119        |
| 7.5. Seguridad . . . . .                  | 119        |
| <b>8. Manual de uso de la skill</b>       | <b>121</b> |
| <b>9. Conclusiones y trabajos futuros</b> | <b>125</b> |
| 9.1. Consecución de objetivos . . . . .   | 125        |
| 9.2. Trabajos futuros . . . . .           | 126        |
| <b>10. Bibliografía</b>                   | <b>127</b> |

# Índice de figuras

|     |   |    |
|-----|---|----|
| 1.  | Pirámides de población de España en futuros años ( <a href="#">geriatricarea.com</a> ) . . . . .                            | 11 |
| 2.  | Población por grupos de edad que han usado Internet en los últimos tres meses (INE, 2018-23) . . . . .                      | 13 |
| 3.  | Población de entre 16 y 74 años que ha usado Internet en los últimos tres meses en la UE (INE, 2020-23) . . . . .           | 14 |
| 4.  | Gradior, un sistema de evaluación y rehabilitación neuropsicológica . . . . .   | 18 |
| 5.  | Mejorar de calidad de vida de adultos mayores con las TIC . . . . .   | 21 |
| 6.  | Ranking de asistentes de voz según el VPIR en 2020 ( <a href="#">statista</a> ) . . . . .                                   | 24 |
| 7.  | Celia, una asistente todoterreno 24 horas al día. . . . .   | 25 |
| 8.  | Interfaz gráfica del juego colores v2 . . . . .   | 26 |
| 9.  | Planificación temporal en el calendario . . . . .   | 28 |
| 10. | Leyenda de la planificación temporal por (sub)tareas . . . . .  | 29 |
| 11. | Diseño de la oca para evaluar las habilidades motoras en Educación Primaria (Ortín et al., <a href="#">2008</a> ) . . . . . | 34 |
| 12. | Tablero completo correspondiente a las casillas de la figura superior (Ortín et al., <a href="#">2008</a> ) . . . . .       | 35 |
| 13. | Un tablero del juego tradicional de la oca . . . . .  | 36 |
| 14. | Patrón general de una historia de usuario . . . . .   | 38 |
| 15. | Matriz de cobertura de requisitos funcionales . . . . .   | 58 |
| 16. | Diagrama de secuencia de CU01 y CU02. . . . .   | 59 |
| 17. | Diagrama de secuencia de CU03, CU04 y CU05. . . . .   | 60 |
| 18. | Diagrama de secuencia de CU06 y CU07 y CU13. . . . .  | 61 |
| 19. | Diagrama de secuencia de CU08, CU09, CU10, CU11 y CU12. . . . .   | 62 |
| 20. | Arquitectura de una skill de Alexa con las tecnologías AWS (Stafford, <a href="#">2018</a> ) . . . . .                      | 63 |
| 21. | Arquitectura de una APL con los cuatro actores principales ( <a href="#">Alexa Developer Documentation</a> ) . . . . .      | 64 |
| 22. | Game Design Document página 1 . . . . .   | 66 |
| 23. | Game Design Document página 2 . . . . .   | 67 |
| 24. | Diagrama conceptual del juego de la oca ( <a href="#">Visual Paradigm</a> ) . . . . .                                       | 68 |
| 25. | Tablero personalizado para la skill del juego de la oca . . . . .   | 69 |
| 26. | Boceto de la pantalla de inicio de la skill . . . . .   | 70 |
| 27. | Boceto de la pantalla con los participantes . . . . .   | 70 |
| 28. | Boceto de la pantalla de lanzamiento de dado . . . . .  | 71 |
| 29. | Boceto de pantalla con la casilla y participante actual . . . . .   | 71 |

|     |   |     |
|-----|---|-----|
| 30. | Diagrama de flujo entre pantallas . . . . .   | 72  |
| 31. | Ejemplo de documento APL básico ( <a href="#">Alexa Developer Documentation</a> ) . . . . . | 75  |
| 32. | Menú de creación de una skill de Alexa . . . . .  | 77  |
| 33. | Menú principal de montaje de la skill de Alexa . . . . .                                    | 78  |
| 34. | Menú del código de la skill de Alexa . . . . .  | 79  |
| 35. | Menú de la consola de desarrolladores de AWS . . . . .                                      | 80  |
| 36. | Menú de creación de un rol IAM . . . . .  | 81  |
| 37. | Rol creado para la skill y sus permisos en AWS . . . . .                                    | 82  |
| 38. | Configuración de las relaciones de confianza del rol . . . . .                              | 83  |
| 39. | El AWS Lambda Execution Role ARN de la skill . . . . .                                      | 84  |
| 40. | Bucket de Amazon S3 para la skill de la oca . . . . .                                       | 85  |
| 41. | Tablas <i>JuegoOca</i> y <i>Jugador</i> creadas en DynamoDB . . . . .                       | 86  |
| 42. | Información de la tabla <i>Jugador</i> de DynamoDB . . . . .                                | 87  |
| 43. | Estructura del código de la función Lambda . . . . .  | 87  |
| 44. | Clase jugador junto con su constructor . . . . .  | 88  |
| 45. | Funciones que componen el módulo del dado . . . . .   | 89  |
| 46. | Importaciones de las baterías de preguntas de los minijuegos . . . . .                      | 89  |
| 47. | Declaración de la clase de una casilla normal . . . . .                                     | 90  |
| 48. | Declaración de la clase de casillas de penalización . . . . .                               | 91  |
| 49. | Clase para casilla del minijuego verdadero o falso . . . . .                                | 92  |
| 50. | Método para obtener una pregunta aleatoria sobre un compañero . . . . .                     | 92  |
| 51. | Método para obtener la solución de una pregunta de fecha . . . . .                          | 93  |
| 52. | Cálculo de días de la semana o meses a partir de la fecha actual . . . . .                  | 93  |
| 53. | Cálculo de la estación del año partiendo de la fecha actual . . . . .                       | 94  |
| 54. | Declaración de la clase Tablero . . . . .   | 95  |
| 55. | Métodos para calcular una posición y buscar un tipo de casilla . . . . .                    | 96  |
| 56. | Simula la estructura de datos <i>enum</i> de los estados del juego . . . . .                | 96  |
| 57. | Función que devuelve un mensaje de ayuda según el estado del juego . . . . .                | 97  |
| 58. | Importaciones y declaración de la clase del juego de la oca . . . . .                       | 97  |
| 59. | Método de inicialización de participantes del juego . . . . .                               | 98  |
| 60. | Método de inicialización del tablero de la oca . . . . .                                    | 98  |
| 61. | Método para pasar al siguiente turno . . . . .  | 99  |
| 62. | Método para anunciar los ganadores al final del juego . . . . .                             | 99  |
| 63. | Primeras líneas del método para avanzar las fichas . . . . .                                | 100 |
| 64. | Importaciones de bibliotecas y módulos para los handlers . . . . .                          | 100 |
| 65. | Todos los handlers que se exportan al <i>index.js</i> . . . . .                             | 101 |

|     |   |     |
|-----|---|-----|
| 66. | Exportación de manejadores a la función Lambda . . . . .                | 105 |
| 67. | Directorio de ficheros JSON para APL . . . . .                          | 105 |
| 68. | Documento APL para la pantalla de inicio . . . . .                      | 106 |
| 69. | Pantalla de bienvenida de la skill . . . . .                            | 107 |
| 70. | Documento APL para mostrar los participantes del juego . . . . .        | 108 |
| 71. | Pantalla postconfiguración mostrando a los participante . . . . .       | 109 |
| 72. | Documento APL para mostrar la animación del dado . . . . .              | 109 |
| 73. | Pantalla mostrando un lanzamiento de dado . . . . .                     | 110 |
| 74. | Documento APL para mostrar la casilla y participante actuales . . . . . | 111 |
| 75. | Pantalla mostrando la casilla y participante actuales . . . . .         | 112 |
| 76. | Directorio de ficheros exportados a otros módulos . . . . .             | 112 |
| 77. | Ejemplo de objeto con la información de una casilla normal . . . . .    | 113 |
| 78. | Ejemplos de preguntas en «Adivina la cifra» . . . . .                   | 113 |
| 79. | Ejemplos de preguntas de «Verdadero o falso» . . . . .                  | 114 |
| 80. | Ejemplos de preguntas acerca de los compañeros de juego . . . . .       | 114 |
| 81. | Ejemplos de preguntas acerca en «Recuerda la fecha» . . . . .           | 115 |
| 82. | Función para asumir un rol IAM con permisos temporales . . . . .        | 116 |
| 83. | Función para guardar los datos de la partida en DynamoDB . . . . .      | 117 |
| 84. | Función para cargar los datos de la partida desde DynamoDB . . . . .    | 118 |
| 85. | Función para cargar los datos de la partida desde DynamoDB . . . . .    | 119 |

# Índice de cuadros

|     |   |     |
|-----|---|-----|
| 1.  | Subobjetivos del trabajo . . . . .                          | 15  |
| 2.  | Clasificación de asistentes virtuales . . . . .             | 23  |
| 3.  | Presupuesto para recursos humanos . . . . .                 | 32  |
| 4.  | Presupuesto para hardware . . . . .                         | 32  |
| 5.  | Presupuesto para software . . . . .                         | 32  |
| 6.  | Tabla general de presupuestos . . . . .                     | 33  |
| 7.  | Historia de usuario nº 1 . . . . .                          | 39  |
| 8.  | Historia de usuario nº 2 . . . . .                          | 39  |
| 9.  | Historia de usuario nº 3 . . . . .                          | 39  |
| 10. | Historia de usuario nº 4 . . . . .                          | 40  |
| 11. | Historia de usuario nº 5 . . . . .                          | 40  |
| 12. | Historia de usuario nº 6 . . . . .                          | 40  |
| 13. | Historia de usuario nº 7 . . . . .                          | 40  |
| 14. | Historia de usuario nº 8 . . . . .                          | 40  |
| 15. | Historia de usuario nº 9 . . . . .                          | 41  |
| 16. | Historia de usuario nº 10 . . . . .                         | 41  |
| 17. | Historia de usuario nº 11 . . . . .                         | 41  |
| 18. | Historia de usuario nº 12 . . . . .                         | 42  |
| 19. | Historia de usuario nº 13 . . . . .                         | 42  |
| 20. | Caso de uso nº 1 . . . . .                                  | 46  |
| 21. | Caso de uso nº 2 . . . . .                                  | 47  |
| 22. | Caso de uso nº 3 . . . . .                                  | 48  |
| 23. | Caso de uso nº 4 . . . . .                                  | 49  |
| 24. | Caso de uso nº 5 . . . . .                                  | 50  |
| 25. | Caso de uso nº 6 . . . . .                                  | 51  |
| 26. | Caso de uso nº 7 . . . . .                                  | 51  |
| 27. | Caso de uso nº 8 . . . . .                                  | 52  |
| 28. | Caso de uso nº 9 . . . . .                                  | 53  |
| 29. | Caso de uso nº 10 . . . . .                                 | 54  |
| 30. | Caso de uso nº 11 . . . . .                                 | 55  |
| 31. | Caso de uso nº 12 . . . . .                                 | 56  |
| 32. | Caso de uso nº 13 . . . . .                                 | 57  |
| 33. | Dimensiones y variables de la ergonomía de la app . . . . . | 73  |
| 34. | Lista de comandos de voz y ámbito de uso . . . . .          | 121 |

|  |     |
|--|-----|
| 35. Tipos de respuestas del usuario a los minijuegos . . . . . | 124 |
|--|-----|

# **1. Introducción**

Este Trabajo de Fin de Grado se enfoca en el desarrollo de un juego para asistentes conversacionales dirigido a personas mayores con mayor riesgo de aislamiento social. Reconociendo los desafíos que enfrenta este grupo demográfico en la era digital, el proyecto busca integrar la tecnología de manera accesible y lúdica. El objetivo es no solo fomentar la conexión social, sino también promover la estimulación cognitiva y el bienestar emocional de las personas mayores.

Dado que las personas mayores están menos familiarizadas con el uso de herramientas digitales en su vida cotidiana puede parecer difícil que se sientan atraídas por juegos que se basan en la tecnología, sin embargo si las personas usuarias en un primer momento comprueban que no solo pueden acceder a este entretenimiento si no que, además se incentivan sus relaciones personales, conocen nuevas amistades y se abre un mundo nuevo de ocio y desarrollo intelectual, comprobaremos que su aislamiento se reduce día a día.

La idea detrás de este TFG podría resumirse con la siguiente cita: «Las actividades físicas, cognitivas y emocionales en edades avanzadas son cruciales para estimular la actividad cerebral y contribuir al mantenimiento de la calidad de vida. Además de los beneficios físicos y cerebrales, los juegos estimulan la interacción social y contribuyen a la socialización y al mantenimiento de la salud emocional y afectiva» (Azevedo & Lemos, 2022).

## **1.1. Motivación y contexto**

Este TFG se integra dentro del proyecto de investigación «Evaluación del uso de robots sociales y sistemas conversacionales en residencias y centros de día para promover el envejecimiento saludable» de la Universidad de Granada, con la profesora Nuria Medina Medina como investigadora principal del mismo, y directora de este Trabajo de Fin de Grado.

Este proyecto, con código C-ING-179-UGR23, «propone el uso y evaluación de Agentes Sociales Interactivos (SIA - Social Interactive Agent) (en particular de Robots Sociales apoyados por Asistentes Conversacionales) que promuevan el envejecimiento saludable y las interacciones sociales de los mayores en la residencia o centro de día, ya que estas interacciones son muy importantes para el mejoramiento de la salud y estado anímico de los mayores. Para maximizar su efectividad, la propuesta integrará experiencias lúdicas y técnicas de interacción multimodal».

Dicho proyecto destaca por su metodología pionera en relación con el estado del arte ya que se focaliza en la utilización de los robots sociales para fomentar el envejecimiento saludable y mejorar la comunicación y los problemas de interacción social en las residencias. El proyecto también se centra en el diseño de experiencias lúdicas integradas en entornos sociales para favorecer los problemas de motivación y adopción de la tecnología.

Que el uso de la tecnología puede ser un apoyo importante para lograr una mayor calidad de vida en los adultos mayores que viven en residencias o asisten a centros de día es una de las premisas en las que se apoya la investigación. «Consecuentemente, este proyecto propone el uso y evaluación de Agentes Sociales Interactivos (SIA - Social Interactive Agent) (en particular de Robots Sociales apoyados por Asistentes Conversacionales) que promuevan el envejecimiento saludable y las interacciones sociales de los mayores en la residencia o centro de día, ya que estas interacciones son muy importantes para el mejoramiento de la salud y estado anímico de los mayores. Para maximizar su efectividad, la propuesta integrará experiencias lúdicas y técnicas de interacción multimodal».

Según se menciona en el artículo *La soledad y el aislamiento social en las personas mayores* (Arruebarrena & Cabaco, 2020), «El aislamiento social se define como una ausencia objetiva de relaciones/contactos sociales y la soledad como la experiencia subjetiva aversiva que se siente al valorar esas relaciones/contactos sociales como insuficiente en cantidad y/o calidad»

En los últimos tiempos, el tema de la soledad en las personas mayores ha ganado atención en los medios de comunicación, describiéndose como una «epidemia» en aumento. Aunque no hay evidencia sólida que respalte la idea de una nueva epidemia, las dificultades metodológicas y la falta de consenso en la medición de la soledad limitan la capacidad de confirmar si las personas mayores se sienten más solas que antes.

A pesar de estas limitaciones, estudios indican tasas de soledad entre las personas mayores en España, oscilando entre el 14 % y el 24 %, e incluso alcanzando el 40 % en algunos casos. Uno de los factores que contribuyen a esta percepción es el aumento en el número de personas mayores viviendo solas. Se proyecta que para 2050, aproximadamente un tercio de la población tendrá más de 65 años, lo que por lógica implica un aumento en el número de personas mayores que viven solas.

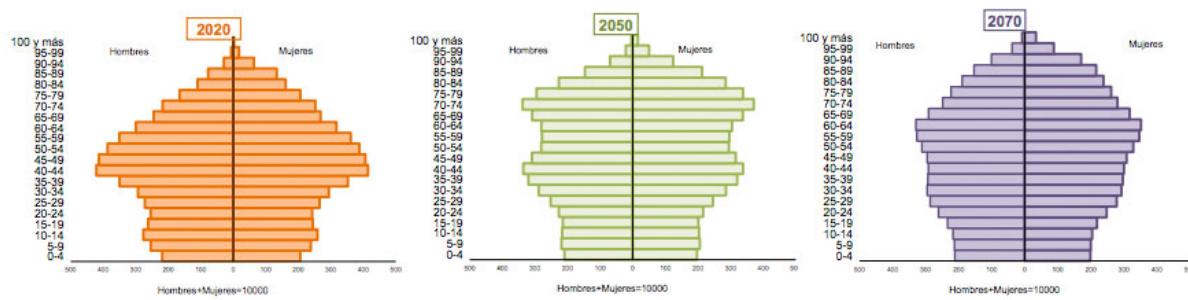


Figura 1: Pirámides de población de España en futuros años ([geriatricarea.com](http://geriatricarea.com))

El aumento de la esperanza de vida de la población adulta en nuestro país ha supuesto que las personas mayores, muchas de ellas que viven solas y poseen nivel económico y cultural medio, supongan un sector amplio de la población que requiere de nuevas formas de ocio y de relacionarse

socialmente. Muchas de estas personas viven solas y el sedentarismo y la falta de interacción social les produce que su desarrollo cognitivo se vea mermado.

También la frecuente automarginación de este grupo demográfico para el uso de herramientas digitales como juegos o aplicaciones, el fenómeno conocido de forma general como brecha digital, contribuye a un mayor aislamiento social. «Muchos adultos mayores tienen acceso a dispositivos móviles, pero no pueden aprovecharlos completamente debido a la falta de conocimiento o el miedo a salir de su zona de confort. Esto resulta en barreras emocionales, dificultades para adquirir nuevas habilidades tecnológicas» (López Garzón & Rozo Barrera, 2023).

A pesar de lo mencionado en el párrafo anterior, el segmento de edad mayor de 60 años no es ajeno a la realidad de que las formas de socialización del siglo XXI están vinculadas a los avances tecnológicos y a las nuevas experiencias lúdicas. Como se señala en *Las competencias digitales en personas mayores: de amenaza a oportunidad*: «el potencial que para las personas mayores ofrece el uso habitual de las TIC es enorme, con una larga lista de oportunidades existentes para el beneficio de este colectivo que deben ser aprovechadas» (Bunbury Bustillo et al., 2022).

Un juego digital que suponga entretenimiento para las personas mayores, al mismo tiempo que una mejora de memoria y ampliación de lenguaje y percepción puede significar un estimulante cambio en su día a día.

### **1.1.1. Impacto social y tecnológico de la pandemia de COVID-19**

Durante la pandemia por el COVID-19 millones de personas en el mundo tuvieron que pasar en pocos días de trabajo presencial al online y en sus relaciones personales debieron adaptar sus costumbres al nuevo escenario virtual.

De esta manera, personas acostumbradas a comunicarse solo por llamadas telefónicas, y muy poco mediante telefonía móvil, se volvieron usuarias de videollamadas diarias ante la necesidad de mantenerse conectados con familiares y amigos, combinada con las restricciones de movimiento.

Este cambio tuvo un impacto profundo en la vida diaria de las personas mayores. Por un lado, les brindó una forma vital de mantenerse conectados con sus seres queridos, incluso cuando no podían reunirse físicamente debido a las medidas de distanciamiento social. Esto ayudó a reducir un poco el riesgo de aislamiento social y proporcionó un medio para el apoyo emocional y la interacción social, lo cual es esencial para su bienestar mental y emocional.

Sin embargo, la transición a las tecnologías digitales también presentó desafíos, especialmente para aquellos menos familiarizados con ellas. Algunos enfrentaron dificultades técnicas al principio, como la configuración de aplicaciones o la resolución de problemas de conexión. Además, la dependencia excesiva de la tecnología para la comunicación también puede aumentar la brecha digital entre aquellos que tienen acceso y conocimientos tecnológicos y aquellos que no los tienen, lo que potencialmente podría aumentar el riesgo de exclusión social para algunos adultos mayores.

A pesar de estos desafíos, la pandemia actuó como un catalizador para la adopción de tecnología entre las personas mayores, lo que les permitió permanecer conectados y participar en la sociedad de manera más activa, incluso en tiempos de crisis. Como resultado, muchas personas mayores han incorporado el uso de tecnologías digitales en su vida diaria incluso después de que levantaran las restricciones de la pandemia, lo que les brinda nuevas oportunidades de participación social y acceso a recursos y servicios en línea (Palacios-Rodríguez et al., [s.f.](#)).

Según datos del Instituto Nacional de Estadística (INE), la población que usa Internet (en los últimos meses el uso de las tecnologías de información y comunicación (TIC) en los hogares ha crecido en los últimos años, si bien sigue existiendo una brecha entre los usuarios y no usuarios (brecha digital) que se puede atribuir a una serie de factores: la falta de infraestructura (en particular en las zonas rurales), la falta de conocimientos de informática y habilidades necesarias para participar en la sociedad de la información, o la falta de interés en lo que la sociedad de la información puede ofrecer).

Al aumentar la edad desciende el uso de Internet, siendo el porcentaje más bajo el que corresponde al grupo de edad de 65 a 74 años (un 79,7 % para los hombres y un 80,5 % para las mujeres) (Instituto Nacional de Estadística, [2023](#)).

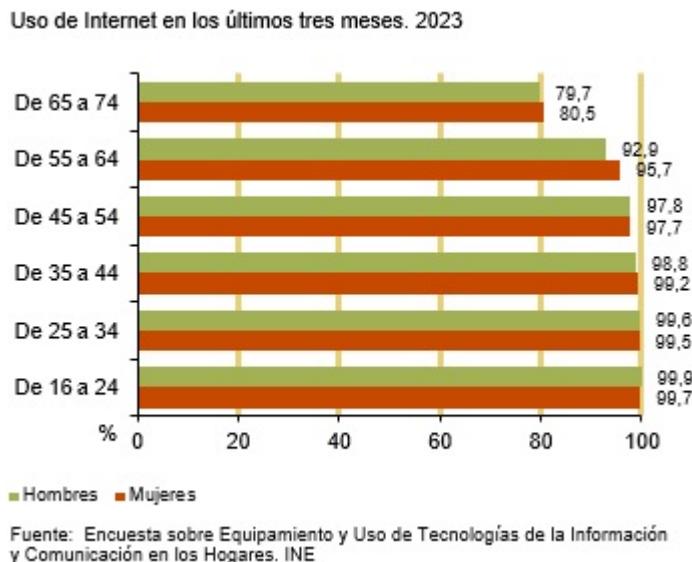
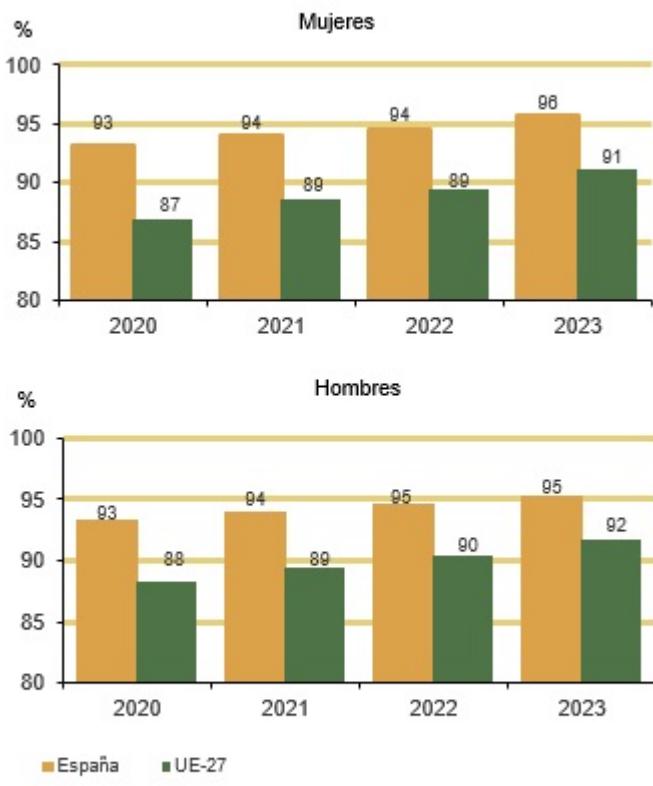


Figura 2: Población por grupos de edad que han usado Internet en los últimos tres meses (INE, 2018-23)

Uso de Internet en los últimos tres meses. España y UE-27.  
Serie 2020-2023



Notas: UE-27: 27 países (desde 2020)  
En UE-27 dato estimado en 2020 y 2023 y ruptura de serie en 2021  
Fuente: Estadísticas de sociedad de la información. Eurostat

Figura 3: Población de entre 16 y 74 años que ha usado Internet en los últimos tres meses en la UE (INE, 2020-23)

Según una encuesta realizada por Canal Sénior, plataforma online de entretenimiento y aprendizaje para personas mayores de 55 años, entre sus usuarios, en el capítulo de juegos y aplicaciones de entretenimiento: «la mayoría de los encuestados prefieren los juegos que permiten el entrenamiento mental, como aplicaciones del tipo Trivial, Apalabradados, Scrabble, Wordle, etc. Esto es muy relevante en personas senior, pues está probado que los juegos que retan a la mente y nos hacen pensar pueden retrasar el envejecimiento cognitivo durante más tiempo».

Todo hace indicar que cada vez hay una relación más estrecha entre mayores y ocio digital. «Es también relevante ver cómo algunos de sus gustos y preferencias son diferentes a los de otros grupos de edad, como por ejemplo su preferencia por la lectura al consumo de series y películas o de contenidos en las redes sociales. Por último, conviene destacar que el uso de las tecnologías digitales como parte de nuestro ocio pueden ayudarnos a tener un envejecimiento activo, sano y con participación elevada en la sociedad.» (Canal Sénior, 2022).

## 1.2. Objetivos

El presente trabajo tiene como objetivo desarrollar un juego digital, que contribuya al creciente conjunto de soluciones innovadoras para combatir el aislamiento social en los adultos mayores.

Este juego no solo buscará proporcionar entretenimiento y diversión, sino que también se centrará en promover la interacción social, el compromiso cognitivo y emocional, y en general, mejorar la calidad de vida de las personas mayores.

Para lograr este objetivo principal, se puede descomponer en varios subobjetivos, recogidos en la siguiente tabla:

| Subobjetivo                     | Descripción  |
|---------------------------------|--|
| 1. Revisión de la literatura    | 1.1. Identificar las investigaciones clave sobre el impacto del aislamiento social en personas mayores<br>1.2. Analizar la diversidad de intervenciones digitales dirigidas a este grupo demográfico   |
| 2. Diseño del juego             | 2.1. Investigar las mejores prácticas en el diseño de juegos digitales accesibles para personas mayores<br>2.2. Considerar las adaptaciones necesarias para abordar posibles limitaciones físicas y cognitivas   |
| 3. Desarrollo técnico           | 3.1. Seleccionar la plataforma y tecnologías más apropiadas para el desarrollo del juego<br>3.2. Asegurar la compatibilidad con dispositivos comunes utilizados por personas mayores<br>3.3. Integrar funcionalidades de accesibilidad, como ajustes de tamaño de fuente y navegación simplificada |
| 4. Contribución al conocimiento | 4.1. Contribuir al creciente cuerpo de conocimientos sobre cómo la tecnología puede mejorar la calidad de vida de los adultos mayores, particularmente en el contexto del aislamiento social.  |
| 5. Participación comunitaria    | 5.1. Colaborar con comunidades de personas mayores para obtener retroalimentación.<br>5.2. Organizar pruebas piloto y grupos de enfoque para evaluar la experiencia del usuario.   |

Cuadro 1: Subobjetivos del trabajo.

## **2. Estado del arte**

En la intersección entre el avance tecnológico y el envejecimiento de la población, los juegos digitales emergen como una herramienta multifacética con el potencial de promover el bienestar y la calidad de vida de los adultos mayores. El estado del arte representa una síntesis dinámica de la investigación, desarrollo y aplicaciones existentes en este campo en constante evolución.

En esta revisión, se explorarán las principales tendencias y avances en cuatro áreas clave: los juegos serios, diseñados con objetivos específicos de aprendizaje o rehabilitación; los juegos digitales, adaptados para satisfacer las necesidades y preferencias de las personas mayores; el estado actual de las tecnologías aplicadas, incluyendo dispositivos y software especializado en el diseño y finalmente, ejemplos de aplicaciones prácticas desarrolladas en esta línea de investigación.

### **2.1. Juegos serios**

Los juegos *serios* se distinguen por su enfoque principal en objetivos educativos o formativos, relegando la diversión a un segundo plano (Ferrer, 2018). Mientras que los juegos tradicionales buscan principalmente entretener al jugador, los juegos serios utilizan la pedagogía para integrar la instrucción dentro de la experiencia de juego. Esto implica que, aunque la diversión sigue siendo un elemento presente, el aprendizaje se convierte en el objetivo primordial.

#### **2.1.1. Origen de los juegos serios**

Hay que remontarse al año 1970 para datar la fecha de aparición del término juego serio o *serious game*. En el contexto de los juegos de mesa, Clark Abt, investigador de EEUU publicó un libro titulado «*Serious Games*» (Abt, 1970), en el que explora las formas en que los juegos pueden utilizarse para instruir e informar. Utiliza enfoques innovadores para la resolución de problemas mediante técnicas de juego en contextos tan diferentes como el aprendizaje de las matemáticas, ciencias sociales y física o las tareas de planificación en el gobierno y la industria.

Pero no es hasta 2002 cuando el concepto de juego serio, tal como se conoce hoy en día, se recoge por Ben Sawyer y David Rejeski en un artículo sobre los juegos serios en el ámbito de la política pública (Sawyer & Rejeski, 2002), que tuvo un gran impacto, gracias a la creación de una asociación para promover el uso de los videojuegos con un propósito serio. Dicha asociación, Serious Games Initiative, de la cuál es cofundador Sawyer a través del Woodrow Wilson Center continúa activa en la actualidad (Wilson Center, 2024).

Precisamente, es Ben Sawyer quien establece como primer juego serio el videojuego America's Army (Gudmundsen, 2006), un simulador de combate para instruir a militares en el campo de batalla a través de la realización de diferentes misiones.

## 2.1.2. Concepto de juego serio

Los juegos serios son una categoría de videojuegos desarrollados principalmente para propósitos educativos y formativos, aunque también pueden tener elementos de entretenimiento. Estos juegos se distinguen por su enfoque en la transmisión de conocimientos y habilidades, a menudo relacionados con temas como la política, la salud, el entrenamiento militar, la educación, el ámbito empresarial, etc (Marcano, 2008).

Se listan a continuación algunas de las características principales que distinguen a los juegos serios del resto:

- **Fin educativo:** su principal objetivo es la formación en lugar del entretenimiento. La adquisición de habilidades técnicas y comprensión de procesos complejos.
- **Realismo:** están vinculados a aspectos de la realidad, favoreciendo la identificación del jugador con el contexto que se está simulando.
- **Ambiente virtual seguro:** proporcionan un entorno tridimensional virtual seguro para la práctica en ciertas áreas, como el entrenamiento militar.
- **Interés temático:** pueden abordar temas políticos, económicos, psicológicos, religiosos, entre otros, a menudo con un enfoque en la educación y el entrenamiento.

Los siguientes son algunos ejemplos reales y de ámbito específico de videojuegos serios:

- **Biomedical Training:** entrenamiento para trabajadores de la salud en emergencias.
- **Food Force:** educación sobre el Programa Mundial de Alimentos de la ONU.
- **Incident Commander:** dirección de acciones en crisis sociales y desastres naturales.
- **Hazmat: Hotzone:** simulación para bomberos en situaciones de emergencia.
- **Yourselfitness:** regímenes de ejercicios físicos y aeróbicos.
- **Real Life 2007:** simulación global para entender condiciones de vida en diferentes países.

Los juegos serios son una herramienta poderosa para el aprendizaje y la formación en distintos campos. Involucran principios pedagógicos y cognitivos para garantizar un entrenamiento efectivo y ofrecen diversas ventajas como la familiarización con interfaces tecnológicas y la mejora de habilidades necesarias para la sociedad digital.

En el contexto de los adultos mayores, los videojuegos serios pueden resultar valiosos para el mantenimiento cognitivo, la estimulación mental y la mejora de habilidades específicas.

### 2.1.3. Juegos serios en la rehabilitación neuropsicológica

Cabe destacar el papel de los juegos serios en el ámbito de la informática médica, en concreto, en el entrenamiento y la rehabilitación neuropsicológica. «En la rehabilitación cognitiva los retos de un juego serio por lo general inciden directamente en un déficit específico, lo que puede repercutir al mismo tiempo en más de uno.» (Regalón & Céspedes, 2019)

En los últimos años, ha habido un notable aumento en la investigación y desarrollo de juegos computarizados para la rehabilitación cognitiva. Ejemplos como RehaCom® y Gradior® (Vanova et al., 2018) han demostrado resultados positivos en varios centros de salud al ofrecer una variedad de ejercicios digitales que abordan áreas cognitivas clave como la atención, la percepción, la memoria y el lenguaje.



Figura 4: Gradior, un sistema de evaluación y rehabilitación neuropsicológica

Otro destacado ejemplo es el módulo EINK de RehaCom®, diseñado para tratar déficits de memoria de trabajo y planificación mediante la simulación de situaciones diarias, como ir de compras. Estos programas permiten adaptar la intervención terapéutica a las necesidades de cada paciente, ajustando la dificultad y la retroalimentación sobre su progreso.

En resumen, los juegos serios ofrecen una vía innovadora y efectiva para abordar las deficiencias cognitivas, a la vez que mantienen la motivación del paciente durante las sesiones de tratamiento.

#### **2.1.4. Juegos tradicionales en grupo y sus ventajas cognitivas**

A continuación, se nombrarán algunos juegos tradicionales y populares que fomentan el contacto social y evitan el deterioro cognitivo en personas mayores, especialmente trabajando la memoria. Dado que esto es justo lo que se busca lograr con este proyecto, estas actividades servirán como fuente de inspiración para la conceptualización y desarrollo del mismo (El Confidencial, [s.f.](#)).

- **Cada oveja con su pareja:** consiste en agrupar objetos en función de su categoría, utilizando todo tipo de elementos. Desde las cartas de una baraja para agrupar por palos, hasta objetos aleatorios dispuestos, como frutas, verduras... Contribuye a mantener la capacidad intelectual, la memoria y la coordinación visual y manual.
- **Veo-veo:** juego mítico que hace las delicias de los más pequeños, y suele ser cuando juegan con alguien más mayor. Uno de los jugadores tendrá que adivinar el objeto elegido por su inicial; se pueden dar pistas sobre el lugar de la sala en que se encuentra.
- **Palabras encadenadas:** se trata de coger la última sílaba de una palabra y encadenar con la siguiente; que la última sílaba de esa palabra sea la primera de la siguiente. Fomenta la memoria y la comunicación, además de la atención y la concentración.
- **Puzzle de refranes:** los adultos mayores son grandes conocedores de refranes populares. El juego consistirá en escribir los refranes en dos partes, en trozos de papel separados. Los jugadores deberán enlazar cada parte para completar el refrán, que puede ir orientado a una temática y luego formar un mural.
- **Quién es quién:** cada persona es asignada una palabra que todos menos él conocen. Dicha persona tendrá que ir haciendo preguntas de sí o no hasta adivinar quién o qué es.
- **Adivina la canción:** se pueden tomar como referencia las canciones más escuchadas del tiempo en el que las personas mayores eran jóvenes. Se reproducirá la canción durante un corto espacio de tiempo y los participantes anotarán el nombre y artista de la canción. Agudeza auditiva, rapidez y capacidad de atención son algunos de sus beneficios.

También destacan juegos de mesa como el Scrabble (creatividad y habilidades lingüísticas), Dominó (planificación y estrategia), Bingo (atención y memoria a corto plazo), Pictionary (comunicación no verbal), rompecabezas (concentración y paciencia), etc (Teleasistencia Vital, [s.f.](#)).

#### **2.2. Juegos digitales para adultos mayores**

Una vez explicado el concepto de juego serio y algunos ejemplos concretos, se hará hincapié en un tipo concreto del mismo: los juegos digitales, que deben su nombre a la tecnologías de digitalización en las que están basados.

Los juegos digitales representan una herramienta innovadora y prometedora para abordar los desafíos asociados con el envejecimiento activo y saludable expuestos anteriormente. A través de dispositivos como computadoras, tabletas, consolas de juegos y teléfonos inteligentes, los adultos mayores tienen acceso a una amplia variedad de experiencias de juego que pueden contribuir a su bienestar físico, cognitivo y social.

Muchos de los juegos digitales populares entre adultos mayores están basados en juegos tradicionales, pues la familiaridad de las dinámicas es un factor que contribuye a su popularidad. Ejemplos de aplicaciones:

- **Sudoku**: el clásico juego de números ayuda a ejercitar la mente y la concentración.
- **Wordscapes**: búsqueda de palabras que desafía la agilidad mental y el vocabulario.
- **Candy Crush Saga**: niveles de rompecabezas que pueden ser adictivos y desafiantes.
- **Jigsaw Puzzles**: selecciones de rompecabezas virtuales de diferentes niveles de dificultad.
- **Solitaire**: versión digital del clásico juego de cartas, fácil de aprender y entretenido.
- **Peak – Brain Games and Training**: variedad de juegos diseñados para ejercitar diferentes áreas del cerebro.
- **Neurona App**: aplicación que pretende estimular atención, memoria, razonamiento y planificación de adultos mayores mediante diversos juegos.
- **Fit Brain Trainer**: cuenta con 360 juegos de agilidad mental, memoria, capacidad visual y de deducción.
- **NeuroNation**: capaz de personalizar el entrenamiento para cada usuario (Cuidalian, [s.f.](#)).

Estos juegos son populares entre adultos mayores debido a su accesibilidad, puesto que son apps que pueden ser descargadas fácilmente en dispositivos móviles y tablets; capacidad para ejercitar la mente, ya que la mayoría estimulan partes del cerebro concretas (la memoria, atención y resolución de problemas) y ofrecer entretenimiento sin demasiada complejidad al tener mecánicas de juego bastante simples.

La investigación en este campo se centra en comprender cómo los juegos digitales pueden promover el envejecimiento activo al mantener la mente activa, mejorar la destreza manual y fomentar la interacción social. Además, se busca diseñar juegos que sean accesibles y atractivos para esta población, teniendo en cuenta las necesidades y preferencias específicas de los adultos mayores.

A medida que la tecnología avanza y la aceptación de los juegos digitales entre los adultos mayores aumenta, resulta importante explorar cómo estos juegos pueden desarrollarse e integrarse de manera efectiva para mejorar la calidad de vida de esta población.



Figura 5: Mejorar de calidad de vida de adultos mayores con las TIC

### 2.2.1. El proyecto WorthPlay

En este campo de investigación, destaca, entre otros, el proyecto *WorthPlay* (Blat et al., 2012), que se enfoca en la investigación y desarrollo de juegos digitales para personas mayores, con el propósito de promover un envejecimiento activo y saludable. En él se reconoce que las tecnologías de la información y la comunicación (TIC) tienen el potencial de reducir el aislamiento social y mejorar el bienestar físico y psicosocial de los adultos mayores.

Para asegurar la efectividad de los juegos digitales, se destaca la necesidad de entender qué aspectos hacen que un juego valga la pena para esta población. Esto implica considerar elementos educativos, de socialización y de entretenimiento, así como la diversidad de preferencias en cuanto a la jugabilidad y el formato del juego.

El proyecto se basa en el campo de la Interacción Persona-Ordenador (IPO), que busca mejorar la experiencia del usuario con las TIC. Se identifican tres olas en la investigación de IPO, desde la ergonomía hasta la experiencia de usuario (UX). WorthPlay se sitúa en la tercera ola, centrándose en la experiencia de juego y la participación del usuario durante un período prolongado en entornos reales.

La investigación también se enfoca en la inclusión social y la superación de barreras de accesibilidad, como la destreza cognitiva y motriz. Se reconocen los estereotipos negativos sobre los adultos mayores en el contexto de los juegos digitales, y se busca desafiar estos estereotipos a través del diseño participativo de juegos.

El proyecto WorthPlay desarrolla prototipos de juegos y los evalúa con la participación activa de personas mayores. Se espera que este enfoque genere resultados significativos tanto para la academia como para la industria de los juegos digitales, contribuyendo a una mejor comprensión de las necesidades y preferencias de los adultos mayores en este campo.

## **2.3. Estado del arte de los asistentes virtuales**

En esta sección, se hablará sobre el estado actual de los asistentes virtuales, que se caracteriza por su creciente sofisticación impulsada por la IA, su diversidad en términos de funciones y aplicaciones, y la presencia destacada de plataformas líderes como Alexa, que continúan innovando y expandiendo las fronteras de esta tecnología.

### **2.3.1. Influencia de la IA y clasificación**

La inteligencia artificial (IA) es una tecnología que ha evolucionado rápidamente y se ha integrado en la vida cotidiana de las personas, especialmente en la automatización de servicios y la toma de decisiones. Ha encontrado aplicaciones en diversos sectores, como la educación, la salud, la economía y la política, transformando la forma en que se prestan servicios y se interactúa con las organizaciones.

El aumento de la población y el crecimiento exponencial de las ciudades requieren sistemas más rápidos y eficientes para la atención al cliente. A raíz de esta necesidad, se han desarrollado asistentes virtuales inteligentes.

Un asistente virtual es un software con acceso a recursos en línea que emplea técnicas de IA y procesamiento del lenguaje natural para proporcionar soporte en tiempo real a usuarios y otorgar acceso a información relevante sobre los servicios ofrecidos por organizaciones públicas y privadas (Menéndez Román, 2023).

La arquitectura de estos sistemas se ha perfeccionado con el tiempo, desde ELIZA, precursor que emulaba a un psicoterapeuta, hasta los chatbots avanzados y versátiles de hoy en día, compatibles con múltiples dispositivos (altavoces, televisores, teléfonos móviles, tablets, etc).

En cuanto a la clasificación, existe más de un criterio para definir de los tipos de asistentes virtuales, pero los principales son: según el grado de interacción con los usuarios, las funciones y finalidades del servicio, los medios de interacción y el grado de afectividad (Hernández & Cruz, 2023).

| Criterio                | Tipos   |
|-------------------------|---|
| Grado de interacción    | - Dirigidos<br>- Conversacionales   |
| Funciones y finalidades | - Comunicación y marketing<br>- Atención al cliente<br>- Mejora de procesos |
| Medios de interacción   | - Texto<br>- Multimedia<br>- Voz  |
| Grado de afectividad    | - No emocionales<br>- Emocionales   |

Cuadro 2: Clasificación de asistentes virtuales

- **Dirigidos:** realizan preguntas predefinidas a los usuarios mediante elementos fijos, controlando la interacción con el usuario.
- **Conversacionales:** permiten una mayor libertad en las preguntas que el usuario quiere hacer, promoviendo una interacción más natural.
- **Comunicación y marketing:** brindan servicios de consulta dentro de aplicaciones móviles o web.
- **Atención al cliente:** asisten a los usuarios resolviendo sus dudas y consultas a través de conversaciones continuas.
- **Mejora de procesos:** tienen el objetivo de reducir el tiempo dedicado a una área específica.
- **No emocionales:** son los chatbots tradicionales y se limitan a dar respuestas oportunas a las solicitudes del usuario.
- **Emocionales:** diseñados para interactuar y comprender a las personas a través de conversaciones informales, permitiendo una atención personalizada.

Con respecto a las tendencias en el desarrollo e implementación de asistentes virtuales en organizaciones públicas y privadas, se observa que actualmente se utilizan asistentes dirigidos, el tipo más dominante es el de atención al cliente, el medio de interacción más empleado es el texto y suelen ser son no emocionales.

### 2.3.2. Alexa, el asistente conversacional más extendido hoy en día

Junto con Siri (Apple) y Google Assistant, Alexa, desarrollada por Amazon, es sin lugar a dudas una de los asistentes virtuales por voz más extendidas. Numerosos estudios confirman el creciente índice de uso e impacto en el mercado que ha supuesto el lanzamiento de Alexa, entre ellos los resultados obtenidos en la clasificación por puntuaciones del *Voice Platform Impact Rating*:



Figura 6: Ranking de asistentes de voz según el VPIR en 2020 ([statista](#))

Fue lanzado en noviembre de 2014, se encuentra alojado en la nube de Amazon y está disponible en una gran variedad de dispositivos, incluyendo Amazon Echo y Echo Plus, así como también en dispositivos de terceros como smartphones, Raspberry Pi y vehículos.

Actualmente, cuenta con una amplia gama de funciones predefinidas, como manejo de alarmas, notificaciones, calendarios y búsqueda en internet para responder preguntas. Sin embargo, su verdadero potencial radica en las *skills*, similares a aplicaciones de terceros que amplían sus funcionalidades, permitiendo interactuar con distintas compañías y acceder a sus servicios desde un mismo lugar. A nivel mundial, Alexa cuenta con más de 56,000 skills disponibles (García Piñeiro, 2022).

Las *skills* de Alexa para el desarrollo de juegos ofrecen una amplia gama de posibilidades para crear experiencias interactivas y entretenidas. Estas habilidades permiten a los desarrolladores crear juegos de diferentes géneros y niveles de complejidad, desde simples juegos de palabras y

adivinanzas hasta juegos de aventuras o trivia más elaborados.

Por los motivos mencionados anteriormente, se va a elegir a Alexa como la asistente virtual para el juego a desarrollar.

## 2.4. Aplicaciones similares

### 2.4.1. CELIA, mucho más que una asistente

Existen algunas iniciativas dirigidas a las personas mayores que están en situación de aislamiento como el asistente virtual Celia ([web oficial Celia](#)), desarrollado por personal del Centro de Investigación en Tecnologías de Telecomunicación de la Universidade de Vigo, atlanTTic, que ya se ha puesto en marcha con éxito ya que su uso es muy sencillo (El Confidencial, [2024](#)).

Las personas interesadas pueden acceder a este asistente virtual desde su teléfono móvil a través de la aplicación gratuita de CELIA, y también por WhatsApp, enviando un mensaje de texto o una nota de voz de manera que establecen una conversación con “Celia”. De esta sencilla manera la persona mayor que vive sola puede preguntar al asistente virtual al levantarse “¿Qué tiempo va a hacer hoy?” y buscar actividades para acudir como exposiciones, conferencias, conciertos que sean al aire libre o en espacios cubiertos dependiendo de la climatología.

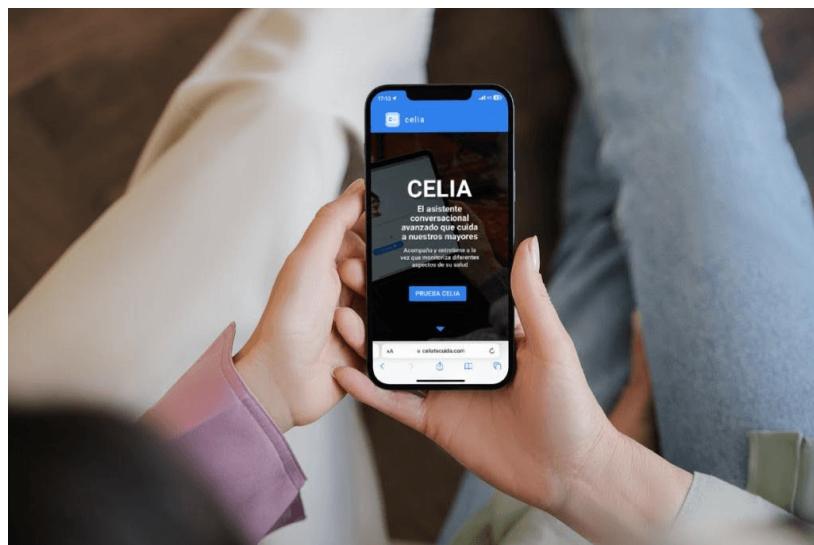


Figura 7: Celia, una asistente todoterreno 24 horas al día.

### 2.4.2. Juego de los trayectos orientado a la detección del deterioro cognitivo

Es una aplicación (o skill) basada en Alexa que puede usarse como una herramienta clínica destinada a la evaluación de la capacidad de memoria de trabajo en pacientes. El juego implementado

consiste en ir memorizando las calles de Jaén de una ruta seleccionada aleatoriamente del mapa cada vez que se inicia una partida (Castillo Moreno, 2022).

Es compatible y accesible a través de cualquier dispositivo de la familia de Alexa y viene con un sistema robusto de almacenamiento en la nube, diseñado para almacenar de manera segura y eficiente toda la información de los usuarios y sus interacciones con el sistema.

Todos estos datos pueden ser consultados mediante una app complementaria destinada exclusivamente al uso del especialista. Así, este último podrá evaluar de manera efectiva y eficiente la capacidad de memoria de trabajo de sus pacientes a partir de la información de sus partidas.

#### 2.4.3. Skill de Alexa para la mejora de la inhibición de respuesta

Consiste en dos juegos principales (animales y colores) con dos modalidades cada uno, empleando la voz y una interfaz gráfica en un dispositivo de Alexa. Estos están dirigidos principalmente a personas con el trastorno por déficit de atención e hiperactivada (TDAH) o variantes similares (Menéndez Román, 2023).

En la primera modalidad del juego de animales, hay 6 rondas y en cada una, se muestra un carrusel de 4 imágenes (2,5 segundos cada una). Se debe pulsar el botón rojo cuando aparezca la imagen que se corresponde al animal que Alexa diga. En el segundo modo de juego, el nombre del animal en cuya aparición hay que presionar el botón viene escrito en pantalla desde el principio, y Alexa dirá el nombre de un animal aleatorio para intentar confundir al jugador.

En cuanto al juego de colores, en la primera variante hay dos pulsadores, un tick y una cruz. Se mostrará por pantalla el nombre de un color, pintado de un color que puede o no ser el mismo. Si coinciden, el jugador debe pulsar el tick y en caso contrario, la cruz. En la segunda versión, se muestran una serie de botones de colores y el jugador debe pulsar la que se corresponda con el color que aparezca escrito.



Figura 8: Interfaz gráfica del juego colores v2

Estos cuatro juegos se han desarrollado tras un trabajo de investigación exhaustivo para determinar qué factores proporcionan más beneficios al grupo de personas al que va dirigido.

### 3. Planificación, metodología y presupuesto

En esta sección, se van a definir los componentes fundamentales en la gestión del proyecto. Estos permitirán definir claramente los objetivos, los recursos disponibles y cómo se utilizarán para alcanzar los resultados deseados dentro de un marco de tiempo y costos preestablecidos.

#### 3.1. Planificación temporal

Se ha elaborado un calendario, marcando con colores los plazos (en días) de las distintas tareas y subtareas en las que se ha desglosado el proyecto.

| Enero | Febrero | Marzo | Abril | Mayo | Junio | Julio | Agosto | Septiembre | Octubre | Noviembre | Diciembre |
|-------|---------|-------|-------|------|-------|-------|--------|------------|---------|-----------|-----------|
| 1 L   | 1 J     | 1 V   | 1 L   | 1 M  | 1 S   | 1 L   | 1 J    | 1 D        | 1 M     | 1 V       | 1 D       |
| 2 M   | 2 V     | 2 S   | 2 M   | 2 J  | 2 D   | 2 M   | 2 V    | 2 L        | 2 M     | 2 S       | 2 L       |
| 3 M   | 3 S     | 3 D   | 3 M   | 3 V  | 3 L   | 3 M   | 3 S    | 3 M        | 3 J     | 3 D       | 3 M       |
| 4 J   | 4 D     | 4 L   | 4 J   | 4 S  | 4 M   | 4 J   | 4 D    | 4 M        | 4 V     | 4 L       | 4 M       |
| 5 V   | 5 L     | 5 M   | 5 V   | 5 D  | 5 M   | 5 V   | 5 L    | 5 J        | 5 S     | 5 M       | 5 J       |
| 6 S   | 6 M     | 6 M   | 6 S   | 6 L  | 6 J   | 6 S   | 6 M    | 6 V        | 6 D     | 6 M       | 6 V       |
| 7 D   | 7 M     | 7 J   | 7 D   | 7 M  | 7 V   | 7 D   | 7 M    | 7 S        | 7 L     | 7 J       | 7 S       |
| 8 L   | 8 J     | 8 V   | 8 L   | 8 M  | 8 S   | 8 L   | 8 J    | 8 D        | 8 M     | 8 V       | 8 D       |
| 9 M   | 9 V     | 9 S   | 9 M   | 9 J  | 9 D   | 9 M   | 9 V    | 9 L        | 9 M     | 9 S       | 9 L       |
| 10 M  | 10 S    | 10 D  | 10 M  | 10 V | 10 L  | 10 M  | 10 S   | 10 M       | 10 J    | 10 D      | 10 M      |
| 11 J  | 11 D    | 11 L  | 11 J  | 11 S | 11 M  | 11 J  | 11 D   | 11 M       | 11 V    | 11 L      | 11 M      |
| 12 V  | 12 L    | 12 M  | 12 V  | 12 D | 12 M  | 12 V  | 12 L   | 12 J       | 12 S    | 12 M      | 12 J      |
| 13 S  | 13 M    | 13 M  | 13 S  | 13 L | 13 J  | 13 S  | 13 M   | 13 V       | 13 D    | 13 M      | 13 V      |
| 14 D  | 14 M    | 14 J  | 14 D  | 14 M | 14 V  | 14 D  | 14 M   | 14 S       | 14 L    | 14 J      | 14 S      |
| 15 L  | 15 J    | 15 V  | 15 L  | 15 M | 15 S  | 15 L  | 15 J   | 15 D       | 15 M    | 15 V      | 15 D      |
| 16 M  | 16 V    | 16 S  | 16 M  | 16 J | 16 D  | 16 M  | 16 V   | 16 L       | 16 M    | 16 S      | 16 L      |
| 17 M  | 17 S    | 17 D  | 17 M  | 17 V | 17 L  | 17 M  | 17 S   | 17 M       | 17 J    | 17 D      | 17 M      |
| 18 J  | 18 D    | 18 L  | 18 J  | 18 S | 18 M  | 18 J  | 18 D   | 18 M       | 18 V    | 18 L      | 18 M      |
| 19 V  | 19 L    | 19 M  | 19 V  | 19 D | 19 M  | 19 V  | 19 L   | 19 J       | 19 S    | 19 M      | 19 J      |
| 20 S  | 20 M    | 20 M  | 20 S  | 20 L | 20 J  | 20 S  | 20 M   | 20 V       | 20 D    | 20 M      | 20 V      |
| 21 D  | 21 M    | 21 J  | 21 D  | 21 M | 21 V  | 21 D  | 21 M   | 21 S       | 21 L    | 21 J      | 21 S      |
| 22 L  | 22 J    | 22 V  | 22 L  | 22 M | 22 S  | 22 L  | 22 J   | 22 D       | 22 M    | 22 V      | 22 D      |
| 23 M  | 23 V    | 23 S  | 23 M  | 23 J | 23 D  | 23 M  | 23 V   | 23 L       | 23 M    | 23 S      | 23 L      |
| 24 M  | 24 S    | 24 D  | 24 M  | 24 V | 24 L  | 24 M  | 24 S   | 24 M       | 24 J    | 24 D      | 24 M      |
| 25 J  | 25 D    | 25 L  | 25 J  | 25 S | 25 M  | 25 J  | 25 D   | 25 M       | 25 V    | 25 L      | 25 M      |
| 26 V  | 26 L    | 26 M  | 26 V  | 26 D | 26 M  | 26 V  | 26 L   | 26 J       | 26 S    | 26 M      | 26 J      |
| 27 S  | 27 M    | 27 L  | 27 S  | 27 L | 27 J  | 27 S  | 27 M   | 27 V       | 27 D    | 27 M      | 27 V      |
| 28 D  | 28 M    | 28 J  | 28 D  | 28 M | 28 V  | 28 D  | 28 M   | 28 S       | 28 L    | 28 J      | 28 S      |
| 29 L  | 29 J    | 29 V  | 29 L  | 29 M | 29 S  | 29 L  | 29 J   | 29 D       | 29 M    | 29 V      | 29 D      |
| 30 M  | 30 S    | 30 M  | 30 J  | 30 D | 30 M  | 30 V  | 30 S   | 30 L       | 30 M    | 30 S      | 30 L      |
| 31 M  | 31 D    | 31 V  | 31 V  | 31 M | 31 S  | 31 S  | 31 J   | 31 J       | 31 M    |           |           |

Figura 9: Planificación temporal en el calendario

## **Calendario: desde el 1 de enero hasta el 1 de septiembre (8 meses = 230 días)**

### **■ 1. Investigación inicial: (15 ene.-11 feb.)**

- 1.1. Motivación, contexto y objetivos (15-25 ene.)
- 1.2. Investigación del estado del arte (26 ene.-11 feb.)

### **■ 2. Análisis del problema y elección de temática: (12 feb.-21 abr.)**

- 2.1. Elección de la temática del juego (12-21 feb.)
- 2.2. Creación de skill y modelo de interacción y oca básica (22 feb.-21 abr.)
- 2.3. Historias de usuario, requisitos del sistema... (1-21 abr.)

### **■ 3. Diseño (22 abr.-5 may.)**

- 3.1. Desarrollo de modelos: arquitectura, conceptual, E/R... (22-28 abr.)
- 3.2. Diseño de la interfaz + cuestiones estéticas y usabilidad (29 abr.-5 may.)

### **■ 4. Análisis tecnológico y desarrollo del código (6 may.-25 ago.)**

- 4.1. Funcionalidad completa del juego ( con minijuegos y APL ) (6 may.-18 ago.)
- 4.2. Configuración de servicios de AWS: IAM, DynamoDB, Amazon y S3 (7 jul.-18 ago.)
- 4.3. Pruebas de ejecución y documentación de código (19-25 ago.)

### **■ 5. Memoria del proyecto (1 feb.-28 ago.)**

### **■ 6. Últimas pruebas y producto final (26 ago.-1 sep.)**

Figura 10: Leyenda de la planificación temporal por (sub)tareas

La planificación establece el marco temporal y los objetivos del proyecto dentro de las fechas límite para su entrega.

| <b>ID</b> | <b>Tarea</b>   | <b>F. inicio</b> | <b>F. final</b> | <b>Duración (d.)</b> |
|-----------|--|------------------|-----------------|----------------------|
| <b>1.</b> | <b>Investigación inicial</b>                                 | <b>15 ene.</b>   | <b>11 feb.</b>  | <b>28</b>            |
| 1.1.      | Motivación, contexto y objetivos                             | 15 ene.          | 25 ene.         | 11                   |
| 1.2.      | Investigación del estado del arte                            | 26 ene.          | 11 feb.         | 17                   |
| <b>2.</b> | <b>Análisis del problema y elección de temática</b>          | <b>12 feb.</b>   | <b>21 abr.</b>  | <b>70</b>            |
| 2.1.      | Elección de la temática del juego                            | 12 feb.          | 21 feb.         | 10                   |
| 2.2.      | Creación de skill, modelo de interacción y oca básica        | 22 feb.          | 21 abr.         | 59                   |
| 2.3.      | Historias de usuario, requisitos del sistema...              | 1 abr.           | 21 abr.         | 21                   |
| <b>3.</b> | <b>Diseño</b>  | <b>22 abr.</b>   | <b>5 may.</b>   | <b>14</b>            |
| 3.1.      | Desarrollo de modelos: arquitectura, conceptual, E/R...      | 22 abr.          | 28 abr.         | 7                    |
| 3.2.      | Diseño de la interfaz, cuestiones estéticas y usabilidad     | 29 abr.          | 5 may.          | 7                    |
| <b>4.</b> | <b>Análisis tecnológico y desarrollo de código</b>           | <b>6 may.</b>    | <b>25 ago.</b>  | <b>112</b>           |
| 4.1.      | Funcionalidad completa del juego (con mini-juegos y APL)     | 6 may.           | 18 ago.         | 106                  |
| 4.2.      | Configuración de servicios de AWS: Amazon S3, IAM y DynamoDB | 7 jul.           | 18 ago.         | 43                   |
| 4.3.      | Pruebas de ejecución y documentación del código              | 19 ago.          | 25 ago.         | 7                    |
| <b>5.</b> | <b>Memoria del proyecto</b>                                  | <b>1 feb.</b>    | <b>28 ago.</b>  | <b>210</b>           |
| <b>6.</b> | <b>Últimas pruebas y producto final</b>                      | <b>26 ago.</b>   | <b>1 sep.</b>   | <b>7</b>             |
|           | <b>Conjunto total del proyecto</b>                           | <b>15 ene.</b>   | <b>1 sep.</b>   | <b>230</b>           |

### **3.2. Metodología**

La metodología describe las técnicas y procesos que se seguirán para desarrollar el proyecto, incluyendo las herramientas y tecnologías utilizadas, los roles y responsabilidades del equipo y los métodos de comunicación y gestión del proyecto. En este caso, se va a emplear la metodología de *desarrollo ágil*.

El método de desarrollo ágil utiliza un enfoque iterativo y flexible, facilitando la adaptación a cambios y la entrega continua de valor al usuario. Esta metodología es especialmente efectiva en entornos donde las necesidades del usuario y el mercado pueden cambiar rápidamente, y donde la colaboración y la comunicación efectiva entre los miembros del equipo y con el cliente son fundamentales para el éxito del proyecto (Duarte & Rojas, 2008).

El desarrollo ágil para una aplicación se divide en varias etapas que se alinean con las prácticas y principios de dicha metodología. Estas fases incluyen:

1. **Análisis del problema:** se centra en comprender las necesidades del usuario y los requisitos del sistema. En el contexto de una aplicación, implica la creación de historias de usuario y casos de uso, que son esenciales para definir las funcionalidades que la aplicación debe ofrecer.
2. **Diseño:** se planifican las soluciones técnicas y se definen los detalles de diseño, como la arquitectura de la aplicación, el modelo E/R y el diseño de la interfaz de usuario. El diseño conceptual y los bocetos y mockups de esta fase preparan el camino para la implementación.
3. **Desarrollo:** se trabaja para implementar las soluciones diseñadas. Esto incluye la codificación, integración de componentes y configuración del entorno de desarrollo.
4. **Pruebas:** su realización permite corregir errores a tiempo. Esto asegura que la aplicación funcione como se espera y cumpla con los requisitos definidos en las etapas anteriores.
5. **Despliegue:** una vez que la aplicación ha sido probada y se ha asegurado de que cumple con los requisitos y expectativas, se despliega para su uso.
6. **Revisión y mejora continua:** tras el despliegue, se puede incluir la recopilación de comentarios de los usuarios, la identificación de áreas de mejora en el proceso de desarrollo y las implementaciones de cambios para mejorarla en iteraciones futuras.

### **3.3. Presupuesto**

El presupuesto es una herramienta clave para gestionar los recursos del proyecto, desde el asignamiento de recursos humanos hasta la adquisición de equipamiento y materiales necesarios.

Permite estimar los costos asociados a cada entregable y recurso requerido, estableciendo un cronograma de gastos y asignando responsabilidades para el control de los mismos.

### **3.3.1. Recursos humanos**

Incluye las personas involucradas en el proyecto, tanto en las etapas previas al desarrollo como durante el mismo.

| Descripción    | Coste (€) |
|----------------|-----------|
| Programador(a) | X         |
| Psicólogo/a    | X         |

Cuadro 3: Presupuesto para recursos humanos

### **3.3.2. Hardware**

Todos los dispositivos físicos y electrónicos necesarios para llevar a cabo la aplicación.

| Descripción                                | Coste (€) |
|--|-----------|
| HP 15s Intel Core i5-1035G1/16GB/1TB/15.6" | 550-650   |
| Alexa Echo Show                            | 70        |

Cuadro 4: Presupuesto para hardware

### **3.3.3. Software**

Los programas

| Descripción               | Coste (€) |
|---------------------------|-----------|
| Visual Paradigm           | 0         |
| Amazon Web Services (AWS) | X/?       |
| Dynamo DB                 | X/?       |
| Amazon S3                 | X/?       |
| TeXstudio                 | 0         |
| Visual Studio Code        | 0         |
| Alexa Developer Console   | 0         |

Cuadro 5: Presupuesto para software

### **3.3.4. Resumen de presupuesto**

tfnjtrjrf

| Descripción      | Coste (€) |
|------------------|-----------|
| Recursos humanos | X         |
| Hardware         | X         |
| Software         | X         |

Cuadro 6: Tabla general de presupuestos

## 4. Análisis del problema

### 4.1. Elección de la temática: el juego de la oca

Antes de pasar a la etapa de diseño y desarrollo, se debe elegir el tipo de juego a implementar, teniendo en cuenta factores de viabilidad tanto del ámbito tecnológico como del psicológico (¿es apropiado para el grupo demográfico al que va dirigido?).

Como se ha visto en la sección 2.1.3., los juegos tradicionales pueden servir como fuente de inspiración para lo que se busca en este proyecto. Sin embargo, existe una gran cantidad de juegos de mesa populares entre los adultos mayores, como es el caso del parchís, el scrabble, el memorama, el dominó, etc. Entonces, ¿por qué decantarse por el juego de la oca?

El principal motivo es la flexibilidad que permite a la hora de diseñar un juego que cumpla unos objetivos específicos. Un ejemplo de ello es la iniciativa de un Centro de Educación Primaria en Murcia, que implementó un programa innovador para la evaluación de las habilidades motrices del estudiantado. (Ortín et al., 2008)

El proceso fue el siguiente: se diseñaron tres tableros de la oca, que se correspondían con los tres ciclos escolares que participaron en el experimento. Cada uno, además de las casillas especiales del juego original (oca, puente, calavera...), disponía de una serie de actividades físicas para poner a prueba a los participantes, entre las que se encuentran: el lanzamiento y captura de objetos, ejercicios de malabares y equilibrio y cooperación con los compañeros.

| ANEXO 1. ACTIVIDADES TABLERO 1: OCA DE MANEJO DE MÓVILES (1º CICLO) |  |
|---|--|
| CASILLA 1   | <b>EL CAMARERO:</b> colocando la pelota encima del vaso boca abajo, impulsar la pelota hacia arriba a la vez que gira el vaso para colar la pelota cuando caiga. Se permite que falle uno del grupo.                                   |
| CASILLA 2   | <b>TRANSPORTE DE SAQUITOS EN EL MENOR TIEMPO POSIBLE:</b> (Ver sesión nº 3 sub-programa). El maestro determinará un tiempo para realizar la prueba.  |
| CASILLA 3   | <b>EL CARRUSEL DE PELOTAS:</b> cada miembro del grupo con un balón de plástico y formando un círculo, a la señal del maestro lanzar la pelota hacia arriba y recibir la del compañero, antes de que caiga al suelo durante 1 minuto.   |
| CASILLA 4   | <b>PELOTA ARRIBA:</b> todos los miembros del grupo deben mantener la pelota en el aire durante 15 segundos. Cada miembro del grupo podrá dar 2 toques como máximo hasta que todos le hayan dado al menos una vez.                      |
| CASILLA 5   | <b>DE OCA A OCA:</b> Pasáis a la casilla nº 9 y... Tiro porque me toca. Vuelves a lanzar el dado.  |
| CASILLA 6   | <b>DE PUENTE A PUENTE:</b> Pasáis a la casilla nº 12 y ... Tiro porque me lleva la corriente. Vuelves a lanzar el dado   |
| CASILLA 7   | Cada miembro del grupo debe realizar 5 botes seguidos con una mano y sin parar otros 5 con la otra mano.   |
| CASILLA 8   | <b>PASES ORDENADOS DE LA SEMANA</b> (Sesión nº 7).   |
| CASILLA 9   | <b>DE OCA A OCA:</b> Pasáis a la casilla nº 15 y... Tiro porque me toca. Vuelves a lanzar el dado  |
| CASILLA 10  | <b>EL TREN BOTADOR:</b> situados en fila y cogidos por la cintura se desplazan botando con la mano libre conducidos por el maquinista que los llevará de un lado a otro de la pista. El maquinista marca el ritmo y el resto lo imita. |

Figura 11: Diseño de la oca para evaluar las habilidades motoras en Educación Primaria (Ortín et al., 2008)



Figura 12: Tablero completo correspondiente a las casillas de la figura superior (Ortín et al., 2008)

Esta medida tuvo gran éxito, pues los seguimientos del progreso de los participantes mostraron una mejoría general en sus destrezas físicas y capacidad de trabajo en equipo, así como una participación activa por parte de los estudiantes de Primaria que no se había registrado hasta el momento.

Por tanto, se ha querido replicar de alguna manera ese enfoque, trasladándolo a un contexto donde el grupo objetivo son los adultos mayores, y teniendo en cuenta la diversidad que puede existir dentro del mismo.

Partiendo de la estructura del tablero original, hay una mayor probabilidad de que los adultos mayores estén familiarizados con su diseño y las reglas básicas, lo que puede contribuir a una experiencia de juego positiva y fluida.



Figura 13: Un tablero del juego tradicional de la oca

Sin embargo, al igual que la oca adaptada para estudiantes de Educación Primaria vista anteriormente, no se tratará de una oca tradicional cualquiera.

Pues aparte de las casillas originales y el objetivo de llegar a la meta, incluirá un sistema de puntos, añadiendo un subobjetivo: el de conseguir la mayor puntuación antes de que termine la partida. Los participantes del juego podrán incrementar su marcador a través de una serie de minijuegos, que serán desencadenados cuando los jugadores caigan en determinadas casillas especiales.

También debe evaluarse la viabilidad de los minijuegos propuestos dentro del juego principal, que es la oca, para lo que se ha consultado con un equipo de psicólogos que han participado en varias experiencias de residencias.

#### 4.1.1. Lista de minijuegos dentro de la oca

Dada la dificultad de implementar algunos en base a las limitaciones de los dispositivos Alexa, y para evitar saturar a los participantes con demasiadas mecánicas nuevas, se han elegido los siguientes cinco minijuegos:

**1. Preguntas de Trivial V/F:** pondrán a prueba los conocimientos generales de los participantes. Se abordarán categorías distintas (geografía, historia, ciencia, arte y cultura. . . ) pero siempre habrá dos opciones de respuesta, *verdadero* o *falso*. Si se acierta, se obtienen puntos, y si no, Alexa proporcionará una breve explicación de la respuesta correcta.

- Alexa: *¿Oceanía es el continente más pequeño de todos?*
  - Agustín: *Verdadero.*
  - Alexa: *¡Correcto! Has ganado 10 puntos.*
- Alternativamente...
- Agustín: *Falso.*
  - Alexa: *Incorrecto, el continente más pequeño del mundo es Oceanía, con 9.000.000 km<sup>2</sup> de superficie.*

**2. Conoce a tus compañeros:** Alexa planteará una pregunta acerca de otro participante, quien deberá colaborar, ya que luego tendrá que confirmar si la respuesta proporcionada por el primer participante es correcta o no. Siempre habrá dos tipos de respuestas, *correcto/a* o *incorrecto/a*.

- Alexa: *Agustín, ¿Roberta ha viajado fuera del país alguna vez? Dime si es correcto o incorrecto.*
- Agustín: *Correcto*
- Alexa: *Roberta, ¿es esta respuesta correcta o incorrecta?*
- Roberta: *Correcta*
- Alexa: *¡Bien! Por conocerlos tan bien, ambos ganáis puntos.*

**3. Adivina la cifra:** Alexa hará preguntas de todo tipo, cuya única respuesta es un número. Puede ser un año, la cantidad de algo... Es de modalidad competitiva, por lo que si el primero en responder la falla, la pregunta rebotará al siguiente, y así sucesivamente hasta que alguien acierte o se llegue de nuevo al primer participante, en cuyo caso Alexa desvelará la solución.

- Alexa: *Agustín, ¿en qué año se descubrió América?*
- Agustín: *1520*
- Alexa: *Incorrecto, la pregunta rebota a Roberta. ¿En qué año se descubrió América?*
- Roberta: *1492*
- Alexa: *¡Correcto! Roberta gana 30 puntos.*

4. **Recuerda la última casilla:** Alexa preguntará por la casilla más reciente previa a la última tirada de dado; es decir, la casilla donde cayó en el turno anterior.

- Alexa: *¿Cuál fue la última casilla en la que caíste en el turno anterior?*
  - Roberta: *La casilla del sombrero*
  - Alexa: *¡Correcto! Has ganado 'x' puntos.*
- Alternativamente...
- Roberta: *Incorrecto, la respuesta correcta era: la casilla del paraguas.*

5. **Recuerda la fecha:** Alexa hará preguntas para poner a prueba la memoria de los participantes en relación con la orientación temporal, es decir, la forma en la que alguien mantiene la noción del tiempo. Las respuestas siempre serán días de la semana, meses o estaciones del año.

- Alexa: *¿Qué día de la semana fue ayer?*
- Agustín: *Jueves ...*
- Alexa: *¿En qué mes estamos?*
- Roberta: *Julio*

## 4.2. Historias de usuario y requisitos

Las historias de usuario son imprescindibles en el desarrollo ágil de software, pero también resultan valiosas en cualquier otra metodología de desarrollo, ya sea tradicional o no. Pues a través de un formato sencillo, recogen de forma clara las necesidades y expectativas de los usuarios y clientes, facilitando la comunicación entre integrantes del equipo de desarrollo y clientes. Un ambiente colaborativo donde las ideas pueden fluctuar a medida que se avanza aumenta las probabilidades de éxito de cualquier proyecto (Menzinsky et al., 2018).

El formato típico de una historia de usuario se basa en los tres elementos siguientes:

**Como [rol del usuario], quiero [objetivo], para poder [beneficio].**

Figura 14: Patrón general de una historia de usuario

Esta estructura facilita empatizar con la persona usuaria, entender qué pretende lograr sin entrar en detalles del cómo o el valor que aporta al producto, conocido como beneficio. Así, además de sintetizar en gran medida la información, permite cierta flexibilidad y la adaptación a cambios e integración de nuevas ideas durante el proceso de desarrollo.

Las historias de usuario ofrecen una perspectiva más centrada en los usuarios finales, mientras que los requisitos funcionales se limitan a describir el comportamiento del sistema. Entonces, merece la pena considerar ambos para que el producto final no solo cumpla con las especificaciones técnicas, sino que también otorgue valor real a quienes va dirigido.

#### 4.2.1. Historias de usuario

Las siguientes historias de usuario establecen los requisitos de sistema del juego de la oca controlada por Alexa.

Estas contemplan desde la creación y configuración de una partida, hasta la gestión de turnos y control del estado del juego, garantizando asistencia activa durante todo el juego.

| HU01: Iniciar el juego |                                  |
|------------------------|----------------------------------|
| <b>Como</b>            | jugador/a                        |
| <b>Quiero</b>          | poder iniciar el juego de la oca |
| <b>Para</b>            | empezar a jugar                  |

Cuadro 7: Historia de usuario nº 1

| HU02: Creación de una nueva partida |   |
|-------------------------------------|---|
| <b>Como</b>                         | jugador/a   |
| <b>Quiero</b>                       | poder iniciar un juego de la oca nuevo, pudiendo elegir cuántos jugadores van a participar, y si va ser por equipos o individualmente |
| <b>Para</b>                         | adaptar el juego a la cantidad y el tipo de participantes   |

Cuadro 8: Historia de usuario nº 2

| HU03: Escuchar las reglas |   |
|---------------------------|---|
| <b>Como</b>               | jugador/a   |
| <b>Quiero</b>             | pedirle a Alexa que me explique las reglas y cualquier otro aspecto relevante del juego |
| <b>Para</b>               | entender cómo jugar antes de comenzar o recordar alguna explicación durante la partida  |

Cuadro 9: Historia de usuario nº 3

| HU04: Jugar un turno |  |
|----------------------|--|
| <b>Como</b>          | jugador/a  |
| <b>Quiero</b>        | que Alexa tire los dados por mí y también mueva mi ficha |
| <b>Para</b>          | jugar mi turno   |

Cuadro 10: Historia de usuario nº 4

| HU05: Guardar la partida |  |
|--------------------------|--|
| <b>Como</b>              | jugador/a  |
| <b>Quiero</b>            | que Alexa tenga la capacidad de guardar el progreso de la partida actual |
| <b>Para</b>              | poder retomar el juego más tarde sin perder el avance                    |

Cuadro 11: Historia de usuario nº 5

| HU06: Continuar la partida anterior |  |
|-------------------------------------|--|
| <b>Como</b>                         | jugador/a  |
| <b>Quiero</b>                       | poder continuar con la partida anterior si abro una nueva sesión |
| <b>Para</b>                         | reanudar la última partida donde la dejé                         |

Cuadro 12: Historia de usuario nº 6

| HU07: Recibir ayuda activa |   |
|----------------------------|---|
| <b>Como</b>                | jugador/a   |
| <b>Quiero</b>              | que Alexa me ofrezca ayuda activa durante el juego    |
| <b>Para</b>                | saber qué hacer por si me pierdo en cualquier momento |

Cuadro 13: Historia de usuario nº 7

| HU08: Finalizar el juego |   |
|--------------------------|---|
| <b>Como</b>              | jugador/a                                     |
| <b>Quiero</b>            | poder finalizar el juego en cualquier momento |
| <b>Para</b>              | terminar la partida cuando lo deseé           |

Cuadro 14: Historia de usuario nº 8

En el desarrollo del juego interactivo para Alexa, se han diseñado una serie de minijuegos que tienen como objetivo poner a prueba los conocimientos y la memoria de los participantes. Estos son los listados en la sección 4.1.1.

A continuación, se presentan las historias de usuario para cada uno de estos minijuegos, que abarcan desde trivial de conocimiento general hasta desafíos de memoria y observación.

| HU09: Minijuego verdadero o falso |  |
|-----------------------------------|--|
| <b>Como</b>                       | jugador/a que ha caído en la casilla del minijuego de verdadero o falso                        |
| <b>Quiero</b>                     | responder a la pregunta planteada por Alexa con verdadero o falso y que me diga si he acertado |
| <b>Para</b>                       | poner a prueba mis conocimientos y ganar puntos  |

Cuadro 15: Historia de usuario nº 9

| HU10: Minijuego conoce a tus compañeros |   |
|---|---|
| <b>Como</b>                             | jugador/a que ha caído en la casilla del minijuego de conoce a tus compañeros                             |
| <b>Quiero</b>                           | responder a la pregunta acerca de otro jugador planteada por Alexa y que me confirmen si es correcto o no |
| <b>Para</b>                             | para demostrar cuánto conozco a mis compañeros y ganar puntos   |

Cuadro 16: Historia de usuario nº 10

| HU11: Minijuego adivina la cifra |  |
|----------------------------------|--|
| <b>Como</b>                      | jugador/a en una ronda del minijuego de adivina la cifra                                     |
| <b>Quiero</b>                    | responder a la preguntas planteada por Alexa con un número y que me diga si he acertado o no |
| <b>Para</b>                      | poner a prueba mis conocimientos y ganar puntos  |

Cuadro 17: Historia de usuario nº 11

| <b>HU12:</b> Minijuego recuerda la última casilla |  |
|---|--|
| <b>Como</b>                                       | jugador/a que ha caído en la casilla del minijuego de recuerda la última casilla |
| <b>Quiero</b>                                     | que responder con el nombre de la casilla en la que caí en el turno anterior     |
| <b>Para</b>                                       | poner a prueba mi memoria y ganar puntos   |

Cuadro 18: Historia de usuario nº 12

| <b>HU13:</b> Minijuego recuerda la fecha |  |
|--|--|
| <b>Como</b>                              | jugador/a que ha caído en la casilla del minijuego de recuerda la fecha                                  |
| <b>Quiero</b>                            | responder a la pregunta planteada por Alexa con el nombre de un día de la semana, mes o estación del año |
| <b>Para</b>                              | poner a prueba mi noción del tiempo y ganar puntos   |

Cuadro 19: Historia de usuario nº 13

#### **4.2.2. Requisitos funcionales**

Esta sección define y describe las características de alto nivel (requisitos funcionales) del sistema que son necesarias para cubrir las necesidades de los usuarios. Se pueden estructurar de la siguiente manera:

##### **RF1: Iniciar la skill de Alexa**

Se debe poder lanzar la skill mediante el comando de invocación.

##### **RF2: Registro de datos de los participantes**

- **RF2.1:** La skill debe preguntar el modo de juego: por equipos o jugadores individuales.
- **RF2.2:** La skill debe preguntar el número de jugadores/equipos que van a participar.
- **RF2.3:** La skill debe registrar el nombre de los jugadores o equipos antes de empezar la partida.

##### **RF3: Simulación de un turno**

- **RF3.1:** La skill debe incluir una función que simule el lanzamiento de dados y determine el número de casillas a avanzar.
- **RF3.2:** La skill debe incluir una función que mueva la ficha del jugador y muestre la casilla en la que ha caído.

##### **RF4: Gestión del estado del juego**

- **RF4.1:** La skill debe ser capaz de guardar el estado actual del juego.
- **RF4.2:** Relacionado con el anterior, se debe poder reanudar la última partida con el estado con el que se ha guardado.
- **RF4.3:** Poder finalizar la partida en cualquier momento (borrando los datos del juego actual a no ser que se haya guardado previamente).
- **RF4.4:** Poder iniciar una nueva partida en cualquier momento (sobrescribiendo los datos guardados de la anterior).

##### **RF5: Explicación del juego**

- **RF5.1:** Debe existir una opción dentro de la skill para explicar las reglas y objetivos del juego mediante un comando de voz.

- **RF5.2:** Debe existir una opción dentro de la skill para explicar los tipos de casillas del tablero mediante un comando de voz.
- **RF5.3:** Debe existir una opción dentro de la skill para explicar detalladamente los tipos de minijuegos mediante un comando de voz.
- **RF5.4:** Debe existir una opción dentro de la skill para que Alexa nombre y explique brevemente todos los comandos disponibles.

## **RF6: Interacción continua**

- **RF6.1:** La skill debe ser capaz de mantener una interacción continua con el usuario, a la espera de una respuesta en todo momento.
- **RF6.2:** La skill debe ofrecer asistencia en todo momento, en caso de que los participantes no sepan qué hacer a continuación y pase cierto tiempo sin recibir una respuesta, o que esta última no sea válida.

## **RF7: Participar en un minijuego**

- **RF7.1:** Cuando se cae en una casilla de minijuego, Alexa debe sacar un elemento aleatorio de la batería de preguntas correspondiente a dicho minijuego.
- **RF7.2:** La skill debe poder capturar la respuesta del participante y verificar si es correcta o no, actualizando sus puntos si es necesario.

### **4.2.3. Requisitos no funcionales**

En este apartado se pueden ver las diferentes cualidades y restricciones del sistema (requisitos no funcionales) que no se relacionan de forma directa con el comportamiento del mismo.

#### **RNF1: Usabilidad**

- **RNF1.1:** La skill debe ser fácil de usar y entender, especialmente diseñada para personas mayores.
- **RNF1.2:** Para evitar dudas, que Alexa explique de forma clara y fácil de entender lo que deben hacer los jugadores.
- **RNF1.3:** Alexa debe dejar un margen flexible de tiempo para esperar una respuesta y si no lo hace, repite la pregunta.
- **RNF1.4:** La interfaz debe ser simple y limitarse a mostrar los elementos relevantes de la partida para evitar saturar a los jugadores y jugadoras.

- **RNF1.5:** La fuente y disposición de elementos debe estar adaptada para la compresión y comodidad de los y las participantes.

### **RNF2: Accesibilidad y compatibilidad**

- **RNF2.1:** Es necesaria una cuenta de Amazon para instalar la skill y poder interactuar con Alexa.
- **RNF2.2:** Es necesario un dispositivo de Alexa con pantalla rectangular (como un Echo Show).
- **RNF2.3:** El dispositivo de Alexa debe tener una conexión estable a Internet.
- **RNF2.4:** La skill solo es compatible con el idioma español.

### **RNF3: Rendimiento**

- **RNF3.1:** Las respuestas a los comandos de voz deben ser rápidas, idealmente no superando los 5 segundos.

### **RNF4: Fiabilidad**

- **RNF4.1:** La skill debe funcionar correctamente en la mayoría de las interacciones, minimizando errores y malentendidos en el reconocimiento de voz.

### **RNF5: Seguridad**

- **RNF5.1:** Seguir las prácticas recomendadas basadas en principio de mínimo privilegio a la hora de utilizar los servicios de AWS.

## 4.3. Casos de uso y sus correspondientes diagramas

Una vez se han detallado las historias y usuario y los requisitos del sistema, el siguiente paso es elaborar los casos de uso, asociados a los requisitos definidos previamente e incluso otros casos de uso. Esta técnica de ingeniería de software permite establecer la forma de interactuar entre los actores, entendidos como las entidades o sistema implicados en cada funcionalidad.

### 4.3.1. Casos de uso

Como los actores siempre van a ser las personas usuarias y la asistente conversacional Alexa, se ha omitido esta fila en la plantilla de casos de uso, manteniendo únicamente en ella los aspectos fundamentales para comprender el curso normal de eventos y excepciones.

| CU01: Iniciar la skill          |  |
|---------------------------------|--|
| <b>Descripción</b>              | La persona usuaria interactúa con Alexa para abrir la skill  |
| <b>Precondiciones</b>           | La skill está desplegada en la cuenta de Amazon vinculada al dispositivo Echo Show, que además debe tener acceso a Internet.   |
| <b>Postcondiciones</b>          | El dispositivo muestra la pantalla de inicio   |
| <b>Referencias</b>              | RF1  |
| <b>Flujo normal de eventos</b>  | <ol style="list-style-type: none"><li>1. La persona usuaria dice «Alexa, abre probando oca» para abrir la skill.</li><li>2. Alexa realiza el lanzamiento de la skill, dando la bienvenida y una breve introducción al juego.</li></ol> |
| <b>Flujo alterno de eventos</b> | <b>1.a.</b> Si la persona usuaria intenta jugar antes de haber lanzado la skill, Alexa dirá que no reconoce ninguno de los comandos.   |

Cuadro 20: Caso de uso nº 1

| CU02: Pedir ayuda               |   |
|---------------------------------|---|
| <b>Descripción</b>              | La persona usuaria interactúa con Alexa para pedirle ayuda  |
| <b>Precondiciones</b>           | La skill está iniciada.   |
| <b>Postcondiciones</b>          | Alexa ofrece un listado de temas que puede explicar.  |
| <b>Referencias</b>              | RF5.1, RF5.2, RF5.3 y RF5.4   |
| <b>Flujo normal de eventos</b>  | <p>1. La persona usuaria dice «Ayuda» para que Alexa pueda asistirle.</p> <p>2. Alexa ofrece una lista de temas que puede explicar: las reglas y objetivos del juego, los tipos de casillas, los minijuegos y todos los comandos disponibles. Alexa dice que para aclarar uno de los temas, se debe decir «Explícame &lt;tema&gt;».</p> <p>3. La persona usuaria dice «Explícame &lt;tema&gt;», siendo &lt;tema&gt; uno de los de la lista ofrecida por Alexa.</p> <p>4. Alexa procede a explicar de forma clara el &lt;tema&gt; preguntado por la persona usuaria.</p> |
| <b>Flujo alterno de eventos</b> | <p><b>3.a.</b> Si la persona usuaria intenta preguntar por un tema que no viene en la lista ofrecida por Alexa, repetirá la lista de temas que puede explicar.</p>  |

Cuadro 21: Caso de uso nº 2

| CU03: Crear una nueva partida   |   |
|---------------------------------|---|
| <b>Descripción</b>              | La persona usuaria interactúa con Alexa para crear una nueva partida del juego de la oca.   |
| <b>Precondiciones</b>           | La skill está iniciada.   |
| <b>Postcondiciones</b>          | Alexa crea una nueva partida con las configuraciones proporcionadas por la persona usuaria.   |
| <b>Referencias</b>              | RF2.1, RF2.2, RF4.4, RF6.1 y RF6.2  |
| <b>Flujo normal de eventos</b>  | <ol style="list-style-type: none"> <li>1. La persona usuaria le dice «Nueva partida» a Alexa.</li> <li>2. Alexa pregunta si los participantes van a jugar por equipos o individualmente.</li> <li>3. La persona usuaria responde: «por equipos» o «por jugadores individuales».</li> <li>4. Alexa pide confirmación, y después pregunta por el número de equipos/jugadores que van a participar en el juego.</li> <li>5. La persona usuaria responde con el número de participantes o equipos.</li> <li>6. Alexa confirma la respuesta, y después pide proceder con el registro de nombre de los equipos/participantes, dando instrucciones de cómo hacerlo.</li> </ol> |
| <b>Flujo alterno de eventos</b> | <p><b>3.a.</b> Si la persona usuaria dice un comando distinto a las dos posibles opciones, Alexa vuelve a preguntarle por el tipo de participantes.</p> <p><b>5.a.</b> Si la persona usuaria responde con algo que no es un número, o es una cifra no válida (menor que 1 y mayor que 5), Alexa repite la pregunta del número de participantes.</p> <p><b>4.a. y 6.a.</b> Si la persona usuaria no le confirma su respuesta a Alexa, le volverá a formular la pregunta.</p>   |

Cuadro 22: Caso de uso nº 3

| CU04: Registro de participantes |   |
|---------------------------------|---|
| <b>Descripción</b>              | Todos los equipos/participantes interactúan con Alexa para registrar sus nombres en el juego.   |
| <b>Precondiciones</b>           | Se está creando una nueva partida y ya se han configurado los siguientes datos: tipo y número de participantes.   |
| <b>Postcondiciones</b>          | Alexa anuncia el nombre de cada participante o equipo, muestra sus colores y nombres por pantalla, y anuncia el comienzo de la partida.   |
| <b>Referencias</b>              | RF2.3, RF6.1 y RF6.2   CU03   |
| <b>Flujo normal de eventos</b>  | <ol style="list-style-type: none"> <li>1. El primer equipo/participante dice «Nuestro/Mi nombre es &lt;nombre&gt;» para que Alexa pueda registrarlo.</li> <li>2. Alexa confirma el nombre registrado y pide el del siguiente equipo/participante.</li> <li>3. El siguiente equipo/participante registra su nombre de la misma forma que el primero.</li> <li>4. Alexa sigue pidiendo nombres hasta que se hayan registrado todos, en cuyo caso hace un resumen de los equipos/participantes registrados y los muestra por pantalla. Después hace una copia de los datos de la partida en la base de datos.</li> </ol> |
| <b>Flujo alterno de eventos</b> | <p><b>1.a. y 3.a.</b> Si el equipo/participante no anuncia su nombre de la forma correcta establecida por Alexa, o Alexa no recibe respuesta en un rato, le recuerda que registre su nombre y cómo hacerlo.</p>   |

Cuadro 23: Caso de uso nº 4

| CU05: Jugar un turno            |  |
|---------------------------------|--|
| <b>Descripción</b>              | El equipo/participante actual juega su turno, compuesto por dos partes: tirar el dado y mover su ficha.  |
| <b>Precondiciones</b>           | La partida ya está configurada y es el comienzo del turno del equipo/participante actual, quien no debe estar atrapado en alguna casilla de penalización.  |
| <b>Postcondiciones</b>          | Alexa muestra por pantalla el color y nombre del equipo/participante actual y la casilla en la que ha caído, explicando el evento que esta ha desencadenado.   |
| <b>Referencias</b>              | RF3.1, RF3.2, RF6.1 y RF6.2  |
| <b>Flujo normal de eventos</b>  | <ol style="list-style-type: none"> <li>1. El equipo/participante actual dice «Tirar dado».</li> <li>2. Alexa muestra la tira de dado por pantalla y anuncia el número del 1 al 6 que ha salido. Después le indica que mueva su ficha.</li> <li>3. El equipo/participante actual dice «Mover ficha».</li> <li>4. Alexa informa sobre su movimiento, mostrando por pantalla el nombre y color del equipo/participante actual, así como en la casilla que ha caído. Alexa la describe y da las instrucciones adicionales necesarias en caso de ser una casilla especial.</li> </ol> |
| <b>Flujo alterno de eventos</b> | <b>1.a. y 3.a.</b> Si el equipo/participante actual dice un comando no accesible en ese momento o distinto al especificado, Alexa le repetirá lo que debe decir para avanzar en el juego.  |

Cuadro 24: Caso de uso nº 5

| CU06: Guardar la partida        |   |
|---------------------------------|---|
| <b>Descripción</b>              | La persona usuaria guarda el estado de la partida actual para que persista de una sesión a otra.  |
| <b>Precondiciones</b>           | La partida actual ya está configurada, y solo puede guardarse al comienzo de turno de cualquier jugador, es decir, antes de tirar el dado.  |
| <b>Postcondiciones</b>          | Alexa anuncia que se han guardado los datos correctamente, al haber sido enviados con éxito a la base de datos.   |
| <b>Referencias</b>              | RF4.1   |
| <b>Flujo normal de eventos</b>  | <p><b>1.</b> La persona usuaria dice «Guardar partida».</p> <p><b>2.</b> Alexa envía los datos de la partida y de los jugadores a la base de datos de DynamoDB y anuncia que la operación tuvo éxito.</p> |
| <b>Flujo alterno de eventos</b> | <b>1.a.</b> No dejará guardar la partida si se encuentra en mitad de un turno (moviendo la ficha o participando en un minijuego).   |

Cuadro 25: Caso de uso nº 6

| CU07: Continuar la partida      |  |
|---------------------------------|--|
| <b>Descripción</b>              | La persona usuaria le indica que cargue los datos de la anterior partida que fueron guardados en la base de datos.   |
| <b>Precondiciones</b>           | Tiene que haber una partida guardada de sesiones anteriores  |
| <b>Postcondiciones</b>          | Se carga los datos de la última partida guardada y se retoma el juego desde el punto en el que se dejó.  |
| <b>Referencias</b>              | RF4.2  |
| <b>Flujo normal de eventos</b>  | <p><b>1.</b> La persona usuaria dice «Continuar última partida».</p> <p><b>2.</b> Alexa carga los datos de la partida guardada en sesiones anteriores y recuerda el turno en el que se quedó el juego.</p> |
| <b>Flujo alterno de eventos</b> | <b>1.a.</b> Si no hay datos guardados, no podrá cargarlos y habrá que crear una partida nueva.   |

Cuadro 26: Caso de uso nº 7

| CU08: Minijuego verdadero o falso |   |
|-----------------------------------|---|
| <b>Descripción</b>                | El equipo/participante actual participa en el minijuego de verdadero o falso para la posibilidad de ganar 10 puntos.  |
| <b>Precondiciones</b>             | El equipo/participante actual ha caído en una casilla del minijuego verdadero o falso.  |
| <b>Postcondiciones</b>            | Se actualizan los puntos del equipo/participante actual si es necesario, y se anuncia el siguiente turno.   |
| <b>Referencias</b>                | RF6.1, RF6.2, RF7.1 y RF7.2   |
| <b>Flujo normal de eventos</b>    | <p><b>1.</b> Alexa le hace una pregunta de trivial de verdadero o falso al equipo/participante actual.</p> <p><b>2.</b> El equipo/participante actual responde con «verdadero» o «falso».</p> <p><b>3.</b> Alexa verifica si su respuesta es correcta o no, actualizando sus puntos en caso afirmativo, o dando una breve explicación de la respuesta correcta en caso contrario.</p> |
| <b>Flujo alterno de eventos</b>   | <p><b>2.a.</b> Si la respuesta proporcionada no es del tipo verdadero o falso, Alexa repetirá la pregunta, aclarando el formato de respuesta que debe dar.</p> <p><b>2.b.</b> Si Alexa no recibe una respuesta válida en un rato, repetirá la pregunta, aclarando el tipo de respuesta que debe dar.</p>  |

Cuadro 27: Caso de uso nº 8

| CU09: Minijuego conoce a tus compañeros |   |
|---|---|
| <b>Descripción</b>                      | El equipo/participante actual participa en el minijuego de conoce a tus compañeros para la posibilidad de ganar 15 puntos.  |
| <b>Precondiciones</b>                   | El equipo/participante actual ha caído en una casilla del minijuego conoce a tus compañeros.  |
| <b>Postcondiciones</b>                  | Se actualizan los puntos de los equipos/participantes involucrados si es necesario, y se anuncia el siguiente turno.  |
| <b>Referencias</b>                      | RF6.1, RF6.2, RF7.1 y RF7.2   |
| <b>Flujo normal de eventos</b>          | <ol style="list-style-type: none"> <li>1. Alexa le hace una pregunta al equipo/participante actual, acerca de otro equipo/participante aleatorio.</li> <li>2. El equipo/participante actual responde con «correcto/a» o «incorrecto/a».</li> <li>3. Alexa le pide al equipo/participante sobre el que iba la pregunta que confirme si la respuesta dada es correcta o incorrecta.</li> <li>4. El sujeto de la pregunta confirma si es correcta o incorrecta la respuesta del equipo/participante actual.</li> <li>5. Si es correcta, Alexa otorga puntos a ambas partes implicadas y pasa al siguiente turno. Si ha fallado, pasa de turno directamente.</li> </ol> |
| <b>Flujo alterno de eventos</b>         | <p><b>2.a.</b> Si la respuesta proporcionada no es del tipo correcto/a o incorrecto/a, Alexa repetirá la pregunta, aclarando el formato de respuesta que debe dar.</p> <p><b>2.b.</b> Si Alexa no recibe una respuesta válida en un rato, repetirá la pregunta, aclarando el tipo de respuesta que debe dar.</p>  |

Cuadro 28: Caso de uso nº 9

| CU10: Minijuego adivina la cifra |   |
|----------------------------------|---|
| <b>Descripción</b>               | El equipo/participante actual participa en el minijuego de conoce a adivina la cifra para la posibilidad de ganar 30 puntos.  |
| <b>Precondiciones</b>            | El equipo/participante actual ha caído en una casilla del minijuego adivina la cifra.   |
| <b>Postcondiciones</b>           | Se actualizan los puntos del equipo/participante que ha acertado si es necesario, y se anuncia el siguiente turno.  |
| <b>Referencias</b>               | RF6.1, RF6.2, RF7.1 y RF7.2   |
| <b>Flujo normal de eventos</b>   | <p><b>1.</b> Alexa le hace una pregunta al equipo/participante actual, cuya respuesta debe ser únicamente un número.</p> <p><b>2.</b> El equipo/participante actual responde con una cifra.</p> <p><b>3.</b> Alexa verifica si es correcta, dándole puntos en caso afirmativo. En caso de fallo, repetirá la pregunta al siguiente equipo/participante.</p> <p><b>4.</b> El siguiente responde con una cifra distinta que piense que pueda ser la correcta.</p> <p><b>5.</b> Alexa continúa preguntando al siguiente, así hasta que alguien acierte o se haya dado una vuelta completa por todos los equipos/participantes, en cuyo caso desvelará la solución.</p> |
| <b>Flujo alterno de eventos</b>  | <p><b>2.a. y 4.a.</b> Si la respuesta proporcionada no es un número, Alexa repetirá la pregunta, aclarando el formato de respuesta que debe dar.</p> <p><b>2.b. y 4.a.</b> Si Alexa no recibe una respuesta válida en un rato, repetirá la pregunta, aclarando el tipo de respuesta que debe dar.</p>   |

Cuadro 29: Caso de uso nº 10

| CU11: Minijuego recuerda la última casilla |  |
|--|--|
| <b>Descripción</b>                         | El equipo/participante actual participa en el minijuego de recuerda la última casilla para la posibilidad de ganar 25 puntos.  |
| <b>Precondiciones</b>                      | El equipo/participante actual ha caído en una casilla de recuerda la última casilla.   |
| <b>Postcondiciones</b>                     | Se actualizan los puntos del equipo/participante actual si es necesario, y se anuncia el siguiente turno.  |
| <b>Referencias</b>                         | RF6.1, RF6.2, RF7.1 y RF7.2  |
| <b>Flujo normal de eventos</b>             | <p><b>1.</b> Alexa le pregunta al equipo/participante actual el nombre de la última casilla en la que estaba, previa a la actual.</p> <p><b>2.</b> El equipo/participante actual responde con el nombre de la casilla en la que piensa que estaba en el turno anterior.</p> <p><b>3.</b> Alexa verifica si su respuesta es correcta o no, actualizando sus puntos en caso afirmativo, o desvelando el nombre de la casilla correcta en caso contrario.</p> |
| <b>Flujo alterno de eventos</b>            | <p><b>2.a.</b> Si la respuesta proporcionada no es un nombre de casilla válido, Alexa repetirá la pregunta, aclarando el formato de respuesta que debe dar.</p> <p><b>2.b.</b> Si Alexa no recibe una respuesta válida en un rato, repetirá la pregunta, aclarando el tipo de respuesta que debe dar. Si no recuerda ningún nombre de casilla, le indica que diga «No me acuerdo de la casilla» para pasar de turno.</p>                                   |

Cuadro 30: Caso de uso nº 11

| CU12: Minijuego recuerda la fecha |  |
|-----------------------------------|--|
| <b>Descripción</b>                | El equipo/participante actual participa en el minijuego de recuerda la fecha para la posibilidad de ganar 20 puntos.   |
| <b>Precondiciones</b>             | El equipo/participante actual ha caído en una casilla de recuerda la fecha.  |
| <b>Postcondiciones</b>            | Se actualizan los puntos del equipo/participante actual si es necesario, y se anuncia el siguiente turno.  |
| <b>Referencias</b>                | RF6.1, RF6.2, RF7.1 y RF7.2  |
| <b>Flujo normal de eventos</b>    | <ol style="list-style-type: none"> <li>1. Alexa le hace una pregunta sobre un día de la semana, mes o estación del año concreto.</li> <li>2. El equipo/participante actual responde con el nombre del día de la semana, mes o estación del año que piense que es la correcta.</li> <li>3. Alexa verifica si se ha acordado bien o no, actualizando sus puntos en caso afirmativo, o recordando la solución en caso contrario.</li> </ol> |
| <b>Flujo alterno de eventos</b>   | <p><b>2.a.</b> Si la respuesta proporcionada no es un día de la semana, mes o estación del año, Alexa repetirá la pregunta, aclarando el formato de respuesta que debe dar.</p> <p><b>2.b.</b> Si Alexa no recibe una respuesta válida en un rato, repetirá la pregunta, aclarando el tipo de respuesta que debe dar.</p>  |

Cuadro 31: Caso de uso nº 12

| CU13: Cerrar la skill           |  |
|---------------------------------|--|
| <b>Descripción</b>              | La persona usuaria interactúa con Alexa para cerrar la skill   |
| <b>Precondiciones</b>           | La skill está iniciada.  |
| <b>Postcondiciones</b>          | Se cierra la skill del juego de la oca, borrando los datos de la partida actual a no ser que se hayan guardado en la base de datos antes.  |
| <b>Referencias</b>              | RF4.3  |
| <b>Flujo normal de eventos</b>  | <p><b>1.</b> La persona usuaria dice «Cierra probando oca» para cerrar la skill.</p> <p><b>2.</b> Alexa manda la orden de terminar la sesión, marcando el fin de la partida y cerrando la skill.</p> |
| <b>Flujo alterno de eventos</b> | <p><b>2.a.</b> Si la persona usuaria intenta jugar después de haber cerrado la skill, Alexa dirá que no reconoce ninguno de los comandos y habrá que abrir la skill de nuevo.</p>                    |

Cuadro 32: Caso de uso nº 13

#### 4.3.2. Matriz de cobertura de requisitos funcionales

A continuación, se comprueba que todos los casos de uso definidos en la sección anterior referencian a un requisitos funcionales como mínimo.

Dicho de otra forma, todos los requisitos funcionales deben estar cubiertos por al menos un caso de uso.

| RF↓ | CU 01 | CU 02 | CU 03 | CU 04 | CU 05 | CU 06 | CU 07 | CU 08 | CU 09 | CU 10 | CU 11 | CU 12 | CU 13 |
|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| RF1 | ×     |       |       |       |       |       |       |       |       |       |       |       |       |
| 2.1 |       |       | ×     |       |       |       |       |       |       |       |       |       |       |
| 2.2 |       |       | ×     |       |       |       |       |       |       |       |       |       |       |
| 2.3 |       |       |       | ×     |       |       |       |       |       |       |       |       |       |
| 3.1 |       |       |       |       | ×     |       |       |       |       |       |       |       |       |
| 3.2 |       |       |       |       | ×     |       |       |       |       |       |       |       |       |
| 4.1 |       |       |       |       |       | ×     |       |       |       |       |       |       |       |
| 4.2 |       |       |       |       |       |       | ×     |       |       |       |       |       |       |
| 4.3 |       |       |       |       |       |       |       |       |       |       |       |       | ×     |
| 4.4 |       |       | ×     |       |       |       |       |       |       |       |       |       |       |
| 5.1 |       | ×     |       |       |       |       |       |       |       |       |       |       |       |
| 5.2 |       | ×     |       |       |       |       |       |       |       |       |       |       |       |
| 5.3 |       | ×     |       |       |       |       |       |       |       |       |       |       |       |
| 5.4 |       | ×     |       |       |       |       |       |       |       |       |       |       |       |
| 6.1 |       |       | ×     | ×     | ×     |       |       | ×     | ×     | ×     | ×     | ×     |       |
| 6.2 |       |       | ×     | ×     | ×     |       |       | ×     | ×     | ×     | ×     | ×     |       |
| 7.1 |       |       |       |       |       |       |       | ×     | ×     | ×     | ×     | ×     |       |
| 7.2 |       |       |       |       |       |       |       | ×     | ×     | ×     | ×     | ×     |       |

Figura 15: Matriz de cobertura de requisitos funcionales

### 4.3.3. Diagramas de secuencia de casos de uso

Los diagramas de secuencia son herramientas en el modelado de sistemas que permiten visualizar cómo los objetos interactúan a lo largo del tiempo, para así poder cumplir con los requisitos funcionales ligados a los casos de uso.

Se ha usado la herramienta de *Visual Paradigm* para diseñar los siguientes diagramas:

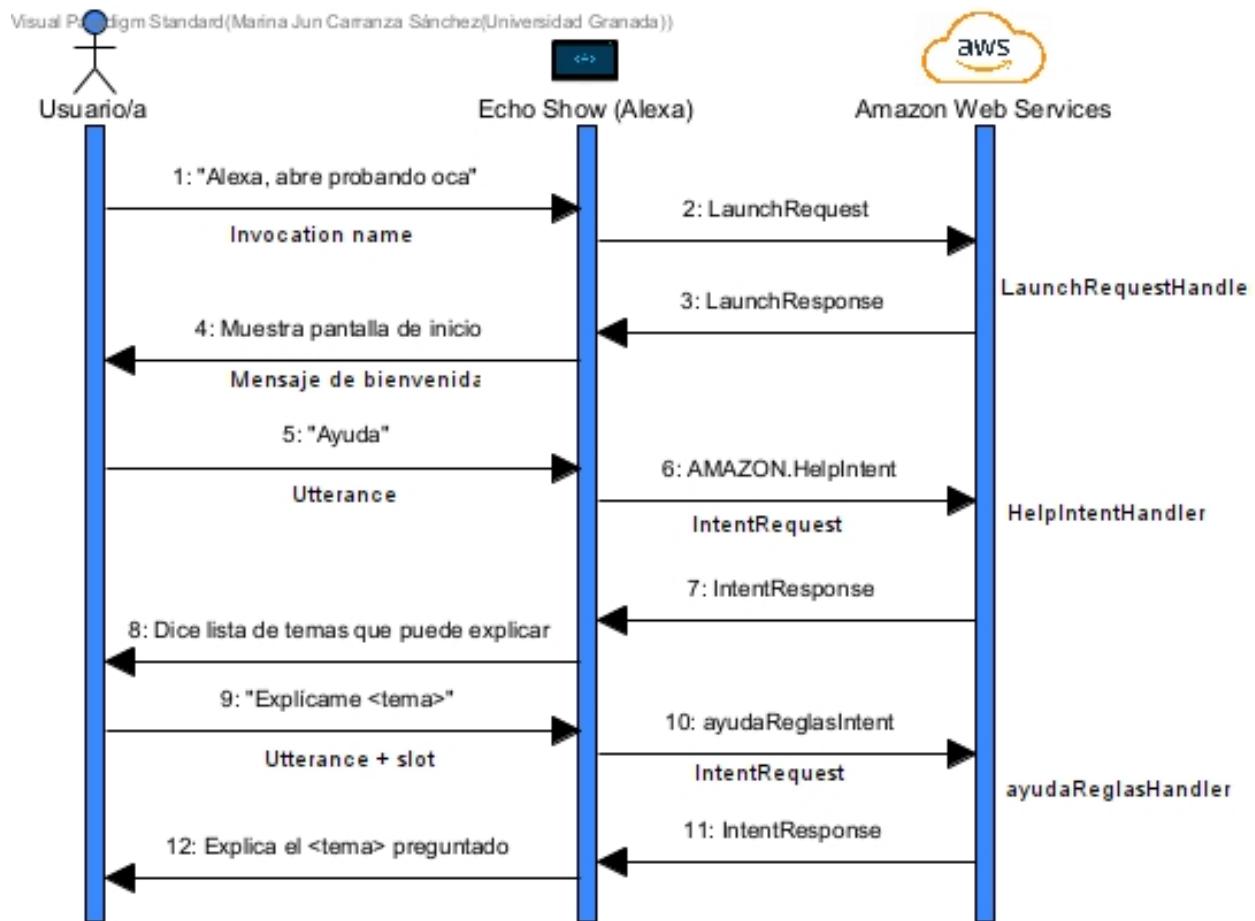


Figura 16: Diagrama de secuencia de CU01 y CU02.

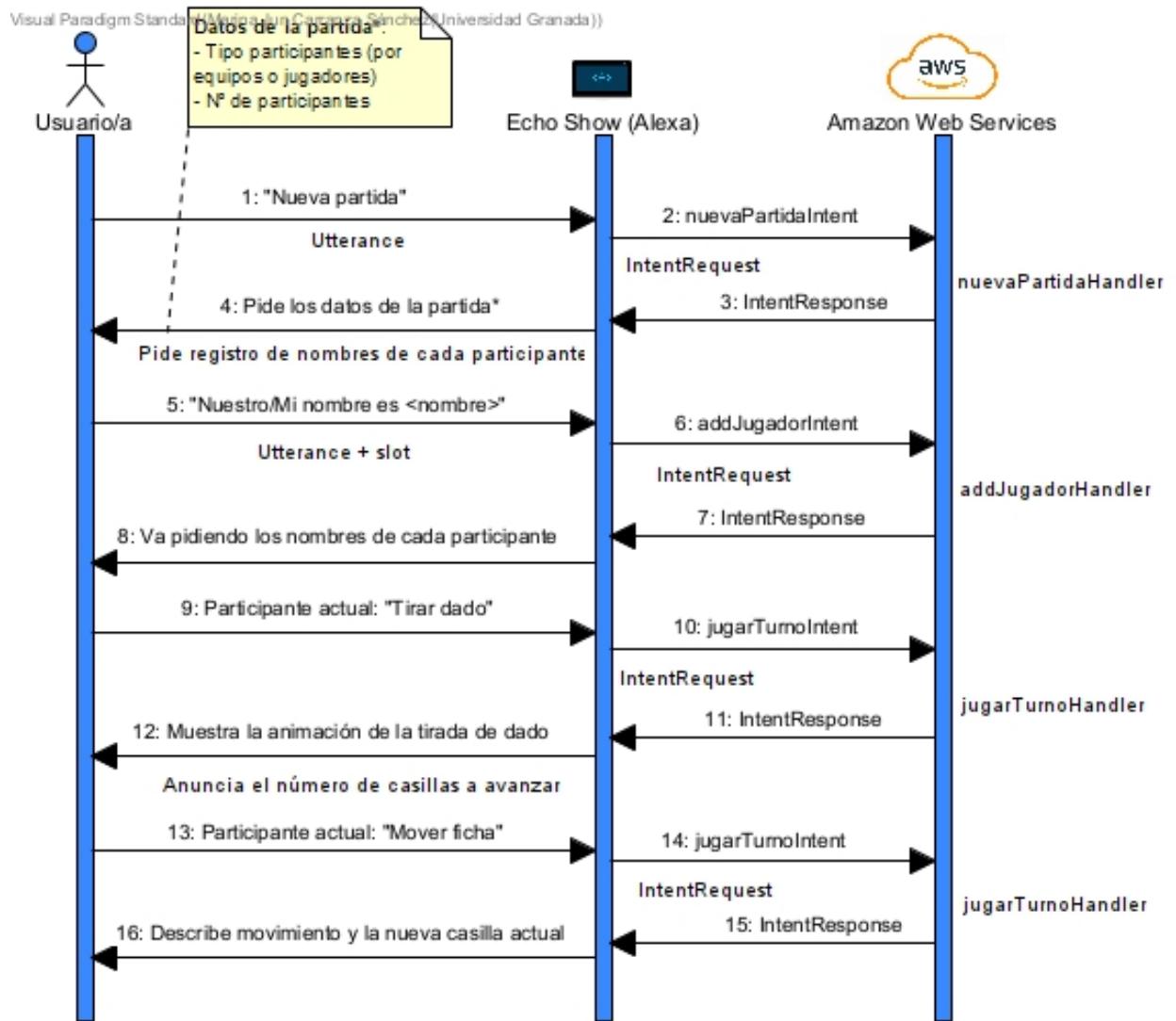


Figura 17: Diagrama de secuencia de CU03, CU04 y CU05.

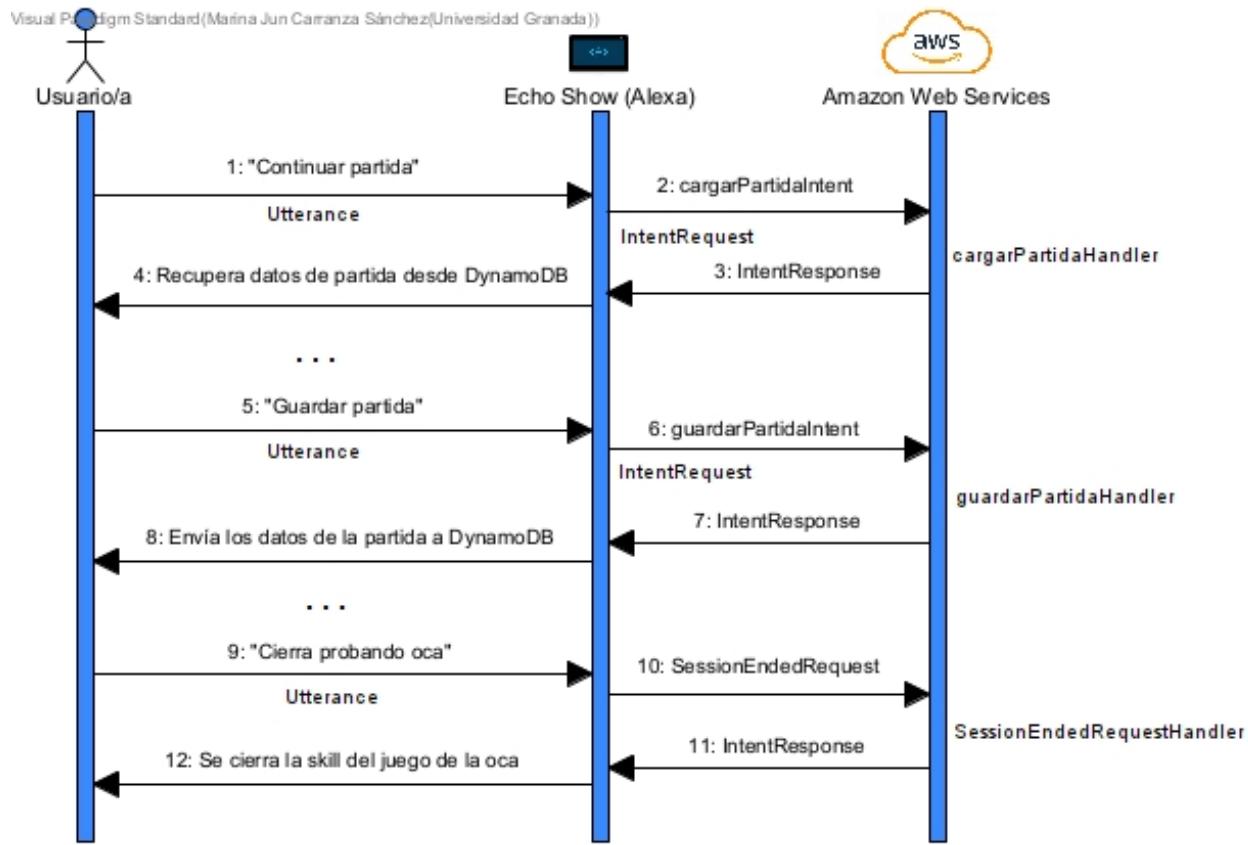


Figura 18: Diagrama de secuencia de CU06 y CU07 y CU13.

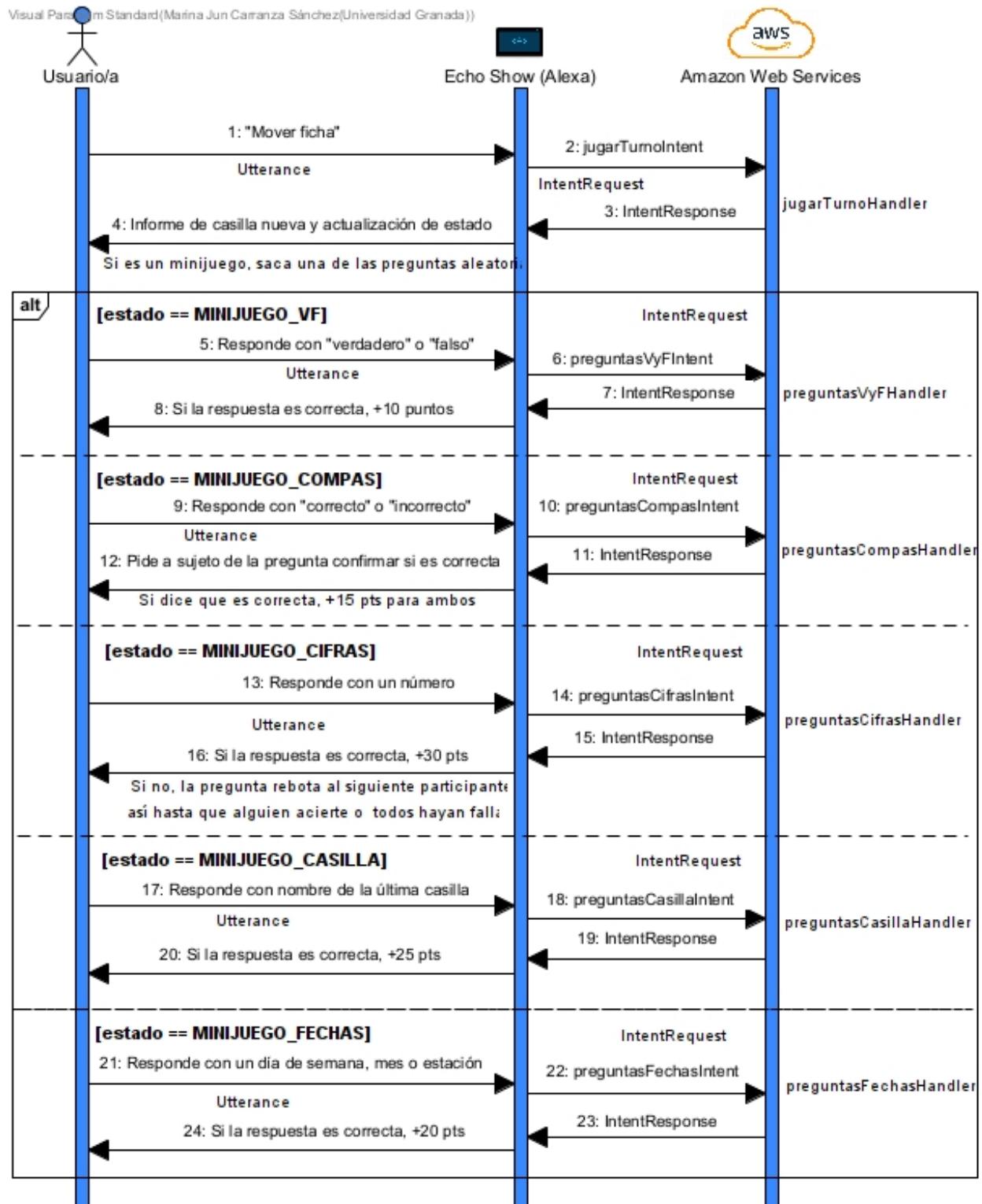


Figura 19: Diagrama de secuencia de CU08, CU09, CU10, CU11 y CU12.

## 5. Diseño

En esta etapa de la metodología ágil, se debe elaborar un diseño que responda a las dificultades que los adultos mayores puedan experimentar probando juegos digitales. Encontrar el punto medio entre entretenimiento y aprendizaje, hacer una correcta integración de los elementos visuales y garantizar la legibilidad son algunos de los aspectos que se tratarán en este apartado.

### 5.1. Diseño de la arquitectura

La arquitectura completa de la skill se construirá sobre varias herramientas de *Amazon Web Services* (AWS) que serán explicadas con mayor detalle en la sección 6. Estas se encuentran englobadas en *AWS Serverless Platform*, una plataforma que permite la creación de skills sin necesidad de disponer de un servidor propio.

Las tecnologías avanzadas de AWS más relevantes en el desarrollo de skills son: AWS Lambda, Amazon DynamoDB y Amazon S3.

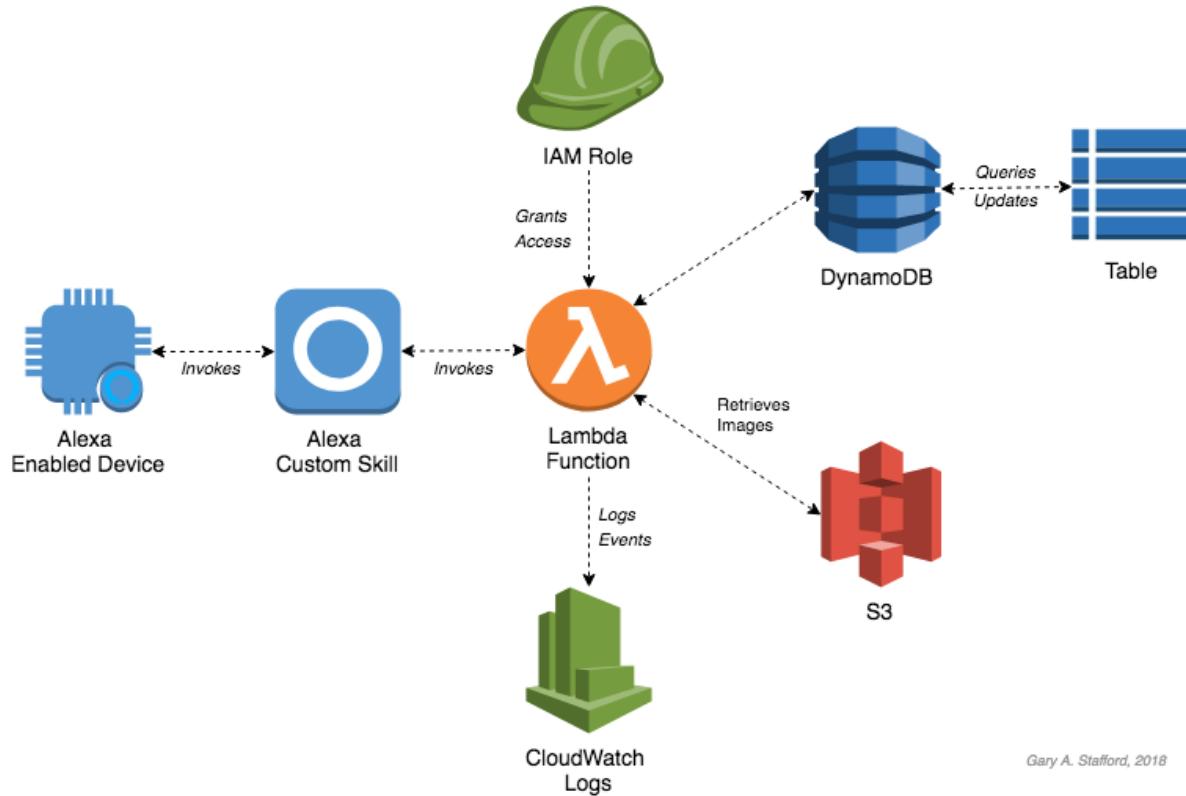


Figura 20: Arquitectura de una skill de Alexa con las tecnologías AWS (Stafford, 2018)

El proceso de creación de la habilidad para Alexa implica varios pasos clave (Stafford, 2018):

1. Definir el modelo de interacción por voz; es decir, cómo los usuarios pueden invocar la skill con diversas intenciones mediante comandos de voz.
2. Diseñar la interfaz, solo en caso de que se vaya a desplegar en dispositivos con pantallas, para mejorar la experiencia de usuario al mostrar información visual complementaria al audio.
3. Configurar las tablas en DynamoDB: para el almacenamiento persistente de información, usando una base de datos.
4. Crear un bucket en S3 donde se alojarán las imágenes y videos requeridos para el paso 2.
5. Programar la skill de Alexa, con funciones que permitan gestionar el flujo de conversación entre los usuarios y Alexa.
6. Escribir la función Lambda, responsable de procesar las entradas del usuario y devolver las respuestas adecuadas, que pueden incluir datos almacenados en DynamoDB y en S3.
7. Modificar el rol IAM predeterminado para que la función Lambda actualice la información de la base de datos según sea necesario.
8. Desplegar y probar la skill para asegurarse de que funciona según lo esperado.

Adicionalmente, para gestionar los elementos visuales que serán mostrados en la pantalla del dispositivo de Alexa, se tiene la siguiente arquitectura para el Lenguaje de Presentación de Alexa, comúnmente conocido como APL.

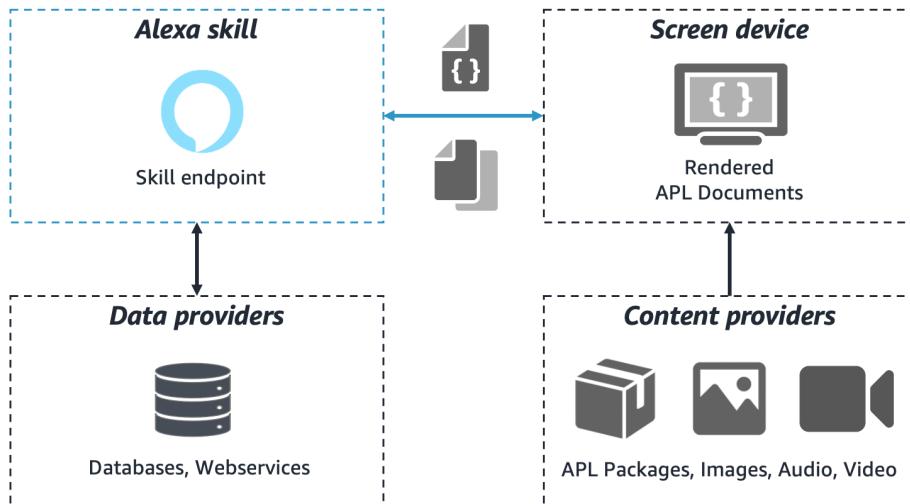


Figura 21: Arquitectura de una APL con los cuatro actores principales ([Alexa Developer Documentation](#))

Los dos agentes esenciales son:

- **La propia skill:** es la que inicia el proceso al enviar primero la plantilla del documento APL al dispositivo.
- **Los dispositivos con pantalla:** algunos dispositivos Alexa, como el Echo Show, son compatibles con este lenguaje de presentación y se encargan de mostrar por pantalla la plantilla APL.

Los anteriores pueden ser complementados por agentes opcionales como:

- **Los proveedores de datos:** normalmente bases de datos como DynamoDB, almacenan de forma externa a la skill información relevante que puede ser consultada y/o modificada.
- **Los proveedores de contenido:** como Amazon S3, que incluyen archivos multimedia externos, que se almacenan públicamente en la red y son referenciados desde la skill mediante URLs.

## 5.2. GDD: Documento de Diseño del Juego

Como bien se menciona en el libro *Game design workshop: a playcentric approach to creating innovative games* (Fullerton, 2008), el Game Design Document (GDD) es un documento que integra todos los aspectos relevantes de un videojuego, desde la mecánica y la narrativa hasta los elementos visuales y de audio.

Es útil porque sirve como roadmap (plan estratégico para alcanzar unos determinados objetivos a largo plazo) para todo el proceso de desarrollo, así como una guía para coordinar las tareas de los miembros del equipo a lo largo del camino hacia alcanzar dichos objetivos.

Se ha hecho una adaptación a una plantilla de uso gratuito para elaborar el documento de diseño del juego.

|   |   |  |
|---|---|--|
| <b>TÍTULO</b>   |   | <b>El Juego de la Oca</b>  |
| <b>DESCRIPCIÓN</b>  |   | Un juego de la Oca con minijuegos para mayores que promueve la socialización y el entretenimiento  |
| <b>TEMÁTICA</b>   | Basada en el juego tradicional, pero ampliada               | <b>GÉNERO</b><br>Juego de mesa interactivo por voz   |
| <b>CARACTERÍSTICAS CLAVE</b> <ul style="list-style-type: none"> <li>- Interacción por voz</li> <li>- Entretenimiento</li> <li>- Equipos/individual</li> </ul>   |   | <ul style="list-style-type: none"> <li>- Accesibilidad</li> <li>- Memoria/cognitivo</li> <li>- Interfaz simple</li> </ul> <ul style="list-style-type: none"> <li>- Educativo</li> <li>- Participativo</li> <li>- Aleatoriedad</li> </ul> |
| <b>ELEMENTOS VISUALES</b> <p>El juego incluye elementos visuales en la pantalla del dispositivo Alexa donde se esté ejecutando, que muestran información esencial (el equipo o jugador/a actual, la casilla actual y el lanzamiento de dado). Esto ayuda a seguir el progreso del juego y a que sea más intuitivo.</p>  |   |  |
| <b>AUDIO Y SONIDO</b> <p>En el juego, toda la interacción se realiza a través de audio, utilizando exclusivamente la voz de Alexa para proporcionar instrucciones, anunciar resultados, y describir el progreso del juego. Sin efectos de sonido adicionales, centrándose en la comunicación verbal clara y directa .</p>   |   |  |
| <b>NARRATIVA</b> <p>La narrativa del juego gira en torno a una travesía divertida y desafiante en un tablero virtual, pudiendo formar equipos o siendo participantes individuales, que deberán asumir un nombre con el que serán referidos durante toda la partida.</p> <p>A medida que avanzan por el tablero, desencadenarán una serie de eventos en función de las casillas en las que caigan. Se enfrentan a una serie de minijuegos que pondrán a prueba su memoria, conocimientos generales y acerca de sus compañeros de juego.</p> <p>Su modalidad por voz crea una experiencia inmersiva que combina entretenimiento y aprendizaje durante el viaje hacia la meta.</p> |   |  |
| <b>AUDIENCIA</b>  | Adultos mayores, individuos con mayor riesgo de aislamiento | <b>PLATAFORMA</b><br>Skill de Alexa para dispositivos Amazon con pantalla  |

Figura 22: Game Design Document página 1

|                                      |   |   |   |
|--------------------------------------|---|---|---|
| <b>OBJETIVO PRINCIPAL DEL JUEGO</b>  | Ser el primer equipo o jugador/a en llegar a la meta  |   |   |
| <b>OBJETIVO SECUNDARIO DEL JUEGO</b> | Ser el equipo o jugador/a con mayor número de puntos acumulados al final del juego.   |   |   |
| <b>FACILITADORES</b>                 |   | <b>OBSTÁCULOS</b>   |   |
| <b>ACTIVOS</b>                       | Conseguir una gran cantidad de puntos gracias a un número elevado de respuestas acertadas en los minijuegos.  | <b>PASIVOS</b>  | Tener mala suerte y caer a menudo en casillas de penalización que ocasionen la pérdida de turnos, sacar números bajos en las tiradas de dado que provoquen un avance lento, no caer en casillas de minijuegos y no poder ganar puntos, etc. |
| <b>BUCLE PRINCIPAL</b>               | <ol style="list-style-type: none"> <li>Se crea y configura una nueva partida o se cargan los datos de la anterior que fueron guardados en sesiones pasadas.</li> <li>El equipo/participante del turno actual dice “tirar dado”.</li> <li>Alexa muestra y anuncia el resultado de la tirada.</li> <li>El equipo/participante del turno actual dice “mover ficha”.</li> <li>Alexa informa del movimiento y la casilla en la que ha caído. Si es una especial, desencadena el evento correspondiente.</li> <li>Alexa anuncia el comienzo del siguiente turno y se repiten los pasos a partir del 2.</li> </ol> | <b>CONDICIONES PARA GANAR</b><br>Ser el primer equipo o jugador/a en llegar a la meta, y/o conseguir más puntos que nadie.                        |   |
| <b>MECÁNICAS Y FUNCIONALIDADES</b>   |   | <b>CONDICIONES PARA PERDER</b><br>Que otro equipo o participante llegue antes a la casilla de meta y no haber conseguido más puntos que el resto. |   |
| <b>EQUIPO DE DESARROLLO</b>          | <p>Marina Jun Carranza Sánchez</p> <p>Contribuir a las iniciativas de emplear los modelos de asistentes conversacionales para mejorar la calidad de vida de las personas mayores.</p>   |   |   |

Figura 23: Game Design Document página 2

### 5.3. Modelo conceptual del juego de la oca

Aunque en una skill de Alexa los modelos conceptuales no se ajustan de la misma forma que lo harían en un sistema orientado a objetos tradicional, sí se puede plantear un diagrama de conceptos que encapsule las clases del juego de la oca y cómo se relacionan entre sí.

Se ha ilustrado el diagrama conceptual de forma aislada a la estructura particular de la skill:

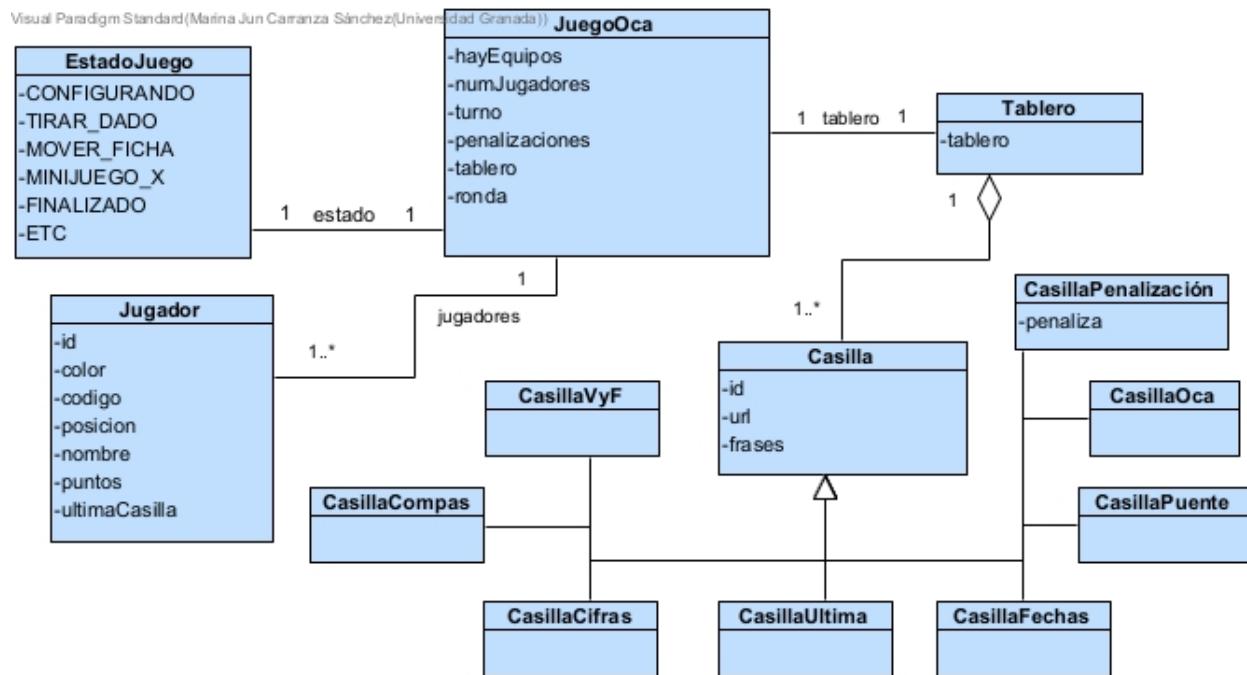


Figura 24: Diagrama conceptual del juego de la oca ([Visual Paradigm](#))

### 5.4. Diseño de la interfaz de usuario

Aunque no se vaya a mostrar en la pantalla del dispositivo de Alexa, se ha realizado un diseño con Adobe Photoshop del tablero personalizado que utilizará la skill de la oca, lo que ha ayudado en gran medida a visualizarlo para poder implementarlo con mayor facilidad más adelante.



Figura 25: Tablero personalizado para la skill del juego de la oca

La distribución de las casillas es la siguiente: 18 especiales tomadas del juego tradicional (oca, puente, laberinto, pozo, cárcel y hotel), 15 de tipo minijuego y 30 normales (no desencadenan ninguna acción especial, salvo la de meta). Por tanto, los porcentajes redondeados de las proporciones de cada una son: 28,57 %, 23,81 % y 47,62 %, respectivamente.

### 5.4.1. Bocetos y mockups

Para que las personas mayores puedan tener una experiencia de usuario accesible y cómoda, el principio de usabilidad número uno a seguir es mantener un diseño sencillo y minimalista, con elementos claros y bien organizados, contrastes aceptables y tamaños de texto grandes.

Es lo que se ha tratado de lograr con los siguientes bocetos, que definen la apariencia de la interfaz, habiendo un total de cuatro estados para mantenerlo lo más simple posible. La herramienta empleada es [Lucidchart](#), un software gratuito para el diseño de modelos, diagramas y planificación de procesos y tareas.

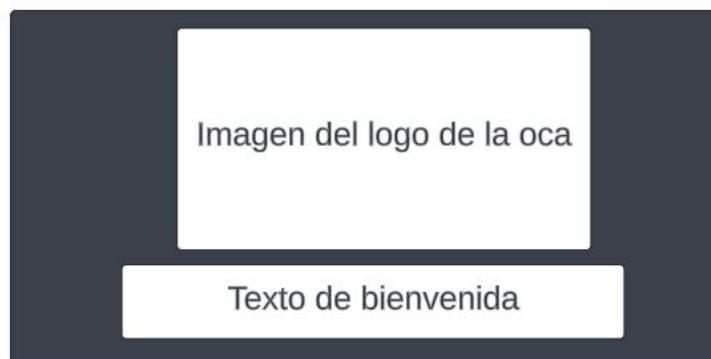


Figura 26: Boceto de la pantalla de inicio de la skill

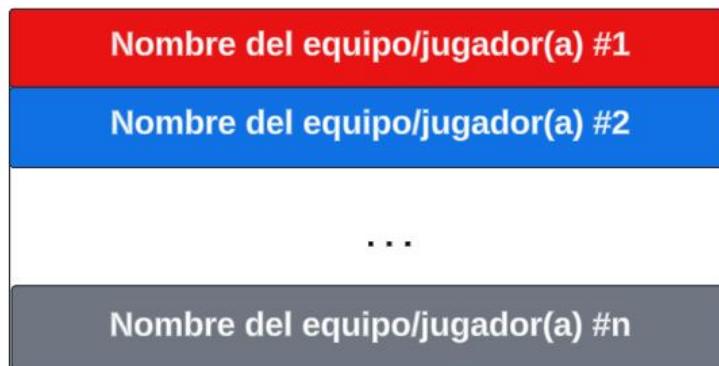


Figura 27: Boceto de la pantalla con los participantes



Figura 28: Boceto de la pantalla de lanzamiento de dado

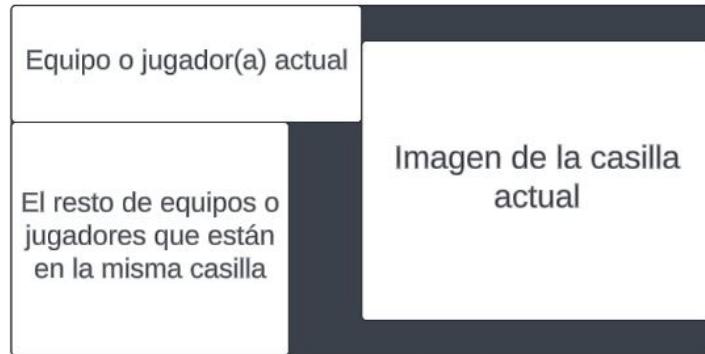


Figura 29: Boceto de pantalla con la casilla y participante actual

#### 5.4.2. Diagrama de flujo entre pantallas

Debido a que solo existen cuatro tipos de pantallas en todo el juego, el diagrama de flujo entre ellas es igual de simple. Pues se pretende que la navegación entre pantallas sea automática e intuitiva, mostrando únicamente los elementos que pueden ayudar a los participantes a visualizar mejor la progresión del juego.



Figura 30: Diagrama de flujo entre pantallas

#### 5.4.3. Cuestiones de estética, usabilidad y accesibilidad

Para el estudio de las variables relevantes a la ergonomía del juego, se ha seguido la metodología utilizada en un juego digital para adultos mayores llamado *Solitaire Quiz*. Este está inspirado en el *Solitario*, un juego de cartas tradicional, y a diferencia del original, incluye contenidos didácticos en forma de pequeños cuestionarios. (Mendoza et al., 2017).

| Categoría        | Variables  |
|------------------|--|
| Diseño del juego | <ul style="list-style-type: none"> <li>- Desafío</li> <li>- Contenido del aprendizaje</li> <li>- Retroalimentación</li> </ul>  |
| Usabilidad       | <ul style="list-style-type: none"> <li>- Ambiente externo al juego</li> <li>- Ambiente interno al juego</li> <li>- Elementos visuales</li> <li>- Dispositivos</li> </ul> |
| Legibilidad      | <ul style="list-style-type: none"> <li>- Texto</li> <li>- Imágenes</li> <li>- Audio</li> </ul>   |

Cuadro 33: Dimensiones y variables de la ergonomía de la app

## 6. Análisis tecnológico

### 6.1. Fundamentos de una skill

Qué es una skill

#### 6.1.1. Términos comunes en el desarrollo de skills

intents  
utterances  
handlers  
slots

#### 6.1.2. Memoria y persistencia de datos

session attributes y DynamoDB entre sesiones.

### 6.2. Alexa Skills Kit (ASK)

ASK es un conjunto de APIs y herramientas que facilitan la integración de nuevas habilidades en Alexa, permitiendo crear distintas clases de skills, desde personalizadas hasta otras específicas para vídeo, listas y hogar inteligente.

El tipo de skill que mejor se ajusta a los objetivos preestablecidos es la personalizada o *Custom Skill*, pues esta permite más flexibilidad a los desarrolladores, permitiéndoles adaptarla a las necesidades de la aplicación a desarrollar.

Se puede importar ASK, a través del nombre de la biblioteca: *ask-sdk-core*.

### 6.3. AWS Serverless Platform

Para el desarrollo de skills de Alexa, se puede optar o bien por la función Lambda de AWS, o bien por un servicio web distinto. La primera opción pertenece a un conjunto de herramientas de desarrollador y servicios en la nube de alto rendimiento que componen la Plataforma sin Servidor de AWS.

Se va a utilizar **AWS Lambda** para la creación de este juego digital, aprovechando así las múltiples funcionalidades y material de apoyo para encaminar el proceso de desarrollo. Además, al poder hacer uso de los otros servicios incluidos en esta plataforma (DynamoDB, Amazon S3, etc), se garantiza cierta centralización y absoluta compatibilidad entre ellos.

## 6.4. Alexa Presentation Language (APL)

La parte visual de la skill se gestiona mediante el lenguaje de presentación de Alexa (APL), a través del envío de documentos APL al dispositivo en forma de una directiva que se verá más adelante.

Un documento APL consiste en un fichero JSON que define la estructura y disposición de elementos a mostrar por la pantalla del dispositivo de Alexa. A continuación se muestra un ejemplo básico de documento APL que imprime por pantalla una cadena de texto:

```
{  
  "type": "APL",  
  "version": "2024.1",  
  "mainTemplate": {  
    "item": {  
      "type": "Text",  
      "text": "Hello, world"  
    }  
  }  
}
```

Figura 31: Ejemplo de documento APL básico ([Alexa Developer Documentation](#))

## 7. Desarrollo

Una vez pasada a la fase de desarrollo, el objetivo es transformar el diseño y requisitos concretados previamente en una aplicación funcional.

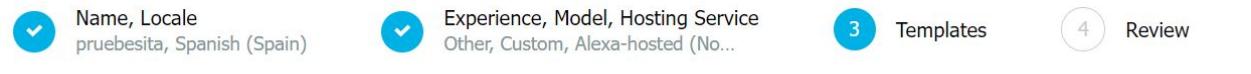
Este proceso implica la creación de la skill, el desarrollo de código y la realización de pruebas, pero no sin antes haber cumplido dos prerrequisitos:

- Tener una cuenta en *Alexa developer console*, que es de carácter totalmente gratuito y permitirá el alojamiento y despliegue de la skill.
- Tener una cuenta de *Amazon Web Services* (AWS), para lo que se debe ingresar un método de pago. Esto es necesario en caso de que el uso de recursos exceda los límites de la oferta gratuita y Amazon tenga que cobrar una cantidad adicional.

### 7.1. Creación de la skill y el modelo de interacción

Hay una gran cantidad de guías disponibles para aprender a crear una skill desde cero, y la propia [página oficial de desarrolladores Alexa](#) ofrece diversos documentos de apoyo, los cuales se han utilizado como material de referencia en esta etapa del proyecto.

1. Una vez iniciada la sesión en la consola de desarrolladores de Alexa, se abre el menú de creación de skills.
2. Se elige la región donde se van a alojar los servicios AWS predeterminados de la nueva skill, en este caso, la opción *eu-west-1*, localizada en Irlanda, que es la más cercana.
3. Se selecciona un nombre para la skill y el idioma para el que va a estar implementado el modelo.
4. Para la decisión del modelo, el que mejor se ajusta a las especificaciones de este proyecto es el *Custom model* (o personalizado), que permite una mayor flexibilidad.
5. Como la idea es que Alexa se encargue de alojar el backend de la skill, se elige una de las dos opciones de *Alexa-hosted*, en este caso la del entorno de ejecución de Node.js v16.x.
6. Se puede añadir una plantilla o importar una skill alojada en un repositorio de Git. En este caso, se ha optado por trabajar a partir de una plantilla básica, que solo incluye un ejemplo simple de «*Hello world*» (ver figura 29). Sobre esta base se añadirán todas las funcionalidades del juego.



## Templates

Select a skill template from the list below or import a skill shared by the Alexa community as a public Git repository.

Templates come with a pre-built interaction model and default endpoint so that you can start testing as soon as the skill is created.

### Start from Scratch

This skill gets you started with the required intents and with code demonstrating "Hello World" functionality if you are building an Alexa-hosted skill.

[Learn more](#)

By [Alexa](#)

### Fact Skill

Build an engaging fact skill about any topic. Alexa will select a fact at random and share it with the user when the skill is invoked. [Learn more](#)

Includes: custom intents, Personalization

By [Alexa](#)

### Scheduling Skill

Build a skill to allow users to schedule appointments on your calendar, receive email confirmations and reminders.

[Learn more](#)

Includes: voice permissions, reminders, API calls, session persistence

By [Dabble Lab](#)

Figura 32: Menú de creación de una skill de Alexa

Una vez creada y finalizada la configuración básica, se abrirá automáticamente el menú de *Build*, desde el que se pueden ajustar varios aspectos de la skill.

Lo primero es declarar el nombre de invocación, que es con el que se llamará a la función de lanzamiento en el momento en el que se abre la skill, y se puede ajustar desde el menú lateral de *Invocations*. Alexa establece una serie de restricciones sobre este parámetro: no puede incluir artículos o preposiciones, debe componerse de mínimo dos palabras, si contiene números estos deben ser escritos de manera completa, no debe incluir mayúsculas ni palabras reservadas como «Alexa, skill, app,» etc.

El nombre de invocación elegido es «probando oca», por tanto cuando se quiera iniciar la skill, habrá que decir: «Alexa, abre probando oca».

Otro elemento desplegable revelante del menú lateral es el modelo de interacción (*Interaction Model*), donde se definen los *intents*. Inicialmente hay cinco predeterminados, de los cuales cuatro son de carácter obligatorio y por tanto no pueden borrarse:

- **AMAZON.CancelIntent**: cancela la acción actual y termina la interacción con la skill.
- **AMAZON.HelpIntent**: proporciona información sobre cómo usar la habilidad.
- **AMAZON.StopIntent**: detiene la acción en curso y finaliza la interacción con la skill.
- **AMAZON.NavigateHomeIntent**: regresa al inicio de o a la pantalla principal.

- **HelloWorldIntent**: ejecuta la acción personalizada predefinida, que consiste en saludar a la persona usuaria. Es la única que puede eliminarse.

También se puede acceder y modificar el archivo JSON del modelo de interacción directamente, aunque es aconsejable utilizar en lugar de ello la interfaz de configuración, ya que cada vez que se monta la skill este fichero se actualiza automáticamente.

Otro elemento interesante son los Assets, que permiten la creación de tipos personalizados de slots, la consulta del historial de compilación de la skill y la configuración del *endpoint*.

Al tratarse de una skill alojada por Alexa, el endpoint se trata de la función Lambda de AWS encargada de la ejecución del código de la skill. Además, por defecto se dispone de tres de ellos, localizados en distintas regiones del mundo: Virginia del Norte, Irlanda y Oregon.

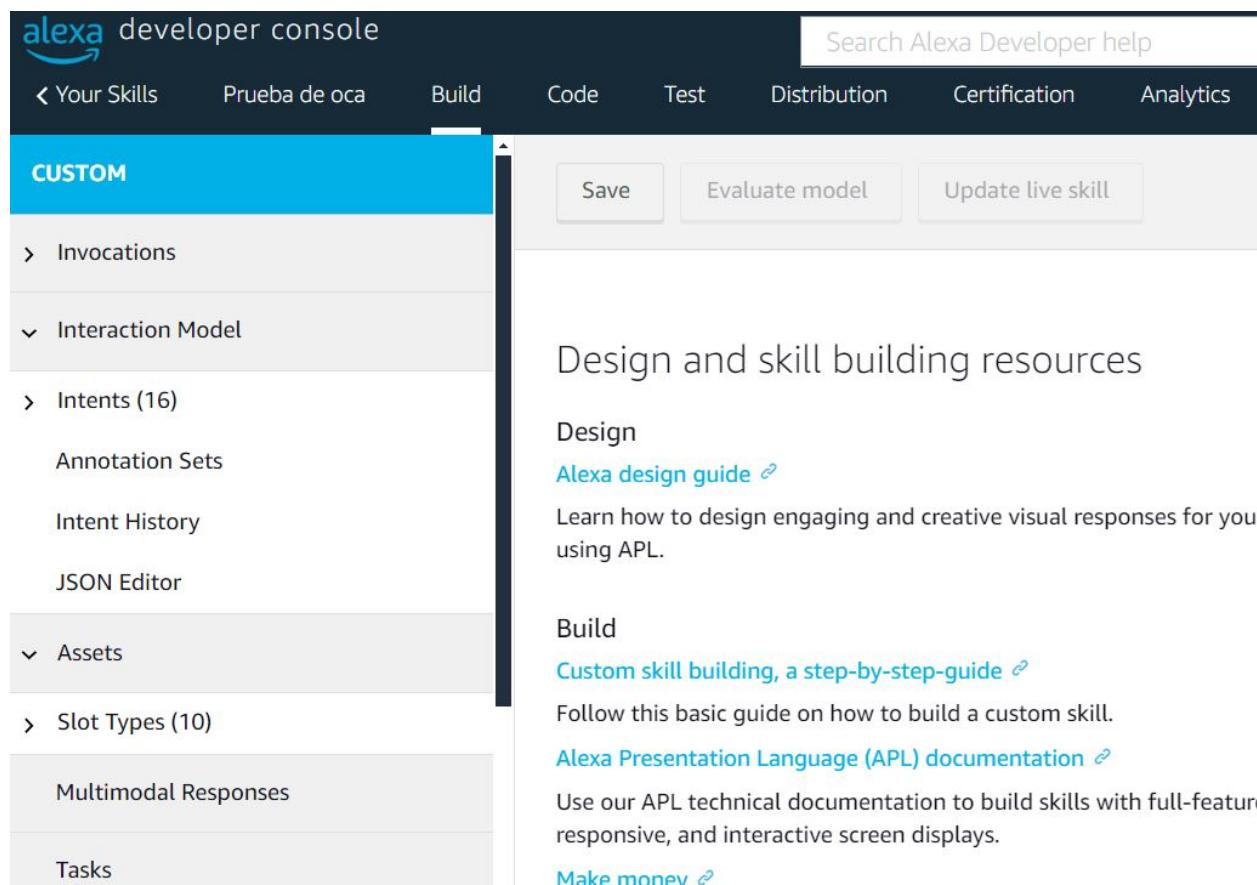


Figura 33: Menú principal de montaje de la skill de Alexa

Si se navega al menú de código desde la barra de herramientas, se puede encontrar la siguiente estructura de archivos:

```

1  /*
2   * This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK for Node.js
3   * Please visit https://alexa.design/cookbook for additional examples on integrating your skill with session persistence, api calls, and more.
4   */
5  const Alexa = require('ask-sdk-core');
6
7  const LaunchRequestHandler = {
8    canHandle(handlerInput) {
9      return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';
10    },
11    handle(handlerInput) {
12      const speakOutput = 'Welcome, you can say Hello or Help. Which would you like to try?';
13
14      return handlerInput.responseBuilder
15        .speak(speakOutput)
16        .reprompt(speakOutput)
17        .getResponse();
18    }
19  };
20
21  const HelloWorldIntentHandler = {
22    canHandle(handlerInput) {
23      return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
24        && Alexa.getIntentName(handlerInput.requestEnvelope) === 'HelloWorldIntent';
25    },
26  };

```

Figura 34: Menú del código de la skill de Alexa

El fichero *index.js* es el más importante, ya que actúa como entrypoint o punto de entrada de la skill, donde se exporta la función de Lambda. Es el que gestiona todos los handlers (manejadores de solicitudes), que determinan el flujo de conversación de Alexa gracias a la capacidad de establecer respuestas concretas a cada intent. Los handlers predefinidos son:

- **LaunchRequestHandler:** se activa cuando se abre la skill sin especificar un comando, respondiendo con un mensaje de bienvenida.
- **HelloWorldIntentHandler:** responde cuando el usuario invoca a *HelloWorldIntent*, que se limita a saludar.
- **HelpIntentHandler:** maneja el *AMAZON.HelpIntent*, puede invocarse cuando se necesita ayuda.
- **CancelAndStopIntentHandler:** gestiona *AMAZON.CancelIntent* y *AMAZON.StopIntent*, que sirve para detener la skill.
- **FallbackIntentHandler:** se activa si dice algo que no coincide con ninguno de los intents definidos.

- **SessionEndedRequestHandler**: se invoca cuando una sesión termina, ya sea por un comando de usuario o error.
- **IntentReflectorHandler**: útil para la depuración de la skill, ya que responde con el nombre del intent con el que fue llamado.
- **ErrorHandler**: sirve para la captura de errores de cualquier tipo.

Otro archivo esencial en cualquier proyecto realizado con Node.js es *package.json*, donde se definen las dependencias y bibliotecas requeridas, además de permitir la configuración de scripts, pruebas e información acerca de la versión, nombre del proyecto, etc.

Por otro lado, los dos ficheros restantes (*local-debugger.js* y *util.js*) no han sido necesarios para el desarrollo del proyecto, pero pueden servir como base para facilitar la depuración local de la skill y definir funciones de utilidad generales.

## 7.2. Configuración de servicios de AWS

La plataforma de servicios AWS ofrece una amplia gama de mecanismos de monitorización, almacenamiento, gestión de políticas, computación, etc. Se caracteriza, entre otras cosas, por la flexibilidad y escalabilidad que permite gracias a su modelo de «pagar solo lo que se usa».

En la consola de desarrolladores de AWS aparecen todos los servicios usados recientemente, facilitando la navegación entre ellos.

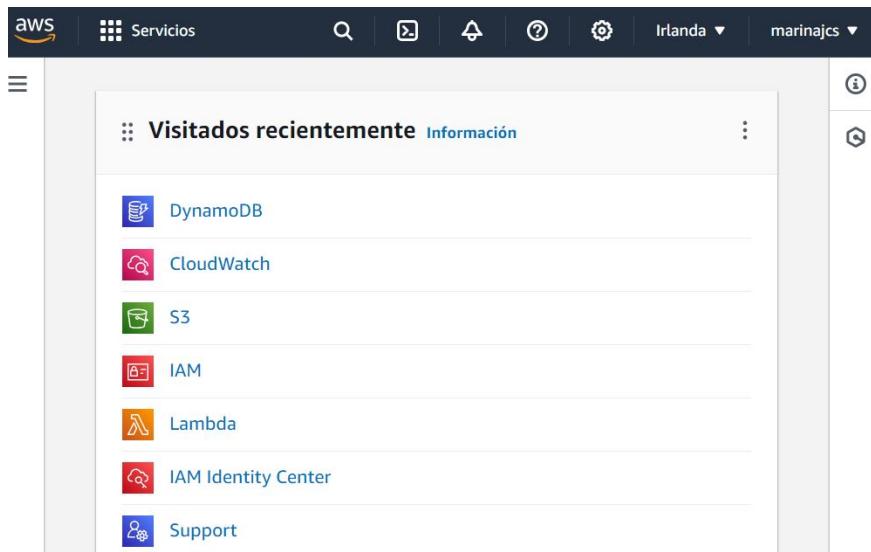


Figura 35: Menú de la consola de desarrolladores de AWS

### 7.2.1. Identity and Access Management (IAM)

El primer servicio utilizado es IAM, con el objetivo de crear un rol capaz de otorgar permisos temporales de acceso a DynamoDB a skills de Alexa.

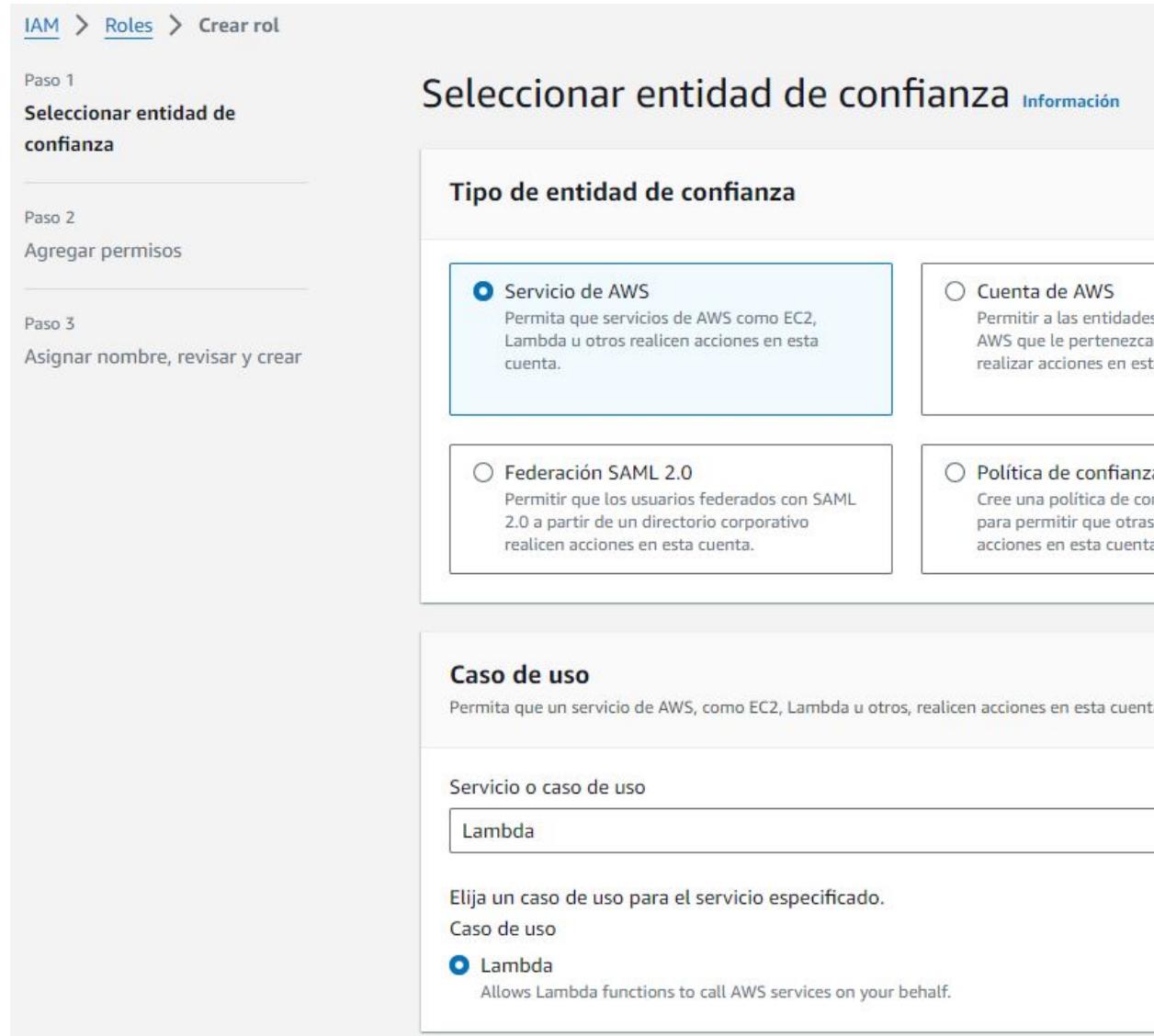


Figura 36: Menú de creación de un rol IAM

En el menú de creación del rol, hay que rellenar los siguientes parámetros de configuración: el tipo de entidad de confianza (un servicio de AWS), el caso de uso (la función de Lambda), las políticas de permisos a agregar (*AmazonDynamoDBFullAccess*) y el nombre del rol que se está creando.

The screenshot shows the AWS IAM Role configuration page for 'rol-dynamo'. At the top, it says 'Allows Lambda functions to call AWS services on your behalf.' Below is a 'Resumen' section with details like creation date, ARN, last activity, and maximum session duration. A navigation bar at the bottom includes 'Permisos', 'Relaciones de confianza', 'Etiquetas', 'Access Advisor', and 'Revocar las sesiones'. The 'Permisos' tab is selected, showing a table of three policies: 'AmazonDynamoDBFullAccess', 'AmazonS3ReadOnlyAccess', and 'AWSLambdaBasicExecutionRole'. Each policy has a preview icon, name, type ('Administrada por AWS'), and the number of entities associated with it (2, 1, 1 respectively).

|                          | Nombre de la política                       | Tipo                 | Entidades asociadas |
|--------------------------|---|----------------------|---------------------|
| <input type="checkbox"/> | <a href="#">AmazonDynamoDBFullAccess</a>    | Administrada por AWS | 2                   |
| <input type="checkbox"/> | <a href="#">AmazonS3ReadOnlyAccess</a>      | Administrada por AWS | 1                   |
| <input type="checkbox"/> | <a href="#">AWSLambdaBasicExecutionRole</a> | Administrada por AWS | 1                   |

Figura 37: Rol creado para la skill y sus permisos en AWS

Para que la skill pueda usar los recursos de AWS que será definidos en las secciones siguientes, se necesita modificar el fichero JSON que especifica las relaciones de confianza del rol. Para ello, se añade la siguiente entrada:

Permisos | **Relaciones de confianza** | Etiquetas | Access Advisor | Revocar las sesiones

## Entidades de confianza

Entidades que pueden asumir este rol en condiciones especificadas.

```
1 - [ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Principal": {  
7                 "AWS": "arn:aws:iam::██████████:role/AlexaHostedSkillLambdaRole"  
8             },  
9             "Action": "sts:AssumeRole"  
10        }  
11    ]  
12 }]
```

Figura 38: Configuración de las relaciones de confianza del rol

El texto censurado es el ARN (Amazon Resource Name) o identificador del rol de ejecución IAM que la función Lambda asume al ejecutarse la skill de Alexa. Este se puede obtener desde la consola de desarrolladores de Alexa, en el menú del código, con la opción *Integrate*, especialmente diseñada para poder vincular la skill a servicios personales de AWS.

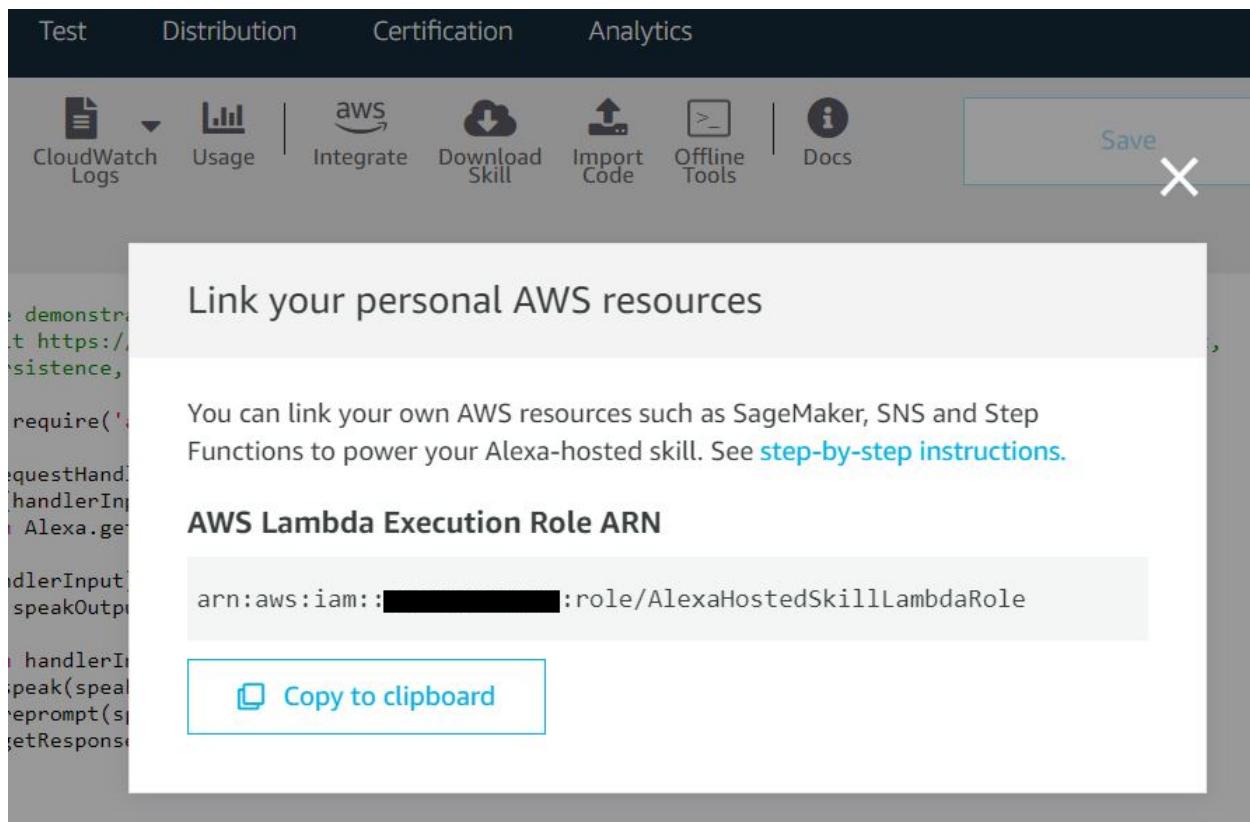


Figura 39: El AWS Lambda Execution Role ARN de la skill

### 7.2.2. Amazon Simple Storage Service (S3)

Como su nombre indica, es un servicio ofrecido en AWS que permite almacenar objetos de diversos tipos (textos, binarios, documentos, audios, comprimidos, etc). Sin embargo, dados los requisitos de este juego, solo será necesario guardar lo siguiente: las imágenes asociadas a cada casilla, la de la pantalla de bienvenida y los videos con las animaciones de los lanzamientos de dado.

En Amazon S3, se ha adoptado el término de *bucket* como un contenedor para almacenar objetos. Cada bucket tiene un nombre único y puede simular una estructura de directorios, además pueden ser de acceso público o privado, al igual que sus elementos, que pueden configurarse a través de políticas de acceso.

Para este proyecto, se ha creado un bucket de nombre *bucket-oca* para guardar todos los archivos multimedia que necesita la skill.

The screenshot shows the AWS S3 console interface for the bucket 'bucket-oca'. At the top, there's a navigation bar with 'Amazon S3 > Buckets > bucket-oca'. Below it, the bucket name 'bucket-oca' is displayed with a 'Información' link. A horizontal menu bar includes 'Objetos', 'Propiedades', 'Permisos', 'Métricas', 'Administración', and 'Puntos de acceso'. The 'Objetos' tab is selected, showing a list of three items:

|                          | Nombre                        | Tipo    | Última modificación               | Tamaño   | Clase de almacenamiento |
|--------------------------|-------------------------------|---------|-----------------------------------|----------|-------------------------|
| <input type="checkbox"/> | <a href="#">casillas-oca/</a> | Carpeta | -                                 | -        | -                       |
| <input type="checkbox"/> | <a href="#">oca.png</a>       | png     | 21 Aug 2024<br>6:22:22 PM<br>CEST | 126.1 KB | Estándar                |
| <input type="checkbox"/> | <a href="#">videos-dado/</a>  | Carpeta | -                                 | -        | -                       |

Below the list are several action buttons: 'Copiar URI de S3', 'Copiar URL', and 'Descargar'. There's also a search bar labeled 'Buscar objetos por prefijo'.

Figura 40: Bucket de Amazon S3 para la skill de la oca

### 7.2.3. DynamoDB

Como se ha mencionado en la sección 6.1.2., los datos de una skill no persisten de una sesión a otra, por lo que es necesario un mecanismo para guardarlos. Aquí entra el papel de las bases de datos, en particular DynamoDB, también integrado en AWS.

Esta base de datos es de tipo NoSQL y es una alternativa interesante no solo por su total compatibilidad con las herramientas actuales del proyecto debido a que es otro servicio Amazon, sino también por su alta capacidad de adaptación a cualquier volumen de datos.

Es conocido por optimizar sus tiempos de respuesta y llevar a cabo un escalado automático en función del tráfico, gracias a su esquema NoSQL, cuyos elementos de las tablas se identifican mediante claves primarias únicas. Estas últimas pueden ser de partición o compuestas (unión de una clave de partición con una de ordenamiento).

Para este proyecto, se han creado dos tablas:

- **JuegoOca:** su clave primaria es *idJuego*, que siempre valdrá 0 debido a que se pretende

guardar los datos de una sola partida entre sesiones. Tiene los elementos generales de un juego de la oca, que son el estado, el turno y ronda actual, el número de jugadores, el tipo de participantes...

- **Jugador:** identificada por la clave primaria *idJugador*, que a la vez determina el orden de turno, y será un valor entero único entre 0 y el el numero de total de participantes menos uno. Los atributos de cada elemento son: el nombre, el color, los puntos, la posición y penalizaciones actuales, etc.

The screenshot shows the AWS DynamoDB console interface. The top navigation bar includes the AWS logo, a services menu, and a dropdown for 'Irlanda'. Below the header, the left sidebar is titled 'DynamoDB' and contains links for 'Panel', 'Tablas' (which is selected), 'Explorar elementos', 'Editor PartiQL', 'Copias de seguridad', 'Exportaciones a S3', 'Importaciones de S3', 'Integraciones', 'Capacidad reservada', and 'Configuración'. A 'DAX' section is also present. The main content area is titled 'Tablas (2) Información' and shows a list of two tables: 'JuegoOca' and 'Jugador'. The 'JuegoOca' table has attributes: 'Nombre' (with a value of 'JuegoOca'), 'Estado' (set to 'Activo'), and 'Clave de partición' (set to 'idJuego (N)'). The 'Jugador' table has attributes: 'Nombre' (with a value of 'Jugador'), 'Estado' (set to 'Activo'), and 'Clave de partición' (set to 'idJugador (N)'). The interface includes search, filter, and pagination controls.

Figura 41: Tablas *JuegoOca* y *Jugador* creadas en DynamoDB

Desde la consola de AWS, dentro del servicio de DynamoDB, se pueden explorar los elementos guardados en las tablas, así como consultar las unidades de lectura/escritura consumidas hasta el momento.

| Elementos devueltos (2)  |              |             |                 |             |            |          |        |               |
|--------------------------|--------------|-------------|-----------------|-------------|------------|----------|--------|---------------|
|                          | C            | Acciones    | Crear ele       | <           | 1          | >        |        |               |
|                          | idJugador... | codigoColor | nombre          | nombreColor | penaliz... | posicion | puntos | ultimaCasilla |
| <input type="checkbox"/> | 1            | #0000FF     | las divinas ... | Azul        | 0          | 0        | 30     | Salida        |
| <input type="checkbox"/> | 0            | #FF0000     | los campe...    | Rojo        | 0          | 1        | 15     | Salida        |

Figura 42: Información de la tabla *Jugador* de DynamoDB

### 7.3. Implementación y estructura del código

La estructura de archivos del código de la función de Lambda que ejecuta la skill del juego es la siguiente:

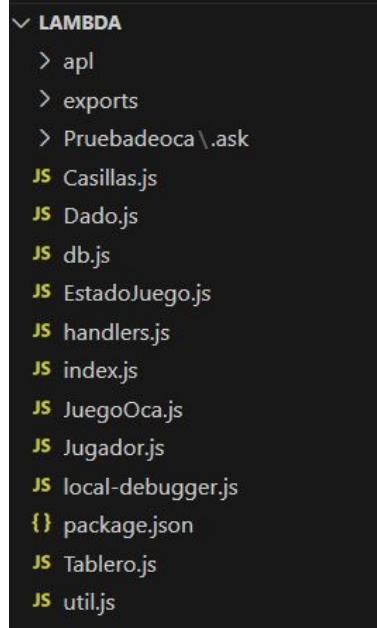


Figura 43: Estructura del código de la función Lambda

A continuación se explicarán los contenidos de los ficheros, salvo: *local-debugger.js*, *package.json* y *util.js*, que ya fueron descritos en la sección 7.1.

### 7.3.1. Lógica del juego de la oca

Consiste en la implementación de las clases definidas en el modelo conceptual de la sección 5.3, que cubren las funcionalidades del juego de la oca.

**Fichero «Jugador.js»** Define la clase encargada de representar a cada participante, ya sea un equipo o jugador/a individual, que va a competir en el juego de la oca.

```
/**  
 * Representa a un participante (equipo o individuo) en el juego.  
 * @class  
 */  
class Jugador {  
    /**  
     * Crea una instancia de Jugador (participante que puede ser un equipo o individuo).  
     * @param {number} id - Identificador único del participante.  
     * @param {string} color - Nombre del color asignado al participante.  
     * @param {string} codigo - Código hexadecimal del color asociado al participante.  
     * @param {number} posicion - Posición actual del participante en el tablero.  
     * @param {number} puntos - Puntos acumulados por el participante.  
     * @param {string} nombre - Nombre del participante.  
     * @param {number} ultimaCasilla - Última casilla en la que estuvo el participante.  
     */  
    constructor(id, color, codigo, posicion, puntos, nombre, ultimaCasilla){  
        this.id = id;  
        this.color = color;  
        this.codigo = codigo;  
        this.posicion = posicion;  
        this.puntos = puntos;  
        this.nombre = nombre;  
        this.ultimaCasilla = ultimaCasilla;  
    }  
}
```

Figura 44: Clase jugador junto con su constructor

También incluye *setters* y *getters* para poder modificar y consultar cada atributo de clase.

**Fichero «Dado.js»** Está constituido por dos funciones auxiliares: la primera, que realiza un lanzamiento de dado, devolviendo un número aleatorio entre 1 y 6, y la segunda, para obtener la dirección URL del vídeo asociado a cada resultado posible de una tirada. Esto será especialmente útil cuando se vaya a implementar el APL más adelante.

```

JS Dado.js > ...
1  /**
2   * Simula el lanzamiento de un dado de seis caras.
3   * @returns {number} Un entero aleatorio entre 1 y 6, representando el resultado de la tirada.
4   */
5  function tirarDado() {
6      return Math.floor(Math.random() * 6) + 1;
7  }
8
9 /**
10 * Obtiene la URL del video correspondiente a la tirada de dado, que se almacena en Amazon S3.
11 * @param {number} n El número obtenido al tirar el dado, que debe estar entre 1 y 6.
12 * @returns {string} La URL del video que muestra la animación del lanzamiento de dado.
13 * @description Devuelve la URL de un video específico basado en el número del dado. Cada resultado
14 * tiene una animación asociada.
15 */
16 function getUrlDado(n) {
17     let url;
18     switch(n) {
19         case 1:
20             url = "https://bucket-oca.s3.eu-west-1.amazonaws.com/videos-dado/dado-1.mp4";
21             break;
22         case 2:
23             url = "https://bucket-oca.s3.eu-west-1.amazonaws.com/videos-dado/dado-2.mp4";
24             break;
25         case 3:
26             url = "https://bucket-oca.s3.eu-west-1.amazonaws.com/videos-dado/dado-3.mp4";
27             break;
28         case 4:
29             url = "https://bucket-oca.s3.eu-west-1.amazonaws.com/videos-dado/dado-4.mp4";
30             break;
31         case 5:
32             url = "https://bucket-oca.s3.eu-west-1.amazonaws.com/videos-dado/dado-5.mp4";
33             break;
34         case 6:
35             url = "https://bucket-oca.s3.eu-west-1.amazonaws.com/videos-dado/dado-6.mp4";
36             break;
37     }
38     return url;
39 }
40

```

Figura 45: Funciones que componen el módulo del dado

**Fichero «Casillas.js»** En las primeras líneas se realizan las importaciones necesarias de los datos de las preguntas que se usarán para los minijuegos, así como las frases de carácter descriptivo que se invocarán cuando se caiga en cualquier casilla.

```

JS Casillas.js > ...
1  const preguntasVF = require('../exports/preguntasVF.json');
2  const preguntasCifras = require('../exports/preguntasCifras.json');
3  const preguntasFechas = require('../exports/preguntasFechas.json');
4  const preguntasCompas = require('../exports/preguntasCompas.json');
5

```

Figura 46: Importaciones de las baterías de preguntas de los minijuegos

Luego, se define la clase que representa una casilla general (o de carácter normal); es decir,

aquella que no desencadena ningún evento especial como desplazarse a otra casilla, tirar el dado de nuevo, iniciar un minijuego, etc.

```

/**
 * Representa una casilla general en el tablero de juego.
 * @class
 */
class Casilla {
    /**
     * Crea una nueva instancia de casilla.
     * @param {string} id - Identificador único de la casilla y su nombre.
     * @param {string} url - URL de la imagen asociada a la casilla.
     * @param {Object} frases - Frases asociadas a la casilla, diferenciadas por si el juego es en equipo o individual.
     */
    constructor(id, url, frases) {
        this.id = id;
        this.url = url;
        this.frases = frases;
    }

    /**
     * Recibe a un participante (equipo o jugador) en la casilla.
     * @param {Jugador} jug - El jugador que ha caído en esta casilla.
     * @param {boolean} hayEquipos - Indica si el juego se está jugando por equipos o no.
     * @returns {string} Descripción del evento que ocurre cuando el participante cae en la casilla.
     */
    recibirJugador(jug, hayEquipos) {
        if (this.id === "META") {
            return `¡Buen trabajo! ${hayEquipos ? 'Habéis ' : 'Has '} caído en la casilla de meta.`;
        } else {
            return `${hayEquipos ? 'Habéis ' : 'Has '} caído en la casilla ${this.id}.` + this.getFraseRandom(hayEquipos);
        }
    }

    /**
     * Devuelve una frase aleatoria de la casilla, adaptada al tipo de participante.
     * @param {boolean} hayEquipos - Indica si el juego se está jugando por equipos o no.
     * @returns {string} Una frase aleatoria adecuada al contexto del juego.
     */
    getFraseRandom(hayEquipos) {
        const desc = hayEquipos ? this.frases.equipo : this.frases.jugador;
        const idx = Math.floor(Math.random() * 3);

        return desc[idx];
    }
}

```

Figura 47: Declaración de la clase de una casilla normal

Se puede ver cómo hay dos métodos: `getFraseRandom()`, encargado de devolver una descripción aleatoria de la casilla normal, habiendo tres posibles opciones para cada una, y `recibirJugador()`, que informa al participante sobre dónde ha caído, incluyendo la descripción de la misma. Este último método será sobrescrito en las distintas clases que heredan de `Casilla`, para adaptar los informes en función del tipo de la instancia. Por ejemplo, si es una casilla de oca, dirá «de oca en oca y tiro porque me toca», si es una de minijuego, anunciará qué tipo de pregunta hará y cómo jugar, etc.

**Casillas especiales tradicionales: oca, puente y penalización** Siguiendo el esquema del juego de mesa tradicional, las casillas de oca permiten moverse a la siguiente del mismo y tirar el

dado de nuevo; las casillas de puente, transportan a la otra casilla del mismo tipo independientemente de si dicho movimiento supone un avance o retroceso en el tablero; y las casillas de penalización, como la del pozo y el laberinto, hacen perder turnos a cualquiera que caiga en ellas.

Las casillas de oca y de puente mantienen una estructura similar a su superclase, con los mismos atributos y la única diferencia en el método sobrescrito de *recibeJugador()*; sin embargo, la clase *CasillaPenalizacion* incluye un nuevo atributo (*penaliza*) que representa el número de turnos que deben pasar antes de poder avanzar de nuevo.

```
class CasillaPenalizacion extends Casilla {
    /**
     * Crea una casilla de tipo "Penalización".
     * @param {string} id - Identificador único de la casilla.
     * @param {string} url - URL de la imagen asociada a la casilla.
     * @param {Object} frases - Frases asociadas a la casilla.
     */
    constructor(id, url, frases, penaliza) {
        super(id, url, frases);
        this.penaliza = penaliza;
    }

    /**
     * Recibe a un participante (equipo o jugador) en la casilla de penalización.
     * @param {Jugador} jug - El jugador que ha caído en esta casilla.
     * @param {boolean} hayEquipos - Indica si el juego se está jugando por equipos o no.
     * @returns {string} Descripción del evento que ocurre cuando el participante cae en la casilla de penalización.
     */
    recibirJugador(jug, hayEquipos) {
        if (hayEquipos){
            return `Mala suerte, habéis caído en la casilla ${this.id}. Perdéis ${this.penaliza} turnos.`;
        }
        return `Mala suerte, ha caído en ${this.id}. Pierdes ${this.penaliza} turnos.`;
    }
}
```

Figura 48: Declaración de la clase de casillas de penalización

**Casillas de minijuegos personalizadas** Se caracterizan porque además de los métodos heredados de las casillas normales, cuentan con un método llamado *getPreguntaRandom()*, que devuelve un elemento aleatorio de la batería de preguntas del tipo correspondiente, con la excepción del minijuego de «recuerda la última casilla», que no necesita el método porque la pregunta siempre es la misma.

Por tanto, la clase que define la casilla del minijuego de verdadero o falso tendría la siguiente apariencia:

```

class CasillaVyF extends Casilla {
    /**
     * Crea una casilla del minijuego "verdadero o falso".
     * @param {string} id - Identificador único de la casilla.
     * @param {string} url - URL de la imagen asociada a la casilla.
     * @param {Object} frases - Frases asociadas a la casilla.
     */
    constructor(id, url, frases) {
        super(id, url, frases);
    }

    /**
     * Devuelve una pregunta aleatoria de la batería de preguntas del minijuego "verdadero o falso".
     * @returns {string} Una pregunta aleatoria adecuada al tipo de minijuego.
     */
    getPreguntaRandom() {
        const ind = Math.floor(Math.random() * preguntasVF.length);
        return preguntasVF[ind]
    }
}

```

Figura 49: Clase para casilla del minijuego verdadero o falso

En el minijuego de «conoce a tus compañeros» también se necesita guardar el nombre del equipo/participante sobre el que va la pregunta, así que se hace una ligera modificación, pasándole como argumento un arreglo con todos los compañeros del jugador o jugadora actual.

```

/**
 * Devuelve una pregunta aleatoria de la batería de preguntas del minijuego "conoce a tus compañeros",
 * junto con el compañero sobre el que irá dicha pregunta.
 * @param {Array} compas - El conjunto de los compañeros de juego, salvo el participante encargado
 * de responder a la pregunta.
 * @returns {Array} Un array con la pregunta y compañero aleatorios.
 */
getPreguntaRandom(compas) {
    const indP = Math.floor(Math.random() * preguntasCompas.length);
    const indC = Math.floor(Math.random() * compas.length);

    return [preguntasCompas[indP], compas[indC]];
}

```

Figura 50: Método para obtener una pregunta aleatoria sobre un compañero

Las clase para la casilla del minijuego «adivina la cifra» es parecida a la de la figura 44. Por otro lado, como en el minijuego de «recuerda la fecha» se manejan tres tipos distintos de preguntas y respuestas (día de la semana, mes o estación del año), y la solución correcta se calcula a partir de la fecha actual, o lo que es lo mismo, no es un valor estático, entonces requiere añadir una serie de métodos adicionales:

```

    /**
     * Calcula y devuelve la solución a una pregunta basada en su tipo e identificador.
     * @param {Object} pregunta - Objeto que contiene el tipo de pregunta y un identificador numérico.
     * @param {string} pregunta.type - El tipo de pregunta ('diaSemana', 'mes', 'estacion').
     * @param {number} pregunta.id - Identificador numérico para calcular la solución en función de la fecha actual.
     * @returns {string} La solución calculada o un mensaje de error si la pregunta no es válida.
     * @description Este método calcula la solución a la pregunta del minijuego recuerda la fecha. Por ejemplo,
     *             si la pregunta es "¿Qué día de la semana fue ayer?", su tipo es 'diaSemana' y su ID es -1,
     *             ya que ID 0 sería el día de la semana actual. El método devolvería el nombre del día de ayer.
     */
    getSolucion(pregunta) {
        const type = pregunta.type;
        const id = pregunta.id;
        const hoy = new Date();

        switch (type) {
            case 'diaSemana':
                return this.getDiaSemana(hoy, id);
            case 'mes':
                return this.getMes(hoy, id);
            case 'estacion':
                return this.getEstacion(hoy, id);
            default:
                return 'Tipo de pregunta no reconocido.';
        }
    }
}

```

Figura 51: Método para obtener la solución de una pregunta de fecha

En la figura anterior, aparecen invocaciones a tres métodos auxiliares que permiten el cálculo correcto de la respuesta pasándole como argumentos el valor de la fecha actual y un id o número de desplazamiento partiendo del actual. Si el id es igual a 0, entonces la solución es el día/mes/estación actual; si el id es 1, la respuesta es el día/mes/estación siguiente al actual; si el id es -1, el anterior al actual... Y así sucesivamente.

```

    /**
     * Calcula y devuelve el día de la semana basado en un desplazamiento desde el actual.
     * @param {Date} fecha - La fecha de referencia actual para el cálculo. Desde 0 (domingo) hasta 6 (sábado).
     * @param {number} id - Desplazamiento de días de la semana a partir del actual.
     * @returns {string} El nombre del día de la semana calculado.
     */
    getDiaSemana(fecha, id) {
        const diasSemana = ['domingo', 'lunes', 'martes', 'miércoles', 'jueves', 'viernes', 'sábado'];
        const diaActual = fecha.getDay(); // 0 (domingo) - 6 (sábado)
        const diaCalculado = (diaActual + id + 7) % 7;
        return diasSemana[diaCalculado];
    }

    /**
     * Calcula y devuelve el mes del año basado en un desplazamiento desde el actual.
     * @param {Date} fecha - La fecha de referencia actual para el cálculo. Desde 0 (enero) hasta 11 (diciembre).
     * @param {number} id - Desplazamiento de meses del año a partir del actual.
     * @returns {string} El nombre del mes calculado.
     */
    getMes(fecha, id) {
        const meses = ['enero', 'febrero', 'marzo', 'abril', 'mayo', 'junio', 'julio', 'agosto', 'septiembre', 'octubre', 'noviembre', 'diciembre'];
        const mesActual = fecha.getMonth(); // 0 (enero) - 11 (diciembre)
        const mesCalculado = (mesActual + id + 12) % 12;
        return meses[mesCalculado];
    }
}

```

Figura 52: Cálculo de días de la semana o meses a partir de la fecha actual

```

/**
 * Calcula y devuelve la estación del año basada en un desplazamiento desde la actual.
 * @param {Date} fecha - La fecha de referencia para el cálculo, utilizando el mes actual.
 * @param {number} id - Desplazamiento de meses del año a partir del actual.
 * @returns {string} El nombre del mes calculado.
 */
getEstacion(fecha, id) {
    const estaciones = ['invierno', 'primavera', 'verano', 'otoño'];
    const mesActual = fecha.getMonth();

    let estacionActual;
    if (mesActual >= 2 && mesActual <= 4) {
        estacionActual = 1;
    } else if (mesActual >= 5 && mesActual <= 7) {
        estacionActual = 2;
    } else if (mesActual >= 8 && mesActual <= 10) {
        estacionActual = 3;
    } else {
        estacionActual = 0;
    }

    const estacionCalculada = (estacionActual + id + 4) % 4;
    return estaciones[estacionCalculada];
}

```

Figura 53: Cálculo de la estación del año partiendo de la fecha actual

**Fichero «Tablero.js»** Es la estructura que almacena todas las casillas del juego en un arreglo con el mismo nombre, y hace operaciones básicas de gestión como añadir nuevas casillas u obtener una concreta a partir de un índice.

```

/**
 * Representa el tablero de juego, gestionando las casillas y la lógica de movimiento.
 * @class
 */
class Tablero {
    /**
     * Crea un tablero del juego de la oca, inicializando con la casilla de salida 0.
     */
    constructor() {
        this.tablero = [];
        this.addCasilla(new Casilla("Salida"));
    }

    /**
     * Añade una nueva casilla al tablero.
     * @param {Casilla} casilla - La casilla a añadir al tablero.
     */
    addCasilla(casilla) {
        this.tablero.push(casilla);
    }

    /**
     * Obtiene una casilla del tablero según el índice pasado.
     * @param {number} numCasilla - El índice de la casilla a obtener.
     * @returns {Casilla|null} La casilla en dicho índice o null si el índice es incorrecto.
     */
    getCasilla(numCasilla) {
        if (this.correcto(numCasilla)) {
            return this.tablero[numCasilla];
        } else {
            return null;
        }
    }
}

```

Figura 54: Declaración de la clase Tablero

Además, es la encargada de calcular una nueva posición a partir del índice de la casilla actual y el número de desplazamiento, asegurándose de que si es mayor que el número de casillas restantes hasta la meta, pueda retroceder y no salirse del límite.

Otra de sus funcionalidades es la de buscar la siguiente casilla de tipo oca, dada la posición de la actual, útil para manejar el evento cuando se cae en una. También cuenta con un método para obtener la otra casilla de puente.

```

    /**
     * Calcula una nueva posición en el tablero a partir de la actual y una tirada de dado.
     * @param {number} actual - La posición actual en el tablero.
     * @param {number} tirada - El número obtenido en la tirada de dado.
     * @returns {number} La nueva posición en el tablero después de avanzar.
     */
    nuevaPosicion(actual, tirada) {
        let nueva_pos = actual + tirada;
        if (nueva_pos >= this.tablero.length) {
            let restantes = this.tablero.length - actual;
            let retrocede = tirada - restantes;
            nueva_pos = (this.tablero.length - 1) - retrocede ;
        }
        return nueva_pos;
    }

    /**
     * Busca la siguiente casilla de tipo 'Oca' tras la posición actual.
     * @param {number} posActual - La posición actual en el tablero.
     * @returns {number|null} El índice de la siguiente casilla de tipo 'Oca' o null si no la encuentra.
     */
    buscarSiguienteOca(posActual) {
        for (let i = posActual + 1; i < this.tablero.length; i++) {
            if (this.tablero[i] instanceof CasillaOca) {
                return i;
            } else if (i === (this.tablero.length-1)) {
                return (this.tablero.length-1);
            }
        }
        return null;
    }
}

```

Figura 55: Métodos para calcular una posición y buscar un tipo de casilla

**Fichero «EstadoJuego.js»** Aquí va el contenido del párrafo.

```

js EstadoJuego.js > ...
1  /**
2   * Emula una estructura de datos enum para los posibles estados del juego.
3   * @readonly
4   * @enum {string}
5   */
6  const EstadoJuego = {
7      INDETERMINADO: 'INDETERMINADO',
8      CONFIGURANDO: 'CONFIGURANDO',
9      REGISTRO_NOMBRES: 'REGISTRO_NOMBRES',
10     TIRAR_DADO: 'TIRAR_DADO',
11     MOVER_FICHA: 'MOVER_FICHA',
12     MINIJUEGO_VF: 'MINIJUEGO_VF',
13     MINIJUEGO_CIFRAS: 'MINIJUEGO_CIFRAS',
14     MINIJUEGO_COMPAS: 'MINIJUEGO_COMPAS',
15     MINIJUEGO_FECHAS: 'MINIJUEGO_FECHAS',
16     MINIJUEGO_CASILLA: 'MINIJUEGO_CASILLA',
17     FINALIZADO: 'FINALIZADO'
18 };
19

```

Figura 56: Simula la estructura de datos *enum* de los estados del juego

```

/**
 * Devuelve un mensaje de ayuda basado en el estado actual del juego.
 * @param {EstadoJuego} estado El estado actual del juego.
 * @param {boolean} [hayEquipos=false] Indica si los participantes van por equipos o no.
 * @param {string} [nombreJ=''] El nombre del jugador o equipo actual.
 * @param {string} [pregunta=''] Pregunta actual en caso de minijuegos que la requieran.
 * @returns {string} Mensaje que indica el estado del juego y qué hacer a continuación.
 */
function informeEstado(estado, hayEquipos = false, nombreJ = '', pregunta = '') {
    let mensaje;
    switch (estado) {
        case EstadoJuego.CONFIGURANDO:
            mensaje = `Se está creando una nueva partida. Registre los datos que se le indiquen.`;
            break;
        case EstadoJuego.REGISTER_NOMBRES:
            mensaje = `Se está registrando el nombre de los ${hayEquipos ? 'equipos' : 'participantes'}. Cada uno debe decir: ${hayEquipos ? 'Nuestro' : 'Mi'} nombre es ${hayEquipos ? 'Nuestro' : 'Mi'} nombre es ${hayEquipos ? 'Amantes' : 'Locos'}`;
            break;
        case EstadoJuego.TIRAR_DADO:
            mensaje = `Es el turno de ${nombreJ}. Por favor, ${hayEquipos ? 'decid' : 'di'}`;
            break;
        case EstadoJuego.MOVER_FICHA:
            mensaje = `Es el turno de ${nombreJ}. Por favor, ${hayEquipos ? 'decid' : 'di'}`;
            break;
    }
}

```

Figura 57: Función que devuelve un mensaje de ayuda según el estado del juego

**Fichero «JuegoOca.js»** Es la clase más extensa de todas, haciendo uso del resto de clases definidas anteriormente, lo que puede observarse en los módulos que han tenido que importarse.

```

const {EstadoJuego} = require('./EstadoJuego.js');
const {Tablero} = require('./Tablero.js');
const {Jugador} = require('./Jugador.js');
const {Casilla, CasillaOca, CasillaPuente, CasillaPenalizacion, CasillaCompas,
       CasillaVyF, CasillaCifras, CasillaUltima, CasillaFechas} = require('./Casillas.js');
const fc = require('./exports/frasesCasillas.json');

/**
 * Clase que representa un juego de la Oca.
 */
class JuegoOca {
    /**
     * Crea una nueva instancia del juego de la Oca.
     */
    constructor() {
        this.hayEquipos = false;
        this.jugadores = [];
        this.numJugadores = 0;
        this.turno = 0;
        this.penalizaciones = [];
        this.tablero = this.crearTableroPrueba();
        this.ronda = 1;
        this.estado = EstadoJuego.INDETERMINADO;
    }
}

```

Figura 58: Importaciones y declaración de la clase del juego de la oca

Esta clase dispone de varios de métodos para implementar la lógica del juego:

- **crearJugadores()**: crea un número específico de jugadores (hasta 5), asignándoles colores y códigos únicos (salvo los nombres, que serán añadidos más tarde), inicializándolos con los valores de partida predeterminados, y actualizando el estado de juego.

```
/**
 * Crea los participantes (equipos o jugadores) del juego.
 * @param {number} n - El número de participantes a crear (máximo 5).
 * @returns {boolean} - Devuelve true si los participantes fueron creados con éxito, false en caso contrario.
 */
crearJugadores(n) {
    let ok = false;
    if (n < 6 && this.estado === EstadoJuego.CONFIGURANDO) {
        this.jugadores = [];
        const colores = ['Rojo', 'Azul', 'Verde', 'Morado', 'Naranja', 'Gris'];
        const codigos = ['#FF0000', '#0000FF', '#008000', '#AA00FD', '#FD7D00', '#8C8C8C'];

        for (let i = 0; i < n; i++) {
            const color = colores[i];
            const cod = codigos[i];
            this.jugadores.push(new Jugador(i, color, cod, 0, 0, '', 'Salida'));
            this.penalizaciones.push(0);
        }

        this.numJugadores = n;
        ok = true;
    }
    return ok;
}
```

Figura 59: Método de inicialización de participantes del juego

- **crearTablero()**: genera un tablero completo con 63 casillas, con diferentes tipos como CasillaOca, CasillaPenalizacion, CasillaVyF, entre otras, para un juego completo de la oca.

```
/**
 * Crea el tablero completo del juego (63 casillas sin contar la de salida, que es la 0).
 * @returns {Tablero} - El tablero definitivo del juego.
 */
crearTablero() {
    let tablero = new Tablero()

    tablero.addCasilla(new Casilla("trompo", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/1.jpg", fc[2]));
    tablero.addCasilla(new CasillaVyF("minijuego verdadero o falso", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/2.jpg"));
    tablero.addCasilla(new Casilla("dinero", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/3.jpg", fc[1]));
    tablero.addCasilla(new CasillaCompas("minijuego conoce a tus compañeros", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/4.jpg"));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/5.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/6.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/7.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/8.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/9.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/10.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/11.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/12.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/13.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/14.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/15.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/16.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/17.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/18.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/19.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/20.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/21.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/22.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/23.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/24.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/25.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/26.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/27.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/28.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/29.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/30.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/31.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/32.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/33.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/34.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/35.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/36.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/37.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/38.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/39.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/40.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/41.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/42.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/43.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/44.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/45.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/46.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/47.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/48.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/49.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/50.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/51.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/52.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/53.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/54.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/55.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/56.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/57.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/58.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/59.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/60.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/61.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/62.jpg", fc[3]));
    tablero.addCasilla(new Casilla("casa", "https://bucket-oca.s3.eu-west-1.amazonaws.com/casillas-oca/63.jpg", fc[3]));
}
```

Figura 60: Método de inicialización del tablero de la oca

- **pasarTurno()**: avanza el turno, recalcula el número de ronda, actualiza el estado del juego y devuelve un mensaje anunciando el turno del siguiente participante. comprobando antes si tiene penalizaciones de turno o no.

```

    /**
     * Avanza el turno al siguiente participante, incrementando el número de ronda si es preciso.
     * @returns {string} - Un mensaje anunciando el turno del siguiente participante.
     */
    pasarTurno() {
        let sig = this.turno + 1;
        if (sig >= this.numJugadores) {
            sig = 0;
            this.ronda++;
        }
        this.turno = sig;
        this.setEstado(EstadoJuego.TIRAR_DADO);

        return this.anunciarTurno();
    }

```

Figura 61: Método para pasar al siguiente turno

- **anunciarGanadores()**: si el juego ha finalizado, devuelve un mensaje anunciando las dos modalidades de ganadores: quien haya llegado antes a la meta y quien haya obtenido más puntos gracias a los minijuegos.

```

    /**
     * Devuelve un mensaje anunciando los ganadores de la partida, en caso de que haya finalizado.
     * @returns {string} - Un mensaje anunciando los ganadores o que la partida sigue en curso.
     */
    anunciarGanadores() {
        let resultados;

        if (this.estado === EstadoJuego.FINALIZADO) {
            const jugEnMeta = this.getJugadorActual();
            resultados = `Enhorabuena ${this.hayEquipos ? 'al equipo' : 'al participante'} ${jugEnMeta.getNombre()}, que ha ganado la \
                partida al llegar a la meta en primer lugar.`;

            const jugMayorPts = this.jugadores.reduce((max, actual) =>
                actual.getPuntos() > max.getPuntos() ? actual : max, this.jugadores[0]
            );

            if (jugMayorPts === jugEnMeta) {
                resultados += `Además, ${this.hayEquipos ? 'habéis' : 'has'} acumulado más puntos que nadie, \
                    ${jugMayorPts.getPuntos()} puntos en total, por tanto también ${this.hayEquipos ? 'conseguís' : 'consigues'} \
                    la Medalla de Minijuegos. ¡Felicitaciones!`;
            } else {
                resultados += `Honorable mención ${this.hayEquipos ? 'al equipo' : 'al participante'} ${jugMayorPts.getNombre()}, \
                    que con ${jugMayorPts.getPuntos()} puntos ha logrado la mayor cantidad, por tanto se lleva la \
                    Medalla de Minijuegos. ¡Felicitaciones!`;
            }
        } else {
            resultados = 'La partida todavía sigue en curso, nadie ha llegado a la meta aún.';
        }
        return resultados;
    }

```

Figura 62: Método para anunciar los ganadores al final del juego

- **avanzarJugador()**: mueve al participante actual una cantidad determinada de casillas, actualizando su posición y manejando eventos específicos basados en el tipo de casilla en la

que cae. También genera un informe sobre el movimiento realizado y actualiza el estado de la partida.

```
/**  
 * Avanza al participante actual una cantidad de casillas concreta, actualizando su posición y manejando los  
 * eventos en función de la casilla en la que caerá.  
 * @param {Jugador} jActual - El participante actual.  
 * @param {number} tirada - El número de posiciones que debe moverse.  
 * @returns {[Casilla, string]|null} - Un array con la nueva casilla y un informe del movimiento, o null si  
 * no es el momento de mover la ficha.  
 */  
avanzaJugador(jActual, tirada){  
    if (this.estado === EstadoJuego.MOVER_FICHA) {  
        let posNueva = 0;  
        let posActual = jActual.getPosActual();  
        let casillaNueva;  
        let informe = `${this.hayEquipos ? 'El equipo' : ''} ${jActual.nombre} estaba en la casilla ${posActual}. `;  
        jActual.setUltimaCasilla(this.tablero.getCasilla(posActual).getId());  
    }  
}
```

Figura 63: Primeras líneas del método para avanzar las fichas

- Métodos *getters* y *setters* de todos los atributos de la clase.
- Métodos adicionales para calcular y hallar: un participante de índice concreto, el equipo o jugador/a del turno actual, sus compañeros de juego, los que se encuentran en una casilla determinada, las penalizaciones de alguno en particular, etc.

### 7.3.2. Fichero «handlers.js»

Con diferencia el fichero más extenso, con unas 900 líneas de código

```
js handlers.js > ...  
1  const Alexa = require('ask-sdk-core');  
2  const AWS = require('aws-sdk');  
3  const dynamoDB = new AWS.DynamoDB.DocumentClient();  
4  const bienvenida = require('./apl/bienvenida.json');  
5  const fichas = require('./apl/fichas.json');  
6  const {JuegoOca} = require('./JuegoOca.js');  
7  const {Jugador} = require('./Jugador.js')  
8  const {tirarDado, getUrlDado} = require('./Dado.js');  
9  const {reglasInfo, casillasInfo, minijuegosInfo, comandosInfo} = require('./exports/frasesAyuda.js');  
10 const {EstadoJuego, informeEstado} = require('./EstadoJuego.js');  
11 const {asumirRol, guardarPartida, cargarPartida} = require('./db.js');  
12  
13 let oca = new JuegoOca();
```

Figura 64: Importaciones de bibliotecas y módulos para los handlers

De la siguiente lista de handlers definidos, se van a explicar todos los que han sido personalizados de alguna forma, exceptuando aquellos mencionados en la sección 7.1. que no hayan sufrido ningún cambio.

```
module.exports = {
    LaunchRequestHandler,
    configuracion1Handler,
    guardarPartidaHandler,
    cargarPartidaHandler,
    ayudaReglasHandler,
    nuevaPartidaHandler,
    addJugadorHandler,
    jugarTurnoHandler,
    preguntasVyFHandler,
    preguntasCifrasHandler,
    preguntasCasillaHandler,
    preguntasCompasHandler,
    preguntasFechasHandler,
    HelpIntentHandler,
    CancelAndStopIntentHandler,
    FallbackIntentHandler,
    SessionEndedRequestHandler,
    IntentReflectorHandler,
    ErrorHandler
};
```

Figura 65: Todos los handlers que se exportan al *index.js*

Cabe mencionar que cada handler está vinculado a un intent con el mismo nombre, que es invocado mediante utterances. Por ejemplo: con la utterance «Nueva partida», se activa la intención *nuevaPartidaIntent*, que provocará la ejecución del manejador *nuevaPartidaHandler*, devolviendo una respuesta procesada a la persona usuaria.

**LaunchRequestHandler** Este handler se activa cuando el usuario abre la skill por primera vez, mediante el comando «Alexa, abre probando oca». Su función principal es dar una bienvenida al usuario y proporcionar opciones iniciales sobre cómo comenzar una partida. Informa al usuario sobre la posibilidad de continuar una partida previamente guardada o iniciar una nueva, advirtiendo que al comenzar una nueva se sobrescribirá la partida anterior.

Si el dispositivo donde se ejecuta dispone de pantalla, se muestra la pantalla de inicio/bienvenida. Finalmente, el handler prepara una respuesta para que el usuario se sienta guiado sobre los próximos pasos, manteniendo la sesión activa para una interacción más fluida.

***configuracion1Handler*** Se invoca cuando el usuario solicita un intent específico para configurar una partida de prueba, útil para el proceso de depuración de errores y testeo de la skill. Este handler omite el proceso normal de configuración del juego y establece un estado predeterminado con dos equipos ficticios con nombres y puntos predefinidos para facilitar las pruebas.

A continuación, notifica al usuario que la partida ha comenzado y le recuerda los comandos necesarios para jugar. Además, muestra información visual sobre los jugadores utilizando directivas APL.

***guardarPartidaHandler* y *cargarPartidaHandler*** El primero responde a la solicitud de guardar el progreso actual del juego. A través de un rol temporal en AWS, accede a DynamoDB y almacena los datos de la partida en curso. Una vez que la partida se guarda correctamente, confirma el éxito o fracaso de la operación.

El segundo realiza el proceso inverso, pues se activa cuando el usuario desea cargar una partida previamente guardada. Vuelve a asumir un rol IAM temporal para conectarse con DynamoDB y recuperar los datos de la partida guardada. Así, permite continuar jugando desde el último punto guardado.

***ayudasReglasHandler*** Invocado cuando la persona usuaria solicita ayuda, mediante el comando «Explícame <tema>».

Dependiendo del tema específico sobre el que se quiere recibir información (como reglas generales, casillas del tablero, minijuegos o comandos), el handler proporciona una explicación detallada. Como en el resto de handlers, mantiene la sesión activa para ofrecer asistencia continua y mejorar la experiencia del usuario.

***nuevaPartidaHandler*** Se encarga de crear una nueva partida cuando el usuario lo solicita. Primero, pregunta por el tipo de participación (individual o en equipos) y luego por el número de participantes desde los slots del intent.

Una vez se han recibido los datos, y confirmado que son correctos, se actualiza el estado del juego y se proporciona una explicación sobre cómo proceder al registro los nombres, indicando el formato correcto y sugiriendo ejemplos.

A partir de este punto, se tiene en cuenta el tipo de participación para ajustar todas las respuestas con menciones a los equipos o jugadores individuales.

***addJugadorHandler*** Es el paso obligatorio que sigue al anterior handler, este gestiona el registro de nombres de los jugadores en el juego. Se activa cuando cada participante proporciona su nombre a través del comando «Mi/Nuestro nombre es <nOMBRE>».

Primero, verifica si el estado del juego está en la fase de registro de nombres y después, guarda el nombre. Si aún faltan participantes por registrar, solicita el siguiente nombre, y una vez que haya recibido todos, hace un resumen de los jugadores o equipos y guarda los datos de partida configurados en DynamoDB. También muestra en pantalla mediante APL los nombres y colores de los participantes recién registrados.

Por último, informa al usuario que la partida está lista para comenzar y guía sobre los próximos pasos: el lanzamiento de dado y movimiento de ficha.

**jugarTurnoHandler** Procesa el turno del participante actual, dividido en dos fases: tirar el dado y mover la ficha. Primero, revisa si el dado ha sido lanzado (el valor del dado de atributo de sesión no es nulo) y en caso contrario, lo lanza y guarda el resultado.

Si el dado ya fue lanzado, realiza el movimiento del jugador en el tablero, avanzando su ficha tantas casillas indicadas el dado y describiendo la casilla en la que ha caído. Dependiendo del tipo que sea, presenta instrucciones específicas sobre cómo proceder.

Por no mencionar que es el encargado de actualizar la interfaz de usuario con información visual sobre la casilla actual y quién está en ella. Una vez gestionados los eventos de la nueva casilla, se actualiza el turno actual junto con el estado del juego.

**Handlers de minijuegos** Los cinco handlers que gestionan las preguntas de cada tipo de minijuego se rigen por un esquema general similar.

1. Se comprueba que el estado del juego se corresponde con el del minijuego invocado, para evitar acceder a estos handlers fuera de turno. Si el estado no es el esperado, lanza un mensaje de error y recordatorio de cómo se debe proceder.
2. Se obtiene la respuesta del participante a través del sistema de slots del intent, que permiten capturar tipos de datos predefinidos.
3. Se recuperan los datos de la pregunta si es necesario, mediante atributos de sesión, en caso de que Alexa tenga que formularla de nuevo.
4. Se procede con la verificación de la respuesta.

Es en el paso 4 de la secuencia anterior donde la implementación difiere según el tipo de minijuego del que se trata:

- **preguntasVyFHandler:** compara la respuesta del participante actual (verdadero o falso) con la solución almacenada en forma de booleano. Si coinciden, se suman 10 puntos, si no, ofrece una breve explicación de la respuesta correcta.

- **preguntasCifrasHandler:** compara el número indicado por el participante actual con la solución en formato de entero. Si coinciden, obtiene 30 puntos, y si no, vuelve a formular la pregunta al siguiente. Así hasta que alguien acierte o se haya preguntado a todos y ninguna respuesta haya sido la correcta, en cuyo caso se revela la solución.
- **preguntasCasillaHandler:** compara la respuesta proporcionada, que es un nombre de casilla, con el ID de su última casilla almacenada. Si coinciden, suma 25 puntos, si no, Alexa revela cuál era.
- **preguntasCompasHandler:** primero se guarda la respuesta (formato: correcto/a o incorrecto/a) del participante, y después Alexa le pide al compañero sobre el que iba la pregunta que valide si la respuesta ofrecida era correcta o incorrecta. Si es correcta, ambos ganan 15 puntos.
- **preguntasFechasHandler:** compara la respuesta del participante actual (día, mes o estación), con la fecha calculada correcta. Si coincide, se suman 20 puntos.

**ErrorHandler** Se ha incluido una comprobación del estado actual del juego, para que pueda adaptar correctamente el mensaje de error a cada situación.

Si el estado indica que uno de los minijuegos está en curso, el mensaje de error incluirá información específica sobre la pregunta actual, y si no, serán instrucciones más fijas en base a dicho estado. Por ejemplo, si es el inicio del turno de algún participante, pero Alexa no recibe respuesta en un tiempo determinado o se intenta acceder a un comando no permitido, Alexa le recordará que es tiene que tirar el dado y cómo debe hacerlo.

### 7.3.3. Fichero «index.js»

Explicado previamente en la sección 7.1., se ha modificado para incluir todas las funciones manejadoras definidas hasta ahora.

```

JS index.js > ...
26     SessionEndedRequestHandler,
27     IntentReflectorHandler,
28     ErrorHandler
29 } = require('./handlers');
30
31 exports.handler = Alexa.SkillBuilders.custom()
32     .addRequestHandlers(
33         LaunchRequestHandler,
34         configuracion1Handler,
35         guardarPartidaHandler,
36         cargarPartidaHandler,
37         HelpIntentHandler,
38         ayudaReglasHandler,
39         nuevaPartidaHandler,
40         addJugadorHandler,
41         jugarTurnoHandler,
42         preguntasVyFHandler,
43         preguntasCifrasHandler,
44         preguntasCasillaHandler,
45         preguntasCompasHandler,
46         preguntasFechasHandler,
47         CancelAndStopIntentHandler,
48         FallbackIntentHandler,
49         SessionEndedRequestHandler,
50         IntentReflectorHandler)
51     .addErrorHandlers(
52         ErrorHandler)
53     .withCustomUserAgent('sample/hello-world/v1.2')
54     .lambda();
55

```

Figura 66: Exportación de manejadores a la función Lambda

#### 7.3.4. Directorio «apl»

Se tiene la siguiente estructura de ficheros JSON para la configuración de directivas de APL dentro de la respuesta de los manejadores:



Figura 67: Directorio de ficheros JSON para APL

**bienvenida.json** Es la pantalla de inicio, es decir, el aspecto que toma nada más iniciar la skill, compuesto por una imagen y un texto de bienvenida.

```
{
  "type": "APL",
  "version": "1.4",
  "import": [
    {
      "name": "alexa-layouts",
      "version": "1.7.0"
    }
  ],
  "mainTemplate": {
    "items": [
      {
        "type": "Container",
        "width": "100%",
        "height": "100%",
        "alignItems": "center",
        "justifyContent": "center",
        "items": [
          {
            "type": "AlexaBackground",
            "backgroundColor": "#FFFFFF"
          },
          {
            "type": "Image",
            "source": "https://bucket-oca.s3.eu-west-1.amazonaws.com/oca.png",
            "width": "70%",
            "height": "70%",
            "scale": "best-fit"
          },
          {
            "type": "Text",
            "text": "¡Bienvenidos/as a la oca!",
            "color": "#000000",
            "fontSize": "60dp",
            "marginTop": "15dp"
          }
        ]
      }
    ]
  }
}
```

Figura 68: Documento APL para la pantalla de inicio



## ¡Bienvenidos/as a la oca!

Figura 69: Pantalla de bienvenida de la skill

**fichas.json** Aparece tras configurar la partida que se va a jugar, con los participante ya registrados. Simplemente muestra los nombres de cada equipo o jugador/a, junto con el color que tienen asociado.

```
    {
      "type": "APL",
      "version": "1.5",
      "theme": "dark",
      "mainTemplate": {
        "parameters": [
          "payload"
        ],
        "items": [
          {
            "type": "Sequence",
            "data": "${payload.datosFichas.properties.jugadores}",
            "width": "100%",
            "height": "100%",
            "scrollDirection": "vertical",
            "items": [
              {
                "type": "Frame",
                "direction": "row",
                "backgroundColor": "${data.codigo}",
                "height": "96dp",
                "width": "100%",
                "items": [
                  {
                    "type": "Text",
                    "text": "${data.nombre}",
                    "color": "white",
                    "fontSize": "60dp"
                  }
                ]
              }
            ]
          }
        ]
      }
    }
```

Figura 70: Documento APL para mostrar los participantes del juego

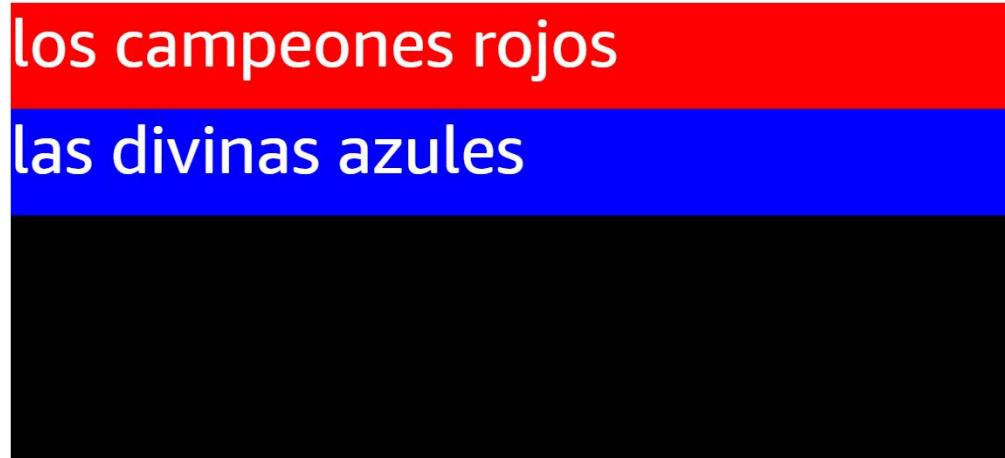


Figura 71: Pantalla postconfiguración mostrando a los participante

**dado.json** Utilizado después de cada lanzamiento de dado para reproducir la animación correspondiente al número obtenido en la tirada (hay seis vídeos en total, uno para cada resultado).

```
{
  "type": "APL",
  "version": "1.8",
  "mainTemplate": {
    "parameters": [
      "payload"
    ],
    "items": [
      {
        "type": "Video",
        "source": "${payload.datosDado.properties.num}",
        "autoplay": true,
        "audioTrack": "none",
        "width": "80%",
        "height": "80%"
      }
    ]
  }
}
```

Figura 72: Documento APL para mostrar la animación del dado

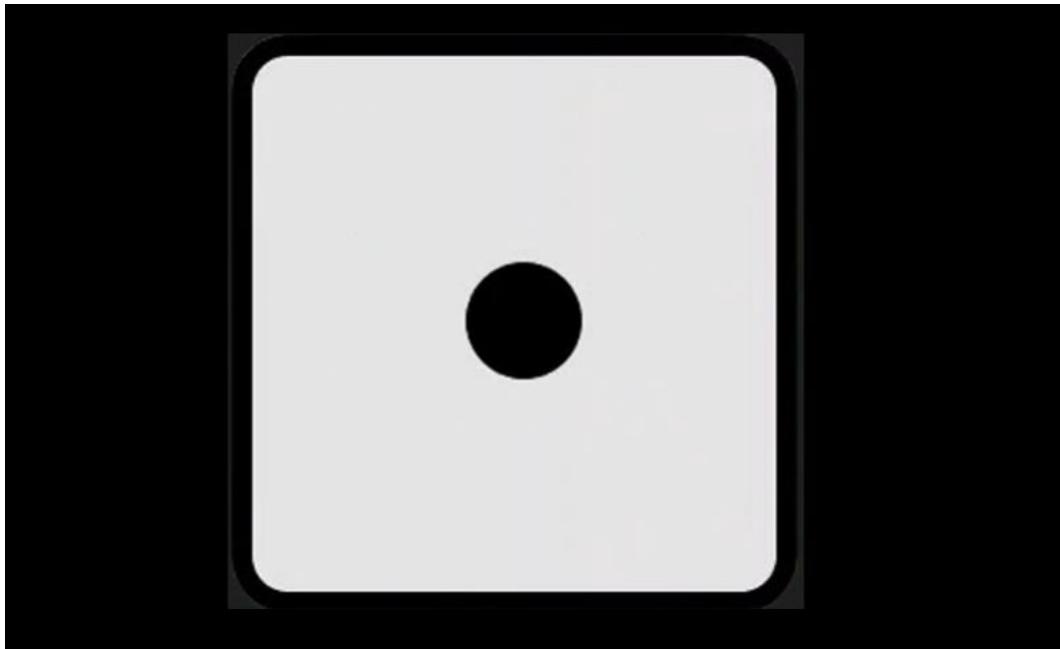


Figura 73: Pantalla mostrando un lanzamiento de dado

***casilla.json*** Muestra en la mitad izquierda los participantes que están en esa casilla, pudiendo haber más de uno en la misma, y en la mitad derecha, la imagen asociada a la casilla con su número o índice dentro del tablero.

```
{
  "type": "APL",
  "version": "1.6",
  "mainTemplate": {
    "parameters": [
      "payload"
    ],
    "items": [
      {
        "type": "Container",
        "width": "100%",
        "height": "100%",
        "items": [
          {
            "type": "Image",
            "source": "${payload.datosCasilla.properties.casillaImg}",
            "width": "50%",
            "height": "100%",
            "position": "absolute",
            "right": "1dp"
          },
          {
            "type": "Frame",
            "width": "50%",
            "height": "30%",
            "backgroundColor": "${payload.datosCasilla.properties.circleColor}",
            "alignSelf": "left",
            "alignItems": "left",
            "items": [
              {
                "type": "Text",
                "text": "${payload.datosCasilla.properties.circleId}",
                "color": "#FFFFFF",
                "fontSize": "60dp"
              }
            ]
          },
          {
            "type": "Sequence",
            "id": "fichaSequence",
            "width": "40%",
            "height": "70%",
            "scrollDirection": "vertical",
            "data": "${payload.datosCasilla.properties.fichas}"
          }
        ]
      }
    ]
  }
}
```

Figura 74: Documento APL para mostrar la casilla y participante actuales



Figura 75: Pantalla mostrando la casilla y participante actuales

### 7.3.5. Directorio «exports»

En esta carpeta se encuentran aquellos datos que van a ser importados en otros módulos, como por ejemplo las baterías de preguntas de los minijuegos o las frases aleatorias de las casillas normales.

```
✓ exports
  JS frasesAyuda.js
  {} frasesCasillas.json
  {} preguntasCifras.json
  {} preguntasCompas.json
  {} preguntasFechas.json
  {} preguntasVF.json
```

Figura 76: Directorio de ficheros exportados a otros módulos

***frasesAyuda.js*** Contiene las descripciones de los distintos temas de los que puede solicitarse una explicación a través de *ayudaReglasIntent*. Hay una variable constante en formato cadena de texto para cada tema posible: las reglas del juego, los tipos de casilla, los minijuegos y los comandos disponibles.

**frasesCasillas.json** Es donde viene almacenada la información de las casillas de tipo normal. Los datos relevantes son: el identificador (id) o nombre de la casilla, el número o índice dentro del tablero y un arreglo de frases posibles para recibir a cada modalidad de participante (individual o equipo).

```
{  
    "id": "lentejas",  
    "num": 3,  
    "jugador": [  
        "Te has encontrado con un saco de lentejas, ¡a cocinar!",  
        "Un plato de lentejas, quien quiera las come y quien no las deja.",  
        "Un montón de lentejas para tí, ¡saludable y delicioso!"  
    ],  
    "equipo": [  
        "Os habéis encontrado con un saco de lentejas, ¡a cocinar!",  
        "Un plato de lentejas, quien quiera las come y quien no las deja.",  
        "Un montón de lentejas para vosotros, ¡saludable y delicioso!"  
    ]  
},
```

Figura 77: Ejemplo de objeto con la información de una casilla normal

**preguntasCifras.json** El conjunto de preguntas posibles para el minijuego de adivinar la cifra, compuestas por el texto de la pregunta en sí y su solución, que siempre es un número entero.

```
{  
    "question": "¿En qué año Cristóbal descubre América?",  
    "answer": "1492"  
},  
{  
    "question": "¿Cuántos días tiene un año bisiesto?",  
    "answer": "366"  
},  
{  
    "question": "¿Cuántos huevos hay en media docena?",  
    "answer": "6"  
},
```

Figura 78: Ejemplos de preguntas en «Adivina la cifra»

**preguntasVF.json** La batería de preguntas de tipo trivial verdadero o falso, compuestas por el texto de la pregunta en, su solución (booleano) y una explicación de la respuesta que se mencionará si se falla.

```
{
  "question": "Hay cinco grupos sanguíneos diferentes.",
  "answer": false,
  "explanation": "Hay cuatro grupos sanguíneos principales: A, B, AB y O."
},
{
  "question": "El verdadero nombre de Madonna es Madonna.",
  "answer": true,
  "explanation": "Su nombre completo es Madonna Louise Ciccone."
},
```

Figura 79: Ejemplos de preguntas de «Verdadero o falso»

***preguntasCompas.json*** Todas las preguntas acerca del resto de participantes que pueden salir en el minijuego de conoce a tus compañeros. Cada objeto se compone de un texto de la pregunta para cada tipo de participante (equipos o individuales).

```
{
  "questionE": "¿Alguien del equipo toca un instrumento musical?",
  "questionJ": "¿Toca un instrumento musical?"
},
{
  "questionE": "¿Alguien del equipo tiene hermanos?",
  "questionJ": "¿Tiene hermanos?"
},
```

Figura 80: Ejemplos de preguntas acerca de los compañeros de juego

***preguntasFechas.json*** El conjunto de posibles preguntas de fechas, cada objeto definido por tres atributos: el tipo de pregunta, que indica a su vez el tipo de respuesta esperada (día de la semana, mes o estación del año), el texto de la pregunta y el id o número de desplazamiento que hay que aplicar sobre la fecha actual para calcular la respuesta correcta.

```

{
  "type": "diaSemana",
  "question": "¿Qué día de la semana será mañana?",
  "id": 1
},
{
  "type": "mes",
  "question": "¿En qué mes estábamos el mes pasado?",
  "id": -1
},
{
  "type": "estacion",
  "question": "¿En qué estación del año estamos?",
  "id": 0
},

```

Figura 81: Ejemplos de preguntas acerca en «Recuerda la fecha»

En los tres ejemplos anteriores, las soluciones son dinámicas y se calcularían empleando los métodos de la clase *CasillaFechas* encargados de aplicar el desplazamiento indicado por el id sobre el valor de la fecha actual (una instancia nueva de *Date*). Por ejemplo, si se hace la pregunta el primer jueves de enero en España, la respuesta a cada una sería: viernes, diciembre e invierno.

### 7.3.6. Fichero «db.js»

Como indica el nombre del fichero, incluirá todas las funciones relativas a operaciones con elementos de las tablas alojadas en DynamoDB. Se destacan las tres principales:

***asumirRol()*** Permite asumir un rol de AWS para obtener credenciales temporales para poder acceder a DynamoDB de forma segura. Se han seguido los pasos de una de las guías de la [documentación oficial de Alexa](#):

1. Crear una instancia de cliente STS (*Security Token Service*) usando *AWS.STS*.
2. Se llama al método *assumeRole()* del cliente STS, proporcionando el ARN del rol que se desea asumir y un nombre para la sesión. Este devuelve credenciales temporales (*AccessKeyId*, *SecretAccessKey* y *SessionToken*).
3. Se termina de configurar el nuevo cliente *DynamoDB.DocumentClient*, utilizando las credenciales temporales obtenidas en el paso previo, que será el objeto que devuelve la función.

```

/**
 * Asume un rol IAM en AWS para obtener credenciales temporales.
 * @returns {AWS.DynamoDB.DocumentClient} Un cliente de DynamoDB configurado con credenciales temporales.
 * @throws {Error} Lanza un error si no puede asumir el rol correctamente.
 */
async function asumirRol() {
    const arnRol = 'arn:aws:iam::533267233233:role/rol-dynamo';

    const STS = new AWS.STS({ apiVersion: '2011-06-15' });

    const credentials = await STS.assumeRole({
        RoleArn: arnRol,
        RoleSessionName: 'sesionRol'
    }, (err, res) => {
        if (err) {
            console.log('asumirRol ERROR: ', err);
            throw new Error('Error al intentar asumir rol');
        }
        return res;
    }).promise();

    const db = new AWS.DynamoDB.DocumentClient({
        apiVersion: '2012-08-10',
        accessKeyId: credentials.Credentials.AccessKeyId,
        secretAccessKey: credentials.Credentials.SecretAccessKey,
        sessionToken: credentials.Credentials.SessionToken
    });

    return db;
}

```

Figura 82: Función para asumir un rol IAM con permisos temporales

**guardarPartida()** Almacena los detalles de una partida en la tabla *JuegoOca* y la información de los jugadores en la tabla *Jugador* en DynamoDB, de la siguiente forma:

1. Prepara un objeto *params* para guardar los datos generales de la partida en la tabla *JuegoOca*: estado, número de jugadores, ronda y turno.
2. Se usa el método *put* del cliente DynamoDB para guardar la información de la partida en la tabla, permitiendo capturar cualquier error que surja durante este proceso.
3. Llama a la función *guardarJugadores()*, encargada de repetir el mismo proceso seguido en los dos pasos anteriores, pero esta vez con los datos de cada participante: nombre, color, puntos, posición, etc.
4. Los resultados de ambas operaciones de guardado se concatenan en un mensaje de texto puramente informativo que será el devuelto por la función.

```


/**
 * Guarda los datos de una partida en DynamoDB, incluyendo la información de los participantes.
 * @param {AWS.DynamoDB.DocumentClient} db - El cliente de DynamoDB.
 * @param {Object} juegoOca - Objeto que contiene los datos del juego.
 * @returns {string} Mensaje de resultado de la operación.
 */
async function guardarPartida(db, juegoOca) {
    let txt = '';
    const params = {
        TableName: 'JuegoOca',
        Item: {
            idJuego: 0,
            estado: juegoOca.estado,
            hayEquipos: juegoOca.hayEquipos,
            numJugadores: juegoOca.numJugadores,
            ronda: juegoOca.ronda,
            turno: juegoOca.turno
        }
    };
    try {
        await db.put(params).promise();
        txt += 'Datos de la partida guardados con éxito en DynamoDB. ';
    } catch (error) {
        console.error('Error al guardar los datos de la partida en DynamoDB:', error);
        txt += `Error al guardar los datos de la partida: ${error.message}`;
    }
    txt += await guardarJugadores(db, juegoOca);

    return txt;
}


```

Figura 83: Función para guardar los datos de la partida en DynamoDB

**cargarPartida()** Leva a cabo el proceso anterior pero en sentido contrario, dicho en otras palabras, recupera la información desde la tabla de DynamoDB y la carga en en contexto de la skill de Alexa.

1. Se configura un objeto *params* para obtener los datos generales de la partida con la clave primaria, *idJuego*, de la tabla *JuegoOca*.
2. Se usa el método *get* del cliente DynamoDB para cargar la información de la partida, y si esta es válida, se actualizan los datos de la instancia actual de *JuegoOca*.
3. Invoca la función *cargarJugadores()*, para repetir el proceso de los pasos 1 y 2, esta vez usando la clave primaria *idJugador*, que coincide con el ID de cada participante.
4. La función devuelve los informes de éxito o fracaso de ambas operaciones de carga de datos.

```

/**
 * Carga los datos de una partida desde DynamoDB, incluyendo la información de los participantes.
 * @param {AWS.DynamoDB.DocumentClient} db - El cliente de DynamoDB.
 * @param {Object} juegoOca - Objeto que recibe y actualiza los datos de partida, incluyendo participantes.
 * @returns {string} Mensaje de resultado de la operación.
 */
async function cargarPartida(db, juegoOca) {
    let txt = '';
    try {
        const params = {
            TableName: 'JuegoOca',
            Key: {
                idJuego: 0
            }
        };

        const result = await db.get(params).promise();

        if (result.Item) {
            const datos = result.Item;
            juegoOca.hayEquipos = datos.hayEquipos;
            juegoOca.numJugadores = datos.numJugadores;
            juegoOca.crearJugadores(datos.numJugadores);
            juegoOca.ronda = datos.ronda;
            juegoOca.turno = datos.turno;
            juegoOca.estado = datos.estado;

            txt += `Datos de la partida cargados con éxito.`;
        } else {
            txt += 'Conexión exitosa, pero no hay ítem.';
        }
    } catch (error) {
        txt += `Error al conectar con DynamoDB: ${error.message}`;
    }

    txt += await cargarJugadores(db, juegoOca);

    return txt;
}

```

Figura 84: Función para cargar los datos de la partida desde DynamoDB

## 7.4. Pruebas

- hacer tablas + nombre prueba + descripción + respuesta esperada + validación + correspondencia CU
- simulador de alexa de dev console - echo show 5

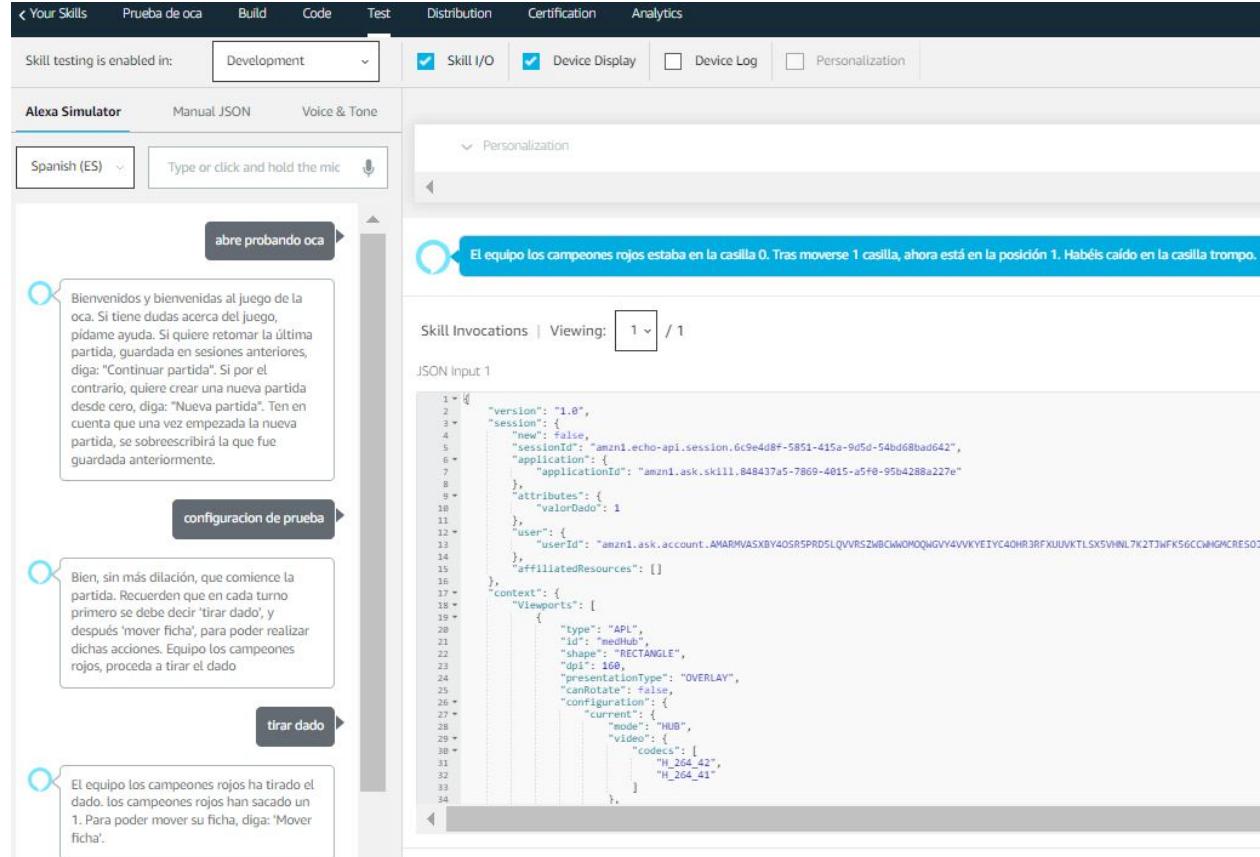


Figura 85: Función para cargar los datos de la partida desde DynamoDB

## 7.5. Seguridad

La seguridad es uno de los requisitos no funcionales más importantes, junto con la usabilidad. Si bien es cierto que en un juego de la oca no se tratan datos de alta importancia que haya que cifrar, es una skill que hace uso de servicios de *Amazon Web Services* (AWS). Estos últimos están vinculados a una cuenta administradora que gestiona los métodos de pago de los servicios cuando estos superan una determinada cuota de uso. Por tanto, para proteger estos servicios y limitar su uso a skills personalizadas concretas, es necesario controlar quién tiene acceso, siguiendo el principio de mínimo privilegio. Esto puede lograrse de forma eficiente mediante los roles de *Identity and Access Management* (IAM).

Estos roles son similares a los usuarios IAM, en el sentido de que permite gestionar el acceso a determinados servicios de AWS de forma centralizada; sin embargo, es más conveniente ya que en lugar de estar asociada a una única entidad, proporciona credenciales a cualquier persona usuaria con duración de una sesión. Además, facilitan el seguimiento de registros (o *logs*) de acciones entre usuarios y servicios.

Aunque Amazon ofrece los servicios de DynamoDB y Amazon S3 de manera gratuita para las skills alojadas en Alexa, no es sin limitaciones: solo puede crearse una tabla en Dynamo y hay un límite de almacenamiento y operaciones para el bucket de S3. Es por este motivo que se ha optado expandir los recursos de la skill con los servicios de una cuenta personal de AWS, para lo que se siguieron los pasos detallados en la sección 7.2.1.

## 8. Manual de uso de la skill

El primer paso es lanzar la skill, para lo que es necesario decir «*Alexa, abre probando oca*».

Alexa dará un mensaje de bienvenida a los jugadores. Para más información acerca del juego, se le puede pedir a Alexa que explique con más profundidad ciertos temas, diciendo: «*Explícame [tema]*». Los temas de ayuda disponibles son:

1. **Las reglas del juego:** hará una descripción general del juego y los objetivos a cumplir para ganar. Estos últimos son: llegar a la meta en primer lugar y/o conseguir la mayor cantidad de puntos a través de los minijuegos.
2. **Las casillas:** ofrece una breve explicación de cada tipo de casilla. Consultar escenario C.
3. **Los minijuegos:** menciona los minijuegos que hay y en qué consiste cada minijuego. Consultar escenario D.
4. **Los comandos de voz:** comenta brevemente los comandos de voz que el usuario puede utilizar para interactuar con la skill. Consultar cuadro 34.

Además, si en cualquier momento se intenta realizar una acción no válida o transcurre un determinado tiempo sin que Alexa reciba respuesta, se activará un mensaje de ayuda recordando el estado actual de la partida y cómo progresar adecuadamente a partir de ese punto.

| Comando                   | Cuándo puede invocarse   |
|---------------------------|--|
| Abrir la skill            | En cualquier momento, mientras no se haya iniciado la skill aún                                  |
| Ayuda/Explicación de tema | En cualquier momento   |
| Nueva partida             | En cualquier momento, pero sobrescribe los últimos datos de partida guardados                    |
| Continuar partida         | En cualquier momento, pero sobrescribe los datos de partida actuales                             |
| Guardar partida           | Al inicio de cualquier turno, antes de tirar el dado   |
| Cerrar la skill           | En cualquier momento, con el riesgo de perder el progreso actual si no se ha guardado la partida |
| Tirar dado                | Al inicio del turno de un participante o después de haber caído en una casilla de oca            |
| Mover ficha               | Después de un lanzamiento de dado  |

Cuadro 34: Lista de comandos de voz y ámbito de uso

### **Escenario A: Iniciar una nueva partida.**

Para comenzar una nueva partida, se debe decir «*Nueva partida*».

Alexa procederá a pedir de forma secuencial los datos necesarios para configurar la partida en proceso de creación, en el siguiente orden:

1. **El modo de juego:** las dos opciones son «por equipos» o «jugadores individuales». Este parámetro afectará a la forma en la que Alexa se refiere a los participantes (en singular o plural).
2. **El número de participantes** (equipos o jugadores) que van a competir. Debe ser una cifra entera entre 1 y 5.
3. **El nombre de cada participante.** Alexa dará indicaciones sobre cómo registrar un nombre y luego irá preguntado a cada participante, uno por uno. La sintaxis que hay que seguir para ello es: «*Mi/Nuestro nombre es [nombre]*».

Una vez registrados todos los datos, Alexa hará un resumen general de la configuración de la partida, tras lo cual se pasará al escenario C.

### **Escenario B: Continuar con la partida anterior.**

Si se ha guardado anteriormente una partida, la próxima vez que se inicie una sesión de la skill se podrá retomar desde el punto en el que se dejó, mediante la instrucción «*Continuar partida*».

Esto permitirá cargar el estado de la partida desde la base de datos y continuar el juego a partir del último turno registrado.

### **Escenario C: Jugar un turno.**

Durante la partida, Alexa anunciará el turno del equipo/participante actual. Cada turno se compone de dos fases clave: el lanzamiento de dado y el movimiento de su ficha.

Para tirar el dado, dicho participante o un integrante del equipo debe decir «*Tirar dado*», lo que llevará a mostrar la animación y el resultado del lanzamiento.

Acto seguido, debe decir «*Mover ficha*» para poder avanzar tantas posiciones como indique el dado. El juego lo colocará en la nueva casilla, que dependiendo de su tipo, puede implicar el final de dicho turno o el inicio de una nueva acción:

- **Casilla normal:** se narra un evento concreto en función de la casilla en la que haya caído, que puede variar de unas veces a otras, y finaliza el turno.

- **Casilla de oca:** se mueve a la siguiente casilla del mismo tipo y puede tirar de nuevo el dado. En el caso en el que haya caído en la oca anterior a la casilla de meta, se traslada a esta última y finaliza el juego.
- **Casilla de puente:** solo existen dos de este tipo, y caer en una provoca el desplazamiento inevitable a la otra, tras lo cual finaliza el turno.
- **Casilla de penalización:** a su vez se distinguen cuatro tipos (pozo, laberinto, hotel y cárcel), entre los que puede variar el número de turnos que hay que esperar antes de poder avanzar de nuevo.
- **Casilla de minijuego:** por lo general, implica que Alexa le haga una pregunta de un tipo concreto para la posibilidad de ganar puntos si se acierta. Para un desglose de los tipos de juegos y respuestas aceptadas, consultar el escenario D.
- **Casilla de meta:** se da la enhorabuena a los ganadores o ganadores (quienes hayan llegado antes a la meta y/o hayan logrado acumular más puntos con los minijuegos), y se da por finalizada la partida.

#### **Escenario D: Participar en un minijuego**

A este escenario se llega solo si se cae en una de las casilla de minijuego, por lo que la cantidad de puntos acumulados no solo depende de cuántas se acierten, sino también de la frecuencia en la que se caiga en ellas para aumentar la posibilidad de sumar puntos al marcador.

Dependiendo del tipo de casilla de minijuego que sea, se dan las siguientes situaciones:

- **Verdadero o falso:** Alexa hace una pregunta de verdadero o falso. Si la respuesta es correcta, se añaden 10 puntos.
- **Adivina la cifra:** Alexa hace una pregunta cuya respuesta es un número. Si se acierta, se ganan 30 puntos, si no, la pregunta rebota al siguiente equipo/participante, y así sucesivamente hasta que alguien acierte o todos hayan dado respuestas incorrectas, entonces Alexa revelará la solución.
- **Conoce a tus compañeros:** Alexa formulará una pregunta acerca de uno de los compañeros, que se elige de manera aleatoria. El primero responde con lo que piense que es correcto, y será el propio sujeto de la pregunta el que tenga que confirmar si su respuesta es correcta o no. Si lo es, ambos ganan 15 puntos.
- **Recuerda la última casilla:** desafía la memoria a corto plazo, pues Alexa preguntará por el nombre de la última casilla en la que estuvo. Si es capaz de acordarse, ganará 25 puntos y si no, se revelará la solución.

- **Recuerda la fecha:** Alexa hace una pregunta cuya respuesta puede ser un día de la semana, mes o estación del año concreta, por la oportunidad de ganar 20 puntos. Para hallar la respuesta correcta, es necesario acordarse de la fecha actual, por lo que pone a prueba la noción del tiempo y memoria de la actualidad de quienes juegan.

Para entender mejor el tipo de respuestas esperadas en cada minijuego, se puede consultar la tabla siguiente.

| Minijuego                  | Formato de respuesta esperada                            |
|----------------------------|--|
| Trivial V/F                | Verdadero o falso  |
| Conoce a los participantes | Correcto/a o incorrecto/a                                |
| Adivina la fecha           | Un número entero positivo                                |
| Recuerda la última casilla | El nombre de una casilla                                 |
| Recuerda la fecha          | El nombre de un día de la semana, mes o estación del año |

Cuadro 35: Tipos de respuestas del usuario a los minijuegos

#### Escenario E: Terminar la partida

Basta con decir «***Cierra probando oca***» para cerrar la skill y por tanto finalizar la partida, sin embargo, esta acción borrará de manera permanente los datos de la partida actual a no ser que se hayan guardado previamente.

## **9. Conclusiones y trabajos futuros**

En conclusión, este TFG que se integra en el proyecto de investigación «Evaluación del uso de robots sociales y sistemas conversacionales en Residencias y Centros de Día para promover el envejecimiento saludable», desarrolla un juego en Alexa cuyo sector de la población al que se dirige son las personas mayores que están en situación de poco contacto social, como contribución al desarrollo de soluciones innovadoras y efectivas que aborden el problema del aislamiento social en este sector de la población.

Para lo cual se plantea un juego digital que además de ofrecer entretenimiento y diversión, promueve la interacción social, el compromiso cognitivo y emocional. Se trata de una nueva versión interactiva del tradicional Juego de la Oca, con minijuegos incorporados que le dotan de originalidad y mucho entretenimiento.

Alexa es el asistente virtual escogido para el juego a desarrollar, debido a que sus skills ofrecen una amplia gama de posibilidades para crear experiencias interactivas y entretenidas. Estas habilidades permiten a los desarrolladores crear juegos de diferentes géneros y niveles de complejidad, desde simples juegos de palabras y adivinanzas hasta juegos de aventuras o trivial más elaborados

### **9.1. Consecución de objetivos**

Los objetivos alcanzados han sido de manera resumida y tras analizar la diversidad de intervenciones digitales dirigidas a este grupo demográfico de personas mayores, el diseño del juego para lo que se han investigado las mejores prácticas en el diseño de juegos digitales accesibles para personas mayores. Para ello ha sido necesario documentarse sobre investigaciones publicadas acerca el impacto del aislamiento social en personas mayores como el artículo *La soledad y el aislamiento social en las personas mayores* (Arruebarrena & Cabaco, 2020), que define el aislamiento social como «una ausencia objetiva de relaciones/contactos sociales y la soledad como la experiencia subjetiva aversiva que se siente al valorar esas relaciones/contactos sociales como insuficiente en cantidad y/o calidad».

Además, se han considerado las adaptaciones necesarias para abordar posibles limitaciones físicas y cognitivas de las personas a las que va dirigido en el desarrollo técnico del juego, como seleccionar la plataforma y tecnologías más apropiadas para el desarrollo del mismo, y asegurar la compatibilidad con dispositivos comunes utilizados por personas mayores.

Otro de los objetivos conseguidos ha sido integrar funcionalidades de accesibilidad, como ajustes de tamaño de fuente y navegación simplificada.

Este trabajo como contribución al conocimiento se centra en cómo la tecnología puede mejorar

la calidad de vida de las personas mayores, particularmente en el contexto del aislamiento social.

Para la implementación del juego se han investigado otros similares así como aplicaciones ya que en la actualidad ya se han presentado algunas iniciativas dirigidas a las personas mayores que están en situación de aislamiento como, por ejemplo, el asistente virtual Celia desarrollado por personal del Centro de Investigación en Tecnologías de Telecomunicación de la Universidad de Vigo, atlanTTic ([El Confidencial, 2024](#)). También se han consultado de manera exhaustiva la documentación oficial de la página de desarrolladores de Alexa.

Otra de las aplicaciones estudiadas ha sido la skill de Alexa para la mejora de la inhibición de respuesta, que consiste en dos juegos principales (animales y colores) con dos modalidades cada uno, empleando la voz y una interfaz gráfica en un dispositivo de Alexa.

## **9.2. Trabajos futuros**

Como aplicación práctica de «Participación comunitaria», se podrían hacer pruebas del juego en residencias de personas mayores o en centros de día, con una persona que actuará como guía para llevar un seguimiento de la experiencia de los usuarios. De esta manera a partir de la retroalimentación de los y las participantes, se podrían hacer modificaciones o ampliaciones de aspectos del juego para adaptarlos mejor a las personas usuarias.

Otras de las áreas que se podrían completar sería la capacidad de guardar más de una partida en la base de datos, en lugar de solo la última. Esta circunstancia sería útil si se fueran a llevar a cabo varias pruebas piloto distintas, con grupos diversos y durante más de una sesión. Así, se podría mantener el progreso de varias partidas simultáneamente.

Además, se podrían ampliar aspectos del juego como hacer varios tableros, implementar minijuegos nuevos, etc, siempre pensando en las personas a las que va dirigida. Se podría proponer al popular programa de CanalSurTV “La tarde con Juan y Medio”, cuyos invitados son personas mayores que hicieran una sección para practicar con este juegos interactivo, de manera que se hiciera popular entre su audiencia.

## 10. Bibliografía

- Abt, C. (1970). Serious games. *New York: Viking Press.*
- Arruebarrena, A. V., & Cabaco, A. S. (2020). La soledad y el aislamiento social en las personas mayores. *Studia Zamorensia*, (19), 15-32.
- Azevedo, M., & Lemos, J. L. (2022). Ancianos activos, días divertidos. Diseño de juegos para personas mayores. *Zincografía*, 124-138.
- Blat, J., Lluis Arcos, J. L., & Sayago, S. (2012). WorthPlay: Juegos digitales para un envejecimiento activo y saludable. *Revista Lychnos*, 8, 16.
- Bunbury Bustillo, E., Osuna-Acedo, S., & Pérez Calle, R. (2022). *Las competencias digitales en personas mayores: de amenaza a oportunidad*, (No. ART-2022-131993).
- Canal Séniior. (2022). *La tecnología en el ocio séniior: de esta forma utilizamos nuestros dispositivos para entretenernos*. Consultado el 15 de julio de 2024, desde <https://canalsenior.es/noticias/tecnologia-en-el-ocio-senior/>
- Castillo Moreno, D. (2022). *Implementación de juegos basados en Alexa orientados a la detección del deterioro cognitivo* [Bachelor's thesis].
- Cuidalian. (s.f.). *Juegos de memoria para personas mayores*. Consultado el 19 de febrero de 2024, desde <https://cuidalian.com/blog/juegos-memoria-personas-mayores/>
- Duarte, A. O., & Rojas, M. (2008). Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo. *Revista Avances en Sistemas e Informática*, 5(2), 159-171.
- El Confidencial. (s.f.). *Apps y juegos para la mente: Ejercitar memoria en personas mayores*. Consultado el 19 de febrero de 2024, desde <https://cuideo.com/blog/apps-juegos-ejercitar-memoria-personas-mayores/>
- El Confidencial. (2024). *Celia, la 'app' gallega que charla con mayores y detecta el Alzheimer*. Consultado el 11 de febrero de 2024, desde [https://www.elconfidencial.com/espana/galicia/2024-01-08/celia-app-gallega-charla-mayores-detecta-alzheimer\\_3805293/](https://www.elconfidencial.com/espana/galicia/2024-01-08/celia-app-gallega-charla-mayores-detecta-alzheimer_3805293/)
- Ferrer, J. R. C. (2018). Juegos, videojuegos y juegos serios: Análisis de los factores que favorecen la diversión del jugador. *Miguel Hernández Communication Journal*, (9), 191-226.

- Fullerton, T. (2008). *Game design workshop: a playcentric approach to creating innovative games*. CRC press.
- García Piñeiro, F. (2022). *Skill de Alexa para monitorización de personas mayores* [Bachelor's thesis].
- Gudmundsen, J. (2006). Movement aims to get serious about games. *USA Today*, 5(19).
- Hernández, P. R., & Cruz, D. V. (2023). Los asistentes virtuales basados en Inteligencia Artificial. *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica*, 11(2), C1-11.
- Instituto Nacional de Estadística. (2023). *Población que usa Internet (en los últimos tres meses). Tipo de actividades realizadas por Internet*. Consultado el 15 de julio de 2024, desde [https://www.ine.es/ss/Satellite?L=es\\_ES&c=INESeccion\\_C&cid=1259925528782&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout](https://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=1259925528782&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout)
- López Garzón, C., & Rozo Barrera, F. (2023). *Narrativa transmedia para ayudar a los adultos mayores a adquirir habilidades tecnológicas* [Bachelor's thesis].
- Marcano, B. (2008). Juegos serios y entrenamiento en la sociedad digital. *Education in the knowledge society (EKS)*, 9(3), 93-107.
- Mendoza, G. A. A., Sauvé, L., & Plante, P. (2017). Adultos mayores y juegos educativos digitales. ¿Qué consideraciones de diseño favorecen su uso? *Edutec, Revista Electrónica de Tecnología Educativa*, (62), 104-116.
- Menéndez Román, S. (2023). *Creación de una skill de Alexa para la mejora de la inhibición de respuesta* [Bachelor's thesis].
- Menzinsky, A., López, G., Palacio, J., Sobrino, M. Á., Álvarez, R., & Rivas, V. (2018). Historias de usuario. *Ingeniería de requisitos ágil*.
- Ortín, N. U., Villanueva, F. U., & López, F. A. (2008). Una propuesta de evaluación para las habilidades motrices básicas en Educación Primaria a través de un juego popular: la oca. *Retos. Nuevas tendencias en Educación Física, Deporte y Recreación*, (14), 35-42.
- Palacios-Rodríguez, A., Rodríguez, J. M. R., & Gómez, G. (s.f.). CAPÍTULO 2 AISLAMIENTO SOCIAL Y USO DE LAS TIC EN PERSONAS MAYORES EN ÉPOCA DE PANDEMIA.

**COVID 19. REVISIÓN SISTEMÁTICA.** *Desafíos de investigación educativa durante la pandemia.*

Regalón, J. A. L., & Céspedes, I. M. (2019). Los juegos serios en el entrenamiento y la rehabilitación cognitiva. *Revista Cubana de Informática Médica*, 11(2), 140-157.

Sawyer, B., & Rejeski, D. (2002). Serious games: Improving public policy through game-based learning and simulation. *Woodrow Wilson International Center for Scholars*.

Stafford, G. (2018). *Building Asynchronous, Serverless Alexa Skills with AWS Lambda, DynamoDB, S3, and Node.js*. Consultado el 8 de julio de 2024, desde <https://programmaticponderings.com/2018/07/24/building - asynchronous - serverless - alexa - skills - with - aws - lambda - dynamodb-s3-and-node-js/>

Teleasistencia Vital. (s.f.). *Los 10 mejores juegos de mesa para personas mayores*. Consultado el 19 de febrero de 2024, desde <https://teleasistenciamayores.com/blog/10-juegos-mesa-personas-mayores/>

Vanova, M., Irazoki, E., García-Casal, J. A., & Botella, C. (2018). The effectiveness of ICT-based neurocognitive and psychosocial rehabilitation programmes in people with mild dementia and mild cognitive impairment using GRADIOR and ehcoBUTLER: study protocol for a randomised controlled trial. *Trials*, (19), 100.

Wilson Center. (2024). *The Serious Games Initiative*. Consultado el 15 de julio de 2024, desde <https://www.wilsoncenter.org/publication-series/serious-games>