Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela politécnica superior de Jaén

Trabajo Fin de Grado

IMPLEMENTACIÓN DE JUEGOS BASADOS EN ALEXA ORIENTADOS A DETECCIÓN DEL DETERIORO COGNITIVO

Alumno: Daniel Castillo Montero

Tutor: Fernando Javier Martínez Santiago

Dpto: Informática

Septiembre, 2022

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Fernando Javier Martínez Santiago , tutor del Trabajo Fin de Grado titulado: Implementación de juegos basados en Alexa orientados a la detección del deterioro cognitivo, que presenta Daniel Castillo Montero, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2022

El alumno:

CASTILLO MONTERO DANIEL - 53598224J
Firmado digitalmente por CASTILLO MONTERO DANIEL - 53598224J
Fecha: 2022.09.07 16:03:03 +02'00'

Daniel Castillo Montero

Los tutores:

MARTINEZ SANTIAGO FERNANDO JAVIER - 26016529X
Firmado digitalmente por MARTINEZ SANTIAGO FERNANDO JAVIER - 26016529X
Fecha: 2022.09.07 15:55:44 +02'00'

Fernando Javier Martinéz Santiago

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1. Especificación del trabajo

1.1. Introducción

1.1.1. Detección temprana del deterioro congnitivo asociado al envejecimiento.

El aumento de la esperanza de vida da lugar a mayor desgaste en la salud de una persona en edades avanzadas. Con frecuencia, este desgaste no tiene un impacto significativo sobre la calidad de vida, sin embargo, siempre existe la posibilidad de que los síntomas empeoren y acabe en una enfermedad grave. Este deterioro afecta de forma negativa principalmente a la autonomía personal.

Es en este desarrollo paulatino donde la detección temprana de los primeros síntomas (especialmente el deterioro de la memoria de trabajo) donde tiene mayor beneficio un diagnóstico temprano. Con esto se puede ralentizar el progreso de la enfermedad y mejorar la calidad de vida del paciente y es la motivación para desarrollar un seguimiento y monitorización del paciente.

Este proceso de seguimiento y almacenamiento de datos implica la recogida y centralización de mucha información del paciente, por lo que es aquí donde entran las TIC, como los asistentes virtuales. Dicha tecnología permite automatizar la recolección de información, distribuir y mantener la aplicación de forma sencilla y personalizar la experiencia siempre de una forma remota, para mayor comodidad del paciente.

1.1.2. Asistentes virtuales

El método más común para la evaluación cognitiva es el uso de test escritos (pruebas de papel y lápiz). Estas pruebas presentan el inconveniente, entre otros, de que los pacientes las consideran monótonas, aburridas y muy similares a un examen (lo que puede producir estrés en la persona). Estos problemas hacen que a largo plazo los participantes se desmotiven y los datos obtenidos pierdan fiabilidad.

Las personas mayores también interactúan en el día a día con dispositivos electrónicos, por ello es posible el uso de herramientas modernas, como los asistentes virtuales ya que estos tienen una apariencia muy similar a otros aparatos electrónicos cotidianos como teléfonos móviles o tablets. Estos frente a otros tipos de tecnología

permiten que la interacción de la persona con el dispositivo sea más natural y transparente (se puede interactuar solo con comandos de voz sencillos, por ejemplo), por lo que su curva de aprendizaje es mucho menor.

Por otro lado, los asistentes virtuales son una herramienta versátil y ubicua que ofrecen un marco de trabajo compatible con otras tecnologías actuales y amigables para el desarrollador. Este motivo permite digitalizar los test tradicionales de papel y lápiz con un diseño más desenfadado y variado que motive al paciente en la realización de los test porque de esta forma pierde el aspecto de examen y se parece más a un juego[9] (conocido como serious games).

1.2. Objetivos del trabajo

El objetivo general de este proyecto consiste en el desarrollo del prototipo de una herramienta clínica que permita a un psicólogo especialista monitorizar la capacidad de memoria de trabajo de los pacientes. Esto objetivo se divide en los siguientes objetivos específicos:

- Una aplicación, conocida como Skill, que será accesible desde cualquier altavoz inteligente de la familia de Alexa
- Un sistema de almacenamiento en la nube donde se almacenará la información relacionada con los usuarios.
- Una aplicación que permita al especialista consultar y recoger la información de los usuarios al interactuar con el sistema.

1.3. Presupuesto

Este apartado tiene como objetivo estimar y justificar el coste económico de desarrollo del proyecto. Para ello, se definirá el software necesario para cumplir con las necesidades del proyecto, de forma que en el segundo (sección de hardware) se podrá elegir un equipo adecuado que cumpla con las necesidades del software con un precio adecuado.

El presupuesto para este proyecto se divide en 3 categorías:

- **Software:** incluye los servicios de software necesarios: SO, servicio de API de navegación y ruta…. Se indicará el uso de cada servicio y si se estimará si el umbral de pago será excedido.
- **Hardware:** incluye el equipo con que se utilizará en la programación y dos servidores: uno para almacenar la base de datos y otro donde se almacenarán imágenes.
- **Personal:** será el presupuesto destinado a la contratación de personal para la gestión y programación.

1.3.1. Coste personal

En este apartado se estima el coste de contratación de personal (desglosado en la Tabla 1) requerido para el proyecto. Aunque este proyecto pertenezca a un Trabajo de Fin de Grado, se supondrá el personal y costes según dicta el BOE[7] y la Consejería de Hacienda, Industa y Energía[8].

Personal	Trabajo	Tiempo en el proyecto	Coste
Programador senior	Programación	12 meses	24.640,37€
Arquitecto de software	Diseño y toma de decisiones sobre el software	4 meses	8.396,33€
Jefe de proyecto	Gestión	4 meses	4.465€
Psicólogo	Parte clínica y modelado de la información	3 meses	5.673€
		Total	**43.174€**

Tabla 1. Coste de personal del proyecto

1.3.2. Coste Software

En este apartado se incluye todo el software de pago, especificando el precio base en caso de ser un servicio de suscripción y el precio completo para la duración del proyecto (desglosado en la Tabla 2). La API de Open Route Service ofrece 3 servicios diferentes, de los cuáles, para este proyecto el más conveniente será la licencia standard[16], que es más que suficiente ya que solo será necesaria una vez en el proyecto.

Software	Descripción	Coste estimado
Office 365 Personal	Documentación	50’40€ (4’20€/mes)
Amazon EC-2 + Amazon S3	Servicios en la nube de Amazon para alojar el backend.	264€ (22.00€/mes)
Open Rout Service	API para generar los recorridos	Licencia estándar gratuita
	Total	**314,40€ (26’20€/mes)**

Tabla 2. Coste del software del proyecto

1.3.3. Coste Hardware

El grueso del cómputo de este proyecto residirá en los servicios de Amazon para Alexa y AWS[14], por lo que con equipos de bajas prestaciones será suficiente siempre y cuando dichos equipos sean compatibles con el software nombrado en este apartado. En la Tabla 3 se muestra el precio del equipo necesario

Hardware	Descripción	Coste estimado
Asus VivoBook	Equipo portátil para desarrollar el proyecto. Incluye S.O., teclado, ratón y pantalla	599€
Alexa Echo Show	Altavoz inteligente de la familia de Alexa	249’99€
	Total	**848,99€**

Tabla 3. Coste hardware del proyecto

1.3.4.Amortización

Una vez estimado el coste de cada apartado del proyecto, se resume en la Tabla 4 el total de costes para la duración del proyecto.

Descripción	Coste estimado
Coste hardware	848,99€
Coste software	314,40€
Coste personal	43.174€
Total	**44.337,39€**

Tabla 4. Resumen de costes del proyecto

Una vez se ha realizado el coste estimado de cada una de las partes en las que se divide el proyecto, se puede calcular la amortización a los bienes materiales, tanto el hardware como el software de este proyecto. El software contratado tiene una vida útil con pagos mensuales, salvo la licencia del sistema operativo que está incluida en el equipo portátil.

De estos bienes mencionados, solo se podrá amortizar el hardware, ya que en este proyecto no hay software permanente. Para estos componentes se calculará una amortización lineal con los coeficientes que considera la Agencia Tributaria[12] dando los resultados expuestos en la siguiente tabla como se muestra en la Tabla 5:

Elemento	Valor inicial	Valor residual	Vida útil	Amortización anual
Asus Vivobook	599€	490€	8 años	13,62€
Alexa echo Show	249,99€	180€	10 años	6’99€
			Total	**20,61€**

Tabla 5. Valor amortizado para cada pieza de hardware

Dado que el proyecto tiene una duración estimada de 1 año, el valor amortizado total será de 20,61€

1.4. Planificación temporal

Debido a que el sistema descrito de forma natural se divide en módulos, para el desarrollo del proyecto se ha optado por un método incremental. Este ha consistido en el planteamiento de un objetivo y una reunión semanal para hacer seguimiento de

dicho objetivo. Una vez satisfecha la meta, se continúa con el siguiente hasta el final de la implementación del proyecto.

El proyecto se ha dividido en los siguientes incrementos descritos en las Ilustraciones 1 a 5 y la Tabla 6 (en el apartado 4 se describirá cada una con más detalle):

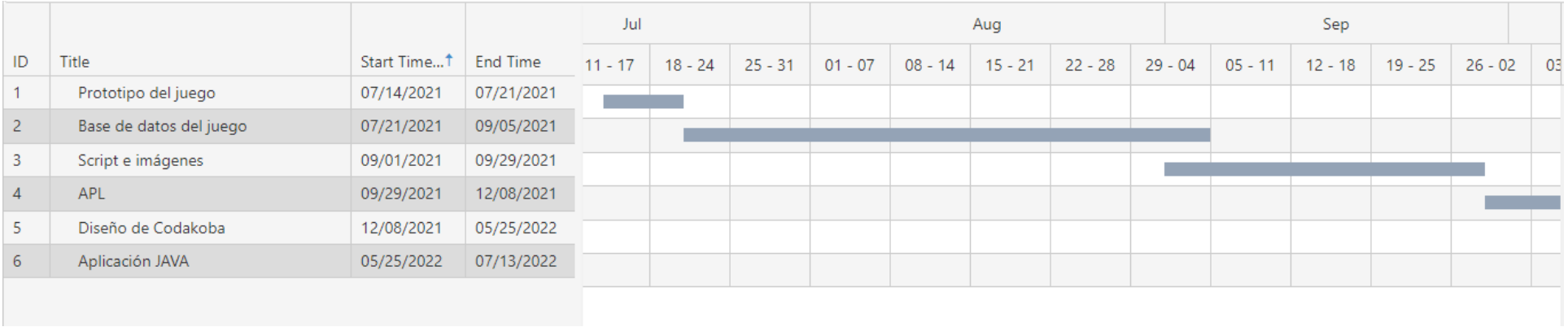


Ilustración 1. Diagrama de Gantt (1/5)

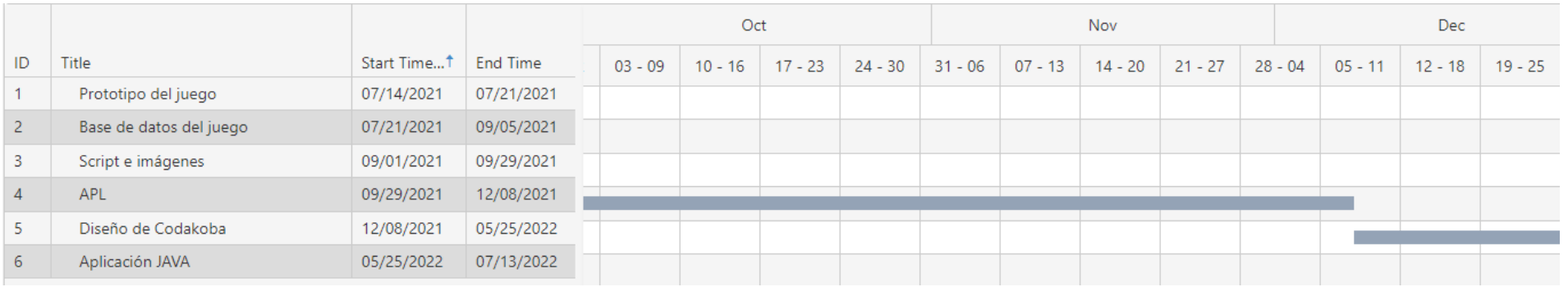
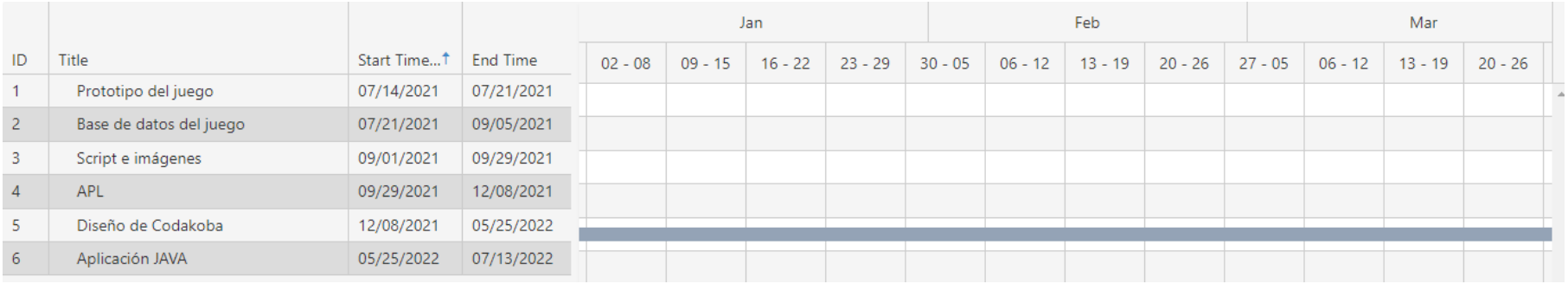
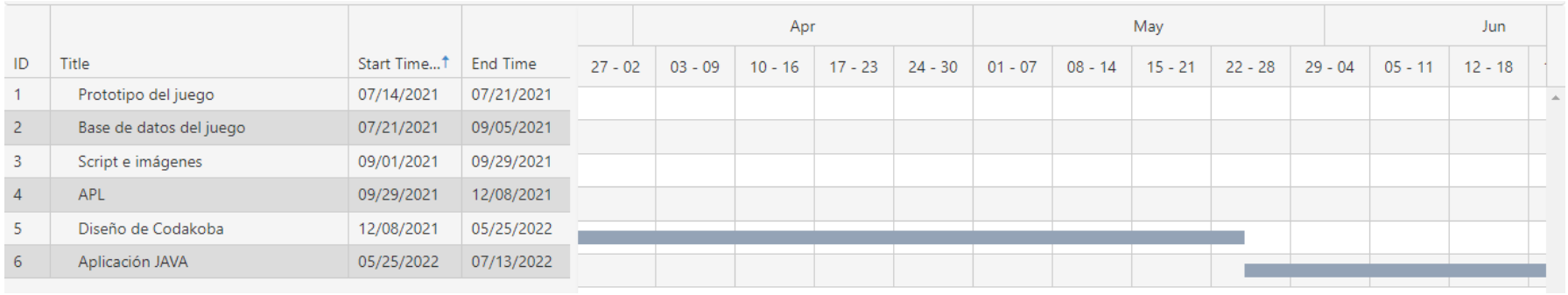


Ilustración 2. Diagrama de Gantt (2/5)



ID	Title	Start Time...	End Time
1	Prototipo del juego	07/14/2021	07/21/2021
2	Base de datos del juego	07/21/2021	09/05/2021
3	Script e imágenes	09/01/2021	09/29/2021
4	APL	09/29/2021	12/08/2021
5	Diseño de Codakoba	12/08/2021	05/25/2022
6	Aplicación JAVA	05/25/2022	07/13/2022

Ilustración 3. Diagrama de Gantt (3/5)



ID	Title	Start Time...	End Time
1	Prototipo del juego	07/14/2021	07/21/2021
2	Base de datos del juego	07/21/2021	09/05/2021
3	Script e imágenes	09/01/2021	09/29/2021
4	APL	09/29/2021	12/08/2021
5	Diseño de Codakoba	12/08/2021	05/25/2022
6	Aplicación JAVA	05/25/2022	07/13/2022

Ilustración 4. Diagrama de Gantt (4/5)

ID	Title	Start Time...	End Time
1	Prototipo del juego	07/14/2021	07/21/2021
2	Base de datos del juego	07/21/2021	09/05/2021
3	Script e imágenes	09/01/2021	09/29/2021
4	APL	09/29/2021	12/08/2021
5	Diseño de Codakoba	12/08/2021	05/25/2022
6	Aplicación JAVA	05/25/2022	07/13/2022

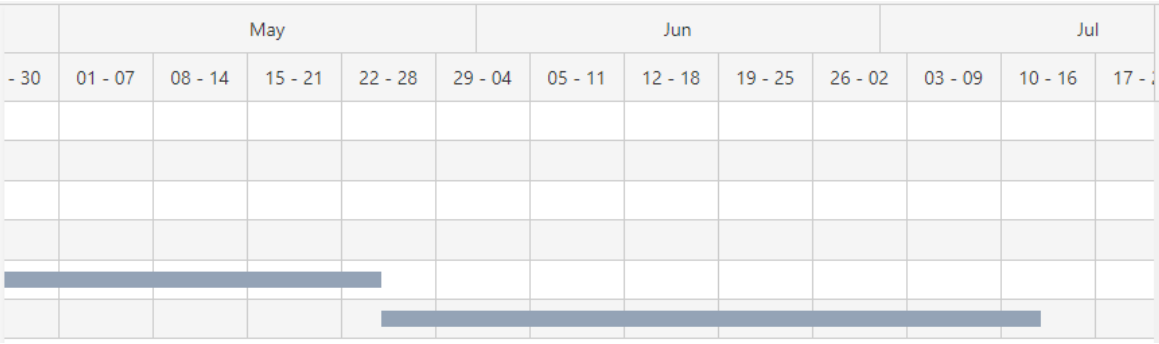


Ilustración 5. Diagrama de Gantt (5/5)

Iteración	Tiempo dedicado	Meta
Iteración 1: Prototipo del juego	14 de julio de 2021 21 de julio de 2021	Estudiar el funcionamiento de ASK, recibir una descripción del juego del psicólogo e implementar un prototipo en Alexa con datos de prueba y sin imágenes.
Iteración 2: Base de datos del juego	21 de julio de 2021 1 de septiembre de 2021	Estudiar el funcionamiento de TypeDB y diseñar la base de datos que almacenará las rutas para el juego.
Iteración 3: Script e imágenes	1 de septiembre de 2021 29 de septiembre de 2021	Implementar el script que poblará la base de datos con las calles y rutas e implementar las consultas en Alexa. Crear el servicio en S3 para las imágenes.
Iteración 4: APL	29 de septiembre de 2021 8 de diciembre de 2021	Estudiar APL para añadir soporte táctil e imágenes en el juego.
Iteración 5: Diseño de Codakoba	8 de diciembre de 2021 25 de mayo de 2022	Determinar la información con el psicólogo la información que se almacenará de los usuarios. Hacer el esquema de Codakoba y las consultas desde Alexa
Iteración 6: Aplicación Java	25 de mayo de 2022 13 de julio	Crear la aplicación en Java para consulta de datos de usuarios.

Tabla 6. Descripción de las iteraciones del proyecto

2. Análisis

2.1. Descripción del Sistema

Con ayuda de la Ilustración 6 se describirá de forma general el funcionamiento del sistema y las partes que lo forman:

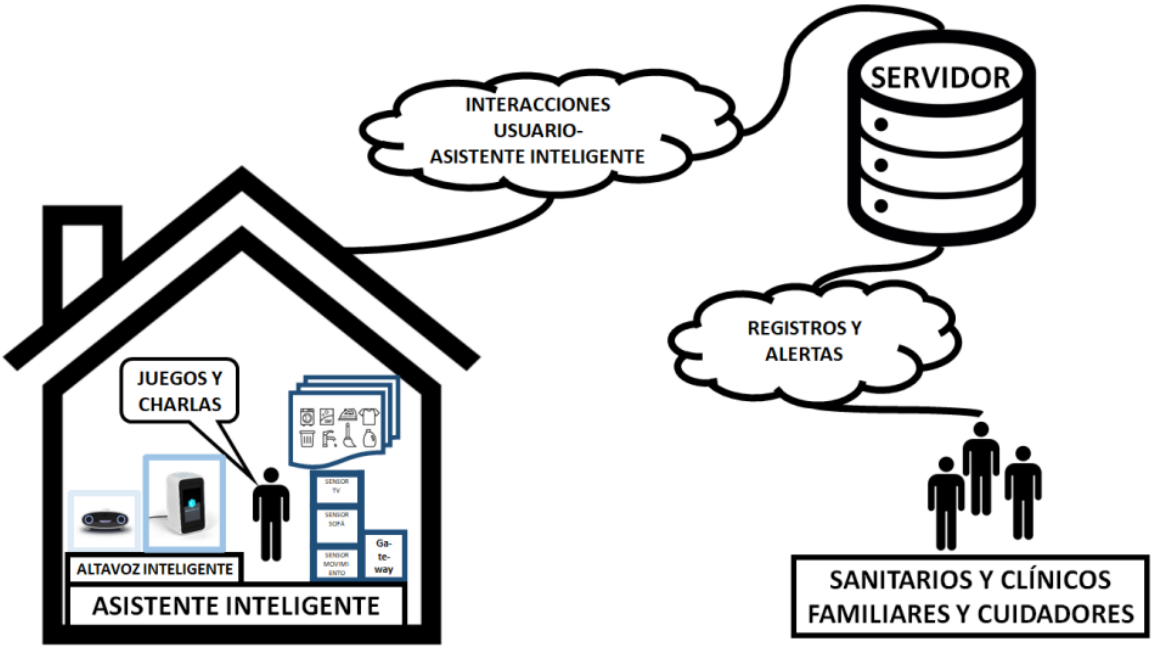


Ilustración 6. Representación gráfica del sistema

- En primer lugar, cada usuario tendrá un altavoz inteligente, concretamente alguno de la familia de Alexa, con el que jugará con cierta frecuencia.
- Los datos de cada partida (respuestas del usuario, tiempo en responder, desempeño…) se almacenarán en un servidor externo a la casa del usuairo y común para todos.
- La información almacenada en el servidor estará a disposición de especialistas que se encargarán de analizar los datos de cada usuario.

2.2. Requisitos

Una vez visto el funcionamiento general del sistema y sus partes se pueden definir los requisitos tanto funcionales como no funcionales del proyecto.

Requisitos funcionales:

- El sistema explicará en cada paso del juego qué es lo que debe responder el usuario.
- El sistema deberá registrar y almacenar cada interacción del usuario con el dispositivo.
- La información que se almacene de la partida seguirá el formato descrito por el especialista.
- El sistema debe permitir recoger información de usuarios, filtrada o sin filtrar y con un formato especificado por los sanitarios.

Requisitos no funcionales:

- El juego se diseñará para altavoces inteligentes de la familia de Alexa con pantalla, pero debe adaptarse automáticamente si no cuenta con esta.
- El sistema y su interfaz debe de ser intuitiva y fácil de interpretar para el usuario.
- La interacción necesaria con el altavoz inteligente debe ser la mínima necesaria.
- El software para acceder y recoger datos debe ser compatible con las últimas versiones de Windows, MacOS y Linux.

3. Diseño

Ahora que están descritos los requisitos del sistema y su funcionamiento general, se puede profundizar más en cada parte del mismo. Esto se verá con ayuda de la Ilustración 7:

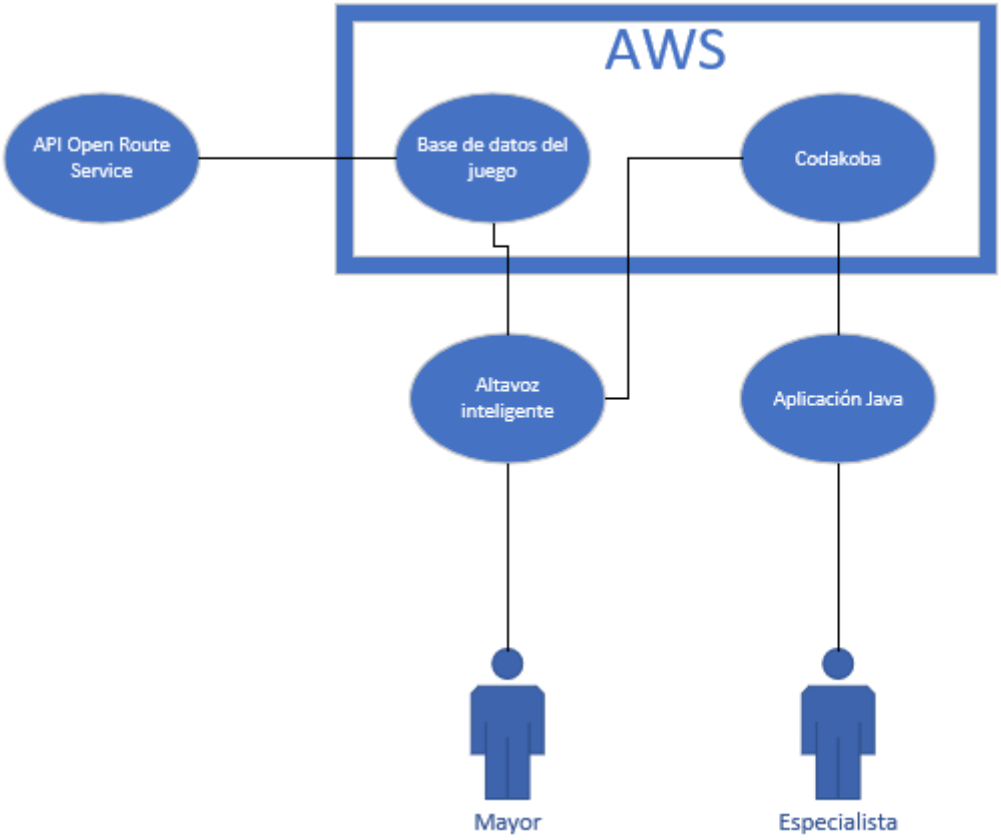


Ilustración 7. Representación gráfica de los diferentes módulos que forman el proyecto

En la ilustración se puede observar:

- La información del juego se generará con un script que aproveche la API de Open Route Service, pero solo hará falta la primera vez para poblar la base de datos.
- Tanto la base de datos con la información del juego como de los usuarios estará alojada en un servidor de AWS. Será un servidor que aloje a ambas.
- El altavoz inteligente solo necesitará acceder a dicho servidor de AWS.
- Los especialistas podrán acceder a la información de los usuarios desde una aplicación Java.
- El mayor solo interactuará a través del dispositivo inteligente y no tendrá acceso a los datos de partida.
- El sistema de base de datos y servidor será totalmente transparente al cualquier tipo de usuario.

3.1. Tecnologías utilizadas

En este apartado se describen las tecnologías utilizadas en el desarrollo del proyecto, así como se justifica su uso.

3.1.1. TypeDB

Para organizar la información se ha optado por usar TypeDB. Este sistema de base de datos se centra en las relaciones entre entidades, con lo que el tipo de consultas que se prevén mejorarán su rendimiento. Además, dispone de un cliente para Python, por lo que facilita mucho el acceso desde Alexa.

El uso de esta tecnología se divide entre la parte servidor y la parte cliente. Para el lado servidor se creará una instancia de este servicio que pone a disposición las bases de datos necesarias. Por otro lado, el acceso a esta instancia desde el lado del cliente se hará mediante el cliente de Python y el de Java oficial para que sea más fácil y con la menor cantidad de servicios intermediarios posible.

3.1.2. Python y módulos de terceros

Como se ha descrito en puntos anteriores, la parte cliente de TypeDB es compatible con Python. Sabiendo esto, además de su versatilidad y la variedad de sus módulos, justifican su elección para

El uso de Python se divide en dos partes: una se corresponde con la generación de información para la base de datos mediante un script y otra con la implementación del juego en Alexa (que se trata en otro apartado).

Los módulos utilizados han sido:

- **Folium[6]**: permite generar un mapa HTML para visualizar la ruta (usado principalmente en la depuración del código).
- **Pillow[11]**: transforma el mapa HTML generado por Folium en un archivo PNG. También se ha usado principalmente en la depuración del código.

- **JSON** y **request**: se han usado para el intercambio de mensajes con la API OpenRouteService[10].

3.1.3.ASK (Alexa Skllt Kit)

Alexa Skill Kit da a elegir entre dos lenguajes para la implementación: Python y Node.js. Por el mismo motivo que en el punto anterior, se ha elegido Python.

Para la implementación del propio juego se han utilizado las herramientas propias de Alexa en Python que permiten la recogida de información del usuario desde el dispositivo y controlar el contenido en pantalla con APL (Alexa Presentation Language).

3.1.4.API OpenRouteService

La generación de las rutas se ha hecho mediante un script que conecte con esta API. Este servicio, a partir de las coordenadas de los puntos de partida y final, da como respuesta una lista de calles que forman la ruta a seguir desde los puntos proporcionados.

Estos puntos de inicio y fin han sido elegidos a mano y se han obtenido sus coordenadas mediante la página web[1] de la misma API para generar después desde el script las rutas que se agregarán posteriormente a la base de datos y que formarán los niveles del juego.

3.1.5.Servicios de Amazon

Para alojar la información en la nube se ha elegido a Amazon, pues son fáciles de configurar y ofrecen unas prestaciones de potencia de cómputo y velocidad de conexión más que suficientes para alojar el proyecto.

[1] ORS Maps

De los servicios que Amazon pone a disposición, se han utilizado dos en específico: Amazon Web Servicies y Amazon S3. Por un lado, con Amazon Web Servicies se instanciarán dos bases de datos: una máquina virtual donde se desplegará la base datos que registrará la interacción de cada usuario en cada partida (esta información será analizada posteriormente por el especialista) y otra base de datos con la información necesaria para jugar: rutas, calles, url de imágenes… Por otro lado, S3 permitirá almacenar las imágenes de cada calle que serán mostradas como refuerzo visual al usuario durante el juego y serán accesibles mediante una url que estará en la base de datos del juego.

3.1.6. Java

Este lenguaje se ha utilizado para implementar la aplicación que permita al especialista recoger los datos desde la base de datos en AWS. Específicamente, la interfaz gráfica la forma la librería de JavaSwing y la conexión y consultas al servidor se gestionan con el cliente de TypeDB para este mismo lenguaje.

3.2. Diagrama de casos de uso

A continuación, veremos cómo será la interacción de los diferentes usuarios con el sistema con los diagramas de casos de uso correspondientes. Los usuarios del sistema podrán tener dos roles diferentes:

- **Mayor:** representa a una persona de la que se desea hacer el seguimiento. Debido a que el sistema está orientado a personas de edad avanzada, la interacción con el sistema será solo la necesaria y lo más simple posible.
- **Especialista:** se refiere a una persona perteneciente al personal médico que observará y analizará los resultados para hacer el seguimiento y pronóstico del mayor.

Las historias de usuario se pueden ver en la Tabla 7:

Actor	Caso de uso	Valor aportado
Como	**Quiero**	**Para poder**
Mayor	Jugar con el altavoz inteligente	Registrar mi desempeño en una partida
Especialista	Obtener los datos de los pacientes	Hacer seguimiento del mayor

Tabla 7. Historias de los tipos de usuario del sistema

3.2.1.Jugar con el altavoz inteligente

En este subapartado se verá el caso de uso en la Tabla 8 y la Ilustración 8 que describe el proceso por el que pasaría un paciente durante una partida cualquiera.

Caso de uso	Registro
Actor primario	Mayor
Sistema	Altavoz inteligente
Participantes	Mayor
Nivel	Objetivo de usuario
Condiciones previas	Ninguna
Operaciones básicas	
1	Pedir ayuda
2	Iniciar partida
3	Salir
Alternativas	
1.A	Explicación
2.A	Iniciar partida

Tabla 8. Caso de uso para jugar con el altavoz inteligente

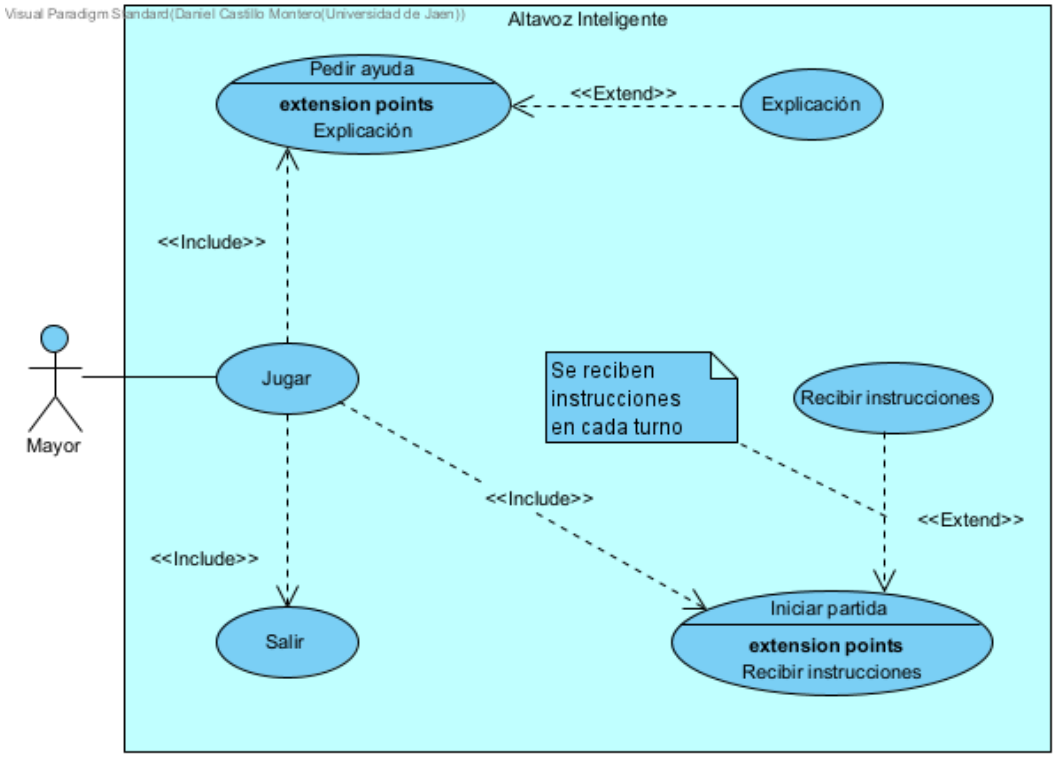


Ilustración 8. Diagrama de caso de uso para jugar con el altavoz inteligente

3.2.2.Obtener los datos de los pacientes

En este subapartado se verá el caso de uso en la Tabla 9 y la Ilustración 9 que describe el proceso por el que pasaría un especialista que quiera consultar la información de las partidas de un mayor.

Caso de uso	Registro
Actor primario	Especialista
Sistema	Aplicación Java
Participantes	Especialista
Nivel	Objetivo de usuario
Condiciones previas	Ninguna
Operaciones básicas	
1	Introducir datos
2	Salir
Alternativas	
1.A	Validar datos

Tabla 9. Caso de uso para recoger datos de los pacientes

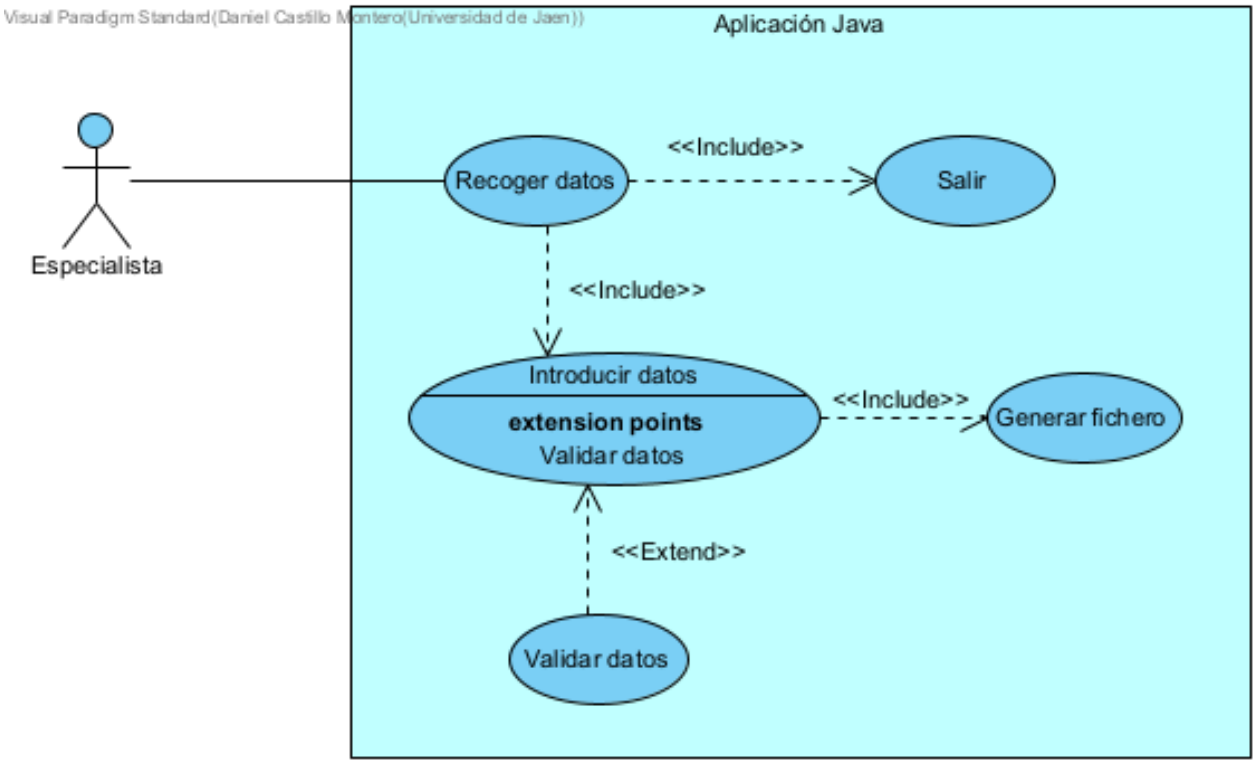


Ilustración 9. Diagrama de caso de uso para recoger datos de los pacientes

3.3. Diagramas de secuencia

3.3.1. Jugar con el altavoz inteligente

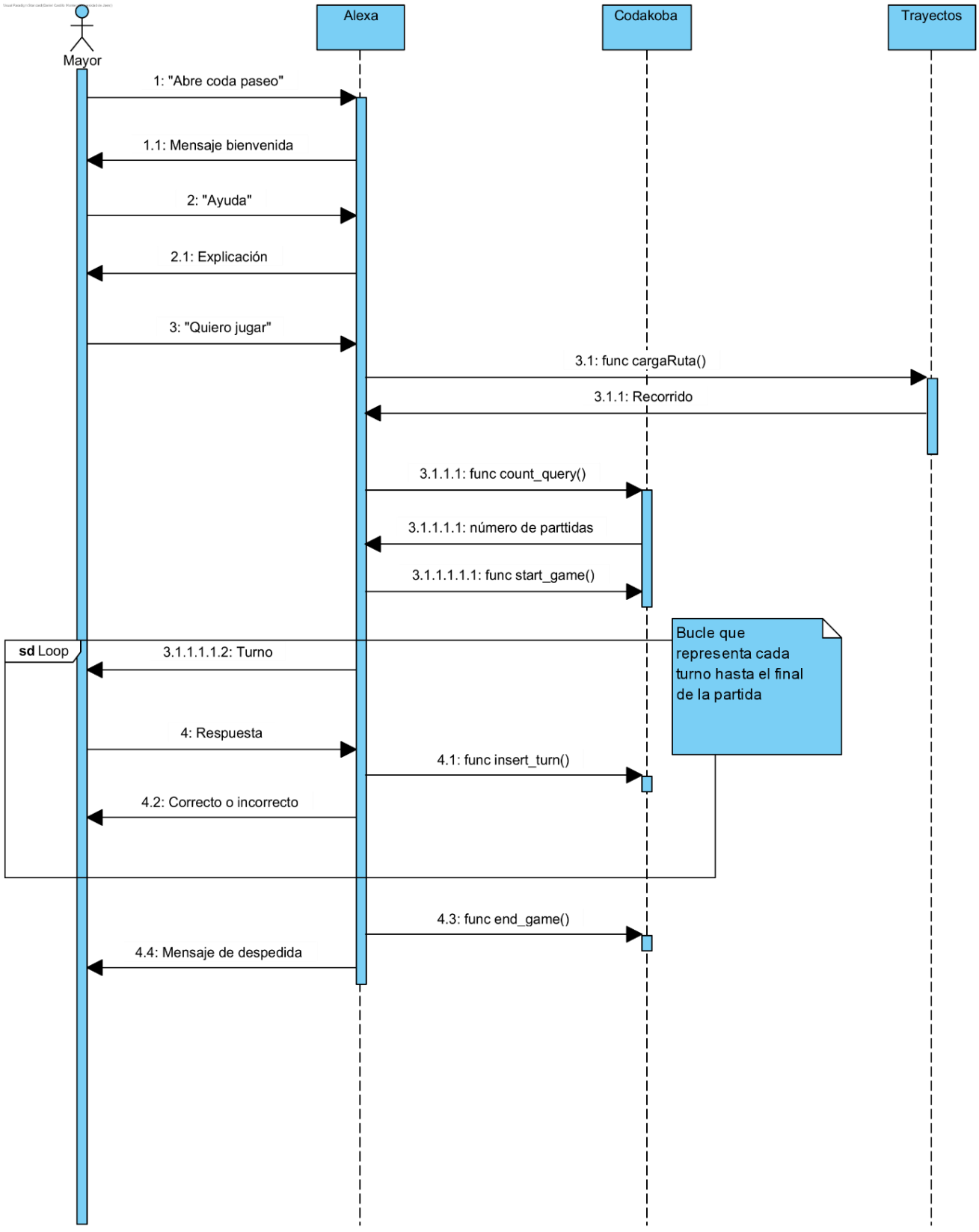


Ilustración 10. Diagrama de secuencia de una partida con el altavoz inteligente

3.3.2. Obtener los datos de los pacientes

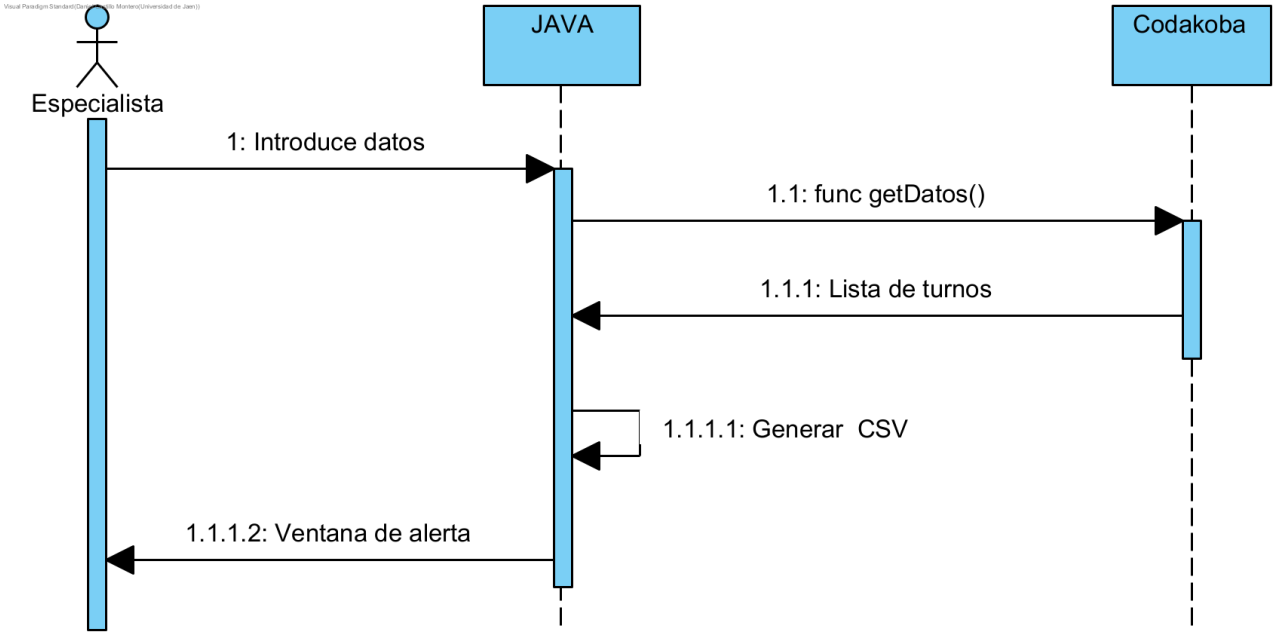


Ilustración 11. Diagrama de secuencia para obtener datos de Codakoba

3.4. Base de datos

Toda la persistencia de datos relacionada con el proyecto se dividirá en dos bases de datos diferentes alojadas en la misma instancia de TypeDB en AWS. A continuación, se describe el diseño de ambas.

3.4.1. Codakoba

Esta será la base de datos donde se almacenará la información la información personal que identifique al mayor y su interacción con el juego. Esta última fue indicada por el psicólogo y es la que se necesita para poder hacer el análisis sobre el desempeño del mayor. Estos datos se describen a continuación:

- Nombre y apellidos del mayor.
- Fecha de la partida.

- Formato de la partida (directo o indirecto), esto es, si se le pidió que la respuesta fuera en el mismo orden en el que se dictó o en orden inverso, respectivamente.
- Longitud de la secuencia
- Primer o segundo intento para ese turno.
- Respuesta dictada por Alexa y respondida por el mayor.
- Tiempo de reacción, que quiere decir el tiempo que hay entre que el mayor recibe la respuesta y responde.

Por otro lado, también hay que almacenar la información relativa al juego (y que sirva para almacenar más en el futuro). Estos datos son:

- Nombre
- Descripción
- Necesidad de pantalla para funcionar

Las partidas están modeladas de forma que cada vez que un mayor inicia una partida, se crea una relación entre una entidad s__Person (la persona que juega) y una entidad Game (a lo que juega). Esta relación tiene un atributo de tipo turn y gracias a la propiedad de TypeDB de ser multievaluado, se pueden asociar varias entidades a la misma relación game-played, de forma que se genera la lista de turnos que han sido jugados para dicha partida.

Con estos requisitos y este dieño, se ha modelado el esquema lógico para Codakoba representado en la Ilustración 12 y una descripción más detallada en la Tabla 10.

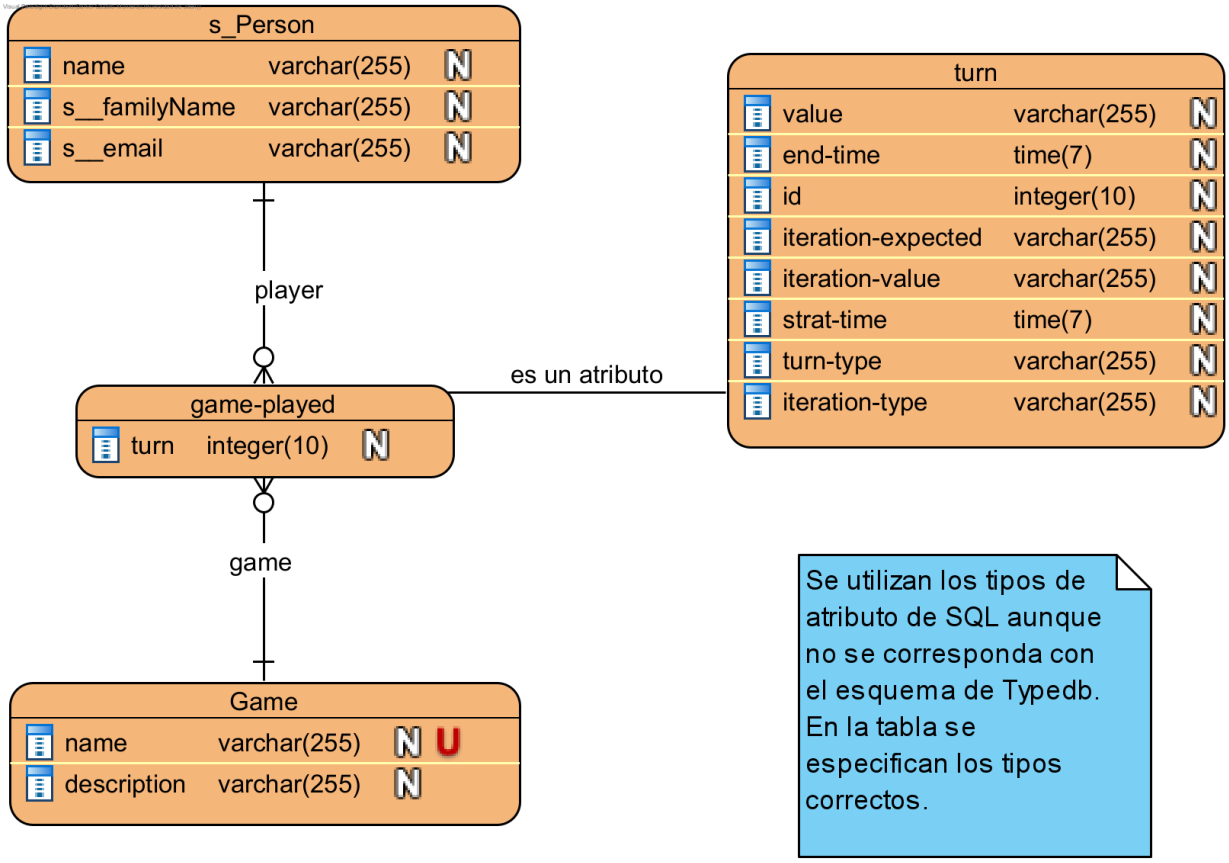


Ilustración 12. Esquema de la base de datos de Codakoba

Entidad	Descripción	Atributo	Tipo	Descripción
S__Person	Representa toda la información asociada al mayor para ser identificado	name	string	Nombre del mayor
		s__familyName	string	Apellido del mayor
		s__email (key)	string	Correo para identificar al mayor. Debe ser el mismo con el que se inicie sesión en el dispositivo de Alexa

Game	Información relevante sobre el juego	name	string	Nombre del juego
		description	string	Breve descripción del juego que se muestra al pedir ayuda durante la partida
game-played	Relación en la que se almacena la información relacionada con una partida	id (key)	long	Identificador de una partida. Se construye como el número de partidas en la base de datos +1.
		game-ended	boolean	Indica si la partida fue acabada o se dejó a medias.
		end-time	datetime	Marca de tiempo para el momento en el que se termina la partida
		start-time	datetime	Marca de tiempo para el momento en el que empieza la partida
		turn	turn	atributo multievaluado con los turnos que se han

				jugado en la partida a la que está asociado.
		variant	long	Diferencia entre el formato directo o indirecto con 0 y 1, respectivamente
turn	Almacena todos los datos relativos a un turno de una partida	id(key)	long	Atributo único para identificar a un turno. Se contruye como el número de turnos en la base de datos + 1
		end-time	datetime	Marca de tiempo cuando se recibe la respuesta del mayor
		iteration-expected	string	Respuesta correcta para el turno actual
		iteraton-value	string	Respuesta dada por el mayor
		iteration-type	string	Indica si el mayor respondió por voz o de forma táctil. Restringido a los

				valores “tactile” y “voice”.
		start-time	datetime	Marca de tiempo cuando se recibe la respuesta del mayor
		try	string	indica si es el primer o el segundo intento para este turno.

Tabla 10. Descripción de componentes de la base de datos de Codakoba

3.4.2.Juego trayectos

En esta base de datos es donde se almacenará la información necesaria para que se desarrolle la partida. Esto incluye rutas, calles y sus imágenes. Al empezar una partida, se elegirá una ruta aleatoria dando más probabilidad de ser elegidas a las rutas que menos se hayan jugado.

A continuación, se muestra el esquema lógico en la Ilustración 13 y una explicación más en detalle en la Tabla 11:

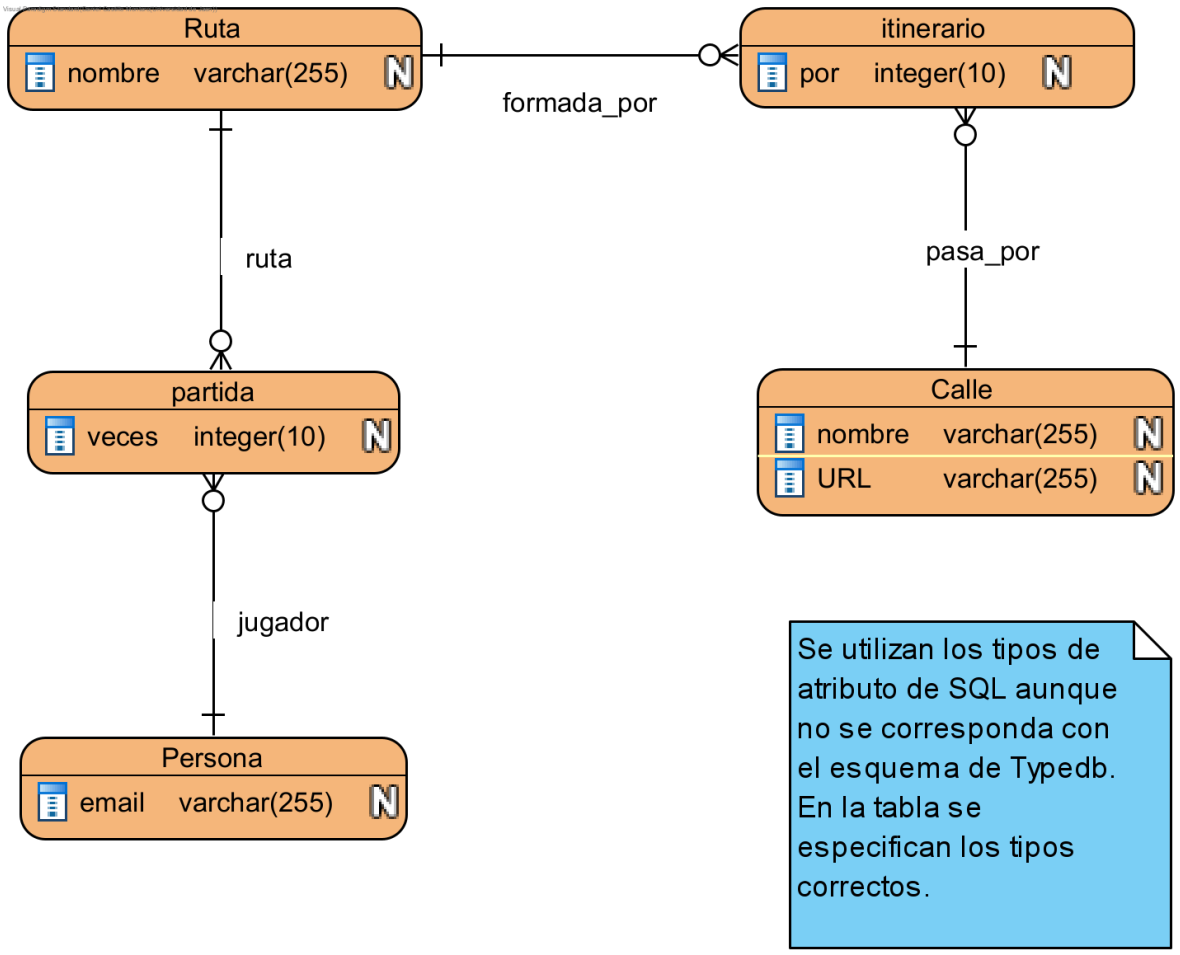


Ilustración 13. Esquema de la base de datos de Trayectos

Entidad	descripción	atributo	tipo	descripción
Ruta		nombre	string	Nombre que identifica al recorrido. Se forma concatenando el origen y el destino de la ruta
partida		veces	long	Nº de veces que una persona ha jugado una ruta.
Persona		email	string	Correo para identificar al mayor. Debe ser el mismo con el que se inicie sesión en el dispositivo de Alexa

itinerario		pos	long	Posición que ocupa una calle en el recorrido.
calle		nombre	string	Nombre de una calle dado por Open Route Service
		URL	string	Dirección que identifica la imagen de esta callen en S3. En el apartado 4.3.4 se explica en detalle como se genera y funciona

Tabla 11. Descripción de componentes de la base de datos de trayectos

4. Desarrollo del proyecto

4.1. Primera iteración: Prototipo del juego

El primer paso fue crear una base para el proyecto desde donde construir el resto del mismo. Esta base consiste en un prototipo del juego que permita ser jugado de principio a fin sin almacenar información del usuario ni datos del juego final.

Este juego fue descrito por el psicólogo en reuniones previas del proyecto. Esta descripción marca las frases con las que el usuario dará órdenes a Alexa durante la partida y lo que se espera de ella al recibirlas (ya que ASK está orientado a eventos). Estas órdenes y sus acciones se describen a continuación en la Tabla 12:

Entrada del usuario (intetns)	Respuesta de Alexa
“Abre coda paseo”	Lanzar la Skill
“Quiero jugar”	Comenzar con primera ronda
“Ayuda”	Explicación breve de cómo se juega
nombre de una calle	Decir si el usuario ha acertado o no
“Salir”	Se cierra la Skill

Tabla 12. Intents del usuario y respuestas de Alexa

El flujo de la partida quedaría según se representa en la Ilustración 14:

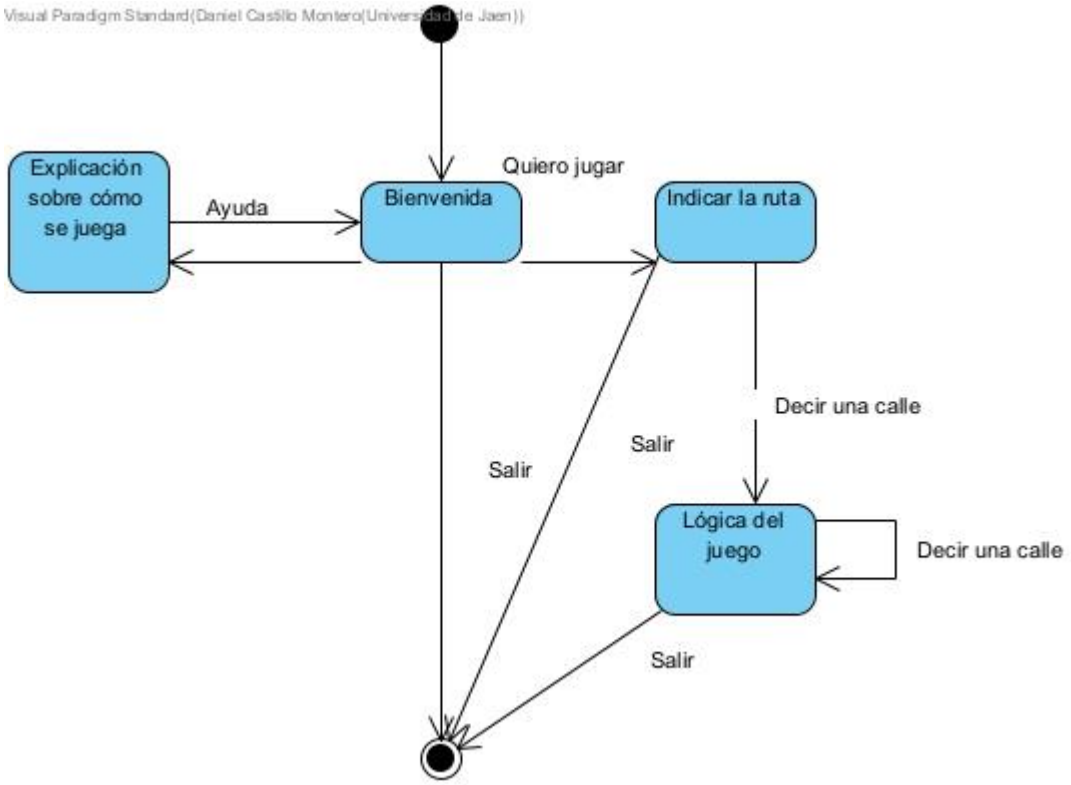


Ilustración 14. Diagrama del flujo de la partida

En este punto el juego es funcional de principio a fin, pero solo funciona mediante iteración por voz y tiene un único recorrido a modo de ejemplo (calle 1, calle 2, calle 3 y calle 4). En posteriores iteraciones de añadirá más funcionalidad e interfaz gráfica.

A continuación, se muestra en la Ilustración 15 un ejemplo en la consola de desarrollador de Alexa de cómo sería interactuar con Alexa al lanzar la skill y pedir una explicación del juego. En el apartado 4.3 se mostrarán más ejemplos durante la partida.

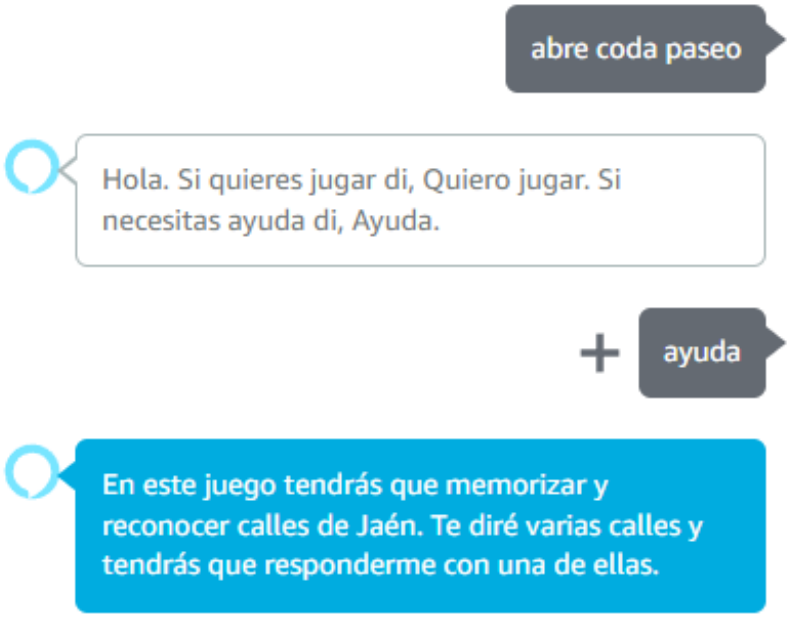


Ilustración 15. Ejemplo de interacción con Alexa en el que explica el funcionamiento del juego

Para añadir más variedad a las respuestas, para indicar si el usuario ha acertado o no se elige una frase aleatoria de entre un conjunto de cada, de forma que las respuestas de Alexa sean menos monótonas para el usuario.

4.2. Segunda iteración: Base de datos del juego

Este tramo del desarrollo del desarrollo tiene como meta implementar el esquema para la base de datos. Tanto el esquema como los detalles del mismo están explicados en el apartado 3.4.2. Ahora solo queda trasladar el diseño lógico a un archivo de texto plano siguiendo la sintaxis propia de TypeDB e importarlo al servidor de AWS. En el siguiente apartado se poblará con información.

4.3. Tercera iteración: Script e imágenes

Una vez creada la base de datos en el servidor y teniendo acceso a la misma, solo falta poblarla con datos. Esta tarea se va a hacer de forma parcialmente automática mediante un script en Python.

Este script aprovechará principalmente la API de Open Route Service y el cliente de TypeDB en Python para su tarea. Como entrada necesitará un fichero que tendrá

las coordenadas de inicio y final de cada ruta y como salida dará un fichero con la lista de calles

4.3.1. Fichero de entrada

Este fichero simplemente estará formado por texto plano con un formato concreto. Este formato será, en líneas individuales:

- Longitud de origen
- Latitud de origen
- Longitud de destino
- Latitud de destino
- Nombre de la ruta

Para más comodidad, se ignoran las líneas en blanco y se pueden añadir comentarios si la línea empieza por '#'. En la Ilustración 16 se muestra como se ve una parte del que se ha usado para este proyecto:

```
#Ruta 0
-3.7973913436942053
37.77790603853329
-3.792912255744428
37.77858723019613
Iglesia de San Félix-Iglesia del Salvador

#Ruta1
-3.7968653848750478
37.77193980114735
-3.7924078398620247
37.77060289734392
Convento de Santa Úrsula-Convento de Santa Clara
```

Ilustración 16. Contenido del fichero de coordenadas conteniendo dos trayectos

Las coordenadas deben introducirse manualmente. Para este caso, se ha usado el mapa[2] que proporciona Open Route Service en su página web, pero podría usarse cualquier otro.

[2] Mapa de Open Route Service que proporcionada las coordenadas

4.3.2. Script

El funcionamiento se divide en la siguiente secuencia:

1. Lee el fichero con las coordenadas, que puede estar en el mismo directorio o en cualquiera si se especifica como argumento.
2. Crea un subdirectorio llamado “datos” donde guardará la información de salida.
3. Para cada ruta:
    1. Hace una consulta al api y obtiene como respuesta un JSON con la ruta a pie.
    2. Almacena el JSON en un fichero de texto plano y el recorrido en un fichero HTML y en PNG.
    3. Guarda en un array la lista de calles ordenadas y sin repetir.
4. Conecta con la base de datos y almacena cada calle con su información (nombre y url a la imagen), cada ruta y la relación de cada ruta con sus calles.

Cada entrada de una calle almacenará una url con el formato “Trayectos/(*nombre_calle)*.png” (como se ve en la Ilustración 17), que se corresponderá con la dirección al archivo dentro del servicio de S3. En el siguiente punto se explicará cómo se usa este enlace.

```
{
    $nombre "Calle de Fermín Palma" isa nombre;
    $url "Trayectos/Calle de Fermín Palma.png" isa URL;
}
{
    $nombre "Calle de Goya" isa nombre;
    $url "Trayectos/Calle de Goya.png" isa URL;
}
{
    $nombre "Calle de la Cruz Roja Española" isa nombre;
    $url "Trayectos/Calle de la Cruz Roja Española.png" isa URL;
}
{
    $nombre "Calle de Úbeda" isa nombre;
    $url "Trayectos/Calle de Úbeda.png" isa URL;
}
{
    $nombre "Calle Duquesa" isa nombre;
    $url "Trayectos/Calle Duquesa.png" isa URL;
}
```

Ilustración 17. Estado de algunas entidades en la base de datos de Trayectos

El pseudocódigo para este script quedaría como se ve en la Ilustración 18:

```
coord = [] #Lista con las coordenadas de origen y destino de las rutas
for linea in ficheroCoordenadas:
    c = extaercoordenada(linea)
    coord.append(c)

crearDirectorio("./datos")

for ruta: #Una iteracción por cada ruta del fichero
    json = hacerConsulta() #Consulta a la API
    json.guardarFichero()

    generaraMapaHTML() #Para esto se usa Folium
    almacenarMapa()

    exportarMapaPNG()

    volcadoBaseDatos()
```

Ilustración 18. Pseudocódigo del scritp que puebla la base de datos

4.3.3. Resultados

Después de terminar la ejecución del script, la información necesaria se encuentra en la base de datos, además de un subdirectorio en la misma ubicación que el script que contiene un nuevo archivo con una lista de todas las calles (que se

explicará en el siguiente apartado), además del JSON recibido dese la api para cada recorrido y un fichero HTML y una imagen en formato PNG que marca sobre un mapa el recorrido. Este último tiene el aspecto que muestra la Ilustración 19:

Ilustración 19. Ejemplo de salida del script para un mapa

4.3.4. Imágenes

Para almacenar las imágenes se ha tenido que buscar una alternativa a TypeDB, ya que este sistema no tiene soporte para archivos binarios. La solución implementada consiste en almacenar las imágenes como tal en S3 (servicio de amazon que se puede acceder con ASK) y en TypeDB en guardar junto a cada calle una URL parcial a la imagen que le corresponde. Más adelante, durante la ejecución del juego, se usa la librería de S3 para formar en enlace completa a una imagen concreta.

La lista de nombres de calles que se obtiene como resultado del script del punto anterior sirve para buscar a mano una foto para cada calle y poder nombrar cada archivo con el nombre que se espera que tenga según la información la base de datos en el servidor. En la Ilustración 20 se muestra un ejemplo:

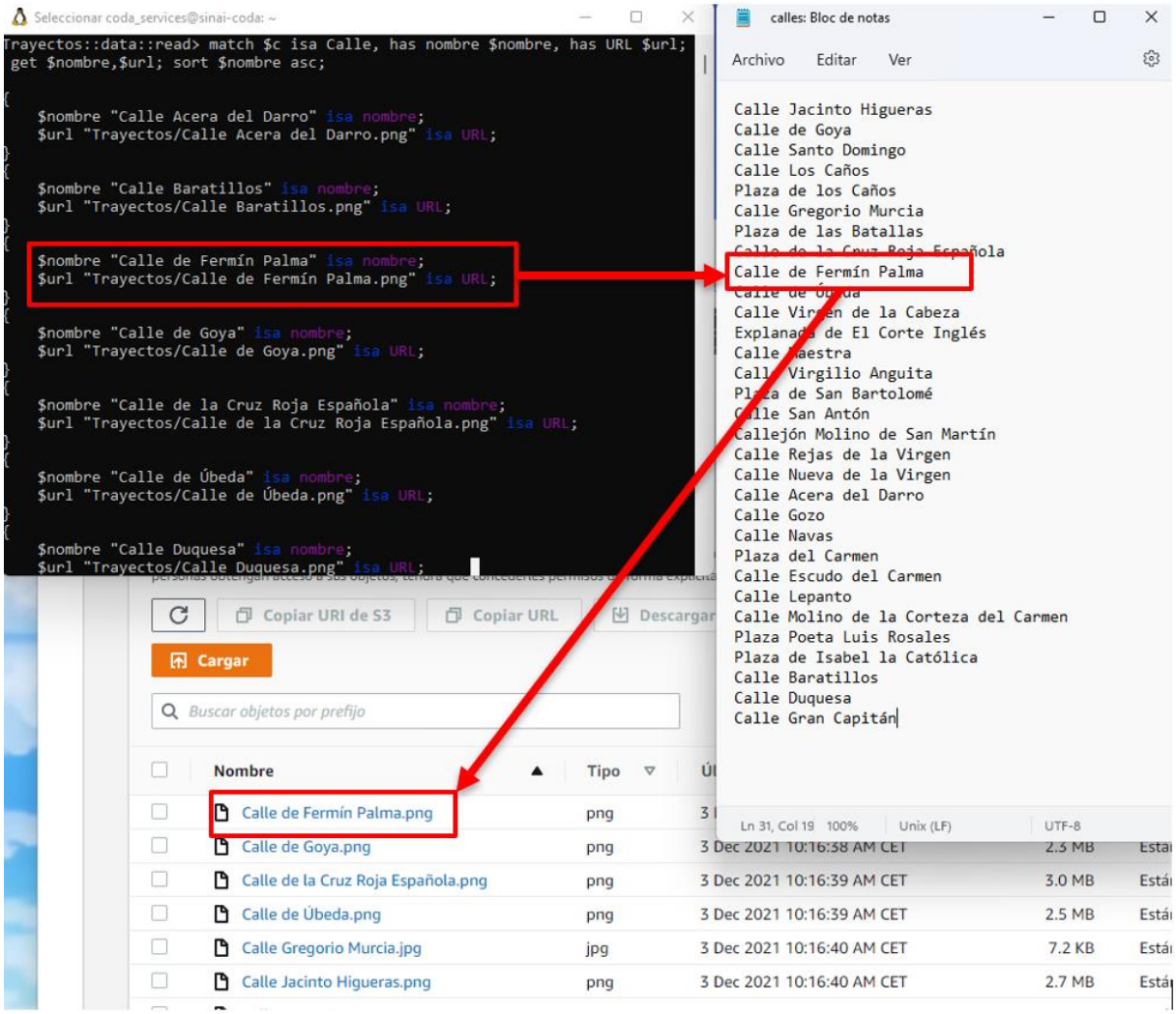


Ilustración 20. Concordancia de la url para la base de datos, fichero de salida y S3

Ahora que se puede conectar con la base de datos, se puede sustituir el recorrido de ejemplo mencionado en el apartado 4.1 por calles reales. En la Ilustración 21 se muestra un ejemplo de lo que ocurre al iniciar una partida y una respuesta del usuario correcta:

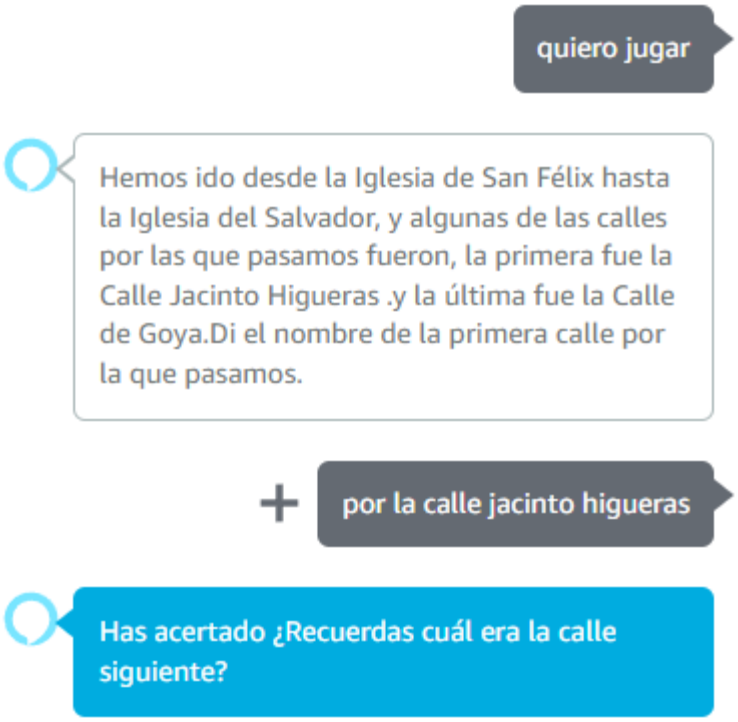


Ilustración 21. Transcripción de una interacción con Alexa

4.4. Cuarta iteración: APL

En este punto del desarrollo el juego es funcional en cualquier dispositivo de Alexa, ya que todos permiten interactuar por voz, sin embargo, algunos modelos disponen de una pantalla táctil. Por ello, la meta de esta iteración es añadir una nueva funcionalidad para aprovechar dicha característica con la librería APL (Alexa Presentation Language) incluida en ASK.

APL permite tanto controlar el contenido que se muestra en pantalla como responder a eventos generados por una pulsación del usuario en la pantalla. Con esto se puede mostrar (si el dispositivo es compatible) una rejilla con las imágenes de cada calle del recorrido con el que se está jugando, pero desordenadas.

Esta rejilla se crea en el editor gráfico de APL y se exporta en formato JSON a un fichero que estará en el mismo directorio que el resto de código de la Skill. Una vez dentro de la skill, solo hay que cargar dicho fichero y pasarle los argumentos necesarios, en este caso sería una lista con los enlaces de cada imagen y el nombre que acompañe a cada una. Estos enlaces son los que se almacenan en TypeDB junto

a cada calle, explicado en el apartado 4.3.4. En la Ilustración 22 se muestra este archivo:

Ilustración 22. Fichero JSON que compone la parrilla de imágenes en pantalla

En la Ilustración 23 se muestra el ejemplo del recorrido que dicta Alexa y lo que se muestra en la pantalla del dispositivo:

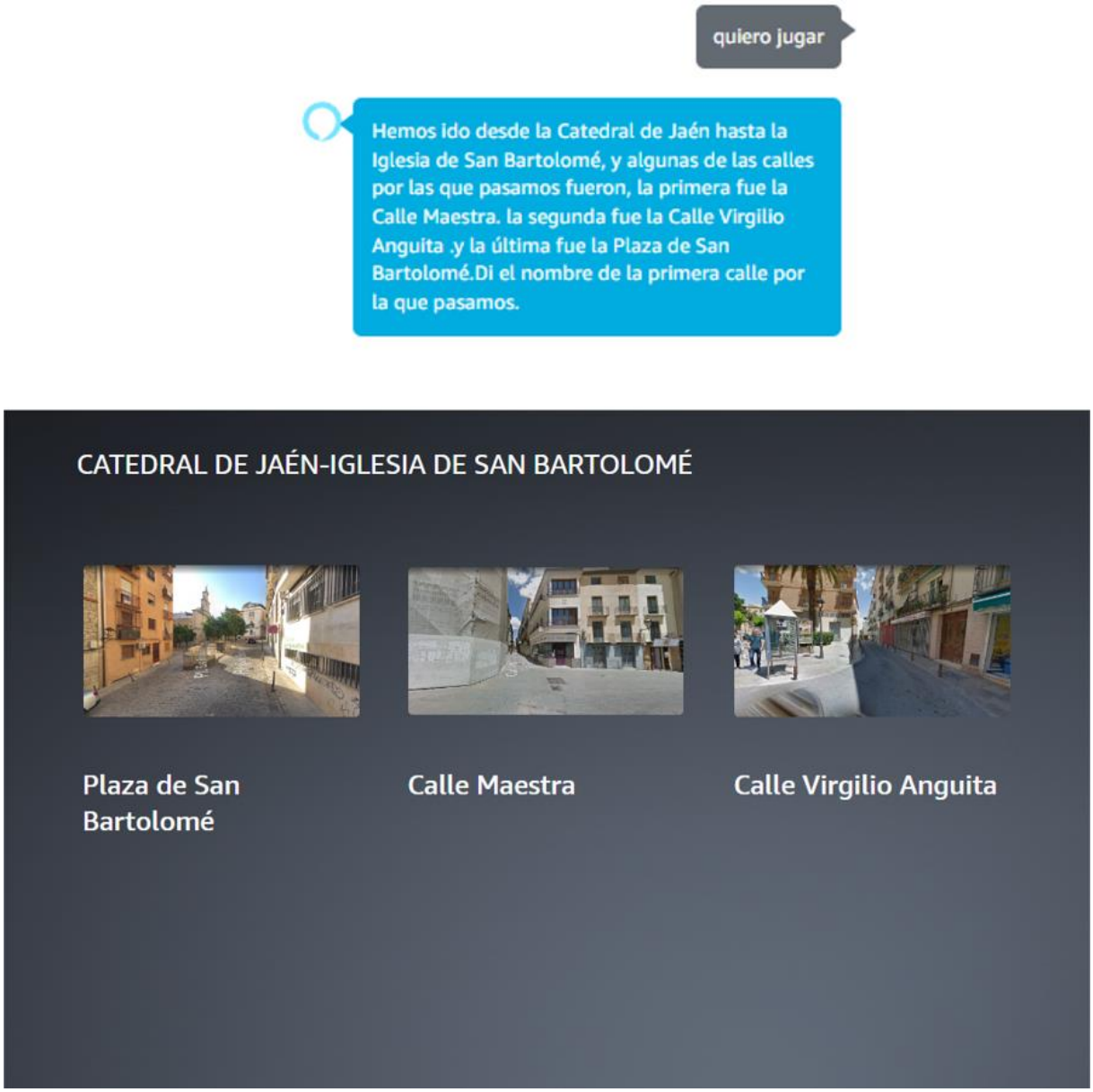


Ilustración 23. Muestra de la pantalla del dispositivo para un trayecto

4.5. Quinta iteración: Diseño de Codakoba

Esta iteración tiene como objetivo implementar el esquema para la base de datos. Tanto el esquema como los detalles del mismo están explicados en el apartado 3.4.13.4.2. Ahora solo queda trasladar el diseño lógico a un archivo de texto plano siguiendo la sintaxis propia de TypeDB e importarlo al servidor de AWS al igual que se hizo con la base de datos Trayectos.

4.6. Sexta iteración: Aplicación Java

En este punto del desarrollo, ya existe un juego con el que el mayor puede interactuar y generar información (Alexa) y un lugar donde almacenarla automáticamente (servidor con la base de datos). La última parte del sistema que falta es la aplicación con la que el profesional podrá recoger la información del mayor para el análisis de la misma. Para mayor compatibilidad y portabilidad, esta aplicación consiste únicamente en un ejecutable de java (GenerarDatos.jar).

El aspecto que tiene la ventana cuando se lanza la aplicación se muestra en la Ilustración 24:

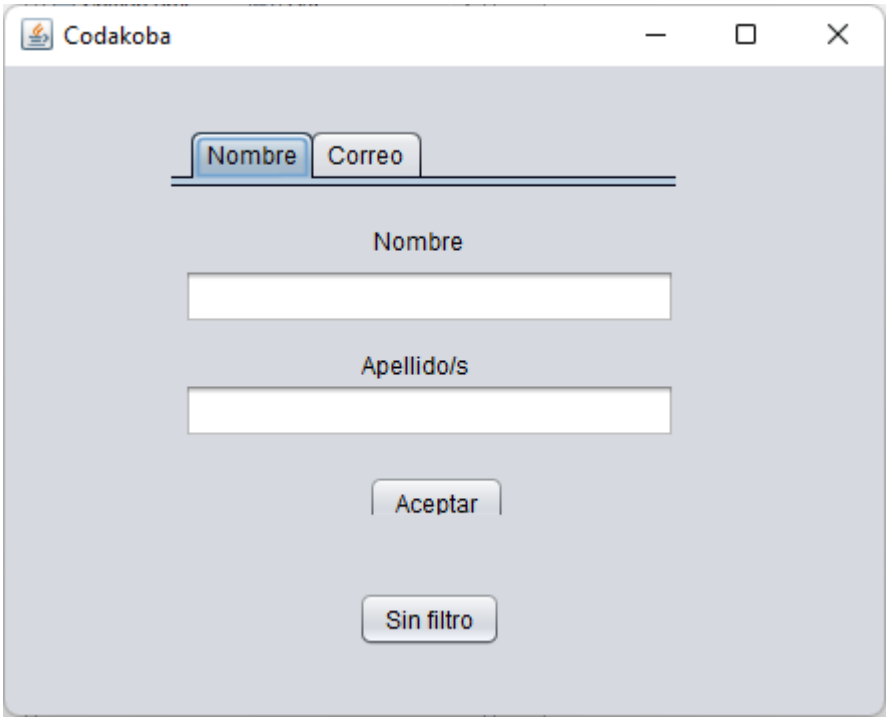


Ilustración 24. Ventana de la aplicación

Esta aplicación permite con una única ventana acceder a la información de 3 formas distintas:

- Filtrar por el nombre y apellidos del mayor (Ilustración 25).

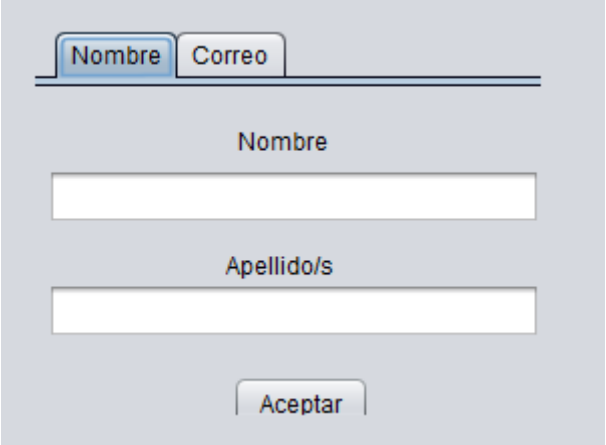


Ilustración 25. Pestaña de búsqueda por nombre y apellidos

- Filtrar por el correo electrónico con el que se registró (Ilustración 26).

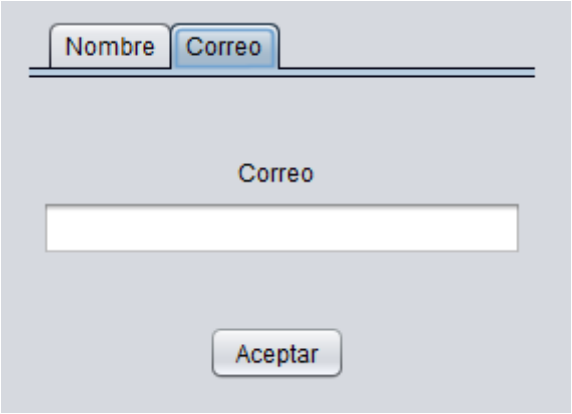


Ilustración 26. Pestaña de búsqueda por correo

- Sin filtrar, recogiendo los datos de todos los mayores registrados en el sistema (Ilustración 27).

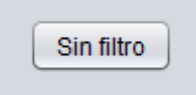


Ilustración 27. Botón de búsqueda sin filtro

A la hora de introducir los datos y en caso de que no se la búsqueda por filtro, es obligatorio que los campos requeridos no se dejen en blanco. En caso de que esto ocurra, se presenta una ventana de alerta indicando que tanto nombre y apellidos o el

correo electrónico son obligatorios (según el tipo de búsqueda). Las búsquedas sin filtro no tienen ninguna condición de error. Estas ventanas se pueden ver en la Ilustración 28:

Ilustración 28. Ventanas de alerta cuando algún campo de búsqueda se deja en blanco

Una vez que se ha hecho la consulta de los datos, pueden darse dos casos y en ambos se mostrará una venta emergente (El botón para aceptar la búsqueda estará desactivado hasta que la ventana emergente se cierre):

- Se han encontrado resultados para la búsqueda y se ha podido generar el fichero. También se mostrará la ruta al archivo con los datos, que será en el mismo directorio en el que se encuentre el ejecutable (Ilustración 29).

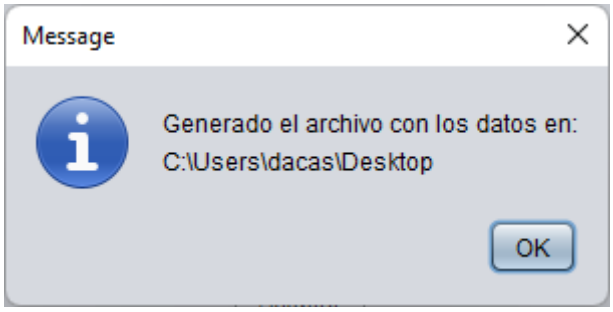


Ilustración 29. Ventana de confirmación cuando se encuentra al menos una entrada en la base de datos junto a la ruta donde se guardó el fichero

- No se han encontrado resultado y no se ha generado el archivo (Ilustración 30).

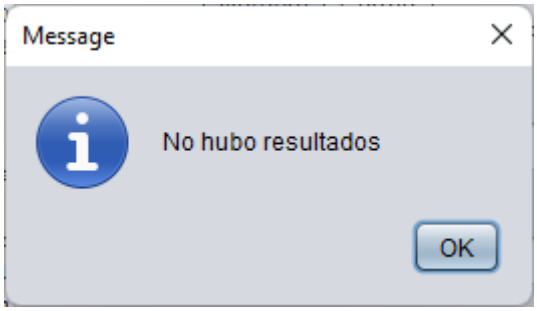


Ilustración 30. Ventana cuando no se encuentran resultados para los datos de búsqueda

El fichero generado tiene como nombre la fecha y la hora de creación concatenados (como se ve en la Ilustración 31), para que sea más fácil la organización, y el formato de los datos es el explicado en el punto sobre Codakoba.

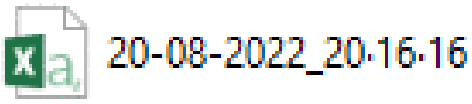


Ilustración 31. Nombre del fichero donde se almacenan los datos recogidos

5. Conclusiones y trabajo futuro

Este proyecto ha resultado en un desarrollo interesante. Está formado por varios módulos donde el reto ha sido identificar las ventajas y limitaciones de cada tecnología usada y la mejor forma de conectar la información entre ellas. De estas tecnologías cabe destacar el papel de ASK y TypeDB. Ambas son herramientas novedosas, pero han provocado problemas esperables al ser software que no ha alcanzado la madurez. Sin embargo, han sido las que han aportado las características más punteras y novedosas en su campo, aparte de tener aún mucho potencial por desarrollar. Estas tecnologías, y especialmente el uso de un altavoz inteligente como Alexa, son las que

han dado una personalidad propia al proyecto. Trabajar con TypeDB ha permitido introducir el concepto de grafos de conocimiento[3] e hipergrafos[5].

El software desarrollado ha sido la herramienta protagonista para un estudio piloto del que se hicieron pruebas a pequeña escala y del que se pueden extraer algunas conclusiones.

El problema principal que se ha descrito es relacionado con la aplicación JAVA. Si bien se pensó en este lenguaje por ser compatible con diversas plataformas, hemos tenido problemas en MAC para ejecutarlo al reconocerlo como software de orígenes desconocidos. Esto ha llevado a pensar que un posible cambio sea sustituir esta parte por una aplicación web (aprovechando el cliente de Node.js de TypeDB).

Por otro lado, este software también forma parte de un proyecto mayor llamado CODA[13] que continúa en desarrollo, de forma que tanto las bases de datos como el código de acceso a las mismas sean compatibles con otras skills y esta funcionalidad se pueda integrar en ellas de forma fácil.

Los objetivos que tiene el proyecto CODA se definen a continuación:

- Desarrollo y validación de un sistema de monitorización del estado cognitivo circunscrito al ámbito de la memoria de una persona mayor en un entorno doméstico simulado (SmartLab). Esto necesita un producto mínimamente viable que requiere tres elementos: una batería de juegos digitales, un agente conversacional y habitación que simule una sala de estar (SmartLab).
- Identificar una serie de marcadores sobre el indicio de deterioro cognitivo.

6. Anexos

6.1. TypeDB

TypeDB[2] es un sistema de base de datos fuertemente tipado que permite descomponer problemas complejos en un sistema lógico. TypeQL[4] (el lenguaje para el esquema) permite abstracciones sobre patrones de datos complejos y de bajo nivel.

Al combinar ambos, se consigue un sistema de definiciones y reglas muy parecido entre el lenguaje del dominio y el lenguaje interpretable por la base de datos.

TypeDB garantiza la integridad y la seguridad de los datos, a la vez que permite realizar inferencias a nivel de datos dentro de la propia base de datos. Este nuevo paradigma ofrece un mayor nivel de expresividad.

La columna vertebral de cualquier base de datos TypeDB es la representación de su dominio: el esquema. El esquema se compone de un conjunto de tipos y reglas, que aprovechan los principios orientados a objetos y la deducción lógica.

Los tipos definidos en su esquema están estructurados en tres categorías:

- **Entidad**: representan los objetos de su dominio.
- **Relación**: representan conexiones n-arias dentro de su dominio. Cuando se utiliza un tipo de relación para conectar tipos, cada tipo debe desempeñar un papel determinado. Esto se representa utilizando un tipo de rol, que proporciona contexto a sus conexiones
- **Atributo**: representan valores.

Utilizando estas sencillas construcciones, se puede construir un esquema de terminología y conocimientos específicos del dominio, de forma que también lo entienda TypeDB. Esto libera al desarrollador de la necesidad de pensar en términos de tablas, documentos o vértices/aristas, y le permite pensar en un nivel más alto de abstracción, en un lenguaje más cercano al natural. El diseño de modelos de datos y la gestión de datos a este nivel de abstracción es una de las principales ventajas de TypeQL y TypeDB.

Una base de datos está hecha de esquema y datos. Los tipos en el esquema se crean mediante subtipos, similar a los lenguajes de programación orientados a objetos. Cada base de datos viene con algunos tipos integrados: entidad, relación y atributo, que ampliará para crear su esquema.

Todos los elementos de datos existentes en la base de datos son instancias de sus tipos. Dado que los tipos son subtipos de entidad, relación, o atributo, decidir cuál

de ellos se utiliza al modelar los datos es una tarea importante a la hora de crear el esquema de TypeDB.

6.2. Alexa skill kit (ASK)

Alexa Skill Kit (ASK)[15] es un framework de desarrollo de software que permite crear contenido, llamado Skill. Las Skills son como las aplicaciones para Alexa. Con una interfaz de voz interactiva, Alexa ofrece a los usuarios una interacción de manos libres con la SKill. Los usuarios pueden usar su voz para realizar tareas cotidianas como revisar las noticias, escuchar música o jugar un juego. Los usuarios también pueden usar su voz para controlar dispositivos conectados a la nube. Por ejemplo, los usuarios pueden pedirle a Alexa que encienda las luces o apague la calefacción. Las Skills están disponibles en dispositivos habilitados para Alexa, como Amazon Echo y Amazon Fire TV, y en dispositivos habilitados para Alexa creados por otros fabricantes.

Un usuario accede al contenido de una Skill pidiéndole a Alexa que la invoque. Cuando un usuario dice la palabra de activación, "Alexa", y habla a un dispositivo habilitado para ello, el dispositivo transmite la frase al servicio en la nube. Alexa reconoce el discurso[1], determina lo que el usuario quiere y luego envía una solicitud para invocar la Skill que puede cumplir con la solicitud. El servidor de Alexa maneja el reconocimiento de voz y el procesamiento del lenguaje natural. Su Skill se ejecuta como un servicio en una plataforma en la nube. Alexa se comunica con su Skill mediante el uso de un mecanismo de solicitud-respuesta a través de la interfaz HTTPS. Cuando un usuario invoca una Skill de Alexa, esta recibe una solicitud POST que contiene un objeto JSON. El cuerpo de la solicitud contiene los parámetros necesarios para que su Skill reciba los parámetros que necesita, realice su lógica y, a continuación, genere una respuesta. De forma adicional, una Skill puede tener contenido complementario en forma de imágenes o contenido táctil.

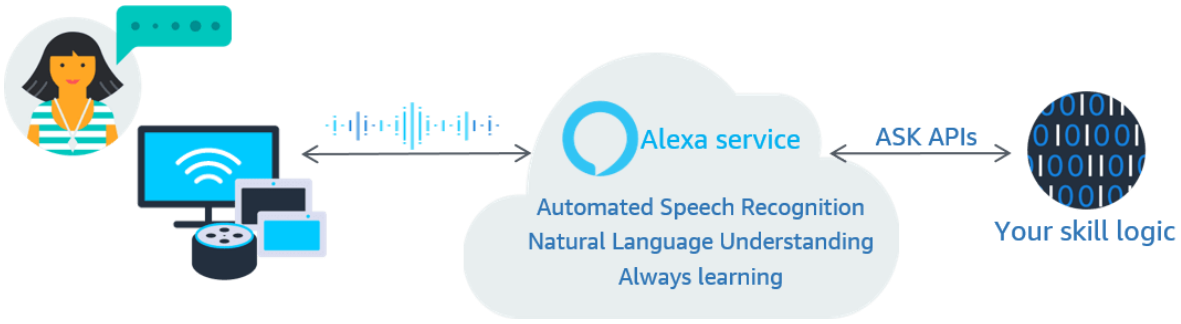


Ilustración 32. Diagrama que ilustra el flujo de funcionamiento en la nube de Alexa

Cada Skill de Alexa tiene un *modelo de interacción de voz* que define las palabras y frases que los usuarios pueden decir para indicar su intención. Este modelo determina cómo los usuarios se comunican y controlan su Skill. Una interfaz de usuario de voz es similar a una interfaz gráfica de usuario en una aplicación tradicional. En lugar de hacer clic en los botones y seleccionar opciones de los cuadros de diálogo, los usuarios realizan sus solicitudes y responden a las preguntas por voz. A menudo, la interacción de voz es de una duración mucho más corta que la interacción con una aplicación. Cuando un usuario hace preguntas y hace solicitudes, Alexa emplea el modelo de interacción para interpretar y traducir las palabras en una solicitud específica a la Skill identificada.

En la tabla 13 se compara una interacción por voz con una Skill con una interacción con una interfaz gráfica tradicional para hacer reservas de vuelos:

Acción	Interfaz de usuario de voz	interfaz gráfica de usuario tradicional
Hacer una solicitud	El usuario dice: "Alexa, quiero volar a Denver desde Seattle".	El usuario hace clic en la aplicación y, a continuación, selecciona los aeropuertos de origen y destino. El usuario se desplaza por la lista de aeropuertos para encontrar Seattle y, a continuación, se desplaza para encontrar Denver.

Recopilar más información del usuario	Alexa responde: "¿Cuándo te gustaría viajar?" y espera una respuesta.	La aplicación muestra un calendario y, a continuación, espera a que el usuario seleccione una fecha.
Proporcionar la información necesaria	El usuario responde: "1 de febrero". La habilidad hace la reserva y espera la confirmación.	El usuario abre el calendario, selecciona 1 de febrero y, a continuación, elige Aceptar. El usuario hace clic en un botón para completar la solicitud y, a continuación, espera la confirmación.
Solicitud completa	Alexa responde: "Su reserva de Seattle a Denver el lunes 1 de febrero está lista".	La aplicación muestra el resultado de la solicitud. El usuario cierra la aplicación.

Tabla 13. Comparación para reservar un vuelo mediante interacción voz frente a una interfaz gráfica

Alexa admite dos tipos de modelos de interacción de voz:

- **Modelo de interacción de voz predefinido**: en este modelo, ASK define el conjunto de palabras que los usuarios dicen para invocar una habilidad. Por ejemplo, un usuario puede decir: "Alexa, enciende la luz" o "Alexa, apaga la televisión".
- **Modelo de interacción de voz personalizado**: el modelo personalizado da más flexibilidad, pero es el más complejo. El desarrollador es quien diseña toda la interacción de voz. Con el modelo personalizado, normalmente se tienen que definir todas las formas en que un usuario puede expresar la misma intención a la Skill. Por ejemplo, "Alexa, planea un viaje de Seattle a Denver", "Alexa, quiero ir de viaje a Denver desde Seattle" y "Alexa, planea un viaje a Denver".

6.3. Memoria de trabajo

La memoria es el proceso cognitivo con el que se codifica, almacena y recupera información o sucesos específicos. La memoria se puede diferenciar en dos tipos:

- **Memoria explícita**: de acceso rápido, capacidad de ser verbalizada y modificable.
- **Memoria implícita**: no puede ser verbalizada y es más rígida que la anterior.

Dentro de la memoria explícita está la memoria de trabajo (u memoria operativa). Este tipo de memoria se especializa en retener información por periodos cortos y que necesita ser accedida de forma inmediata, por lo que constantemente se añade, elimina y sustituye información. Este tipo de memoria en la protagonista en el comportamiento consciente y las acciones inmediatas. Debido a esto es fundamental para las tareas cotidianas en la vida una persona.

7. Bibliografía

[1]Sistemas de diálogo. (s.f.). Obtenido de https://www.ugr.es/~rlopezc/sistemas_dialogo.htm (visitado el 06/09/2022)

[2]¿Qué es un esquema en TypeDB? (s.f.). Obtenido de https://docs.vaticle.com/docs/schema/overview (visitado el 06/09/2022)

[3]Grafos de conocimiento. (s.f.). Obtenido de https://www.grapheverywhere.com/knowledge-graph/ (visitado el 06/09/2022)

[4]TypeQL, lenguage de consulta de TypeDB. (s.f.). Obtenido de https://docs.vaticle.com/docs/query/overview (visitado el 06/09/2022)

[5]Hipergrafos en TypeDB (Szymon Klarman). Obtenido de https://blog.vaticle.com/modelling-data-with-hypergraphs-edff1e12edf0 (visitado el 06/09/2022)

[6]Creación de mapas con Folium. (s.f.). Obtenido de https://mappinggis.com/2018/10/folium-utilizando-leaflet-con-python/ (visitado el 06/09/2022)

[7]Disposición 5652 del BOE núm. 83 de 2022. (s.f.). Obtenido de https://www.boe.es/boe/dias/2022/04/07/pdfs/BOE-A-2022-5652.pdf (visitado el 22/08/2022)

[8]TIC, I. d. (s.f.). Obtenido de https://www.juntadeandalucia.es/export/drupaljda/DGTD_instruccion_perfiles_TIC.pdf (visitado el 22/08/2022)

[9]Ludificación o gamificación de actividades. (s.f.). Obtenido de https://www.educaciontrespuntocero.com/noticias/gamificacion-que-es-objetivos/ (visitado el 06/09/2022)

[10]Open Route Service. (s.f.). Obtenido de https://openrouteservice.org/ (visitado el 06/09/2022)

[11]Pillow para creación de imágenes en formato PNG. (s.f.). Obtenido de https://pypi.org/project/Pillow/ (visitado el 26/12/2021)

[12]Tabla de coeficientes de amortización lineal. (s.f.). Obtenido de https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/ (visitado el 22/08/2022)

[13]Martínez-Santiago, F.; García-Viedma, R.; Rueda-Ruíz, A.; Ureña-Cámara, M.; García-Fernández, Á.& Ureña-López, L. (2020). A Proposal to Improve Voice-based Interfaces for Elders using Daily-living Activity Identification. 6th International Conference on Information and Communication Technologies for Ageing Well and e-Health - TEG, Prague, Czech Republic, May 3-5, 2020

[14]¿Qué es Amazon Web Services? (s.f.). Obtenido de https://aws.amazon.com/es/what-is-aws/ (visitado el 06/09/2022)

[15]¿Qué es una Skill de Alexa? (s.f.). Obtenido de https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html (visitado el 06/09/2022)

[16] Licencias de uso de Open Route Service (s.f.) Obtenido de https://openrouteservice.org/plans/ (visitado el 27/12/2021)