Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Gestión de Emergencias por Coronavirus (COVID-19)

Coronavirus (COVID-19) Emergency Management

Alba Cruz Torres

La Laguna, 29 de mayo de 2020

Dª. **Pino Caballero Gil**, con N.I.F. 45.534.310-Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D. **Carlos B. Rosa Remedios**, con N.I.F. 43.786.084-H responsable de la Unidad de Tecnologías de la Información y la Comunicación de Gestión de Servicios para la Salud y Seguridad en Canarias, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

”*Gestión de Emergencias por Coronavirus (COVID-19)*”

ha sido realizada bajo su dirección por Dª. **Alba Cruz Torres**, con N.I.F. 79.073.834-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 29 de mayo de 2020

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutora, Pino Caballero Gil, y a mi cotutor, Carlos B. Rosa Remedios, por su especial atención, apoyo y orientación a través de cada una de las etapas de este proyecto.

También quiero agradecer a todos mis compañeros de la carrera, así como mis amigos más cercanos, quienes me han apoyado concienzudamente y han hecho mi camino más fácil y ameno de recorrer.

Por último, me gustaría agradecer a mi familia, por brindarme el mayor respaldo que he recibido durante toda mi vida. En especial, quiero mencionar a mi padre, quien siempre creyó en mí y sé que estaría plenamente orgulloso de lo que he sido capaz de conseguir.

Licencia

Resumen

Uno de los problemas principales al que la sociedad tiene que hacer frente hoy en día es la situación sanitaria tan grave que se ha extendido a nivel mundial en pocos meses, debido a un nuevo coronavirus ya denominado COVID-19. Esta crisis ha conducido a graves consecuencias tanto en toda la población como en los centros de emergencias y hospitalarios. Uno de los inconvenientes más trascendentales ha sido el colapso de las líneas de atención de emergencias del ciudadano, provocando tiempos de espera enormes y dando lugar así a pacientes o afectados por este virus que no llegan a ser atendidos. Herramientas de Inteligencia Artificial que realicen este tipo de trabajos, es decir, trabajos en los que el sentido humano gana mayor importancia, parecen apropiadas para ofrecer una solución a este problema. No obstante, aún hoy en día sigue siendo muy difícil emular completamente el comportamiento humano para ayudar, de esta manera, a disminuir el estrés que genera este tipo de situaciones.

Por este motivo, se investigaron otras formas de implementar una solución con las capacidades tecnológicas del momento que pudiera cumplir con estos objetivos. Por tanto, se ha construido y desarrollado una herramienta que ofrece una experiencia conversacional con los usuarios para evitar que los operadores de emergencias empleen más tiempo del que poseen en ejecutar el test de coronavirus a cada usuario por individual, ahorrando así tiempos de espera y la carga de trabajo de los mismos y dando lugar, en consecuencia, a una mejor gestión de estas emergencias. Este proyecto aborda el desarrollo de esta solución, así como analiza los resultados que se han obtenido de la investigación e implementación llevada a cabo.

Palabras clave: COVID-19, crisis sanitaria, emergencias, experiencia conversacional, afectados, seguridad.

Abstract

One of the main problems that society has to face nowadays is the so serious health situation that it has spread worldwide in a few months, due to a new coronavirus already called COVID-19. This crisis has led to serious consequences in the entire population and also in emergency and hospital centers. One of the most significant issues has been the collapse of the citizen's emergency hotlines, causing enormous waiting times and bringing about affected patients or people by this virus who do not get to be treated. Artificial Intelligence tools that carry out this type of work, that is, work in which the human sense gains greater importance, seem appropriate to offer a solution to this problem. However, even today it is still very difficult to completely emulate human behavior to help reducing the stress that this kind of situation generates.

For this reason, other ways of implementing a solution with the technological capabilities of the moment that could fulfill these objectives were investigated. Therefore, a tool that offers a conversational experience with users has been built and developed to prevent emergency operators from spending more time than the time they have in executing the coronavirus test individually to each user,producing savings in waiting times and their workload and consequently leading to better management of these emergencies. This project addresses the development of this solution, as well as discuss the results that have been obtained from the research and implementation accomplished.

Keywords: COVID-19, health crisis, emergencies, conversational experience, affected people, security.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

Desde sus orígenes, la tecnología ha experimentado una constante evolución y paulatina introducción en todos los ámbitos de la vida cotidiana, siendo así la responsable de multitud de progresos y avances que se han venido sucediendo en nuestra sociedad. A pesar de dichos progresos, se consideran inmensas las posibilidades que aún nos deparan los avances tecnológicos. Entre ellos, en el momento actual toma especial significación el desarrollo de la Inteligencia Artificial [1] que, de forma resumida, se puede enunciar como aquella combinación de algoritmos que pretende emular el comportamiento humano trasladando dicha inteligencia a los equipos.

El objetivo esencial de la Inteligencia Artificial, por lo tanto, reside en la consecución de una toma de decisiones autónoma por parte de los equipos. No obstante, la principal problemática radica en la codificación del comportamiento humano ya que, además de no contar con protocolos o estándares determinados para cada situación, existen ciertos rasgos humanos, tales como el sentido de la ética o la moral, así como el campo de los sentimientos que son difícilmente codificables [2].

Esta problemática se hace especialmente latente en el campo de estudio que ocupa el presente documento: las emergencias. Asimismo, y por añadidura a las dificultades inherentes a la codificación de la conducta humana, se observan otros inconvenientes en el citado campo. Tanto es así, que actualmente son las llamadas telefónicas el medio más rápido y eficaz para atender una emergencia. Dicha herramienta plantea posibles complicaciones, tales como la indisponibilidad de un número de operadores suficiente para la atención de las llamadas o el desbordamiento de las

líneas en aquellos casos en los que las emergencias afectan a un número significativo de personas.

Por todo lo anterior, el presente estudio aspira a optimizar los tiempos y medios de la atención de las emergencias a través del diseño de una aplicación que haga de intermediario entre el operador y el sujeto que sufre una urgencia o emergencia. Concretamente, el desarrollo de la propuesta se contextualizará y basará en uno de los mayores desafíos para la atención de las emergencias: la gestión de la COVID-19 [3]. Se trata, en definitiva, de un escenario propicio para el análisis del abordaje de una emergencia acuciante y comunitaria.

En definitiva, lo que se pretende es el empleo de los avances tecnológicos hoy en día disponibles para un eficiente y eficaz desempeño de todos los procesos implicados en la solicitud y asistencia en caso de emergencia, de forma que esto redunde en un mayor y satisfactorio estado de seguridad ciudadana.

1.2. Objetivos

El objetivo principal de este proyecto es el de conseguir realizar una herramienta capaz de contribuir a la disminución de la saturación de las líneas del Centro Coordinador de Emergencias y Seguridad (CECOES) 1-1-2 del Gobierno de Canarias [4] y de los tiempos de espera medios, de manera que a sus operadores sólo les lleguen los casos con más probabilidades de ser posibles contagios por coronavirus. Y que además disminuya el volumen de trabajo y el estrés que genera este tipo de situaciones a dichos trabajadores.

Asimismo, para poder cumplir plenamente este reto es necesario garantizar que la herramienta es, en primer lugar, capaz de filtrar las llamadas con mayores posibilidades de estar relacionadas con la COVID-19; por otro lado, es competente a la hora de disminuir el tiempo de las llamadas, de forma que la herramienta recolecte la mayor cantidad de información posible con el fin de facilitar el trabajo a los teleoperadores y, por último, está provista de un nivel de seguridad que respalde completamente la integridad de los datos sensibles que se manejen (véase Figura 1.1).

Figura 1.1: Objetivos específicos y general

En suma, se desea ofrecer una herramienta que sea capaz de aliviar esta situación de crisis, dando como resultado, en primer lugar, la reducción de la saturación de las líneas telefónicas del CECOES, para poder garantizar así una correcta asistencia a las solicitudes de los ciudadanos, además de la disminución del volumen de trabajo de los operadores de los centros. Del mismo modo, se quiere que la herramienta contribuya en el aumento del sentimiento de calma y tranquilidad de la población, reforzando la sensación seguridad de los ciudadanos, y que, por último, se consiga intensificar la confianza de la población en la competencia y gran capacidad de actuación de los servicios públicos.

1.3. Fases del proyecto

El desempeño de los objetivos propuestos ha requerido que el proyecto se divida en un total de seis fases (véase Figura 1.2), pudiendo a su vez distinguir un número específico de tareas en cada una de ellas. A continuación, se enuncian y explican brevemente las citadas fases:

1. Planificación, análisis y requisitos: En esta primera fase se ha analizado el problema que se quiere abordar y las soluciones similares existentes, dando lugar al establecimiento de los requisitos del proyecto. Se ha decidido crear una herramienta capaz de filtrar, por un lado, aquellos casos con mayor probabilidad de estar relacionados con la COVID-19 y, por otro lado, aquellos de carácter grave y urgente que requieran de una respuesta inmediata.

Para ello, se ha partido de una prueba rutinaria para clasificar los casos según las respuestas del sujeto. Las preguntas que han compuesto la

Figura 1.2: Fases del proyecto

prueba se han ido ajustando y adaptando obligatoriamente a las nuevas circunstancias y peculiaridades de la situación sanitaria.

2. Tecnologías y herramientas: Durante este período, se han estudiado las tecnologías que harían posible el desarrollo del trabajo cumpliendo con los requisitos establecidos y se han seleccionado las más adecuadas (véase Capítulo 3).

3. Diseño: Es esta la etapa en la que se han diseñado los bocetos de la parte visual de la aplicación, donde se muestra la información recogida con la herramienta. De igual manera, se ha llevado a cabo el diseño de la base de datos, habiendo hecho una investigación previa sobre la adecuación entre los diferentes tipos de bases de datos existentes y las características de la situación y de los datos objeto de tratamiento (véase Capítulo 4, sección 4.1).

4. Implementación: A lo largo de esta fase se ha hecho frente al desarrollo y escritura del código necesario para implementar tanto el Back-end como el Front-end del proyecto, así como los errores encontrados durante la implementación (véase Capítulo 4, secciones 4.2 y 4.3)

5. Testeo de funcionalidades: Este apartado se ha mantenido en constante funcionamiento a lo largo de todas las fases del proyecto, incluyendo

su creación, desarrollo y finalización. El testeo ha sido indispensable para la correcta comprobación de funciones y detección de errores.

6. Posibles mejoras: La dedicación de este período ha versado sobre la indagación, investigación y estudio de nuevas ideas que pudieran aportar mayor calidad y mejor funcionamiento a la herramienta (véase Capítulo 6).

1.4. Estructura de la memoria

La presente memoria se divide en tres grandes bloques: introducción, implementación y desarrollo, y conclusiones. Cada uno de estos bloques agrupa, a su vez, un conjunto de capítulos. En aras de facilitar la comprensión del documento y de su estructura se propone la tabla que aparece en la Figura 1.3.

ESTRUCTURA DE LA MEMORIA
1.- INTRODUCCIÓN DE LA MEMORIA
Capítulo 1. Introducción
Capítulo 2. Aproximación a la aplicación
Capítulo 3. Tecnologías
2.- IMPLEMENTACIÓN Y DESARROLLO
Capítulo 4. Implementación
Capítulo 5. Presupuesto
3.- CONCLUSIONES
Capítulo 6. Conclusiones y líneas futuras
Capítulo 7. Summary and conclusions
Apéndice A: Código del proyecto

Figura 1.3: Estructura de la memoria

El primer bloque de la memoria engloba la introducción del proyecto. En ella se abordan la motivación del estudio, la definición del problema que se desea resolver, los objetivos que se necesita cumplir y una aproximación a la propia aplicación resultante. Además, después de haber efectuado una exhaustiva investigación se detectan aquellas otras soluciones similares disponibles en el mercado y se realiza un recorrido a través de las tecnologías implementadas en el desarrollo del proyecto, así como los motivos de su elección. Dentro del primer bloque de introducción se contienen los capítulos 1, 2 y 3, correspondientes a la introducción, la aproximación a la aplicación y las tecnologías.

El segundo bloque de la memoria está compuesto por los capítulos 4 y 5, referidos al diseño e implementación llevados a cabo para generar la herramienta deseada y al presupuesto requerido para construir la aplicación. Por un lado, se advierten las fases implicadas en el diseño tanto de la base de datos como del apartado visual de la herramienta. Se observan, asimismo, los bocetos y mockups ideados al igual que su evolución. Por otro lado, se detalla el proceso de desarrollo del proyecto, incluyendo aquellos fragmentos de código que se han estimado más interesantes, sus respectivas explicaciones y las principales dificultades a las que se ha hecho frente en el avance de dicho proyecto. Finalmente, se expone el cálculo del presupuesto requerido para el desenvolvimiento del proyecto.

Por último, se encuentra el bloque en el que se realiza una valoración del resultado final y de los caminos que se han seguido para llegar hasta él. De la misma manera, se abordan las posibles mejoras e implementaciones que se podrían añadir, con el fin de dar lugar a una aplicación más adecuada y adaptada a los requerimientos de ese instante, así como las posibles rutas que podría tomar su evolución, teniendo en consideración la situación tecnológica del momento, los recursos disponibles y, sobre todo, las necesidades de la empresa para la que se está trabajando. Este bloque está compuesto por los capítulos 6 y 7, en el que se añaden unas conclusiones y líneas futuras también en inglés.

Finalmente, se ha añadido un apéndice a este último bloque que contiene los links a los repositorios de GitHub en los que se ha alojado el código para implementar cada una de las fases del desarrollo de esta aplicación.

Capítulo 2

Aproximación a la aplicación

2.1. Problemática

Hasta ahora, en las Islas Canarias se han gestionado las emergencias por medio de llamadas telefónicas. Normalmente, hay alrededor de tres o cuatro teleoperadores que se encargan de contestar las llamadas en primera estancia, indagar y recabar la información necesaria y establecer a qué sector derivar dicha llamada, de forma que este se encargue de gestionar los recursos necesarios para ponerle solución. En el CECOES se trabaja con un sistema de colas. Esto quiere decir que, si en el momento de realizar una llamada todos los operadores están ocupados, se pondrá en espera hasta que un teleoperador quede libre. Esto ha resultado ser bastante eficaz durante muchos años.

Dentro de la gestión llevada a la práctica en los centros de emergencias hasta el momento, en un día normal de trabajo se reciben en torno a 5.000 llamadas. No obstante, cuando ocurre un incidente no rutinario, como por ejemplo un terremoto o un apagón en una isla al completo, esa cantidad de llamadas aumenta en un porcentaje muy alto. Esto quiere decir que, aun incrementando el número de teleoperadores, sería casi imposible atender todas esas llamadas en el momento exacto en el que se realizan. Como se ha nombrado anteriormente, al recibir tantas llamadas, las que no puedan ser atendidas en ese momento se pondrán en espera. Y aquí surge el problema que atañe a este proyecto: ¿Qué pasaría si una de esas llamadas en espera corresponde a una persona sufriendo un infarto? ¿Cuáles son las probabilidades de que esa persona sobreviva si no es atendida lo más rápido posible?

En el caso del coronavirus, se están registrando tiempos de espera de veinticinco minutos o más. Hay incluso llamadas que no llegan a ser

contestadas y, por tanto, nunca son atendidas. Esto hace que las personas que requieren dicho servicio, al ver que no responden a su llamada, decidan acudir a los hospitales poniendo en riesgo a los demás e incluso a ellos mismos y desencadenando así un aumento en la saturación de los centros sanitarios.

A causa de que este virus está ocasionando la saturación de los centros hospitalarios, así como de las líneas de emergencias que tratan de asistir a todos los afectados, se han detectado en España, a fecha 29/05/2020, 237.906 contagios y alrededor 27.119 fallecidos [5]. Mientras, el archipiélago canario acumula 2.337 casos positivos y 160 fallecidos [6]. Debido a dicha situación, el CECOES ha visto sus líneas tan desbordadas, que han tenido que habilitar otra línea dedicada sólo a la COVID-19 y, aun así, siguen desbordados.

2.2. Aplicaciones similares

Al tratarse de una situación sanitaria totalmente nueva, hay muy pocas aplicaciones similares conocidas. Sin embargo, muchas entidades y trabajadores autónomos han querido apoyar la causa e intentar ayudar con alguna herramienta creada en un tiempo récord.

Este proyecto en concreto está basado en un chatbot llamado Health Bot Service de Microsoft Azure [7], en el que se adaptó a la COVID-19 una de sus plantillas disponibles. En ella se hace directamente una encuesta a los ciudadanos para filtrar quiénes son posibles casos de contagio. Durante la encuesta, se realizan preguntas de carácter excluyente, ya que según las respuestas se establece si es un posible caso de contagio o no y para hacer esto posible, se usa un diagrama de flujo que representa los pasos a seguir o los caminos a tomar de acuerdo con dichas respuestas (véase Figura 2.1). Muchas empresas e instituciones han hecho uso ya de este tipo de herramientas, como por ejemplo el Providence Health & Services [8], que es un sistema de atención médica sin ánimo de lucro. Fue una de las primeras empresas que incorporó el Health Bot Service a su página web para poder orientar así a los ciudadanos estadounidenses y ofrecerles pautas de cómo deben proceder según cada situación personal (véase Figura 2.2).

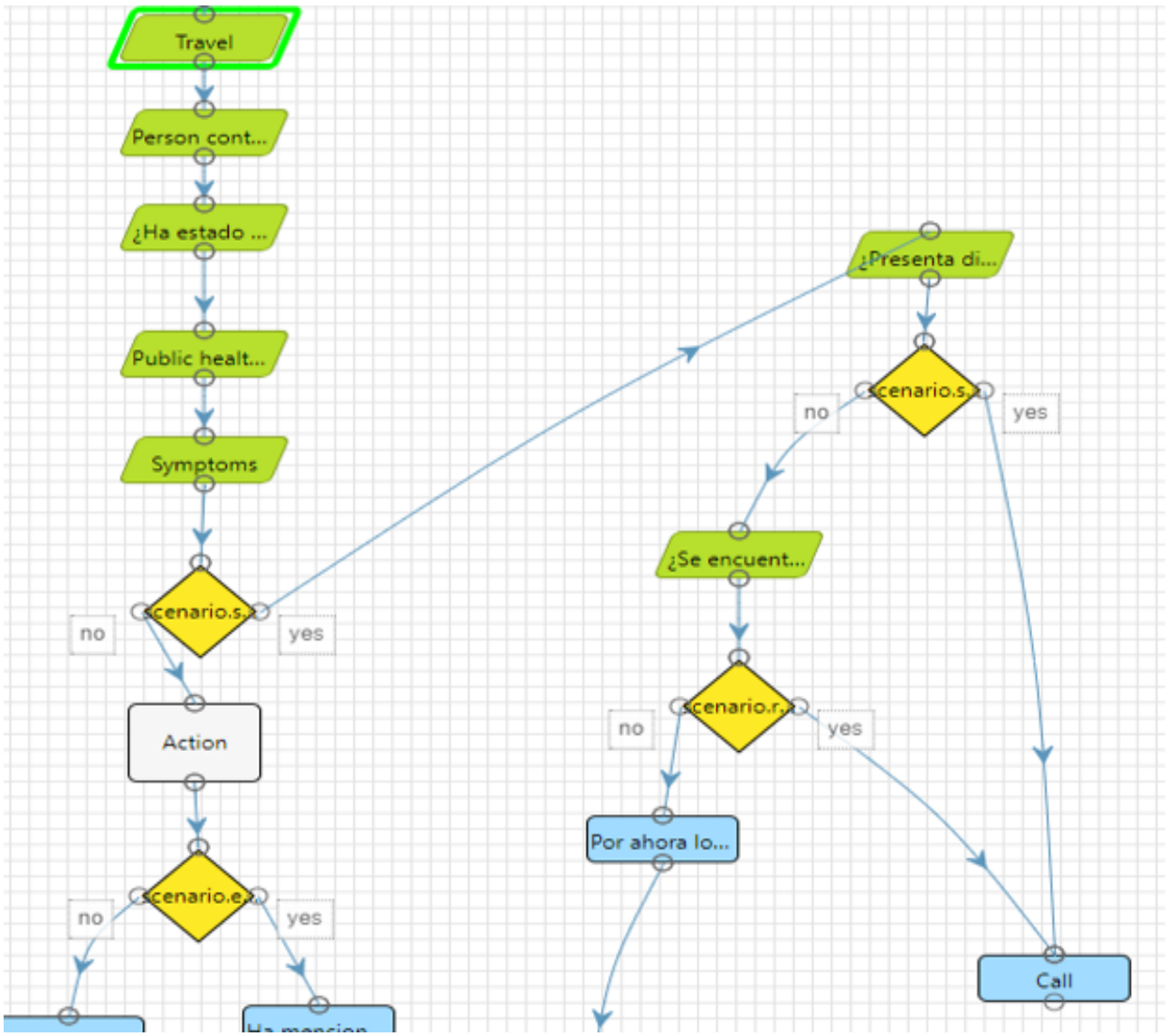

Figura 2.1: Diagrama de flujo de Health Bot Service de Microsoft

Desde otro ángulo, cabe destacar que ha ganado gran importancia la alianza que se ha establecido entre Google y Apple para el desarrollo de una tecnología de seguimiento de contactos [9]. Esta técnica será usada por las autoridades del sector sanitario para cuantificar y desacelerar el contagio. Para que la técnica pueda funcionar, se necesita que las personas infectadas den un aviso sobre las personas con las que han mantenido contacto y que, por consiguiente, les avisen sobre las medidas de salud a tomar y la realización de pruebas. A todo este proceso se le ha denominado como técnica de "*contact racing*"[10].

Esta tecnología funcionará a través de comunicación Bluetooth entre teléfonos iOS y Android. Por tanto, los usuarios deberán activar dicha función para formar parte del programa. Inmediatamente, el dispositivo efectuará el envío de una señal (*beacon*) con un número identificador de

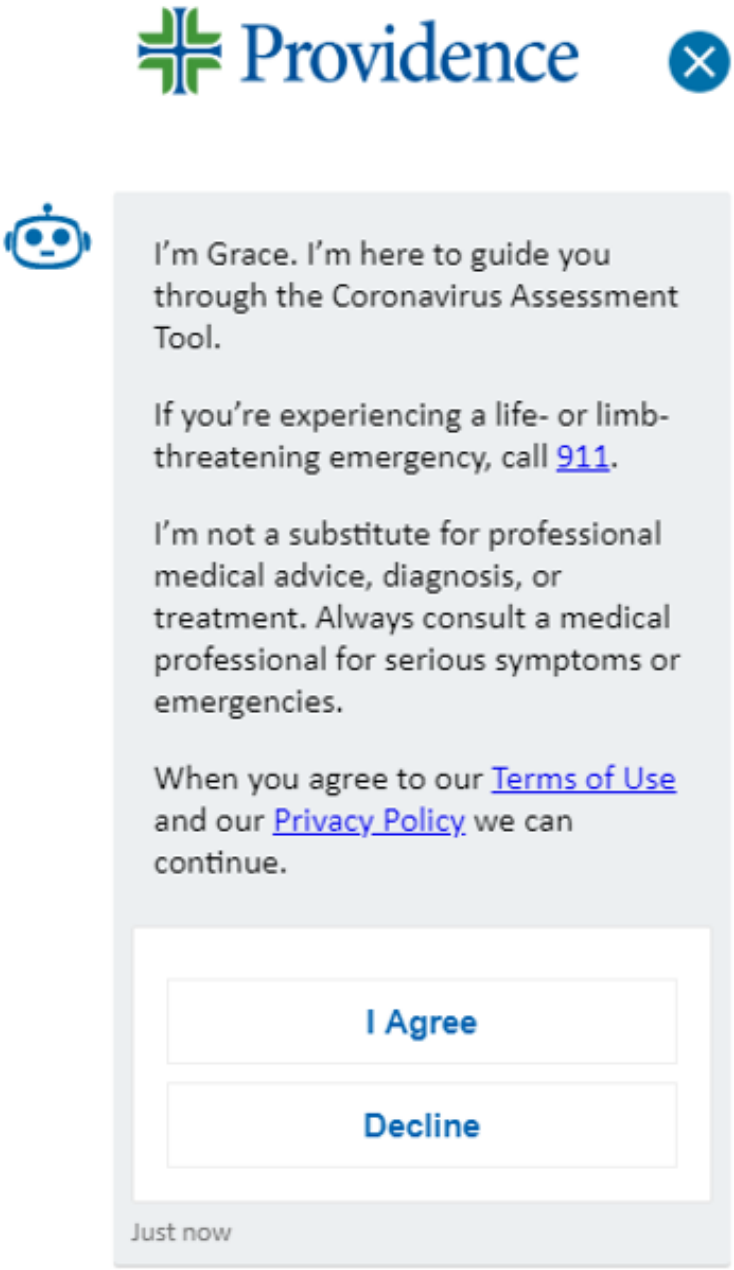

Figura 2.2: Chatbot COVID-19 de Providence.org

cada usuario y sin incluir la información personal del mismo, ya que lo que se desea ante todo es garantizar la privacidad y seguridad de los datos. Por ello, tanto Google como Apple modificarán el tiempo de exposición vía Bluetooth a un máximo de 30 minutos y, adicionalmente, los metadatos enviados estarán completamente cifrados con el fin de evitar cualquier tipo de ataque.

De la misma manera, en España se hallan algunas aplicaciones similares como la desarrollada por el Gobierno de España: Asistencia COVID-19 [11]. Se trata de una aplicación tanto web como móvil que da servicio a

las comunidades autónomas de Canarias, Cantabria, Castilla-La Mancha, Extremadura e Islas Baleares y, por añadidura, ofrece otras propuestas para las comunidades autónomas que se ven excluidas del listado anterior (véase Figura 2.3).

Gracias al uso de esta aplicación los ciudadanos españoles pueden recibir pautas y aclaraciones a seguir dependiendo de su estado de salud y, de igual manera, mantenerse informados de cualquier cambio en su condición sanitaria, ya que les permite realizar el test cada 24 horas. Una vez realizada la encuesta en la que se formulan preguntas tanto de carácter personal como sanitario, se lleva a cabo una evaluación de las respuestas y se ofrecen tanto los resultados como las recomendaciones a seguir.

Figura 2.3: Aplicación COVID-19 del Gobierno de España

Por otro lado, en Canarias se ha ideado un sistema para anticiparse a un incremento de los contagios [12]. Se trata de un método que alerta de un brote del virus fundamentado en las llamadas efectuadas a la línea del teléfono COVID habilitado por el Servicio Canario de Salud. Su modo de operar está basado en la clasificación de las llamadas que se reciben en tres grupos: por un lado, el número de personas que llama cada día; por otro lado, cuántas de ellas requirieron atención sanitaria y, por último, a cuántas personas citaron con su médico de cabecera.

Con esos datos recolectados en tiempo real, se estudia si ese número tiene alguna correlación estadística con los casos positivos que empiezan a presentar síntomas ese mismo día (véase Figura 2.4). Como resultado se tiene una información que les produce un margen de cinco días de preeminencia a los encargados de la situación sanitaria presente. De este modo se da lugar a la previa preparación de los dispositivos sanitarios necesarios para cada momento, pudiendo adelantarse a la situación para sobrellevarla de la manera más eficiente posible.

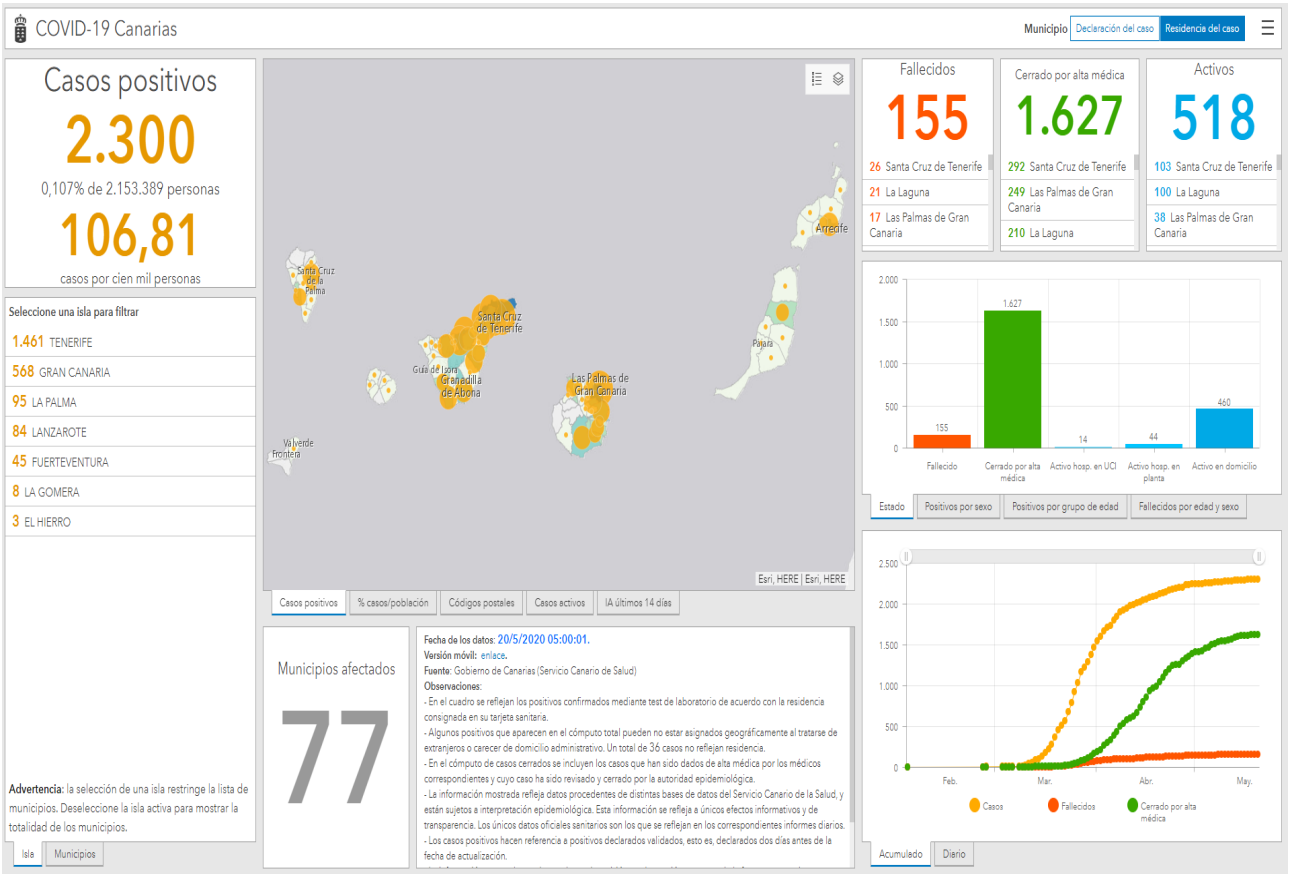

Figura 2.4: Situación en Canarias por COVID-19

2.3. Conceptualización de la propuesta

Después de haber llevado a cabo una investigación de a qué problemática se quería hacer frente y una vez establecidos los objetivos que se quería alcanzar, se ha procedido a la conceptualización del proyecto. Para hacer esto posible, se ha razonado y asimilado todos los componentes que rodean al problema mencionado y se ha determinado así, cómo interrelacionar cada una de las partes.

Una vez generado el desarrollo lógico de las ideas, se ha producido un esquema en el que se muestra claramente el ciclo de vida de la herramienta y la estructura principal que la compone (véase Figura 2.5). Dicho ciclo de vida está compuesto por tres fases:

Figura 2.5: Conceptualización de la propuesta

Efectuar test y evaluar respuestas: En esta fase se realiza la encuesta al sujeto que efectúa la llamada a la vez que se evalúan las respuestas para filtrar así los casos con más probabilidades de ser contagios por COVID-19 o los casos más críticos.

Tratar datos: Durante esta fase se guardarán los datos recolectados en la base de datos, teniendo especial cuidado y manteniendo siempre la seguridad de los datos con carácter personal del sujeto.

Mostrar resultados: En esta última fase se expone a los teleoperadores que manejarán la situación los resultados recogidos en la encuesta junto con los datos personales del sujeto.

Capítulo 3

Tecnologías

Con el propósito de implementar correctamente la aplicación y de garantizar el completo cumplimiento de los requisitos necesarios para hacerla funcionar plenamente, se ha hecho una selección de las tecnologías más adecuadas para efectuar la nombrada tarea. Haciendo énfasis en el ciclo de vida de la aplicación, se ha compuesto un diagrama (véase Figura 3.1) en el que se muestran las fases por las que transita la herramienta, además de las tecnologías que participan en cada una de ellas y aquellas necesarias para hacer posible la conexión entre dichas fases.

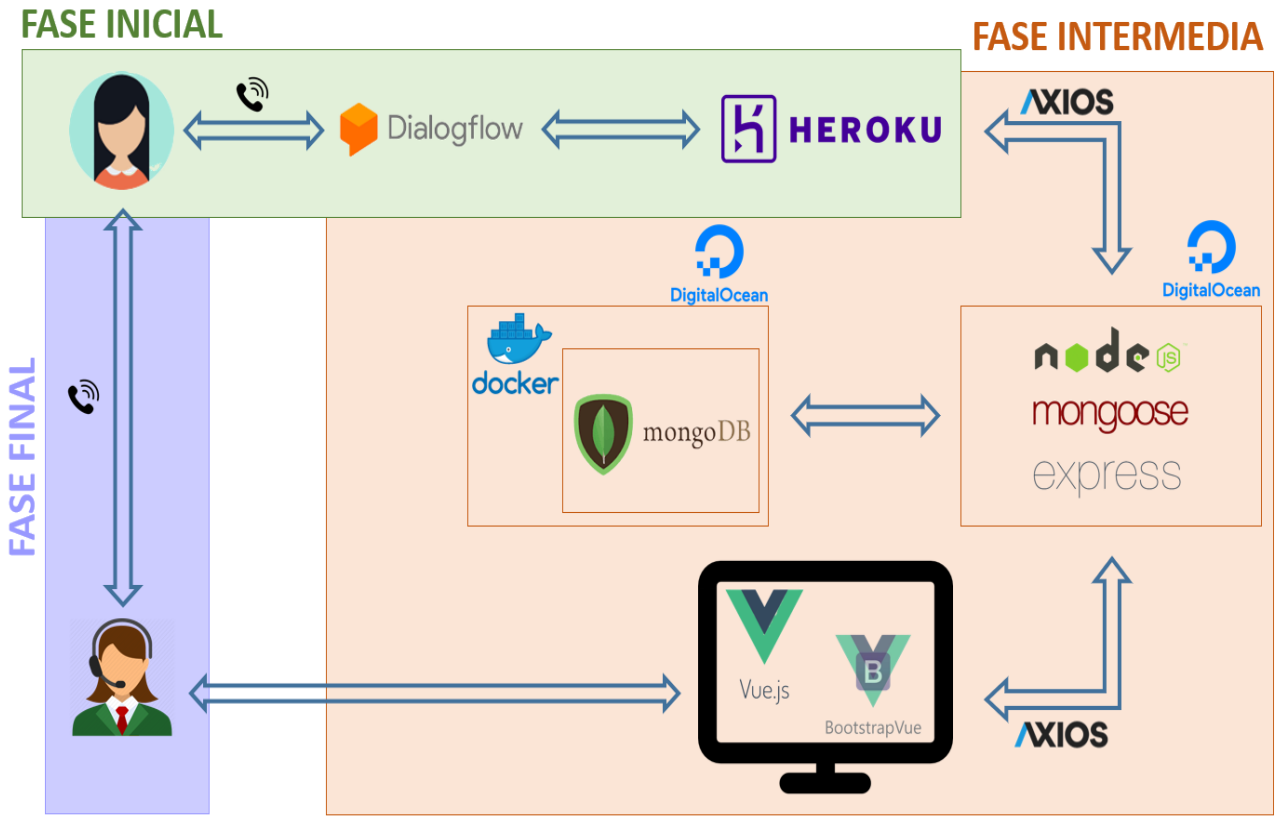

Figura 3.1: Tecnologías implicadas en cada fase y sus conexiones

3.1. Fase inicial

Durante la primera fase se inicia el proceso de llamada y se resuelve la encuesta haciendo uso de Dialogflow [13], una herramienta de Google para la creación de Chatbots y gracias a la cual se ha creado y construido la experiencia conversacional con el usuario. Para ello, se ha instanciado un agente virtual que permite manejar las conversaciones con los usuarios, ya que entiende el lenguaje natural del ser humano. El flujo básico de conversación entre un usuario y el agente virtual de Dialogflow es el siguiente: en primer lugar, el usuario ofrece un input, que puede ser un comando de voz, una petición o simplemente una pregunta. Acto seguido, el agente extrae cada uno de los parámetros de dicho input para trabajar a partir de ellos. Por último, el agente devuelve una respuesta que se corresponde con el input y que está previamente programada.

Con el objeto de conectar Dialogflow con el código implementado y de tal forma que se recojan los datos respondidos por el usuario, se ha hecho uso de Heroku [14] para alojar el código, una plataforma como servicio (PaaS) que provee un servicio de computación en la nube y que, adicionalmente, proporciona la transmisión continua de los registros más recientes, teniendo así un mayor control del comportamiento del código con Dialogflow. Es decir, cada vez que un usuario pone en marcha una conversación con el agente virtual, se puede seguir el flujo de la misma en todo momento, de tal forma que se puedan detectar errores o conductas no deseadas.

3.2. Fase intermedia

Con el fin de cumplir el objetivo de la herramienta, se ha implementado una aplicación NodeJS [15] con ExpressJS [16] para generar una API REST [18] con la que poder hacer las peticiones básicas a la base de datos. Pero, ¿qué es Express? Pues se trata de un framework de JavaScript [17] que ofrece una infraestructura de aplicaciones web la cual proporciona un conjunto sólido de características para las aplicaciones web y móviles. Mediante esta API creada, podremos efectuar métodos GET, POST, PUT y DELETE, es decir, tendremos la posibilidad de leer los datos de las llamadas y los sujetos; añadir nuevas llamadas así como el sujeto correspondiente a esta; actualizar cualquier fallo o equívoco en la introducción de los datos y, por último, borrar las llamadas y los datos de carácter personal.

Por lo tanto, una vez terminada la llamada y recolectadas las respuestas y datos del ciudadano, se estudia si se trata de un caso que requiere de una atención más inmediata o prioritaria y, si es así, se da comienzo al proceso de tratamiento de datos. Para almacenar dichos datos se ha usado un sistema de base de datos NoSQL [19] llamado MongoDB [20], el cual almacena los datos como documentos que tienen una estructura similar a las estructuras JSON. Luego, esos documentos se colocan dentro de colecciones, que no son más que nuestras tablas de llamadas y sujetos. Por último, para que la API sea capaz de manejar y trabajar con la base de datos de manera más sencilla, se ha utilizado MongooseJS [21], un ODM (*Object Data Modeling*) con el que se pueden definir objetos a través de esquemas asignados a los documentos de MongoDB, gracias a que ofrece una gran variedad de funcionalidades para crear y trabajar con esquemas de MongoDB, es decir, proporciona una capa de abstracción para efectuar las consultas que se desea hacer a la base de datos.

Ahora bien, ¿cómo se garantiza la integridad y seguridad de los datos personales? En este caso, la aplicación incorpora un paquete npm llamado mongoose-field-encryption [22], que utiliza el algoritmo AES con el fin de, por un lado, encriptar los datos cada vez que se añaden a la base de datos y, por otro lado, desencriptarlos siempre que se hagan peticiones o llamadas que los requieran. Dicha encriptación se efectúa por medio de una clave secreta y una sal o conjunto de bits aleatorios o generados a través de la función *saltGenerator()*. La sal generada y el valor encriptado resultante se concatenan usando el símbolo : y se muestran de esta manera en la base de datos.

Por otra parte, se ha utilizado DigitalOcean [23] para alojar la aplicación NodeJS nombrada, pues se trata de un proveedor de servidores virtuales privado que incentiva a un ahorro del tiempo y de la dedicación al código, ya que se encarga él mismo de la creación de la infraestructura, es decir, tiene como propósito ofrecer una arquitectura *serverless*. De igual manera, se ha ubicado en esta herramienta el sistema de base de datos que se ha utilizado en la aplicación. No obstante, este último se ha creado dentro de un contenedor de Docker [24], que es una tecnología que automatiza el despliegue de aplicaciones dentro de contenedores de software, suministrando una capa adicional de abstracción y automatización para modularizar una aplicación sin verse condicionada por el sistema operativo.

Finalmente, para hacer todo esto posible, es decir, la conexión de todas las herramientas nombradas con la base de datos, se ha utilizado Axios [25], una librería de JavaScript capaz de ejecutarse tanto en el navegador como en NodeJS para facilitar todo tipo de operaciones HTTP. Dicha librería también es utilizada en el desarrollo de la sección web para poder ejecutar las llamadas a cada una de las rutas especificadas para llevar a cabo las distintas peticiones a la base de datos.

3.3. Fase final

En esta última fase, se muestra la información recolectada al operador correspondiente. Por lo tanto, para desarrollar la parte visual de la aplicación se ha hecho uso de VueJS [26] que es un marco de JavaScript que permite la construcción de interfaces de usuario de una forma muy sencilla. Se ha elegido este framework ya que permite la implementación de infinidad de funcionalidades de manera creativa y nada restrictiva, además de que su curva de aprendizaje es muy sencilla. VueJS está caracterizado por el trabajo con componentes, que son elementos en los cuales se encapsula código reutilizable y en los que encontramos código HTML, JavaScript y estilos de CSS. La idea es atomizar cada uno de los aspectos existentes en la página web, de tal forma que cada componente se corresponda con una funcionalidad lo más simple posible y que las vistas se compongan luego de un conjunto de estos componentes creados.

Por último y para dar lugar a un mejor diseño de la aplicación, se ha integrado en el código un framework llamado BootStrapVue [27], que sirve para construir proyectos web responsivos combinando las características y beneficios de VueJS junto con las singularidades de uno de los frameworks más populares del mundo denominado Bootstrap [28], que es una librería multiplataforma que contiene plantillas con multitud de elementos de diseño basados en HTML y CSS, así como extensiones de JavaScript adicionales. Gracias a esta combinación se da lugar al Front-end de la aplicación, cumpliendo con los requisitos y patrones de diseño y se consigue también cumplir con uno de los objetivos más importantes de la sección visual de la aplicación, es decir, expresar de manera clara y concisa los datos recolectados.

Capítulo 4

Implementación

Después de haber efectuado un recorrido a lo largo del proyecto, dilucidando claramente cuál es su motivación principal y los objetivos que quiere alcanzar, es hora de explicar tanto el proceso de diseño de la aplicación como el proceso de desarrollo del código y los problemas encontrados durante su evolución. Para ello, se dividirá el siguiente capítulo en cuatro secciones principales: Diseño, Back-end, Front-end y Problemas encontrados.

4.1. Diseño

Durante esta fase de diseño se ha llevado a cabo el proceso de conceptualización visual del espacio y de la estructura de la aplicación, dando lugar así a los bocetos y diagramas que se utilizarán más adelante como modelo para la construcción de una aplicación usable y accesible. En este proyecto se ha necesitado efectuar el diseño tanto de la base de datos y de la conexión entre sus tablas como el diseño de la parte más visual o la interfaz de usuario, que en este caso serán los teleoperadores del CECOES.

En primer lugar, se ha hecho frente a la representación conceptual de la base de datos. Por este motivo, se analizaron los datos que se iban a tratar y cuál era la mejor manera de almacenarlos. En el caso que atañe a este proyecto, se llegó a la conclusión de que sólo serían necesarias dos tablas, una para las respuestas de la encuesta y otra para los datos personales del sujeto. A continuación, se ha procedido al trazado del diagrama que representará las conexiones entre las distintas colecciones que conforman la base de datos, así como los registros que las componen (véase Figura 4.1). Estos registros se corresponden con los resultados de la encuesta relacionada con la COVID-19 por un lado y, los datos personales del sujeto, por otro.

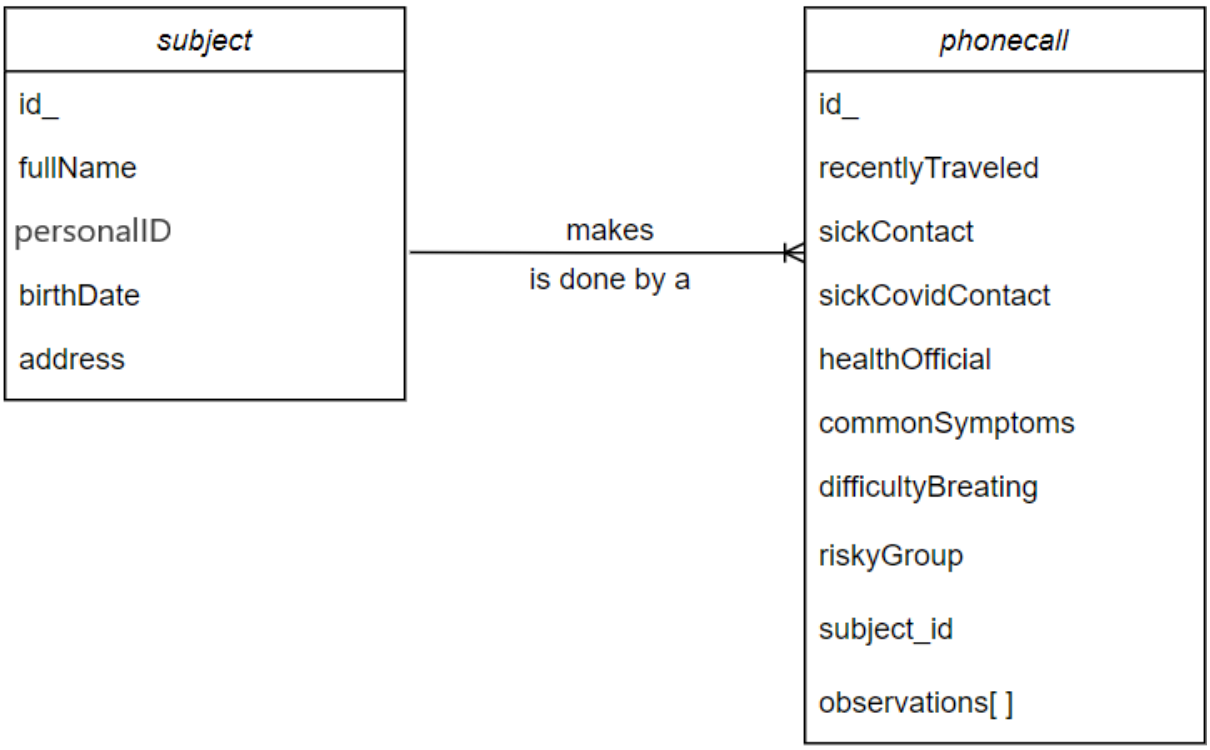

Figura 4.1: Diagrama base de datos

Inmediatamente después de haber construido el diagrama de la base de datos, se ha comenzado con la conceptualización de los bocetos visuales de la aplicación. En esta primera aproximación al diseño web se ha optado por un estilo simple y sobrio, pues la finalidad de dichos bosquejos es la de ofrecer una idea general y abstracta de cómo se desea expresar la información tratada durante el uso de la herramienta (véase Figura 4.2). En cuanto a la organización de los datos en la pantalla, se ha decidido dividirlos en cuatro bloques:

- Test: dedicado a las respuestas del sujeto a la encuesta de la COVID-19.
- Sujeto: contiene los datos personales del afectado necesarios y útiles para el operador.
- Recursos: indica los servicios puestos en marcha para manejar la situación.
- Observaciones: compuesto por comentarios hechos sobre los operadores.

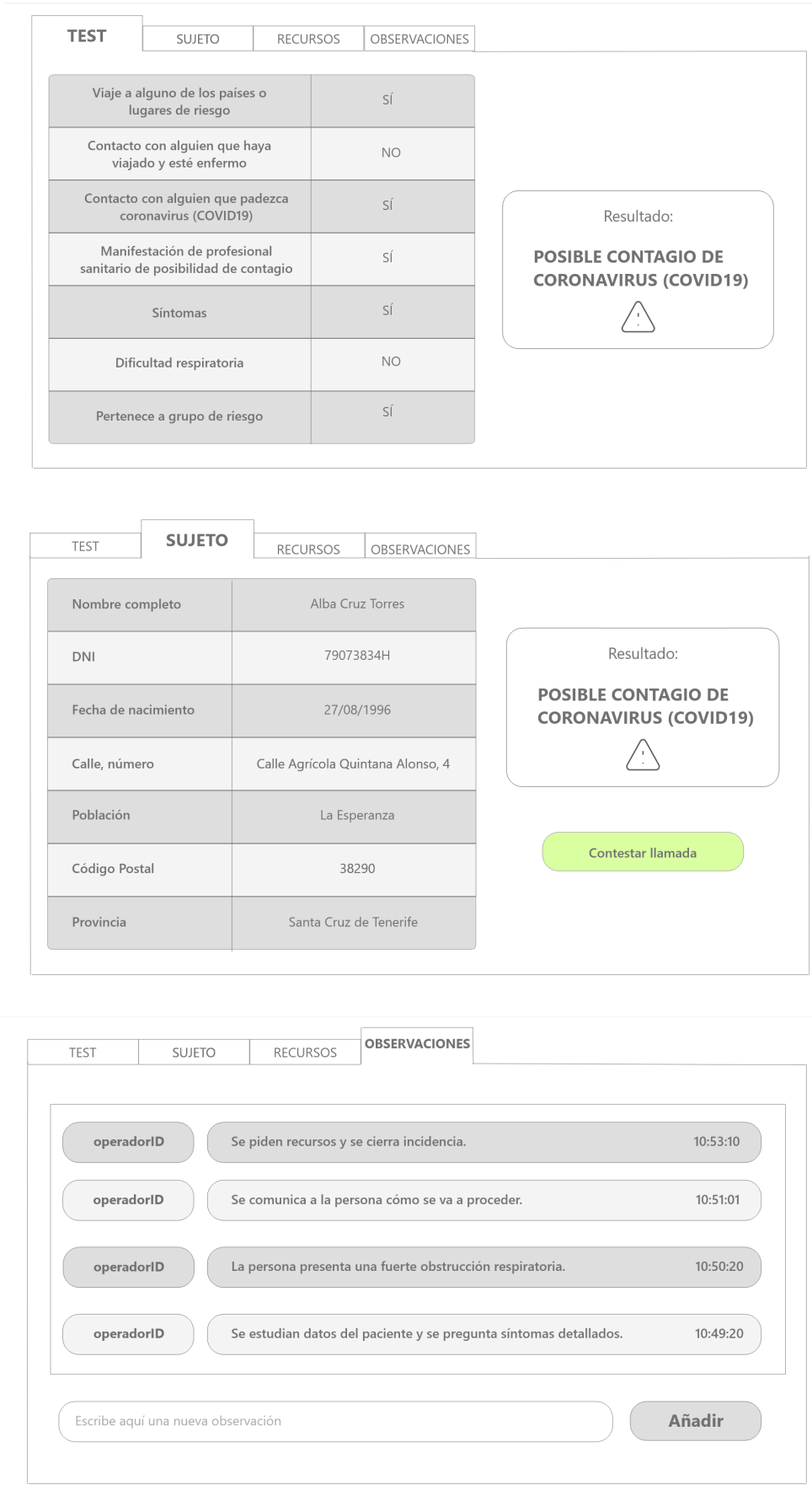

Figura 4.2: Bocetos iniciales de las vistas pertenecientes a la aplicación web

Más adelante y con el fin de ejecutar un diseño con un mayor grado de conveniencia, se ha tenido en consideración el tipo de empresa para la que se estaba trabajando y se ha diseñado una posible propuesta dentro de los estándares encontrados en las aplicaciones y páginas web ya existentes en dicha organización. Por lo tanto, se ha querido mantener el diseño sobrio y simple de los bocetos, pero añadiendo cierta tonalidad en las partes más funcionales de la aplicación. No obstante, se ha mantenido la importancia y la atención sobre los datos que al final son los que constituyen el grueso de la aplicación. Mediante una combinación de ligeros patrones modernos de diseño se han obtenido como resultado los mockups que se muestran en la Figura 4.3.

4.2. Back-end

A la hora de hacer posible las principales funcionalidades de la aplicación, se ha desarrollado, en primer lugar, la experiencia conversacional con el usuario, es decir, la encuesta sobre COVID-19. Para ello, se ha partido de la estructura básica de código que utiliza Dialogflow para establecer la ruta a seguir y emprender así una conversación completa con el usuario. Dicha estructura básica no es más que un objeto que contiene la respuesta que debe ofrecer la herramienta al usuario, habiendo estudiando previamente la situación exacta de la conversación. En ese objeto, se integra el valor textual de dicha respuesta dentro el elemento con clave "*speech*", (véase Figura 4.4). Finalmente, se envía la respuesta en formato JSON para que Dialogflow pueda continuar la conversación con el usuario.

No obstante, aún no se ha explicado cómo se decide qué respuesta ofrecer al usuario según cómo este actúe a la hora de resolver la encuesta. Es por esto que en Dialogflow se ha generado un caso o *Intent* por cada una de las preguntas que el usuario debe responder. Dichas preguntas evalúan la situación de cada uno de los sujetos o afectados que contactan con la línea específica de coronavirus del centro de emergencias. Asimismo, se debe hacer énfasis en el hecho de que mientras se lleva a cabo la encuesta, se van estudiando las respuestas y tomando un camino u otro dependiendo de éstas. Con el fin de identificar los resultados, se ha asignado una palabra clave a cada una de las preguntas, las cuales además se corresponden con los registros de las colecciones en la base de datos, de tal forma que se puedan identificar de manera rápida y sencilla.

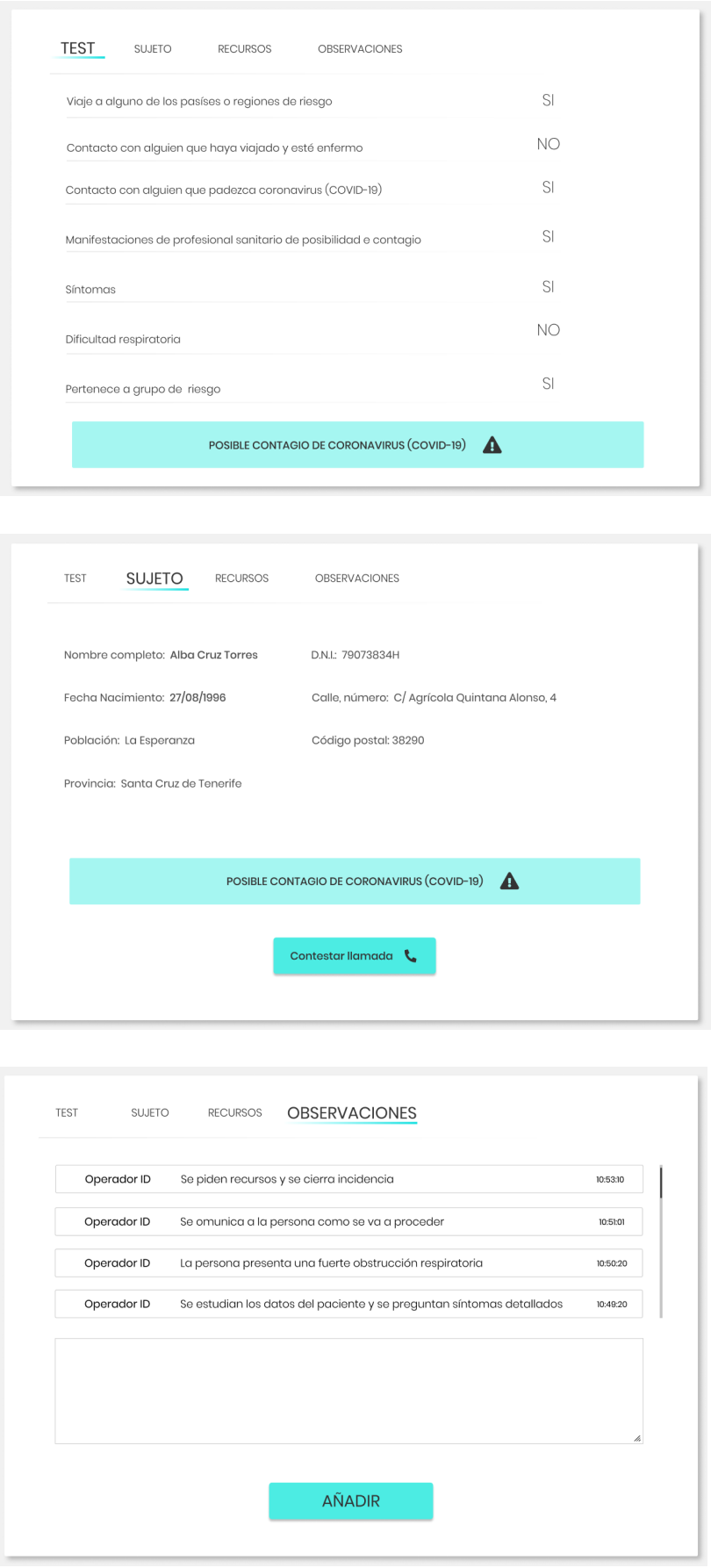

Figura 4.3: Mockups de las vistas correspondientes a la aplicación web

```
const speechResponse = {
    google: {
        expectUserResponse: true,
        richResponse: {
            items: [
            {
                simpleResponse: {
                textToSpeech: speech
                }
            }
            ]
        }
    }
}
return res.json({
    payload: speechResponse,
    data: speechResponse,
    fulfillmentText: speech,
    speech: speech,
    displayText: speech,

});
```

Figura 4.4: Estructura básica de una respuesta de Dialogflow

La clasificación de estas preguntas se realiza por medio de tres fases. La primera fase consta de las preguntas que hay que realizar obligatoriamente a cada encuestado, puesto que dependiendo de las respuestas a éstas se continuará preguntando para saber si se trata de un caso más crítico o se dará por finalizada, ya que no se corresponde con un caso que sea necesario seguir estudiando. Esta primera fase está compuesta por las preguntas que se muestran en la Figura 4.5.

Preguntas	Palabras clave
¿Ha viajado recientemente a alguno de los siguientes países? - España: las Comunidades Autónomas de Madrid y de La Rioja o los municipios de La Bastida y Vitoria (País Vasco). - China: todas las provincias, incluyendo Hong Kong y Macao. - Italia: todo el país. - Francia: departamentos de Haut-Rhin (Grabd Est) y l'Oise (Hauts-de-France). - Alemania: departamento de Heinsberg (Renania del Norte-Westfalia) - Japón: isla de Hokkaidō. - Corea del Sur. - Singapur. - Irán.	RecentlyTraveled
¿Ha estado en contacto con alguien que haya viajado a estos países y que ahora esté enfermo?	SickContact
¿Ha estado en contacto con alguien que conozca que esté contagiado de coronavirus (COVID-19)?	SickCovidContact
¿Algún profesional sanitario le ha manifestado que podría haber estado expuesto al coronavirus (COVID-19)?	HealthOfficial
¿Presenta alguno de los siguientes síntomas? - Fiebre - Tos - Moqueo nasal - Dolor de garganta	CommonSymptoms

Figura 4.5: Preguntas obligatorias y palabras clave correspondientes

Una vez respondidas las preguntas obligatorias, se ejecutará la segunda fase siempre y cuando el usuario o afectado haya respondido que sí presenta los síntomas indicados. Esto es así ya que se dará prioridad a estos usuarios sobre los demás, a los que se comunicará que no presentan indicios importantes y discriminatorios de ser posibles contagios de coronavirus. Por lo tanto, se les ofrecerá las distintas pautas y recomendaciones a seguir, además de la posibilidad de realizar la encuesta de nuevo siempre que lo deseen o noten alguna alteración en sus síntomas. En cambio, si se pasa a la segunda fase, se preguntará la información que podría categorizar las diferentes situaciones de manera más crítica o importante, basándolas en los síntomas e inconvenientes más significativos de la enfermedad que ocupa a este proyecto (véase Figura 4.6).

Preguntas	Palabras clave
¿Presenta dificultad respiratoria?	DifficultyBreathing
¿Se encuentra en alguna de estas situaciones? - Persona con 65 años o superior. - Enfermedades cardíacas (por ejemplo: enfermedad de las arterias coronarias, enfermedad valvular o insuficiencia cardíaca). - Enfermedades pulmonares (por ejemplo: asma o enfermedad de obstrucción pulmonar crónica). - Enfermedades inmunosupresoras (por ejemplo: quimioterapia, trasplante o diabetes).	RiskyGroup

Figura 4.6: Preguntas de filtrado y palabras clave

Por último, se encuentra la tercera y última fase en la que se preguntan los datos de carácter personal del sujeto. Se transitará a esta etapa siempre y cuando el usuario haya respondido afirmativamente la pregunta relacionada con la dificultad respiratoria o una vez habiendo respondido que no a esta pregunta, haya comentado que pertenece a alguno de los grupos de riesgo. Estos son los dos casos en los que se daría mayor prioridad a estas llamadas y se comenzaría a pedir los datos personales que se muestran en la Figura 4.7.

Preguntas	Palabras clave
¿Podría decirme su nombre completo?	FullName
¿Podría decirme los dígitos y letra de su Documento Nacional de Identidad?	PersonalID
¿Podría decirme su fecha de nacimiento?	BirthDate
¿Podría decirme la dirección en la que se encuentra?	Address

Figura 4.7: Preguntas de datos personales y palabras clave

Por tanto, para poder dotar a Dialogflow de las posibles preguntas a efectuar y dar lugar así a una experiencia conversacional más adecuada y cercana al usuario, se ha implementado una función compuesta por una sentencia *Switch-Case* en la que se especifican cada una de las preguntas según el flujo de conversación explicado anteriormente. Por ello, cada uno de los casos de la función se corresponderá con los intents implementados en Dialogflow, respectivamente, ya que estos intents nos permiten mantener la conexión entre Dialogflow y el código de manera correcta (véase Figura 4.8). Esta conexión con Dialogflow seguirá en marcha hasta que se termine de realizar la encuesta o lo que es decir cuando se haya alcanzado el último caso o intent dependiendo de cada situación (véase Figura 4.9).

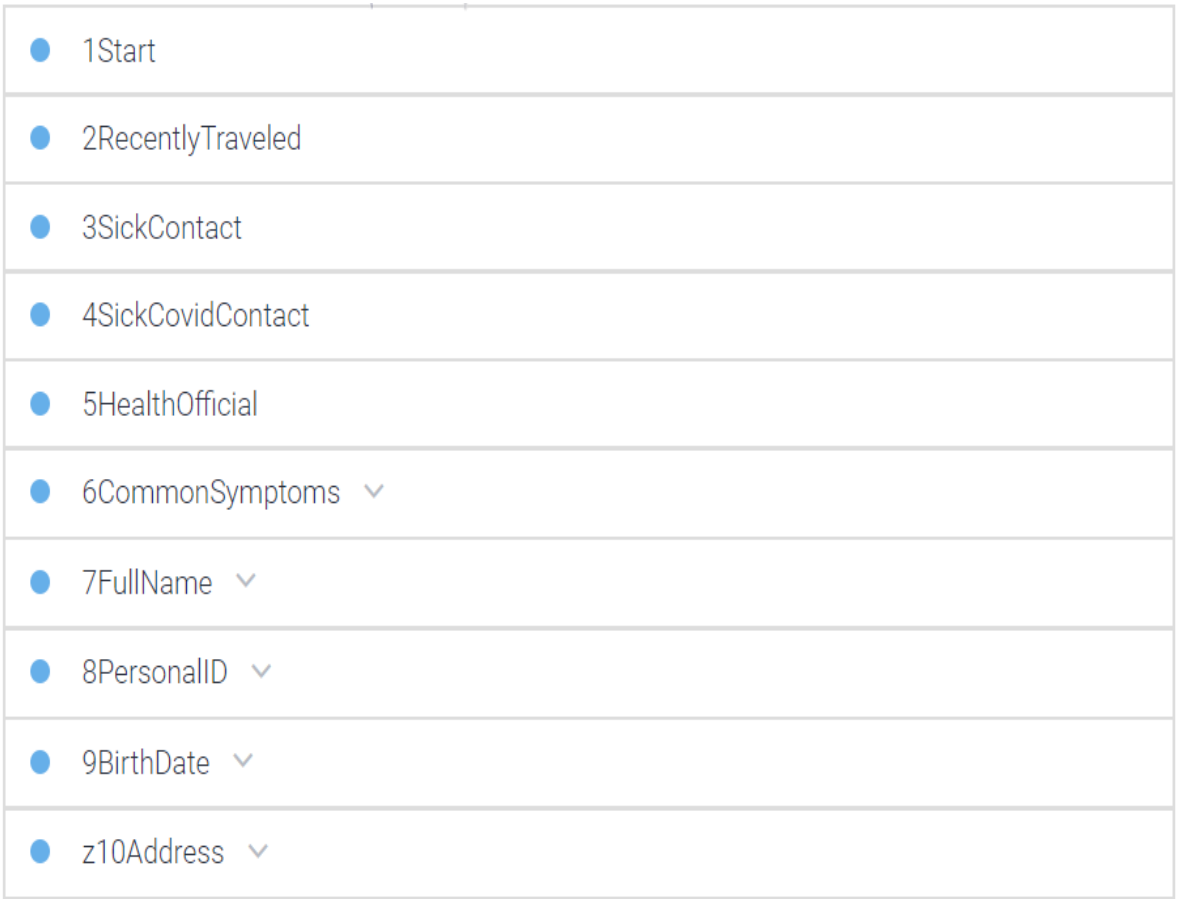

Figura 4.8: Intents de Dialogflow

```
const cases = async (intent, parameters) => {
    switch(intent){
        case "Start":
            return questions.start;
        case "RecentlyTraveled":
            return questions.recentlyTraveled;
        case "SickContact":
            if(parameters.recentlyTraveled == "Sí"){
                answers.recentlyTraveled = true;
            }
            return questions.sickContact;
        case "SickCovidContact":
            if(parameters.sickContact == "Sí"){
                answers.sickContact = true;
            }
            return questions.sickCovidContact;
        case "HealthOfficial":
            if(parameters.sickCovidContact == "Sí"){
                answers.sickCovidContact = true;
            }
            return questions.healthOfficial;
        case "CommonSymptoms":
            if(parameters.healthOfficial == "Sí"){
                answers.healthOfficial = true;
            }
            return questions.commonSymptoms;
```

Figura 4.9: Fragmento de la función con las distintas situaciones de conversación

Una vez llegados a este momento tenemos dos posibilidades: una primera en la que no se tienen sospechas de que el sujeto presente un posible contagio y una segunda en la que el usuario podría ser un caso de padecimiento de coronavirus, en el cual se procederá a hacer una petición a la base de datos para guardar por un lado, los datos o respuestas de la llamada y, por otro lado, los datos personales del afectado. Para hacer dichas peticiones, se han implementado dos funciones que envían los datos del sujeto que efectúa la llamada, de manera que, una vez se añada y se tenga el identificador del mismo en la base de datos, se pueda crear la instancia de su llamada (véase Figura 4.10). En estas funciones lo que se hace son peticiones POST a la base de datos mediante axios y utilizando la dirección ip del servidor de DigitalOcean donde está alojada la aplicación de NodeJS, Express y Mongoose con la que se maneja la base de datos.

```
const axios = require("axios");
const URL = "http://178.62.41.123:3000";

const sendPhonecallToDB =  async () => {

    axios.post(URL + "/phonecalls/addPhonecall", answers)
    .then(response => console.log(response))
    .catch(e => console.log(e));
}

const sendSubjectToDB = async () => {

    axios.post(URL + "/subjects/addSubject", subject)
    .then(async response => {
        answers.subject_id = response.data;
        await sendPhonecallToDB();
    })
    .catch(e => console.log(e));
}
```

Figura 4.10: Funciones que realizan peticiones POST a la base de datos

Dicha aplicación NodeJS con Express y Mongoose da lugar a la API REST que hace posible la conexión entre el Dialogflow y la base de datos. Para constituirla, se ha implementado, en primer lugar, el servidor web con NodeJS y la configuración de Express. Es en este momento en el que se lleva a cabo la conexión específica con la base de datos de MongoDB, a través del método *connect()* que ofrece Mongoose. Cabe destacar que se han especificado unas opciones de conexión de Mongoose, de manera que sólo se pueda acceder a la base de datos con la introducción de un usuario y contraseña autorizados, garantizando así que no haya un robo de los datos o que éstos puedan verse comprometidos y, finalmente, se mandará un mensaje de confirmación por consola una vez realizada dicha conexión (véase Figura 4.11).

Por último, se crea el servidor web por medio del método *listen()* que proporciona Express, que permitirá escuchar la dirección ip y el puerto especificados, que se corresponderán con la dirección a utilizar cada vez

que se quiera hacer una petición a la base de datos.

```
const express = require("express");
const mongoose = require("mongoose");
const app = express();

const phonecallRouter = require("../routers/phonecallRouter");
const subjectRouter = require("../routers/subjectRouter");

app.use("/phonecalls", phonecallRouter);
app.use("/subjects", subjectRouter);

const connectionOptions =  {
    auth: { authSource: "admin" },
    user: "root",
    pass:         ,
    useNewUrlParser: true,
    useUnifiedTopology: true
};
mongoose.connect("mongodb://161.35.42.76:27017/emergenciasdb", connectionOptions)
.then(console.log("DB connection was succesful"))
.catch(err => console.log(err));

app.set("port", process.env.PORT || 3000);
const port = app.get("port");

app.listen(port, () => console.log(`Listening on port ${port}!`));
```

Figura 4.11: Servidor de la API REST

Asimismo, se deben añadir las distintas rutas principales a las que responderá la aplicación, una para cada una de las colecciones de la base de datos. Luego, se han generado las rutas específicas a cada una de las acciones y peticiones que se pueden realizar con esta API, es decir, hacen referencia a los puntos finales o *endpoints* de la aplicación. Dichas rutas se generan gracias a las funcionalidades de direccionamiento que aporta Express mediante el método *Router()* (véase Figura 4.12 y Figura 4.13). Cada una de las colecciones de la tabla tiene su ruta principal y sus rutas particulares, de forma que si se desea añadir un nuevo sujeto a la colección de sujetos se utilizarán las rutas establecidas para gestionar y manejar los datos relativos a los sujetos.

```
const express = require("express");
const router = express.Router();
const phonecallController = require("../controllers/phonecallController");

router.get("/", phonecallController.showAll);
router.get("/:id", phonecallController.showPhonecall);
router.get("/showFirst", phonecallController.showFirstPhonecall);
router.get("/getObservations/:id", phonecallController.getObservations);
router.post("/addPhonecall", phonecallController.addPhonecall);
router.post("/addObservation/:id", phonecallController.addObservation);
router.delete("/:id", phonecallController.deletePhonecall);
router.delete("/deleteObservations/:id", phonecallController.deleteObservations);
router.put("/:id", phonecallController.updatePhonecall);

module.exports = router;
```

Figura 4.12: Rutas o endpoints de la colección de llamadas

```
const express = require("express");
const router = express.Router();
const subjectController = require("../controllers/subjectController");

router.get("/", subjectController.showAll);
router.get("/:id", subjectController.showSubject);
router.post("/addSubject", subjectController.addSubject);
router.delete("/:id", subjectController.deleteSubject);
router.put("/:id", subjectController.updateSubject);

module.exports = router;
```

Figura 4.13: Rutas o endpoints de la colección de sujetos

Ahora bien, ¿cómo responde la aplicación cada vez que se llama a alguna de estas rutas? La aplicación responde a las peticiones por medio de los controladores que son los que establecen qué acciones y operaciones se efectuarán sobre la base de datos según cada endpoint. Cada una de las colecciones posee las funciones necesarias para implementar un *CRUD* (*Create, Read, Update, Delete*), lo cual significa añadir, leer, actualizar y borrar datos, respectivamente (véase Figura 4.14). Todas estas funciones serán las que luego se utilicen para implementar el Front-end, ejecutando cada una de ellas cada vez que sea necesario en la página web. Es por esto que se crean, por ejemplo, los endpoints específicos para manejar y trabajar en el Front-end con las distintas observaciones que los teleoperadores pueden redactar sobre las situaciones que deben gestionar.

```
const Phonecall = require('../models/Phonecall');

async function showAll(req, res) {
    const phonecall = await Phonecall.find()
    res.json(phonecall);
}

async function showPhonecall(req, res){
    const id = req.params.id;
    const foundPhonecall = await Phonecall.findById(id);
    res.json(foundPhonecall);
}

async function addPhonecall(req, res) {
    const newPhonecall = new Phonecall(req.body);
    newPhonecall.save();
    res.json(newPhonecall);
}


async function getObservations(req, res) {
    let temp = [];
    let observations = [];
    try{
        await Phonecall.findById(req.params.id)
        .then(phonecall => {
            temp = phonecall.observations;
            for(let i = temp.length-1; i>=0; i--){
                observations.push(temp[i]);
            }
            res.json(observations);
        }).catch(function(err){
            console.log(err);
        });
    } catch(e){
        next(e);
    }
}

async function addObservation(req, res) {
    console.log(req.params.id);
    try{
        await Phonecall.findById(req.params.id)
        .then(phonecall => {
            phonecall.observations.push(req.body);
            phonecall.save();
            res.json(phonecall);
        }).catch(function(err){
            console.log(err);
        });
    } catch(e){
        next(e);
    }
}
```

Figura 4.14: Controladores de la colección de llamadas

Por último, se necesita implementar los constructores o modelos de los documentos u objetos de la base de datos. Éstos se crean a partir de los distintos esquemas que se asignan a las colecciones de MongoDB. Con estos esquemas se define la forma que tendrán los documentos dentro de esa colección. Es en este momento en el que se debe establecer el requisito de encriptar y desencriptar los datos personales de los sujetos cada vez que se realicen peticiones a la base de datos. Por ello, el modelo de los sujetos estará compuesto por una función en la que a través del paquete mongoose-field-encryption se indican qué campos se quiere encriptar por medio de la clave secreta y la sal generada previamente (véase Figura 4.15).

```
const mongoose = require('mongoose');
const mongooseFieldEncryption = require("mongoose-field-encryption").fieldEncryption;
const Schema = mongoose.Schema;
const key = "mtivpiy2ohlbnl9zfx4hqo";

const SubjectSchema = new Schema({
    fullName: {
        type: String,
        required: true
    },
    personalID: {
        type: String,
        unique: true,
        required: true
    },
    birthDate: {
        type: Date,
        required: true
    },
    address:{
        type: String,
        required: true
    },
    createdAt:{
        type: Date
    },
    updateAt: {
        type: Date
    }
});

SubjectSchema.plugin(mongooseFieldEncryption, {
    fields: ["fullName", "dni", "birthDate", "address"], secret: key});
module.exports = mongoose.model('subjects', SubjectSchema);
```

Figura 4.15: Modelo de la colección de sujetos con función de encriptación

4.3. Front-end

Después de haber llevado a cabo la encuesta y de haber tratado y guardado los datos correctamente, se ha procedido a la implementación del sitio web siguiendo los mockups diseñados. Para dicho desarrollo se ha usado VueJS y se han generado tres vistas que se corresponden con tres de los cuatro tabs o pestañas de información presentes. Cada una de las vistas está formada por los componentes o instancias reutilizables de Vue con los que se consigue visualizar cada uno de los datos y posicionarlos en la pantalla de forma adecuada. Dichos tabs han sido implementados gracias al componente *b-tabs* ya existente que ofrece BootstrapVue (véase Figura 4.16).

Para poder viajar entre los distintos tabs se ha hecho uso del *Router* de VueJS, con el que se han establecido las distintas rutas a las que se puede acceder desde la página web. Siempre que se haga click sobre alguna de las tabs se ejecutarán las funciones que indicarán al router a qué ruta debe ir después de dicho evento (véase Figura 4.17).

```
<template>
    <div class="phonecall">
        <TabsPhonecall></TabsPhonecall>
    </div>
</template>

<script>
// @ is an alias to /src
import TabsPhonecall from '@/components/TabsPhonecall.vue'

export default {
  name: 'Phonecall',
  components: {
      TabsPhonecall
  },
}
</script>
```

```
<div class="tabsPhonecall">
    <b-card no-body>
        <b-tabs card>
            <b-tab :title-link-class="'tab-title-class-active'" title="TEST" active>
                <Container :nmbr="sizeTest"/>
                <div class="info">
                    <div class="alerta">
                        <p>POSIBLE CONTAGIO DE CORONAVIRUS</p>
                    </div>
                </div>
            </b-tab>
            <b-tab :title-link-class="'tab-title-class'" title="SUJETO" v-on:click="toSubject()">
            </b-tab>
            <b-tab :title-link-class="'tab-title-class'" title="RECURSOS">
            </b-tab>
            <b-tab :title-link-class="'tab-title-class'" title="OBSERVACIONES" v-on:click="toObservations()">
            </b-tab>
        </b-tabs>
    </b-card>
</div>
```

Figura 4.16: Vista de llamadas y componente principal de tabs de llamadas

```
<script>
import Container from "./Container"
export default {
    name: 'TabsPhonecall',
    components: {
        Container
    },
    data(){
        return {
            sizeTest: 7,
        };
    },
    methods: {
        toSubject() {
            this.$router.push({path:'/subject'});
        },
        toObservations() {
            this.$router.push({path:'/observations'});
        }
    }
}
</script>
```

Figura 4.17: Funciones para cambiar de ruta en la aplicación web

Inmediatamente después de instanciar las vistas, hay que componerlas con los datos que se quieren mostrar. Por ello, se ha implementado un componente llamado *Container* que se encargará de dividir cada una de las pantallas en las filas y columnas correspondientes a cada vista. Por ejemplo, en la vista del test, se mostrarán los resultados de la encuesta y se sabe que las preguntas dedicadas a la COVID-19 son siete de las once realizadas, mientras que las otras cuatro se corresponden con las de los datos personales del sujeto. Es por ello que a este componente se le pasará como propiedad el número de filas que ha de tener dependiendo de si es la vista del test o la del sujeto. Por tanto, una vez conocido el número de filas, se procede a la creación de un *grid*, que no es más que una cuadrícula de varios elementos que en este caso serán cada una de las preguntas con sus respuestas correspondientes (véase Figura 4.18). Previamente, se han recolectado dichas preguntas y respuestas efectuando las peticiones de lectura de datos necesarias a la base de datos (véase Figura 4.19).

```
createGrid(rows, cols){
  let container = document.getElementById("ctr");
  container.style.setProperty('--grid-rows', rows);
  container.style.setProperty('--grid-cols', cols);
  let contP1 = 0, contP2 = 0, contS1 = 0, contS2 = 0;
  for (let c = 0; c < (rows * cols); c++) {
      let cell = document.createElement("div");
      if((c%2) == 0){
        if(rows == 7){
          cell.innerText = Object.values(this.phonecallText)[contP1];
          contP1++;
        }
        else{
          cell.innerText = Object.values(this.subjectText)[contS1];
          contS1++;
        }
      }
      else{
        if(rows == 7){
          if(Object.values(this.phonecall)[contP2]){
            cell.innerText = "Sí";
            contP2++;
          }
          else{
            cell.innerText = "No";
            contP2++;
          }
        }
        else{
          cell.innerText = Object.values(this.subject)[contS2]
          contS2++;
        }
      }
      container.appendChild(cell).className = "grid-item contenedor";
      this.centerAnswersBorders();
  };
},
```

Figura 4.18: Función para crear la cuadrícula con preguntas y respuestas

```
async getAnswers(){
  await this.$store.dispatch("getResponses");
  this.phonecall = {...this.$store.getters.getAnswers};
  const subjectId = { ...this.phonecall }.subject_id;
  const suj = await this.$store.dispatch("getSubject", subjectId);
  this.subject = {...this.$store.getters.getSubject};
},
```

Figura 4.19: Función para pedir los datos almacenados en la base de datos

Por último, se debe hacer hincapié en la parte más importante que permite a la aplicación funcionar correctamente y esto es la petición y acceso a los datos almacenados en la base de datos. Este proceso se ha llevado a cabo por medio del *Store* de Vue, el cual es un patrón de manejo del estado o flujo de comportamiento de la aplicación. Dicho store está formado por varias secciones: actions, mutations, state y, por último, getters. Cada una de estas secciones tiene una tarea específica que permite controlar el torrente de datos en la aplicación de la manera deseada.

Las *Actions* o acciones son métodos que dan la posibilidad de ejecutar operaciones asíncronas. En esta sección se implementa cada uno de los métodos necesarios para efectuar las distintas peticiones a la base de datos que se utilizarán más adelante en el desarrollo de la aplicación web. Para ello, se utiliza el mismo paquete axios nombrado anteriormente, indicando la dirección ip, el puerto y la ruta a la que realizar dicha petición (véase Figura 4.20).

```
actions: {
  async getResponses({ commit }) {
    await axios.get(URL + "/phonecalls")
    .then(function (response) {
      const resp = response.data[0];
      commit("setValuesPhonecall", resp);
    })
    .catch(function(error){
      console.log(error);
    });
  },

  async getSubject({commit}, id){
    let months = ["enero", "febrero", "marzo", "abril", "mayo", "junio",
    "julio", "agosto", "septiembre", "octubre", "noviembre", "diciembre"]
    await axios.get(URL + "/subjects/" + id)
    .then(response => {
      let sujeto = response.data;
      let fecha = Object.values(sujeto)[3].split("T")[0].split("-");
      fecha[1] = months[fecha[1]-1];
      sujeto.birthDate = fecha[2] + " de " + fecha[1] + " de " + fecha[0];
      commit("setValuesSubject", sujeto);
    })
    .catch(err => console.log(err));
  },
```

Figura 4.20: Algunas de las acciones o métodos asíncronos del Store de Vue

Las *Mutations* o mutaciones son métodos con los que se cambia el estado del store. Constituyen eventos síncronos en los que se realizan modificaciones de estado reales. Por tanto, una vez se tengan los datos devueltos por las acciones, se ejecutan estas mutaciones que almacenarán dichos datos en el estado (véase Figura 4.21).

```
mutations: {
  setValuesPhonecall (state, values){
    state.phonecall = { ...state.phonecall, ...values};
  },
  setValuesSubject (state, values){
    state.subject = { ...state.subject, ...values};
  },
  setNewObs(state, values) {
    state.phonecall.observations = {...state.phonecall.observations, values};
  }
},
```

Figura 4.21: Mutaciones o métodos síncronos del Store de Vue

El *State* o estado es la fuente de verdad que impulsa la aplicación. Se trata de un objeto que contiene las claves y valores de los elementos o datos que serán necesarios para dar sentido a la herramienta (véase Figura 4.22).

```
state: {
  phonecall: {
    _id: "",
    recentlyTraveled: "",
    sickContact: "",
    sickCovidContact: "",
    healthOfficial: "",
    commonSymptoms: "",
    difficultyBreathing: "",
    riskyGroup: "",
    subject_id: "",
    observations: []
  },
  subject: {
    fullName: "",
    personalID: "",
    birthDate: "",
    address: ""
  }
},
```

Figura 4.22: Estado del Store de Vue

Por último, los *Getters* son métodos que permiten pedir y leer datos almacenados en el estado, es decir, cada vez que se quiera mostrar un dato en la aplicación, se ejecutará el getter necesario que devolverá el dato o elemento en cuestión (véase Figura 4.23).

```
getters: {
  getAnswers(state){
    return state.phonecall;
  },
  getSubject(state){
    return state.subject;
  },
  getObservations(state){
    return state.phonecall.observations;
  },
  getId(state){
    return state.phonecall._id;
  }
}
);
```

Figura 4.23: Funciones getter del Store de Vue

Con todo esto se consigue mostrar los datos tanto de la encuesta como los del afectado a los teleoperadores, que después de haber revisado dicha información se pondrán en contacto con el sujeto y tomarán las decisiones convenientes a cada situación. Por esto, se ha habilitado un botón para contestar la llamada que desplegará un cronómetro de manera que el operador tenga un control del tiempo en todo momento. De igual manera, junto con este cronómetro se desplegarán dos botones más, uno para poner en pausa la llamada, lo cual puede ser muy útil siempre que el teleoperador necesite algo de tiempo o intimidad para realizar consultas a otros compañeros y otro para finalizar la llamada. Cada vez que se pulse este último botón, se procederá a eliminar el caso que se acaba de gestionar, actualizar la página web y mostrar los datos de la siguiente llamada.

Finalmente, la pestaña de las observaciones se ha generado siguiendo la misma estructura que las otras dos pestañas, es decir, creando un grid con los elementos a introducir. Esta cuadrícula dependerá del número de observaciones registradas a esa llamada y tendrá siempre tres columnas

para mostrar el identificador del operador que ha añadido la observación, el mensaje de la misma y, por último, la fecha y hora en la que se añadió. De esta forma, cualquier operador que lea dichas observaciones podrá entenderlas de un modo más rápido e intuitivo. Además, se ha incluido un cuadro de texto para poder introducir una nueva observación y añadirla después con el botón implementado para ello. Cada vez que se haga click sobre este botón, se llamará a la acción del store correspondiente y se actualizará la página en cuestión para que muestre la nueva observación, haciendo uso de los vigiladores o *watchers* que ofrece Vue, los cuales observan si se produce algún cambio en las variables o elementos que se le indique (véase Figura 4.24).

```
watch:{
    newObservation: async function(){
        let first = document.getElementsByClassName("grid-item");
        const size = Object.values(first).length;
        if(size != 0){
            Object.values(first).map(n => n && n.remove());
        }
        await this.getData();
    }
}
```

Figura 4.24: Watcher que vigila cambios en las observaciones y actúa en consecuencia

4.4. Problemas encontrados

Finalmente, se procederá a comentar los inconvenientes a los que se ha tenido que hacer frente durante el desarrollo de esta aplicación. Si bien es cierto que esta herramienta no se ha visto perjudicada por una gran cantidad de problemas emergentes, es importante destacar algunos aspectos importantes que dificultaron su implementación.

En primer lugar, la curva de aprendizaje de Dialogflow ha sido bastante compleja, ya que consta de mucha información y trabajo para conocer y aprender cómo llevar a cabo la conexión del código con la herramienta. No obstante, una vez comprendido el flujo de comportamiento, se vio necesario el alojamiento del código en alguna plataforma en la que construir, correr y operar la aplicación en la nube, pues no se podía establecer la conexión

desde un servicio local al servicio público de Dialogflow. Es por esta causa que se eligió y se estudió Heroku, para subir así el código a la nube a través de un repositorio de GitHub. Cada vez que se realizaba una modificación en dicho código, éste se actualizaba directamente en la aplicación alojada en Heroku.

Más adelante, debido a la subida a Heroku del código de conexión con Dialogflow, se ocasionaron algunos inconvenientes a la hora de hacer las peticiones a la base de datos, pues en ese momento el servidor estaba construido en local y, por lo tanto, desde este código alojado en la nube no se podía realizar una petición a esa dirección local que contenía el servidor de la base de datos. Es por esto que se decidió crear una máquina virtual o droplet de DigitalOcean que actuara como servidor, sin tener que crearlo manualmente, es decir, con la única tarea de administrarlo. Asimismo, se ha creado otro droplet para alojar el contenedor de Docker con la base de datos MongoDB, pues se producía el mismo error que en los dos casos anteriores: el contenedor se había creado localmente.

Por otro lado, cabe destacar que a causa de la situación sanitaria tan grave que se está viviendo hoy en día, no se pudo preguntar al cliente e investigar sobre los requerimientos y las características que debía presentar la pestaña de los *Recursos*, que ha sido nombrada anteriormente pero no se ha visto incluida ni en el diseño ni en el desarrollo de la aplicación. Esto se debe a que se tenía prevista una visita a las inmediaciones de la empresa, de tal forma que se pudieran recabar más datos e información sobre esta pestaña. Sin embargo, dicha visita no pudo efectuarse debido al estado de alarma impuesto en todo el país.

Por último, pero no menos importante, se encontraron algunos inconvenientes u obstáculos a la hora de afrontar el enfoque a tomar durante la implementación del Front-end, pues hay muchísimas formas distintas de ejecutarlo, gracias a la posibilidad de desarrollar componentes propios y de poder utilizar todos los componentes ya existentes que ofrecen los frameworks especiales para VueJS. Pensar este enfoque de manera concienzuda es importante, ya que luego puede dar lugar a un desarrollo más fácil, rápido y eficiente. De lo contrario, puede ocasionar código complicado, rebuscado o mal estructurado [29].

Capítulo 5

Presupuesto

El Centro Coordinador de Emergencias y Seguridad (CECOES) 1-1-2 del Gobierno de Canarias requiere un servicio en el que se desarrolle una aplicación web para aliviar la situación de crisis sanitaria vivida actualmente y que se genere un informe final de desarrollo y evolución del proyecto. Se ha estipulado que la aplicación debe cumplir los requisitos que se muestran en la Tabla 5.1.

Requisitos
Efectuar encuesta COVID-19 a usuarios
Filtrar casos más críticos
Tratar y almacenar datos de manera segura
Mostrar resultados a través de página web

Tabla 5.1: Requisitos aplicación

Para la construcción de dicha aplicación la empresa cliente ofrece una plantilla de salida de un Chatbot sobre coronavirus creada por el servicio Health Bot Service de Microsoft Azure. Si bien es cierto que la ejecución y construcción del proyecto se ha llevado a cabo en un tiempo total de cuatro meses, la fase inicial de establecimiento de requisitos se ha efectuado previamente. Este conjunto de meses dedicado a la total planificación y elaboración del proyecto se corresponde con el primer quinquemestre del año 2020. Para organizar el trabajo, se ha generado un cronograma en el que se muestran las tareas a realizar junto con esa fase previa en la que se han fijado los requerimientos que debe cumplir la aplicación (véase Tabla 5.2).

Cronograma	Tareas
Enero	Establecer requisitos con el cliente
Febrero (Comienzo del proyecto)	Investigación y familiarización con herramientas
1ª mitad Marzo	Diseño de base de datos y mockups
2ª mitad Marzo - Abril	Implementación Back-end
Mayo	Implementación Front-end
Abril - Mayo	Redacción de informe de proyecto

Tabla 5.2: Cronograma y tareas a realizar

Con el fin de estipular un presupuesto asequible y adecuado al proyecto, se ha utilizado como guía el Informe de Empleo Informática - Mayo 2020 ofrecido por el Portal de Empleo en Informática, Telecomunicaciones y Tecnologías: *tecnoempleo.com* [30], de donde se han sacado los salarios anuales brutos de los profesionales TIC necesarios para el desarrollo de la aplicación. Los costes por hora y totales de cada una de las tareas nombradas anteriormente, son los que se recogen en la Tabla 5.3.

Tareas	Horas	Coste por hora	Total
Investigación y familiarización con herramientas	30	16€	480€
Diseño de base de datos	12	17€	204€
Diseño de mockups	20	18€	360€
Implementación Back-end y Front-end	80+64 = 144	16€	2304€
Redacción informe de proyecto	90	17€	1530€
Total	-	-	4878€

Tabla 5.3: Coste total por tareas a realizar

En cuanto a las herramientas utilizadas para hacer posible el correcto funcionamiento de la aplicación, se debe hacer énfasis en dos de ellas: Heroku y DigitalOcean, las cuales se han utilizado para alojar el código en las distintas fases del proyecto. Estos costes que se muestran a continuación son de carácter ilustrativo pues, en cualquier caso la empresa puede efectuar una investigación sobre qué herramientas o paquetes con qué características les conviene más para cada situación. El coste por mes orientativo de uso de las nombradas herramientas es el que se engloba en la Tabla 5.4.

Herramientas	**Especificaciones**	**Coste por mes**
Heroku	Aplicación con funcionalidad compleja que requiere gran disponibilidad	229,25€
DigitalOcean	4GB de memoria, 2vCPUs, 4TB de transferencia	18,34€

Tabla 5.4: Coste mensual por uso de herramientas Heroku y DigitalOcean

Finalmente, después de haber realizado un estudio tanto de los perfiles necesarios para llevar a cabo el proyecto como del número de horas para efectuar cada tarea y del coste por hora, se ha calculado como resultado un presupuesto total de 4878€. Con dicho presupuesto, se cubre la construcción y desarrollo de la aplicación, así como la redacción del informe del proyecto y el uso de las herramientas de hosting y servidores necesarias. Una vez entregado el proyecto resultante a la empresa, ésta se encargará de migrarlo a los dominios y servidores propios de la misma y llevará a cabo el estudio de las herramientas que le sean más adecuadas.

Capítulo 6

Conclusiones y líneas futuras

Finalmente, después de un largo trabajo de investigación y construcción de la aplicación, se han extraído varias conclusiones. En primer lugar, es un hecho que la población mundial se ha visto afectada por una crisis sanitaria muy dura para los tiempos que corren, debido a la cual han muerto ya alrededor de 343.000 personas en todo el mundo. Si bien es cierto que España no estaba preparada para afrontar una situación de esta magnitud, la mayor parte de la población española ha sido capaz de aguantar y aceptar las medidas estrictas de confinamiento impuestas para evitar que creciera el número de muertos por no tomar las precauciones adecuadas.

Por otro lado, a causa de dicha situación sanitaria, especialmente durante las primeras semanas de crisis, las líneas de los centros de emergencias se colapsaron, al igual que los centros hospitalarios. Por tanto, un aprendizaje de la situación vivida ha sido la necesidad de desarrollar herramientas tecnológicas que permitan aliviar el trabajo de los teleoperadores de emergencias.

Precisamente el objetivo de este trabajo ha sido el diseño e implementación de una herramienta que apoye el trabajo de los teleoperadores, realizándolo como si se tratara de un humano y no una máquina quien estuviera interactuando con los usuarios. Una herramienta como esta podría ser de gran utilidad, no sólo en una situación crítica como la actual, sino en cualquier día de trabajo en un centro de emergencias. Una de las situaciones relativamente frecuentes y que permitiría afrontar una versión modificada de la herramienta desarrollada, sería cuando un desastre como por ejemplo un terremoto, oculta posibles llamadas de casos graves tales como un infarto. En casos como ese se podría activar una versión modificada de la herramienta desarrollada para discernir las llamadas

relacionadas con el desastre de otras posibles llamadas no relacionadas con el desastre. Así se podría afrontar la posibilidad de que algún caso grave no sea atendido con la brevedad necesaria.

En cuanto a las posibles mejoras de esta aplicación, cabría destacar, principalmente, la integración con los medios de telefonía necesarios para hacerla funcionar al completo, ya que por ahora funciona a través del servicio de voz de Google, debido a que no se posee una línea telefónica específica con la que hacer las pruebas necesarias. Sin embargo, lo único que haría falta sería la integración con una plataforma de telefonía, ya que el funcionamiento y comportamiento conversacional está totalmente implementado.

En segundo lugar, sería importante adaptar la encuesta a las necesidades del momento, así como las preguntas sobre datos personales del sujeto, las cuales siempre pueden variar con el fin de adecuarse a la información que sea más útil para los operadores de emergencias. Esto es debido a que la situación sanitaria relacionada con este virus ha sido tan cambiante que sería conveniente mantenerse atento en todo momento a cualquier modificación emergente y adaptarse a estos cambios lo más rápido posible.

Además, como se mencionó anteriormente, se tiene el deseo de desarrollar por completo la aplicación añadiendo la pestaña de *Recursos*, en la que los operadores pueden poner en marcha ciertos servicios, ya sea enviar una ambulancia o alguno de los cuerpos de policía, de manera que se muestre cuándo se puso en marcha dicho servicio, cuánto tiempo tardó en llegar ese servicio al lugar en concreto y cuándo se dio por finalizado el mismo.

Por último, una posible mejora estaría relacionada con el posible uso de aplicaciones de Inteligencia Artificial para intentar emular el comportamiento humano en este tipo de herramientas de interacción con humanos.

En conclusión, con el desarrollo de esta herramienta se ha querido ayudar, en la manera de lo posible, a uno de los cuerpos de trabajo que están en el frente de batalla día tras día, dando una posible solución temporal a estos centros de emergencias que alivie la carga de trabajo del momento y la cual pueda evolucionar a una aplicación de uso permanente y útil para el trabajo diario de dichos cuerpos.

Capítulo 7

Summary and conclusions

Finally, after a long researching and building work, several aspects have been concluded. Firstly, it is a fact that the world population has been affected by a very severe health crisis for these times and due to this situation around 343.000 people worldwide have already died. Although it is true that Spain was not prepared to face a situation of this magnitude, the great part of the Spanish population has been able to stand and accept the strict confinement measures imposed, because it is unacceptable to suffer more deaths because of not taking the proper precautions.

On the other hand, because of this health situation, especially during the first weeks of the crisis, the lines of the emergency centres collapsed, as did the hospital centres. Therefore, one lesson learned from the situation experienced was the need to develop technological tools that would allow the work of emergency tele-operators to be alleviated.

Precisely the objective of this work has been the design and implementation of a tool that supports the work of tele-operators, performing it as if it were a human and not a machine that was interacting with users. A tool like this could be very useful, not only in a critical situation like the current one, but also in any day of work in an emergency center. One of the relatively frequent situations that would allow a modified version of the developed tool, would be when a disaster such as an earthquake, hides possible calls of serious cases such as a heart attack. In such cases, a modified version of the developed tool could be activated to discern disaster-related calls from other possible non-disaster-related calls. This could confront the possibility that a serious case may not be attendend to as quickly as necessary.

Regarding the possible improvements of this application, it should be

highlighted, mainly, the integration with the necessary telephony means to make it work completely, since for now it works through Google's voice service, because it does not have a specific telephone hotline with which to carry out the necessary tests. Nevertheless, the only necessary thing would be the integration with a telephony platform, as the conversational behavior is fully implemented.

Secondly, it would be important to adapt the survey to the needs of the moment, as well as the questions about the subject's personal data, which can always be altered in order to arrange to the information that is most useful for emergency operators. This is due to the fact that the health situation related to this virus has been so changing that it would be advisable to pay special attention to any emerging modification and get adapted to these changes as quickly as possible.

Furthermore, as mentioned in the previous chapter, we would like to fully develop the application by adding the *Resources* tab, in which operators can launch certain services, either by sending an ambulance or one of the police forces, so it is shown in this tab when that service started, how long it took for that service to arrive at the specific place and when it was terminated.

Finally, a possible improvement would be related to the possible use of Artificial Intelligence applications to try to emulate human behaviour in this type of human interaction tools.

In conclusion, the aim of developing this tool has been to help, as much as possible, one of the work forces that are on the battlefront day after day, giving a possible temporary solution to these emergency centers to relieve the workload of the moment and which can evolve into a permanent use application that is useful for the daily work of these forces.

Apéndice A

Código del proyecto

El código que ha hecho posible la implementación total de la aplicación se ha alojado en los tres repositorios de GitHub que se muestran a continuación.

A.1. Repositorio del código agente virtual Dialogflow

Agente Virtual Dialogflow

A.2. Repositorio del código Back-end

Back-end

A.3. Repositorio del código Front-end

Front-end

Bibliografía

[1] Pascual, Juan Antonio, (2019), *Inteligencia Artificial: qué es, cómo funciona y para qué se está utilizando*.

https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917

[2] Stalman, A. *La ambigüedad de lo humano y la Inteligencia Artificial*.

http://andystalman.com/la-ambiguedad-del-humano-la-inteligencia-artificial/

[3] Enfermedad por nuevo coronavirus, COVID-19,

https://www.mscbs.gob.es

[4] Centro Coordinador de Emergencias y Seguridad (CECOES) 1-1-2 del Gobierno de Canarias,

http://www.112canarias.com/info/

[5] Situación actual COVID-19 España,

https://cnecovid.isciii.es/covid19/

[6] Situación actual COVID-19 Canarias,

https://grafcan1.maps.arcgis.com

[7] Health Bot Service Microsoft Azure,

https://azure.microsoft.com

[8] Providence Health & Services,

https://coronavirus.providence.org/

[9] Gutiérrez, Ó., (2020) *Apple y Google explican su tecnología de seguimiento de contactos*.

https://www.cnet.com/es/noticias/apple-google-colaboracion-covid-19-coronavirus-respuestas/

[10] Fernández de Lara, C., (2020) *Apple y Google liberan API para sistema de rastreo COVID-19*.

https://expansion.mx/tecnologia/2020/04/24/apple-y-google-mejoran-privacidad-de-sistema-de-rastreo-de-covid-19

[11] Asistencia COVID-19,

https://asistencia.covid19.gob.es/

[12] Hernández, María Jesús, (2020) *Canarias idea un sistema para anticiparse a un brote del virus*.

https://www.eldia.es/sociedad/2020/05/11/canarias-idea-sistema-anticiparse-brote/1077440.html

[13] Dialogflow,

https://dialogflow.com/

[14] Heroku,

https://www.heroku.com/

[15] NodeJS,

https://nodejs.org/

[16] ExpressJS,

https://expressjs.com/

[17] JavaScript,

https://developer.mozilla.org/es/docs/Web/JavaScript

[18] BBVA Open4U, (2016) *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*.

https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[19] Elmasri, R., & Navathe, S., (2017). *Fundamentals of database systems* (Vol. 7). Pearson.

[20] MongoDB,

https://www.mongodb.com/

[21] MongooseJS,

https://mongoosejs.com/

[22] Mongoose Field Encryption package,

https://www.npmjs.com/package/mongoose-field-encryption

[23] DigitalOcean,

https://www.digitalocean.com/

[24] Docker,

https://www.docker.com/

[25] Axios package,

https://www.npmjs.com/package/axios

[26] VueJS,

https://vuejs.org/

[27] BootstrapVue,

https://bootstrap-vue.org/

[28] Bootstrap,

https://getbootstrap.com/

[29] Martin, R. C., (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

[30] Portal de Empleo en Informática, Telecomunicaciones y Tecnologías. *Informe Empleo Informática - Mayo 2020*.

https://www.tecnoempleo.com/informe-empleo-informatica.php