



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO
GRADO EN INGENIERIA INFORMÁTICA

POSTCOVID-AI Chatbot

Facilitating data collection of people level human behavior

Autor

Juan del Río Gómez

Directores

Oresti Baños Legrán

Miguel Damas Hermoso



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 2023

POSTCOVID-AI Chatbot

Facilitating data collection of people level human behavior

Autor

Juan del Río Gómez

Directores

Oresti Baños Legrán

Miguel Damas Hermoso

POSTCOVID-AI Chatbot: facilitating data collection of people level human behavior

Juan del Río Gómez

Palabras clave: chatbot, bienestar, cuestionarios, aplicación web

Resumen

Este proyecto se centra en el desarrollo de un agente conversacional que permita la recolección de datos relacionados con el bienestar de las personas. Forma parte del proyecto POSTCOVID-AI como solución escalable a su método de recogida de datos mediante uso de otras tecnologías. El objetivo es facilitar esta recogida de datos para su posterior análisis a través de una herramienta que sea fácil de utilizar y atractiva a todo usuario.

El chatbot guía al usuario a través de preguntas específicas sobre su estado de ánimo, actividades diarias, hábitos de sueño y alimentación entre otros aspectos. Además de recoger datos, el bot está programado mediante un conjunto de instrucciones establecidas que permiten a los usuarios interactuar con él.

Gracias a estas pautas, la persona puede tener una conversación con el bot y a su vez este le irá haciendo una serie de preguntas previamente definidas. Todo esto se puede configurar a través de una aplicación web orientativa a la que se tendrá acceso para poder modificar las preguntas a realizar, agruparlas, añadir mensajes al bot, planificar cuestionarios y consultar los datos en tiempo real.

POSTCOVID-AI Chatbot: facilitating data collection of people level human behavior

Juan del Río Gómez

keywords: chatbot, well being, questionnaires, web application

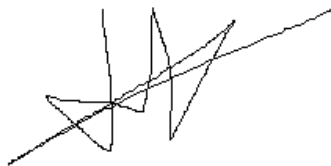
Abstract

This project focuses on the development of a conversational agent that allows the collection of data related to the well-being of people. It is part of the POSTCOVID-AI project as a scalable solution to its method of data collection using other technologies. The objective is to facilitate this data collection for subsequent analysis through a tool that is easy to use and attractive to all users.

The chatbot guides the user through specific questions about their mood, daily activities, sleeping and eating habits among other aspects. In addition to collecting data, the bot is programmed through a set of established instructions that allow users to interact with it.

Thanks to these guidelines, the person can have a conversation with the bot and in turn it will ask a series of previously defined questions. All this can be configured through a web application that can be accessed to modify the questions to be asked, group them, add messages to the bot, plan questionnaires and consult the data in real time.

Yo, **Juan del Río Gómez**, alumno de la titulación GRADO EN INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 46069380N, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.



Fdo: Juan del Río Gómez

Granada a 8 de octubre de 2023 .

D. **Oresti Baños Legrán(tutor1)**, Profesor del Área de Ingeniería de Sistemas y Automática del Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

D. **Miguel Damas Hermoso (tutor2)**, Profesor del Área de de Ingeniería de Sistemas y Automática del Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***POSTCOVID-AI Chatbot***, ha sido realizado bajo su supervisión por **Juan del Río Gómez (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 8 de Octubre de 2023.

Los directores:

Oresti Baños Legrán (tutor1)

Miguel Damas Hermoso(tutor2)

Agradecimientos

En primer lugar quisiera agradecer a mi familia, sin su ayuda hoy no estaría donde estoy.

A todos mis compañeros de carrera y amigos, siempre dispuestos y atentos a ayudarme a resolver problemas del día a día.

Y a todos los profesores que me han aportado experiencias positivas gracias a su enseñanza durante el transcurso de esta bonita etapa.

Índice general

1. Introducción	17
1.1. Motivación	18
1.2. Objetivos	19
1.3. Estructura	20
2. Estado del arte	21
2.1. Chatbots	21
2.1.1. Tipos de chatbots	22
2.1.2. Canales de comunicación	23
2.1.3. Ventajas y desventajas de usar chatbots	24
2.2. Trabajos previos	24
3. Planificación y presupuesto	27
3.1. Planificación	27
3.2. Presupuesto	28
3.2.1. Requerimientos software	28
3.2.2. Recursos humanos	29
4. Diseño	31
4.1. Requerimientos	31
4.2. Arquitectura	32
5. Implementación	35
5.1. Modelos	35
5.2. Aplicación Web	36
5.2.1. Patrón MVT	37
5.2.2. Login	38
5.2.3. Home	40
5.2.4. Gestores	41
5.2.5. Preguntas	41
5.3. Chatbot	50
5.3.1. Módulo de bienvenida y registro	50
5.3.2. Módulo conversacional	51
5.3.3. Módulo cuestionario	55

5.4. Tecnologías	58
6. Despliegue e instalación	61
6.1. Despliegue	61
6.2. Instalación	63
6.2.1. Configuración del entorno	63
6.2.2. Montaje de desarrollo	64
7. Conclusiones	67
7.1. Aspectos a mejorar	68
7.2. Valoración personal	68
A. Guía de Uso	69
A.1. Ingreso	69
A.2. Gestión	70
A.2.1. Administración de preguntas	70
A.2.2. Administración de Preámbulos	71
A.2.3. Administración de Bloques	72
A.2.4. Respondiendo al cuestionario	73
A.2.5. Administración de Respuestas	75

Índice de figuras

1.1. Comparación de personas con problemas mentales antes y durante la pandemia por (<i>“Impacto de la covid-19 en la salud física y mental de la población adulta española” 2021</i>)	18
2.1. Crecimiento del uso de chatbots en los últimos años (<i>“Código Chatbot: cuando la humanidad se robotiza” 2023</i>)	22
3.1. Planificación temporal	28
4.1. <i>POSTCOVID-AI TELEGRAMBOT</i> arquitectura	33
5.1. Diagrama de clases	35
5.2. Diagrama tabla usuarios	38
5.3. Login usuarios	38
5.4. Formulario de registro	39
5.5. Home	40
5.6. Actualización usuario	41
5.7. Buscador	41
5.8. Listado de preguntas	42
5.9. Añadir pregunta	43
5.10. Modificar Pregunta	43
5.11. Borrar Pregunta	44
5.12. Listado de Bloques	44
5.13. Añadir bloque	45
5.14. Planificación Bloque	47
5.15. Editar Preámbulo	47
5.16. Lista respuestas	48
5.17. Mensaje de bienvenida	51
5.18. Respuesta del bot a un mensaje no entendido	53
5.19. Respuestas del bot a las preguntas frecuentes	53
5.20. Mensaje cuestionario pendiente	54
5.21. Cuestionario de preguntas	56
5.22. Mensaje del bot a una respuesta incorrecta	57
5.23. Diagrama de flujo del bot	58

6.1. Creación del bot en Telegram	62
6.2. Customización del bot	62
A.1. Vista de registro	70
A.2. Vista de creación de pregunta	71
A.3. Vista de listado de preguntas	71
A.4. Vista creación de preámbulo	72
A.5. Vista creación de bloques	73
A.6. Vista listado de bloques	73
A.7. Introducción del bot	73
A.8. Cuestionario	74
A.9. Ejemplo de conversación	74
A.10.Vista listado de respuestas	75
A.11.Respuestas CSV	75

Capítulo 1

Introducción

En la historia de la humanidad surgen momentos y etapas que trascienden lo cotidiano y se convierten en hitos definitorios de una era. Uno de estos hitos que han marcado una etapa adversa en la historia moderna ha sido la pandemia de COVID-19, cuyos síntomas han afectado en cada rincón del planeta y han sacudido los cimientos de nuestra existencia de manera profunda.

Cada uno de nosotros en cierta medida, hemos experimentado un crisol de emociones y desafíos que invitan a la introspección y la búsqueda de significado a la adversidad. La soledad ocasionada por el confinamiento y alejamiento de nuestras personas queridas, la ansiedad, la depresión y el estrés se volvieron parte del día a día de muchas personas. En medio de todo este revuelo, la mente humana, enrevesada, reaccionó planteándose algunas preguntas existenciales ¿Qué valores importan en tiempos de crisis? ¿Cómo podemos encontrar la fortaleza para luchar contra la incertidumbre?

Esto ha provocado que la población haya tenido que experimentar un periodo de adaptación a esta nueva normalidad. Ha habido un gran cambio en el desarrollo del cerebro de la población juvenil provocado por el confinamiento, la vulnerabilidad psicosocial y aumento a la exposición a las redes sociales. Como vemos en la *Figura 1.1* esto afecta a las habilidades socioemocionales, generando conductas de agresividad, falta de empatía, ansiedad, síntomas depresivos, dificultades para la resolución de conflictos (*Elizabeth Muller 2020*).

Y no solo en este sentido, ya que el bienestar se refiere a un estado general de satisfacción y equilibrio en la vida y se compone de varios aspectos, como la salud física, la salud mental, la calidad de vida y el sentido de propósito y conexión con los demás (*HIFAS 2023*). Por lo que debemos tener en cuenta que el bienestar no es un aspecto individual, sino que se encuentra asociado al entorno en el que vivimos y que nos ayuda a hacer frente a los factores

estresantes de la vida, a aprender y trabajar bien, y a contribuir a nuestra comunidad.

En definitiva, la importancia del bienestar se ha vuelto crucial, ya que la pandemia ha afectado significativamente a la calidad de vida de las personas. Es esencial abordar el bienestar como una cuestión de interés público y trabajar juntos para construir una sociedad más saludable y equitativa.

Este proyecto nace en este contexto, con el objetivo de contribuir a la mejora de la salud mental a través de la tecnología, ofreciendo un medio cómodo para la recogida de datos que permita la detección de tendencias relacionadas con problemas de salud mental y nos ayude a combatir este problema.

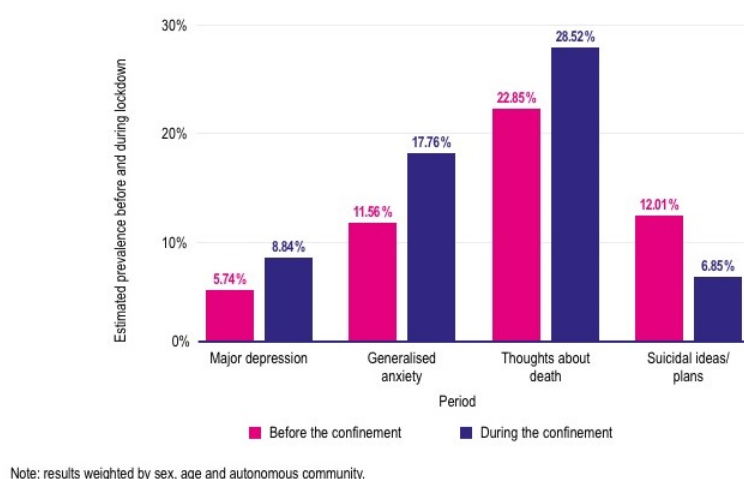


Figura 1.1: Comparación de personas con problemas mentales antes y durante la pandemia por (“*Impacto de la covid-19 en la salud física y mental de la población adulta española*” 2021)

1.1. Motivación

La nueva normalidad socioeconómica tras el brote de COVID-19 plantea desafíos sin precedentes tanto para los gobiernos como para los ciudadanos. Desde el inicio de la pandemia los gobiernos han tenido acceso a diversos datos económicos, pero la información precisa sobre cómo la nueva normalidad y las políticas específicas influyen en la conducta individual y grupal y, en consecuencia, en el bienestar de la sociedad, es escasa, indirecta y abierta a la interpretación. Como argumenta (*Pena Trapero 2009*) son necesarios datos que proporcionen un indicador cuantitativo confiable de cómo las diferentes medidas y su impacto en el contexto económico, social y conductual

afectan el bienestar general de la población. Gracias al uso masivo de teléfonos inteligentes podemos recoger datos ubicuos, continuos y anónimos de la población, así como datos emocionales, sociales, conductuales y relativos al bienestar.

En este contexto, la creación de un agente conversacional en (*Telegram*) se nos presenta como una ingeniosa solución para esta recolección de datos. La motivación de este proyecto radica en la importancia de poder brindar al usuario un espacio confidencial y seguro a través de conversaciones interactivas y personalizadas para que las personas puedan expresar sus emociones sin ningún tipo de pensamiento crítico. Además Telegram consta de un diseño atractivo permitiendo la integración de cuestionarios, encuestas, enlaces, lo que facilita el acceso a recursos de apoyo.

El objetivo del bot es la recaudación de datos de usuarios mediante el uso de un agente conversacional como alternativa al tradicional cuestionario de preguntas. Para que al usuario le sea atractivo responder este tipo de cuestionarios que usualmente suelen ser un poco monótonos, se ha implementado una parte de interacción básica. Leyendo varias fuentes y opiniones de usuarios acerca de su experiencia con chatbots, en (*kettle y Lee 2023*) llegan a la conclusión de que a las personas entrevistadas les es mucho más fácil poder abrirse y contestar preguntas acerca de su salud con un chatbot con el que pudieran conversar sabiendo que no hay lugar a ninguna crítica ni prejuicio por su parte, además de ser una experiencia calificada como más entretenida y divertida que realizar un cuestionario. Partiendo de esta base y con esa idea se ha desarrollado este proyecto.

Toda esta información recogida puede ser muy útil para investigadores y profesionales del sector, ya que proporcionan una visión más precisa de las necesidades y desafíos a los que se enfrentan los ciudadanos.

"De vez en cuando, una nueva tecnología, un antiguo problema y una gran idea se convierten en una innovación"

– Dean Kamen. Creador del Segway y el iBOT.

1.2. Objetivos

Este proyecto tiene como objetivo el desarrollo de un chatbot con el que recabar información personal sobre aspectos claves en el bienestar de las personas, de forma que nos permita clasificarla y estructurarla para generar un análisis.

Para lograr este fin se plantean los siguientes objetivos específicos:

- Creación de un chatbot en Telegram que nos permita la realización de

cuestionarios interactivos.

- Desarrollo de una parte interactiva entre el chatbot y el usuario, mediante la clasificación de diferentes expresiones de comunicación previamente definidas.
- Desarrollo de una aplicación web destinada a la creación, gestión y control de cuestionarios, mensajes mostrados por el bot, planificación de estos cuestionarios, administración de usuarios y visualización de respuestas.
- Normalización de los datos obtenidos en un formato estructurado para garantizar que estén en un estado óptimo para su posterior utilización.

1.3. Estructura

En este primer capítulo hemos visto una introducción del proyecto, comentando la motivación del mismo y los objetivos a cumplir a lo largo de este desarrollo. En el segundo capítulo, nos pondremos un poco más en contexto y veremos ejemplos de chatbots en la actualidad relacionados con nuestro tema e intentaremos entender como funcionan, como nos ayudan y forman parte de nuestro día a día.

En el tercer y cuarto capítulo hablaremos sobre la planificación y etapas seguidas en el ciclo de vida del proyecto. Indagaremos en las tecnologías usadas, los requerimientos necesarios de nuestro trabajo, exponiendo las funcionalidades de todas las partes implicadas, problemas a abordar y como satisfacerlos. Seguidamente en los capítulos posteriores encontraremos toda la información necesaria para entender el desarrollo y la implementación del proyecto, para finalizar con la puesta a punto, conclusiones, aspectos a mejorar y una guía de uso para el usuario.

Capítulo 2

Estado del arte

2.1. Chatbots

Un chatbot o agente conversacional (*Adamopoulou y Moussiades 2020*) es un programa inteligente que mediante lenguaje escrito u oral pueden interactuar con los usuarios a través de una conversación de lenguaje natural.

La Inteligencia Artificial se integra cada vez más en nuestra vida cotidiana con la creación y el análisis de software y hardware inteligentes, llamados agentes inteligentes. Los agentes inteligentes pueden realizar una gran variedad de tareas, desde trabajos manuales hasta operaciones sofisticadas. Un chatbot es un ejemplo típico de sistema de IA y uno de los ejemplos más elementales y extendidos de Interacción Persona-Ordenador inteligente. Se trata de un programa informático que responde como una entidad inteligente cuando se conversa con él a través de texto o voz y que entiende uno o más lenguajes humanos mediante el Procesamiento del Lenguaje Natural (PLN). En el léxico, un chatbot se define como 'Un programa informático diseñado para simular una conversación con usuarios humanos, especialmente a través de Internet' (*English 2023*).

En las últimas décadas, el uso de chatbots ha tenido un crecimiento bastante significativo como vemos en la *Figura 2.1*. En el ámbito empresarial los chatbots se han convertido en una forma cada vez más popular de que las empresas interactúen con sus clientes (*Journal TechInnovation 2022*). Proporcionan una forma cómoda y eficaz de obtener respuestas a las consultas de los clientes, ayudarles a realizar compras e incluso ofrecerles recomendaciones personalizadas. Como resultado, cada vez más empresas están implementando chatbots en sus operaciones de atención al cliente. En ese artículo se analizan las diversas formas en que las empresas están aprovechando la tecnología de chatbot para mejorar su experiencia de servicio al cliente y cómo se pueden utilizar también en otras áreas de negocio.

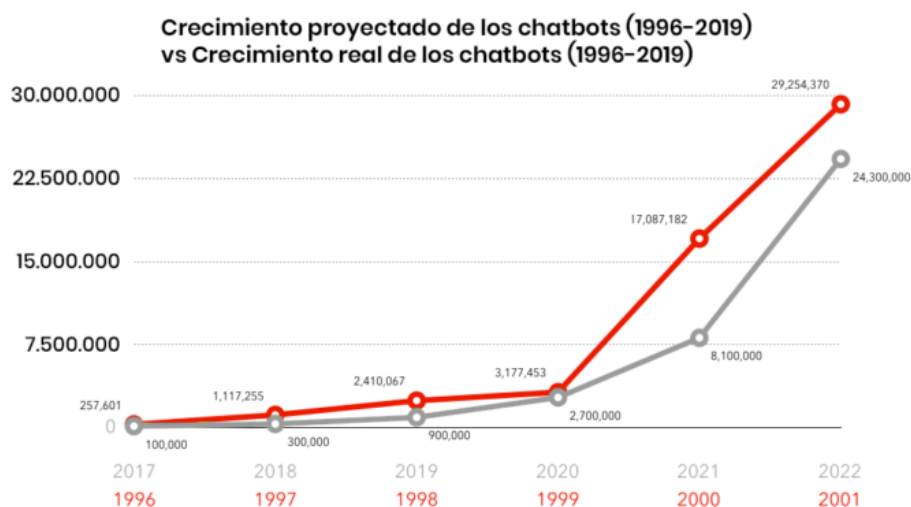


Figura 2.1: Crecimiento del uso de chatbots en los últimos años (*“Código Chatbot: cuando la humanidad se robotiza” 2023*)

2.1.1. Tipos de chatbots

Un chatbot funciona de un par de maneras: con directrices establecidas y mediante aprendizaje automático (ML).

Set Guidelines Chatbots: Un chatbot que funciona con un conjunto de directrices establecidas está limitado en su conversación. Sólo puede responder a un número determinado de peticiones y vocabulario y es tan inteligente como su código de programación. Un ejemplo de bot limitado es un bot bancario automatizado que hace algunas preguntas a la persona que llama para entender lo que quiere hacer.

Machine Learning Chatbots: Un chatbot que funciona mediante aprendizaje automático tiene una red neuronal artificial inspirada en los nodos neuronales del cerebro humano. El bot está programado para autoaprender a medida que se le presentan nuevos diálogos y palabras. En efecto, a medida que un chatbot recibe nuevos diálogos de voz o textuales, aumenta el número de consultas a las que puede responder y la precisión de cada respuesta que da.

Los chatbots se utilizan en diferentes sectores y se crean con diferentes fines. Hay bots de venta al por menor diseñados para recoger y encargar la compra, bots meteorológicos que dan la previsión del tiempo del día o de la semana, y bots amistosos que simplemente hablan con la gente que necesita un amigo.

2.1.2. Canales de comunicación

Los canales de comunicación desempeñan un papel importante a la hora de ayudar a las marcas y a los clientes a conectarse entre sí para diversas formas de compromiso e interacciones. La elección del canal adecuado es vital para que nuestro chatbot proporcione una experiencia efectiva y eficiente en los usuarios. Algunos de los más usados son:

Sitio Web y App móvil: Uno de los canales más populares es en el sitio web de la empresa o en algunos casos en la App que dispongan. El chatbot puede responder preguntas frecuentes de los clientes y proporcionar asistencia en tiempo real.

Facebook: Otra forma es integrándolo el chatbot en una página de Facebook. A través de Facebook Messenger los clientes pueden recibir respuestas automáticas al momento. Además Facebook brinda la oportunidad de integrar el chatbot en tu página para que responda comentarios de tu muro de forma automática.

WhatsApp: Creando una cuenta en WhatsApp Business API, el bot puede responder instantáneamente al usuario y segmentar las conversaciones.

Telegram: Surge como alternativa a la preeminente WhatsApp. Desde entonces, ha sido percibida como una aplicación más segura que su rival, debido a sus políticas de encriptación. Además permite la creación de bots en grupos de hasta 100.000 personas.

Google My Business: Permite chatear con cualquier persona que encuentre tu negocio a través de Google.

Al fin y al cabo, no hay ningún canal de comunicación mejor que otro para implementar chatbots. Cada uno tiene sus beneficios y te puede dar más o menos facilidades, pero lo importante es elegir el canal que más se adecue a tu necesidad y más sencillo le resulte al cliente. Debe responder a la pregunta de ¿Para que quiero implementar un chatbot y cómo me va a ayudar?. Imaginemos que tienes una academia de inglés, y tu forma de venderte y mostrarte al público es a través de tu página web y redes sociales. Sería inútil que hicieras un chatbot en Telegram, y con la funcionalidad añadida de el trabajo adicional que tendría que hacer el cliente para contactar con él, que casi nadie haría, si ofreces tus servicios en tu sitio web. Sería mejor implementarlo en tu página, para que cada persona que tenga cualquier duda acerca de ¿Qué horario tienes las clases por la tarde? o ¿Cuanto cuesta cada clase? o infinidad de preguntas frecuentes que debas someterte a responder

en tu día a día, puedan preguntarlo directamente al chatbot y le sea fácil al usuario e incluso una experiencia positiva.

2.1.3. Ventajas y desventajas de usar chatbots

Ventajas	Desventajas
<ul style="list-style-type: none">- Menor coste que trabajadores humanos- Online 24/7- Puede usarse como herramienta de ventas, marketing y recopilación de información	<ul style="list-style-type: none">- Puede no entender las consultas de los usuarios- Carece de emoción y no es personalizable- Necesitan mantenimiento constante

2.2. Trabajos previos

Los chatbots se pueden aplicar a diferentes áreas. En el caso de salud mental podemos ver bots conversacionales que actúan como psicólogos para las personas como por ejemplo (*Danielle Ramo 2019*). Proponen un bot conversacional para proporcionar habilidades de psicología positiva y promover el bienestar entre los jóvenes después del tratamiento del cáncer. Reclutaron a adultos jóvenes en los 5 años siguientes a la finalización de un tratamiento activo contra el cáncer a una prueba de 4 semanas para probar la efectividad del bot. Este incluía 4 semanas de habilidades de psicología positiva, valoraciones diarias de emociones, vídeos y otros materiales. Los análisis examinaron la participación del chatbot y los comentarios abiertos sobre la simpatía y la utilidad percibida, y compararon los grupos experimental y de control con respecto a los síntomas de ansiedad y depresión y los cambios en las emociones positivas y negativas entre el inicio y las 4 semanas. Los participantes calificaron su experiencia de útil y recomendada. Los comentarios abiertos señalaron su naturaleza no crítica como una ventaja particular del chatbot.

Otro ejemplo de bot conversacional orientado a la salud es (*Williams 2021*) que plantea un agente virtual para ayudar a adolescentes a afrontar los cambios importantes en su vida, como terminar los estudios o la formación, iniciar una carrera profesional o entablar una relación íntima que pueden desencadenar o amplificar problemas de salud mental subyacentes, lo que a veces provoca un malestar psicológico o un funcionamiento inadaptado.

(*“Moodfit” 2016*) es una aplicación que pretende ayudar a los consumidores a comprender y mejorar su estado de ánimo, aumentar su resiliencia y alcanzar objetivos. Utiliza principios cognitivo-conductuales (TCC) como el registro de pensamientos, la atención plena, la meditación y el diario de gratitud para tratar las fluctuaciones del estado de ánimo que pueden deberse a la depresión, el estrés y la ansiedad. Los consumidores pueden crear sus propios objetivos o elegir entre los predeterminados, como estado de ánimo, sueño, gratitud y nutrición. La pantalla de inicio muestra el porcentaje de objetivos que el consumidor ha alcanzado junto con cuántos días ha guardado su estado de ánimo de forma consecutiva. También hay recordatorios que ofrecen consejos para mejorar el estado de ánimo, así como una sección que hace un seguimiento del progreso del usuario con gráficos.

(*“Moodpath” 2016*) es un chatbot diseñado para evaluar el estado de ánimo y la salud mental de los usuarios a lo largo del tiempo. Realiza cuestionarios diarios para recopilar información sobre los síntomas de depresión y ansiedad con los que proporciona informes detallados y sugerencias basadas en los resultados. Ayuda al usuario a seguir, controlar y comprender sus quejas de forma estructurada para intentar descubrir que comportamiento negativo podría estar relacionado con un peor bienestar.

Escasez de proveedores, citas limitadas, largas listas de espera: los obstáculos para prestar atención de salud mental son enormes y dificultan el progreso de las iniciativas de equidad sanitaria. Esta es la premisa con la que se creó (*“Woebot” 2017*), con la idea de ayudar tanto a profesionales como a usuarios. Ofrece apoyo para la salud mental y recopila información sobre el bienestar emocional de los usuarios a lo largo de conversaciones. Esto permite adaptar sus interacciones y recomendaciones según las necesidades de cada individuo.

Capítulo 3

Planificación y presupuesto

En el mundo de la gestión de proyectos, nos encontramos con varias etapas que guían el rumbo y éxito de cualquier iniciativa. Es un ciclo continuo con diversas fases interconectadas conocidas como el 'Ciclo de vida de un proyecto'.

En esta sección haremos un recorrido comentando el ciclo de vida del proyecto, desglosando la planificación, y haciendo una estimación sobre el coste que supone el desarrollo del mismo.

3.1. Planificación

La planificación de un proyecto tiene como objetivo su realización en las condiciones ideales y asegurando los mejores resultados. Este ciclo de vida es un marco conceptual que divide la evolución de un proyecto en etapas secuenciales, cada una con sus diferentes objetivos. Para el desarrollo de este proyecto se han seguido las siguientes etapas.

- **Estudio** de distintos tipos de chatbots y trabajos similares que nos servirán como ejemplo a la hora de definir la funcionalidad del proyecto. Realización de cursos y estudio de la documentación de las tecnologías que vamos a usar.
- **Documentación** de la memoria realizada concurrentemente en todas las etapas del proyecto.
- **Análisis** del proyecto, identificando los requisitos, entidades y componentes, así como la forma de interactuar entre sí.
- **Diseño** de la base de datos, generación de mockups para las interfaces de usuario y arquitectura.

- **Implementación** del software teniendo en cuenta todo los aspectos estudiados en las etapas previas.
- **Evaluación y corrección de errores** realizando pruebas constantes y exponiendo todos los casos de uso posibles para la detección de errores y su correcto arreglo.
- **Despliegue** del proyecto en un contenedor software y subida a github para que su instalación sea lo más sencilla posible.

Planificación	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre
Documentación								
Estudio								
Análisis								
Diseño								
Implementación								
Evaluación y corrección								
Despliegue								

Figura 3.1: Planificación temporal

3.2. Presupuesto

3.2.1. Requerimientos software

Todos los recursos software y tecnologías utilizadas en el proyecto son gratuitos. Partiendo desde el IDE utilizado en mi caso Visual Studio Code, como los framework Django y Foundation, Docker, todas las librerías, e incluso la conexión a la API de Telegram. Además se encuentra subido en un repositorio público de github, que nos ayuda a realizar copias y controlar versiones.

El único gasto en recursos sería el despliegue del proyecto en un servidor. Si quisieramos migrarlo a la nube en Amazon Web Services un precio medio teniendo en cuenta varios factores como el tamaño de la instancia EC2, capacidad de almacenamiento de la base de datos y transferencia de datos nos costaría aproximadamente entre 40 y 70 euros por mes según (*“AWS” 2022*), que es lo que cuesta la arquitectura más básica.

Cabe destacar que Amazon cobra toda la transferencia de datos que sucede por fuera de su misma red, es decir, cada dato que nosotros queramos mover dentro y fuera de AWS nos costará.

3.2.2. Recursos humanos

Teniendo en cuenta que este proyecto es desarrollado única y exclusivamente por mí, el coste de recursos humanos, contando las horas efectivas trabajadas en el proyecto por el autor y estableciendo un salario base de un programador de unos 14,62€ la hora según (*Talent 2022*) sería el siguiente:

Actividad	Horas	Coste
Análisis y diseño	25h	365€
Implementación	200h	2924€
Evaluación y corrección	80h	1169€
Despliegue	10h	146€
Documentación	50h	731€
Total	365h	5336€

Cuadro 3.1: Tabla presupuesto

Capítulo 4

Diseño

4.1. Requerimientos

Antes de empezar con el desarrollo de nuestro proyecto, debemos definir la funcionalidad del mismo explicando detalladamente como sería el comportamiento de cada una de las partes implicadas.

- **Web:** Creación de una interfaz con la que poder interaccionar con las preguntas del bot y mensajes a mostrar.
 - Implementación de un login y un registro para que solo los usuarios registrados puedan acceder al panel de control.
 - Panel de control, primera vista del usuario que funciona como guía para las operaciones a realizar.
 - Apartado de preguntas, bloques, preámbulos, respuestas y gestión de usuarios.
 - Vistas de detalle y operaciones de modificación, clonado y borrado de cada uno de los apartados anteriores.
 - Autenticación y permisos en la realización de operaciones solo para usuarios específicos.
 - Descarga de las respuestas proporcionadas por los usuarios en formato CSV.
- **Chatbot:** Creación de un chatbot en línea que realice un cuestionario de preguntas y a su vez una lógica interna que permita una interacción con el usuario.
 - Mensaje de bienvenida al usuario cuando entre en el chat del bot.

- Almacenamiento del usuario como un nuevo integrante de nuestro sistema si ningún tipo de cuestionario gracias a las funciones específicas de Telegram que permiten la obtención de los datos del usuario que se encuentra activo en el chat.
- Entorno de interacción por parte del bot con el usuario para poder tener una conversación lo más humana posible mediante el análisis de ciertas palabras e ideas previamente establecidas.
- Cuestionario de preguntas que se encuentran activas para los usuarios en un momento concreto. Estas preguntas varían dependiendo de la planificación de los cuestionarios y la frecuencia de cada uno de ellos.
- Recolección y almacenamiento de las respuestas proporcionadas en la base de datos de forma automática.
- Creación de avisos a todos los usuarios cuando haya cuestionarios pendientes, dependiendo de la hora y frecuencia de cada uno de ellos.
- Realización de este proceso de forma cíclica sin que el usuario pierda el hilo conductor de la conversación y además que cada proceso sea totalmente independiente para cada usuario.

4.2. Arquitectura

El objetivo del proyecto es conseguir información estructurada a partir de las experiencias y emociones de los usuarios. Con este objetivo se ha diseñado una arquitectura que consta de tres componentes principales: aplicación web, chatbot y base de datos. La aplicación web actúa como panel de control para que el administrador tenga acceso a la gestión de los cuestionarios, el chatbot se encarga de interactuar con los usuarios realizando preguntas con el objetivo de obtener respuestas. Por último la base de datos que almacena los datos relevantes y permite un fácil acceso a esta información. Gracias a esto se consigue una solución integral que combina la facilidad de uso del chatbot en Telegram con la capacidad de gestión y análisis dada por la aplicación web.

En la *Figura 4.2*, se presenta de manera detallada la arquitectura que hemos empleado en nuestro proyecto. Esta representación gráfica es fundamental para comprender la estructura subyacente de nuestro sistema y cómo se interconectan sus diversos componentes. A través de esta figura, se destila una visión completa de la disposición y relación de los elementos clave que conforman nuestra solución.

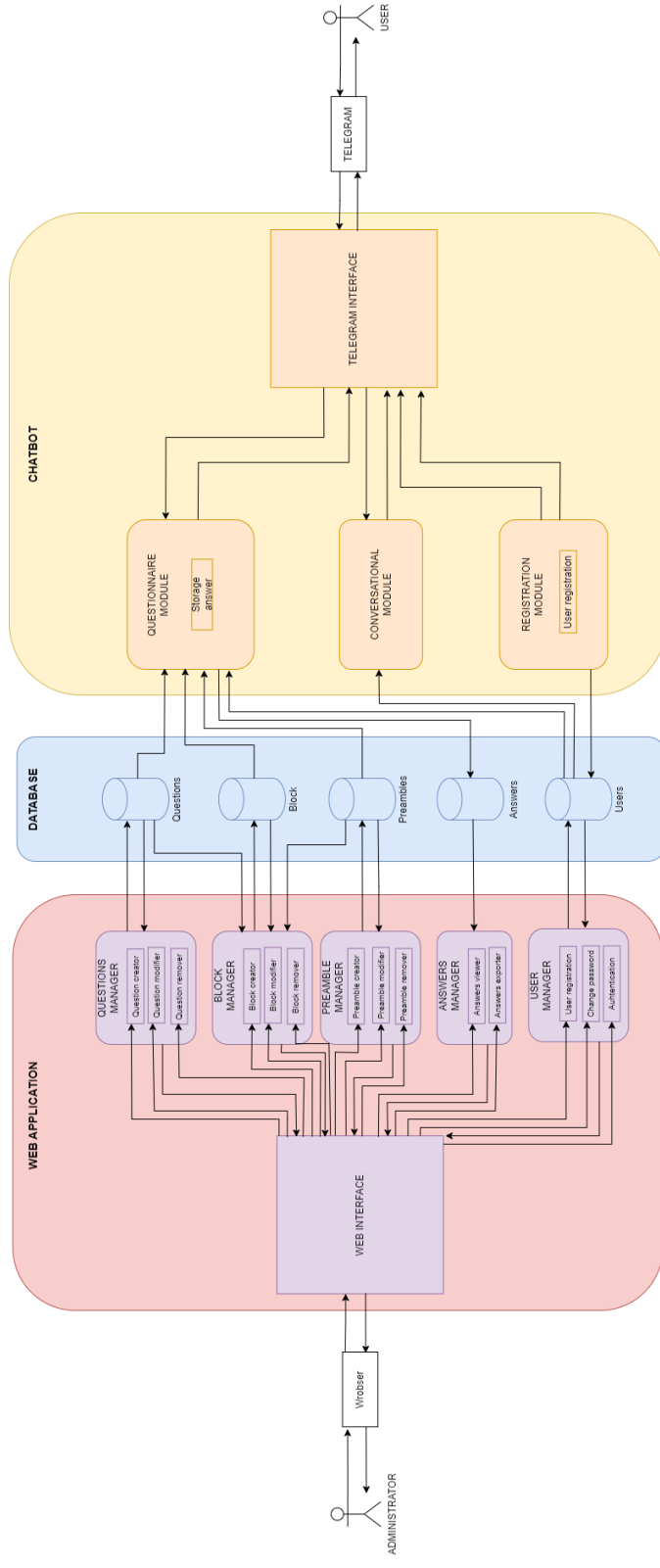


Figura 4.1: *POSTCOVID-AI TELEGRAMBOT* arquitectura

La base de datos se encuentra estructurada en cinco modelos: preguntas, preámbulos, bloques, respuestas y usuarios. Posteriormente se analizarán con más detalle.

La aplicación web esta formada por cinco módulos. El primero es el question manager, que permite la creación, modificación y borrado de preguntas. Seguido del preamble manager, el cual tiene la misma funcionalidad que el anterior pero con los preámbulos. Después se encuentra el block manager, que también consta con las operaciones básicas para la gestión de los bloques. Tanto las preguntas como los preámbulos interactúan con este módulo. En cuarto lugar se encuentra el answer manager, que guarda las respuestas obtenidas por los usuarios e interactúa con el último módulo. User manager, contiene la información de los usuarios registrados en nuestro sistema.

La parte del chatbot está formada por tres módulos. Un primer módulo de bienvenida y registro, donde el bot se presenta al usuario y lo guarda como un nuevo registro en la base de datos. Un segundo módulo que actúa como agente conversacional. Y por último, un tercer módulo que representa el cuestionario de preguntas. Este módulo interactúa con las preguntas, bloques y preámbulos creados a través de la aplicación web. Recoge las respuestas de los usuarios y las guarda en la base de datos.

Esta arquitectura permite la creación, gestión y control de cuestionarios interactivos, mensajes generados, administración de usuarios a través de la aplicación web y la recogida y clasificación de respuestas.

Capítulo 5

Implementación

5.1. Modelos

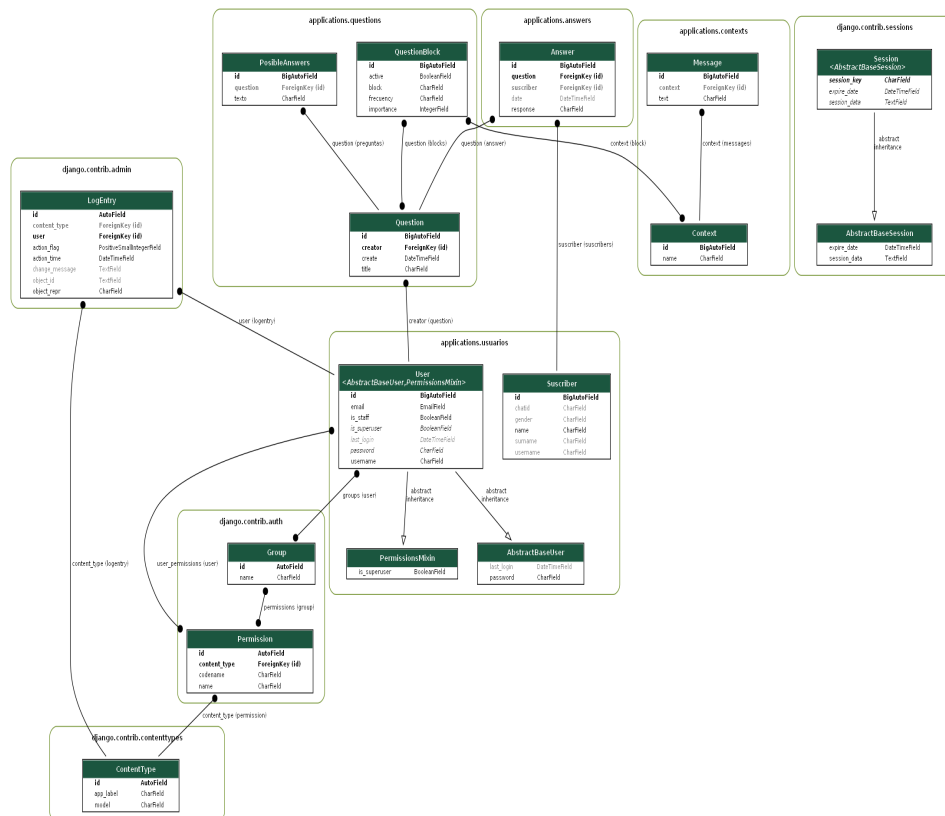


Figura 5.1: Diagrama de clases

En la *Figura 5.1* se ve el diagrama de clases que muestra las relaciones entre los modelos creados en el sistema. Siguiendo el modelo de la ORM de Django (*“Django ORM” s.f.*), este nos permite interactuar con bases de datos relacionales de una manera más orientada a objetos. Esta base de datos consta de cinco modelos principales:

- **Preguntas:** Contiene la información de todas las preguntas almacenadas. Esta se divide en dos: Pregunta y Posibles respuestas. Dentro de la primera especificaría el título de la pregunta junto con otros campos como la fecha o usuario de creación y la segunda contiene las respuestas a esa pregunta.
- **Bloques de Preguntas:** Las preguntas se estructuran en bloques. Cada bloque puede tener las preguntas que desee, junto a otros atributos para que permitan su planificación. Los bloques también tienen asociado un preámbulo.
- **Preámbulos:** Dentro de los preámbulos se guardan los mensajes a mostrar por el bot cuando se hable de un tema concreto. Si un bloque tiene asociado un preámbulo, en el momento que se realice el cuestionario asociado a ese bloque de preguntas el bot mostrará cualquiera de los mensajes de ese preámbulo de forma aleatoria. De esta forma nos aseguramos la interactividad y que la experiencia sea diferente para cada usuario.
- **Usuarios:** Guarda los agentes registrados en nuestro sistema. Se divide en dos tipos: los administradores que tienen acceso al panel de control y los usuarios comunes que son los que interactúan con el bot.
- **Respuestas:** Contiene las respuestas de los usuarios a las preguntas. Solo se almacenan las respuestas posibles de cada pregunta para así confirmar que la información sea correcta.

5.2. Aplicación Web

La aplicación web se ha creado con el objetivo de proporcionar una interfaz atractiva y manejable usada como panel de administración de forma que podamos interactuar con los cuestionarios, respuestas e información mostrada en el bot. Esta solución garantiza la integridad y el almacenamiento seguro de los datos almacenados en una base de datos centralizada.

La aplicación web funciona como intermediario entre el usuario y el bot. Esta se encarga de la creación de cuestionarios, del procesamiento y visualización de las repuestas de manera organizada, lo que facilita su posterior análisis. Nace como solución a satisfacer la importancia de la automatización

y agilidad en la recopilación de datos, logrando un enfoque integral para la administración y respuestas de usuarios.

El chatbot se ha pensado para ser utilizado por los usuarios. Sin embargo, la aplicación solo podrá ser usada por el administrador o las personas autorizadas para su uso. En este capítulo se explican los pasos seguidos para su desarrollo.⁹

5.2.1. Patrón MVT

Django es un framework basado en el modelo MVC (Model-View-Controller). MVC es un patrón arquitectónico que separa una aplicación en tres componentes lógicos principales: el modelo, la vista y el controlador. Cada uno de estos componentes está diseñado para manejar aspectos de desarrollo específicos de una aplicación. Además, MVC es uno de los marcos de desarrollo web estándar de la industria más utilizados para crear proyectos escalables y extensibles.

Django implementa este patrón MVC de una manera peculiar y con algunas variaciones que ellos llaman MTV, que viene siendo la de Model, Template, View (*Gore 2021*).

- M significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- T significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.
- V significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelos y las plantillas.

Esta es la lógica seguida para el desarrollo. Si nos adentramos en el código, vemos que existe una carpeta llamada *'applications'* que contiene varias subcarpetas y en la que cada una es una tabla diferente en la base de datos. Dentro de estas subcarpetas es donde estan creadas las operaciones relacionadas con los modelos y las vistas. En un nivel superior encontramos la carpeta *'templates'*, que incluye las plantillas que dan funcionalidad a estas vistas, tambien separadas por modelos. Y por último dentro de la carpeta *'static'* se encuentran los archivos que dan apariencia a estas plantillas.

5.2.2. Login

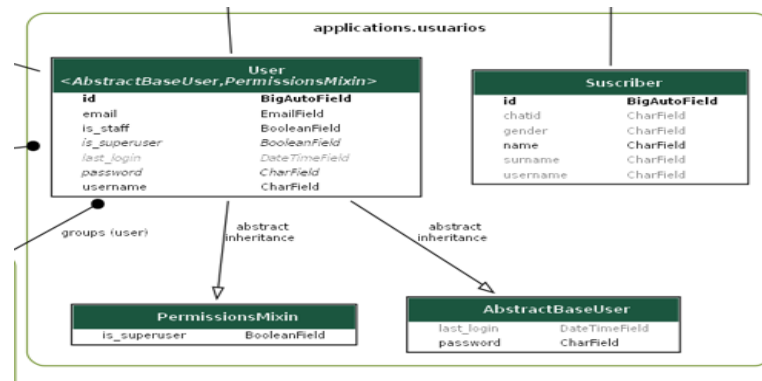


Figura 5.2: Diagrama tabla usuarios

Existen 2 tipos de usuarios en el sistema como vemos en la *Figura 5.2*

El primer usuario, identificado como *User* que actúa como administrador, y es el que cuenta con todos los privilegios para poder acceder y modificar las tablas de la base de datos además de poder acceder al sistema. Y el segundo tipo *Suscriber* que son los usuarios registrados por el bot sin ningún tipo de autoridad.

En Django, las operaciones asociadas a los usuarios se manejan a través del sistema de autenticación que tiene incorporado. Permite manejar cuentas de usuario, grupos, permisos, inicios y cierres de sesión, reestablecimiento de contraseñas y sesiones de usuario basadas en cookies. Todo esto ya viene implementado por defecto, pero proporciona opciones para reemplazar y así ampliarlo y personalizarlo para satisfacer las necesidades de tu proyecto.

Figura 5.3: Login usuarios

El login, que se ve en la *Figura 5.3*, es la primera página que se muestra cuando intentamos entrar. Solo pueden acceder a la aplicación los usuarios registrados.

En caso de no estar registrado se provee de un formulario para ello como se muestra en la *Figura 5.4* . Este rellena los campos de la tabla User que crea un nuevo registro de usuario en el sistema para que pueda autenticarse y acceder a él. Además otorga todos los permisos de administrador para que pueda realizar cualquier tipo de acción.

The figure consists of two screenshots of a web application's registration form, titled "Sign Up".

The top screenshot shows the form in its initial state. It has four input fields: "Username", "Email" (containing "User@gmail.com"), "Password", and "Repeat password". Below these fields is a blue button labeled "Register".

The bottom screenshot shows the form after some data has been entered. The "Username" field contains "juanito" and the "Email" field contains "juanitodrg@correo.ugr.es". The "Password" and "Repeat password" fields are filled with masked characters (dots). A red error message is displayed above the password fields: "Password must contains at least 8 characters". The "Register" button remains at the bottom.

Figura 5.4: Formulario de registro

Cada usuario que desee registrarse deberá especificar un username, email y la contraseña. El formulario cuenta con validaciones como que el nombre de usuario no se encuentre repetido en el sistema, escribir la contraseña dos veces y que coincidan o que esta tenga más de 8 caracteres.

5.2.3. Home

Una vez logueados en el sistema, nos redirige al home. El home es la vista principal de la aplicación, la primera página que los usuarios ven cuando acceden al sitio web *Figura 5.5*. Proporciona información clave y navegación a otras secciones. Este consta de 4 secciones principales: Preguntas, Bloques, Preámbulos, Respuestas.

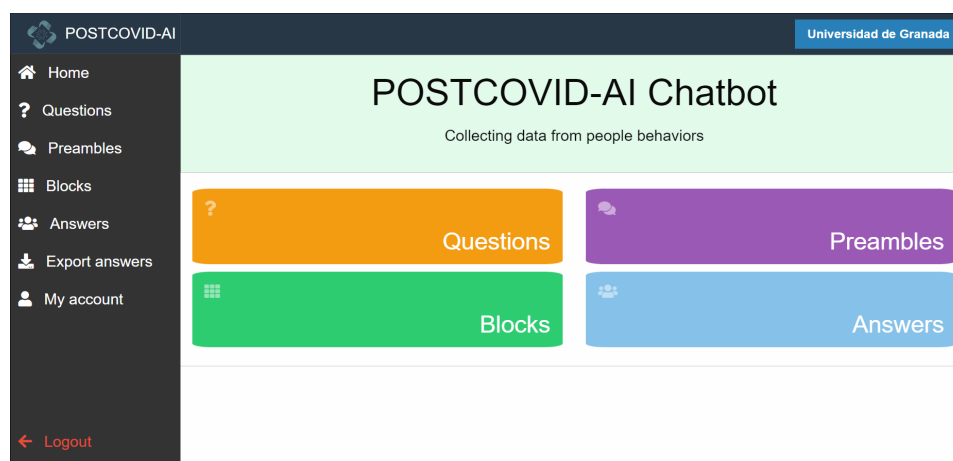


Figura 5.5: Home

Todas las vistas además cuentan con una barra lateral desplegable para realizar algunas operaciones de manera rápida como la redirección a las otras vistas, exportar respuestas, modificar la información del usuario y hacer logout.

Por ejemplo, en la *Figura 5.6* se muestra la vista de modificar usuario donde nos muestra un formulario con los datos del usuario que se encuentra activo y nos da la posibilidad de cambiar la contraseña, pudiendo proporcionar una nueva.

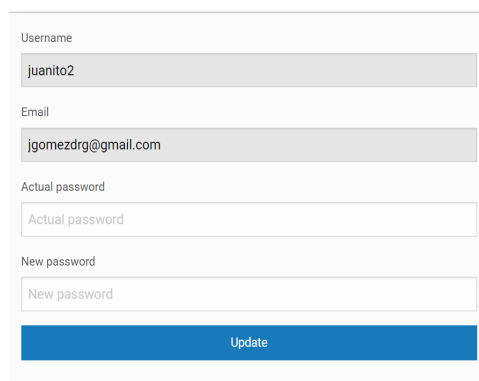
El formulario de actualización de usuario está organizado verticalmente. Comienza con el campo 'Username' que contiene el texto 'juanito2'. A continuación es el campo 'Email' con el correo 'jgomezdr@gmail.com'. Luego hay dos campos de contraseña: 'Actual password' y 'New password', ambos con el texto de marcador de posición 'Actual password' y 'New password' respectivamente. Al final del formulario hay un botón azul rectangular con el texto 'Update' en blanco.

Figura 5.6: Actualización usuario

5.2.4. Gestores

Todas las secciones que veremos consisten en tablas paginadas que muestran la información guardada de cada registro pertenecientes a cada modelo. Además todas constan de un botón para añadir un nuevo registro, eliminar, modificar y clonar.

Se ha implementado un buscador en cada una de las vistas de cada sección.

La barra de búsqueda consiste en un campo de texto rectangular con el texto de marcador de posición 'Search Question...' en gris. A la derecha del campo de texto hay un botón gris con un icono de lupa (Q) en blanco.

Figura 5.7: Buscador

Este buscador mostrará la lista de registros cuyos nombres coincidan con aquello escrito, para facilitar el acceso a la información deseada de manera más eficiente.

5.2.5. Preguntas

Dentro de la tabla de preguntas se muestra de forma general los campos mas relevantes para distinguirlas, mostrado en la *Figura 5.8*, como el enunciado, el bloque al que pertenecen, la fecha y el usuario que la ha creado.

Como se ve en la parte superior derecha, se encuentra un botón para añadir una pregunta. Si navegamos por las filas de la tabla y clicamos en una se nos abrirá la vista para modificar la pregunta. Además en la parte derecha de cada fila hay un botón para eliminar el registro deseado.

Question	Creator	Date	Actions
Soy competente y capaz en las tareas que son importantes para mi	juanito	July 12, 2023, 7:12 p.m.	
Mis preocupaciones interfieren en el camino de lo que quiero conseguir	juanito	July 12, 2023, 6:34 p.m.	
Parece que la mayoría de la gente lleva su vida mejor que yo	juanito	July 12, 2023, 6:34 p.m.	
Mis emociones interfieren en como me gustaria que fuera mi vida	juanito	July 12, 2023, 6:33 p.m.	
Mis recuerdos dolorosos me impiden llevar una vida plena	juanito	July 12, 2023, 6:33 p.m.	
Me preocupa no ser capaz de controlar mis preocupaciones y sentimientos	juanito	July 12, 2023, 6:33 p.m.	

Figura 5.8: Listado de preguntas

Si se pulsa en el botón para añadir una pregunta se abrirá una nueva página con un formulario con los datos necesarios para registrarla. A nivel de base de datos los campos que requiere cada pregunta son los siguientes:

- *id*
- *title*: Enunciado de la pregunta.
- *creator*: Foreign key a Usuario.
- *date*: Fecha en la que se crea la pregunta.

El *id* como en todas las tablas es la clave primaria. A su vez, la pregunta tiene una clave externa a la tabla *Posibles Respuestas*. Por lo que cada pregunta tiene varias posibles respuestas y cada posible respuesta tiene asignada una pregunta. En este caso, como se ve en la *Figura 5.9*, para registrar la pregunta solo se necesita proporcionar el título y las posibles respuestas ya que el campo de creador se autorellena con el usuario que la crea y en fecha se rellena con la actual.

Dentro del panel de posibles respuestas se deben añadir las respuestas a esa pregunta, cada una separada por línea. A su vez cuando se cree la pregunta también se crearán los registros de las respuestas asociadas a la pregunta.

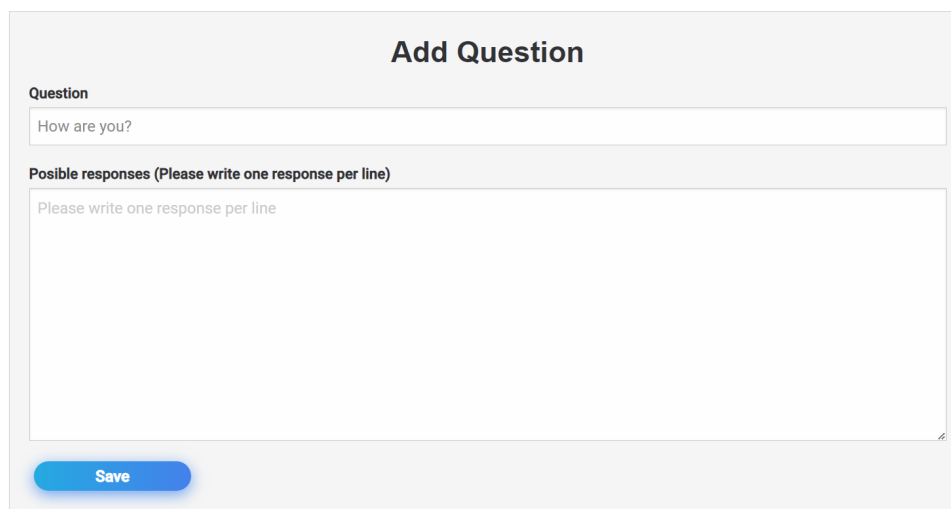


Figura 5.9: Añadir pregunta

En la *Figura 5.10* se observa el formulario de modificación de la pregunta en el que se puede cambiar el título y las respuestas. Si otro usuario modifica la pregunta también se actualiza el campo del creador y fecha.

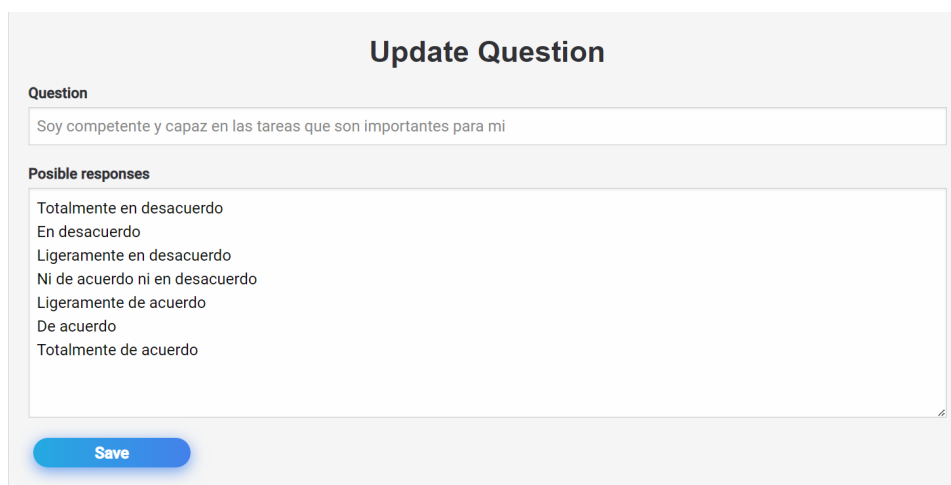


Figura 5.10: Modificar Pregunta

También se pueden eliminar y clonar preguntas, tal como se ve en la *Figura 5.7*. Concretamente, si se selecciona el botón de eliminar (que se ve como un símbolo de papelera roja a la derecha de cada fila), aparecerá una alerta para verificar si realmente se desea borrar la pregunta (*Figura 5.11*) y en caso afirmativo se realiza. También vemos un botón azul justo al lado, que es el botón de clonar. Este botón creará una nueva pregunta idéntica a

la seleccionada.

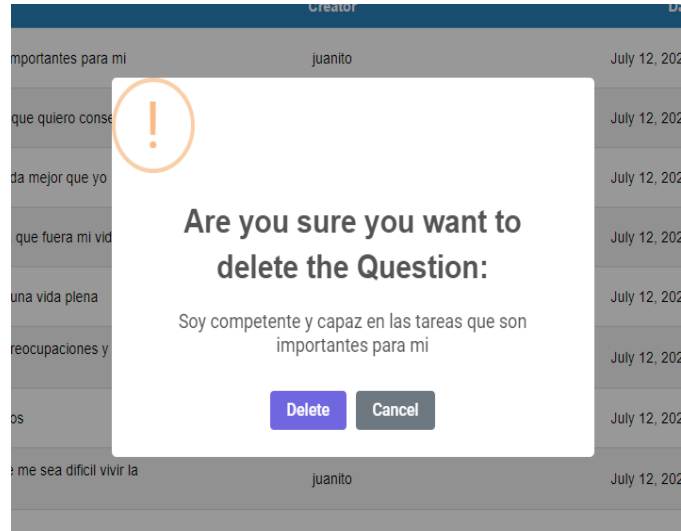


Figura 5.11: Borrar Pregunta

Bloques













Home Blocks					
Search Block...		Q	+ Block		
Block	Frecuency	Creator	Created	Active	Actions
Bloque Satisfaccion semanal	Weekly	juanito	Oct. 11, 2023, 5:09 p.m.	<input checked="" type="checkbox"/>	 
Bloque GAD-7	Once	juanito	Oct. 11, 2023, 5:09 p.m.	<input checked="" type="checkbox"/>	 
Bloque COVID-19	Weekly	juanito	Oct. 11, 2023, 5:09 p.m.	<input checked="" type="checkbox"/>	 
testingblock	Daily	juanito	Oct. 11, 2023, 5:03 p.m.	<input type="checkbox"/>	 
Bloque Satisfaccion	Daily	juanito	Oct. 10, 2023, 1:06 p.m.	<input checked="" type="checkbox"/>	 
Bloque PANAS	Once	juanito	Oct. 10, 2023, 1:06 p.m.	<input type="checkbox"/>	 

Figura 5.12: Listado de Bloques

La *Figura 5.12* muestra la tabla que contiene los registros de los bloques de preguntas en el sistema. Al igual que en la vista anterior cuenta con las operaciones de añadir, modificar y eliminar un bloque.

Los campos necesarios para registrar un bloque en la base de datos son los siguientes:

- *id*
- *block*: Nombre del bloque.
- *active*: Atributo que indica si el bloque se encuentra activo.
- *operating*: Indica si el bloque se encuentra operando en ese momento.
- *questions*: Foreign Key a Pregunta.
- *preamble*: Foreign key a Preámbulo.
- *frequency*: Frecuencia del bloque.
- *time*: Hora de la planificación.
- *day*: Día de la planificación.
- *duration*: Duración activa del bloque.
- *creator*: Foreign key a Usuario.
- *date*: Fecha en la que se crea el bloque.

El campo *questions* es una relacion de tipo *ManyToMany* con la tabla de preguntas. Un mismo bloque tendrá la posibilidad de asignar el número de preguntas que desee. Todas esas preguntas a su vez pertenecerán a ese bloque.

Add Block

Block:
Block name

Preamble:
PANAS
Flourishing scale
GAD-7
Diario
Journal, etc.

Features

Frequency:
Once

Importance:
High priority

Planning

Time (HH:MM:SS): 11:48:32 **Day:** Monday **Duration (Minutes):** 60

Questions

- ☐ Soy competente y capaz en las tareas que son importantes para mí
- ☐ Mis preocupaciones interfieren en el camino de lo que quiero conseguir
- ☐ Parece que la mayoría de la gente lleva su vida mejor que yo
- ☐ Mis emociones interfieren en como me gustaría que fuera mi vida
- ☐ Mis recuerdos dolorosos me impiden llevar una vida plena
- ☐ Me preocupa no ser capaz de controlar mis preocupaciones y sentimientos

Figura 5.13: Añadir bloque

Como se ve en la *Figura 5.13*, se puede elegir mediante un checkbox las preguntas que se desee añadir y un recuadro en el que muestra una lista de los preámbulos creados en el sistema, para elegir uno y asignarlo al bloque.

Una vez se han añadido preguntas al bloque y se le ha asignado un preámbulo, faltaría definir la frecuencia del bloque, su importancia y el atributo activo. Estos atributos están más relacionados con el bot. La frecuencia define cada cuanto tiempo se debe realizar el cuestionario asociado al bloque. Normalmente, las preguntas dentro de un mismo bloque tienen un sentido y un propósito común. Por ejemplo, hay preguntas sobre salud, horas de sueño, alimentación que necesitan de realizarse todos los días o una vez por semana. Mientras que hay otras que con solo hacerse una vez ya bastaría. De ahí nace la idea de agrupar las preguntas en bloques. Dentro de la frecuencia puede elegir entre:

- Once
- Daily
- Weekly

Otro campo sería la importancia del bloque (High importance, Normal, Low importance). Este campo se añadió para que a la hora de que el bot haga el cuestionario de un bloque, priorice los de mayor importancia ante los otros. El atributo activo va a definir la disponibilidad de ese bloque, es decir, si este se encuentra seleccionado, el bloque estará activo para los usuarios y este se ejecutará dependiendo de su planificación, pero si se encuentra inactivo nunca se realizará el cuestionario asociado (cuando se crea un bloque por defecto está desactivado, solo se activará marcando la casilla). Ahora vendría la planificación del bloque. Una de las finalidades de este trabajo es el diseño y desarrollo de un sistema de cuestionarios dinámicos, que permite a los usuarios planificar la fecha y hora de activación de los bloques, así como definir la duración de los mismos. El sistema proporciona flexibilidad y adaptabilidad para ajustar estos parámetros, permitiendo una personalización efectiva de la experiencia del usuario. Para ello se proporcionan los atributos de *time*, *day*, *duration* de forma que se pueda automatizar este proceso.

En la figura *Figura 5.14*, se puede ver como sería el ejemplo de planificar un bloque una vez a la semana todos los miércoles a las 12:00 horas y con una duración activa de 60 minutos.

The screenshot shows a web interface with two main sections: 'Features' and 'Planning'. The 'Features' section includes a 'Frequency' dropdown menu set to 'Weekly', an 'Active' checkbox that is checked, and an 'Importance' dropdown menu set to 'High priority'. The 'Planning' section includes three input fields: 'Time (HH:MM:SS)' set to '12:00:00', 'Day' set to 'Monday', and 'Duration (Minutes)' set to '60'.

Figura 5.14: Planificación Bloque

Preámbulos

Los preámbulos contienen los mensajes a mostrar por parte del bot en el chat del usuario cuando se va a tratar un tema concreto. Cuando se entra en un bloque el bot muestra un mensaje a forma de introducción que nos pone en contexto acerca del tema a tratar en el cuestionario. Esta tabla se crea como idea para que haya variedad a la hora de mostrar estos mensajes introductorios.

The screenshot shows a form titled 'Update Preamble'. It has a 'Preamble' text input field containing 'BRS'. Below this is a 'Messages' section with a large text area containing three paragraphs of text. At the bottom of the form is a blue 'Save' button.

Figura 5.15: Editar Preámbulo

En la *Figura 5.15* se puede ver un ejemplo que nos aclara con mayor detalle la función del preámbulo. El tema es AAQ-II, estas preguntas sirven para evaluar el nivel de aceptación psicológica y disposición a actuar de una persona. Cuando se realice el cuestionario asociado se mostrará uno de los tres mensajes que dispone.

La forma de crearlos es parecida a la de las preguntas que se explica

previamente. Existen 2 tablas: *Preámbulos* y *Mensajes*. La clase preámbulo contiene los nombres, y la de mensajes los enunciados de la información a mostrar, que tienen una clave foránea a preámbulo.

Clase Preámbulo:

- *name*: Nombre.

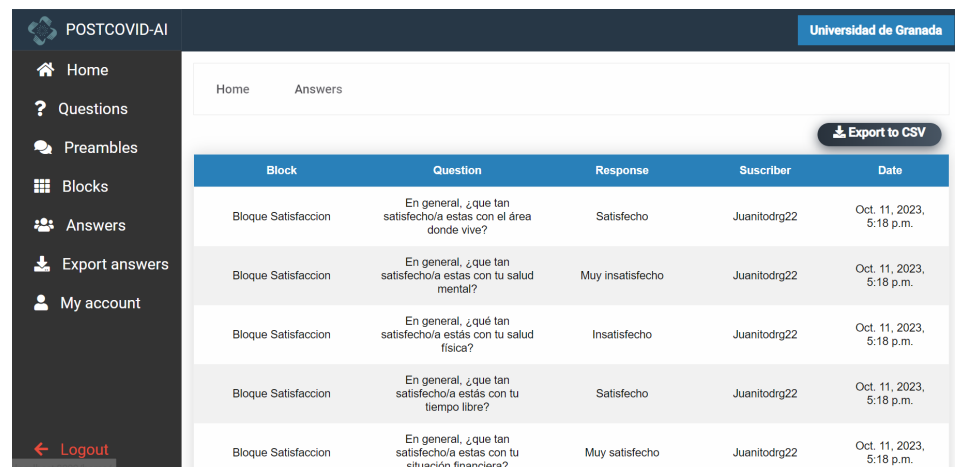
Clase Mensaje

- *text*: Enunciado del mensaje.
- *preamble*: Foreign key a Preámbulo.

Para crearlo se especifica el nombre y los mensajes, cada uno separado por una línea, que crea todos los registros y los asocia al preámbulo actual.

Respuestas

Por último nos encontramos el apartado de respuestas, mostrado en la *Figura 5.16*. Esta vista es simplemente a modo de visualización ya que el administrador no podrá realizar operaciones con ellas. Estas se crean automáticamente cuando un usuario responde a la pregunta hecha por el bot.



Block	Question	Response	Suscriber	Date
Bloque Satisfaccion	En general, ¿que tan satisfecho/a estas con el área donde vive?	Satisfecho	Juanitodrg22	Oct. 11, 2023, 5:18 p.m.
Bloque Satisfaccion	En general, ¿que tan satisfecho/a estas con tu salud mental?	Muy insatisfecho	Juanitodrg22	Oct. 11, 2023, 5:18 p.m.
Bloque Satisfaccion	En general, ¿qué tan satisfecho/a estás con tu salud física?	Insatisfecho	Juanitodrg22	Oct. 11, 2023, 5:18 p.m.
Bloque Satisfaccion	En general, ¿que tan satisfecho/a estás con tu tiempo libre?	Satisfecho	Juanitodrg22	Oct. 11, 2023, 5:18 p.m.
Bloque Satisfaccion	En general, ¿que tan satisfecho/a estas con tu situación financiera?	Muy satisfecho	Juanitodrg22	Oct. 11, 2023, 5:18 p.m.

Figura 5.16: Lista respuestas

Los campos para registrar una respuesta son los siguientes:

- *id*
- *block*: Foreign key a Bloque.
- *question*: Foreign key a Pregunta.
- *response*: Respuesta del usuario.
- *suscriber*: Foreign key a Usuario.
- *date*: Fecha de la Respuesta.

Esta vista cuenta además con un botón en la parte superior derecha para exportar las respuestas a formato CSV. Los archivos CSV almacenan los datos separados con comas, por lo que cuando se guarda el texto y los números es fácil moverlos de un programa a otro. Al fin y al cabo este es el propósito de nuestro proyecto, recaudar información y entregarla en un formato específico para que el análisis sea sencillo.

La función que lo realiza es la siguiente:

```
def export_to_csv(request):
    answers = Answer.objects.all()
    buffer = io.StringIO()
    writer = csv.writer(buffer, delimiter=',')

    writer.writerow([smart_str('Block'), smart_str('Question'),
                    smart_str('Response'), smart_str('Subscriber'), smart_str('Date')])

    for answer in answers:
        block = smart_str(answer.block.block)
        question = smart_str(answer.question.title)
        response = smart_str(answer.response)
        subscriber_username = smart_str(answer.suscriber.username)
        date = smart_str(answer.date.strftime('%Y-%m-%d'))

        writer.writerow([block, question, response,
                        subscriber_username, date])

    csv_file_name = 'answers_export.csv'
    with open(csv_file_name, 'w', encoding='latin1') as csv_file:
        csv_file.write(buffer.getvalue())

    with open(csv_file_name, 'rb') as csv_file:
        response = HttpResponse(csv_file.read(), content_type='text/csv')
        response['Content-Disposition'] = f'attachment; filename={
            csv_file_name}'

    return response
```

Python cuenta con una librería llamada csv. Esta librería implementa clases para leer y escribir datos tabulados en formato CSV.

5.3. Chatbot

El agente conversacional se encuentra formado por 3 módulos diferenciados: Módulo de bienvenida y registro de usuario, módulo conversacional y cuestionario.

5.3.1. Módulo de bienvenida y registro

La librería *python-telegram-bot* es la herramienta con la que vamos a interactuar con la API de Telegram. La librería emerge como una herramienta indispensable en este ámbito, proporcionando a los desarrolladores un marco completo para construir, desplegar y gestionar bots de Telegram sin esfuerzo.

Esta nos va a permitir diferentes posibilidades a la hora de crear nuestro bot como son la gestión de actualizaciones para controlar diferentes tipos de entradas de manera eficiente, envío de mensajes de texto y contenido multimedia, teclados y botones, comandos personalizados, respuestas interactivas, middleware y filtros. Todo lo relacionado a la descripción de estas operaciones se encuentra especificado en la documentación oficial (*“python-telegram-bot” 2015-2023*).

Este primer módulo tiene como objetivo la presentación de nuestro bot con el usuario y su registro en nuestra base de datos. Contiene una función que es la primera que se ejecuta cuando cualquier persona empieza una conversación. Se activa con el comando */start* que se envía mecánicamente y es el único comando necesario para la conversación. Cuando este evento se detecta se realiza una llamada a la función la cual muestra un mensaje de bienvenida al usuario presentando los temas a tratar. Gracias a la librería usada podemos obtener los datos que el usuario posea en su Telegram personal (como el nombre, nombre de usuario, apellidos, email, id).

```
suscriber, created = Suscriber.objects.get_or_create(
    chatid = chat_id,
    name=name,
    surname=surname,
    username=username,
)
```

La función de Django *get_or_create()* nos permite filtrar una serie de datos introducidos dentro de una tabla en la base de datos. En caso de que ningún registro coincida exactamente con todos los datos, crea una nueva instancia con los parámetros pasados. De esta forma se registran los usuarios. Para asegurarnos de la concurrencia de nuestro programa, se ha creado un atributo en usuario llamado *chatid* que almacena el id propio del chat.

Esto se hace con la intención de no declarar variables globales dentro del código ya que si hubiera varias personas hablando al mismo tiempo habría problemas de concurrencia. Por eso cada consulta y cada mensaje es específico para cada uno intentando operar siempre desde la base de datos, lo que además influye positivamente en la eficiencia ya que las consultas y operaciones son más rápidas.

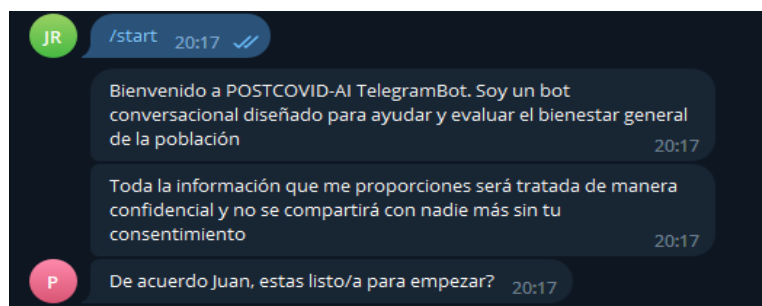


Figura 5.17: Mensaje de bienvenida

La figura *Figura 5.17* muestra como es el mensaje de presentación del bot. Nos pone en contexto y nos habla sobre su propósito. Finalmente nos hace una pregunta y dependiendo de la respuesta del usuario nos redirigirá a un módulo u otro.

5.3.2. Módulo conversacional

Este módulo es una etapa intermedia, que funciona como una especie de sala de espera a que el usuario cuente con cuestionarios a responder. Dentro de él, se puede tener una conversación de lo más común y simple con el bot, con un apartado que muestra preguntas frecuentes en caso de que el usuario tenga cualquier tipo de duda.

El usuario puede entrar dentro de este módulo de diferentes maneras. Una de ellas es respondiendo de forma negativa en la pregunta anterior. Dentro de este módulo hay unas palabras, frases o expresiones previamente definidas por las cuales el bot responde en función de lo que escribas. Cabe destacar que no se utiliza modelos de lenguaje como GPT (Generative Pre-Trained Transformer) para generar respuestas mas elaboradas. GPT es un modelo que utiliza algoritmos avanzados de procesamiento de lenguaje natural para generar respuestas a preguntas y comentarios de los usuarios en tiempo real (*Floridi 2020*). En nuestro caso esa no es la finalidad de nuestro bot. Como ya se ha indicado el propósito principal es la recogida de datos, con la funcionalidad añadida de que el usuario pueda tener una conversación escueta con el bot para que esta recogida se realice de la forma más humanamente posible, pero esta quedando siempre en segundo plano.

Este módulo se ejecuta de forma recursiva cada vez que el usuario escribe un mensaje en el chat. Funciona de la siguiente manera:

1. Recogida del mensaje: Cuando el usuario manda cualquier tipo de mensaje el bot lo recibe como texto de entrada y lo almacena.

2. Procesamiento: Este proceso implica limpiar y normalizar el texto introducido, convirtiéndolo a minúsculas y reduciendo las palabras a su forma base.

3. Análisis: Conlleva la identificación del contexto del mensaje con el propósito de extraer información relevante o identificar intenciones.

Esta identificación se hace dentro del archivo *replyMessages.py* situado dentro de la carpeta /chatbot. En el podemos encontrar diferentes listas como estas.

```
#HELLO AND GOOD BYE
greetings = ['hola', 'hello', 'buenos dias', 'buenas tardes', 'buenas',
             'saludos', 'hi', 'good']
farewell = ['adios', 'bye', 'good bye', 'hasta luego', 'nos vemos', 'hasta pronto', 'buenas noches', 'que tengas un buen dia', 'hasta la proxima', 'que vaya bien']
```

Este es un ejemplo de lista para establecer el contexto de saludo y despedida. Estas listas definen una serie de palabras y expresiones concretas que cualquier persona hispanohablante utilizaría si quisiera referirse o hablar sobre un tema. Si cualquier palabra dentro de la respuesta del usuario se encuentra contenida en esta lista el bot identifica la intención del mensaje y le asociará un contexto.

4. Generación de respuesta: Una vez encontrado el contexto del mensaje debemos buscar una respuesta digna que se adecue al tema tratado. Para ello dentro de este mismo archivo mencionado encontramos varios diccionarios. Estos contienen las respuestas que el bot debe proporcionar para cada contexto. Por ejemplo:

```
'greetings':{
    '0': 'Hola! Puedo ayudarte en algo?',
    '1': 'Muy buenas! Necesitas ayuda?',
    '2': 'Hola! Como puedo asistirte hoy?'
},
'farewell':{
    '0': 'Adios !. Que tengas un buen dia!',
    '1': 'Hasta luego!. Espero verte de nuevo pronto!',
    '2': 'Adios. !No dudes en volver!',
}
```

Cuando declaramos el contexto de saludo como este ejemplo, vemos que hay una serie de mensajes. He intentado que haya un mínimo de unos tres mensajes por cada contexto para que la respuesta proporcionada por el bot no sea siempre igual e intentar que esta varíe. A la hora de mostrarlo en el chat se hace con la función que se muestra a continuación, la cual envía el mensaje al usuario concreto que lo ha escrito y selecciona una respuesta aleatoria dentro de las que el diccionario disponga.

```
context.bot.send_message(chat_id=user.id, text=messages['greetings'][  
    str(random_var)])
```

Por último debemos tratar que pasaría si el bot no reconoce el mensaje del usuario, ya sea por hablar de un tema que el bot no tiene respuesta o por escribir mal una palabra o expresión. En este caso, en la etapa anterior de análisis el bot no podría clasificar la respuesta. Cuando esto sucede se muestra lo que vemos en la *Figura 5.18*.

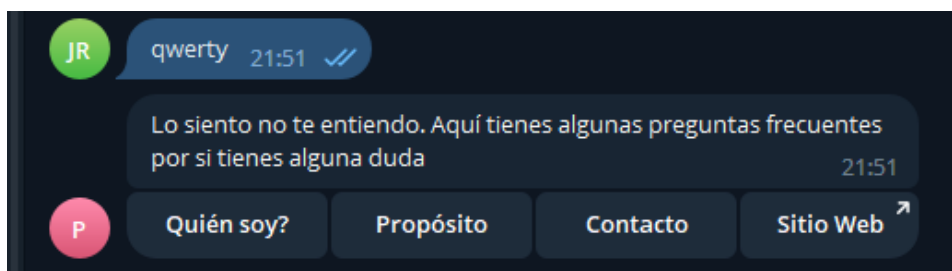


Figura 5.18: Respuesta del bot a un mensaje no entendido

El mensaje contiene botones con algunas preguntas frecuentes que puedan ser de interés al usuario (como información sobre el bot, su propósito, sitio web y contacto del proyecto a que pertenece). En la *Figura 5.19* se pueden ver los mensajes que se mostrarían si se pulsa cada uno de los botones:

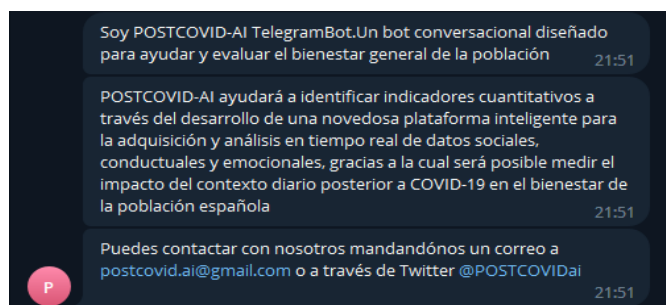


Figura 5.19: Respuestas del bot a las preguntas frecuentes

Calendario de cuestionarios

Desde el momento que se ejecuta el programa se crea un proceso recursivo dentro del mismo para cada uno de los bloques registrados en el sistema. Este proceso se ejecuta de forma ininterrumpida dependiendo de la planificación que le hayamos dado a cada bloque. Es decir, si un bloque concreto se encuentra planificado para que se repita de forma diaria a las 14:00 horas, todos los días de la semana a esa hora en concreto se ejecutará el proceso. Este avisará a todos los usuarios registrados de que cuentan con un cuestionario activo y que pueden pasar a responderlo. Esto se hace mediante la ayuda de la librería *apscheduler*, que permite planificar procesos y ejecutarlos en tiempos determinados. Los denomina como 'Job'.

```
self.scheduler.add_job(
    self.advise,
    replace_existing=True,
    trigger=CronTrigger(day_of_week=self.block.days, hour=
        self.block.time.hour, minute=self.block.time.
        minute, second=self.block.time.second, timezone=
        pytz.timezone('Europe/Madrid')),
    id=str(self.block.id),
    name=self.block.block,
)
```

Esta es la manera de crearlos, que como se ve en el código depende totalmente de la planificación del bloque. Lo que hace es llamar a otro método que muestra el mensaje mostrado en la *Figura 5.20* al usuario y activa el cuestionario correspondiente.

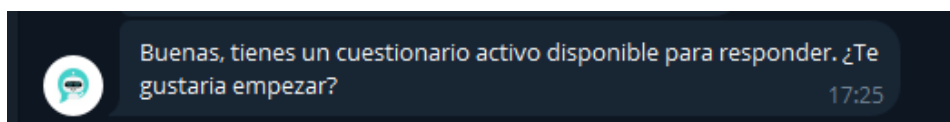


Figura 5.20: Mensaje cuestionario pendiente

Los cuestionarios cuentan con una duración definida en cada bloque. Estos solo se encontrarán activos durante el tiempo que hayamos proporcionado (puede ser 30 minutos, 1 hora, 2 horas...). Cuando un cuestionario se activa en su momento planificado, lanza otro 'Job' que se ejecutará una vez pasado el tiempo de duración del bloque con la función de desactivarlo. De esta manera, si un cuestionario está planificado, por ejemplo, el lunes a las 12:00 horas y tiene una duración de una hora, el bot avisará de que se cuenta con un cuestionario activo y todos los usuarios tendrán exactamente una hora para responderlo. Pasada esta hora el cuestionario se cierra y hasta la siguiente planificación no se podrá volver a responder.

También debemos de tener en cuenta cuando se modifique, se cree o se borre un bloque. Para ello el programa cuenta con un proceso interno que

se ejecuta cada 30 segundos que ejecuta el siguiente método:

```
def actualize(context):
    global aux_blocks
    current_blocks = list(QuestionBlock.objects.all().values('id'))
    id_list1 = [item['id'] for item in current_blocks]
    id_list2 = [item['id'] for item in aux_blocks]
    different_blocks = list(set(id_list1) ^ set(id_list2))
    if different_blocks is not None:
        for item in different_blocks:
            try:
                block = QuestionBlock.objects.get(id=item)
                ScheduleJob(context, block, scheduler)

            except:
                ScheduleJob.remove_schedule(context, str(item))

    aux_blocks = current_blocks
```

La función de este método es buscar modificaciones en la base de datos y actualizar los calendarios. En el caso de crear un nuevo bloque creará un nuevo calendario, al borrar un bloque existente eliminará su calendario asociado con el fin de que no se ejecute más, o al modificarlo actualizará el actual.

5.3.3. Módulo cuestionario

Pasamos ahora al último módulo de nuestro bot. Este módulo sigue el ejemplo de un cuestionario de preguntas clásico en el que el bot va haciendo una serie de preguntas y se debe responder con una de las opciones posibles conforme al criterio de cada uno.

Para entrar, el bot ya sea al principio o cuando se cuente con preguntas pendientes, se hará la pregunta de si se quiere comenzar el cuestionario. Solo se entrará en él cuando se responda de forma afirmativa a esta pregunta. Una vez estemos dentro, el bot explicará de forma escueta como funciona y elegirá la primera pregunta. He aquí cuando entra en juego una de las funciones principales *chooseQuestion(user)*.

```
def choose_question(user):
    global first
    first = False
    bloques = QuestionBlock.objects.filter(operating=True, active=True)
    bloques = bloques.order_by('-importance')
    block_counter = 0
    question_result = []
    for block in bloques:
        questions = Question.objects.filter(blocks=block).order_by('create')
        first_value = questions.first()
        for question in questions:
            if is_answered(user, question, block) == False:
```

```

        question_result.append(question)
        question_result.append(block)
        if question == first_value:
            first = True
            break
        block_counter += 1
    return question_result

```

Esta función recorre los bloques de preguntas que se encuentran activos en el sistema ordenándolos por importancia. Comprueba las preguntas una a una dentro de cada bloque filtrando las respuestas para cada usuario y comprobando si ha sido respondida. En este punto tenemos dos posibilidades. Que el usuario sea nuevo y no ha realizado ningún cuestionario, por lo que la primera pregunta que compruebe será la devuelta ya que no hay ningún registro de respuestas por su parte. Y en caso contrario, que el usuario ya haya realizado varios cuestionarios. En este supuesto, se resolvería de la misma forma que el Job entiende cuando un usuario cuenta con preguntas pendientes, es decir, comparando las fechas de las respuestas con la fecha actual, restándola y si es mayor a la frecuencia del bloque, significa que esa pregunta se encuentra pendiente, si no pasaría a la siguiente. Además el método identifica la primera pregunta perteneciente a cada bloque, para que cuando esta se ejecute muestre cualquiera de los mensajes asociados al preámbulo de ese bloque de preguntas.

Para el cuestionario, se ejecuta recursivamente la función *handleanswer(update, context)* que es la encargada de realizar las preguntas, separándolas por bloques, mostrar los mensajes asociados a los preámbulos y recoger las respuestas como lo vemos en la *Figura 5.21*. Solo termina cuando el usuario no cuente con preguntas pendientes. En cada iteración hace una llamada a la función anterior *chooseQuestion(user)*, eligiendo de forma dinámica la siguiente pregunta a realizar.

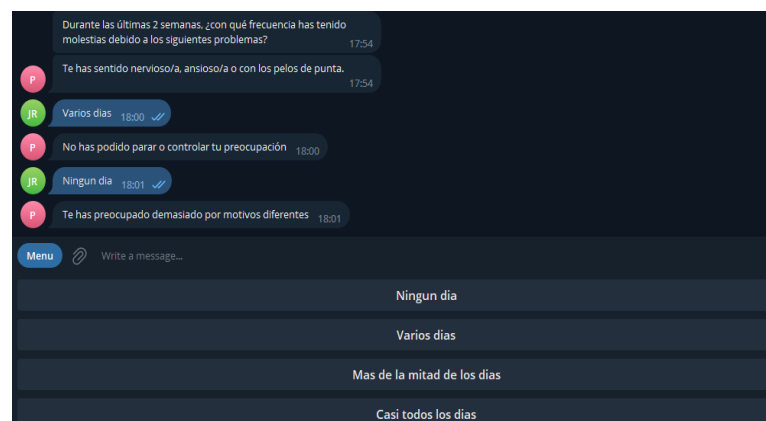


Figura 5.21: Cuestionario de preguntas

Ya sabemos que cada pregunta cuenta con posibles respuestas de forma independiente. Cuando el bot hace la pregunta actualiza el teclado de Telegram y da la opción solo de contestar mediante botones que contienen las respuestas a la pregunta. Esto se hace de la siguiente manera:

```
#Function to get a custom keyboard in Telegram
def custom_keyboard(values):
    schema = [[str(value)] for value in values]
    return ReplyKeyboardMarkup(schema, one_time_keyboard=True,
                               resize_keyboard=True)
```

El parámetro pasado contiene las respuestas a la pregunta y haciendo uso de la función *ReplyKeyboardMarkup()* proporcionada por la librería de Telegram creamos un teclado personalizado con los valores que deseemos. Solo se puede responder a la pregunta si el mensaje esta dentro de estas respuestas. Esto se ha hecho para asegurarse que los datos obtenidos son correctos y reales, ya que si se responde cualquier cosa que no tuviera nada que ver se guardaría como respuesta a la pregunta. Por lo que en cada iteración de la función se comprueba si el mensaje proporcionado por el usuario es válido. En caso negativo, como se muestra en la *Figura 5.22* el bot no pasaría a la siguiente pregunta y volverá a realizarla de forma indefinida hasta que se responda de forma correcta.

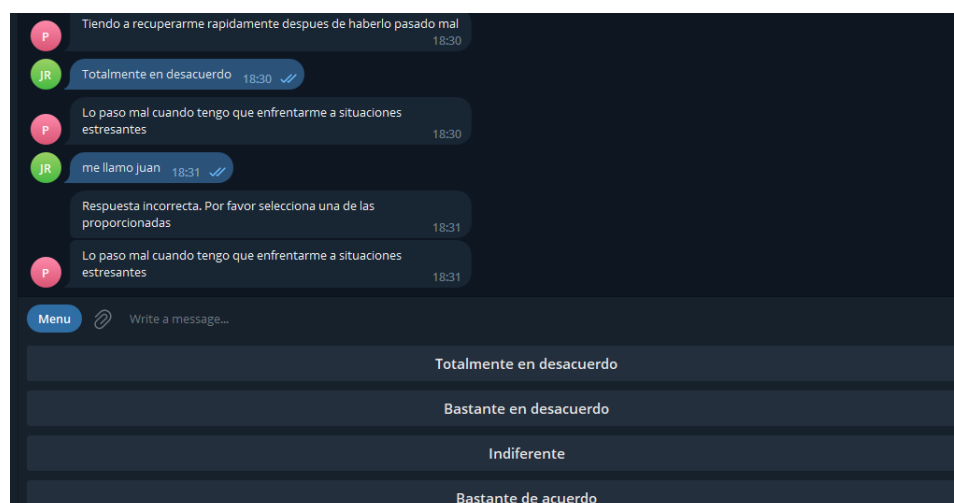


Figura 5.22: Mensaje del bot a una respuesta incorrecta

Solo cuando esta sea correcta guardará la respuesta en la base de datos. La función finaliza cuando el usuario haya terminado de responder todas las preguntas y lo redirigirá de forma automática al módulo conversacional.

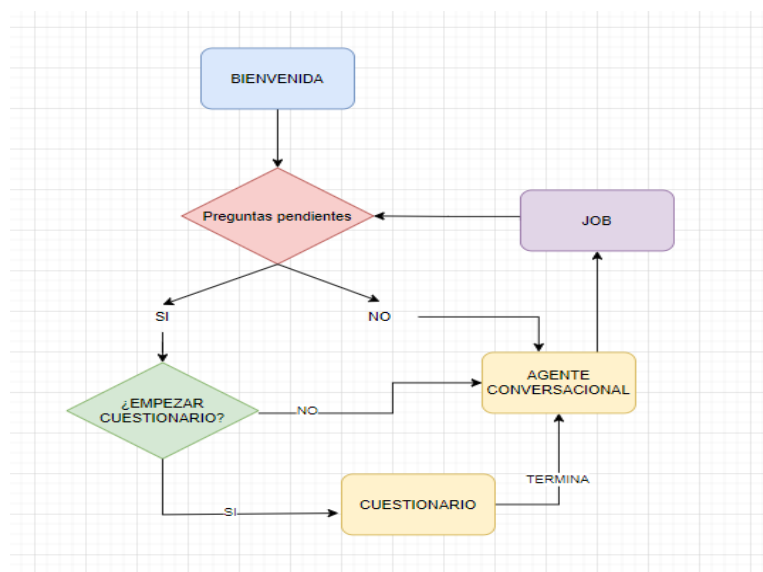


Figura 5.23: Diagrama de flujo del bot

En la *Figura 5.23* se ve como sería el diagrama de flujo que sigue el bot y la lógica que utiliza.

Recapitulando todo el proceso, cuando se inicie una conversación con el bot, este te registrará en la base de datos como un nuevo usuario. El sistema comprobará si hay algún cuestionario planificado para ese momento, en caso afirmativo el bot avisará al usuario y preguntará si desea realizar el cuestionario que se encuentre activo. Si se accede a responder, en el momento que termine se le llevará al módulo conversacional en el que puede charlar con el bot. Si no accede a responder, no pasaría nada, el cuestionario terminaría según lo planificado y no habría respuestas de el usuario para ese cuestionario. Una vez dentro de este módulo conversacional, el bot volverá a avisar a el usuario cuando haya de nuevo un cuestionario activo y continuaría con la misma dinámica.

5.4. Tecnologías

Para el desarrollo usaremos herramientas de Front-End, que nos ayudarán a dar cuerpo a nuestro sitio web y herramientas de Back-End para el control y lógica tanto de la web como del bot. Las tecnologías utilizadas para la implementación del proyecto son las siguientes:

- **Python** como lenguaje principal del bot y de la web.
- **Django** es un framework de aplicaciones web gratuito y de código

abierto (open source) escrito en Python para el desarrollo de nuestra web.

- **Telegram** como plataforma para alojar el chatbot
- **Python-telegram-bot** es una librería de Python para la creación de nuestro bot que nos permite conectarnos a la API de Telegram.
- **PostgreSQL** es un sistema de gestión de base de datos relacional orientado a objetos.
- **Psycopg2** para adaptar la base de datos a Python.
- **HTML5** lenguaje para la elaboración de páginas web.
- **CSS3** como lenguaje de diseño gráfico para crear la presentación de nuestra página.
- **Javascript** para dar dinamismo a nuestra web y ayudar en la experiencia del usuario.
- **Foundation** como framework de ayuda a la hora de crear la interfaz de usuario.
- **Docker** para automatizar el despliegue de la aplicación dentro de un contenedor software.
- **Github** como tecnología que nos ayuda a realizar copias del proyecto y controlar versiones.

Capítulo 6

Despliegue e instalación

6.1. Despliegue

El despliegue del proyecto se ha realizado en un repositorio público de mi cuenta de GitHub, en el que encontramos todo el código disponible.

Para la creación del bot y que se encuentre activo para todo aquel que quiera conversar se ha utilizado la API de Telegram. La aplicación de mensajería pone a disposición de cualquier usuario la creación de bots en su plataforma de manera relativamente sencilla con la ayuda de su robot *BotFather*. Se trata de un bot disponible en cualquier versión de Telegram, escritorio o app móvil, y que se encarga de controlar otros bots. Gracias a la API de que dispone Telegram para bots vamos a poder acceder a una gran cantidad de herramientas que nos ayuda a la hora de desarrollar la lógica de nuestro bot. Para ello debemos buscar a *@BotFather* en el buscador de Telegram y empezar una conversación con él. Este es el bot oficial que te permitirá crear y administrar tus propios bots.

Como observamos en la *Figura 6.1*, escribiendo el comando `/newbot` iniciamos el proceso de creación. Nos solicitará que elijamos un nombre, que será el mostrado en las conversaciones, y un username que deberá de terminar en `_bot` o `Bot`. Una vez creado, nos proporcionará un token de acceso único para nuestro bot. Este es el token con el que vamos a interactuar con la API de Telegram.

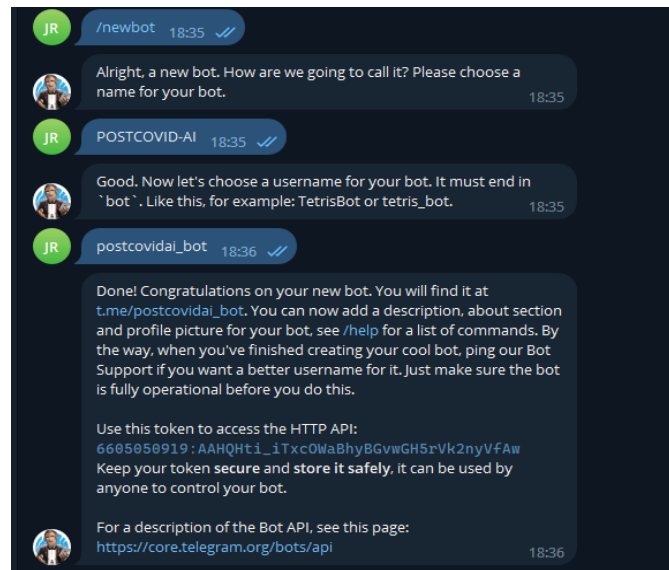


Figura 6.1: Creación del bot en Telegram

Hecho esto, tal como se muestra en la *Figura 6.2*, podemos personalizar nuestro bot como deseemos, cambiando la descripción, el estado, la foto de perfil, comandos personalizados...



Figura 6.2: Customización del bot

6.2. Instalación

6.2.1. Configuración del entorno

Antes de explicar como montar el proyecto, son necesarias tener instaladas previamente algunas tecnologías. La primera de ella es Docker. Dependiendo del sistema operativo que estemos utilizando, debemos descargarlo de una forma u otra. En el caso de Windows, desde la documentación oficial (“Docker” s.f.), nos permite descargar un archivo ejecutable que nos hará la instalación de manera automática. En caso de usar Linux, podemos descargarlo usando los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/
  docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null$
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

El lenguaje de programación utilizado es python, por lo que debemos de tenerlo descargado y declarado como variable de entorno en nuestro sistema. En su página oficial, explica como hacerlo (“Python” s.f.)

El gestor de base de datos utilizado es PostgreSQL. También es posible descargarlo en Windows utilizando el instalador que proveen (“PostgreSQL” s.f.) o en el caso de Linux ejecutando los siguientes comandos:

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt
$(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc
| sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get -y install postgresql
```

6.2.2. Montaje de desarrollo

Una vez tengamos todo lo necesario, ya podemos pasar al montaje del proyecto. Para ello, descargamos el código del repositorio público de github que contiene el proyecto. Podemos hacerlo a mano o mediante comando.

```
$ git clone https://github.com/juantiog22/Thesis.git
```

Se ha usado Docker ya que permite entregar código con mayor rapidez, estandarizar las operaciones de las aplicaciones, transferir el código con facilidad y ahorrar dinero al mejorar el uso de recursos. Con Docker, obtiene un solo objeto que se puede ejecutar de manera fiable en cualquier lugar. La sintaxis sencilla y simple de Docker le aporta un control absoluto, por lo que todo lo necesario a la implantación del proyecto ya se encuentra proporcionado. Dentro de la carpeta de */Thesis* ejecutamos este comando:

```
$ docker-compose up -d --build
```

Esto creará las imágenes de docker correspondientes a la interfaz web, el bot y la base de datos. Además instalará todos los paquetes necesarios situado dentro del archivo *'requirements.txt'*. Para la creación de la base de datos, se ha exportado una por defecto que se encuentra dentro del archivo *'dumpfile.sql'*. Esta base de datos cuenta con preguntas, bloques y preámbulos previamente creados con ejemplos reales de como serían algunos cuestionarios guiándome según las reglas e información relevante que me proveen desde (“POSTCOVID-AI” 2023). En el caso de querer establecer esta base de datos tras haber hecho el paso anterior debemos restaurar la nuestra con la información que contiene este archivo. Para ello:

```
$ docker-compose exec db bash
$ pg_restore -U postgres -d postgres /var/lib/postgres/data
/dumpfile.sql
```

Esto vuelca toda la información en nuestra base de datos y migrará todos los datos de forma automática en la base de datos que provee PostgreSQL por defecto. En el caso de no querer usar esa base de datos y querer crear otra nueva pero con todos los cambios debemos de crearla previamente. Desde la terminal de psql podemos hacerlo con el siguiente comando:

```
$ createdb -U "nombre_propietario" "nombre_base_datos"
```

Si hacemos esto, debemos hacer un par de cambios antes de la instalación. En el archivo *'docker-compose.yml'*, podemos cambiar las variables relacionadas con el entorno de la base de datos antes de construir la imagen.

- `POSTGRES_USER=postgres`
- `POSTGRES_DB=postgres`
- `POSTGRES_PASSWORD=postgres`

También dentro de la ruta `'/telegrambot/telegrambot/settings/local.py'` es donde se define la base de datos con la que vamos a trabajar, podemos cambiarlo modificando los parámetros como deseemos.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql',  
        'NAME': 'postgres',  
        'USER': 'postgres',  
        'PASSWORD': 'postgres',  
        'HOST': 'db',  
        'PORT': 5432,  
    }  
}
```

Tendríamos que cambiar los datos relacionados al usuario, contraseña y nombre. Si queremos crear una base de datos desde cero y completamente nueva, simplemente tras construir la imágenes de docker ejecutamos lo siguiente:

```
$ docker-compose run web python manage.py makemigrations  
$ docker-compose run web python manage.py migrate
```

Esto generará una base de datos completamente nueva con todas las tablas necesarias, lista para funcionar, aunque estará vacía de registros. Una vez que se haya completado esta fase, ejecutamos el siguiente comando, y nuestro proyecto estará en funcionamiento y preparado para su uso.

```
$ docker-compose up
```

Es importante asegurarse de que no haya otro proceso corriendo en el mismo puerto a la hora de instalarlo para que no nos encontremos con ningún error. Si desde el navegador accedemos a la url: `http://localhost:8000/` podemos acceder a la aplicación web y además el bot se encontrará escuchando y esperando actualizaciones desde la aplicación de Telegram.

Capítulo 7

Conclusiones

Para concluir nuestro proyecto, vamos a analizar si los objetivos propuestos han sido llevados a cabo. Finalmente expondremos algunos aspectos a mejorar como trabajo futuro y una opinión personal evaluando mi experiencia tras el desarrollo del proyecto.

- Se ha conseguido implementar un chatbot que recoja datos relacionados con el bienestar de las personas.
- Se ha conseguido diseñar una arquitectura y una base de datos que cumple con lo que necesitábamos. Tanto la interfaz web, como la base de datos y el bot en telegram se encuentran mutuamente conectados. La base de datos contiene la información de usuarios, preguntas, bloques, mensajes, respuestas y todo almacenado de forma legible y estructurada.
- La interfaz web responde adecuadamente a lo que estábamos buscando. Una aplicación sencilla e intuitiva para poder interactuar con el bot y las respuestas recogidas.
- El bot es capaz de tener una conversación con el usuario de forma escueta para mejorar la interacción, además de realizar cuestionarios según se planificación.
- El almacenamiento de las respuestas en la base de datos se realiza de forma automática y correcta, con la funcionalidad añadida de poder exportarlas para su posterior análisis.

El desarrollo del proyecto se ha alargado algo más de lo planteado, debido a ciertos cambios en el diseño que conforme se ha ido implementado han ido surgiendo, también sumado al desconocimiento de algunas tecnologías usadas. Pero tras tener todo esto en cuenta, el resultado ha sido satisfactorio ya que todo los requisitos planteados han sido cubiertos.

7.1. Aspectos a mejorar

La funcionalidad de nuestro proyecto puede ser mejorada de diversas formas. En el caso del bot sobre todo, se puede mejorar el tema de la interacción con el usuario usando sistemas de procesamiento de lenguajes más avanzados. En el caso de la aplicación web un punto potencial de mejora sería la creación de una API que nos permita la interacción con la base de datos de una forma más eficiente. Se podría realizar mejoras significativas en la experiencia de usuario (UI/UX) de la interfaz y en lo que respecta a la seguridad y tratamiento de datos. Este último apartado es clave para mejorar el proyecto, garantizar la seguridad de los datos y cumplimiento con las regulaciones pertinentes.

Todos estos aspectos a mejorar se podrían abordar en futuras etapas del proyecto o en trabajos posteriores.

7.2. Valoración personal

Este trabajo ha sido mi mayor reto afrontado en el transcurso de mi etapa como estudiante en la Universidad de Granada al comprometerme para su desarrollo en un entorno de trabajo real. Además, junto con lenguajes, técnicas y campos en los que no poseía conocimiento especializado, mi interés en participar en diversas competencias ha representado una fase de aprendizaje en solitario sumamente enriquecedora. En esta línea, he intentado utilizar siempre cosas útiles y empleadas por programadores y empresas en la actualidad, por lo que todo lo adquirido me ayudará en el desarrollo de futuros proyectos en mi trayectoria profesional.

Tras haber realizado este proyecto me he dado cuenta de cómo las tecnologías pueden enriquecer y facilitar nuestras vidas cotidianas. Observamos que la tecnología en sí misma no es un fin, sino un medio para lograr el florecimiento humano. A medida que nos adentramos cada vez más en un mundo más tecnológico, es fundamental reconocer que estas herramientas son manifestaciones de la capacidad humana para imaginar, crear y transformar su realidad.

"La tecnología no es nada. Lo importante es que tengas fe en la gente, que sean básicamente buenas e inteligentes, y si les das herramientas, harán cosas maravillosas con ellas."

– Steve Jobs.

Apéndice A

Guía de Uso

En este apéndice, aprenderemos a usar de forma general el proyecto, explicando paso a paso cada funcionalidad, poniendo a prueba algunos casos de uso y examinando la función de cada parte implicada.

A.1. Ingreso

Tras haber explicado todo el proceso de instalación y tecnologías necesarias para montar el proyecto, vamos a explicar como acceder a él.

Si hemos seguido los pasos previos mostrados en el apartado de instalación, el proyecto ya debería de estar montado, las imágenes en docker ya construidas y la base de datos ya creada y migrada. Para ejecutarlo simplemente desde la carpeta inicial del proyecto abrimos una terminal y escribimos el siguiente comando:

```
$ docker-compose up
```

Este comando inicializa los contenedores de docker y ejecuta el proyecto en el entorno local de nuestro ordenador. Para acceder a la aplicación web, debemos de usar un navegador como puede ser Google Chrome y escribimos en el buscador esta dirección : `http://localhost:8000/` que abrirá el proyecto en una ventana de nuestro navegador.

Una vez estemos dentro nos aparecerá la vista de inicio de sesión. Como no tenemos usuarios creados, vamos al apartado de registro y creamos un nuevo superusuario con los campos requeridos (*Figura A.1*) que constará con los permisos para realizar cambios y administrar los datos.

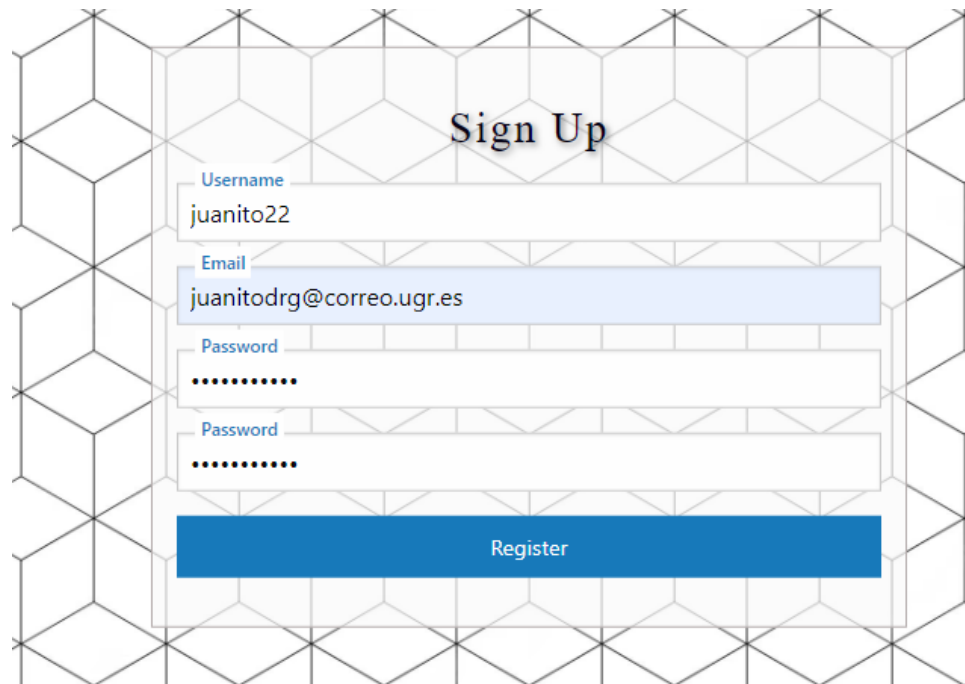
A registration form titled "Sign Up" is displayed within a decorative frame of overlapping hexagons. The form contains four input fields: "Username" with the value "juanito22", "Email" with the value "juanitodrg@correo.ugr.es", and two "Password" fields, both masked with dots. A blue "Register" button is positioned at the bottom of the form.

Figura A.1: Vista de registro

Una vez registrados, ya podemos hacer login y acceder al panel de administración.

A.2. Gestión

A.2.1. Administración de preguntas

Lo primero de todo es definir las preguntas que queremos que el bot realice a los usuarios. Para ello dentro del apartado de preguntas, en la esquina superior derecha se encuentra la funcionalidad de añadir pregunta. Clicamos y nos abrirá la vista para añadirla. Son necesarios los campos del enunciado y posibles respuestas a la pregunta.

En la *Figura A.2* se ve un ejemplo de una pregunta de prueba que se ha creado. Como vemos se le ha asignado cuatro posibles respuestas. Se ha añadido también dos preguntas más, cada una con sus respuestas específicas para que luego se puedan identificar.

Figura A.2: Vista de creación de pregunta

En la *Figura A.3* vemos las tres preguntas que se acaban de crear con algunos datos relevantes. Si desde aquí se clic en la fila de la pregunta se abrirá una vista para modificar la pregunta o si le damos al icono de la papelera, aparecerá un modal para eliminarla.

Question	Creator	Date	Actions
Pregunta de prueba 3	juanito	Oct. 17, 2023, 10:48 a.m.	 
Pregunta de prueba 2	juanito	Oct. 17, 2023, 10:48 a.m.	 
Pregunta de prueba 1	juanito2	Oct. 17, 2023, 10:48 a.m.	 

Figura A.3: Vista de listado de preguntas

A.2.2. Administración de Preámbulos

Aquí se definirán los preámbulos, es decir, los mensajes que mostrará el bot cuando haga el cuestionario asociado a un bloque. Para añadirlo, desde la página principal nos dirigimos al apartado de preámbulos que nos abrirá la vista de los preámbulos registrados y al igual que antes creamos uno.

Se crea de igual forma que la pregunta como se ve en la *Figura A.4*, con la diferencia que el primer campo es el nombre del preámbulo y el segundo los mensajes asociados a él. Cada mensaje deberá ocupar una línea diferente en el formulario.

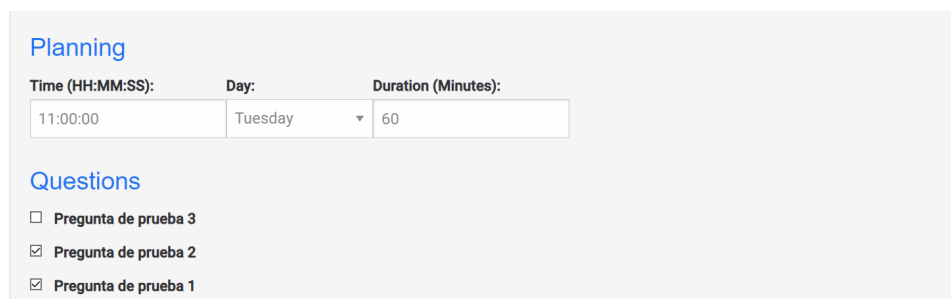
The screenshot shows the 'Add Preamble' form in the POSTCOVID-AI application. On the left is a dark sidebar with the application logo and navigation links: Home, Questions, Preambles, Blocks, Answers, Export answers, and My account. At the bottom of the sidebar is a 'Logout' button. The main content area has a top navigation bar with 'Home', 'Preambles', and 'Add Preambles'. The 'Add Preamble' form includes a 'Preamble' text field with the placeholder 'Preámbulo de prueba'. Below it is a 'Messages (Please write one message per line)' text area containing three lines of example text: 'Hola este es un mensaje de prueba', 'Responde las preguntas de este cuestionario de prueba', and 'Este es un ejemplo de un mensaje de prueba'. A blue 'Save' button is located at the bottom right of the form.

Figura A.4: Vista creación de preámbulo

A.2.3. Administración de Bloques

Pasamos a la creación de bloques. Al igual que en los ejemplos anteriores en el apartado de bloques, podemos añadir nuevos. Elegimos el nombre, las preguntas que queramos que contenga, el preámbulo, elegimos la frecuencia y la planificación *Figura A.5*.

The screenshot shows the 'Add Block' form in the POSTCOVID-AI application. The sidebar and top navigation bar are identical to the previous figure. The 'Add Block' form has a 'Block:' text field with the placeholder 'Bloque de prueba 1' and a blue 'Save' button. Below this is a 'Context:' section with a list of items: 'Satisfacción semanal', 'BRS', 'Preámbulo de prueba 1', and 'Preámbulo de prueba 2'. The 'Preámbulo de prueba 1' item is currently selected. Under the 'Context:' section is a 'Features' section. It contains two dropdown menus: 'Frequency:' with 'Once' selected, and 'Importance:' with 'High priority' selected.



Planning

Time (HH:MM:SS): 11:00:00 Day: Tuesday Duration (Minutes): 60

Questions

- ☐ Pregunta de prueba 3
- ☒ Pregunta de prueba 2
- ☒ Pregunta de prueba 1

Figura A.5: Vista creación de bloques

He creado dos bloques para este ejemplo, como se ve en la *Figura A.6*. Uno con las dos primeras preguntas y el primer preámbulo, y otro con la última pregunta y otro preámbulo.

Block	Frequency	Creator	Created	Active	Actions
Bloque de prueba 1	Once	juanito	Oct. 17, 2023, 11:09 a.m.	<input checked="" type="checkbox"/>	 
Bloque de prueba 2	Once	juanito	Oct. 17, 2023, 10:58 a.m.	<input checked="" type="checkbox"/>	 

Figura A.6: Vista listado de bloques

A.2.4. Respondiendo al cuestionario

Una vez hecho esto, ya podemos pasar a hablar con el bot. Nos dirigimos al chat de Telegram y empezamos la conversación *Figura A.7*.

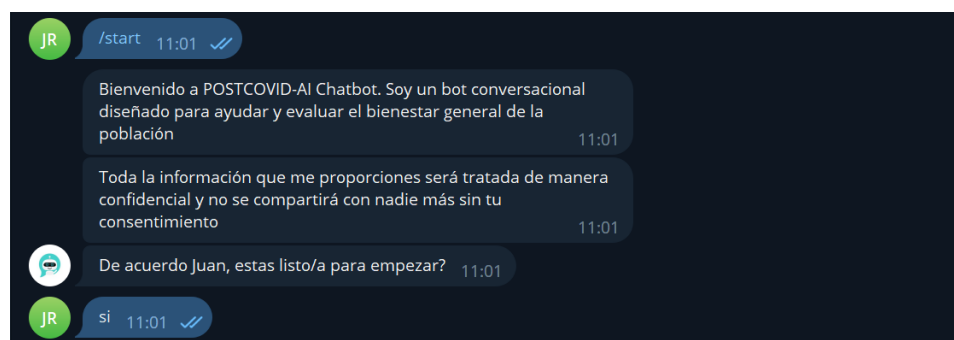


Figura A.7: Introducción del bot

El bot detecta que tenemos preguntas pendientes debido a que hemos creado un nuevo bloque y lo hemos planificado para el momento actual, por lo que pasaría a realizar el cuestionario *Figura A.8*. Empieza realizando las

preguntas asociadas al último bloque creado. Primeramente muestra uno de los mensajes del preámbulo y realiza la primera pregunta, respondemos y realiza la segunda con sus respuestas correspondientes. Y así con cada uno de los bloques que se encuentren activos.



Figura A.8: Cuestionario

Una vez se hayan respondido todas las preguntas, el bot nos informará de que hemos respondido todos los cuestionarios pendientes *Figura A.9*. En este momento es cuando entramos en la sala de espera, en la que podemos tener una conversación con el bot y este avisará al usuario en el momento que otro cuestionario se encuentre activo según las planificaciones.

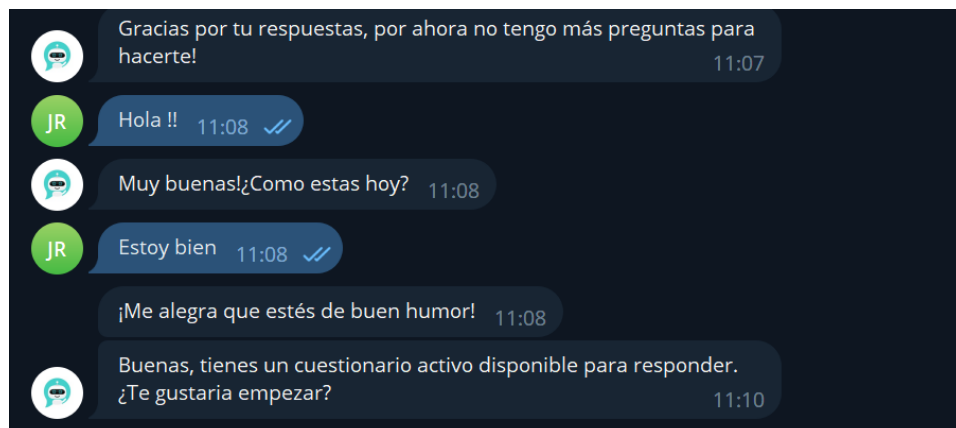


Figura A.9: Ejemplo de conversación

A.2.5. Administración de Respuestas

Finalmente, volvemos a la aplicación y esta vez nos dirigimos al último apartado de respuestas. Podemos ver en la *Figura A.10* que se han generado las repuestas a la preguntas que hemos respondido.

Block	Question	Response	Suscriber	Date
Bloque de prueba 1	Pregunta de prueba 2	Respuesta 2 - Pregunta 2	Juanitodrg22	Oct. 17, 2023, 11:11 a.m.
Bloque de prueba 1	Pregunta de prueba 1	Respuesta 1 - Pregunta 1	Juanitodrg22	Oct. 17, 2023, 11:11 a.m.
Bloque de prueba 2	Pregunta de prueba 3	Respuesta 2 - Pregunta 3	Juanitodrg22	Oct. 17, 2023, 11:06 a.m.

Figura A.10: Vista listado de respuestas

Si clicamos en el botón de exportar a CSV nos descarga un archivo que contiene ordenado por columnas la información a las respuestas proporcionadas (*Figura A.11*).

	A	B	C	D	E
1	Block	Question	Response	Subscriber	Date
2	Bloque de prueba 1	Pregunta de prueba 1	Respuesta 1	Juanitodrg22	17/10/2023
3	Bloque de prueba 1	Pregunta de prueba 2	Respuesta 2	Juanitodrg22	17/10/2023
4	Bloque de prueba 2	Pregunta de prueba 3	Respuesta 2	Juanitodrg22	17/10/2023

Figura A.11: Respuestas CSV

Bibliografía

- Adamopoulou, Eleni y Lefteris Moussiades (2020). “An Overview of Chatbot Technology”. En: URL: https://link.springer.com/chapter/10.1007/978-3-030-49186-4_31?ref=blog.min.io.
- Anokye, Reindolf (2020). “Epidemiology of mental health problems in COVID-19: a review”. En: URL: <https://www.w3schools.com>.
- “AWS” (2022). En: URL: <https://aws.amazon.com/>.
- Celi-Parraga, Ricardo Javier (2021). “Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional textual”. En: *AlfaPublicaciones*, 3(4.1), 40–52. URL: <https://doi.org/10.33262/ap.v3i4.1.123>.
- “Código Chatbot: cuando la humanidad se robotiza” (2023). En: URL: <https://comuniza.com/blog/codigo-chatbot-para-empresas>.
- Cruz, Cristhian Santa (2022). “Curso Django y Django Rest Framework Profesional”. En: URL: <https://www.udemy.com/course/curso-profesional-de-programacion-web-con-python-y-django/?kw=curso+django+y+django+rest+frame&src=sac>.
- Danielle Ramo, PhD (2019). En: URL: <https://mhealth.jmir.org/2019/10/e15018/>.
- “Django Documentation” (s.f.). En: (). URL: <https://docs.djangoproject.com/en/4.2/>.
- “Django ORM” (s.f.). En: (). URL: <https://docs.djangoproject.com/en/4.2/topics/db/queries/>.
- “Docker” (s.f.). En: (). URL: <https://docs.docker.com/desktop/install/windows-install/>.

- Elizabeth Muller, Ashley (2020). "The mental health impact of the covid-19 pandemic on healthcare workers, and interventions to help them: A rapid systematic review". En: *Psychiatry Research*. URL: <https://www.sciencedirect.com/science/article/pii/S0165178120323271/pdf?md5=645ade4bebbc7a39ee7322d6c059c73b&pid=1-s2.0-S0165178120323271-main.pdf>.
- English, Lexico Dictionaries (2023). "Definición y significado de chatbot". En: URL: <https://www.lexico.com/en/definition/chatbot>.
- Floridi, Luciano (2020). "GPT-3: Its Nature, Scope, Limits, and Consequences". En: URL: <https://link.springer.com/article/10.1007/s11023-020-09548-1>.
- "Github" (2023). En: URL: <https://github.com>.
- Gore, Himanshu (2021). "Django: Web Development Simple Fast". En: URL: <http://www.annalsofrscb.ro/index.php/journal/article/view/6301/4788>.
- HIFAS, Editorial (2023). "Bienestar emocional: las claves para estar y sentirse bien". En: URL: <https://hifasdaterra.com/blog/bienestar-emocional/>.
- "Impacto de la covid-19 en la salud física y mental de la población adulta española" (2021). En: URL: <https://elobservatoriosocial.fundacionlacaixa.org/-/impacto-de-la-covid-19-en-la-salud-fisica-y-mental-de-la-poblacion-adulta-espanola>.
- Journal TechInnovation* (2022). Springer Nature. URL: <https://revistas.unesum.edu.ec/JTI/index.php/JTI/article/view/12>.
- kettle, Liam y Yi-Ching Lee (2023). "Human Factors: The Journal of the Human Factors and Ergonomics Society". En: *Sage Journals*. URL: <https://journals.sagepub.com/doi/abs/10.1177/00187208231162453>.
- "Moodfit" (2016). En: URL: <https://www.getmoodfit.com/>.
- "Moodpath" (2016). En: URL: <https://moodpath.es.aptoide.com/app>.
- OMS (2022). "Mental Health and COVID-19: Early evidence of the pandemic's impact". En: URL: https://www.who.int/publications/i/item/WHO-2019-nCoV-Sci_Brief-Mental_health-2022.1.

- Pena Trapero, Bernardo (2009). "The Measurement of Social Welfare: A Critical Review". En: URL: <http://www.revista-eea.net/documentos/27206.pdf>.
- "POSTCOVID-AI" (2023). En: URL: <https://projects.ugr.es/postcovid-ai/>.
- "PostgreSQL" (s.f.). En: (). URL: <https://www.postgresql.org/>.
- "Python" (s.f.). En: (). URL: <https://www.python.org/>.
- "python-telegram-bot" (2015-2023). En: URL: <https://python-telegram-bot.org/>.
- Smutny, Pavel y Petra Schreiberova (2020). "Chatbots for learning: A review of educational chatbots for the Facebook Messenger". En: *Elsevier*. URL: <https://www.sciencedirect.com/science/article/pii/S0360131520300622/pdf?md5=fd500687e51f6ceacc8de69616f6f4837&pid=1-s2.0-S0360131520300622-main.pdf>.
- Talent (2022). En: URL: <https://es.talent.com/salary?job=programador#:~:text=El%20salario%20programador%20promedio%20en,hasta%20%E2%82%AC%2038.000%20al%20a%C3%B1o..>
- "Web Online Tutorials" (2023). En: URL: <https://www.w3schools.com>.
- Williams, Ruth (2021). "21-Day Stress Detox: Open Trial of a Universal Well-Being Chatbot for Young Adults". En: *Soc. Sci., 2021*. URL: <https://www.mdpi.com/2076-0760/10/11/416>.
- "Woebot" (2017). En: URL: <https://woebothealth.com>.

