universidad de león

Escuela de Ingenierías
Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Máster

Desarrollo de una *app* paliativa contra el *bullying* y la violencia de género aplicando *Sentimental Analysis*

Development of a palliative app against bullying and gender violence applying Sentimental Analysis

Autor: César Gutiérrez Pérez
Tutor: Héctor Alaiz Moretón

(Diciembre, 2019)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA INFORMÁTICA
Trabajo de Fin de Máster

ALUMNO: César Gutiérrez Pérez

TUTOR: Héctor Alaiz Moretón

TÍTULO: Desarrollo de una *app* paliativa contra el *bullying* y la violencia de género aplicando *Sentimental Analysis*

TITLE: Development of a palliative app against bullying and gender violence applying Sentimental Analysis

CONVOCATORIA: Diciembre, 2019

RESUMEN:

Connect se presenta como una solución paliativa destinada a las víctimas de *bullying* y de violencia de género. El objetivo principal consiste en brindar a estas personas un canal de comunicación anónimo, fiable, rápido y seguro, para contactar con psicólogos especializados que les ayuden a superar su situación y a denunciar a los agresores.

Se propone una aplicación móvil, para *iOS* y *Android*, destinada a establecer una comunicación en línea con psicólogos. En segundo lugar, se plantea la creación y el posterior uso de un modelo predictivo de *Machine Learning* para clasificar los mensajes de los usuarios víctimas y mejorar así el uso y la experiencia que ofrece la aplicación.

La aplicación móvil se ha desarrollado con *Angular 7* de forma modular utilizando el modelo vista controlador. A su vez, se ha implementado *Socket.IO* para establecer comunicaciones en tiempo real, rápidas y seguras. Además, se ha llevado a cabo un mecanismo de ocultación para proteger al usuario, que consiste en enmascarar la funcionalidad real mostrando una calculadora al abrir la aplicación, que solo muestra la aplicación verdadera tras introducir un código secreto elegido por el usuario.

El modelo predictivo, basado en el análisis de sentimientos, se utiliza para clasificar la polaridad de los mensajes que envían los usuarios de la aplicación. Con el uso de este modelo se pretende detectar el grado de amenaza que sufre un determinado usuario y reflejar el grado de progreso de este, para ver como mejora o empeora mientras utiliza la aplicación.

ABSTRACT:

Connect is presented as a palliative solution for victims of bullying and gender-based violence. The main objective is to provide these people with an anonymous, reliable, fast and safe channel of communication, to contact specialized psychologists that can help them overcome their situation and to denounce the aggressors.

A mobile application is proposed, for *iOS* and *Android*, to establish an online communication with psychologists. Secondly, the creation and subsequent use of a Machine Learning predictive model is proposed to classify the messages of victim users and thus improve the use and experience offered by the application.

The mobile application has been developed with *Angular 7* in a modular way using the view controller model. At the same time, *Socket.IO* has been implemented to establish communications in real time, fast and secure. In addition, a concealment mechanism has been implemented to protect the user, which consists of masking the real functionality by displaying a calculator when opening the application, which only shows the real application after entering a secret code chosen by the user.

The predictive model, based on Sentiment Analysis, is used to classify the polarity of messages sent by users using the application. The use of this model is intended to detect the degree of threat that a user suffers and reflect the degree of progress of him, to see how it improves or worsens while using the application.

Palabras clave: bullying, violencia de género, smartphone, comunicación, tiempo real, Angular, Python, Machine Learning, Sentiment Analysis.

Firma del alumno:	**VºBº Tutor/es:**

Índice de contenidos

Índice de figuras

Índice de tablas

0. Introducción

0.1 Contexto y antecedentes del proyecto

Este proyecto surge para dar respuesta a la creciente necesidad de quienes sufren algún tipo de acoso o de abuso, de ponerse en contacto con psicólogos de forma anónima y segura para solucionar la situación.

En España, durante el año 2017 se registraron un total de 1.054 denuncias por casos de acoso escolar, según datos recogidos Cuerpo Nacional de Policía, Guardia Civil y cuerpos de Policía Local, lo que implica un aumento de un 11,65% respecto al año anterior.[1] Esta cifra asciende hasta el 90% en el caso del acoso a través de Internet y las nuevas tecnologías.[2]

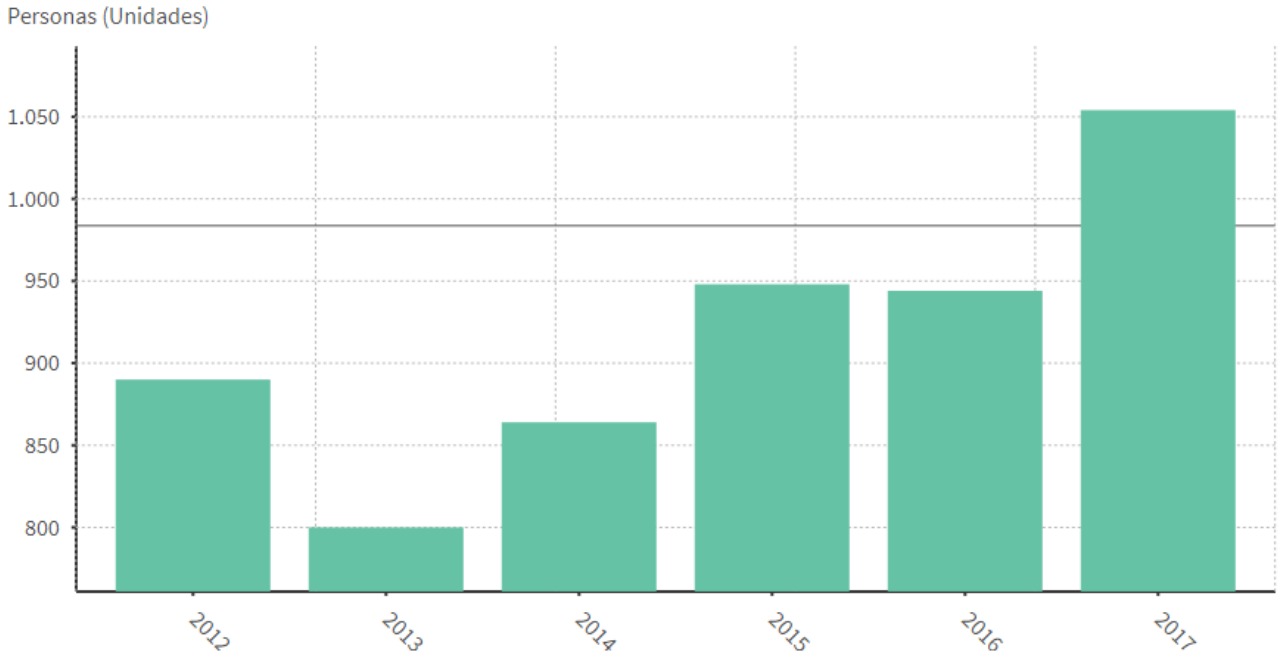


Figura 0.1 - Denuncias de bullyng en España cada año

La Fundación ANAR afirma que la edad media en la que comienzan a producirse las primeras situaciones de acoso escolar se sitúa en los 12,2 años, afectando en un mayor número de casos a las niñas (66%).[2]

Según indica el informe Violencia Viral, realizado por la asociación *Save the Children* en base a una investigación realizada con 400 jóvenes de entre 18 y 20 años, se estima que 7 de cada 10 personas ha sufrido algún tipo de violencia en el entorno digital siendo menor de edad.[3]

La mayoría de personas que sufre este tipo de acoso reconoce que le da miedo y vergüenza contar lo que les pasa, especialmente a su familia. Con frecuencia tiene relaciones familiares complicadas y desconoce, por falta de información, los peligros de internet.[4]

Quizás el dato más relevante y preocupante a su vez, es que solo uno de cada tres menores afectados se atreve a denunciar esta situación. La vergüenza y el miedo a posibles represalias o a sentirse señalados por la sociedad que les rodea son algunas de las principales causas.[2]

Los datos sobre el *bullying* no son uniformes en todo el territorio español, sino que varían mucho dependiendo de cada provincia.[5]

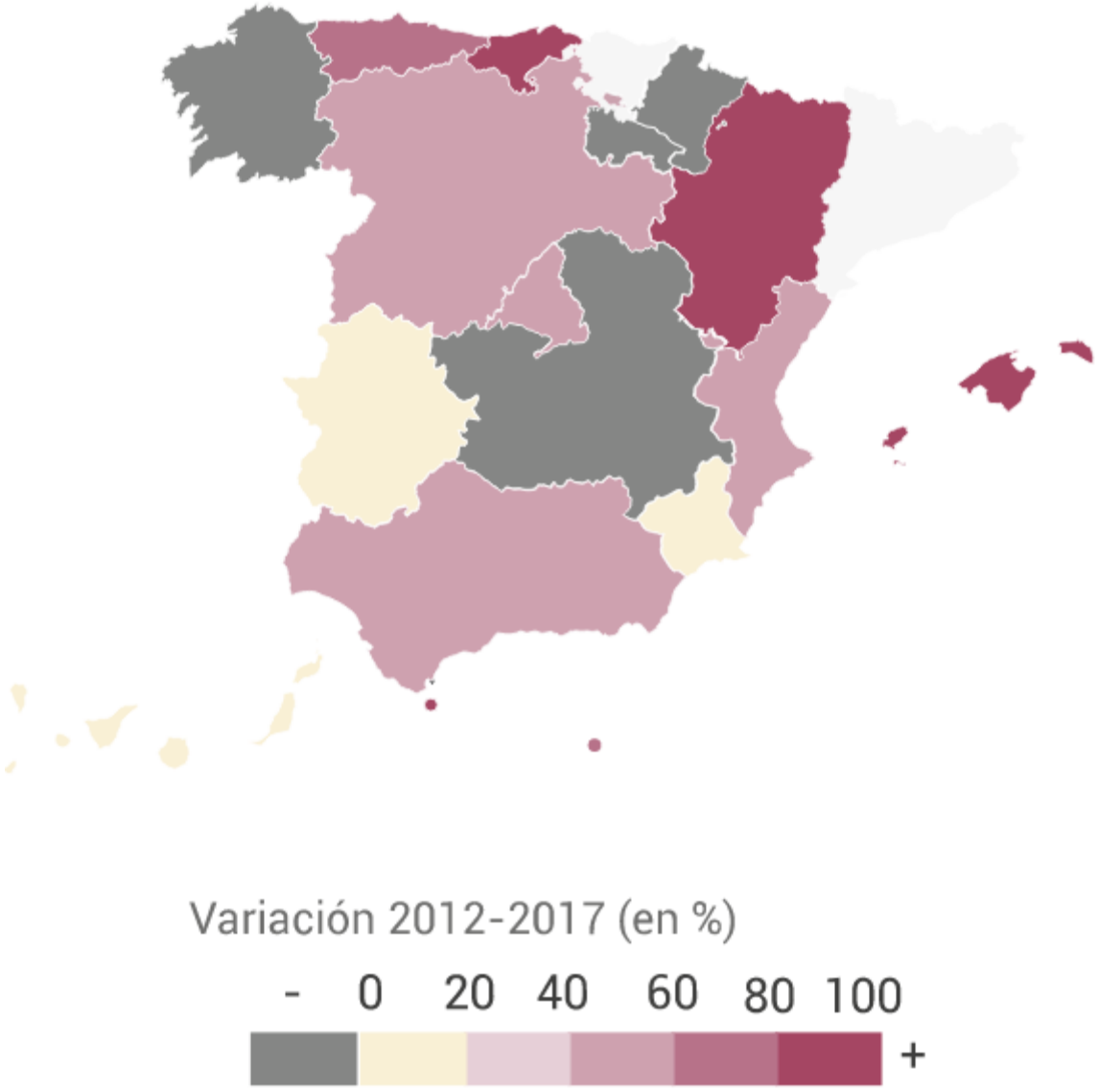


Figura 0.2 - Denuncias de bullying por comunidades

Como se puede apreciar en la figura 0.2, Cantabria es la provincia con mayor índice de acoso escolar de España, mientras que Navarra es la menor. Los datos de Cataluña y el País Vasco no están disponibles por tener las competencias transferidas.[5]

Según datos del Ministerio de Educación los tipos de acoso pueden ser: psicológico, físico, ciberacoso, social o sexual. A continuación, se explica cada uno de forma breve:[6]

1. Acoso psicológico: Se refiere a los insultos, hablar mal de alguien, amenazas para generar miedo o poner motes.
2. Acoso físico: Golpes y empujones, robos y roturas de pertenencias o, en último caso, peleas y palizas.
3. Ciberacoso: La gran mayoría de los menores denuncia insultos o risas a través de las redes, seguidos de grabaciones que, en algunos casos, incluyen contenido sexual.
4. Acoso social. La forma más frecuente es no dejar participar en actividades a la víctima, ignorar o hacer el vacío, vejaciones por discapacidad o vejaciones en grupo por el aspecto físico.
5. Acoso sexual: En este grupo se incluyen insultos o comentarios obscenos, acoso o intimidación o abusos sexuales.

Por otro lado, durante el año 2018, se presentaron en España un total de 166.961 denuncias por violencia de género, un dato bastante elevado, que presenta un aumento del 0,4% respecto al año anterior. En torno a un 69% de las denuncias presentadas fueron por la propia víctima, directamente en el juzgado o a través de atestados policiales. Las denuncias por intervención directa de la Policía se sitúan en torno al 15% de los casos, mientras que el número de denuncias presentadas por familiares de la víctima alcanzó un porcentaje superior al 4%.[7]

Según el Barómetro Juventud y Género, el 49% de los jóvenes españoles ha afirmado que la violencia de género ha ido en aumento en los últimos años. En el informe también se muestra el número de mujeres que ha sido testigo de algún caso de violencia de género en su círculo más cercano. Así, casi la mitad de las mujeres han sido testigo de algún tipo de violencia, destacando la revisión del móvil por parte de su pareja, que lo afirma el 48,8% de las encuestadas. El 50,9% asegura haber sido testigo de amenazas o insultos empleando internet o el móvil y el 48,5% afirma que conoce situaciones en las que la pareja controla todo lo que hace.[8]

El mayor número de denuncias por violencia de género se corresponde con jóvenes entre 16 y 25 años y supone un 47% del total.

Un estudio realizado por el Observatorio contra la Violencia Doméstica y de Género refleja que las mujeres que se atreven a romper el silencio y denunciar a su agresor no tienen garantizada su seguridad en muchos de los casos.[9]

Al igual que ocurría con el *bullying*, los datos no son uniformes en todo el panorama nacional, ya que presentan una notable variación dependiendo de cada provincia.

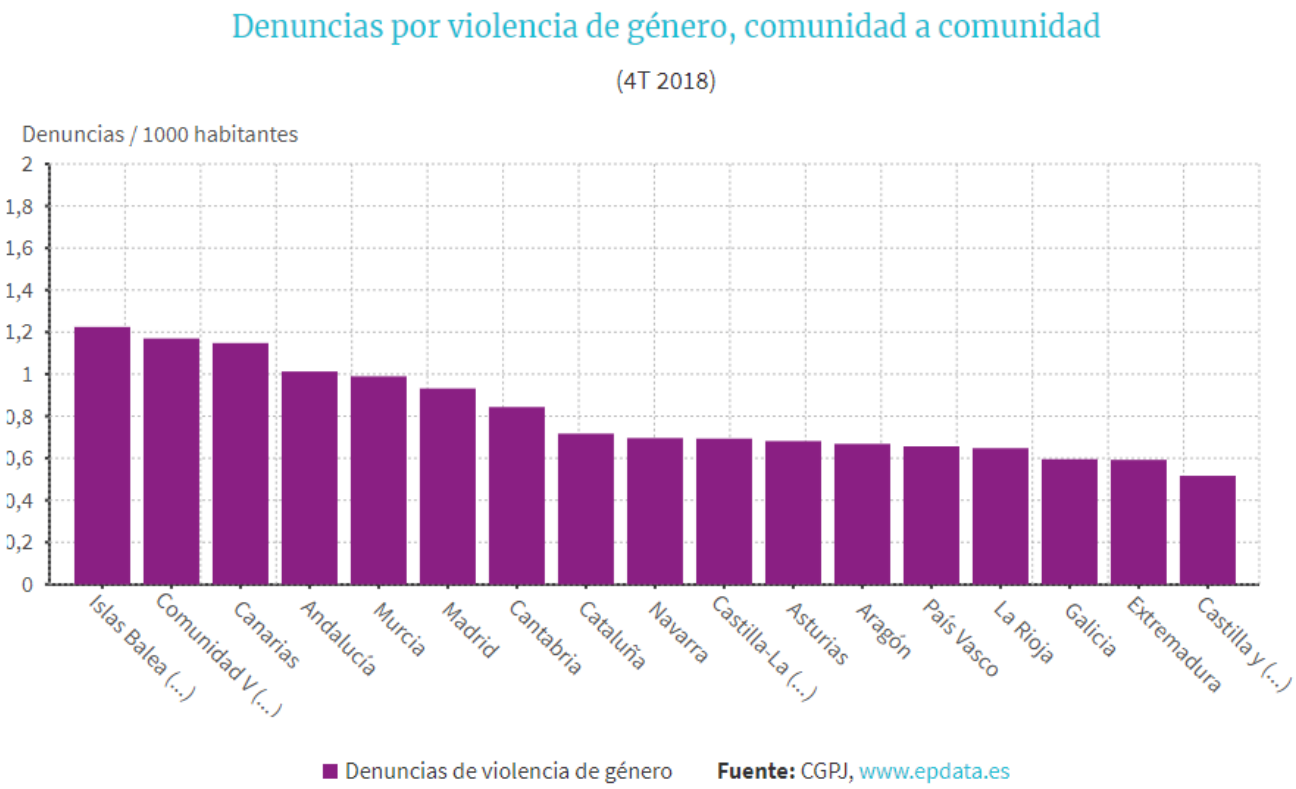


Figura 0.3 - Denuncias de violencia de género por comunidad

Como se observa en la figura 0.3, la mayor tasa de denuncias realizadas por víctimas de violencia de género pertenece a las Islas Baleares, mientras que la menor tasa corresponde a la comunidad de Castilla y León.[10]

Al igual que ocurre con el *bullying*, uno de los principales problemas de este tipo de violencia es el silencio de las víctimas.[9] Según el Observatorio contra la Violencia Doméstica y de Género, el silencio de la víctima es un factor de riesgo para la vida de las personas que sufren este tipo de violencia. Se debe, por tanto, concienciar a la víctima de la necesidad de denunciar y de ponerse en contacto con una persona especializada, por ejemplo, un psicólogo.

El concepto de violencia de género incluye una amplia variedad de actitudes y actuaciones que pueden dañar a la persona desde diferentes dimensiones. Si bien no en todos los casos se agrede a la persona desde todos los ámbitos, dentro de la violencia de género podemos encontrar los siguientes tipos de violencia:[11]

1. Violencia física: Se considera violencia física todo aquel acto en que se inflige un daño físico a la víctima que a través de la agresión directa.
2. Violencia psicológica: La víctima se ve humillada, minusvalorada y atacada psicológicamente.
3. Violencia sexual: Se refiere concretamente al tipo de situaciones en que una persona es forzada o coaccionada para llevar a cabo actividades de índole sexual.
4. Violencia económica: Se basa en la reducción y privación de recursos económicos a la víctima.
5. Violencia patrimonial: Consiste en la usurpación o destrucción de objetos, bienes y propiedades.
6. Violencia social: Se basa en la limitación, control y la inducción al aislamiento social de la persona.
7. Violencia vicaria: Consiste en amenazar, agredir e incluso matar a los hijos de una persona para infligir algún tipo de control sobre esta.

0.2 Justificación del proyecto

El proyecto que se expone a continuación se presenta como una solución capaz de cubrir la demanda generada por las personas que son víctimas de cualquier tipo de acoso o de abuso. Se centra principalmente en el acoso que afecta a menores de edad y en la violencia de género. Ambos tienen algo en común: las personas que lo sufren no se atreven a denunciarlo ni a ponerse en contacto con un experto porque sienten vergüenza, miedo a sufrir represalias o temen sentirse señaladas por la sociedad.

La principal forma de solucionar este problema es brindar a estas personas un canal de comunicación anónimo, fiable y seguro, en el que se pongan en contacto con expertos que les ayuden a superar su situación y a denunciar a los agresores.

Connect ofrece un conjunto de aplicaciones, tanto móviles como webs, que permiten a los usuarios establecer una comunicación con psicólogos que cubran los problemas específicos

de cada víctima, centrándose en dos puntos clave: el anonimato y la posibilidad de ofrecer a estas personas una alternativa viable que les ayude a paliar su problema permitiéndoles tener sesiones de forma inmediata y en cualquier momento del día.

Del mismo modo, los psicólogos también cuentan con una suite de aplicaciones que les permiten gestionar de forma sencilla y personalizada las conversaciones con los usuarios a los que atienden, permitiendo especificar el número que pueden atender. Por último, todos los psicólogos cuentan con un panel privado y uno público por cada víctima, en el que pueden añadir notas sobre las sesiones para sus futuras consultas o información relevante que consideren que otros psicólogos deben tener en cuenta.

Para entender *Connect* se debe recordar el perfil psicológico de las personas que han sufrido o están sufriendo cualquier tipo de acoso, teniendo en cuenta que la mayoría de estas personas tienden a cerrarse en su mundo ocultando lo ocurrido al resto de personas. *Connect* ofrece una forma de romper esta barrera ofreciendo a los usuarios una herramienta de comunicación anónima y una vía de desahogo sin tener que enfrentarse al contacto de forma física, punto al que ofrecen reticencia.

Por otro lado, este proyecto supone un gran reto en el ámbito tecnológico, ya que abarca diversas áreas de estudio de la rama informática como son: la comunicación, la seguridad, la ingeniería del software, el diseño conceptual, el análisis de datos o el *Machine Learning*.

Otra de las grandes ventajas de este proyecto es que se ha realizado todo el desarrollo desde cero, utilizando los estándares y las tecnologías más novedosas que existen a día de hoy.

0.3 Alcance del proyecto

El proyecto *Connect* consta de cuatro subproyectos, cada uno con una función clara, específica y perfectamente diferenciada de los demás. En esta memoria se tratará de explicar de forma detallada la aplicación móvil que utilizarán los usuarios que son víctimas de acoso o abuso para ponerse en contacto con expertos que les brinden ayuda. Además de esta aplicación móvil, *Connect* cuenta también con una aplicación web destinada a los psicólogos que se encargarán de ayudar a estas personas. Algo que comparten estas dos aplicaciones es un *Back-end* que integra una *API REST*, encargada de ofrecer una interfaz entre sistemas con el fin de obtener datos o generar operaciones sobre esos datos en todos los formatos posibles. No hay que olvidar tampoco el uso de *Machine Learning* para la

creación de un modelo predictivo que se encargará de clasificar los mensajes de los usuarios víctimas.

0.3.1 Aplicación móvil para usuarios que sufren acoso o abuso

La aplicación móvil para los usuarios que sufren acoso está desarrollada tanto para *iPhone* como para dispositivos *Android*. Consta, en primer lugar, de una sección de registro en la que el usuario debe crear una cuenta rellenando unos pocos datos que son muy relevantes para brindarle ayuda. Estos datos no incluyen información personal, tan solo información sobre el tipo de problema o la situación en la que se encuentra el usuario, con el fin de buscar psicólogos expertos en ese campo, siempre garantizando el anonimato de la persona si esta así lo desea. Una vez el usuario ya tenga una cuenta se le ofrecerá activar el mecanismo de ocultación de la aplicación, que consiste en enmascarar la funcionalidad real de la aplicación y mostrar una aplicación básica no relacionada como, por ejemplo, una calculadora. De esta forma, si un tercero se hace con el móvil de la víctima no puede ver si utiliza *Connect* porque se mostraría otra aplicación.

Una vez el usuario inicia sesión en la aplicación tiene acceso a una lista de psicólogos especializados en su problema concreto. Los psicólogos se muestran en base a un algoritmo que tiene en cuenta la valoración media de cada psicólogo y la disponibilidad que establece cada uno.

Los usuarios pueden valorar en todo momento al psicólogo con el que se comunican y también tienen acceso a la media de valoraciones de cada uno de estos.

El chat para la comunicación con psicólogos está basado en una *API REST* robusta y en la implementación de un *socket* para poder realizar comunicaciones en tiempo real.

Los usuarios tienen a su disposición un botón del pánico, que consiste en un mecanismo salvavidas que enviará la ubicación del usuario a la policía en caso de que el usuario tenga la necesidad.

0.3.2 Modelo predictivo con Machine Learning

El proyecto cuenta también con un modelo predictivo entrenado con un conjunto de datos de datos extraídos de *Twitter*, que se han clasificado según su polaridad en positivos, negativos y neutros. Este modelo se encargará de calcular la polaridad de los mensajes que envían los usuarios víctimas y se utilizará para:

- Detectar de grado de amenaza que sufre el usuario mediante un análisis de sentimientos de sus mensajes con el fin de asignar mayor prioridad a los usuarios que sufren mayor grado de amenaza.
- Reflejar el grado de progreso de un usuario. Esta funcionalidad permite obtener estadísticas de como un usuario mejora o empeora mientras utiliza la aplicación.

La creación del modelo predictivo engloba todos los *scripts* en lenguaje *Python* que se han utilizado para las diversas tareas y pruebas necesarias, como son:

- Selección del conjunto de datos.
- Limpieza de datos.
- Entrenamiento de diferentes modelos predictivos.
- Pruebas de validación cruzada y comparativa de modelos.
- Integración del modelo predictivo en la aplicación de usuarios víctima.

0.3.3 Aplicación para psicólogos

La aplicación destinada a los psicólogos es una parte complementaria de *Connect* que ha sido desarrollada por Jesús García Potes y no se tratará en esta memoria, salvo que alguna parte sea imprescindible para entender el proyecto.

0.3.4 Back-end

Connect cuenta también con un *Back-end* desarrollado con el fin de crear un sistema de comunicación entre las aplicaciones de usuario y de psicólogo de forma efectiva. La seguridad y la eficiencia han sido dos puntos importantes a tener en cuenta durante el desarrollo.

El *Back-end* implementa una *API REST* encarda de ofrecer una interfaz de programación de aplicaciones que se apoya en la arquitectura *REST* para el desarrollo de aplicaciones en red. Se pueden realizar todo tipo de consultas mediante estas cuatro operaciones: *GET*, *POST*, *PUT* y *DELETE*.

Además de la *API REST*, el *Back-end* tiene implementado un *socket* para realizar comunicaciones rápidas y en tiempo real entre un usuario y un psicólogo. Se ha tenido en cuenta la seguridad y se ha implementado un *middleware* para garantizar la comunicación de extremo a extremo.

0.4 Tecnologías utilizadas

Antes de comenzar con la planificación y el desarrollo de este proyecto, se realizó un estudio previo de las tecnologías más modernas, seguras y con mayor soporte.

En este proyecto se utilizan una gran variedad de tecnologías que son muy diferentes entre sí.

- **Aplicación móvil para usuarios que sufren acoso o abuso**: Consiste en una aplicación móvil hibrida desarrollada con tecnologías web, para *Android* e *iOS*.
    - Ha sido desarrollada con *Angular 7* de forma modular utilizando el modelo vista controlador (*MVC*) para poder controlar la escalabilidad futura. *Angular* es un *framework* de código abierto desarrollado por Google con el objetivo de crear aplicaciones web dinámicas y modernas. *Angular* permite crear aplicaciones interactivas y dinámicas de una sola página. Estas aplicaciones incluyen plantillas con enlaces a los datos, inyección de dependencias y manejo de *AJAX* de forma nativa y enfocado al uso de *API REST*.[12]

Figura 0.4 - Logo de Angular

    - Una interfaz web en *HTML5*, desarrollada con el *framework Bootstrap*[13] junto con *CSS* propio.

Figura 0.5 - Logo de HTML5

Figura 0.6 - Logo de CSS3

Figura 0.7 - Logo de Bootstrap

- **Análisis de Datos y *Machine Learning***: Engloba el software necesario para seleccionar un conjunto de datos de entrenamiento y un algoritmo de predicción basado en el análisis de sentimientos:
    - La totalidad de este software ha sido desarrollado con *Python 3.6*. *Python* es una herramienta computacional poderosa utilizada para resolver tareas complicadas en el área de finanzas, economía, ciencia de los datos y *Machine Learning*. *Python* cuenta también con una gran y activa comunidad de personas que se encargan de mejorar esta herramienta y de crear librerías útiles para el análisis de datos y *Machine Learning*.[14]

Figura 0.8 - Logo de Python

    - Algunas librearías que se han utilizado para este proyecto son las siguientes:
        - *Numpy*: Proporciona una estructura de datos de matriz que tiene algunos beneficios sobre las listas nativas de *Python*. Proporciona potentes estructuras de datos, como las matrices multidimensionales.[15]
        - *Pandas*: Cuenta con estructuras de datos que necesarias para limpiar los datos en bruto y que sean aptos para el análisis, el procesado y la visualización de datos de forma sencilla.[16]
        - *NLTK*: Es un conjunto de bibliotecas y herramientas para el procesamiento del lenguaje natural (*PLN*). Permite realizar múltiples tareas relacionadas con el análisis de texto.[17]
        - *Scikit-learn*: Es una biblioteca de *Machine Learning* que ofrece una serie de algoritmos de clasificación, regresión, *clustering* y reducción de dimensionalidad, para desarrollar modelos predictivos.[18]
        - *Joblib*: Es una herramienta muy útil para exportar y cargar modelos predictivos en *Python*.[19]

- ***Back-end***: Hace referencia a la parte del desarrollo web que se encarga de toda la lógica y de la comunicación entre las aplicaciones del lado del cliente. Ha sido desarrollado con las siguientes tecnologías:
    - *Node.js*: Es un entorno que utiliza *JavaScript* de lado del servidor de forma asíncrona y con una arquitectura orientada a eventos.[20]

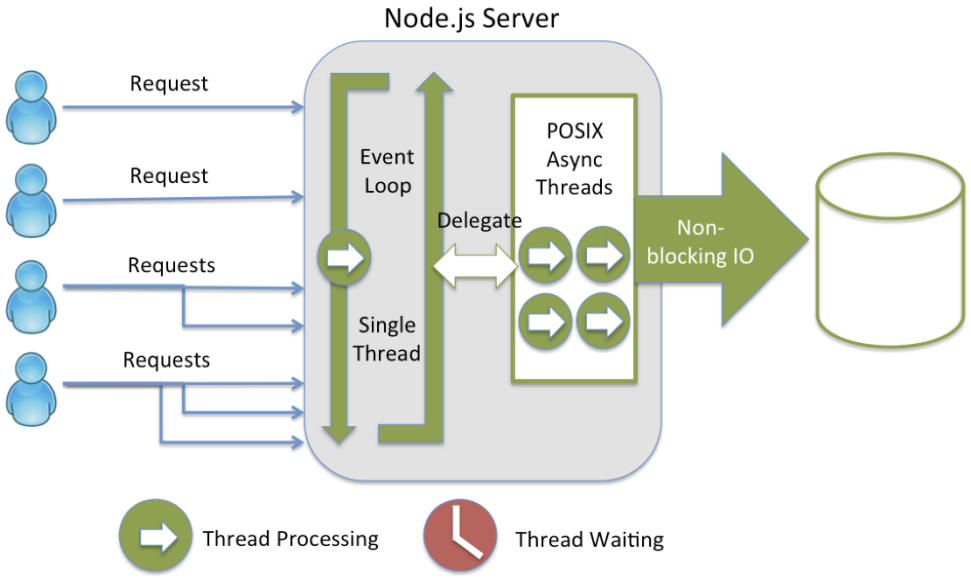


Figura 0.9 - Diagrama de un servidor Node.js

    - *Express*: Es un *framework* para *Node.js* que sirve para crear aplicaciones web de forma minimalista y flexible en poco tiempo, ya que proporciona funcionalidades como enrutamiento, opciones para gestionar sesiones y cookies.[21]
    - *Socket.IO*: Es una librería de *JavaScript* para *Node.js* que permite una comunicación bidireccional en tiempo real entre cliente y servidor.[22] Se basa principalmente en *websockets* pero también puede usar otras alternativas como *sockets* de *Adobe Flash*, *JSONP polling* o *long polling* en *AJAX*, seleccionando la mejor alternativa para el cliente en tiempo de ejecución.[23]
    - *MongoDB*: Es una base de datos *NoSQL* orientada a documentos. Esto significa que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en *BSON*, que es una representación binaria de *JSON*.[24]

1. Estado del arte

1.1 Descripción del problema

Las aplicaciones móviles son programas informáticos creados para que las personas puedan acceder a diversas tareas mediante dispositivos móviles. El futuro de las *apps* móviles está en auge, ya que en el año 2017 se estimaron alrededor de 268.000 millones de descargas en aplicaciones móviles. El ranking europeo de descargas de apps móviles está encabezado por España, ya que usar una aplicación móvil es más rápido, ameno e interactivo que un ordenador. En la sociedad es común el uso del teléfono móvil y hoy en día los *smartphones* se han convertido en un tercer brazo, por lo que es lógico que haya aumentado el uso de las *apps* móviles.[25]

En relación al proyecto, en los últimos años han surgido un gran número de aplicaciones encargadas de poner en contacto a usuarios con expertos, que pueden ir desde psicólogos, médicos y hasta abogados. La principal diferencia de esta forma de interaccionar, frente a las relaciones cotidianas, es la sencillez y rapidez, pues vivimos en un mundo globalizado en el que cada segundo es valorado de forma incalculable.

Muchas de estas aplicaciones están destinadas únicamente a proporcionar un canal de comunicación a personas que sufren diversos problemas de acoso o abuso. Estas aplicaciones surgen para cubrir la necesidad que tienen estas personas de ponerse en contacto con expertos de forma fácil y rápida.[26]

Muchas de estas víctimas sufren en silencio porque no se atreven a denunciar su situación ni a ponerse en contacto con un experto por miedo a sufrir represalias o sentirse señaladas por la sociedad. Esto ocasiona que el problema se interiorice en la persona y que cuando se atreva a dar el paso el problema ya este interiorizado y sea mucho más difícil de solucionar.[27]

Las aplicaciones que ofrecen ayuda de forma online constituyen una alternativa rápida y sencilla para solicitar ayuda y denunciar la situación en la que viven estas personas. El auge de las aplicaciones móviles en el campo de la psicología ha supuesto una revolución a la hora de establecer canales de comunicación alejados de las vías tradicionales. Ya no es necesario acudir a una consulta físicamente, ya que estas aplicaciones suponen el establecimiento de consultas rápidas, anónimas y eficaces. [25]

Haciendo una investigación del uso de los teléfonos en los menores españoles, se concluyó que el número de niños que tienen un *smartphone* es cada vez es mayor, y que cada año la edad en la que lo adquieren es más baja. Sin entrar en detalle de si se trata de un punto negativo o positivo, se observa que la mejor forma de llegar a este público era mediante el uso de *smartphones*, ofreciéndoles una *app* que permita ponerse en contacto con expertos de forma anónima.[28]

Teniendo el punto de entrada de los usuarios claro, la parte de los psicólogos no podría ser otra que un chat para comunicarse con estos usuarios, ofreciéndoles una forma sencilla, rápida y eficaz de poder ayudar a estas personas. Este canal de comunicación debe ser una aplicación web, donde el psicólogo pueda manejar en todo momento, el número de usuarios con los que puede trabajar y donde existe un panel de notas para anotar toda la información del usuario, por si en algún momento decide, traspasar el usuario a otro psicólogo.

Un punto a tener en cuenta en este tipo de aplicaciones es el análisis de datos para sacar conclusiones relevantes y la inteligencia artificial para mejorar la experiencia que se puede brindar a los usuarios. Por este motivo, se plantea utilizar un modelo predictivo basado en el análisis de sentimientos que se encargue de predecir la polaridad de los mensajes que envía un usuario para asignar mayor prioridad a usuarios que sufren problemas más graves y para reflejar el grado de progreso de un usuario desde que utilizo la aplicación por primera vez.

El análisis de sentimientos es el campo de estudio que analiza las opiniones, sentimientos, evaluaciones, actitudes y emociones de las personas a partir del lenguaje escrito. Es una de las áreas de investigación más activas en el procesamiento del lenguaje natural y también es ampliamente estudiada en minería de datos, minería web y minería de texto.[29] La creciente importancia del análisis de sentimientos coincide con el crecimiento de las redes sociales como *Twitter*, las reseñas, los debates en los foros y los blogs. Por primera vez en la historia humana, ahora tenemos un gran volumen de datos grabados en forma digital para su análisis.[30]

1.2 Análisis de la competencia

En los últimos años han surgido una gran variedad de aplicaciones encargadas de poner en contacto a usuarios, que sufren diversos problemas, con expertos, que pueden ir desde psicólogos, médicos hasta abogados. Estas aplicaciones ofrecen una buena alternativa a esas personas que, por miedo o vergüenza, no son capaces de denunciar la situación que viven.

Entre estas caben destacar las siguientes:

1. **Psonrie**: Es un servicio de orientación psicológica online y completamente anónimo, especializado en problemas del día a día. Permite poner en contacto a un usuario con un psicólogo profesional cuando lo necesite. Está disponible tanto para *Android* como para *iOS*.[31]
2. **Psicoglobal**: Es una aplicación destinada a la terapia psicológica online en general. Es útil para pacientes que se han trasladado o viajan frecuentemente.[32]
3. **Ypsihablamos**: Este servicio ofrece psicología online de calidad. Los usuarios, en primer lugar, explican su problema y se les asigna un psicólogo especializado.[33]
4. **Workingminds**: Es un servicio que ofrece la posibilidad de realizar videollamadas con un psicólogo de forma flexible y económica.[34]
5. **Andrea.**: Es una aplicación que proporciona al usuario la posibilidad de alertar de forma anónima situaciones de acoso escolar.
6. **mediQuo**: Es una aplicación que dispone de varias salas de chat grupales para comunicarse con médicos y especialistas. Tiene un propósito general.[35]
7. **B-resol**: Esta aplicación ofrece una solución para luchar contra el acoso, el *ciberbullying*, los trastornos alimentarios y otros problemas que afectan a los adolescentes.[36]
8. **ZeroAcoso**: Es una herramienta basada en la comunicación online de forma anónima y confidencial que hace posible abordar el problema que supone el acoso escolar.[37]
9. **Kitestring**: Es una aplicación que se encarga de vigilarte cuando estás fuera y de alerta a tus amigos si no respondes.[38]

A continuación, se adjunta una tabla comparativa de estas aplicaciones en la que se muestran las características principales de cada una de ellas:

Tabla 1.1 - Comparativa 1 de aplicaciones

Aplicación	Funcionalidad	Tipo de acción	Anonimato	Ocultación	Público objetivo	Uso de *Machine Learning*
Psonrie	Orientación psicológica online de forma anónima	Paliativa	Sí	No	Genérico	No
Psicoglobal	Terapia psicológica online en general	Paliativa	No	No	Genérico	No
Vpsihablamos	Psicología online de forma general	Paliativa	No	No	Genérico	No
Workingminds	Videollamadas con un psicólogo de forma flexible y económica	Paliativa	No	No	Genérico	No
Andrea.	Alertar de forma anónima de acoso escolar	Preventiva	Sí	No	Específico	No
MediQuo	Chats grupales para comunicarse con médicos y especialistas	Paliativa	No	No	Genérico	No
B-resol	Enfocada a combatir el acoso escolar	Preventiva	Sí	No	Específico	No
ZeroAcoso	Combatir el acoso escolar de forma anónima	Preventiva	Sí	No	Específico	No
Kitestring	Vigilarte si estás fuera y alertar si no respondes	Preventiva	No	Vía SMS	Específico	No
Connect	Comunicación con psicólogos específicos de forma anónima y segura	Paliativa	Sí	Sí	Específico	Sí

Tabla 1.2 - Comparativa 2 de aplicaciones

Aplicación	Detección de intrusión	Colaboración entre expertos	Colaboración guiada entre usuarios	Consulta telefónica	Consulta videollamada	Sesiones instantáneas
Psonrie	No	No	No	Sí	Sí	Sí
Psicoglobal	No	No	No	Sí	Sí	No
Vpsihablamos	No	No	No	Sí	Sí	No
Workingminds	No	No	No	Sí	Sí	No
Andrea.	No	No	No	No	No	Sí
MediQuo	No	No	No	No	No	Sí
B-resol	No	No	Sí	No	No	Sí
ZeroAcoso	No	No	No	No	No	Sí
Kitestring	No	No	No	No	No	Sí
Connect	Sí	Sí	Sí	No	No	Sí

1.3 Diferencias y mejoras frente otras soluciones

Connect es una aplicación desarrollada específicamente para las personas que sufren acoso escolar o violencia de género. Proporciona una comunicación rápida, segura y específica para cada usuario.

Las principales diferencias que suponen una mejora en el uso de *Connect* frente a otras aplicaciones son:

1. El anonimato: No se utilizan el nombre y apellidos de la víctima ni otros datos personales.
2. Tratamiento paliativo y preventivo: Muchas de las herramientas listadas ofrecen un tratamiento preventivo o paliativo, mientras que las preventivas excluyen a todas las personas que ya están sufriendo el problema, la realidad es que esto no debería ocurrir e intentar frenarlo es una buena medida. Las aplicaciones paliativas se centran en curar un problema arraigado queremos poder solucionar el problema antes de que tenga consecuencias devastadoras para el usuario.
3. Aplicación enfocada a un público específico: Está destinada al *bullying* y a la violencia de género. La gran mayoría de aplicaciones se centran en prevenir el problema, pero no en el público que ya ha sufrido cualquier acoso o abuso, dejando excluidas a estas personas.
4. Ocultación: Enmascarado de la funcionalidad principal de la aplicación mediante una calculadora que activa la aplicación con un código secreto.
5. Colaboración entre psicólogos: Para una mayor disponibilidad, *Connect* tiene un panel de notas que asigna a cada usuario para que los psicólogos pueden dejar reflejada la información relevante del usuario.
6. Precio: El resto de aplicaciones ofrecen sus servicios facturando a los usuarios las sesiones que realizan, mientras que *Connect* dispone de psicólogos especializados, que se ofrecen de forma altruista empleando parte de su tiempo en sesiones, de esta forma el psicólogo obtiene la marca de psicólogo *Connect*.
7. Botón del pánico: Dispone de un mecanismo que se encarga de obtener la localización de la víctima y enviársela a la policía, en caso de que esta esté en peligro.

8. Inteligencia artificial para mejorar el servicio: *Connect* cuenta con un modelo predictivo de *Machine Learning* orientado a mejorar la calidad del servicio. Este modelo se encarga de:
    a. Detectar el grado de amenaza que sufre el usuario mediante un análisis de sentimientos que se aplica a sus mensajes con el fin de asignar mayor prioridad a los usuarios que sufren mayor grado de amenaza.
    b. Reflejar el grado de progreso de un usuario mediante un análisis de sentimientos de sus mensajes. Ofrece estadísticas de como un usuario mejora o empeora mientras utiliza la aplicación.

También existen algunas funcionalidades con las cuales *Connect* no cuenta. Muchas de ellas serán integradas en el futuro y otras no se incluirán, porque el objetivo de *Connect* está alejado de la función que realizan dichas aplicaciones. Hay una serie de aplicaciones como *Nerea*, *B-resol* y *Zero-acoso* que se centran en la prevención del problema, algo muy diferente a la función de *Connect*, que trata de resolverlo una vez este se ha producido. La idea de *Connect*, en un futuro, es evolucionar hacia un tratamiento preventivo y paliativo, pero se ha considerado que inicialmente el tratamiento paliativo es prioritario.

2. Planificación

A continuación, se describe detalladamente la planificación que se ha llevado a cabo durante el desarrollo del proyecto, mostrando la planificación inicial y planificación final junto con los cambios llevados a cabo durante el desarrollo.

Inicialmente para realizar la planificación se dividió el proyecto en pequeñas tareas, con un tiempo y unos recursos estimados para llevarlas a cabo y se establecieron las circunstancias para considerar las tareas terminadas.

Para estimar la planificación se empleó la metodología *Kanban*. Se dividió el proyecto en diversas tareas, para cada una de las cuales se estimó el valor o grado de importancia, el tiempo de trabajo, los recursos necesarios, la dependencia con otras tareas y las pruebas de aceptación necesarias.

Las asignaciones que se establecieron fueron las siguientes:

- Recursos humanos: El proyecto se realizó en compañía de Jesús García Potes, por lo que la asignación de recursos humanos fue dividida entre ambos, donde cada uno realizó una parte independiente, que en conjunto se complementaban dando lugar a *Connect*.
- Recursos materiales: Los recursos materiales necesarios para realizar el proyecto son los siguientes:
    - Un PC con *Ubuntu* que se utilizó para desarrollar las aplicaciones que conforman *Connect* y realizar baterías de tests.
    - Un *Smartphone* con un sistema operativo *Android* para realizar todas las pruebas necesarias de la aplicación orientada a los usuarios que sufren acoso o abuso.
    - Un *iPhone* para instalar la aplicación desarrollada en *iOS* y poder probar la aplicación orientada a los usuarios que sufren acoso o abuso.
    - Una *Raspberry* configurada como servidor para alojar el *Back-end* y permitir las conexiones remotas.
- Asignación del tiempo: La asignación del tiempo se realizó teniendo en cuenta que las personas que se encargaron de la realización del proyecto no disponen de demasiado tiempo, al ser estudiantes, como un trabajador normal.

2.1 Planificación inicial

Tabla 2.1 - Tarea 1: Investigación Inicial

T1. Investigación Inicial	5 días
Descripción:	
Consiste en realizar un análisis de la problemática que supone el acoso y el abuso y analizar las soluciones que existen actualmente valorando cada una de estas de forma objetiva.	
Recursos:	
Internet.	
Criterios de aceptación:	
Se recopilarán datos de los casos de acoso y de abuso que se producen en España, así como también estadísticas donde se recoja como se producen estos casos y porque las personas no se atreven a denunciarlo.	

Tabla 2.2 - Tarea 2: Búsqueda y análisis de las soluciones existentes

T2. Búsqueda y análisis de las soluciones existentes	8 días
Descripción:	
Recopilación de información sobre aplicaciones de psicología. Se ha de realizar una tabla comparativa entre las distintas aplicaciones que cubren esta necesidad con las características más relevantes e innovadoras de cada una.	
Recursos:	
Internet.	
Criterios de aceptación:	

Se estudiarán todas las aplicaciones que ofrezcan ayuda a personas de forma online sin importar que sean para un fin general como la psicología general o para uno específico como el acoso y el abuso.

Tabla 2.3 - Tarea 3: Diseño teórico de la funcionalidad del proyecto

T3. Diseño teórico de la funcionalidad del proyecto	**6 días**
Descripción:	
Con los datos recopilados sobre las aplicaciones disponibles en el mercado se realizará una selección de las características principales que ha de tener una aplicación de psicología paliativa enfocada a el acoso escolar y la violencia de género.	
Recursos:	
Recopilación de información de otras aplicaciones.	
Criterios de aceptación:	
Las funcionalidades que se han de implementar en el proyecto son las características principales de aplicaciones de psicología paliativas que se puedan añadir de forma viable en un plazo de 8 meses.	

Tabla 2.4 - Tarea 4: Asignación de recursos

T4. Asignación de recursos	**2 días**
Descripción:	
Se establecerá una asignación de los recursos necesarios para realizar el proyecto. Los principales recursos que se han de tener en cuenta son: recursos humanos, recursos materiales, asignación del tiempo.	
Recursos:	
Recursos humanos, recursos materiales.	

Criterios de aceptación:
En la asignación de recursos humanos se buscarán personas cualificadas y con experiencia en la realización de proyectos de desarrollo de aplicaciones móviles. Para la asignación de recursos materiales se tendrá en cuenta los recursos necesarios para el desarrollo del proyecto. En la asignación del tiempo se realizará un horario de trabajo por cada miembro del equipo.

Tabla 2.5 - Tarea 5: Análisis y selección de tecnologías.

T5. Análisis y selección de tecnologías.	10 días
Descripción:	
Se realizará un análisis de las diferentes tecnologías que permitan implementar las funcionalidades del proyecto, así como las adaptaciones oportunas para cumplir con los requisitos. Se procederá a buscar las tecnologías que mejor se ajusten a los requisitos.	
Recursos:	
Entorno de desarrollo para realizar pruebas, búsquedas por internet y foros de desarrollo.	
Criterios de aceptación:	
Se considerará realizada esta tarea, una vez que se establezcan las tecnologías que mejor se ajusten para la posterior implementación de la arquitectura deseada, será necesario realizar una tabla comparativa para verificar que se han escogidos las tecnologías más ventajosas.	

Tabla 2.6 - Tarea 6: Aprendizaje de las tecnologías

T6. Aprendizaje de las tecnologías	15 días
Descripción:	
Una vez seleccionadas las tecnologías necesarias para la consecución del proyecto, se investigarán dichas tecnologías en mayor profundidad, se debe adquirir conocimiento de las mismas para la posterior realización del proyecto. También se realizarán unos pequeños programas para encaminar dichas tecnologías hacia la consecución del proyecto.	
Recursos:	
Documentación de cada una de las tecnologías y entorno de desarrollo para realizar pequeñas implementaciones.	
Criterios de aceptación:	
Se considerará realizada esta tarea, una vez que se establezcan las tecnologías que mejor se ajusten para la posterior implementación de la arquitectura deseada, será necesario realizar una tabla comparativa para verificar que se han escogidos las tecnologías más ventajosas.	

Tabla 2.7 - Tarea 7: Definición de los objetivos del proyecto

T7. Definición de los objetivos del proyecto	8 días
Descripción:	
Una vez se conozcan bien las tecnologías necesarias para la consecución del proyecto, se especificarán cuáles son los objetivos que debe cumplir el proyecto así como también el plazo de consecución de estos.	
Recursos:	
Recursos humanos, recursos materiales y asignación del tiempo.	
Criterios de aceptación:	

Se finalizará esta tarea en el momento en el que se defina con precisión los objetivos y la forma de cumplir los mismos. Se realizará una documentación detallada reflejando todos los objetivos especificados, así como los criterios de aceptación de estos.

Tabla 2.8 - Tarea 8: Diseño de la arquitectura

T8. Diseño de la arquitectura	**8 días**
Descripción:	
Se realizará un estudio de diferentes modelos de arquitectura y las ventajas que suponen unos frente a otros y posteriormente se definirá una arquitectura que sea acorde y se ajuste a los objetivos del proyecto.	
Recursos:	
Internet, tecnologías seleccionadas, manuelas de desarrollo de software	
Criterios de aceptación:	
Se escogerán como válidas todas las posibles arquitecturas que se adapten a los objetivos del proyecto y se finalizará esta tarea en el momento en que se tenga diseñada una arquitectura que supera las pruebas necesarias de aceptación. Además se debe documentar dicha arquitectura de forma que se pueda entender perfectamente en el futuro.	

Tabla 2.9 - Tarea 9: Pruebas con API REST

T9. Pruebas con API REST	12 días
Descripción:	
Se realizarán una serie de implementaciones con una *API REST* con la finalidad de entender su funcionamiento y los modelos de datos que se utilizan. Después se realizarán pruebas con un cliente básico.	
Recursos:	
Entorno de desarrollo para realizar las pruebas.	
Criterios de aceptación:	
Se dará por finalizada la tarea después de evaluar las implantaciones desarrolladas en la *API REST* y comprobar que se comprende correctamente el uso de *APIs* y se dispone de los conocimientos necesarios sobre su uso.	

Tabla 2.10 - Tarea 10: Requisitos de la aplicación de usuarios víctimas

T10. Requisitos de la aplicación de usuarios víctimas	4 días
Descripción:	
Se estudiarán los requisitos que debe tener la aplicación destinada a las víctimas de acoso y abuso para ofrecer a esas una solución paliativa, segura y anónima. Es importante que la aplicación también sea sencilla de utilizar y agradable para la vista.	
Recursos:	
Información sobre el acoso escolar y la violencia de género y guías de diseño.	
Criterios de aceptación:	

Se aceptaran todos aquellos criterios que, tras analizar exhaustivamente, sean viables y ofrezcan funcionalidad real de la aplicación, así como un factor de diferenciación frente a otras aplicaciones.

Tabla 2.11 - Tarea 11: Desarrollo de la aplicación de usuarios víctimas

T11. Desarrollo de la aplicación de usuarios víctimas	**20 días**
Descripción:	
Consistirá en la fase de desarrollo de la aplicación destinada a las víctimas de acoso y abuso. Se implementarán todos los requisitos establecidos en la fase anterior y se finalizará con una versión estable de la aplicación.	
Recursos:	
Entorno de desarrollo, *iPhone* y *smartphone Android*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya concluido con el desarrollo de la aplicación y se compruebe que se han contemplado todos los requisitos establecidos anteriormente.	

Tabla 2.12 Tarea 12: Requisitos del Back-end

T12. Requisitos del Back-end	**4 días**
Descripción:	
Se estudiarán los requisitos que debe tener el *Back-end* que se encargará de centralizar las comunicaciones entre las aplicaciones de usuario y psicólogo. Es importante realizar una documentación exhaustiva de como se ha de implementar este sistema.	
Recursos:	
Manual de *Node.js*	

Criterios de aceptación:
Se aceptarán todos aquellos criterios que, tras analizar exhaustivamente, sean viables y ofrezcan funcionalidad real al *Back-end*.

Tabla 2.13 Tarea 13: Creación y configuración de un servidor

T13. Creación y configuración de un servidor	3 días
Descripción:	
Consiste en la creación de un servidor que este expuesto de forma remota para poder integral en el *Back-end*. Se realizarán las configuraciones oportunas para garantizar la seguridad.	
Recursos:	
Raspberry Pi 3.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya concluido con la creación y configuración de un servidor a partir de una *Raspberry Pi*. El servidor ha de ser seguro.	

Tabla 2.14 - Tarea 14: Desarrollo del Back-end

T14. Desarrollo del Back-end	20 días
Descripción:	
Consistirá en la fase de desarrollo del *Back-end*. Se creará una aplicación utilizando *Node.js* y las librearías que sean necesarias para garantizar que el *Back-end* sea robusto y seguro.	
Recursos:	
Raspberry Pi 3 y *Node.js*.	

Criterios de aceptación:
Se considera terminada esta fase una vez que se haya concluido con el desarrollo del *Back-end* y se compruebe que se han contemplado todos los requisitos establecidos anteriormente.

Tabla 2.15 - Tarea 15: Creación de la API REST

T15. Creación de la API REST	12 días
Descripción:	
Consistirá en la implementación de una *API REST* utilizando la librearía *express*. Se diseñará un *middleware* que ofrezca seguridad y diferentes *endpoints* para obtener datos desde las aplicaciones del cliente.	
Recursos:	
Entorno de desarrollo.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya concluido con el desarrollo de una *API REST* robusta y segura que garantice la correcta comunicación entre en *Back-end* y las aplicaciones del cliente.	

Tabla 2.16 - Tarea 16: Implementación de un socket para las comunicaciones en tiempo real

T16. Implementación de un socket para las comunicaciones en tiempo real	10 días
Descripción:	
Consistirá en la implementación de un *socket* utilizando la librearía *Socket.IO*. Se diseñará un *middleware* para garantizar la seguridad y se utilizarán diferentes *endpoints* para comunicar en tiempo real la aplicación de los usuarios con la aplicación de los psicólogos.	
Recursos:	

Entorno de desarrollo.
Criterios de aceptación:
Se considera terminada esta fase una vez que se haya concluido con el desarrollo de una comunicación en tiempo real que sea robusta y segura, garantizando la correcta comunicación entre la aplicación de los usuarios con la aplicación de los psicólogos.

Tabla 2.17 - Tarea 17: Búsqueda de un conjunto de datos para entrenar un modelo predictivo

T17. Búsqueda de un conjunto de datos para entrenar un modelo de Machine Learning	**5 días**
Descripción:	
Consistirá en buscar un conjunto de datos constituido por oraciones en los que se clasifique la polaridad de cada una de estas. Se analizarán varias fuentes de datos en busca de un conjunto que sea válido.	
Recursos:	
Datasets de Internet.	
Criterios de aceptación:	
Los datos deben ser suficientes y tener un alto grado de calidad para asegurar el funcionamiento de los algoritmos predictivos que se realizarán posteriormente. Se valorará el tamaño del conjunto de datos y la calidad de este.	

Tabla 2.18 - Trea 18: Limpieza del conjunto de datos seleccionado

T18. Limpieza del conjunto de datos seleccionado	6 días
Descripción:	
Se realizarán una serie de *scripts* en *Python* para cargar el conjunto de datos seleccionado y aplicar diversas tareas de limpieza sobre este. Por ejemplo se han de eliminar datos duplicados y asegurar que el conjunto esté correctamente balanceado.	
Recursos:	
Dataset, *Python*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya concluido con la limpieza del conjunto de datos y se haya comprobado que el conjunto es apto para entrenar modelos predictivos.	

Tabla 2.19 - Tarea 19: Análisis de los diferentes algoritmos de Machine Learning

T19. Análisis de los diferentes algoritmos de Machine Learning	5 días
Descripción:	
Se realizará un estudio de los diferentes algoritmos de *Machine Learning* que existen y sus ventajas frente a otro y se harán pequeñas implementaciones para probar cada uno de los modelos.	
Recursos:	
Dataset, *Python*.	
Criterios de aceptación:	

Se considera terminada esta fase una vez que se hayan encontrado los mejores algoritmos predictivos para el conjunto de datos seleccionado anteriormente. Se establecerán métricas que permitan medir la precisión de cada uno de los modelos predictivos.

Tabla 2.20 - Trea 20: Entrenamiento de diferentes modelos predictivos

T20. Entrenamiento de diferentes modelos predictivos	**5 días**
Descripción:	
Se entrenará cada uno de los modelos seleccionados en la fase anterior con el conjunto de datos y se crearan distintos modelos predictivos que establezcan la polaridad de una oración.	
Recursos:	
Dataset, *Python*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se hayan generado los modelos predictivos para cada uno de los algoritmos seleccionados en la fase anterior. Se calculará la polaridad de varias oraciones para comprobar que todos ellos funcionan bien.	

Tabla 2.21 - Tarea 21: Pruebas de validación cruzada

T21. Pruebas de validación cruzada	**4 días**
Descripción:	
Se realizarán pruebas de validación cruzada para evaluar los resultados de los modelos predictivos y garantizar que los resultados estadísticos son independientes de la partición entre los datos de entrenamiento y los prueba.	
Recursos:	

Dataset, *Python*.
Criterios de aceptación:
Se considera terminada esta fase una vez que se hayan realizado todas las pruebas de validación cruzada en cada uno de los diferentes modelos predictivos que hayan sido seleccionados.

Tabla 2.22 - Tarea 22: Selección del mejor modelo predictivo

T22. Selección del mejor modelo predictivo	**2 días**
Descripción:	
Con las pruebas de validación cruzada realizadas anteriormente se recopilarán resultados de diferentes métricas para elegir el modelo predictivo que mejor resultado ofrezca.	
Recursos:	
Dataset, *Python*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya seleccionado un único modelo predictivo que proporcione el mayor grado de acierto frente al resto de modelos.	

Tabla 2.23 - Tarea 23: Integración del modelo predictivo

T23. Integración del modelo predictivo	**4 días**
Descripción:	
Esta tarea consistirá en la integración del modelo predictivo en el *Back-end* de forma que se aplique una predicción de la polaridad en los mensajes que envían los usuarios víctimas para reflejar en la aplicación del psicólogo su polaridad y realizar otras operaciones de interés posteriormente.	

Recursos:
Back-end, modelo predictivo.
Criterios de aceptación:
Se considera terminada esta fase una vez que se haya integrado perfectamente el modelo predictivo en el *Back-end* y se compruebe que calcula la polaridad de los mensajes correctamente.

Tabla 2.24 - Tarea 24: Evaluación de la aplicación destinada a usuarios que sufren acoso o abuso.

T24. Evaluación de la aplicación destinada a usuarios que sufren acoso o abuso.	**3 días**
Descripción:	
Esta tarea consistirá en la evaluación de la aplicación destinada a usuarios víctimas para comprobar el correcto funcionamiento. Se aplicarán tests unitarios para comprobar que cada uno de los módulos desarrollados funciona correctamente según lo esperado.	
Recursos:	
Entorno de pruebas, *iPhone*, *Smartphone Android*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya evaluado la totalidad del software desarrollado y se haya detectado cada error y documentado correctamente para solucionarlo posteriormente.	

Tabla 2.25 - Tarea 25: Corrección de errores en la aplicación.

T25. Corrección de errores en la aplicación destinada a usuarios que sufren acoso o abuso.	3 días
Descripción:	
Esta tarea consistirá en la corrección de cada uno de los errores detectados en la fase anterior en la aplicación destinada a los usuarios víctimas.	
Recursos:	
Entorno de pruebas, *iPhone*, *smartphone Android*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se hayan corregido todos los errores detectados en la totalidad del software desarrollado y se haya garantizado su correcto funcionamiento	

Tabla 2.26 - Tarea 26: Evaluación mediante tests unitarios del Back-end

T26. Evaluación mediante tests unitarios del Back-end	3 días
Descripción:	
Esta tarea consistirá en la evaluación del *Back-end* para comprobar el correcto funcionamiento. Se aplicarán tests unitarios para comprobar que cada uno de los módulos desarrollados funciona correctamente según lo esperado.	
Recursos:	
Entorno de pruebas.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya evaluado la totalidad del software y se haya detectado cada error y documentado correctamente para solucionarlo posteriormente.	

Tabla 2.27 - Tarea 27: Corrección de errores en el Back-end

T27. Corrección de errores en el Back-end	3 días
Descripción:	
Esta tarea consistirá en la corrección de cada uno de los errores detectados en la fase anterior en el *Back-end*.	
Recursos:	
Entorno de pruebas.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se hayan corregido todos los errores detectados en la totalidad del software desarrollado y se haya garantizado su correcto funcionamiento.	

Tabla 2.28 - Tarea 28: Despliegue del software

T28. Despliegue del software	2 días
Descripción:	
En esta fase se desplegará todo el software desarrollado para su puesta en producción.	
Recursos:	
Raspberry Pi, *iPhone*, *smartphone Android*.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya puesto en producción todo el software desarrollado anteriormente y se compruebe que este funciona según lo esperado.	

Tabla 2.29 - Tarea 29: Manual de uso de la aplicación de usuarios víctimas

T29. Manual de uso de la aplicación de usuarios víctimas	12 días
Descripción:	
Se realizará un manual de uso de la aplicación destinada a los usuarios que son víctimas de acoso y de abuso donde se explique de forma detallada como utilizar la aplicación y que ventajas ofrece esta.	
Recursos:	
Software desarrollado.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya concluido el manual donde se recoja como utilizar la aplicación, las características de esta, y que ventajas ofrece frente a otras aplicaciones.	

Tabla 2.30 - Tarea 30: Métricas y sistemas de evaluación futura

T30. Métricas y sistemas de evaluación futura	6 días
Descripción:	
Definición de posibles métricas de parámetros para mejorar la mantenibilidad del software sabiendo en que puntos podría mejorar y la definición de las medidas que se tomaran para afrontar las dificultades.	
Recursos:	
Software desarrollado.	
Criterios de aceptación:	
Se considera terminada esta fase una vez que se haya documentado dicha información, incluyendo las posibles mejoras o formas de afrontar las dificultades.	

La estimación inicial es que el proyecto requiere entorno a unos 211 días de trabajo para ser realizado completamente. Se comenzó el proyecto en la fecha 01/11/2018 y se concluyó en 29/06/2019. Seguidamente en la tabla 2.31 se muestran las tareas con sus respectivas fechas de inicio y finalización, y en la figura 2.1 se muestra el diagrama de Gantt con la planificación inicial del proyecto.

Tabla 2.31 - Planificación inicial

Actividad	Duración (días)	Fecha de inicio	Fecha de fin
T1	5	01/11/2018	06/11/2018
T2	8	07/11/2018	15/11/2018
T3	6	16/11/2018	22/11/2018
T4	2	23/11/2018	25/11/2018
T5	10	26/11/2018	06/12/2018
T6	15	07/12/2018	22/12/2018
T7	8	23/12/2018	31/12/2018
T8	8	01/01/2019	09/01/2019
T9	12	10/01/2019	22/01/2019
T10	4	23/01/2019	27/01/2019
T11	20	28/01/2019	17/02/2019
T12	4	18/02/2019	22/02/2019
T13	3	23/02/2019	26/02/2019
T14	20	27/02/2019	19/03/2019
T15	12	20/03/2019	01/04/2019
T16	10	02/04/2019	12/04/2019
T17	5	13/04/2019	18/04/2019
T18	6	19/04/2019	25/04/2019
T19	5	26/04/2019	01/05/2019

T20	5	02/05/2019	07/05/2019
T21	4	08/05/2019	12/05/2019
T22	3	13/05/2019	16/05/2019
T23	4	17/05/2019	21/05/2019
T24	3	22/05/2019	25/05/2019
T25	3	26/05/2019	29/05/2019
T26	3	30/05/2019	02/06/2019
T27	3	03/06/2019	06/06/2019
T28	2	07/06/2019	09/06/2019
T29	12	10/06/2019	22/06/2019
T30	6	23/06/2019	29/06/2019
Total	**211**	**01/11/2018**	**29/06/2019**

2.2 Planificación final

Al concluir el proyecto la planificación que se estableció inicialmente no se pudo cumplir de forma rigurosa ya que en algunos casos se necesitó más tiempo y en otros sobraba tiempo realmente. Pero en la consecución del proyecto la planificación fue muy bien estimada ya que el tiempo total se aproximó al tiempo estimado inicialmente.

Como conclusión, la planificación inicial en conjunto se ha realizado correctamente y se ha ajustado a la realidad, porque la planificación final solamente ha aumentado 6 días el tiempo estimado para la finalización del proyecto. El tiempo que ha trascurrido desde el inicio hasta el final del proyecto es de 217 días.

A continuación, una tabla con los desfases en la planificación.

Tabla 2.32 - Desfase en la planificación

Actividad	Duración inicial (días)	Desfase (días)	Duración final (días)
T1	5	0	5
T2	8	-2	6
T3	6	0	6
T4	2	1	3
T5	10	0	10
T6	15	-4	11
T7	8	0	8
T8	8	2	10
T9	12	0	12
T10	4	0	4
T11	20	-1	19
T12	4	0	4
T13	3	2	5
T14	20	0	20
T15	12	-1	11
T16	10	0	10
T17	5	0	5
T18	6	3	9

T19	5	3	8
T20	5	0	5
T21	4	0	4
T22	3	-2	1
T23	4	0	4
T24	3	0	3
T25	3	1	4
T26	3	0	3
T27	3	-1	2
T28	2	0	2
T29	12	5	17
T30	6	0	6
Total	**211**	**6**	**217**

2.3 Diagramas de Gantt para la planificación

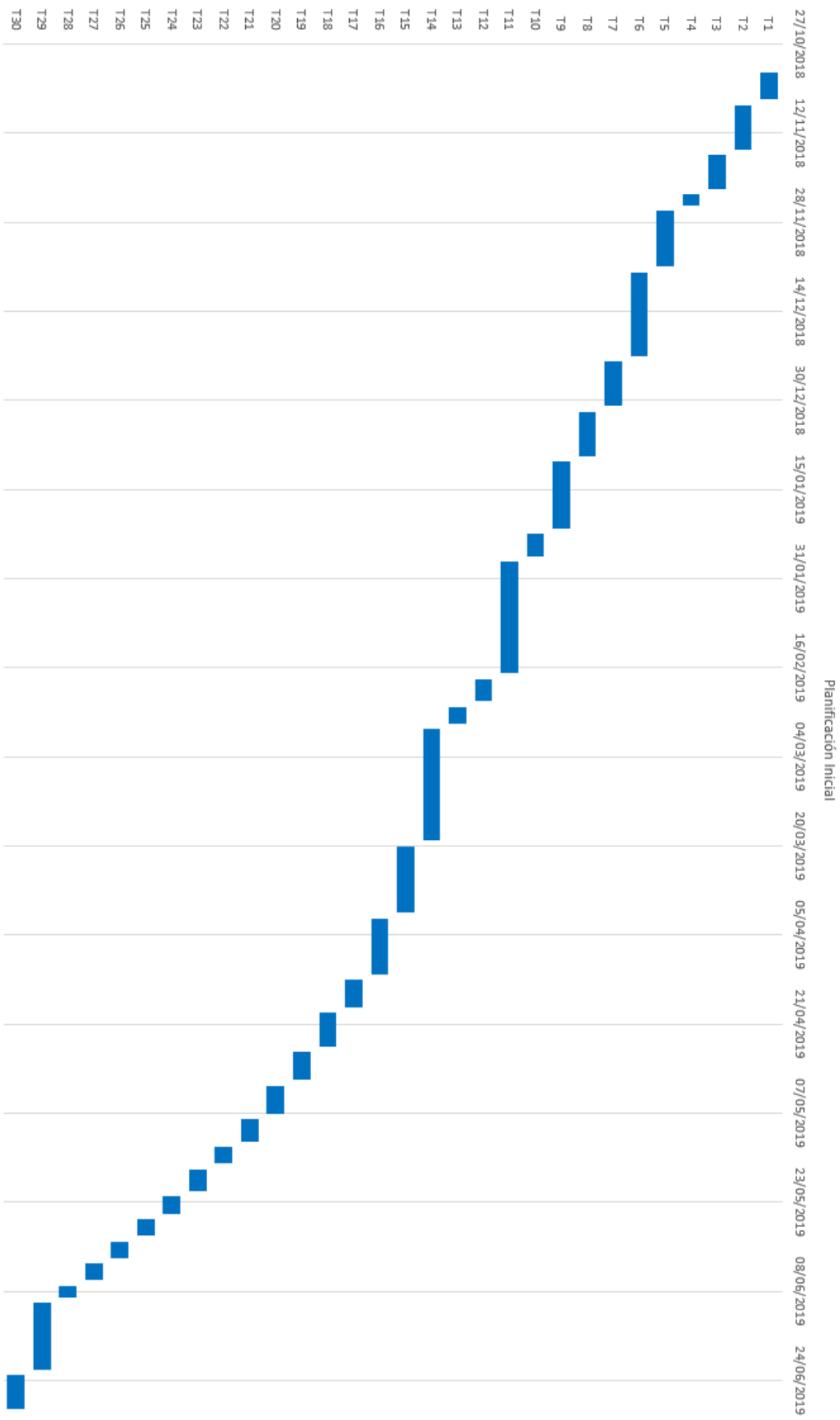


Figura 2.1 - Diagrama de Gantt para la planificación inicial

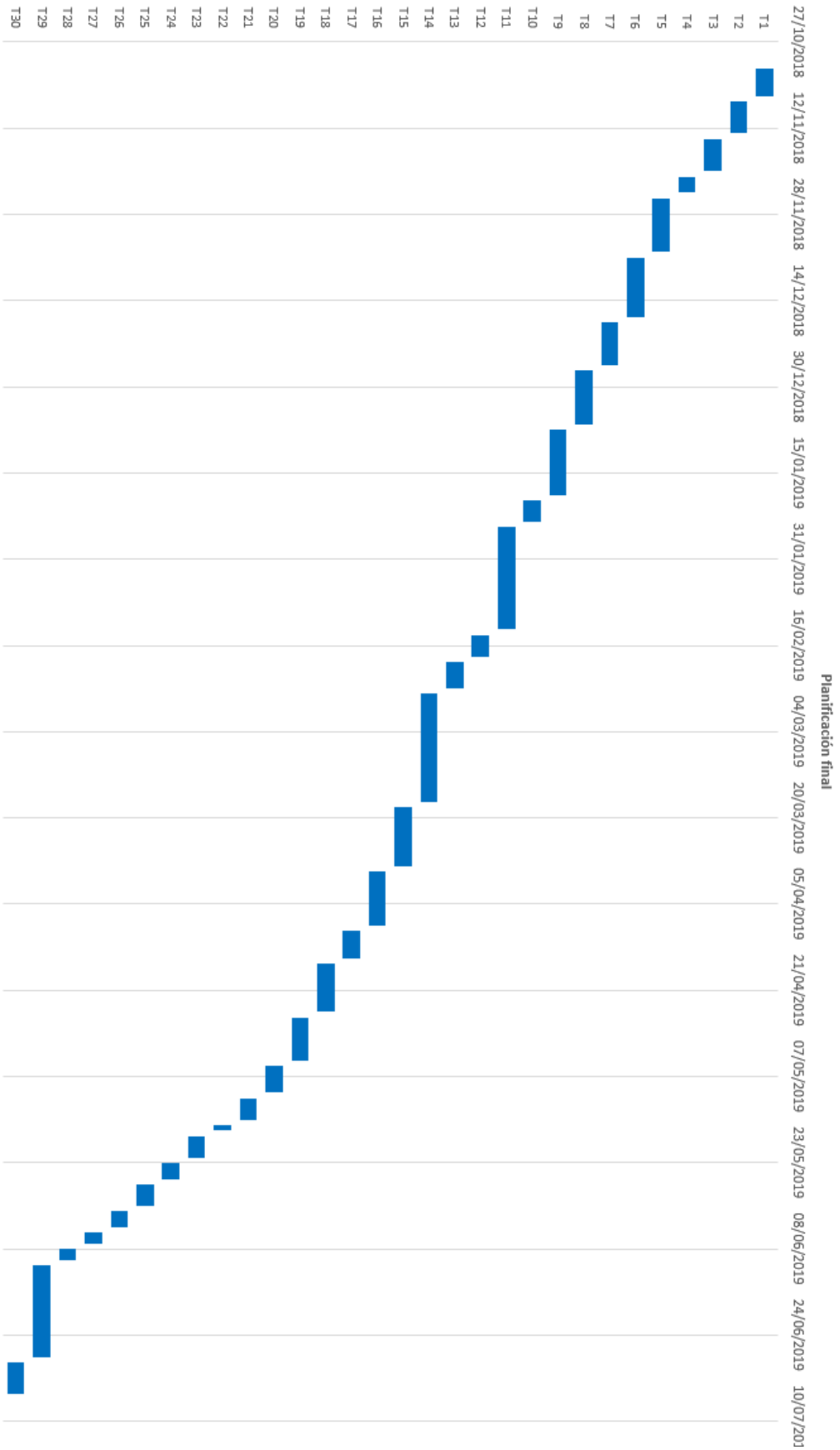


Figura 2.2 - Diagrama de Gantt para la planificación final

3. Requisitos

El objetivo principal del proyecto es desarrollar un canal de comunicación entre las víctimas que sufren acoso escolar o violencia de género y los psicólogos especializados en estos problemas que puedan brindar ayuda a estas personas. Para lograr este objetivo son necesarios una serie de requisitos, que van desde las necesidades de los posibles usuarios hasta las especificaciones que debe cumplir el software a desarrollar para satisfacer dichas necesidades.

Los requisitos que debe de tener el sistema se dividen principalmente en dos categorías:

3.1 Requisitos funcionales

El software desarrollado tiene que cumplir una serie de especificaciones en forma de requisitos para poder cumplir los objetivos necesarios. Estos requisitos representan las funcionalidades que ha de tener el sistema para que cubra las necesidades de todos los posibles clientes y les ofrezca la mejor experiencia de uso posible. En la tabla 3.1 se muestran los requisitos funcionales:

Tabla 3.1 - Requisitos funcionales

Numero	Requerimiento	Prioridad
RF1	Inicio de sesión seguro basado en *JSON Web Token* y un *middleware*.	5
RF2	Función de auto camuflaje para ocultar la funcionalidad real de la aplicación.	5
RF3	Posibilidad de que usuario pueda borrar todos sus datos si así lo desea.	3
RF4	Posibilidad de elegir a los psicólogo con los que se desea hablar.	4
RF5	Valoración de los psicólogos por parte de los usuarios.	3

RF6	Visualización de la valoración media de cada psicólogo.	3
RF7	Botón del pánico para enviar a la policía la ubicación de la persona si esta está en peligro.	5
RF8	Posibilidad de modificar los datos de usuario si este lo desea.	2
RF9	Eliminar chats de forma individual cuando el usuario lo desea.	2
RF10	Posibilidad de buscar un fragmento de texto en un chat.	1
RF11	Diseño material y minimalista que facilite la usabilidad.	4
RF12	Adaptación de la aplicación al tamaño de cualquier pantalla de móvil.	5
RF13	Botones de ayuda en determinadas secciones para mostrar diálogos explicativos.	4
RF14	Visualización de la última conexión del psicólogo, indicando si está en línea en ese momento.	5
RF15	Mostar la fecha y la hora en la que se envió cada mensaje.	4
RF16	Sistema de notificaciones para mostrar el número de chats con mensajes sin leer y el número de mensajes sin leer de un chat concreto.	4

RF17	Ofrecer fluidez con la implementación de un *scroll* infinito para cargar los mensajes a medida que el usuario se desplace por el chat.	3
RF18	Posibilidad de que el usuario pueda eliminar su cuenta de *Connect*, así como sus datos relacionados con esta.	4
RF19	Mostrar una lista de selección de psicólogos basada en el problema concreto que sufre el usuario.	4
RF20	Comunicaciones en tiempo real mediante un *socket*.	5
RF21	Informe de eventos del chat y del usuario en tiempo real. Algunos ejemplos son: conexión, desconexión, nuevo chat, chat eliminado.	2
RF22	*Middleware* en la *API REST* para ofrecer seguridad en las comunicaciones.	5
RF23	*Middleware* en el *socket* para ofrecer seguridad en las comunicaciones en tiempo real.	5
RF24	Base de datos no relacional, basada en documentos y que sea flexible.	5
RF25	Uso de un *logger* para registrar todos los eventos, incluidos los errores, que ocurren en el *Back-end*.	2
RF26	Uso de *JSON Web Token* para garantizar la integridad de las comunicaciones.	4

RF27	Identificar los mensajes negativos del usuario víctima, con un algoritmo de *Machine Learning*, para mostrárselos en rojo al psicólogo.	4
RF28	Utilizar un algoritmo predictivo basado análisis de sentimientos para ver como evoluciona un usuario durante el tiempo que usa la aplicación.	4

3.2 Requisitos no funcionales

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

A continuación, se muestran en la tabla 3.2 los requisitos no funcionales más relevantes para el proyecto:

Tabla 3.2 - Requisitos no funcionales

Numero	Requerimiento	Prioridad
RNF1	Desempeño	4
RNF2	Confiabilidad	4
RNF3	Usabilidad	5
RNF4	Adaptabilidad	5
RNF5	Rendimiento	4
RNF6	Seguridad	5
RNF7	Escalabilidad	4
RNF8	Usabilidad	4
RNF9	Eficiencia	4
RNF10	Facilidad de mantenimiento	3

RNF11	Portabilidad	4
RNF12	Estabilidad	4
RNF13	Disponibilidad	5
RNF14	Interfaz	3
RNF15	Coste	3
RNF16	Accesibilidad	3
RNF17	Facilidad de uso	4

4. Objetivos

4.1 Objetivos generales

Una vez se han sido identificados y definidos los requisitos, se procede a identificar los objetivos del proyecto de forma general.

El proyecto consistirá en la creación de un canal de comunicación rápido, seguro y anónimo que sirva como una solución ante la creciente necesidad que tienen las personas que sufren algún tipo de acoso o de abuso, de ponerse en contacto con psicólogos de forma anónima para solventar esta necesidad.

Se desarrollará una aplicación móvil que permita a los diferentes usuarios establecer una comunicación con psicólogos, que cubran los problemas específicos de cada usuario, centrándose en dos puntos claves: el anonimato y la posibilidad de ofrecer a estas personas una alternativa viable que les ayude a sobrellevar su problema permitiéndoles tener sesiones de forma inmediata y en cualquier momento del día.

El proyecto se puede dividir en tres partes que son complementarias:

1. En primer lugar, se creará una aplicación móvil para dispositivos *iOS* y *Android*. Esta aplicación se desarrollará mediante tecnologías web. Esta aplicación está destinada a las víctimas y se utilizará para que estos usuarios se pongan en contacto con psicólogos de forma anónima. La aplicación ofrecerá una comunicación en tiempo real.
2. Por otro lado, se creará un *Back-end* que será el encargado de establecer la comunicación entre las aplicaciones de los usuarios víctimas y de los psicólogos. Esté *Back-end* ofrecerá una *API REST* para realizar diversas tareas e integrará un *socket* para realizar las comunicaciones en tiempo real. También integrará una base de datos *MongoDB* que se utilizará para almacenar todos los datos de los usuarios.
3. Por último, una de las funcionalidades más importantes de *Connect* es el uso de la inteligencia artificial para mejorar el servicio y ofrecer una clasificación de los mensajes de las víctimas. Para ello será necesario crear una herramienta en *Python* que genere modelos predictivos de análisis de sentimientos con un conjunto de datos, y posteriormente, realizar predicciones de polaridad sobre los mensajes de las víctimas.

4.2 Objetivos técnicos

Posteriormente a la definición de los objetivos generales del proyecto y la especificación de los requisitos, tanto funcionales como los no funcionales, es el momento de entrar a explicar los objetivos desde el punto de vista técnico, describiendo como deberán implementarse.

Los objetivos principales que se han establecido durante esta fase se explican detalladamente a continuación:

4.2.1 Desarrollo del Front-end

Se creará una aplicación móvil con *Angular7* que posteriormente será exportada con *Cordova* para crear una aplicación *iOS* y otra *Android*.

Angular es un *framework*, gratuito y *open source*, destinado a facilitar la creación de aplicaciones web modernas y por componentes. *Angular* utiliza como lenguaje de programación principal *TypeScript*, un superconjunto de *JavaScript* que facilita mucho el desarrollo.[39]

Apache Cordova es un marco de desarrollo móvil de código abierto. Permite utilizar tecnologías web como *HTML5*, *CSS3* y *JavaScript* para un desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo de cada plataforma móvil por separado. Las aplicaciones se ejecutan dentro de una envoltura para cada plataforma y dependen de un estándar *API* para acceder a cada uno de los sensores del dispositivo móvil, permitiendo también el acceso a otros datos y al estado de la red.[40]

4.2.1.1 Autenticación

La aplicación deberá utilizar un inicio de sesión seguro. Se implementará *JSON Web Token* que consiste en un estándar abierto basado en *JSON* para crear un *token* que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros. El caso más común de uso de los *JWT* es para manejar la autenticación en aplicaciones móviles o web. Cuando el usuario se quiere autenticar manda sus datos de inicio de la sesión al servidor, este genera el *JWT* y se lo manda a la aplicación cliente, luego en cada petición el cliente envía este *token* que el servidor usa para verificar que el usuario este correctamente autenticado y saber quién es. Otro uso de este estándar consiste en transferir datos entre servicios de nuestra aplicación y asegurarnos de que no se comprometa su integridad.[41]

Cada vez que se acceda a la aplicación se comprobará si el *token JWT* es válido. Válido significa que es posible desencriptarlo con una clave conocida y que su fecha de validez no ha expirado. En el caso de que alguna de estas dos condiciones no se cumpla se deberá redirigir al usuario a la página de inicio de sesión para loguearse de nuevo.

Si el usuario es nuevo, y no tiene una cuenta, se le ofrecerá la posibilidad de registrarse introduciendo una serie de datos. Estos datos son:

- Email
- Nombre de usuario
- Contraseña
- Tipo de problema

El tipo de problema hace referencia a uso que va a hacer el usuario de la aplicación. Existirán dos tipos de problema que son el acoso escolar y la violencia de género. Los psicólogos a los que tiene acceso el usuario se mostrarán en base al problema que haya seleccionado.

Por otro lado, el usuario podrá borrar su cuenta si así lo desea. Si elige esta opción se deberán eliminar todos los datos relacionados con el usuario como son:

- Información de perfil
- Notas de los psicólogos
- Conversaciones
- Ubicaciones

4.2.1.2 Comunicación en tiempo real

Será necesaria la comunicación en tiempo real con los psicólogos de la forma más segura y eficiente posible, y para ello se utilizarán *websockets*. Los *websockets* son una tecnología que permite una comunicación bidireccional entre cliente y servidor sobre un único *socket* TCP. En cierta manera es un buen sustituto de *AJAX* como tecnología para obtener datos del servidor, ya que no tenemos que pedirlos, el servidor nos los enviará cuando haya nuevos.[42]

Para usar *websockets* se utilizará *Socket.IO*, que es una herramienta que nos permite crear comunicaciones en tiempo real entre cliente y servidor. *Socket.IO* es una librería que nos permite controlar eventos en tiempo real a través de conexiones *TCP* y nos ayuda a evitar problemas de compatibilidad entre equipos. Está desarrollado completamente en

JavaScript y, su objetivo es hacer que las aplicaciones en tiempo real tengan posibilidad de ejecutarse en cualquier navegador, incluidos los dispositivos móviles, salvando las diferencias entre los diferentes protocolos.

En primer lugar, se debe de implementar la escucha de las tres señales más importantes para el funcionamiento de *Socket.IO* que son:

- *connect*: Se encarga de informar de que se ha producido la conexión con el *Back-end* a través de *websockets*.
- *disconnect*: Informa a la aplicación de que se ha producido una desconexión con el *Back-end*.
- *error*: Esta señal indica que se ha producido un error durante alguna de las transmisiones a través de *websockets*.

Una vez controladas estas tres señales, será necesario implementar los dos métodos básicos que se deben utilizar para realizar cualquier transmisión de datos a través de *webservices*. Estos métodos son:

- *emit*: Esta función se utiliza para enviar datos al servidor. Se debe especificar el nombre del evento y los datos.
- *listen*: Esta función se utiliza para escuchar un evento del servidor. Se debe especificar el nombre del evento que se desea escuchar.

4.2.1.3 Módulos

Un módulo es uno de los elementos principales con los que se puede organizar el código de las aplicaciones en *Angular*. Lo adecuado es desarrollar diferentes módulos y agrupar distintos elementos en unos u otros. El orden se realizará de una manera lógica, atendiendo a nuestras propias preferencias, el modelo de negocio o las preferencias del equipo de desarrollo.[43]

La aplicación se ha desarrollado de forma modular para facilitar el desarrollo y ofrecer un alto grado de escalabilidad futura. Los principales módulos de la aplicación son los siguientes:

1. **Chats**: Este módulo se encargará de gestionar toda la funcionalidad que deben tener los chats. El usuario tendrá acceso una lista de chats y podrá acceder a cada uno de forma individual. Se ofrecerán una serie de acciones:
    a. Obtención de chats: Se obtendrá la lista de chats en los que el usuario participa y se mostrará cada uno en una fila.
    b. Obtención de un chat: Se obtendrá un chat de forma individual. En el chat se concatenará la información del otro usuario.
    c. Eliminar chat: El usuario debe tener la posibilidad de eliminar un chat si lo desea.
    d. Buscar en un chat: Esta función permitirá buscar una determinada cadena de texto en un chat.
    e. Eliminar un chat: Esta función permitirá la opción de eliminar un chat de forma individual.
    f. Eliminar todos los chats: Se deberán eliminarán todos los chats si el usuario lo sea. En cualquier momento el usuario puede elegir esta opción.
2. **Usuarios**: Este módulo se encargará de gestionar toda la funcionalidad que está asociada a los usuarios. Se ofrecerán una serie de acciones:
    a. Obtención de un psicólogo: Se solicitará la información de un psicólogo. Esta información consta de: nombre, última conexión y *rating*.
    b. Obtención de psicólogos disponibles: Proporcionará una lista de los psicólogos que hay disponibles en ese momento. Más tarde el usuario escogerá al psicólogo con el que sea hablar.
    c. Evaluar a un psicólogo: Esta función consistirá en establecer una nota numérica al psicólogo, con el que se está chateando, para expresar el grado de conformidad.
    d. Actualizar información de usuario: El usuario debe tener la posibilidad de actualizar su información de registro en cualquier momento.

3. **Mensajes**: Este módulo se encargará de gestionar toda la funcionalidad que deben tener los mensajes asociados a un chat. El usuario tendrá acceso a los mensajes de un chat en el que participe y también podrá enviar nuevos mensajes al psicólogo que participa en ese chat. Se ofrecerán una serie de acciones:
    a. Obtención de mensajes de un chat: Se proporcionará al usuario la lista de mensajes del chat que haya seleccionado. Los mensajes incluirán información de la fecha de envió y de si se han leído o no.
    b. Enviar mensaje: Esta función permite al usuario enviar un nuevo mensaje al psicólogo con el que se comunica.
    c. Lectura de mensajes: Cuando el usuario acceda a un chat se deberán de marcar los mensajes que se muestren como leídos.

4.2.1.4 Ocultación

Una de las funcionalidades más importantes de la aplicación será la ocultación de esta, mostrando una aplicación sencilla que no esté relacionada. Esta función consistirá en que, una vez el usuario acceda a la aplicación, se debe de mostrar una calculadora, de esta forma no se muestra la aplicación real y se protege al usuario frente a un tercero que pueda hacerse con el *smartphone*.

Esta funcionalidad debe ser opcional y se activará desde el menú de configuración de la aplicación una vez se acceda a esta.

Si se activa la ocultación, se debe solicitar un código secreto que sea numérico. Este código se introducirá en la calculadora para mostrar la aplicación real.

Cada vez que el usuario salga de la aplicación y vuelva a acceder a esta se debe activar la ocultación a través de la calculadora.

4.2.1.5 Notificaciones

Otra funcionalidad que debe incorporar la aplicación es el uso de notificaciones para informar al usuario en todo momento de los eventos que se producen durante el uso de la aplicación.

Los eventos que se han de notificar son los siguientes:

1. Nuevo mensaje
2. Conexión de un psicólogo
3. Desconexión de un psicólogo
4. Solicitud aceptada para hablar con un psicólogo
5. Solicitud denegada para hablar con un psicólogo
6. Chat eliminado para un psicólogo

Las notificaciones se mostrarán al usuario a través de la interfaz móvil de la forma más apropiada.

4.2.1.6 Diseño material y minimalista

A la hora de desarrollar una aplicación, uno de los objetivos más importantes es el papel que juega el diseño de la interfaz que va a utilizar el usuario. Este diseño ha de ser lo más usable posible, y a su vez, permitir al usuario acceder a todas las funcionalidades de la aplicación de la forma más sencilla e intuitiva que se pueda.

Material Design es una normativa de diseño enfocado en la visualización del sistema operativo *Android* e *iOS*, las plataformas web y otras plataformas. Según Google la definición es la siguiente: "*Material Design* es un lenguaje visual que sintetiza los principios clásicos del buen diseño con la innovación de la tecnología y la ciencia". Una de sus principales características es la experiencia de usuario y animación de elementos. Cuando se mueven y se transforman los objetos, sus transiciones deben ser suaves, lo que asegura la continuidad de la experiencia. El movimiento es una consideración imprescindible en *Material Design* lo que inspira a animar iconos y pequeñas interacciones que hacen la navegación más agradable.[44]

Por otro lado, el término minimalista, se refiere a todo aquello que se haya despojado de los excesos, dejando sólo lo esencial. El diseño minimalista, es una muestra de esto, un diseño en su forma más básica. Se distingue por sus elementos ligeros, colores, formas y texturas. Su lema "Menos es más", hace referencia a los ambientes que contienen pocos elementos decorativos, además de sus componentes principales: la comodidad y funcionalidad. Se utilizan colores puros, con superficies o fondos monocromáticos, de tonos suaves.[45]

4.2.2 Desarrollo del Back-end

Se implementará un *Back-end* con el fin de crear un sistema de comunicación entre las aplicaciones de usuario y de psicólogo de forma efectiva. La seguridad y la eficiencia deben ser dos puntos importantes a tener en cuenta durante el desarrollo.

El *Back-end* se desarrollará con *Node.js*, un entorno de ejecución para *JavaScript* construido con el motor de *JavaScript* V8 de Chrome.

El *Back-end* alojará una *API REST* encarda de ofrecer una interfaz de programación de aplicaciones que se apoya en la arquitectura *REST* para el desarrollo de aplicaciones en red. Para ello se utilizará *Express*, un *framework* para *Node.js* que sirve para crear aplicaciones web de forma minimalista y flexible en poco tiempo, ya que proporciona funcionalidades como enrutamiento, opciones para gestionar sesiones y cookies. Se pueden realizar todo tipo de consultas mediante estas cuatro operaciones: *GET*, *POST*, *PUT* y *DELETE*.[46]

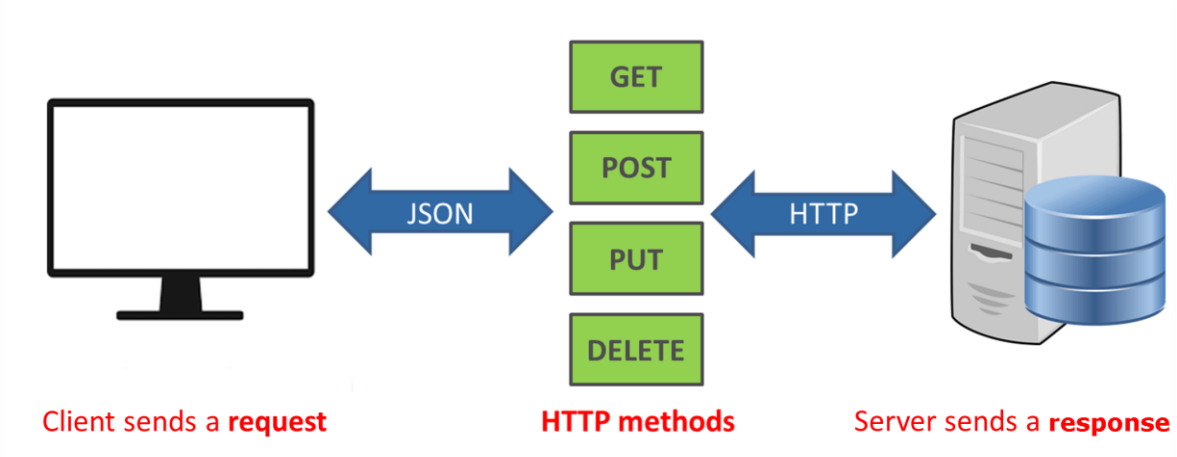


Figura 4.1 - Diagrama de una API REST

Además de la *API REST*, el *Back-end* tiene implementado un *socket* para realizar comunicaciones rápidas y en tiempo real entre un usuario y un psicólogo. Se ha tenido en cuenta la seguridad y se ha implementado un *middleware* para garantizar la comunicación de extremo a extremo.

Para almacenar todos los datos se utilizará *MongoDB*, una base de datos *NoSQL* orientada a documentos. Esto significa que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en *BSON*, que es una representación binaria de *JSON*.

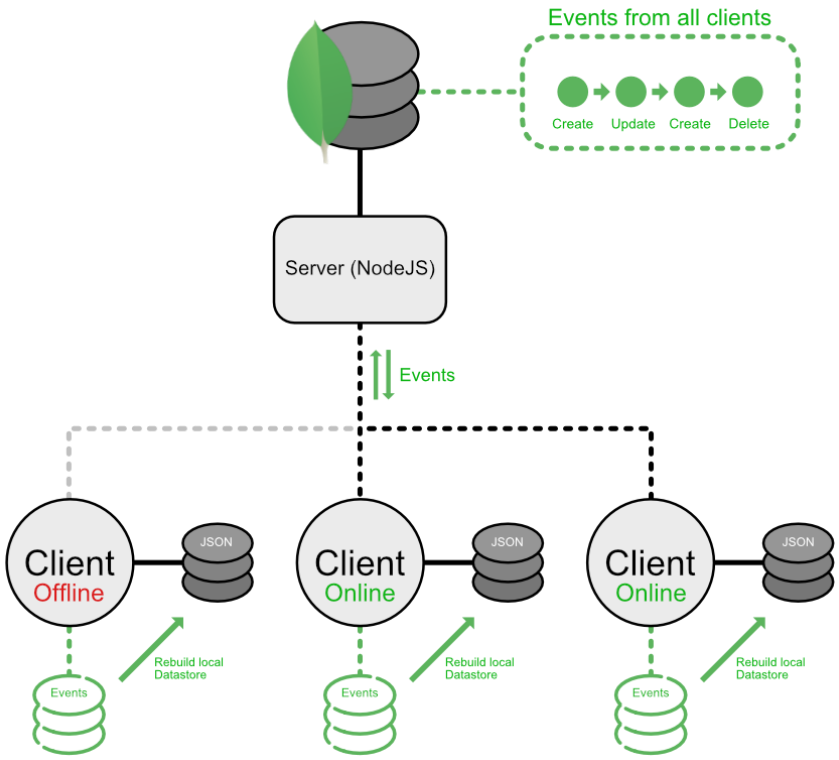


Figura 4.2 - Diagrama de una base de datos MongoDB

4.2.2.1 Autenticación

El *Back-end* implementará un método se autenticación segura. Para ello se desarrollará un *middleware* que se encargará de bloquear las peticiones de clientes que no están autorizados. Este *middleware* debe comprobar que el *token* de usuario es válido y que no ha expirado. El *middleware* es una funcionalidad incluida en *Express* que captura el tráfico antes de llegar a los controladores y decide sí debe bloquearlo o no.

Un *middleware* puede realizar las siguientes tareas:[21]

- Ejecutar cualquier código.
- Realizar cambios en la solicitud y los objetos de respuesta.
- Finalizar el ciclo de solicitudes y respuestas.
- Invocar la siguiente función de *middleware* en la pila.

Cuando un usuario inicie sesión se generará un *token JWT* en el *Back-end* con una clave conocida y se enviará este a la aplicación del cliente. Este *token* es el que el cliente debe enviar en cada petición que realice contra el *Back-end*.

4.2.2.2 API REST

Se desarrollará una *API REST* para ofrecer a las aplicaciones clientes un servicio a través del cual puede obtener todos los datos necesarios para el correcto funcionamiento de la aplicación. Se implementarán una serie de rutas y controladores para obtener y modelar todos los datos.

La *API* se ha desarrollado de forma modular, para facilitar el desarrollo y ofrecer un alto grado de escalabilidad futura. Las rutas de la *API REST* se pueden dividir en 4 grupos, que son los siguientes:

- Autenticación: Hace referencia a los *endpoints* encargados de permitir el acceso a los usuarios clientes mediante sus cuentas de usuario. Se deben implementar los siguientes métodos:
    - Registro de nuevo usuario.
    - Inicio de sesión.
    - Comprobación del *token* de usuario.
    - Eliminar cuenta de usuario.
- Chats: Engloba todos los *endpoints* que se encargan de ofrecer a los clientes todos los datos relacionados con los chats e implementar las operaciones que se pueden realizar en estos. Debe tener los siguientes métodos:
    - Obtención un chat concreto.
    - Obtención de la lista chats en los que participa un usuario.
    - Creación un chat.
    - Actualizar estado del chat.
    - Eliminar todos los chats de un usuario.
    - Eliminar un chat concreto.

- Mensajes: Hace referencia a todos los *endpoints* que se encargan de ofrecer a los clientes los mensajes de un determinado chat e implementar las operaciones que se pueden realizar en estos. Debe tener los siguientes métodos:
    - Obtención de mensajes de un chat.
    - Envío de mensajes.
    - Indicar que se ha leído un determinado mensaje.
- Usuarios: Engloba todos los *endpoints* que se encargan de ofrecer a los clientes todos los datos relacionados con los usuarios que desempeñan el papel de psicólogos e implementar las operaciones que se pueden realizar en estos. Debe tener los siguientes métodos:
    - Obtención de la lista de psicólogo disponible.
    - Obtener de los datos de un psicólogo en concreto.
    - Actualización de los datos del propio usuario.
    - Añadir valoración a un psicólogo.

4.2.2.3 Comunicación en tiempo real

Uno de los objetivos más importantes es implementar un sistema de comunicación en tiempo real que permita comunicar a los usuarios de la aplicación móvil con los psicólogos. Para cumplir este objetivo será necesario utilizar *Socket.IO*, una librería de *JavaScript* para *Node.js* que permite una comunicación bidireccional en tiempo real entre cliente y servidor.

Este objetivo consistirá en la implementación de un *socket* mediante la librearía *Socket.IO*. Se diseñará un *middleware* para garantizar la seguridad y se utilizarán diferentes *endpoints* para comunicar en tiempo real la aplicación de los usuarios con la aplicación de los psicólogos.

Una vez un usuario acceda a la aplicación móvil se establecerá una conexión con el *Back-end* a través de un *socket*. En el *Back-end* se debe de implementar la escucha tres señales principales para asegurar el correcto funcionamiento de la comunicación a través de *sockets*:

- Conexión: Se encargará de informar de que un nuevo cliente se ha conectado a la aplicación.
- Desconexión: Informará al *Back-end* de que se ha producido una desconexión de un usuario de la aplicación móvil.
- Emparejamiento: Esta función se encargará de asociar la conexión mediante un *socket* con la cuenta de usuario correspondiente a ese cliente.

Al igual que en la *API REST*, se implementará un método se autenticación segura para permitir solo las comunicaciones a usuarios correctamente logueados. Para ello se desarrollará un *middleware* a nivel de *socket* que se encargará de bloquear las conexiones de clientes que no están autorizados. Se utilizará un *token JWT* en cada transmisión que será comprobado para asegurar su validez. El *middleware* es una funcionalidad incluida en *Socket.IO* que captura la petición antes de llegar a los controladores y decide sí debe permitir la conexión o no.

Se debe contemplar una lista en la memoria del *Back-end* con los usuarios que están conectados en ese momento y que esté asociada al *socket* de cada uno. De esta forma será posible la comunicación entre clientes a través de su identificador de usuario independientemente del *socket*. Esto permitirá asegurar la comunicación, aunque un usuario se desconecte y se vuelva a conectar. La lista debe implementar una serie de funciones como son:

- Añadir nuevo usuario
- Eliminar un usuario
- Obtención de un usuario
- Obtención de todos los usuarios

4.2.2.4 Registro de todas las acciones

Con el objetivo de reflejar todos los eventos y situaciones que ocurren durante el periodo de ejecución y funcionamiento del *Back-end* se utilizará un *logger* que se encargará de escribir en un fichero de texto todos los eventos y errores.

Para realizar este objetivo se utilizará la librería *Winston*.[47] *Winston* está diseñado para crear un sistema de registro simple y universal con soporte para múltiples formatos. *Winston* se puede configurar a través de niveles. Cada nivel indica el grado de gravedad que supone un evento y es especificado por el desarrollador. Cada uno tiene una prioridad específica y cuanto mayor sea esa prioridad, más importante se considerará el mensaje. Los niveles que se utilizan son los siguientes:

- Error
- Alerta
- Información
- Depuración

Los archivos de registro se almacenarán en un fichero de texto que incluirá en su nombre la fecha de ese día. Se utilizará el formato "*YYYY-MM-DD*".

4.2.3 Desarrollo de un modelo predictivo con Machine Learning

Otro de los objetivos principales del proyecto será el desarrollo de un modelo predictivo, basado en el análisis de sentimientos, que calcule la polaridad de los mensajes de los usuarios. Con el uso de este modelo se pretende:

- Detectar el grado de amenaza que sufre un determinado usuario, con el fin de indicar a los psicólogos quienes son los usuarios más vulnerables, para poder atenderles primero.
- Reflejar el grado de progreso del usuario midiendo la polaridad de sus mensajes para ver como mejora o empeora mientras utiliza la aplicación.

Para la consecución de este objetivo se utilizarán *scripts* en lenguaje *Python* que realizarán pequeñas tareas de selección, limpieza y procesamiento de datos, y posteriormente se crearán diferentes modelos predictivos con el fin de compararlos y escoger el modelo que mejor resultados ofrezca.

4.2.3.1 Selección de un conjunto de datos

El primer paso consistirá en buscar un conjunto de datos en Internet que esté constituido por oraciones en las que se clasifique la polaridad de cada una de estas en positivo, negativo o neutro. Puesto que la aplicación está destinada a los usuarios de España, el conjunto de datos tiene que ser en español. Se analizarán varias fuentes de datos en busca de un conjunto que sea válido.

Las principales fuentes de datos que se analizarán serán conjuntos de datos en *Twitter*, es decir *tweets*. Debido a que en esta red social la gente plasma de forma muy clara sus sentimientos es posible encontrar un conjunto de datos donde se clasifique la polaridad de los *tweets*.[48]

Una vez seleccionados los conjuntos de datos, se realizará un *script* en *Python* para analizarlos brevemente comprobando si es apto. Para que sea apto será necesario que todos los *tweets* tengan una polaridad asociada y que el conjunto de datos está bien balanceado, es decir se tiene que mantener una proporción equitativa de *tweets* positivos, negativos y neutros. También se debe tener en cuenta el tamaño del conjunto de datos seleccionado, ya que tiene que ser una muestra representativa.[49]

4.2.3.2 Limpieza de datos

Una vez seleccionado un buen conjunto de datos será necesario aplicarle una serie de tareas de limpieza de datos con el fin de mejorar la calidad del conjunto y que sea apto para el entrenamiento de un modelo predictivo.

En primer lugar, se deben de eliminar las características de los *tweets* que no estén relacionadas con la polaridad, ya que no será necesario su uso. Si el conjunto de datos estuviese dividido en partes sería necesario juntarlas en un mismo archivo para facilitar el trabajo de análisis. Después, deben de eliminarse los datos duplicados ya que pueden afectar negativamente al modelo.

En cuanto a los *tweets*, puede que sea necesario aplicarles algún tipo de limpieza. Para realizar estas tareas se deberá utilizar la librería de *Python NLTK*, un conjunto de bibliotecas y herramientas para el procesamiento del lenguaje natural que permite realizar múltiples tareas relacionadas con el análisis de texto.

Las tareas de limpieza básicas que se deben aplicar a los *tweets*, para mejorar la calidad y el procesamiento, son las siguientes:

- Eliminar los caracteres extraños.
- Eliminar los símbolos de puntuación.
- Convertir todo el texto a minúsculas.
- Lematizar todas las palabras del texto.

Una vez aplicadas las tareas de limpieza se deberá de almacenar el conjunto de datos limpio en un nuevo fichero, en formato CSV.

4.2.3.3 Entrenamiento de modelos predictivos

Se realizará un estudio de los diferentes algoritmos de *Machine Learning* que existen y sus ventajas frente a otros, así como los requisitos que tiene cada uno y como se deben de implementar.

Una vez esté listo el conjunto de datos limpio se procederá a entrenar diferentes algoritmos predictivos con esos datos. Es importante a determinar si utilizar algoritmos de clasificación binaria o de clasificación multiclase.[50]

Será necesario determinar la clasificación se realizará en tres categorías o solo en 2. En el caso de utilizar 3 categorías están serían:

- Neutro
- Positivo
- Negativo

Si se utilizan dos categorías estas serían:

- Negativo
- No negativo

Es importante realizar varias pruebas con diferentes algoritmos para determinar el tipo de clasificación que será más efectiva.

Tiene sentido utilizar clasificadores binarios porque realmente solo se necesitan identificar los mensajes negativos y diferenciarlos de los mensajes positivos.

Independientemente del tipo de clasificadores que se escojan será necesario utilizar algún método para ajustar los parámetros del algoritmo y así conseguir el mejor modelo posible con el conjunto de datos que se aporte.

4.2.3.4 Pruebas de validación cruzada y comparativa de modelos

Una vez se haya determinado el tipo de clasificadores a utilizar y se hayan elegido los algoritmos predictivos que se van a implementar, se debe proceder a la realización de pruebas con el fin de determinar cuál de los modelos predictivos ofrece mejores resultados frente al resto.

Para realizar estas comparativas son necesarias las pruebas de validación cruzada. La validación cruzada es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Es una de las técnicas más utilizada en proyectos de inteligencia artificial para validar modelos predictivos.

Para realizar estas pruebas será necesario establecer las métricas que se utilizarán para comparar los modelos. Las principales métricas que se utilizan para la clasificación son las siguientes:

- Matriz de confusión o error
- Precisión
- Sensibilidad
- Especificidad o tasa negativa real
- *F1-Score*
- Área bajo la curva o *AUC ROC*
- Pérdida logarítmica

Después de realizar las pruebas de validación cruzada, se seleccionará el mejor modelo predictivo de todos a partir de los resultados de las métricas.

4.2.3.5 Integración del modelo predictivo

Una vez se haya seleccionado el mejor modelo predictivo de los que se han comparado en la fase anterior, este se debe integrar como parte de *Connect* para realizar predicciones en la polaridad de los mensajes de los usuarios víctima.

Para realizar esta integración se creará una herramienta en *Python* que obtendrá una lista de mensajes de los usuarios víctima a través de una consulta a la base de datos de *Connect*.

Con los mensajes de la lista se podrá predecir su polaridad mediante el uso el modelo estadístico. Una vez se calcule la polaridad de cada uno se devolverán a la base de datos utilizando otra consulta de *Mongo*.

Esta herramienta funcionará a modo de demonio, es decir, comprobará continuamente, mediante una consulta a la base de datos de *Mongo*, si existen mensajes de usuarios victimas que no tengan polaridad, y de ser así, se obtendrán dichos mensajes para predecir su polaridad.

Un demonio o servicio es un programa que se ejecuta en segundo plano, fuera del control interactivo de los usuarios del sistema ya que carecen de interfaz con estos. El término demonio se usa fundamentalmente en sistemas *UNIX* y basados en *UNIX*.[51]

5. Descripción técnica

Una vez han sido identificados y explicados los objetivos se procederá a explicar de una forma más detallada como se ha realizado el desarrollo del proyecto y si se han cumplido los objetivos establecido y de qué forma.

Para explicar cómo ha sido el desarrollo del proyecto es necesario separar de nuevo el proyecto en tres partes diferenciadas pero complementarias. Estas partes se pueden denominar subproyectos, debido a que unidas constituyen la totalidad del proyecto.

5.1 Aplicación destinada a usuarios víctimas

La aplicación destinada a los usuarios víctimas se ha desarrollado utilizando tecnologías web, concretamente *Angular 7*. Una vez finalizado el desarrollo de esta se ha exportado utilizando *Apache Cordova*. A continuación, se detalla cómo se ha realizado el desarrollo de esta.

En primer lugar, la aplicación se creó partiendo de cero, es decir un proyecto totalmente nuevo. El entorno de desarrollo es un equipo *Ubuntu* que cuenta con todo el software necesario para la creación de la aplicación. Se escogió *Ubuntu* debido a la gestión cómoda de instalación y actualización de software, a que se los escritorios son intercambiables y configurables y que es un sistema operativo *open source*.

A continuación, se describe y se justifica por secciones como se realizó la creación de la aplicación:

5.1.1 Autenticación

La aplicación consta de un inicio de sesión seguro mediante un *token JWT*, que también garantiza la seguridad e integridad de las comunicaciones futuras.

5.1.1.1 Registro

Cuando un usuario utiliza la aplicación por primera vez deberá registrarse por medio de un formulario en el que se solicita información básica como:

- Email
- Nombre de usuario
- Contraseña
- Tipo de problema

Este formulario se ha desarrollado en forma de componente de *Angular* y consta de tres partes que están muy relacionadas:

- Vista *HTML*
- Controlador
- Estilos

Todos los componentes que se describan en la aplicación constarán de estas tres partes básicas.

Puesto que se va a aprovechar toda la funcionalidad de *Angular* este formulario utilizará *two data binding*. Esta técnica consiste en un enlace de datos que une una fuente de datos entre el origen de datos y las vista, y se encarga de sincronizarlos. Esto provoca que cada cambio de datos se refleje automáticamente en los elementos que están sincronizados. De esta forma cuando el usuario escribe sus datos, automáticamente se actualizan en el controlador y en el momento en el que se realiza el registro los datos ya están reflejados. Esta técnica se utilizará con todos los formularios de la aplicación, así como también se aplicará a cualquier dato que el usuario pueda modificar desde la vista. De esta forma el controlador tiene acceso al dato actualizado en tiempo real sin tener que solicitarlo.

Una vez el usuario haya completado el registro y decida enviar sus datos de registro se comprobará que todos los datos se han completado de forma correcta y se procederá a formular una petición a un servicio que se encarga de las acciones de autenticación.

```
signup() {
  if (this.user.email && this.user.username && this.user.password && this.user.problem_type) {

    this._authService.signUp(this.user).subscribe(
      response => {
        if (response.identity && response.token) {
          this.error = null;
          this._authService.setSession(response.identity, response.token);
          $('#signup').modal('hide');
          this._socketService.emit('pairing');
        }
      },

      error => {
        this.error = error._body ? JSON.parse(error._body).info : 'Se ha producido un error.';
      }
    );
  } else {
    this.error = 'Debe rellenar todos los datos.';
  }
}
```

Figura 5.1 - Método signup

Si la respuesta del servidor es correcta se iniciará sesión de forma automática con esta cuenta de usuario y se realizará una petición por *socket* para emparejar al usuario.

5.1.1.2 Inicio de sesión

Si el usuario ya tiene una cuenta, en lugar de registrarse deberá iniciar sesión con sus credenciales. Para eso se utilizará otro componente de *Angular* encargado únicamente de realizar el logueo de los usuarios de la aplicación.

Este componente utilizará otro formulario en el que únicamente se solicitará el nombre de usuario y la contraseña. Una vez el usuario haya completado el formulario y decida iniciar sesión se procederá a formular una petición a un servicio que se encarga de las acciones de autenticación.

```
signin() {
  if (this.user.username && this.user.password) {
    this._authService.signIn(this.user).subscribe(
      response => {
        if (response.identity && response.token) {
          this.error = null;
          this._authService.setSession(response.identity, response.token);
          $('#signin').modal('hide');
          this._socketService.emit('pairing');
        }
      },

      error => {
        this.error = error._body ? JSON.parse(error._body).info : 'Se ha producido un error.';
      }
    );
  } else {
    this.error = 'Debe rellenar todos los datos.';
  }
}
```

Figura 5.2 - Método signin

Si la respuesta del servidor es correcta se iniciará sesión en la aplicación con esta cuenta de usuario y se realizará una petición por *socket* para emparejar al usuario.

5.1.1.3 Comprobación del token JWT

La aplicación tiene persistencia y almacena los datos de usuario en el almacenamiento local. Esta funcionalidad permite que no sea necesario iniciar sesión cada vez que se abre la aplicación.

Cuando el usuario abra la aplicación, si ya ha iniciado sesión anteriormente, solamente se comprobará si el *token JWT* sigue siendo válido. En caso contrario se cerrará la sesión de forma inmediata obligando al usuario a iniciarla de nuevo.

```
getSession() {
  let session = this._authService.getSession();

  if (session.identity && session.token) {
    this._authService.checkToken(session.token).subscribe(
      response => {
        this.session = session;
      },

      error => {
        this._authService.removeSession();
        this.session = { identity: '', token: '' };
      }
    );
  } else {
    this.session = { identity: '', token: '' };
  }

}
```

Figura 5.3 - Método getSession

Para comprobar si el *token* es válido se formulará, desde el componente principal de la aplicación, una petición al servicio que se encarga de las acciones de autenticación.

5.1.1.4 Servicio encargado de la autenticación

Además de utilizar componentes, *Angular* también permite el uso de servicios. Los servicios añaden funcionalidad extra y la separan de los componentes ofreciendo de esta forma una abstracción.

El servicio encargado de la autenticación es el responsable de establecer la comunicación con la *API REST* e integra cada una de las funciones a las que se hace referencia en el registro, el inicio de sesión y la comprobación del *token*.

```
signUp(user: any) {
  let params = JSON.stringify(user);

  let headers = new Headers({
    'Content-Type': 'application/json'
  });

  let options = new RequestOptions({ headers: headers });

  return this._http.post(environment.SERVER_URL + '/auth/signup', params, options).map(res => res.json());
}
```

Figura 5.4 - Método signUp del servicio

```
signIn(user: any) {
  let params = JSON.stringify(user);

  let headers = new Headers({
    'Content-Type': 'application/json'
  });

  let options = new RequestOptions({ headers: headers });

  return this._http.post(environment.SERVER_URL + '/auth/signin', params, options).map(res => res.json());
}
```

Figura 5.6 - Método signIn del servicio

```
checkToken(token: string) {
  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http.get(environment.SERVER_URL + '/auth/checkToken', options).map(res => res.json());
}
```

Figura 5.5 - Método checkToken del servicio

5.1.2 Chats

Esta sección hace referencia a todo el desarrollo de las conversaciones con los psicólogos y se detalla su funcionamiento.

5.1.2.1 Visualización de lista de chats

La interfaz en la que se muestra la lista de chats se desarrollará mediante un componente de *Angular* y su única función será la de mostrar los chats.

Para mostrar al usuario la lista de chats en los que participa será necesario realizar una petición a través de un servicio, que se encarga de las operaciones de los chats, para obtener la lista de chats actualizada de un usuario. Esta lista se mostrará en la interfaz de una forma limpia en la que cada chat representará una única fila.

Es importante mostrar los chats que tienen mensajes sin leer con algún símbolo distintivo para notificar al usuario. Se ha utilizado un pequeño círculo que aloja en su interior el número de mensajes sin leer de un chat.

```
getChats() {
  this._chatService.getChats().subscribe(
    response => {
      if (response.length > 0) {
        this.chats = response;

        const unReadChats = this.chats.filter(chat => chat.unread_count > 0).length;
        document.getElementById('unreadCount').innerText = unReadChats.toString();

        if (unReadChats == 0) document.getElementById('unreadCount').style.display = 'none';
        else document.getElementById('unreadCount').style.display = 'block';
      } else {
        this.chats = [];
      }
    },

    error => {
      if (error.status == 401) this._authService.removeSession();
    }
  );
}
```

Figura 5.7 - Método getChats

Una vez el usuario tenga acceso a la lista de chats, al pulsar sobre uno de ellos, esté se abrirá en un nuevo componente.

```
openChat(chat_id: string) {
  this.router.navigate(['chat', chat_id]);
}
```

Figura 5.8 - Método openChat

5.1.2.2 Visualización del chat seleccionado

La interfaz en la que se muestra el chat seleccionado se desarrollará mediante un componente de *Angular* y sus funciones serán las siguientes:

- Obtener información del chat
- Obtener los mensajes
- Buscar mensajes
- Enviar un nuevo mensaje
- Eliminar chat

En primer lugar, cuando el usuario acceda a un chat concreto se deben obtener la información del chat y la lista de mensajes. Estas tareas se realizan de forma paralela aprovechando que *JavaScript* permite la ejecución asíncrona.

Para obtener la información del chat será necesario realizar una petición a través del servicio encargado de las operaciones de los chats.

```
getChat(chat_id: string) {
  this._chatService.getChat(chat_id).subscribe(
    response => {
      this.chat = response;
    },

    error => {
      if (error.status == 401) this._authService.removeSession();
    }
  );
}
```

Figura 5.9 - Método getChat

De la misma forma se realizará otra petición para obtener los mensajes del chat, pero en esta ocasión se utilizará el servicio encargado de las operaciones con mensajes.

```
getMessages(chat_id: string) {
  this._messageService.getMessages(chat_id, this.messages ? this.messages.length : 0, this.messagesPerPage).subscribe(
    response => {
      if (this.messages) this.messages = response.concat(this.messages);
      else this.messages = response;
    },

    error => {
      if (error.status == 401) this._authService.removeSession();
    }
  );
}
```

Figura 5.10 - Método getMessages

La funcionalidad encargada de filtrar los mensajes por un determinado campo de texto se realizará desde la vista, aprovechando que *Angular* permite filtrar elementos aplicando directivas que ya vienen incorporadas.

```
<ng-container *ngFor="let message of messages">
  <ng-container *ngIf="!search_query || message.message.toLowerCase().includes(search_query.toLowerCase())">
    <div *ngIf="message.user == chat.user; then own_message; else unrelated"></div>

    <ng-template #own_message>
      <div class="row justify-content-end mt-2">
        <div class="card own">
          <div class="card-body">
            {{ message.message }}
            <span class="small text-muted text-right m-0 ml-1">{{ message.creation_time | smartdate }}</span>
          </div>
        </div>
      </div>
    </ng-template>

    <ng-template #unrelated>
      <div class="row justify-content-start mt-2">
        <div class="card unrelated">
          <div class="card-body">
            {{ message.message }}
            <span class="small text-muted text-right m-0 ml-1">{{ message.creation_time | smartdate }}</span>
          </div>
        </div>
      </div>
    </ng-template>
  </ng-container>
</ng-container>
```

Figura 5.11 - Filtrado de mensajes por un campo de texto

Como se observa en la figura 5.11 utilizando la directiva "*ngIf*" se muestra únicamente, mediante una condición, los mensajes que incluyen la cadena de texto seleccionada.

En la parte inferior de esta interfaz se encuentra un *input* para introducir los mensajes que se desean enviar, junto con un botón para enviarlos. No se permitirán mensajes en blanco y se eliminarán los espacios que haya al principio y al final de cada mensaje. Para enviar mensajes se realizará una petición al servicio encargado de las operaciones con los mensajes.

Si el usuario lo desea puede eliminar un chat en el que participe. Cuando lo haga se realizará una petición al servicio de chats que se encargará de transmitir la petición a la *API REST*. Una vez sea eliminado un chat, el usuario será redirigido a la interfaz en la que se muestra la lista de chats.

```
deleteChat() {
  this._chatService.deleteChat(this.chat._id).subscribe(response => {
    this._router.navigateByUrl('/chats');
  });
}
```

Figura 5.12 - Método deleteChats

5.1.2.3 Creación de un nuevo chat con un psicólogo

Se proporcionará al usuario una interfaz en la que pueda obtener una lista de psicólogos con los que entablar una conversación. Esta lista de psicólogos constituye un nuevo componente de *Angular*. Desde aquí el usuario podrá seleccionar al psicólogo que desee e iniciar una conversación con él.

Las funciones que debe implementar este componente son las siguientes:

- Obtención de la lista de psicólogos disponibles.
- Creación de un chat entre el usuario y el psicólogo.

Cuando el usuario acceda a esta interfaz se debe obtener la lista de psicólogos disponibles y mostrarla de forma visible e interactiva para el usuario.

Para obtener la lista de psicólogos será necesario realizar una petición a través del servicio encargado de las operaciones de los chats.

```
getExperts() {
  this._userService.getExperts().subscribe(
    response => {
      this.experts = response.experts;
    },

    error => {
      if (error.status == 401) this._authService.removeSession();
    }
  );
}
```

Figura 5.13 - Método getExperts

Cuando el usuario seleccione al psicólogo con el que desea iniciar un chat, se ejecutará la función encargada de la creación de un nuevo chat a través del servicio encargado de las operaciones de los chats.

```
addChat() {
  this._chatService.createChat(this.selected_expert_id).subscribe(
    response => {
      if (response.message == 'OK') this._router.navigateByUrl('/chats');
    },

    error => {
      if (error.status == 401) this._authService.removeSession();
      if (error.status == 409) $('#errorAddChat').modal('show');
    }
  );
}
```

Figura 5.14 - Método addChat

5.1.2.4 *Servicio encargado de los chats*

El servicio encargado de los chats es el responsable de establecer la comunicación con la *API REST* e integra cada una de las funciones a las que se hace referencia en la visualización de la lista de chats y de un chat seleccionado. Las funciones que realiza son:

- Obtención de la lista chats
- Obtención de un chat concreto
- Eliminar un chat

```
getChats() {
  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: this._authService.getSession().token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http.get(environment.SERVER_URL + '/chat/chats', options).map(res => res.json());
}
```

Figura 5.15 - Método getChats del servicio

```
getChat(chat_id: string) {
  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: this._authService.getSession().token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http.get(environment.SERVER_URL + '/chat/fullInfo/' + chat_id, options).map(res => res.json());
}
```

Figura 5.17 - Método getChat del servicio

```
deleteChat(chat_id: string) {
  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: this._authService.getSession().token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http.delete(environment.SERVER_URL + '/chat/' + chat_id, options).map(res => res.json());
}
```

Figura 5.16 - Método deleteChat del servicio

5.1.2.5 Servicio encargado de los mensajes

El servicio encargado de los mensajes es el responsable de las iteraciones que se pueden realizar sobre los mensajes de un determinado chat. Se encarga de establecer la comunicación con la *API REST* e integra cada una de las funciones necesarias para interactuar con los mensajes. Las funciones que realiza son:

- Obtención de mensajes de un chat
- Envío de mensajes
- Marcar como leído los mensajes

```
getMessages(chat_id: string, offset = 0, limit = 20) {
  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: this._authService.getSession().token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http
    .get(environment.SERVER_URL + '/message/messages/' + chat_id + '?offset=' + offset + '&limit=' + limit, options)
    .map(res => res.json());
}
```

Figura 5.18 - Método getMessages del servicio

```
sendMessage(chat_id: string, message: string) {
  let params = JSON.stringify({ message: message });

  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: this._authService.getSession().token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http
    .post(environment.SERVER_URL + '/message/sendMessage/' + chat_id, params, options)
    .map(res => res.json());
}
```

Figura 5.19 - Método sendMessage del servicio

```
readMessages(chat_id: string, message_id: string) {
  let headers = new Headers({
    'Content-Type': 'application/json',
    Authorization: this._authService.getSession().token
  });

  let options = new RequestOptions({ headers: headers });

  return this._http
    .get(environment.SERVER_URL + '/message/readMessages/' + chat_id + '/' + message_id, options)
    .map(res => res.json());
}
```

Figura 5.20 - Método readMessages del servicio

5.1.3 Botón del pánico

Una de las funcionalidades más importantes que se ha desarrollado durante el proyecto es la implementación de un botón del pánico que tiene la función de obtener las ubicaciones del usuario, solicitando previamente su permiso, para enviársela a la policía informando de que el usuario se encuentra en peligro y necesita ayuda de forma inminente. El botón del pánico se ha desarrollado en un componente de *Angular*.

En la parte superior de la interfaz se encuentra ubicado un botón de ayuda que muestra un modal en el que se puede visualizar un menú de ayuda que informa al usuario de la funcionalidad de este botón.

Para obtener la ubicación se ha utilizado la *API* del navegador mediante una petición desde el controlador.

```
initGeolocation() {
  if (navigator && navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(this.successCallback.bind(this), this.errorCallback.bind(this));
  }
}
```

Figura 5.21 - Método para obtener la geolocalización

Se han tenido en cuenta las situaciones que pueden ocurrir cuando se solicite la ubicación y esta no se devuelva como se espera. Estas situaciones son las siguientes:

- El usuario no aún no ha conducido el permiso necesario para obtener la ubicación.
- La información de la localización no está disponible.
- Se ha excedido el tiempo para obtener la ubicación.
- El usuario ha denegado la petición.

Todos estos casos han sido controlados para ofrecerle está información al usuario.

```
errorCallback(error: PositionError) {
  this.errorInGeolocation = 'Se ha producido un error obteniendo la ubicación.';

  switch (error.code) {
    case error.PERMISSION_DENIED:
      this.errorInGeolocation = 'Permita la ubicación para usar esta funcionalidad.';
      break;
    case error.POSITION_UNAVAILABLE:
      this.errorInGeolocation = 'La información de la localización no está disponible.';
      break;
    case error.TIMEOUT:
      this.errorInGeolocation = 'Se ha execido el tiempo para obtener la ubicación.';
      break;
  }
}
```

Figura 5.22 - Método para controlar los errores en la geolocalización

5.1.4 Acciones de un usuario sobre su cuenta

Se ha desarrollado un componente de *Angular* que se encarga de gestionar y mostrar todas las acciones que los usuarios pueden realizar sobre su cuenta. Esta sección mostrará un menú de opciones que el usuario puede realizar.

5.1.4.1 Modificación de los datos de su cuenta de acceso

Se mostrará al usuario un formulario en el que puede modificar sus datos asociados a su cuenta como son:

- Nombre de usuario
- Email
- Tipo de problema
- Contraseña

Una vez que el usuario haya rellenado y decida realizar la modificación será necesario realizar una petición a través del servicio, que se encarga de las operaciones de los usuarios, para ejecutar esta acción.

```
updateUser() {
  this._userService.updateUser(this.user).subscribe(
    response => {
      this.user = response.user;
      this.success = 'El usuario se actualizó correctamente.';
      this.error = '';
    },
    error => {
      this.error = 'Se ha producido un error actualizando el usuario.';
      if (error._body) this.error = JSON.parse(error._body).info;
      console.log(error);
    }
  );
}
```

Figura 5.23 - Método updateUser

5.1.4.2 Cerrar sesión

Otra de las acciones que el usuario puede realizar desde este componente es el cierre de sus sesiones. Cuando decida cerrarla, se eliminará su *token JWT* y su identificación del almacenamiento local, y será redirigido directamente al inicio de sesión.

```
closeSession() {
  this._authService.removeSession();
  this._socketService.disconnect();
}
```

Figura 5.24 - Método closeSession

5.1.4.3 Eliminar de todos los chats

El usuario podrá decidir en cualquier momento si desea eliminar todos los chats en los que participa. Si elige esta opción se realizará una petición a través del servicio, que se encarga de las operaciones de los usuarios, para ejecutar esta acción.

Una vez se hayan eliminado todos los chats se mostrará un mensaje en la interfaz para notificar al usuario de que la acción de ha ejecutado de forma correcta.

5.1.4.4 Borrar la cuenta de usuario

El usuario podrá decidir en cualquier momento si desea eliminar por completo su cuenta, incluyendo los datos relacionados con esta. Si elige esta opción se realizará una petición a través del servicio, que se encarga de las operaciones de los usuarios, para ejecutar esta acción.

Una vez se hayan ejecutado esta acción el usuario será redirigido al menú principal de la aplicación, donde si desea seguir utilizando la aplicación, deberá crear una cuenta de usuario nueva.

```
deleteAccount() {
  this._authService.deleteAccount().subscribe(response => {
    this._authService.removeSession();
  });
}
```

Figura 5.25 - Método deleteAccount

5.1.5 Ocultación de la aplicación

Se ha desarrollado un componente de *Angular* que se encarga de activar, desactivar y gestionar la ocultación de la aplicación. Esta funcionalidad se encarga de mostrar una calculadora cuando se accede a la aplicación con el fin de ocultar la verdadera aplicación y proteger a los usuarios.

Desde esta sección se mostrará una opción que al activarla solicitará un código. Este código es el que se debe introducir en la calculadora para mostrar la verdadera funcionalidad de la aplicación. Si no se introduce el código secreto la calculadora seguirá funcionando a modo de interfaz que realiza operaciones matemáticas básicas.

A continuación, se muestran los dos métodos básicos encargados de activar y desactivar la ocultación de la aplicación.

```
enableStealth() {
  this._stealthService.activate(this.secretCode);
}

disableStealth() {
  this._stealthService.deactate();
}
```

Figura 5.26 - Métodos para activar y desactivar la ocultación

Una vez activada la ocultación, cuando el usuario salga de la aplicación y vuelva a acceder a ella se mostrará la calculadora.

5.1.6 Comunicación en tiempo real

Para implementar un sistema de comunicación en tiempo real que sea rápido y seguro se ha utilizado *Socket.IO* que permite realizar este tipo de comunicaciones a través de *websockets*.

Puesto que muchos componentes de la aplicación necesitan este tipo de comunicación se ha creado un servicio en el que se han implementado los dos métodos básicos para proporcionar una abstracción a todos los componentes que utilicen las funciones de este servicio.

Las accesiones básicas que debe realizar un *socket* son el envío y la escucha de determinados eventos. Estos eventos se han implementado de la siguiente manera:

El método encargado de emitir un evento al servidor recibe los siguientes parámetros:

- Nombre del evento
- Datos del evento
- Función de retorno que se ejecuta después de emitir el evento

```
emit(event: string, data?: any, callback?: Function) {
  console.log(`Emitiendo ${event}`);

  const payload = {
    user: this._authService.getSession(),
    data: data
  };
  this.socket.emit(event, payload, callback);
}
```

Figura 5.28 - Método para emitir eventos a través del socket

Por otro lado, el método encargado de escuchar un evento y ejecutar una función al recibirlo solo recibe el nombre del evento que está escuchando.

```
listen(event: string) {
  return this.socket.fromEvent(event);
}
```

Figura 5.27 - Método para escuchar eventos a través del socket

Con estas dos funciones básicas se puede establecer una comunicación en tiempo real con el servidor.

Para ello los componentes implementarán las funciones de este servicio, y en el caso de la escucha de un evento, se ejecutará una acción cuando se reciba.

Un ejemplo de comunicación en tiempo real se produce en el componente que muestra un chat determinado. Este componente creará una señal implementando el método de escucha del *socket* para recibir nuevos mensajes. En el momento en el que se reciba un mensaje nuevo se disparará una acción que consistirá en mostrar el nuevo mensaje en el panel de mensajes del chat.

```
getMessageSignal() {
  return this._socketService.listen('new-message').subscribe((response: any) => {
    if (this.chat.expert.id == response.message.user) {
      this.messages.push(response.message);
      this._messageService.readMessages(response.message.chat_id, response.message._id).subscribe();
      this.scrollToBottom();
    }
  });
}
```

Figura 5.29 - Señal para escuchar nuevos mensajes a través del socket

5.2 Back-end

El *Back-end* encargado de almacenar todos los datos que se generan con el uso de las aplicaciones clientes y de comunicar a los usuarios victimas con los psicólogos se ha desarrollado utilizando *Node.js*.

Normalmente, el cuello de botella en toda la arquitectura de aplicación web, el tráfico, la velocidad de procesador y la velocidad de memoria, era el número máximo de conexiones concurrentes que podía manejar un servidor. *Node.js* resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de generar un nuevo hilo para cada conexión y de asignarle la memoria acompañante, cada conexión dispara una ejecución de evento dentro del proceso del motor de *Node*. *Node* también asegura que nunca se quedará en punto muerto, porque no se permiten bloqueos y porque no se bloquea directamente cuando se realizan operaciones de *E/S*.[52] *Node* consigue que un servidor que lo ejecute pueda soportar decenas de miles de conexiones concurrentes.

Node.js incorpora varios módulos básicos de forma nativa, como por ejemplo el módulo de red, que proporciona una capa para programación de red asíncrona y otros módulos fundamentales, como *Path*, *FileSystem*, *Buffer* o *Timers*. También se pueden incorporar módulos propios o de terceros a un proyecto de *Node*, en forma de dependencias que se especifican en un fichero llamado *package.json*.

El *Back-end* se ha construido partiendo de cero en un entorno de desarrollo en el que se ha utilizado un equipo *Ubuntu* que cuenta con todo el software necesario para la creación del *Back-end*.

El servidor donde se aloja el *Back-end* está en una *Raspberry Pi 3* configurada como un servidor y expuesta de forma remota en el puerto 3977. Este servidor contiene una *API REST* y un *socket* para las comunicaciones en tiempo real.

5.2.1 API REST

El *Back-end* implementa una *API REST* encarda de ofrecer una interfaz de programación de aplicaciones que se apoya en la arquitectura *REST* para el desarrollo de aplicaciones en red. Para ello se utilizará *Express*, un *framework* para *Node.js* que sirve para crear aplicaciones web de forma minimalista y flexible en poco tiempo, ya que proporciona funcionalidades como enrutamiento, opciones para gestionar sesiones y *cookies*. Esta *API*

REST permite realizar todo tipo de consultas mediante estas cuatro operaciones: *GET*, *POST*, *PUT* y *DELETE*.

La *API REST* ofrece a las aplicaciones clientes un servicio a través del cual puede obtener todos los datos necesarios para el correcto funcionamiento de la aplicación. Se implementarán una serie de rutas y controladores para obtener y modelar todos los datos. Además, se incluye un *middleware* que garantiza seguridad utilizando un *token* generado con *JSON Web Token*.

La *API* se ha desarrollada de forma modular, para facilitar el desarrollo y ofrecer un alto grado de escalabilidad futura.

5.2.1.1 Middleware

Se ha diseñado un *middleware* a nivel de *API* para garantizar la seguridad y la integridad de las comunicaciones con las aplicaciones de los clientes.

Este *middleware* se implementa en las rutas a las que solamente puede acceder un usuario que previamente haya iniciado sesión en la aplicación. Antes de que la petición llegué al controlador es capturada por el *middleware* que comprueba si lleva la cabecera de autorización con el *token JWT*. De no ser así devolverá un error 401 que indica que el cliente no se ha autenticado. Si existe la cabecera de autenticación se comprobará si el *token* es válido. Para ser considerado válido debe de poder desencriptarse con una clave secreta que solo conoce el *Back-end*, y que la fecha de validez no haya expirado aún.

Si el *token* es correcto se liberará la petición y esta ya será procesada por el controlador que le corresponda.

```
exports.ensureAuth = function(req, res, next) {

  if (!req.headers.authorization) {
    return res.status(401).send({info: 'La petición no tiene la cabecera de autenticación'});
  }

  const payload = jwt.verify(req.headers.authorization.replace(/['"]+/g, ''));
  if (!payload) return res.status(401).send({info: 'El token ha experidado'});

  req.user = payload;
  next();

};
```

Figura 5.30 - Middleware de la API REST

5.2.1.2 Endpoints

Las aplicaciones clientes solicitarán los datos y realizarán las comunicaciones con la *API REST* a través de *endpoints*. Los *endpoints* son las *URLs* a través de las cuales se recibe y retorna información de la *API*. Cada una de estas rutas está asociadas a un controlador que es quien se encarga de procesar las peticiones que recibe y de devolver una respuesta al cliente.

Las rutas de la aplicación se dividen en 4 clases que además están ubicadas en ficheros diferentes:

- Autenticación
- Chats
- Mensajes
- Usuarios

A continuación, se muestran y se explican todos los *endpoints* que están integrados en la *API REST*.

Tabla 5.1 - Endpoint SignUp

SignUp	
Descripción	Se encarga de crear una nueva cuenta de usuario.
Grupo	Autenticación
Método	Post
Ruta	auth/signup
Parámetros	• username • email • password • problemType

Respuesta	Devuelve la información de la cuenta de usuario que se ha creado y un *token JWT* para la autenticación del cliente.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.2 - Endpoint SignIn

SignIn	
Descripción	Se encarga de generar un *token* de inicio de sesión para el cliente.
Grupo	Autenticación
Método	Post
Ruta	auth/signin
Parámetros	• username o emial • password
Respuesta	Devuelve un *token JWT* para la autenticación del cliente.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.3 - Endpoint CheckToken

CheckToken	
Descripción	Se encarga de comprobar si el *token* que se recibe es válido.
Grupo	Autenticación
Método	Get
Ruta	auth/checkToken
Parámetros	• token (a través de la cabecera de autenticación)
Respuesta	Devuelve el mensaje “ok”.
Respuesta en caso de error	• Indica que el *token* no es válido. Código 401

Tabla 5.4 - Endpoint DeleteAccount

DeleteAccount	
Descripción	Se encarga de eliminar todos los datos relacionados con una cuenta de usuario.
Grupo	Autenticación
Método	Delete
Ruta	auth/
Parámetros	• userID (a través de la cabecera de autenticación)
Respuesta	Devuelve el mensaje “ok”.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400.

	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.5 - Endpoint GetChatInfo

GetChatInfo	
Descripción	Se encarga de devolver la información de un chat.
Grupo	Chats
Método	Get
Ruta	chat/fullInfo
Parámetros	• chatID
Respuesta	Devuelve la información del chat solicitado mediante su ID.
Respuesta en caso de error	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.6 - Endpoint CreateChat

CreateChat	
Descripción	Se encarga de crear un chat nuevo entre un usuario y un psicólogo.
Grupo	Chats
Método	Get
Ruta	chat/create
Parámetros	• expertID (ID del psicólogo)

Respuesta	Devuelve la información del chat que se ha creado.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.7 - Endpoint GetChats

GetChats	
Descripción	Se encarga de obtener la lista de chats en los que participa el usuario.
Grupo	Chats
Método	Get
Ruta	chat/chats
Parámetros	• userID (a través de la cabecera de autenticación) • offset (el último chat que tiene el usuario) • limit (él número de chats que se desea obtener)
Respuesta	Devuelve la lista de chats en los que participa el usuario.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400.

	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.8 - Endpoint DeleteChats

DeleteChats	
Descripción	Se encarga de eliminar todos los chats en los que participa el usuario.
Grupo	Chats
Método	Delete
Ruta	chat/chats
Parámetros	• userID (a través de la cabecera de autenticación)
Respuesta	Devuelve el mensaje “ok”.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.9 - Endpoint DeleteChat

DeleteChat	
Descripción	Se encarga de eliminar un chat concreto de la lista de chats del usuario.
Grupo	Chats
Método	Delete

Ruta	chat/
Parámetros	• chatID
Respuesta	Devuelve el mensaje “ok”.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.10 - Endpoint SendMessage

SendMessage	
Descripción	Se encarga de almacenar un mensaje en la base de datos y enviarlo al usuario destinatario.
Grupo	Mensajes
Método	Post
Ruta	message/sendMessage
Parámetros	• userID (a través de la cabecera de autenticación) • chatID • message
Respuesta	Devuelve el mensaje que se envió si la petición es correcta.
Respuesta en caso de error	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.11 - Endpoint GetMessages

GetMessages	
Descripción	Se encarga de obtener todos los mensajes de un chat del usuario.
Grupo	Mensajes
Método	Get
Ruta	message/messages
Parámetros	• userID (a través de la cabecera de autenticación) • chatID • offset (el último mensaje que tiene el usuario) • limit (el número de mensajes que se desea obtener)
Respuesta	Devuelve la lista de mensajes de un chat determiando.
Respuesta en caso de error	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.12 - Endpoint ReadMessages

ReadMessages	
Descripción	Se encarga de marcar como leído un mensaje y todos los que se han enviado anteriormente en ese chat.
Grupo	Mensajes
Método	Get

Ruta	message/readMessages
Parámetros	• userID (a través de la cabecera de autenticación) • chatID • messageID
Respuesta	Devuelve el mensaje “ok”.
Respuesta en caso de error	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.13 - Endpoint GetExperts

GetExperts	
Descripción	Se encarga de obtener la lista de psicólogos que hay disponibles para un determinado usuario.
Grupo	Usuarios
Método	Get
Ruta	user/experts
Parámetros	• userID (a través de la cabecera de autenticación) • problemType
Respuesta	Devuelve la lista de psicólogos disponibles para ese usuario teniendo en cuenta el tipo de problema que sufre.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400.

	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.14 - Endpoint PersonalInfo

PersonalInfo	
Descripción	Se encarga de obtener la información personal del cliente.
Grupo	Usuarios
Método	Get
Ruta	user/
Parámetros	• userID (a través de la cabecera de autenticación)
Respuesta	Devuelve la información personal del usuario cliente de la aplicación.
Respuesta en caso de error	• Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.15 - Endpoint UpdateInfo

UpdateInfo	
Descripción	Se encarga de actualizar la información personal del cliente.
Grupo	Usuarios
Método	Put
Ruta	user/

Parámetros	• userID (a través de la cabecera de autenticación) • username • email • problemType • password
Respuesta	Devuelve la información personal del cliente actualizada.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

Tabla 5.16 - Endpoint AddRating

AddRating	
Descripción	Se encarga de almacenar en la base de datos la valoración que se hecho el usuario de un psicólogo con el que mantiene una conversación.
Grupo	Usuarios
Método	Post
Ruta	user/addRating
Parámetros	• userID (a través de la cabecera de autenticación) • expertID (ID del psicólogo) • rating

Respuesta	Devuelve el rating que el usuario le ha establecido al psicólogo.
Respuesta en caso de error	• Petición incorrecta en caso de que falten datos. Código 400. • Error en caso de que se produzca un error durante el procesamiento de la petición. Error 500.

5.2.2 Socket.IO

Socket.IO es una librería de *JavaScript* para *Node.js* que permite una comunicación bidireccional en tiempo real entre cliente y servidor. Se basa principalmente en *websockets* pero también puede usar otras alternativas.

Se ha implementado un *socket* en el *Back-end* para permitir una comunicación rápida y segura en tiempo real entre cliente y servidor.

Una vez un usuario acceda a la aplicación móvil se establecerá una conexión con el *Back-end* a través de un *socket*. En el *Back-end* se he implementado la escucha de tres señales principales, para asegurar el correcto funcionamiento de la comunicación a través de *sockets*:

- Conexión: Se encargará de informar de que un nuevo cliente se ha conectado a la aplicación.

```
function connection(socket) {
  io = this;

  socket.use(md_socket.ensureAuth);

  socket.on('disconnect', disconnect);
  socket.on('pairing', pairing);
}
```

Figura 5.31 - Método para detectar la conexión de un cliente

- Desconexión: Informará al *Back-end* de que se ha producido una desconexión de un usuario de la aplicación móvil.

```
function disconnect() {
  const user = session.getUserBySocketID(this.id);
  if (!user) return;
  const last_session = Date.now();

  session.remove(this.id);
  user_service.update(user.id);
  this.broadcast.emit('user-event', {
    user: { id: user.id, last_session: last_session },
    type: 'disconnect'
  });
}
```

Figura 5.32 - Método para detectar la desconexión de un cliente

- Emparejamiento: Esta función se encargará de asociar la conexión mediante *socket* con la cuenta de usuario correspondiente a ese cliente.

```
function pairing(data, callback) {
  this.broadcast.emit('user-event', { user: { id: data.user.id }, type: 'connect' });
  session.add(this.id, data.user);
  if (callback) callback({ pairing: 'OK' });
}
```

Figura 5.33 - Método para emparejar un cliente

5.2.2.1 Middleware

Se ha diseñado un *middleware* para garantizar la seguridad y la integridad de las comunicaciones en tiempo real a través de *Socket.IO*.

Este *middleware* se implementa en todos los eventos del *socket* para que solamente puedan comunicarse con el *Back-end* los usuarios que previamente hayas iniciado sesión en la aplicación. Antes de que la petición llegué al controlador es capturada por el *middleware* que comprueba si lleva la cabecera de autorización con el *token JWT*. De no ser así devolverá un error 401 que indica que el cliente no se ha autenticado. Si existe la cabecera de autenticación se comprobará si el *token* es válido. Para ser considerado válido debe de poder desencriptarse con una clave secreta que solo conoce el *Back-end*, y que la fecha de validez no haya expirado aún.

```
exports.ensureAuth = function(packet, next) {
  if (!packet[1].user.token) {
    return next(new Error(401));
  }

  const payload = jwt.verify(packet[1].user.token.replace(/['"]+/g, ''));
  if (!payload) return next(new Error(401));

  packet[1].user = payload;

  next();
};
```

Figura 5.34 - Middleware de Socket.IO

Si el *token* es correcto se liberará la petición y esta ya será procesada por el controlador que le corresponda.

5.2.2.2 *Emparejamiento de usuarios*

El emparejamiento se utiliza para asociar la conexión mediante *socket* con la cuenta de usuario correspondiente a ese cliente. Cuando un nuevo usuario se conecte a través del *socket*, se ejecutará un evento denominado *pairing* con el identificador de ese usuario. Este método establecerá una relación entre el identificador de usuario y el identificador del *socket*, de esta forma se conoce en todo momento la lista de usuarios que están conectados en un momento determinado.

La lista de relaciones entre el identificador del *socket* y el del usuario necesita implementar una serie de operaciones que se detallan a continuación:

Tabla 5.17 - Lista de sockets: addUser

addUser	
Acción	Se encarga de añadir a la lista un nuevo usuario que se acaba de conectar mediante el *socket*.
Parámetros	• userID • socketID
Respuesta	Ninguna.

Tabla 5.18 - Lista de sockets: removeUser

removeUser	
Acción	Se encarga de eliminar un usuario de la lista cuando este se desconecta.
Parámetros	• socketID
Respuesta	Ninguna.

Tabla 5.19 - Lista de sockets: getUser

getUser	
Acción	Se encarga de obtener un usuario mediante su identificador de usuario.
Parámetros	• userID
Respuesta	Devuelve el usuario.

Tabla 5.20 - Lista de sockets: getUserBySocketID

getUserBySocketID	
Acción	Se encarga de obtener un usuario mediante su identificador de *socket*.
Parámetros	• socketID
Respuesta	Devuelve el usuario.

Tabla 5.21 - Lista de sockets: getUsers

getUsers	
Acción	Se encarga de obtener la lista completa de usuarios que están conectados en un momento determinado.
Parámetros	Ninguno.
Respuesta	Devuelve la lista de usuarios conectados.

5.3 Modelo predictivo con Machine Learning

Una parte muy importante durante el desarrollo del proyecto es la creación de un modelo predictivo que obtenga la polaridad de los mensajes de los usuarios víctimas. Este modelo se basa en el análisis de sentimientos y su uso es imprescindible para dos funciones principales:

- Detectar de grado de amenaza que sufre el usuario mediante un análisis de sentimientos de sus mensajes con el fin de asignar mayor prioridad a los usuarios que sufren mayor grado de amenaza.
- Reflejar el grado de progreso de un usuario mediante un análisis de sentimientos de sus mensajes. Esto nos ofrece estadísticas de como un usuario mejora o empeora mientras utiliza nuestra aplicación.

Para la creación de este modelo han sido necesarios una serie de *scripts* en lenguaje *Python* que realizaran pequeñas tareas de selección, limpieza y procesamiento de datos. Posteriormente se han creado diferentes modelos predictivos con el fin de compararlos y escoger el modelo que mejor resultados ofrezca.

Para crear un modelo predictivo robusto es necesario seguir una serie de pasos obligatorios que se describen a continuación.

5.3.1 Selección de un conjunto de datos

En primer lugar, se ha realizado una búsqueda en Internet de conjuntos de datos constituidos por oraciones que estén clasificadas por su polaridad. Es decir, es necesario que cada oración tenga una polaridad asociada que puede ser positiva, negativa o neutra. Puesto que la aplicación está destinada a los usuarios de España, y el modelo se aplicará a oraciones en español, el conjunto de datos tiene que estar en español también.

Se han analizado varias fuentes de datos de diversa precedencia llegando a la conclusión de que los datos de *Twitter* son muy buenos. Debido a que en esta red social la gente plasma de forma muy clara sus sentimientos es posible encontrar un conjunto de datos donde se clasifique la polaridad de los *tweets*.

Se ha utilizado un conjunto de datos categorizado con las categorías positivo, negativo y neutro. Este conjunto es un corpus de *TASS*.[53]

```
<tweet>
 <tweetid>142378325086715906</tweetid>
 <user>camilaruiiz</user>
 <content>Hoy me he levantado de mal humor y no tengo ganas de hacer nada</content>
 <date>2015-12-02T00:03:32</date>
 <lang>es</lang>
 <sentiments>
  <polarity><value>N</value></polarity>
 </sentiments>
 <topics>
  <topic>política</topic>
 </topics>
</tweet>
<tweet>
```

Figura 5.35 - Tweet etiquetado

TASS es un Taller de Análisis de Sentimientos en español organizado cada año por la Sociedad Española del Procesado del Lenguaje Natural. Ha sido necesario ponerse en contacto con la asociación para obtener el corpus. Los archivos del corpus están en formato *XML* y contienen miles de *tweets* en español junto con su sentimiento.[54]

Se ha comprobado que todos los *tweets* tienen una polaridad asociada y que además el conjunto está correctamente balanceado, es decir se tiene que mantener una proporción equitativa de *tweets* positivos, negativos y neutros. En total hay 68017 *tweets*.

5.3.2 Limpieza de datos

Una vez seleccionado el conjunto de datos de *TASS* será necesario aplicarle una serie de tareas de limpieza de datos con el fin de mejorar la calidad del conjunto y que sea apto para el entrenamiento de un modelo predictivo.

La limpieza de datos es una de las tareas importantes que normalmente se deben llevar a cabo para que el conjunto de datos se pueda usar de forma eficaz para el aprendizaje automático. Los datos sin procesar son a menudo ruidosos no confiables y es posible que les falten valores. El uso de estos datos para entrenar un modelo puede producir resultados engañosos.

En primer lugar, se ha convertido el conjunto de datos a formato *CSV* con el fin de procesarlo mejor en el futuro y trabajar con la librería *Pandas*. Durante este proceso se han eliminado una serie de características que no eran interesantes y no servían para entrenar el modelo.

```
general_tweets_corpus_train = pd.DataFrame(columns=('content', 'polarity', 'agreement'))
xml = objectify.parse('./xml/general-train-tagged.xml')
tweets = xml.getroot().getchildren()

for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'],
                       [tweet.content.text, tweet.sentiments.polarity.value.text, tweet.sentiments.polarity.type.text]))
        row_s = pd.Series(row)
        row_s.name = i
        general_tweets_corpus_train = general_tweets_corpus_train.append(row_s)
general_tweets_corpus_train.to_csv('./csv/general-train-tagged.csv', index=False, encoding='utf-8')
```

Figura 5.36 - Limpieza del dataset: Paso 1

Una vez creado el archivo *CSV* ya se puede proceder con la limpieza de datos utilizando la librearía *Pandas*.

Lo primero es eliminar los *tweets* duplicados ya que no tiene sentido tener una oración repetida y puede empeorar el modelo.

```
tweets_corpus = tweets_corpus.drop_duplicates()
tweets_corpus.shape
```

Figura 5.37 - Limpieza del dataset: Paso 2

Una vez eliminados los *tweets* duplicados el tamaño del conjunto disminuye levemente pero no demasiado.

Una de las características es "*agreement*", que representa una indicación del nivel de acuerdo o desacuerdo del sentimiento expresado dentro del contenido, con dos valores posibles: acuerdo o desacuerdo. Esto se utiliza en aquellos *tweets* con polaridad neutra debido a una mezcla de palabras positivas y negativas. Por lo tanto, es necesario eliminar los *tweets* en los que no exista un acuerdo en la polaridad ya que pueden generar ruido.

```
tweets_corpus = tweets_corpus.query('agreement != "DISAGREEMENT"')
tweets_corpus.shape
```

Figura 5.38 - Limpieza del dataset: Paso 3

El siguiente paso será eliminar todas las *URLs* que están contenidas en los *tweets* ya que no tiene sentido utilizarlas.

```
tweets_corpus = tweets_corpus[-tweets_corpus.content.str.contains('^http.*$')]
tweets_corpus.shape
```

Figura 5.39 - Limpieza del dataset: Paso 4

Una vez limpiado el conjunto de datos es necesario guardarlo en un nuevo archivo con el formato *CSV*.

5.3.3 Entrenamiento de modelos predictivos

En esta fase se realizará un estudio de diferentes algoritmos de *Machine Learning* con el objetivo ver las ventajas frente a otros.

Se ha optado por utilizar modelos de clasificación binaria que clasifiquen las oraciones en dos categorías:

- Negativas
- No negativas

La clase negativa se representará con 1 y la clase no negativa con 0.

```
tweets_corpus['polarity_bin'] = 0
tweets_corpus.loc[tweets_corpus['polarity'].isin(['N', 'N+']), 'polarity_bin'] = 1
tweets_corpus['polarity_bin'].value_counts(normalize=True)
```

Figura 5.40 - Entrenamiento de modelos: Paso 1

El siguiente paso será utilizar *NLTK* y otras librerías de procesamiento de lenguaje natural con el fin de limpiar las oraciones durante la creación de los modelos. Es importante eliminar los signos de puntuación y las *"stop words"*, que son las palabras más frecuentes en español y pueden perjudicar al modelo.

```
nltk.download("stopwords")
spanish_stopwords = stopwords.words('spanish')

non_words = list(punctuation)

#Simbolos de interrogación y admiración invertidos que solo hay en español.
non_words.extend(['¿', '¡'])
non_words.extend(map(str,range(10)))
```

Figura 5.41 - Entrenamiento de modelos: Paso 2

Después se implementará un método propio para separar y procesar las palabras de una oración con el fin de mejorar la extracción de características. Esta función se encargará de eliminar las *stop words* y los símbolos de puntuación, así como de utilizar la raíz semántica de cada palabra en lugar de la palabra. Utilizar la raíz de la palabra en lugar de la palabra ofrece mejores resultados en el entrenamiento de los modelos debido a que un masculino y un femenino de una palabra se asociarán a la misma raíz semántica, lo mismo ocurre con los singulares y plurales y las conjugaciones de los verbos.

```
stemmer = SnowballStemmer('spanish')
def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

def tokenize(text):
    # remove non letters
    text = ''.join([c for c in text if c not in non_words])
    # tokenize
    tokens =  word_tokenize(text)

    # stem
    try:
        stems = stem_tokens(tokens, stemmer)
    except Exception as e:
        print(e)
        print(text)
        stems = ['']
    return stems
```

Figura 5.42 - Entrenamiento de modelos: Paso 3

El siguiente paso es extraer las características de cada oración mediante un método que construye una matriz de recuentos de las palabras de cada oración. Esta función se denomina *CountVectorizer*. [55]

```
vectorizer = CountVectorizer(
    analyzer = 'word',
    tokenizer = tokenize,
    lowercase = True,
    stop_words = spanish_stopwords
)
```

Figura 5.43 - Entrenamiento de modelos: Paso 4

En este método se debe especificar la unidad de la oración a analizar, el método que se utilizará para separar cada unidad, la lista de *stop words* y si se desea utilizar solo letras minúsculas.

El siguiente paso es probar diferentes modelos y obtener los mejores parámetros de entrenamiento de cada uno. Para esto se utilizará un método de ajuste y puntuación denominado *GridSearchCV*. Este método recibirá como parámetros el tipo de modelo que se desea utilizar, el rango de parámetros de que desea configurar y la métrica de evaluación.

El rango de parámetros que se va a utilizar para obtener los mejores es el siguiente:

```
parameters = {
    'vect__max_df': (0.5, 1.9),
    'vect__min_df': (10, 20,50),
    'vect__max_features': (500, 1000),
    'vect__ngram_range': ((1, 1), (1, 2)),  # unigrams or bigrams
    'cls__C': (0.2, 0.5, 0.7),
    'cls__loss': ('hinge', 'squared_hinge'),
    'cls__max_iter': (500, 1000)
}
```

Figura 5.44 - Entrenamiento de modelos: Paso 5

Una vez definido el rango de parámetros falta crear cada uno de los modelos y buscar los mejores parámetros de cada uno. Para ello es necesario crear las *pipelines* que combinen el método de extracción de características con la creación propia del modelo. De esta forma cuando se entrene un modelo, automáticamente se extraerán las características y luego se realizará el entrenamiento.

A continuación, se muestra cómo se obtienen los mejores parámetros para un modelo que utiliza el estimador *LinearSVC*.

```
pipeline = Pipeline([
    ('vect', vectorizer),
    ('cls', LinearSVC()),
])


grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1 , scoring='roc_auc')
grid_search.fit(tweets_corpus['content'].values.astype('U'), tweets_corpus['polarity_bin'])
```

Figura 5.45 - Entrenamiento de modelos: Paso 6

Este proceso se repetirá con otros estimadores distintos como son los siguientes:

- Regresión logística
- Clasificadores bayesianos
- Arboles de decisión
- Clasificadores KNN

Una vez finalice en entrenamiento de *GridSearchCV* se pueden obtener los mejores parámetros de entrenamiento.

```
grid_search_LinearSVC.best_params_
```

Figura 5.46 - Entrenamiento de modelos: Paso 7

5.3.4 Pruebas de validación cruzada y comparativa de modelos

Una vez determinados los clasificadores que se van a utilizar y los mejores parámetros de cada uno de ellos, es necesario crear cada uno de los modelos con esos parámetros y entrenarlos. Posteriormente realizarán las pruebas necesarias con el fin de determinar cuál de los modelos predictivos ofrece mejores resultados frente al resto.

Se han utilizado pruebas de validación cruzada para evaluar cada uno de los modelos con diferentes métricas. La validación cruzada es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba.

Las pruebas se realizarán sobre cada uno de los estimadores nombrados en el punto anterior.

En el caso de la regresión logística, se creará una *pipeline* que combinará el algoritmo de extracción de características con el estimador *LogisticRegression*, y se utilizarán los parámetros obtenidos en el punto anterior para poder crear el mejor modelo posible.

```
pipeline_LogisticRegression = Pipeline([
    ('vect', CountVectorizer(
            analyzer = 'word',
            tokenizer = tokenize,
            lowercase = True,
            stop_words = spanish_stopwords,
            min_df = 20,
            max_df = 0.5,
            ngram_range=(1, 1),
            max_features=1000
            )),
    ('cls', LogisticRegression(C=.5,max_iter=500,multi_class='ovr',
             random_state=None,
             penalty='l2',
             tol=0.0001
             )),
])
```

Figura 5.47 - Pruebas de validación: Paso 1

Para el clasificador *SVM* se creará una *pipeline* que combinará el algoritmo de extracción de características con el estimador *LinearSVC*, y se utilizarán los parámetros obtenidos en el punto anterior para poder crear el mejor modelo posible.

```
pipeline_LinearSVC = Pipeline([
    ('vect', CountVectorizer(
            analyzer = 'word',
            tokenizer = tokenize,
            lowercase = True,
            stop_words = spanish_stopwords,
            min_df = 20,
            max_df = 0.7,
            ngram_range=(1, 1),
            max_features=1000
            )),
    ('cls', LinearSVC(C=.2, loss='squared_hinge',max_iter=500,multi_class='ovr',
             random_state=None,
             penalty='l2',
             tol=0.0001
             )),
])
```

Figura 5.48 - Pruebas de validación: Paso 2

Como se puede observar en la figura 5.48 lo parámetros son muy parecidos en ambos estimadores.

El siguiente paso sería realizar las pruebas de validación cruzada para ambos estimadores. Para ello es necesario definir la métrica de evaluación y el número de iteraciones que se van a realizar.

```
scores = cross_val_score(
    pipeline_estimator,
    corpus_data_features_nd[0:len(tweets_corpus)],
    y=tweets_corpus.polarity_bin,
    scoring='roc_auc',
    cv=5
    )

scores.mean()
```

Figura 5.49 - Pruebas de validación: Paso 3

Para realizar el resto de pruebas es necesario separar el conjunto de datos en dos partes para poder realizar el entrenamiento con cada uno de los modelos. Estas dos partes son:

- Datos de entrenamiento
- Datos de prueba

Los datos de prueba constituirán un 20% del total del conjunto de datos y se realizará de forma aleatoria con una determinada *semilla*.

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
```

Figura 5.50 - Pruebas de validación: Paso 4

Después de tener el conjunto de datos de entrenamiento y el de test se procede a entrenar ambos modelos.

```
pipeline_LogisticRegression.fit(X_train, y_train)
prediction_LogisticRegression = pipeline_LogisticRegression.predict(X_test)

pipeline_LinearSVC.fit(X_train, y_train)
prediction_LinearSVC = pipeline_LinearSVC.predict(X_test)
```

Figura 5.51 - Pruebas de validación: Paso 5

Una vez que haya finalizado el entrenamiento de ambos modelos es necesario comparar los resultados obtenidos. Para ello se utilizarán la función *classification_report*, que proporciona el resultado de varias métricas, y *accuracy_score*, que proporciona la exactitud. También se mostrará una matriz de confusión.

```
print(confusion_matrix(y_test, prediction))
print(classification_report(y_test, prediction))
print(accuracy_score(y_test, prediction))
```

Figura 5.52 - Pruebas de validación: Paso 6

6. Resultados

Una vez concluida la fase de desarrollo, se ha puesto la aplicación en producción. A continuación, se muestran los resultados obtenidos, tanto de la aplicación móvil destinada a usuarios víctimas, como del conjunto de *scripts* encargados de la creación, el entrenamiento y la aplicación del modelo predictivo de *Machine Learning*. Se utilizan capturas de pantalla para mostrar de forma clara los resultados obtenidos.

6.1 Aplicación destinada a usuarios víctimas

Una vez el usuario instale la aplicación en su *smartphone*, esta se mostrará en su menú de aplicaciones con el nombre "Connect" y el logo original de la aplicación.

Figura 6.1 - Logo de Connect

Cuando la abra la aplicación por primera vez se mostrará un *splash* mientras la aplicación se prepara para iniciar.

Figura 6.2 - Splash de Connect

Una vez la aplicación esté lista, se mostrará la interfaz de inicio de sesión, donde el usuario deberá introducir sus credenciales, en caso de que las tenga, para poder acceder a la aplicación. Estas credenciales son:

- Nombre de usuario o email
- Contraseña.

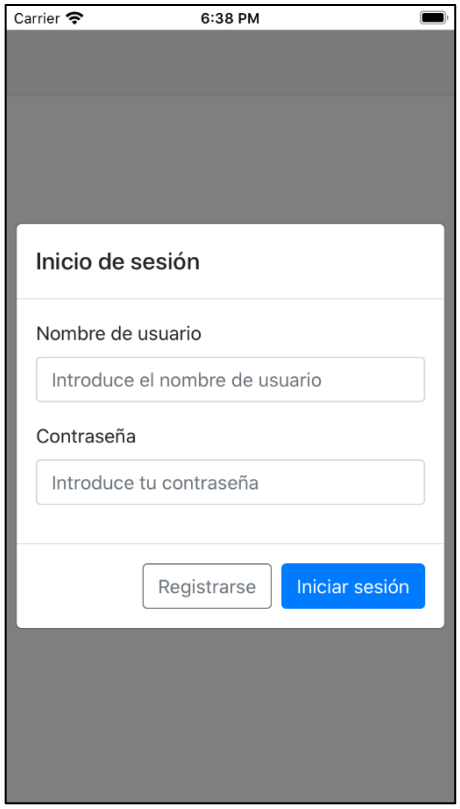


Figura 6.3 - Interfaz de inicio de sesión

Si el usuario no tiene una cuenta, deberá crear una previamente. Para crear una cuenta deberá pulsar “Registrarse” y accederá a la interfaz de registro.

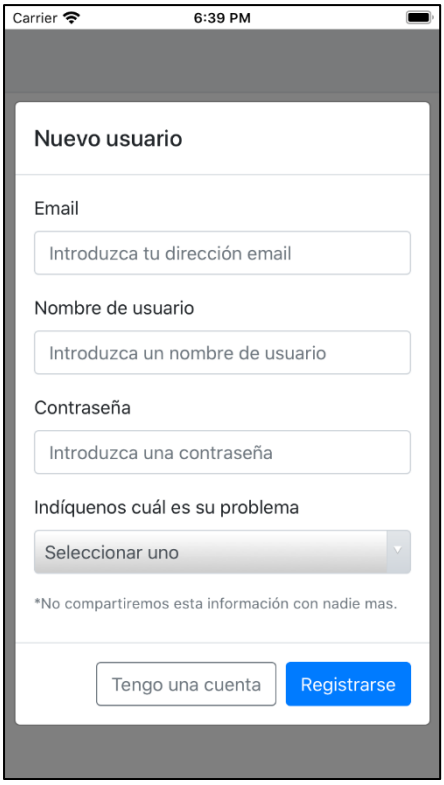


Figura 6.5 - Interfaz de registro

Será necesario rellenar los campos email, nombre de usuario, tipo de problema y contraseña para poder crear una cuenta de usuario.

Figura 6.4 - Interfaz de registro completada

Es importante especificar correctamente el tipo de problema que sufre el usuario porque los psicólogos con los que se comunicará serán especialistas en ese problema en concreto.

Una vez el usuario haya iniciado sesión o se haya registrado accederá a la interfaz principal de la aplicación, donde se muestra la lista de chats en los que participa.

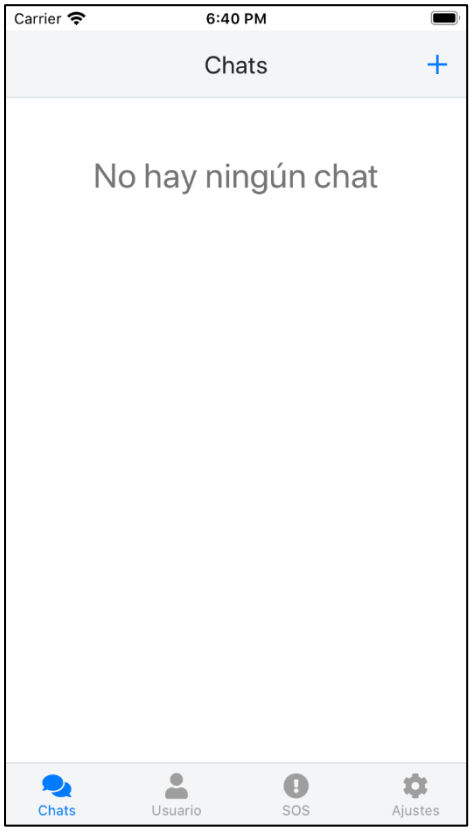


Figura 6.6 - Interfaz de visualización de chats

Para crear un nuevo chat el usuario debe pulsar el botón que se muestra en la parte superior derecha.

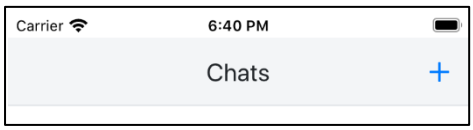


Figura 6.7 - Botón para añadir chat

Una vez el usuario pulse este botón se mostrará la lista de psicólogos disponibles con los que puede iniciar un chat, junto con su valoración general y el número de valoraciones que tiene en total.

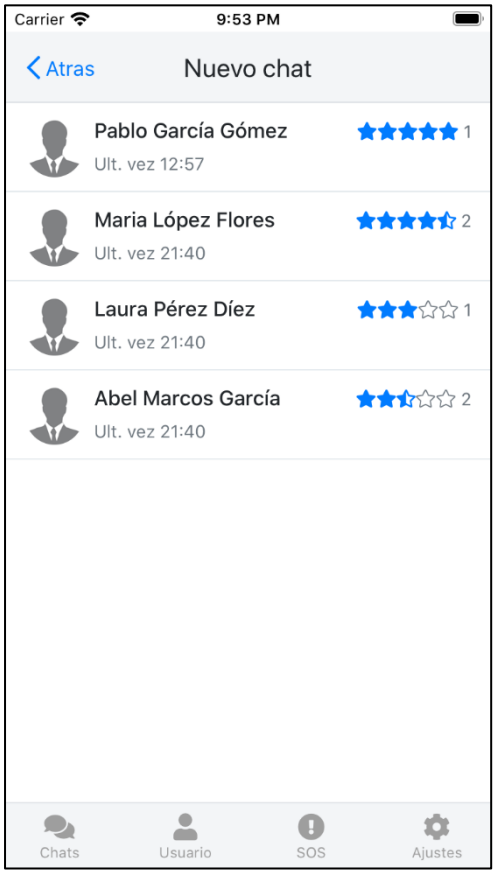


Figura 6.9 - Interfaz para mostrar los psicologos

Cuando se seleccione al psicólogo con el que se desea iniciar una conversación, se mostrará un modal para confirmar esta acción.

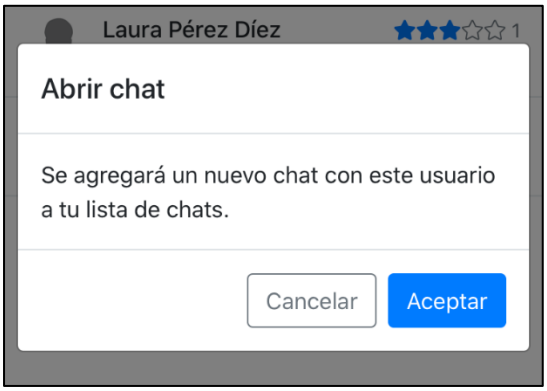


Figura 6.8 - Modal para crear un chat

Una vez el usuario haya confirmado esta acción, se añadirá un chat con el psicólogo escogido a la interfaz de chats del usuario.

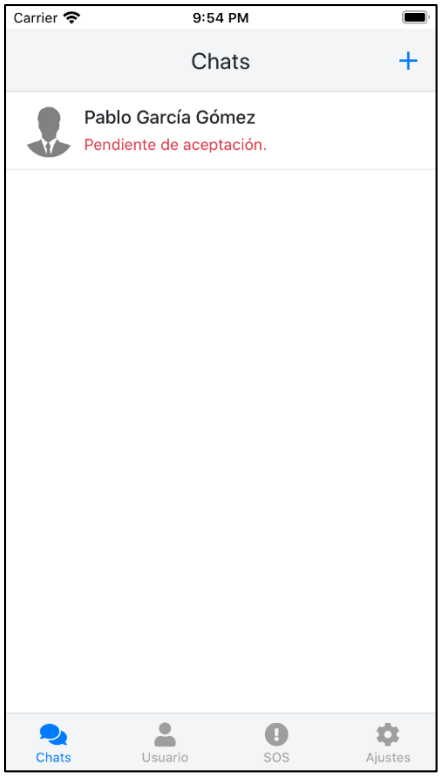


Figura 6.11 - Chat pendiente de aceptación

Será necesario que el psicólogo acepte la invitación del usuario para iniciar un chat. Una vez sea aceptada, el usuario ya podrá comunicarse de forma normal con el psicólogo.

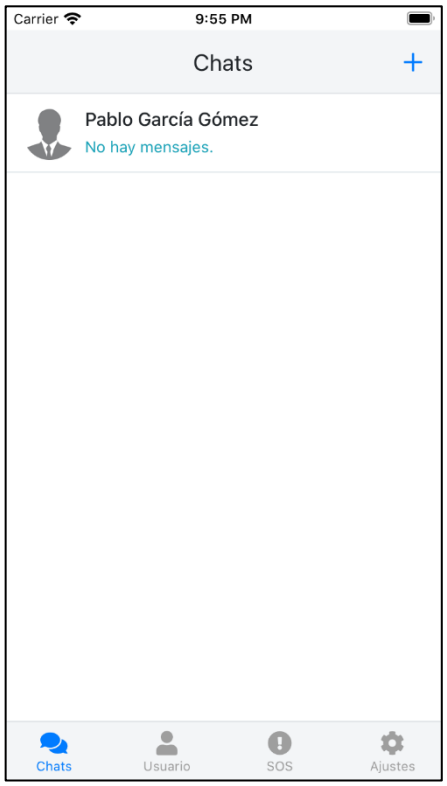


Figura 6.10 - Chat aceptado

Al pulsar sobre un determinado chat, se accederá a la interfaz que muestra un chat concreto, junto con sus mensajes y la información del psicólogo.

Desde esta interfaz el usuario puede escribir un mensaje y enviarlo, de forma bastante intuitiva.

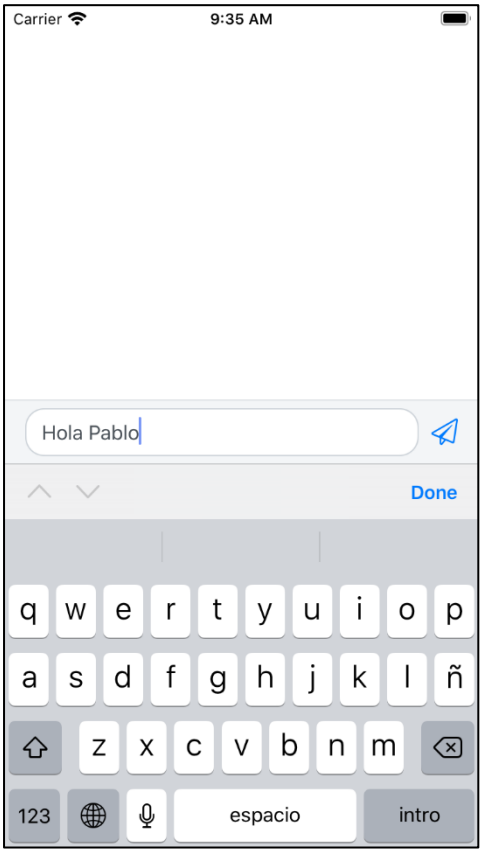


Figura 6.12 - Interfaz para mostrar un chat

Cuando el usuario acabé de redactar su mensaje y lo envié, este se mostrará en la interfaz en forma de lista junto al resto de mensajes.

En esta interfaz también se puede observar la última conexión del psicólogo, indicando si está en línea en ese momento.

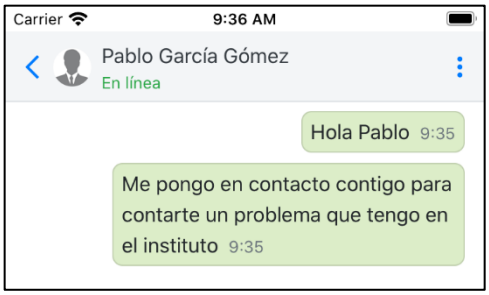


Figura 6.13 - Mensajes del usuario en el chat

Los mensajes del usuario se muestran en color verde y alineados hacia la derecha, mientras que los del psicólogo se muestran en color gris y a la izquierda.

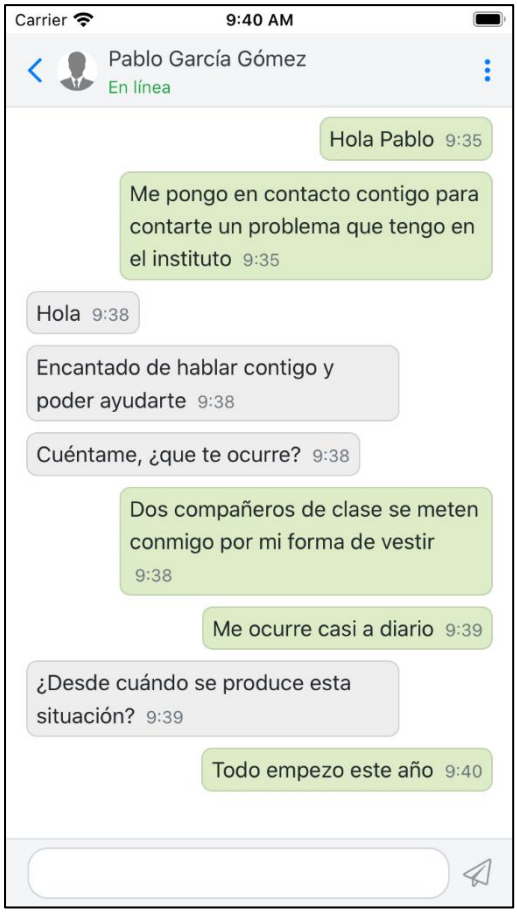


Figura 6.14 - Conversación con un psicólogo

En la parte superior derecha de esta interfaz se muestra un botón para acceder a un menú de opciones. Desde este menú se puede realizar una búsqueda de un campo de texto en el chat y eliminar el propio chat.

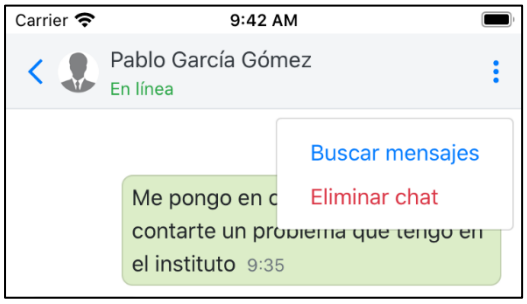


Figura 6.15 - Menú de opciones en un chat

Al pulsar “Buscar mensajes” se mostrará un *input* para introducir la búsqueda. Esta búsqueda se realizará en tiempo de ejecución.

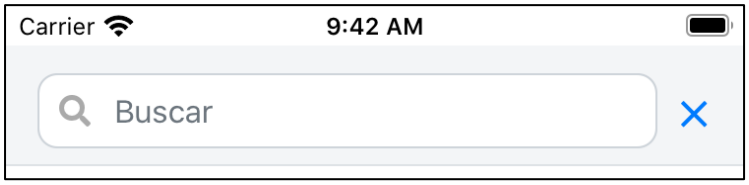


Figura 6.16 - Input para buscar mensajes

Cuando se introduzca un campo de texto, se realizará una búsqueda de todos los mensajes que contienen ese campo de texto, sin distinguir entre letras minúsculas o mayúsculas, y se mostrarán en la interfaz únicamente esos mensajes.

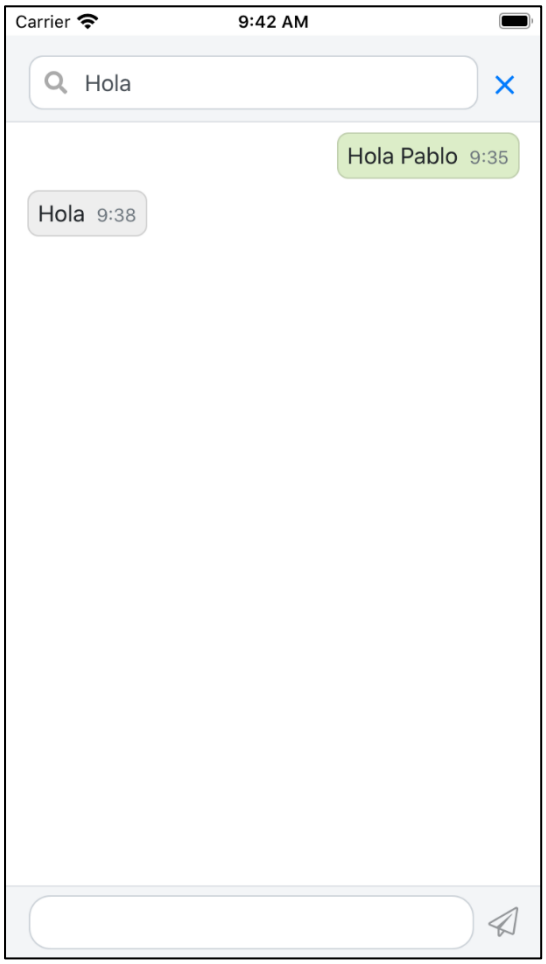


Figura 6.17 - Búsqueda en un chat

Otra de las funcionalidades de la aplicación es la visualización de la información de un psicólogo con el que el usuario mantiene una conversación. Cuando el usuario pulsa sobre el nombre del psicólogo, se muestra un modal con la información de este. En el modal que se despliega se muestra el nombre completo del psicólogo, su última conexión y su valoración general, así como el número de usuarios que le han valorado.

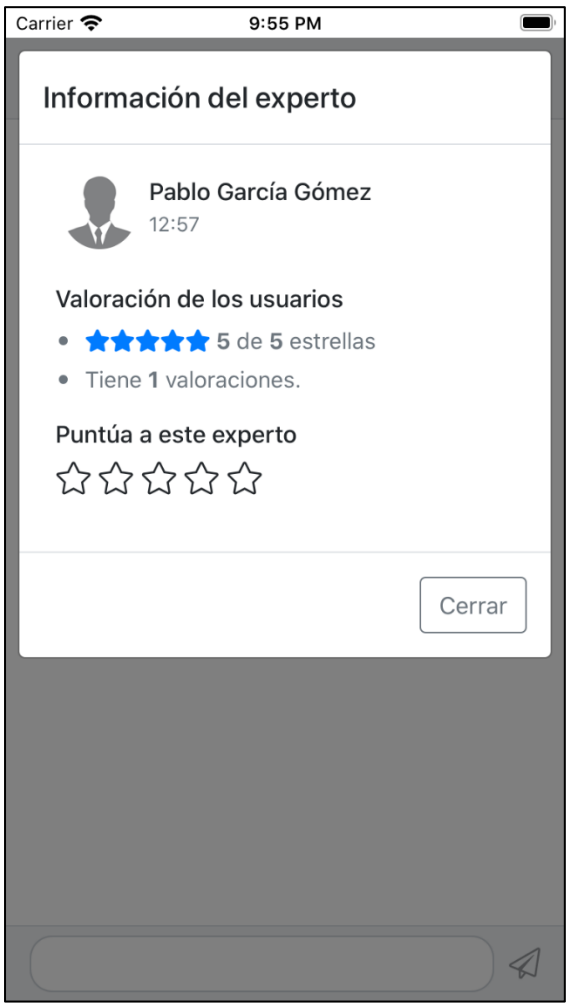


Figura 6.18 - Modal de información del psicólogo

Desde este modal también es posible realizar una valoración al psicólogo. Para ello se ha de seleccionar el número de estrellas con el que el usuario desea valorar al psicólogo. Una estrella sería la peor valoración posible y cinco la mejor.

Una vez se haya realizado la valoración al psicólogo, se mostrará la valoración que acaba de hacer el usuario y se actualizará la valoración global en caso de que haya cambiado.

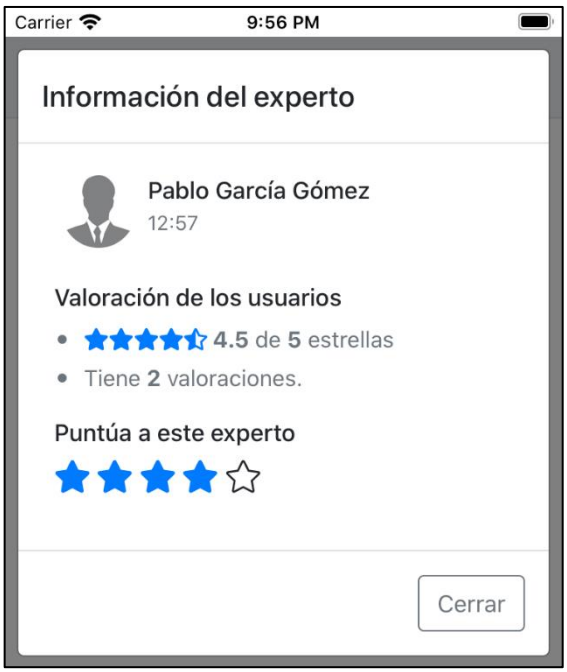
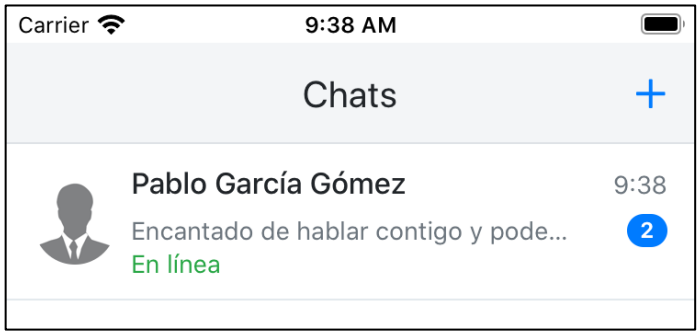


Figura 6.19 - Valoración de un psicólogo

Desde cualquier interfaz que se muestre en la aplicación, tras iniciar sesión, se podrá visualizar una notificación que indica el número de chats con mensajes sin leer.

Figura 6.20 - Notificación de chats con mensajes sin leer

Cuando se pulse sobre el botón “Chats”, que se encuentra en la parte inferior izquierda, se accederá de nuevo a la interfaz que muestra la lista de chats. Desde esta interfaz se puede ver visualizar también, el número de mensajes sin leer de un determinado chat.



Para acceder a la interfaz de opciones de usuario se debe pulsar el botón “Usuario”, que se encuentra en la parte inferior izquierda, junto al botón “Chats”.

Desde esta interfaz se podrán modificar algunos datos de la cuenta de usuario, como son el nombre de usuario, el email y la contraseña, pero no será posible modificar el tipo de problema. Además, se muestran dos funciones más que son “Cerrar sesión” y “Eliminar Cuenta”. “Eliminar cuenta” se encarga de borrar todos los datos asociados a esa cuenta de usuario, incluyendo las conversaciones con psicólogos.

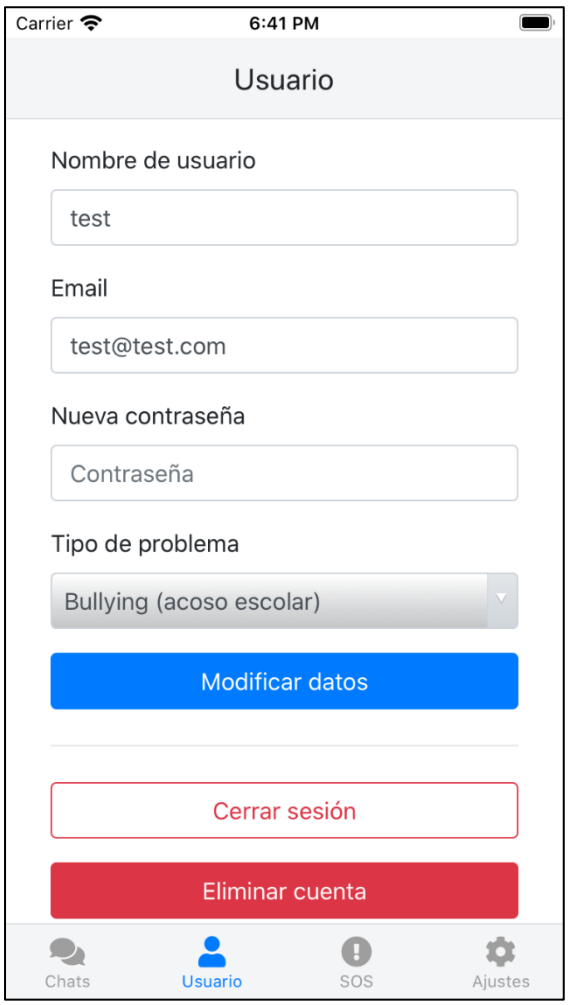


Figura 6.21 - Interfaz de opciones de usuario

Una de las funcionalidades más importantes de la aplicación es el botón del pánico, que se encarga de emitir un aviso a la policía con la ubicación del usuario. Para hacer uso de esta función se debe pulsar el botón “SOS”, que se encuentra en la parte inferior, junto al botón “Usuario”.

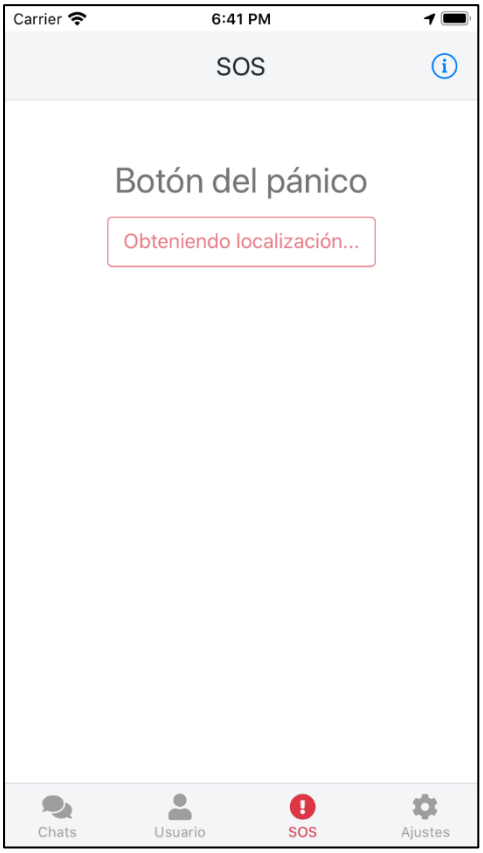


Figura 6.22 - Interfaz de botón del pánico

Si es la primera vez que se utiliza esta función, se solicitará la ubicación al usuario para poder hacer uso de esta funcionalidad.

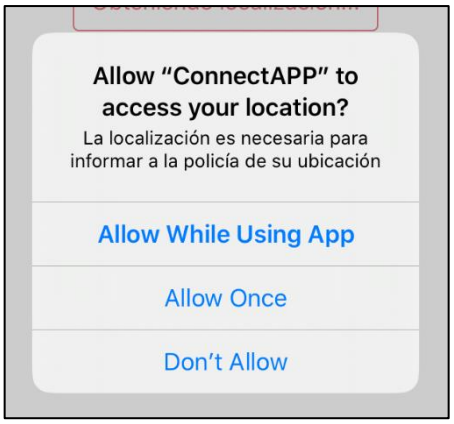


Figura 6.23 - Solicitud de ubicación

Si el usuario se niega a permitir la ubicación, no podrá utilizar esta función y se le mostrará el siguiente mensaje.

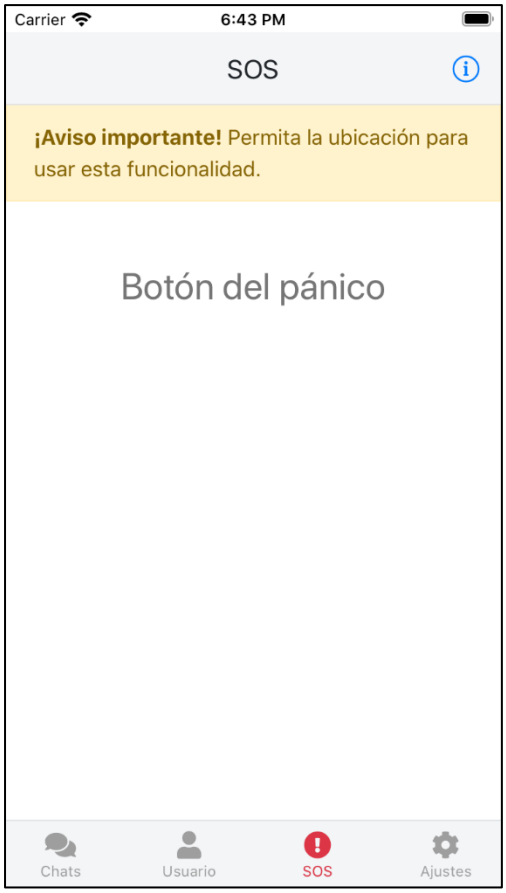


Figura 6.24 - Mensaje de obtención de ubicación no permitida

Si, por el contrario, el usuario permite la ubicación, se mostrará el botón “Estoy en peligro” encargado de enviar la localización a la policía.

Figura 6.25 - Botón del pánico

Al pulsar este botón se mostrará un modal informando de que se ha enviado la localización a la policía.

Figura 6.26 - Modal de aviso a la policía

En la interfaz del botón del pánico se muestra un botón de ayuda en la parte superior derecha, que muestra un modal de información al pulsarlo.

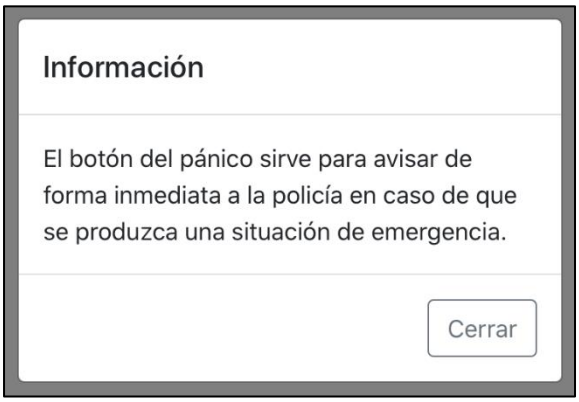


Figura 6.27 - Modal de ayuda de botón del pánico

Otra de las funcionalidades importantes de *Connect* es la ocultación de la aplicación, mostrando una simple calculadora al abrir la aplicación.

Para activar esta funcionalidad hay que dirigirse a la interfaz de ajustes, pulsando el botón “ajustes”, en la parte inferior derecha.

Por defecto, esta funcionalidad esta desactivada, y es el usuario quien la debe activar cuando lo desee.

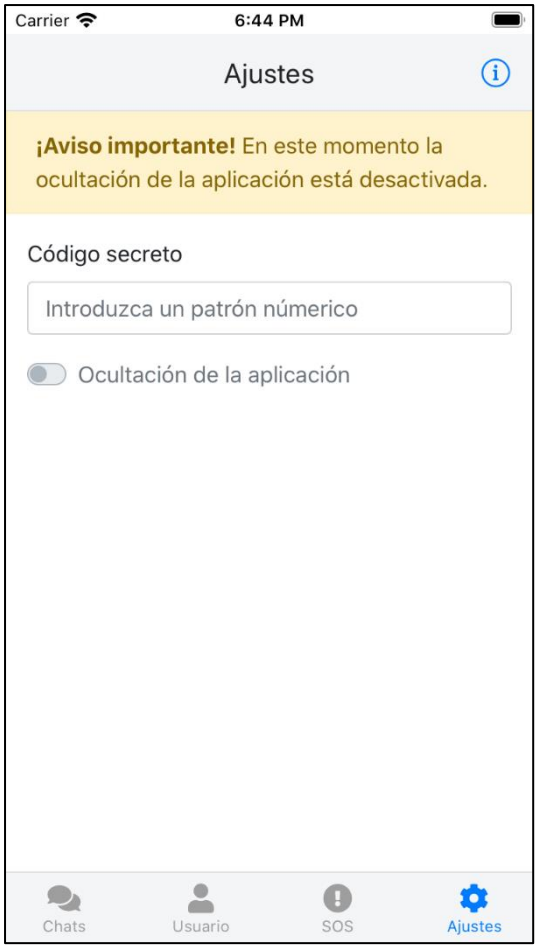


Figura 6.29 - Interfaz de ajustes

En la parte superior derecha se muestra un botón de ayuda que muestra una serie de instrucciones acerca de la ocultación de la aplicación.

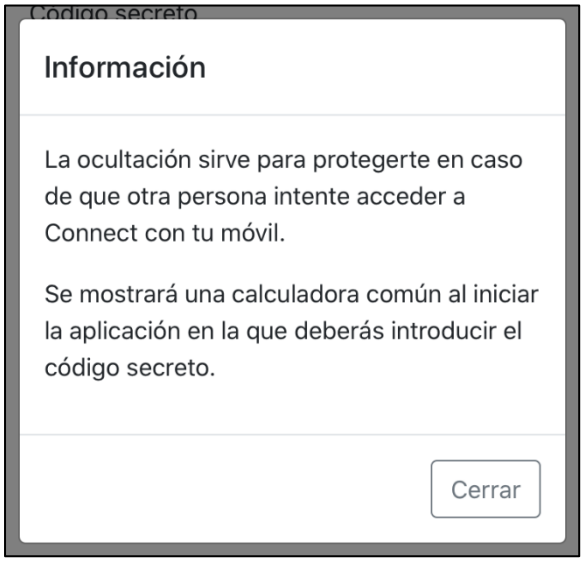


Figura 6.28 - Modal de ayuda para la ocultación

Para activar la ocultación, primero será necesario elegir un código secreto, que será el que se introduzca en la calculadora para mostrar la aplicación real.

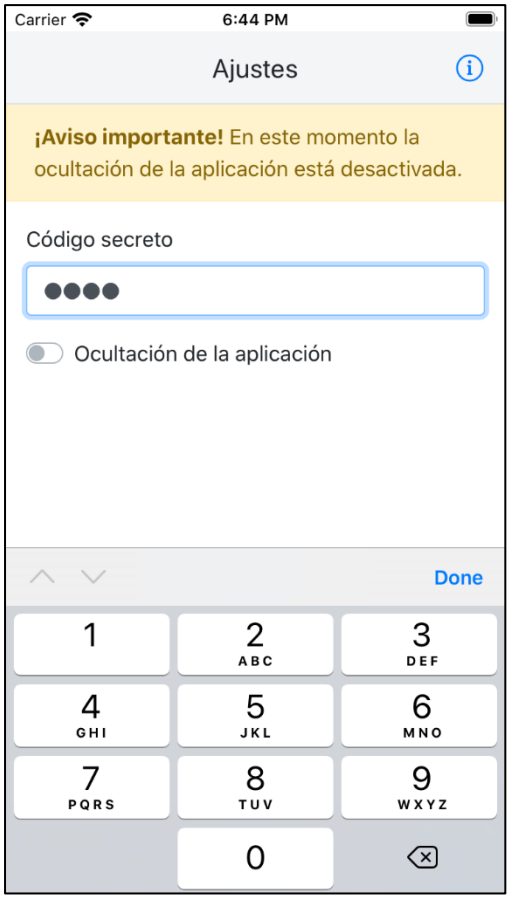


Figura 6.30 - Introducción del código secreto

El siguiente paso será pulsar sobre el *switch* para activar la ocultación de la aplicación. Para desactivarla bastaría con pulsarlo de nuevo.

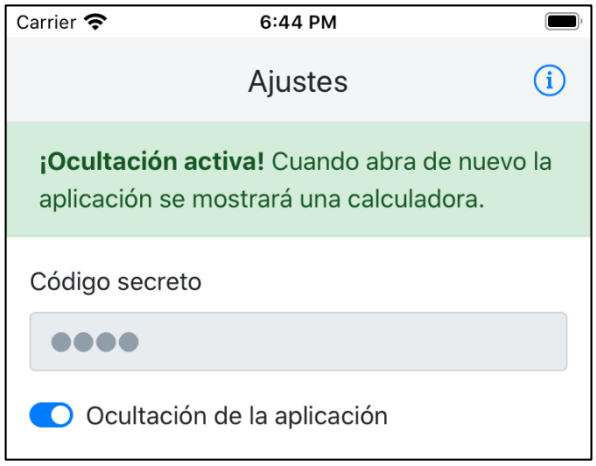


Figura 6.31 - Activación de la ocultación

Si la ocultación esta activa, el icono de la aplicación no será el original, sino un icono de una calculadora. Este icono se mantendrá mientras la ocultación este activada.

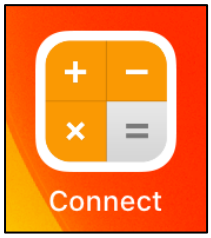


Figura 6.32 - Logo falso de Connect

Al abrir la aplicación se mostrará una calculadora que realiza operaciones matemáticas simples y solo mostrará la aplicación real si se introduce el código secreto.

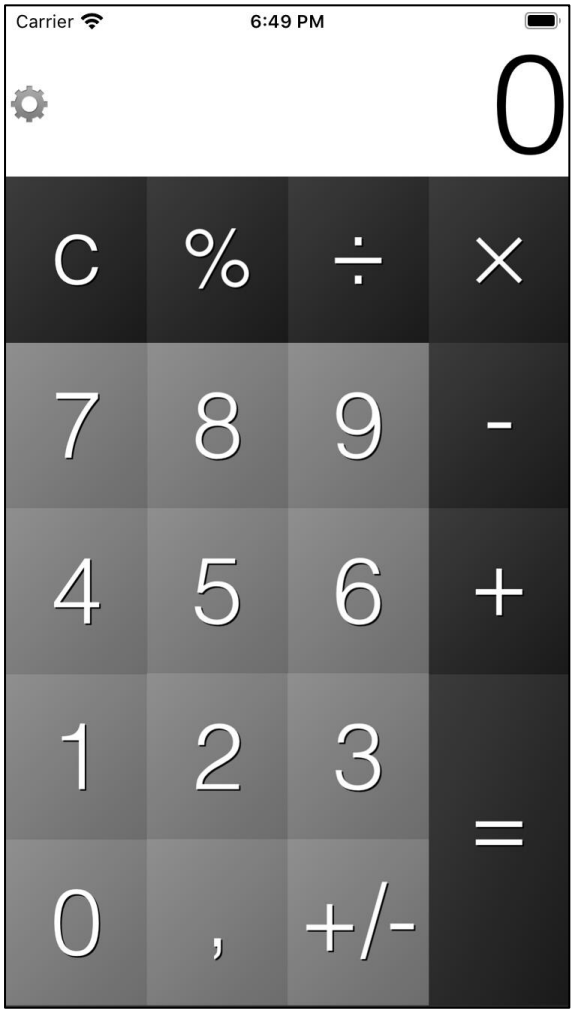


Figura 6.33 - Calculadora para ocultar la aplicación

Una vez se introduzca el código secreto en la calculadora, se mostrará de forma inmediata la aplicación real.

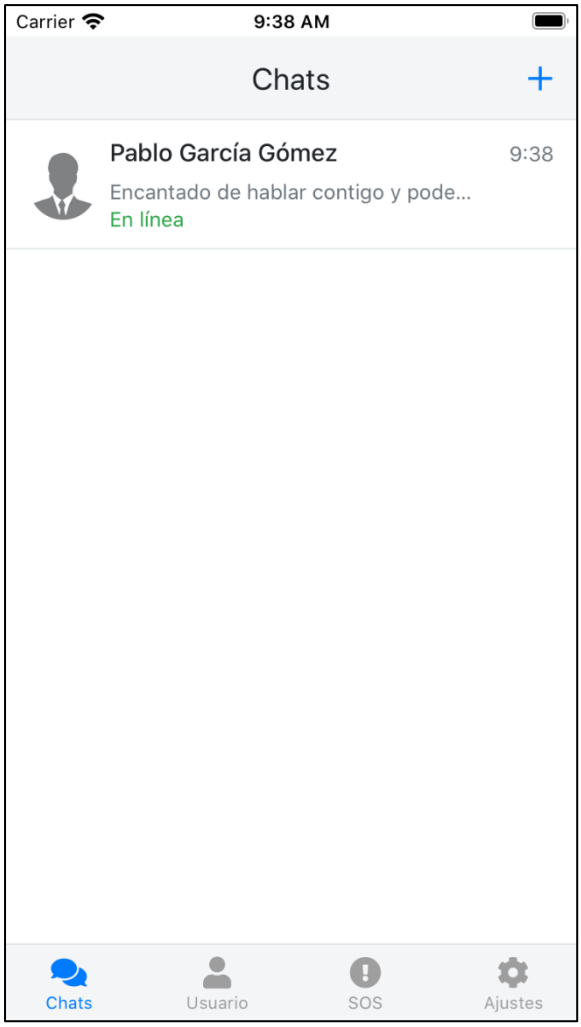


Figura 6.34 - Aplicación real

6.2 Modelo predictivo de Machine Learning

Para crear un modelo predictivo que sea capaz de calcular la polaridad de los mensajes de los usuarios han sido necesarias algunas tareas previas, que se han sido realizadas mediante *scripts* en *Python*. Estas tareas comprenden la limpieza, el procesado, la creación de diferentes modelos y, por último, las pruebas de validación.

Se ha utilizado un conjunto de datos de *Twitter*, recopilados por *TASS*[53], en los que se clasifica la polaridad de cada *tweet*.

```
In [11]: tweets_corpus.sample(10)
```

Out[11]:

	content	polarity	agreement
39564	. @elmundo Mirad, enfermos de la unidad de sal...	N	NaN
7012	Se puede llamar zorra a una periodista o a cua...	N	NaN
6314	#ElCambioAndaluz Arenas planteará anticipos re...	NONE	AGREEMENT
40584	Tengo q escribir mi art económico de mañana pa...	P+	NaN
6438	@SSantiagosegura tintín...he perfao...no atisb...	P+	AGREEMENT
45626	Con Gemma del Corral en las jornadas de #mujer...	NONE	NaN
8945	Hoy m acompañan en @DHAM_13tv @sanchezcastejo...	NONE	NaN
59684	¿Alguien puede ponerme link del artículo de ho...	P+	NaN
50298	A recoger a mi peque que llevo desde el vierne...	NONE	NaN
2869	Rajoy apuesta por la bajada general de salario...	N	NaN

Figura 6.35 - Dataset TASS

6.2.1 Limpieza de datos

El conjunto de datos inicial constaba de 68017 registros, pero después de aplicarle una serie de tareas de limpieza de datos, el número de registros es de 66550. Algunas de las tareas de limpieza que se aplicaron fueron:

- Eliminación de duplicados.
- Eliminación de registros en los que no está clara su polaridad.
- Eliminación de registros que contienen *URLs*.

```
In [12]: tweets_corpus.shape
Out[12]: (68017, 3)

In [13]: tweets_corpus = tweets_corpus.drop_duplicates()
         tweets_corpus.shape
Out[13]: (67602, 3)

In [14]: tweets_corpus = tweets_corpus.query('agreement != "DISAGREEMENT"')
         tweets_corpus.shape
Out[14]: (66875, 3)

In [15]: tweets_corpus = tweets_corpus[-tweets_corpus.content.str.contains('^http.*$')]
         tweets_corpus.shape
Out[15]: (66550, 3)
```

Figura 6.36 - Limpieza de datos: Fase 1

Por último, se han aplicado una serie de operaciones sobre los datos para que puedan ser utilizados para entrenar un modelo.

```
In [43]: tweets_corpus['content'] = tweets_corpus['content'].astype(str)
         tweets_corpus['content'] = tweets_corpus['content'].replace('', np.nan)
         tweets_corpus = tweets_corpus.dropna(axis=0, subset=['content'])
         tweets_corpus.shape
Out[43]: (66550, 3)
```

Figura 6.37 - Limpieza de datos: Fase 2

6.2.2 Procesamiento y análisis de datos

Una vez limpio el conjunto de datos, se procede a crear una nueva columna donde se indique la polaridad de forma más clara para mostrar la distribución de cada una de las categorías.

```
In [32]: tweets_corpus['polarity_'] = 'Neutro'
         tweets_corpus.loc[tweets_corpus['polarity'].isin(['N']), 'polarity_'] = 'Negativo'
         tweets_corpus.loc[tweets_corpus['polarity'].isin(['N+']), 'polarity_'] = 'Muy negativo'
         tweets_corpus.loc[tweets_corpus['polarity'].isin(['P']), 'polarity_'] = 'Positivo'
         tweets_corpus.loc[tweets_corpus['polarity'].isin(['P+']), 'polarity_'] = 'Muy Positivo'
         tweets_corpus['polarity_'].value_counts().plot.barh()

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1d401b57e80>
```

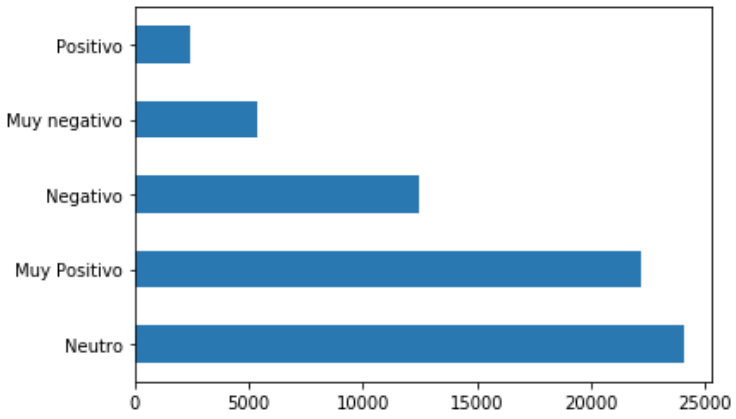


Figura 6.39 - Distribución inicial de categorías

Es necesario reducir estas cinco categorías a tres, con el fin de mejorar el entrenamiento del modelo posteriormente. Cuanto menor sea el número de categorías, mejor resultado ofrecerá el modelo.

```
In [33]: tweets_corpus.loc[tweets_corpus['polarity'].isin(['N+']), 'polarity_'] = 'Negativo'
         tweets_corpus.loc[tweets_corpus['polarity'].isin(['P+']), 'polarity_'] = 'Positivo'
         tweets_corpus['polarity_'].value_counts().plot.barh()

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1d401bb4c50>
```

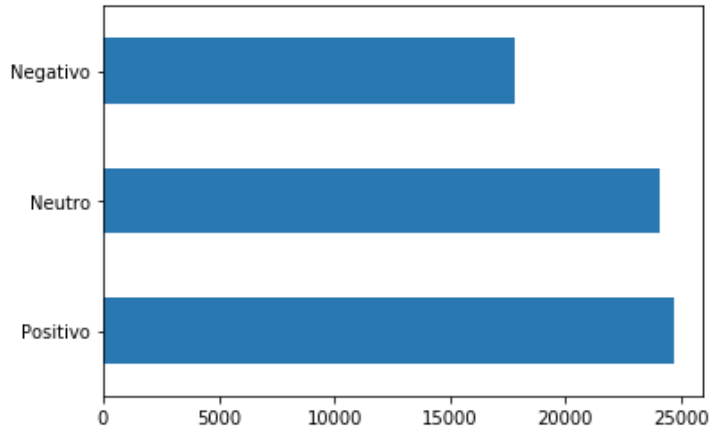


Figura 6.38 - Distribución final de categorías

Tras realizar las pruebas oportunas se concluyó que los modelos que mejor resultado ofrecían eran los de clasificación binaria, y puesto que solo se necesita identificar los mensajes negativos, se utilizaron algoritmos de clasificación binaria finalmente. Para ello se crearon dos clases: la negativa y la no negativa.

```
In [8]: tweets_corpus['polarity_'] = 'NoNegative'
        tweets_corpus.loc[tweets_corpus['polarity'].isin(['N', 'N+']), 'polarity_'] = 'Negative'
        tweets_corpus['polarity_'].value_counts()

Out[8]: NoNegative    48739
        Negative      17811
        Name: polarity_, dtype: int64
```

Figura 6.40 - Distribución de la clasificación binaria

El siguiente paso consiste en comprobar si las dos categorías que se van a utilizar están correctamente balanceadas.

```
In [47]: tweets_corpus['polarity_'] = 'NoNegative'
         tweets_corpus.loc[tweets_corpus['polarity'].isin(['N', 'N+']), 'polarity_'] = 'Negative'
         tweets_corpus['polarity_'].value_counts().plot.pie()

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1a9b5eaa470>
```

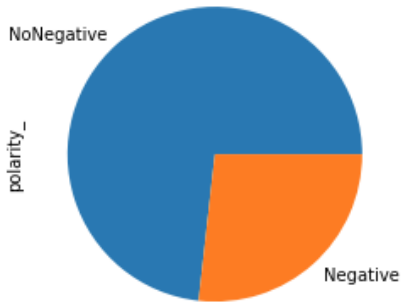


Figura 6.41 - Categorías mal balanceadas

Como se puede observar en la figura 6.41, las clases están mal balanceadas porque hay muchos más registros de la clase no negativa que de la negativa. Por lo tanto, es necesario equilibrar las dos clases.

```
In [48]: # Balanceo el dataset
         no_negative = tweets_corpus['polarity_'] == 'NoNegative'
         tweets_corpus[no_negative] = tweets_corpus[no_negative].sample(17811)
         tweets_corpus['polarity_'].value_counts().plot.pie()

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1a9b5eaa240>
```

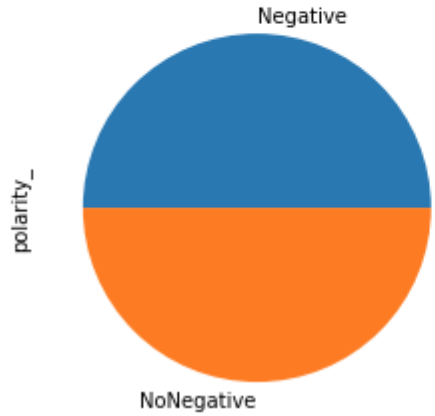


Figura 6.42 - Categorías balanceadas correctamente

6.2.3 Entrenamiento de modelos predictivos

Los estimadores que se han elegido para crear los modelos son la regresión lineal y las máquinas de vectores de soporte o *SVM*. Se seleccionaron estos dos estimadores porque fueron los que mejores resultados ofrecían.

Para ajustar los modelos se ha utilizado el algoritmo *GridSearchCV* que se encarga de generar varios modelos con diferentes parámetros y comprobar cuáles de esos parámetros son los que generan el mejor modelo.

Para la regresión logística estos son los mejores parámetros:

```
In [53]: grid_search_LogisticRegression.best_params_

Out[53]: {'cls__C': 0.5,
          'cls__max_iter': 500,
          'vect__max_df': 0.5,
          'vect__max_features': 1000,
          'vect__min_df': 20,
          'vect__ngram_range': (1, 1)}
```

Figura 6.43 - Parámetros para LogisticRegression

Para el modelo generado con máquinas de vectores de soporte estos son los mejores parámetros:

```
In [55]: grid_search_LinearSVC.best_params_

Out[55]: {'cls__C': 0.2,
          'cls__loss': 'squared_hinge',
          'cls__max_iter': 500,
          'vect__max_df': 0.5,
          'vect__max_features': 1000,
          'vect__min_df': 20,
          'vect__ngram_range': (1, 1)}
```

Figura 6.44 - Parámetros para LinearSVC

6.2.4 Pruebas de validación cruzada y comparativa de modelos

Para validar los dos modelos creados se ha utilizado un conjunto de métricas que permiten estimar cual es el mejor modelo de los dos. Las métricas que se han utilizado son: precisión, sensibilidad, exactitud y *f1-score*. La función *classification_report* permite visualizar en forma de tabla los resultados.

A continuación, se muestran los resultados de validación de la regresión logística:

```
In [29]: print(confusion_matrix(y_test,prediction_LogisticRegression))
         print(classification_report(y_test,prediction_LogisticRegression))
         print(accuracy_score(y_test, prediction_LogisticRegression))

         [[3192  388]
          [ 537 1054]]
                       precision    recall  f1-score   support

             Negative       0.86      0.89      0.87      3580
              Neutral       0.73      0.66      0.70      1591

             accuracy                           0.82      5171
            macro avg       0.79      0.78      0.78      5171
         weighted avg       0.82      0.82      0.82      5171

         0.8211177721910655
```

Figura 6.45 - Pruebas de validación de LogisticRegression

Con el estimador de máquinas de vectores de soporte se realizó el mismo proceso y estos son los resultados:

```
In [30]: print(confusion_matrix(y_test,prediction_LinearSVC))
         print(classification_report(y_test,prediction_LinearSVC))
         print(accuracy_score(y_test, prediction_LinearSVC))

         [[3168  412]
          [ 512 1079]]
                       precision    recall  f1-score   support

             Negative       0.86      0.88      0.87      3580
              Neutral       0.72      0.68      0.70      1591

             accuracy                           0.82      5171
            macro avg       0.79      0.78      0.79      5171
         weighted avg       0.82      0.82      0.82      5171

         0.8213111583832914
```

Figura 6.46 - Pruebas de validación de LinearSVC

Ambos modelos ofrecen muy buenos resultados, pero el modelo *LinearSVC* es algo mejor que la regresión logística. El siguiente paso fue mejorar el modelo *LinearSVC* probando nuevos parámetros y distintas opciones de preprocesamiento.

Por último, se realizaron pruebas de validación cruzada con la versión final del modelo predictivo, utilizando la métrica *roc_auc* o área bajo la curva *ROC*.

```
In [9]: scores = cross_val_score(
            model,
            corpus_data_features_nd[0:len(tweets_corpus)],
            y=tweets_corpus.polarity_bin,
            scoring='roc_auc',
            cv=5
            )

        scores.mean()

Out[9]: 0.8720956859765572
```

Figura 6.47 - Pruebas de validación cruzada

6.2.5 Uso del modelo predictivo

La integración del modelo predictivo en las aplicaciones de *Connect* tiene como finalidad clasificar la polaridad de los mensajes que envían los usuarios víctimas. Con el uso de este modelo se pretende:

- Detectar el grado de amenaza que sufre un usuario, con el fin de indicar a los psicólogos quienes son los usuarios más vulnerables, para poder atenderles primero.
- Reflejar el grado de progreso del usuario midiendo la polaridad de sus mensajes para ver como mejora o empeora mientras utiliza la aplicación.

Con el fin de indicar a los psicólogos cuales de los mensajes que reciben de las víctimas son negativos, se ha utilizado el color rojo para destacar estos mensajes.

Figura 6.48 - Mensajes negativos destacados

Además, se ha estimado cual es el grado de amenaza que sufre cada víctima mediante un cálculo que consiste en el número de mensajes negativos entre los menajes totales. Si el porcentaje de mensajes negativos es inferior al 10% el riesgo es bajo, si está entre el 10% y el 30% el riesgo es medio y si supera el 30% supone un alto riesgo.

Figura 6.50 - Clasificación del riesgo de la víctima

El psicólogo puede ver el riesgo que se ha establecido para cada víctima desde la interfaz de la aplicación.

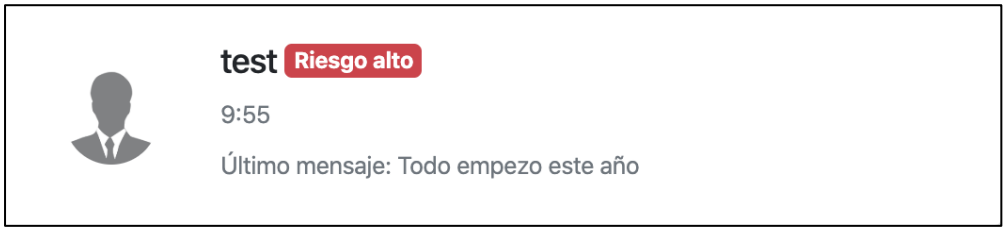


Figura 6.49 - Ejemplo de estimación del riesgo de una víctima

Por último, es posible visualizar el grado de progreso de una víctima durante el tiempo que utiliza la aplicación. Este es posible midiendo el porcentaje de mensajes negativos de manera mensual.

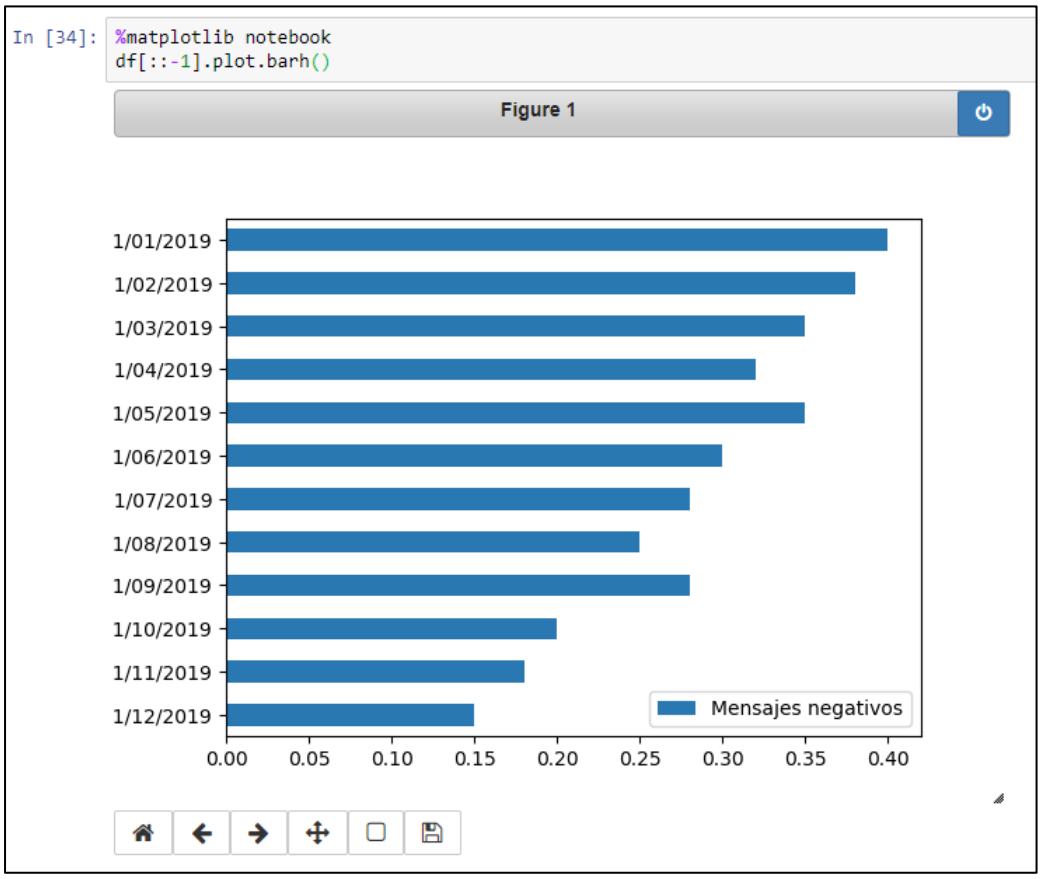


Figura 6.51 - Gráfica de progreso de una víctima

7. Conclusiones

Tras la consecución del proyecto se puede afirmar que se han realizado con éxito la gran parte de los objetivos especificados. Durante el desarrollo hubo objetivos que supusieron un gran reto o más trabajo del estimado inicialmente, debido al uso de tecnologías de diferentes ámbitos y a la complejidad de algunas de las tareas del proyecto.

La aplicación móvil se ha desarrollado con las tecnologías web más punteras hasta este momento, garantizando eficiencia, modularidad y escalabilidad. El objetivo principal de la aplicación es proporcionar un canal de comunicación anónimo, fiable, rápido y seguro, que permita a las víctimas de *bullying* y de violencia de género contactar con psicólogos para superar su situación. Otro objetivo que se considera esencial, es la creación de un mecanismo que la víctima pueda activar en caso de sufrir una situación extrema o de que su vida corra peligro. Es importante que la aplicación cubra la mayoría de situaciones de violencia que pueda sufrir el usuario, por eso también se diseñó un sistema de ocultación que enmascara la aplicación real, mostrando otra completamente diferente. Muchas de las personas que sufren este tipo de situaciones no pueden hacer un uso libre de sus *smartphones* y corren el peligro de que un tercero, incluso el agresor, descubra que están utilizando una aplicación para solicitar ayuda.

El uso de modelos predictivos supone un gran reto en las aplicaciones destinadas a ofrecer ayuda a las personas que sufren este tipo de problemas. Cuando se realizó el análisis de la competencia, ninguna de las aplicaciones que se analizaron implementaba *Machine Learning* para mejorar la funcionalidad y la experiencia de uso, por lo que se convirtió en un factor de diferenciación muy determinante. Gracias al uso de modelos predictivos es posible ofrecer a los psicólogos estadísticas sobre como mejoran en el tiempo las personas a las que ofrecen ayuda a través de *Connect*.

Considero que *Connect* es un proyecto novedoso y revolucionario que puede aportar un gran beneficio a la sociedad actual, ya que ofrece una solución paliativa completa capaz que cubrir las desventajas de la psicología tradicional. *Connect* permite a las víctimas ponerse en contacto con un psicólogo en el momento en que lo necesiten, sin cita previa. Además, pueden cambiar de psicólogo en cualquier momento de forma sencilla, incluso comunicarse con varios. La ocultación que ofrece rompe la barrera de las víctimas que sienten vergüenza al contar su situación o su libertad está coartada por sus agresores, y por lo tanto no acudirían nunca a una consulta física con un psicólogo.

8. Líneas futuras

Los objetivos principales del proyecto se han cumplido correctamente siguiendo las especificaciones iniciales, mientras que algunos de los objetivos secundarios, se han cumplido solo parcialmente. Además, han surgido nuevos objetivos que tienen como finalidad mejorar la aplicación y añadir nuevas funcionalidades. Estos nuevos objetivos permitirán aumentar el alcance del proyecto y lograr un público mayor.

Los objetivos que solo se han cumplido parcialmente y por lo tanto es necesario cumplir en su totalidad son los siguientes:

- Integración del modelo predictivo: Se ha desarrollado con éxito un modelo predictivo que calcula la polaridad de los mensajes, pero es necesario integrarlo completamente en la aplicación. Esta integración consistirá en un mecanismo que calcule la polaridad de cada mensaje que se envía antes de guardarlo en la base de datos. Actualmente se utiliza un *script* que, cada cierto tiempo, recoge los mensajes de la base de datos y predice su polaridad.
- Mecanismo de ocultación: El mecanismo de ocultación que se ha implementado, enmascara la aplicación mostrando otra diferente y, además, cambia el icono de esta. Sería necesario también modificar el nombre de la aplicación, ya que un tercero que conozca *Connect* puede ver que la víctima utiliza una aplicación con este nombre y sospechar. Actualmente no se puede modificar el nombre debido a problemas técnicos que impiden la modificación de este una vez compilada la aplicación.

Los objetivos nuevos que han surgido, para ser implementados en un futuro, son los siguientes:

- Ampliar el público objetivo: Además de ofrecer ayuda paliativa en los casos de *bullying* y de violencia de género, se ampliaría el servicio a otros problemas, en los que las víctimas no se atrevan a contar su situación ni a solicitar ayuda. Algunas de estas situaciones serían el racismo, la homofobia o el acoso laboral.
- Soporte para un servicio jurídico: Además del servicio proporcionado por los psicólogos, se incluirá otro servicio similar, constituido por abogados especializados en cada una de las situaciones en las que tenga cobertura *Connect*. La finalidad de este servicio será ayudar y asesorar a las víctimas para que denuncien su situación.

- Interfaces de ocultación: Además de la calculadora, se proporcionará al usuario un conjunto de interfaces para enmascarar la aplicación, permitiéndole escoger la que prefiera. Estas interfaces deben tener una funcionalidad similar y permitir, de alguna forma, introducir un código secreto que muestre la aplicación real.
- Aplicaciones móviles nativas: Con el fin mejorar la interacción de los usuarios, se desarrollarán aplicaciones de forma nativa para *iOS* y *Android*. Las aplicaciones nativas proporcionan un mejor rendimiento generalmente y además permiten acceder a recursos del *smartphone* de forma más sencilla.

9. Lista de referencias

[1] Epdata, «Acoso escolar, datos, cifras y estadísticas», 2019. [En línea]. Disponible en: https://www.epdata.es/datos/acoso-escolar-datos-cifras-estadisticas/257/espana/106.

[2] ABC, «Menores y acoso escolar: estos son los síntomas que alertan sobre el problema», 2019. [En línea]. Disponible en: https://www.abc.es/familia/educacion/abci-menores-y-acoso-escolar-estos-sintomas-alertan-sobre-problema-201905190149_noticia.html.

[3] C. López, «Casi 530.000 menores sufren acoso en las redes sociales», 2019. [En línea]. Disponible en: https://www.lavanguardia.com/vida/20190704/463278910418/save-the-children-acoso-ciberbullying-ninos-victimas-bullying.html.

[4] K. Nalini y L. Jaba Sheela Professor, «A survey on Datamining in Cyber Bullying», vol. 7, pp. 1865-1869, 2014.

[5] R. R. Ballesteros, «Acoso escolar: Mapa del acoso escolar: España supera por primera vez las mil víctimas en un año», 2018. [En línea]. Disponible en: https://www.elconfidencial.com/espana/2018-10-18/mapa-acoso-escolar-espana-mil-victimas-ano_1631192/.

[6] A. García, «Educación detecta 5.557 posibles casos de acoso escolar en un año | Sociedad | EL PAÍS», 2019. [En línea]. Disponible en: https://elpais.com/sociedad/2019/04/30/actualidad/1556609306_685394.html.

[7] RTVE, «Violencia de género: Más denuncias, condenas y órdenes de protección en 2018», 2019. [En línea]. Disponible en: http://www.rtve.es/noticias/20190308/violencia-genero-mas-denuncias-condenas-ordenes-proteccion-2018/1897660.shtml.

[8] F. Morales, «Aumentan las denuncias por violencia de género en 2018 | Expansión», 2018. [En línea]. Disponible en: https://www.expansion.com/sociedad/2018/11/24/5bf998c122601d585d8b45d5.html.

[9] D. Chiappe, «La mayoría de las víctimas de violencia de género es asesinada en su

casa sin haber denunciado | El Correo», 2019. [En línea]. Disponible en: https://www.elcorreo.com/sociedad/victimas-violencia-genero-20190925140328-ntrc.html.

[10] Epdata, «Madrid - Datos y estadísticas sobre violencia de género por comunidad autónoma», 2019. [En línea]. Disponible en: https://www.epdata.es/datos/datos-graficos-violencia-genero/49/madrid/304.

[11] O. C. Mimenza, «Los 7 tipos de violencia de género (y características)». [En línea]. Disponible en: https://psicologiaymente.com/forense/tipos-violencia-de-genero.

[12] «Angular», 2019. [En línea]. Disponible en: https://angular.io/.

[13] «Bootstrap · The most popular HTML, CSS, and JS library in the world.», 2019. [En línea]. Disponible en: https://getbootstrap.com/.

[14] «Python», 2019. [En línea]. Disponible en: https://www.python.org/.

[15] «NumPy», 2019. [En línea]. Disponible en: https://numpy.org/.

[16] «Pandas: Python Data Analysis Library», 2019. [En línea]. Disponible en: https://pandas.pydata.org/.

[17] «NLTK: Natural Language Toolkit», 2019. [En línea]. Disponible en: https://www.nltk.org/.

[18] «scikit-learn: machine learning in Python», 2019. [En línea]. Disponible en: https://scikit-learn.org/stable/.

[19] «Joblib: running Python functions as pipeline jobs», 2019. [En línea]. Disponible en: https://joblib.readthedocs.io/en/latest/.

[20] R. R. McCune, «Node. js Paradigms and Benchmarks», *Striegel, Gr. Os F*, vol. 11, pp. 1-6, 2011.

[21] «Express - Infraestructura de aplicaciones web Node.js», 2019. [En línea]. Disponible en: https://expressjs.com/es/.

[22] «Socket.IO», 2019. [En línea]. Disponible en: https://socket.io/.

[23] Hipertextual, «Socket.io, comunicación bidireccional en tiempo real», 2014. [En línea]. Disponible en: https://hipertextual.com/archivo/2014/08/socketio-javascript/.

[24] «MongoDB», 2019. [En línea]. Disponible en: https://www.mongodb.com/es.

[25] Applicantes, «La tendencia de las aplicaciones móviles está en auge y crear una propia puede marcar la diferencia : Applicantes – Información sobre apps y juegos para móviles», 2019. [En línea]. Disponible en: https://applicantes.com/aplicaciones-moviles-smartphones-empresas/.

[26] P. An y A. Bullying, «Method and apparatus for providing an anti - bullyng service», vol. 2, 2017.

[27] X. Peytibi, «Apps contra el acoso sexual», 2019. [En línea]. Disponible en: https://www.elperiodico.com/es/apps-para-el-ciudadano-comprometido/20190710/apps-contra-el-acoso-sexual-7548386.

[28] A. Herranz, «Smartphones y menores: más uso, más exposición a riesgos, pero menos daños | El Comercio», 2018. [En línea]. Disponible en: https://www.elcomercio.es/tecnologia/smartphones-menores-exposicion-20180227122329-ntrc.html.

[29] B. Pang y L. Lee, *Opinion Mining and Sentiment Analysis: Foundations and Trends in Information Retrieval*, vol. 2, n.º 1-2. 2008.

[30] B. Liu, «Sentiment analysis and opinion mining», *Synth. Lect. Hum. Lang. Technol.*, vol. 5, n.º 1, pp. 1-184, 2012.

[31] «Psonríe». [En línea]. Disponible en: https://www.psonrie.com/.

[32] «PsicoGlobal». [En línea]. Disponible en: https://www.psicoglobal.com/.

[33] «Ypsihablamos». [En línea]. Disponible en: https://www.ypsihablamos.com/.

[34] «WorkingMinds». [En línea]. Disponible en: https://www.workingminds.es/.

[35] «mediQuo». [En línea]. Disponible en: https://www.mediquo.com/.

[36] «b-resol». [En línea]. Disponible en: https://b-resol.com/.

[37] «ZeroAcoso». [En línea]. Disponible en: https://www.zeroacoso.org/.

[38] «Kitestring». [En línea]. Disponible en: https://www.kitestring.io/.

[39] J. Á. López, «Ventajas de utilizar Angular, un framework JavaScript», 2018. [En línea]. Disponible en: https://blog.aitana.es/2018/04/10/ventajas-de-utilizar-angular/.

[40] «Apache Cordova», 2019. [En línea]. Disponible en: https://cordova.apache.org/docs/es/latest/guide/overview/.

[41] S. Daniel, «Introducción a JSON Web Tokens (JWT)», 2016. [En línea]. Disponible en: https://platzi.com/blog/introduccion-json-web-tokens/.

[42] C. Azaustre, «Usando Websockets con NodeJS y SocketIO», 2015. [En línea]. Disponible en: https://carlosazaustre.es/websockets-como-utilizar-socket-io-en-tu-aplicacion-web/.

[43] M. A. Alvarez, «Trabajar con módulos en Angular», 2017. [En línea]. Disponible en: https://desarrolloweb.com/articulos/trabajar-modulos-angular.html.

[44] Cleventy, «¿Qué es Material Design?», 2019. [En línea]. Disponible en: https://cleventy.com/que-es-material-design/.

[45] «Características del diseño minimalista moderno», 2017. [En línea]. Disponible en: https://www.hazloconceramicos.com/blog/5-caracteristicas-del-diseno-minimalista-moderno/.

[46] J. González Rubio, «Desarrollo de una aplicación web y una API REST para realizar envíos colaborativos en Node.js», mar. 2017.

[47] «Winston», 2019. [En línea]. Disponible en: https://www.npmjs.com/package/winston#logging-levels.

[48] R. Passonneau, «Sentiment Analysis of Twitter Data», *Proc. Work. Lang. Soc. Media (LSM 2011)*, n.º June, pp. 30-38, 2011.

[49] T. I. Jain y D. Nemade, «Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis», *Int. J. Comput. Appl.*, vol. 7, n.º 5, pp. 12-21, 2010.

[50] A. Mathur y G. M. Foody, «Multiclass and binary SVM classification: Implications for training and classification users», *IEEE Geosci. Remote Sens. Lett.*, vol. 5, n.º 2, pp. 241-245, 2008.

[51] J. Torres, «Servicios y demonios en Linux», 2013. [En línea]. Disponible en: https://medium.com/jmtorres/servicios-y-demonios-en-linux-a424366336ac.

[52] S. Tilkov y S. Vinoski, «Node.js: Using JavaScript to build high-performance network programs», *IEEE Internet Comput.*, vol. 14, n.º 6, pp. 80-83, 2010.

[53] «TASS: Workshop on Semantic Analysis at SEPLN», 2015. [En línea]. Disponible en: http://www.sepln.org/workshops/tass/.

[54] H. Cerezo-costas, D. Celix-salgado, y E. Costa-montenegro, «GTI-Gradiant at TASS 2015: A Hybrid Approach for Sentiment Analysis in Twitter», *Tass 2015*, pp. 35-40, 2015.

[55] C. R. Association for Computational Linguistics. Meeting (45th : 2007 : Prague, R. E. Association for Computational Linguistics., P. T. Pham, D. Huang, A. Y. Ng, y C. Potts, «Learning Word Vectors for Sentiment Analysis», *Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol. - Vol. 1*, pp. 142-150, 2007.

[56] «PM2», 2019. [En línea]. Disponible en: https://pm2.keymetrics.io/.

Anexo A. Manual de instalación

Aplicación web

En el desarrollo de la aplicación móvil se han utilizado tecnologías web, concretamente *Angular 7*. Es posible ejecutar la aplicación en cualquier navegador de escritorio utilizando *Angular CLI*.

El primer paso es abrir una terminal y acceder al directorio del proyecto de *Angular*. Después será necesario instalar todas las librerías necesarias que incluye el proyecto, ejecutando el siguiente comando:

- **`npm install`**

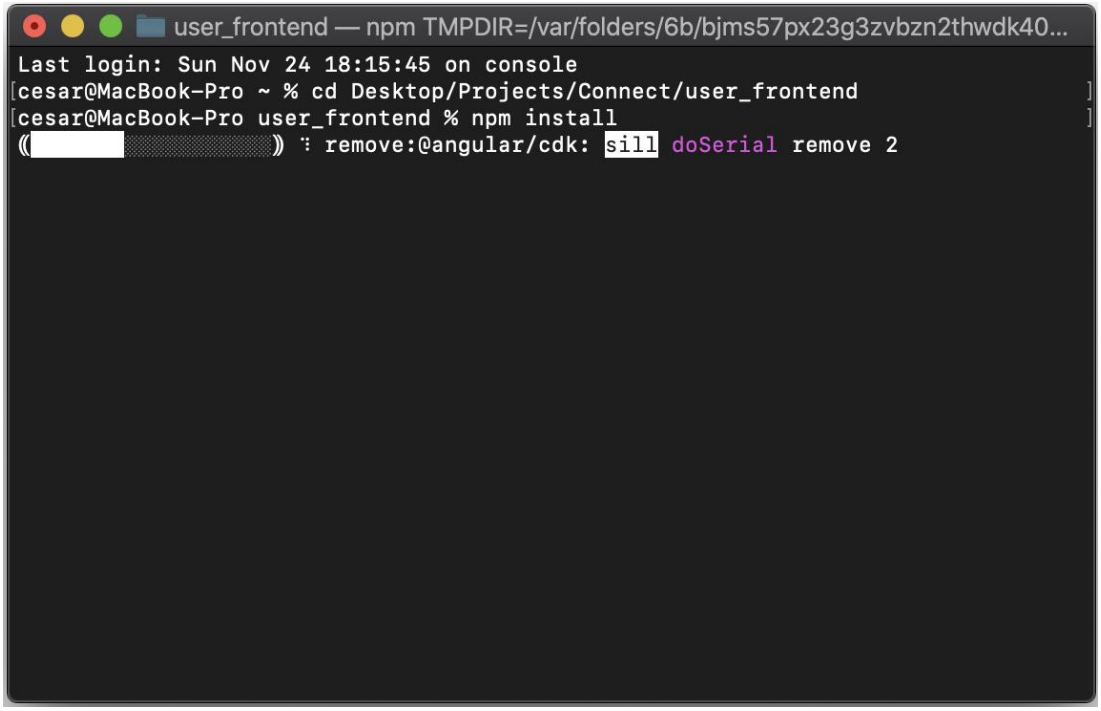


Figura A.1 - Aplicación web: Instalación de dependencias

Una vez se hayan instalado todas las dependencias, ya es posible ejecutar la aplicación con el compilador que incluye *Angular CLI*. Se ejecutará el siguiente comando:

- **`ng serve –port 4200`**

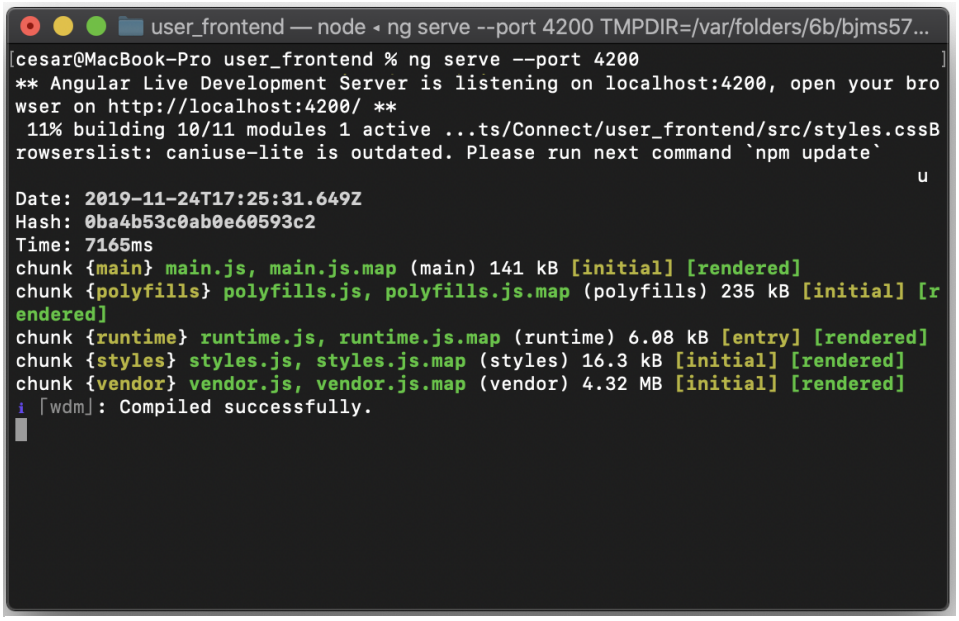


Figura A.2 - Aplicación web: Ejecución en modo debug

Al introducir la *URL "http://localhost:4200/"* en el navegador, se ejecutará la aplicación. Para ver mejor la interfaz será necesario abrir las herramientas del navegador y simular un dispositivo móvil.

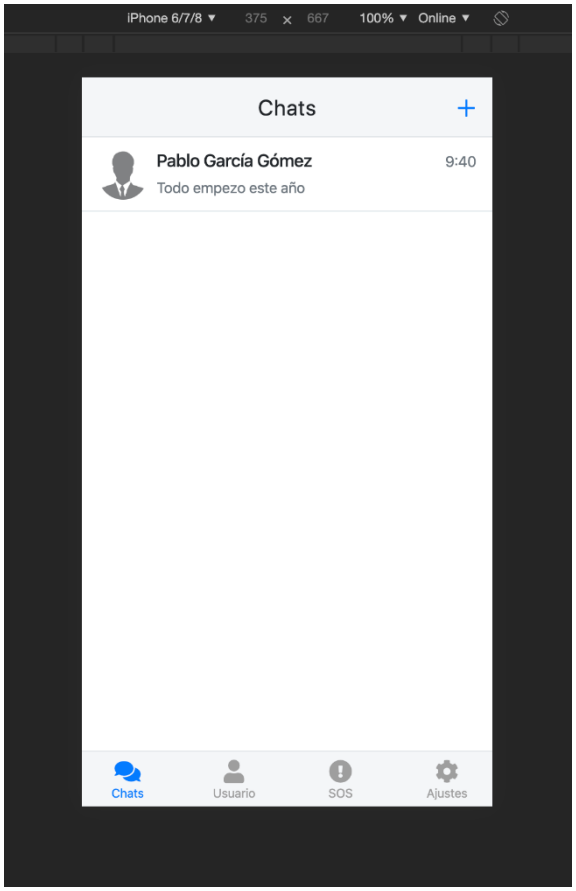


Figura A.3 - Aplicación web: Interfaz de la aplicación

Angular también permite exportar la aplicación en modo de producción. Una vez exportada la aplicación, esta funcionará en cualquier servidor web, ya que solo bastaría con colocar los archivos que *Angular* exporta e introducirlos en la raíz de un servidor web. Para exportar la aplicación en modo producción se ejecutará el siguiente comando:

- **`ng build --prod`**

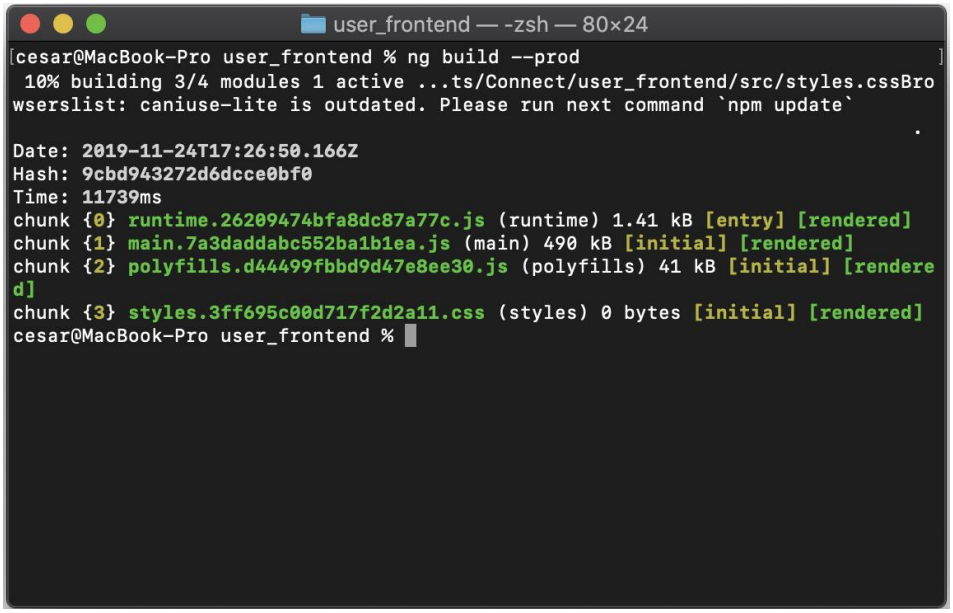


Figura A.4 - Aplicación web: Exportación de la aplicación

Los archivos que se han generado se encuentran en el directorio "*/dist/Connect*", en la raiz del proyecto y contienen toda la aplicación.

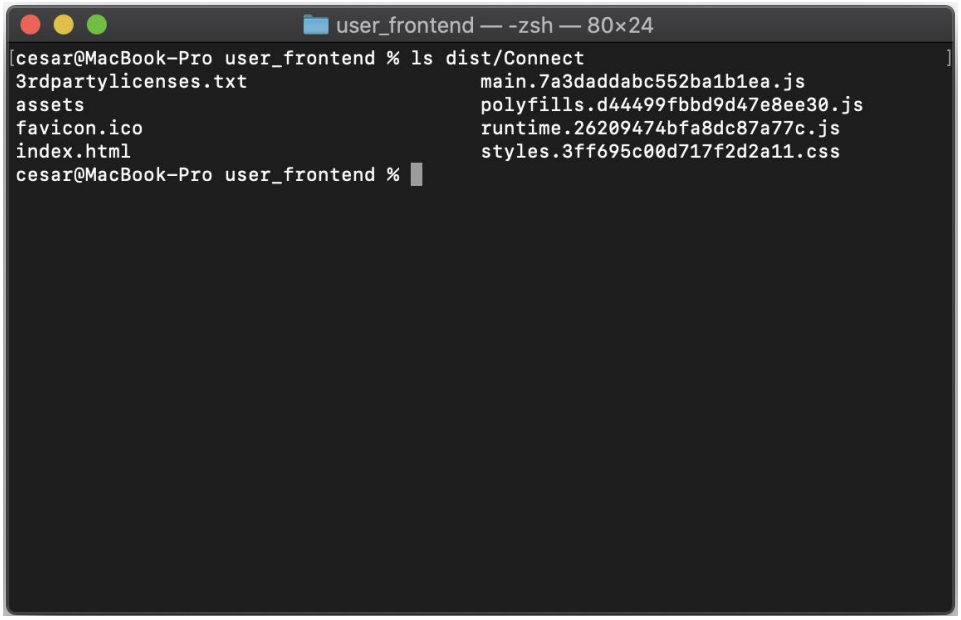


Figura A.5 - Aplicación web: Archivos de la aplicación exportada

Integración en Cordova

Para crear las aplicaciones móviles para *iOS* y *Android* es necesario utilizar la herramienta *Cordova*.

El primer paso es crear el directorio donde se almacenará el proyecto de *Cordova* y situarse en su interior para instar las dependencias necesarias del proyecto. Se utilizará el siguiente comando:

- **`cordova craete ConnectAPP cesar.connect "ConnectAPP "`**

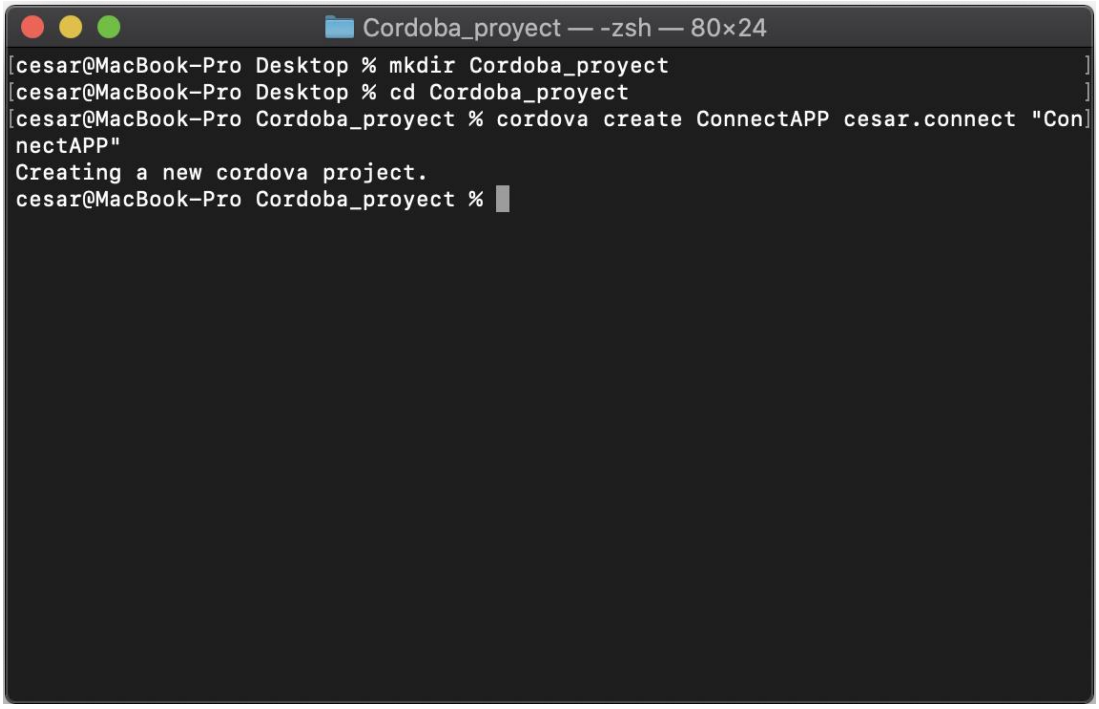


Figura A.6 - Integración en Cordova: Creación del proyecto

Una vez creado el proyecto de *Cordova* es necesario alojar en su interior la aplicación web exportada en modo de producción.

Para ello es necesario situarse en el proyecto de *Angular* y exportar la aplicación en el interior del directorio "*/www*", que está en la raíz del proyecto de *Cordova*. Se utilizará el siguiente comando:

- **`ng build –prod –base-href . –output-path [Cordova Proyect]/www`**

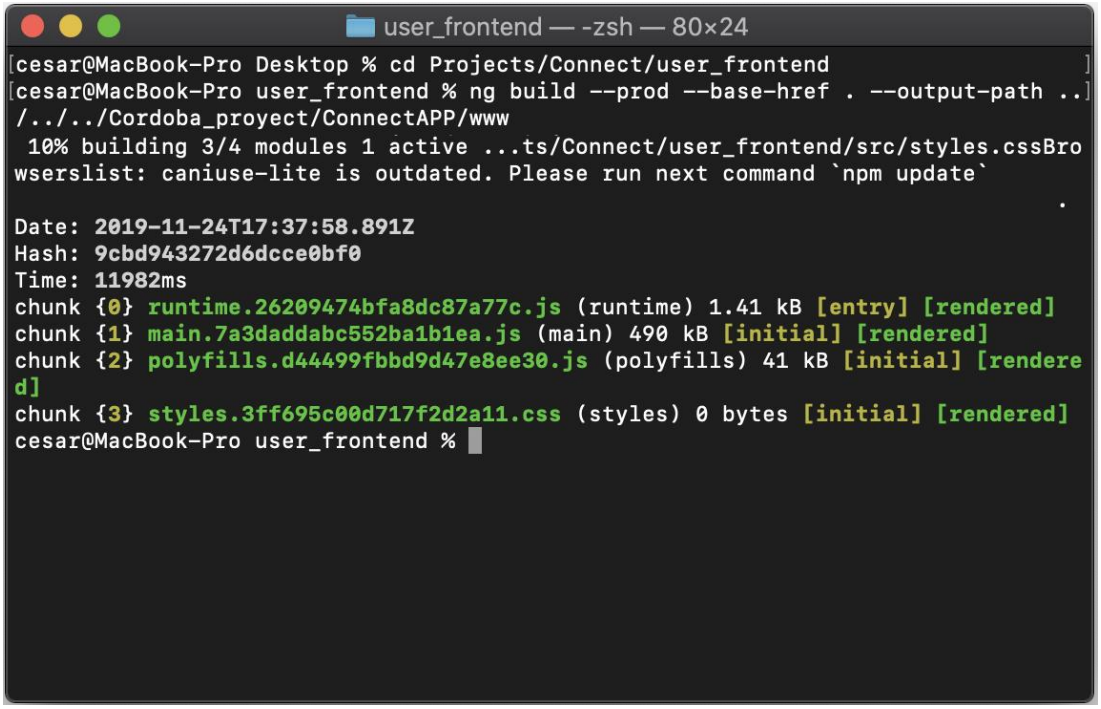


Figura A.7 - Integración en Cordova: Importación del proyecto de Angular en Cordova

Para que la aplicación funcione correctamente es necesario abrir el dichero "*index.html*" ubicado en el directorio "*/www*" del proyecto de *Cordova* e incluir un *script* de *Cordova* con la siguiente línea:

- **<script type="text/javascript" src="cordova.js"></script>**

```
<script type="text/javascript" src="cordova.js"></script>
```

Figura A.8 - Integración en Cordova: Script necesario de Cordova

El siguiente paso es instalar el *plugin* encargado de traducir la solicitud de geolocalización que se hace desde *JavaScript*. Se utilizará el siguiente comando:

- **cordova plugin add cordova-plugin-geolocation**

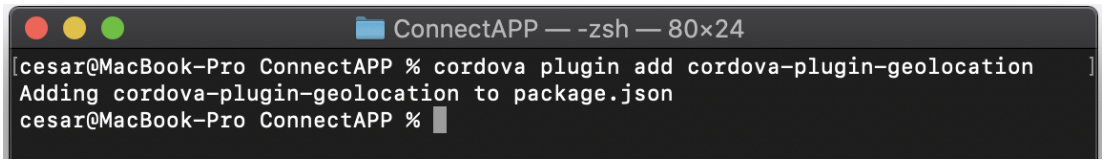


Figura A.9 - Integración en Cordova: Instalación del plugin para la geolocalización

Por último se necesita establecer el mensaje que se le mostrará al usuario en la solicitud de la localización. Para ello será necesario modificar el archivo "*config.xml*", ubicado en la raiz del proyecto de *Cordova*. Se añadirán las siguientes lineas:

```
<edit-config target="NSLocationWhenInUseUsageDescription" file="*-Info.plist" mode="merge">
    <string>Permitir geolocalización</string>
</edit-config>
```

Figura A.10 - Integración en Cordova: Mensaje de solicitud de la geolocalización

Aplicaciones móviles

Una vez integrada la aplicación en *Cordova*, el siguiente paso es crear las aplicaciones móviles para *iOS* y *Android*.

Será necesario añadir las dependencias necesarias de *iOS* y *Android* para integrar la aplicación. Es bastante sencillo y se realiza con los siguientes comandos:

- **`cordova platform add ios`**
- **`cordova platform add android`**

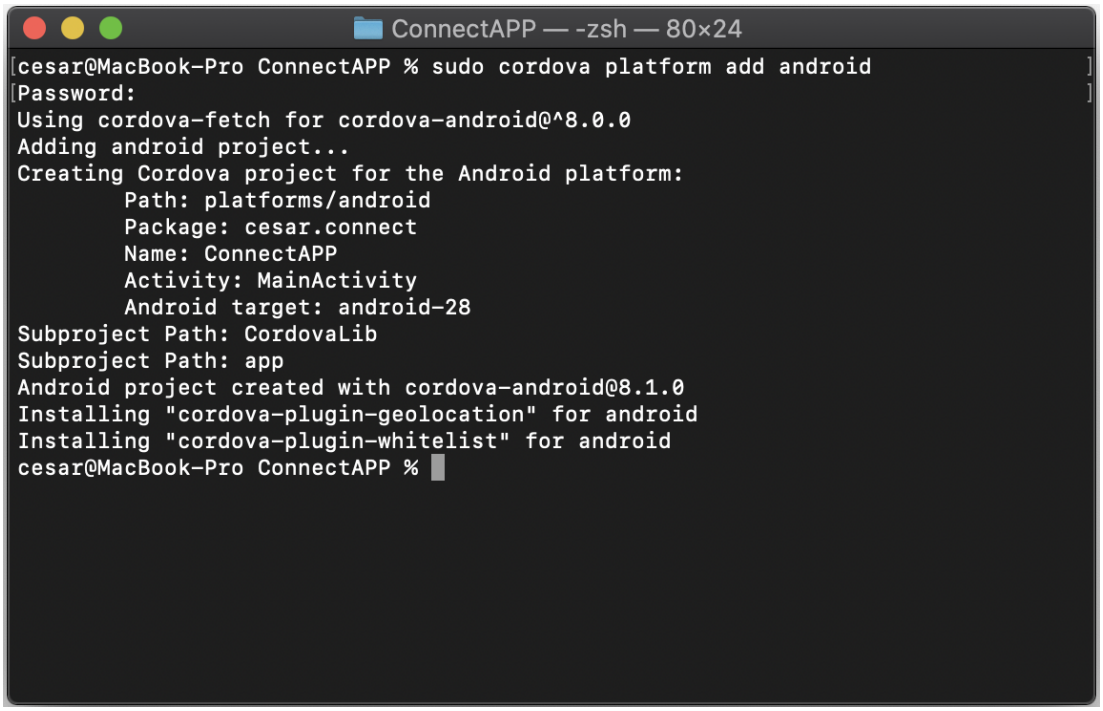


Figura A.11 - Aplicaciones móviles: Integración en las plataformas iOS y Android

El siguiente paso es obtener la lista de dispositivos móviles que se encuentran disponibles, tanto físicamente como de forma simulada. Se utilizarán los siguientes comandos:

- **`cordova run android -list`**
- **`cordova run ios --list`**

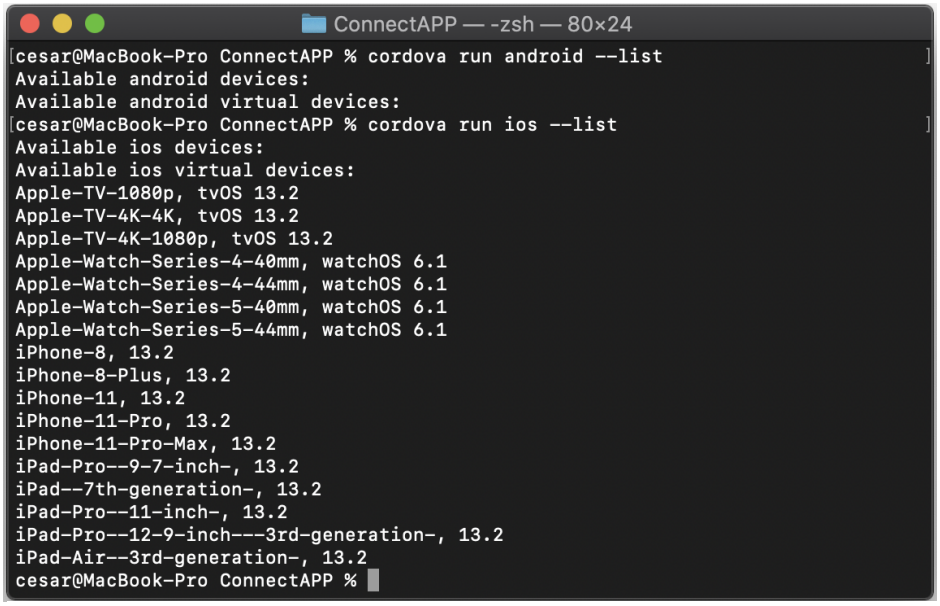


Figura A.12 - Aplicaciones móviles: Obtención de dispositivos móviles disponibles

Por último, para ejecutar la aplicación se debe elegir entre una de las dos plataformas y elegir un dispositivo móvil en concreto con el siguiente comando:

- **`cordova run [platform] -target="[ mobile device]"`**

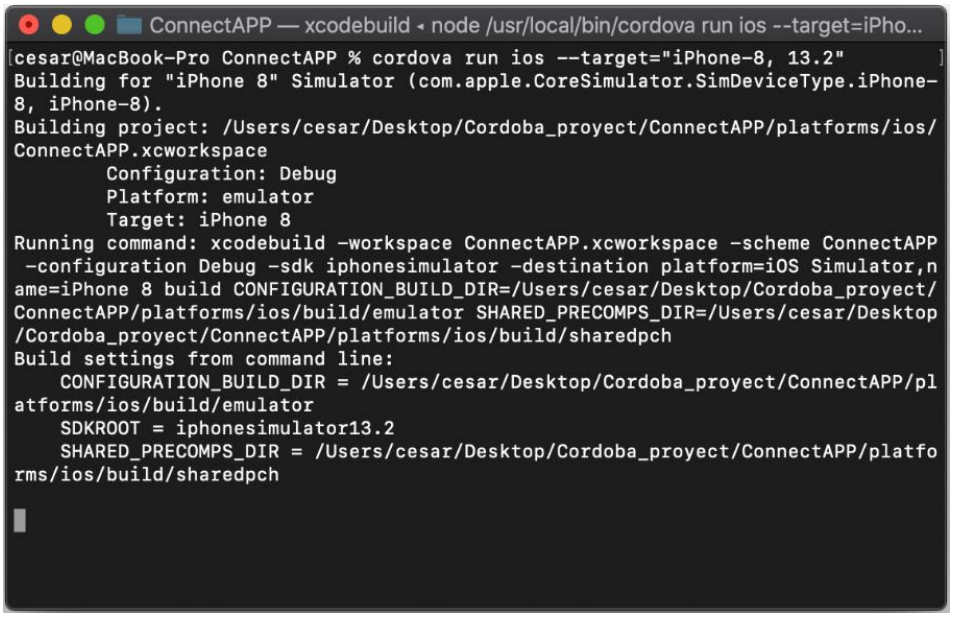


Figura A.13 - Aplicaciones móviles: Iniciar ejecución en un dispositivo móvil

Si es la primera vez que se ejecuta la aplicación se requerirán algunos minutos para la instalación. Una vez haya finalizado esta, se informará al usuario.

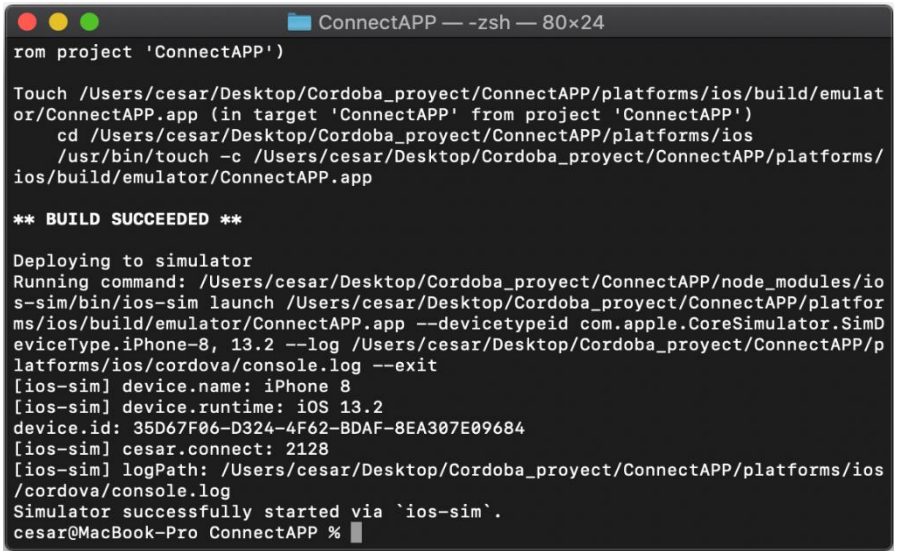


Figura A.14 - Aplicaciones móviles: Aplicación instalada en un dispositivo móvil

Inmediatamente después, la aplicación se abrirá de forma automática en el dispositivo móvil o en el simulador.

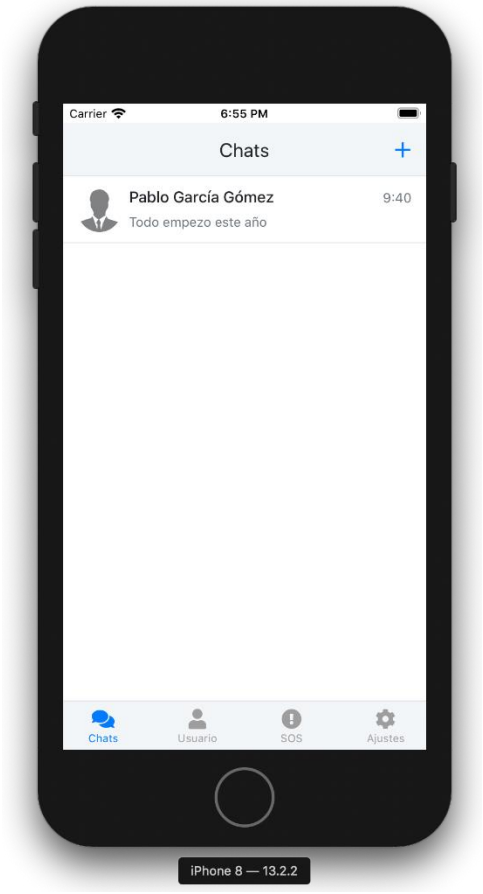


Figura A.15 - Aplicaciones móviles: Simulador ejecutando la aplicación

Back-end

El primer paso es abrir una terminal y acceder al directorio donde se encuentre ubicado el *Back-end*. Después será necesario instalar todas las dependencias necesarias que incluye el proyecto, ejecutando el siguiente comando:

- **`npm install`**

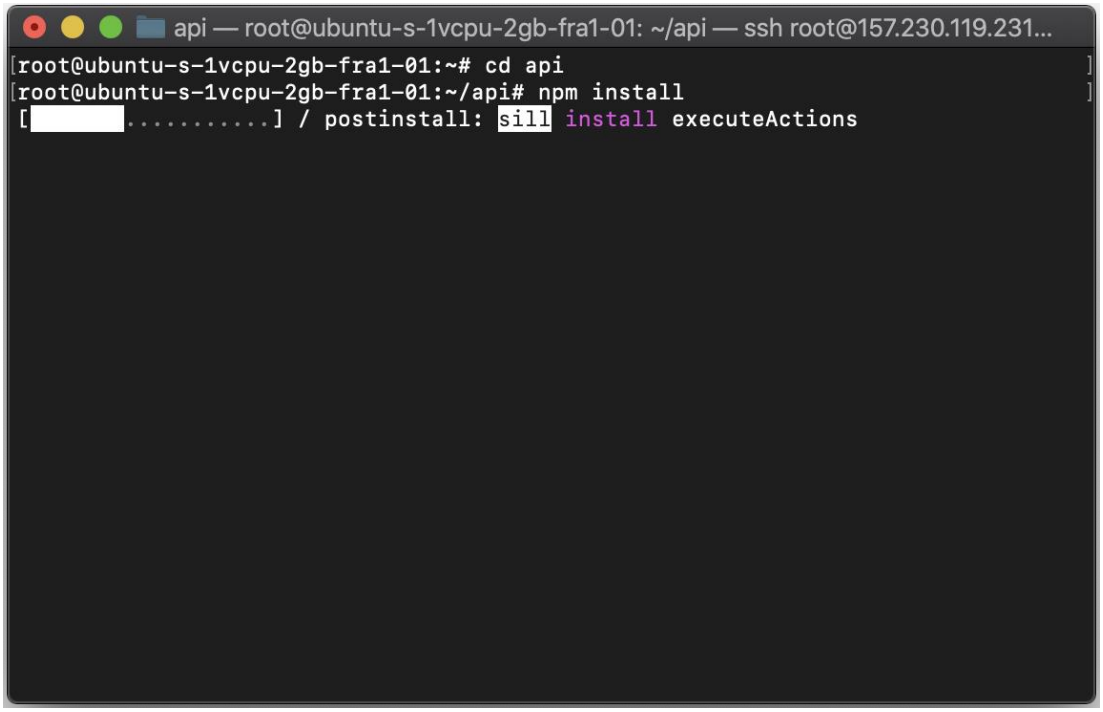


Figura A.16 - Back-end: Instalación de dependencias

Una vez se hayan instalado todas las dependencias, ya es posible ejecutar el *Back-end* con la interfaz de comandos de *npm*. Se ejecutará el siguiente comando:

- **`npm start`**

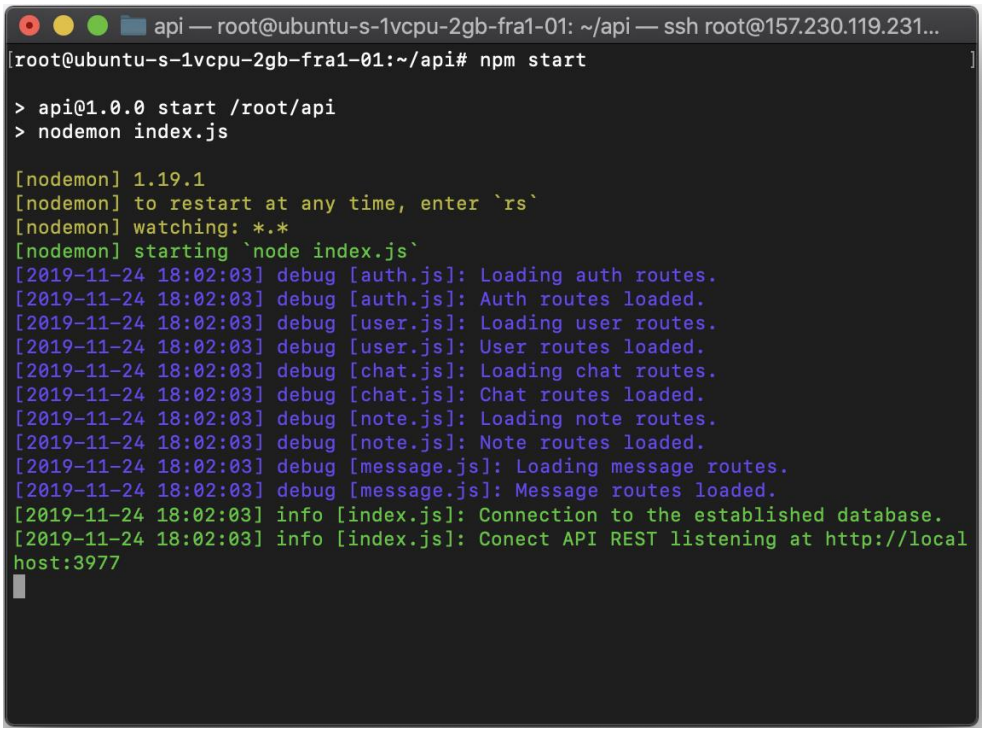


Figura A.17 - Back-end: Ejecución del Back-end

Para mantener el *Back-end* en un modo ejecución constante se necesita un administrador de procesos demonio, como *PM2*.[56] Con siguiente comando se ejecutará el *Back-end* a modo de demonio:

- **`pm2 start index.js`**

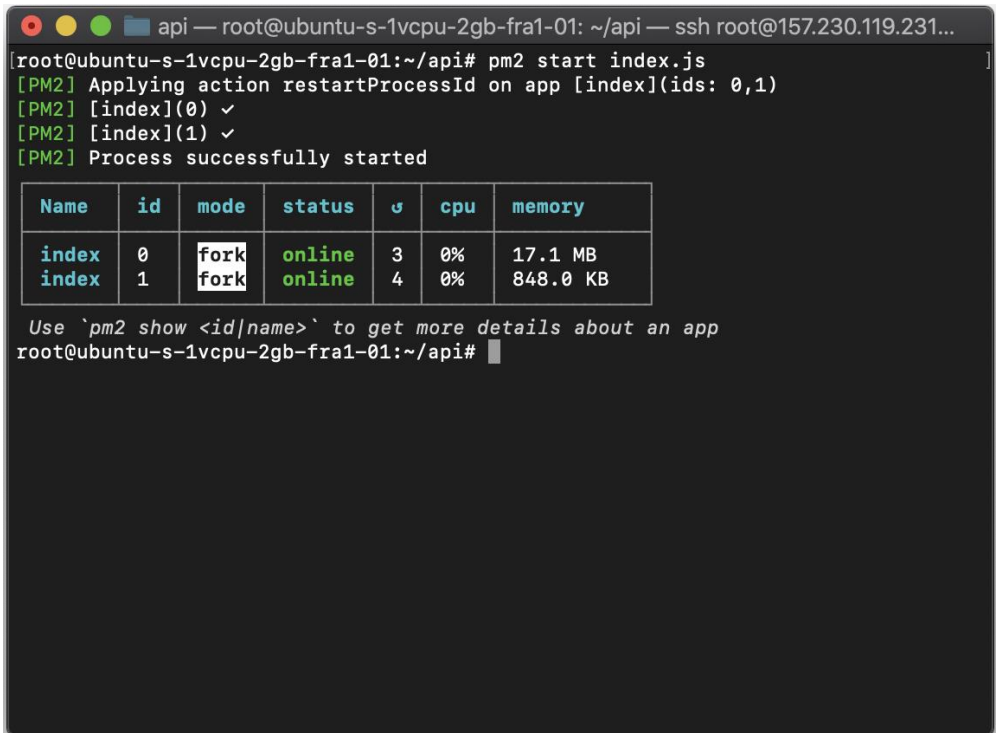


Figura A.18 - Back-end: Creación de un proceso demonio

El *Back-end* implementa un registro de la mayoría de las acciones que realiza durante su ejecución. Este registro se encuentra distribuido en varios archivos, cada uno con la fecha concreta del día en el que se registraron las acciones. El registro se encuentra ubicado en el directorio "*/logs*", en la raíz del proyecto.

Para ver un archivo de registro concreto se puede utilizar el siguiente comando:

- **`cat [yyyy-MM-dd]-results.log`**

Figura A.19 - Back-end: Registro de acciones del Back-end

Anexo B. Presupuesto del proyecto

Hardware

El proyecto *Connect* contará con un entorno para el desarrollo y las pruebas, además de un entorno de producción constituido por dos servidores, uno en el que se alojará el *Back-end* y en el otro la aplicación web destinada a los psicólogos. Para ello se utilizará el servicio que proporciona la web *Digital Ocean* como *hosting*.

El coste del entorno para de desarrollo y las pruebas es el siguiente:

- *Raspberry Pi 3*: Se alojará el *Back-end* y la aplicación web destinada a los psicólogos. El coste es de 50€.
- Equipo *Macbook*: Empleado para el desarrollo de la aplicación móvil destinada a las víctimas y para la simulación de esta en los dispositivos móviles. El coste es de 1500€.
- Equipo de sobremesa: Empleado para el desarrollo del *Back-end* y de la aplicación web destinada a los psicólogos. El un coste es de 600€.

El coste del entorno de producción es el siguiente:

- Servidor destinado al *Back-end*: Se solicitará un servidor *Ubuntu* con un coste mensual de 20€ con las siguientes características:
  - ✓ Memoria *RAM* de 4GB
  - ✓ Dos *CPUs*
  - ✓ Almacenamiento *SSD* de 80GB
- Servidor destinado al *Front-end* de los psicólogos: Se solicitará otro servidor *Ubuntu* con un coste mensual de 10€ con las siguientes características:
  - ✓ Memoria *RAM* de 2GB
  - ✓ Un *CPUs*
  - ✓ Almacenamiento *SSD* de 50GB

Tabla B.1 - Anexo B: Costes de Hardware

Concepto	Coste
Raspberry Pi 3	50,00 €
Equipo *Macbook*	1.500,00 €
Equipo de sobremesa	600,00 €
Servidor destinado al *Back-end*	240,00 €
Servidor destinado al *Front-end* de los psicólogos	120,00 €
Total	**2.510,00 €**

Software

Puesto que todo el proyecto se va a desarrollar con software *open source* no existen costes adicionales de este tipo.

Dominio

Será necesario adquirir un dominio para utilizar una dirección web conocida que permita acceder a la aplicación web, así como un subdominio para el *Back-end*. Se solicitará un dominio web con la extensión "*.es*" y con el nombre "*connect*". El coste anual es de 25€.

Tabla B.2 - Anexo B: Costes de dominio

Concepto	Coste
Dominio	25,00 €
Total	**25,00 €**

Personal técnico

La estimación salarial por hora que se aplicó para la realización del proyecto es de 7€/hora. Durante el desarrollo del proyecto participaron dos personas que conformaron el personal técnico:

- César Gutiérrez Pérez: Empleó un total de 217 días trabajando aproximadamente 4 horas diarias.
- Jesús García Potes: Empleó un total de 157 días trabajando aproximadamente 5 horas diarias.

Tabla B.3 - Anexo B: Costes de personal técnico

Concepto	Coste
Salario del técnico 1 (César Gutiérrez)	8.680,00 €
Salario del técnico 2 (Jesús García)	7.800,00 €
Total	**16.480,00 €**

Coste total de proyecto

Tabla B.4 - Anexo B: Coste total del proyecto

Concepto	Coste
Hardware	2.510,00 €
Dominio	25,00 €
Personal técnico	16.480,00 €
Total	**19.015,00 €**