Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de servidores de recepción y presentación de datos para el análisis y la gestión de QoX por expertos.

Development of servers for receiving and presenting data for the analysis and management of QoX by experts.

Autora

Marta Lampaya Pérez

Directora

Rosa Mora Marín

Ponente

José Ruiz Mas

Escuela de Ingeniería y Arquitectura

Curso 2020/2021

ÍNDICE

ÍNDICE de FIGURAS

ÍNDICE de TABLAS

ACRÓNIMOS

QoX/QoE – Quality of Experience (Calidad de Experiencia)

TFG – Trabajo Final de Grado

ITU-T – International Telecommunication Union (Unión Internacional de Telecomunicaciones)

TLS – Transport Layer Security (Seguridad de la Capa de Transporte)

SSL -- Secure Sockets Layer (Capa de Puertos Seguros)

HTTP -- Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto)

HTTPS -- Hypertext Transfer Protocol Secure (Protocolo seguro de transferencia de hipertexto)

CSS -- Cascading Style Sheets (Hoja de estilos en cascada)

SQL -- Structured Query Language (Lenguaje de consulta estructurada)

IDE -- Integrated Development Environment (Entorno de desarrollo integrado)

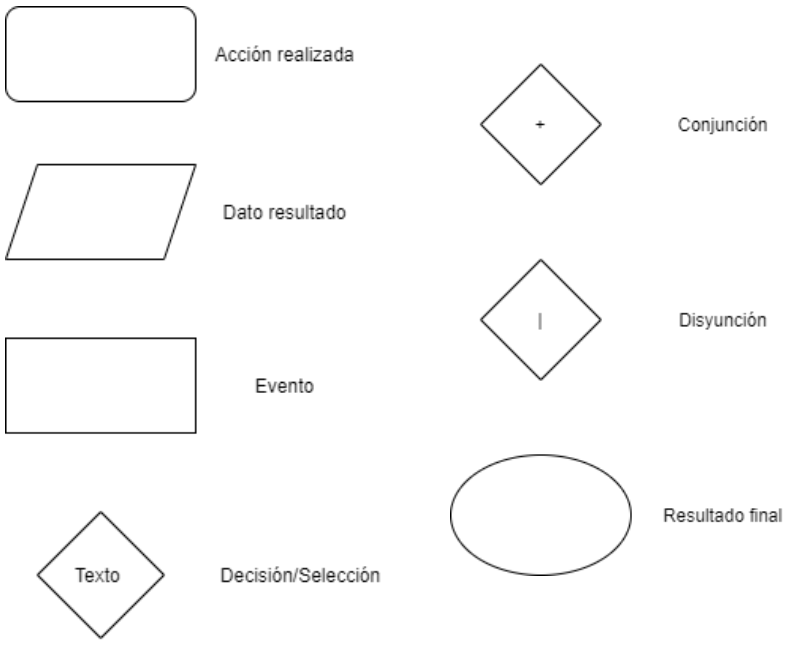
MVC -- Modelo Vista Controlador

HMTL -- HyperText Markup Language (Lenguaje de Marcas de Hipertexto)

csv -- Comma-Separated Values (Valores Separados Por Comas)

TCP -- Transmission Control Protocol (Protocolo de Control de Transmisión)

LEYENDA DIAGRAMAS DE FLUJO



RESUMEN

En este documento titulado "Desarrollo de servidores de recepción y presentación de datos para el análisis y la gestión de QoX por expertos." se expone el proyecto realizado como Trabajo Fin de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación en la especialidad de Telemática durante el curso 2020/2021.

Este Trabajo Fin de Grado desarrolla una infraestructura de estudio con el fin de permitir al Grupo EducaViva de la Universidad de Zaragoza realizar investigaciones en el ámbito de la calidad de experiencia como expertos en educación acerca del uso de las nuevas tecnologías por los jóvenes, tanto en el ámbito escolar como en el personal.

A lo largo de todo este proyecto se ha trabajado con el fin de cubrir las necesidades expuestas por los expertos a lo largo de diversas reuniones. Se mostrará la metodología que se ha seguido para conseguir plasmar los requerimientos de dichos expertos en la infraestructura implementada. Así mismo, se incidirá en la elección de las herramientas y el proceso de desarrollo de la arquitectura llevada a cabo para su construcción.

La solución definida es un entorno de estudio que puede servir como base para futuros trabajos, y deberá ser evaluado y optimizado con el objetivo de realizar un uso real y continuado de su infraestructura.

1. INTRODUCCIÓN

1.1. Ubicación del trabajo

La aparición y la evolución de nuevas tecnologías en multitud de dispositivos móviles han introducido una nueva forma de vida en nuestra sociedad. Son muchas las ventajas que nos proporcionan en el día a día permitiéndonos estar conectados en cualquier momento y lugar a un sinfín de recursos. Sin embargo, esta capacidad de conexión también es un potente medio de distracción, sobre todo, en los más jóvenes.

Para algunas personas, el uso de las nuevas tecnologías puede convertirse en un abuso e incluso en una conducta adictiva. Las adicciones comportamentales, llamadas también adicciones sin droga, son un tipo de adicción en la cual se da una pérdida de control y una dependencia psicológica de una determinada conducta o actividad, que en sí misma, no es perjudicial para la salud de las personas y que tiene como consecuencias, igual que las adicciones con sustancia, la dependencia, la tolerancia, el síndrome de abstinencia y la interferencia grave en la vida cotidiana de la persona provocando un desinterés por otras actividades. Desde el punto de vista biológico, se ha visto que estas adicciones comportamentales actúan a través de receptores neuronales y sistemas de neurotransmisión muy similares a los que generan la adicción a las drogas tóxicas.

En este contexto, este Trabajo de Fin de Grado (TFG) se centra en el desarrollo de una infraestructura con herramientas que permitan la gestión y la mejora de la Calidad de Experiencia (QoX o QoE), en el área de la utilización de las nuevas tecnologías con el fin de velar por que su uso sea saludable. La QoX [1] se define según la ITU-T como la aceptabilidad general de una aplicación o servicio, tal como lo percibe subjetivamente el usuario final. A su vez, la aceptabilidad general puede verse influenciada por las expectativas y el contexto del usuario. La QoX incluye los efectos completos del sistema de extremo a extremo (cliente, terminal, red, infraestructura de servicios, etc.).

La infraestructura desarrollada deberá dotar a los expertos de los instrumentos necesarios para mejorar estas expectativas, así como el estado emocional de los usuarios finales de los dispositivos móviles que repercuten en su percepción de la QoX. De esta forma, se pretende poder llegar a conseguir un uso más responsable y eficiente de las nuevas tecnologías desde la reeducación de los usuarios implicados. Para lograr este resultado, en primer lugar, será imprescindible estudiar en profundidad los comportamientos de dichos usuarios en relación con la utilización de las nuevas tecnologías para identificar patrones y conductas tanto individuales como colectivas. Este TFG se ubica, concretamente, en esta primera fase de recogida y presentación de datos de dicha utilización en diversos dispositivos móviles por multitud de usuarios.

1.2. Objetivos

Como se ha comentado en el apartado anterior, el propósito general de este TFG es desarrollar una infraestructura para otorgar a expertos las herramientas necesarias

para poder evolucionar en el campo de la QoX en relación con la adecuada utilización de las nuevas tecnologías por usuarios finales.

Para conseguir este propósito, el objetivo del trabajo será desarrollar un entorno de *back-end* compuesto por diversos servidores que se encargarán de:

- La recepción de datos de utilización de multitud de dispositivos móviles de usuario.
- El almacenamiento y la distribución de todos los datos con el fin de hacer una primera clasificación de toda la información recibida.
- La provisión de un servicio web a los expertos que presentará la información en diferentes formatos con el objetivo de que dichos expertos puedan analizar y gestionar los datos obtenidos.

Asimismo, para probar la potencialidad de la infraestructura diseñada, como último paso se desarrollarán:

- Diversos prototipos de clientes que simulen el envío de los diferentes tipos de mensajes posibles con los sistemas de recepción de datos que formarán un primer sistema básico de pruebas sobre el entorno implementado.
- Un servidor que se comunique con los dispositivos móviles de los usuarios para demostrar la posibilidad de un envío de datos personalizado a cada uno de ellos con el que probar la capacidad de hacer realimentación al usuario a partir de la información obtenida y los estudios realizados.

Los principales problemas a abordar en el desarrollo de este entorno serán:

- La conexión de los sistemas de recepción con multitud de dispositivos móviles. Para esta conexión se establecerá un protocolo y un formato fijo de mensajes que permitirá automatizar el proceso de comunicación y recogida de datos.
- La comunicación de los sistemas de recepción con los sistemas de almacenamiento que proporcionarán la primera distribución de la información recopilada.
- La extracción de información desde los sistemas de presentación de datos para su representación en la aplicación web. Para aportar un mayor beneficio se buscará realizar una presentación de la información recogida de forma clara y eficiente en una plataforma que permita la interacción a usuarios comunes.

Con el propósito de lograr una infraestructura funcional y conseguir que la representación de los datos aporte valor, se pretende contar con la colaboración de grupos de expertos en educación, que nos proporcionarán los requerimientos necesarios para conseguir diseñar una infraestructura útil que se adapte a sus necesidades partiendo de escenarios reales de utilización.

La colaboración antes mencionada junto con la toma de decisiones sobre las herramientas de desarrollo y despliegue de la arquitectura del entorno de back-end y la puesta en marcha de los equipos y la implantación de dicho entorno en ellos, permitirá obtener una primera definición de la infraestructura que se quiere implementar en este

TFG y la realización de un primer prototipo que pueda servir en un futuro como base para la elaboración de la infraestructura en un entorno completamente operativo y optimizado.

1.3. Estructura de la memoria

En este primer capítulo se ha introducido el contexto en el que se realizará el TFG y los objetivos que pretende. En el capítulo 2, *Materiales y métodos*, se enumeran las tecnologías y protocolos utilizados para la implementación del trabajo, exponiendo los motivos de la elección y analizando sus fortalezas y debilidades.

El capítulo 3, *Diseño de la infraestructura*, comienza con una visión general de la infraestructura a desarrollar, describiendo su arquitectura e identificando sus partes fundamentales. Tras ello, se explica detalladamente la estructura, configuración y programación de la infraestructura de recepción, almacenamiento, gestión y presentación desarrollada.

En el capítulo 4, *Resultados*, se describe la metodología y el razonamiento con el que se realizan los informes para la presentación de datos. Finalmente, esta memoria concluye con un capítulo 5 de *Conclusiones y líneas futuras* que se pueden llevar a cabo tratando este trabajo como base.

También se incluyen tres anexos que abordan en detalle la implementación de los servidores explicados en el capítulo 3, y un cuarto anexo que muestra el intercambio de mensajes.

2. MATERIALES Y MÉTODOS

En este capítulo se describen los elementos que componen la arquitectura *software* de la infraestructura desarrollada. La toma de decisiones sobre las tecnologías que se han escogido para la realización de este trabajo se basa en la búsqueda de herramientas de uso libre que permitan la mayor interoperabilidad posible, un despliegue rápido pero eficaz y válido para una primera construcción del entorno *back-end*.

Este entorno se ha desarrollado en su totalidad en un equipo de un laboratorio de la EINA con dirección IP: 155.210.158.24 y con nombre de dominio registrado por la universidad: gtc1pc1.cps.unizar.es a través de conexión remota para permitir el trabajo a distancia. El entorno creado responde a una arquitectura cliente-servidor tal como el mostrado en la *Fig. 1*.

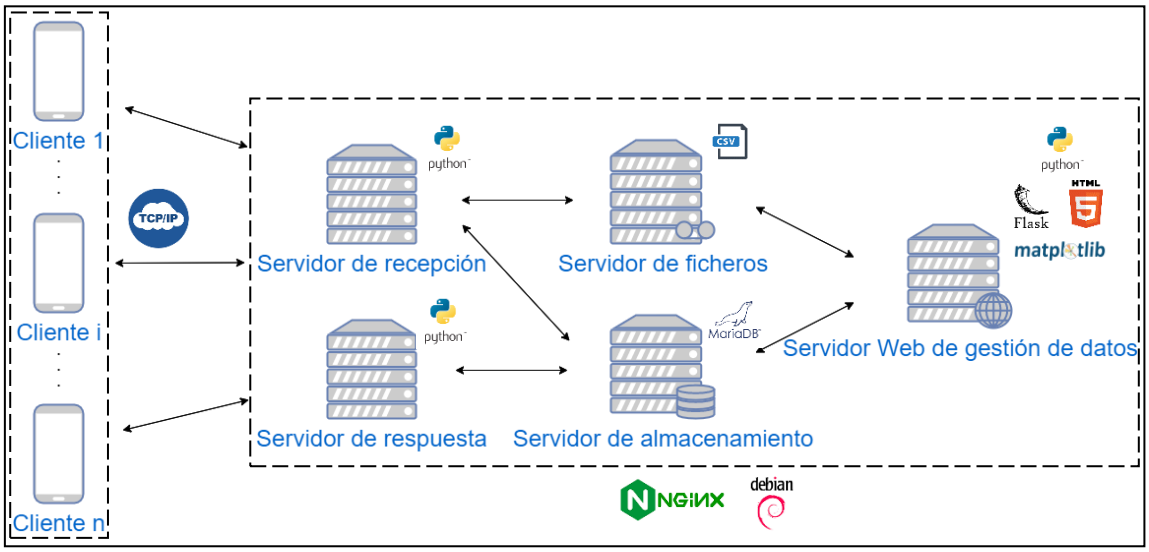


Figura 1. Arquitectura software de la infraestructura desarrollada

En esta arquitectura los clientes se corresponden con los dispositivos móviles que registraran las interacciones de los usuarios con ellos, estos datos serán enviados al servidor de recepción que se comunicara con el servidor de ficheros y almacenamiento para registrar los mensajes recibidos. A su vez, el servidor de respuesta proporcionara realimentación a los clientes con los datos existentes en el servidor de almacenamiento. Por último, el servidor web de gestión de datos se conectará a los servidores de ficheros y almacenamiento para proporcionar a los expertos una herramienta de acceso a la información recogida y procesada por dicho sistema de servidores.

A continuación, se detallan los distintos elementos necesarios para construir la arquitectura *software* desarrollada.

2.1. Sistema operativo: *Debian*

Un sistema operativo es el *software* principal de un sistema informático que gestiona los recursos de *hardware* y provee servicios a los programas de aplicación de *software*, ejecutándose en modo privilegiado respecto de los restantes.

El sistema operativo seleccionado para el equipo en el que se ha realizado el proyecto es *Debian GNU/Linux* [2]. Este sistema ofrece un sistema operativo basado en software libre y de distribución gratuita de tipo Unix (GNU), es decir, es un sistema operativo portable, multitarea y multiusuario, con el núcleo *Linux*. Es uno de los sistemas operativos más estables y seguros en la actualidad que posee miles de paquetes pre-compilados estables.

2.2. Base de datos: *MariaDB* y *DBeaver*

El sistema de almacenamiento estará constituido por una base de datos. Se llama base de datos a un conjunto de información perteneciente a un mismo contexto, ordenada de modo sistemático para su posterior recuperación, análisis y/o transmisión.

MariaDB [3] es el sistema de administración de bases de datos de código abierto, desarrollada por los creadores de *MySQL*, con el que Debian trabaja por defecto en la versión que hemos instalado en el equipo. *MariaDB* presenta mejoras que aumentan la velocidad y el rendimiento, reducen el número de alertas del compilador y progresa en la facilidad de uso.

Como interfaz gráfica para gestionar la base de datos se utiliza *DBeaver* [4], que es una herramienta universal para bases de datos. Es muy potente, no solo por estar disponible con una rápida instalación en cualquier sistema operativo, sino también por la cantidad de sistemas gestores de bases de datos que soporta: *MySQL*, *MariaDB*, *PostgreSQL, SQLite, Oracle, SQL Server, Sybase, MS Access…*

El objetivo fundamental del interfaz es facilitar todas las tareas comunes, no sólo de administración de la base de datos. También actúa como capa de aislamiento del esquema conceptual (tablas y relaciones) de la base de datos real siendo el flujo de datos entre el interfaz y el sistema transparente para el usuario.

2.3. Servidor web como proxy inverso: *NGINX, uWSGI y Certbot*

Nginx [5] es un servidor web/*proxy* inverso ligero de alto rendimiento. Se trata de un *software* multiplataforma que podremos instalar fácilmente en el equipo con sistema operativo *Debian*. En este trabajo se utilizará como *proxy* inverso, es decir, como servidor *proxy* situado en el alojamiento de los servidores web. Todo el tráfico procedente de Internet y con destino en alguno de esos servidores web es recibido por el servidor *proxy* que lo reenvía al recurso específico lo que aporta una capa adicional de seguridad y protección y facilita el cifrado SSL, por ejemplo, a través de *Certbot*. Además, el *proxy* puede distribuir la carga entre varios servidores web de manera eficiente y puede descargar trabajo a dichos servidores almacenando contenido estático como imágenes u otro contenido gráfico y contenido dinámico pero que pueda ser reutilizable en alguna medida. La principal ventaja de *Nginx* como servidor es que consume muchos menos recursos que la mayoría de los servicios que hacen su misma función al servir contenido estático.

uWSGI [6] es una aplicación de *software* que "tiene como objetivo desarrollar una pila completa para crear servicios de alojamiento", es decir, se trata de un

contenedor de servidor de aplicaciones que tiene como objetivo proporcionar una pila completa para desarrollar e implementar aplicaciones y servicios web. Lo vamos a usar para servir aplicaciones web *Python* junto con el servidor web *Nginx*, ya que este ofrece soporte directo para el protocolo binario nativo de comunicación con otros servidores 'uwgsi'. El flujo de datos será:

Cliente HTTP <-> Nginx <-> uWSGI <-> Aplicación Python

Para aportar seguridad a nuestro servicio web, como ya hemos dicho, con *Nginx* se puede hacer de forma sencilla a través de *Certbot* [7]. Se trata de una herramienta para gestionar de forma automática certificados TLS/SSL (Transport Layer Security/ Secure Sockets Layer) y automáticamente configurar el cifrado HTTPS (Hypertext Transfer Protocol Secure) en nuestro servidor web. Permite obtener los certificados de *Let's Encrypt* y autoconfigurar nuestro sistema. *Let's Encrypt* es actualmente una de las CA más grandes a nivel mundial, permite a cualquier persona obtener un certificado TLS/SSL para incorporar a su página web el protocolo HTTPS de forma fácil y gratuita.

2.4. Lenguaje programación de servidores: *Python* y *Pycharm*

Para implementar los servidores se va a utilizar *Python* [8] como ya habíamos introducido en el punto anterior. *Python* es un lenguaje de programación de escritura rápido, escalable, robusto y de código abierto, orientado a objetos con una sintaxis sencilla que cuenta con una amplia biblioteca de herramientas, que hacen de *Python* un lenguaje de programación único. Una de las ventajas principales es la posibilidad de crear un código con gran legibilidad que facilita su comprensión e implementación. Pero, sobre todo, *Python* es un lenguaje gratuito con una gran comunidad en activo, que proporciona soporte. Además, *Python* es un programa con SQL embebido que va a permitir implementar la conexión de los diferentes servidores con la base de datos en el entorno de *back-end* desarrollado de manera sencilla.

PyCharm [9] es un entorno de desarrollo integrado (IDE), específicamente utilizado para el lenguaje Python que proporciona una finalización del código inteligente, inspecciones del código, indicación de errores sobre la marcha y arreglos rápidos, así como refactorización de código automática y completas funcionalidades de navegación.

A continuación, se describen brevemente las bibliotecas más importantes que se han utilizado para el desarrollo del servidor web de presentación de datos.

2.4.1. Aplicación web: *Flask*

La biblioteca *Flask* [10] nos permite crear de manera sencilla aplicaciones web con *Python* bajo el patrón MVC (Modelo Vista Controlador), es decir, diferenciando y separando lo que es el modelo de datos (base de datos), la vista (página HTML (*HyperText Markup Language*)) y el controlador, que es donde se gestiona las peticiones de la aplicación web. *Flask* es un *microFramework* que nos da un esquema de trabajo y una serie de utilidades y funciones que facilitan y abstraen la construcción de páginas web dinámicas proporcionando las herramientas necesarias para crear una aplicación

web funcional que, si en algún momento necesita nuevas funcionalidades, puede utilizar un conjunto muy grande de extensiones que se pueden instalar con *Flask*.

Una de las ventajas más importantes de *Flask* es que es compatible con *wsgi* que es el protocolo que utilizaremos en los servidores web de *Nginx* para servir las páginas web escritas en *Python*. Otra de las ventajas es que cuando se trabaja con aplicaciones web hechas en *Python* se posee un controlador que recibe todas las peticiones que hacen los clientes. Se tiene que determinar a qué ruta está accediendo el cliente para ejecutar el código necesario y *Flask* lo facilita con un buen manejo de rutas. Además, soporta de manera nativa el uso de cookies seguras.

Usaremos el comando *@app.route* para convertir una función *Python* regular en una función vista de *Flask*, que transforma el valor de devolución de la función en una respuesta HTTP (Hypertext Transfer Protocol) que se mostrará mediante un cliente HTTP, como un navegador web. Por ejemplo, el valor '/' en *@app.route(*) indicará que esta función responderá a las solicitudes web para la URL /, que es la URL principal. Con el comando *render_template()* podremos establecer la páginas en formato HTML que se visualizan en cada una de las rutas. *Flask* nos permitirá que nuestras plantillas aparte de contener código HTML tengan comando de *Python* embebidos. Esto nos va a aportar la interactividad y el dinamismo necesario para realizar el servicio web.

2.4.2. Librería de representación de gráficas: *Matplotlib*

Para la representación de los datos en los diferentes tipos de gráficas que se usan en el servidor de presentación se va a utilizar la librería de *Python Matplotlib* [11]. Se trata de una librería para la generación de gráficos y visualizaciones estáticas, animadas e interactivas a partir de datos contenidos en listas o *arrays*.

Las gráficas creadas de forma dinámica con la información de la base de datos se guardarán como imágenes y se insertaran en las plantillas que componen la página web gracias a la conectividad entre *Python* y la base de datos, y *Python* y el servidor web con *Flask*.

2.4.3. Librería de conexión

La conexión de los servidores con *MariaDB* y entre servidores, se realizará con la librería *mariadb*, que permitirá la administración de la base de datos mediante comandos SQL embebidos y el módulo *socket*, que posibilitará intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

2.5. Lenguaje en el servidor web: HTML y CSS

El lenguaje utilizado para la escritura de las páginas web será HTML ya que es el componente más básico de la web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia y presentación de una página web o su funcionalidad y su comportamiento. En este sentido utilizamos CSS (*Cascading Style Sheets*), un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un

lenguaje de marcado. Asimismo, en este trabajo para aportar la funcionalidad a las páginas web, se utilizará la potencialidad de *Flask*, que permite utilizar comandos de *Python* directamente en los archivos HTML (*templates*).

2.6. Ficheros *raw* de la base de datos: csv

Para la generación de ficheros *raw* de las tablas de la base de datos que permiten la exportación del sistema de almacenamiento se va a utilizar el formato de ficheros csv (*comma-separated values*). Son archivos de texto en el cual los caracteres están separados por comas, haciendo una especie de tabla en filas y columnas. Las columnas quedan definidas por cada coma, mientras que cada fila se define mediante una línea adicional en el texto. Este tipo de fichero permite mover las tablas de la base de datos a otras bases de datos o a programas de hojas de cálculo y de computación para realizar su procesado.

2.7. Protocolo de comunicación: *sockets* TCP

Para la comunicación cliente-servidor entre los dispositivos móviles y los servidores del entorno *back-end* se va a utilizar la comunicación vía *socket* TCP (*Transmission Control Protocol*). Este protocolo de transporte permite que dos equipos se conecten e intercambien flujos de datos garantizando la entrega de datos y paquetes en el mismo orden en que se enviaron de forma confiable y sin errores. TCP tiene control de concurrencia, lo que significa que las solicitudes iniciales serán pequeñas, aumentando de tamaño a los niveles de ancho de banda que los ordenadores, servidores y redes puedan soportar.

Con la biblioteca estándar de *Python socket* se crean conectores que se pueden configurar para que actúen como un servidor y escuchen mensajes entrantes, o se conecten a otras aplicaciones como un cliente. Tras establecerse la conexión TCP, la comunicación será bidireccional, es decir, se pueden recibir y enviar mensajes.

3. DISEÑO DE LA INFRAESTRUCTURA

El escenario principal para el que se desarrolla esta infraestructura se plantea en entornos de investigación con el fin de conseguir mejorar la QoX en el área de la utilización de nuevas tecnologías. Se pretende conseguir que los expertos tengan acceso a una aplicación web que, en cualquier momento y lugar, les proporcione tener a su alcance toda la información actualizada necesaria para realizar los estudios pertinentes sobre las conductas de utilización de dispositivos móviles por usuarios finales.

Por ello, se toma como punto de partida en este capítulo de diseño de la infraestructura la valiosa información adquirida en las reuniones llevadas a cabo con expertos en la materia (Grupo EducaViva de la Universidad de Zaragoza). A continuación, se describirán los requerimientos de la infraestructura, así como su arquitectura explicando en detalle la funcionalidad y la implementación de cada una de las partes que la componen.

3.1. Requerimientos

Este apartado recoge la información proporcionada por los expertos en las reuniones antes mencionadas. Con ello se ha llegado a la siguiente descripción de las funcionalidades del sistema:

- Registrar los datos personales de los usuarios finales de los dispositivos móviles (edad, sexo, nacionalidad…) así como el identificador asignado a cada uno según el grupo de estudio al que pertenecen. Estos datos serán necesarios para poder sacar conclusiones acerca de las conductas tanto individuales como de grupo.
  Además de esta información, cada usuario registrará un alias personal y se le asignará un número de usuario único para permitir su identificación en el sistema de almacenamiento.
- Permitir que los usuarios puedan tener la aplicación en diferentes dispositivos móviles.
- Recoger y almacenar los mensajes de datos de uso de los dispositivos recibidos de los usuarios finales. Será necesario que estos mensajes contengan la fecha en la que ocurrió la interacción con el dispositivo, el tipo de interacción realizada y los detalles de esta.
- Extraer diferentes datos de las interacciones de los usuarios con los dispositivos móviles para permitir la realización de los estudios de la conducta en la utilización de dichos dispositivos.

Realizando el análisis pertinente se ha considerado obtener las siguientes interacciones para la cuantificación de criterios educativos:

- Contenido emocional
  - Detección de emoticonos.
- FOMO: ansiedad a perderse algo
  - Nº de desbloqueos del móvil.
  - Nº de veces que entro en las redes sociales y tiempo en ellas.

    - Tiempo en borrar/acceder a las notificaciones.
    - Nº de publicaciones (de cualquier tipo) que se hacen.
- Multitarea
    - En diferentes dispositivos: Mediante la identificación por parte del usuario podemos ver las conexiones que se han hecho de manera simultánea en ambos dispositivos.
    - En el mismo dispositivo: Coincidencia en tiempo de varias conexiones.
- *Phubbing*
    - Mediante geolocalización: Viendo si varios dispositivos de estudio se encuentran cerca y el tipo de conexiones y frecuencia en ese tiempo. (No viable en nuestra infraestructura)
    - A partir de una suposición temporal: Estudiando intervalos temporales en los que la probabilidad de interacción con otras personas es mayor y, de igual manera que en el caso anterior, observando las conexiones y frecuencias de las mismas en ese intervalo.

El primer y segundo grupo de interacciones llegará directamente como eventos desde el dispositivo móvil, mientras que el tercer y cuarto grupo se podrán obtener con el estudio de los datos presentados.

3.2. Arquitectura de la infraestructura

La arquitectura de la infraestructura desarrollada sigue un esquema cliente-servidor. Un cliente se corresponde con los dispositivos móviles de distintos usuarios, donde se generará la información de utilización sobre dichos dispositivos y se adaptaran al formato convenido para la comunicación con el servidor. Cuando el usuario inicie sesión en la aplicación cliente, esta recopilará los datos de uso del dispositivo sin necesidad de que dicho usuario interactúe con ella.

La otra perspectiva de cliente es la del experto que analiza los datos adquiridos y ubicados en el servidor, desde el navegador web para extraer la información en diferentes formatos. Se plantea la posibilidad de que haya diferentes expertos que utilicen la aplicación por lo que la clasificación por grupos permitirá la distinción de las diferentes líneas de investigación.

La parte del servidor está compuesta por el servidor de recepción de mensajes, que se encarga de introducir los datos adquiridos desde los clientes en la base de datos a la que está conectado y que constituirá el sistema de almacenamiento, el servidor web de gestión de datos que soporta toda la lógica de la aplicación para la presentación de la información con programas y páginas conectado también a dicho sistema de almacenamiento, el servidor de respuesta y, por último, un servidor de ficheros. En este extremo, que es el desarrollado ampliamente en este proyecto, reside toda la información y la funcionalidad de la aplicación.

La *Fig. 2* muestra un esquema funcional de la arquitectura del sistema descrita y la *Tabla 1* proporciona información de los objetivos fundamentales de cada uno de sus elementos.

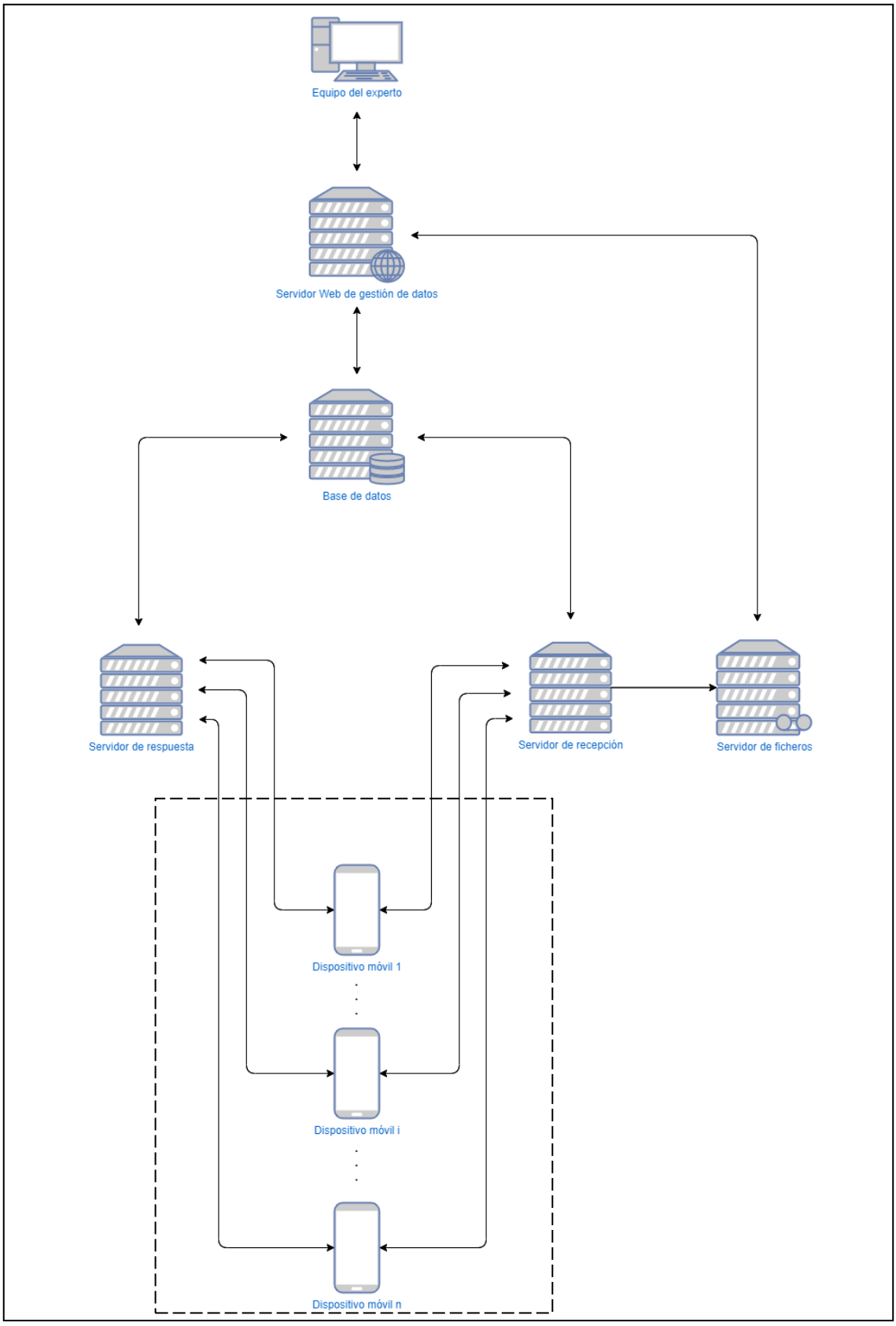


Figura 2. Esquema funcional de la arquitectura del sistema

Tabla 1. Objetivos de cada elemento de la arquitectura del sistema

Elementos de la Arquitectura	**Objetivos**
Base de datos	Constituye el sistema de almacenamiento de la información
Servidor de recepción	Se encarga de admitir las conexiones de los distintos clientes, clasificar los mensajes y actuar de acuerdo con lo establecido según su contenido.
Servidor de respuesta	Su cometido es enviar mensajes personalizados a los distintos clientes.
Servidor de gestión de datos	Constituye la estructura de la aplicación web y soporta toda su funcionalidad. Su propósito es proporcionar y representar la información de la base de datos de manera clara y eficiente.
Servidor de ficheros (log, csv)	Se encarga de almacenar en ficheros (log,) creados dinámicamente desde el servidor de recepción, todos los mensajes recibidos de los distintos dispositivos móviles de los usuarios finales. También almacena de forma temporal y dinámica los archivos *raw* (CSV) que se pueden descargar desde el servidor web.
Clientes de pruebas	Clientes de prueba para la simulación de la comunicación entre cliente y servidor. Cada uno de ellos realiza el envío de los diferentes tipos de mensajes.

En los siguientes apartados se presenta una descripción detallada de la infraestructura desarrollada.

3.2.1. Base de datos

La base de datos sustenta el sistema de almacenamiento de la infraestructura desarrollada en este proyecto. Dicha base de datos debe guardar toda la información de la aplicación. Por un lado, deberá registrar los datos personales de los usuarios de forma anonimizada y los mensajes de datos provenientes de sus correspondientes dispositivos móviles. Por otro lado, para hacer la traducción entre los identificadores numéricos y sus descripciones será necesario que la base de datos contenga otras tablas de información. Para hacer el diseño se han utilizado los requerimientos obtenidos en las reuniones ya mencionadas con los expertos en la materia.

- **Modelo y estructura de la base de datos**

En la *Fig. 3* se muestra el diseño de las tablas que se han creado en la base de datos y como se relacionan:

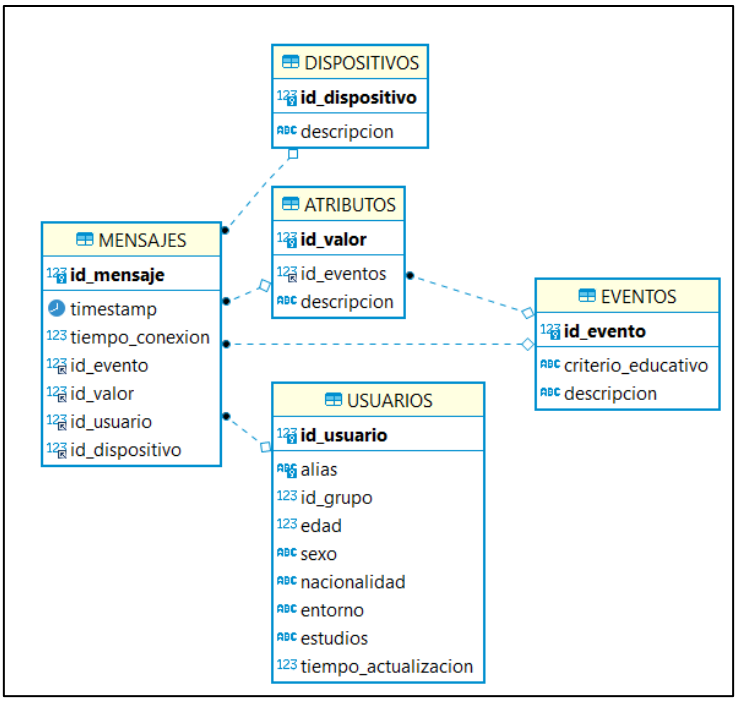


Figura 3. Diseño de tablas de la Base de Datos

En la *Tabla 2* se describe el objetivo de cada una de las tablas anteriores.

Tabla 2. Tablas de la base de datos

Nombre	**Descripción**
USUARIOS	Registro de la información de los usuarios finales de los dispositivos móviles.
DISPOSITIVOS	Tabla con los diferentes tipos de dispositivos móviles posibles.
MENSAJES	Almacenamiento de los mensajes de datos de uso de los dispositivos.
EVENTOS	Tabla con las posibles interacciones directamente extraíbles de los dispositivos móviles.
ATRIBUTOS	Tabla con los posibles valores que pueden tomar los eventos.

3.2.2. Servidor de recepción

El servidor de recepción debe ser capaz de estar escuchado de forma continua y funcionar de manera completamente automatizada. A su vez, es imprescindible que tenga la capacidad de recibir varias conexiones de clientes de forma simultánea. También, debe soportar la recepción de distintos tipos de mensajes en un formato prefijado para la comunicación con la aplicación residente en los dispositivos móviles de los usuarios finales. En la Tabla 3 se muestran los tipos de mensaje intercambiados con su correspondiente significado:

Tabla 3. Tipos de mensajes

Identificador	Descripción del mensaje
1	Registro de nuevo usuario
2	Datos de utilización del dispositivo
3	Inicio de sesión
4	Actualización de datos personales

Para cada mensaje entrante, el servidor de recepción proporcionará un mensaje de respuesta que aportará funcionalidad extra a la aplicación y servirá para tener fluidez en la conexión cliente-servidor que se ha establecido.

A continuación, se muestra el formato de los mensajes y se explica su contenido y funcionalidad[1]:

- Mensaje de registro de nuevo usuario:

1\n
descripción_dispositivo;alias;edad;sexo;nacionalidad;entorno;estudios;
identificador_grupo\n

El mensaje con ID = 1 se corresponde con la fase de registro del usuario en la aplicación. En este mensaje estarán contenidos los datos personales del usuario, junto al alias elegido, así como el identificador de grupo que se le haya asignado por parte de los investigadores y el tipo de dispositivo en el que se está utilizando la aplicación. En caso de que el alias ya se encuentre en uso se devolverá un error al cliente, de lo contrario, se enviara un mensaje de vuelta al usuario con el identificador de usuario y el identificador de dispositivo separados con ';'.

- Mensaje de datos de utilización del dispositivo:

2\n
id_usuario;id_dispositivo\n
timestamp;id_evento;id_valor;tiempo_conexion\n
… (N posibles líneas de datos)
timestamp;id_evento;id_valor;tiempo_conexion\n

El mensaje con ID = 2 corresponde al envío de datos sobre la utilización del dispositivo móvil. En este mensaje deberá estar indicado el identificador de usuario asignado en la fase de registro junto al identificador de dispositivo. A continuación, se incluirá un número de líneas indeterminado correspondiente a todos los eventos generados desde el terminal. Cada línea debe contener tanto el evento que se ha producido como el momento y la duración de dicho evento. El tiempo de actualización que debe usar el usuario para realizar el envío de datos de forma periódica se envía como respuesta en la comunicación.

[1] El carácter \n indica cambio de línea

- Mensaje de inicio de sesión:

3\n *alias;id_usuario;descripción_dispositivo\n*

El mensaje con ID = 3 atañe a la posible fase de inicio de sesión en diferentes tipos de dispositivos por un mismo usuario final. Este mensaje debe contener tanto el alias elegido de forma personal por dicho usuario como el identificador de usuario que se le ha sido asignado en el proceso de registro. Esto permite que el usuario se identifique de forma univoca en el servidor. También deberá enviar el tipo de dispositivo en el que se ha iniciado la sesión. Si el inicio de sesión se ha realizado de manera correcta se envía al cliente un mensaje de vuelta con el identificador de usuario y el identificador de dispositivo obtenido a partir de la descripción del tipo de dispositivo también incluida en el mensaje separados por ";". En caso contrario, se devuelve un mensaje de error.

- Mensaje de actualización de datos personales:

4\n *id_usuario;\n* *id_grupo;edad;sexo;nacionalidad;entorno;estudios\n*

El mensaje con ID = 4 se corresponde a la fase de actualización de datos personales por parte del usuario final desde la aplicación en cualquiera de los dispositivos posibles. En este mensaje, será necesario que se incluya el identificador de usuario y la información que se desea actualizar. Una vez que la actualización de los datos se ha realizado de manera exitosa se envía un mensaje de 'OK' al cliente.

- **Implementación servidor de recepción**

Para poder implementar un servidor de recepción capaz de estar esperando las conexiones de los distintos clientes, en primer lugar, es necesario crear un socket activo para la comunicación TCP. A este socket, hay que asignarle la dirección IP del equipo en el que se encuentra el servidor. Al tratarse de un servidor al que se pretende que se conecten los clientes es necesario asignar al socket un puerto fijo, se ha elegido el 9999. Además, como se espera que el número de conexiones sea elevado se establece una cola de espera para nuevas conexiones y se abre un nuevo hilo (*thread*) para cada conexión que posibilitará la concurrencia de conexiones.

A continuación, se muestra el funcionamiento completo del servidor representado mediante diagramas de flujo*. En la *Fig. 4* se puede ver el funcionamiento global del servidor de recepción.

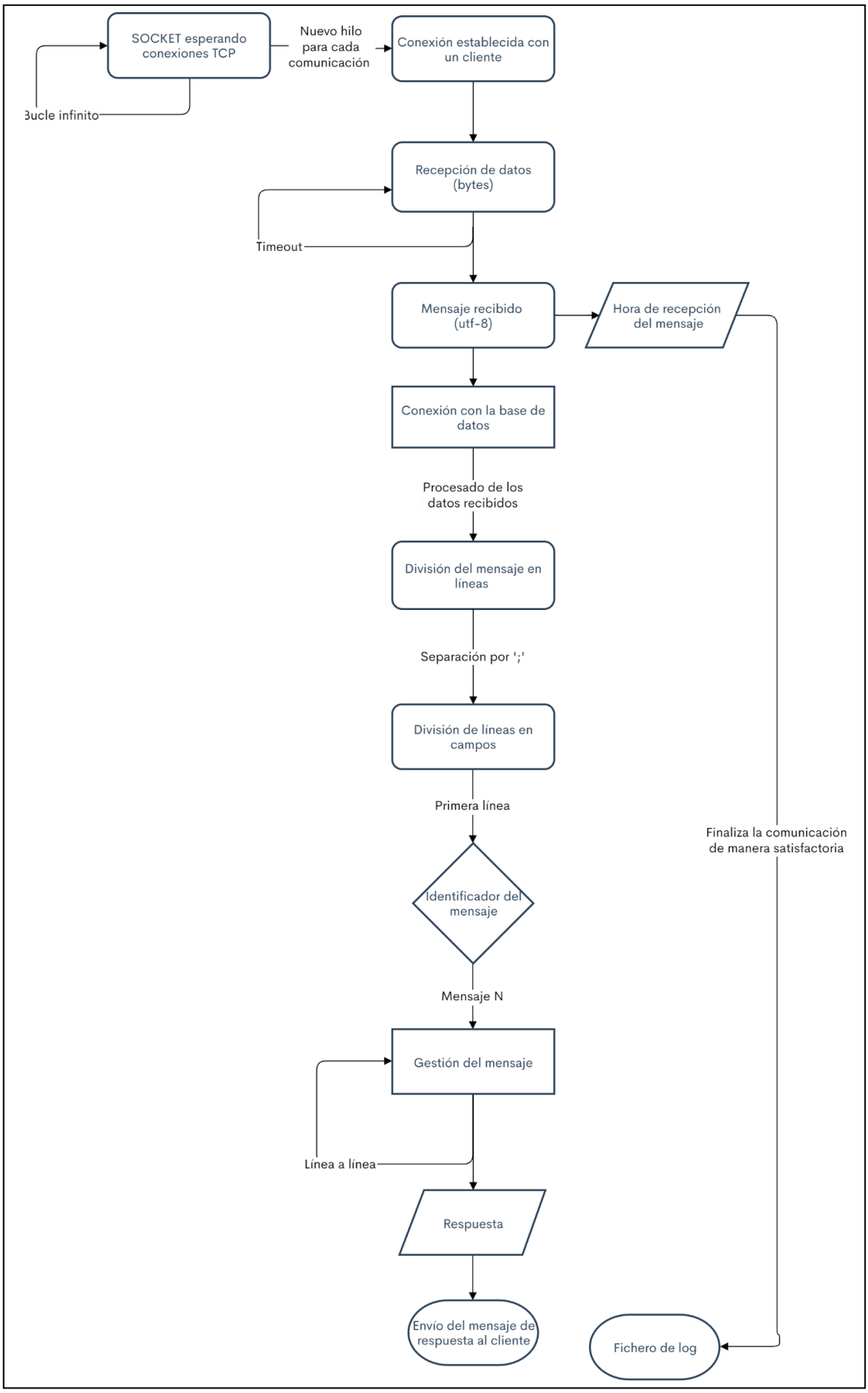


Figura 4. Funcionamiento general del servidor de recepción

En la *Fig. 5* se muestra el comportamiento detallado del servidor cuando llega un mensaje tipo 1.

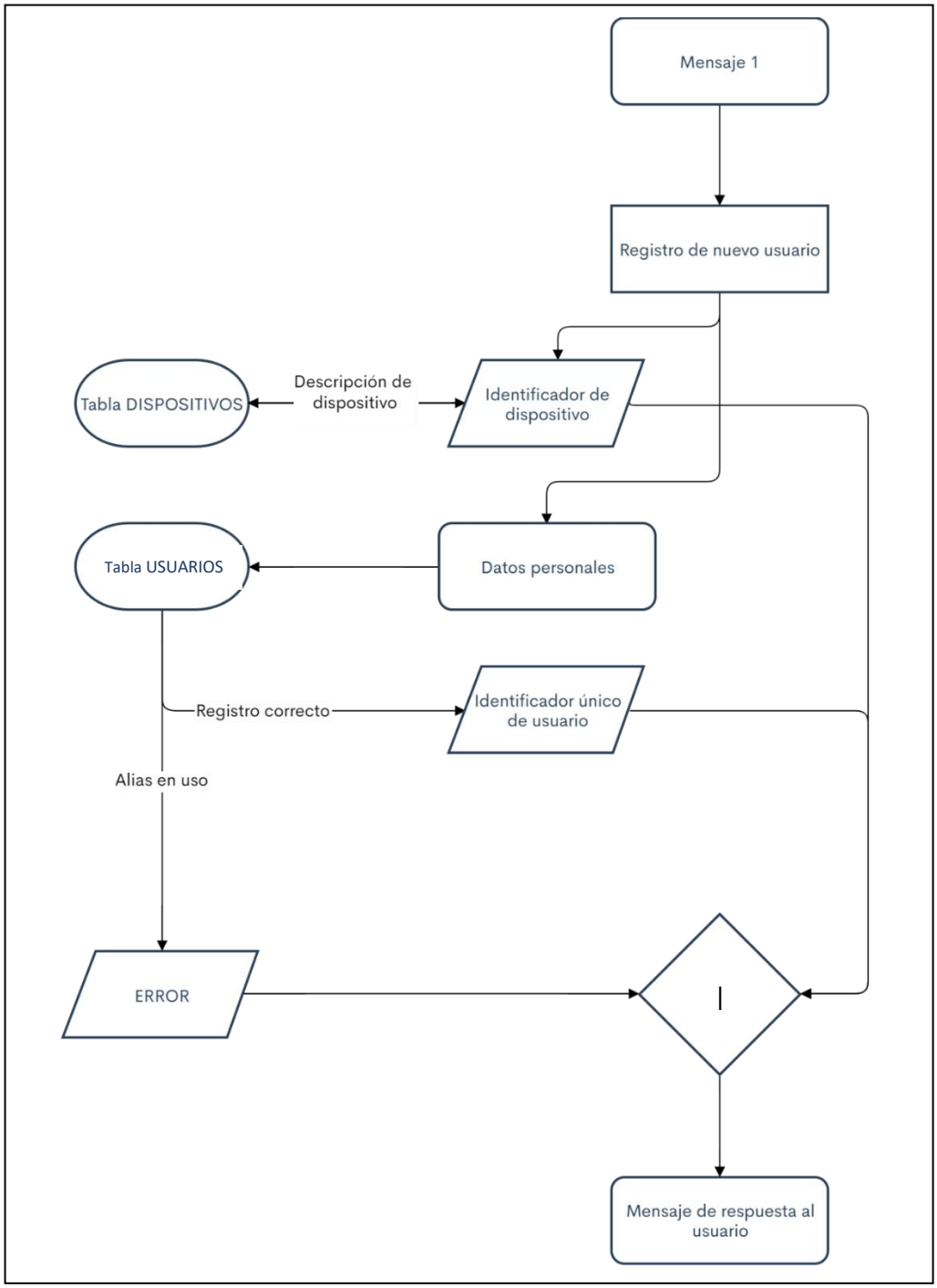


Figura 5. Funcionamiento del servidor de recepción para el mensaje tipo 1

En la *Fig. 6* se detalla el comportamiento del servidor ante la recepción de un mensaje tipo 2.

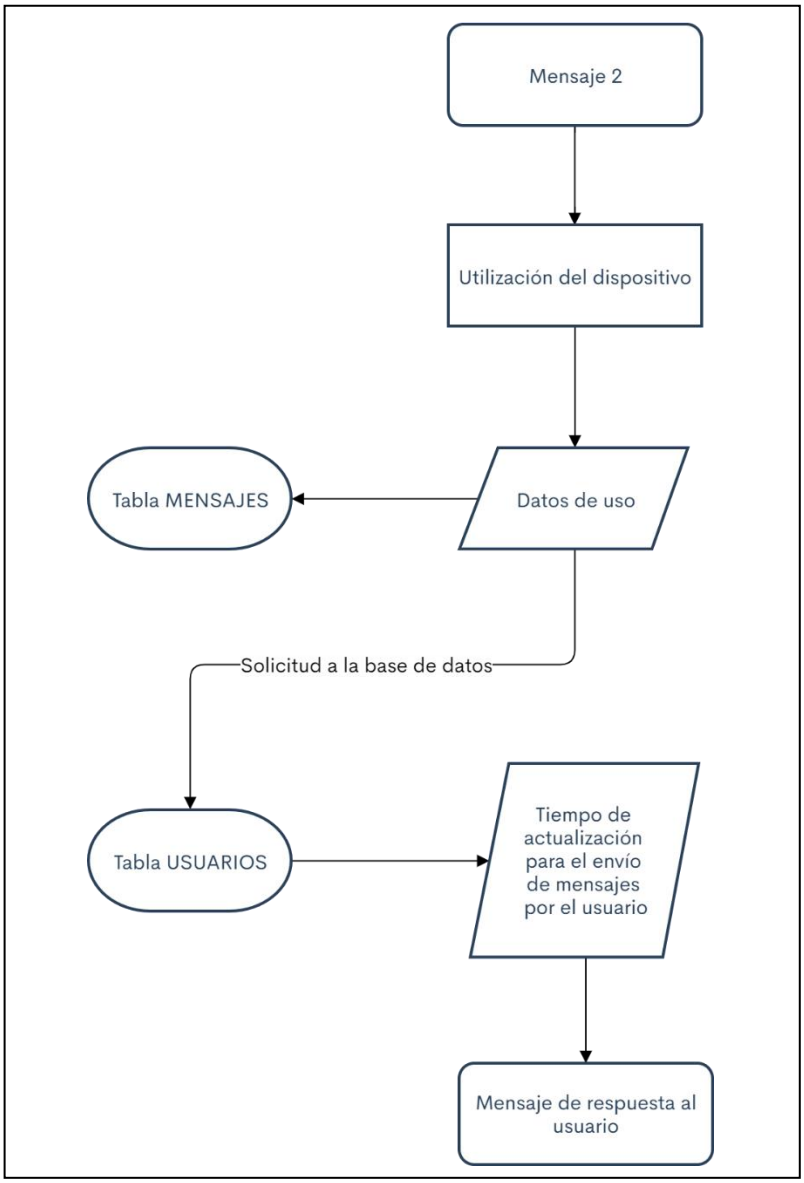


Figura 6. Funcionamiento del servidor de recepción para el mensaje tipo 2

En la *Fig. 7* se muestra el comportamiento del servidor cuando llega un mensaje tipo 3.

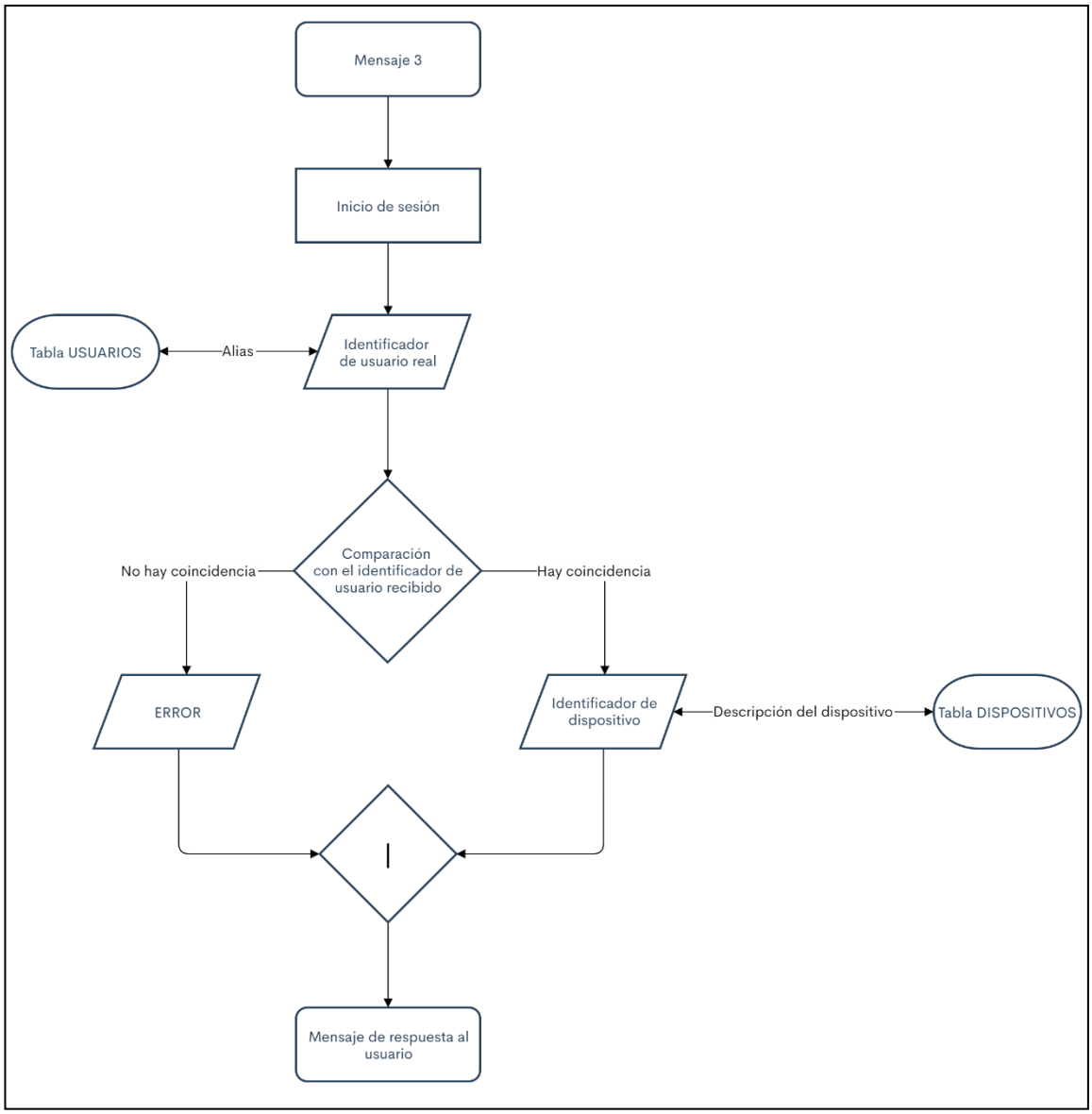


Figura 7. Funcionamiento del servidor de recepción para el mensaje tipo 3

En la *Fig. 8* se muestra detalladamente el comportamiento del servidor de recepción cuando llega un mensaje tipo 4.

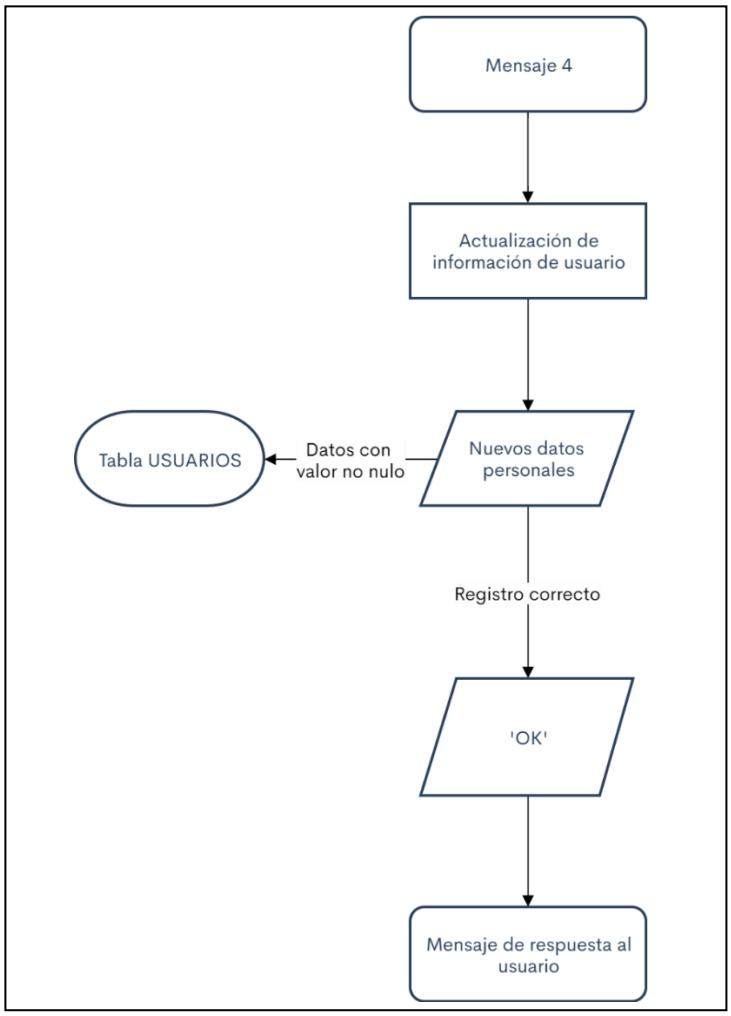


Figura 8. Funcionamiento del servidor de recepción para el mensaje tipo 4

En el *Anexo I* se encuentra una explicación completa y detallada de la implementación que se ha realizado para lograr el funcionamiento descrito para el servidor de recepción.

- **Intercambio de mensajes entre el servidor de recepción y un cliente**

En la *Fig. 9* se muestra el intercambio de mensajes que se producen en diferentes fases de la comunicación entre un cliente y el servidor de recepción.

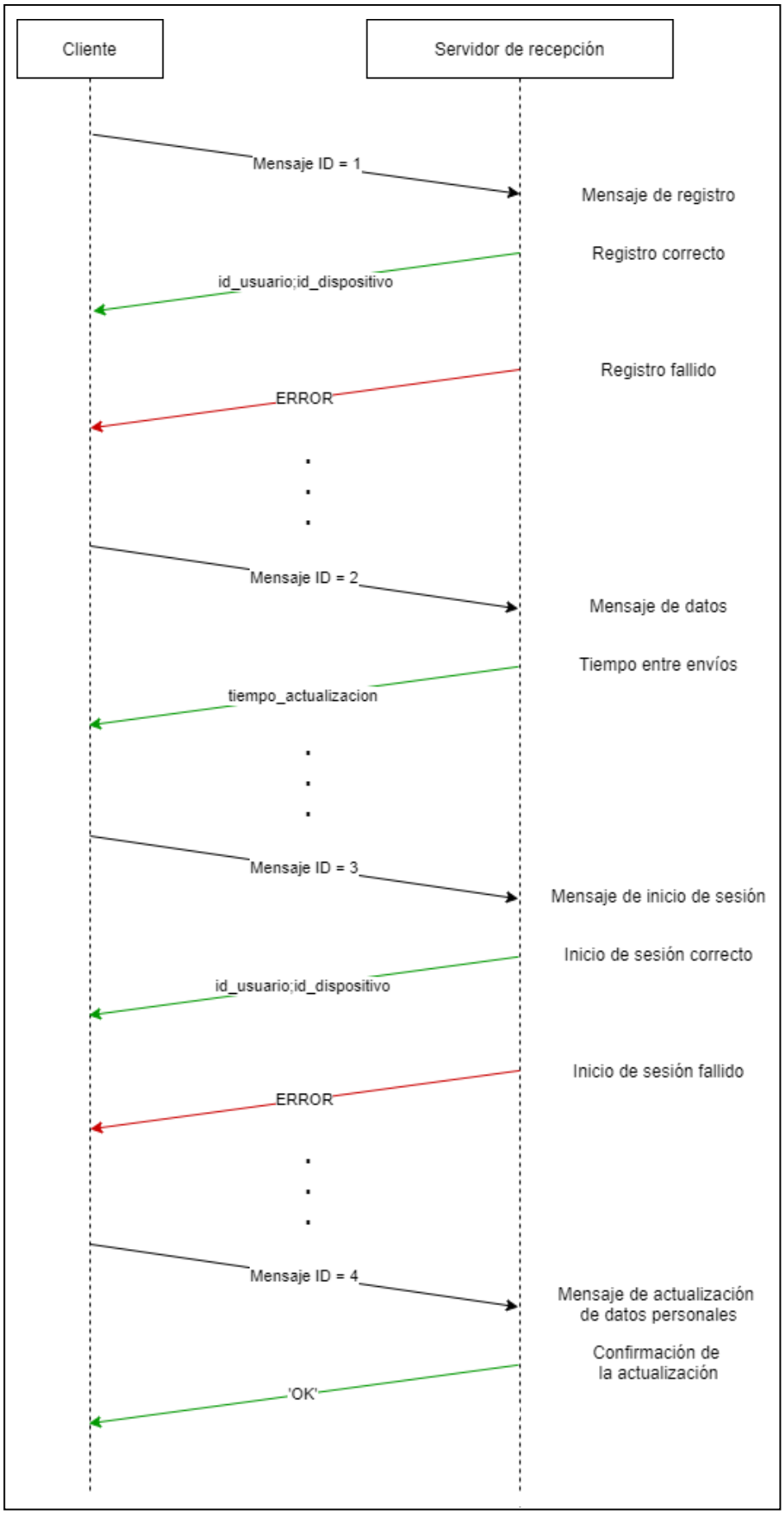


Figura 9. Intercambio de mensajes entre el servidor de recepción y un cliente

A continuación, se muestra un ejemplo de captura de intercambio real de mensajes. En la *Fig. 10* se ve el mensaje recibido por el servidor de recepción para el registro de un nuevo usuario, y en la *Fig. 11* el mensaje de respuesta del servidor al cliente tras el registro exitoso.

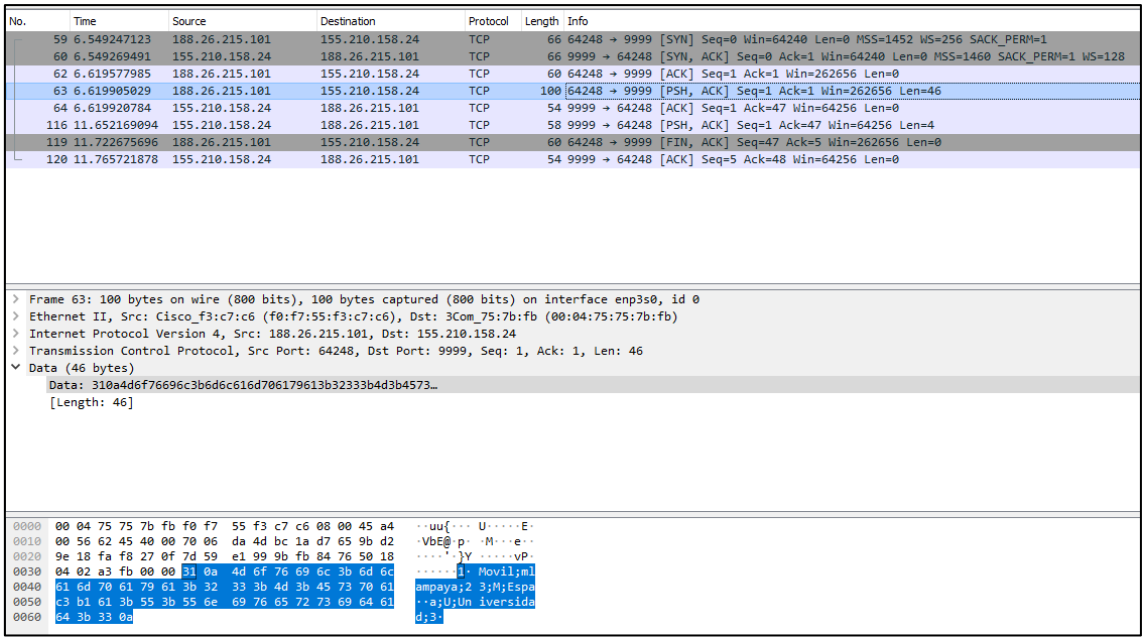


Figura 10. Captura mensaje de registro (ID = 1)

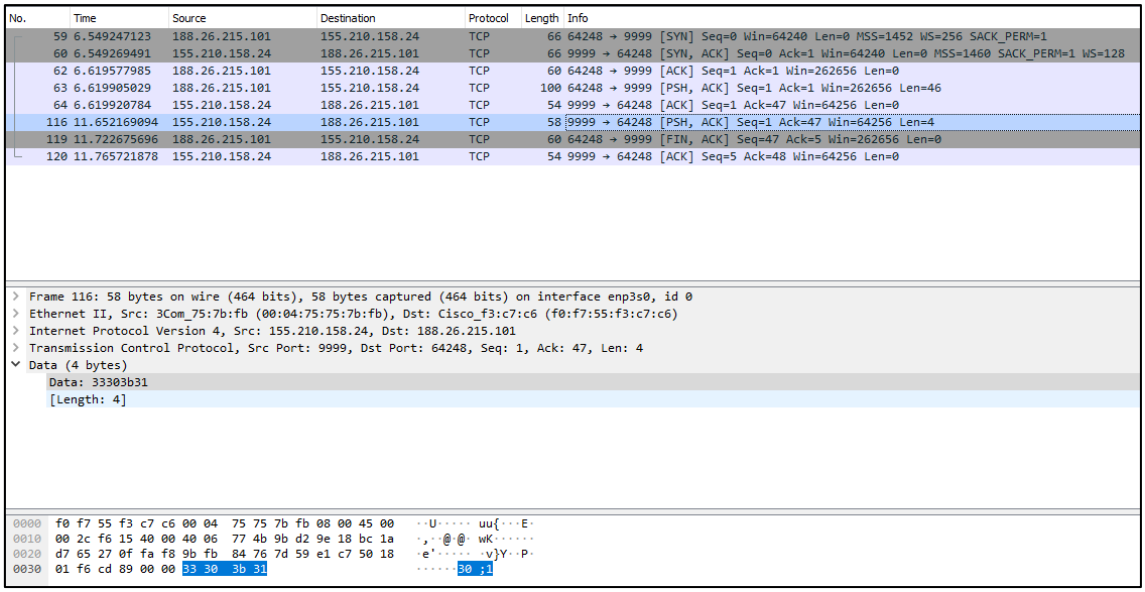


Figura 11. Captura mensaje de respuesta al registro

En el *Anexo II*, se puede ver las capturas relativas al resto de tipos de mensaje intercambiados en la comunicación.

3.2.3. Servidor de respuesta

El servidor de respuesta esta creado con el objetivo de mostrar la potencialidad de funcionamiento del entorno de *back-end* desarrollado. Como primera aproximación

de funcionamiento, deberá ser capaz de recibir múltiples conexiones concurrentes desde la aplicación de distintos dispositivos móviles. En esta conexión los clientes deberán enviar su número de identificación de usuario asignado. En función de este, se enviará un mensaje personalizado con el alias correspondiente, tal como se muestra a continuación:

HOLA, 'alias'\n *:X\n*

donde la notación :X de la 2º línea corresponde a:

Caritas para el mensaje				
:D	:)	:|	:|	:O

Con esto, se pretende demostrar la posibilidad de hacer un sistema de retroalimentación al usuario que, de forma personalizada, le permita conocer su conducta respecto a la utilización de su correspondiente dispositivo móvil e incluso poder recibir educación por parte de los expertos.

- **Implementación servidor de respuesta**

Para poder implementar el servidor de respuesta como en el caso del servidor de recepción, deberemos crear un socket activo esperando conexiones TCP de distintos clientes en la dirección del equipo servidor. El puerto elegido para escuchar conexiones es el 6666. Para que este servidor sea capaz de atender un número elevado de conexiones concurrentes se crea un nuevo hilo para gestionar cada una de las comunicaciones y se establece una cola de espera. Para poder atender conexiones de manera ininterrumpida hay un bucle infinito en el cual se crean e inician dichos hilos.

En la *Fig. 12*, que se encuentra a continuación, se describe esquemáticamente mediante un diagrama de flujo el funcionamiento del servidor de respuesta.

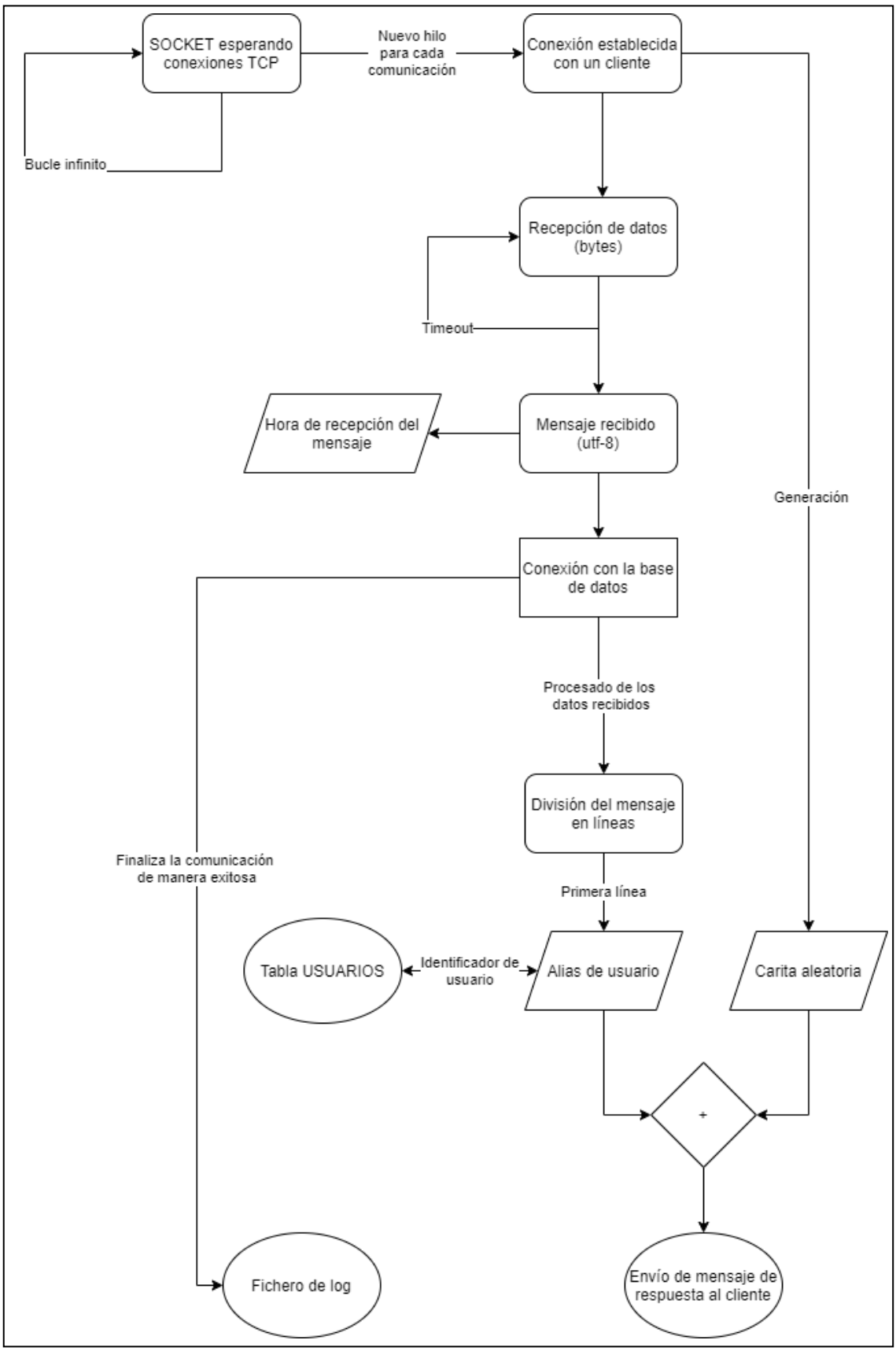


Figura 12. Funcionamiento del servidor de respuesta

En el *Anexo III* se encuentra una explicación completa y detallada de la implementación que se ha realizado para lograr el funcionamiento descrito para el servidor de respuesta.

- **Intercambio de mensajes entre el servidor de respuesta y un cliente**

En la *Fig. 13* se muestra el mensaje recibido por el servidor de respuesta para la identificación del usuario, y en la *Fig. 14* el mensaje de realimentación del servidor al cliente de forma personalizada.

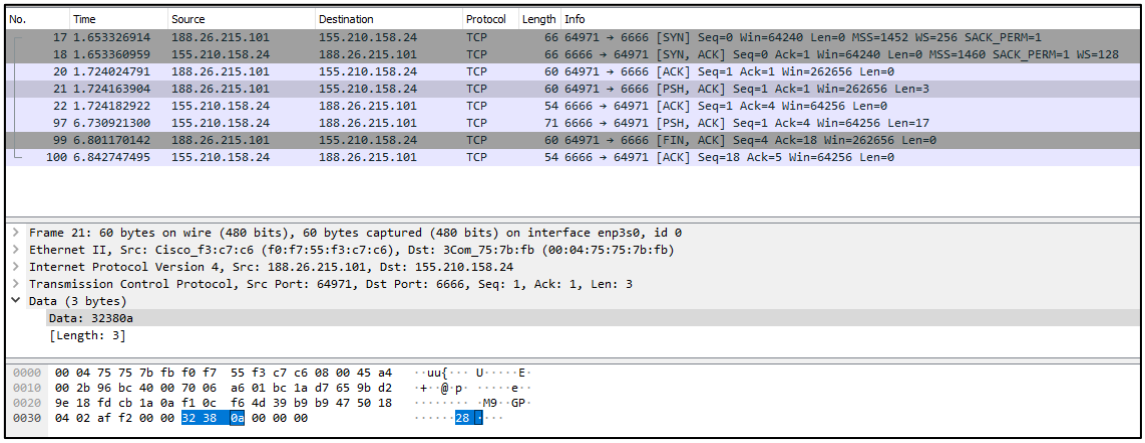


Figura 13. Captura mensaje de identificación de usuario

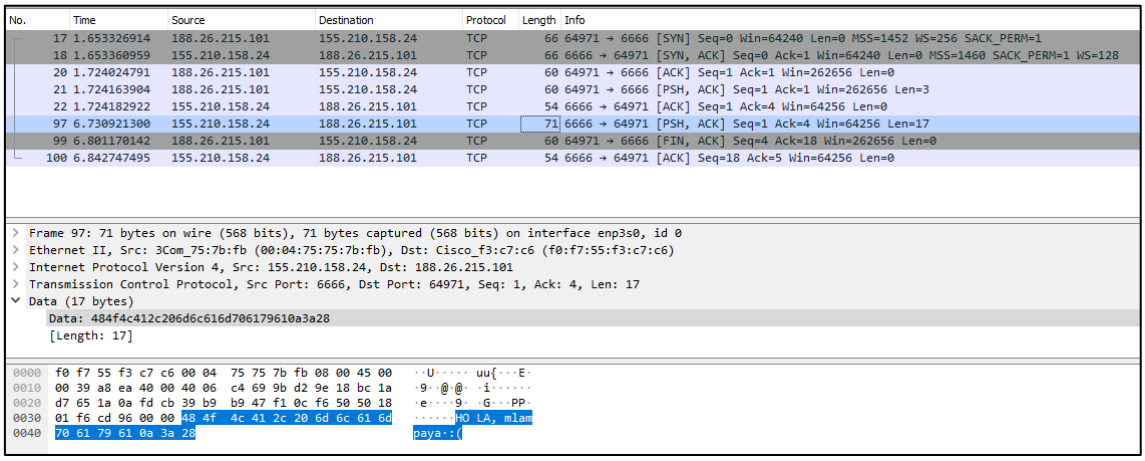


Figura 14. Captura mensaje de realimentación personalizado

3.2.4. Clientes de pruebas

Para probar los servidores de recepción y respuesta hemos creado una serie de clientes de prueba que simulan los posibles comportamientos que pueden tener los clientes en la comunicación con dichos servidores. Todos estos clientes de prueba tienen la misma estructura, ya que la única diferencia entre ellos debe ser el propio contenido de los mensajes que envían. A continuación, se incluye la *Tabla 4* con la correspondencia entre los diferentes clientes de pruebas y el funcionamiento del servidor a probar:

Tabla 4. Relación clientes de pruebas con el servidor a probar

Servidor	**Cliente de pruebas**	**Tipo de mensaje que envía**
Servidor de recepción	Cliente1	Mensaje de registro de usuario (ID: 1)
	Cliente2	Mensaje de datos de utilización del dispositivo (ID: 2)
	Cliente3	Mensaje de inicio de sesión (ID: 3)
	Cliente4	Mensaje de actualización de datos personales (ID: 4)
Servidor de respuesta	Cliente5	Mensaje de identificación de usuario

- **Implementación clientes de pruebas**

En primer lugar, se debe crear un socket, este socket se conecta a la dirección IP del servidor y el puerto 9999 caso de conectarnos al servidor de recepción, o el puerto 6666 caso de conectarnos al servidor de respuesta. Una vez se ha establecido la comunicación TCP con el servidor de manera correcta, se envía al servidor el mensaje de prueba. Después de realizar el envío el cliente se queda escuchando a espera de una respuesta por parte del servidor. Finalmente, cuando se recibe el mensaje devuelto por el servidor, se imprime su contenido por pantalla y se finaliza la conexión.

A continuación, en la *Fig. 15* se muestra el funcionamiento general de los clientes de pruebas.

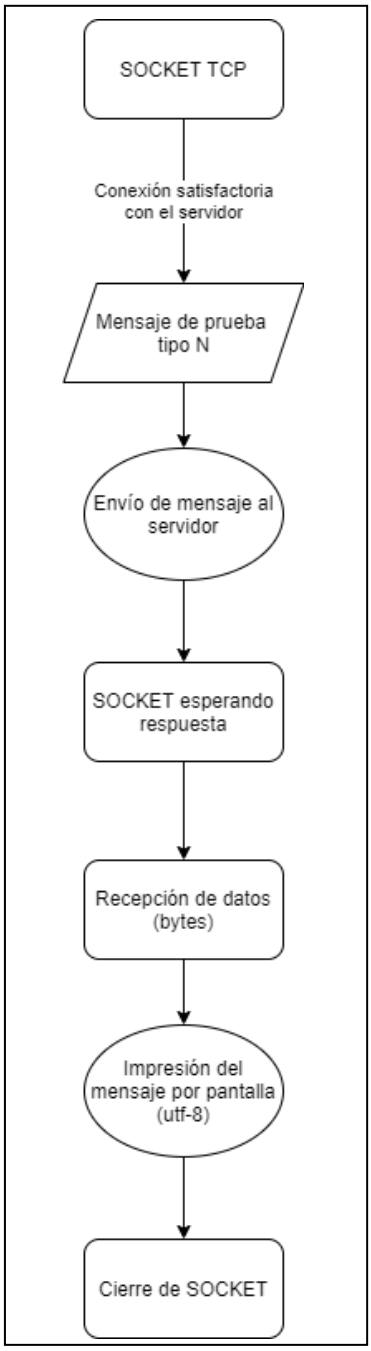


Figura 15. Funcionamiento general de los clientes de pruebas

3.2.5. Servidor web de gestión de datos

El servidor web de gestión de datos, además de proporcionar y representar la información de la base de datos, constituye la estructura de la aplicación web y ofrece al experto toda la funcionalidad expuesta en los apartados anteriores. Este servidor proporcionará la conexión al sistema de almacenamiento de forma transparente para el usuario común del servicio web y generará dinámicamente las páginas a las que acceden los investigadores, ya sea desde un PC o cualquier dispositivo con conexión a internet.

En la *Fig. 16* se muestra la página principal o de inicio del servidor web y posteriormente se explican sus principales componentes.

Figura 16. Página de inicio del servidor web de gestión de datos

En la parte de presentación de datos tenemos los apartados siguientes:

- *Muestra de usuarios.* Permite mostrar el número de usuarios totales que se hayan registrado en la base de datos, y su distribución mediante gráficas en función de sus datos personales.
  A su vez permitirá acceder a ver con más detalle la clasificación de usuarios mediante tablas en función del grupo de estudio al que pertenecen.

- *Búsqueda avanzada.* Permite obtener información filtrada del sistema de almacenamiento. El formulario está compuesto por diferentes filtros de búsqueda agrupados por datos de usuario, dispositivos, eventos y fecha (*Fig. 17*). También permite seleccionar una vista de los resultados individual o colectiva.

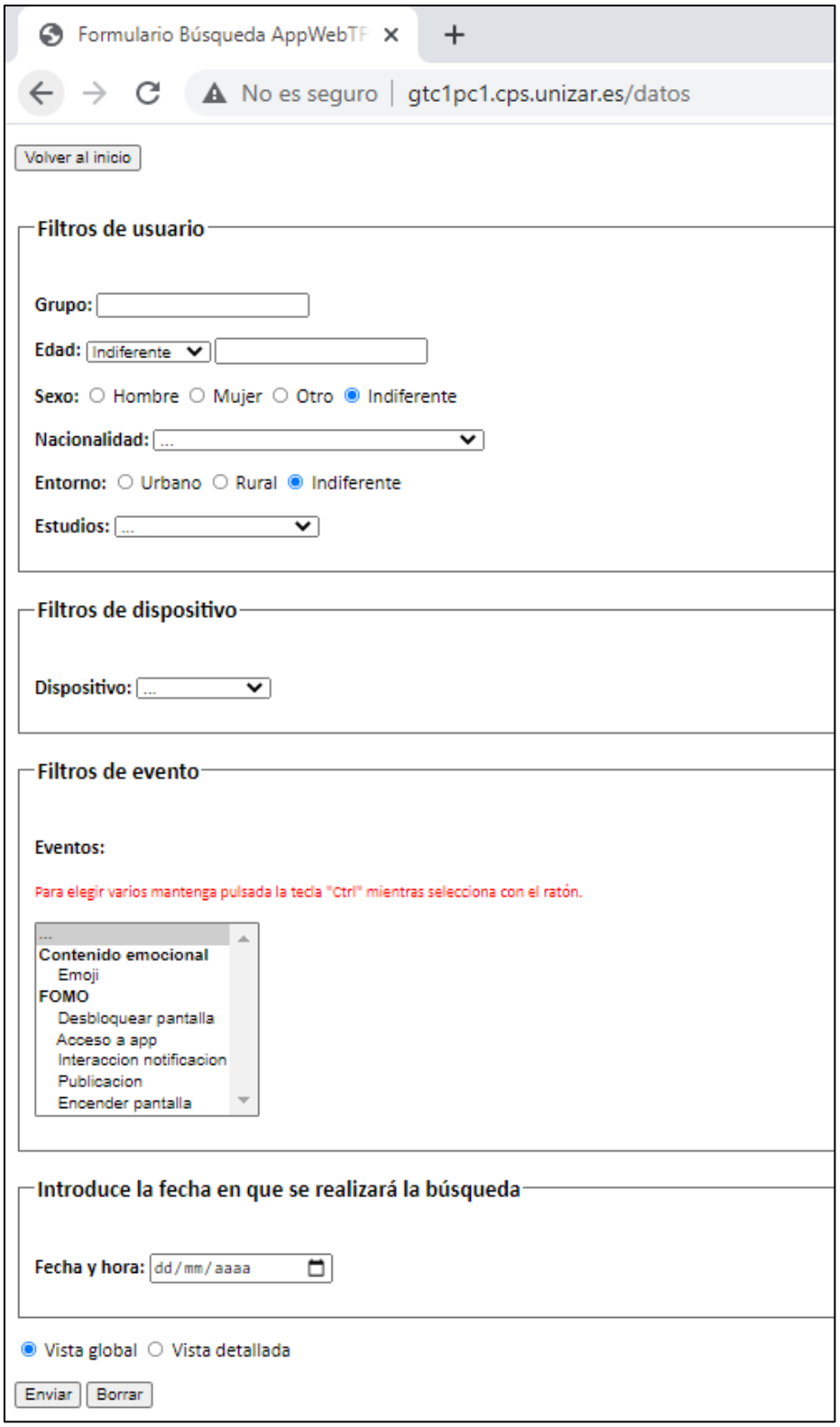


Figura 17. Formulario de búsqueda de datos

Como resultado de la búsqueda se mostrará el número total de mensajes de datos que se han recibido según los parámetros de la búsqueda y el número total de usuarios que han interactuado en dichos mensajes. A su vez, se podrá ver la distribución mediante gráficas de estos mensajes y de dichos usuarios. Finalmente, permitirá descargar una tabla en formato csv con el resultado de la

búsqueda realizada. Este fichero de salida permite que los expertos puedan generar o alimentar sus propias herramientas de análisis estadístico.

La metodología utilizada en la representación de los datos para el apartado de presentación se explicará de manera detallada y en su totalidad en el siguiente capítulo de este documento.

En la parte de ficheros raw tenemos el siguiente apartado:

- *Tablas de datos.* Permite visualizar las tablas de datos que contiene el sistema de almacenamiento y contendrá un enlace para descargar cada una de ellas en formato .csv con el fin de consentir la exportación de datos y, junto con las obtenidas en el apartado anterior permitir a los investigadores hacer un procesado de los datos independiente. Estos enlaces descargan los ficheros raw que contienen las tablas desde el servidor de ficheros que los almacena de forma temporal y dinámica. En la *Fig. 18* se muestra el formulario de selección de tabla de la base de datos.

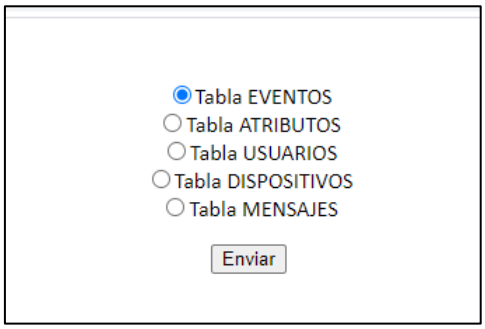


Figura 18. Formulario de elección de tabla

Por último, en la parte de mantenimiento tenemos el apartado siguiente:

- *Mantenimiento base de datos.* Permitirá a los superusuarios realizar el mantenimiento del sistema de almacenamiento accediendo a la base de datos con el fin de insertar, actualizar o borrar datos de cualquiera de sus tablas. Para restringir el acceso a los usuarios comunes del resto del servicio web, será necesario introducir las credenciales de acceso a la base de datos (*Fig. 19*), tras lo cual se podrá elegir la tabla y la acción a realizar (*Fig. 20*) y finalizar con la acción elegida (*Fig. 21, Fig. 22 y Fig. 23*).

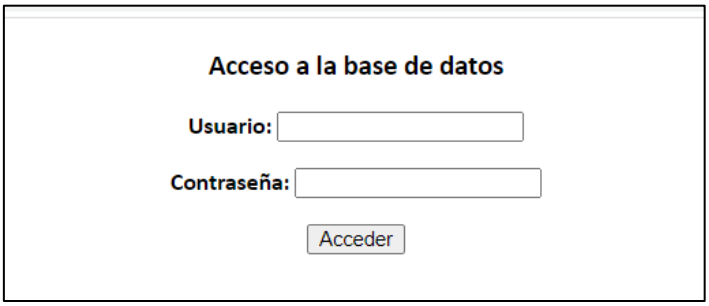


Figura 19. Formulario de acceso a la base de datos desde el Mantenimiento

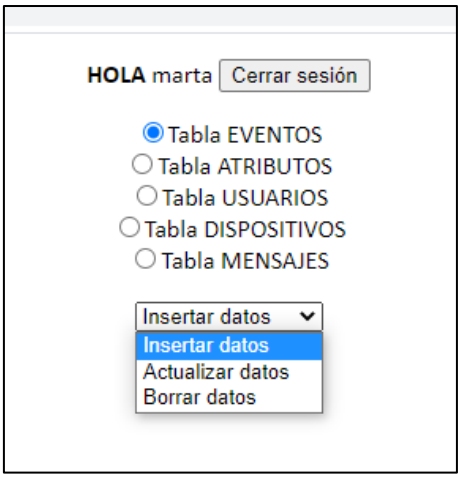


Figura 20. Formulario de elección de tabla y acción a realizar en la base de datos

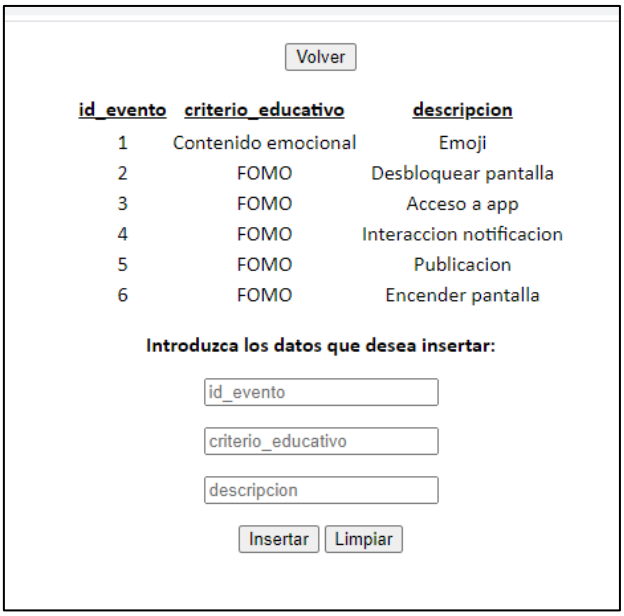


Figura 21. Formulario de insertar datos en tabla

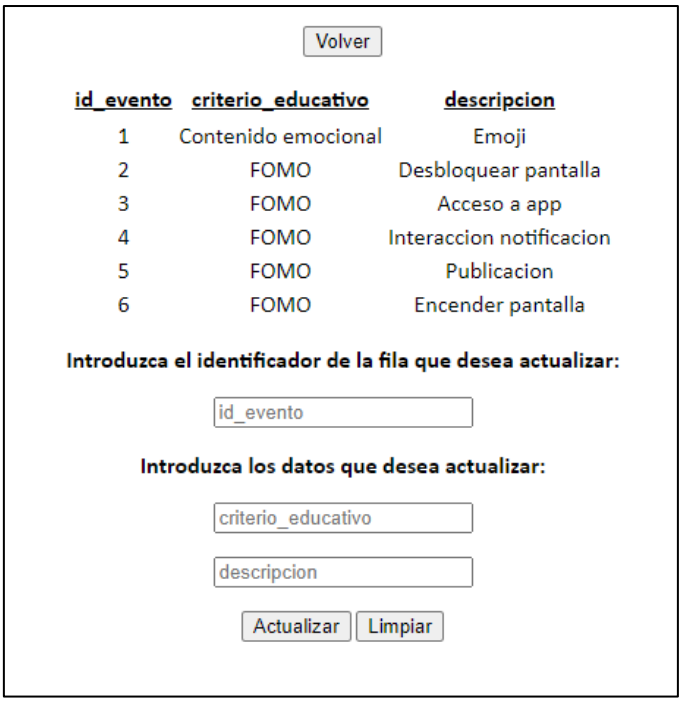


Figura 22. Formulario de actualizar datos en tabla

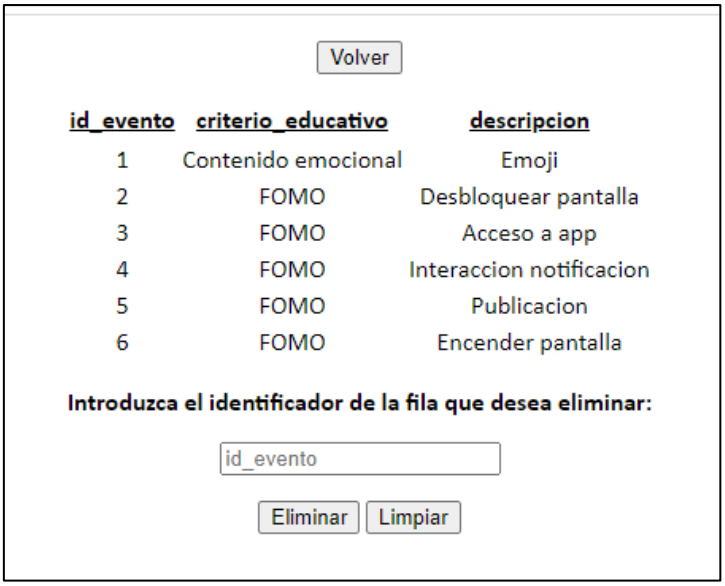


Figura 23. Formulario de borrar datos en tabla

Como soporte para el servicio web se ha creado una página de error que permite visualizar los fallos en la conexión a la base de datos.

- **Implementación servidor web de gestión de datos**

La estructura del servicio web y así como su funcionalidad está desarrollada a través de la biblioteca *Flask* nos permite crear de manera sencilla aplicaciones web con *Python*.

En primer lugar, se crea una instancia de aplicación *Flask* con el nombre app. Una vez creada la instancia podremos lanzar la aplicación en el equipo. Esta instancia app, se

utiliza para gestionar las solicitudes web entrantes y enviar respuestas al usuario. Para crear el punto de entrada a nuestra aplicación desde *WSGI* importaremos la instancia de la aplicación desde nuestro proyecto de *Python* a en un archivo *wgsi.py*. Para que sea sólido a largo plazo se creara un archivo de configuración *uWGSI* y un archivo de unidad *systemd* que permitirá que el sistema *init* de *Debian* inicie automáticamente uWGSI y haga funcionar la aplicación de *Flask* en segundo plano cuando el servidor cargue.

A continuación, se desarrolla el comportamiento del servidor para las URL que componen el servicio web en diferentes diagramas de flujo. En la *Fig. 24* se muestra el comportamiento detallado del servidor web en la sección *Muestra de usuarios*.

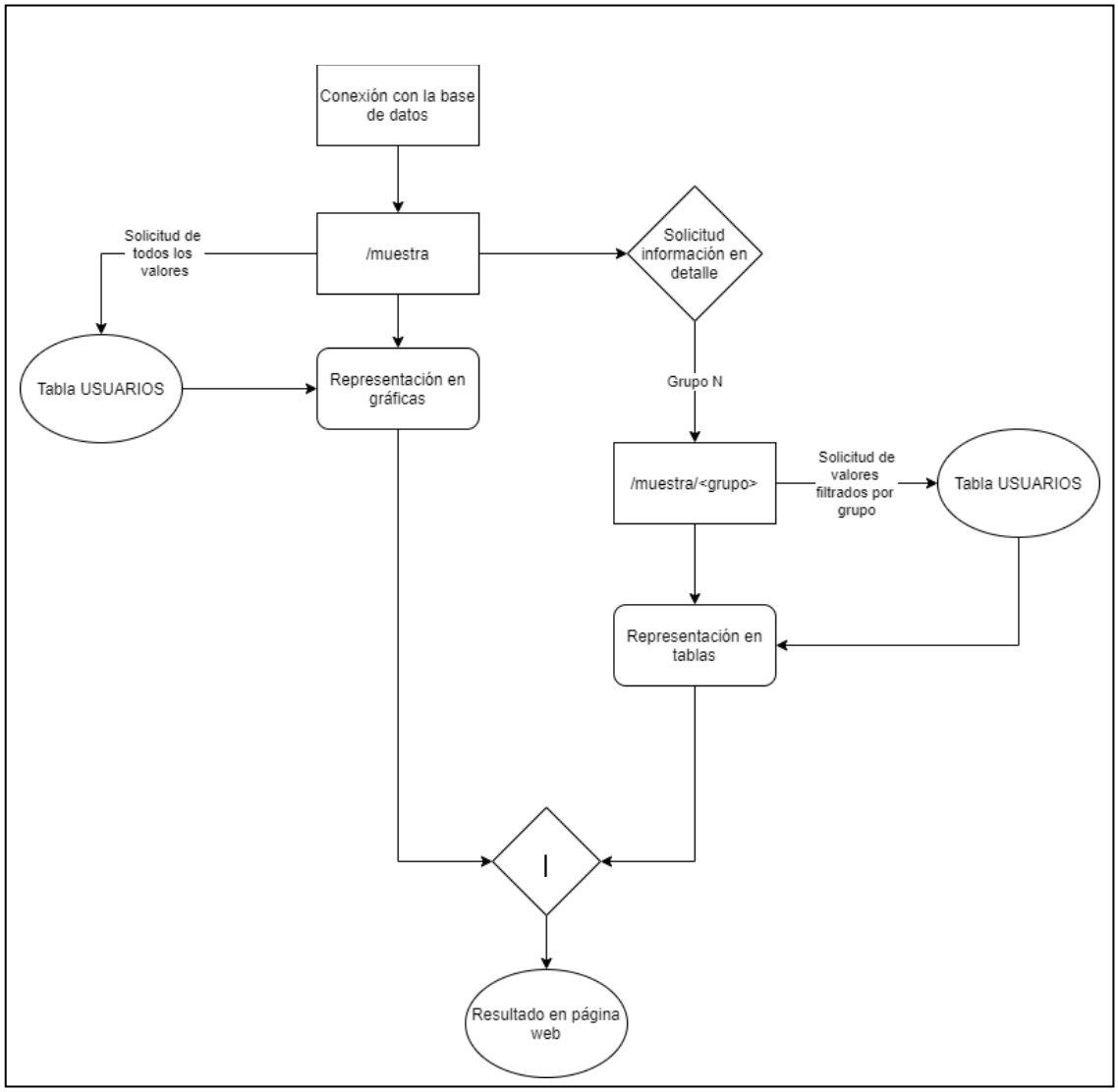


Figura 24. Comportamiento del servidor web para Muestra de usuarios

En la *Fig. 25* se puede observar el comportamiento del servidor para la sección *Búsqueda avanzada*.

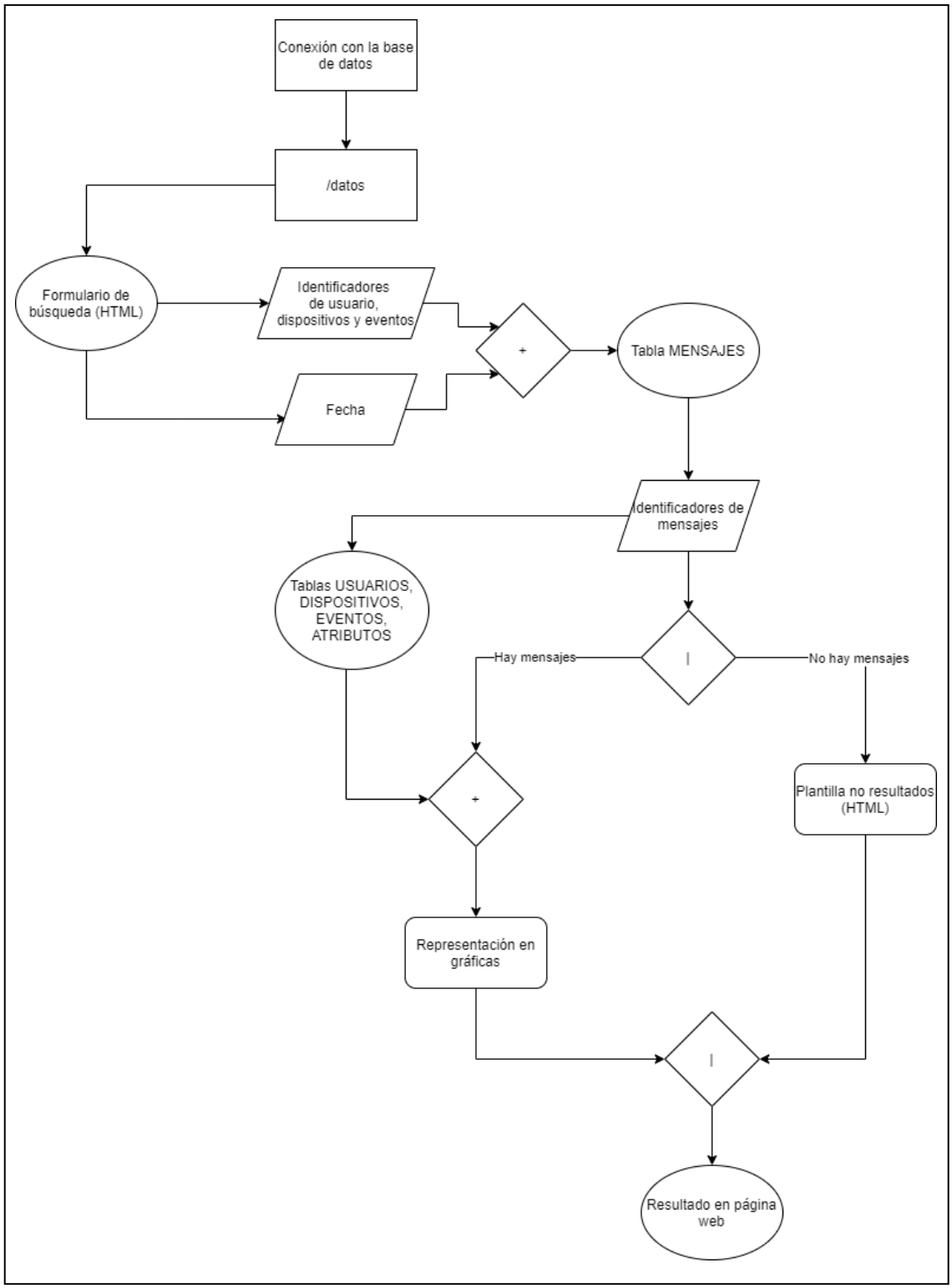


Figura 25. Comportamiento del servidor web para Formulario de búsqueda

En la *Fig. 26* detalladamente el comportamiento del servidor para la sección *Tablas de datos*.

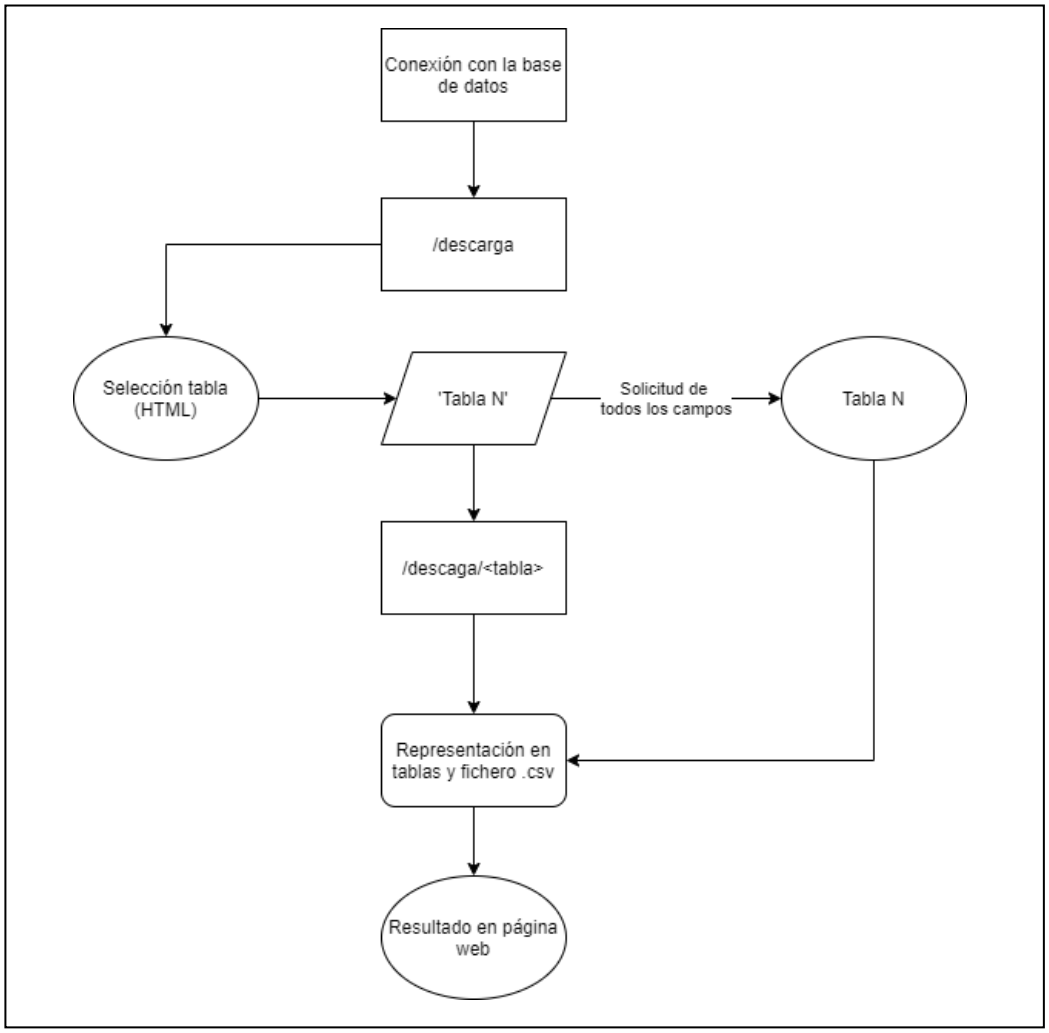


Figura 26. Comportamiento del servidor web para Tablas de datos

En la *Fig. 27* se muestra el comportamiento del servidor web para la sección *Mantenimiento base de datos*.

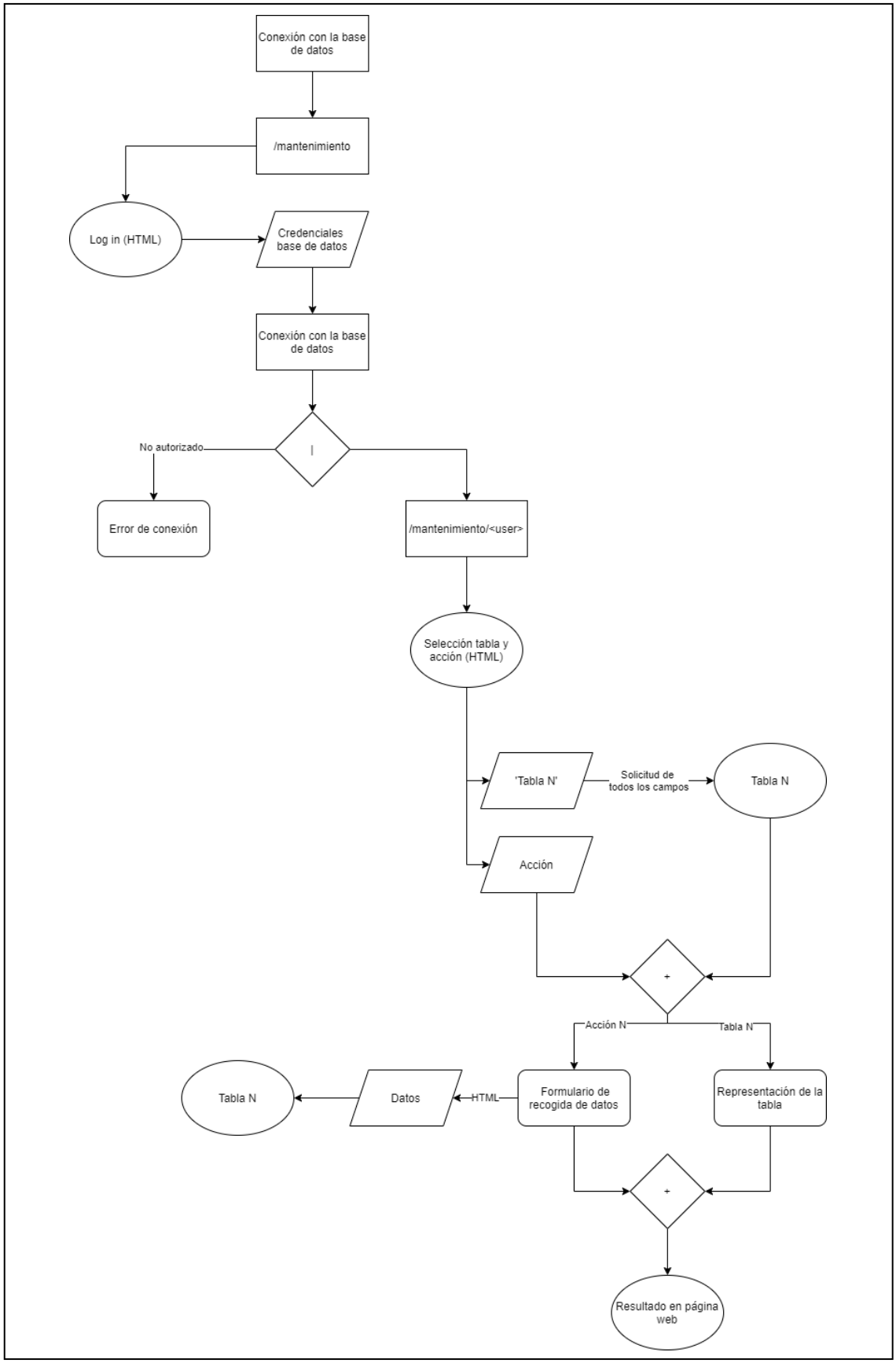


Figura 27. Comportamiento del servidor web para Mantenimiento

En el *Anexo IV* se encuentra una explicación completa y detallada de la implementación que se ha realizado para lograr el comportamiento descrito en los diagramas anteriores para el servidor web de gestión de datos.

3.2.6. Servidor de ficheros

El servidor de ficheros debe ser capaz de almacenar en ficheros creados dinámicamente desde el servidor de recepción, todos los mensajes recibidos de los distintos dispositivos móviles de los usuarios finales. Estos mensajes de *log* se crearán de forma dinámica cada día y por cada usuario que realice conexiones con el servidor. También será necesario un servidor de ficheros que contenga copias temporales de los ficheros *csv* de las tablas de datos que se crean de forma dinámica para que sean "descargables" desde la aplicación web.

La implementación de estos servidores se ha llevado a cabo de manera sencilla creando una carpeta accesible por el servidor de recepción y por el servidor web, respectivamente.

4. RESULTADOS: PRESENTACIÓN DE INFORMES

En este capítulo se van a describir la metodología y el razonamiento con el que se realizan los informes para la presentación de datos. Para ello se han planteado diferentes casos de uso realizables a través de la aplicación web:

- Obtención de los datos de los usuarios registrados en el sistema.
- Búsqueda de resultados sobre la actividad de los usuarios en los dispositivos móviles.
- Extracción de la información almacenada en el sistema mediante ficheros raw.

Estos informes de presentación se han realizado como una primera aproximación para cubrir las necesidades indicadas por los expertos en las reuniones para la realización de sus investigaciones.

En el primer caso de uso, que se corresponde a la sección *Muestra de usuarios,* se han utilizado gráficas de barras que permiten representar un conjunto de datos muy grande de forma clara y sencilla. Así en la *Fig. 28* se puede ver el resultado que muestra la página al inicio.

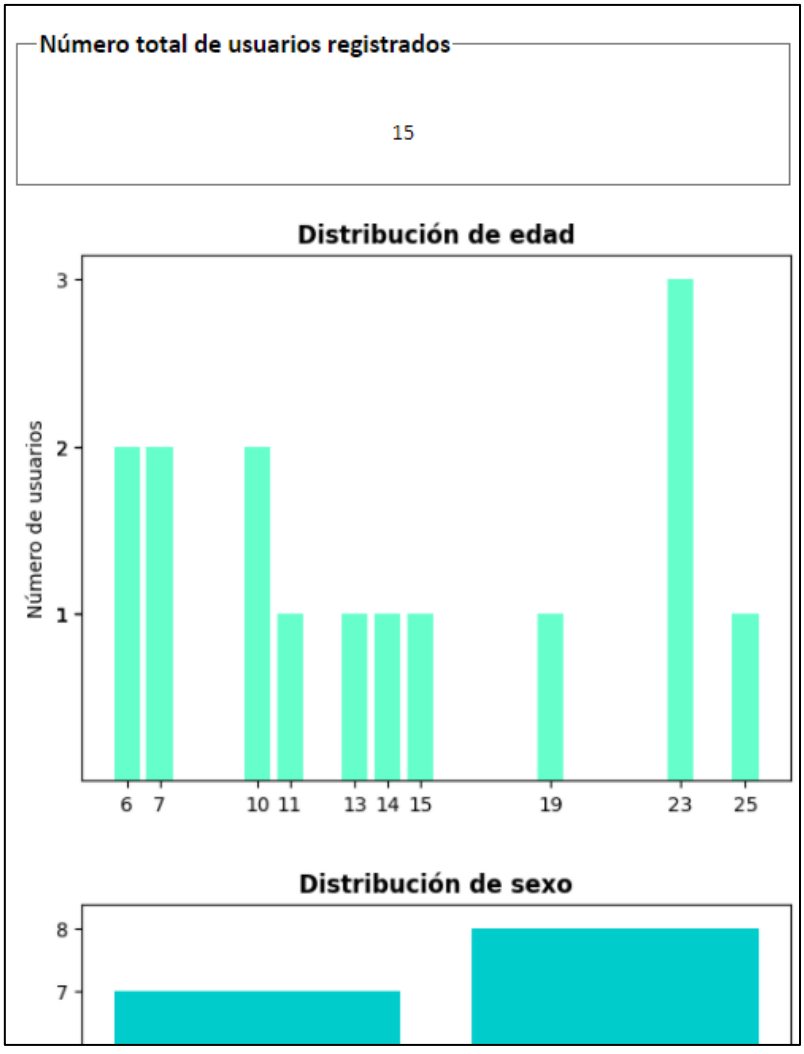


Figura 28. Muestra de usuarios vista principal

De un solo vistazo se pueden hacer rápidamente comparaciones, generalizaciones acerca de los datos y deducir tendencias. En este caso nos resultan útiles dado que hacemos la representación de datos en los que no se pretende evaluar la información, si no tener una visión de conjunto de la muestra de usuarios con la que se está trabajando.

Para aportar más detalle, se ha planteado la elección de un determinado grupo de estudio en el que se muestran los datos en tablas de frecuencia (*Fig. 29*). Ahora si se pretende que la información sea más concreta de forma que se permite conocer exactamente la distribución de los usuarios.

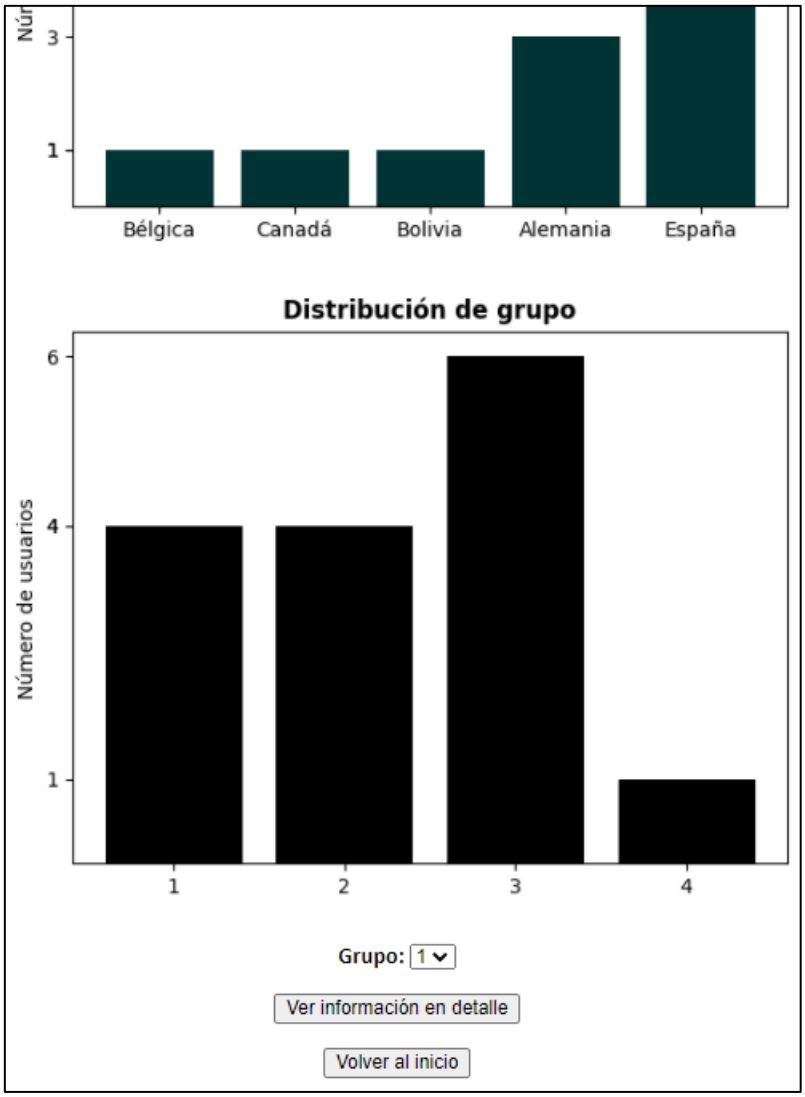


Figura 29. Muestra de usuarios elección de grupo

En la *Fig. 30*, se muestra el resultado obtenido al elegir un grupo de estudio determinado.

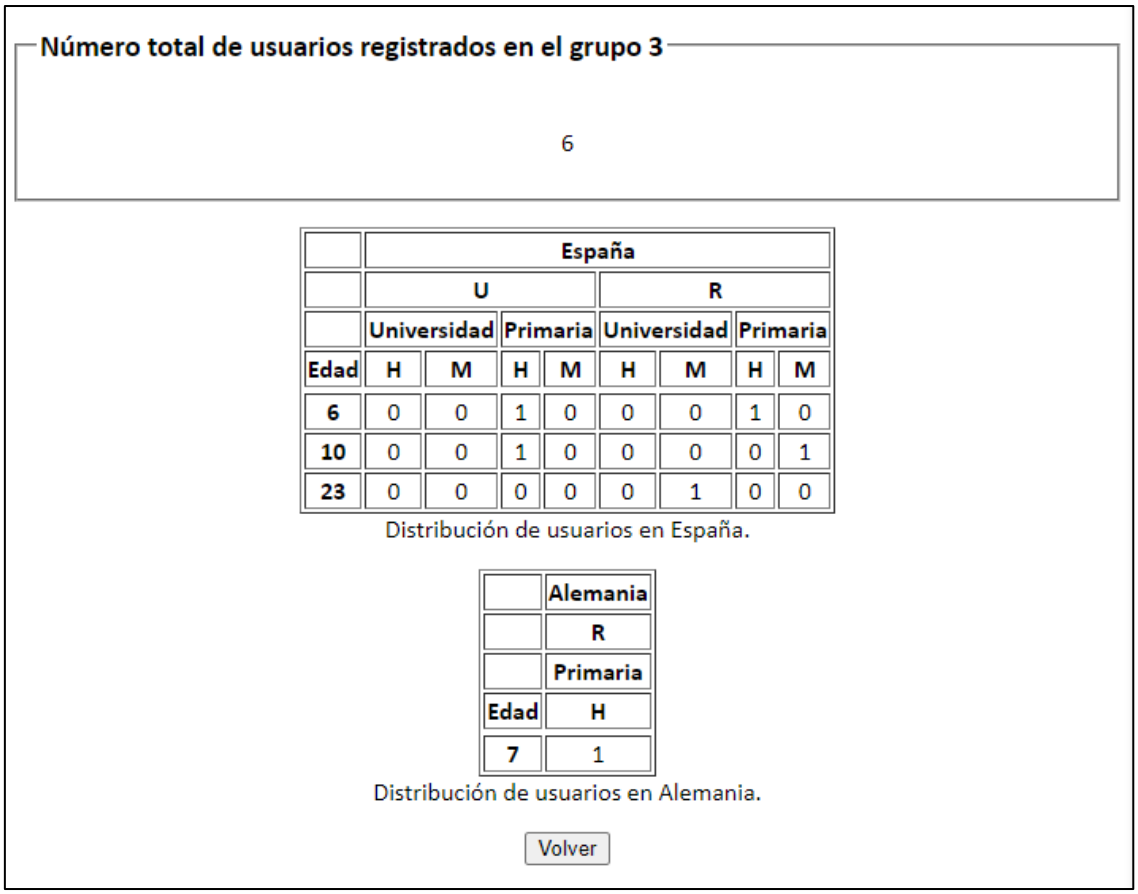


Figura 30. Muestra de usuarios filtrado por grupo

En el segundo caso de uso, tanto en la visión global que permite conocer el comportamiento de un conjunto de usuarios requerido, como en la visión detallada que aporta la información de manera individual para cada uno de los usuarios de dicho conjunto, se han elegido los gráficos circulares como método principal de representación.

Este tipo de gráficos ofrece la ventaja de ser muy "visuales" a la hora de examinar e interpretar la información. Gracias a la proporcionalidad de las porciones y la distribución de los porcentajes se facilita la rápida visualización de comportamientos individuales y en los diferentes grupos de usuarios, facilita la comprensión de sus actividades e intereses, ayuda a comprender las estadísticas obtenidas y a desarrollar nuevas herramientas de uso experto.

Para lograr los objetivos finales de investigación descritos en apartados anteriores, el experto debe ser capaz de examinar la simultaneidad de las interacciones en un mismo o en diferentes dispositivos. Esto se consigue gracias a los filtros, pero también a la gráfica de barras en tres dimensiones que proporciona la duración de las conexiones (eje z) a lo largo del tiempo (eje x) para las distintas aplicaciones (eje y) que se han considerado en el desarrollo del proyecto. Todo esto se puede mostrar tanto de forma global para un conjunto de usuarios como de forma individual para cada uno de ellos.

En la *Fig. 31* se muestra el resultado inicial de la búsqueda avanzada global.

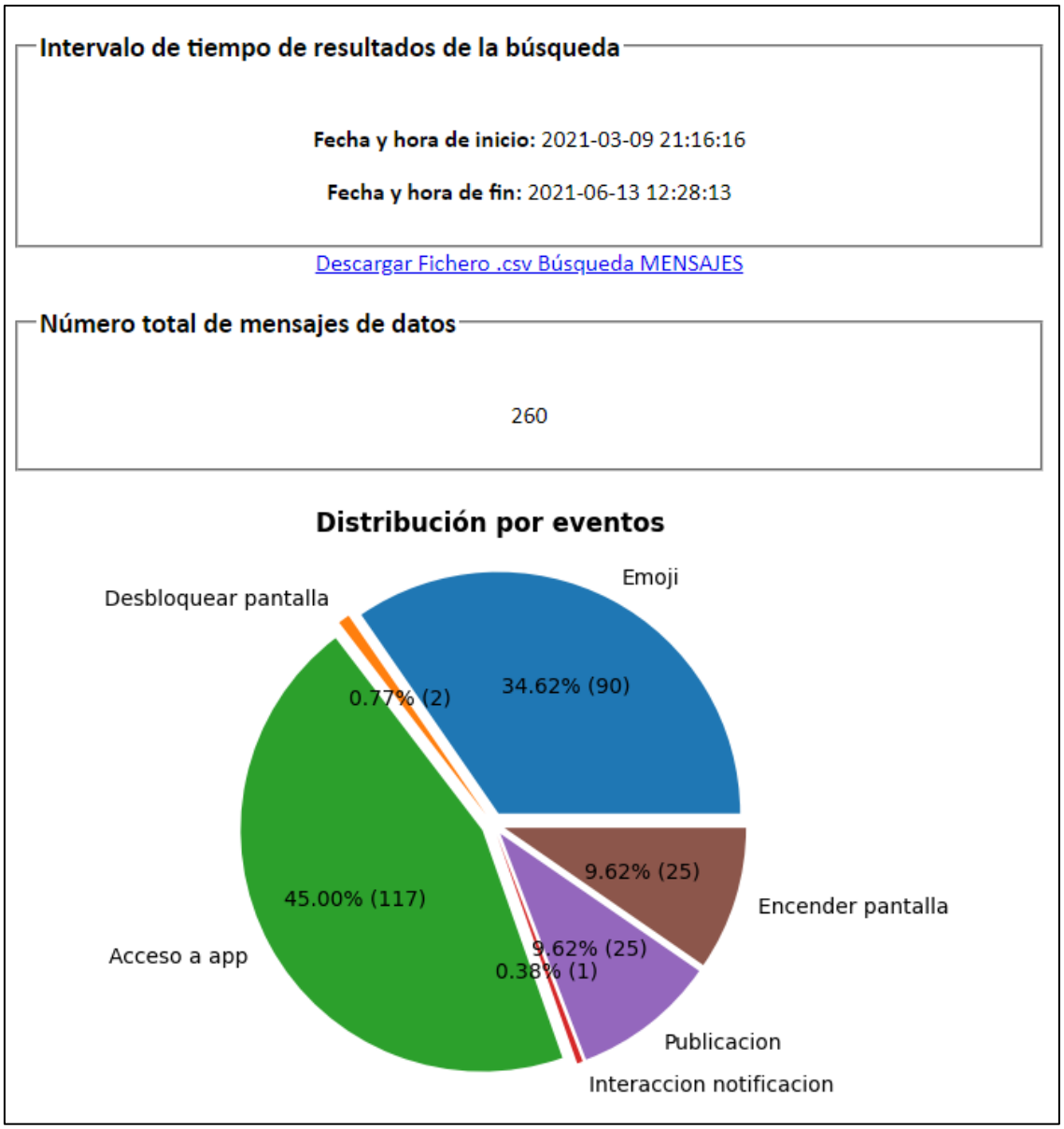


Figura 31. Resultado de búsqueda de datos global por intervalo de tiempo

En la *Fig. 32* se puede ver el gráfico circular y el grafico de barras de la distribución del acceso a aplicaciones.

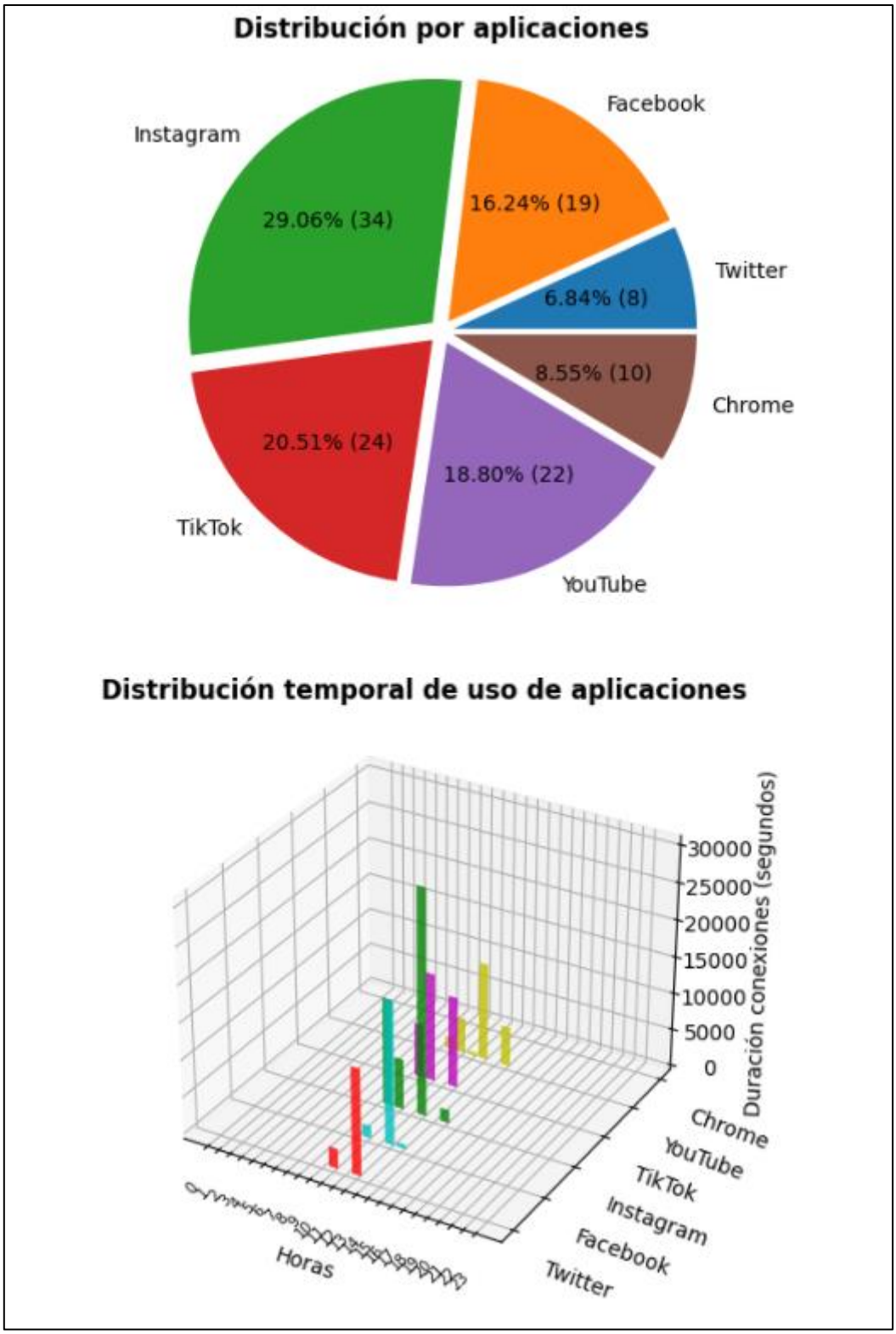


Figura 32. Resultado de búsqueda de datos global por acceso a aplicaciones

En la *Fig. 33* se muestra el grafico circular de la distribución del uso de emoticonos.

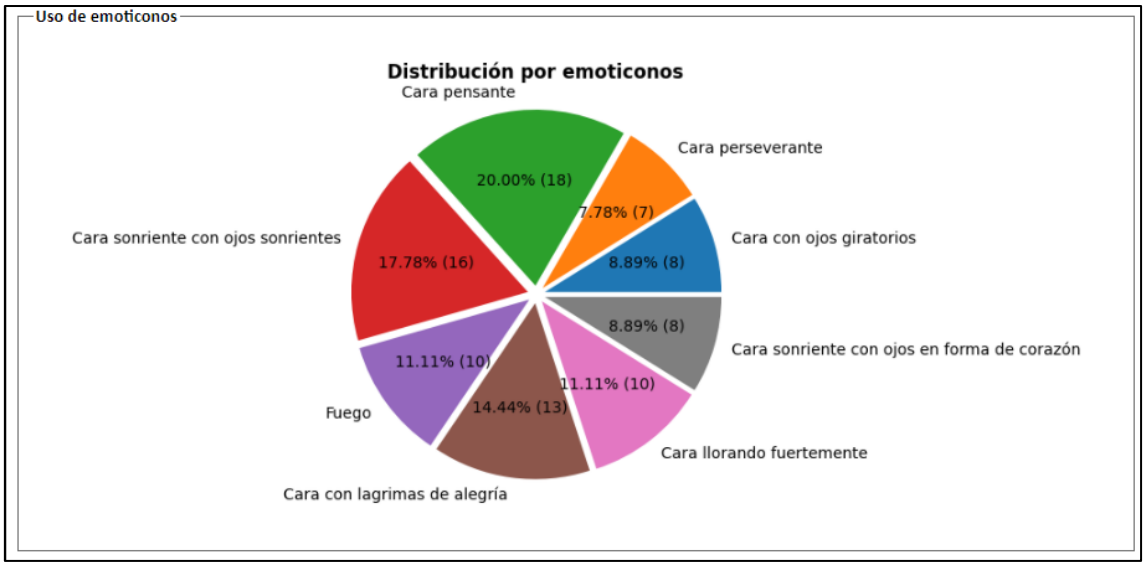


Figura 33. Resultado de búsqueda de datos global uso de emoticonos

En la *Fig. 34* se representan las gráficas circulares que muestran la distribución de los usuarios.

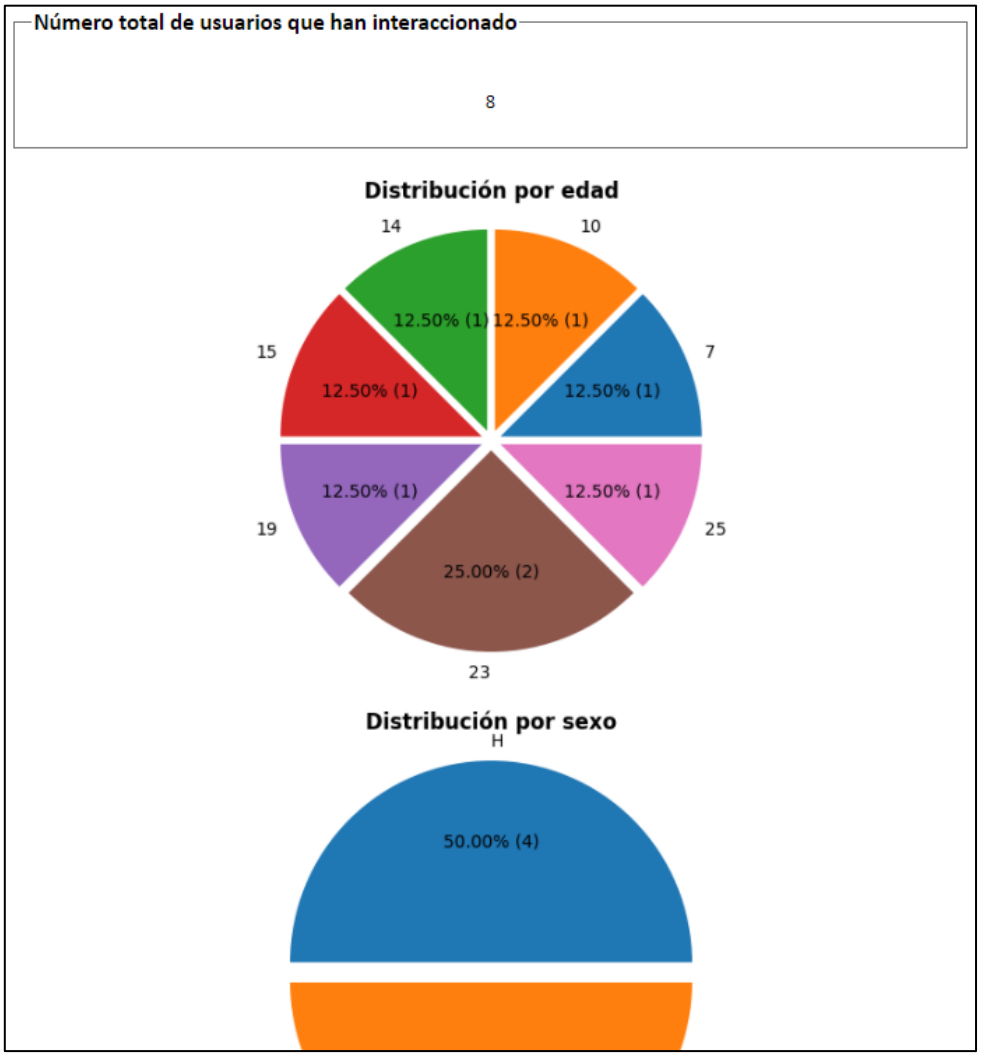


Figura 34. Resultado de búsqueda de datos global por distribución de usuarios

En la *Fig. 35* se muestra el resultado de la búsqueda individual por usuario.

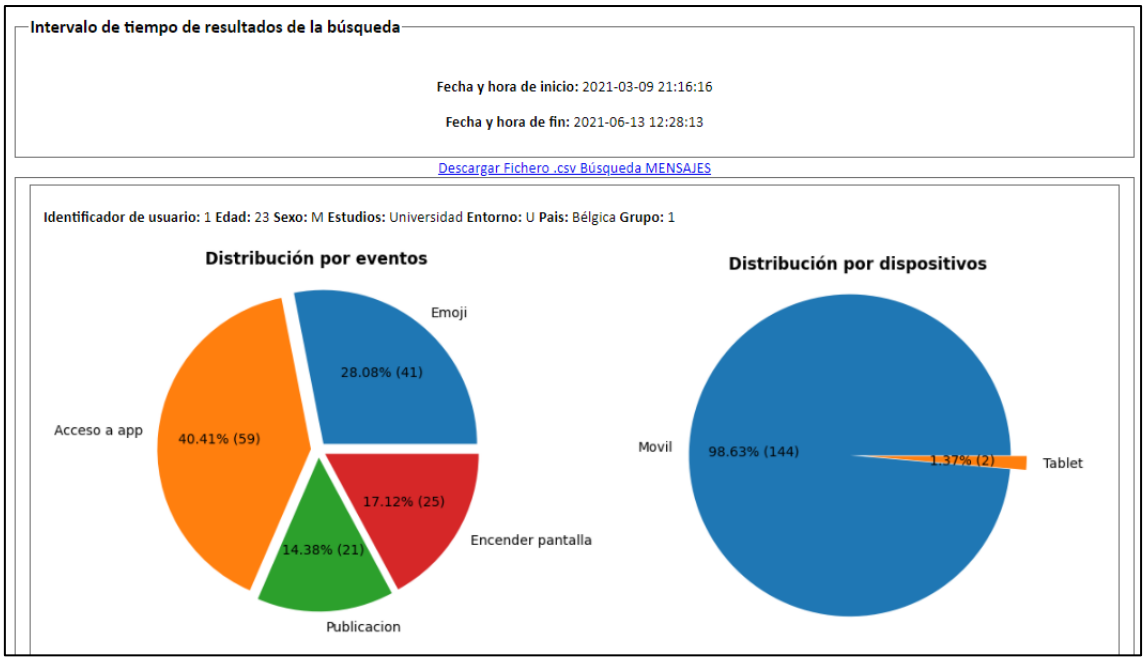
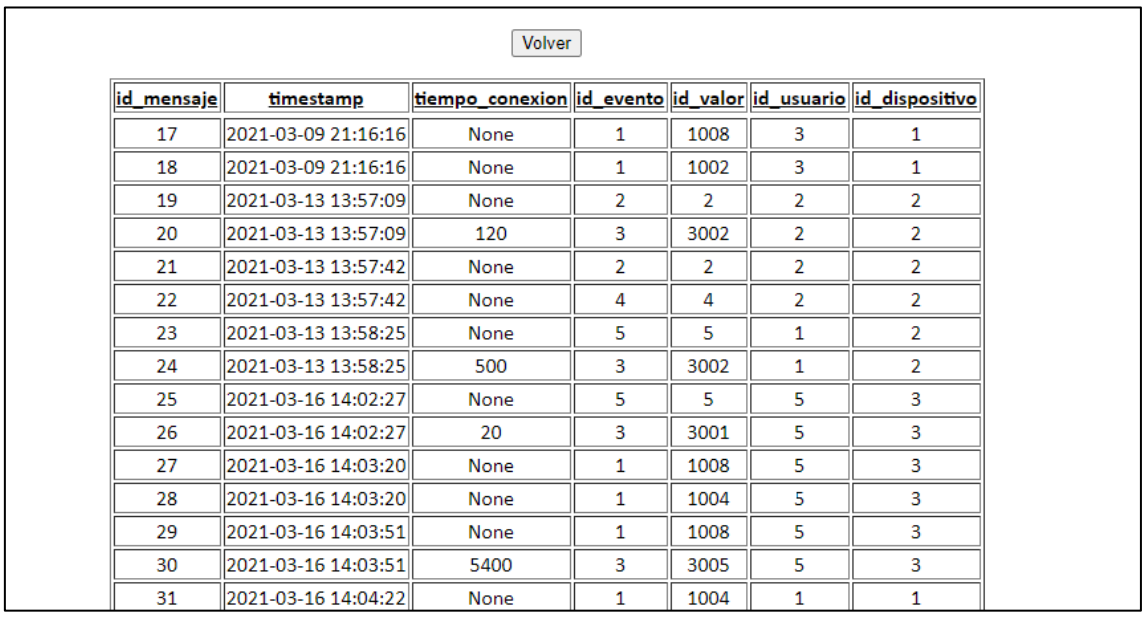


Figura 35. Resultado de búsqueda detallada de datos por usuario

Para el tercer y último caso de uso, la presentación de los datos se hace mediante tablas (*Fig. 36*), que como ya se introdujo en el capítulo *Materiales y métodos,* tendrán su equivalente versión descargable en formato csv para permitir la exportación de la información de la base de datos y el procesado independiente de este sistema. Como también es el caso de la sección de mantenimiento, se busca aportar transparencia ofreciendo los datos en bruto a los usuarios expertos y superusuarios del sistema.



id_mensaje	timestamp	tiempo_conexion	id_evento	id_valor	id_usuario	id_dispositivo
17	2021-03-09 21:16:16	None	1	1008	3	1
18	2021-03-09 21:16:16	None	1	1002	3	1
19	2021-03-13 13:57:09	None	2	2	2	2
20	2021-03-13 13:57:09	120	3	3002	2	2
21	2021-03-13 13:57:42	None	2	2	2	2
22	2021-03-13 13:57:42	None	4	4	2	2
23	2021-03-13 13:58:25	None	5	5	1	2
24	2021-03-13 13:58:25	500	3	3002	1	2
25	2021-03-16 14:02:27	None	5	5	5	3
26	2021-03-16 14:02:27	20	3	3001	5	3
27	2021-03-16 14:03:20	None	1	1008	5	3
28	2021-03-16 14:03:20	None	1	1004	5	3
29	2021-03-16 14:03:51	None	1	1008	5	3
30	2021-03-16 14:03:51	5400	3	3005	5	3
31	2021-03-16 14:04:22	None	1	1004	1	1

Figura 36. Resultado de descarga de tabla de datos

5. CONCLUSIONES Y LÍNEAS FUTURAS

En este trabajo se ha desarrollado una infraestructura para proporcionar a los investigadores del Grupo EducaViva de la Universidad de Zaragoza las herramientas necesarias para poder mejorar el campo de la QoE en relación con la adecuada utilización de las nuevas tecnologías por usuarios finales desde la perspectiva educativa. La infraestructura se basa en un sistema de recepción de mensajes sobre la utilización de los dispositivos móviles. Estos mensajes son almacenados y procesados de acuerdo con los requerimientos obtenidos en las reuniones con el grupo de investigación, y son presentados a través de un servidor web.

5.1. Conclusiones

Los objetivos expuestos en el primer capítulo de este documento han sido alcanzados de forma que:

- La recepción de mensajes de datos de utilización se realiza de forma concurrente y continuada permitiendo la conexión simultanea de multitud de dispositivos móviles de usuario en cualquier momento. Asimismo, la posibilidad de recibir diferentes tipos de mensajes identificados de forma única mediante el protocolo de comunicación establecido con un formado de mensajes determinado aporta la capacidad de realizar una comunicación completa con estos dispositivos de forma totalmente automatizada.
- El almacenamiento y la distribución de todos los datos se fundamenta en una base de datos constituida por diferentes tablas que permiten hacer una primera clasificación y estructuración de toda la información recibida a través del sistema de recepción de mensajes.
- El servicio web proporciona a los expertos una herramienta de acceso a la información registrada en la base de datos en diferentes formatos, siguiendo con los requerimientos obtenidos en las reuniones realizadas, con los que se ha conseguido trasladar dichos requerimientos a un instrumento de trabajo con el que los expertos pueden analizar y gestionar los datos obtenidos de forma clara y eficiente.
- Los diversos prototipos de clientes han permitido probar el correcto funcionamiento de todo el sistema durante el periodo de desarrollo permitiendo la evolución y la mejora de la arquitectura desarrollada mientras se realizaba la implementación.
- El servidor de respuesta ha permitido demostrar la posibilidad de realizar realimentación por parte de los expertos a medida que los estudios y las investigaciones evolucionan.

5.2. Líneas futuras

Como líneas futuras de trabajo se pueden mencionar principalmente:

- Evolucionar los servidores de recepción y el proceso de almacenamiento en la base de datos consiguiendo abarcar un mayor número de formatos de mensajes.
- Añadir funcionalidad al servicio web para la gestión de datos permitiendo una mayor parametrización en las búsquedas de información y en la representación de los resultados.
- Desarrollar en su totalidad el sistema de respuesta a los usuarios creando una herramienta que permita a los expertos realizar realimentación tanto de manera automática como personalizada a los distintos usuarios en función de los resultados obtenidos en las investigaciones.

Finalmente, es importante destacar que este trabajo ha sido realizado como un primer prototipo o aproximación de la infraestructura requerida, que en el caso de implantarse en un entorno real deberá ser evaluada y optimizada en su totalidad.

BIBLIOGRAFÍA

[1] Unión Internacional de Telecomunicaciones. Definition of Quality of Experience. (última visita 11/06/2021) https://www.itu.int/md/T05-FG.IPTV-IL-0050/es

[2] Debian, the universal operating system. (última visita 11/06/2021) https://www.debian.org/index.es.html

[3] MariaDB Foundation. (última visita 11/06/2021) https://mariadb.org/

[4] DBeaver community, Free universal database tool. (última visita 11/06/2021) https://dbeaver.io/

[5] Nginx, part of F5. (última visita 11/06/2021) https://www.nginx.com/

[6] uWGSI, the uWGSI Project. (Última visita 11/06/2021) https://uwsgi-docs.readthedocs.io/en/latest/

[7] Certbot. EFF. (última visita 11/06/2021) https://certbot.eff.org/

[8] Python. (última visita 11/06/2021) https://www.python.org/

[9] Pycharm. (última visita 11/06/2021) https://www.jetbrains.com/es-es/pycharm/

[10] Flask, web development, one drop at a time. (última visita 11/06/2021) https://flask.palletsprojects.com/en/2.0.x/

[11] Matplotlib, visualitation with Python. (última visita 11/06/2021) https://matplotlib.org/

Anexo I. IMPLEMENTACIÓN DEL SERVIDOR DE RECEPCIÓN

En este anexo se explica detalladamente la implementación del servidor de recepción mediante el programa *ServerSocket.py*.

Para recibir conexiones de manera continua se implementa un bucle infinito que acepta las conexiones de los clientes e inicia un nuevo hilo para la comunicación con cada uno de ellos. Cada hilo gestionará la comunicación TCP con el cliente de forma independiente.

Una vez establecida la comunicación de manera satisfactoria se necesita un bucle encargado de recibir los datos del cliente con un timeout para finalizar la espera de paquetes. Estos datos recibidos en paquetes de bytes se van concatenando en una misma variable de respuesta en formato utf-8. Una vez se tiene el mensaje completo se registra la hora de la recepción para poder crear un registro de logs de las comunicaciones realizadas con los distintos clientes.

El siguiente paso es establecer la conexión con la base de datos que nos permite registrar y clasificar los datos de los mensajes recibidos y obtener la información necesaria para enviar la respuesta a los clientes.

Cuando estamos conectados a la base de datos de manera correcta podemos empezar el procesado de los paquetes recibidos. En primer lugar, se separa el mensaje en las diferentes líneas que lo componen. Se ha establecido en las reuniones que la primera línea recibida designe el tipo de mensaje del que se trata. Como ya se comentó anteriormente hay cuatro tipos de mensajes posibles, a cada uno se ellos se le ha asignado un identificador numérico para una sencilla clasificación del contenido recibido.

Una vez se ha clasificado el mensaje recibido se procede a trabajar con cada una de las líneas siguientes de manera individual recorriéndolas con un bucle. En cada una de estas líneas, en caso de contener más de un dato de interés, se ha convenido que se utilizará el separador ‘;’ para dividir la información. De esta manera será común a todos los tipos de mensajes hacer una primera separación de los datos de cada una de las líneas de manera consecutiva. En los siguientes puntos se explica cómo actúa el servidor con cada uno de los mensajes recibidos:

- La información del mensaje de registro de nuevo usuario servirá para que el servidor se conecte a la base datos y obtenga el identificador correspondiente a la descripción de dispositivo de la tabla DISPOSITIVOS. A continuación, el resto de los datos del mensaje se guardan en un vector y se insertan en la tabla de USUARIOS. Si se han podido introducir los datos de forma correcta se pide a la base de datos el identificador de usuario asignado en el registro. En caso de que el alias ya se encuentre en uso se devolverá un error al cliente, de lo contrario, se enviara un mensaje de vuelta al usuario con el identificador de usuario y el identificador de dispositivo separados con ‘;’.

- La información del mensaje de datos de utilización del dispositivo se inserta en la tabla MENSAJES de la base de datos. Una vez que se ha hecho el registro del mensaje se solicita a la base de datos el tiempo de actualización que debe usar dicho cliente para realizar el envío de datos de forma periódica desde la tabla USUARIOS correspondiente al usuario que ha enviado la información. Este tiempo se envía de vuelta a dicho usuario como respuesta en la comunicación.

- Con la información extraída del mensaje de inicio de sesión, el servidor solicita el identificador de usuario correspondiente al alias recibido para poder contrastar los datos de acceso. En caso de que el identificador de usuario introducido por el usuario en el inicio de sesión y el obtenido a través de la base de datos mediante el alias coincidan, el inicio de sesión se ha realizado de manera correcta. Si esto ocurre, se envía al cliente un mensaje de vuelta con el identificador de usuario y el identificador de dispositivo obtenido a partir de la descripción del tipo de dispositivo también incluida en el mensaje. En caso contrario, si ambos identificadores de usuario no coinciden, se devuelve un mensaje de error.

- El mensaje de actualización de datos personales contendrá todos los huecos de los datos que se pueden actualizar, pero solo habrá información en aquellos que realmente se vayan a cambiar. De esta manera después de hacer la separación de los datos de la línea se comprobará cuáles de ellos tienen un valor no nulo y se realizara la actualización de dicho valor en la tabla de USUARIOS. Una vez que la actualización de los datos se ha realizado de manera exitosa se envía un mensaje de 'OK' al cliente.

Por último, cuando se envía la respuesta al usuario de manera que la comunicación ha finalizado de manera satisfactoria, se guarda en un archivo de log nombrado con la fecha actual y el numero identificador de usuario el mensaje que se ha recibido en el inicio de la conexión.

Para aportar robustez al servidor de recepción se cuenta con un comando de captura ante posibles errores en la comunicación con la base de datos.

Anexo II. CAPTURAS DE INTERCAMBIO DE MENSAJES ENTRE UN CLIENTE Y EL SERVIDOR DE RECEPCIÓN

En la *Fig. 37* se ve el mensaje de datos sobre la utilización de un dispositivo recibido por el servidor de recepción, y en la *Fig. 38* el mensaje de respuesta del servidor al cliente tras el fin de la recepción de datos.

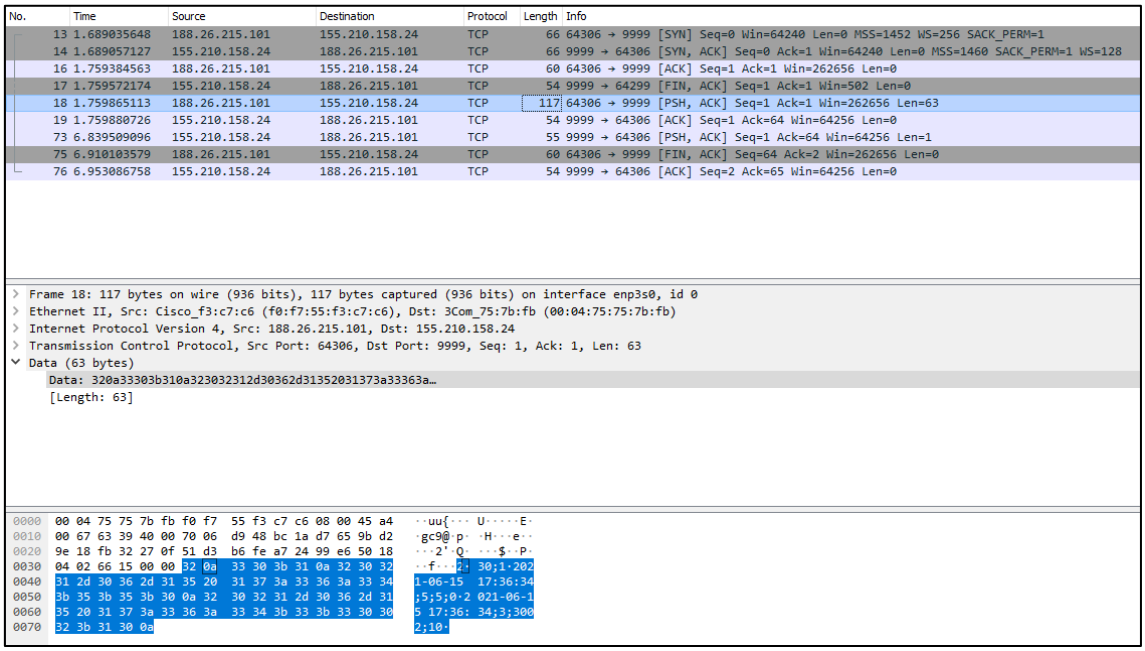


Figura 37. Captura mensaje de datos (ID = 2)

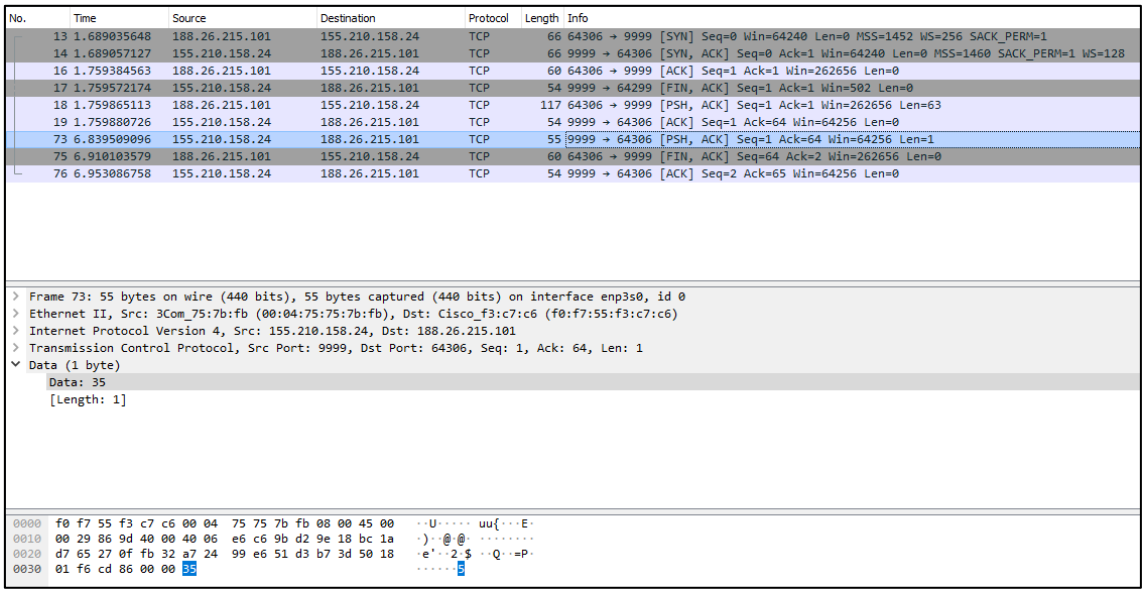


Figura 38. Captura mensaje de respuesta al mensaje de datos

En la *Fig. 39* se ve el mensaje enviado al servidor de recepción para realizar el inicio de sesión, y en la *Fig. 40* el mensaje de respuesta del servidor al cliente tras el inicio de sesión exitoso.

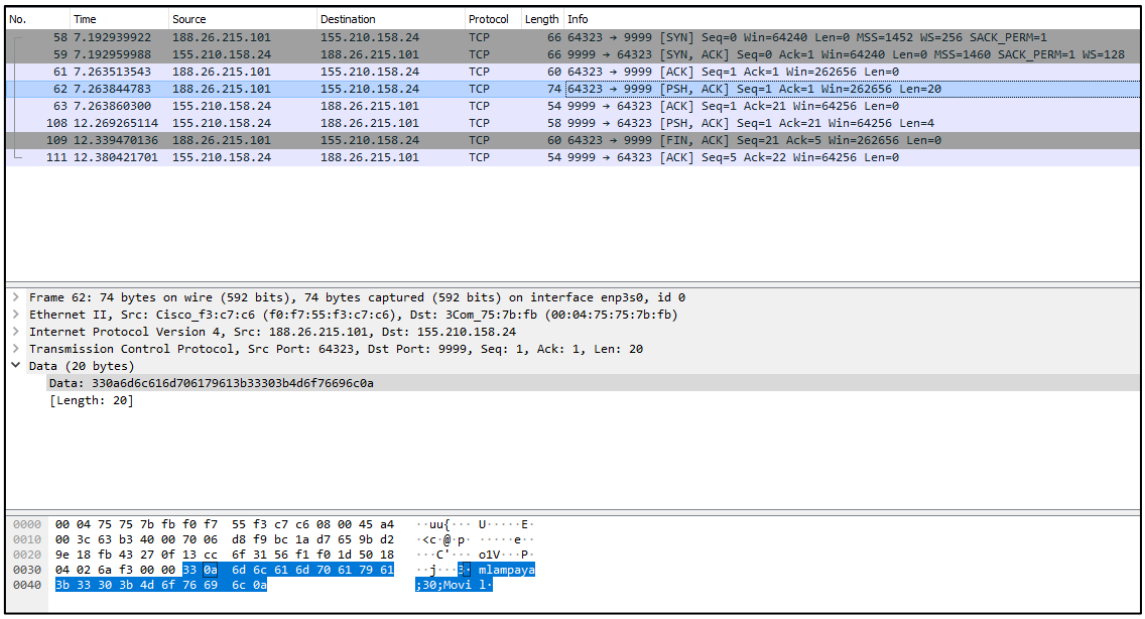


Figura 39. Captura mensaje de inicio de sesión (ID = 3)

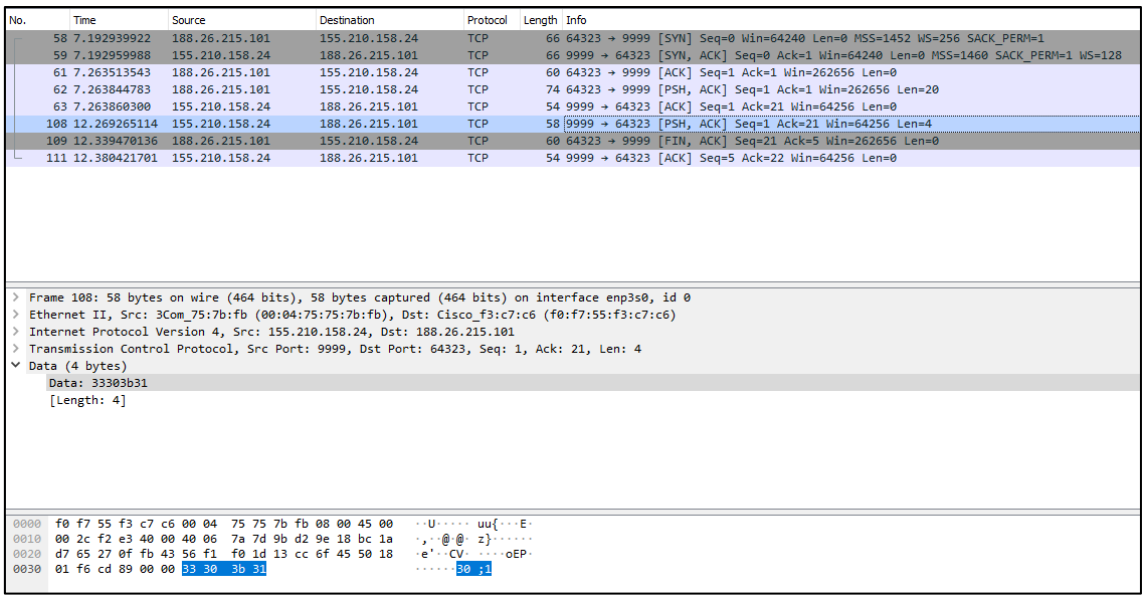


Figura 40. Captura mensaje de respuesta al inicio de sesión

En la *Fig. 41* se ve el mensaje recibido en servidor para realizar la actualización de datos personales del usuario, y en la *Fig. 42* el mensaje de respuesta del servidor al cliente tras la actualización exitosa.

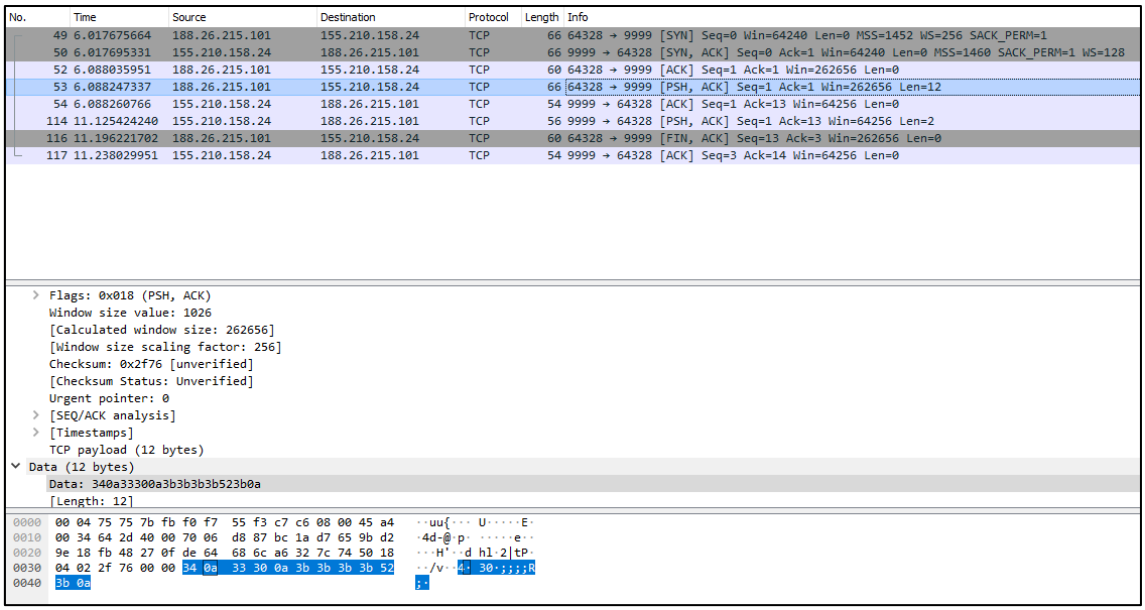


Figura 41. Captura mensaje de actualización de datos personales (ID = 4)

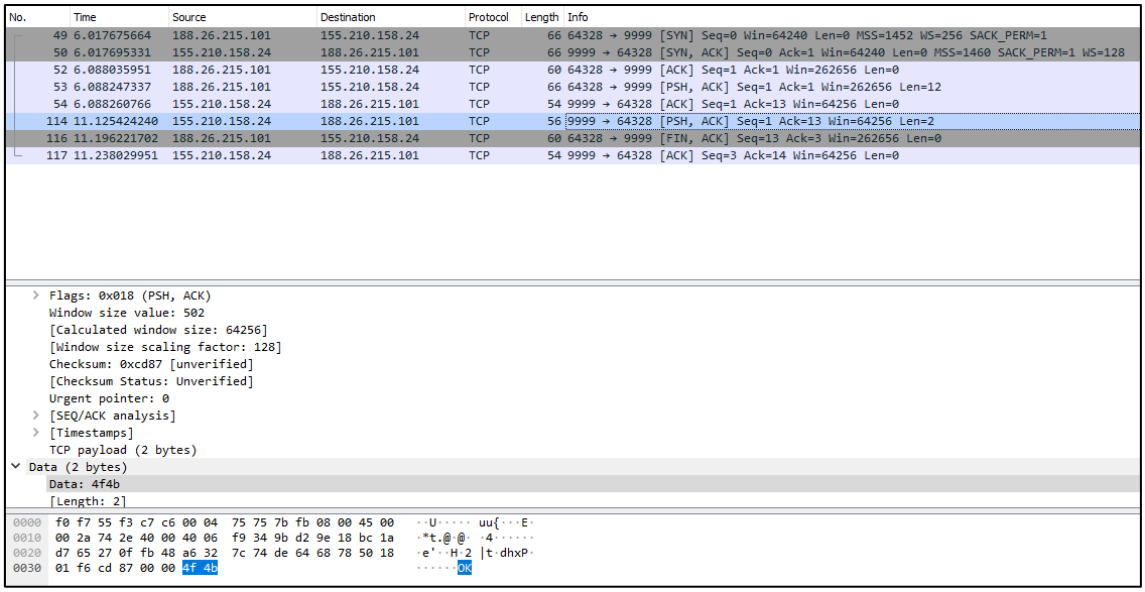


Figura 42. Captura mensaje de respuesta a la actualización de datos personales

Anexo III. IMPLEMENTACIÓN DEL SERVIDOR DE RESPUESTA

En este anexo se explica detalladamente la implementación del servidor de respuesta mediante el programa *ServerIni.py*.

Una vez establecida la comunicación con el cliente se implementa un bucle con un timeout, encargado de finalizar la espera de paquetes, que recibirá el mensaje del usuario como paquetes de bytes de datos. Estos paquetes se concatenan en la variable respuesta en formato utf-8. En este punto, para poder hacer el registro de logs de las comunicaciones de los distintos clientes, se guarda la hora de la recepción del mensaje.

A continuación, nos conectamos a la base de datos lo que nos permitirá obtener cualquier tipo de información que se encuentre registrada en la base de datos sobre el usuario que ha realizado la conexión. Para identificar a dicho usuario se extrae el identificador de usuario del mensaje recibido. En este caso como el mensaje solo contendrá este dato, será fácil extraerlo separando el mensaje por líneas y tomando el valor de la primera línea.

A modo de prueba, una vez que disponemos de dicho identificador obtendremos el alias correspondiente en la tabla USUARIOS de la base de datos. También generaremos de manera aleatoria una carita entre las siguientes posibles:

Caritas para el mensaje				
:D	:)	:|	:|	:O

De esta forma creamos un mensaje del tipo:

HOLA, 'alias'\n *:X\n*

Este mensaje será enviado como respuesta al usuario en la comunicación establecida. Una vez se ha completado el envío de manera exitosa, se guarda el mensaje recibido en el inicio de la conexión en un fichero de log nombrado con la fecha de la recepción y el identificador de usuario.

Finalmente, se cuenta con un comando de captura ante posibles errores en la comunicación con la base de datos.

Anexo IV. IMPLEMENTACIÓN DEL SERVIDOR WEB DE GESTIÓN DE DATOS

En este anexo se explica detalladamente la implementación del servidor de web mediante el programa *AppWebTFG.py*.

Será necesario realizar la conexión con la base de datos en cada sección para obtener tanto los identificadores numéricos con los que se calcular como se distribuyen los datos, como los descriptores que permitirán etiquetar las representaciones realizadas de una manera clara.

- Para componer la sección *Visualizar muestra de usuarios* (/muestra), se solicita a la base de datos todos los valores de la tabla USUARIOS para su representación en gráficas. Para ver la información en detalle, se añade un botón en la plantilla HMTL que permite escoger el grupo de estudio que se quiere visualizar y con esto se muestra una página (/muestra/<grupo>) con la información clasificada en tablas.

- En la sección *Acceder al formulario de búsqueda* (/datos) se extraerán todos los filtros seleccionados en los campos del formulario HTML (este formulario se crea de forma dinámica en función de la información de las tablas de la base de datos). Con los valores de estos filtros se obtienen los identificadores de usuario, dispositivo y evento que cumplen con los requisitos de la búsqueda relacionada. Una vez se tienen estos identificadores y el día del que se quiere obtener los datos se puede seleccionar los identificadores de mensaje de datos correspondientes de la tabla MENSAJES. En caso de que no haya resultados se mostrara una plantilla HTML por defecto, si no, se procederá a la representación detallada de toda la información que contienen estos mensajes tanto del tipo de interacción con el dispositivo como del usuario que ha realizado dicha interacción. Para realizar la representación es necesario volver a extraer los identificadores de usuario, dispositivo y evento a partir de los identificadores de mensaje obtenidos y, con estos identificadores conseguir sus descripciones.

- Para realizar la sección *Descargar tablas de datos* (/descarga), en primer lugar, se deberá seleccionar la tabla a mostrar en la plantilla HTML. Una vez extraído este dato se obtiene la tabla completa de la base de datos y se redirecciona a la página (/descarga/<tabla>) que la muestra y contiene el enlace para descargarla en formato csv.

- En la sección *Acceder al mantenimiento* (/mantenimiento) se extraen las credenciales de conexión introducidas en la plantilla HTML. Con estas credenciales se hace la conexión a la base de datos. En caso de no estar autorizado se muestra la página de error por defecto. Si la conexión se hace de

forma satisfactoria, se redirecciona a la página (/mantenimiento/<user>) que permite seleccionar tanto la tabla como la acción a realizar a través del formulario HMTL presentado. De esta plantilla se extrae la tabla y la acción lo que se usa para redirigir a la página (/mantenimiento/<user>/<TABLA_accion>) que ofrece el formulario adaptado a cada tipo de petición realizada. Una vez el usuario introduce los datos en el formulario se recogen los valores y se ejecuta la petición en la correspondiente tabla de la base de datos. Con cada acción realizada se recarga automáticamente la página para mostrar a tiempo real los cambios producidos. De nuevo, en caso de ocurrir un error en la interacción con la base de datos se mostrará un mensaje de error.