UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A WEB TOOL TO ANALYSE INSTANT MESSAGING CHATS

JAIME CONDE SEGOVIA

ENERO 2023

TRABAJO DE FIN DE GRADO

Título:	DISEÑO Y DESARROLLO DE UNA HERRAMIENTA WEB PARA ANÁLISIS DE CHATS DE MENSAJERÍA INSTANTÁNEA
Título (inglés):	DESIGN AND DEVELOPMENT OF A WEB TOOL TO ANALYSE INSTANT MESSAGING CHATS
Autor:	JAIME CONDE SEGOVIA
Tutor:	JUAN FERNANDO SÁNCHEZ RADA
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A WEB TOOL TO ANALYSE INSTANT MESSAGING CHATS

JAIME CONDE SEGOVIA

ENERO 2023

Resumen

Con 14 años compré mi primer teléfono inteligente. Con ello, gran parte de las interacciones con las personas de mi entorno más cercano, tenían lugar a través de aplicaciones de mensajería instantánea: principalmente WhatsApp. Por entonces no era consciente de la información que se perdía por estas vías, pero con el tiempo me he dado cuenta de la importancia de la información no verbal que se manifiesta en gestos, emociones, entonación y; finalmente, con la acumulación de todos estos factores, puede llegar a deteriorar una relación si no se reacciona correctamente.

Han sido numerosas las ocasiones en las que he sentido la necesidad de comprobar si mi relación con alguna persona cercana estaba decayendo, o se trataba únicamente de un sentimiento sin fundamentos. Es por ello que, en 2019, comencé el desarrollo de una herramienta de código libre para el análisis de conversaciones exportadas por aplicaciones de mensajería instantánea. Al tratar información tan personal, la arquitectura estaba clara: todo el procesamiento debe hacerse en el cliente y evitar enviar información a servidores salvo la autorización del usuario final.

Palabras clave: mensajería instantánea, aplicación web, javascript, código libre, WhatsApp, privacidad

Abstract

Keywords:

Agradecimientos

Me gustaría expresar un especial agradecimiento a mi tutor, Juan Fernando Sánchez Rada, que en las buenas y en las malas, ha prestado ayuda en todo lo posible, ofreciendo apoyo y soluciones cuando ha sido necesario; en el proyecto y en lo personal.

Agradezco a mis compañeros de la universidad, familia y amigos por acompañarme durante esta etapa de mi vida y arrojar luz allá donde hubo sombra.

A mi compañero y amigo Carlos García-Mauriño Dueñas por alojarme una instancia de ShareLatex en su servidor, por sus consejos, así como por el incondicional apoyo que he recibido por su parte durante todo el proyecto.

A mi compañero y amigo Guillermo García Grao, por apoyar y contribuir en los primeros pasos del desarrollo del código de este proyecto.

Por último, quiero expresar mi agradecimiento al grupo *GSI* en la ETSIT-UPM, que ha hecho posible realizar este trabajo sobre uno de mis primeros proyectos personales.

"In God we trust. All others must bring data." - W. Edwards Deming

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

1.1. Contexto y motivación

Con 14 años compré mi primer teléfono inteligente. Con ello, gran parte de las interacciones con las personas de mi entorno más cercano, tenían lugar a través de aplicaciones de mensajería instantánea: principalmente WhatsApp. Por entonces no era consciente de la información que se perdía por estas vías, pero con el tiempo me he dado cuenta de la importancia de la información no verbal que se manifiesta en gestos, emociones, entonación y; finalmente, con la acumulación de todos estos factores, puede llegar a deteriorar una relación si no se reacciona correctamente.

Han sido numerosas las ocasiones en las que he sentido la necesidad de comprobar si mi relación con alguna persona cercana estaba decayendo, o se trataba únicamente de un sentimiento sin fundamentos. Es por ello que, en 2019, comencé el desarrollo de una herramienta de código libre para el análisis de conversaciones exportadas por aplicaciones de mensajería instantánea. Al tratar información tan personal, la arquitectura estaba clara: todo el procesamiento debe hacerse en el cliente y evitar enviar información a servidores salvo la autorización del usuario final.

1.2. Objetivos

Con este trabajo se pretende diseñar una aplicación web que analice datos de conversaciones de WhatsApp y ofrezca al usuario una visualización de datos relevantes de la misma, tanto generales como de la evolución en el tiempo.

Podemos segregar los objetivos en los siguientes:

- Creación de una aplicación web.
- Compatibilidad con Progressive Web App (PWA).
- Compatibilidad con datos de conversaciones grupales e individuales.
- Compatibilidad con datos de conversaciones con y sin contenido multimedia.
- Compatibilidad con todos los formatos de exportación de la aplicación WhatsApp para distintos sistemas operativos.
- Cálculo de estadísticas generales de los datos.
- Visualizaciones gráficas para los distintos estadísticos.
- Posibilidad de interacción con los gráficos.
- Posibilidad de exportar los resultados.
- Implementación de tests para estabilidad y comprobación del código fuente.
- Documentación del proyecto para desarrolladores.

1.3. Estructura del documento

En esta sección describiremos la estructura de este documento, así como una breve descripción de los capítulos. La estructura es la siguiente:

Capítulo 1. Introducción.

Se introduce la motivación que ha llevado al desarrollo del proyecto, así como los objetivos que busca alcanzar y resolver. Todo esto son ingredientes para la adecuada resolución de un problema. Asimismo, se busca explicar la estructura del documento.

Capítulo 2. Tecnologías habilitantes.

En este capítulo se explicarán las tecnologías utilizadas que permiten la realización del proyecto, fundamentando la elección de las mismas de manera segregada en cliente, servidor y desarrollo.

Capítulo 3. Análisis de requisitos.

Se detallan el análisis de requisitos, casos de uso.

Capítulo 4. Arquitectura.

Se introducirá la arquitectura general, entrando en detalle en decisiones de diseño, módulos y unidades lógicas en las que se divide, así como la implementación de las mismas.

Capítulo 5. Caso de estudio.

Se ha escogido un caso de uso para analizar en detalle. Se explica el funcionamiento completo desde el punto de vista del usuario.

Capítulo 6. Conclusiones y futuras líneas de trabajo.

En este capítulo se extraen las conclusiones del proyecto, así como posibles formas de continuar el desarrollo del proyecto tras esta memoria.

CAPÍTULO 2

Tecnologías habilitantes

2.1. Tecnologías en el cliente

2.1.1. React

Para el desarrollo del cliente web se ha elegido el framework React. React es una librería para el desarrollo de front-end en JavaScript. Es de código libre y permite construir interfaces de usuario mediante la definición de componentes reutilizables. Es impulsado y mantenido por Meta, así como una gran comunidad de individuos y compañías.

Cuenta con la mayor comunidad de desarrolladores frente a sus competidores: AngularJS, VueJS, NextJS. Esto facilita el desarrollo, con mayor cantidad de librerías disponibles y mayor facilidad para solventar errores debido a la cantidad de usuarios.

React y JSX forman un stack tecnológico que nos permite sustituir completamente el stack conformado por HTML, JavaScript y CSS, ganando modularidad durante el desarrollo.

Para modificar el contenido que se muestra al usuario, cuenta con acceso al DOM, que representa la página web como un árbol de nodos, que pueden ser modificados con JavaScript.

2.2. Progressive Web App

El mercado de las aplicaciones está segmentado en función a las distintas tiendas de aplicaciones de cada plataforma. Esto hace que el desarrollo multiplataforma sea complejo. Recientemente han ido surgiendo frameworks como Flutter, aunque la comunidad de desarrollo es todavía pequeña.

Es por ello que para este proyecto nos hemos acogido a las PWA, cuyo soporte ha sido desarrollado para navegadores basados en Chromium, así como Safari; mientras Firefox ha decidido no acoger el estándar de la Web abierta. [1]

Mediante un archivo de manifiesto, se detalla el título, descripción, imágenes y acciones que la aplicación puede ejecutar o intervenir una vez instalada, permitiendo interacción con el sistema operativo.

Con ello, nos permite mantener una sola fuente de código mientras podemos ejecutar nuestra aplicación en la mayoría de los clientes web.

Además, en noviembre de 2016 el tráfico web móvil superó al tráfico web de escritorio. Desde entonces, se ha mantenido la distancia ligeramente, por lo que tendría sentido optimizar la aplicación hacia clientes móviles. [2]

2.2.1. WebAssembly

WebAssembly define un formato de código binario portable e instrucciones correspondientes a modo de interfaz para facilitar interacciones entre programas y el entorno del host. Se trata de un estándar abierto que apunta a soportar la ejecución de cualquier lenguaje en cualquier sistema operativo.

Para ello, requiere de la integración del soporte de WebAssembly por los navegadores. Los navegadores principales (Chrome, Firefox, Safari y Edge) ya soportan la versión 1.0 de WebAssembly.

WebAssembly nos permite ejecutar programas desde nuestra página web en el host con alto rendimiento, puesto que las instrucciones son precompiladas, tarda menos en cargar al ser un binario y permite ejecutar órdenes a bajo nivel, obteniendo un rendimiento más cercano al nativo del sistema en el que se ejecuta (con menor número de capas intermediarias).

2.2.2. Expresiones regulares

Las expresiones regulares son una secuencia de caracteres y operadores especiales que definen y controlan una búsqueda de patrones y filtros en un texto de destino.

Haremos uso de las mismas para segmentar los mensajes recibidos en texto plano, obteniendo así diferentes grupos de texto que conformarán la fecha del mensaje, la hora, el nombre del contacto y el cuerpo del mensaje. Pese a ser conocidos por ser un problema más que una solución, con el correcto uso de la tecnología, puede simplificar mucho el desarrollo. Esto se debe a que podríamos obtener resultados similares mediante el uso de funciones que separen la cadena de caracteres en los lugares adecuados, eliminen caracteres innecesarios y formen los grupos anteriormente descritos, cosa que aumentaría la complejidad del desarrollo y añadiría unidades lógicas a la lógica de negocio.

2.3. Tecnologías en el servidor

2.3.1. GNU/Linux

Pese a no haber dominado el mercado de los escritorios, Linux se encuentra en el 96,3 % de los servidores del mundo [3]. Se trata de un kernel de código libre, por lo que se alinea con este trabajo perfectamente. Junto con los programas asociados y necesarios para un servidor, como puede ser SSH, nos permite operar y administrar el sistema para ejecutar programas y servicios a ofrecer.

2.3.2. Traefik

Un proxy inverso nos permite interceptar peticiones a nuestro servidor y redirigir el tráfico al servicio back-end adecuado. Además, permite la implementación de certificados SSL, que facilita el tráfico seguro en las conexiones externas.

Hemos elegido Traefik frente a alternativas como Nginx o Apache, que, pese a ser contendientes más establecidos, no ofrecen una integración sencilla con Docker. Además, Traefik ofrece integración nativa con Let's Encrypt, autoridad de certificados sin ánimo de lucro que, mediante una prueba, expide certificados para los dominios bajo nuestra propiedad.

2.4. Herramientas de desarrollo

A lo largo del desarrollo del proyecto, hemos hecho uso de las siguientes tecnologías para facilitar el control y versionamiento del código, propiedad intelectual, despliegue de infraestructura y comprobación de errores.

2.4.1. Git

Git es un sistema de control de versiones distribuido, de código libre y gratuito, diseñado para manejar proyectos de cualquier tamaño, independientemente del número de contribuidores, de forma rápida y eficiente. Nos permite versionar cambios de nuestro código, así como recuperar versiones anteriores y desarrollar nuevas funciones en paralelo en diferentes ramas, entre otros.

Además, se trata de un sistema distribuido, por lo que evitamos la centralización del código; lo que nos permite mantener el control de versiones sin necesidad de estar conectados a Internet, ya que contamos con nuestro propio nodo local. Una vez tengamos acceso a Internet, podemos sincronizar nuestro trabajo con el nodo origen u otros.

2.4.2. Docker

Para servir el cliente desarrollado en React, se ha utilizado Docker como tecnología de contenerización y virtualización ligera. Esto nos permitirá aislar nuestra aplicación en una capa superior al sistema operativo, permitiendo ejecutar la misma en cualquier sistema operativo basado en Linux, independientemente de las dependencias que este tenga instalado; siempre y cuando cuente con Docker.

Hemos optado por virtualización ligera para reducir el impacto en el servidor, permitiendo que este varíe sus recursos ocupados en función a la demanda dentro de las capacidades de nuestro servidor.

2.4.3. Software Libre

Durante las fase de desarrollo, se usará Git como sistema de control de versiones, unido a la publicación total del código en repositorios de acceso libre y gratuito, pudiendo definirlo como “Open Source Software” bajo la licencia GNU General Public License v3.0 [4]. Esta licencia nos permite, en resumen:

1. Cualquiera puede copiar, modificar y distribuir este software.
2. Se debe incluir la licencia con todas y cada distribución de este código.
3. Se permite el uso privado de este software.
4. Se permite el uso de este software con fines comerciales.
5. En caso de construir un negocio únicamente de este código, se arriesga a publicar la fuente del resto del código derivado.
6. En caso de modificación, se deben indicar los cambios realizados al código.
7. Cualquier modificación de este código debe ser distribuida con la misma licencia, GPLv3.
8. Este software se provee sin ningún tipo de garantía.
9. Ni el autor del código ni la licencia pueden ser marcadas como responsables de daños producidos por el software.

El código fuente está disponible en Codeberg para consulta, contribuciones y seguimiento de errores. Se ha elegido Codeberg como alternativa de código abierto a otras plataformas como GitHub, ya que está basada en Gitea.

2.4.4. Tests unitarios

A lo largo del desarrollo se han implementado numerosos tests unitarios para comprobar el funcionamiento de las diferentes unidades lógicas que componen la aplicación. Se ha utilizado Jest como librería para estas pruebas, y se ha implementado una cobertura del 85 % de las unidades lógicas de la lógica de negocio. La decisión de utilizar esta librería tiene en cuenta su compatibilidad con React y las facilidades que ofrece durante el desarrollo.

CAPÍTULO 3

Análisis de requisitos

3.1. Introducción

El siguiente diagrama UML representa un resumen visual de los casos de uso que se describirán a continuación:

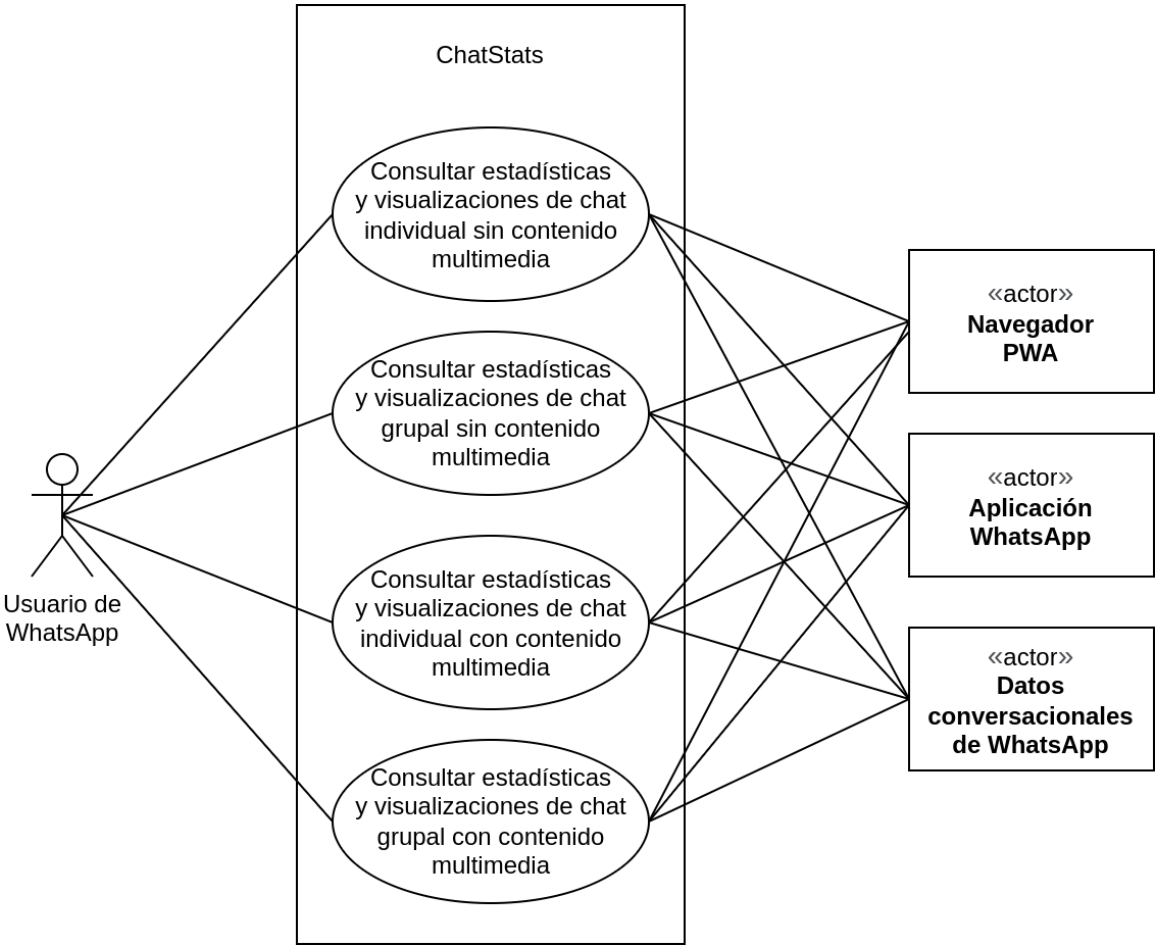

Figura 3.1: Diagrama UML

3.2. Casos de uso

3.2.1. Consultar estadísticas y visualizaciones de chat individual sin contenido multimedia

3.2.1.1. Nombre del caso de uso

Consultar estadísticas y visualizaciones de chat individual sin contenido multimedia.

3.2.1.2. Actores

Usuario de WhatsApp. Navegador con soporte Progressive Web App (PWA). Aplicación WhatsApp Datos conversacionales de WhatsApp.

3.2.1.3. Resumen

El usuario de WhatsApp usará un archivo previamente exportado desde la aplicación WhatsApp como archivo de entrada. Podrá hacerlo desde el explorador de archivos del navegador o compartiendo el archivo en formato *txt* desde el menú de compartir de su sistema operativo, si tiene la Progressive Web App (PWA) instalada. El archivo no incluirá los mensajes multimedia ni su contenido. Una vez el archivo se ha introducido, el usuario puede confirmar la selección y comenzar el análisis del archivo de texto, cálculo de estadísticas y datos para gráficos. Tras el análisis, se mostrarán las diferentes estadísticas y visualizaciones en la ventana del navegador.

3.2.1.4. Secuencia de acciones

1. El usuario introduce el archivo de entrada, ya sea mediante la selección desde el explorador de archivos del navegador, como compartiendo el archivo mediante la PWA si estuviese instalada.
2. El usuario confirma la selección del archivo de entrada.
3. Se realiza el *parseo* del texto, se calculan las estadísticas y se preparan las estructuras de datos para la visualización, mientras el usuario espera en una pantalla de carga. Estas operaciones se realizan en el cliente.
4. Se muestran gráficos interactivos, así como las estadísticas calculadas previamente.

3.2.2. Actores del sistema

3.2.2.1. Usuario de WhatsApp

Se trata del usuario único en el que se centran nuestros casos de uso. Este es un usuario de WhatsApp, ya que ChatStats es compatible con los archivos de chat exportados por esta aplicación.

3.2.2.2. Navegador con soporte Progressive Web App (PWA)

Nuestra aplicación web se ejecuta en un navegador, y por tanto, se trata de un actor para nuestro sistema. Además, si el navegador soporta Progressive Web App (PWA), la aplicación web puede ser instalada, ofreciendo mayor integración con el sistema operativo.

3.2.2.3. Aplicación WhatsApp

Para exportar y generar los datos conversacionales que ChatStats espera como entrada, es necesario hacer uso de la aplicación de WhatsApp. Desde dicha aplicación, se puede seleccionar si la exportación se debe realizar con archivos multimedia o sin los mismos.

3.2.2.4. Datos conversacionales de WhatsApp

Es el fichero de datos que nustro sistema espera como entrada. En este caso de uso, se trata de un fichero de texto plano con todos los datos conversacionales exportados por la aplicación WhatsApp para un chat individual, sin incluir los archivos multimedia.

3.2.3. Especificación suplementaria

3.2.3.1. Reglas de dominio

3.2.3.2. Requisitos no funcionales

- **Interfaces:** leer y analizar archivos de texto plano *txt*.
- **Operación:** Interfaz de Usuario para navegador web, smartphone y tablet.
- **Seguridad:** Los datos no deben ser enviados a ningún servidor externo sin la autorización previa del usuario. Estos datos, además, no deben ser almacenados de manera temporal o permanente en ningún servidor. Todas las conexiones entre cliente y servidor deben estar cifradas con SSL.
- **Portabilidad:** La aplicación podrá ejecutarse en los navegadores: Chrome, Firefox, Edge y Safari; siendo compatible con PWA en los navegadores que lo soporten.

3.2.3.3. Restricciones

De implementación: lenguaje JavaScript tanto para el servidor como para el cliente, y Docker para el despliegue. El código ha de ser libre.

De interfaz: uso de archivos *txt* para importar los chats exportados desde la aplicación WhatsApp.

Legales: RGPD.

3.2.4. Consultar estadísticas y visualizaciones de chat grupal sin contenido multimedia

3.2.4.1. Nombre del caso de uso

Consultar estadísticas y visualizaciones de chat grupal sin contenido multimedia.

3.2.4.2. Actores

Usuario de WhatsApp. Navegador con soporte Progressive Web App (PWA). Aplicación WhatsApp Datos conversacionales de WhatsApp.

3.2.4.3. Resumen

El usuario de WhatsApp usará un archivo previamente exportado desde la aplicación WhatsApp como archivo de entrada. Podrá hacerlo desde el explorador de archivos del navegador o compartiendo el archivo en formato *txt* desde el menú de compartir de su sistema operativo, si tiene la Progressive Web App (PWA) instalada. El archivo no incluirá los mensajes multimedia ni su contenido. Una vez el archivo se ha introducido, el usuario puede confirmar la selección y comenzar el análisis del archivo de texto, cálculo de estadísticas y datos para gráficos. Tras el análisis, se mostrarán las diferentes estadísticas y visualizaciones en la ventana del navegador.

3.2.4.4. Secuencia de acciones

1. El usuario introduce el archivo de entrada, ya sea mediante la selección desde el explorador de archivos del navegador, como compartiendo el archivo mediante la PWA si estuviese instalada.

2. El usuario confirma la selección del archivo de entrada.

3. Se realiza el *parseo* del texto, se calculan las estadísticas y se preparan las estructuras de datos para la visualización, mientras el usuario espera en una pantalla de carga. Estas operaciones se realizan en el cliente.

4. Se muestran gráficos interactivos, así como las estadísticas calculadas previamente.

3.2.5. Actores del sistema

3.2.5.1. Usuario de WhatsApp

Se trata del usuario único en el que se centran nuestros casos de uso. Este es un usuario de WhatsApp, ya que ChatStats es compatible con los archivos de chat exportados por esta aplicación.

3.2.5.2. Navegador con soporte Progressive Web App (PWA)

Nuestra aplicación web se ejecuta en un navegador, y por tanto, se trata de un actor para nuestro sistema. Además, si el navegador soporta Progressive Web App (PWA), la aplicación web puede ser instalada, ofreciendo mayor integración con el sistema operativo.

3.2.5.3. Aplicación WhatsApp

Para exportar y generar los datos conversacionales que ChatStats espera como entrada, es necesario hacer uso de la aplicación de WhatsApp. Desde dicha aplicación, se puede exportal un chat grupal y seleccionar si la exportación se debe realizar con archivos multimedia o sin los mismos.

3.2.5.4. Datos conversacionales de WhatsApp

Es el fichero de datos que nustro sistema espera como entrada. En este caso de uso, se trata de un fichero de texto plano con todos los datos conversacionales exportados por la aplicación WhatsApp para un chat de grupo, sin incluir los archivos multimedia.

3.2.6. Especificación suplementaria

3.2.6.1. Reglas de dominio

3.2.6.2. Requisitos no funcionales

- **Interfaces:** leer y analizar archivos de texto plano *txt*.
- **Operación:** Interfaz de Usuario para navegador web, smartphone y tablet.

- **Seguridad:** Los datos no deben ser enviados a ningún servidor externo sin la autorización previa del usuario. Estos datos, además, no deben ser almacenados de manera temporal o permanente en ningún servidor. Todas las conexiones entre cliente y servidor deben estar cifradas con SSL.
- **Portabilidad:** La aplicación podrá ejecutarse en los navegadores: Chrome, Firefox, Edge y Safari; siendo compatible con PWA en los navegadores que lo soporten.

3.2.6.3. Restricciones

De implementación: lenguaje JavaScript tanto para el servidor como para el cliente, y Docker para el despliegue. El código ha de ser libre.

De interfaz: uso de archivos *txt* para importar los chats exportados desde la aplicación WhatsApp.

Legales: RGPD.

3.2.7. Consultar estadísticas y visualizaciones de chat individual con contenido multimedia

3.2.7.1. Nombre del caso de uso

Consultar estadísticas y visualizaciones de chat individual con contenido multimedia.

3.2.7.2. Actores

Usuario de WhatsApp. Navegador con soporte Progressive Web App (PWA). Aplicación WhatsApp Datos conversacionales de WhatsApp.

3.2.7.3. Resumen

El usuario de WhatsApp usará un archivo previamente exportado desde la aplicación WhatsApp como archivo de entrada. Podrá hacerlo desde el explorador de archivos del navegador o compartiendo el archivo en formato *txt* desde el menú de compartir de su sistema operativo, si tiene la Progressive Web App (PWA) instalada. El archivo incluirá los mensajes multimedia. Una vez el archivo se ha introducido, el usuario puede confirmar la selección y comenzar el análisis del archivo de texto, cálculo de estadísticas y datos para

gráficos. Tras el análisis, se mostrarán las diferentes estadísticas y visualizaciones en la ventana del navegador, con las visualizaciones de contenido multimedia correspondientes.

3.2.7.4. Secuencia de acciones

1. El usuario introduce el archivo de entrada, ya sea mediante la selección desde el explorador de archivos del navegador, como compartiendo el archivo mediante la PWA si estuviese instalada.

2. El usuario confirma la selección del archivo de entrada.

3. Se realiza el *parseo* del texto, se calculan las estadísticas y se preparan las estructuras de datos para la visualización, mientras el usuario espera en una pantalla de carga. Estas operaciones se realizan en el cliente.

4. Se muestran gráficos interactivos, así como las estadísticas calculadas previamente.

3.2.8. Actores del sistema

3.2.8.1. Usuario de WhatsApp

Se trata del usuario único en el que se centran nuestros casos de uso. Este es un usuario de WhatsApp, ya que ChatStats es compatible con los archivos de chat exportados por esta aplicación.

3.2.8.2. Navegador con soporte Progressive Web App (PWA)

Nuestra aplicación web se ejecuta en un navegador, y por tanto, se trata de un actor para nuestro sistema. Además, si el navegador soporta Progressive Web App (PWA), la aplicación web puede ser instalada, ofreciendo mayor integración con el sistema operativo.

3.2.8.3. Aplicación WhatsApp

Para exportar y generar los datos conversacionales que ChatStats espera como entrada, es necesario hacer uso de la aplicación de WhatsApp. Desde dicha aplicación, se puede seleccionar si la exportación se debe realizar con archivos multimedia o sin los mismos. Este caso de uso comprende la exportación con contenido multimedia.

3.2.8.4. Datos conversacionales de WhatsApp

Es el fichero de datos que nustro sistema espera como entrada. En este caso de uso, se trata de un fichero de texto plano con todos los datos conversacionales exportados por la aplicación WhatsApp para un chat individual, incluyendo los archivos multimedia.

3.2.9. Especificación suplementaria

3.2.9.1. Reglas de dominio

3.2.9.2. Requisitos no funcionales

- **Interfaces:** leer y analizar archivos de texto plano *txt*.
- **Operación:** Interfaz de Usuario para navegador web, smartphone y tablet.
- **Seguridad:** Los datos no deben ser enviados a ningún servidor externo sin la autorización previa del usuario. Estos datos, además, no deben ser almacenados de manera temporal o permanente en ningún servidor. Todas las conexiones entre cliente y servidor deben estar cifradas con SSL.
- **Portabilidad:** La aplicación podrá ejecutarse en los navegadores: Chrome, Firefox, Edge y Safari; siendo compatible con PWA en los navegadores que lo soporten.

3.2.9.3. Restricciones

De implementación: lenguaje JavaScript tanto para el servidor como para el cliente, y Docker para el despliegue. El código ha de ser libre.

De interfaz: uso de archivos *txt* para importar los chats exportados desde la aplicación WhatsApp.

Legales: RGPD.

3.2.10. Consultar estadísticas y visualizaciones de chat grupal con contenido multimedia

3.2.10.1. Nombre del caso de uso

Consultar estadísticas y visualizaciones de chat grupal con contenido multimedia.

3.2.10.2. Actores

Usuario de WhatsApp. Navegador con soporte Progressive Web App (PWA). Aplicación WhatsApp Datos conversacionales de WhatsApp.

3.2.10.3. Resumen

El usuario de WhatsApp usará un archivo previamente exportado desde la aplicación WhatsApp como archivo de entrada. Podrá hacerlo desde el explorador de archivos del navegador o compartiendo el archivo en formato *txt* desde el menú de compartir de su sistema operativo, si tiene la Progressive Web App (PWA) instalada. El archivo incluirá los mensajes multimedia del grupo. Una vez el archivo se ha introducido, el usuario puede confirmar la selección y comenzar el análisis del archivo de texto, cálculo de estadísticas y datos para gráficos. Tras el análisis, se mostrarán las diferentes estadísticas y visualizaciones en la ventana del navegador, con las visualizaciones de contenido multimedia correspondientes.

3.2.10.4. Secuencia de acciones

1. El usuario introduce el archivo de entrada, ya sea mediante la selección desde el explorador de archivos del navegador, como compartiendo el archivo mediante la PWA si estuviese instalada.

2. El usuario confirma la selección del archivo de entrada.

3. Se realiza el *parseo* del texto, se calculan las estadísticas y se preparan las estructuras de datos para la visualización, mientras el usuario espera en una pantalla de carga. Estas operaciones se realizan en el cliente.

4. Se muestran gráficos interactivos, incluyendo las visualizaciones multimedia, así como las estadísticas calculadas previamente.

3.2.11. Actores del sistema

3.2.11.1. Usuario de WhatsApp

Se trata del usuario único en el que se centran nuestros casos de uso. Este es un usuario de WhatsApp, ya que ChatStats es compatible con los archivos de chat exportados por esta aplicación.

3.2.11.2. Navegador con soporte Progressive Web App (PWA)

Nuestra aplicación web se ejecuta en un navegador, y por tanto, se trata de un actor para nuestro sistema. Además, si el navegador soporta Progressive Web App (PWA), la aplicación web puede ser instalada, ofreciendo mayor integración con el sistema operativo.

3.2.11.3. Aplicación WhatsApp

Para exportar y generar los datos conversacionales que ChatStats espera como entrada, es necesario hacer uso de la aplicación de WhatsApp. Desde dicha aplicación, se puede seleccionar si la exportación se debe realizar con archivos multimedia o sin los mismos. Este caso de uso comprende la exportación con contenido multimedia.

3.2.11.4. Datos conversacionales de WhatsApp

Es el fichero de datos que nustro sistema espera como entrada. En este caso de uso, se trata de un fichero de texto plano con todos los datos conversacionales exportados por la aplicación WhatsApp para un chat grupal, incluyendo los archivos multimedia.

3.2.12. Especificación suplementaria

3.2.12.1. Reglas de dominio

3.2.12.2. Requisitos no funcionales

- **Interfaces:** leer y analizar archivos de texto plano *txt*.
- **Operación:** Interfaz de Usuario para navegador web, smartphone y tablet.
- **Seguridad:** Los datos no deben ser enviados a ningún servidor externo sin la autorización previa del usuario. Estos datos, además, no deben ser almacenados de manera temporal o permanente en ningún servidor. Todas las conexiones entre cliente y servidor deben estar cifradas con SSL.
- **Portabilidad:** La aplicación podrá ejecutarse en los navegadores: Chrome, Firefox, Edge y Safari; siendo compatible con PWA en los navegadores que lo soporten.

3.2.12.3. Restricciones

De implementación: lenguaje JavaScript tanto para el servidor como para el cliente, y Docker para el despliegue. El código ha de ser libre.

De interfaz: uso de archivos *txt* para importar los chats exportados desde la aplicación WhatsApp.

Legales: RGPD.

CAPÍTULO 4

Arquitectura

4.1. Introducción

En este capítulo trataremos la fase de diseño de este proyecto, así como los detalles de implementación de dicha arquitectura. Primero, presentaremos una vista general del proyecto, dividido en módulos. Con ello tenemos la intención de facilitar al lector una vista general de la arquitectura del proyecto. Posteriormente, presentaremos cada módulo de forma separada y en mayor profundidad.

Finalmente, también cubriremos la arquitectura del lado del servidor, así como la del cliente.

4.2. Arquitectura de procesamiento

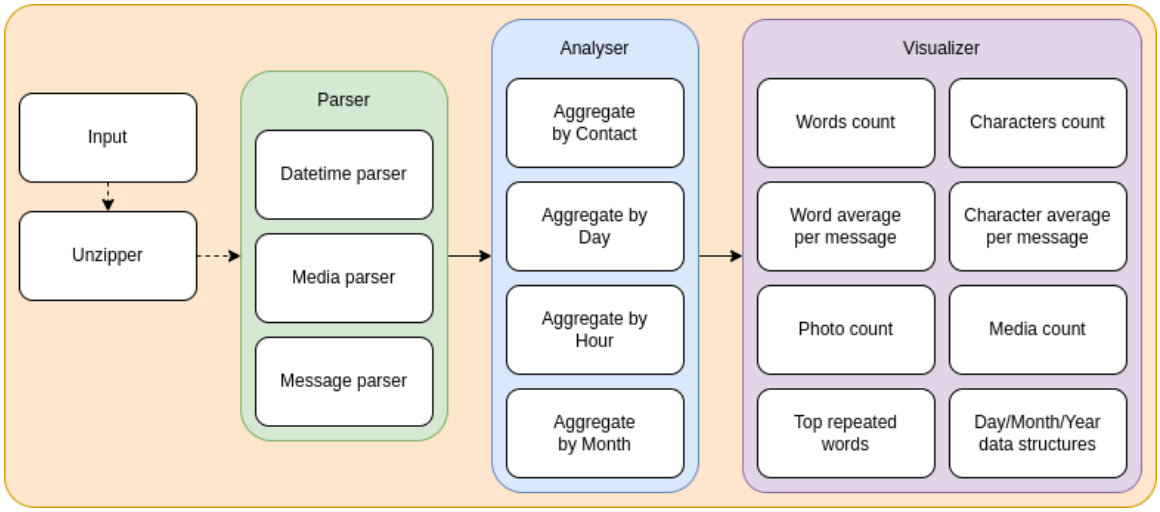

Figura 4.1: Arquitectura de procesamiento

4.2.1. Input

Se trata de un conjunto de funciones que acceden y leen el archivo de entrada que el usuario ha proporcionado. Este archivo puede ser en texto plano o comprimido, como se explica en el siguiente módulo.

4.2.2. Descompresor (Opcional)

Mientras que en los sistemas operativos Android, la aplicación WhatsApp exporta los chats en texto plano, en los sistemas operativos iOS se exportan en un archivo comprimido en formato *zip*. En este último caso, dentro del archivo se encuentra un fichero denominado *_chat.txt*, junto con el resto de contenido multimedia (fotos, vídeos y documentos) en caso de haber optado por exportarlos.

4.2.3. Parser

Este módulo se compone de diferentes submódulos para parsear los diferentes tipos de objetos contenidos en cada grupo de la cadena de texto.

Todo comienza por encontrar una expresión regular que pueda encontrar coincidencias en el archivo de texto, así como dividir estas coincidencias en los grupos correspondientes.

Analizamos a continuación un mensaje de ejemplo:

```
17/07/2022, 01:28 - Juan Pedro: Este es un mensaje de prueba
```

Podemos observar que un mensaje se compone de la fecha, la hora, el nombre del contacto y el propio cuerpo del mensaje.

Para poder separar cada mensaje se ha llegado a la siguiente expresión regular:

```
/(\d{2}\/\d{2}\/\d{4}),\s(\d(?:\d)?:\d{2})\s-\s([^:]*):\s(.*?)(?=\s*\d
    {2}\/\d{2}\/\d{4},\s|$)/ug
```

Los grupos de captura que lo componen se describen en detalle en el Apéndice D.

4.2.3.1. Parser de cadenas de caracteres de tiempo

Este submódulo recoge las coincidencias del grupo de captura 1 y 2, devolviendo un objeto de tipo *Date*, lo cual nos permitirá operar con el fácilmente. Este módulo considera el *locale* del cliente, invirtiendo mes y día para el caso *en_US*.

4.2.3.2. Parser de archivos multimedia

Existen dos opciones para exportar un chat: con contenido multimedia o sin él.

Con contenido multimedia

En el primer caso, WhatsApp exporta el fichero de texto plano, además de todas las imágenes, vídeos, música, notas de voz o documentos del chat. En caso de iOS, todos los archivos se agrupan en un archivo comprimido en *zip*, mientras que en Android se exportan todos los archivos como individuales, permitiendo al usuario compartirlos con la aplicación deseada (o guardarlos).

En un mensaje con contenido multimedia, observaremos que el cuerpo incluirá el nombre del fichero que se ha exportado con la extensión del formato del mismo. Por ello, categorizamos como *voice_message*, *video_file*, *sticker* o *image* en función a la extensión; *.opus*, *.mp4*, *.webp* o *.jpg*, respectivamente.

Se indican a continuación dos mensajes de ejemplo:

```
17/10/2022, 21:11 - Juan Pedro: PTT-20221017-WA0078.opus (file attached)
20/10/2022, 10:37 - Juan Pedro: IMG-20221020-WA0013.jpg (file attached)
14/11/2022, 18:58 - Juan Pedro: VID-20221114-WA0039.mp4 (file attached)
```

```
24/11/2022, 19:13 - Jaime Conde: STK-20220717-WA0090.webp (file attached)
```

Sin contenido multimedia

Para el segundo caso, cada vez que un mensaje sea contenido multimedia, aparecerá *¡Media omitted¿* (multimedia omitido). Estos pueden ser fotos, vídeos, música, notas de voz o documentos. Se definirán como *undefined* o indefinidos, ignorándose en las visualizaciones y módulos posteriores.

```
17/07/2022, 01:33 - Juan Pedro: <Media omitted>
```

4.2.3.3. Parser de mensajes

Este módulo convierte las cadenas de textos en un objeto JSON con la siguiente estructura:

```
{
  date: new Date("2022-10-17T10:37:00"),
  from: "Juan Pedro",
  text: "IMG-20221020-WA0013.jpg",
  type: "message",
  media_type: "image"
}
```

En caso de tratarse de un mensaje de texto (no multimedia), el "*media_type*" será *undefined* (indefinido) y "*text*" contendrá el cuerpo del mensaje.

Serán estos los objetos que se utilizarán más adelante para calcular las estadísticas y las estructuras de datos de visualización.

4.2.4. Agregadores

Los mensajes se encuentran segregados en una lista, por lo que a continuación, el módulo de agregador se encargará de agregar los mensajes en diferentes grupos. En el código los hemos llamado polarizadores. Se describen los distintos submódulos a continuación:

4.2.4.1. Agregador por contacto

ChatStats se encarga de calcular las estadísticas de cada contacto para visualizarlas y mostrarlas en comparación con el resto de contactos. Hablamos de numerosos contactos, puesto que es compatible con chats individuales y grupales.

El resultado de este submódulo será un objeto JSON con una clave por cada contacto (su nombre), que contendrá un array de los mensajes enviados por este. Se indica un ejemplo:

```
{
  "Jaime": [...messagesByJaime],
  "Juan Pedro": [...messagesByJuanPedro],
  ...
}
```

donde los array de mensajes contienen objetos definidos en el *Parser de mensajes*.

4.2.4.2. Agregador por día

Este agregador toma como entrada la salida del submódulo anterior: los mensajes agregados por contacto. Con ello se procede a agregarlos, además, por día de la semana: de lunes a domingo. Se usará el nombre del día de la semana como subclave.

El resultado son objetos con la siguiente estructura:

```
{
  "Jaime": {
    "monday": [...messagesByJaimeOnMonday],
    "tuesday": [...messagesByJaimeOnTuesday],
    ...,
    "sunday": [...messagesByJaimeOnSunday]
    },
  "Juan Pedro": {
    "monday": [...messagesByJuanPedroOnMonday],
    "tuesday": [...messagesByJuanPedroOnTuesday],
    ...,
    "sunday": [...messagesByJuanPedroOnSunday]
  },
  ...
}
```

El objetivo de esta estructura de datos es visualizar la distribución de los mensajes a lo largo de la semana, en media.

4.2.4.3. Agregador por hora

Este agregador toma también como entrada los mensajes agregados por contacto. Con ello se procede a agregarlos, además, por hora del día, usando la hora en formato 24 horas como subclave de agregación: de 00 a 23 horas.

El resultado son objetos con la siguiente estructura:

```
{
  "Jaime": {
    "00": [...messagesByJaimeAt00],
    "01": [...messagesByJaimeAt01],
    ...,
    "23": [...messagesByJaimeAt23]
  },
  "Juan Pedro": {
    "00": [...messagesByJuanPedroAt00],
    "01": [...messagesByJuanPedroAt01],
    ...,
    "23": [...messagesByJuanPedroAt23]
  },
  ...
}
```

El objetivo de esta estructura de datos es visualizar la distribución de los mensajes a lo largo del día, en media.

4.2.4.4. Agregador por mes

Este agregador toma también como entrada los mensajes agregados por contacto. Con ello se procede a agregarlos, además, por mm/YYYY, por lo que deja de tratarse de un agregador acotado: pueden haber tantos meses como se haya hablado.

El resultado son objetos con la siguiente estructura:

```
{
  "Jaime": {
    "10/2022": [...messagesByJaimeOnOctober2022],
    "11/2022": [...messagesByJaimeOnNovember2022],
    ...
  },
  "Juan Pedro": {
    "10/2022": [...messagesByJuanPedroOnOctober2022],
    "11/2022": [...messagesByJuanPedroOnNovember2022],
```

```
      ...
    },
    ...
}
```

El objetivo de esta estructura de datos es visualizar la distribución de los mensajes a lo largo del tiempo, con una agregación mensual.

4.2.5. Visualizador

Este módulo prepara los datos para ser representados por la librería de visualización elegida: *ChartJS*. Además, también se procesan datos para otras librerías de visualización, como las nubes de palabras o *word clouds*.

4.2.5.1. Contador de palabras

Este submódulo cuenta las palabras que hay en los mensajes de cada contacto y calcula la suma total de las mismas, obteniendo el número de palabras totales enviadas por cada contacto.

4.2.5.2. Contador de caracteres

Este submódulo cuenta los caracteres que hay en los mensajes de cada contacto y calcula la suma total de los mismos, obteniendo el número de caracteres totales enviados por cada contacto.

4.2.5.3. Media de palabras por mensaje

Este submódulo devuelve el número medio de palabras por mensaje para cada contacto.

4.2.5.4. Media de caracteres por mensaje

Este submódulo devuelve el número medio de caracteres por mensaje para cada contacto.

Tanto el número de caracteres como el número de palabras suelen indicar la misma información respecto a qué contacto escribe más.

4.2.5.5. Contador de multimedia

En caso de que existan objetos JSON con el campo "*media_type*" distinto de *undefined*, este módulo contará cuántos archivos multimedia de cada tipo ha mandado cada contacto.

4.2.5.6. Contador de palabras más repetidas

Este módulo elimina las palabras más comunes del español y el inglés, así como otros mensajes que WhatsApp añade, como *Media ommited* o *This message has been deleted*.

Las listas de palabras más comunes del español e inglés se han recopilado de distintas fuentes, combinado y eliminado repeticiones.

A la salida de este módulo, se entrega un diccionario con las palabras más repetidas y el número de veces que aparece cada una.

4.2.5.7. Generador de estructuras de datos por día, hora y mes

Para el gráfico de barras con el número de mensajes en el tiempo, *ChartJS* necesita una estructura de datos para mensajes por día, siendo los días la variable independiente y el número de mensajes la variable dependiente.

4.3. Arquitectura en el servidor

Se muestra a continuación una grafo de la arquitectura en el servidor, donde se muestran las capas que lo componen.

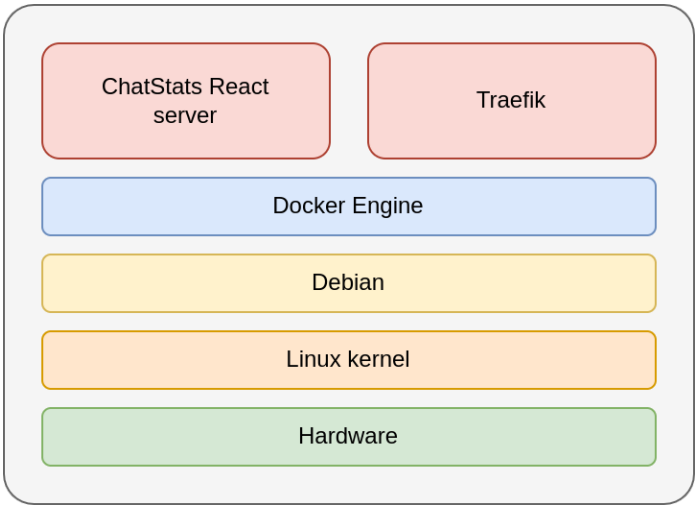

Figura 4.2: Arquitectura en el servidor

Se ha decidido no instalar el software directamente sobre el sistema operativo, evitando problemas de dependencias y distintas versiones de las mismas para los componentes del sistema operativo. Asimismo, se evitan problemas de seguridad que puedan venir por vulnerabilidades en el código fuente y sus dependencias.

Hemos elegido virtualización ligera para ejecutar nuestro código en contenedores, por las razones que se exponen:

- Se contienen las dependencias de terceros en una imagen.
- En caso de vulnerabilidad, solo se expone el contenedor y no el sistema completo.
- Los recursos se ocupan dinámicamente en función a las necesidades, al contrario que con la virtualización completa.
- Permite el despliegue en cualquier sistema operativo compatible con Linux, salvo arquitecturas ARM (que no es típica en servidores).

4.3.1. ChatStats React Server

Se trata del servidor de React que sirve el contenido. Tras construir la versión de producción con el código fuente, este contenedor sirve el contenido estático final, que enviará al cliente completamente cuando este solicite la aplicación web.

4.3.2. Traefik

Se ha decidido usar Traefik como *proxy* inverso, que se sitúa frente al servidor de ChatStats para redirigir las peticiones realizadas a su contenedor correspondiente en el puerto adecuado.

Además, Traefik gestiona los certificados SSL haciendo uso de *Let's Encrypt*: autoridad sin ánimo de lucro que provee certificados para la capa TLS sin coste alguno.

CAPÍTULO 5

Caso de estudio

5.1. Introducción

In this chapter we are going to describe a selected use case. This description will cover the main Wool features, and its main purpose is to completely understand the functionalities of Wool, and how to use it.

5.2. Rule edition

...

CAPÍTULO 6

Conclusiones y futuras líneas de trabajo

En este capítulo se exponen las conclusiones extraídas del proyecto, así como recomendaciones y posibles futuras líneas de trabajo.

6.1. Conclusiones

Académicamente, este proyecto ha desarrollado una doble función: la de proyecto personal y, la de trabajo de fin de grado.

La motivación principal del proyecto era desarrollar una aplicación web para ayudar a analizar conversaciones de WhatsApp, permitiendo detectar y mejorar los problemas encontrados; y este pro

6.2. Objetivos conseguidos

Creación de una aplicación web.

Compatibilidad con Progressive Web App (PWA).

Compatibilidad con datos de conversaciones grupales e individuales.

Compatibilidad con datos de conversaciones con y sin contenido multimedia.

Compatibilidad con todos los formatos de exportación de la aplicación WhatsApp para distintos sistemas operativos.

Cálculo de estadísticas generales de los datos.

Visualizaciones gráficas para los distintos estadísticos.

Posibilidad de interacción con los gráficos.

Implementación de tests para estabilidad y comprobación del código fuente.

6.3. Futuras líneas de trabajo

- Migración de los módulos con mayor carga de trabajo a Web Assembly (WASM), permitiendo obtener un rendimiento mayor y cercano al nativo del cliente que ejecuta la aplicación.
- Mayor facilidad para añadir módulos, mediante un sistema de carpetas y plugins.
- Inclusión de visualizaciones para el análisis de sentimientos y su evolución en el tiempo.
- Añadir soporte para otras plataformas de mensajería como Telegram.
- Implementación de *cache* en la Progressive Web App (PWA) para poder ejecutar la aplicación sin necesidad de acceso a Internet (una vez instalada).
- Mejora de la documentación para futuros contribuyentes al proyecto.

APÉNDICE A

Impacto del proyecto

A.1. Impacto social

A.1.1. Introducción

Este proyecto aporta herramientas para el estudio y análisis de las relaciones personales llevadas a cabo mediante la aplicación WhatsApp, permitiendo evaluar y mejorar y tomar decisiones las mismas mediante la observación de los resultados.

A.1.2. Descripción de impactos relevantes relacionados con el proyecto

El cuidado de las comunicaciones que tienen lugar a través de WhatsApp puede ayudar a las personas a dedicar el tiempo necesario a mantener relaciones saludables y darse cuenta de posibles fallos que están teniendo lugar en la comunicación a largo plazo, o analizar su comportamiento.

Otro impacto relevante a la ética del proyecto es las consideraciones de privacidad en un tema tan importante como son las conversaciones privadas del usuario, que se ha considerado prioritario durante las decisiones de diseño y arquitectura.

A.1.3. Conclusiones

Las consideraciones sociales de este proyecto son el núcleo del desarrollo del mismo, así como la causa por la que el proyecto comenzó en un primer lugar.

APÉNDICE B

Presupuesto económico

B.1. Costes

COSTE DE MANO DE OBRA (coste directo)		
Horas	Precio/hora	Total
360	12 €	4,320.00 €

COSTE DE RECURSOS MATERIALES (coste directo)				
	Precio de compra	Uso en meses	Amortización (en años)	Total
Portátil personal	1,500.00 €	6	5	150.00 €
Tablet	1,099.00 €	6	5	109.90 €
Escritorio ETSIT	130 €	6	5	15.60 €
Silla ETSIT	50 €	6	5	5.00 €
COSTE TOTAL DE RECURSOS MATERIALES				280.50 €

GASTOS GENERALES (costes indirectos)	15 %	de CD	690.08 €
BENEFICIO INDUSTRIAL	6 %	of CD+CI	317.43 €

MATERIAL FUNGIBLE	
Impresión y encuadernación	50.00 €

PRESUPUESTO SUBTOTAL		5,658.01 €
IVA APLICABLE	21 %	1,188.18 €

PRESUPUESTO TOTAL	6,846.19 €

APÉNDICE C

Código

C.1. Acceso al código fuente

Acceso principal: `https://codeberg.org/devve/chatstats`

Copia (mirror): `https://github.com/d3vv3/chatstats`

APÉNDICE D

Expresiones regulares

D.1. Grupos de captura

Como se decribe en el Capítulo 4, poder separar cada mensaje se ha llegado a la siguiente expresión regular, cuyos grupos se explicaran a continuación:

```
/(\d{2}\/\d{2}\/\d{4}),\s(\d(?:\d)?:\d{2})\s-\s([^:]*):\s(.*?)(?=\s*\d
    {2}\/\d{2}\/\d{4},\s|$)/ug
```

Se denotan los distintos grupos de captura por las agrupaciones realizadas con los paréntesis. Se exponen:

Grupo de captura 1: fecha

```
(\d{2}\/\d{2}\/\d{4})
```

Se encarga de la fecha en formato *dd/mm/YYYY*, denotado con "$\backslash d\{X\}$" que indica que se buscan X dígitos de 0 a 9 seguidos, separados por un "/". En las expresiones regulares hay que escapar los "/" o *slash* con un "\" o *backslash*.

Durante numerosas pruebas se ha observado que la fecha siempre sigue este formato, independientemente del sistema operativo. No hay consistencia entre los ajustes del parámetro *locale* de *en_US* y *es_ES*, siendo *mm/dd/YYY* y *dd/mm/YYYY* respectivamente. Para ello, el cliente accederá al *locale* para actuar en consecuencia más adelante. No se ha probado para otras configuraciones.

Grupo de captura 2: hora

```
(\d(?:\d)?:\d{2})
```

Se encarga de la la hora en formato *hh:MM*, aunque en alguna ocasión se ha observado que no hay consistencia si la hora es de un solo dígito, pudiendo aparecer 01:00 o 1:00 en función del dispositivo y la versión de WhatsApp que ejecuta. Es por ello que la expresión regular es algo más compleja y se buscan dígitos a la izquierda del carácter "*:*" independientemente del número de repeticiones del mismo. Los minutos si han mostrado consistencia, por lo que se utiliza "$\backslash d\{2\}$".

Grupo de captura 3: contacto

```
([^:]*)
```

Se encarga del nombre del contacto. Busca la repetición de caracteres ilimitados a excepción del carácter "*:*", ya que éste es un separador.

Grupo de captura 4: mensaje

```
(.*?)
```

Se encarga del cuerpo del mensaje. Busca la repetición de caracteres ilimitados, incluyendo caracteres unicode para tener los emoticonos en cuenta. Esta búsqueda de caracteres se realiza de manera perezosa, expandiendo las coincidencias en caso posible, siempre que no coincida con el siguiente grupo de captura (*lookahead*).

Look ahead o mirada hacia delante

```
(?=\s*\d{2}\/\d{2}\/\d{4},\s|$)
```

Si únicamente contáramos con el grupo de captura 4, solo se reconocería el primer mensaje, puesto que se reconocería el resto del texto como cuerpo del primer mensaje. Para solucionarlo, en el grupo de captura 4 se intentan reconocer el menor número posible de coincidencias, hasta el siguiente patrón reconocido. Este patrón es una mirada hacia delante conformada por la misma expresión regular que en el grupo de captura 1, acompañada seguida de una coma “,” y un espacio.

Bibliografía

[1] Jared Newman. "Firefox just walked away from a key piece of the open web". https://www.fastcompany.com/90597411/mozilla-firefox-no-ssb-pwa-support. Online; accedido a 11 de octubre de 2022.

[2] Pablo G. Bejerano. "El tráfico web procedente de smartphones ya supera al de ordenadores". https://blogthinkbig.com/el-trafico-web-procedente-de-smartphones-ya-supera-al-de-ordenadores, 2014. Online; accedido a 14 de octubre de 2022.

[3] Barry Elad. "Linux Statistics 2022 – Market Share, Usage Data and Facts". https://www.enterpriseappstoday.com/stats/linux-statistics.html. Online; accedido a 10 de octubre de 2022.

[4] Free Software Foundation. "GNU General Public License". https://www.gnu.org/licenses/gpl-3.0.en.html. Online; accedido a 18 de octubre de 2022.