

Trabajo fin de grado

Plataforma web para el análisis de mensajería instantánea



Juan Bautista Menchero Amigo

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Plataforma web para el análisis de mensajería instantánea

Autor: Juan Bautista Menchero Amigo
Tutor: Esther Guerra Sánchez

junio 2022

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución comunicación pública y transformación de esta obra sin contar con la autorización de los titulares de la propiedad intelectual.

La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (*arts. 270 y sgts. del Código Penal*).

DERECHOS RESERVADOS

© 24 de Mayo de 2022 por UNIVERSIDAD AUTÓNOMA DE MADRID
Francisco Tomás y Valiente, nº 1
Madrid, 28049
Spain

Juan Bautista Menchero Amigo

Plataforma web para el análisis de mensajería instantánea

Juan Bautista Menchero Amigo

C\ Francisco Tomás y Valiente N° 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi madre.

Este título es tan tuyo como mío.

*El regalo más grande que le puedes dar a los demás
es el ejemplo de tu propia vida.*

Bertolt Brecht

AGRADECIMIENTOS

Principalmente quiero agradecer a mi madre, *María Jesús Amigo Rodríguez*, la única persona que se ha esforzado en que yo llegara hasta aquí más que yo mismo. Te lo debo todo, gracias por transmitirme los valores del amor, la bondad, el trabajo, la fuerza, la flexibilidad,... Eres la persona que más admiro.

A mi padre, *Juan Bautista Menchero García*, por haberme inculcado la importancia del conocimiento y la constancia. Por despertar mi interés por todas las cosas como fuente inagotable de felicidad.

A mi hermano, *Hugo Jiménez Amigo*, por mostrarme la vida que quería seguir y no para la que estaba predestinado, y haberme limpiado y allanado el camino que era desconocido para ambos. Por animarme y guiarme a ser la mejor versión de mí que puedo ser.

A *Aitana Rickert Llacer*, por haberme acompañado en cada línea de código y cada página de la carrera con tu luz infinita. Por haberme cuidado y dado todo lo que necesitaba cuando yo no era capaz de darlo de vuelta, pero sobre todo por enseñarme a ser mejor persona, aunque no estés para verlo.

A *Álvaro Martínez Morales* (yo sí que me sé tus apellidos), por ser lo mejor que me ha dado esta etapa, no podía soñar nada que supere haberme llevado alguien más en mi familia.

RESUMEN

En este trabajo se ha diseñado e implementado tanto el software como la infraestructura necesarias para ofrecer una aplicación web accesible por el público para la generación de reportes de interés sobre su uso de aplicaciones de mensajería instantánea.

El objetivo es contribuir a la democratización de los datos, así como concienciar a los usuarios de la información que se puede extraer de ellos.

En el contenido de esta memoria se exploran un conjunto de buenas prácticas de la arquitectura de software llevadas a un entorno real en producción, con enfoque en la alta disponibilidad, la escalabilidad, la integración continua y la inmutabilidad, aprovechando la velocidad de desarrollo y abstracciones que permiten las plataformas en la nube actuales.

PALABRAS CLAVE

Computación en la nube, Análisis de datos, Ingesta de datos, Alta disponibilidad, Escalabilidad, Integración continua, Inmutabilidad, AWS, Amazon Web Services, Terraform, Redshift, Vue

ABSTRACT

This work details the design and implementation of both the software and infrastructure that are necessary to provide public access to a web application for the generation of reports of interest on their use of instant messaging applications.

It aims to contribute to the democratization of the access to data, as well as to raise awareness among users of the information that can be extracted from their private data.

The content of this report explores a set of best practices of software architecture taken into a real production environment, focusing on high availability, scalability, continuous integration and immutability, taking advantage of the speed of development and abstractions allowed by current cloud platforms.

KEYWORDS

Cloud computing, Data analysis, Data ingestion, High availability, Scalability, Continuous integration, Immutability, AWS, Amazon Web Services, Terraform, Redshift, Vue

ÍNDICE

1 Introducción	1
1.1 Motivación	1
1.2 Objetivos	1
1.3 Organización de la memoria	2
2 Estado del arte	3
2.1 Derecho de acceso	3
2.2 Información destilada de libre acceso	4
2.2.1 Spotify Wrapped	4
2.2.2 Tinder Insights	5
2.2.3 Chat Visualizer	6
2.2.4 Limitaciones de las herramientas analizadas	6
2.3 Computación en la nube	7
2.4 Bases de datos	8
2.5 Frameworks de frontend	8
3 Análisis de requisitos	11
3.1 Reportes individuales	11
3.2 Reportes comparativos	12
3.3 Requisitos no funcionales	12
4 Diseño	13
4.1 Experiencia de usuario	13
4.1.1 Presentación	14
4.1.2 Instrucciones de exportación	14
4.1.3 Reportes individuales	15
4.1.4 Reporte comparativo	16
4.2 Infraestructura	18
4.2.1 Almacenamiento del código	18
4.2.2 Proveedor de cloud	18
4.2.3 Infraestructura como código	19
4.2.4 Hosting de la plataforma	19
4.2.5 Base de datos	19
4.2.6 Control de acceso	20

4.3 Planificación	21
5 Implementación	23
5.1 Interfaz de usuario	23
5.2 Procesado de datos	26
5.3 Infraestructura como código	27
6 Pruebas	29
7 Conclusiones y trabajo futuro	31
7.1 Conclusiones	31
7.2 Trabajo futuro	32
Bibliografía	34

LISTAS

Lista de figuras

2.1	Spotify Wrapped	4
2.2	Tinder Insights	5
2.3	Chat Visualizer	6
2.4	Cuota de mercado de proveedores de cloud	7
2.5	Frameworks de frontend	8
4.1	Pantalla de presentación del proyecto	14
4.2	Pantalla de instrucciones	15
4.3	Reporte de emojis más usados	15
4.4	Reporte de uso acumulativo por horas	16
4.5	Pantalla de consentimiento de datos	17
4.6	Pantalla de métricas comparativas	17
4.7	Diagrama de arquitectura de la plataforma	18
5.1	División por componentes	23
5.2	Ejemplo de componente en Vue	24
5.3	Estructura de ficheros	25
5.4	Diagrama de flujo de Vuex	26
5.5	Esquema del histórico de mensajes de Telegram	27
5.6	Ejemplo de tarea ejecutada por GitHub Actions	28

INTRODUCCIÓN

Antes de comenzar con el desarrollo técnico del proyecto conviene contextualizar las motivaciones 1.1 y objetivos 1.2 del mismo. Así como plantear un esquema de la organización de esta memoria 1.3 para facilitar la comprensión del objetivo de cada capítulo.

1.1. Motivación

En la actualidad, como usuarios de un gran número de servicios informáticos gratuitos, generamos una cantidad inmensa de información sobre nosotros mismos todos los días a la que solo tienen acceso las plataformas que nos prestan los servicios que consumimos [1].

La mayoría de la población, en muchos casos hasta ajena de la definición de metainformación, ni siquiera puede plantearse ser conscientes de las conclusiones que son capaces de sacar sobre nosotros mismos a partir de nuestros datos, lo que conlleva un descuido cada vez más asumido sobre nuestra privacidad.

Este trabajo propone las bases para una plataforma de acceso libre y gratuito, que podrá expandirse gracias a la comunidad opensource, a lo que típicamente resultaría un proyecto interno de análisis de datos para cualquier empresa que tuviera acceso a los mismos para atajar esa falta de conocimiento.

1.2. Objetivos

La intención principal de este trabajo es concienciar al público de la información que los actuales propietarios de sus datos pueden obtener sobre su intimidad, para, en última instancia, promover una gestión más consciente de su información.

En segundo plano también se pretende entretenir a los visitantes del sitio con los infográficos generados, siendo el boca a boca el único método viable actualmente para publicitar la plataforma y tener un mayor alcance para el mensaje.

En tercer lugar, este proyecto pretende tener recorrido más allá del ámbito académico, sentando las bases para una comunidad de desarrollo de código abierto, con un enfoque en la separación de conceptos [2] para añadir nuevas analíticas de manera independiente para favorecer la contribución de código por parte de la comunidad.

Por último y a título personal, el desarrollo realizado ha servido como oportunidad para extraer conclusiones creativas a partir de lenguaje natural y metadatos, así como explorar tecnologías de cloud provistas por AWS para la ingesta y procesado de los datos y aplicar buenas prácticas de la arquitectura de software en un entorno de producción real.

1.3. Organización de la memoria

En este trabajo presento una aplicación web donde los usuarios pueden analizar sus copias de seguridad de aplicaciones de mensajería instantánea (como Telegram o What's App) y visualizar varios reportes con información que se puede destilar en base a esos históricos.

A lo largo de los siguientes capítulos de la memoria detallo:

- 1.– **Estado del arte:** Se profundiza en el marco legal que permite la realización de este proyecto, el estudio de otros proyectos similares así como un pequeño resumen del contexto y alternativas de las principales tecnologías elegidas (AWS, Redshift y Vue).
- 2.– **Análisis de requisitos:** Se recopilan todos los requisitos tanto funcionales como no funcionales identificados, en los que se basan las decisiones tecnológicas detalladas más adelante.
- 3.– **Diseño:** Se realiza una propuesta de experiencia de usuario que satisfaga todos los requisitos funcionales indicados previamente, la infraestructura que de soporte a los requisitos no funcionales y una descripción de la planificación de la ejecución a seguir.
- 4.– **Implementación:** Profundiza en la solución de código adoptada tanto para la infraestructura como para la plataforma.
- 5.– **Pruebas:** Se resumen las pruebas realizadas para garantizar la calidad del proyecto.
- 6.– **Conclusiones y trabajo futuro:** Recopila las conclusiones alcanzadas tras el desarrollo de este trabajo y propone una serie de futuros pasos a seguir tras su defensa.

ESTADO DEL ARTE

En este capítulo se profundiza tanto en el marco legal 2.1 que permite la realización de este proyecto, así como en el estudio de otros proyectos similares 2.2 y un pequeño resumen del contexto y alternativas de las principales tecnologías elegidas (AWS 2.3, Redshift 2.4 y Vue 2.5).

2.1. Derecho de acceso

El 14 de abril de 2016 se aprobó en el Parlamento Europeo el Reglamento General de Protección de Datos [3], entrando en vigor el 24 de Mayo de 2016 y concediendo un periodo de aplicación de dos años hasta el 24 de Mayo de 2018. A partir del 25 de Mayo de 2018 todas las empresas, organizaciones, organismos o instituciones dentro del marco europeo comenzaron a tener la obligación, bajo multa por incumplimiento de hasta 20 millones de euros, de ofrecer a los usuarios de una manera fácilmente accesible y legible una copia de sus datos almacenados.

Gracias a esta ley podemos solicitar ante cualquier plataforma que almacene o trate nuestros datos una copia de seguridad de toda nuestra información que tengan disponible. Desde Google [4], Facebook [5], Twitter [6], Spotify [7], Tinder [8] o, en el caso de la información de la que es sujeto el análisis de este trabajo, a aplicaciones de mensajería instantánea como What's App [9] o Telegram [10].

Esto nos permite tener acceso a una fuente de datos para el proyecto de manera sencilla y fácilmente ingerible por el sistema propuesto, cuando disponer de fuentes reales de datos es usualmente la parte más complicada a la hora de aterrizar una idea conceptual en un entorno de producción.

No obstante esta ley, aunque obliga a las empresas a compartir la propiedad de los datos crudos con el individuo que los genere, no impone bajo ninguna cláusula que se compartan las conclusiones destiladas que generan a partir de esa información cruda.

2.2. Información destilada de libre acceso

En esta sección analizaremos algunas aplicaciones existentes con forma u objetivos similares, tratando de observar si hay un nicho de usuarios interesados que den viabilidad al proyecto, para posteriormente enumerar las limitaciones de estos a subsanar.

2.2.1. Spotify Wrapped

Desde 2016 Spotify lanza anualmente una campaña de marketing [11] que permite a sus usuarios visualizar una compilación de sus datos de uso como los mostrados en la figura 2.1, comparándolos con el resto de la comunidad que utiliza la aplicación. Desde resúmenes meramente estadísticos, así como análisis de emociones o intereses.

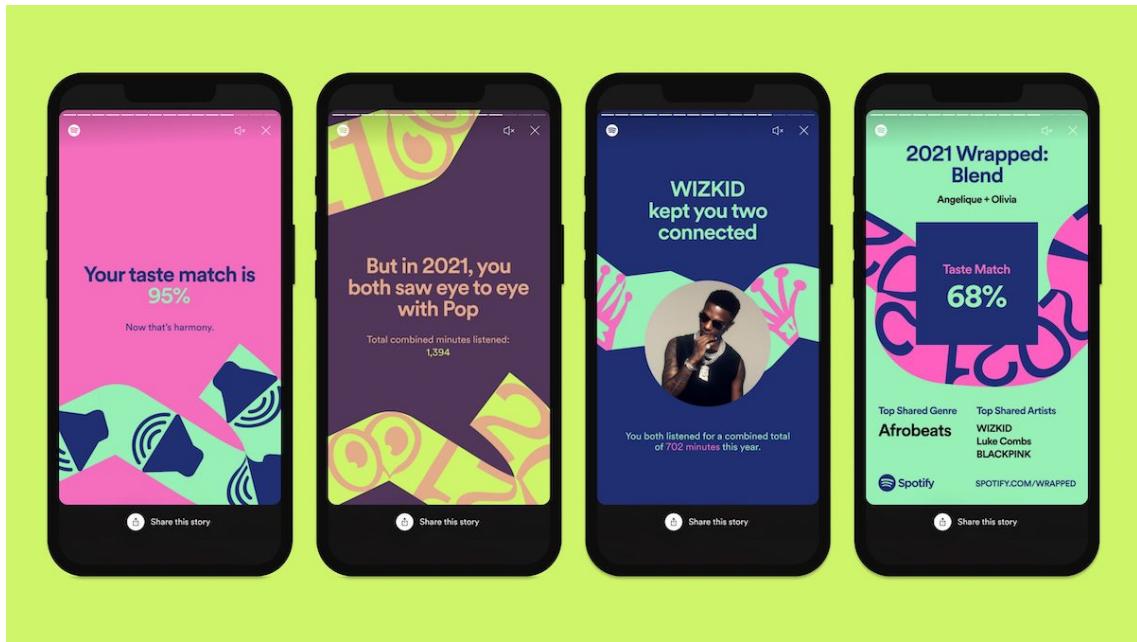


Figura 2.1: Ejemplos de pantallas de reporte de la iniciativa Spotify Wrapped. Fuente: Spotify.com

Esta campaña ha tenido un gran impacto social y cultural, llegando a formar parte de la cultura pop de las nuevas generaciones, y generando en el inconsciente colectivo inquietud e interés por los datos a una población regularmente ajena y desinteresada de su información en la era digital.

2.2.2. Tinder Insights

Creada en 2019 por Dora Szucs (Software Engineer) y Krisztina Szucs (UX Designer) de manera independiente [12], como se puede observar en la figura 2.2 da acceso a información estadística sobre el uso de la aplicación Tinder, como la cantidad de mensajes enviados y recibidos, tiempo medio de chat, ...

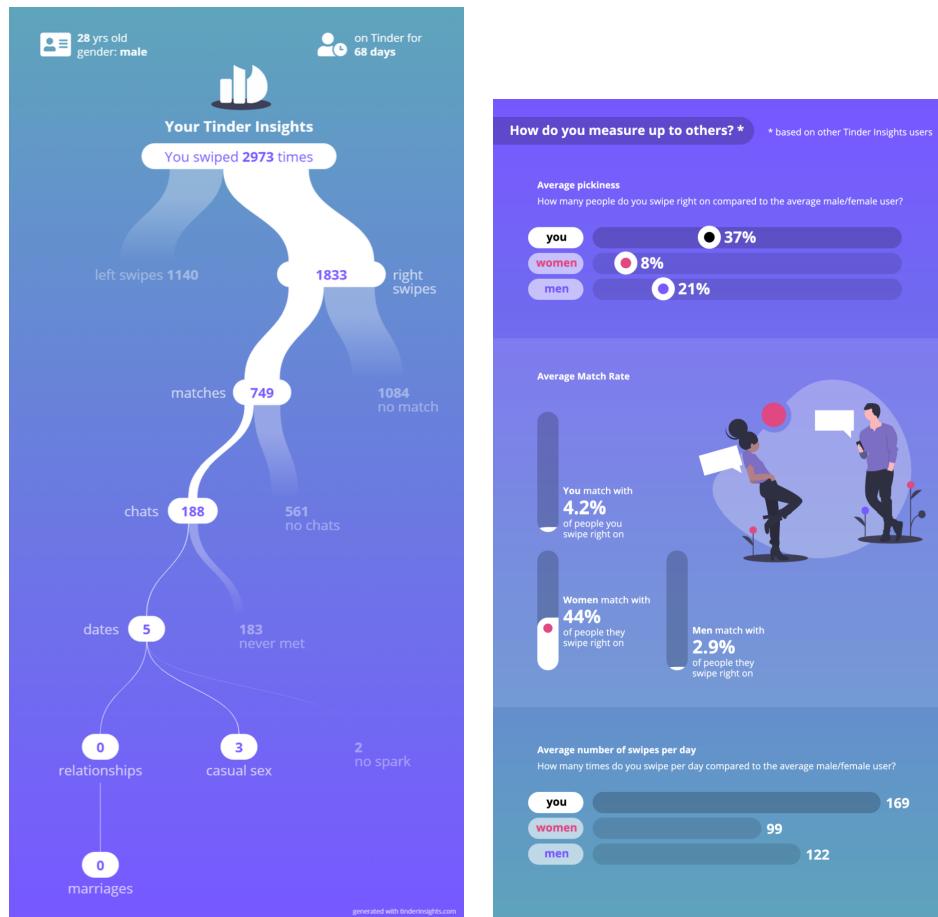


Figura 2.2: Ejemplos de pantallas de reporte de Tinder Insights. Fuente: TinderInsights.com

Siguiendo el mismo planteamiento que nuestro proyecto, requieren que el usuario solicite paralelamente desde un flujo externo a la aplicación su copia de seguridad, sirviendo como ejemplo práctico de que el interés por la información sobre uno mismo puede ser suficiente para que un gran numero de población comparta sus datos privados de uso con una aplicación de terceros para analizarla (incluyendo sus conversaciones a través de la app).

2.2.3. Chat Visualizer

Una herramienta que permite analizar la actividad de un chat de What's App de manera estadística [13] (72K chats analizados en el momento de la redacción de esta memoria). Algunos ejemplos de sus infográficos se pueden observar en la figura 2.3.

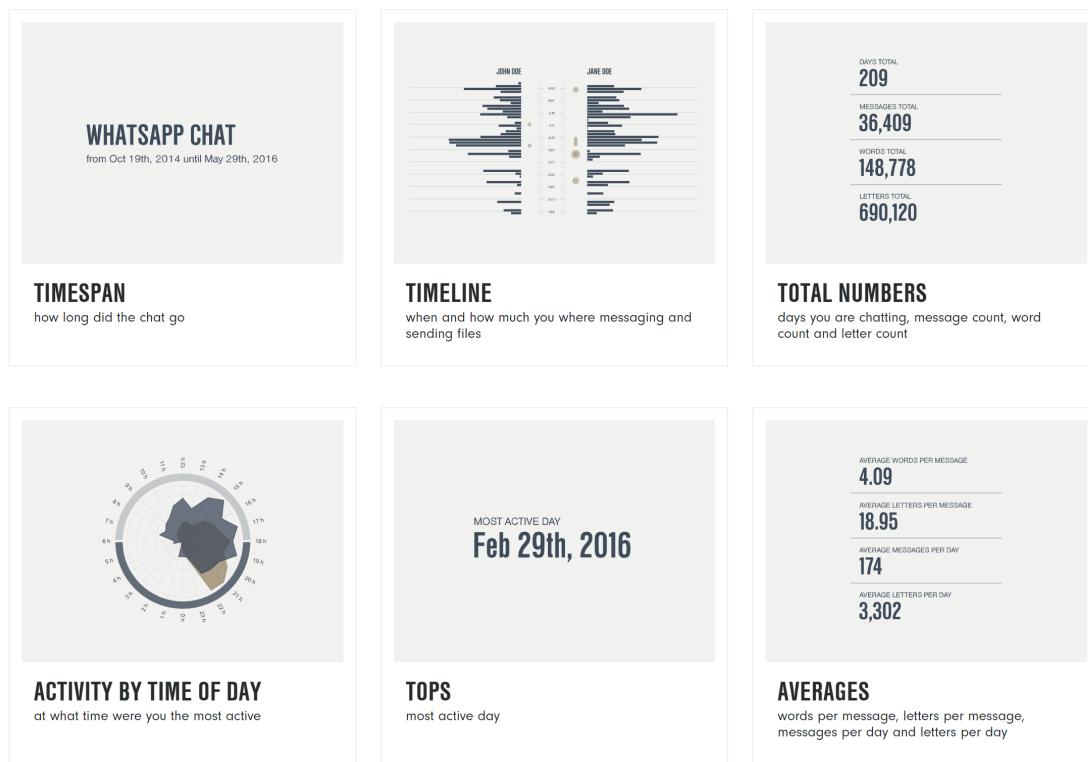


Figura 2.3: Ejemplos de pantallas de reporte de Chat Visualizer. Fuente: ChatVisualizer.com

2.2.4. Limitaciones de las herramientas analizadas

Estas herramientas existentes, aunque de gran calidad, ninguna ataja el objeto de estudio de este trabajo: La información sensible de los históricos de mensajes.

Las limitaciones que este trabajo pretende subsanar son:

- No ofrecer las conclusiones que se pueden obtener de los datos
- No es posible analizar más de una conversación en conjunto
- Otras aplicaciones de mensajería como Telegram quedan excluidas
- No garantizar la privacidad del usuario al procesar los datos en el lado del servidor

2.3. Computación en la nube

La computación en la nube [14] es la oferta de servicios de computación tales como servidores, bases de datos, redes privadas, software, herramientas de análisis, ... a través de internet, sin la necesidad de disponer de hardware físico en propiedad y beneficiándose de la economía de escala que aporta el compartir recursos de forma flexible entre un gran número de usuarios.

Actualmente hay numerosos proveedores de cloud, a continuación se esbozan las principales ventajas de aquellos con más cuota de mercado según el gráfico de la figura 2.4:

- **Amazon Web Services:** Mayor presencia en el mercado, gran número de servicios, disponibilidad global con escalabilidad y replica entre regiones, 99.99 % de disponibilidad garantizada [15].
- **Microsoft Azure:** Destacan sus servicios de aprendizaje automático e inteligencia artificial [16].
- **Google Cloud:** Como creadores de Kubernetes, es la solución mejor integrada para la utilización de contenedores de Docker en organizaciones con arquitectura de microservicios [17].
- **Digital Ocean:** Enfocados en dar servicio a desarrolladores, y no tanto a grandes organizaciones con complejos requisitos de infraestructura [18].

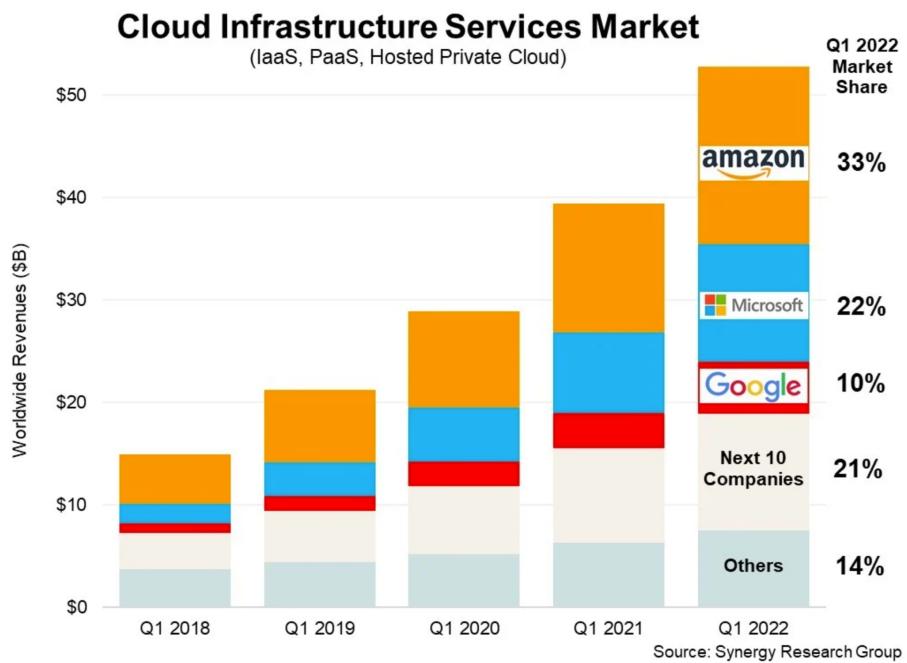


Figura 2.4: Histórico de la progresión de cuota de mercado de proveedores de cloud. Fuente: Synergy Research Group

2.4. Bases de datos

A la hora de seleccionar una base de datos para un proyecto, es importante conocer los tipos de bases de datos disponibles en función de su forma de datos y relaciones, y la forma de almacenamiento en memoria.

En función de la forma de los datos podemos diferenciar las bases de datos relacionales de las no relacionales. Donde las relacionales almacenan la información de manera estructurada, con una forma constante y una declaración explícita de las relaciones entre los diversos elementos (MySQL, Oracle, SQL Server, PostgreSQL, Redshift, ...). Y mientras que las no relacionales almacenan la información como documentos independientes, que no tienen por qué mantener la misma forma ni tener relaciones explícitas (MongoDB, Redis, Elasticsearch, Cassandra, ...).

Según cómo se almacenan los datos en memoria, podemos distinguir las bases de datos columnares de las orientadas a filas. Según la cuál las columnares almacenan de manera contigua cada columna o atributo, para acelerar el análisis estadístico de toda una tabla (Redshift, BigQuery, ...). Y las orientadas a filas almacenan en memoria contigua la información de cada entidad, permitiendo un acceso rápido a todos los datos de una misma entidad (SQLServer, PostgreSQL, ...).

2.5. Frameworks de frontend

El espectro del frontend es muy cambiante, en apenas dos o tres años las tecnologías más presentes pueden pasar a ser consideradas obsoletas, pero gracias a iniciativas como StateOfJS [19], una encuesta global sobre el ecosistema front, se puede consultar de manera muy visual el estado actual de esta rama.

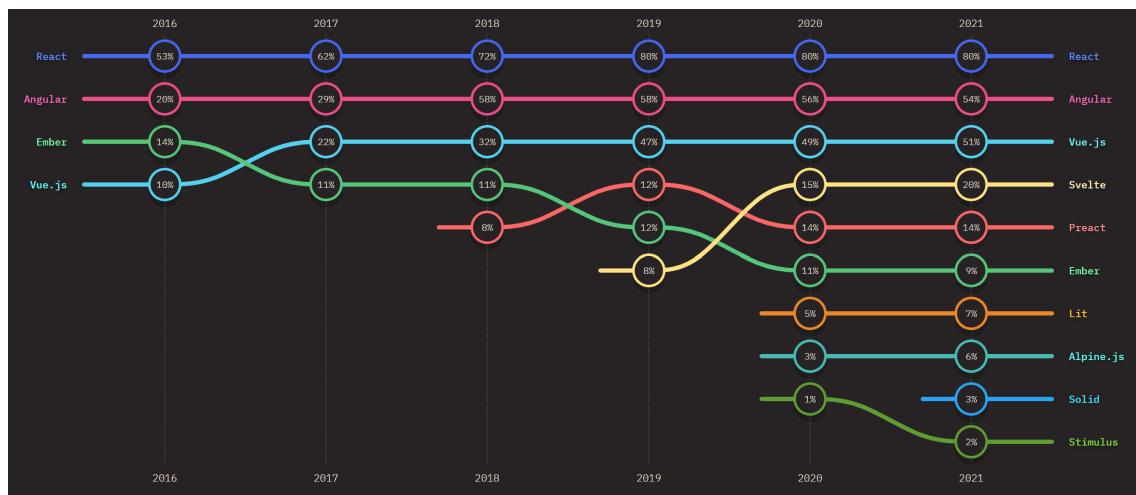


Figura 2.5: Progresión de uso de frameworks de frontend. Fuente: StateOfJS.com

Como podemos observar en la figura 2.5, Vue.JS parece ser el único framework con un crecimiento constante y consistente en los últimos cinco años, pudiendo significar un declive del resto de competidores en los próximos años en favor de este framework, destacable por su suave curva de aprendizaje y su sintaxis de HTML enriquecida, integrando bucles, condicionales y bindings de datos en las propias etiquetas nativas de HTML.

ANÁLISIS DE REQUISITOS

En esta sección se recopilan todos los requisitos tanto funcionales como no funcionales 3.3 identificados, en los que se basan las decisiones tecnológicas detalladas más adelante. Los requisitos funcionales se dividen en dos secciones, reportes individuales 3.1 y reportes globales 3.2. Siendo los primeros aquellos reportes a analizar exclusivamente en el cliente sobre la información del usuario de manera independiente, y los globales aquellas métricas comparadas con las de otros usuarios almacenadas en la base de datos del proyecto.

3.1. Reportes individuales

- **RF-01 [Presentación del proyecto]:** Presentar el proyecto de manera completa y breve, para evitar perder al usuario antes de conocer de qué trata.
- **RF-02 [Guía de exportación]:** Guiar al usuario durante el proceso de exportación de sus datos de la aplicación de mensajería externa seleccionada.
- **RF-03 [Procesar datos de Telegram]:** Permitir subir el histórico de mensajes de Telegram.
- **RF-04 [Procesar datos de What's App]:** Permitir subir el histórico de mensajes de What's App.
- **RF-05 [Emojis más usados]:** Mostrar una lista con los emojis más usados.
- **RF-06 [Uso diario]:** Desglosar la actividad diaria de la aplicación.
- **RF-07 [Horas de sueño o trabajo]:** Mostrar durante qué horas se puede deducir que el usuario duerme o trabaja.
- **RF-08 [Intereses]:** Resumir las temáticas principales de las conversaciones del usuario.
- **RF-09 [Interacciones]:** Mostrar las personas que han aparecido o desaparecido de la vida del usuario a partir de su frecuencia de mensajes.
- **RF-10 [Estado anímico]:** Simular una progresión anímica basándose en los emojis más utilizados según el periodo.
- **RF-11 [Contactos favoritos]:** Mostrar a qué contactos responde más rápido el usuario.
- **RF-12 [Contactos más sociales]:** Visualizar un ranking con las personas en las que el usuario comparte más grupos.

3.2. Reportes comparativos

- **RF-13 [Compartir estadísticas]:** Permitir seleccionar al usuario si desea compartir sus estadísticas sin comprometer los datos originales para poder generar comparativas.
- **RF-14 [Comparativa de longitud de mensajes]:** Comparar la longitud de mensaje media del usuario con el resto de la población.
- **RF-15 [Comparativa de uso]:** Comparar la cantidad de mensajes enviados por el usuario con el resto de la población.

3.3. Requisitos no funcionales

- **RNF-01 [Alta disponibilidad]:** Garantizar un tiempo de disponibilidad del 99.9 % como para que no haya usuarios que decidan no usar la aplicación por no estar disponible.
- **RNF-02 [Disponibilidad geográfica]:** Disponer de presencia del sitio web en las caches globales para permitir su acceso regular desde cualquier parte del mundo.
- **RNF-03 [Escalabilidad horizontal]:** Permitir escalar de manera virtualmente infinita tanto en datos como en procesamiento, para cubrir un posible crecimiento de popularidad del sitio.
- **RNF-04 [Intuitividad de uso]:** Ofrecer una interfaz gráfica intuitiva para permitir el acceso a cualquier tipo de usuario.
- **RNF-05 [Tiempos de respuesta]:** Que los tiempos de generación de reportes sean inferiores a 2 segundos para no interferir en el uso síncrono de la aplicación por parte del usuario.
- **RNF-06 [Interés personal]:** Mostrar reportes interesantes para que los usuarios tengan una motivación a la hora de utilizar la plataforma.
- **RNF-07 [Interés social]:** Utilizar infográficos llamativos para favorecer que se compartan por redes sociales y publicitar la herramienta.
- **RNF-08 [Mensaje de concienciación]:** Aportar información que conciente del compromiso a la intimidad que supone otorgar acceso a los históricos de chat.
- **RNF-09 [Privacidad de los datos]:** Evitar que la información sensible del usuario salga de su ordenador para garantizar la privacidad de sus datos.
- **RNF-10 [Comunidad de desarrollo]:** Simplificar la colaboración con el código fuente para la ampliación de los reportes mostrados.

DISEÑO

En esta sección se presenta el resumen del proceso de diseño de la aplicación, desde la definición de la experiencia de usuario a partir de los requisitos funcionales antes expuestos 4.1, así como la arquitectura e infraestructura necesarias para dar soporte a los requisitos técnicos no funcionales 4.2 y una descripción de la planificación de la ejecución a seguir 4.3.

Durante el diseño estrictamente visual se han utilizado principios de diseño descritos en el libro de referencia White Space Is Not Your Enemy [20], no obstante no se detallan en esta sección por considerarse fuera del ámbito técnico del proyecto.

Aunque el proceso real de diseño ha sido realizado de manera iterativa, se presenta organizado por categorías para una mayor facilidad de consulta.

4.1. Experiencia de usuario

La plataforma pretende tener el menor número de pantallas y elementos posibles para facilitar su uso, intentando reducir el número de clicks necesarios para consultar los reportes, como decisión de diseño principal a la hora de satisfacer los requisitos RNF-04 [Intuitividad de uso] y RF-01 [Presentación del proyecto].

Según uno de los principios de experiencia de usuario más citados, "3-Click rule" [21] ninguna pieza de información en un sitio web debe estar a más de 3 clicks de distancia para evitar perder al usuario en el flujo de la aplicación. Aunque dicho principio ha sido desmitificado por algunos estudios [22], en este proyecto existe ya una gran barrera de usabilidad al no disponer de ninguna forma inmediata de acceder a los datos (ni What's App ni Telegram proveen ninguna API de acceso para integrar sistemas externos a sus datos) y requerir una serie de pasos intermedios por el usuario fuera de la plataforma para obtener el histórico de datos. Por ello la interfaz se ha limitado a las cuatro siguientes secciones.

4.1.1. Presentación

En esta pantalla el objetivo principal es comunicar de la manera más concisa posible el valor que aporta la aplicación. La elección de palabras observable en la figura 4.1 intenta motivar el interés apelando a la intimidad comprometida de los usuarios. También se aplican técnicas de neuromarketing [23] a la hora de seleccionar el texto del botón principal, incitando a la acción específica por parte del usuario. Acompañada de un corto video de ejemplo donde pueden previsualizar un ejemplo del reporte generado por la aplicación, para mantener el texto lo más reducido posible y no perder el interés del usuario, ya que es comúnmente referenciado que un usuario decide si permanecer o no en una página que visita por primera vez en los primeros 10-20 segundos desde que entra [24].

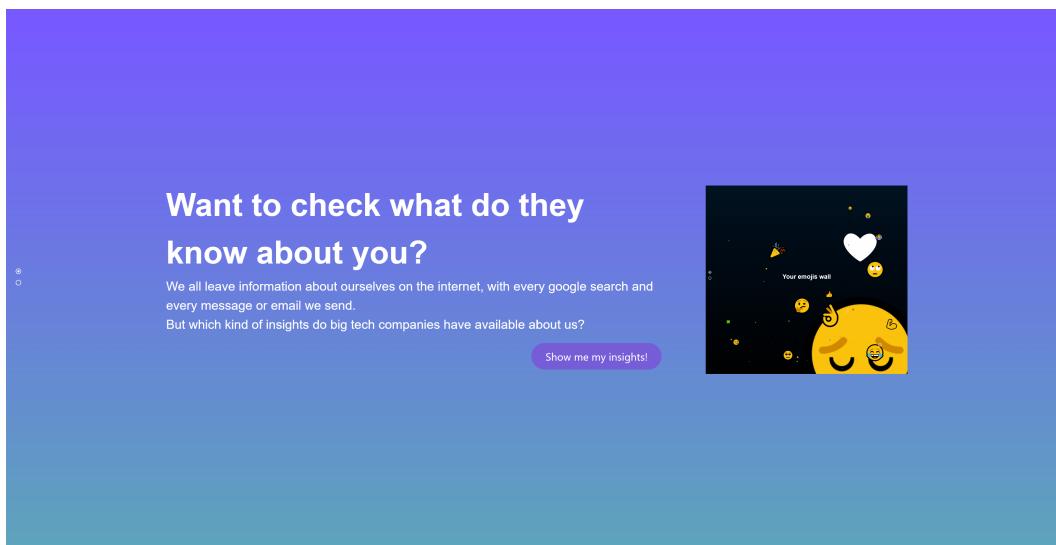


Figura 4.1: Pantalla de presentación del proyecto

4.1.2. Instrucciones de exportación

En la figura 4.2 se presentan las instrucciones para generar la copia de seguridad de manera tanto escrita como visual para acompañar lo máximo posible al usuario durante el proceso que debe realizar inevitablemente fuera de la plataforma. Se pretende así cumplir con los requisitos RF-02 [Guía de exportación] y RF-03 [Procesar datos de Telegram].

Antes de que el usuario comience con la copia de seguridad se avisa de que ninguno de sus datos saldrá de su ordenador para que la privacidad no sea un impedimento de uso y cumplir con el RNF-09 [Privacidad de los datos], la elección de palabras y acciones también pretende dejar constancia implícita de lo mismo. También se incluye una nota mencionando que los usuarios pueden acceder al código fuente disponible en GitHub, motivado tanto por intentar comenzar una comunidad de desarrollo entorno a la plataforma como se indicaba en el RNF-10 [Comunidad de desarrollo], como en un ejercicio de transparencia para transmitir más confianza respecto a la forma de procesar sus datos privados.

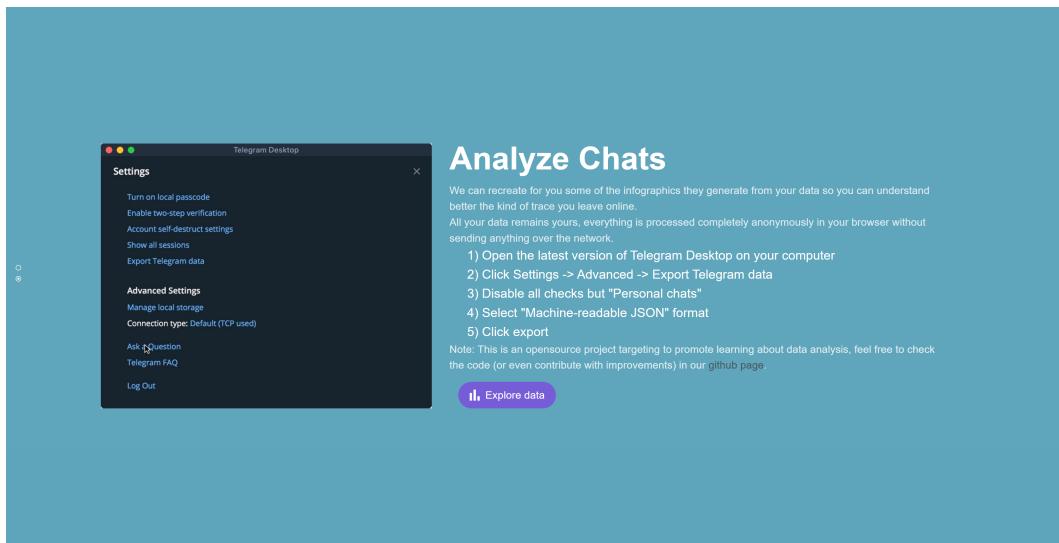


Figura 4.2: Instrucciones para exportar el histórico

4.1.3. Reportes individuales

Esta sección consta de múltiples pantallas, una por reporte generado, imitando la experiencia de usuario de Spotify Wrapped para favorecer la posibilidad de compartir la información obtenida por redes sociales, acorde con el RNF-07 [Interés social]. Algunos ejemplos de los reportes generados se pueden observar tanto en la figura 4.3 (RF-05 [Emojis más usados]) como en la figura 4.4 (RF-06 [Uso diario]).

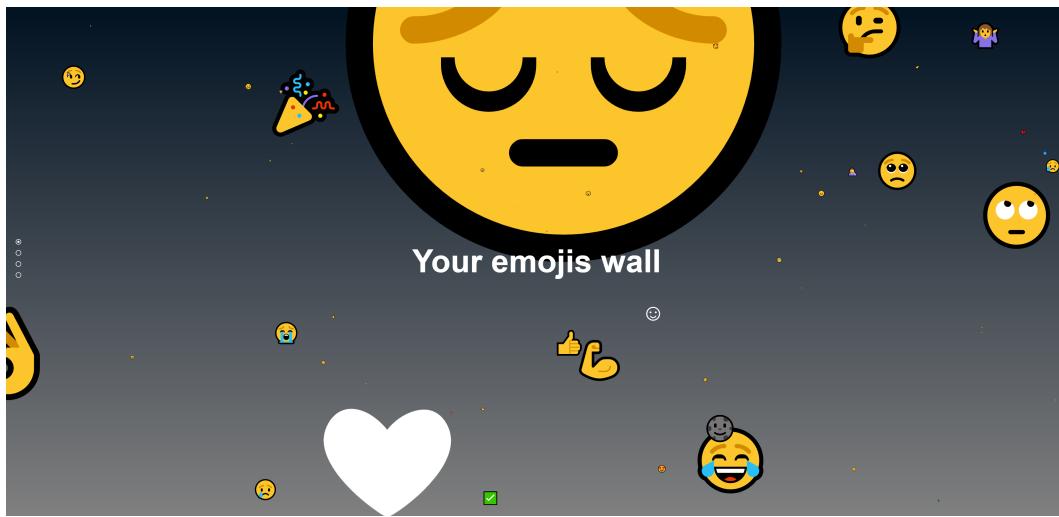


Figura 4.3: Pantalla con los emojis más usados por el usuario

Con estos reportes sobre el uso personal de la aplicación, que escapan al conocimiento que puede tener de manera intuitiva el usuario sobre sí mismo, se pretende cumplir con el RNF-06 [Interés personal], aportando información sobre el propio individuo que puede ser desconocida para él o ella. Paralelamente, la elección estética de los reportes pretende huir de mostrar la información de manera

meramente textual sino utilizando infográficos llamativos, expresivos y concisos, pretendiendo satisfacer el requisito RNF-07 [Interés social].

Detalles como la curva de uso horario permiten visualizar de manera implícita los horarios de sueño y actividad social del usuario, además se intenta aproximar un valor explícito para las horas de acostarse y retomar la actividad según lo definido en el requisito RF-07 [Horas de sueño o trabajo], esperando aportar información suficientemente íntima como para promover una concienciación con sus datos privados (RNF-08 [Mensaje de concienciación]).



Figura 4.4: Diagrama radial con el uso acumulativo por horas

4.1.4. Reporte comparativo

Para acceder al reporte comparativo primero se solicita el permiso del usuario para almacenar sus datos como aparece en la figura 4.5. Aunque no es necesario almacenar los datos del usuario para poder compararlos, se opta por esta estrategia para fomentar que comparten su reporte y mejorar la calidad del análisis global.

De esta forma se permite así al usuario poder utilizar la herramienta de análisis individual sin compartir ningún dato, y decidir si desea compartir parte de su información con fines meramente estadísticos para desbloquear la comparativa con otros usuarios, acorde con el requisito RF-13 [Compartir estadísticas].

Tras dar el consentimiento para compartir sus datos, automáticamente se desbloquea el reporte global y se permite acceder a la comparativa de la figura 4.6.

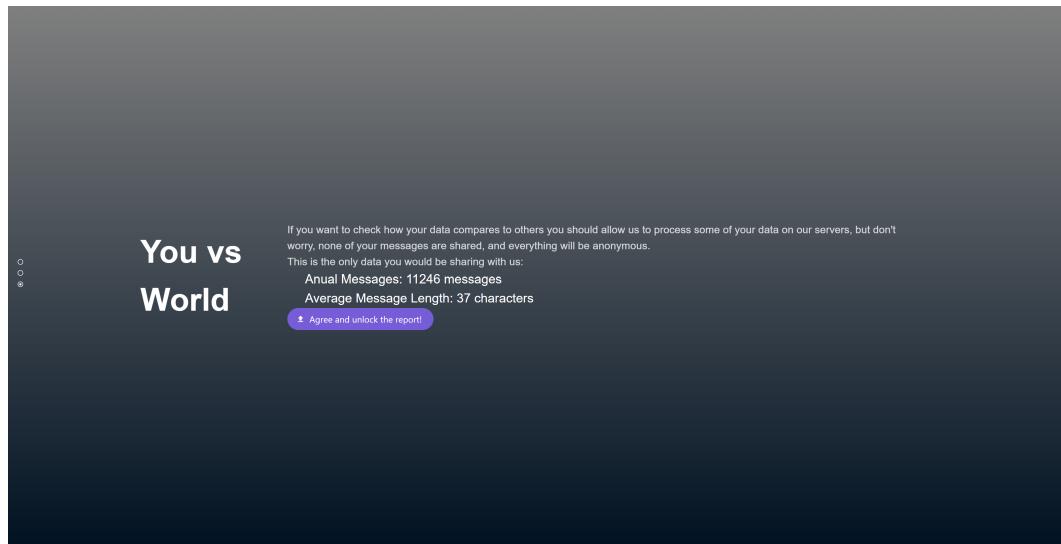


Figura 4.5: Aceptación de datos a compartir

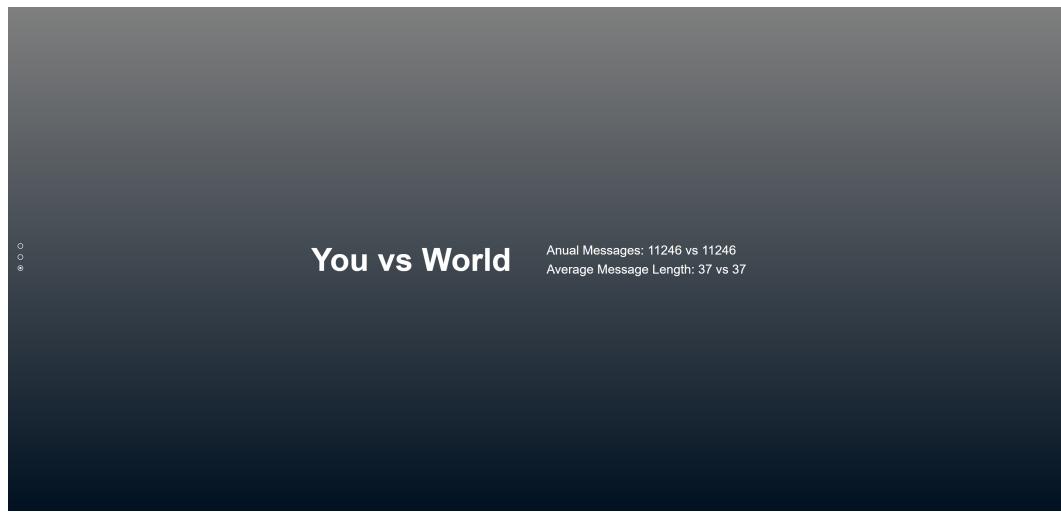


Figura 4.6: Pantalla de reporte comparativo

4.2. Infraestructura

Para dar soporte a la plataforma web y su necesidad de almacenamiento de datos, así como todos los requisitos no funcionales anteriormente detallados, se han tomado la siguiente serie de decisiones a la hora de plantear la arquitectura de servicios cuyo resultado se muestra en la figura 4.7.

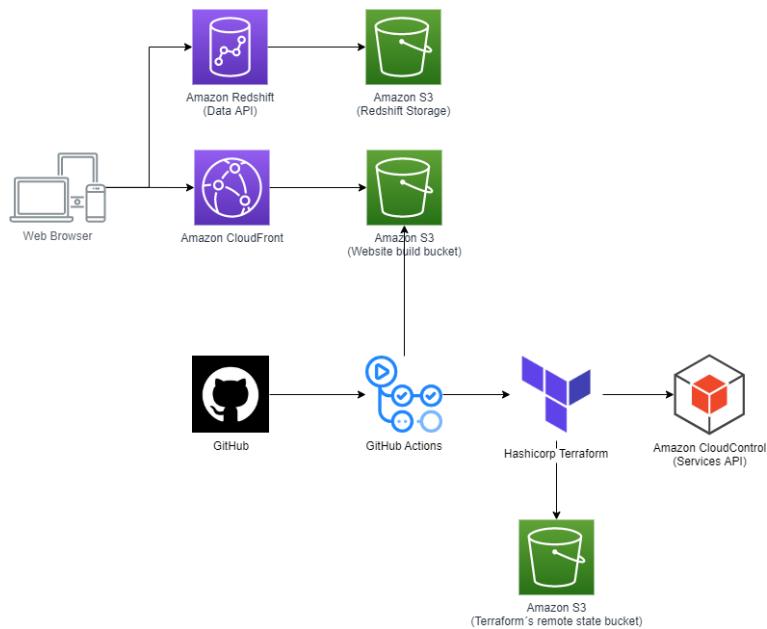


Figura 4.7: Diagrama conceptual de los servicios utilizados en la plataforma

4.2.1. Almacenamiento del código

De todas las opciones alternativas de almacenamiento de código (GitHub, BitBucket, GitLab, Beanstalk,...) se ha optado por GitHub por ser el estándar en la comunidad open source (ya que en el requisito RNF-10 [Comunidad de desarrollo] se pretende generar una comunidad de desarrollo entorno al proyecto), ofrecer servicios de integración continua como GitHub Actions o herramientas para la gestión de claves como GitHub Secrets así como por ser un servicio gratuito.

4.2.2. Proveedor de cloud

Los principales proveedores de computación en la nube, como Google o Microsoft, ofrecen servicios similares como se detalla en el capítulo anterior, no obstante se ha optado por Amazon Web Services por su transparencia a la hora de ofrecer alta disponibilidad y escalabilidad gracias a servicios como S3 (utilizado tanto para el hosting de la aplicación, como por Redshift como sistema de almacenamiento de sus datos) así como por tener una mayor cuota del mercado y seguir en crecimiento.

4.2.3. Infraestructura como código

Almacenar la definición de la infraestructura como código permite aplicar el principio de infraestructura inmutable [25] cuyo principal beneficio es la confiabilidad. Como el riesgo de fallo es inevitable, asegurar un proceso de recuperación ante desastres resulta fundamental, quedando cubierto con un versionado de la infraestructura rápidamente accesible y desplegable, al ser esta desecharable. Se opta por Terraform por su fácil integración con GitHub Actions y AWS.

Terraform funciona comparando el nuevo estado de infraestructura definido contra el anterior, y ejecutando una serie de pasos mediante las APIs a las que tiene acceso para cambiar en el entorno todo aquello que sea necesario para dejarlo como en la nueva definición del estado.

En este caso en particular, al utilizar GitHub Actions para ejecutar esos cambios, se requiere de lo que se denomina un “estado remoto”. Sin un estado remoto, Terraform genera un fichero local que no sería capaz de salvar en GitHub, por ello requiere configurar a mano un almacenamiento para el estado de manera remota. Se considera una práctica estándar en la integración entre Terraform y AWS utilizar un bucket de S3 para ello.

En el ámbito del proyecto se emplea Terraform para definir y configurar los tres recursos que se detallan a continuación (un bucket de S3 para almacenar el build de la plataforma, CloudFront como CDN y Redshift para almacenar y realizar los reportes globales).

4.2.4. Hosting de la plataforma

Para el hosting del cliente web se utiliza AWS S3 (Simple Storage Service) por su seguridad y disponibilidad de acceso RNF-01 [Alta disponibilidad], la facilidad de sincronización con GitHub Actions a la hora de realizar los despliegues y permitir una rápida configuración para la redirección de dominios al contenido web almacenado. Se utiliza en conjunto con CloudFront como servicio de CDN encargado de acelerar el acceso en sus Edge Locations (caches con réplicas de contenido distribuidas geográficamente para garantizar un rápido acceso a la plataforma en cualquier lugar del mundo RNF-02 [Disponibilidad geográfica]) y ofrecer una conexión HTTPS segura RNF-09 [Privacidad de los datos].

4.2.5. Base de datos

Basándonos en las categorizaciones expuestas en el capítulo anterior y teniendo en cuenta tanto la elección de proveedor de cloud, como la naturaleza de los datos a almacenar (un resumen de estructura fija, anonimizado y ya procesado del usuario), y que el uso de los datos almacenados va a ser realizar un reporte estadístico de cada una de las columnas de todos los usuarios en conjunto y no individualmente, la opción que más se adecúa a las necesidades del proyecto es el servicio de

AWS Redshift, por ser relacional, columnar y proveer de escalado y replicado automáticos RNF-03 [Escalabilidad horizontal].

Al funcionar encima de S3 garantiza la misma disponibilidad, escalabilidad y fiabilidad que S3, y al ser una base de datos columnar permite realizar análisis sobre conjuntos de atributos (el caso principal de uso de la aplicación) con un menor tiempo de respuesta RNF-05 [Tiempos de respuesta].

También ofrece una API para ejecutar sentencias de SQL a través del SDK de AWS, evitando así la necesidad de una capa intermedia para el backend mientras las características del proyecto no lo requieran.

De todos los servidores disponibles encargados de realizar el procesamiento de Redshift disponibles, DC2 es el seleccionado en la configuración al estar incluido gratuitamente en el tier básico de AWS [26], no obstante permite cambios si las necesidades de procesamiento aumentan.

4.2.6. Control de acceso

Todos los servicios disponibles en AWS están restringidos al acceso público por defecto, para ello es necesario configurar una serie de políticas de autenticación y autorización que permitan acceder a los usuarios tanto al contenido de la web estática generada y almacenada en S3, y distribuida por CloudFront, como para realizar consultas desde el cliente web a la Data API de Redshift.

Para ello el servicio de gestión de usuarios y acceso principal de Amazon Web Services es IAM, que se ha utilizado también para generar los accesos a los servicios de AWS para GitHub Actions y Terraform.

4.3. Planificación

Tras el diseño de la experiencia de usuario y la infraestructura que le dará soporte, se elabora un plan secuencial de hitos para el proceso de desarrollo. Pretende servir tanto de guía a la hora de afrontar el desarrollo del proyecto, como para analizar el progreso del proyecto, permitiendo aún así la flexibilidad para cambiar el orden de prioridades durante la evaluación del transcurso del mismo.

Se dividen en las siguientes historias de usuario:

- 1.– **Pantalla de presentación del proyecto:** Establecer el flujo de Desarrollo e Integración continuas entre GitHub y AWS. Permitiendo la compilación y despliegue en producción de un proyecto Vue con una pantalla estática de contenido informativo.
- 2.– **Primer reporte para Telegram:** Se añade la pantalla de guía para la exportación de los datos, el procesado de los datos de Telegram en el cliente (con una definición de estructura de datos genérica que permita importar datos de otros servicios de mensajería como What's App fácilmente en el futuro) y una primera pantalla de reporte.
- 3.– **Reporte global de una métrica:** Incluye la pantalla de aceptación de cesión de datos, registrar y configurar el servicio de Redshift para almacenar los datos cedidos por todos los usuarios, conectar el cliente a la Data API de Redshift y visualizar una métrica que compare al usuario de la aplicación con el resto de datos almacenados.
- 4.– **Ampliar reportes individuales:** Añadir el resto de pantallas con las analíticas definidas anteriormente.
- 5.– **Ampliar métricas globales:** Completar el reporte global con más métricas.
- 6.– **Dar soporte a What's App:** Incluir un paso de elección de aplicación de mensajería a analizar, que lleve a la pantalla de acompañamiento correspondiente. Traducir los datos de What's App al formato genérico definido en la aplicación y conectarlos con el pipeline de procesado de datos.

IMPLEMENTACIÓN

Con la definición de la experiencia de usuario deseada y la arquitectura conceptual diseñada, en este capítulo se profundiza en la solución de código adoptada tanto para la infraestructura en la sección 5.3 como para la parte visual en la sección 5.1 y analítica en la sección 5.2 de la plataforma, pasando por la elección de lenguajes, frameworks y librerías que permiten acelerar el desarrollo de la aplicación, en la que se ha puesto especial foco en la reutilización de código ya existente y la estrategia de negocio del bootstrapping [27].

Se puede acceder al resultado de la implementación en la url <http://wdtk-front.s3-website.eu-west-3.amazonaws.com/> así como leer el código en el repositorio <https://github.com/jmenchero/what-do-they-know/>.

5.1. Interfaz de usuario

Tras lo planteado en la sección 2.5, la selección final para el framework de frontend es Vue.JS, por su crecimiento de uso en la comunidad así como por su asequible curva de aprendizaje.

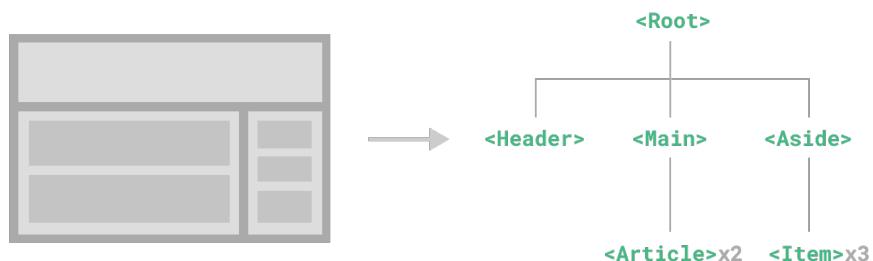


Figura 5.1: Diagrama de asociación entre organización visual y estructura de ficheros. Fuente: [VueJS.org](https://vuejs.org)

Vue.JS es un framework que extiende HTML, JavaScript y CSS [28]. Estos lenguajes se integran bajo una sintaxis propia enriquecida que permite la separación conceptual del código en componentes

con estructura, funcionalidad y estética reutilizables, atendiendo a una asociación entre componentes gráficos y ficheros como se observa en la figura 5.1. Se siguen así los principios de reusabilidad de componentes gráficos descritos en el libro Atomic Design [29]. Gracias a esta modularización se espera facilitar la contribución de código por parte de la comunidad (RNF-10 [Comunidad de desarrollo]).

```

ActionButton.vue X
front > components > atoms > ActionButton.vue > {} "ActionButton.vue"
  1  <template>
  2    <b-button
  3      class="action-button"
  4      type="is-primary"
  5      rounded
  6      @click="handleClick"
  7      >{{ text }}</b-button>
  8  </template>
  9
 10 <script>
 11 export default {
 12   name: 'ActionButton',
 13   props: ['text'],
 14   methods: {
 15     handleClick () {
 16       this.$emit('action-clicked')
 17     }
 18   }
 19 </script>
 20
 21 <style scoped>
 22   .action-button {
 23     margin: 1rem;
 24     font-size: 1vw;
 25     float: right;
 26   }
 27 </style>
 28
 29

```

Figura 5.2: Ejemplo de componente de la plataforma en Vue

En la figura 5.2 se describe un componente real de la aplicación de tipo botón. Donde se puede observar cómo el componente es definido por una plantilla de la estructura HTML con contenido inyectable (en este caso el texto a mostrar en el botón), seguido de la funcionalidad JavaScript capaz de manejar los datos a los que tiene acceso el ámbito del componente y emitir eventos a su padre, y finalizando con las clases CSS que definen la especificación estética del componente.

El framework ofrece a su vez un sistema de reactividad integrado que permite regenerar nodos HTML ante cambios de los datos almacenados, facilitando una carga dinámica del contenido. También permite introducir conceptos de programación estructurada dentro del paradigma declarativo de HTML, siendo más realista con la integración actual entre JavaScript y el lenguaje de marcado.

El framework Nuxt.JS también ha sido utilizado en la aplicación, por ser una capa por encima de Vue.JS, Node.JS y Webpack. Este framework introduce a su vez el concepto de páginas, que no son más que componentes de Vue para los que automáticamente se genera una ruta de acceso web [30].

Esto permite omitir el paso de configuración de rutas resultando en una estructura más intuitiva de la aplicación separando los conceptos de páginas y componentes.

A su vez, la ausencia de un generador de rutas dinámico permite generar una distribución estática de la web (gracias al modo de compilación Single Page Application). Se puede así prescindir de un servidor web dinámico, permitiéndonos el uso de AWS S3 como hosting estático y AWS CloudFront como CDN (en el caso de no compilar un contenido estático, no podríamos hacer uso de un CDN para garantizar el acceso global a la aplicación, sino que deberíamos proveer de réplicas del servidor web sincronizadas a lo largo del globo).

También se ha empleado la librería Fullpage.JS, que permite la navegación en formato carrusel [31], similar a la de la iniciativa de referencia Spotify Wrapped. El conjunto de estas tres tecnologías, Vue.JS para los componentes, Nuxt.JS para las páginas y Fullpage.JS para la navegación en formato carrusel, ha permitido asociar la división conceptual de las dos principales secciones de la experiencia de usuario (introducción y análisis) en dos ficheros de páginas diferentes, y cada pantalla o reporte en su propio fichero de componente aislado como se aprecia en la figura 5.3.

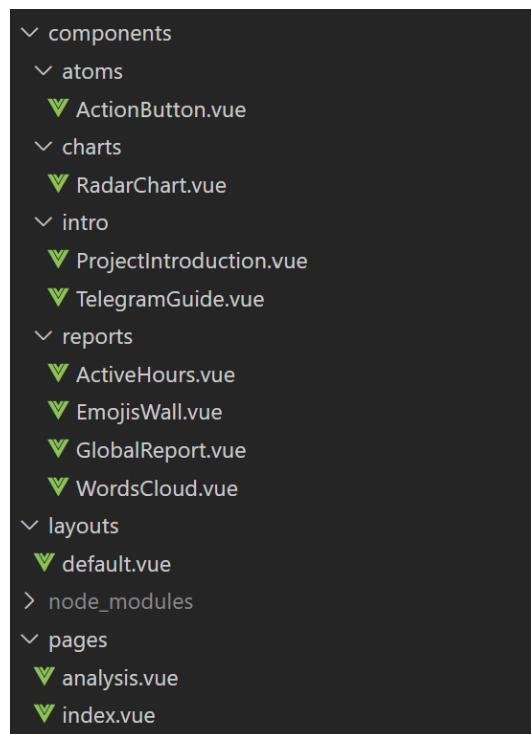


Figura 5.3: Estructura de ficheros de los componentes visuales de la plataforma

Cada componente de la lista de reportes se encarga de describir la organización de los elementos visuales que contiene su reporte, sus características estéticas y de transformar los datos, ya preparados y listos para consumir, al infográfico correspondiente.

Para ciertos elementos gráficos se emplea la librería Buefy, que provee de una serie de componentes responsive atómicos (como botones, slides, ...) listos para ser empleados en el sistema de datos

reactivos de Vue [32].

5.2. Procesado de datos

Una vez cargado el histórico de mensajes del usuario en memoria comienza el procesado de los datos con el objetivo de elaborar resúmenes de los mismos, simples e independientes, con un formato fácilmente legible por las vistas que se encargarán de mostrar sus resultados de manera visual.

Para ello existen dos módulos principales en la aplicación, por un lado encontramos la conocida como store utilizando la librería Vuex, una implementación para Vue del State Management Pattern [28]. Este patrón consiste en la aplicación de un flujo de una sola dirección en los datos. Así las acciones mutan el estado de los datos, y este es representado en la vista, pero manteniendo dicho estado en un singleton común a toda la aplicación para permitir su consulta por parte de toda la aplicación. De esta forma se asegura que solo se consultan los datos en modo lectura, y que cualquier mutación se canaliza a través de un solo punto como se detalla en la figura 5.4.

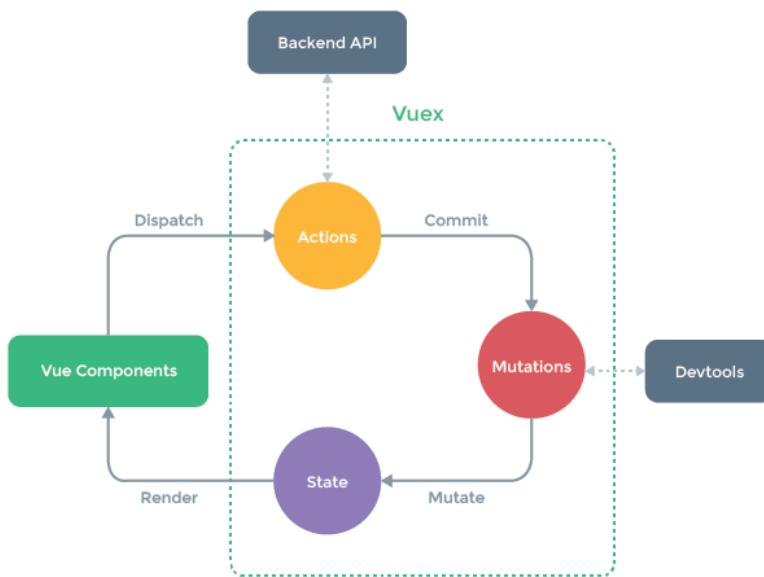


Figura 5.4: Diagrama de flujo de Vuex. Fuente: VueJS.org

En este módulo se encuentran por un lado las acciones encargadas de procesar los datos generando los reportes, y por otro el estado de la aplicación, con la información de los reportes almacenada y accesible por cualquier componente visual de la misma.

En la figura 5.5 se puede visualizar un resumen de la estructura de los datos a ingerir. Una vez en la memoria de la aplicación se provee de una serie de métodos (uno por cada reporte) que se

```
{
  personal_information: { user_id, first_name, last_name, phone_number, username, bio }
  profile_pictures: [{ date, photo }]
  contacts: [{ first_name, last_name, phone_number, date }]
  frequent_contacts: [{ id, category, type, contact_name, rating }]
  chats: [
    {
      type,
      id,
      messages: [{ id, type, date, from, from_id, text }]
    }
  ]
}
```

Figura 5.5: Esquema del histórico de mensajes de Telegram

encargan de realizar las analíticas correspondientes para posteriormente almacenarlas en la memoria persistente de la aplicación. Tras lo cual, los reportes ya están disponibles para su consulta por parte de las vistas.

Cada uno de esos métodos acepta una lista de chats, con a su vez una lista de mensajes por cada chat, como parámetro, y devuelven un objeto con el reporte en el formato esperado. De esta manera aseguramos que puedan ser compatibles con cualquier servicio de mensajería instantánea con tan solo una transformación previa de la estructura de los datos si es necesario.

Como las librerías de análisis de datos para JavaScript en cliente son limitadas, todas estas funciones analíticas están implementadas a mano para permitir un ajuste más fino de su funcionalidad y performance. Tan solo se utiliza la librería Lodash para facilitar algunas transformaciones de datos así como por su sintaxis funcional, útil para el análisis [33].

El otro módulo es el encargado de realizar las conexiones entre el cliente y la API de datos de Redshift, tanto para volcar los datos de aquellos usuarios que acepten compartirlos como para realizar las consultas analíticas correspondientes para obtener los reportes globales. En concreto, solo se almacenan en Redshift algunos datos estadísticos puntuales como la longitud media de los mensajes o la cantidad total de mensajes, para evitar almacenar cualquier dato comprometido del usuario.

5.3. Infraestructura como código

Para la definición de la infraestructura como código se han empleado dos lenguajes descriptivos. Por un lado se utiliza Terraform, cuya sintaxis permite definir una serie de recursos y servicios en AWS de manera descriptiva, entre los que se encuentran los buckets de S3, la configuración de acceso público al build, el servicio de Redshift con su API de datos y la definición del CDN de CloudFront. Y para el proceso de integración continua se emplea YAML, el lenguaje utilizado por GitHub Actions para la definición de sus procesos de integración, en el caso de este proyecto, encargado de ejecutar las actualizaciones de Terraform correspondientes, compilar la plataforma web y desplegar los cambios en el bucket de hosting cada vez que se agrega un commit a la rama de producción del repositorio.

Podemos observar los pasos siendo ejecutados en una tarea de GitHub actions en la figura 5.6.

> ✓ Set up job	4s
> ✓ Build hashicorp/terraform-github-actions/init@v0.4.0	12s
> ✓ Build hashicorp/terraform-github-actions/validate@v0.3.7	13s
> ✓ Build hashicorp/terraform-github-actions/apply@v0.4.0	10s
> ✓ Build jakejarvis/s3-sync-action@master	24s
> ✓ Checkout 🎙	3s
> ✓ Terraform Init	5s
> ✓ Terraform Validate	1s
> ✓ Terraform Apply	15s
> ✓ Setup node env 🏛	0s
> ✓ Get yarn cache directory path 🔐	0s
> ✓ Cache node_modules 📁	4s
> ✓ Install dependencies 🎨	8s
> ✓ Build static dist 🎨	26s
> ✓ Sync S3	5s
> ✓ Post Cache node_modules 📁	1s
> ✓ Complete job	0s

Figura 5.6: Acción finalizada satisfactoriamente tras un commit correcto en el repositorio

PRUEBAS

Como se espera liberar el acceso a la plataforma al público general, resulta fundamental asegurar la funcionalidad de la herramienta. Para ello se han utilizado cinco históricos reales de voluntarios y voluntarias pertenecientes a la Escuela Politécnica Superior de la Universidad Autónoma de Madrid para probar el flujo de datos y generar los reportes. Dichos informes no se comparten para conservar la privacidad de los voluntarios. También se ha empleado un histórico sintético de apenas 20 mensajes para garantizar que el flujo de datos y los cálculos sean correctos, este sí accesible en el repositorio del proyecto [34] (en el fichero `.tests/synthetic.json`). Dichas pruebas se han realizado a lo largo de la implementación tras cada etapa de desarrollo y han resultado en una serie de correcciones que se detallan en la siguiente lista de commits:

- 1.– (Feb 1, 2022 - 394b724) **Add store persistance**: Al actualizar el navegador se perdían los análisis generados.
- 2.– (Mar 23, 2022 - e43ca44) **Fix emojis wall limits**: Los emojis se salían en algunos casos de los límites de la pantalla.
- 3.– (Mar 23, 2022 - 7d0172f) **Make active hours data reactive**: El gráfico de horas no respondía a cambios en los datos tras la carga.
- 4.– (May 24, 2022 - 2105139) **Fix redshift naming requirements**: Diferentes atributos de la configuración de Redshift requerían de formatos diferentes.
- 5.– (Jun 10, 2022 - 70bf47c) **Refactor how messages are stored and analysed**: La persistencia no funcionaba debido a que se superaban los límites de almacenamiento de los navegadores.
- 6.– (Jun 10, 2022 - 3692b3d) **Fix unavailable redshift result by awaiting**: Ciertas consultas no devolvían datos porque se consultaba su resultado antes de que este estuviera disponible.
- 7.– (Jun 12, 2022 - 0b1f44a) **Fix visuals from emojis wall**: Los emojis tapaban textos y botonería en algunos casos.

Se ha descartado realizar tests unitarios en las fases de prototipado de la herramienta ya que la estructura era sujeto de cambio constante, pero estos se incluirán en futuras versiones una vez se alcance la estabilidad estructural del código, así como tests end-to-end para garantizar que la funcionalidad básica esté siempre cubierta.

CONCLUSIONES Y TRABAJO FUTURO

Este trabajo pretende sentar las bases de un proyecto de más envergadura que pueda seguir creciendo tras su defensa, por ello resulta importante recopilar en este capítulo tanto las conclusiones alcanzadas durante el desarrollo de la plataforma en la sección 7.1 como los pasos a seguir tras su presentación en la sección 7.2.

7.1. Conclusiones

Los objetivos principales de este proyecto eran concienciar al público de la información que los actuales propietarios de sus datos pueden obtener sobre su intimidad, entretenir a los visitantes del sitio con los infográficos generados y sentar las bases para una comunidad de desarrollo de código abierto.

Esto se ha conseguido mediante la implementación de una herramienta web accesible a través de internet [35] en la que se ha guiado a los usuarios en el proceso de obtención de sus datos y dado acceso a un set de infográficos con información elaborada a partir de sus históricos de aplicaciones de mensajería instantánea. A su vez la implementación modular y de código abierto, accesible a través de GitHub [34], espera generar una comunidad colaborativa una vez se publicite entre la comunidad de desarrolladores.

A su vez, este proyecto me ha permitido investigar y aplicar buenas prácticas del desarrollo y arquitectura de software, en un entorno en producción real y accesible por el público, así como trabajar en un proyecto con un proceso de integración continua definido desde el principio y con tecnologías de cloud computing como las provistas por Amazon AWS, pudiendo hacerme cargo de todas las fases de un proyecto público, desde su conceptualización, diseño de experiencia de usuario, arquitectura e implementación hasta su publicación.

7.2. Trabajo futuro

Durante la realización de este proyecto se han identificado una serie de posibles ampliaciones y mejoras que se podrán incluir en futuras versiones del mismo. Dichas mejoras se detallan a continuación:

- 1.– Mejorar las políticas de acceso IAM.
- 2.– Migrar claves de acceso a GitHub secrets.
- 3.– Contratar a un UX Designer para realizar infográficos más llamativos y proveer una estética más consistente.
- 4.– Comprar un dominio independiente para la aplicación.
- 5.– Mejorar el análisis de horas de sueño.
- 6.– Promocionar la plataforma entre la comunidad de desarrolladores.
- 7.– Tests unitarios para garantizar la calidad de los reportes tras posibles cambios.
- 8.– Tests end to end para garantizar la funcionalidad básica de la plataforma en cada despliegue.
- 9.– Asegurar funcionamiento con diferentes versiones de los datos exportados.
- 10.– RF-04 [Procesar datos de What's App]: Permitir subir el histórico de mensajes de What's App.
- 11.– RF-08 [Intereses]: Resumir las temáticas principales de las conversaciones del usuario.
- 12.– RF-09 [Interacciones]: Mostrar las personas que han aparecido o desaparecido de la vida del usuario a partir de su frecuencia de mensajes.
- 13.– RF-10 [Estado anímico]: Simular una progresión anímica basándose en los emojis más utilizados según el periodo.
- 14.– RF-11 [Contactos favoritos]: Mostrar a qué contactos responde más rápido el usuario.
- 15.– RF-12 [Contactos más sociales]: Visualizar un ranking con las personas en las que el usuario comparte más grupos.

BIBLIOGRAFÍA

- [1] B. Vuleta, "How much data is created every day?," *SeedsScientific.com*, 2021. (Visitar).
- [2] N. Natesan, "How to implement design pattern," *CastsOfSoftware.com*, 2019. (Visitar).
- [3] DOUE, "Reglamento general de protección de datos," *Diario Oficial de la Unión Europea*, 2016. (Visitar).
- [4] "Exportar datos de google." Centro de soporte de Google. Última visita en Junio de 2022. (Visitar).
- [5] "Exportar datos de facebook." Centro de ayuda de Facebook. Última visita en Junio de 2022. (Visitar).
- [6] "Exportar datos de twitter." Centro de ayuda de Twitter. Última visita en Junio de 2022. (Visitar).
- [7] "Exportar datos de spotify." Gestión de cuentas de Spotify. Última visita en Junio de 2022. (Visitar).
- [8] "Exportar datos de tinder." <https://account.gotinder.com/data>. Última visita en Junio de 2022. (Visitar).
- [9] "Exportar datos de whatsapp." FAQ de What's App. Última visita en Junio de 2022. (Visitar).
- [10] "Exportar datos de telegram." <https://telegram.org/blog/export-and-more>. Última visita en Junio de 2022. (Visitar).
- [11] "Spotify wrapped." https://en.wikipedia.org/wiki/Spotify_Wrapped. Última visita en Junio de 2022. (Visitar).
- [12] "Tinder insights." <https://tinderinsights.com/>. Última visita en Junio de 2022. (Visitar).
- [13] "Chat visualizer." <https://chatvisualizer.com/>. Última visita en Junio de 2022. (Visitar).
- [14] M. J. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, Paas, and Iaas)*. USA: John Wiley and Sons Inc, 1st ed., 2014.
- [15] "Amazon web services." <https://aws.amazon.com/>. Última visita en Junio de 2022. (Visitar).
- [16] "Microsoft azure." <https://azure.microsoft.com/>. Última visita en Junio de 2022. (Visitar).
- [17] "Google cloud." <https://cloud.google.com/>. Última visita en Junio de 2022. (Visitar).
- [18] "Digital ocean." <https://www.digitalocean.com/>. Última visita en Junio de 2022. (Visitar).
- [19] "State of js." <https://stateofjs.com/>. Última visita en Junio de 2022. (Visitar).
- [20] K. Golombisky and R. Hagen, *White Space Is Not Your Enemy*. USA: A. K. Peters, Ltd., 3rd ed., 2016.
- [21] B. Vuleta, "Three-click rule," *SeedsScientific.com*, 2021. (Visitar).
- [22] B. Vuleta, "Three-click rule myth," *SeedsScientific.com*, 2021. (Visitar).
- [23] M. Grief, "9 ways to make your e-commerce call to action irresistible," *SiteTuners.com*, 2019. (Visitar).
- [24] J. Nielsen, "How long do users stay on web pages?," *NNGroup.com*, 2011. (Visitar).
- [25] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "Devops: Introducing infrastructure-as-code," in *2017 IEEE/ACM 39th International Conference on Software Engineering*

- ring Companion (ICSE-C)*, pp. 497–498, 2017.
- [26] B. Vuleta, “Redshift pricing,” *SeedsScientific.com*, 2021. (Visitar).
- [27] R. T. Harrison, C. M. Mason, and P. Girling, “Financial bootstrapping and venture development in the software industry,” *Entrepreneurship and Regional Development*, 2004.
- [28] “Vuejs.” <https://vuejs.org/>. Última visita en Junio de 2022. (Visitar).
- [29] B. Frost, *Atomic Design*. USA: Brad Frost, 1st ed., 2016.
- [30] “Nuxtjs.” <https://nuxtjs.org/>. Última visita en Junio de 2022. (Visitar).
- [31] “Fullpagejs.” <https://alvarotrigo.com/fullPage/es/>. Última visita en Junio de 2022. (Visitar).
- [32] “Buefy.” <https://buefy.org/>. Última visita en Junio de 2022. (Visitar).
- [33] “Lodash.” <https://lodash.com/>. Última visita en Junio de 2022. (Visitar).
- [34] “Repositorio del proyecto en github.” <https://github.com/jmenchero/what-do-they-know/>. Última visita en Junio de 2022. (Visitar).
- [35] “Acceso al proyecto.” <http://wdtk-front.s3-website.eu-west-3.amazonaws.com/>. Última visita en Junio de 2022. (Visitar).



Universidad Autónoma
de Madrid