Universidad Internacional de Valencia

Implementación de una API para optimizar el consumo de datos en Innguma

Titulación: Máster Universitario en Desarrollo de Aplicaciones y Servicios Web

Alumno/a: Luengo Echeverria, Daniel
DNI: 72277358L

Convocatoria: Segunda

Curso Académico: 2022-2023

Tutor/a del TFM: Eneko Arza Landaluce

Co-Director/a del TFM: Rubén Pérez Ibáñez

De:
Planeta Formación y Universidades

Universidad Internacional de Valencia

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En el siguiente documento, se describe el Trabajo de Fin de Máster titulado como 'Implementación de una API para optimizar el consumo de datos en Innguma' el cual ha sido realizado en la empresa Innguma[1]. El proyecto está completamente ligado a las asignaturas realizadas en el Máster Universitario en Desarrollo de Aplicaciones y Servicios Web. Por lo que, aunque no sea un tema completamente personal ni un tema propuesto por el propio máster, es un trabajo adecuado, que cumple con los requisitos especificados en el máster.

A continuación, en la sección 1.1, se explicará el contexto del proyecto. Por otro lado, en el apartado 1.2, se explicará a qué se dedica la empresa Innguma para entender las necesidades que se van a cumplir con la realización del proyecto. Y por último, se expondrá cuales son las motivaciones del proyecto, en la sección 1.3.

1.1. Contexto

El desarrollo del proyecto se ha realizado junto la empresa Innguma. Esta empresa está especializada en la inteligencia competitiva y en la vigilancia tecnológica, lo cual permite estar constantemente informado sobre todo lo que puede llegar a ser interés para sus cliente.

La empresa ya mencionada, mediante el proyecto 'Implementación de una API para optimizar el consumo de datos en Innguma', ha tratado mejorar el uso de los datos para que sean consumidos de una manera más eficiente y útil. De esta manera, los clientes y usuarios podrán crear informes de una manera sencilla haciendo uso de herramientas de 'Business Intelligence (BI)', ya que podrán crear peticiones para recoger los datos que ofrece la API creada. Por otro lado, también se ha implementado un nuevo sistema de votaciones, para las noticias publicadas dentro de la aplicación de Innguma.

1.2. Innguma

Innguma es una spin-off creada por el centro tecnológico Ideko, que a su vez forma parte de Danobatgroup. La empresa nació debido a la necesidad de crear una aplicación basada en la inteligencia competitiva, la cual permitiría al cliente estar informado de todo aquello que

[1] `https://www.innguma.com/`

resulte de interes para su negocio. La versión inicial se creó en Ideko, una versión simple con la que se pudieron cumplir las necesidades de esta propia empresa.

Con dicha aplicación, la empresa en cuestión puede realizar una vigilancia tecnológica con la que se puede obtener información de diferentes fuentes. Estas son personalizables por el propio usuario, pudiendo especificar cual es la información relevante y útil. Gracias al hecho de estar informado sobre todo lo que te rodea, a la hora de tomar decisiones importantes, que puede incluso llegar a determinar el rumbo de la empresa, el simple hecho de respaldar tus decisiones en información verídica, hace que la decisión que se tome, sea más posible de ser la correcta.

Debido a la gran utilidad de la aplicación, se decidió empezar a comercializarla en el mercado. Al ver que el producto se vendía bien e Ideko ya no podía hacerse cargo del mantenimiento y del desarrollo de la aplicación; es entonces cuando se creó Innguma como empresa. Y esta sería la que se encargaría de los nuevos desarrollos y del mantenimiento de la aplicación.

Actualmente Innguma cuenta con un pequeño equipo de trabajo. Entre las que se pueden encontrar, desde el equipo de marketing, encargados de la distribución del producto, de la comunicación, entre otras cosas; hasta desarrolladores de software, que serían los encargados de actualizar y mantener el programa.

Aun teniendo un equipo reducido, cuenta con una gran cantidad de clientes, a nivel nacional, e incluso a nivel internacional con varios clientes en Latinoamérica. Debido a que la aplicación es muy versátil y útil, la variedad de clientes es amplia, en la que se puede encontrar algunos como Kutxabank, Orona, Soraluce, etc.

1.3. Motivación del proyecto

Innguma actualmente cuenta con una sección de reportes donde los usuarios pueden consultar diferentes datos sobre los analistas, lectores, items, categorías y etiquetas; datos que se encuentran dentro de la propia aplicación. Pero estos datos no están bien estructurados, por lo que no se recogen eficientemente. Por lo que hoy por hoy, no hay una manera de que los usuarios visualicen los datos de una manera coherente y organizada.

Por este motivo, hay tres motivaciones principales para el desarrollo de este proyecto. Por un lado, se ha visto la necesidad de recoger el feedback de los lectores sobre la noticias que se publican en Innguma. Es por ello que se quiere crear un nuevo sistema de votaciones dentro de la aplicación de Innguma, que haga uso de esta nueva arquitectura de tablas, donde los usuarios de la aplicación podrán realizar votaciones a las noticias.

Por otro lado, se quiere crear una API capaz de consumir todos los datos accesibles de la aplicación de Innguma de una manera más eficiente. Esto implica tener que reorganizar las tablas de la base de datos para que los datos tengan coherencia y relación entre ellos. Los datos de las votaciones que se ha comentado, también hará uso de estas tablas y podrán ser consumidos mediante el uso de la API.

Por último, el hecho de haber creado una API posibilita a empresas terceras hacer uso de esta para poder crear informes de una manera sencilla mediante el uso de herramientas de Business Intelligence, como pueden ser Power BI o Data Studio.

También comentar que como objetivo secundario, el hecho de haber reestructurado las tablas y al haber creado una API, facilita desarrollos futuros dentro de Innguma, ya que se ha facilitado el acceso a estos datos, haciendo que el consumo de estos sea más rápido y eficaz.

Capítulo 2

Objetivos del proyecto

Antes de empezar con el desarrollo de un proyecto es imprescindible definir cuáles van a ser los objetivos que se quieren cumplir, ya que proporciona una dirección clara en la que el proyecto debe desarrollarse y también sirven como criterio de éxito para el producto final. Es cierto que existen varios tipos de objetivos, pero en el caso del proyecto denominado como “Implementación de una API para optimizar el consumo de datos en Innguma” se han definido dos tipos de objetivos, los técnicos (sección 2.1) y los de gestión (sección 2.2), por que son los objetivos más comunes dentro de un proyecto tecnológico.

2.1. Objetivos técnicos

Los objetivos técnicos se refieren a los logros y resultados específicos que se desean alcanzar en términos de aspectos técnicos, tecnológicos o de desarrollo. Estos objetivos están relacionados con la implementación y entrega exitosa de soluciones técnicas y productos finales. Los objetivos técnicos identificados son los siguientes:

- Diseñar y crear la arquitectura de nuevas tablas de la Base de Datos
- Desarrollo de la aplicación de Innguma en la parte del servidor para la recolección de datos mediante el uso PHP y Joomla
- Implementación de un nuevo sistema de votación para las noticas publicadas en Innguma, desarrollando la lógica en el servidor y en el cliente; y el diseño del Front-End
- Diseñar y desplegar una API para el consumo de datos mediante Flask
- Uso de plataformas de Business Intelligence para la creación de reportes

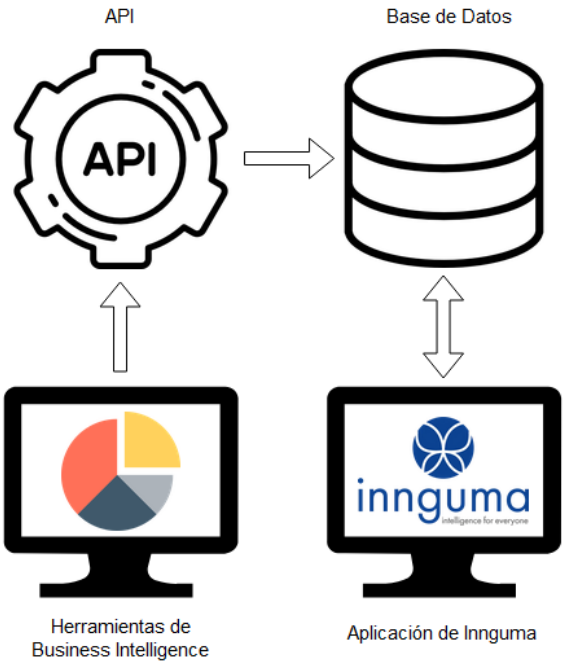

Figura 2.1: Arquitectura del sistema Fuente: Elaboración propia

2.2. Objetivos de gestión

Los objetivos de gestión están relacionados con la planificación, coordinación y supervisión de las actividades del proyecto, así como con el logro de los resultados deseados dentro de los plazos y presupuestos establecidos. Los objetivos de gestión identificados son los siguientes:

- Evaluar y planificar las funcionalidades y tecnologías para el desarrollo de la implementación de la API, la implementación de las tablas de la Base de Datos, la programación de la parte del cliente y del servidor, y finalmente, la creación de informes mediante herramientas de Business Intelligence
- Ejecutar la planificación, entre las que se encuentran el desarrollo de las funcionalidades especificadas
- Guardar y compartir el código con los demás trabajadores de la empresa para que pueda ser revisable por todos mediante el uso de GitLab
- Realizar una documentación del código y del proyecto, para facilitar a un futuro desarrollador la posibilidad de editar el código de una manera más sencilla y entendible, para que pueda mantener y actualizar el estado de la funcionalidad desarrollada
- Hacer uso de un diagrama de Gantt para la gestión del proyecto mediante la politicas establecidas por la metodología predictiva

Capítulo 3

Marco tecnológico

El marco tecnológico consiste en explicar cuáles son las tecnologías, las herramientas y las plataformas que se van a utilizar en el transcurso del desarrollo del proyecto. Cabe destacar que al haber realizado el proyecto en la empresa Innguma, gran parte del marco tecnológico ya viene predefinido, es decir, que se ha de utilizar las tecnologías que ya se están usando dentro de la propia empresa.

Las tecnologías que se van a comentar son: el gestor de contenidos que se va a utilizar, en la sección 3.1. Cuál es el tipo de Base de Datos que se va a usar, en el apartado 3.2. Que tecnología se ha seleccionado para poder levantar un entorno de desarrollo para poder programar con una versión local de la aplicación en la sección 3.3. El software que se va a utilizar para el control de versiones del código en el apartado 3.4. Cuales son los lenguajes de programación con los que se va a escribir el código durante el transcurso del proyecto en el punto 3.6 del documento. Que tecnologías se van a utilizar para la creación de la propia API en la sección 3.7. Y finalmente, el software que se va a usar para la creación de reportes en el apartado 3.8:

3.1. Gestor de contenidos

Se ha hecho uso del sistema de gestión de contenidos llamado **Joomla**, el cual permite desarrollar sitios webs de gran relevancia debido a la posibilidad de poder crear sitios webs interactivos y dinámicos. Por otro lado, te permite crear, modificar y eliminar fácilmente el contenido que se muestra en la web, ya que te ofrece un panel de administración el cual permite configurar una gran cantidad de diferentes opciones (Webempresa, s.f.).

A su vez, Joomla cuenta con una gran comunidad, por lo que se puede encontrar varios foros. En estos, muchos usuarios exponen sus dudas y problemas. Y a consecuencia de ello, otros desarrolladores ayudan a estos a dar una solución a las preguntas.

Otro gestor de contenidos es, WordPress, una herramienta que al igual de Joomla sirve para crear y gestionar la información de una página web. Es cierto que WordPress es más popular, llegando a ser líder a nivel mundial. Sin embargo, el equipo de Innguma en su día decidió decantarse en crear su propia aplicación usando Joomla, debido a su gran versatilidad y porque ofrecía solución a las necesidades que tenía Innguma. Por otro lado, WordPress está

más enfocado para usuarios sin tanta experiencia en la programación, ya que tiene un uso más sencillo, pero no llega a ser tan personalizable, a diferencia de Joomla.

Joomla	**WordPress**
Sistema de gestión de contenidos de propósito general	Sistema de gestión de contenidos de blogs y sitios web
Requiere un conocimiento más avanzado	Fácil de usar y aprender
Altamente personalizable con extensiones	Menos personalizable, a través de temas y plugins
Seguridad con actualizaciones regulares	Requiere atención constante a las actualizaciones de seguridad

Cuadro 3.1: Características principales de Joomla Y WordPress

3.2. Base de datos

Una Base de Datos es la encargada de almacenar todos los datos y de conectarlos entre sí en una unidad lógica. Los desarrolladores tienen la posibilidad de manipular los datos como más les convenga. Pudiendo añadir, borrar o modificarlos a placer (IONOS, s.f.).

Joomla ofrece la posibilidad de conectar la página web con diferentes bases de datos, como puede ser MySQL, SQL Server o PostgreSQL. Los tres tienen sus ventajas y sus desventajas. Por ejemplo, PostgreSQL es recomendable para sistemas que requieran de queries más complejas, pero esta tiene una dificultad mayor. MySQL es idóneo para los proyectos basados en web que utilizan las Bases de Datos para guardar y manipular los datos. Estos dos tipos anteriormente comentados son de código abierto, es decir, son gratis. A diferencia de SQL Server que es de pago.

MySQL	SQL Server	PostgreSQL
Sistema de gestión de bases de datos relacional de código abierto	Sistema de gestión de bases de datos propietario desarrollado por Microsoft	Sistema de gestión de bases de datos relacional de código abierto
Ampliamente utilizado en aplicaciones web y de código abierto	Comúnmente utilizado en entornos empresariales y de Windows	Popular en aplicaciones web y de código abierto
Soporte para múltiples plataformas, incluyendo Windows, Linux y macOS	Principalmente compatible con plataformas Windowsx	3. Compatible con diversas plataformas, incluyendo Windows, Linux y macOS
Características de rendimiento y escalabilidad destacadas para aplicaciones de lectura intensiva	Ofrece características avanzadas de seguridad y gestión en entornos empresariales	Conocido por su robustez y características avanzadas de extensibilidad

Cuadro 3.2: Características principales de MySQL, SQL Server y PostgreSQL

En el figura 3.1 podemos observar la diferencia de popularidad entre los tres tipos de Bases de Datos. Donde se puede observar que MySQL ha sido la más popular en los últimos cinco años.

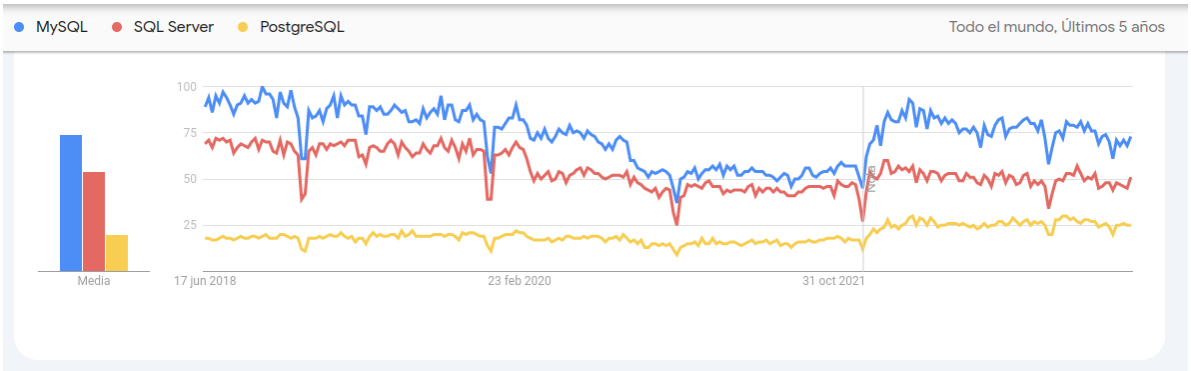

Figura 3.1: Comparación de búsquedas sobre las Bases de Datos Fuente: Elaboración propia

Teniendo en cuenta que MySQL es el más recomendado para los proyectos basados en web, como es el caso de la aplicación de Innguma, el equipo decidió utilizar **MySQL**. Además de ser la más popular y con la que más soporte cuenta.

Por otro lado, como software de administración de la Base de Datos, se ha decidido utilizar **DBeaver**. Esta selección ha sido personal debido a estar ya familiarizado con el mismo. Existe una gran variedad de estas aplicaciones y cada una tiene sus ventajas y desventajas. Pero prácticamente tienen las mismas funciones principales, ya sea la conexión a la base de datos; la creación, la edición y la eliminación de los datos; y la ejecución de queries. La aplicación de Innguma también cuenta con su propia herramienta de administración de Base de Datos, phpMyAdmin. Aunque no tenga tantas funcionalidades como DBeaver, también es útil utilizarla cuando se requiere hacer trabajos sencillos, como puede ser simplemente consultar los datos de alguna tabla.

3.3. Entorno de desarrollo

Un entorno de desarrollo es un espacio de trabajo que permite a los desarrolladores crear una aplicación o realizar cambios en ella sin afectar a la versión real de la aplicación. Estos cambios pueden incluir el mantenimiento, la depuración, la aplicación de parches o el propio desarrollo de la aplicación (Diego, 2023).

El entorno de desarrollo se ha creado mediante el uso de **Docker** (Anabel, 2023). Una plataforma de virtualización ligera que permite empaquetar una aplicación y todas sus dependencias en un contenedor, lo que garantiza la portabilidad y reproducibilidad del entorno de desarrollo.

De esta manera se puede levantar y parar el entorno con una gran facilidad. Ayudando a cualquier desarrollador a crear como una especie de copia de la versión de la aplicación que está en producción, para poder realizar cualquier cambio, sin interferir en el funcionamiento de la aplicación real.

Existen otras tecnologías equivalentes a Docker, como puede ser Kubernetes. Esta plataforma también sirve para administrar contenedores. Pero por la misma razón por la que se han elegido otras tecnologías, Innguma previamente hacia uso de Docker, por lo que la selección ya estaba predefinida.

3.4. Control de versiones

A la hora de realizar un desarrollo que implique programación y la edición de archivos, es altamente recomendable hacer uso de una herramienta de control de versiones. Un sistema que registra los cambios realizados en un conjunto de archivos a lo largo del tiempo, permitiendo mantener un historial de modificaciones, gestionar ramas de desarrollo, fusionar cambios y colaborar de manera efectiva en proyectos de software.

Para ello, se ha hecho uso de **Git**, el sistema de control de versiones más utilizado a nivel mundial por su velocidad, eficiencia y flexibilidad (Atlassian, s.f.). Por otro lado, se ha utilizado **GitLab**, una plataforma web que proporciona un conjunto completo de herramientas para el desarrollo de software basado en Git. Es una solución de alojamiento de repositorios Git (Walsh, 2023).

Gracias a estas tecnologías mencionadas el desarrollo del código es más sencillo y ordenado. Facilitando enormemente al desarrollador poder ir implementando diferentes funcionalidades teniendo un registro de estas.

3.5. Gestión del proyecto

La gestión de un proyecto es de vital importancia, ya que puede tener un impacto significativo en el éxito y el resultado final del proyecto. Una gestión correcta aumenta significativamente cumplir con los objetivos establecidos,permitiendo tener un mayor control de los recursos. Por otro lado, nos permite seguir la planificación establecida para tener el rumbo del desarrollo más claro, de esta manera la organización de dicho proyecto va a ser más óptima y eficaz.

Es por esto que es altamente recomendable usar algún tipo de metodología para poder asegurar el correcto desarrollo del proyecto. Hay dos tipos de metodologías que se han tenido en cuenta a la hora de hacer una elección, y son:

- Metodología predictiva: Se basa en una planificación exhaustiva y un enfoque secuencial. Donde el proyecto se separa en diferentes fases que se ejecutan de una manera lineal y secuencial. Los requisitos y los objetivos se definen desde inicio y es complicado introducir cambios en medio del proceso, ya que supone una modificación de las fases (Donetonic, s.f.).
- Metodología ágil: Se centran en la flexibilidad, la entrega iterativa y la colaboración continua con el cliente. Existen varios tipos como puede ser Scrum o Kanban. No es necesario definir todos los requisitos y objetivos al inicio del proyecto, ya que se trabaja en iteraciones. Fomenta la interacción entre los miembros y el cliente, ya que quiere garantizar el ajuste a las necesidades cambiantes (Donetonic, s.f.).

Viendo que este proyecto es un trabajo individual y no hay un cliente que especifique los objetivos y requisitos que desea. La **metodología** más adecuada es la **predictiva**. Sabiendo esto, se ha decidido hacer uso de un **diagrama de Gantt** (Se puede observar el diagrama en las figuras 4.1 y 4.2), una herramienta de gestión de proyectos que se utiliza para visualizar y planificar las tareas y actividades de un proyecto en un tiempo determinado. Este tipo de diagrama se representa gráficamente como un gráfico de barras en el que cada barra representa una tarea y su duración, definiendo una fecha de inicio y final.

Existen varios programas y aplicaciones que permiten crear diagramas de Gantt; como pueden ser Tom's Planner, Microsoft Project o Trello. Aun así se ha decidido hacer uso de una hoja de cálculo de Google Drive, ya que es la manera más personalizable que puede adecuarse a las necesidades del proyecto.

3.6. Lenguajes de programación

Primero, es importante destacar que la aplicación de Innguma está dividida en dos partes de código, una sería la parte del cliente, en la que se usarán los tres lenguajes predominantes de stack básico, que son **HTML, CSS y JavaScript**. Con el HTML se ha definido la propia

estructura de la página, con CSS el diseño de esta; y con JavaScript, todas las interacciones que el usuario puede realizar con los diferentes elementos de la pantalla.También comentar que se ha hecho uso de varias librerías. Entre otras cosas, para facilitar la creación de las vistas dándoles un toque más profesional; para ello, se hará uso de la librería Bootstrap, la cual también nos ayuda a crear un diseño responsivo. Es decir, que se ajustará correctamente a diferentes tamaños de pantalla.

Y por otro lado, se encontraría la parte del servidor. Como ya se ha comentado, se ha hecho uso del gestor de contenidos de Joomla. Para el desarrollo de esta se ha de utilizar el lenguaje de programación de **PHP**, un lenguaje de código abierto diseñado específicamente para el desarrollo web. Siendo uno de los lenguajes de programación más populares para crear aplicaciones web dinámicas y sitios web interactivos.

Y por último, para el desarrollo de la aplicación API, la cual se desarrollará mediante Flask. Un framework que permite crear aplicaciones web de una manera rápida y sencilla haciendo uso del lenguaje de programación de **Python**.

3.7. Creación de la API

API es el acrónimo en inglés de “interfaz de programación de aplicaciones”, un software intermedio que permite que los programas se comuniquen entre sí. En este proyecto, será el intermediario entre la aplicación o los clientes con la Base de Datos de Innguma. De esta manera, se podrá recoger los datos necesarios de una manera sencilla, solamente realizando una llamada HTTP al endpoint correspondiente con los parámetros correctos (MulesSoft, s.f.).

Para elegir las tecnologías que se han utilizado para la creación de la API, se ha hecho un estudio de los diferentes métodos disponibles en el mercado. Algunas arquitecturas que se han considerado son las siguientes:

3.7.1. Flask, Serverless, API Gateway y AWS Lambda

Consiste en crear una API mediante un desarrollo local usando el framework de Flask. De esta manera se podrá hacer el testeo de esta de una manera más fácil. Para hacer el despliegue de la API en los servicios de Amazon, se usará el framework de Serverless. Este automatizará el despliegue en AWS, creando los endpoint necesarios en API Gateway y creando sus respectivas funciones lambdas con el código necesario.

Ventajas	Desventajas
Escalabilidad	Complejidad de configuración de la API
Flexibilidad en el desarrollo	Los costes dependen del tráfico y las cuotas de Amazon
Gestión de tráfico y seguridad	Despliegue algo más complejo (comparado con Zappa)
Mayor flexibilidad en el despliegue	

Cuadro 3.3: Ventajas y desventajas de la plataforma de Flask, Serverless, API Gateway y AWS Lambda

3.7.2. Flask, Zappa, API Gateway y AWS Lambda

Es una solución muy parecida a la opción anterior, con la diferencia de que en el momento de hacer el despliegue de la API a los servicios de AWS, se hace uso de Zappa en vez de Serverless. Zappa es un framework que está enfocado en hacer despliegues solamente en AWS, facilitando el proceso. Pero serverless es algo más avanzado ya que ofrece la posibilidad de hacer el despliegue en diferentes servicios, sin estar limitado a Amazon, de esta forma, también ofrece más posibilidades.

Ventajas	Desventajas
Escalabilidad	Complejidad de configuración de la API
Flexibilidad en el desarrollo	Los costes dependen del tráfico y las cuotas de Amazon
Gestión de tráfico y seguridad	Solamente se puede desplegar en AWS
Despligue sencillo en AWS	

Cuadro 3.4: Ventajas y desventajas de la plataforma de Flask, Zappa, API Gateway y AWS Lambda

3.7.3. API Gateway y AWS Lambda

Este método consiste en hacer solamente uso de los servicios de AWS API Gateway y AWS Lambda para hacer el desarrollo de la propia API. Ahorrando el trabajo de desarrollar la aplicación API en local y el despliegue.

Ventajas	Desventajas
No se usa frameworks de terceros, todo se hace en AWS	Los endpoint hay que crearlos uno a uno en API gateway y vincularlos con las funciones Lambda
No hace falta hacer un despliegue de la versión local	Las funciones Lambda también hay que crearlas una a una
Gestión de tráfico y seguridad	Limitaciones al realizar el testeo de una versión de prueba
	Escalabilidad muy pobre
	Poca flexibilidad

Cuadro 3.5: Ventajas y desventajas de la plataforma de API Gateway y AWS Lambda

3.7.4. Selección

Habiendo hecho un estudio de los diferentes métodos posibles, se ha decidido hacer uso del primer método explicado: Flask, Serverless, API Gateway y AWS Lambda ya que se ofrece una opción más profesional, ayudando a crear una aplicación API completa, flexible y escalable.

Otra opción que se ha barajado es hacer uso de FastAPI, un framework, equivalente a Flask, que sirve para desarrollar aplicaciones API en Python. Pero al tener conocimientos sobre Flask, y que además, Flask es más escalable a FastAPI, la elección fue bastante rápida.

3.7.5. Validaciones

Una vez teniendo la API funcional, es importante asegurarse de que todos los endpoints funcionan correctamente, es por ello que se va a utilizar el software **Postman**. Una herramienta de desarrollo de software que permite probar APIs de una manera sencilla y eficiente. Donde se puede crear solicitudes HTTP y recibir respuestas en tiempo real, lo que permite verificar y validar el funcionamiento de la API creada (Muradas, 2019).

3.8. Business Intelligence

Las herramientas de Business Intelligence son programas informáticos que ayudan a las empresas a recopilar, organizar y analizar datos de diversas fuentes diferentes. Permitiendo presentar estos datos de una manera gráfica y fácil de entender a través de gráficos, informes y tableros de control. De esta manera, ayudan a las empresas a poder hacer un análisis más eficaz de los datos permitiéndoles tomar decisiones basadas en datos sólidos en lugar de suposiciones o intuiciones (Signaturit, 2021).

Hay una gran variedad de herramientas en el mercado. Pero entre las más utilizadas, se pueden destacar: Power BI, Tableau y Google Data Studio. Cada una de ellas tienen sus ventajas y desventajas, por lo que a la hora de elegir una de ella, hay que analizar cual se adecua más a las necesidades del proyecto.

Las principales características de la herramientas nombradas anteriormente, se puede observar en la siguiente tabla:

Power BI	**Tableau**	**Google Data Studio**
Desarrollador: Microsoft	Desarrollador: Tableau Software	Desarrollador: Google
Amplia gama de opciones de visualización	Excelente en visualización de datos	Enfocado en simplicidad y facilidad de uso
Fuerte integración con productos Microsoft	Permite conexiones a múltiples fuentes de datos	Integración con productos de Google
Ofrece varias opciones de licencia, incluyendo gratuita	Ofrece opciones de licencia, incluyendo versión gratuita	Totalmente gratuito y basado en la nube

Cuadro 3.6: Comparación de características de Power BI, Tableau y Google Data Studio.

Es cierto que durante el transcurso de la carrera, se han impartido clases sobre Google Data Studio; por lo que el conocer cómo funcionaba la herramienta de antemano era un gran aliciente para seleccionarla. Por otro lado está Tableau; pero al no conocer dicha herramienta, se ha preferido no hacer uso de ella. Y por último, esta Power BI; primero comentar que Innguma cuenta con una licencia de Microsoft, por lo que se dispone de la versión de pago de Power BI. Además también se conocía con anterioridad el como usar dicha herramienta; por lo que la opción por la que se ha decantado para la creación de reportes ha sido Power BI.

Capítulo 4

Metodología

A la hora de desarrollar un proyecto en cualquier ámbito es indispensable hacer uso de una metodología para garantizar el éxito del mismo. Como se ha indicado en el apartado del marco tecnológico, sección 3, se ha decidido hacer uso de una metodología predictiva ya que cuentan con una serie de ventajas que van a permitir dar una respuesta eficaz al desarrollo del producto.

Ventajas que se pueden destacar de las metodologías predictivas:

- Requieren una planificación exhaustiva al comienzo del proyecto. Definiendo todos los requisitos y objetivos antes de iniciar el desarrollo. De esta manera se tiene una idea clara del camino que se debe seguir para conseguir llevar a cabo el producto exitosamente
- Son efectivas para controlar los costos y plazos del proyecto. Facilitando el seguimiento y la gestión de desviaciones
- Se presta una atención significativa a la documentación, beneficiando la trazabilidad de requisitos, auditorías y futuras referencias

4.1. Diagrama de Gantt

Existen varias herramientas dentro de las metodologías predictivas. Pero la más utilizada, siempre ha sido el diagrama de Gantt, una herramienta de gestión de proyectos que se utiliza para visualizar y planificar las tareas y actividades de un proyecto en un tiempo determinado (Martins, 2022).

Como se ha comentado en el marco tecnológico, se ha hecho uso de una hoja de cálculo de Google Drive para crear el diagrama, ya que al ser totalmente personalizable, se ajusta a las necesidades del proyecto perfectamente.

A continuación, en las figuras 4.1 y 4.2 se puede observar cómo se han distribuido las tareas a lo largo del tiempo, separado en dos imágenes:

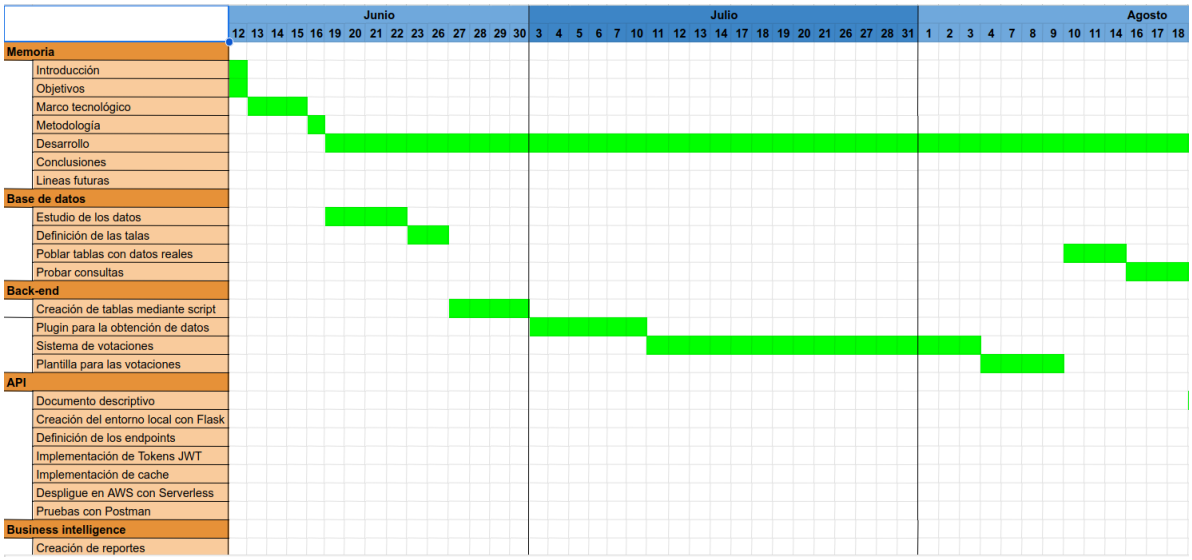

Figura 4.1: Primera parte del Diagrama de Gantt Fuente: Elaboración propia

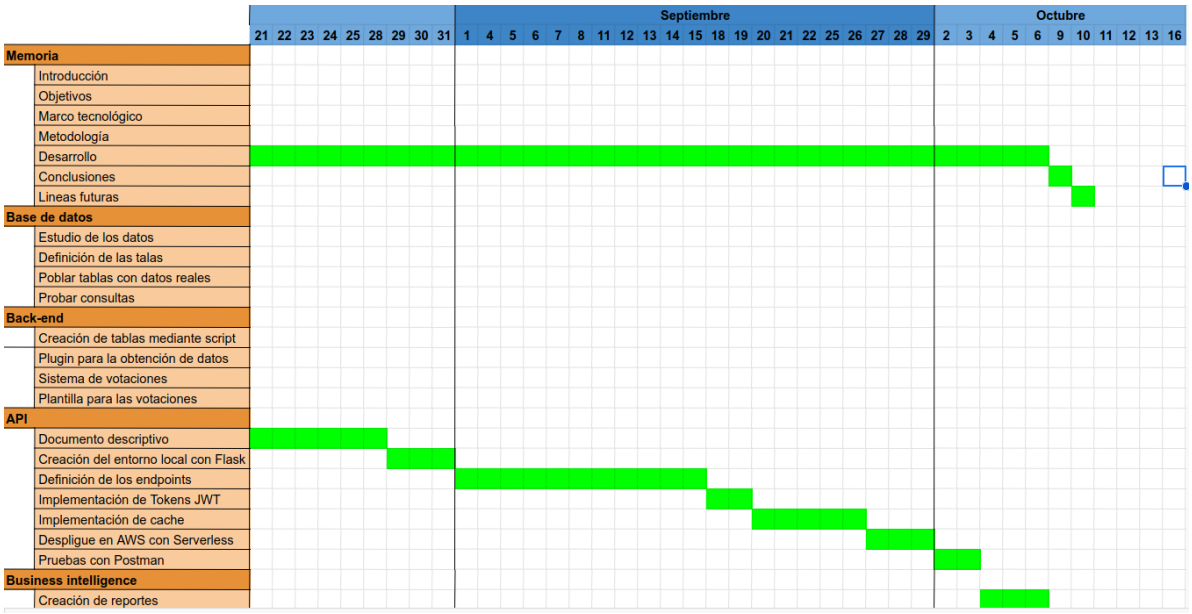

Figura 4.2: Segunda parte del Diagrama de Gantt Fuente: Elaboración propia

Como se puede ver en el diagrama, las tareas están separadas en 5 grandes bloques que son la memoria, la Base de Datos, el Back-end, la API y el Business Intelligence. Cada una de ellas tiene sus propias subtareas:

- Memoria: La memoria consiste en documentar todo el proceso del desarrollo del proyecto

  - Introducción: Consiste en la parte introductoria de la documentación, con el objetivo de dar contexto a las personas que van a leer la memoria
  - Objetivos: Una lista de objetivos que se quieren cumplir al completar el proyecto
  - Marco tecnológico: Determinar cuales son las tecnologías y los programas que se han utilizado en el desarrollo, y las diferentes alternativas que hay en el mercado
  - Metodología: Como se ha organizado las tareas para poder cumplir con el desarrollo del producto
  - Conclusiones: Después de terminar el trabajo, consiste en determinar cuales son las conclusiones
  - Líneas futuras: En caso de haber cosas pendientes por hacer o posibles mejoras que se pueden implementar, se describirán en el siguiente apartado del documento

- Base de Datos: Herramienta para almacenar y organizar los datos de una forma coherente y eficaz
  - Estudio de los datos: Antes de empezar a definir las tablas, ha sido necesario estudiar los datos que se guardan con anterioridad en las Bases de Datos de Innguma, y que nuevos datos se quieren añadir con la creación de las tablas
  - Definición de las tablas: Creación de un modelo de entidad relación para tener de una forma visual y eficaz, la estructura en la que se guardarán los datos en las tablas
  - Probar consultas: Realizar consultas que serán posteriormente implementadas a la API, para asegurarse de que se puede recoger toda la información deseada
  - Probar tablas con datos reales: Al utilizar la aplicación de Innguma, se irán guardando los datos correspondientes en las tablas. Consiste en hacer pruebas con dichos datos

- API: Programa que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades de una manera sencilla
  - Documento descriptivo: Antes de empezar con el desarrollo de la API, es importante crear un documento que recoja todos los detalles que se encontrarán en la API; como pueden ser, los endpoints, los parámetros, los códigos de respuesta, los Tokens JWT, la implementación de una memoria caché, etc.
  - Creación de entorno con Flask: Para el desarrollo de la API, se ha tenido que crear un entorno local con Flask, que esté implementado junto a Serverless, para poder programar la API de una manera eficaz
  - Definición de los endpoints: Consiste en recoger los datos necesarios de la Base de Datos, con sus queries correspondientes. Dependiendo a qué endpoint se le llame, se devolverá un JSON con unos datos u otros
  - Implementación de Tokens JWT: Para validar las solicitudes que se realizan a la API, se hace uso de los Tokens JWT
  - Implementación de caché: Para agilizar las solicitudes que devuelvan la misma información, implementar un sistema de memoria caché

    - Despliegue en AWS con serverless: Desplegar la aplicación de la API creada a los servicios de Amazon Web Services con el uso del framework Serverless
    - Pruebas con Postman: Validar todos los endpoint que devuelvan todo los datos correspondientes

- Business Intelligence: Herramientas que permiten consumir datos y mostrarlos mediante tablas, gráficos
    - Creación de reportes: Hacer uso de herramientas de Business Intelligence para crear informes, donde se muestran datos recogidos de la API mediante tablas, gráficos, etc.

4.2. Historias de usuario

Las historias de usuario son una pieza fundamental en el proceso de desarrollo de software. Estas historias son narrativas concisas que encapsulan necesidades, requisitos y funcionalidades clave desde la perspectiva del usuario. Su objetivo es definir de manera clara y sencilla qué debe hacer el sistema o la aplicación para satisfacer las expectativas y necesidades del usuario final. En esencia, las historias de usuario ayudan a responder a la pregunta fundamental: "¿Qué desea el usuario lograr con esta funcionalidad?". Estas historias suelen ser breves, directas y escritas en lenguaje natural, lo que las convierte en herramientas efectivas para la planificación y priorización de tareas en proyectos de desarrollo de software (Rehkopf, s.f.).

A continuación se pueden observar la lista de historias de usuarios que se han identificado para el desarrollo del proyecto:

(H1) Historia de usuario 1: Almacenar datos en la Base de Datos
Descripción: Como usuario quiero poder almacenar los datos de una manera eficiente en la Base de Datos
Validación: ■ Guardar los datos en la nueva arquitectura de tablas. ■ Los tipos de datos deben de ser los más adecuados para cada recurso.
Prioridad: Alta
Estimación: 40h
Dependencia: Ninguna

Cuadro 4.1: Historia de usuario 1: Almacenar datos en la Base de Datos

(H2) Historia de usuario 2: Sistema de votaciones
Descripción: Como usuario quiero poder votar las noticias de Innguma eficientemente
Validación: ▪ Cada usuario solamente podrá votar cada noticia una vez. ▪ La votación ira con un sistema de estrellas, con el valor entre 1 y 5. ▪ Los usuarios podrán ver la valoración media de la noticia.
Prioridad: Media
Estimación: 72h
Dependencia: Almacenar datos en la Base de Datos

Cuadro 4.2: Historia de usuario 2: Sistema de votaciones

(H3) Historia de usuario 3: Aplicación API
Descripción: Como usuario quiero poder realizar solicitudes a una API para poder recoger los datos
Validación: ▪ Todos los endpoints definidos deberán devolver la información deseada. ▪ La API contará con diferentes endpoints, donde cada una devolverá unos datos diferentes. ▪ El acceso a la API solo estará disponible para los usuarios que dispongan de las credenciales correctas.
Prioridad: Alta
Estimación: 120h
Dependencia: Almacenar datos en la Base de Datos y el Sistema de votaciones

Cuadro 4.3: Historia de usuario 3: Aplicación API

(H4) Historia de usuario 4: Reportes Business Intelligence
Descripción: Como usuario quiero disponer de una plantilla sobre un reporte creado con Power BI
Validación: ▪ Los datos deben de ser obtenidos haciendo peticiones a la API. ▪ El reporte contará con diferentes tablas y gráficos..
Prioridad: Media
Estimación: 16h
Dependencia: Almacenar datos en la Base de Datos, el Sistema de votaciones y la aplicación API

Cuadro 4.4: Historia de usuario 4: Reportes Business Intelligence

4.3. Mockups

Antes de empezar con la implementación del código, es necesario tener una idea clara de cómo va a ser el diseño del Front-end de la aplicación; en este caso, como se va a ver el nuevo sistema de votaciones de las noticias publicadas en Innguma.

Los mockups, son una parte crucial de diseño, que aportan claridad, ahorran recursos y mejoran la comunicación con el cliente. De esta manera se puede conseguir la aprobación de este, una etapa más temprana del proyecto, sin tener que dedicar tanto tiempo al desarrollo de la misma.

Por esta razón, el mockup que se ha creado sirve para visualizar cómo se va a ver el sistema de votación. Como se puede observar en la figura 4.1:

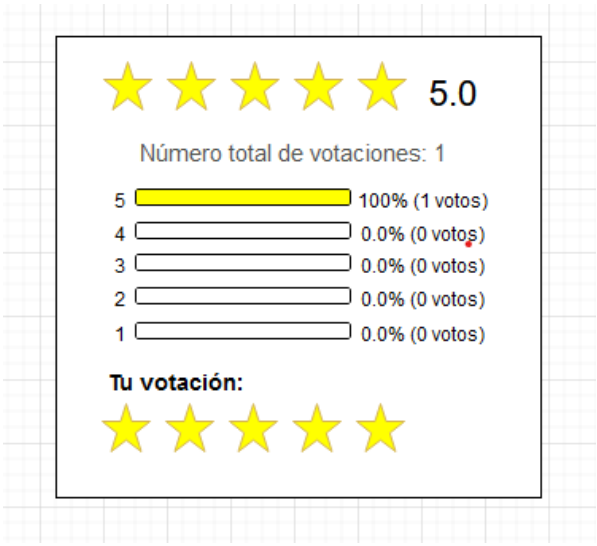

Figura 4.3: Mockup del Sistema de votación Fuente: Elaboración propia

La idea principal ha sido que en la parte superior el usuario pueda observar la media de votaciones de las noticias. Justo debajo, se podrá ver el recuento de votaciones que se han realizado dentro de la propia noticia. En la mitad, se podrá ver las votaciones de una manera más detallada, pudiendo saber cuantos votos de cada puntuación (de 1 a 5 estrellas) se han realizado. Y finalmente, mediante las estrellas, cada usuario podrá ver cuál ha sido su voto dentro de la noticia que está visualizando.

4.4. Arquitectura del sistema

Antes de empezar con la explicación del propio desarrollo en sí, es importante tener una idea clara de cuál es la arquitectura completa del sistema, para tener una visión global del proyecto y poder ubicar cada apartado de una manera más sencilla y eficaz. La arquitectura se muestra en la figura 4.4:

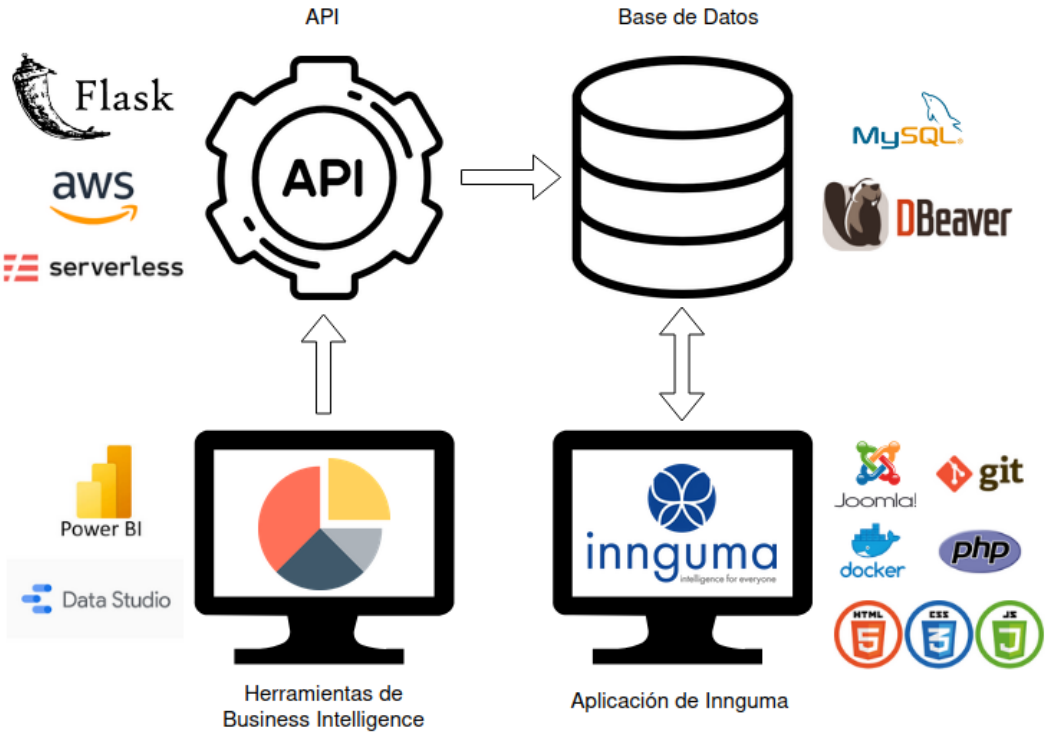

Figura 4.4: Arquitectura del sistema con las herramientas que se han utilizado Fuente: Elaboración propia

Por un lado se encuentra la propia aplicación con la que ya cuenta Innguma. Donde se desarrollarán diferentes cambios; tanto en el Back-End y el Front-end, haciendo uso del gestor de contenidos de Joomla. Como el nuevo sistema de votaciones, obtención de datos mediante plugins y scripts de creación de las tablas de las Bases de Datos, mediante la instalación del componente principal de la aplicación.

Por otro lado, está la Base de Datos. Donde se ha hecho uso de una Base de Datos MySQL para crear una nueva arquitectura de tablas para poder guardar eficientemente todos los datos que posteriormente se van a querer consumir.

También se puede encontrar la aplicación API. Que será una nueva aplicación desarrollada desde cero (independiente a la aplicación de Innguma) mediante Flask, que será desplegada mediante Serverless en los diferentes servicios de Amazon Web Services; con el objetivo de poder consumir los datos de la Base de Datos de una manera sencilla y eficaz, mediante peticiones a los endpoints de la API.

Y por último, se encuentran las herramientas de Business Intelligence, como pueden ser Power BI de Microsoft o Data Studio de Google. Con las que se podrán recoger los datos de la aplicación de Innguma, mediante el uso de la API. De esta manera se podrán crear tablas, gráficos, etc.

4.5. Base de Datos

Antes de crear la API es importante disponer de todos los datos necesarios para que se puedan consumir correctamente. Innguma ya cuenta con su propia base de datos y con su arquitectura de tablas. Pero en el momento de hacer la definición de la API (que se comentará en la sección 5.2) y un estudio de los datos que se quieren otorgar, se ha visto la necesidad de crear nuevas tablas y de editar tablas ya existentes.

4.5.1. Modelo Entidad Relación

Un diagrama de entidad relación es una herramienta gráfica utilizada en el diseño de bases de datos para representar la estructura lógica de un sistema o aplicación. Es decir explica de una forma visual y sencilla, cómo se van a estructurar las tablas para poder guardar los datos correctamente (Lucidchart, s.f.-a).

En la siguiente imagen se puede observar la estructura de las tablas y sus relaciones, dando una solución al sistema que se quiere implementar posteriormente. Como ya se ha comentado, Innguma ya contaba con su base de datos, la cual se ha tenido que modificar. Los nombres de las tablas que se muestran en rojo, quiere decir que son tablas creadas desde cero; es decir, que no existían de antes. Por otro lado en la tabla “jos_k2_attachment”, se ha añadido un nuevo atributo. Por su puesto, también se han creado las nuevas relaciones necesarias entre las tablas.

A la hora de determinar el nombre de las tablas y sus atributos, se ha seguido el patrón de las que ya existían con anterioridad. El nombre comenzará con el prefijo de jos, prefijo impuesto por Joomla; y las palabras se separarán con una barra baja.

En la figura 4.5, se puede observar, el Modelo Entidad Relación que tiene la Base de Datos

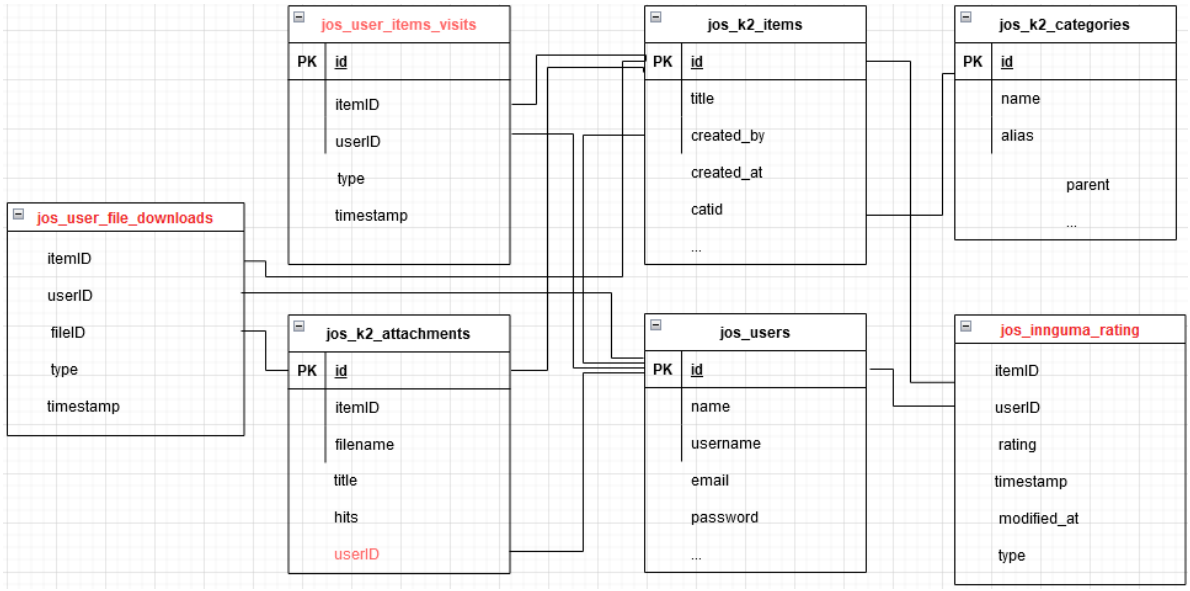

Figura 4.5: Modelo entidad relación de la Base de Datos Fuente: Elaboración propia

4.5.2. Tablas

A continuación se explica para qué se utiliza cada tabla creada dentro de la nueva arquitectura de la Base de datos:

- jos_user_file_downloads: Una tabla con el objetivo de guardar datos sobre la relación entre un usuario y que adjunto ha descargado.
- jos_user_items_visits: La tabla guardará las visitas que hacen los usuarios a las noticias publicadas en la plataforma.
- jos_k2_items: Tabla donde se guardan todas las noticias que se publican en la aplicación.
- jos_k2_categories: Tabla donde se guarda el listado completo de las diferentes categoría de la aplicación.
- jos_k2_attachments: Tabla donde se guardan todos los adjuntos que se suben dentro de las noticas en la plataforma de Innguma.
- jos_users: Tabla donde se guardan todos los datos relacionados con los usuario de la plataforma.
- jos_innguma_rating: Nueva tabla donde se guardarán las valoraciones que hacen los usuarios a las noticias de la plataforma.

Para poder ver los atributos que contiene cada tabla de una forma más detallada, se podrá consultar el apéndice “Atributos de las tablas de la Base de Datos” en el apéndice A.

Capítulo 5

Resultados

En este capitulo, se explicarán los resultados obtenidos con el desarrollo del proyecto. Donde se comentará los cambios implementados dentro de la propia aplicación de Innguma, en la sección 5.1. Por otro lado, se explicará el desarrollo para conseguir la aplicación API y los resultados de la misma, en el apartado 5.2. Y finalemente, el reporte que se ha creado haciendo uso de la herramienta de Business Intelligence en la sección 5.3:

5.1. Aplicación de Innguma

Una vez habiendo definido las tablas que se han de crear, pasamos al desarrollo de las funcionalidades dentro de la aplicación de Innguma; donde se podrá diferenciar tres bloques principales: La creación de las tablas de la Base de Datos mediante un script, creación de un nuevo plugin de sistema para rellenar las tablas con los datos correspondientes y finalmente, un nuevo sistema de votación para las noticias de Innguma.

5.1.1. Creación de tablas

A la hora de crear las nuevas tablas definidas con anterioridad, no se ha optado por hacerlo de una manera manual, ya que si es cierto que esta forma puede agilizar notoriamente la velocidad de implementación; también viene seguido de varias desventajas.

Es por eso que se ha decidido hacer la creación de estas mediante un script que se lanza cada vez que se instala una nueva versión del componente de k2 (componente principal de Joomla). Aprovechando que para hacer uso de las nuevas implementaciones que se han programado en el backend, son todas parte del componente, al hacer la instalación de la nueva versión, ya se tendrá todos los cambios aplicados.

Al hacer la creación de las tablas mediante script, también ofrece otra gran cantidad de ventajas. Ya que una vez teniendo el script listo, solamente habrá que ejecutarlo (se lanza con la instalación del componente de k2) y se implementarán todos los cambios automáticamente. Como Innguma cuenta con una gran cantidad de clientes, y cada uno tiene su propio entorno de trabajo; esto conlleva también a que cada uno tenga su propia Base de Datos, por lo que la creación de las nuevas tablas hay que realizarlas para todos los cliente; por lo que el script agiliza el proceso enormemente, facilitando la reproducción de la estructura de la

Base de Datos en diferentes sistemas y evitando problemas de sincronización, garantizando la consistencia.

Para la creación de las tablas, se ha hecho uso de JDatabase; un sistema que permite que Joomla sea independiente de un sistema de gestión de Base de Datos en específico. JDatabase ofrece una serie de funciones y métodos que permiten realizar operaciones con la Base de Datos de una manera eficiente y segura.

En la figura 5.1 se muestra un ejemplo de como se ha creado la tabla de innguma_rating mediante el uso de los métodos que ofrece JDatabase:

```
$query = "CREATE TABLE IF NOT EXISTS `#__innguma_rating` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `itemID` int(11) NOT NULL,
    `userID` int(11) NOT NULL,
    `type` varchar(255) NOT NULL,
    `rating` TINYINT NOT NULL,
    `timestamp` datetime NOT NULL,
    `modified_at` datetime NOT NULL,
    PRIMARY KEY (`id`),
    KEY `itemID` (`itemID`),
    KEY `userID` (`userID`)
  ) ENGINE=MyISAM DEFAULT CHARSET=utf8;";
$db->setQuery($query);
$db->query();
```

Figura 5.1: Creación de la tabla de innguma_rating Fuente: Elaboración propia

5.1.2. Obtención de datos

Para rellenar las tablas con los datos necesarios se ha hecho uso del sistema de eventos (triggers) que ofrece Joomla. Los eventos son momentos clave en la ejecución de Joomla, como la carga de una página, el inicio o final de una tarea, la publicación de contenido, etc. Cuando ocurre un evento específico, se desencadena una notificación y se permite que las extensiones personalizadas (plugins, módulos, componentes) reaccionen y realicen acciones específicas.

En el caso de las tres tablas: “jos_user_file_downloads”, “jos_user_items_visits” y “jos_innguma_rating” se ha creado un plugin de sistema para rellenarlas con los datos correspondientes. Este, es una extensión que se utiliza para agregar funcionalidad adicional o modificar el comportamiento del sistema principal de Joomla. Los plugins de sistema son una parte fundamental de la arquitectura de Joomla y permiten a los desarrolladores extender y personalizar la funcionalidad de forma modular. Pero en el caso del plugin que se ha creado; este será el encargado de recoger tres eventos diferentes y rellenar su respectiva tabla con los

datos correspondientes.

Para la tabla “jos_user_file_downloads”, se recoge el evento “onK2AfterDownload”. Evento que se lanza cada vez que un usuario descarga un adjunto perteneciente a una noticia. En el momento que se recoge el evento, se recoge que usuario ha sido el encargado de hacer la descarga y cuál ha sido el adjunto descargado; de esta manera se rellena dicha tabla con los datos necesarios.

Por otro lado, para la tabla “jos_user_items_visits” se recoge el evento “onAfterDispatch” propio de Joomla. Este evento se lanza cada vez que se carga cualquier vista de la aplicación, por lo que es necesario identificar cuando corresponde a la vista de una noticia. Para ello, dentro de la url, en el parámetro “option” podemos mirar si está en la vista de una noticia de “k2” o “dbcreator”, parámetro que determina el el tipo de la noticia, el cual se guardará en la columna “type”. Al recoger el evento habrá que identificar qué usuario ha visitado la noticia y guardar los datos en la tabla.

Por último, en la tabla “jos_innguma_rating”, se guardará la valoración de un usuario respecto a una noticia. Para ello, cada vez que en una vista de noticia, cuando el usuario vote dicha noticia, se lanzará el evento “onAfterVoteK2item”, el cual será capturado por el plugin de sistema creado y se recogerá los datos necesarios para rellenar la tabla correctamente. La lógica del sistema de votación y el lanzamiento del evento se explicará más detalladamente en la siguiente sección 5.1.3.

5.1.3. Sistema de votación

Innguma ya contaba con un sistema de votación que era propio de Joomla. Pero este, no cumplía con los requisitos necesarios. Ya que este sistema, solamente mostraba al usuario la media de los votos mediante el uso de estrellas donde la votación podía ir entre 1 y 5. Por otro lado, también se le permitía a cualquier usuario votar todas las veces que quisiese, incrementando o decrementando la media de la noticia a su gusto; lo cual no tenía ningún sentido. La media de la noticia se mostraba mediante las estrellas como se puede observar en la figura 5.2:

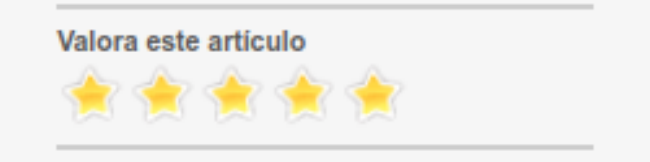

Figura 5.2: Sistema de votación antiguo Fuente: Elaboración propia

Al haber creado una nueva tabla de votaciones, donde se guardará la información de la relación entre usuario y noticia, guardando la votación que se ha dado. De esta manera se ha limitado el número de votos a una noticia del mismo usuario a uno. Obviamente, pudiendo editar el voto, en caso de haber votado con anterioridad.

Por otro lado, al tener las votaciones de todos los usuarios, en la respectiva noticia, se dio la posibilidad de mostrar el número de votos por cada estrella y la media de esta. Haciendo que el sistema de votación actual muestre más información útil, comparando al sistema anterior. El nuevo sistema se mostrará de la siguiente manera, como se puede ver en la figura 5.3:

Figura 5.3: Sistema de votación nuevo Fuente: Elaboración propia

Primero se mostrará la media de votos mediante las estrellas, de una manera más gráfica; pero también se mostrará la media con el uso del número, a su derecha. Seguido del contador del número total de usuarios que han votado la noticia. Después, como ya se ha comentado, se mostrará la información más detalladamente, por cada número de votación disponible (del 1 al 5), con su propia media y el número de votos. Finalmente, también se hará uso de las estrellas para mostrar al usuario cuál ha sido su votación.

Para poder implementar este nuevo sistema, ha requerido realizar bastantes cambios en diferentes partes de la aplicación. Primero en la parte del backend para lanzar el evento onAfterVoteK2item, comentado anteriormente, para poder recogerlo en el plugin y rellenar las tablas de la Base de Datos, cada vez que un usuario realice una votación. También para recoger los datos actualizados de la Base de Datos para poder actualizar la vista después de que un usuario haya votado, como puede ser la media actualizada o el número de votos.

Por otro lado, en el frontend, se ha hecho uso de JavaScript para poder realizar una llamada Ajax, en el momento que el usuario haya votado, para poder enviar al servidor la votación realizada. Y en el momento que el servidor haya realizado todos los cambios, recogerá la respuesta de la llamada para actualizar los datos del sistema de votación mediante el uso del propio JavaScript.

Pero al crear este nuevo sistema surgió un problema. ¿Qué pasaría con todas las votaciones que se hayan realizado en el antiguo sistema? ¿Se perderían? Es por ello, que se decidió, que en el momento de crear las tablas, en el script que se lanza en la instalación del componente, también se rellenará la tabla de innguma_rating siguiendo el siguiente algoritmo: Si la

media de la noticia, el decimal, está por debajo de x.3, se guardará el valor de x en la nueva tabla, todas las veces que se haya votado la noticia. Si la media está por encima de x.7, se guardará el valor de x+1; y en caso de que la media esté entre x.3 y x.7, se guardará el valor de x.5 en la Base de Datos. De esta manera, si es cierto que no tendríamos la votación real de las noticias antiguas, pero si un aproximamiento. Pero en el momento que se vayan añadiendo nuevas noticias y votando con el nuevo sistema, la media del voto, como la demás información, sí que será real.

Resumiendo, con este sistema de votación, se llevará un registro más real de los votos que ha recibido una noticia; porque cada usuario podrá votar solamente una vez. Y se mostrará de una forma más detallada todos los votos que ha recibido la noticia.

5.2. API

Habiendo recogido los datos necesarios y guardarlos en la base de datos, es necesario poder consumir dicha información. Es por ello que se ha desarrollado una API para el consumo de estos. De esta manera, la API permitirá que otras aplicaciones o sistemas interactúen con los datos y funcionalidades, facilitando la integración con otros servicios o plataformas.

Por otro lado una API bien definida, facilita enormemente la escalabilidad del sistema, ya que las nuevas funcionalidades que se puedan crear para la aplicación, se pueden integrar fácilmente. Por lo que crear una API puede proporcionar una mayor flexibilidad, colaboración y oportunidades de crecimiento de la aplicación.

Antes de crear una la API, es importante crear un documento con la definición de la misma. Donde se explicará los siguientes apartados:

- Descripción de la API
- Acceso a la API
- Recursos y datos a recoger
- Definición de los endpoints
- Parámetros disponibles en las peticiones
- Respuestas de la API
- Códigos de respuesta

5.2.1. Recursos

Los recursos son las entidades de las que se va a recoger información de la base de datos. Y posteriormente a partir de estos recursos se definirá los endpoints correspondientes para realizar las peticiones y recoger los datos necesarios en cada una. La API actualmente contará con 4 recursos, que son los siguientes:

- **Lectores:** Son los usuarios principales. Acceden a la aplicación para leer las noticias publicadas por los analistas. De esta manera, estarán informados en todo momento.

- **Analistas:** Son los usuarios encargados de realizar la vigilancia tecnológica y la inteligencia competitiva. Es decir, son los que recogen la información relevante y la publican para que los miembros de su organización (los lectores) estén enterados de los que sea necesario. Por otro lado, los analistas también pueden ejercer la función de lectores y leer las noticias que publican los demás analistas.
- **Noticias:** Las noticias son los ítems que han sido seleccionados por los analistas y han sido publicados para que los lectores las puedan consumir. Se podría considerar la información relevante que se ha conseguido al realizar la vigilancia tecnológica.
- **Categorías:** Las categorías se podrían considerar las carpetas donde los analistas publican las noticias. Los lectores, se pueden suscribir a estas y cada vez que se publique una noticia, podrán leerla.

Para poder echar un vistazo a la información que se podrá obtener de cada recurso, consultar el apéndice B “Recursos de la API”.

5.2.2. Endpoints

Una vez habiendo definido los recursos y entidades que tendrá la API, se han definido los endpoints. Estas, son las URLs específicas a las que se puede enviar una petición para interactuar con una API. Cada recurso, anteriormente mencionado, cuenta con una serie de endpoints. Cada endpoint devolverá una información u otra del recurso solicitado. Estos son los endpoints que se han definido:

- Lectores:
  - /v1/statsreaders - Devuelva la lista de lectores
  - /v1/statsreaders/overview - Devuelve la información completa de todos los lectores
  - /v1/statsreaders/id/overview - Devuelve la información completa del usuario especificado mediante el id
  - /v1/statsreaders/id/items/viewed - Devuelve la lista de noticias que hayan sido visitadas por del lector especificado
  - /v1/statsreaders/id/items/voted - Devuelve la lista de noticias que hayan sido visitadas por del lector especificado
  - /v1/statsreaders/id/items/files/downloads - Devuelve la lista de documentos descargados y su noticia asociada del lector especificado
  - /v1/statsreaders/id/categories/subscriptions - Mostrar la lista de categorías a las que está suscrito el lector especificado
- Analistas:
  - /v1/statsanalyst - Devuelve la lista de los analistas
  - /v1/statsanalyst/overview - Devuelve la información completa respecto a los analistas
  - /v1/statsanalyst/id/overview - Devuelve la información completa respecto al analista que se a definido mediante el id en el endpoint

- /v1/statsanalyst/id/items/published - Muestra la lista de noticias que han sido subidas por el analista especificado en el endpoint
- /v1/statsanalyst/id/items/published/votes - Se recoge la lista de noticias que han sido publicados por el analista y que hayan sido votados por los lectores
- /v1/statsanalyst/id/items/viewed - Devuelve la lista de items con visitas que hayan sido subidos por el analista definido en el id del endpoint
- /v1/statsanalyst/id/items/files/downloads - Devuelve la lista de adjuntos descargados por el analista definido en el id del endpoint
- /v1/statsanalyst/id/items/files/uploads - Devuelve la lista de documentos subidos por el analista definido en el id del endpoint
- /v1/statsanalyst/id/categories/subscription - Devuelve la lista de categorías a las que el analista definido con el id en el endpoint está suscrito

- Noticias:
  - /v1/statsnews - Devuelve la lista de noticias
  - /v1/statsnews/overview - Muestra la información completa de las diferentes noticias
  - /v1/statsnews/id/overview - Muestra la información completa de la noticia especificada en el endpoint mediante el id
  - /v1/statsnews/id/votes - Se recoge el listado de usuarios que han votado la noticia especificada mediante el id
  - /v1/statsnews/id/visits - Se recoge el listado de usuarios que han visitado la noticia especificada mediante el id
- Categorías:
  - /v1/statscategories - Devuelve la lista de categorías
  - /v1/statscategories/overview - Devuelve la información completa de todas las categorías
  - /v1/statscategories/id/overview - Devuelve la información completa de la categoría especificada en el id del endpoint
  - /v1/statscategories/id/subscriptions - Lista de usuarios suscritos a la categoría seleccionada
  - /v1/statscategories/id/items/votes - Devuelve una lista de las noticias que han sido votadas dentro de la categoría seleccionada
  - /v1/statscategories/id/items/published - Devuelve la lista de noticias publicadas dentro de la categoría
  - /v1/statscategories/id/items/files/downloads - Devuelve la lista de adjuntos que hayan sido descargados dentro de la categoría seleccionada
  - /v1/statscategories/id/items/files/uploads - Muestra una lista de los adjuntos que hayan sido subidos dentro de la categoría

5.2.3. Códigos de respuestas

Para saber cómo ha ido la petición HTTP, la API deberá devolver un código de respuesta junto un mensaje explicativo, dependiendo la diferentes situaciones que se puedan dar. Los códigos que se han definido son los siguientes (Mozilla, s.f.):

- **200** - La petición ha sido exitosa y la respuesta incluirá los datos correspondientes al endpoint que se haya llamado, teniendo en cuenta los parámetros de filtros que el usuario ha definido.

```
{
"code":200,
"status":"success",
"name":"Lista de lectores",
"numberOfItems": 4,
"data": {}
}
```

- **400** - El servidor no ha podido interpretar correctamente la solicitud. Ya sea por una sintaxis incorrecta de la llamada o por hacer un uso incorrecto de los parámetros de la solicitud. Se devolverá un JSON con el código respuesta y un mensaje explicando la razón del error.

```
{
  "code": 400,
  "status": "error",
  "message": "Bad Request: The server cannot process the request due
     to a client error"
}
```

- **404** - El servidor no pudo encontrar el contenido solicitado, ya que el recurso solicitado no existe. En las llamadas a las que se accede a una fila de una entidad en concreto mediante el id, puede que el id otorgado por el usuario en la llamada no exista y se de este error. En este caso, se devolverá un JSON con el código respuesta y un mensaje explicando la razón del error.

```
{
  "code": 404,
  "status": "error",
  "message": "Not Found: The server cannot find the requested
     resource"
}
```

- **500** - Error inesperado de servidor. Este error no corresponde a un fallo del usuario, si no del cliente. Y se puede dar por el incorrecto funcionamiento del servidor. Se devolverá un JSON con el código respuesta y un mensaje explicando la razón del error.

```
{
  "code": 404,
  "status": "error",
  "message": "Internal Server Error: The server encountered an
     unexpected condition that prevented it from fulfilling the
     request"
}
```

5.2.4. Diagrama de secuencia

El diagrama de secuencia es una herramienta gráfica que se utiliza para representar la interacción entre diferentes objetos o componentes de un sistema a lo largo del tiempo. Este tipo de diagrama es especialmente útil para modelar y comprender la secuencia de acciones y eventos que ocurren en una aplicación web durante la ejecución (Lucidchart, s.f.-b). En este caso, se ha utilizado para mostrar el recorrido completo después de que el usuario haya hecho una solicitud a la API.

El diagrama de secuencia se puede observar en la figura 5.4:

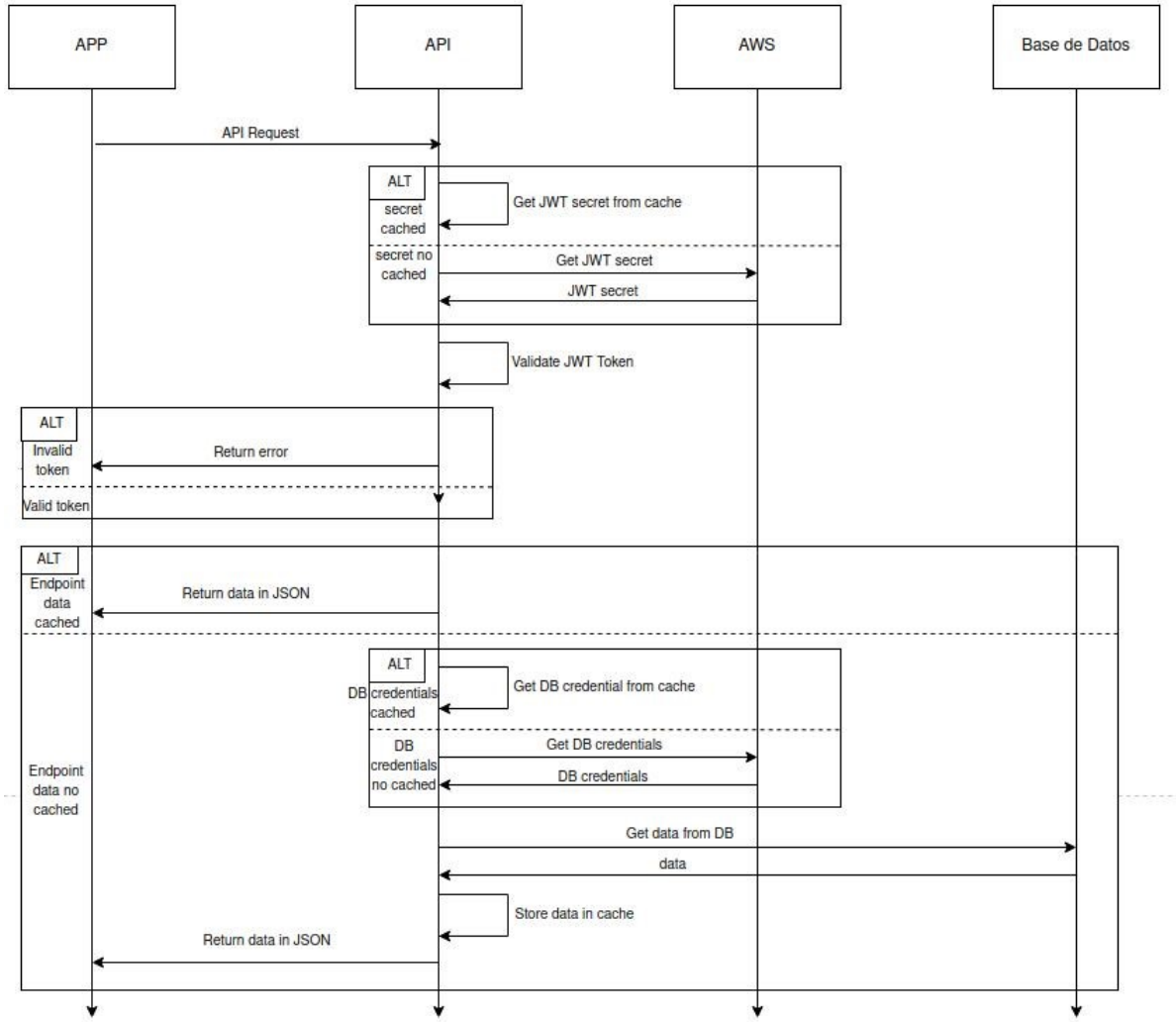

Figura 5.4: Diagrama de secuancia de una petición a la API Fuente: Elaboración propia

Como se puede observar hay cuatro entidades, que son:

- La **app** es donde los usuarios pueden ver los datos de Innguma mediante la representación de tablas, gráficos, etc. Es la encargada de hacer las peticiones a la API para recoger los datos que se van a mostrar.
- La **API**, la cual es la encargada de gestionar y procesar las diferentes peticiones que se realizan desde la APP. También es la encargada de realizar la conexión con Amazon Web Services y la Base de Datos.
- **Amazon Web Services** es donde se almacenan las diferentes credenciales de la base de datos y el secret del Token de JWT, necesario para validar las solicitudes. Se guardan mediante el uso de AWS secrets.
- La **Base de Datos**, donde se guarda toda la información necesaria sobre la aplicación y los usuarios.

Las solicitudes se realizan desde la propia app, en el momento en el que el usuario cargue alguna vista, de esta manera se solicitan los datos a la API para poder rellenar los gráficos/-tablas...

Al haber desarrollado una API, también se posibilita que se hagan informes con el uso de herramientas de Business Intelligence, por lo que es importante que la gestión de las peticiones siga unos estándares. Es por eso, que las peticiones se pueden realizar en dos formatos diferentes.

En la primera, el subdominio, el cual determina a qué base de datos hay que realizar la conexión, se manda mediante la cabecera X-subdomain siguiendo las reglas establecidas sobre las peticiones HTTP. Pero como esta manera no sigue los estándares, la otra forma disponible de mandar el subdominio será mediante el uso del payload del token de JWT.

Al llegar la solicitud de la API y recoger el token de la llamada, primero se mira si se dispone del JWT secret en la caché. En caso de no estar, hay que hacer una petición a AWS secret para recogerlo. Y por consiguiente, habrá que cachear durante un tiempo determinado.

Al disponer del secret, se usará el token de JWT para verificar que la solicitud es válida y que está permitido recoger los datos del endpoint correspondiente. En caso de no ser válida, se devolverá al usuario que ha realizado la petición, una respuesta con el error indicando que la petición no es válida.

En caso de ser válido, el siguiente paso será mirar en caché si los datos del endpoint solicitado están cacheados o no. En caso de estarlo, simplemente se recogerá dichos datos de la caché y se le devolverá al usuario los datos solicitados en formato JSON.

Si los datos del endpoint no están disponibles en la caché, habrá que volver a mirar la caché, pero en este caso, para recoger las credenciales de acceso de la Base de Datos, teniendo en cuenta el subdominio de la misma. Si no están cacheados, habrá que volver a realizar una llamada a AWS para recoger dichas credenciales. Al guardar las credenciales en AWS, añadimos una nueva capa de seguridad, para que no sean vulnerables ante un supuesto ataque.

Una vez se disponga de las credenciales de acceso, se podrá entrar en la función del propio endpoint y realizar todas las llamadas necesarias a la Base de Datos para recoger todos los datos solicitados y montar el JSON.

Antes de devolver el JSON al usuario que ha realizado la petición, se tiene que cachear los datos recogidos, para agilizar las próximas llamadas (solamente la peticiones que devuelven los mismo). Es por ello, que se ha definido una estrategia para determinar cuándo se debe invalidar los datos de la caché. Por un lado, se hace uso de varios parámetros para identificar la llamada realizada de una manera única. Los parámetros son:

- Subdominio: Parámetro que determina a qué Base de Datos hay que conectarse
- Endpoint: URL a la que se le está realizando la petición

- Id: Hay endpoints a los que se le añade un ID. Por ejemplo, para recoger los datos de un lector en concreto
- Filtros: Al realizar las llamadas los endpoints también se le puede pasar varios filtros, como puede ser el campo por el que se ordenarán los datos obtenidos

En caso de que todos los parámetros se repitan, se puede asegurar que los datos que se tienen que devolver van a ser los mismos, en una llamada que en otra. Por lo que en este caso se hará uso de la caché para agilizar la petición.

Por otro lado, también hay que invalidar la caché, en el momento en el que pase un tiempo determinado. Ya que es posible que se haya añadido nueva información en la Base de Datos y haya que volver a recoger los datos.

5.2.5. Token JWT

Un token JWT (JSON Web Token) es una cadena codificada que está formada por tres partes: header, payload y signature. Como ya se ha comentado anteriormente se va a hacer uso del token para determinar si las peticiones que se están realizando por parte de los usuarios son válidas o no. Ya que solo los usuarios que dispongan del secret, serán capaces de realizar llamadas válidas. (Poddar, 2022a)

- Encabezado (header): Contiene información sobre el tipo de token y el algoritmo de firma utilizado. Por ejemplo:

```
{
  "alg":HS256,
  "typ":"JWT"
}
```

- Payload: Contiene la información que se desea transmitir. En este caso, el subdominio, para saber la Base de Datos a la que se tiene que hacer la solicitud para recoger los datos. Ejemplo:

```
{
  "subdomain":innguma
}
```

- Signature: La firma se utiliza para verificar que el token no ha sido alterado durante la transmisión. La firma se crea utilizando el encabezado codificado, el payload codificado, una clave secreta y un algoritmo de firma. La firma se agrega como una tercera parte después de la última coma en el token.

```
HMACSHA256{
   base64UrlEncode(header) + "." +
   base64UrlEncode(payload),
   your-secret
}
```

Y el ejemplo completo del token JWT sería el siguiente, con sus tres respectivas partes diferenciadas por colores:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.
Ey0RZ2qy7uWBwsSf7hLtoyQhr-Vibv5tcbKtcaYp2

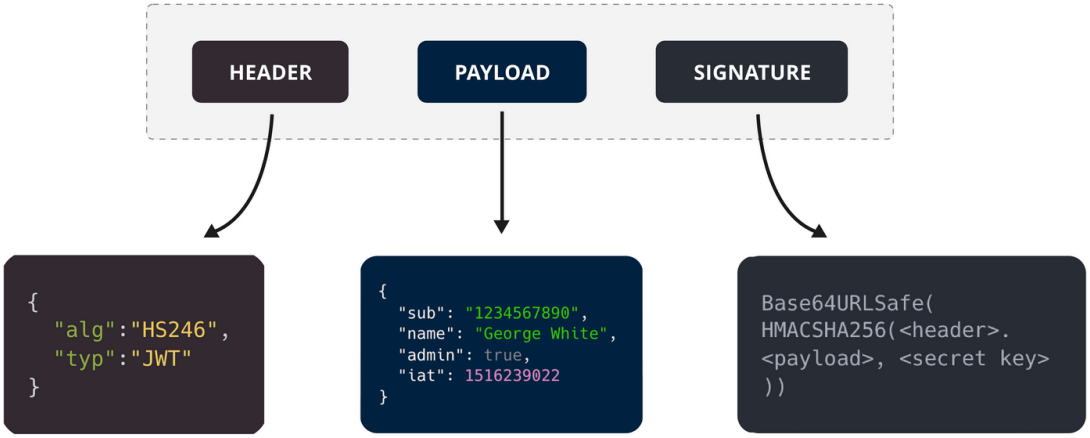

Figura 5.5: Estructura de un Token JWT
Fuente: (Poddar, 2022b)

El Token JWT, seguirá esta estructura cuando la petición es necesario que siga unos estándares, para que pueda ser implementado sin dificultades en herramientas de Business Intelligence (como Power BI); ya que estos, siguen estándares.

5.2.6. Arquitectura de la API

El desarrollo de la aplicación API se ha hecho haciendo uso de **Flask**, un framework que permite crear aplicaciones en Python de una manera sencilla y flexible. Una vez la API esté desarrollada hay que hacer el despliegue a los servicios de Amazon, para ello, se ha utilizado **Serverless**, otro framework que automatiza el proceso de despliegue de la aplicación a Amazon API Gateway y AWS Lambda.

La API va a hacer uso de varios servicios de AWS (Amazon Web Services). Por un lado se utilizará **Amazon API Gateway**, un servicio completamente administrado que te permite crear, publicar y administrar APIs de manera rápida y sencilla. Permite definir los endpoints de la API, establecer la autenticación y autorización de las mismas, configurar el enrutamiento de las llamadas...

También se va a utilizar el servicio **AWS Lambda** que ofrece Amazon. Ya que con API Gateway podemos enrutar las solicitudes HTTP a las funciones lambda. Estas funciones serán las encargadas de interactuar con la Base de Datos, para poder recoger los datos solicitados

por la petición.

La combinación de Amazon API Gateway y AWS Lambda permitirá construir una API segura, altamente escalable y fácil de mantener. Con esta arquitectura sin servidor, se asegura el enfoque en el desarrollo de la aplicación y brindar una experiencia óptima a los usuarios.

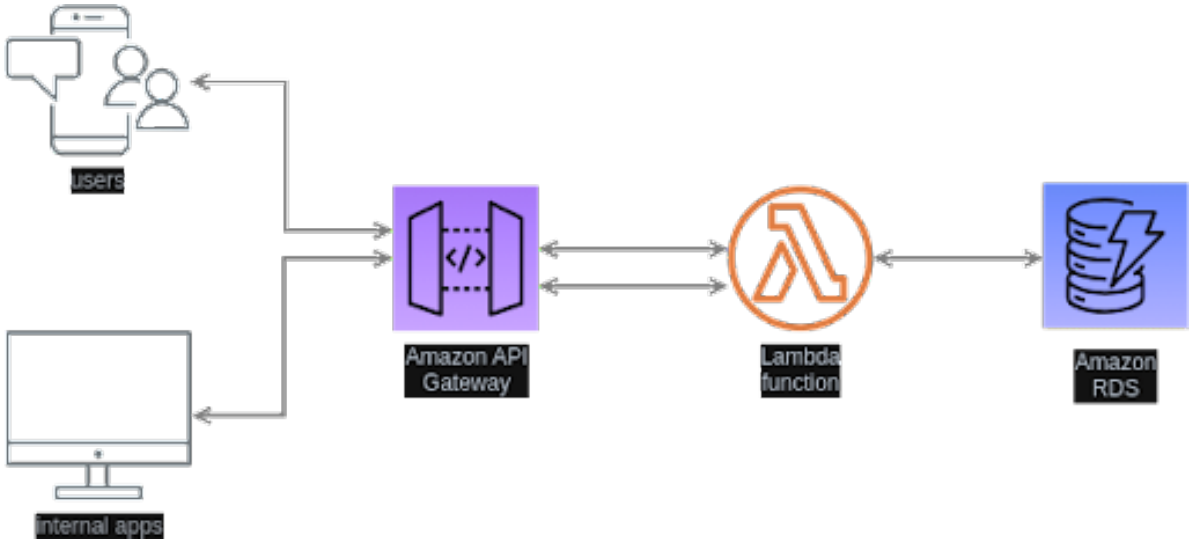

Figura 5.6: Arquitecturá de los servicos AWS para la implementación de la API Fuente: Elaboración propia

5.2.7. Validación con Postman

Por otro lado, se ha hecho uso del software Postman. Una herramienta de desarrollo de software que permite probar APIs de una manera sencilla y eficiente. Donde se puede crear solicitudes HTTP y recibir respuestas en tiempo real, lo que permite verificar y validar el funcionamiento de la API creada (Muradas, 2019).

Para probar los diferentes endpoints de la API, se creará una colección donde se realizarán diferentes llamadas con el objetivo de probar todos los endpoints definidos en la API. De está manera, se comprobará de que la API funciona correctamente. La colección con las diferentes llamadas se puede observer en la siguiente figura 5.7:

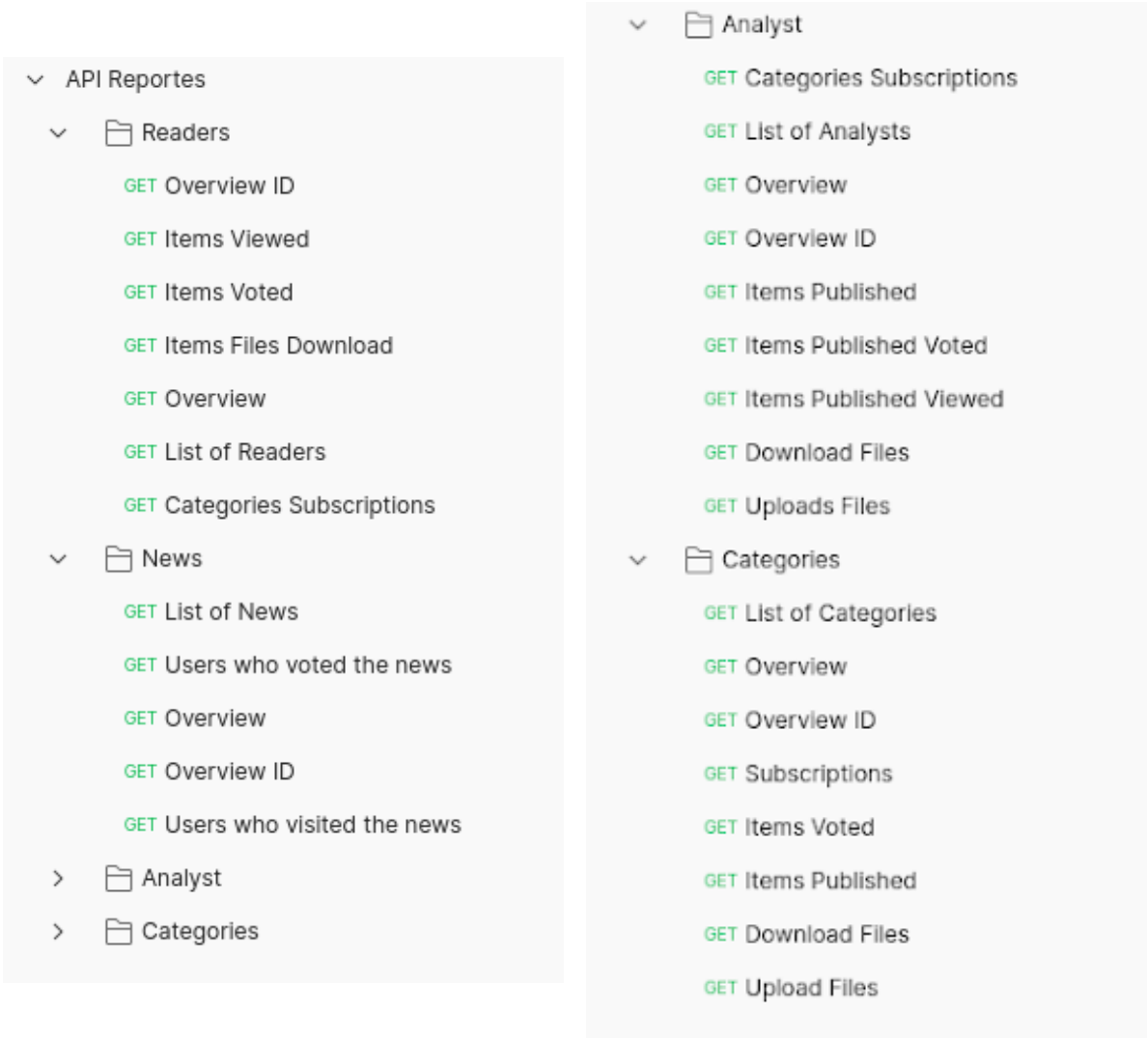

Figura 5.7: Colección de llamadas en Postman Fuente: Elaboración propia

Cada una de las llamadas de la colección hará la solicitud a un endpoint de la API diferente. De esta manera, se prueba que todas devuelven la respuesta deseada y que funcionan correctamente.

Por otro lado, a la hora de definir el endpoint, se ha hecho uso de una variable de entorno denominada como base_url. De esta manera, podemos cambiar las pruebas de los endpoint en local y en la nube, cambiando solamente el valor de la variable. Para el entorno local, por ejemplo, el valor será 'localhost:5000'.

Figura 5.8: Llamada al endpoint realizada desde Postman Fuente: Elaboración propia

Y en el caso de este endpoint, que sería para recoger simplemente el listado de analistas de la aplicación, la respuesta devuelta es el siguiente JSON:


```
{
    "data": [
        {
            "analyst": "analyst",
            "email": "analyst@gmail.com",
            "id": 66
        }
    ],
    "name": "Lista de analistas",
    "numberOfItems": 1
}
```


Figura 5.9: Respuesta de la llamada en Postman Fuente: Elaboración propia

5.3. Business Intelligence

Como se ha mencionado en el marco tecnológico, las herramientas de Business Intelligence son programas con los cuales las empresas pueden recopilar, organizar y analizar datos de diversas fuentes diferentes. Permitiendo presentar estos datos de una manera gráfica y fácil de entender a través de gráficos, informes y tableros de control. De esta manera, ayudan a las empresas a poder hacer un análisis más eficaz de los datos permitiéndoles tomar decisiones basadas en datos sólidos en lugar de suposiciones o intuiciones.

Antes de empezar a hacer uso de estas herramientas, en este caso, Power BI de Microsoft. Es importante que la API ya sea funcional, porque los datos que se van a mostrar en Power BI, se van a recoger haciendo peticiones a esta API. Y por otro lado, para que la API funcione, es necesario que la aplicación de Innguma tenga todos los cambios implementados y que la Base de Datos guarde los datos en las nuevas tablas. Es decir, la creación de informes mediante Power BI es el último paso para poder completar el proyecto exitosamente.

Comentar, que el informe que se ha creado con sus respectivas tablas, gráficos, etc. Es un ejemplo de los diferentes datos que pueden recoger y visualizar las terceras empresas que tengan acceso a la API. Algunos de los datos visualizados son los siguientes:

Primero se ha creado una tabla donde se podrá ver el listado completo de los usuarios dentro del entorno de la aplicación de Innguma. Donde se mostrará el id, el email y el nombre de cada usuario. Dicha tabla se puede observar en la figura 5.10:

ID	Email	Usuario
63	__cronidkr__@ideko.es	cronidkr
62	admin@innguma.com	Administrator
67	afernandez@ideko.es	Aitor
66	analyst@gmail.com	analyst
71	damazmela@innguma.com	Danel Mazmela
65	dani@dani.com	dani
70	enarza@innguma.com	Eneko Arza
68	imramirez@innguma.com	Imobach Ramírez
69	jokarriki@innguma.com	Joseba Karriki
64	user@innguma.com	user

Figura 5.10: Tabla con el listado de los usuarios Fuente: Elaboración propia

Por otro lado, se ha creado un gráfico de barras apiladas donde se puede observar el número total de accesos que cada usuario ha realizado dentro de la aplicación de Innguma. En el eje Y se mostrará el nombre de cada usuario y en el eje X se mostrará el número de accesos a la plataforma. Los datos se mostrarán de una manera descendente; es decir, primero se mostrará los usuarios con más accesos. El gráfico se puede observar en la figura 5.11:

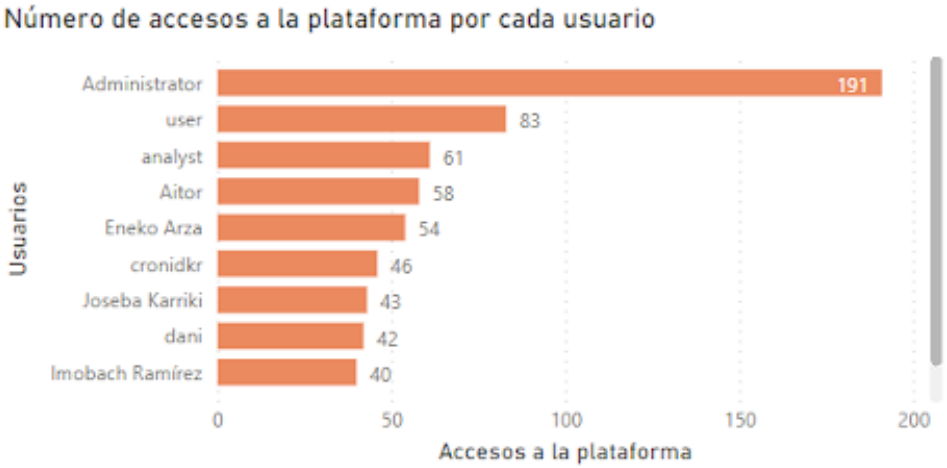

Figura 5.11: Gráfico con el número de accesos a la plataforma por cada usuario Fuente: Elaboración propia

El siguiente es un gráfico de anillos, donde se podrá observar el número de noticias que ha visitado cada usuario. Gracias a este gráfico, se puede ver de una manera sencilla el porcen-

taje de noticias que ha visitado cada usuario en comparación con los demás y determinar qué usuario es más activo dentro de la aplicación de Innguma. El gráfico se puede observar en la figura 5.12:

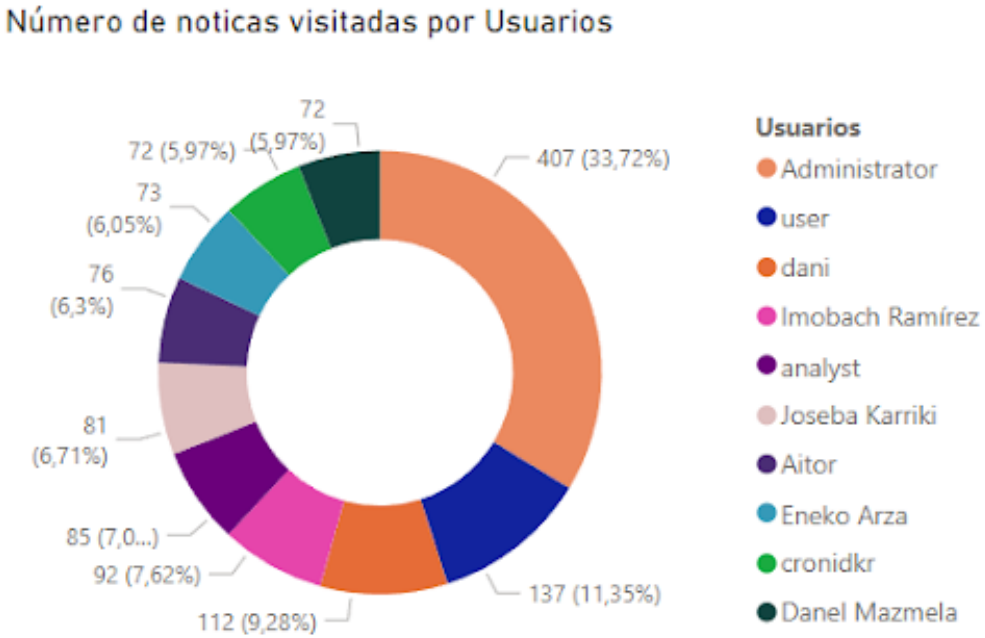

Figura 5.12: Gráfico con el número noticias visitadas por cada usuario Fuente: Elaboración propia

También se puede mostrar información específica de cada usuario. En este caso se está mostrando mediante tarjetas la fecha de la última noticia votada y la fecha de la última noticia visitada por parte del usuario Administrator, como se puede observar en la figura 5.13:

Figura 5.13: Tarjetas con la información específica del usuario Administrator Fuente: Elaboración propia

Por último, Power BI cuenta con una serie de filtros, que se denomina como segmentación

de datos. Gracias a esta funcionalidad, se puede editar la información que se va a visualizar, para que se muestran los datos que cumplan los filtros establecidos. En este caso se ha añadido un filtro por nombre de usuario, donde se puede seleccionar el usuario del cual se quiere visualizar la información. En es caso se ha elegido Administrator como se puede observar en la figura 5.14:

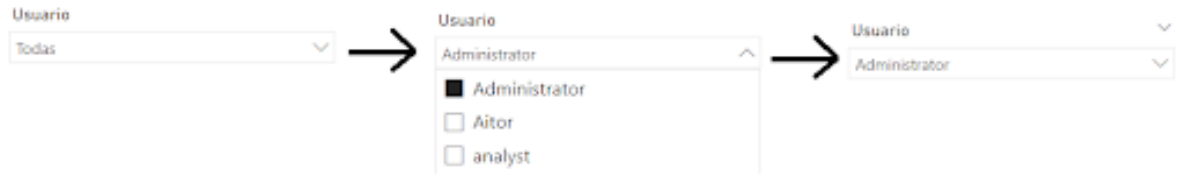

Figura 5.14: Segmentación de datos por usuarios Fuente: Elaboración propia

De esta manera los demás gráficos del informe se actualizarán con los datos, para solo mostrar la información correspondiente del usuario seleccionado. En el caso del gráfico de Número de accesos a la plataforma por cada usuario y del gráfico Número de noticias visitadas por Usuarios se actualizarán con solamente la información del usuario Adminstrator como se puede observar en la figuras 5.15 y 5.16

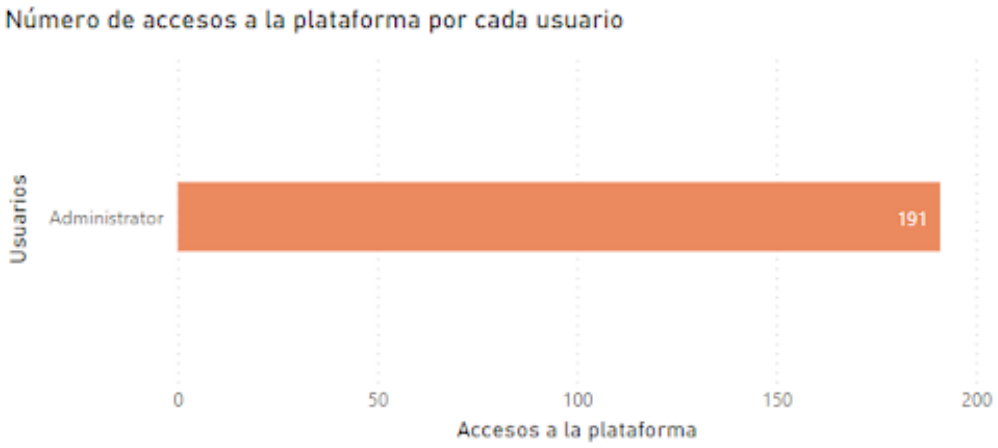

Figura 5.15: Gráfico con el número de accesos a la plataforma por Administrator Fuente: Elaboración propia

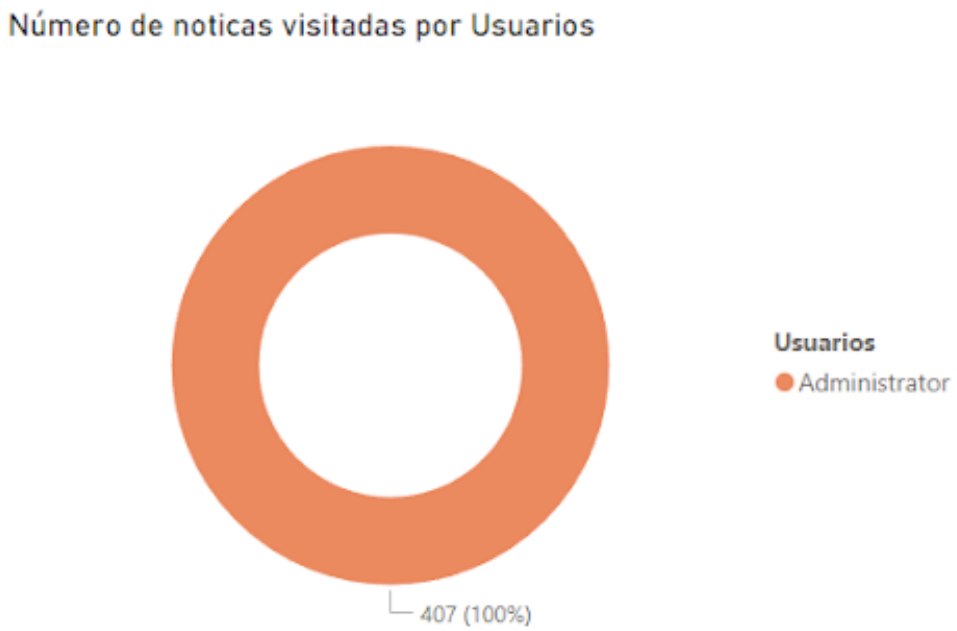

Figura 5.16: Gráfico con el número noticias visitadas por Administrator Fuente: Elaboración propia

Capítulo 6

Conclusiones

En cuanto a las conclusiones del proyecto, se han conseguido cumplir los objetivos marcados en el inicio del proyecto, teniendo en cuenta el tiempo limitado con el que se contaba. Ya que se han desarrollado funcionalidades diferentes dentro de la arquitectura del sistema vista en la sección 4.4. Por un lado, se ha estado trabajando con una Base de Datos, para la creación de tablas, la manipulación de los datos y la obtención de estos. Además, ha habido un desarrollo dentro de la aplicación de Innguma, la cual ha conllevado a programar en la parte del servidor y en la parte del Front-End; haciendo uso de diferentes lenguajes. También se ha creado una aplicación independiente haciendo uso de Flask y de Python, creando una API completamente funcional. Y por último se ha hecho uso de herramientas de Business Intelligence para la creación de reportes, y poder visualizar los datos de una manera rápida y eficiente, haciendo peticiones a la API anteriormente creada.

El proyecto ha sido útil para la propia empresa de Innguma, ya que cumplía con la necesidades de la misma. Como se ha comentado en la sección 1.3, en las motivaciones del proyecto, la información de la aplicación no estaba bien estructurada, y era muy complicado hacer un uso eficiente de esos datos; por ello la necesidad de la reestructuración de las tablas de la Base de Datos y la creación de la API. Por otro lado, también era necesario recoger el feedback de los usuarios dentro de la aplicación de una manera más eficiente y visual; por ello, la creación del nuevo sistema de votaciones de las noticias que se publican en Innguma.

También cabe destacar que el proyecto que se ha realizado, está vinculado completamente a las asignaturas impartidas en el transcurso del máster. Como pueden ser las asignaturas: Ingeniería de software, Bases de Datos, Desarrollo de aplicaciones Web en el lado del cliente (Front-End) y en el lado del servidor (Back-End), Análisis de datos web y Computación en la nube.

Capítulo 7

Trabajos futuros

Aunque es cierto que todo los desarrollado ya es completamente funcional, hay varios desarrollos que han faltado por implementar debido a la falta de tiempo. Por otro lado, a medida que se iba desarrollando, la empresa ha observado nuevas funcionalidades que se podrían añadir, que no habían sido contempladas a la hora de definir el proyecto. Aun así, el estado actual del proyecto ha cumplido exitosamente con los objetivos marcados en el inicio de la misma.

A continuación se detallará las nuevas funcionalidades que se quieren implementar y la nuevas necesidades que han surgido con el transcurso del proyecto:

- **Nuevos endpoints de la API:** Como ya se ha comentado, a medida que el proyecto iba avanzando, la empresa ha observado la necesidad de añadir nuevos endpoints. Esta necesidad se ha identificado en una fase bastante avanzada del desarrollo; por lo que se ha decidido terminar el proyecto con los endpoints que se han definido al inicio. Y una vez habiendo concluido el máster, se añadirán los nuevos endpoint que hacen falta.
- **Despliegue de la API en AWS:** Actualmente la API cuenta con su versión en local; es decir, aún no está puesta en marcha en los servicios de AWS. Esto se debe a que como aún falta por implementar nuevos endpoint, no se ha visto la necesidad de hacer el despliegue todavía. Ya que tener una aplicación en marcha en los servicios de AWS cuesta dinero; y para tener una aplicación que no está completamente terminada, no se ha visto necesario hacer dicho despliegue. Si es cierto, que como el entorno local ya está integrado y configurado con el framework de Serverless, para el despliegue de la API solamente es necesario la ejecución de varios comandos, por lo que la mayor parte del trabajo ya está hecho.
- **Implementación de caché:** Aunque se ha definido la lógica de la caché y cómo debe funcionar dentro de la aplicación API, no se ha llegado a implementar este tipo de memoria por falta de tiempo. Gracias a esta funcionalidad, aumentaremos la velocidad de las solicitudes que realicen los usuarios y disminuimos la carga de trabajo de la Base de Datos. Por lo que en un futuro, va a ser importante hacer la implementación de la memoria caché para mejorar la eficiencia de la API.

- **Usar secrets de AWS:** Para guardar las credenciales de acceso a la Base de Datos de una forma segura, se quiere hacer uso del servicio de Amazon de AWS secrets. Esto nos permite almacenar este tipo de información de una manera segura.
- **Creación de gráficos en Front-End:** Actualmente solo se consume la API con el uso de herramientas de Business Intelligence, como puede ser la aplicación de Power BI de Amazon. La nueva funcionalidad que se quiere crear consiste en un nuevo apartado dentro de la propia aplicación de Innguma, para mostrar a los usuarios los datos que se pueden obtener de la API de una manera sencilla y visual. Mediante el uso de gráficos, tablas, etc.

Como se puede observar, las mejoras que se le pueden implementar a la arquitectura del sistema creado son muy amplias. A medida de que se van desarrollando estas nuevas funcionalidades, lo más seguro es que vayan surgiendo nuevas ideas y necesidades que mejorarán el sistema.

Apéndice A

Atributos de las tablas de la Base de Datos

En el siguiente apéndice se mostrará de una forma detallada la diferentes tablas que se han creado para la nueva estructura de la Base de Datos y los atributos de cada una:

- jos_user_file_downloads: Una tabla con el objetivo de guardar datos sobre la relación entre un usuario y que adjunto ha descargado. Los atributos que tiene son:
  - itemID: ID de la noticia a la que pertenece el adjunto
  - userID: ID del usuario que ha descargado el adjunto
  - fileID: ID del adjunto
  - type: El tipo de la noticia a la que pertenece el adjunto
  - timestamp: Determina la hora y el dia en el que se ha descargado el adjunto
- jos_user_items_visits: La tabla guardará las visitas que hacen los usuarios a las noticias publicadas en la plataforma, con los siguientes atributos:
  - itemID: ID único para identificar la noticia a la que se ha realizado la visita
  - userID: ID del usuario que ha realizado la visita
  - type: El tipo de la noticia
  - timestamp: Determina la hora y el dia en el que se ha realizado la visita
- jos_k2_items: Tabla donde se guardan todas las noticias que se publican en la aplicación. Con los siguientes atributos principales:
  - title: Título descriptivo de la noticia
  - created_by: Usuario que ha subido la noticia a la plataforma
  - created_at: Fecha en la que se ha subido la noticia a la plataforma
  - catid: ID de la categoría a la que pertenece la noticia
- jos_k2_categories: Tabla donde se guarda el listado completo de las diferentes categoría de la aplicación, con estos atributos principales:
  - id: ID único que identifica la categoría

  - name: Nombre de la categoría
  - alias: Sobrenombre por el cual es conocida la categoría
  - parent: ID de la categoría padre a la que pertenece (si que la tiene)

- jos_k2_attachments: Tabla donde se guardan todos los adjuntos que se suben dentro de las noticas en la plataforma de Innguma, el cual tiene los siguientes atributos:
  - id: ID único que identifica el adjunto
  - filename: Nombre del adjunto
  - title: Título descriptivo del adjunto
  - hits: Número de veces que se ha descargado el adjunto
  - userID: ID del usuario que ha subido el adjunto a la plataforma
- jos_users: Tabla donde se guardan todos los datos relacionados con los usuario de la plataforma. Cuenta con una gran cantidad de atributos, pero estos se podrían considerar los principales:
  - name: Nombre completo del usuario (Nombre y apellidos)
  - username: Apodo con el que se le conoce dentro de la plataforma
  - email: Email único que se utiliza para el inicio de sesión
  - password: Contraseña encriptada del usuario
- jos_innguma_rating: Nueva tabla donde se guardarán las valoraciones que hacen los usuarios a las noticias de la plataforma. Con los siguiente atributos:
  - itemID: ID de la noticia
  - userID: ID del usuario que ha votado la noticia
  - rating: Valoración otorgada por el usuario, entre 1 y 5
  - timestamp: Determina la hora y el dia en el que se ha realizado la valoración
  - modified_at: En caso de editar la valoración se guardará la fecha y la hora que se ha realizado la modificación
  - type: Tipo de la noticia

Apéndice B

Recursos de la API

En el siguiente apéndice, se podrá observar de una forma más detallada los recursos que se han definido de la API, y todo la información que se puede recoger de cada uno de ellos:

- Lectores: Son los usuarios principales. Acceden a la aplicación para leer las noticias publicadas por los analistas. De esta manera, estarán informados en todo momento. Se recogerá esta información:
  - Nombre y apellidos
  - Correo
  - Última fecha de conexión
  - Última noticia leída
  - Total de noticias vistas
  - Sus votaciones a noticias
  - Última votación
  - Noticia más vista
  - Categoría más vista
  - Categorías suscritas
  - Número de documentos descargados
- Analistas: Son los usuarios encargados de realizar la vigilancia tecnológica y la inteligencia competitiva. Es decir, son los que recogen la información relevante y la publican para que los miembros de su organización (los lectores) estén enterados de los que sea necesario. Por otro lado, los analistas también pueden ejercer la función de lectores y leer las noticias que publican los demás analistas. La API devolverá la siguiente información sobre los analistas:
  - Nombre y apellidos
  - Correo
  - Fecha de última conexión
  - Acceso totales a la plataforma
  - Fecha de la última publicación realizada

- Última noticia publicada
- Última noticia leída
- Total de noticias subidas
- Total de votaciones realizadas
- Última votación realizada
- Noticia con más visitas
- Categoría más recurrente a la hora de publicar
- Categoría más vista
- Categorías suscritas
- Número de documentos suscritos
- Número de documentos subidos

- Noticias: Las noticias son los ítems que han sido seleccionados por los analistas y han sido publicados para que los lectores las puedan consumir. Se podría considerar la información relevante que se ha conseguido al realizar la vigilancia tecnológica. Los datos devueltos por la API son los siguientes:

  - Título de la noticia
  - Fecha de publicación
  - Fecha de la última lectura
  - Analista que ha subido la noticia
  - Número de visitas
  - Votos recibidos
  - Categoría a la que pertenece
  - Etiquetas de la noticia
  - Número de veces que se ha descargado el documento adjunto
  - Los adjuntos de la noticia
  - Comentarios del analista
  - Los campos extra
  - Comentarios de los lectores
  - Valoración de la noticia

- Categorías: Las categorías se podrían considerar las carpetas donde los analistas publican las noticias. Los lectores, se pueden suscribir a estas y cada vez que se publique una noticia, podrán leerla. Se podrá recoger los siguientes datos de la API:

  - Nombre de la categoría
  - Fecha de creación
  - Fecha de última publicación dentro de la categoría
  - Fecha de última lectura dentro de la categoría
  - Número de visitas

- Número de votos
- Última votación
- Número de noticias publicadas
- Noticia con más visitas
- Número de lectores suscritos
- Número de documentos descargados
- Número de documentos subidos

Referencias

Anabel, E. (2023). *¿qué es docker?* Descargado de https://dominicode.com/que-es-docker/

Atlassian. (s.f.). *Qué es git.* Descargado de https://www.atlassian.com/es/git/tutorials/what-is-git

Diego, V. (2023). *¿qué es un entorno de desarrollo y en qué se diferencia de un entorno de desarrollo integrado (ide)?* Descargado de https://www.hostinger.es/tutoriales/que-es-un-entorno-de-desarrollo

Donetonic. (s.f.). *Metodología waterfall vs metodología agile.* Descargado de https://www.google.com/search?client=firefox-b-d&q=Se+basa+en+una+planificaci%C3%B3n+exhaustiva+y+un+enfoque+secuen-+cial

IONOS, D. G. (s.f.). *Bases de datos: qué tipos hay y para qué se usan.* Descargado de https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos/

Lucidchart. (s.f.-a). *Qué es un diagrama entidad-relación.* Descargado de https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

Lucidchart. (s.f.-b). *Tutorial de diagrama de secuencia uml.* Descargado de https://www.lucidchart.com/pages/es/diagrama-de-secuencia

Martins, J. (2022). *Diagrama de gantt: qué es y cómo crear uno con ejemplos.* Descargado de https://asana.com/es/resources/gantt-chart-basics

Mozilla. (s.f.). *Códigos de estado de respuesta http.* Descargado de https://developer.mozilla.org/es/docs/Web/HTTP/Status

MulesSoft. (s.f.). *¿qué es una api? (interfaz de programación de aplicaciones).* Descargado de https://www.mulesoft.com/es/resources/api/what-is-an-api

Muradas, Y. (2019). *Qué es postman y primeros pasos.* Descargado de https://openwebinars.net/blog/que-es-postman/

Poddar, R. (2022a). *What is a jwt? understanding json web tokens.* Descargado de https://supertokens.com/blog/what-is-jwt

Poddar, R. (2022b). *What is a jwt? understanding json web tokens.* Descargado de https://supertokens.com/blog/what-is-jwt

Rehkopf, M. (s.f.). *Historias de usuario con ejemplos y plantilla.* Descargado de https://www.atlassian.com/es/agile/project-management/user-stories

Signaturit. (2021). *¿qué es business intelligence (bi) y qué herramientas existen?* Descargado de https://blog.signaturit.com/es/que-es-business-intelligence-bi-y-que-herramientas-existen

Walsh, D. (2023). *Github vs. gitlab: diferencias, similitudes y usos.* Descargado de https://blog.hubspot.es/website/github-gitlab

Webempresa. (s.f.). *¿qué es joomla?* Descargado de https://www.webempresa.com/joomla.html