UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Freenalytics – Servicio de recopilación y analíticas de uso con soporte para cualquier tipo de aplicación para un almacenamiento y aprovechamiento de datos confiable

Christian Eduardo López Argüello

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 20 de diciembre de 2022

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Freenalytics – Servicio de recopilación y analíticas de uso con soporte para cualquier tipo de aplicación para un almacenamiento y aprovechamiento de datos confiable

Christian Eduardo López Argüello

Nombre del profesor, Título académico **Daniel Fellig Goldvechmiedt, MSc.**

Quito, 20 de diciembre de 2022

© DERECHOS DE AUTOR

Nombres y apellidos: Christian Eduardo López Argüello

Código: 200613

Cédula de identidad: 1718342098

Lugar y fecha: Quito, 20 de diciembre de 2022

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Hoy en día, capturar los datos de uso de usuarios de cualquier tipo de plataforma virtual se ha convertido en una necesidad y no en un lujo. La analítica de datos es una herramienta sumamente útil que los desarrolladores y administradores de proyectos aprovechan para conocer a sus usuarios, saber cómo ellos interactúan con sus aplicaciones y cómo mejorar la experiencia de usuario.

Existen varias herramientas de analítica en el mercado como Google Analytics, Bitly u Hotjar, pero estas herramientas están limitadas solo a aplicaciones Web. Además, estas plataformas no son de fuente abierta, por lo que se desconoce el uso que las empresas detrás de estas puedan dar a los datos de sus usuarios. Esto puede provocar un sentimiento de desconfianza de las partes interesadas de cualquier proyecto que necesiten sus servicios.

Por ende, en este proyecto, se propone una solución genérica, open source y que pueda ser alojada on-premises para cualquier tipo de aplicación, que requiera de analíticas de uso y para quien necesite del control total de sus datos.

Palabras clave: Analíticas de uso, fuente abierta, en-premisas, servicio web, herramienta de desarrollo de software.

ABSTRACT

Nowadays, capturing the usage data of the users of any type of virtual platform has become a necessity and not a luxury. Usage analytics are incredibly useful tools that software developers and project managers utilize to learn about their users, to know how they interact with their applications and how to improve their user experience.

There are tools out there on the market such as Google Analytics, Bitly or Hotjar, but they're limited only for Web applications. Additionally, since these platforms are not open source, it is impossible to know for sure what use may the companies behind these products be giving to the data of their users. This can provoke distrust in the stakeholders of any project that need such services.

Hence, in this project, a generic and open-source solution is proposed. One that can be hosted on-premises for any type of application that requires usage analytics and for people who need total control over their data.

Key words: Usage analytics, open-source, on-premises, web service, software development tool.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

No table of figures entries found.

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La captura de datos de uso de aplicaciones es una herramienta sumamente importante para los desarrolladores, administradores de proyectos, diseñadores y demás partes involucradas en el desarrollo de una aplicación. Existen varios tipos de datos que los ingenieros pueden captar sobre cómo sus usuarios interactúan con sus aplicaciones como: clics, duración de la sesión, demográfica, errores, entre otros. Capturar estos datos permite a los desarrolladores tomar mejores decisiones en cuanto al diseño de su aplicación, así, mejorando la calidad de su servicio y la experiencia de sus usuarios.

Actualmente en el mercado existen múltiples alternativas de servicios y plataformas que permiten la captura de estos tipos de datos, servicios como Google Analytics, Hotjar, Bitly y muchos otros más. Aunque por lo general suelen ser limitadas a aplicaciones tipo Web o móvil. No obstante, cada uno de ellos tienen sus pros y sus contras que pueden representar factores decisivos al momento de su uso por parte de desarrolladores de Software. Algunos de estos servicios son de uso gratuito, aunque por lo general existen términos y condiciones que obligan al cliente a compartir estos datos con la empresa operadora del servicio como es el caso de Google Analytics según sus propios términos de uso. (Google, 2019)

Para los desarrolladores más preocupados por los datos capturados por sus aplicaciones, la utilización de una plataforma externa se convierte en un mal necesario, ya que entienden que el momento que sus datos entran en un servicio externo entonces dejan de ser sus datos porque ya no tienen el control total sobre ellos. Una opción viable para combatir esta falta de confianza sería el uso de una plataforma alojada On-Premises (en las propias instalaciones), pues el equipo de desarrollo tendría total control sobre el almacenamiento de estos datos.

En este trabajo, se propone Freenalytics, una solución genérica, de fuente abierta, que puede ser alojada On-Premises y que pueda integrarse con cualquier tipo de aplicaciones sin ningún tipo de limitación.

Objetivos generales

1. Implementar una plataforma de analíticas de uso confiable y de fuente abierta para cualquier tipo de aplicación de Software.
2. Facilitar a los desarrolladores de Software la captura de datos de uso de sus usuarios y su aprovechamiento para aplicaciones que no puedan hacer uso de las plataformas de analíticas de uso ya existentes.

Objetivos específicos

1. Utilizar metodologías de desarrollo ágiles para implementar un proyecto de Software robusto.
2. Utilizar TDD o Test-Driven-Development y pipelines de CI/CD para asegurar una buena calidad de código y automatizar los procesos de publicación y despliegue.
3. Generar una comunidad de desarrollo de fuente abierta dispuesta a contribuir en el proyecto a futuro a través del uso de herramientas de código abierto o que impulsen los proyectos de fuente abierta como GitHub.

DESARROLLO

Terminología

Analíticas

La analítica en el Software se puede definir como el análisis de los datos de Software para los administradores e ingenieros con el propósito de empoderar el desarrollo de Software y sus equipos para ganar una mejor visión sobre sus datos y realizar mejores decisiones. (Menzies, Zimmermann, 2013)

La analítica de uso de una aplicación tiene como objetivo recuperar información del uso de una plataforma. Este tipo de información puede incluir demográfica de los usuarios, los dispositivos que utilizan, qué partes de la aplicación utilizan, los horarios más activos, etc…

Otro tipo de analítica de uso es Crashlytics, este tipo de recolección de datos tiene como objetivo entender los posibles errores que pueden existir en una aplicación y sus causas.

Desarrollo ágil

El desarrollo ágil abarca un proceso iterativo al momento de realizar el desarrollo de software. Esta metodología nace de la necesidad de adaptar el proceso de desarrollo de manera efectiva a los requerimientos cambiantes de los productos de software. (Ahmed et al., 2010)

Las prácticas del desarrollo ágil de basan de los 12 principios siguientes (Agile Alliance, s.f):

1. *La prioridad principal es la de satisfacer al cliente a través de la entrega continua de Software valioso.*
2. *Los procesos ágiles son abiertos a los cambios de requerimientos, aún cuando son en una etapa tardía del desarrollo en virtud de la ventaja de competitividad del cliente.*
3. *Entregar Software funcional por periodos semanales, con preferencia al rango de tiempo menor.*

4. *El personal de negocios y los desarrolladores deben trabajar en conjunto diariamente.*
5. *Construir proyectos con individuos motivados, brindándoles el ambiente y apoyo que necesitan y confiando en que harán un buen trabajo.*
6. *La manera más eficiente de transmitir información es través de una conversación cara a cara.*
7. *Software funcional es la métrica principal de desempeño.*
8. *Los procesos ágiles promueven el desarrollo sustentable. El ritmo de desarrollo del proyecto debe ser consistente indefinidamente.*
9. *Atención continua a la excelencia técnica y buen diseño mejoran la agilidad.*
10. *Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
11. *Las mejores arquitecturas, requerimientos y diseños emergen de equipos auto organizados.*
12. *En intervalos regulares, el equipo debe reflexionar sobre cómo ser más eficiente y ajusta su comportamiento acorde.*

Kanban

Kanban es un marco de trabajo utilizado para implementar un desarrollo ágil. El eje principal de este marco es la visualización de las tareas. Kanban se implementa a través del uso de un tablero que ofrezca una vista rápida de las tareas a realizar en sus respectivos estados (por hacer, en progreso, completado). (Radigan, s.f)

Test-Driven-Development (TDD)

TDD o Test-Driven-Development es una metodología de desarrollo de código limpio que se enfoca en la escritura de pruebas unitarias, previamente a la implementación del código, para así escribir una implementación mínima, modular y robusta. Esta metodología se basa en un algoritmo iterativo que asegura el desarrollo de software robusto (Beck, 2002):

1. Escribir una prueba que falle para una unidad de la implementación.
2. Escribir la implementación mínima que satisfaga la prueba.
3. Refactorizar la implementación y repetir.

Pruebas unitarias

Las pruebas unitarias en Software son pruebas que "prueban unidades individuales (módulos, funciones, clases) en aislamiento del resto del programa" (Beck, 2002). La automatización de la verificación del funcionamiento de las unidades del programa es la ventaja primordial de las pruebas unitarias, ya que estas permiten realizar cambios dentro del programa con confianza de que ningún otro módulo pueda fallar por consecuencia. El desarrolllo de pruebas unitarias es sumamente importante en el desarrollo de sistemas robustos y de alta calidad.

Continuous Integration (CI)

El Continuous Integration o integración continua, es una práctica que consta en la automatización de la integración de los cambios en el código. Sin una práctica apropiada de CI, los desarrolladores necesitan coordinar y comunicar manualmente cuando incluyen cambios en el código. A través del CI, los desarrolladores pueden integrar cambios de forma rápida y confiable. (Rehkopf, s.f)

Continuous Delivery (CD)

El Continuous Delivery o entrega continua, es una extensión del Continuous Integration, donde el objetivo es el de desplegar código de manera automática en un ambiente de pruebas y/o de

producción. A través de esta práctica, los desarrolladores pueden desplegar su código de manera rápida, permitiendo así despliegues pequeños y realizados en periodos cortos de tiempo. (Rehkopf, s.f)

Web Service

Un Web Service o servicio Web es una función de Software de máquina a máquina hosteada en la red en una ubicación direccionable. Un servicio Web expone una ocultando los detalles de la implementación, del software y hardware utilizado. Los servicios Web son utilizados para integrar varias aplicaciones o varias partes de una misma aplicación de una manera de bajo acoplamiento. (IBM, 2021)

API REST

Una API REST es una interfaz que sigue las reglas de la arquitectura REST. REST significa Representational State Transfer o transferencia de estado representacional. Una API REST es un servicio web que utiliza un formato de representación de datos como JSON (JavaScript Object Notation), XML (Extensible Markup Language), entre otros, para la transferencia de información a través del protocolo HTTP. Un punto importante de una API REST es que la comunicación entre cliente y servidor no guarda ningún tipo de estado entre peticiones, es decir, cada petición es independiente. Las APIs REST siguen reglas de diseño las cuales hacen uso de los métodos HTTP (GET, POST, DELETE, PUT, PATCH…) para definir el tipo de operación a realizar. (Red Hat, 2020)

Metodología de desarrollo

El desarrollo de este proyecto se realizó utilizando metodologías ágiles para asegurar que el proceso sea lo más eficiente y resiliente. Para esto, se utilizó la herramienta de seguimiento de

tareas Trello para implementar la metodología Kanban, con la cual se organizaron los quehaceres por prioridad, permitiendo así una clara definición y visualización del progreso en el desarrollo de la plataforma. Mediante el uso de estas herramientas, se llevó a cabo una recopilación de los requisitos de la plataforma para definir el alcance y permitir una planificación del cronograma de actividades, tal como recomiendan Haik y Shahin (2011) para llevar a cabo un trabajo adecuado de diseño de ingeniería.

En cuanto a la programación del proyecto, se utilizó la metodología de TDD o Test-Driven-Development para los componentes que aprovecharían de mejor manera de las pruebas unitarias. Esta práctica asegura que la calidad y la integridad del código se mantenga a lo largo del ciclo de desarrollo de la aplicación. Con toda la aplicación probada con pruebas unitarias, cualquier tipo de inserción o modificación de componentes en la plataforma, los potenciales errores asociados a estas operaciones se reducen considerablemente.

Continuous Integration / Continuous Delivery

Para asegurar un proceso robusto de pruebas y publicación de la plataforma, se optó por el uso de técnicas de Continuous Integration (CI) y Continuous Delivery (CD) a través de la herramienta propia de GitHub llamada GitHub Actions. Esta herramienta permite definir pipelines que se ejecutan por la acción de algún evento en el repositorio de GitHub como la apertura o cierre de un Pull Request (PR) o la subida de commits en alguna rama del repositorio. Estos pipelines son nada más una lista de pasos que un agente (en este caso una máquina virtual en las instalaciones de GitHub) seguirá al momento de darse un evento específico dentro del repositorio.

A través del uso de esta herramienta de DevOps, se pretende que la plataforma propuesta en este proyecto cumpla con los estándares de desarrollo de Software IEEE/ISO/IEC 32675-2021

que describen métodos y prácticas de DevOps para la construcción de sistemas confiables y robustos incluyendo su compilación, empaquetamiento y despliegue. (IEEE, 2022)

En el caso de la plataforma, se han definido pipelines de CI/CD para automatizar el proceso de integración, despliegue y publicación de los distintos componentes de todo el proyecto como son en este caso: la imagen de Docker de la plataforma, la librería de integración web y los sitios web de documentación de la plataforma y del API REST respectivamente.

Al ser este un proyecto con varios componentes independientes se ha decidido de separar estos en repositorios distintos que hagan uso pipelines de CI/CD. A continuación, se presentan los pipelines que han sido definidos para los repositorios que hagan uso de esta herramienta.

1. Para el repositorio freenalytics/freenalytics (Plataforma)

En este repositorio existen tres pipelines que se ejecutan cuando existe un nuevo PR que modifique el código para el servidor o para el cliente web. Al ejecutarse, estos pipelines clonan el repositorio dentro del agente, se configura el ambiente de Node.js con la versión 16.15.1, se instalan las dependencias del proyecto, se ejecuta el linter (que asegura un estilo consistente en el código) y se ejecutan las pruebas unitarias.

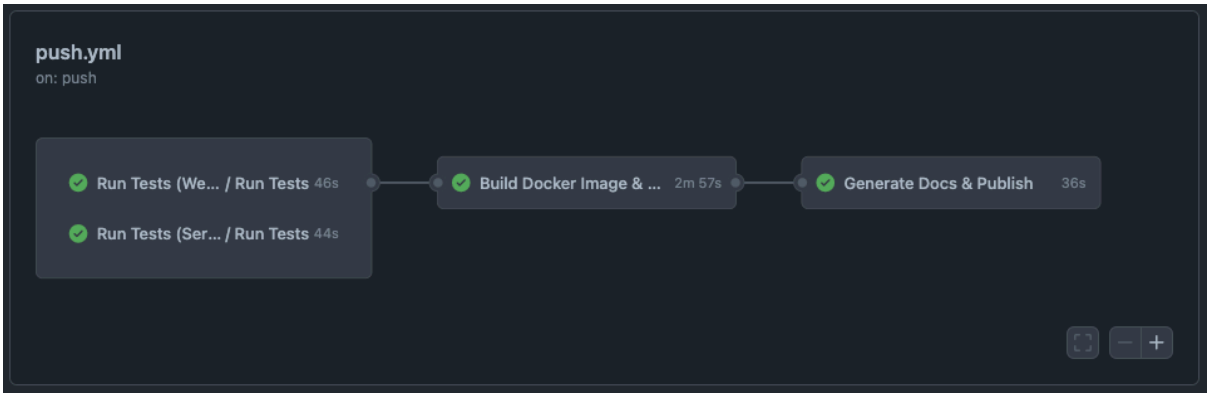


Figura 1 Pipeline de CI/CD para el evento Push del repositorio de la plataforma

En el caso del último pipeline, este se ejecuta con el evento Push en la rama “master”, la cual es la rama principal del proyecto. En este caso, por cada push a esta rama, se ejecuta el linter y

las pruebas. En caso de que las pruebas sean exitosas, el pipeline continua, compilando el proyecto y generando una imagen de Docker, la cual al final es publicada en el registro de imágenes de Docker de GitHub Packages. Finalmente, se genera la documentación del API REST y se publica en el repositorio freenalytics/api-docs el cual contiene el código de esta página web.

2. Para el repositorio freenalytics/freenalytics-connector-web (Librería de integración web)

En este repositorio existen dos pipelines, uno que inicia por el evento de nuevo PR y otro por el evento Push. En el caso del pipeline de PR, funciona de manera similar al anterior, es decir: se clona el repositorio dentro del agente, se configura el ambiente de Node.js con la versión 16.15.1, se instalan las dependencias de la librería, se ejecuta el linter y las pruebas unitarias.

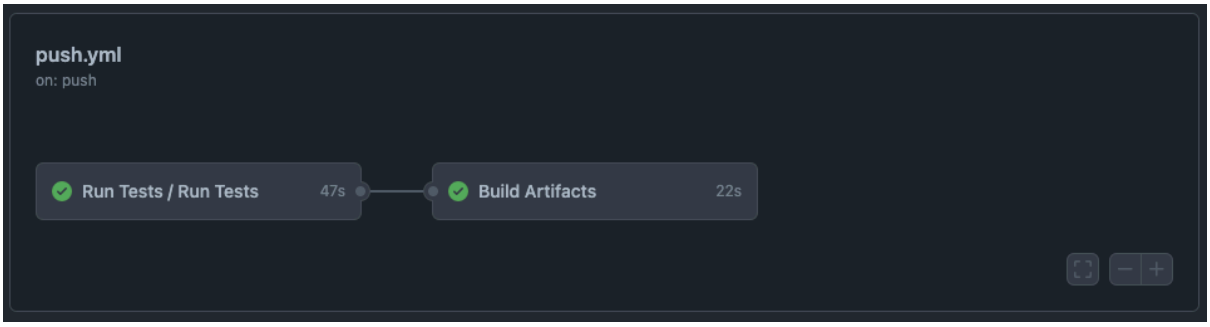


Figura 2 Pipeline de CI/CD para el evento Push del repositorio de la librería de integración web

En cuanto al pipeline del evento Push, se ejecutan las pruebas del linter y las pruebas unitarias. En caso de que estas sean exitosas, se compilan los artefactos de la librería y se las publican en la rama dist del repositorio para su uso en ambientes de producción.

3. Para el repositorio freenalytics/freenalytics.github.io (Documentación de la plataforma)

Finalmente, el repositorio de documentación posee solamente un pipeline que se ejecuta por el evento Push. En este caso, no se hace uso del evento PR porque en este repositorio se realizan

subidas de los commits directamente. No hay necesidad para pruebas unitarias dado a que este repositorio posee únicamente código en formato Markdown, es decir, texto simple.

Este pipeline realiza las siguientes tareas: se clona el repositorio dentro del agente, se configura el ambiente de Python con la versión 3.8.x, se instalan las dependencias del sitio de documentación y luego se compila y publica el sitio web.

Dado a que esta documentación está realizada con MkDocs, existe un proceso de "compilación" que transforma las páginas de Markdown en un sitio web con HTML estático.

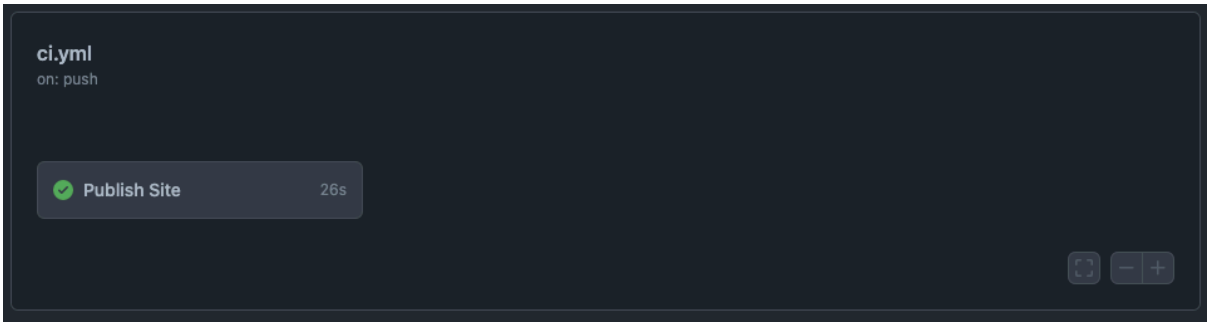


Figura 3 Pipeline de CI/CD para el evento Push del repositorio de la página de documentación de la plataforma

Diseño de la plataforma

La plataforma que se propone en este trabajo es un sistema de fuente abierta para la recopilación y visualización de datos de uso de aplicaciones de cualquier tipo que pueda ser fácilmente albergable on-premises o en la propia computadora de la persona que desee hacer eso de la plataforma.

Descripción

Freenalytics es un servicio que expone un API REST al desarrollador con el fin de integrar sus aplicaciones con la plataforma para captar cualquier tipo de información que él o ella desee almacenar. Estos datos pueden ser: dónde hace clic el usuario en una interfaz, mensajes de error que puedan suceder, número de personas que visitan un sitio web, y muchas otras cosas

más. Los datos que un desarrollador desea recuperar son definidos a través del uso de un lenguaje declarativo llamado JSON Schema (JSON Schema, s.f) que permite definir la estructura y el tipo de los datos que serán subidos y almacenados.

Una vez realizado esto, el desarrollador tendrá un espacio para su aplicación, que contiene un dashboard de visualizaciones de los datos que pueden actualizarse a intervalos definidos, una sección que le permite ver todos sus datos en forma de tabla, una función que le permite exportar todos sus datos en formato CSV y una sección de información que le informará los pasos que debe seguir para subir sus datos en su espacio en la plataforma.

Adicionalmente, la plataforma está disponible como una imagen de Docker lo cual permitirá facilidad de levantar este servicio en cualquier máquina junto con sus requerimientos, de tal forma que cualquier persona pueda tener su instancia propia y privada.

Arquitectura

La plataforma está realizada siguiendo una arquitectura MERN Stack, la cual define una arquitectura Full Stack para aplicaciones que hagan uso de la base de datos no relacional MongoDB. En esta arquitectura, se hace uso de una base de datos MongoDB, la cual está expuesta a través de un servicio web escrito en Node.js con el framework Express.js, el cual es consumido por un cliente realizado en React. (MongoDB, s.f)

La ventaja de esta arquitectura es el uso de un solo lenguaje para escribir el cliente y el servidor. Dado a que Node.js y React funcionan con JavaScript, es posible realizar una aplicación Full Stack con solo un lenguaje. Sin embargo, en este proyecto se optó por el uso de TypeScript, el cual es un superset de JavaScript (Microsoft. s.f). Es decir que, cualquier código de JavaScript es código válido de TypeScript.

TypeScript permite escribir JavaScript con tipos estáticos, lo cual ayuda con la reducción de errores potenciales que se pueden verificar en tiempo de transpilación en lugar de tiempo de

ejecución. Esto ahorra mucho tiempo al desarrollador porque las herramientas de desarrollo como los IDEs que se integran bien con TypeScript tienen acceso a un Intellisense mejorado y más preciso que permite guiar al desarrollador de mejor manera en su código.

La siguiente figura presenta de manera visual la arquitectura implementada en toda la plataforma:

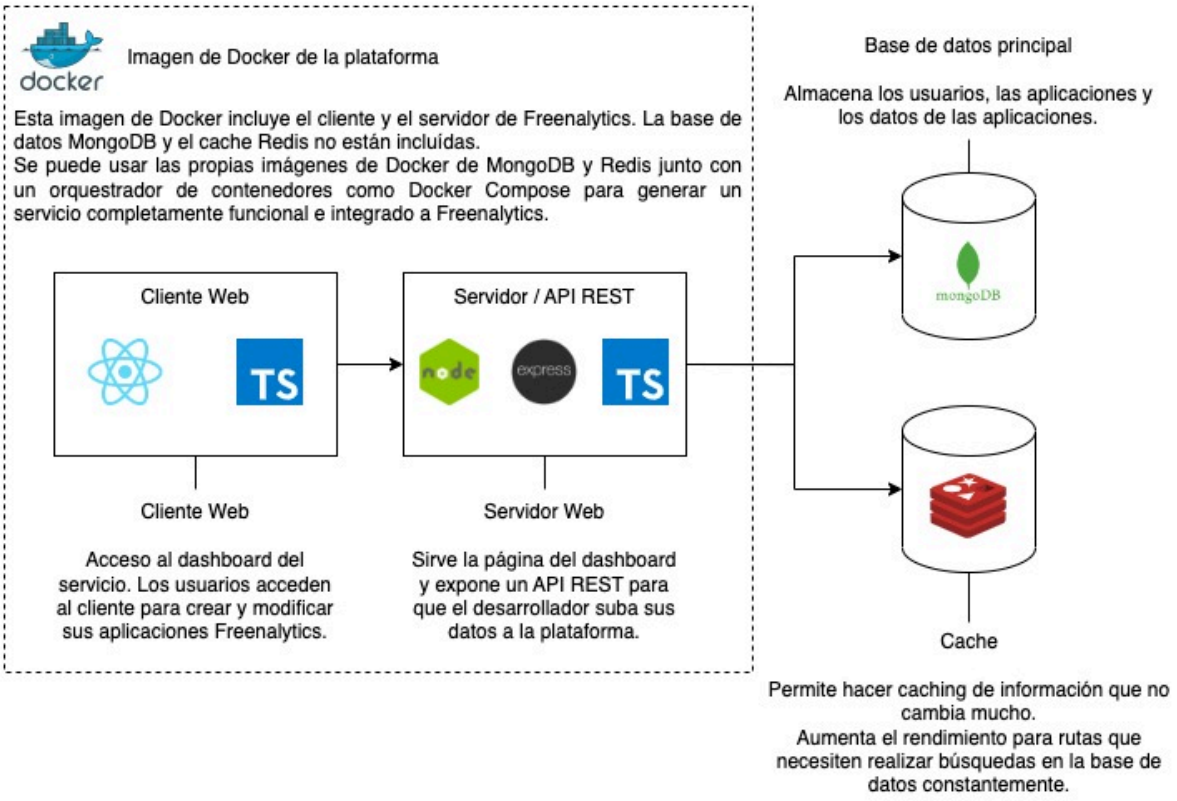


Figura 4 Arquitectura de la plataforma

La plataforma está compuesta por cuatro componentes: el servidor, el cliente Web, la base de datos principal (MongoDB) y el cache (Redis).

Servidor

Empezando por el servidor, este componente es un servicio web realizado en Node.js con el framework Express.js. Este servicio es el corazón de la plataforma. Aquí se expone un API REST que permite la creación de usuarios, el inicio de sesión, la creación, modificación y

eliminación de las aplicaciones para cada usuario, la subida y lectura de los datos de la aplicación, y la exportación de los datos en formato CSV.

Adicionalmente, el servidor utiliza la librería Passport.js la cual permite integrar autenticación en Node.js. Con esta librería se definen estrategias que permiten autenticar a los usuarios para así tener rutas protegidas en el API que requieran de algún tipo de mecanismo de autenticación. En este caso, se ha optado por autenticación a través de tokens JWT, los cuales se insertan en el header Authorization para cualquier tipo de petición HTTP. De esta manera, el momento que un usuario inicia sesión en la plataforma, el servidor le responde con un token JWT que lo identifica y le da acceso al resto de las rutas del API. Las credenciales del usuario están almacenadas en la propia base de datos MongoDB. En cuanto a la contraseña del usuario, esta está guardada como un hash junto a su propio salt para asegurar que nunca se guarde de forma insegura en forma de texto plano y asegurar la confidencialidad de sus credenciales.

Cliente Web

El cliente Web es una aplicación realizada en React la cual implementa una interfaz visual que permite al usuario interactuar con el servidor. Este cliente está realizado de igual manera en TypeScript, junto con el framework visual Bulma para la renderización de componentes estilizados y SASS como lenguaje de estilos, el cual reemplaza a CSS incluyendo funcionalidades adicionales para acelerar el desarrollo de hojas de estilo.

Desde aquí, el usuario puede registrarse en la plataforma (si el registro de usuarios está habilitado), administar sus aplicaciones y visualizar y exportar sus datos.

Base de datos MongoDB

La base de datos de la plataforma es una base de datos no relacional basada en MongoDB. Se optó por el uso de una base de datos no relacional dada la naturaleza de los datos que los desarrolladores subirán a la plataforma. Como no existe una predefinición de la estructura de

los datos, no hay manera de almacenarlos en una base de datos relacional. Se ha optado por MongoDB por su madurez en el campo de bases de datos no relacionales y por su facilidad de albergar a través de Docker.

Esta base de datos contiene toda la información de la plataforma; desde usuarios, aplicaciones y datos de cada aplicación.

Cache Redis

Al momento de que un usuario crea una aplicación en la plataforma, debe de incluir un esquema de la estructura de los datos que planea subir a la plataforma. Este esquema escrito en JSON Schema no solo sirve para estructurar los datos sino también para validarlos. Esto quiere decir que por cada vez que se suban datos a la plataforma, el servidor debe de recuperar el esquema de la aplicación y validar los datos subidos contra el esquema, así evitando que se suban datos mal estructurados a la base de datos.

El problema de este requerimiento es que, por cada vez que se suban los datos, el servidor deberá recuperar el esquema de la base de datos, lo cual puede ser una operación un poco lenta, considerando que una aplicación puede subir muchos datos en poco tiempo.

Dado a que el esquema es único y no modificable para cada aplicación, almacenarlo en un cache resulta ser una excelente solución que acelerará el proceso de búsqueda del cache.

Con esto en mente, se optó por el uso de Redis, un servicio de cacheo que almacena información asociada a una llave en memoria RAM. Esto hace que las peticiones a Redis sean sumamente eficientes.

Para dar un ejemplo, en el caso de que la búsqueda del esquema en la base de datos MongoDB tome 100ms, al ser almacenada en Redis, su recuperación puede llegar a ser de hasta 1ms o incluso menos. Esto garantiza un enorme beneficio en términos de tiempo y rendimiento de la plataforma.

Otra ventaja de este servicio de cache es que también existe una imagen oficial para Docker, permitiendo un fácil despliegue junto a la plataforma.

Casos de uso

Se ha identificado que la plataforma tendrá un único actor que será el usuario o desarrollador que haga uso de la plataforma. Las maneras de interactuar con la plataforma se describen a continuación a través del diagrama de casos de uso siguiente:

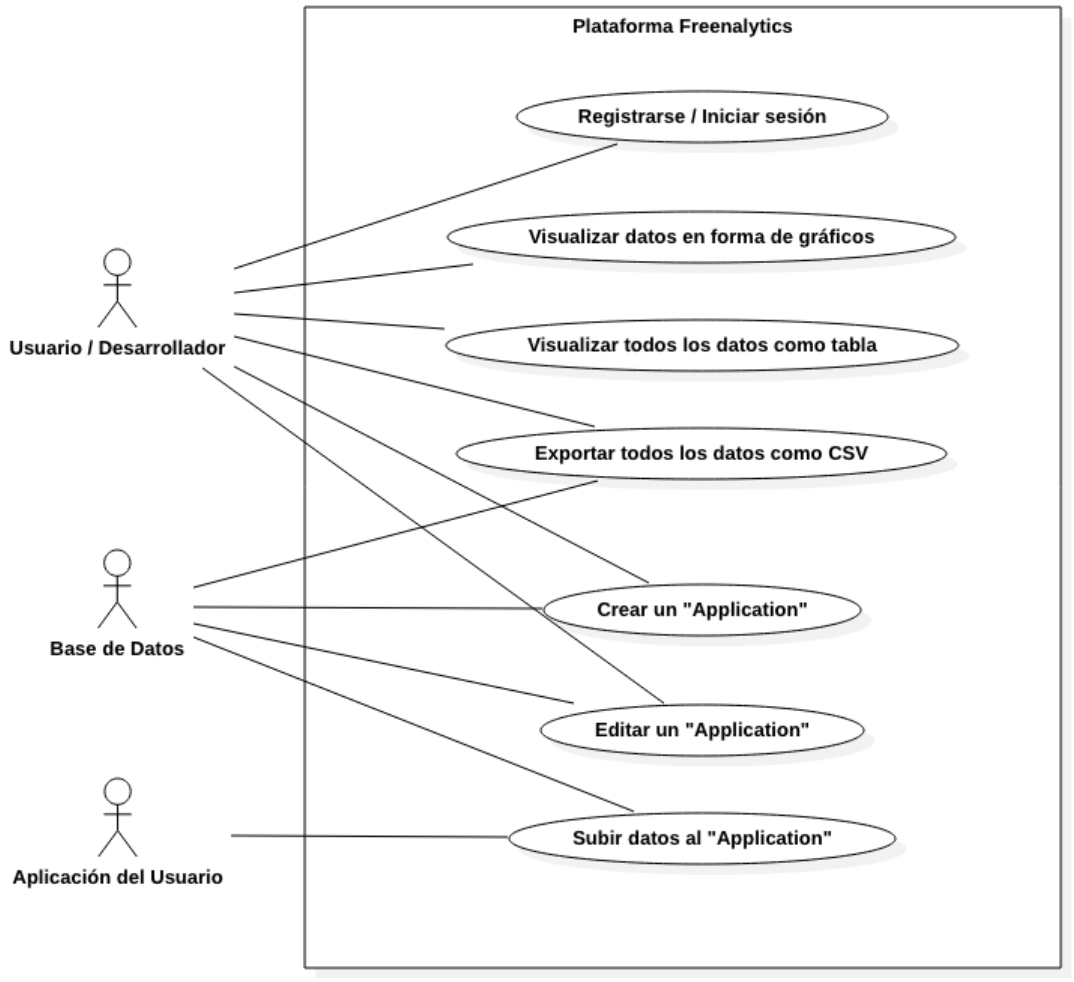


Figura 5 Casos de uso de la plataforma para un desarrollador

Típicamente, el usuario podrá registrarse o iniciar su sesión en la plataforma para poder interactuar con ella. Una vez dentro, tiene la opción de crear una nueva aplicación o editar cualquiera de sus aplicaciones ya existentes. Dentro del espacio de cada aplicación, el usuario

puede visualizar sus datos en forma de gráficos a manera de un Dashboard, visualizar las entradas de los datos en forma de tabla y exportarlos en formato CSV. Finalmente, a través del API REST, el usuario podrá subir su información desde su propia aplicación.

Flujo de interacción

La principal interacción que existe entre el usuario y la aplicación es la subida de datos. Por lo general, esta interacción se realiza programáticamente desde la aplicación del usuario como reacción a algún tipo de evento dentro de la misma. Para subir datos a la plataforma, la aplicación realiza una llamada HTTP POST a la ruta asignada a la aplicación con un Request Body de tipo application/json con los datos que se desean subir.

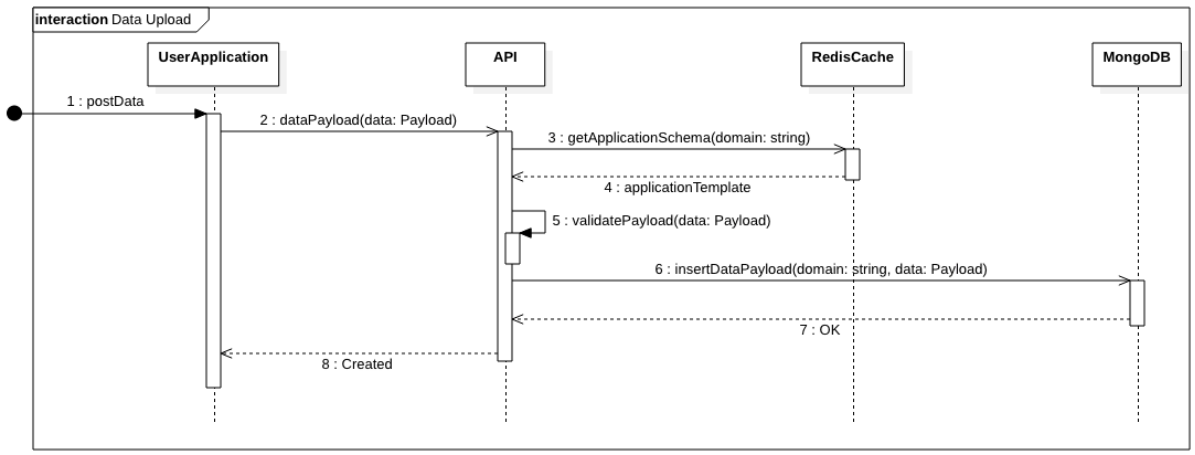


Figura 6 Diagrama de secuencia del manejo de subida de datos a la plataforma

Desde el servidor, se recupera el esquema de aplicación desde el cache Redis, se valida los datos enviados y en caso de ser válidos se insertan en la base de datos.

Uso de la plataforma

Para utilizar la plataforma se necesita primero tener una instancia de Freenalytics corriendo en alguna máquina accesible desde el Internet, típicamente algún tipo de servidor. Para facilitar el despliegue y distribución de la aplicación, se generó una imagen de Docker que permite levantar este servicio de manera rápida. Dado los requisitos de MongoDB y Redis que tiene la

plataforma, se recomienda utilizar algún tipo de orquestrador de contenedores, en este caso, se hará uso de Docker Compose.

El código siguiente genera un stack de Docker Compose con todos los servicios listos para ser utilizados:

```yaml
version: '3.9'

services:
  freenalytics:
    image: ghcr.io/freenalytics/freenalytics:latest
    restart: unless-stopped
    depends_on:
      - mongo
      - redis
    ports:
      - '3000:3000'
    environment:
      MONGODB_URI: mongodb://root:password@mongo:27017/freenalytics?authSource=admin
      REDIS_URI: redis://redis:6379
      JWT_SECRET: MY_SUPER_SECRET
      REGISTRATION_OPEN: true

  mongo:
    image: mongo:latest
    restart: unless-stopped
    volumes:
      - ./data-mongo:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: password

  redis:
    image: redis:latest
    restart: unless-stopped
    volumes:
      - ./data-redis:/data
    command: redis-server --loglevel warning
```

Figura 7 Código del archivo docker-compose.yml para levantar la plataforma Freenalytics

Usando el comando *docker-compose up -d* el servidor levantará los servicios necesarios para correr la plataforma. Una vez iniciado, la plataforma estará disponible en el puerto 3000.

Terminado el proceso de despliegue de la plataforma, el usuario puede crear su cuenta o iniciar sesión en la plataforma. Luego, tendrá acceso a una página que incluye todas las aplicaciones asociadas a su cuenta.

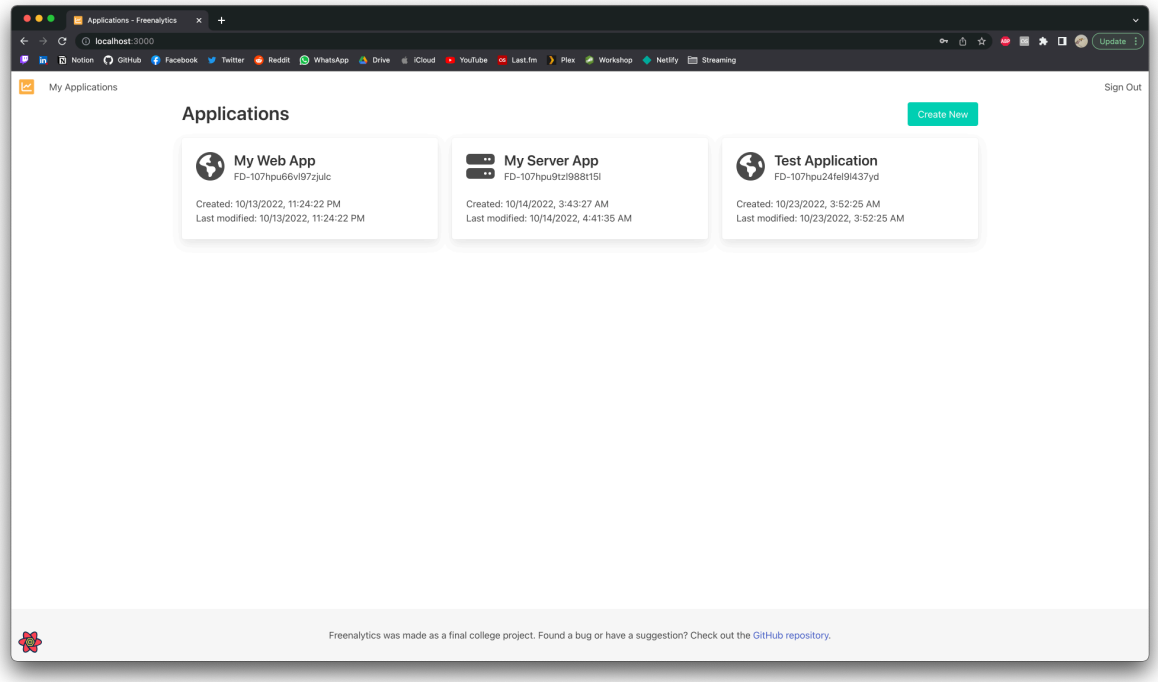


Figura 8 Página de aplicaciones

Dentro de esta página, el usuario puede crear una nueva aplicación, donde deberá ingresar el nombre de esta, el tipo de aplicación correspondiente, el JSON Schema de los datos de la aplicación y adicionalmente, enlaces a las librerías de integración (llamadas conectores) en caso de existir.

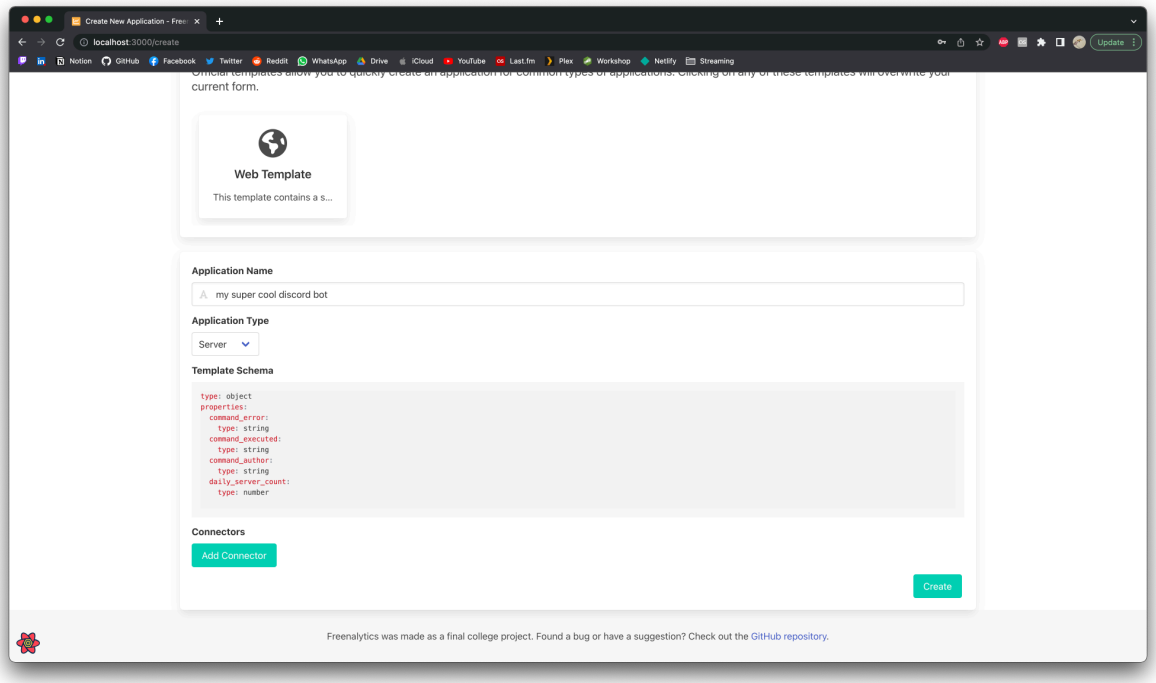


Figura 9 Página de creación de aplicación

Al terminar el proceso de creación de la aplicación, la página web redireccionará al usuario al Dashboard de la aplicación donde podrá encontrar todas las funcionalidades relevantes a su aplicación. Desde gráficos de visualización, hasta tablas de datos, este es el espacio en la plataforma asignado para su aplicación.

Existe una sección que incluye información de sobre cómo subir los datos a la plataforma. En el caso de la aplicación generada en la figura mostrada anteriormente, una simple llamada HTTP POST con la carga correcta es suficiente, algo que es posible realizar en cualquier tipo de lenguaje de programación.

Adicionalmente, el usuario tiene acceso a una sección donde podrá ver todos sus datos subidos en forma de tabla con la opción de expotar todo en formato CSV para permitir cualquier tipo de procesamiento de datos personalizado.

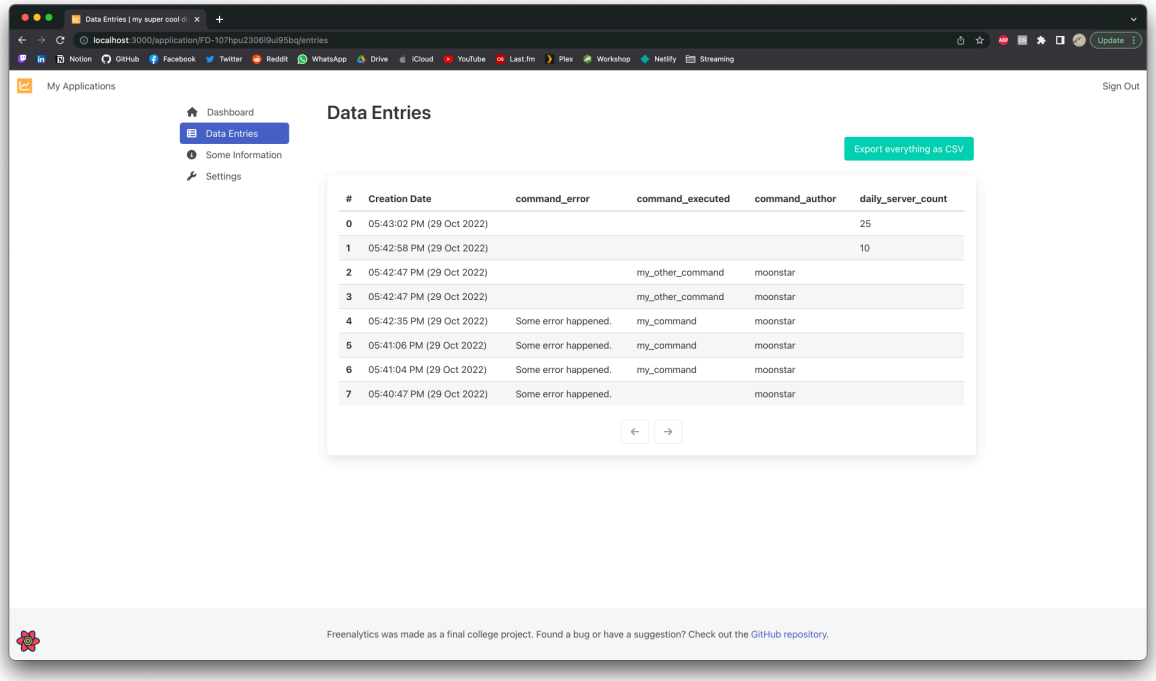
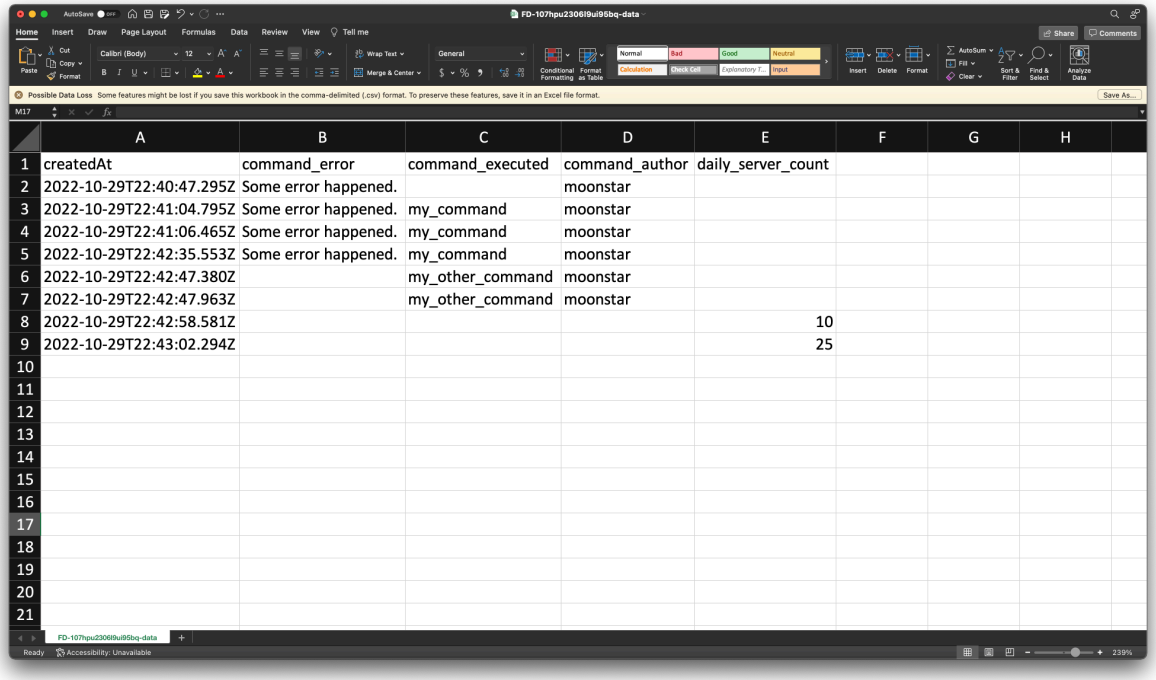


Figura 10 Página de entradas de datos en formato de tabla

Dando clic en el botón “Export everything as CSV” el servidor enviará un CSV al navegador con el cual el usuario tendrá acceso a todos sus datos como se presenta a continuación.



	A	B	C	D	E	F	G	H
1	createdAt	command_error	command_executed	command_author	daily_server_count			
2	2022-10-29T22:40:47.295Z	Some error happened.		moonstar				
3	2022-10-29T22:41:04.795Z	Some error happened.	my_command	moonstar				
4	2022-10-29T22:41:06.465Z	Some error happened.	my_command	moonstar				
5	2022-10-29T22:42:35.553Z	Some error happened.	my_command	moonstar				
6	2022-10-29T22:42:47.380Z		my_other_command	moonstar				
7	2022-10-29T22:42:47.963Z		my_other_command	moonstar				
8	2022-10-29T22:42:58.581Z				10			
9	2022-10-29T22:43:02.294Z				25			
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								

Figura 11 Vista del CSV exportado

En las siguientes secciones se presentan dos ejemplos de aplicaciones que se integran con Freenalytics que fueron realizadas con el objetivo de ilustrar y documentar el funcionamiento de la plataforma con casos reales de uso.

Caso de uso 1: Sitio Web

Descripción

Como ejemplo de integración de la plataforma con una aplicación Web, se creó un sitio web con tres páginas simples, una de inicio con información sobre la aplicación, un artículo falso y una página de información adicional. Esta aplicación fue realizada en HTML puro.

Inicialización de la aplicación en la plataforma

Para integrar esta página Web se necesita crear primeramente la aplicación en Freenalytics para tener un espacio en donde subir los datos. Para esto, se crea un nuevo espacio de monitoreo dentro de Freenalytics que haga uso de la plantilla oficial para Web que viene incluida en la plataforma.

Esta plantilla permite generar rápidamente todo lo necesario para el monitoreo de una aplicación que pueda integrarse directamente usando uso de la librería de integración para una aplicación Web. Dentro de esta plantilla se incluye un esquema de datos que se ajusta a los requerimientos de la librería.

```yaml
type: object
properties:
  page_title:
    type: string
  url_route:
    type: string
  user_time_in_page:
    type: number
  user_scrolled:
    type: boolean
  user_first_visit:
    type: boolean
  user_location:
    type: string
  referrer:
    type: string
  num_of_clicks:
    type: integer
  element_clicked:
    type: object
    properties:
      url_route:
        type: string
      tag_name:
        type: string
      class_name:
        type: string
      id:
        type: string
      page_x:
        type: integer
      page_y:
        type: integer
      page_width:
        type: integer
      page_height:
        type: integer
      client_x:
        type: integer
      client_y:
        type: integer
      client_width:
        type: integer
      client_height:
        type: integer
```

Figura 12 Esquema de datos de la plantilla oficial Web

Con este esquema definido, la aplicación está lista para recopilar información de uso de la página Web como: el nombre de la página donde se encuentra el usuario, la ruta que está visitando, cuánto tiempo pasa el usuario en la página, si el usuario se desplazó dentro de la página, si el usuario digitó la URL del sitio directamente o si vino de algún otro sitio, cuantos

clics realizó, en qué elementos hizo clic, las coordenadas de los clics dentro de la página, entre otros.

Con esta información, el desarrollador del sitio Web tiene conocimiento de cómo los usuarios interactúan con su aplicación, en qué elementos hacen más clics, qué páginas son las más interesantes para ellos, quién refiere a los usuarios hacia su aplicación, y mucho más.

Uso de la librería freenalytics-connector-web

La integración del sitio Web con la plataforma es sumamente fácil. Haciendo uso de la plantilla oficial Web, el desarrollador tiene acceso a una librería oficial que conecta su aplicación con Freenalytics.

Para realizar esta conexión, el desarrollador necesita agregar las siguientes líneas dentro del encabezado de su documento HTML:

```
<script type="text/javascript"
src="https://cdn.jsdelivr.net/gh/freenalytics/freenalytics-connector-
web@v1.1.0/connector.min.js"></script>
<script type="text/javascript" src="/analytics.js"></script>
```

Figura 13 Código HTML de integración con la plataforma

Adicionalmente, el desarrollador necesita crear un archivo llamado *analytics.js* dentro de la raíz de su servidor Web con las instrucciones siguientes:

```
const client = new freenalytics.Client({
  apiUrl: 'http://localhost:4000/api',
  domain: 'FD-107hpu34tlb7s7mro'
});
client.initialize();
```

Figura 14 Código JavaScript de integración con la plataforma

Donde *apiUrl* deberá ser actualizado con la URL del API de la instancia de Freenalytics del desarrollador y *domain* con el nombre del dominio de la aplicación que fue creada en la plataforma. Con esto, el sitio Web estará listo para subir datos a la plataforma automáticamente.

Visualización de los datos de analítica

A continuación, se presentan algunas de las gráficas disponibles en el Dashboard de la aplicación de Freenalytics para el sitio Web de ejemplo.

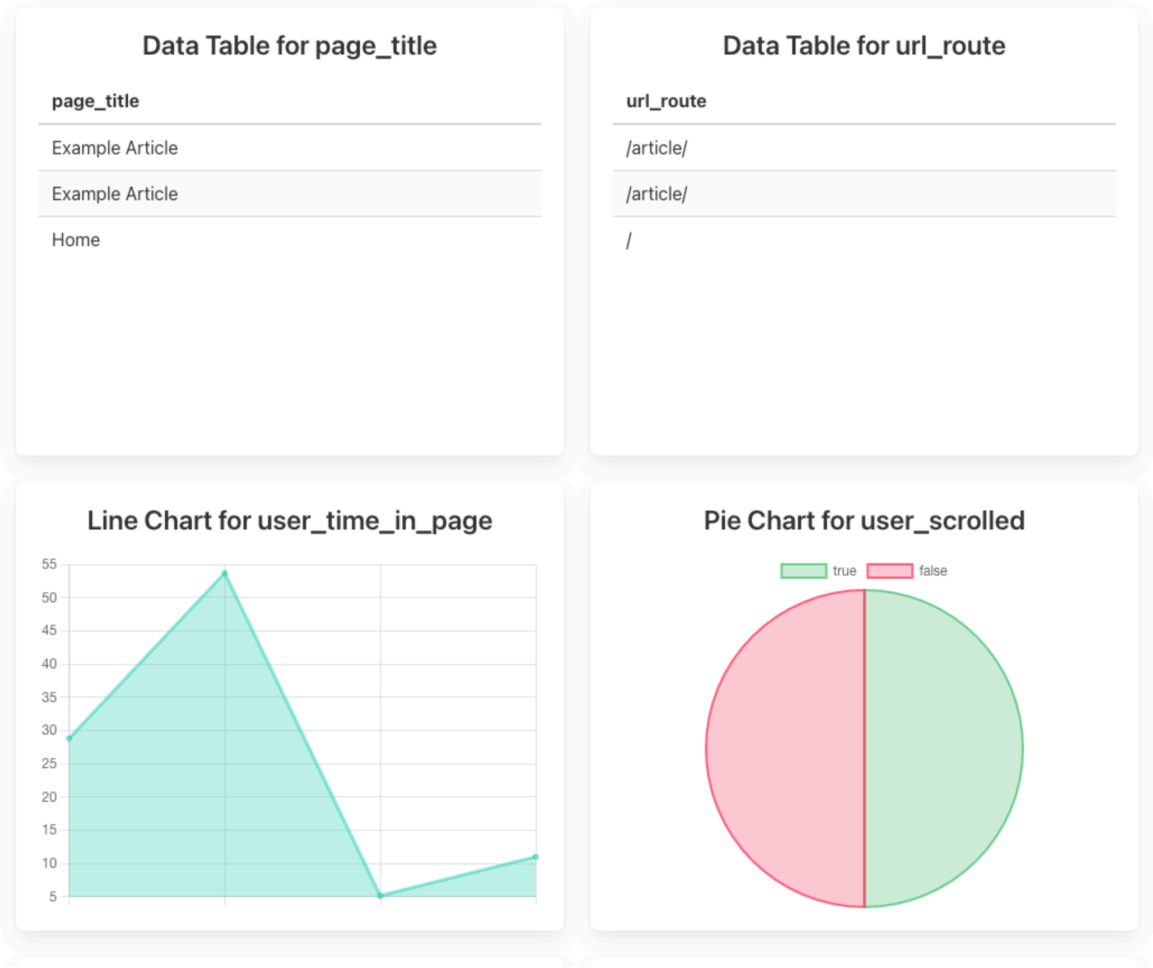


Figura 15 Gráficos de visualización de los datos de la aplicación Web (1)

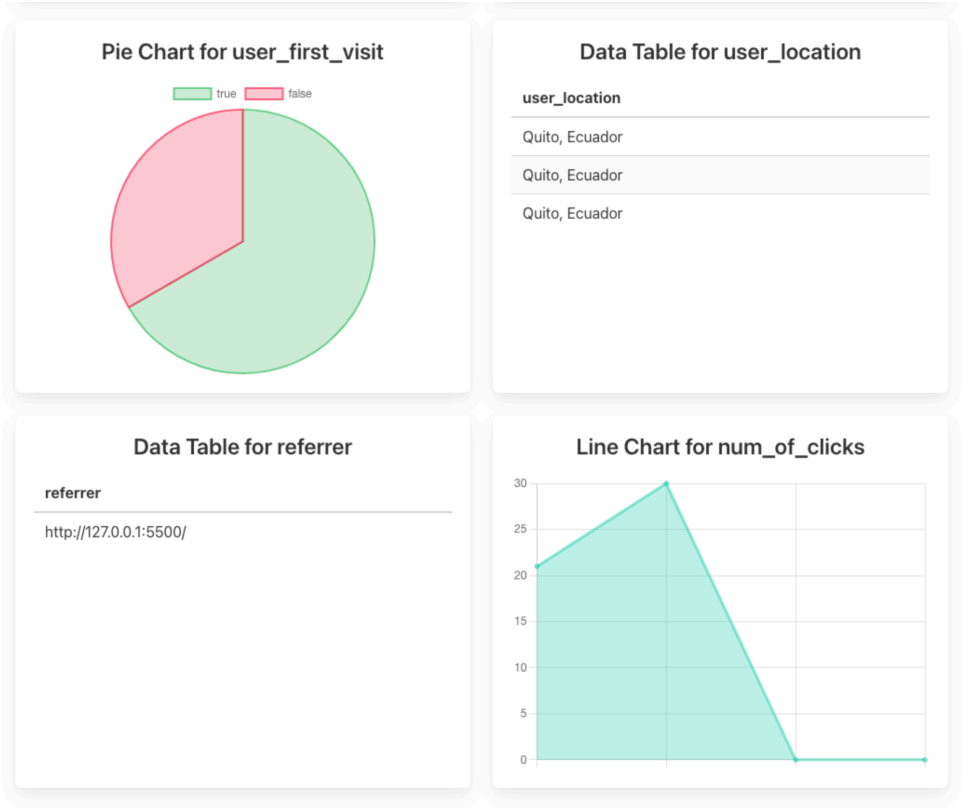


Figura 16 Gráficos de visualización de los datos de la aplicación Web (2)

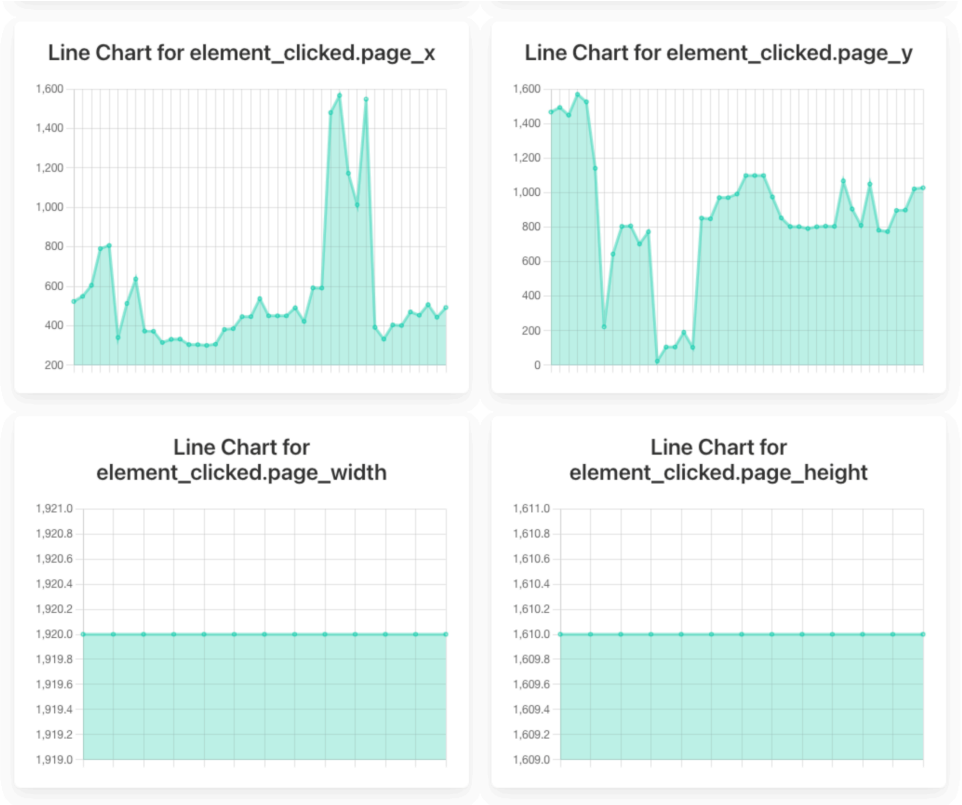


Figura 17 Gráficos de visualización de los datos de la aplicación Web (3)

Generación de vista de clics

Como un ejemplo del tipo de tratamiento o procesamiento de los datos que se puede realizar con la funcionalidad de exportación CSV, se creó un pequeño script en Python junto a las librerías Selenium, Pandas, Numpy y Matplotlib, que permite generar capturas de las pantallas del sitio Web con una visualización integrada de los clics de los usuarios. Esto permite tener una visualización más digerible de cómo usan los usuarios el sitio Web.

El script es bastante simple y realiza lo siguiente:

1. Con Selenium, se genera una captura de pantalla de la página Web que se desea procesar.
2. Usando Pandas, se extrae un arreglo de coordenadas normalizadas de los clics que los usuarios realizaron en esa página.
3. Con Matplotlib, se agregan círculos en las coordenadas antes calculadas a la captura realizada anteriormente, generando así una visualización de los clics de los usuarios.

A continuación, se presentan las capturas procesas generadas por el script para cada una de las páginas del sitio Web de ejemplo.

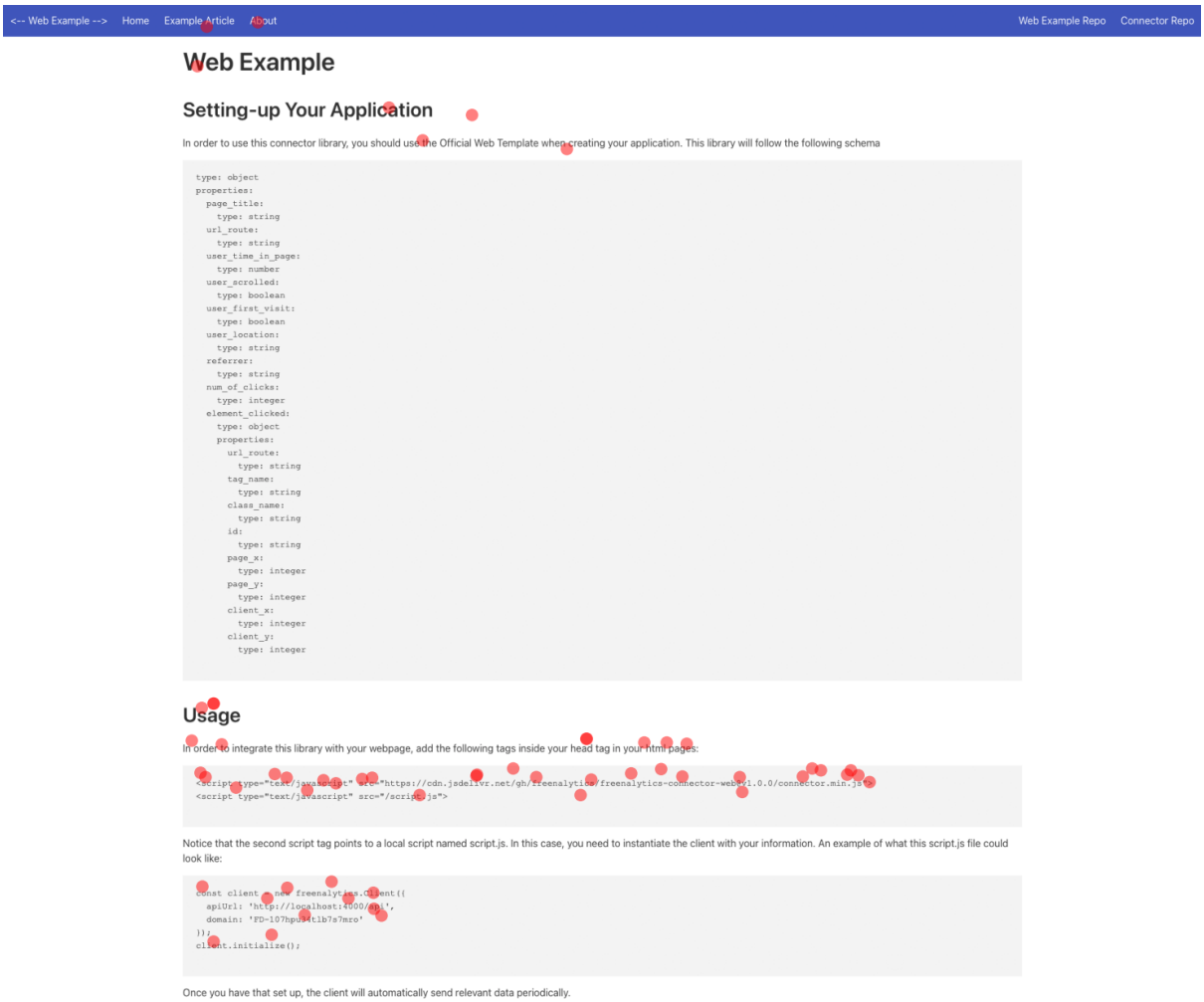


Figura 18 Captura procesada de los clics de los usuarios para la página /

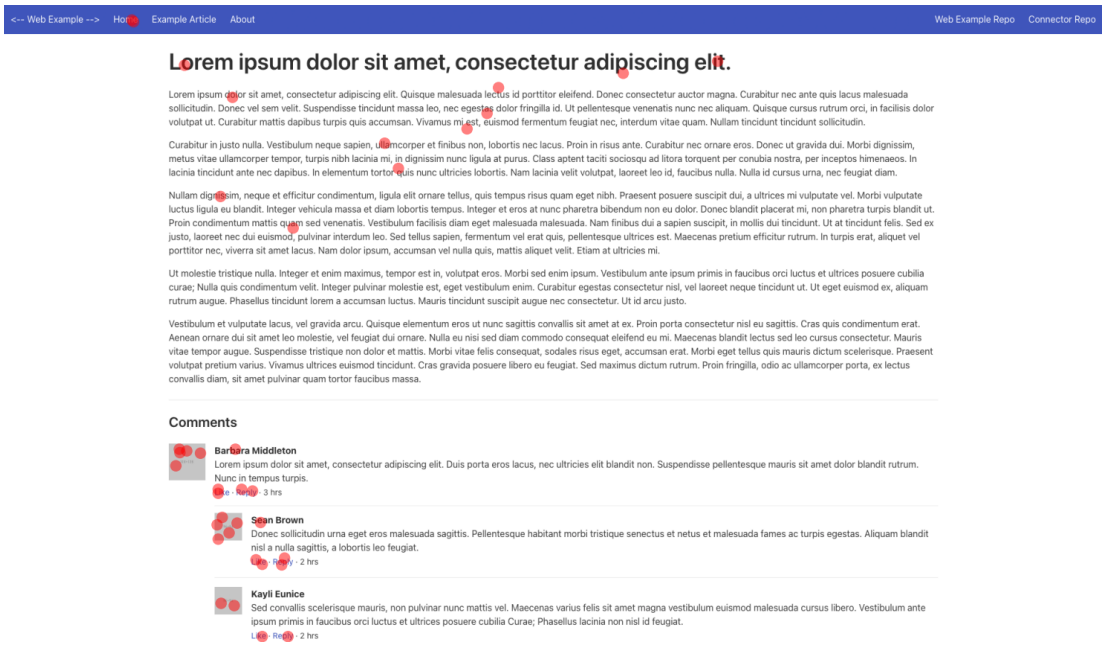


Figura 19 Captura procesada de los clics de los usuarios para la página /article

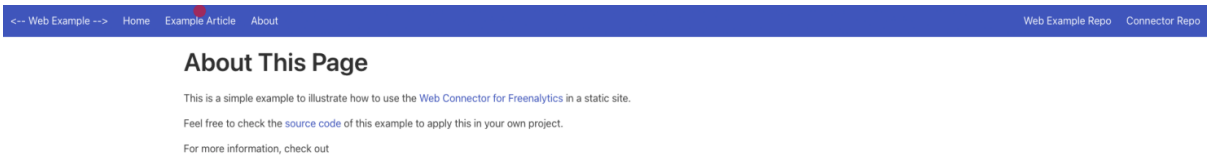


Figura 20 Captura procesada de los clics de los usuarios para la página /about

Con esta visualización es ahora más fácil determinar las zonas más "calientes" dentro del sitio Web. Se conoce así qué páginas son las más visitadas, las más interactuadas, lo que permite mejores decisiones en el futuro sobre el desarrollo del sitio Web. Con este conocimiento, sería posible generar una mejor estrategia de diseño, donde las partes más importantes se encuentren en las zonas "calientes" para una mejor experiencia de usuario.

Caso de uso 2: Servicio de chatbot con comandos para la plataforma de mensajería instantánea Discord

Descripción

Como ejemplo de integración de la plataforma con una aplicación no convencional, se creó un chatbot de ejemplo para la aplicación de mensajería instantánea Discord. Discord es un servicio enfocado en jugadores de videojuegos de computador que permite comunicación por voz y por mensajes de texto basados en canales similares a cómo funcionaba IRC (Internet Relay Chat). (Discord, s.f)

Este servicio posee un API que permite a desarrolladores crear integraciones con Discord en forma de bots con los que los usuarios de la plataforma pueden interactuar a través de comandos de texto.

En este ejemplo, se creó un pequeño bot con dos comandos, uno que saluda al usuario por su nombre y otro que genera un error con una probabilidad del 80%. El ejemplo está integrado con Freenalytics para recopilación del uso de los comandos del bot.

Inicialización de la aplicación en la plataforma

De manera análoga a la aplicación Web antes vista, la integración de esta aplicación requiere la creación de un espacio para la subida de datos en Freenalytics. Para esto, se crea una aplicación con el siguiente JSON Schema:

```yaml
type: object
properties:
  command_name:
    type: string
  command_author:
    type: string
  command_success:
    type: boolean
  command_error_message:
    type: string
  server_count:
    type: integer
  member_count:
    type: integer
```

Figura 21 Esquema de datos del bot de Discord

Con este esquema, se podrá recopilar: el nombre del comando utilizado, el nombre del usuario que ejecutó el comando, el resultado del comando (si hubo error o no) y el mensaje de error (en caso de existir). Adicionalmente se tendrá la información del número de servidores a los que el bot está conectado y la cantidad de usuarios que tienen acceso al bot.

En términos de subida de datos existen dos formas de activación:

1. Cuando un comando es ejecutado, se enviarán los campos *command_name*, *command_author*, *command_success* y *command_error_message*.
2. En un intervalo de 10 segundos, se enviarán los campos *server_count* y *member_count*.

Código del cliente para comunicarse con la plataforma

Dado a que esta aplicación no proviene de ningún tipo de plantilla, el desarrollador debe implementar un cliente que pueda comunicarse con el API de Freenalytics.

En este ejemplo, el código de este cliente es el siguiente:

```
const axios = require('axios');
const logger = require('@greencoast/logger');
const { Message } = require('discord.js');

const DEFAULT_INTERVAL = 10000;

class FreenalyticsClient {
  constructor(client, options) {
    FreenalyticsClient.validateOptions(options);

    this.client = client;
    this.apiUrl = options.apiUrl;
    this.domain = options.domain;
    this.interval = Math.floor(options.interval ?? DEFAULT_INTERVAL);

    this.rest = axios.create({
      baseURL: this.apiUrl
    });
  }

  initialize() {
    this.registerEvents();
    setInterval(this.postIntervalHandler.bind(this), this.interval);
  }

  registerEvents() {
    this.client.on('commandExecute', this.handleCommandExecution.bind(this));
    this.client.on('commandError', this.handleCommandError.bind(this));
  }

  handleCommandExecution(command, trigger) {
    const author = trigger instanceof Message ?
      `${trigger.author.username}#${trigger.author.discriminator}` :
      `${trigger.user.username}#${trigger.user.discriminator}`;

    return this.postPayload({
      command_name: command.name,
      command_author: author,
      command_success: true
    });
  }

  handleCommandError(error, command, trigger) {
    const author = trigger instanceof Message ?
      `${trigger.author.username}#${trigger.author.discriminator}` :
      `${trigger.user.username}#${trigger.user.discriminator}`;

    return this.postPayload({
      command_name: command.name,
      command_author: author,
      command_success: false,
      command_error_message: error?.message ?? 'An unknown error occurred.'
    });
  }

  postIntervalHandler() {
    const guildCount = this.client.guilds.cache.size;
    const memberCount = this.client.guilds.cache.reduce((sum, guild) => sum + guild.memberCount, 0);

    return this.postPayload({
      server_count: guildCount,
      member_count: memberCount
    });
  }

  async postPayload(payload) {
    try {
      await this.rest.post(`/applications/${this.domain}/data`, payload);
      logger.debug(`Successfully posted analytics with payload: ${JSON.stringify(payload)}`);
    } catch (error) {
      logger.error(error.message);
    }
  }

  static validateOptions(options) {
    if (!options.apiUrl) {
      throw new Error('options.apiUrl is required.');
    }

    if (!options.domain) {
      throw new Error('options.domain is required.');
    }

    if (options.interval) {
      if (isNaN(options.interval) || options.interval < 1) {
        throw new Error('options.interval should be a positive number.');
      }
    }
  }
}
module.exports = FreenalyticsClient;
```

Figura 22 Código de JavaScript para integrar el bot de Discord con la plataforma

Este cliente recibe como parámetro el objeto cliente del bot que le permitirá obtener toda la información relevante que necesita ser enviada a Freenalytics. La inicialización de este cliente registra los eventos *commandExecuted* y *commandError* que son parte de la interfaz del cliente de la librería utilizada para comunicarse con el API de Discord. Estos eventos son disparados cada vez que un usuario ejecuta un comando. Adicionalmente, el cliente crea un manejador de intervalos que envía la información de la cantidad de servidores y usuarios del bot cada 10 segundos.

La implementación anterior sirve nada más de referencia para ilustrar el tipo de trabajo que un desarrollador necesita realizar para integrar su aplicación con Freenalytics.

Visualización de los datos de analítica

A continuación, se presentan algunas de las gráficas disponibles en el Dashboard de la aplicación de Freenalytics para el bot de Discord de ejemplo.

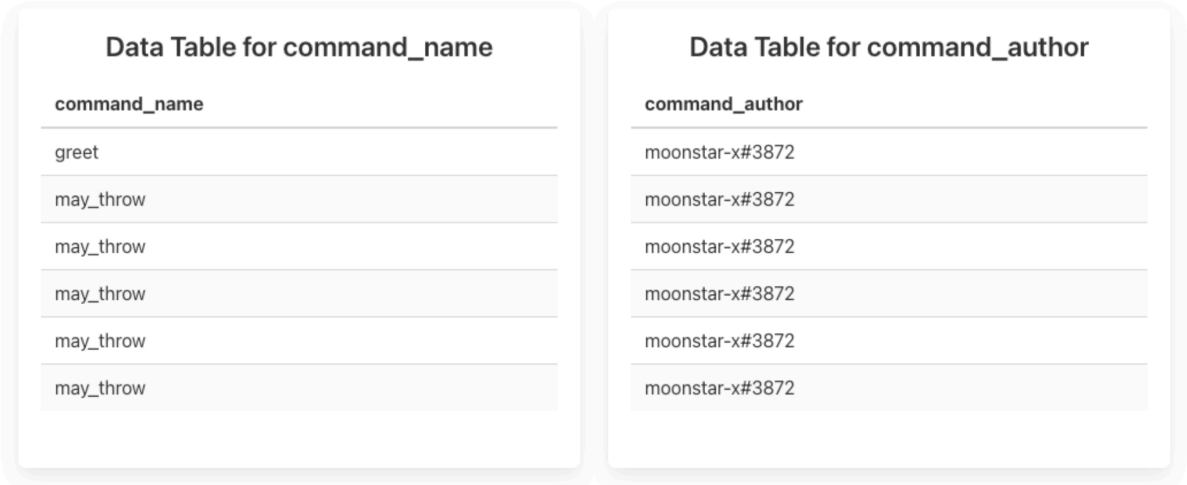


Figura 23 Gráficos de visualización de los datos del bot de Discord (1)

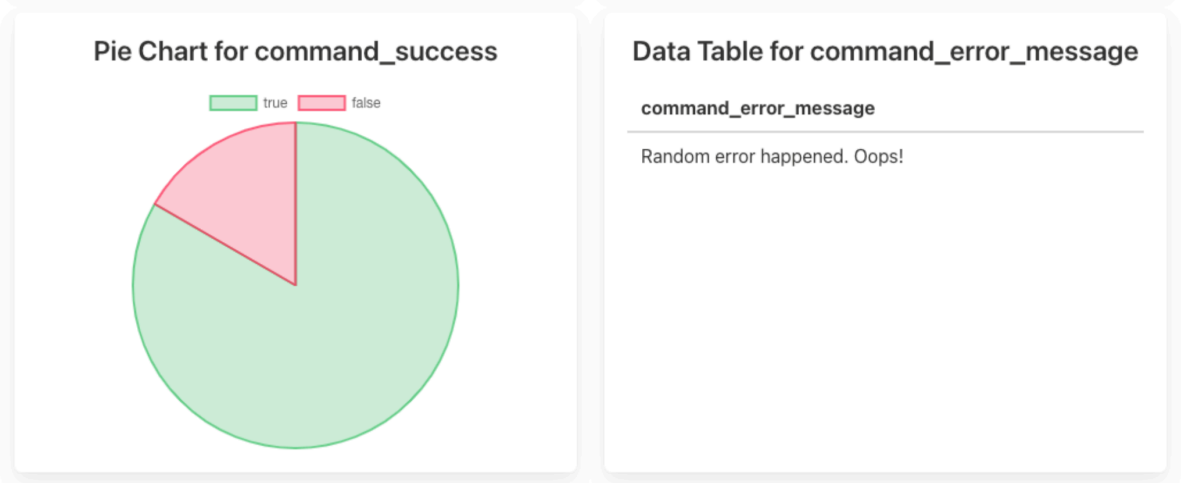


Figura 24 Gráficos de visualización de los datos del bot de Discord (2)

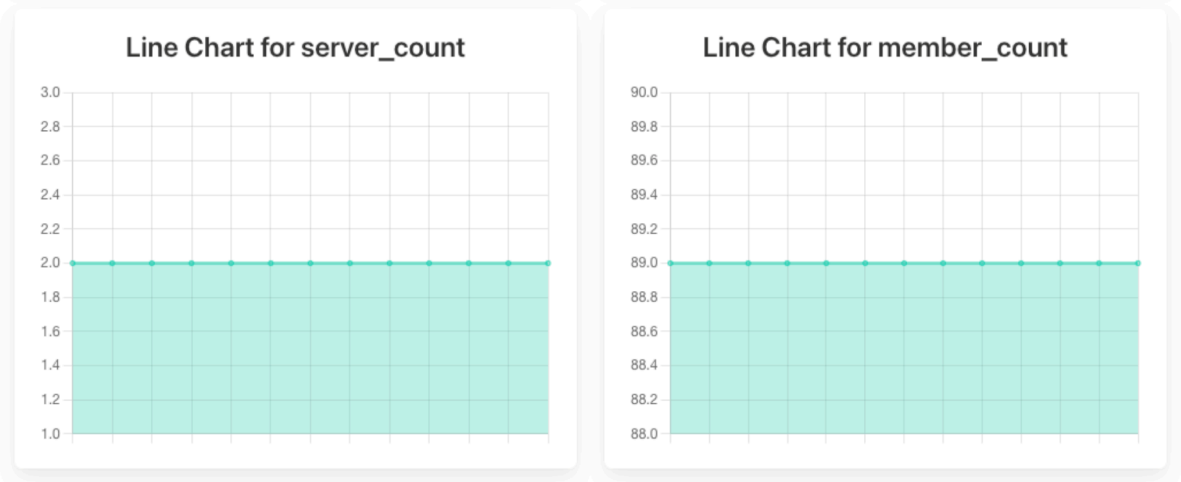


Figura 25 Gráficos de visualización de los datos del bot de Discord (3)

CONCLUSIONES

Finalmente, la necesidad de llevar registro de analíticas de datos de uso de aplicaciones es siempre presente al momento de desarrollar cualquier tipo de sistema. Desde que se empezó con el desarrollo de la plataforma, el repositorio de GitHub ha ganado dos estrellas al momento de la escritura de este documento. Cabe recalcar que en ningún momento se realizó publicidad de ningún tipo, y aquellas estrellas se obtuvieron por usuarios que encontraron el proyecto por su propia cuenta, presuntamente por la necesidad de alguna plataforma o servicio que satisfaga sus necesidades de analíticas de datos.

A través de este proyecto se ha puesto en práctica varias herramientas, tecnologías y utilidades para el desarrollo de una aplicación Full Stack. Se ha creado un proyecto robusto y completo que hace uso de tecnologías modernas y relevantes como son Node.js y React, incluyendo el uso de Docker para permitir la contenerización de la plataforma para un despliegue rápido. Adicionalmente, a través del uso de pipelines de CI/CD se ha implementado un proceso automatizado de prueba, compilación y publicación que cumple con los estándares de desarrollo de Software actuales.

Asimismo, se han alcanzado los objetivos de este trabajo en su totalidad; desde la implementación de una metodología de desarrollo orientada a las pruebas unitarias, hasta la creación de una comunidad de fuente abierta por medio de la creación de páginas de documentación con gran detalle sobre el proceso de desarrollo para atraer a personas dispuestas a contribuir al proyecto.

No obstante, la plataforma actualmente se encuentra en una etapa temprana de desarrollo y tiene algunas limitaciones como son las limitadas visualizaciones que existen actualmente y la falta de un servicio de análisis de datos incluido en la propia plataforma.

Recomendaciones

A partir de la experiencia de desarrollo de este proyecto se han recopilado algunas recomendaciones:

1. Un diseño extenso es necesario en la etapa inicial de ideación de cualquier proyecto. El diseño debe no solo tomar en cuenta de la solución que se trabajará sino también del potencial uso que se le dará. En el caso de Freenalytics, la simpleza del proceso de creación de aplicaciones a través de una definición de un único esquema estructural de los datos termina siendo problemática porque limita de gran manera lo que la plataforma puede realizar en cuanto a visualización y análisis de los datos, dado a que no hay semántica en la estructura de los datos que se definen. Esto puede resolverse a través de un proceso de creación de aplicaciones más extenso que permita al usuario definir cómo quiere que sus datos se muestren para que él o ella pueda dar el sentido a sus datos.
2. La utilización de herramientas modernas es de gran ayuda. Notablemente, seguir estándares de calidad del proceso de desarrollo es imprescindible en cualquier tipo de proyecto. En este caso, la automatización de los procesos de desarrollo (DevOps) permite ahorrar tiempo, reducir potenciales errores y alcanzar al usuario final más rápidamente.

Trabajo Futuro

Tomando en cuenta las limitaciones de la plataforma antes mencionadas, se definió algunas tareas que se podrían realizar a futuro para mejorar el funcionamiento de la plataforma:

1. Se podría crear un lenguaje declarativo que pueda reemplazar el uso de JSON Schema, que sea más fácil y corto de escribir y que permita definir campos que requieren de extensa validación. Esto implica crear el lenguaje basado en YML en sí y un módulo

que permita realizar validación de los datos para asegurar que su estructura sea la adecuada.

2. El proceso de creación de aplicaciones debe incluir alguna manera para que los usuarios puedan definir cómo son visualizados sus datos. La manera actual de inferir la visualización basada en el tipo de dato no es suficientemente buena porque no todos los datos deberían verse igual a pesar de ser del mismo tipo.
3. La implementación de herramientas de visualización más "inteligentes" serían de gran beneficio para los usuarios. En el sitio Web de ejemplo que se exploró anteriormente se realizó un trabajo adicional para poder generar una visualización más digerible de los clics en una página Web específica. Este tipo de visualización podría estar integrada en la plataforma para que así los usuarios no tengan que implementar sus propias soluciones de tratamiento de datos.
4. Para incluir procesos de minería de datos, la integración con plataformas, servicios o aplicaciones ya existentes (como Weka) podría ser de gran utilidad para incluir funcionalidades de minería de datos en la propia plataforma.

REFERENCIAS BIBLIOGRÁFICAS

Agile Alliance. (s.f). The 12 Principles behind the Agile Manifesto. *Agile Alliance.* Extraído el 12 de noviembre de 2022 de: https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., Sarwar, S. Z. (2010). Agile software development: Impact on productivity and Quality. *2010 IEEE International Conference on Management of Innovation & Technology.* doi: 10.1109/ICMIT.2010.5492703

Beck, K. (2002). *Test-driven development by example.* Boston, MA: Addison-Wesley.

Discord. (s.f). Discord. Extraído el 15 de diciembre de 2022 de https://discord.com/

Google. (June 17, 2019). *Google Analytics Terms of Service.* Extraído el 19 de diciembre de 2022 de https://marketingplatform.google.com/about/analytics/terms/us/

Haik Y., Shahin T. (2011). *Engineering Design Process* (2nd ed.). Pacific Grove, CA: Cengage Learning.

IBM. (2021). What is a web service? *CICS Transaction Server for z/OS.* Extraído el 12 de noviembre de 2022 de: https://www.ibm.com/docs/en/cics-ts/5.2?topic=services-what-is-web-service

IEEE. (8 September 2022). ISO/IEC/IEEE International Standard—Information Technology—devops—building reliable and secure systems including Application Build, package and deployment. doi: 10.1109/IEEESTD.2022.9882056

JSON Schema. (s.f). JSON Schema | The home of JSON Schema. Extraído el 14 de diciembre de 2022 de: https://json-schema.org/

Menzies, T., & Zimmermann, T. (2013). Software analytics: So what? *IEEE Software, 30*(4), 31-37. doi:10.1109/ms.2013.86

Microsoft. (s.f). TypeScript is JavaScript with syntax for types. Extraído el 14 de diciembre de 2022 de https://www.typescriptlang.org/

MongoDB. (s.f). MERN Stack Explained. Extraído el 14 de diciembre de 2022 de https://www.mongodb.com/mern-stack

Radigan, D. (s.f). Kanban. *Atlassian.* Extraído el 12 de noviembre de 2022 de: https://www.atlassian.com/agile/kanban

Red Hat. (2020). What is a REST API?. *Understanding APIs.* Extraído el 12 de noviembre de 2022 de: https://www.redhat.com/en/topics/api/what-is-a-rest-api

Rehkopf M. (s.f). Continuous delivery principles. *Atlassian.* Extraído el 12 de noviembre de 2022 de: https://www.atlassian.com/continuous-delivery/principles

Rehkopf M. (s.f). What is continuous integration?. *Atlassian.* Extraído el 12 de noviembre de 2022 de: https://www.atlassian.com/continuous-delivery/continuous-integration

ANEXO A: LISTA DE ENLACES RELEVANTES DEL PROYECTO

A continuación, se presenta una lista de los enlaces Web relevantes al proyecto:

- Organización de GitHub que contiene los repositorios de código del proyecto: https://github.com/freenalytics
- Repositorio del código del proyecto: https://github.com/freenalytics/freenalytics
- Repositorio del código de la librería de integración Web: https://github.com/freenalytics/freenalytics-connector-web
- Repositorio del código de la aplicación Web de ejemplo: https://github.com/freenalytics/example-web
- Repositorio del código de la herramienta de visualización de clics: https://github.com/freenalytics/tools-page-clicks-view
- Repositorio del código del chatbot para Discord de ejemplo: https://github.com/freenalytics/example-discord-bot
- Sitio Web de documentación de la plataforma https://freenalytics.github.io/
- Sitio Web de documentación del API REST de la plataforma: https://freenalytics.github.io/api-docs/