Organizaciones UP:
Ingeniería detrás de la plataforma

Jose Naranjo
Universidad del Pacífico

1 Requerimientos

1.1 Requerimientos de la plataforma

1.1.1 Requerimientos funcionales

La plataforma debe ser capaz de capturar información relevante acerca del comportamiento de los usuarios, poder identificarlos a través de un conjunto de variables, funcionar siempre, tener respaldos, formas de recuperarse y debe funcionar en línea.

1.1.2 Requerimientos no funcionales

La plataforma debería funcionar con celulares, tablets, y computadoras de todo tipo; debe ser fácil de usar y de modificar de acuerdo a las necesidades cambiantes; debería ser barata y minimizar costos en la medida de lo posible.

1.2 Requerimientos de analítica

1.2.1 Requerimientos funcionales

La plataforma debe poder crear perfiles de los usuarios que nos permitan preveer sus intereses y encontrar tendencias para ofrecerles servicios relacionados o marketing dirigido. Debe también poder evaluar la reputación de las organizaciones, su presencia en las redes sociales, el interés que hay por ellas y debe ser capaz de exportar el conocimiento a través de visualizaciones.

1.2.2 Requerimientos no funcionales

La plataforma debe trabajar con buena data, limpia y verificada, en el formato adecuado de manera que cada herramienta pueda trabajar sobre ella sin problemas. Debe arrojar gráficos representativos, en un formato fácil de utilizar, entender y que sea visual.

Figure 1: Diagrama general

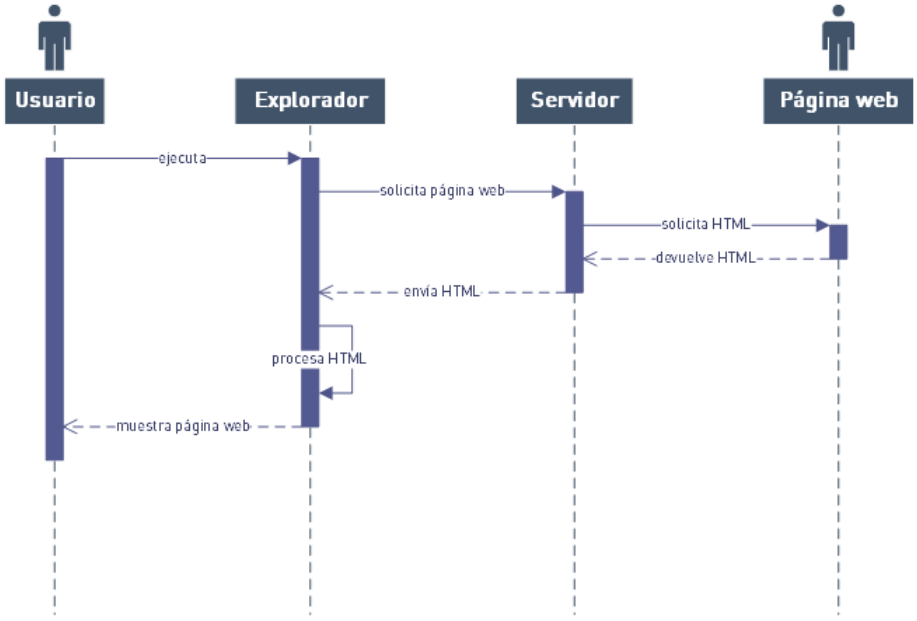
1.3 Casos de uso

1.3.1 Usuario ingresa a página web

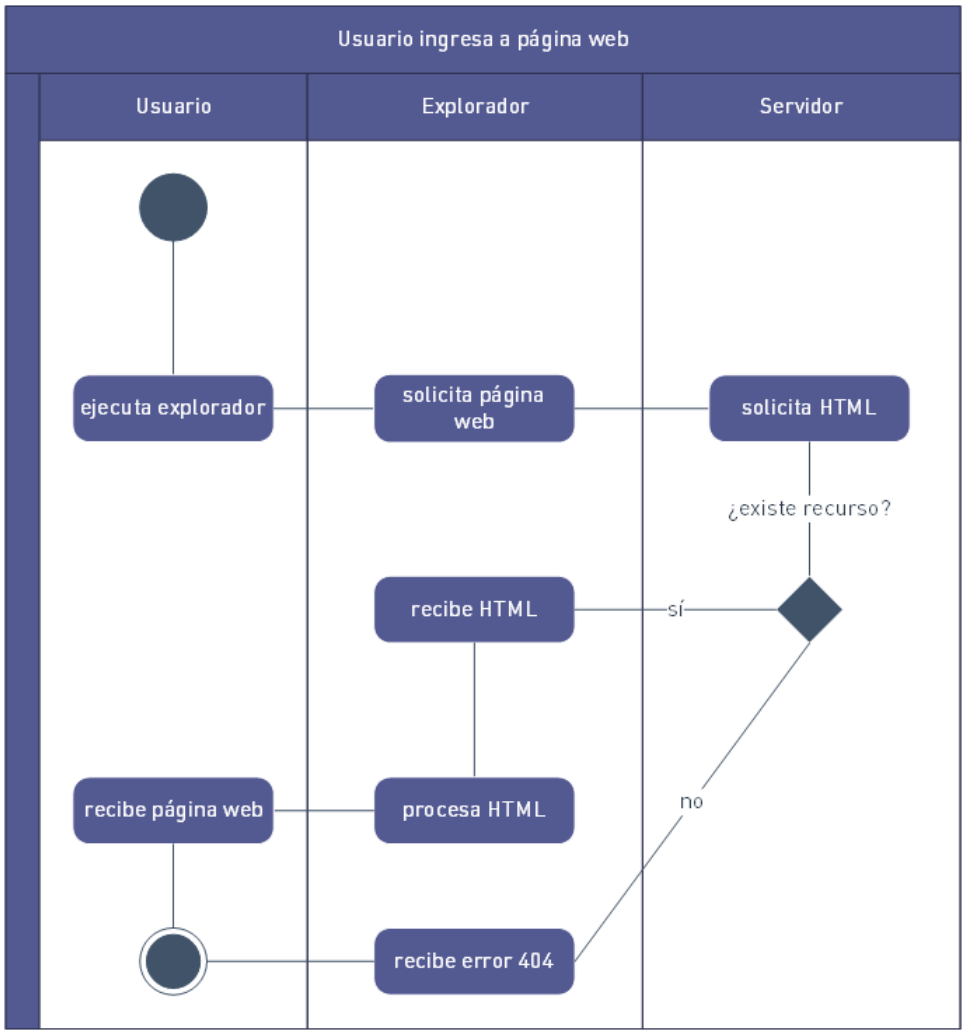
El usuario solicita la página web a través de su explorador y este contacta al servidor para retornarle lo que ha solicitado.

1. Caso de uso

2. Diagrama de secuencia



3. Diagrama de actividades

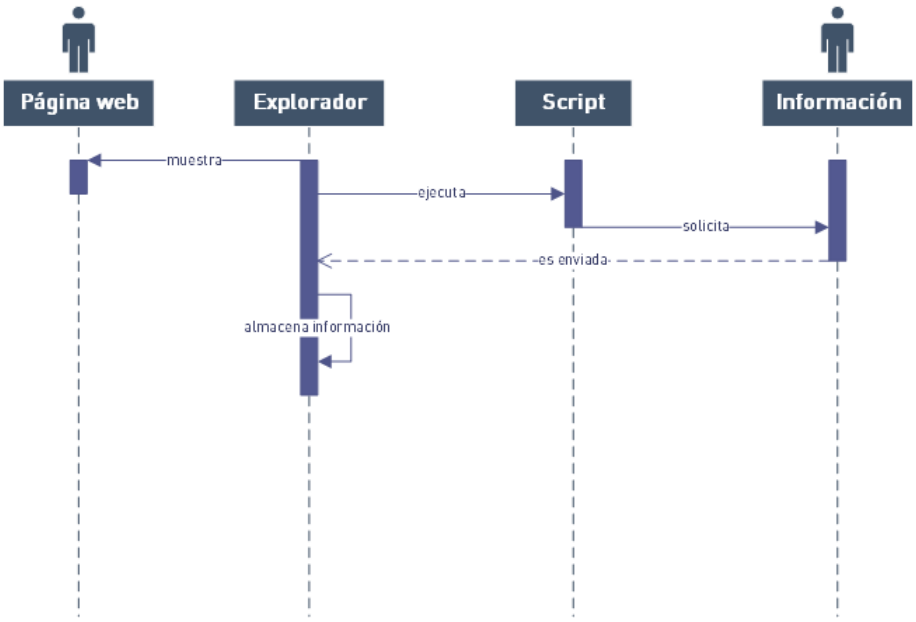


1.3.2 Información es recolectada localmente

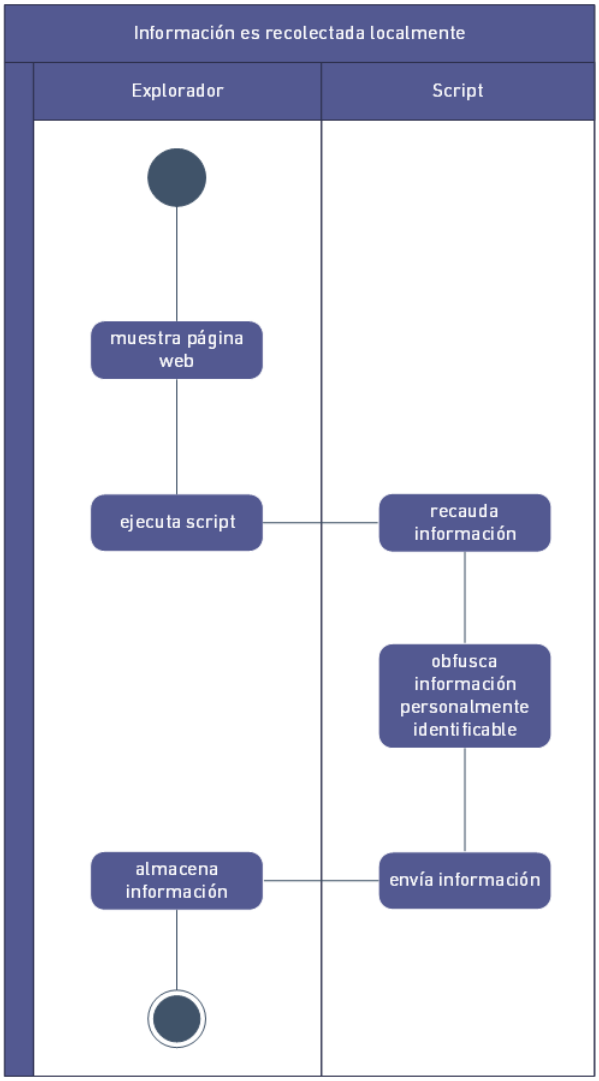
La página web, al cargar, carga el contenido en HTML, el diseño en CSS y los scripts de recolección de datos en JavaScript. Estos scripts obtienen la información que buscan y se almacenan en el caché temporal del explorador.

1. Caso de uso

2. Diagrama de secuencia



3. Diagrama de actividades

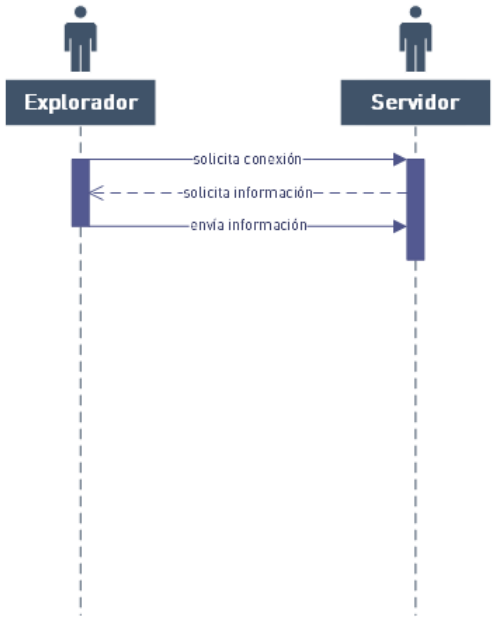


1.3.3 Información es enviada al servidor

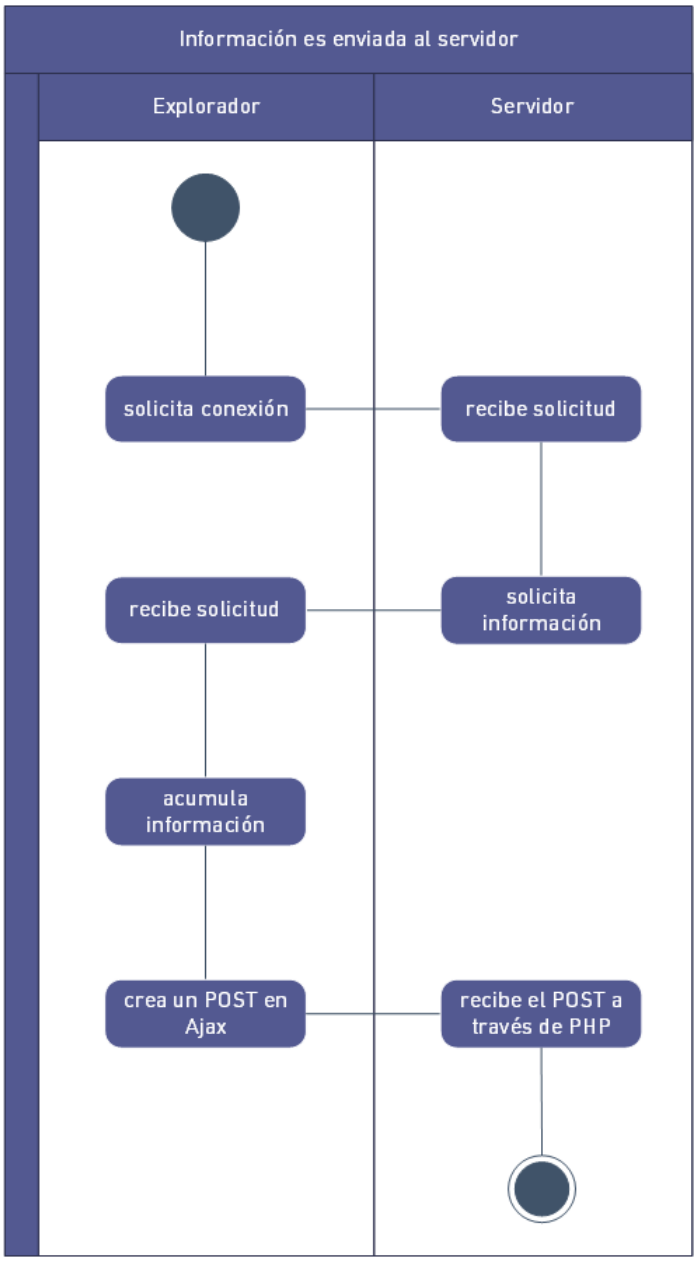
La información cacheada se envía en un POST de Ajax hacia el servidor, quien está esperando que llegue esta data.

1. Caso de uso

2. Diagrama de secuencia



3. Diagrama de actividades

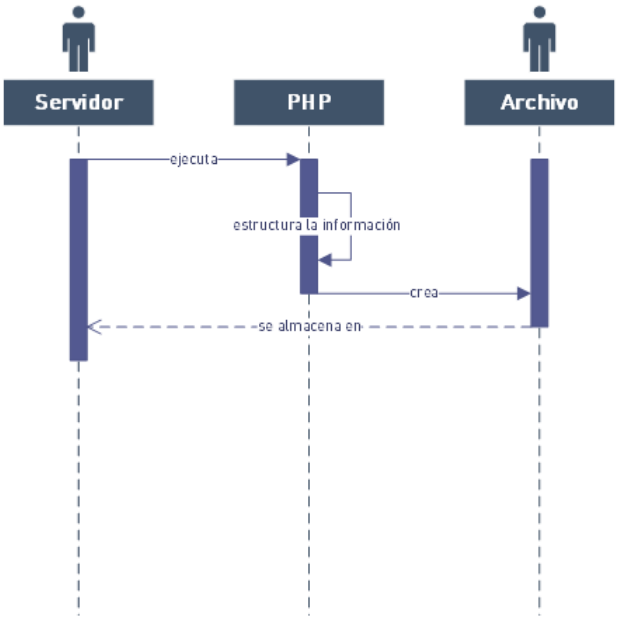


1.3.4 Información es almacenada en un archivo

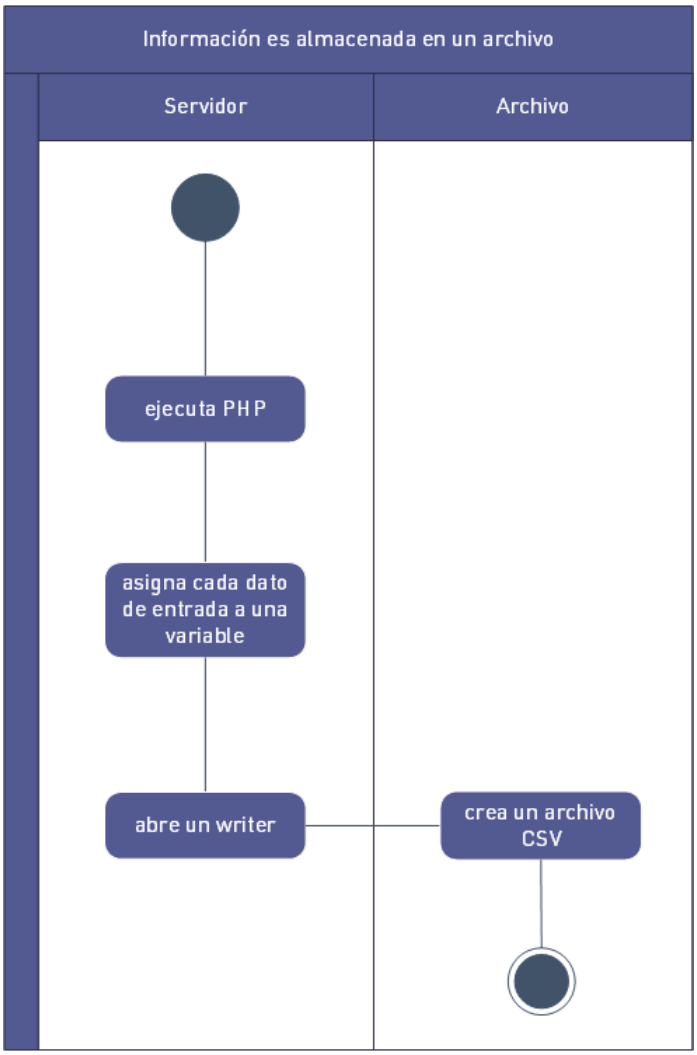
El servidor recibe el POST de Ajax y lo procesa a través de PHP, el cual escribe un archivo CSV con la data.

1. Caso de uso

2. Diagrama de secuencia



3. Diagrama de actividades

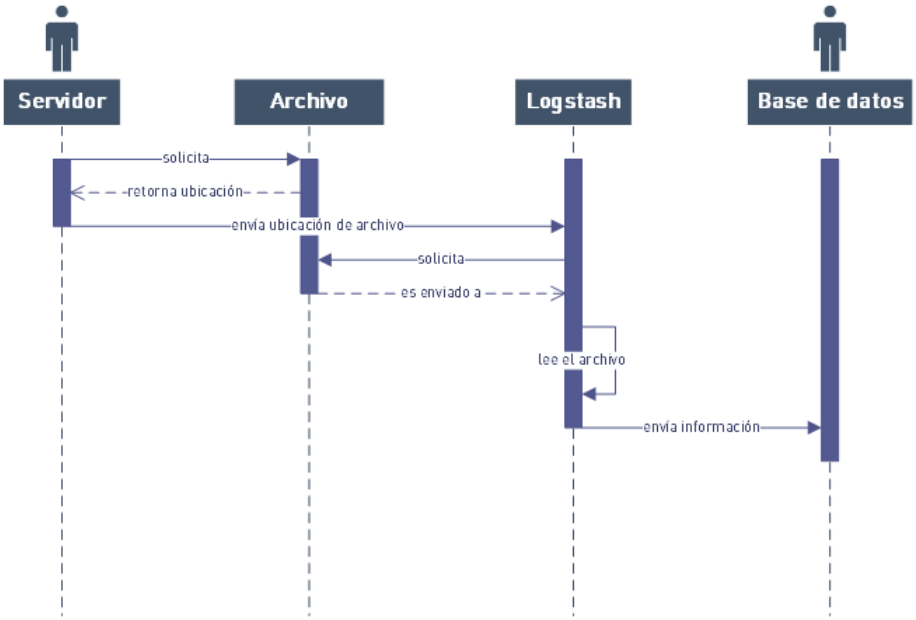


1.3.5 Información es enviada a la base de datos

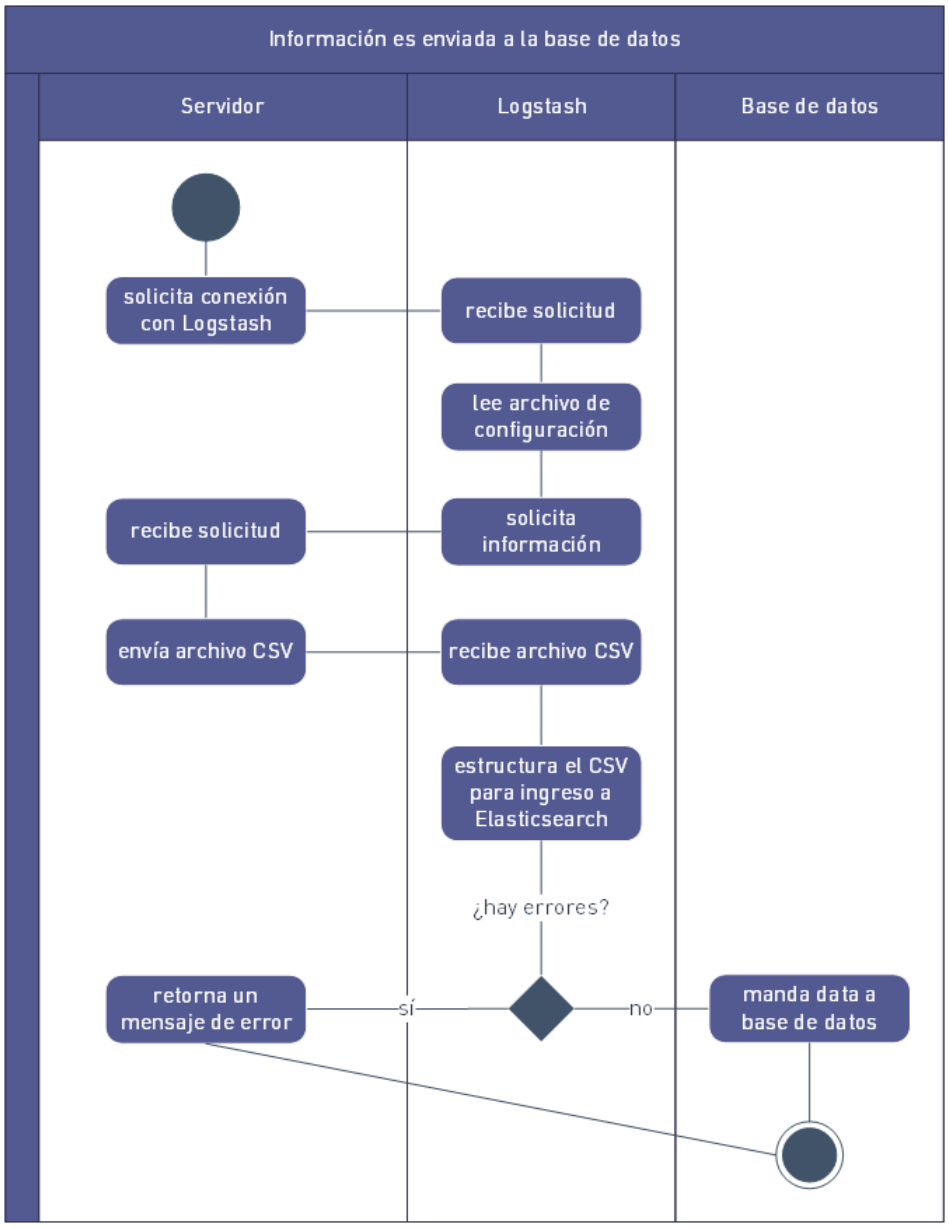
El archivo CSV con la data es procesado por Logstash y la data es enviada a la base de datos Elasticsearch.

1. Caso de uso

2. Diagrama de secuencia



3. Diagrama de actividades

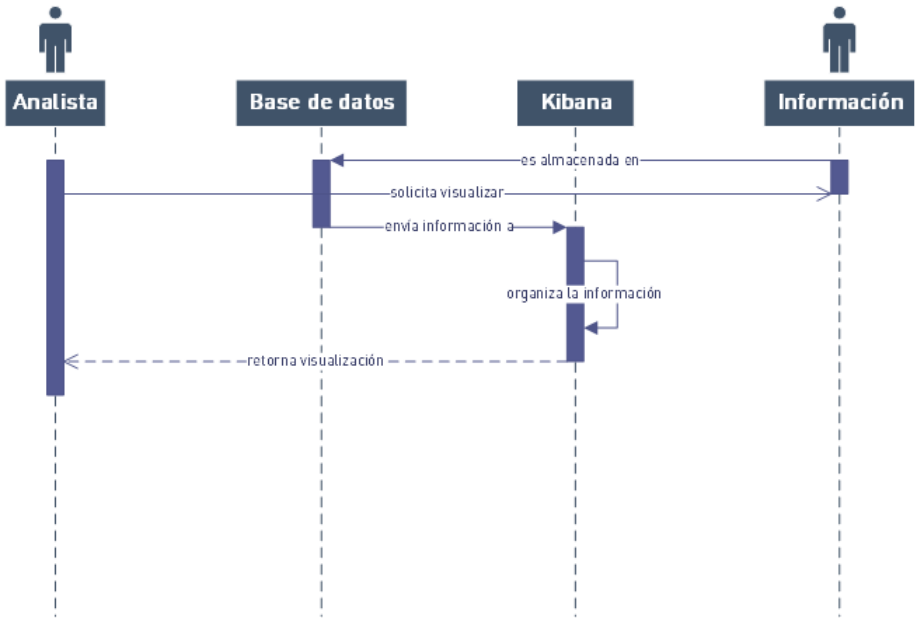


1.3.6 Información es visualizada por el analista

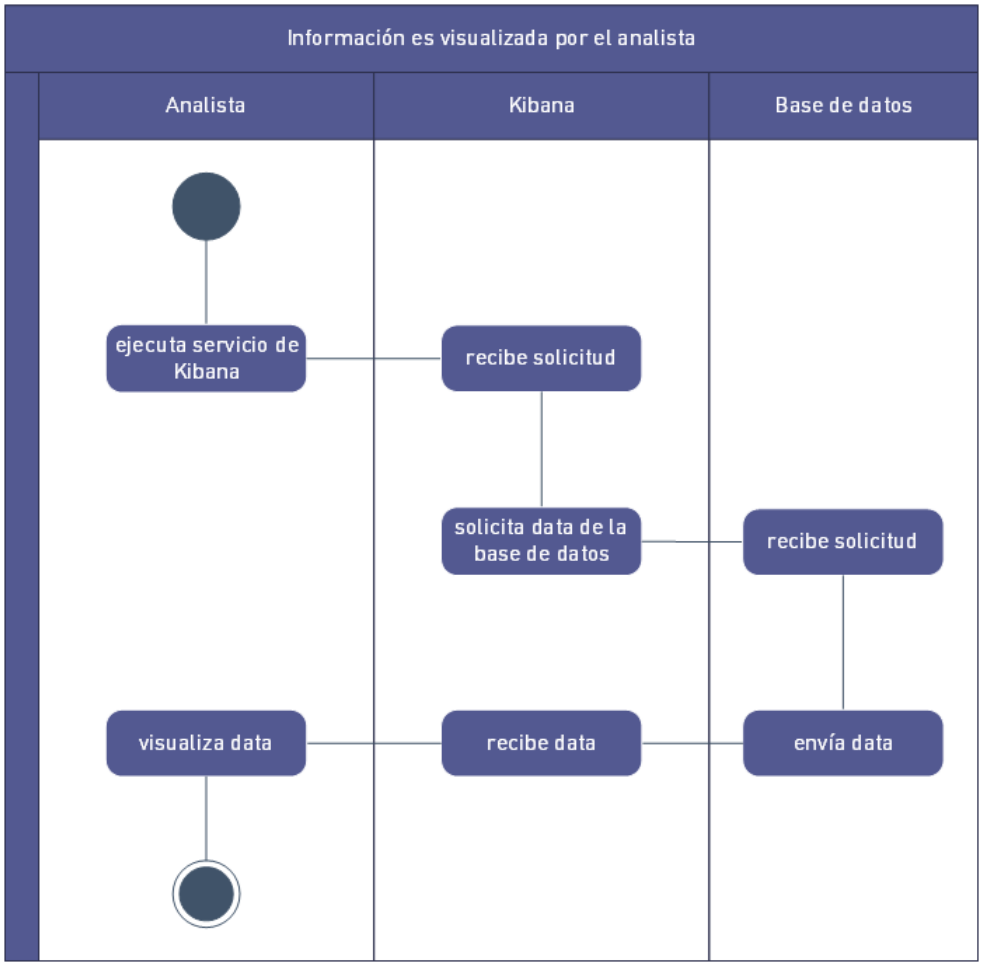
La base de datos Elasticsearch envía la data a Kibana para la visualización, donde luego un analista puede accederla.

1. Caso de uso

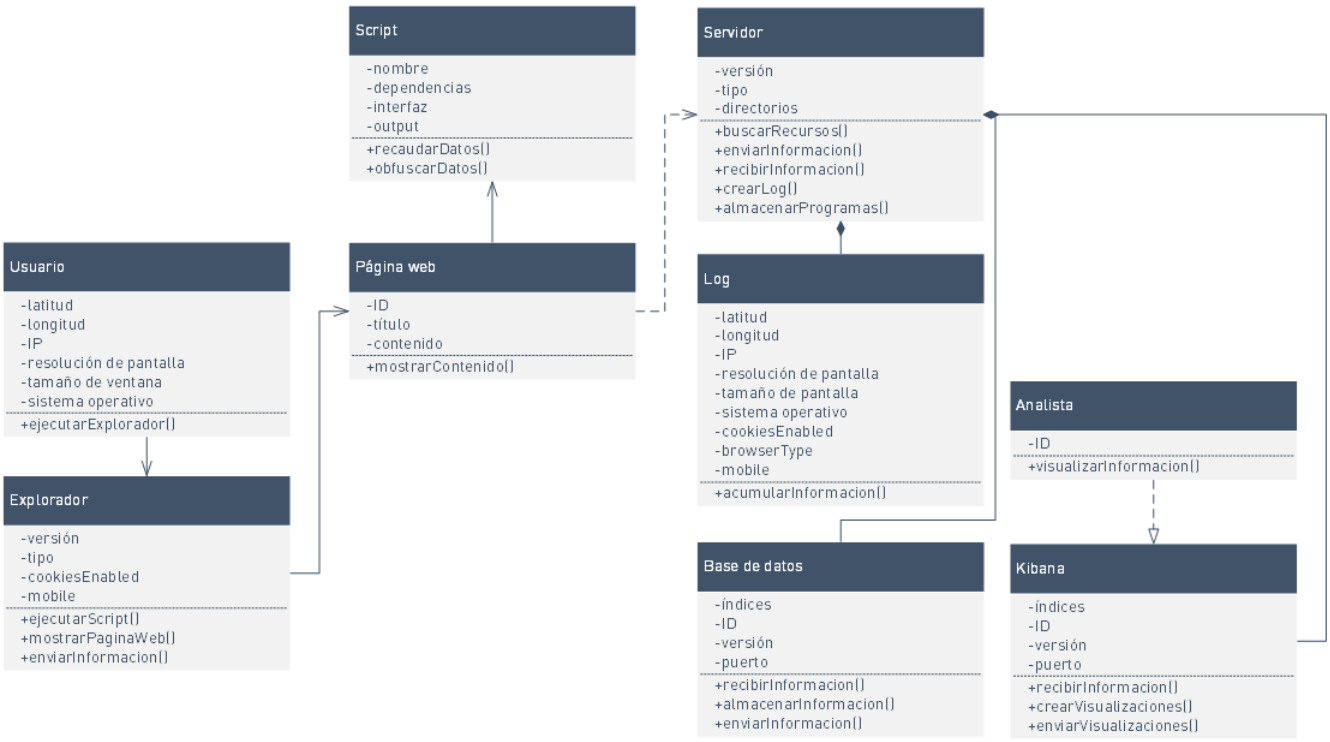
2. Diagrama de secuencia



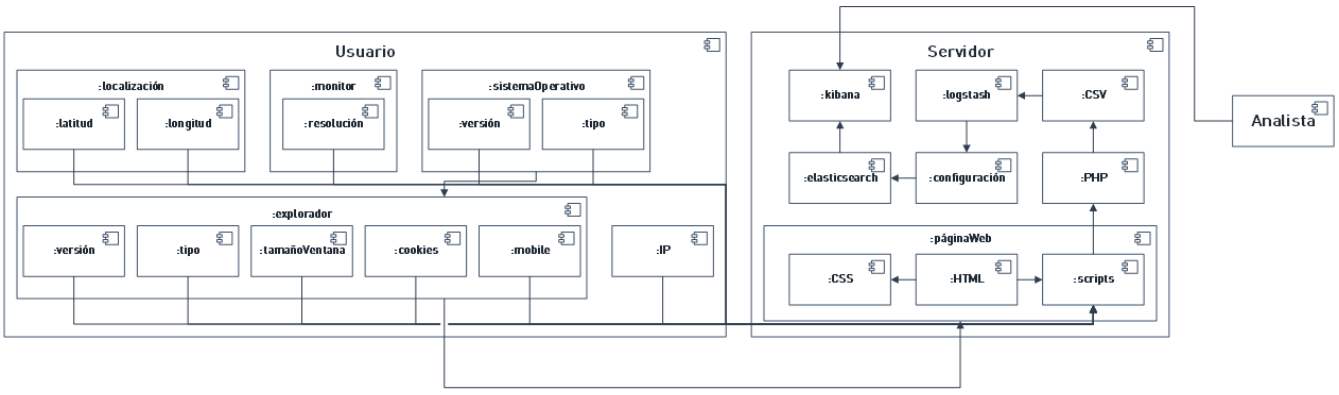
3. Diagrama de actividades



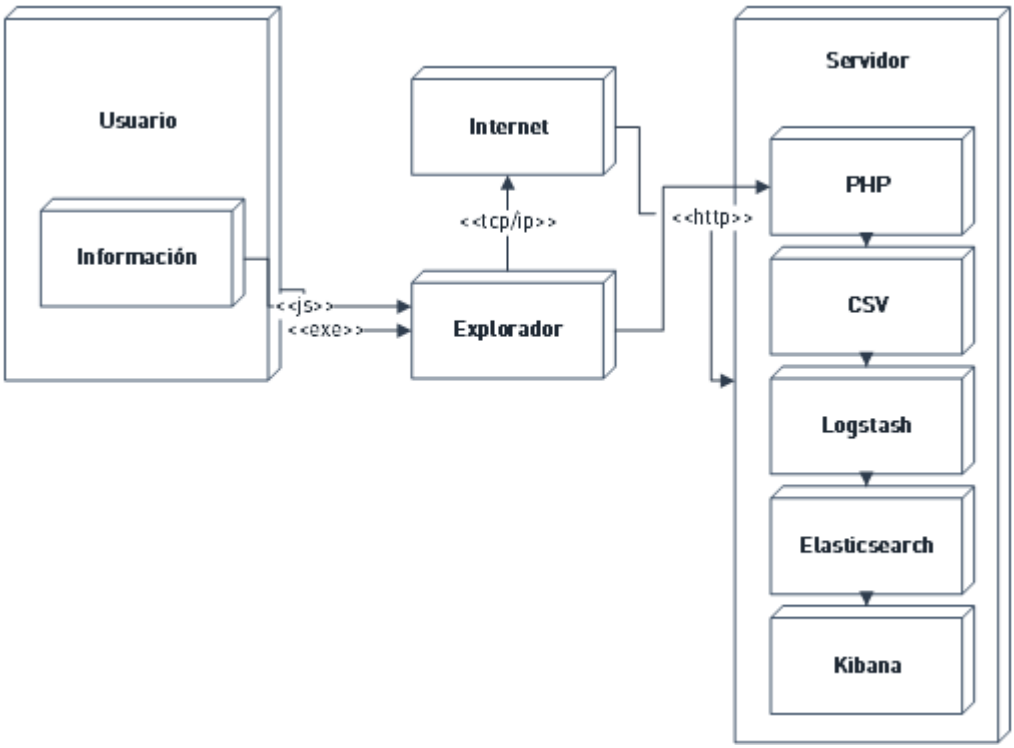
1.4 Diagrama de clases



1.5 Diagrama de componentes



1.6 Diagrama de despliegue



2 Código

2.1 Lenguajes de programación

2.1.1 HTML5

Se utiliza HTML5, última y actual versión de HTML, para la presentación de la información de la página web ante el usuario. Se caracteriza por tener compatibilidad con la mayoría de exploradores modernos, tanto en dispositivos móviles como en ordenadores.

2.1.2 CSS3

Se utiliza CSS3 para el diseño de la página web ante el usuario. Además se incorpora un *framework* de diseño de Google Material Design llamado Materialize que incorpora sus propios archivos CSS y otras funciones que van de la mano en JavaScript y jQuery.

2.1.3 JavaScript

Se utiliza JavaScript para la lógica detrás de la página web, tanto la recolección de datos como la obfuscación de los mismos. Se separan los distintos scripts de acuerdo a sus funcionalidades.

2.1.4 jQuery

Se utiliza jQuery para modificar los elementos de la página web de acuerdo al contexto, sea para ajustar el contenido de la página actual o cargar los elementos del framework de diseño.

2.1.5 Ajax

Se utiliza Ajax para realizar un POST desde el lado del cliente hacia el lado del servidor con la data recolectada.

2.1.6 PHP

Se utiliza PHP para recibir la data del POST, organizarla y escribirla en un archivo CSV.

2.2 Archivos de configuración

2.2.1 Logstash.conf

El archivo de configuración de Logstash nos permite establecer qué acciones se desean realizar con la información de entrada. Primero se lee el archivo CSV, se almacena cada columna como un atributo de una tabla en una base de datos, se transforman los valores de entrada a cadenas y números reales con punto flotante (ie. *float*) y se establece qué índice (ie. tabla) se desea utilizar y qué plantilla (ie. *template*) se va a aplicar al ingresar los datos a Elasticsearch.

2.2.2 elastic-template.json

El archivo plantilla determina la manera como se trabajarán los datos una vez que ingresen a Elasticsearch. Aquí se determina que las cadenas deben ingresar también de manera cruda (ie. *raw*), que la latitud y longitud deben juntarse para crear un objeto *geopoint* que nos permita visualizar las coordenadas como puntos en un mapa.

2.3 Herramientas de código

2.3.1 GitHub

GitHub es una plataforma web que permite almacenar repositorios Git y ofrece las funcionalidades heredadas de este para gestionar un mejor control de cambios. A través de esta herramienta se puede programar de manera colaborativa, reportar problemas, plantear mejoras y realizar un seguimiento del producto de software.

2.3.2 Sublime Text

Sublime Text es un editor de textos diseñado para trabajar con código. El espacio de trabajo funciona con espaciado singular (ie. *monospace*) lo cual permite asignarle el mismo espacio a cada carácter y codificar de manera ordenada. Este editor lo utilizamos para trabajar todos los archivos que se producen en el sistema. Permite además incorporar *plugins* para realizar ediciones

en HTML más sofisticadas, separar la pantalla para trabajar presentación y diseño (ie. HTML y CSS) a la vez y reconocer errores tipográficos o sintácticos en los distintos archivos.

2.3.3 Vim

Vim es un editor de textos a través de consola que se utiliza especialmente cuando se trabaja solo en consola. En el proyecto lo utilizamos para realizar algunas modificaciones rápidas en los archivos y para editar el código que está en el lado del servidor.

2.3.4 Cmder

Cmder es un *software bundle* que incluye muchas herramientas de consola para poder realizar acciones más complejas que con la consola *default* de Windows. Incorpora distintos comandos muy comunes en Linux a sus equivalentes en Windows y permite trabajar directamente con Git y Vim al añadirlos automáticamente al *path*.

2.4 Reglas de código

2.4.1 Comentarios

El código se encuentra comentado de acuerdo a qué tan complejas son las actividades de lógica que se realizan. Por ejemplo, el archivo HTML no tiene comentarios en los elementos *div* como *head, body* o *nav* pero sí comentarios cuando se cargan bibliotecas o se realiza una carga de datos a través de un script. Los scripts están bastante comentados y algunos archivos de configuración no permiten comentarios.

2.4.2 Número de caracteres

El número de caracteres por línea se escogió de ser 120 y se aplica a todos los archivos de código del proyecto.

2.4.3 Declaración de variables

Las variables se declaran en la parte superior del código de manera que se inicializan y ejemplifican de qué se trata el código. Esta regla aplica especialmente a los scripts de recaudación de datos que trabajan con muchas variables.

2.4.4 Encapsulamiento

La lógica de la plataforma se encuentra en los distintos scripts, los que a su vez son completamente independientes entre ellos. Cada script está encapsulado en un archivo separado y una refactorización de una variable solo afectaría localmente al script en cuestión.

2.4.5 Interfaces

Las funciones y métodos de JavaScript trabajan con una sola interfaz que recibe un determinado número de parámetros. Para las funciones más complejas existen comentarios sobre las interfaces y sus entradas, pero por lo general se manejan nombres significativos y el uso de las interfaces es fácil de intuir.

3 Test

3.1 Tipos de tests

Los tipos de test empleados consisten de:

1. Testeo unitario: en el que se comprueba que la plataforma satisface los requerimientos y que la estructura está de acuerdo con el diseño planteado.
2. Testeo funcional: en el que se valida que la información esté conforme a las especificaciones y se revisa si es que las funciones implementadas proveen los servicios y métodos requeridos.
3. Testeo de desempeño: en el que se verifica si es que la plataforma tiene la capacidad adecuada y el tiempo de respuesta esperado.
4. Testeo de condiciones de límites: en el que se prueba la reacción de la plataforma ante valores extremos para ver si funcionan. Por ejemplo, se probó que ocurría con la latitud y longitud si es que era un número fuera del rango.
5. Testeo alfa: en el que se despliega el sistema para uso de los desarrolladores para explorar funciones y tareas.
6. Testeo beta: en el que el sistema se despliega a usuarios externos para poder determinar qué funciones explorar.
7. Testeo de aceptación y calificación: en el que se verifica que el software cumpla con los requerimientos y que esté validado por usuarios para que se realicen las funciones esperadas antes del despliegue final.

3.2 Herramientas de tests

El principal testeo del código se realiza a través de Sublime Text y los plugins Codebug y Xdebug que permiten detener la ejecución del código JavaScript y PHP para entender el estado de las variables y su composición y así encontrar errores.

También se utilizó mucho exploradores como Google Chrome y Firefox Developer Edition para probar la página y utilizar las herramientas de desarrollo web disponibles como la consola para ver si es que han ocurrido errores.

Finalmente, se utiliza un servidor Apache a través del paquete XAMPP que nos permite probar la plataforma localmente antes de desplegarla a Amazon Web Services.

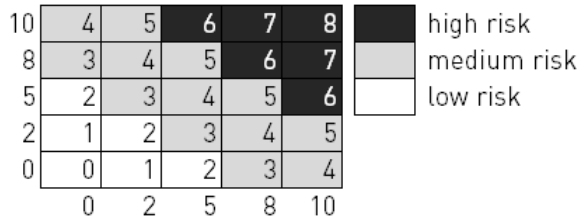


Figure 2: Matriz de riesgos

4 Mantenimiento

4.1 Estrategia de mantenimiento

El mantenimiento de la plataforma se da luego de la implementación, a medida que surgen mejoras o problemas en el sistema de analítica. Actualmente, el proyecto está en GitHub completamente *open source* lo que nos permite abrir *tickets* y recibir apoyo de la comunidad. Las mejoras al núcleo de la plataforma se realizan constantemente y de manera gratuita pero los problemas particulares en caso un cliente necesite de la plataforma se darán con un costo extra. Asimismo, si es que hubieran mejoras propuestas para casos particulares del cliente, se evaluaría cada circunstancia y se determinaría un precio. Contaremos además con un paquete de mantenimiento en el que agendaremos mantenimiento preventivo, documentaremos el historial del mantenimiento y generaremos reportes.

5 Análisis de riesgos

5.1 Seguridad

Los riesgos de seguridad son que el servidor de Amazon sea interceptado por un tercero (RS1), que alguien tenga acceso a nuestros archivos internos, como por ejemplo encontrando el DNS de nuestra instancia y buscando en la estructura del servidor (RS2) y que nuestras computadoras reciban acceso indebido por terceros y se vulnere la integridad o identidad del proyecto (RS3).

- RS1: impacto 10, probabilidad 2, riesgo 5
- RS2: impacto 5, probabilidad 2, riesgo 5
- RS3: impacto 8, probabilidad 5, riesgo 5

El riesgo de seguridad es moderado, ni muy alto ni muy bajo. Esto nos inclina a continuar con los esfuerzos de seguridad para evitar que incremente la probabilidad de que ocurran y se torne en un riesgo alto.

5.2 Privacidad

Los riesgos de privacidad más importantes soon que los reportes de la data sean visualizados por personas cuyo acceso no es planificado (RP1), que la data sea personalmente identificable

(RP2) y que la data sea cruzada con otra para obtener información personal de nuestros clientes, vulnerando su privacidad y seguridad (RP3).

- RP1: impacto 5, probabilidad 5, riesgo 4
- RP2: impacto 8, probabilidad 2, riesgo 4
- RP3: impacto 10, probabilidad 2, riesgo 5

El riesgo de privacidad es moderado-bajo. Esto nos dice que nuestra data no es muy personalmente identificable y por lo tanto no es muy probable de afectarnos a nosotros como desarrolladores de la plataforma ni a nuestros usuarios. No obstante, hay uno que otro detalle que se debe considerar por lo que continuarán los esfuerzos de privacidad, especialmente si se están considerando nuevas variables que podrían incrementar el riesgo.

6 Presupuesto y cronograma

6.1 Equipo

Jose Naranjo: Encargado de diseñar la página web en HTML, el diseño en CSS, la lógica de recaudación y obfuscación de datos en JavaScript y jQuery; **Javier Zárate**: Encargado de realizar la conexión entre el lado del cliente y el lado del servidor (Ajax y PHP), Elasticsearch y Kibana; **Samantha Pacheco**: Encargada de realizar las visualizaciones en Kibana y generar el marco legal correspondiente; **Isaías Hoyos**: Encargado de realizar la conexión entre Logstash y Elasticsearch, y de realizar las visualizaciones en Kibana.

6.2 Entregables

Los entregables son: la página web en formato HTML5, los scripts en JavaScript, el diseño en CSS3, la lógica de escritura del servidor en PHP, el log en CSV, los archivos de configuración en CONF, las plantillas en JSON y los resultados de la visualización de Kibana en un *dashboard iframe* en formato HTML.

La página web con su diseño original se presentó el 17 de junio de 2016, con su nuevo diseño el 29 de junio de 2016 y ese mismo día se realizaron las visualizaciones en Kibana.

6.3 Costo de la plataforma

El costo de la plataforma se separa en el desarrollo, implementación y mantenimiento. El desarrollo involucra los costos relacionados a la mano de obra detrás de la programación de la página web y del servidor; la implementación, los costos relacionados a los activos necesarios como servidores; el mantenimiento, las correcciones, ajustes y mejoras luego de la implementación. Todo el software utilizado es en sí *open source* y grautito, desde las herramientas de código hasta el sistema de bases de datos y visualización.