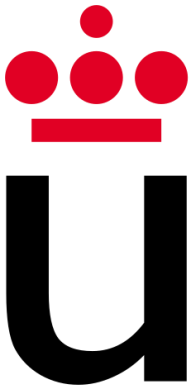


Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso Académico 2022/2023

Trabajo de Fin de Grado

ANÁLISIS DE BOTS DE TWITTER USANDO LA LEY DE BENFORD

Autora: Yamilé Lucero Dipaz Roa

Tutora: Almudena Sierra Alonso

RESUMEN

En este Trabajo de Fin de Grado (de ahora en adelante TFG) se analizarán cuentas de usuarios de ***Twitter*** con el objetivo de conocer si se trata realmente de usuarios reales o de ***bots***. Para este análisis, se va a desarrollar la aplicación "Newcomb".

La metodología utilizada para este análisis es la ley de Benford. Más adelante se detallará en profundidad sobre esta ley, pero se podría decir que, en resumen, con la ley de Benford se puede afirmar si algo es un fraude o no partiendo de un conjunto de datos.

En el caso de este TFG, el conjunto de datos a analizar será el número de seguidores de los seguidores de la cuenta de ***Twitter*** dada como muestra.

Este TFG se basa en una aplicación web cuya interfaz permite al usuario introducir el *username* de un usuario de ***Twitter*** e internamente mediante peticiones ***API*** obtiene la información de los seguidores del usuario, después de aplicar la ley de Benford se podrá decir cuál es el porcentaje estadístico que tiene esa cuenta para no ser considerada un ***bot***.

Esta aplicación consta de una **base de datos**, la cual existe para agilizar la obtención de datos ya que la ***API*** de ***Twitter*** tiene límites de peticiones por tiempo, e incluso a veces puede no estar disponible por sobrecarga de peticiones. En esta **base de datos** se guarda la información de cuentas analizadas por primera vez para que, en caso de volver a pedir un análisis de la misma cuenta, esta información se extraiga de allí y no desde una nueva petición ***API*** a ***Twitter***.

La información almacenada en la **base de datos** de una cuenta de Twitter es el *username*, el número de *followers* y el *id* de la cuenta de ***Twitter*** del usuario. Este último dato es el más importante ya que es un valor único y es el que evita una segunda iteración sobre el número de seguidores del usuario.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Las redes sociales suponen un punto de visibilidad y conexión con una comunidad, en este caso si nos centramos en Twitter, podemos ver que en esta red social en particular se puede compartir contenido de forma innovadora, comunicarnos con personas de otra parte del mundo, seguir temas de interés y de actualidad, etc…

Actualmente, está normalizada la presencia de bots en cualquier red social ya que estas nos permiten utilizar sus APIs para automatizar acciones, por ejemplo, en Twitter se pueden poner tweets, reaccionar a contenido, enviar mensajes directos a usuarios, empezar a seguirlos…

Gracias a la Ley de Benford, entre otras cosas, podemos llegar a identificar bots según sus cálculos, esto acción es importante ya que los bots pueden degradar la calidad de la información en las redes sociales con la difusión de información falsa.

Conocí esta ley gracias a una serie documental de Netflix llamado Superconectados, dónde un periodista científico llamado Latif Nasser se encarga de explicar las cosas que nos rodean en nuestro día a día. Después de ver el documental quería comprobar por mí misma la validez de la ley.

También he de decir que, aunque esta ley lleve el apellido del físico Frank Benford, el primero en descubrirla fue el matemático Simon Newcomb unos años antes. Benford fue el que hizo algunas pruebas y se encargó de postular la ley como “ley de los números anómalos” o “ley del primer dígito”.

Es por esto último que la aplicación desarrollada en este TFG para demostrar la ley de Benford se llama “Newcomb”.

A lo largo de este documento, se comentarán los objetivos que se plantearon para el desarrollo de la aplicación y la planificación empleada para ello.

También se ahondará sobre los puntos técnicos que forman parte del *core* del software como el propio algoritmo, la base de datos y el código del proyecto.

2. OBJETIVOS

2.1 Descripción del problema

En general, los bots pueden llegar a ser herramientas muy útiles, por ejemplo, existen bots de monitorización que consultan una web cada 30 segundos y en el caso de que la web no responda, este bot generará una alerta para el administrador avisando de que el entorno está caído.

Particularizando en Twitter, podemos ver algunas cuentas llevadas por bots que sirven para informar sobre las novedades de Netflix, de terremotos alrededor del mundo, otros te ayudan a descargar vídeos o incluso como recordatorios de otros tweets.

Sin embargo, estos bots también pueden ser confundidos con usuarios reales y pueden ser utilizados para promover información falsa en masa creando hashtags que se vuelven tendencia. Es aquí donde surge el verdadero problema y se ve la importancia de identificarlos.

Por poner un ejemplo del problema que pueden crear los bots, existe una cuenta llamada BotSentinel que se encarga de analizar la desinformación y el acoso en Twitter. A principios de 2022, esta empresa se encargó de analizar las diferentes campañas de acoso a la actriz Meghan Markle, la duquesa de Sussex casada con el príncipe Harry de Inglaterra. En este análisis se detectó que 83 cuentas eran responsables del 70% del odio recibido, lo que quiere decir que estas cuentas de forma organizada lograban alcanzar un nivel de odio lo bastante severo para afectar en la decisión de Meghan Markle y Harry. En este enlace se puede ver el reporte completo https://botsentinel.com/reports/documents/duke-and-duchess-of-sussex/report-01-18-2022.pdf.

2.2 Hipótesis de solución

En primer lugar, hay que explicar lo que es la ley de Benford. Se trata de una fórmula matemática que analiza un conjunto de números aleatorios y comprueba que el 30.1% de los números empiezan por el dígito 1, el 17.6% empiezan por el dígito 2, el

12.5% por el dígito 3, el 9.7% por el dígito 4, el 7.9% por el dígito 5, el 6.7% por el dígito 6, el 5.8% por el dígito 7, el 5.1% por el dígito 8 y el 4.6% por el dígito 9. Según esta gráfica, el porcentaje que deben cumplir los primeros dígitos de los números sería como se muestra en la *Ilustración 1*:

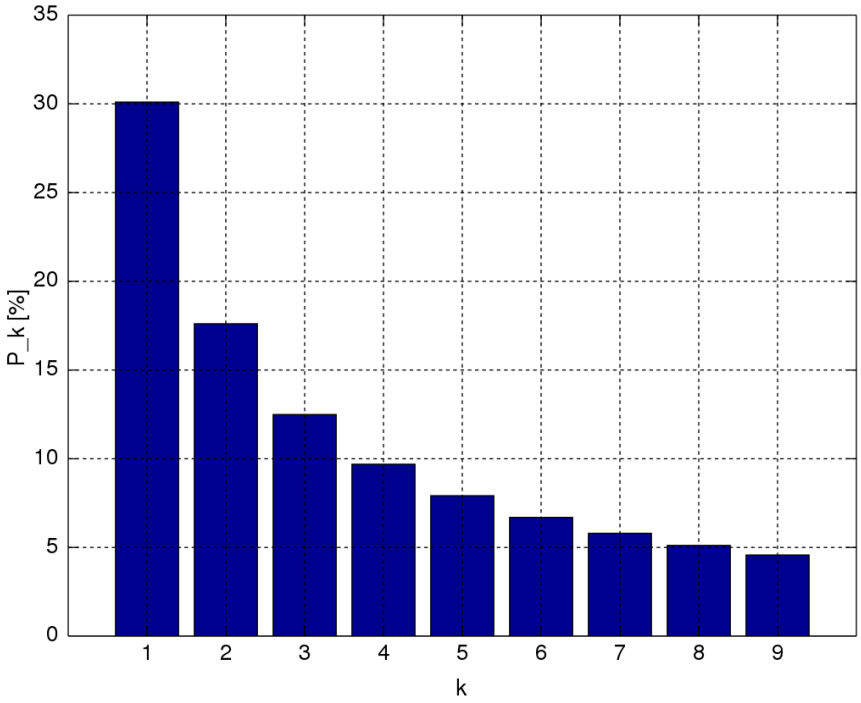


Ilustración 1. Gráfica de la Ley de Benford

Matemáticamente, podemos decir que el primer dígito no nulo ***n*** **(n = 1, 2, 3, 4, 5, 6, 7, 8, 9)** tiene una frecuencia igual a $\log_{10}(n + 1) - \log_{10}(n)$.

Esta ley es utilizada por el Departamento de Hacienda de Estados Unidos en el que, si existe una mínima desviación de algún porcentaje y no se cumple la Ley de Benford, esto significa un indicio para investigar el posible fraude fiscal.

Volviendo al tema del TFG, la idea es emplear esta ley para analizar cuentas de Twitter y ver si se trata de posibles bots. Para ello es importante tener de antemano cuentas de bots identificados, por ejemplo, algunos de esta web https://borjaruizcm.com/mejores-bots-twitter/ o simplemente, ir a los tweets citados del tweet anunciado por TwitterDev el 9 de Febrero de 2023, anunciando con una semana de antelación que el uso de la API de Twitter será de pago, por lo que miles de cuentas de bots se verán obligados a cerrar ya que no van a poder ser mantenidas *(ver Ilustración 2)*.

Ilustración 2. Comunicado sobre el pago de la API de Twitter

2.3 Objetivo principal

El objetivo de este TFG es hacer una aplicación web que dado un *username* de una cuenta de Twitter se encargue de recoger la cantidad del número de seguidores de los seguidores de esta cuenta utilizando la propia API de Twitter para ello.

Con estos datos se aplicará la Ley de Benford y se visualizará en una gráfica cómo se cumple o no la ley.

2.4 Objetivos parciales

Este objetivo principal se divide en los siguientes objetivos parciales:

- Acceso a la API de Twitter.
- Desarrollo de la base de datos.
- Desarrollo de Newcomb.

2.4.1 Acceso a la API de Twitter

El primer paso es tener acceso a la API de Twitter para poder obtener los datos para aplicar la Ley de Benford de forma automatizada.

Se solicitó el acceso a la API dando detalle del uso que se quería hacer con ella, y se obtuvieron credenciales para poder utilizarla con permisos de lectura ya que para este proyecto no se necesitan permisos de escritura. Es decir, con el acceso que se recibió por parte de Twitter la API no puede publicar un tweet, sino que recopila los datos de la información ya existente. La documentación de la API está en la dirección https://developer.twitter.com/en/docs/twitter-api.

La conexión con esta API está facilitada por una librería de Java llamada twitter4j (https://twitter4j.org/).

2.4.2 Desarrollo de la base de datos

Para mantener la información obtenida de Twitter y no tener que volver a recuperarla si se necesitaba analizar de nuevo, se desarrolló una base de datos. Se van a mencionar las siguientes cuestiones a tener en cuenta para el desarrollo de ésta:

- La llamada al endpoint de la API para obtener la información de los seguidores es https://api.twitter.com/1.1/users/show.json. Tenía una limitación de 15 llamadas cada 15 minutos, es decir, el análisis de cuentas con muchos seguidores requería mayor tiempo a la aplicación, este fue la razón principal para la creación de la base de datos.
- El objetivo del estudio de la Ley de Benford no requiere que los datos sean exactos ya que unos valores aproximados sirven de la misma manera. Es decir, si por ejemplo hoy revisamos los datos de los seguidores de un usuario y mañana los volvemos a consultar de nuevo, es muy poco probable que el número haya variado, y en caso de que sí lo haya hecho, lo más seguro es que no haya cambiado el valor de su primer dígito, que es el único dato que tenemos en cuenta.
- La consulta de datos en la base de datos implica mayor agilidad en cuentas analizadas previamente.

Para evitar caer en las limitaciones de Twitter, la aplicación antes de analizar una cuenta comprueba si tenemos por lo menos el 80% de los seguidores de ésta. En ese caso el análisis consume la información almacenada en base de datos, en caso contrario se analizan los seguidores de la cuenta para almacenar esos datos.

Esta base de datos local está creada y controlada por el software MySQL Workbench 8.0 CE.

2.4.3 Desarrollo de Newcomb

En primer lugar, hay que comentar que un proyecto Spring es un framework de Java que se utiliza para crear aplicaciones web escalables y fáciles de mantener.

El esqueleto de Newcomb es una plantilla de Spring con las configuraciones que se pueden ver en la *Ilustración 3*:

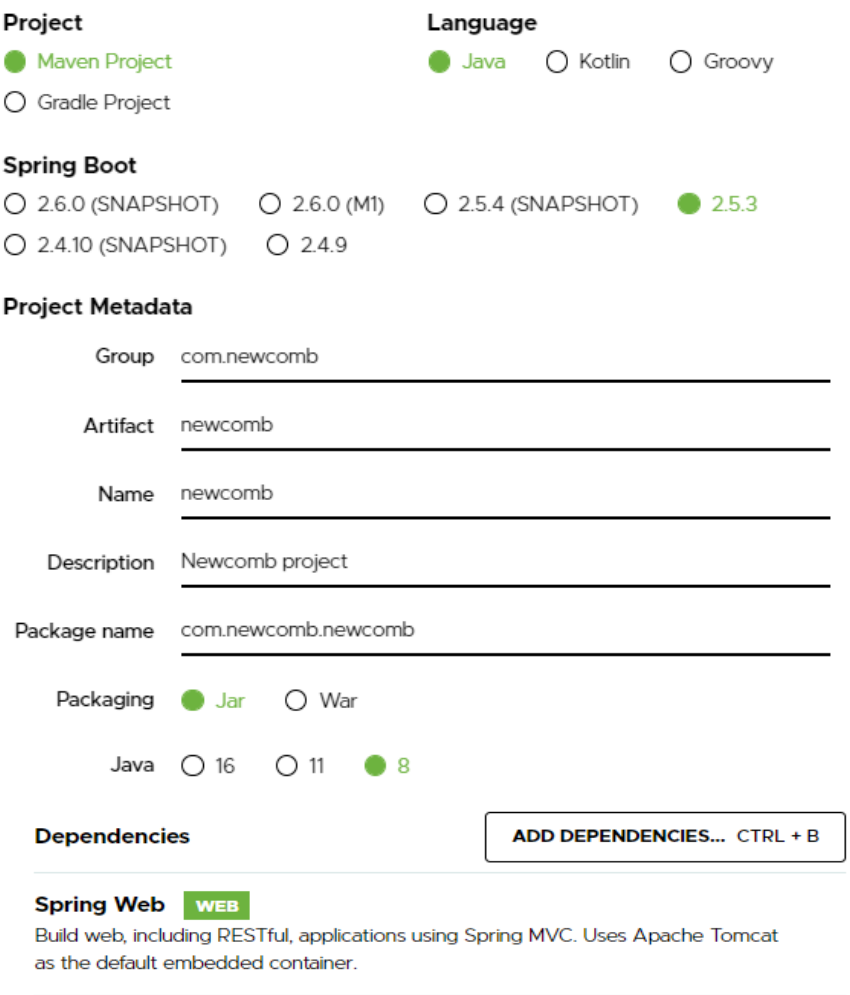


Ilustración 3 Spring initializr

Estas plantillas se pueden encontrar en la siguiente página web: https://start.spring.io/.

2.5 Metodología empleada para el desarrollo de Newcomb

La metodología empleada fue incremental, es decir, el avance de la aplicación web fue evolutivo, con cambios pequeños que agregaron nuevas funcionalidades al producto final.

Este tipo de metodología es flexible y permite que se puedan tratar los problemas que surgen a lo largo de un desarrollo.

2.6 Planificación para el desarrollo de Newcomb

En cuanto a la planificación aplicada durante el proyecto, se ha utilizado un diagrama de hitos en el que se establecieron los hitos y el tiempo dedicado en cada uno de ellos.

Hito	Estimación
Hito 1 Base del proyecto	
Hito 1.1 Implementación de la base de un proyecto Spring Boot	3h
Hito 1.2 Implementación de Algoritmo de la ley de Benford	1 día
Hito 2 Componentes	
Hito 2.1 Creación de base de datos	6h
Hito 2.3 Creación de servicio web (interfaz de usuario)	10h
Hito 3 Detalles	
Hito 3.1 Añadiendo estilo a la web (Bootstrap y .css)	5h
Hito 3.2 Conexión back-end y front-end	2 días
Hito 3.3 Refactorización de código	5h
Hito 3.4 Solución de problemas encontrados	4h
Hito 3.5 Spike de librería chart.js para gráficos	3h
Hito 3.6 Uso de chart.js en la interfaz	4h
Hito 4 Documentación	
Hito 4.1 Memoria	5 días

Tabla 1 Diagrama de Hitos

Los problemas que surgieron a lo largo del desarrollo y afectaron a los tiempos de la planificación inicial fueron:

- En primer lugar y el más importante, como ya se adelantó en el punto **Desarrollo de la Base de Datos**, tener datos almacenados en BD no estaba planificado ya que el análisis de una cuenta se puede hacer usando exclusivamente las llamadas a la API. El problema que se vio es que esto era muy lento debido a la cantidad de seguidores de las cuentas analizadas y a las limitaciones de la API, siendo la más importante la limitación de las peticiones a ésta https://developer.twitter.com/en/support/twitter-api/error-troubleshooting, en la *Ilustración 4* podemos ver el error más detallado del límite de llamadas:

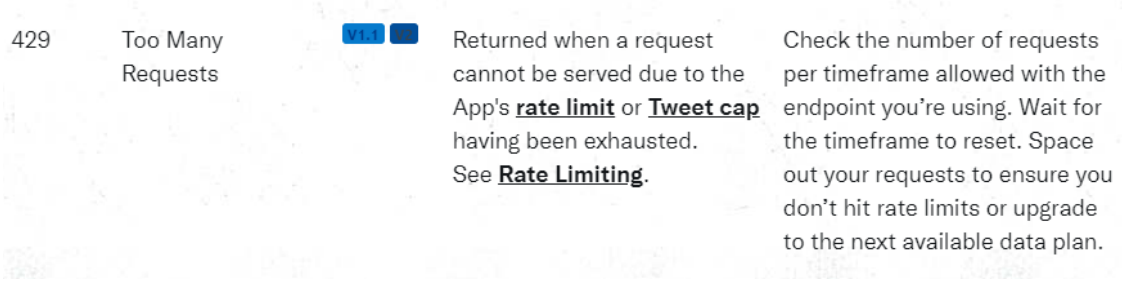


Ilustración 4 Error 429 de la API de Twitter por demasiadas peticiones

Esta limitación obliga a cargar la información de 899 cuentas en 5 minutos aproximadamente para evitar que se produzca un error 429.

- Problema al tratar los resultados devueltos por la API de Twitter cuando un usuario tiene más de 5000 seguidores. Una vez hecha la implementación de lectura de la información de las cuentas se producían errores cuando éstas tenían más de 5000 seguidores, en este caso, Twitter devuelve estos resultados paginados, y para recorrerlos en su totalidad se requirió de una pequeña implementación extra en la lectura de la información de las cuentas, para más información sobre la documentación de esta paginación de resultados está la siguiente url https://developer.twitter.com/en/docs/twitter-api/v1/pagination.

3. DESCRIPCIÓN INFORMÁTICA

3.1 Especificación de los requisitos del software a desarrollar

De acuerdo con el ciclo de vida de desarrollo del software, se establecieron en primer lugar los siguientes requisitos de la aplicación web:

- Funcionalidad: la aplicación debe aplicar la ley de Benford al número de seguidores de los seguidores de una cuenta analizada y estos resultados deben poder ser visibles para el usuario de alguna forma gráfica.
- Usabilidad: la interfaz debe ser intuitiva y navegable. También debe utilizar un diseño con una estética atractiva para la web cuyo color debe ser compatible con los colores de la plataforma de Twitter.
- Compatibilidad: Newcomb debe ser adaptado para los navegadores web como Chrome, Firefox y Safari y para dispositivos móviles, es decir, debe ser responsive.
- Velocidad: el tiempo para poder visualizar los resultados del análisis dado una cuenta debe ser inmediato para dar feedback al usuario.

3.2 Diseño de la base de datos y del software

Se utiliza una BD de MySQL relacional debido a la paridad de los datos que se introducen en el análisis. Estos datos se organizan en la tabla **users**:

Nombre	Descripción	Tipo de valor
ID	Identificador de la tabla	int (entero)
Username	Nombre del usuario de twitter	Varchar(45) (cadena de caracteres)
Followers	Número de seguidores	int (entero)
TwitterId	Identificador del usuario en twitter	int (entero)

Tabla 2 Usuarios y sus campos

Siendo el campo **twitterId** la clave primaria de la tabla **users** ya que se trata de un valor único no nulo que concuerda como identificador de las cuentas almacenadas en la BD de Twitter.

La conectividad de la BD de MySQL con la aplicación Java será mediante la interfaz JDBC, las siglas de Java Database Connectivity, esta conexión permite hacer operaciones CRUD (Create, Read, Update y Delete) con los datos que se vayan manejando.

Esta BD fue gestionada por **MySQL Workbench**, herramienta visual para diseñar y administrar BBDD. Esta herramienta fue elegida debido a la fácil gestión y el uso habitual de esta a nivel laboral.

3.3 Implementación: tecnología usada para el desarrollo

Spring Boot

Spring Boot es un framework desarrollado para el trabajo con Java como lenguaje de programación. Se trata de un entorno de desarrollo de código abierto y gratuito. (School, 2022).

Trabajar con este framework como programadora, ayuda a enfocarte en el código y no en la arquitectura del proyecto. De hecho, el equipo de Spring tiene una web para generar el esqueleto de un proyecto Spring, la url es https://start.spring.io/.

El modelo Spring Boot se basa en la arquitectura Spring MVC también llamado modelo-vista-controlador. Este modelo se puede explicar como se indica en la *Ilustración 5*:

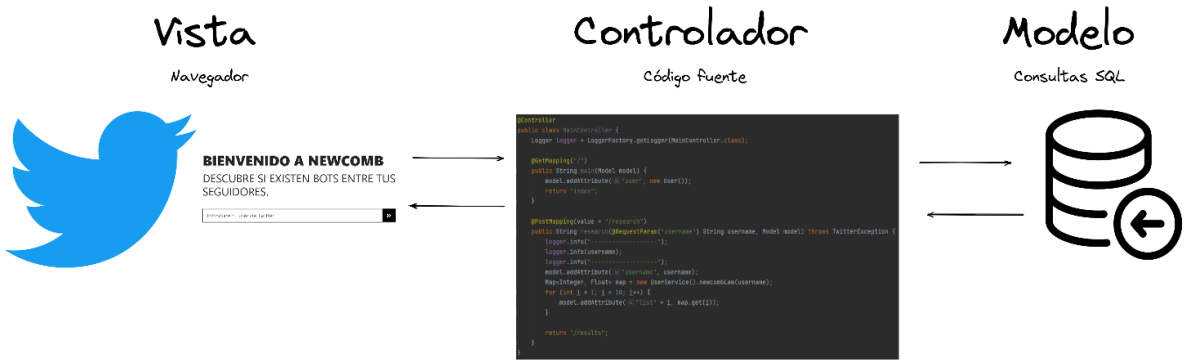


Ilustración 5 Modelo-Vista-Controlador

En este esquema, la vista es el navegador donde el usuario interactúa con la aplicación web, el controlador se encarga de recibir lo introducido por el usuario y gestionar lo que tiene que devolver al usuario consumiendo información almacenada en la BD.

Así mismo, podemos estructurar un proyecto Spring Boot como se muestra en la *Ilustración 6:*

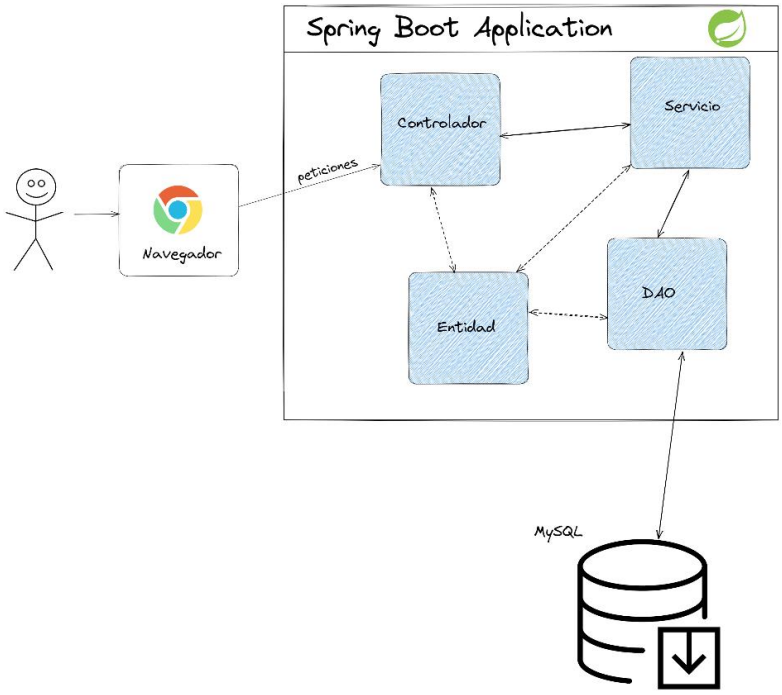


Ilustración 6 Estructura de proyecto Spring Boot

En esta ilustración podemos ver que el usuario interactúa con el navegador siendo esto la *vista*, ésta llama a la capa intermedia que es el *controlador*, que se encarga de gestionar las peticiones recibidas por el cliente apoyándose en servicios, entidades y DAOs, cuya información es proporcionada por la capa del *modelo*.

En la práctica y aplicado a este TFG, la estructura utilizada queda como se muestra en la *Ilustración 7:*

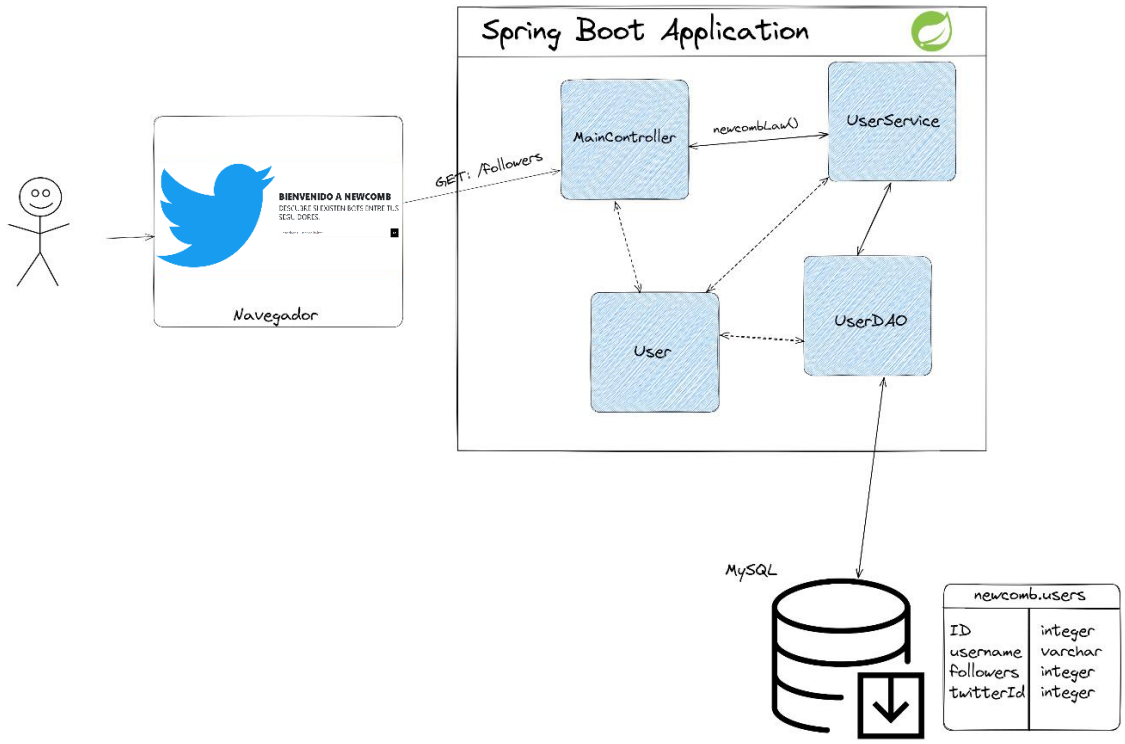


Ilustración 7 Estructura de la aplicación web Newcomb basada en Spring Boot

Maven

Apache Maven es una potente herramienta de gestión de proyectos que se utiliza para gestión de dependencias, como herramienta de compilación e incluso como herramienta de documentación. Es de código abierto y gratuita. (Alarcón, 2022)

Para este TFG, se utiliza Maven como gestor de dependencias, las dependencias utilizadas son las siguientes:

- Twitter4j-stream y twitter4j-core: se trata de librerías de Java que sirve como capa para comunicarse con la API de Twitter, una vez obtenido las credenciales de la API solo hace falta configurar estos datos para poder hacer las llamadas.
- Hibernate-jpa-2.1-api, hibernate-core, mysql-connector-java: se trata de librerías que gestionan y permiten la conexión a la BD.

- Spring-boot-starter-thymeleaf y spring-boot-starter-web: se trata de librerías que permiten utilizar springboot, estableciendo la arquitectura para su uso, esto es HTML, XML, JavaScript y CSS.

Java

Partiendo del uso de Spring Boot y Maven, el uso de este lenguaje es casi obligatorio, además de que la librería twitter4j es exclusiva para Java.

Este lenguaje también fue elegido por la familiaridad y el uso que se le da a nivel laboral.

Thymeleaf

Se trata de una plantilla de Spring para el uso de la capa de la vista con el controlador.

La documentación se encuentra en https://www.baeldung.com/thymeleaf-in-spring-mvc. En la *Ilustración 8* podemos ver que la llamada al botón con el círculo rojo hace una petición al controlador.

Ilustración 8 Interacción Vista - Controlador

En la *Ilustración 9*, se puede ver que Thymeleaf nos permite hacer la llamada al *action* con la información contenida en el input, en este caso, el contenedor que recoge el dato del username del usuario.

```
<form th:action="@{/research}" method="post">
    <div class="form-group">
        <div class="input-group flex-nowrap" >
            <input name = "username" id="usernar
            <button id="arrow-button" type="but
        </div>
    </div>
</form>
```

Ilustración 9 Invocación al controlador usando thymeleaf

El *action* invoca a esta llamada del controlador que a su vez hace la llamada al servicio UserService, en concreto al método newcombLaw() al que le pasa como parámetro el username del usuario. Estas llamadas se pueden ver en la *Ilustración 10*.

```
@PostMapping(value = "/research")
public String research(@RequestParam("username") String username, Model model)
    logger.info("--------------------");
    logger.info(username);
    logger.info("--------------------");
    model.addAttribute( s: "username", username);
    Map<Integer, Float> map = new UserService().newcombLaw(username);
    for (int i = 1; i < 10; i++) {
        model.addAttribute( s: "list" + i, map.get(i));
    }

    return "/results";
}
```

Ilustración 10 Código fuente del controlador que recopila la información para aplicar la ley de Benford

Chart.js

Es una librería de código abierto que permite la creación de gráficos dados unos datos. En este proyecto se ha empleado esta librería para mostrar en una gráfica los datos de la ley de Benford y la del usuario analizado para contrastarlos visualmente. La *Ilustración 11* muestra un ejemplo para el usuario de Twitter BlancaPaloma_rb.

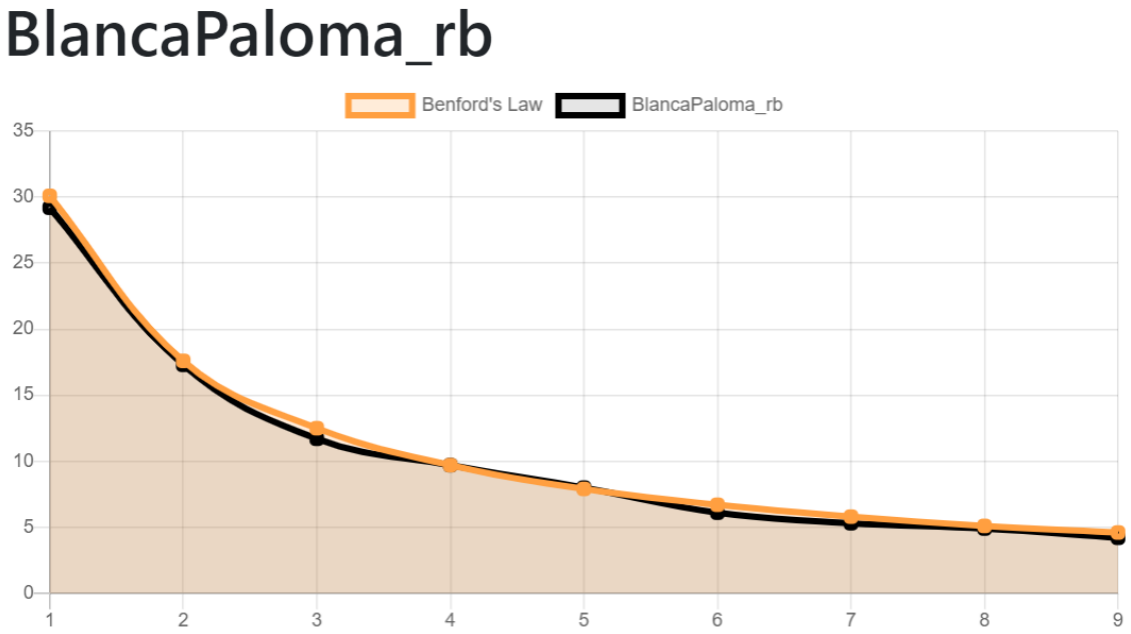


Ilustración 11 Gráfico de la Ley de Benford en el username BlancaPaloma_rb

Este gráfico es representado por el código mostrado en la *Ilustración 12*:

```
<script th:inline="javascript">
        const data = {
            '1' : [[${list1}]],
            '2' : [[${list2}]],
            '3' : [[${list3}]],
            '4' : [[${list4}]],
            '5' : [[${list5}]],
            '6' : [[${list6}]],
            '7' : [[${list7}]],
            '8' : [[${list8}]],
            '9' : [[${list9}]]
        };
        var ctx = document.getElementById('myChart').getContext('2d');
        var myChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: Object.keys(data),
                datasets: [{
                    data: [30.1, 17.6, 12.5, 9.7, 7.9, 6.7, 5.8, 5.1, 4.6],
                    label: "Newcomb's Law",
                    borderColor: "rgb(255, 0, 0)",
                    backgroundColor: "rgb(255, 0, 0, 0.1)",
                }, {
                    data: Object.values(data),
                    label: [[${username.toUpperCase()}]],
                    borderColor: "rgb(0, 0, 0)",
                    backgroundColor: "rgb(0, 0, 0, 0.1)",
                }]
            },
        });
</script>
```

Ilustración 12 Código javascript del gráfico de la ilustración 11

GitHub

Todo el código está almacenado en un repositorio privado de la herramienta GitHub, se trata de un gestor de versiones con historial para que todos los cambios/commits realizados en el proyecto puedan ser consultados en cualquier momento. Otra de las ventajas es el poder compartir el código con otros desarrolladores para recibir feedback sobre el proyecto, reportes de bugs y posibles mejoras.

Además, IntelliJ IDEA ofrece como servicio, hacer commits desde el propio IDE al repositorio de GitHub.

El enlace al repositorio se encuentra en https://github.com/Yami3006/newcomb, en la *Ilustración 13* podemos ver los últimso commits de los cambios de código de forma cronológica (de abajo hacia arriba).

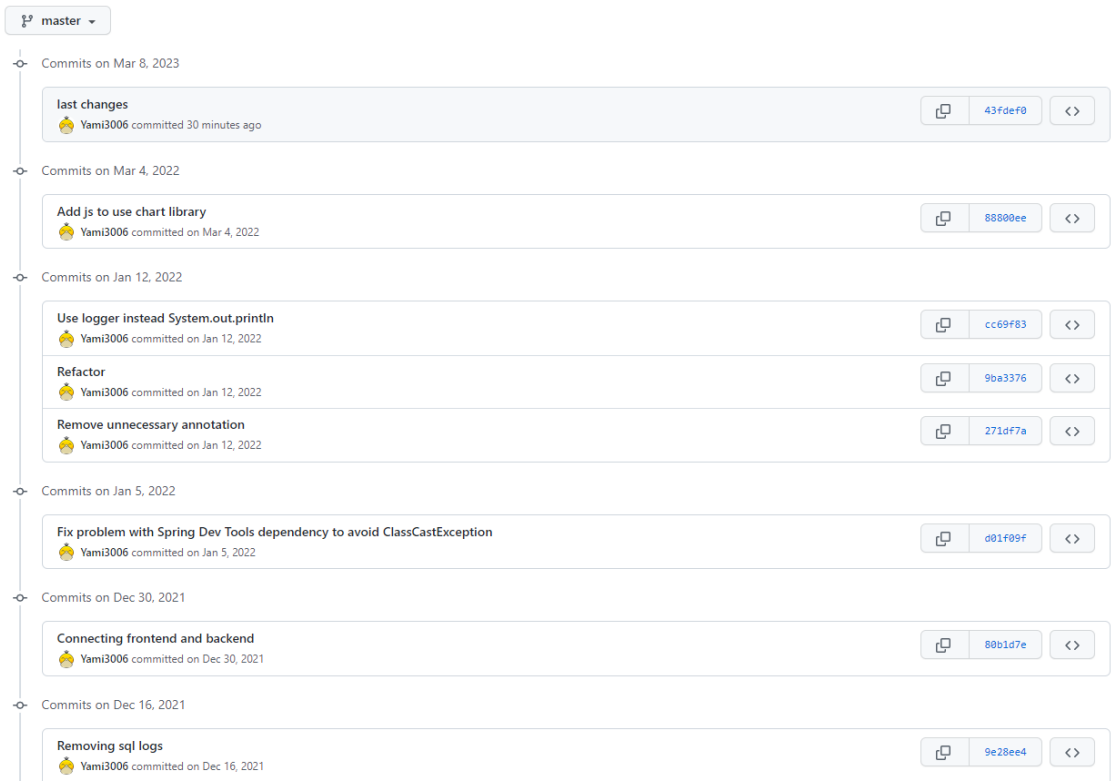


Ilustración 13 Commits en el repositorio de GitHub

SonarLint

Se trata de una extensión de IntelliJ IDEA que ayuda a detectar bugs, code smells, problemas de calidad, uso de métodos deprecados, etc… en el código que estás programando. Muchos de los commits son fixes de mejora para que el código del proyecto sea aprobado por SonarLint.

Como ejemplo vemos en la *Ilustración 14* un mensaje de SonarLint al utilizar *system.out.println* en el proyecto.

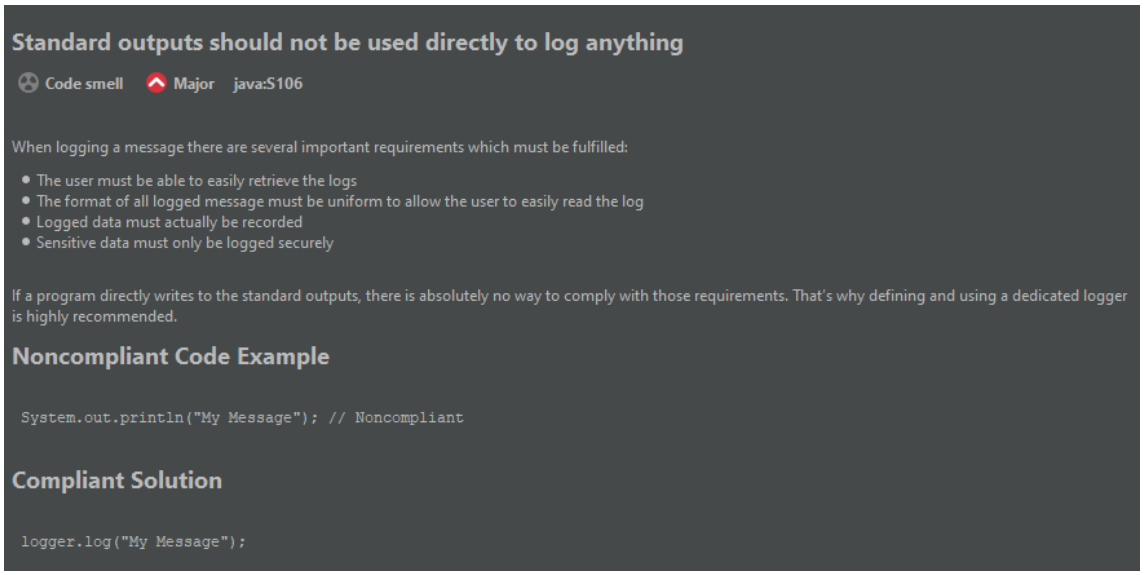


Ilustración 14 Code smell de SonarLint al usar System.out.println

El mensaje además de indicar el code smell que está provocando esa línea de código, aconseja sobre la otra opción recomendable a emplear, en este caso, un *logger*. Además, podemos ver los motivos que justifican que esa es la mejor opción:

- El usuario debe poder recuperar fácilmente las trazas de logs.
- El formato de las trazas de los logs debe ser uniforme para poder facilitar la lectura al usuario.
- Las trazas de los logs deben ser almacenadas.
- Los datos confidenciales deben trazarse de forma segura.

El cambio que se hizo para este code smell fue el commit presentado en la *Ilustración 15:*

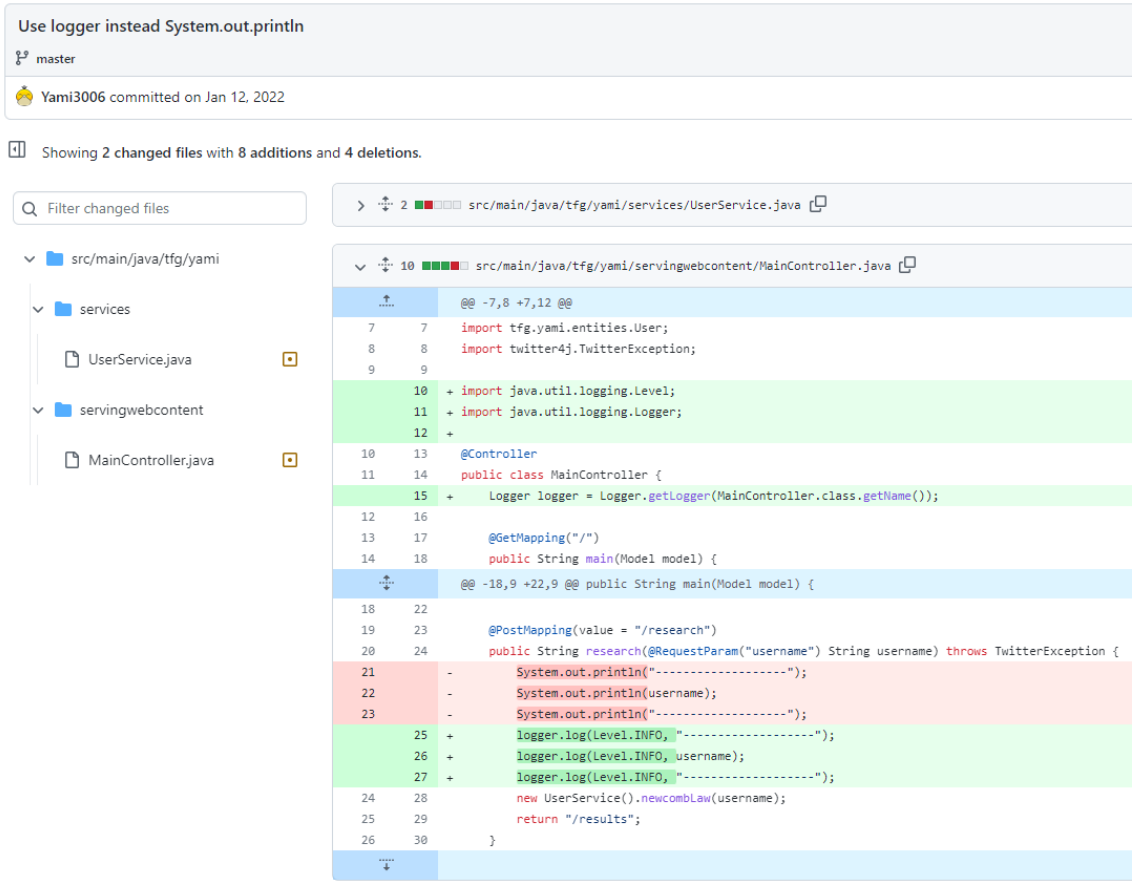


Ilustración 15 Uso de un Logger en vez de system.out.println

3.4 Algoritmo de Newcomb

En el siguiente fragmento de código podemos ver el esqueleto del método principal que se encarga de aplicar la ley de Benford a la cuenta que le fue introducida en el *input* de la página principal.

```
    public Map<Integer, Float> newcombLaw(String twitterUsername)
throws TwitterException {
        init();
        Long twitterUsernameId =
getTwitterinstance().showUser(twitterUsername).getId();

        int followers =
getTwitterinstance().showUser(twitterUsername).getFollowersCount();
        int followersInDB;
        try {
            followersInDB = getUsersInDB(twitterUsernameId,
followers);
        } catch (TwitterException e) {
            return new HashMap<>();
        }
        logger.info("El usuario {} tiene {} seguidores y hay {} en
BBDD", twitterUsername, followers, followersInDB);

        if (followersInDB == 0) {
            return new HashMap<>();
        }

        if (!shouldApplyLaw(followersInDB, followers)) {
            uploadUserFollowers(twitterUsername, followers);
            followersInDB = getUsersInDB(twitterUsernameId,
followers);
        }
        return applyNewcombLaw(twitterUsername, followersInDB);
    }
```

En primer lugar, como vemos en el siguiente fragmento de código se hace un *init()* para abrir la conexión a la BD e inicializar la entidad de usuarios:

```
    public static void init(){
        EntityManager entityManager = Persistence
                .createEntityManagerFactory("users")
                .createEntityManager();
        userDAO = new UserDAO(entityManager);
    }
```

Después, recopilamos datos del usuario a estudiar:

- *getTwitterinstance().showUser(twitterUsername).getId()* es una llamada a la API que se encarga de obtener el ID en Twitter de la cuenta dada.

- *getTwitterinstance().showUser(twitterUsername).getFollowersCount()* es una llamada a la API que se encarga de obtener el número de seguidores actuales en Twitter de la cuenta dada.
- *getUsersInDB(twitterUsernameId, followers)* es una llamada a un método interno que se encarga de obtener el número de seguidores de los cuales tenemos información almacenada en BD de la cuenta dada.

Después de obtener esa información, en el siguiente fragmento de código se puede ver una condición que establece si tenemos suficiente información para aplicar la ley, esta condición es que haya información de por lo menos el 80% de los seguidores del usuario dado.

```
    private boolean shouldApplyLaw(int followersInDB, int followers) {
        return followers*0.8 < followersInDB;
    }
```

Si no se cumple la condición, se empiezan a hacer múltiples peticiones a la API para obtener información de todos los seguidores de la cuenta dada.

Con cada llamada a la API se tiene en cuenta si estamos cerca de pasar el límite de llamadas, en caso de que estemos cerca del límite se espera el tiempo recomendado por la API en ese momento (*twitterUser.getRateLimitStatus().getSecondsUntilReset()*) antes de que devuelva un mensaje de error.

```
    private void uploadUserFollowers(String twitterUsername, int
followers) throws TwitterException {
        long twitterUsernameId =
getTwitterinstance().showUser(twitterUsername).getId();
        logger.info("Iniciando carga de los followers del usuario {}",
twitterUsername);
        long currCursor = -1;
        int rep = 0;
        IDs ids;
        do {
            ids =
getTwitterinstance().getFollowersIDs(twitterUsernameId, currCursor);
            long[] lIds = ids.getIDs();
            for (int i = 0; i < lIds.length; i++) {
                try {
                    if (!existsUser((int) ids.getIDs()[i])) {
                        twitter4j.User twitterUser =
getTwitterinstance().showUser(lIds[i]);
                        if
(twitterUser.getRateLimitStatus().getRemaining() < 2) {
                            logger.info("Vamos a esperar {}
segundos.", twitterUser.getRateLimitStatus().getSecondsUntilReset());
                            TimeUnit.SECONDS.sleep((long)
twitterUser.getRateLimitStatus().getSecondsUntilReset() + 2);
                        }
```

```
                        userDAO.persist(twitterUser.getScreenName(),
twitterUser.getFollowersCount(), (int) lIds[i]);
                    }
                } catch (TwitterException | InterruptedException e) {
                    logger.warn("No se ha podido analizar al usuario
de ID {}", lIds[i]);
                    logger.warn(e.getMessage());
                    Thread.currentThread().interrupt();
                }
            }
            rep++;
        } while ((currCursor = ids.getNextCursor()) != 0 && rep <
Math.ceil(followers/5000.0));
    }
```

Finalmente, cuando se tiene la información recopilada en BD se procede al análisis de los resultados. En primer lugar, se crean 9 listas para almacenar los distintos números de seguidores según empiecen por el dígito 1, 2, 3, … 9

```
    private Map<Integer, Float> applyNewcombLaw(String
twitterUsername, int followersInDB) throws TwitterException {
        long twitterUsernameId =
getTwitterinstance().showUser(twitterUsername).getId();
        int followers =
getTwitterinstance().showUser(twitterUsername).getFollowersCount();
        logger.info("Aplicando Newcomb Law");
        logger.info("El usuario {} tiene el ID {}", twitterUsername,
twitterUsernameId);
        Map<Integer, Float> map = new HashMap<>();
        List<Long> list1 = new ArrayList<>();
        List<Long> list2 = new ArrayList<>();
        List<Long> list3 = new ArrayList<>();
        List<Long> list4 = new ArrayList<>();
        List<Long> list5 = new ArrayList<>();
        List<Long> list6 = new ArrayList<>();
        List<Long> list7 = new ArrayList<>();
        List<Long> list8 = new ArrayList<>();
        List<Long> list9 = new ArrayList<>();
```

En este segundo paso, se obtiene el número de seguidores de cada cuenta que sigue a la cuenta analizada, y se clasifica en las listas mencionadas previamente dependiendo de si el primer dígito de este número de seguidores vale 1, 2, 3, 4, 5, 6, 7, 8 y 9.

```
IDs ids;
        int rep = 0;
        long currCursor = -1;
        do {
            ids =
getTwitterinstance().getFollowersIDs(twitterUsernameId, currCursor);
            long[] lIds = ids.getIDs();

            for (int i = 0; i < lIds.length; i++) {
                if ((getDBUser((int) lIds[i])) != null) {
```

```
                    switch
(getFirstDigit((Objects.requireNonNull(getDBUser((int)
lIds[i])).getFollowers()))) {
                        case 1:
                            list1.add(lIds[i]);
                            break;
                        case 2:
                            list2.add(lIds[i]);
                            break;
                        case 3:
                            list3.add(lIds[i]);
                            break;
                        case 4:
                            list4.add(lIds[i]);
                            break;
                        case 5:
                            list5.add(lIds[i]);
                            break;
                        case 6:
                            list6.add(lIds[i]);
                            break;
                        case 7:
                            list7.add(lIds[i]);
                            break;
                        case 8:
                            list8.add(lIds[i]);
                            break;
                        case 9:
                            list9.add(lIds[i]);
                            break;
                        default:
                            logger.info("El usuario de ID {} tiene 0
followers", lIds[i]);
                            break;
                    }
                }
            }
            rep++;
        } while ((currCursor = ids.getNextCursor()) != 0 && rep <
Math.ceil(followers/5000.0));

        logger.info("El usuario {} tiene {} seguidores y hay {} en
BBDD", twitterUsername, followers, followersInDB);
```

Por último, se devuelve un objeto *hashmap* cuya clave corresponde a 1, 2, 3, 4, 5, 6, 7, 8 y 9 y el valor corresponde al número de cuentas cuyo número de seguidores empiezan por esos dígitos en proporción al número total de los seguidores totales.

```
        map.put(1, ((float) list1.size() * 100) / followersInDB);
        map.put(2, ((float) list2.size() * 100) / followersInDB);
        map.put(3, ((float) list3.size() * 100) / followersInDB);
        map.put(4, ((float) list4.size() * 100) / followersInDB);
        map.put(5, ((float) list5.size() * 100) / followersInDB);
        map.put(6, ((float) list6.size() * 100) / followersInDB);
```

```
        map.put(7, ((float) list7.size() * 100) / followersInDB);
        map.put(8, ((float) list8.size() * 100) / followersInDB);
        map.put(9, ((float) list9.size() * 100) / followersInDB);

        for (Map.Entry<Integer, Float> entry : map.entrySet()) {
            logger.info("Los followers que empiezan por {} son: {}%",
entry.getKey(), entry.getValue());
        }
        return map;
    }
```

3.5 Pruebas

Pruebas unitarias (caja negra):

Se realizan pruebas en el único campo de la aplicación donde se pueden introducir datos.

- Caso de Prueba 01: no introducir username. Está controlado que este campo sea obligatorio y no puede estar vacío, de lo contrario aparece una advertencia como se muestra en la *Ilustración 16*:

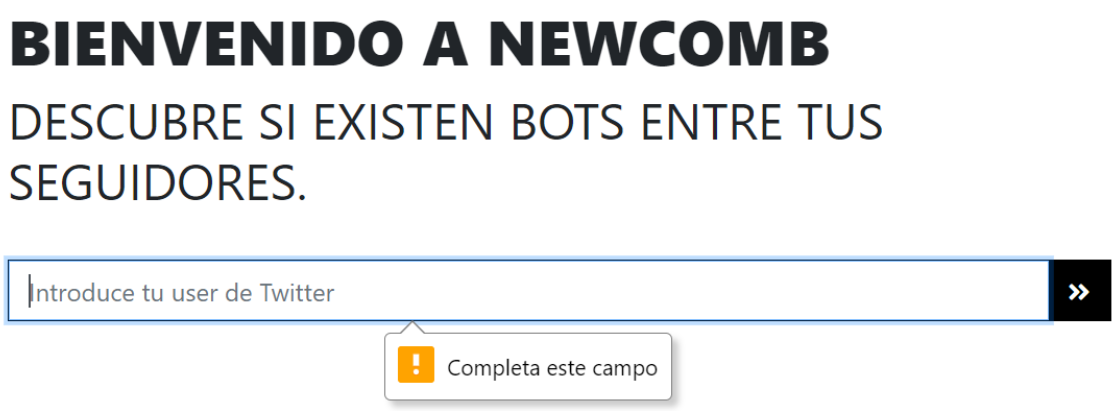


Ilustración 16 Mensaje de validación del campo username vacío

- Caso de Prueba 02: valor del campo username: este campo está limitado a quince caracteres, que es la misma limitación que existe en Twitter para el nombre de los usuarios. En la *Ilustración 17* se puede ver que el atributo maxlength del campo está a quince.

Ilustración 17 Limitación de caracteres en el campo username

Pruebas unitarias (caja blanca)

- Caso de Prueba 01: valor del campo username: en este campo se debe validar si existe o no el usuario en Twitter. En caso de que no exista se muestra un mensaje de error. En la *Ilustración 18* podemos ver que esto no es así, sino que se muestra un error de código no controlado. Incorrecto

Ilustración 18 Error no controlado

Pruebas de integración

Como consecuencia de la ejecución de las pruebas unitarias se corrigieron los siguientes errores:

- Caso de prueba 01: error de usuario no existente controlado, como se puede ver en la *Ilustración 19*.

Ilustración 19 Error controlado en caso de que el usuario no exista

4. EXPERIMENTOS/VALIDACIÓN

A continuación, vamos a explicar los experimentos que se han realizado para demostrar el cumplimiento de la Ley de Benford.

En la *Ilustración 20* tenemos un gráfico lineal donde la línea naranja representa la gráfica que deben seguir las cuentas para demostrar que no son bots.

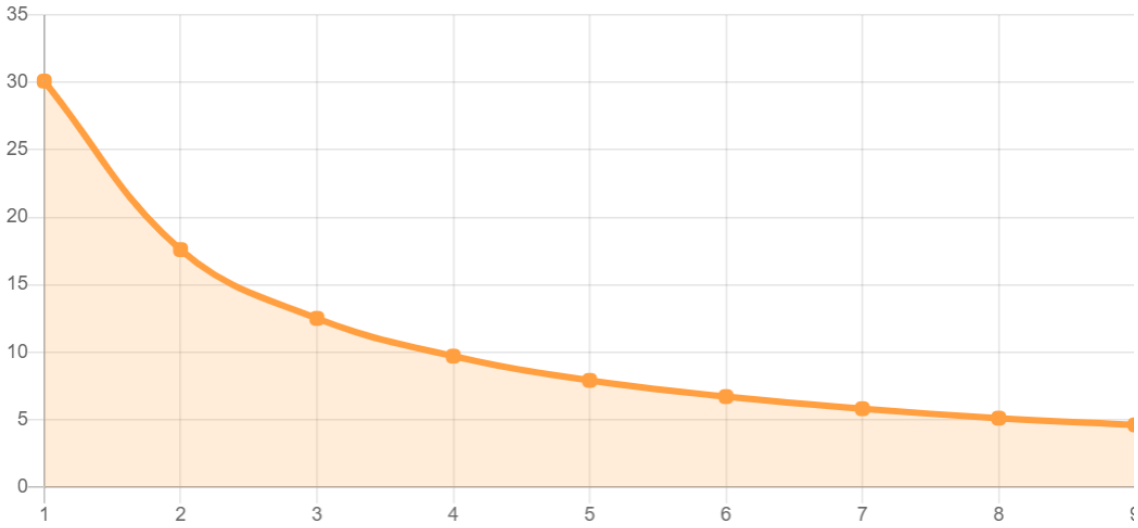


Ilustración 20 Gráfico lineal de la ley de Benford

Esta línea corresponde a los valores mostrados en la *Tabla 3*:

Seguidores de la cuenta que tengan un número de seguidores que empiecen con este dígito	Porcentaje que hace cumplir para la Ley de Benford
1	30,1%
2	17,6%
3	12,5%
4	9,7%
5	7,9%
6	6,7%
7	5,8%
8	5,1%
9	4,6%

Tabla 3 Porcentaje de los valores que cumplen la Ley de Benford

Los experimentos que se han hecho consisten en analizar 4 muestras de cuentas de bots y 4 muestras de cuentas de personas reales. Si algún porcentaje varía en al menos un 1%, se establecerá que esa cuenta analizada NO cumple la ley de Benford.

Estas cuentas son:

- BidhunQuotes: es una cuenta bot que se encarga de twittear citas de Memorias de Idhún. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las mostradas en la *Ilustración 21*:

```
10:41:54.819 INFO  tfg.yami.services.UserService - El usuario BidhunQuotes tiene 72 seguidores y hay 72 en BBDD
10:41:54.820 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 19.444445%
10:41:54.820 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 18.055555%
10:41:54.820 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 20.833334%
10:41:54.820 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 5.5555553%
10:41:54.820 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 9.722222%
10:41:54.821 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 5.5555553%
10:41:54.821 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 8.333333%
10:41:54.821 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 4.1666665%
10:41:54.821 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 8.333333%
```

Ilustración 21 Trazas de BidhunQuotes

A partir de estos resultados, la aplicación muestra un gráfico lineal en la *Ilustración 22* donde podemos ver las notables diferencias entre la línea negra respecto a la naranja, siendo la línea naranja la ya explicada previamente y la negra la del usuario analizado BidhunQuotes.

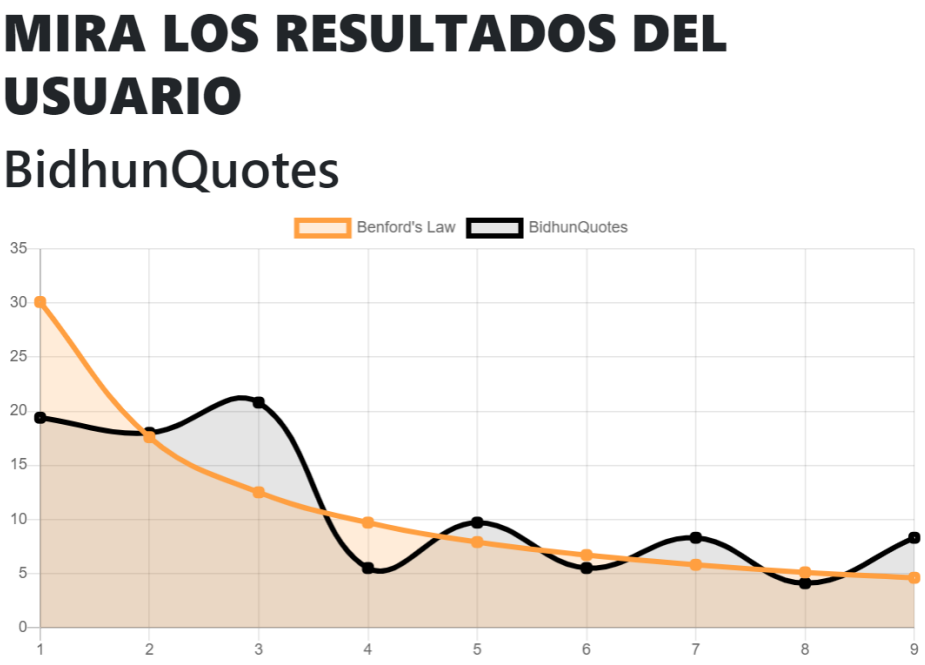


Ilustración 22 Gráfico lineal de BidhunQuotes

Podemos ver que no se cumple la ley ya que estas líneas difieren en casi todos los dígitos (todos menos el 2) y por lo tanto se confirma que es un bot sin ninguna duda.

Dígito	Ley de Benford	BidhunQuotes	Diferencia
1	30,1	19,4	10,7
2	17,6	18	-0,4
3	12,5	20,8	-8,3
4	9,7	5,5	4,2
5	7,9	9,7	-1,8
6	6,7	5,5	1,2
7	5,8	8,3	-2,5
8	5,1	4,1	1
9	4,6	8,3	-3,7

Tabla 4 Análisis de BidhunQuotes

- precioCompra: es una cuenta bot que se encarga de reportar los cambios de los precios de los productos del supermercado Mercadona. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores son las mostradas en la *Ilustración 23:*

```
10:58:04.210 INFO  tfg.yami.services.UserService - El usuario precioCompra tiene 1307 seguidores y hay 1307 en BBDD
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 31.981638%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 19.127773%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 12.394797%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 8.951798%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 6.7329764%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 6.2739096%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.126243%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 3.9020658%
10:58:04.210 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 4.5141544%
```

Ilustración 23 Trazas de precioCompra

Los resultados de la aplicación muestran que esta cuenta tampoco cumple la Ley de Benford porque presenta hasta 4 variaciones (dígito 1, 2, 5 y 8) de más de un 1% respecto a la línea naranja, se confirma que se trata de un bot. (Ver *Ilustración 24* y *Tabla 5*).

MIRA LOS RESULTADOS DEL USUARIO

precioCompra

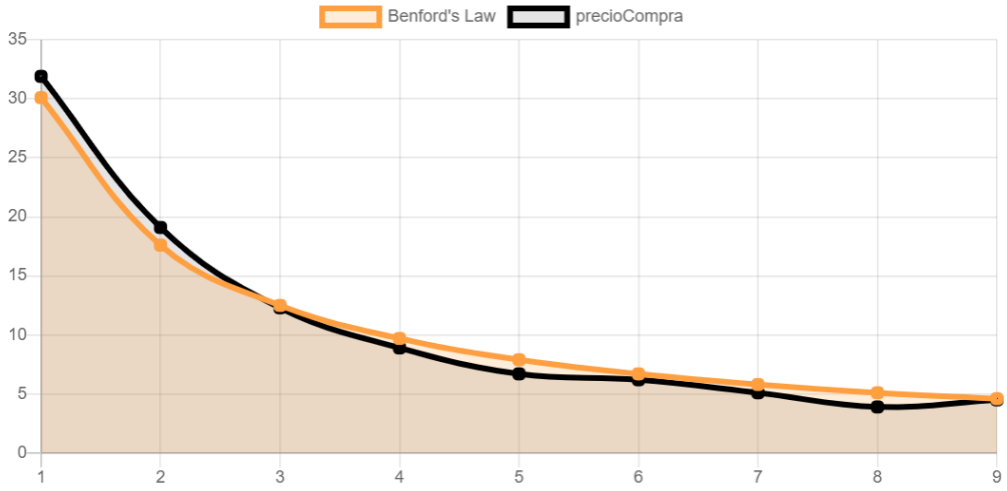


Ilustración 24 Gráfico lineal de precioCompra

Dígitos	Ley de Benford	precioCompra	Diferencia
1	30,1	31,9	-1,8
2	17,6	19,1	-1,5
3	12,5	12,3	0,2
4	9,7	8,9	0,8
5	7,9	6,7	1,2
6	6,7	6,2	0,5
7	5,8	5,1	0,7
8	5,1	3,9	1,2
9	4,6	4,5	0,1

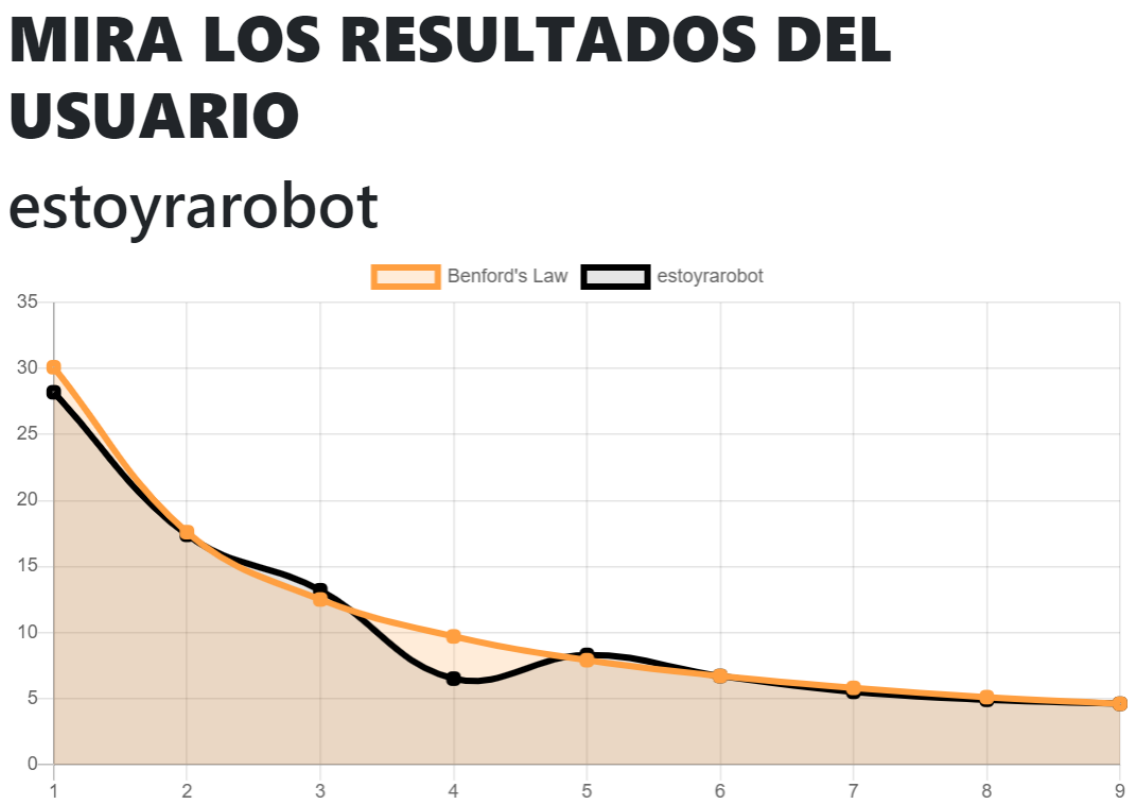
Tabla 5 Análisis de precioCompra

- estoyrarobot: es una cuenta bot que publica tweets con letras del grupo de música Cuarteto de Nos. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las mostradas en la *Ilustración 25*:

```
12:03:54.520 INFO  tfg.yami.services.UserService - El usuario estoyrarobot tiene 686 seguidores y hay 686 en BBDD
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 28.279882%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 17.492712%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 13.265306%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 6.559767%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 8.309038%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 6.705539%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.5393586%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 4.9562683%
12:03:54.521 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 4.664723%
```

Ilustración 25 Trazas de estoyrarobot

Como se puede observar en la *Ilustración 26* y *Tabla 6* hay más de 1% de diferencia entre ambas líneas en los puntos 1 y 4, se confirma que se trata de un bot.



Dígitos	Ley de Benford	estoyrarobot	Diferencia
1	30,1	28,2	1,9
2	17,6	17,4	0,2
3	12,5	13,2	-0,7
4	9,7	6,5	3,2
5	7,9	8,3	-0,4
6	6,7	6,7	0
7	5,8	5,5	0,3
8	5,1	4,9	0,2
9	4,6	4,6	0

Tabla 6 Análisis de estoyrarobot

- lazarobot: se trata de una cuenta bot que comenta los anglicismos utilizados en artículos de la prensa española. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las son las mostradas en la *Ilustración 27*:

```
20:06:26.256 INFO  tfg.yami.services.UserService - El usuario lazarobot tiene 1815 seguidores y hay 1816 en BBDD
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 29.405287%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 17.345816%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 11.178414%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 10.187224%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 8.094713%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 6.4427314%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.61674%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 5.2863436%
20:06:26.256 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 4.6255507%
```

Ilustración 27 Trazas de lazarobot

Podemos ver que difiere más de un 1% en el punto 4 respecto a la gráfica naranja. Podemos confirmar que se trata de un bot. (Ver *Ilustración 28* y *Tabla 7*).

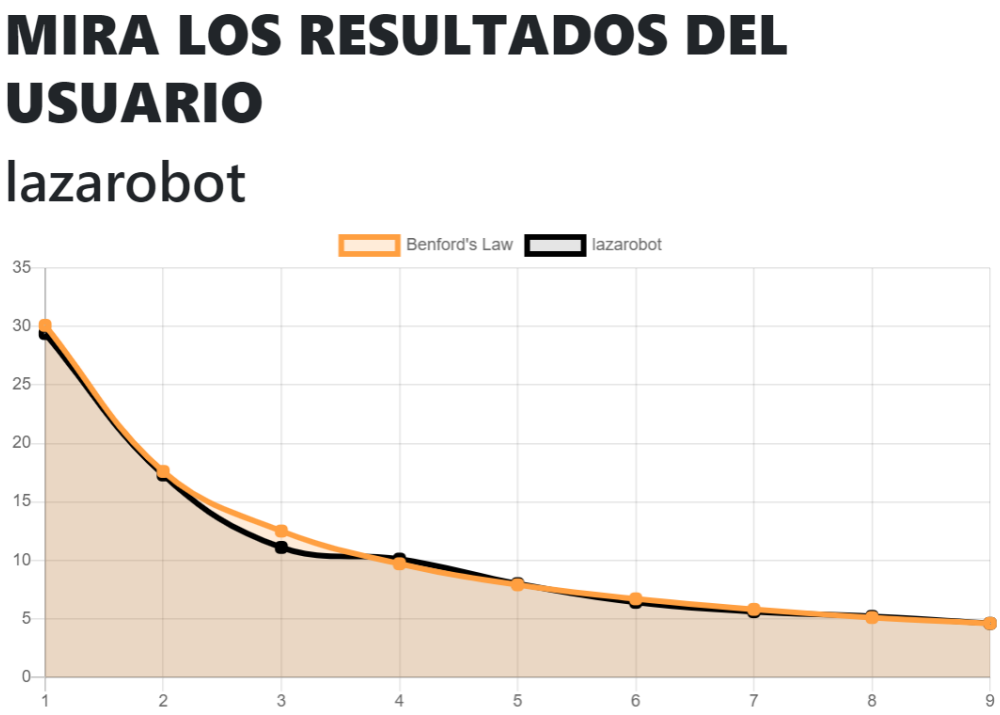


Ilustración 28 Gráfico lineal de lazarobot

Dígitos	Ley de Benford	lazarobot	Diferencia
1	30,1	29,4	0,7
2	17,6	17,3	0,3
3	12,5	11,1	1,4
4	9,7	10,1	-0,4
5	7,9	8,0	-0,1
6	6,7	6,4	0,3
7	5,8	5,6	0,2
8	5,1	5,2	-0,1
9	4,6	4,6	0

Tabla 7 Análisis de lazarobot

- BlancaPaloma_rb: se trata de la cuenta de twitter de nuestra representante a eurovisión este 2023. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las mostradas en la *Ilustración 29*:

```
00:15:53.064 INFO  tfg.yami.services.UserService - El usuario BlancaPaloma_rb tiene 7036 seguidores y hay 7035 en BBDD
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 29.267946%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 17.384506%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 11.769723%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 9.751244%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 8.073916%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 6.169154%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.302061%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 4.9466953%
00:15:53.064 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 4.2643924%
```

Ilustración 29 Trazas de BlancaPaloma_rb

Podemos ver que la gráfica es muy parecida y la máxima diferencia que hay es del 0,9% en el dígito 1, por lo que cumple la ley de Benford y podemos decir que se trata de una cuenta real. (Ver *Ilustración 30* y *Tabla 8*).

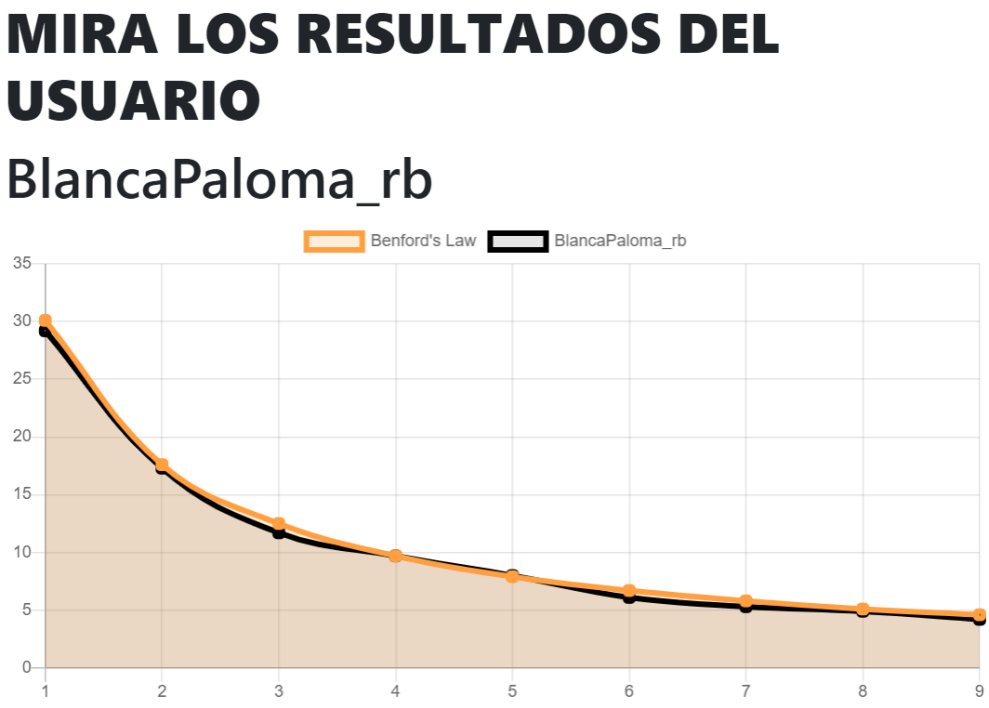


Ilustración 30 Gráfico lineal de BlancaPaloma_rb

Dígitos	Ley de Benford	BlancaPaloma_rb	Diferencia
1	30,1	29,2	0,9
2	17,6	17,38	0,22
3	12,5	11,7	0,8
4	9,7	9,75	-0,05
5	7,9	8,07	-0,17
6	6,7	6,1	0,6
7	5,8	5,3	0,5
8	5,1	4,9	0,2
9	4,6	4,2	0,4

Tabla 8 Análisis de BlancaPaloma_rb

- FUSANOCTA: se trata de la cuenta de twitter de una aspirante a representar eurovisión 2023. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las mostradas en la *Ilustración 31*:

```
20:43:08.141 INFO  tfg.yami.services.UserService - El usuario FUSANOCTA tiene 8369 seguidores y hay 8097 en BBDD
20:43:08.141 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 29.23305%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 17.53736%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 12.634309%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 9.979005%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 7.5707054%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 6.496233%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.397061%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 4.7054462%
20:43:08.142 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 3.8409288%
```

Ilustración 31 Trazas de FUSANOCTA

En este caso la gráfica también es muy parecida y cumple la ley, ya que sólo llega al 0,9% de diferencia del porcentaje respecto a la línea naranja en el dígito 1. (Ver *Ilustración 32* y *Tabla 9).*

MIRA LOS RESULTADOS DEL USUARIO

FUSANOCTA

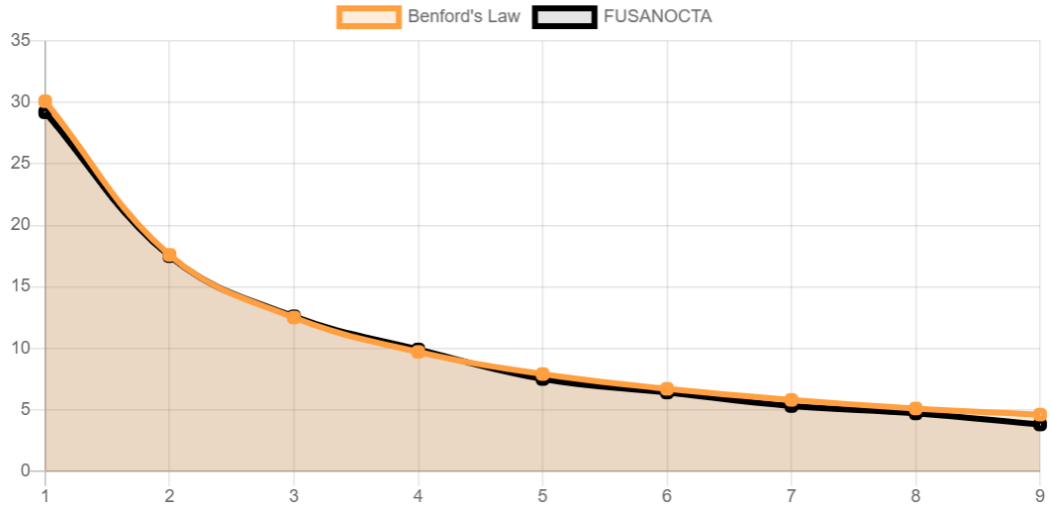


Ilustración 32 Gráfico lineal de FUSANOCTA

Dígitos	Ley de Benford	FUSANOCTA	Diferencia
1	30,1	29,2	0,9
2	17,6	17,5	0,1
3	12,5	12,6	-0,1
4	9,7	9,9	-0,2
5	7,9	7,5	0,4
6	6,7	6,4	0,3
7	5,8	5,3	0,5
8	5,1	4,7	0,4
9	4,6	3,8	0,8

Tabla 9 Análisis de FUSANOCTA

- Cariniopop: se trata de un grupo musical español conformado por 3 chicas cuya música es indie. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las mostradas en la *Ilustración 33*:

```
20:40:28.868 INFO  tfg.yami.services.UserService - El usuario cariniopop tiene 24267 seguidores y hay 24267 en BBDD
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 29.377344%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 17.756624%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 12.737462%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 9.580913%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 7.710059%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 6.251288%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.183995%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 4.549388%
20:40:28.868 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 4.017802%
```

Ilustración 33 Trazas de cariniopop

En la gráfica resultante podemos ver que cumple casi fielmente la Ley de Benford por lo que se trata de una cuenta real. (Ver *Ilustración 34* y *Tabla 10*).

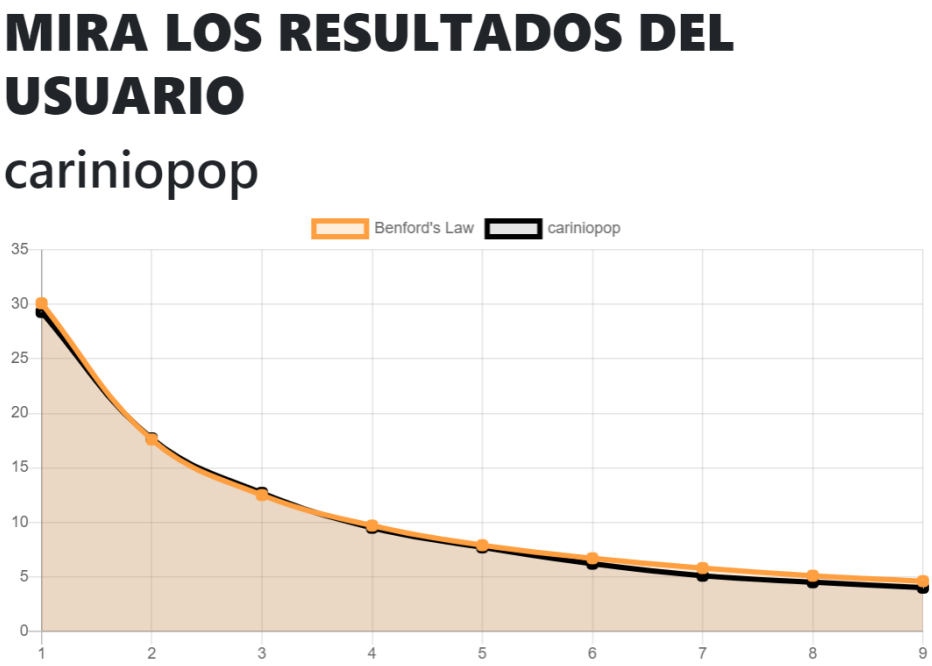


Ilustración 34 Gráfico lineal de cariniopop

Dígitos	Ley de Benford	cariniopop	Diferencia
1	30,1	29,3	0,8
2	17,6	17,7	-0,1
3	12,5	12,7	-0,2
4	9,7	9,5	0,2
5	7,9	7,7	0,2
6	6,7	6,2	0,5
7	5,8	5,1	0,7
8	5,1	4,5	0,6
9	4,6	4,0	0,6

Tabla 10 Análisis de cariniopop

- helioroque_: se trata de un influencer que empezó haciendo vídeos sobre el metro de Madrid en TikTok. Las trazas del análisis y el porcentaje de los dígitos de los seguidores de los seguidores de esta cuenta son las mostradas en la *Ilustración 35*:

```
20:58:56.323 INFO  tfg.yami.services.UserService - El usuario helioroque_ tiene 13932 seguidores y hay 13902 en BBDD
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 1 son: 29.794273%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 2 son: 16.904043%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 3 son: 12.508991%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 4 son: 9.696446%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 5 son: 7.3298807%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 6 son: 5.9128184%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 7 son: 5.1575313%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 8 son: 4.6108475%
20:58:56.323 INFO  tfg.yami.services.UserService - Los followers que empiezan por 9 son: 4.2943463%
```

*Ilustración 35 Trazas de helioroque_*

En la gráfica y en la tabla podemos ver que con la información analizada concluimos que cumple la ley de Benford y por lo tanto no es un bot. (Ver *Ilustración 36* y *Tabla 11*).

MIRA LOS RESULTADOS DEL USUARIO

helioroque_

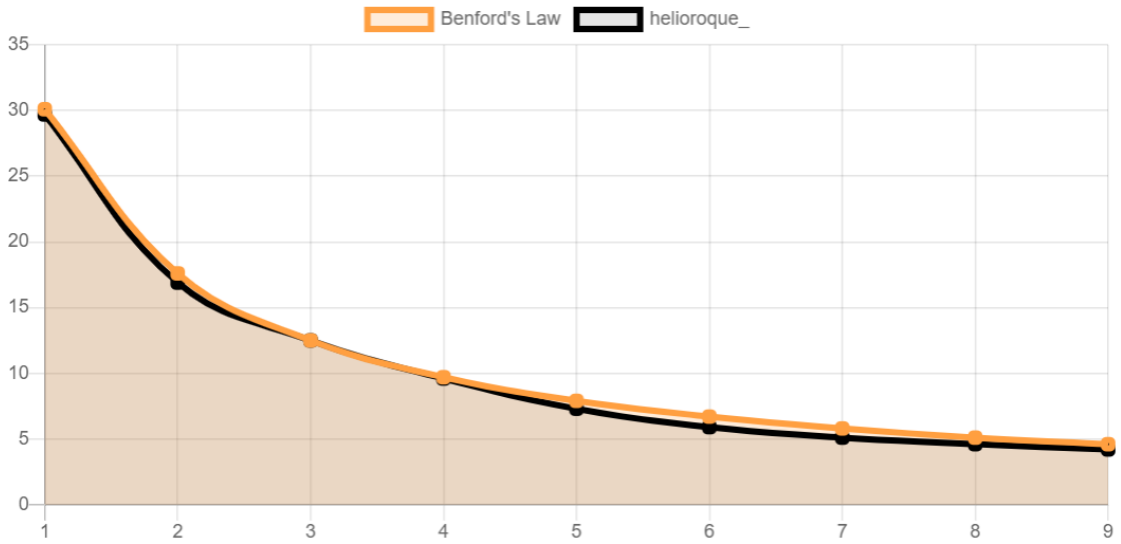


*Ilustración 36 Gráfico lineal de helioroque_*

Dígitos	Ley de Benford	helioroque_	Diferencia
1	30,1	29,7	0,4
2	17,6	16,9	0,7
3	12,5	12,5	0
4	9,7	9,6	0,1
5	7,9	7,3	0,6
6	6,7	5,9	0,8
7	5,8	5,1	0,7
8	5,1	4,6	0,5
9	4,6	4,2	0,4

*Tabla 11 Análisis de helioroque_*

5. CONCLUSIONES

Logros principales alcanzados

Personalmente, mi mayor logro ha sido terminar este TFG, era una tarea que tenía pendiente y solo siento satisfacción después de terminar algo que ha llevado mucho tiempo, esfuerzo y dedicación. Ahora puedo estar feliz conmigo misma porque por fin puedo decir que he terminado una etapa.

En primer lugar, he aprendido a gestionar un proyecto software desde cero, planificando los tiempos de cada etapa y el tiempo empleado en la resolución de errores, que no es despreciable.

En cuanto al desarrollo del software, este trabajo ha supuesto poner en práctica los conocimientos adquiridos durante el grado y durante mi trayectoria profesional de casi 5 años, estos conocimientos involucran tanto el frontend como el backend, la utilización de una API externa, he utilizado tecnologías que se usan en la actualidad, algo que deseaba poner en práctica ya que en el trabajo hago un mantenimiento de un proyecto legacy que poco a poco necesita ser actualizado con nuevas librerías. Sin duda este proyecto servirá como base para evolucionar mis proyectos profesionales y personales.

Posibles trabajos futuros

Actualmente y tal y como está la situación en cuanto al uso de la API de Twitter, mi primera propuesta como trabajo futuro sería la de aplicar la ley de Benford/Newcomb a otras webs con el mismo fin, detectar bots.

El resto de las redes sociales tales como Facebook, Instagram, Youtube y Telegram tienen APIs gratuitas, por lo que solo necesitaríamos un previo estudio de las APIS de estas redes sociales y el mismo algoritmo utilizado en este proyecto para el análisis de las muestras.

En este proyecto hay una posibilidad de mejora en velocidad si se utilizase más de una credencial para el acceso a la API, de este modo, los límites por llamadas a los endpoints desaparecerían.

Por último, en esta aplicación se ha analizado una única muestra de datos (el número de seguidores de los seguidores de la cuenta a analizar), para seguir comprobando su validez se pueden analizar más datos relevantes de la cuenta como el número de tweets, el número de "me gustas", el número de imágenes publicadas, etc…

6. GLOSARIO DE TÉRMINOS

- **Twitter**: “es una plataforma social de microblogging, que sirve básicamente para comunicarse mediante mensajes cortos y de forma gratuita con otras personas o usuarios registrados en ella.” (Facchin, 2019)

- **Bot**: “término que proviene de acortar la palabra *robot*. Es un programa que realiza tareas repetitivas, predefinidas y automatizadas. Los bots están diseñados para imitar o sustituir el accionar humano. Operan en forma automatizada, por lo que pueden trabajar mucho más rápido que una persona. Algunos bots cumplen funciones útiles, como buscar y catalogar páginas web o ayudar a los clientes de una empresa con sus problemas; otros, sin embargo, son enteramente maliciosos y se utilizan para hacerse con el mando de sistemas ajenos.” (Kaspersky, 2022)

- **API**: “significa *interfaz de programación de aplicaciones*. En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas. La documentación de una API contiene información sobre cómo los desarrolladores deben estructurar esas solicitudes y respuestas.” (AWS, 2022)
- **Endpoint:** url indicada en una API para interactuar con ella y así obtener, introducir, editar, borrar (GET, POST, PUT, DELETE) información de una aplicación web.
- **Base de datos**: “es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático.” (Oracle, 2022)
- **Framework: “**es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código.” (Arimetrics, 2021; Arimetrics, 2021)

- **Commit:** “es la acción de guardar o subir archivos a un repositorio remoto (una actualización de los cambios). También puede hacerse al repositorio local (depende de donde se haya creado el repositorio)” (Cuadrados, 2015)

7. BIBLIOGRAFÍA

Alarcón, J. M. (1 de Junio de 2022). Obtenido de https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx

Arimetrics. (05 de Agosto de 2021). Obtenido de Arimetrics: https://www.arimetrics.com/glosario-digital/framework

AWS. (18 de Septiembre de 2022). Obtenido de AWS: https://aws.amazon.com/es/what-is/api/#:~:text=API%20significa%20%E2%80%9Cinterfaz%20de%20programaci%C3%B3n,de%20servicio%20entre%20dos%20aplicaciones.

Cuadrados, D. P. (19 de Agosto de 2015). *CodigoFacilito*. Obtenido de CodigoFacilito: https://codigofacilito.com/articulos/commits-administrar-tu-repositorio

Facchin, J. (17 de Abril de 2019). *Web Escuela*. Obtenido de Web Escuela: https://webescuela.com/que-es-twitter-como-funciona/

Kaspersky. (18 de Septiembre de 2022). Obtenido de Kaspersky: https://latam.kaspersky.com/resource-center/definitions/what-are-bots

Oracle. (18 de Septiembre de 2022). Obtenido de Oracle: https://www.oracle.com/es/database/what-is-database/

School, T. (25 de Octubre de 2022). *Tokio School*. Obtenido de https://www.tokioschool.com/noticias/spring-boot/